Getting Started With

DEEPAK RAWAT

APRIL/2024

<u>Burp Suite</u>

Burp Suite is a comprehensive and powerful tool designed for web application security testing, focusing on identifying and analyzing vulnerabilities in web applications. Developed by PortSwigger, a leading provider of software security solutions, Burp Suite has become an essential asset for security professionals, penetration testers, and developers who aim to ensure the security and integrity of web applications.

The software is available in both free and paid versions, with the Professional edition offering additional features and functionalities. It can be run on various platforms, including Windows, macOS, and Linux, and is often used in conjunction with popular web browsers like Chrome, Firefox, and Safari.

Burp Suite simplifies the process of web application security testing by providing a user-friendly interface and a range of integrated tools. It enables users to analyze the communication between the client and the server, identify hidden or overlooked endpoints, automate vulnerability discovery, and analyze the application's behaviour and response.

By utilizing Burp Suite, users can perform various security testing techniques, such as intercepting and modifying HTTP(S) requests and responses, automatically discovering and mapping web application structures, automating the process of sending multiple requests with different parameters, and examining individual HTTP requests in detail. This

comprehensive approach to web application security testing helps ensure that applications are secure and less vulnerable to attacks.

In summary, Burp Suite is an indispensable tool for security professionals, penetration testers, and developers who seek to maintain the security and integrity of web applications. Its user-friendly interface, combined with its powerful features and functionalities, make it an essential component in the world of web application security testing.

Burp Suite system requirements

The system requirements for Burp Suite are largely dependent on your intended use for the software. While you can generally perform most tasks on a relatively low-spec machine, some use cases (for example, running multiple scans concurrently) may require significantly more power to run without a noticeable effect on performance.

CPU cores / memory

- **Minimum: 2x cores, 4GB RAM** - This spec is suitable for basic tasks such as proxying web traffic and simple Intruder attacks. While Burp Suite may run on a machine with a lower specification than this, we do not recommend doing so for performance reasons.
- **Recommended: 2x cores, 16GB RAM** - This is a good general-purpose spec.
- **Advanced: 4x cores, 32GB RAM** - This spec is suitable for more intensive tasks, such as complex Intruder attacks or large automated scans.

Free disk space

- **Basic installation**: 1GB
- **Per project file**: 2GB

Note:

While 2GB is the recommended minimum free disk space for a project, note that project files can get significantly larger than this (potentially up to many tens of GB), depending on factors such as the amount of proxy history included, the number of scans run, and the number of Repeater tabs open.

Operating system and architecture

Burp Suite supports the latest versions of the following operating systems:

- Windows (Intel 64-bit)
- Linux (Intel and ARM 64-bit)
- OS X (Intel 64-bit and Apple M1)

Embedded browser

Burp's browser has some additional operating system and architecture requirements. It is not compatible with the following:

- Older versions of Windows, including Windows 7, Windows 8/8.1, Windows Server 2012, and Windows Server 2012 R2.
- Instances of Burp Suite that run via the JAR file on Apple Silicon and ARM 64-bit based systems. If you want to use Burp's browser on systems with these chip sets, make sure that you install Burp using the native platform installers.

Note

You can still run multiple instances of Burp simultaneously when using the platform installer versions. This functionality is not limited to instances of Burp run from the JAR file.

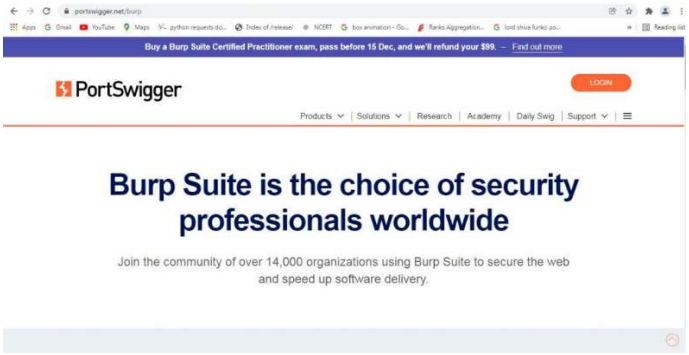
Download

Burp Suite Community Addition :
https://portswigger.net/burp/communitydownload

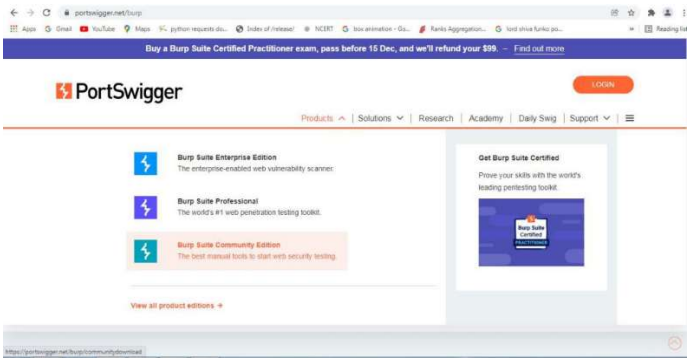
Burp Suite Professional : https://portswigger.net/burp/pro

Burp Suite Enterprise : https://portswigger.net/burp/enterprise

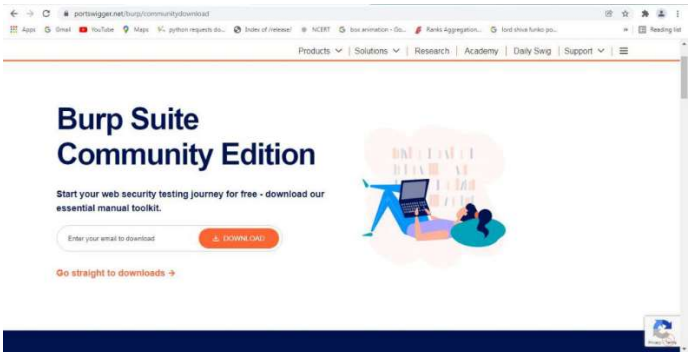
Step 1: Visit the official Burp Suite website using any web browser.



Step 2: Click on Products, a list of different Burp Suites will open, choose Burp suite Community Edition as it is free, click on it.

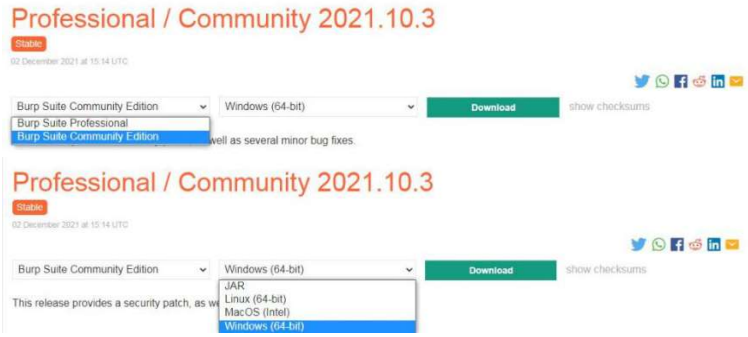


Step 3: New webpage will open, which will ask for email id, and other option is Go Straight to downloads. Click on Go straight to downloads.

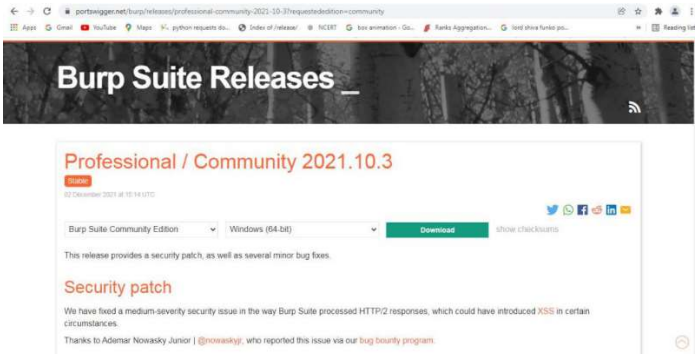


Step 4: After clicking on Go straight to downloads new webpage will open which will contain two versions of burp suite one is Burp suite

community edition and the other is burp suite professional along with compatibility for different operating systems.



Step 5: Choose Burp suite Community Edition along with Windows (64-bit). Click on the download button, downloading of the executable file will start shortly. It is a big 210 MB file that will take some time depending on download speed.

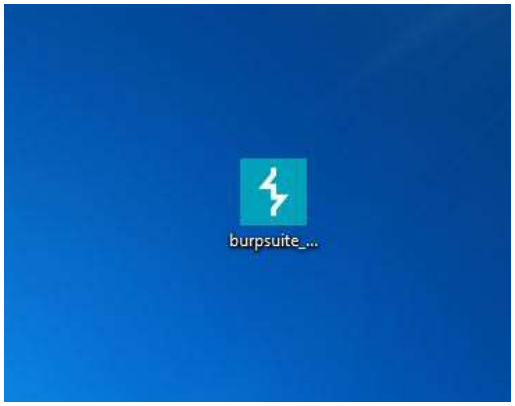


Installation

Installing Burp Suite on Windows

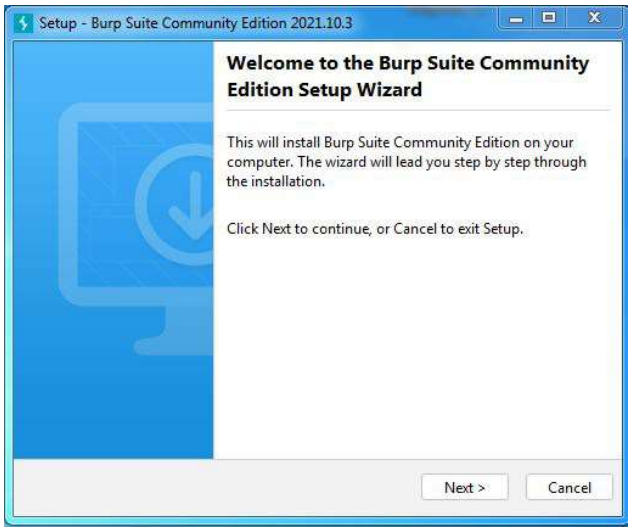
Follow the below steps to install Burp Suite on Windows:

Step 1: Now check for the executable file in downloads in your system and run it.

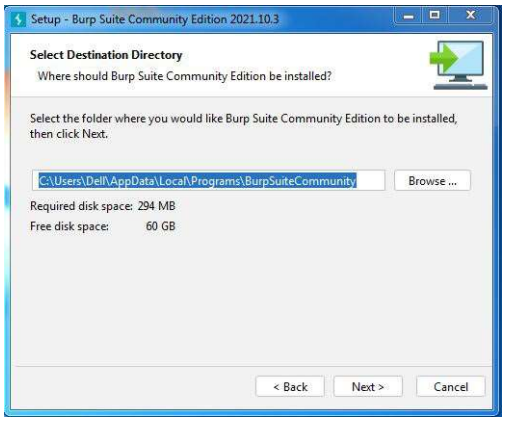


Step 2: Loading of Installation Wizard will appear which will take a few seconds.

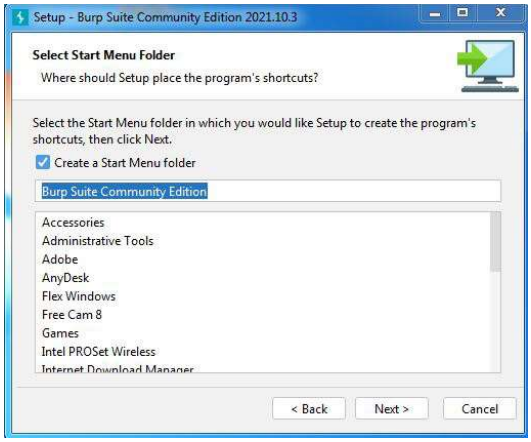
Step 3: After this Setup screen will appear, click on Next.



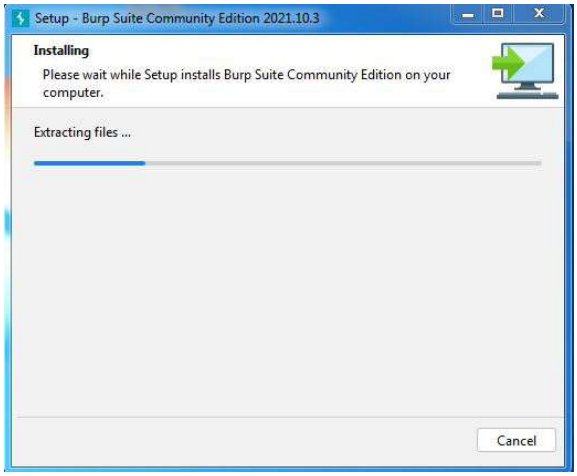
Step 4: The next screen will be of installing location so choose the drive which will have sufficient memory space for installation. It needed a memory space of 294 MB.



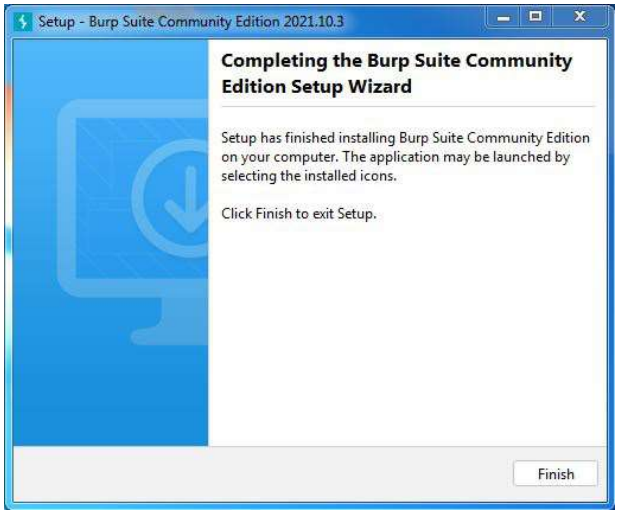
Step 5: Next screen will be of choosing Start menu folder so don't do anything just click on Next Button.



Step 6: After this installation process will start and will hardly take a minute to complete the installation.

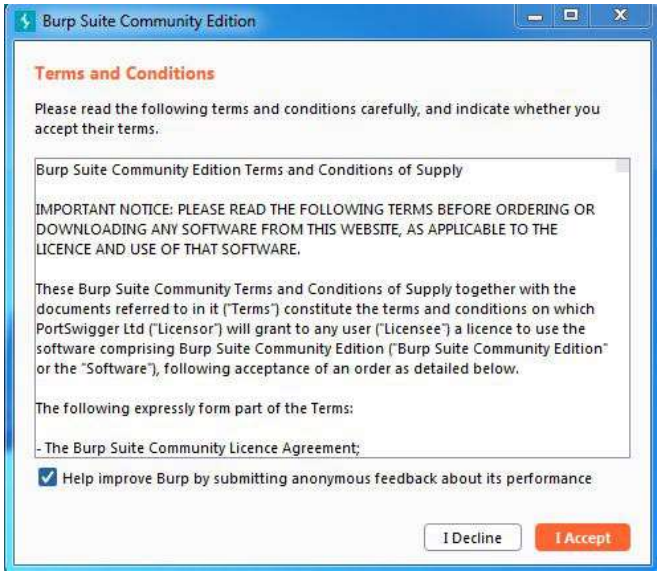


Step 7: Click on Finish after the installation process is complete.

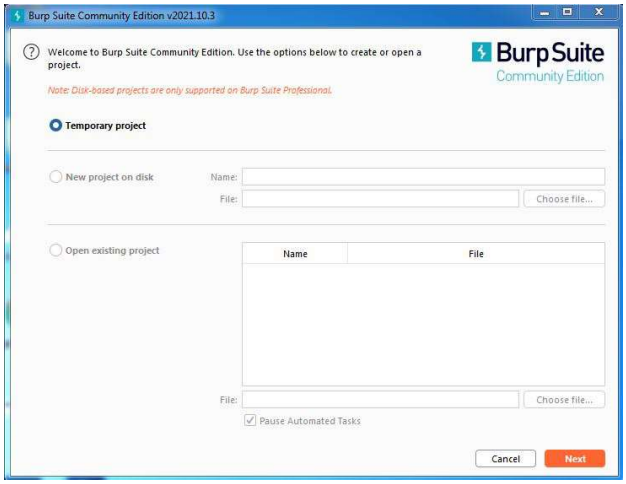


Step 8: Burp suite is successfully installed on the system and an icon is created on the desktop.

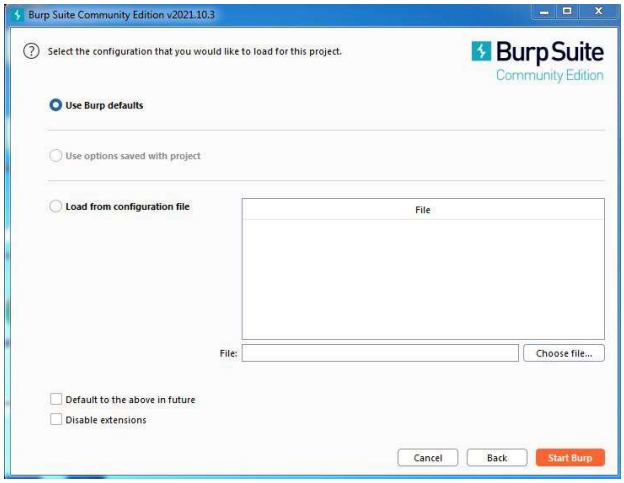
Step 9: Run the software, screen containing terms and conditions will appear Click on I Accept.



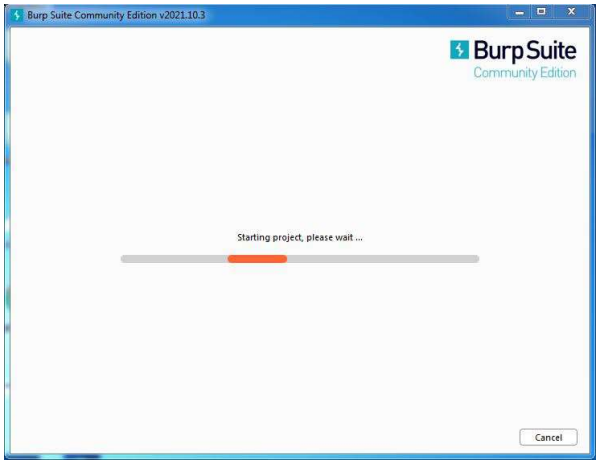
Step 10: New screen containing information regarding the project will appear, choose temporary project and click Next.



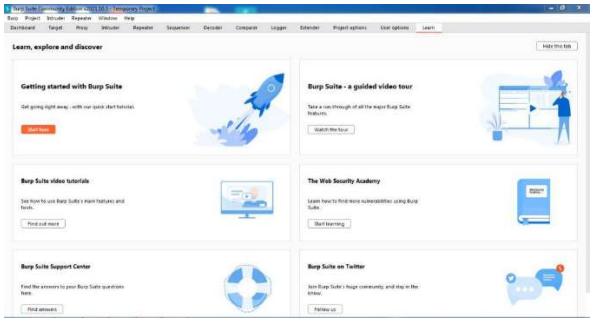
Step 11: Next screen is about using default settings or loading from configuration file, click on Use Burp Defaults.



Step 12: Project will start loading.



Step 13: Finally new project window will appear.

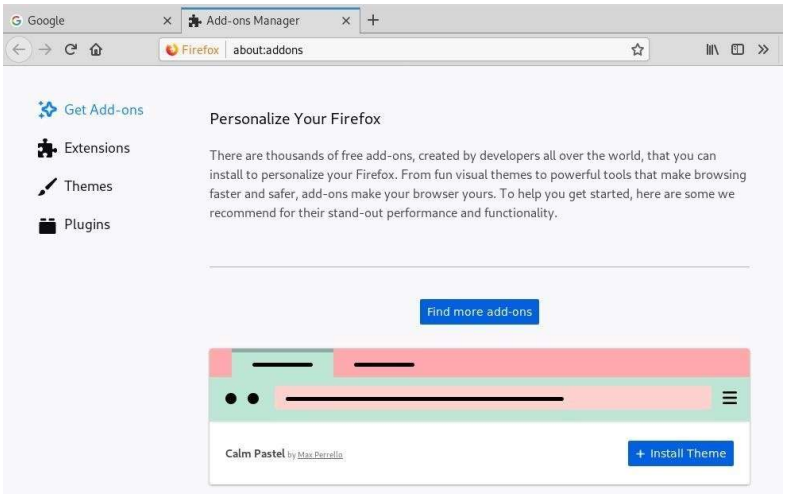


Congratulations!! At this point, you have successfully installed Burp Suite on your windows system.

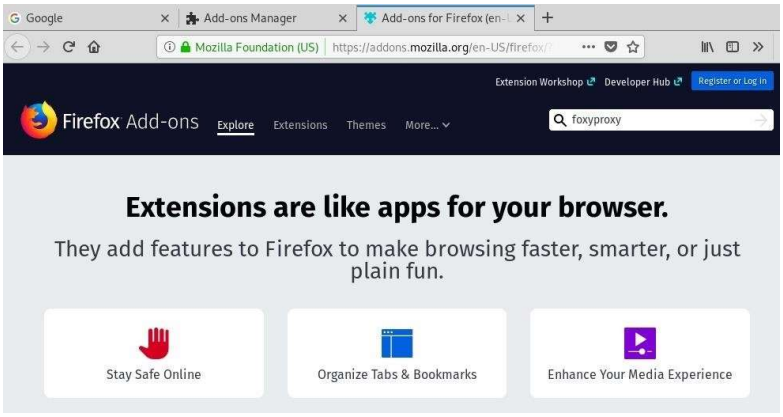
Configuration

Add Foxy Proxy to Firefox

The first thing we need to do is start Firefox and navigate to the Add-ons Manager. You can do so by using the Ctrl Shift p shortcut, clicking the "Open menu" button in the toolbar then "Add-ons," or hitting "Tools" in the menu bar followed by "Add-ons."

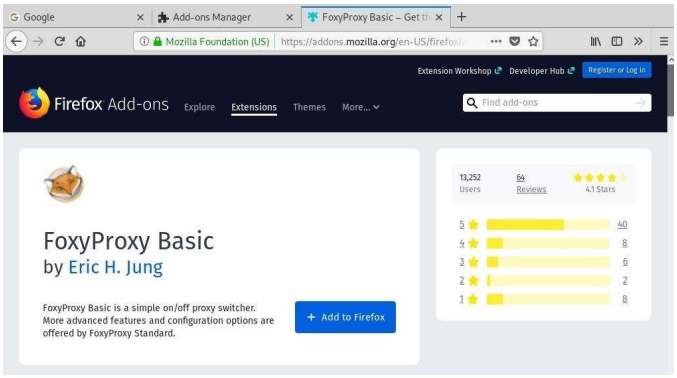


Click "Find more add-ons" on the Personalize Your Firefox page for "Get Add-ons," and search for Foxy Proxy.

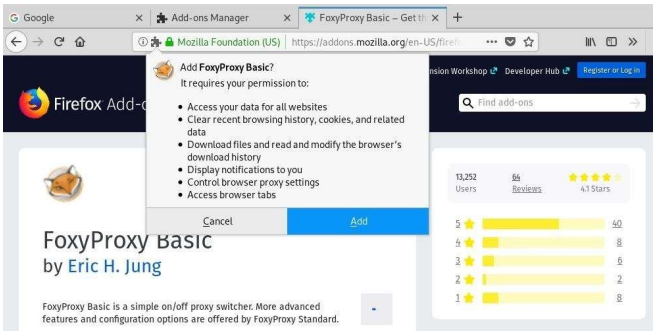


We will use Foxy Proxy Basic as it offers enough functionality for what we need. Alternatively, instead of going through all of the above steps, you can just go directly to FoxyProxy Basic's extension page.

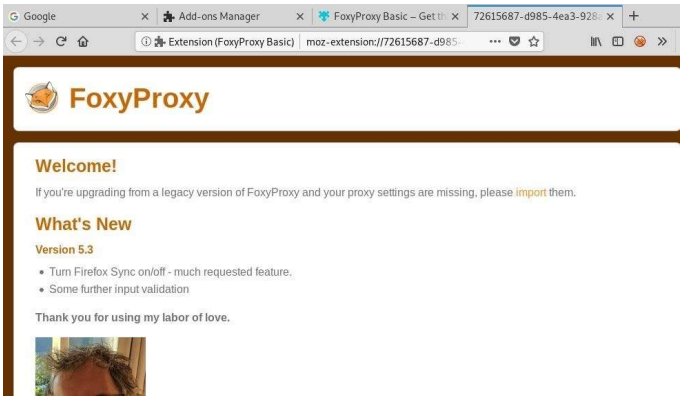
We can then click "Add to Firefox" to add the extension.



Make sure to hit "Add" on the prompt to allow access to what it needs.

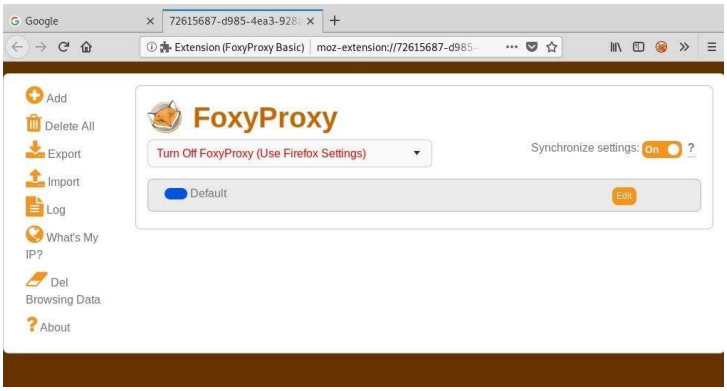


We will then be directed to FoxyProxy's page, which includes a changelog and a bit more information.

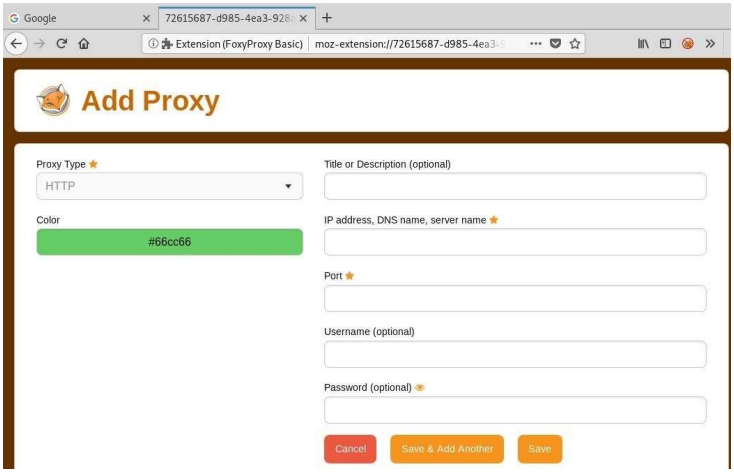


Add a Custom Proxy

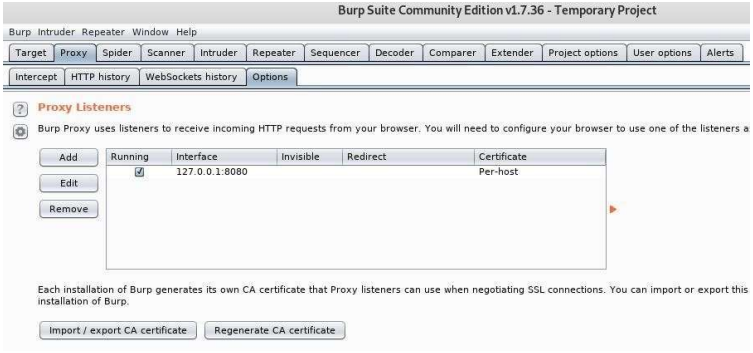
There should now be a little icon in the upper-right area of the browser, next to bookmarks or whatever else is in the toolbar. Click the icon and select "Options" to go to the settings page.



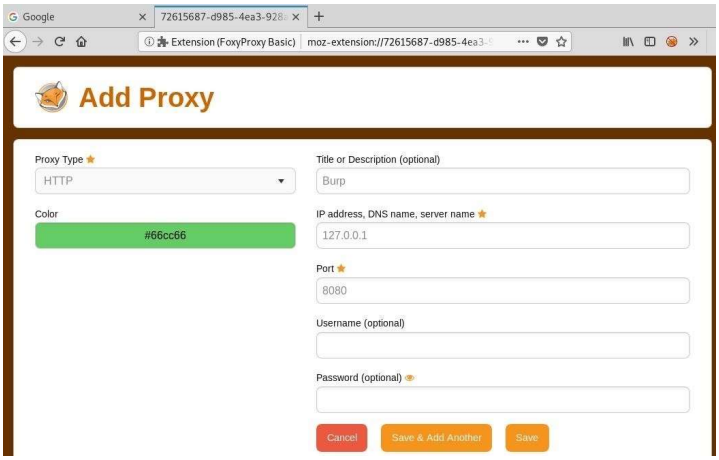
Next, click "Add" to add a custom proxy.



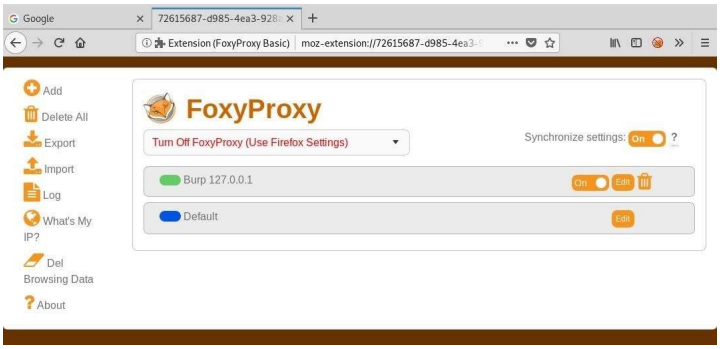
With Burp Suite up and running, go to the "Options" tab under "Proxy." We just want to confirm the default IP address and port since it needs to match in FoxyProxy.



Now we can fill in the information and give it a title to keep things organized.



Click "Save," and our proxy should now appear on the main settings page.



Now, all we have to do is enable it while Burp is running, allowing us to effortlessly switch the proxy on and off or even switch between different proxies. Click the icon and select "Use proxy Burp for all URLs (ignore patterns)" to turn it on.

Installing Burp's CA certificate in Firefox

Note

These steps are only necessary if you want to use your own external browser for manual testing with Burp. If you prefer, you can just use Burp's browser, which is preconfigured to work with Burp Proxy already. To access Burp's browser, go to the Proxy > Intercept tab, and click Open Browser.

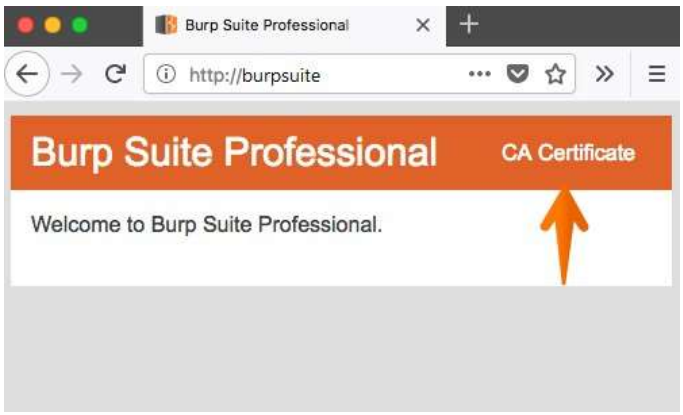
Note

If you previously installed a different CA certificate generated by Burp, you should remove it before installing a new one.

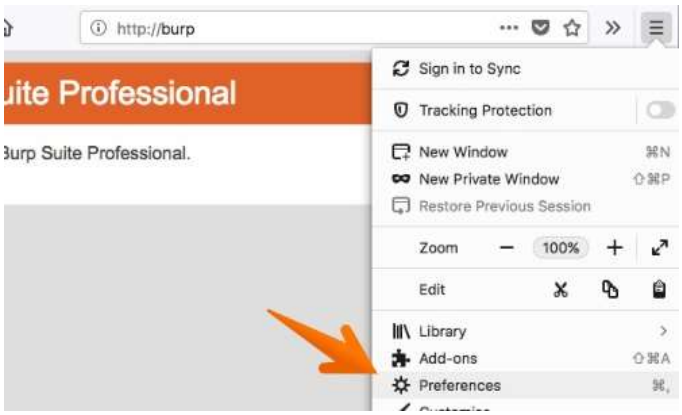
To install Burp's CA certificate in Firefox, proceed as follows:

With Burp running, visit http://burpsuite in Firefox. You should be taken to a page that says "Welcome to Burp Suite Professional". If not, please refer to the proxy troubleshooting page. Depending on what went wrong, you may be taken there automatically.

In the top-right corner of the page, click CA Certificate to download your unique Burp CA certificate. Take note of where you save this.

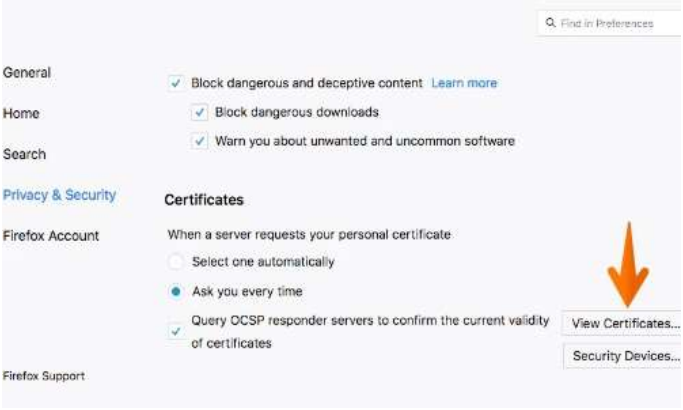


In Firefox, open the burger menu and click Preferences or Options.

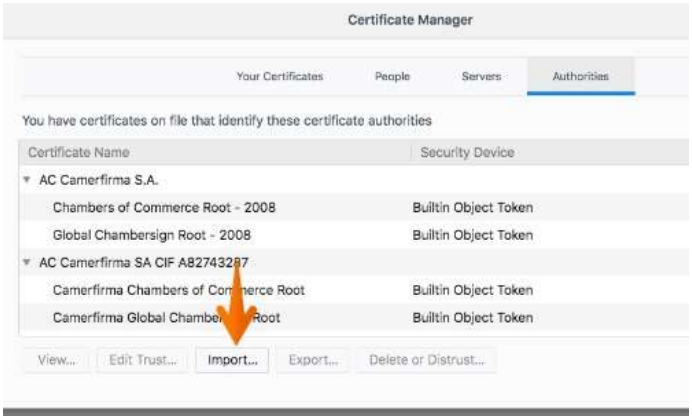


From the navigation bar on the left of the screen, open the Privacy and Security settings.

Scroll down to the Certificates section and click the View certificates button.

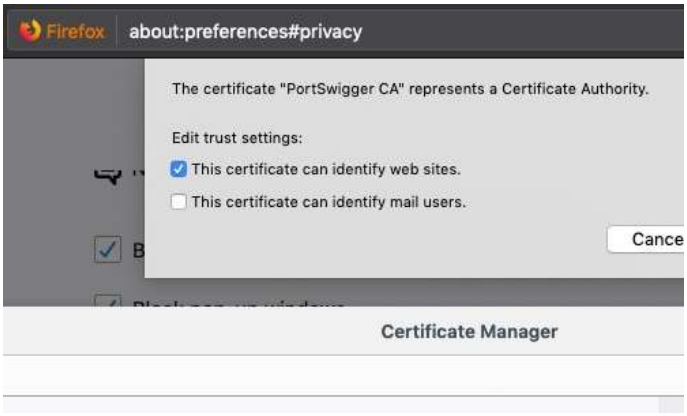


In the dialog that opens, go the Authorities tab and click Import. Select the Burp CA certificate that you downloaded earlier and click Open.



When prompted to edit the trust settings, make sure the checkbox This certificate can identify websites is selected and click OK.

Close and restart Firefox. With Burp still running, try and browse to any HTTPS URL. If everything has worked, you should now be able to browse to the page without any security warnings.



Removing Burp's CA certificate from Firefox

To remove Burp's CA certificate from Firefox, go back to the View certificates > Authorities dialog and select PortSwigger CA. Then, click Delete or Distrust, click OK, and restart Firefox.

Platform Authentication, Upstream Proxy

Servers, SOCKS Proxy Platform Authentication

There are certain scenarios where the application hosted on the target web server is protected by authentication. In such a case, we need to configure the credentials in Burp Suite. In the absence of credentials, Burp Suite won't be able to access the protected portion of the application and miss out on potential checks. To configure platform authentication, navigate to “User Options ➤ Connections ➤ Platform Authentication” as shown in Figure 3-11.

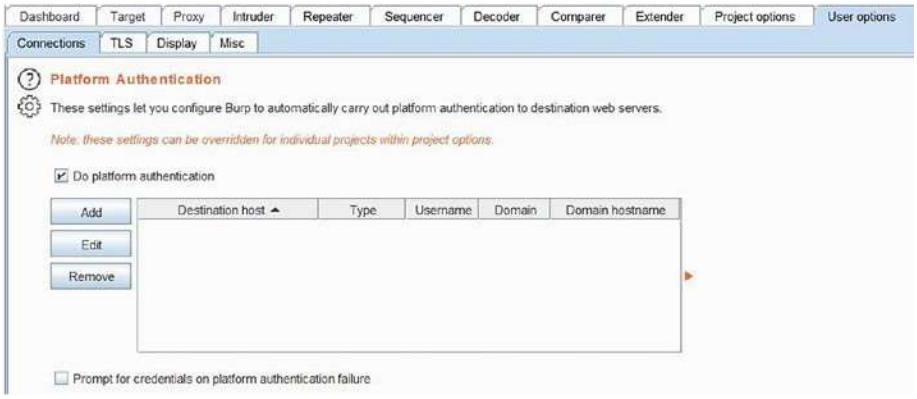


Figure 3-11. Configuring the Platform Authentication

Click on the ‘Add’ button and a pop-up window will appear as shown in Figure 3-12.

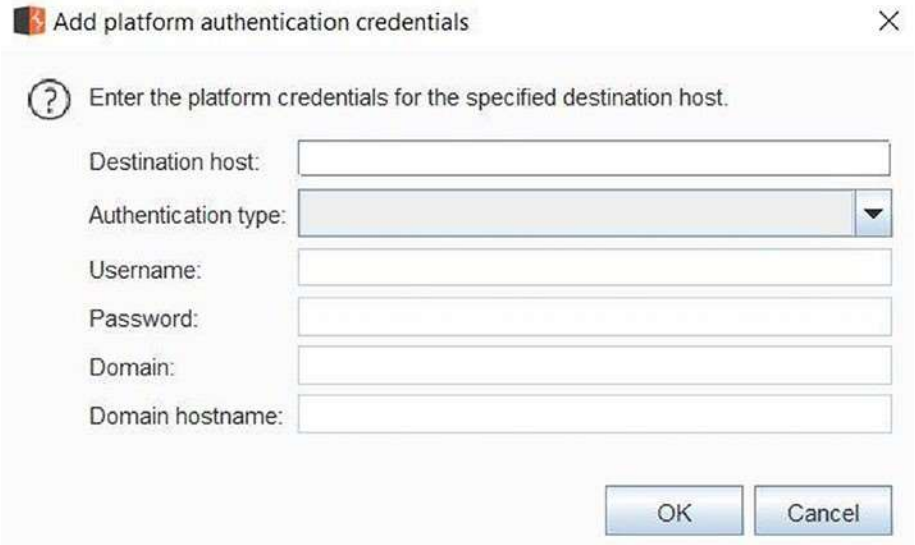


Figure 3-12. Setting up the Platform Authentication

We need to configure the destination host either in the form of an IP address or hostname, authentication type either of Basic, NTLM V2, NTLM V1, Digest, Username and Password, and the Domain if applicable. Once these settings are recorded, Burp Suite can seamlessly access the protected part of the application with the help of these credentials.

Upstream Proxy Servers

While testing applications in certain network environments, it may happen that there’s no direct access to that target application. In such a case, we might need to connect to a proxy server first and then connect to the target application. Burp Suite allows easy configuration of upstream proxy servers. Simply navigate to “User Options ➤ Connections ➤ Upstream Proxy Servers” as shown in Figure 3-13.

Figure 3-13. Configuring the Upstream Proxy Servers

Click on the 'Add' button as shown in Figure 3-14, and configure the required proxy settings.

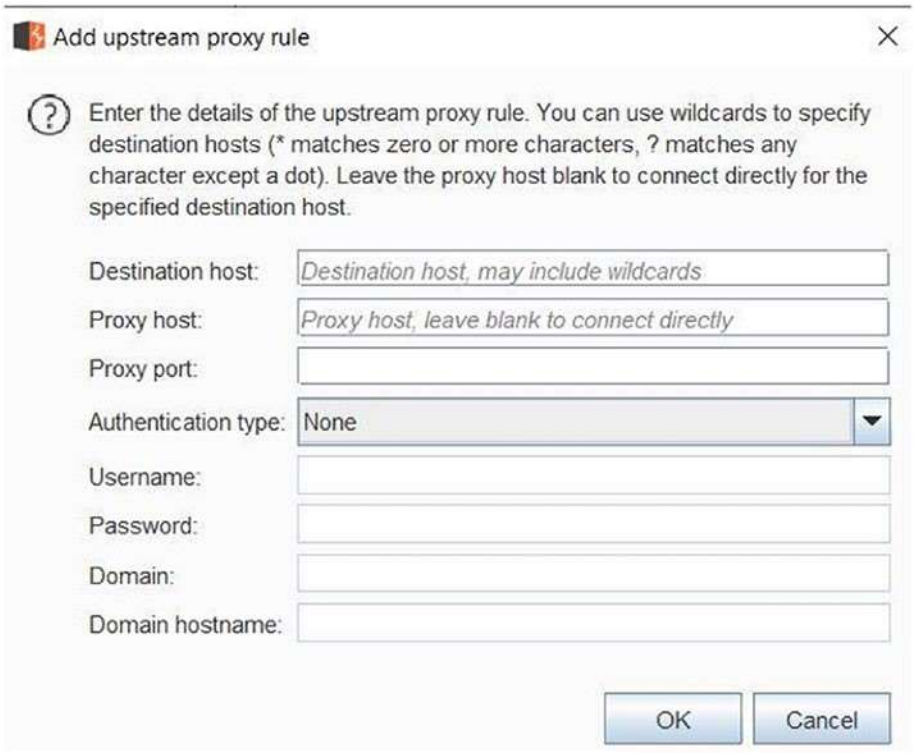


Figure 3-14. Adding the upstream proxy rule

SOCKS Proxy

Burp Suite also allows you to make all connection requests through a

SOCKS proxy. To configure Burp Suite with SOCKS Proxy, navigate to "User Options ➤ Connections ➤ SOCKS Proxy" as shown in Figure 3-15, and configure the required proxy settings.

Figure 3-15. Adding SOCKS proxy

Hotkeys

The Burp Suite tool has many tabs, tools, and options that we can work with. We'll be discussing them in detail in upcoming chapters. At first, the Burp Suite tools and tabs might seem overwhelming. But they all get familiar as you start using them. While all these tools and tabs can be accessed with the click of a button, at times while working on projects, it is much easier to use keyboard shortcuts than using mouse clicks.

The Burp Suite tool offers configuration of Hotkeys, which are nothing but the keyboard shortcuts to access certain tools or tabs. To configure

Hotkeys, navigate to User Options ➤ Misc ➤ Hotkeys as shown in Figure 3-16.

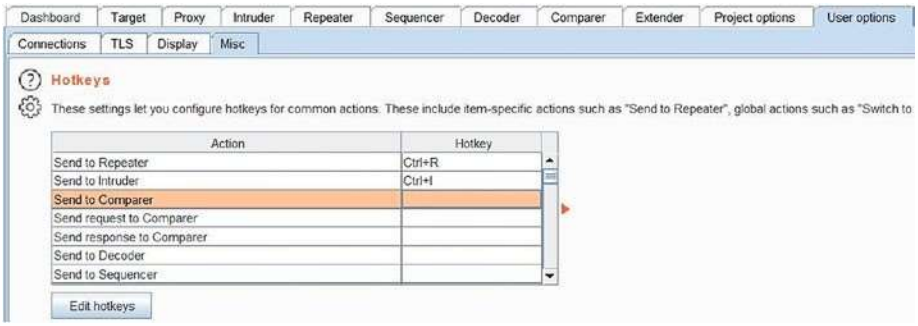


Figure 3-16. Configuring the Burp Suite Hotkeys

By default, Hotkeys for common functionalities within the Burp Suite are already configured. However, there's an option 'Edit hotkeys' either to change the default hotkeys or configure hotkeys for additional functionalities.

The Table 3-1 lists some of the common default hotkeys.

Table 3-1. Default Hotkeys in Burp Suite

Hotkey	Purpose
Ctrl + r	send to repeater
Ctrl + i	send to intruder
Ctrl + F	Forward intercepted proxy message
Ctrl + shift + t	switch to target
Ctrl + shift + p	switch to proxy
Ctrl + shift + i	switch to intruder
Ctrl + shift + r	switch to repeater
Ctrl + shift + o	switch to project options
Ctrl + shift + U	UrL decode
Ctrl + shift + B	Base-64 decode
Ctrl + B	Base-64 encode

Apart from the hotkeys in the above table, all other standard keyboard shortcuts for selecting all text, cut, copy, and paste work in the standard way.

Project Backups

While working on projects using Burp Suite, large amounts of data in the form of requests and responses get generated. It becomes necessary to save a copy of this data at regular intervals. Instead of doing this task manually, Burp Suite offers a feature to take a backup of data automatically after specified intervals.

To enable automatic backups, navigate to User Options ➤ Misc ➤ Automatic Project Backup as shown in Figure 3-17.

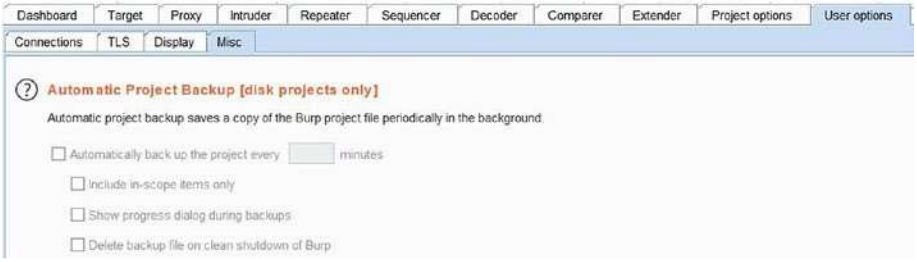


Figure 3-17. Configuring automatic project backups

Using this feature, we can specify the duration in minutes after which Burp Suite will automatically trigger the backup.

Rest API

While we use the Burp Suite mostly for manual application security testing, there could be so many other tools and use cases that need to work along with the Burp Suite. There could be other security tools that need to integrate with Burp Suite or certain custom automation scenarios as well that need to automatically trigger actions in Burp Suite.

For all such purposes, Burp Suite provides users with a REST API interface. REST stands for Representational State Transfer and API stands for Application Programming Interface. REST API is the most popular way of interconnecting different applications.

To enable the Burp Suite REST API, go to “User Options ➤ Misc” as shown in Figure 3-18.

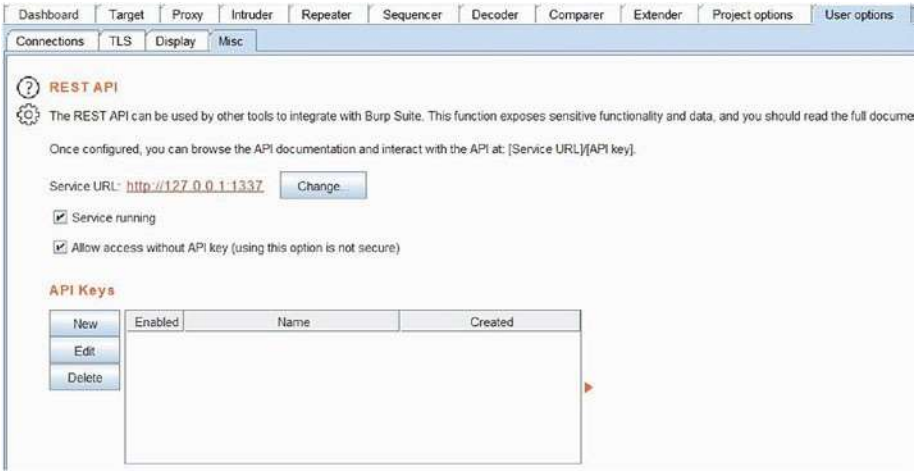


Figure 3-18. Setting up the Burp Suite REST API interface

To enable the Burp Suite REST API, simply check the option “Service running.” The REST API will be available by default on http://127.0.0.1:1337 as shown in Figure 3-19.

Figure 3-19. The Burp Suite REST API Interface

The REST API interface lists all the verbs or methods supported, the endpoints to be called, the parameters to be passed, and the expected responses. Now it completely depends on a particular scenario or use case on how this REST API interface can be utilized.

Performance Feedback

As like with any other tool, Burp Suite has a provision to capture and send certain diagnostic data that could be useful in improving Burp Suite’s performance. This is a completely optional feature, and if enabled it only collects data about internal functioning and not about specific users or project data. Burp Suite also provides features to log all exceptions or to report a specific bug to the Burp Suite team.

To use Performance Feedback options, navigate to User Options ➤ Misc ➤ Performance Feedback as shown in Figure 3-20.

Figure 3-20. Configuring the performance feedback

Project Options

These options include Hostname resolution, Out-of-Scope Requests, Redirections, TLS Configuration, Session Handling Rules, Cookie Jar. and Macros.

Timeouts

Handling requests and responses is the core functionality of Burp Suite. There could be scenarios like the target application is down, or there are connectivity issues wherein Burp Suite needs to decide how long it should wait for a response for a request before dropping it off. These settings are defined by the timeout values. They are configured by default and can be left untouched unless there's an explicit need to change the timeout values. For changing the default timeout values, navigate to "Project Options ➤ Connections ➤ Time Outs" as shown in Figure 3-21.

Figure 3-21. Configuring the request timeouts

Hostname Resolutions

Hostname resolution usually happens with the help of either the local host file or the network DNS. However, Burp Suite allows for custom hostname resolutions. This might be useful in particular scenarios where an application hosted on an intranet needs to be accessed using a specific hostname or URL.

To define custom hostname resolution rules, navigate to "Project

Options ➤ Connections ➤ Hostname Resolutions" as shown in Figure 3- 22.

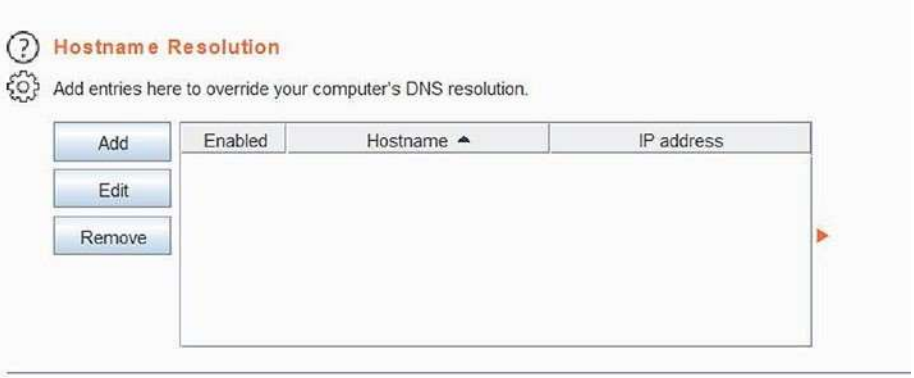


Figure 3-22. Configuring the hostname resolution

Click on the 'Add' button and then you'll get a pop-up window to enter a custom Hostname and associated IP address as shown in Figure 3-23.

Figure 3-23. Adding the hostname resolution rule

Out-of-Scope Requests

Once the browser is configured to work along with Burp Suite, Burp Suite will capture all the HTTP traffic across all tabs by default. This traffic can be overwhelming and distracting. Out of all the traffic that is captured, we need to concentrate only on the required target that we are testing. This can be achieved using a scope that we will be covering in a later chapter. Burp Suite provides a feature to simply drop all the requests that are out of scope. This can be done by navigating to "Project Options ➤ Connections ➤ Out-of-scope Requests" as shown in Figure 3-24.

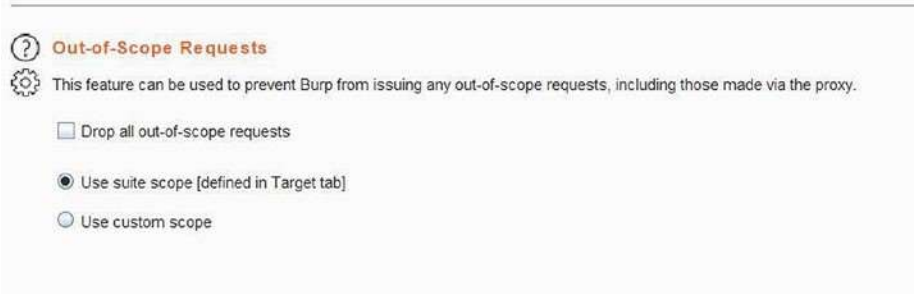


Figure 3-24. Configuring the rules for Out-of-Scope requests

Selecting the “Use custom scope” option we can explicitly add URLs that we wish to exclude from the scope and drop off the proxy as shown in Figure 3-25.

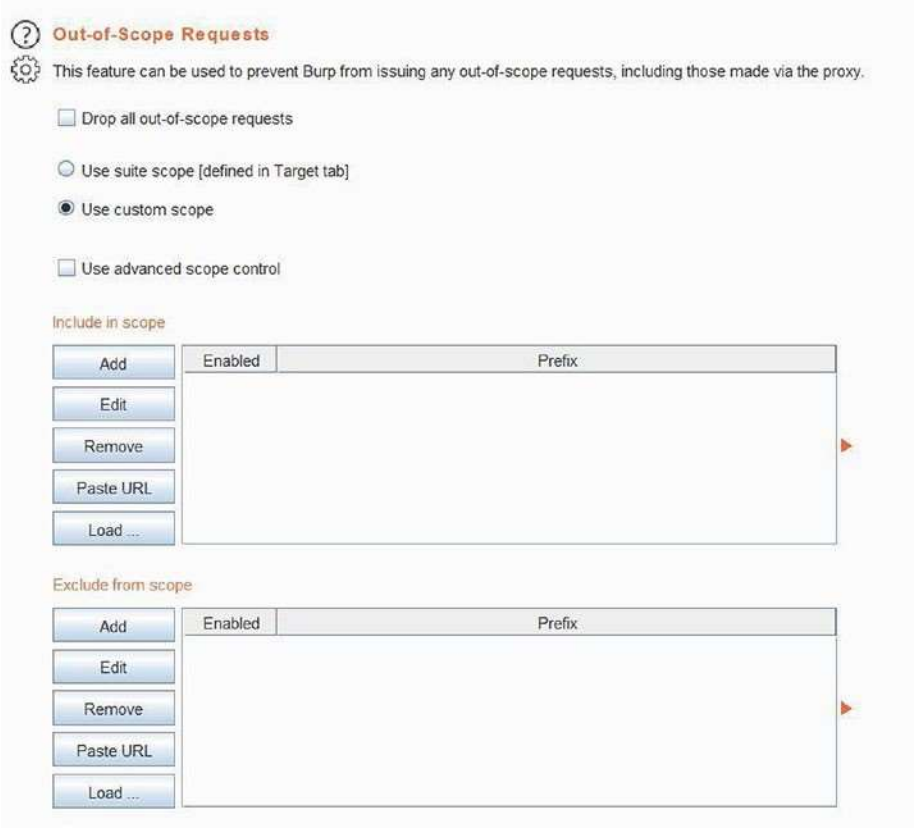


Figure 3-25. Defining rules for Out-of-Scope requests

Redirections

Automated application scanning requires processing of HTTP redirections. The scan engine has to take action once it detects any page redirection. Burp Suite has redirection rules configured by default and those can be left untouched unless there's an explicit need to change them, or there's a need for additional configuration of JavaScript-driven redirections. For configuring redirection rules, navigate to “Project Options ➤ HTTP ➤ Redirections” as shown in Figure 3-26.

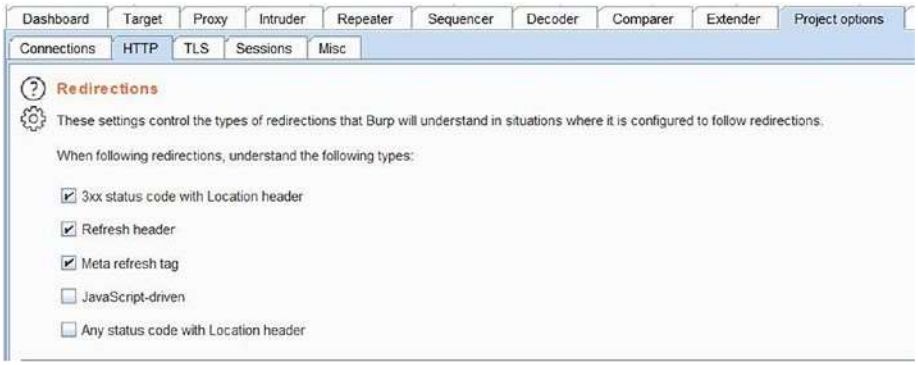


Figure 3-26. Configuring the redirections

Cookie Jar

Any application scanning / testing tool needs to maintain a repository of cookies that it will use to manage the ongoing application sessions. The in-session detection capability is specifically required when performing automated scanning. Burp Suite stores the application cookies in a container called “Cookie Jar.” By default, the Cookie Jar monitors the Proxy traffic to extract and store any cookies; however, we can explicitly instruct Burp Suite to monitor and capture cookies out of other tools like Scanner, Repeater, Intruder, Sequencer, and Extender. This can be done by navigating to “Project Options ➤ Sessions ➤ Cookie Jar” as shown in Figure 3-27.

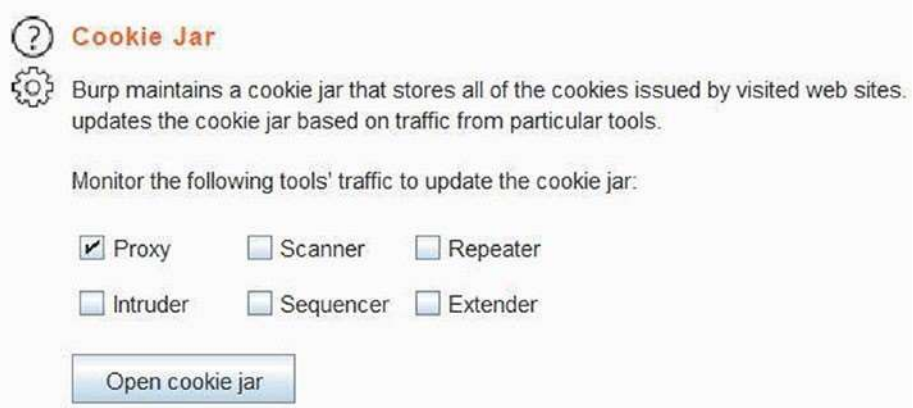


Figure 3-27. Configuring the Cookie Jar options

Macros

In the process of application security testing, it may be required to perform a certain sequence of actions repeatedly. Burp Suite provides an excellent functionality of macros to achieve this. The macro editor is available at "Project Options ➤ Sessions ➤ Macros" as shown in Figure 3-28. You can simply click on the 'Add' button and follow the wizard to record steps.

Figure 3-28. Configuring the macros

proxy, User options, and project options

Summary

In this chapter we learned about configuring the Burp Suite proxy along with a CA Certificate. We then glanced at several options like platform authentication, upstream proxy, socks proxy, etc. Next, we explored several other useful configurations including hotkeys, project backups, using Burp Suite API, project options, hostname resolutions, scoping, and redirections.

In the next chapter we'll explore the Burp Suite dashboard, target tab, and engagement tools.

Dashboard, Target, and Engagement Tools

In the last chapter, we saw some basics about configuring the proxy, user options, and project options. In this chapter we'll get started with getting familiar with the Burp Suite dashboard, target, and engagement tools.

Dashboard

Dashboard, as the name suggests, is that important part of Burp Suite that essentially summarizes different activities and tasks that are running across components. Figure 4-1 shows a typical view of the Burp Suite dashboard.

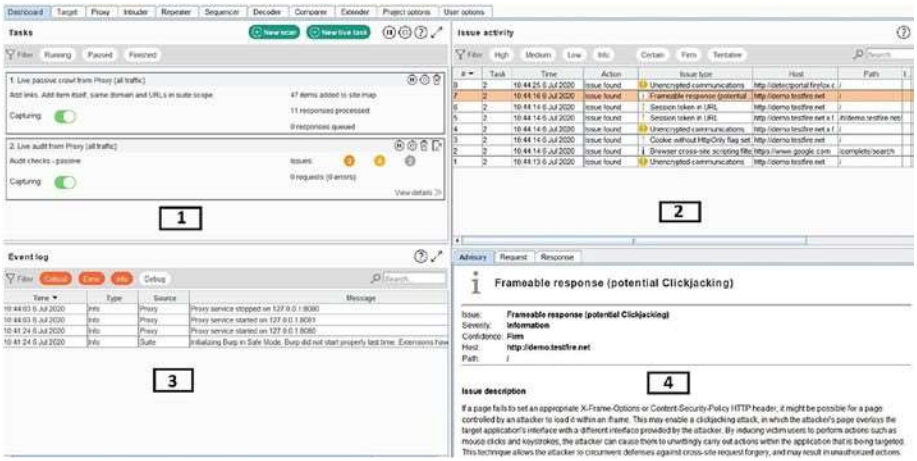


Figure 4-1. The Burp Suite Dashboard

For better understanding, we'll divide the dashboard into four parts and try to explain each in detail. For learning purposes, refer to the numbers from 1 to 4 in Figure 4-1.

The first part of the dashboard shows data around the ongoing scans as shown in Figure 4-2. The scans can be either passive or active. If there are multiple scans running, then we can filter them based on their state: running, paused, or finished. We also get to see a high-level summary of issues found in the scans. There's also an option to create a new scan task, which we will be exploring separately in an upcoming chapter.

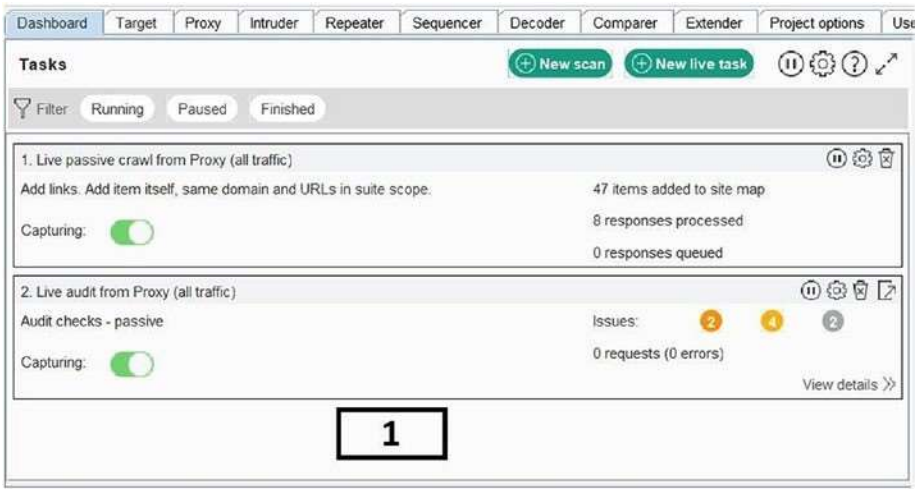


Figure 4-2. Tasks in the Burp Suite Dashboard

The second part of the dashboard is as shown in Figure 4-3. It might happen that either a passive scan or an active scan is running, and in either case the list of vulnerabilities found is highlighted in this section. This section also highlights information like time when the issue was found, the issue type, the host or target on which the issue was found, the vulnerable URL Path, severity of the issue, and the confidence level. Burp Suite may flag different confidence levels for different issues based on the responses; however, the issues need manual verification to ascertain their validity.

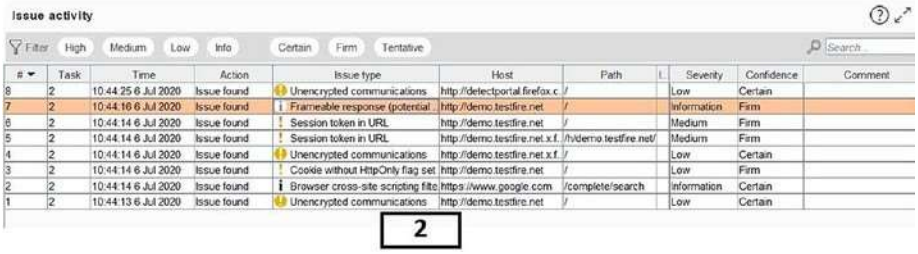


Figure 4-3. Issue activity in the Burp Suite Dashboard

The third part of the dashboard summarizes the Burp Suite functional events as shown in Figure 4-4. These mainly include the status of the proxy service, TLS connection failures (if any), authentication failures, timeouts, etc. For

example, if your system already has port 8080 configured with some other service, this part of the dashboard will highlight that proxy service couldn't be started on port 8080. Or if for any reason the proxy service stops, then it will also get highlighted here. Overall, it gives a picture about whether the Burp Suite proxy and related services are running properly or not.

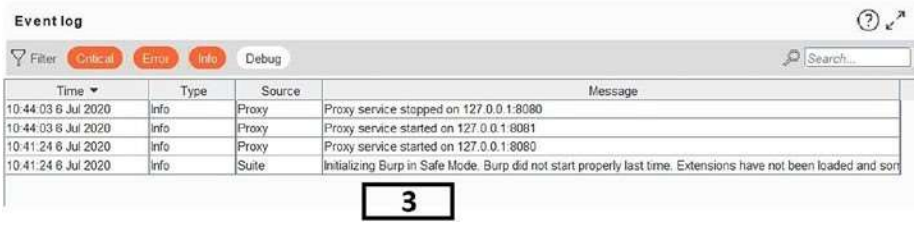


Figure 4-4. The event log in the Burp Suite Dashboard

The last part of the dashboard as shown in Figure 4-5 highlights issue details. If you wish to see details of any of the issues highlighted during the passive or active scan, then simply click that issue as shown in Figure 4- 3 and details will be available accordingly. The issue details include the complete issue description and remediation recommendation, along with the actual request and response.

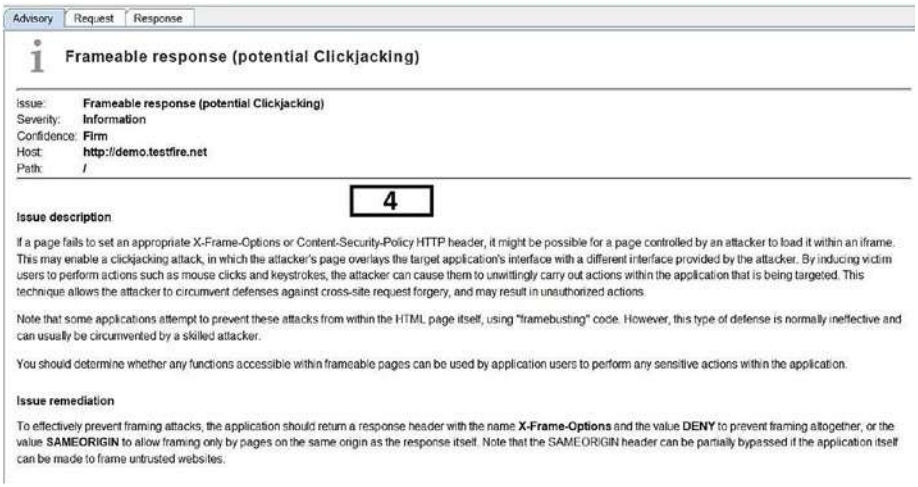


Figure 4-5. Issue details in the Burp Suite Dashboard

Target Tab

Like the Burp Suite dashboard that we saw in the previous section, the Target tab is an equally important work area. The target tab again has multiple panes as shown in Figure 4-6.

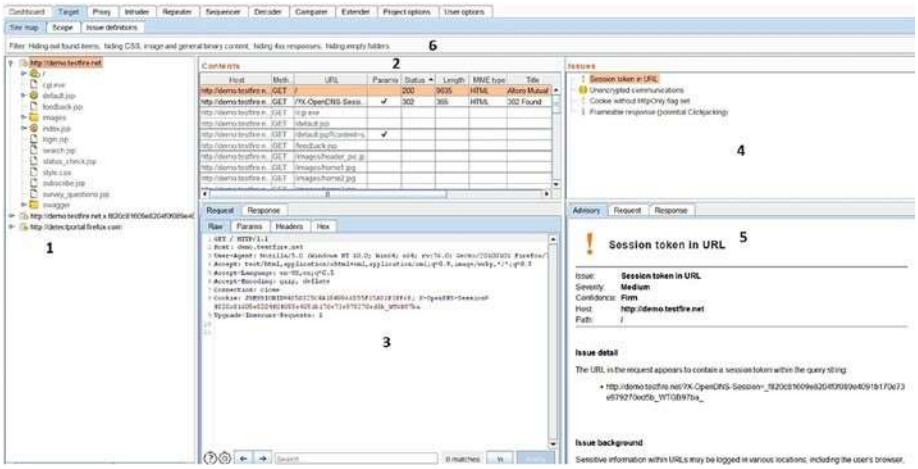


Figure 4-6. Target tab in Burp Suite

To understand different parts of the target tab, let's take one part at a time referring to the numbers 1 to 6 as per Figure 4-6.

The leftmost section in the target tab numbers as '1' in Figure 4-6 is shown in Figure 4-7. This section creates the hierarchy of the site that we are browsing through. It lists all the leaf nodes, folders, etc., in the form of a well-defined tree structure. This helps in getting an idea about how big the site / application could be or what its contents are in common. It is very similar to a sitemap.

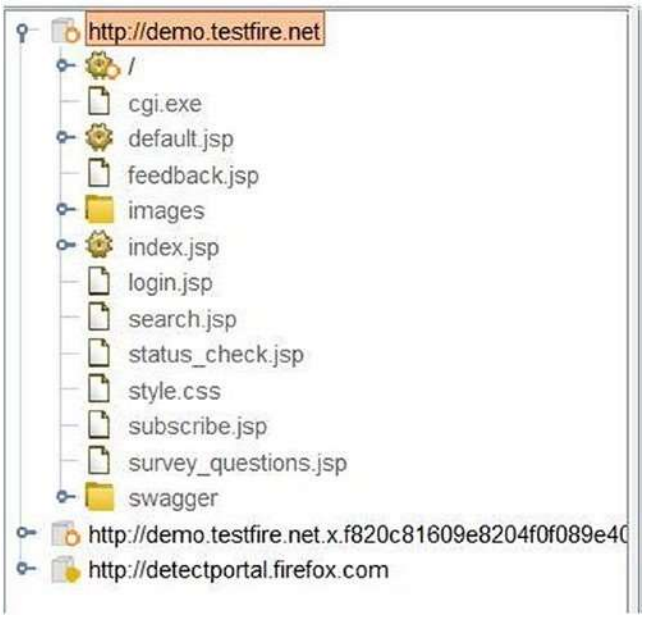


Figure 4-7. Application map / hierarchy

The next section numbered as '2' as per Figure 4-6 is shown in Figure 4-8. This section lists down all the HTTP requests that were made along with other details like the exact host, HTTP method used, target URL, if the URL had any parameters, HTTP status response code, content length, MIME type, and the title of the page, if any. Simply going through this section can help you find interesting URLs especially those having parameters to inject.

Contents

Host	Meth...	URL	Params	Status ▲	Length	MIME type	Title
http://demo.testfire.n...	GET	/		200	9635	HTML	Altoro Mutual
http://demo.testfire.n...	GET	/?X-OpenDNS-Sessi...	✓	302	365	HTML	302 Found
http://demo.testfire.n...	GET	/cgi.exe					
http://demo.testfire.n...	GET	/default.jsp					
http://demo.testfire.n...	GET	/default.jsp?content=s.	✓				
http://demo.testfire.n...	GET	/feedback.jsp					
http://demo.testfire.n...	GET	/images/header_pic.jp.					
http://demo.testfire.n...	GET	/images/home1.jpg					
http://demo.testfire.n...	GET	/images/home2.jpg					

Figure 4-8. List of requests

The next section numbered as ‘3’ as per Figure 4-6 is shown in Figure 4-9. This section shows the actual HTTP request and response. You can see the raw request by default but also see the request in form of parameters. You can also see all of the headers used in the response.

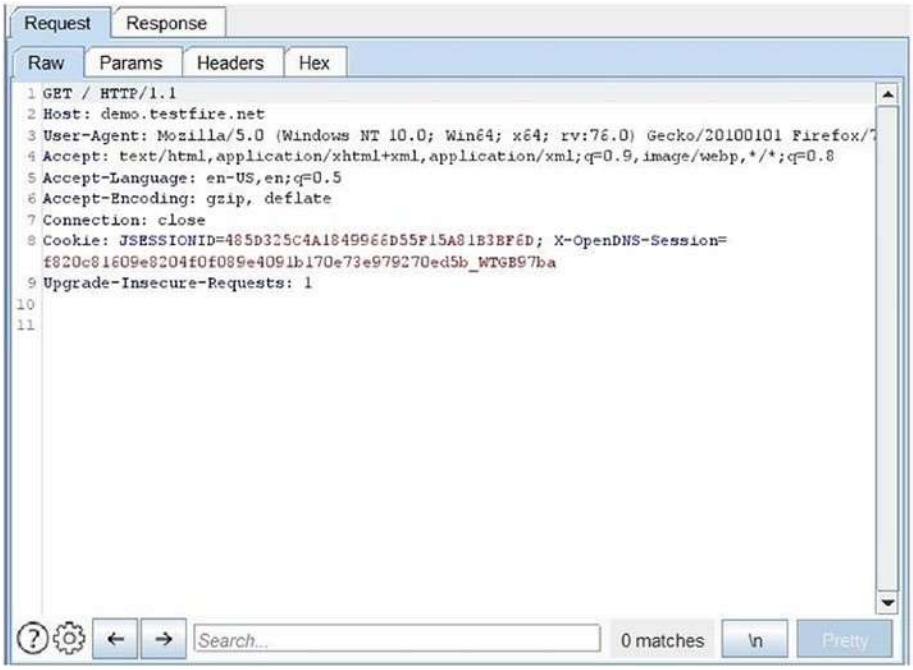


Figure 4-9. The request and response viewer

The next section numbered as ‘4’ as per Figure 4-6 is shown in Figure 4-10. This section shows the list of issues that were found either during the passive scan or active scan. The issues are classified as per the severity, with highest severity issues being shown at the top.

Figure 4-10. Issues found in the target application

The next section numbered as '5' as per Figure 4-6 is shown in Figure 4-11. This section shows the issue details for any of the selected issues. It contains the issue description along with remediation recommendations. In this section it is also possible to view the actual request and response based on which the issue was highlighted.

Figure 4-11. Issue details

The next section numbered as '6' as per Figure 4-6 is shown in

Figure 4-12. Once we configure our browser to work along with the Burp Suite, a lot of traffic may get captured. Hence to filter out only the required data, several filters can be used. Some common and useful filters include filtering by request type, MIME type, status code, extension, etc. This feature also allows searching for a particular item within the data collected through proxy.

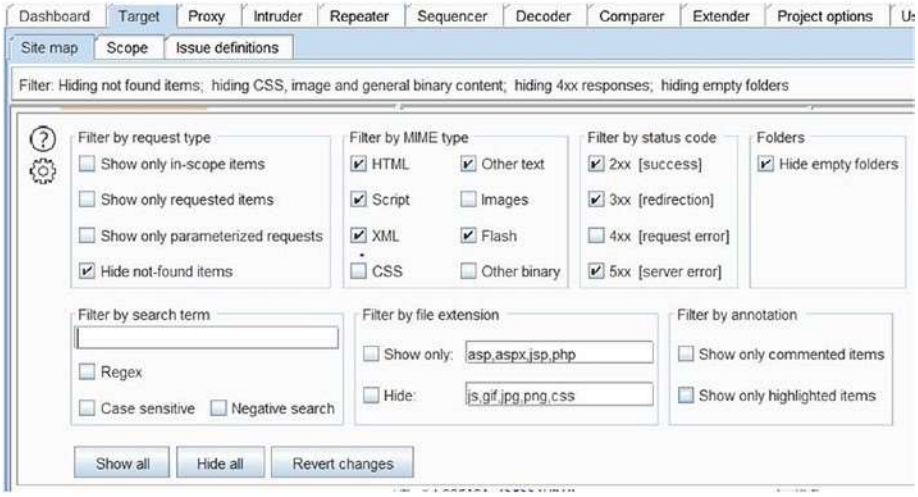


Figure 4-12. Site map filters

Engagement Tools

Engagement tools are nothing but small utilities that help in performing some additional tasks within Burp Suite. In this section we'll go through several such engagement tools serving different purposes. To access the engagement tools, simply right-click the target URL against which you wish to run the engagement tools as shown in Figure 4-13.

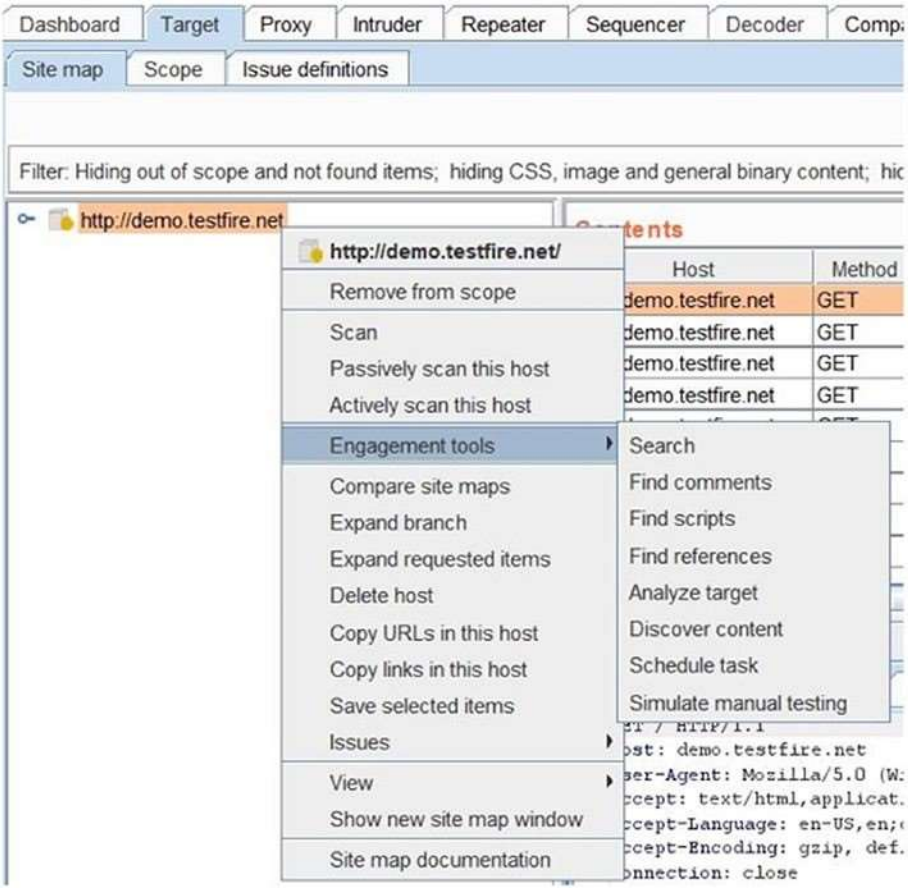


Figure 4-13. The Burp Suite Engagement Tools

The first engagement tool is the simple search as shown in Figure 4- 14. This allows users to search for any keyword within the requests or responses from the target selected.

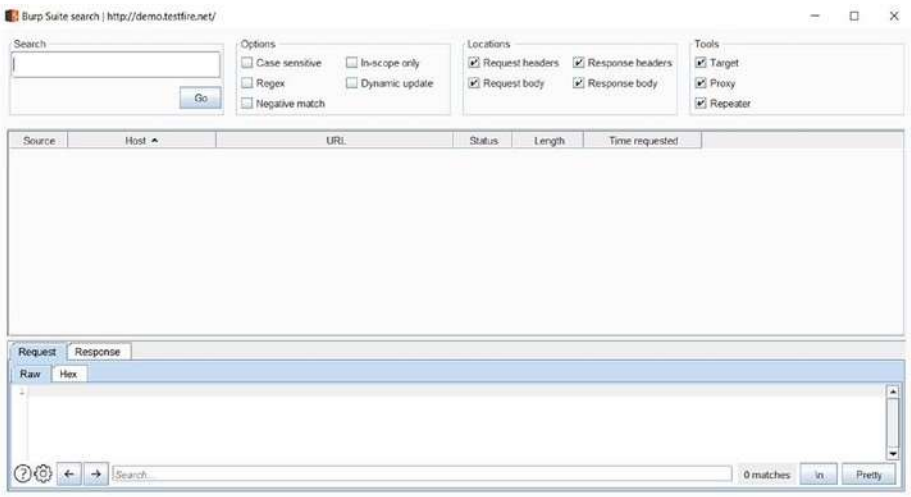


Figure 4-14. Search tool

The next engagement tool is the 'Find comments' as shown in Figure 4-15. This simply crawls through the collected data and finds any code comments. It is worthwhile to go through these comments as there's always a possibility of finding something sensitive in the comments.

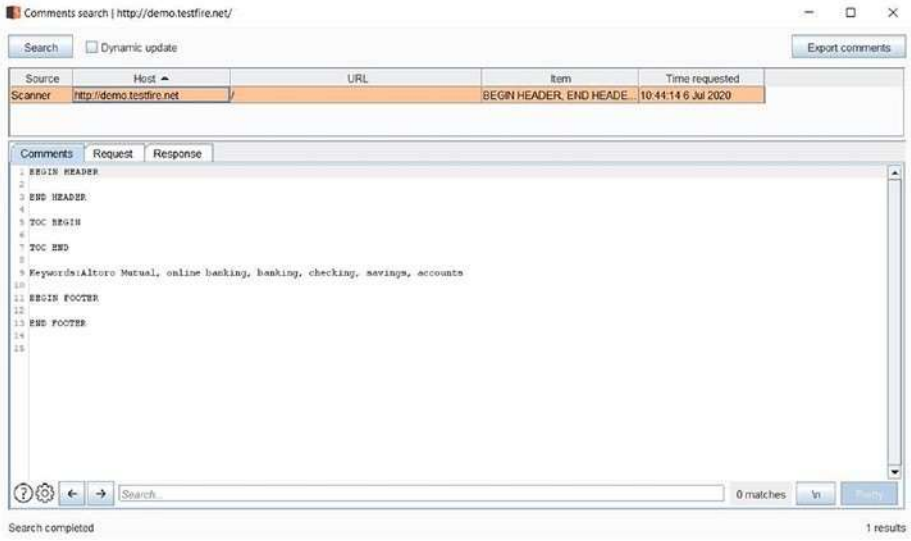


Figure 4-15. Comment finder tool

The next engagement tool is the ‘Find scripts’ as shown in Figure 4-16. This tool simply searches for all scripts within the scope of the target being selected.

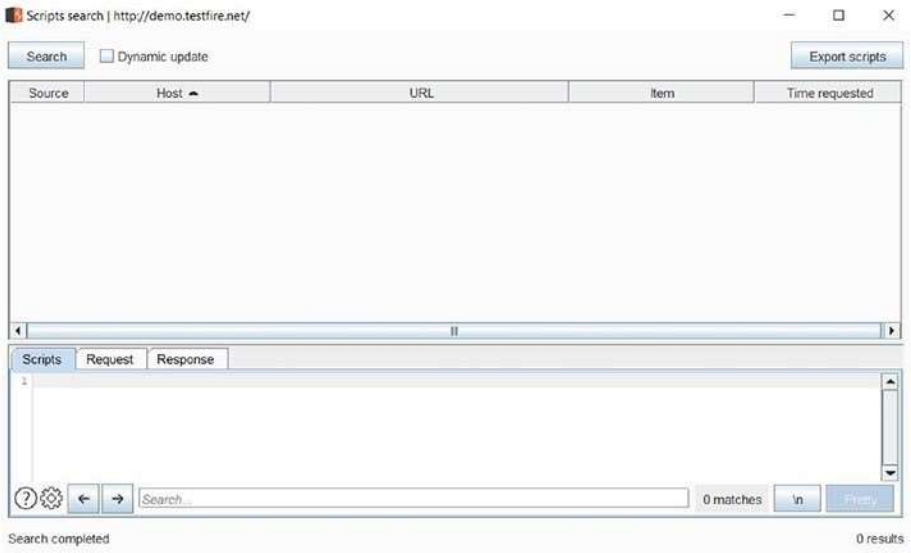


Figure 4-16. Script searching tool

The next engagement tool is the ‘Find references’ as shown in Figure 4- 17. This tool lists down all the Burp Suite components where it is able to find any reference of the selected target. For example, the URL ‘demo.testfire.net’ appeared in Scanner as well as Target within the Burp Suite as shown in Figure 4-17.

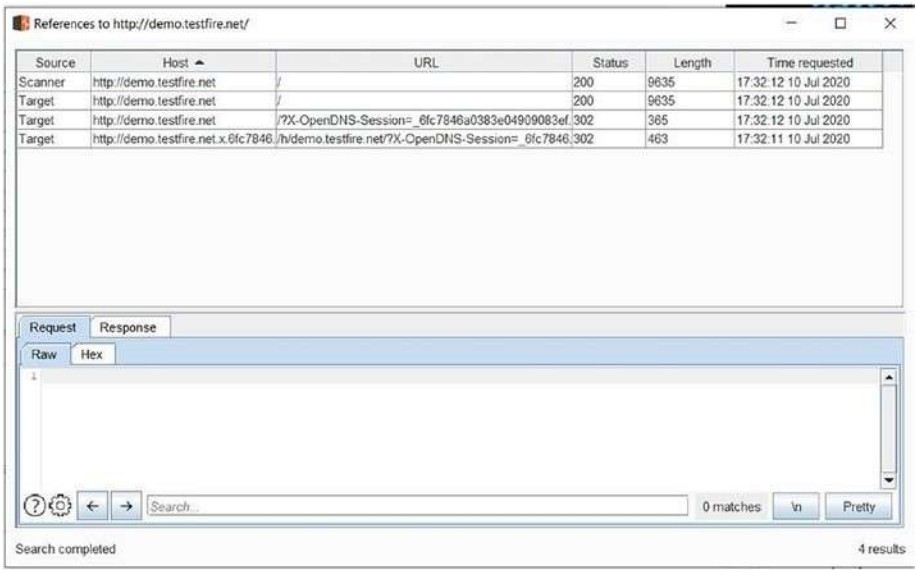


Figure 4-17. Reference finder tool

The next engagement tool is the 'Target Analyzer' as shown in Figure 4- 18. This utility provides statistics mostly around sizing of the application in terms of the number of dynamic URLs, number of static URLs, number of parameters, etc. These statistics help the tester to get an estimate of effort that will be required to test the application.

Figure 4-18. The target analyzer

The next engagement tool is the ‘Content Discovery’ as shown in Figure 4-19. This tool helps define the spidering or crawling rules. For instance, this helps define spidering rules when a new target is discovered, like how much length and breadth of it should be spidered or whether to spider only files or directories as well. This creates a well-structured sitemap. This is optional and can be left as the default.

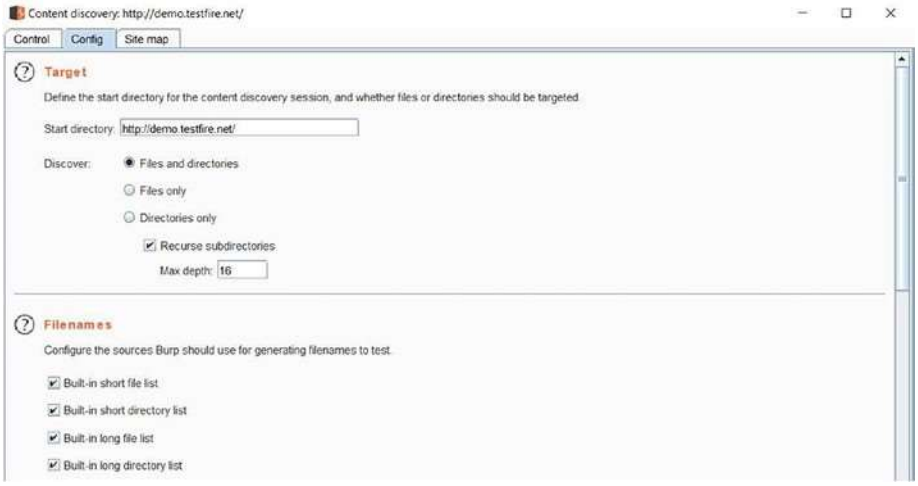


Figure 4-19. The content discovery tool

The next engagement tool is the ‘Schedule Task’ as shown in Figure 4- 20. This is a simple utility to schedule any of the Burp Suite tasks. Using this you can either pause or resume any of the tasks.

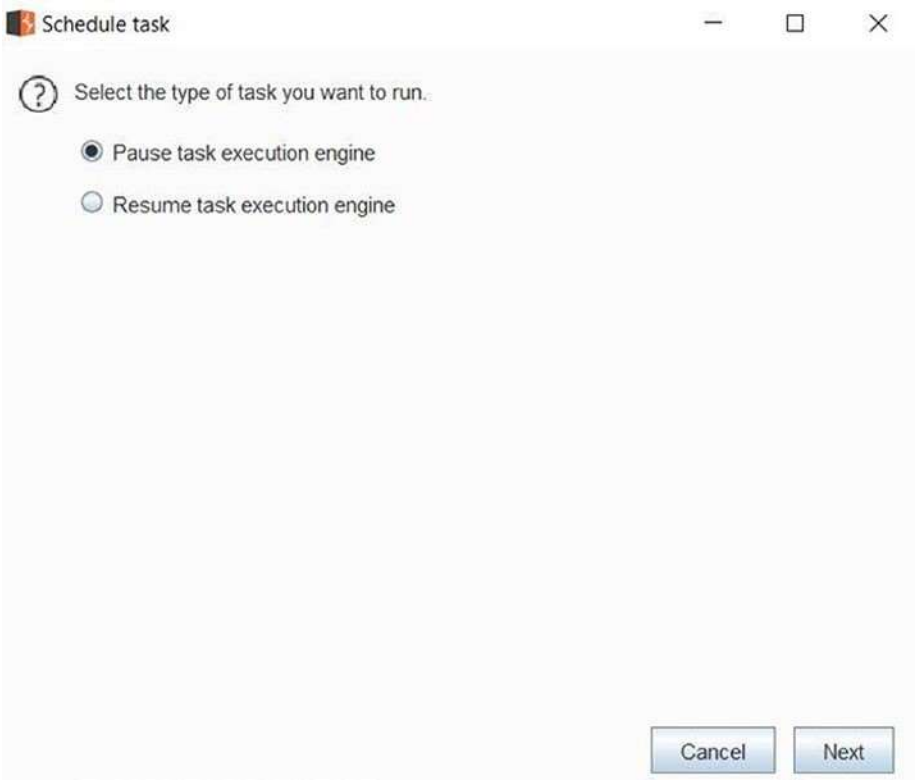
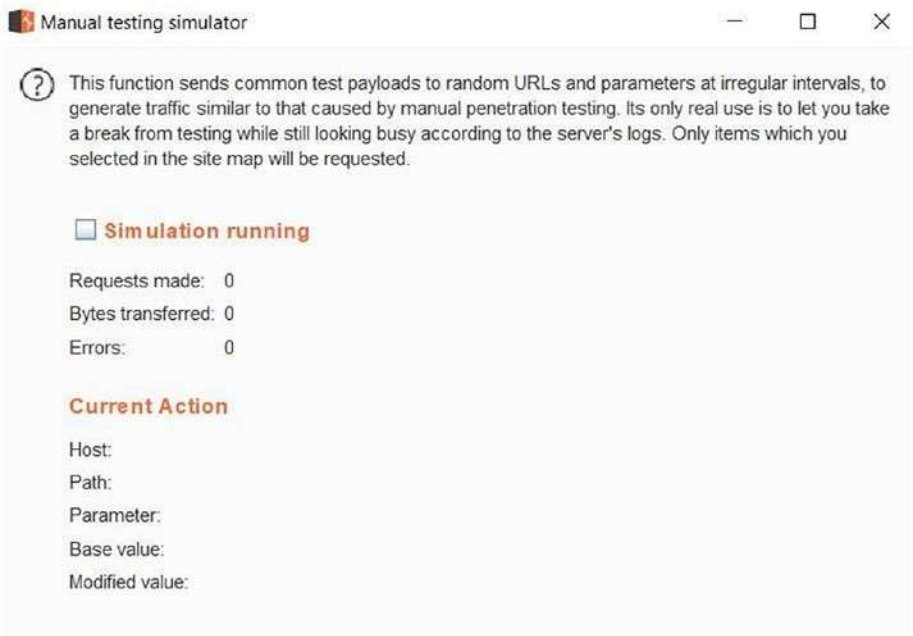


Figure 4-20. The Burp Suite task scheduler

The next engagement tool is the ‘Manual Testing Simulator’ as shown in Figure 4-21. This tool will send out HTTP requests to the target at random intervals and can help keep the session alive. This can be helpful in a scenario where the tester is on break and there’s a possibility of the application session getting timed out due to inactivity.

Dashboard, target, and engagement tools



Summary

In this chapter we learned the basics of the Burp Suite dashboard, target tab, and also discussed several useful engagement tools.

In the next chapter we'll learn about how the Burp Suite Intruder can be used to automate attack scenarios like Brute Force, etc.

Intruder

In the last chapter, we saw some basics about the Burp Suite dashboard, target, and engagement tools. Now that we have seen the basics of intercepting requests and interpreting the summary on the dashboard, we will move ahead with using the Intruder tool. Intruder has advanced fuzzing capabilities that can be used in various attack scenarios.

Introduction to Intruder

Before we get into the details of various options within Intruder, it's important to understand what Intruder is and how it can be helpful in web application security

testing. Intruder is part of Burp Suite, which can be used effectively for fuzzing and performing a brute force attack.

There might be an application with a login page wherein the user needs to enter credentials to proceed further. From a security testing perspective, it would be worthwhile to test this login page for default credentials, weak passwords, or lockout mechanisms. This is where Intruder can come in handy. Given a list of usernames and passwords, Intruder can try all those combinations to see if any of them match.

We can also consider another scenario wherein we have an interesting request that we wish to investigate further to check if it's vulnerable to SQL injection or cross-site scripting. Again, Intruder can help with this. We can simply point Intruder to the URL and parameter we wish to test and feed it with a list of SQL injection or cross- site scripting payloads. It will then try and insert all the payloads we provided into the parameter we want to test and get us the responses. Once this is done, we need to check the responses to see if any of the payload actually resulted in exploitation of the vulnerability.

Thus, Intruder tremendously helps in any of the test scenarios where we have two of the following things:

A URL and a parameter to test

List of payloads to be submitted to the parameter

Now let's try to understand how we can send a request to Intruder. We have already seen the target tab and the hotkeys in previous chapters. Any request can be sent to Intruder in two ways:

Right-click the request you wish to send and click on 'Send to Intruder' as shown in Figure 5-1.

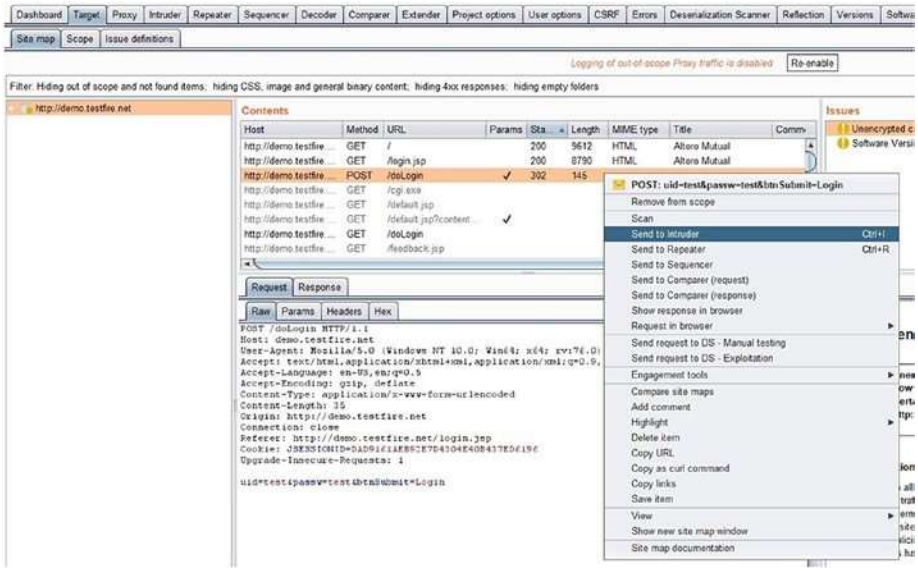


Figure 5-1. Send request to Intruder

Select the request you want to send and press the hotkey combination ‘Ctrl + I’.

Now that we have sent the request to Intruder, let’s see what options need to be configured further.

Target Tab

The first tab in Intruder is the Target tab. This lists the target URL and port that we wish to attack through Intruder as shown in Figure 5-2.

Figure 5-2. Configuring the attack target in Intruder

There’s also an option to use HTTPS in case the target URL is using a secure communication channel.

Positions

The next tab within Intruder is the Positions tab as shown in Figure 5-3.

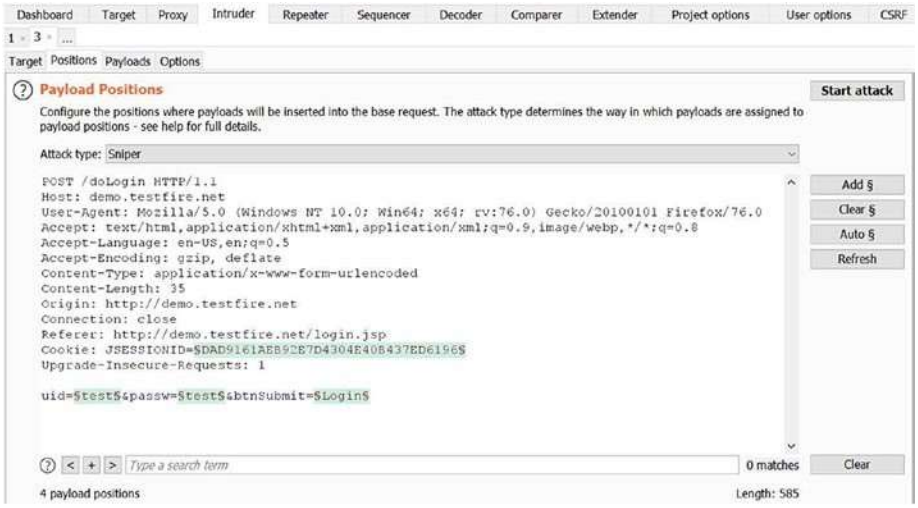


Figure 5-3. Configuring the positions in Intruder

Whenever a request is sent to Intruder, it scans the request for probable insertion points and marks them as variables preceding and ending with the '$' sign. There are three options with regard to selecting the insertion points:

Add $ – This option is used to add a new insertion point. Simply point the cursor to the start and end of the insertion point and click on 'Add $'.

Clear $ – This option will simply remove all the insertion points that were either selected manually or automatically.

Auto $ – This option will scan the request and try to automatically set insertion points marking them with the '$' sign.

Once we are sure about the insertion points or parameters that we want to target, the next step is selecting the type of attack. There are four different attack types available as shown in Figure 5-4.

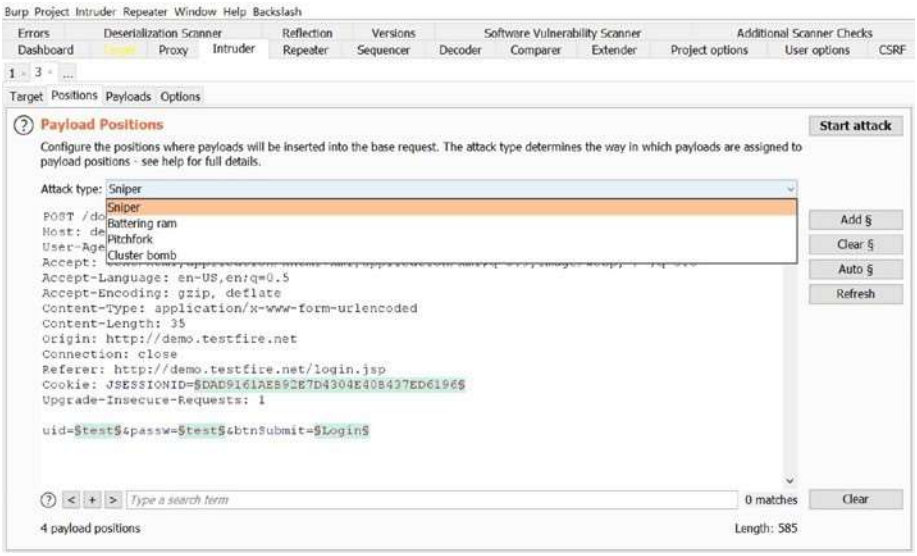


Figure 5-4. Selecting the attack type in Intruder

The four attack types are the following:

Sniper – This type of attack uses a single set of payloads. In this case Intruder inserts payloads into each of the insertion points at once and then iterates through it.

Battering ram – This type of attack uses a single set of payloads. In this case Intruder iterates through payloads by inserting the same payload at all insertion points at once.

Pitchfork – This type of attack uses multiple sets of payloads. In this case Intruder uses different payload for each of the insertion points.

Cluster bomb – This type of attack uses multiple sets of payloads. For each of the defined insertion points, there's a different payload set. Intruder iterates through each of the payload sets and all permutations of payload combinations are then tested. Due to the number of possible permutations and combinations in the case of a cluster bomb, a large number of requests would be generated.

Choosing the correct attack type depends on the attack scenario and the number of insertion points that need to be targeted simultaneously. See Figure 5-5.

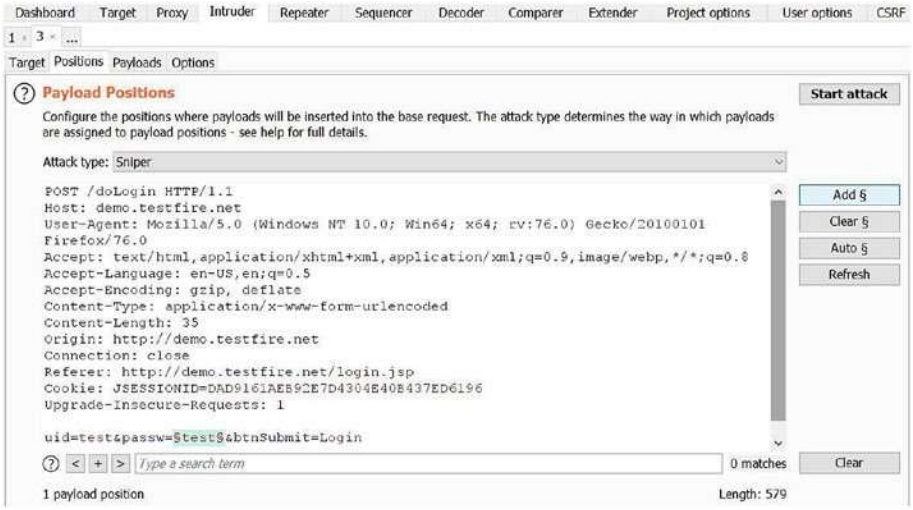


Figure 5-5. Attack type and positions in Intruder

Once the payload positions are configured and type of attack is selected, we can move ahead to configuring the actual payloads.

Payloads

Payload is the data that Intruder would iteratively insert in the selected insertion points. Payloads can differ widely based on the scenario or the attack that we are targeting. In the case of the login page that we are discussing, the payload would be a list of probable passwords. Burp Suite provides various payload types and the most commonly used one is the list. You can create your own list by adding elements one at a time as shown in Figure 5-6 or you can also select a predefined list that Burp Suite offers readily.

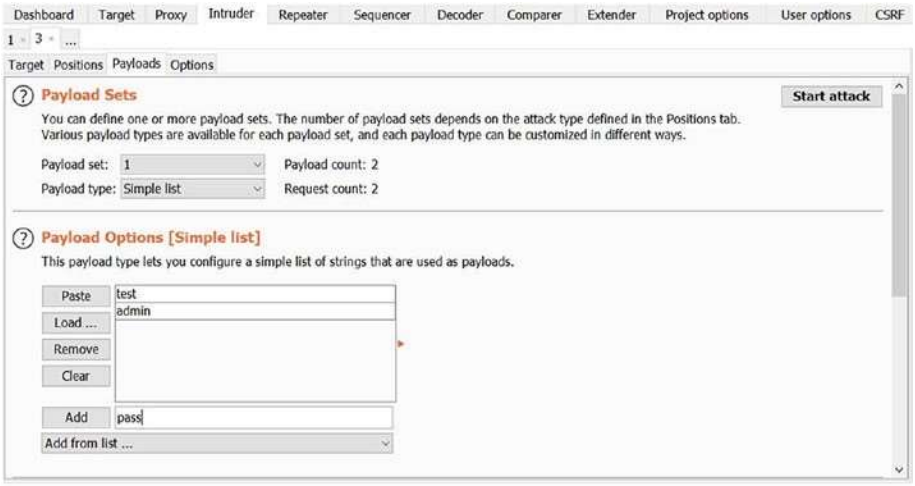


Figure 5-6. Selecting payloads in Intruder

Burp Suite has several predefined lists in the form of usernames, passwords, short words, fuzzing payloads for SQL injection and cross-site scripting, directories, extensions, etc. Depending on the type of attack, we can either use the predefined list or create our own list as shown in Figure 5-7.

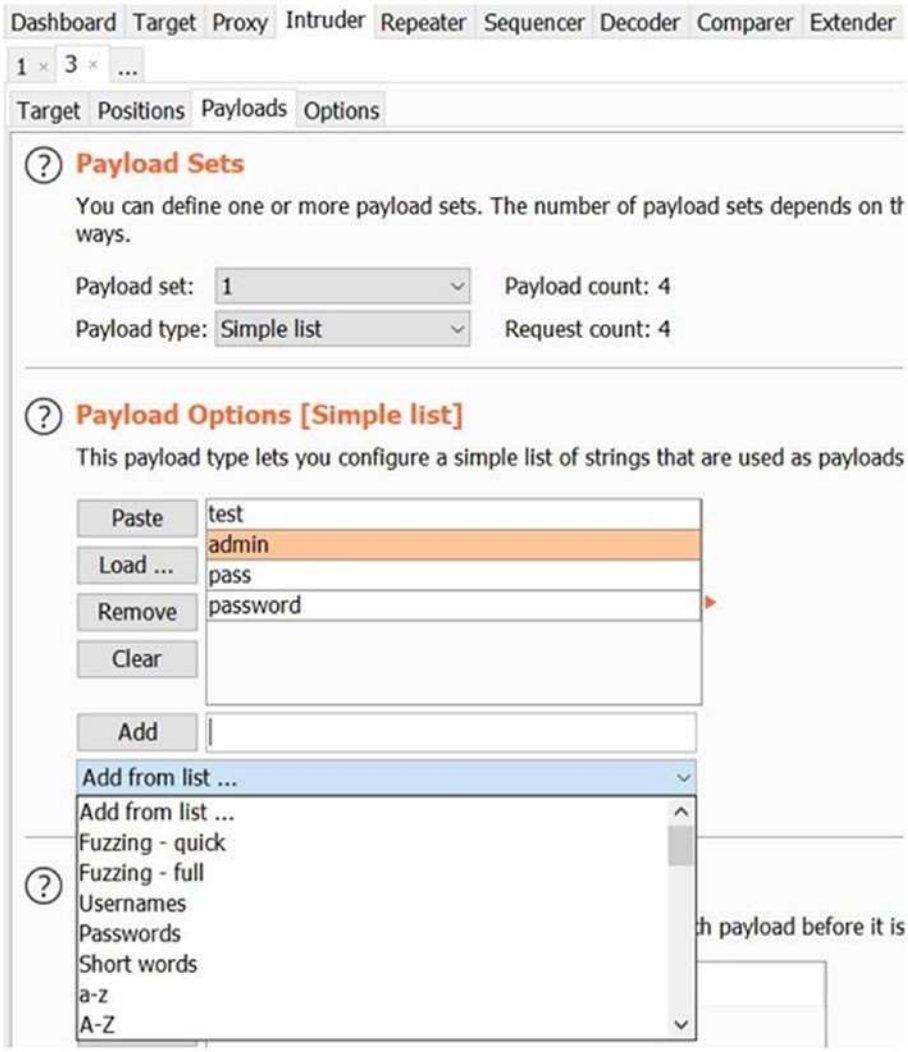


Figure 5-7. Selecting Intruder payloads from various options

Now that we have configured the positions as well as the payloads, we can launch the attack by clicking the 'Start attack' button. A new window will pop up as shown in Figure 5-8, and the payloads we provided will be submitted in insertion points we defined earlier – one request at a time.

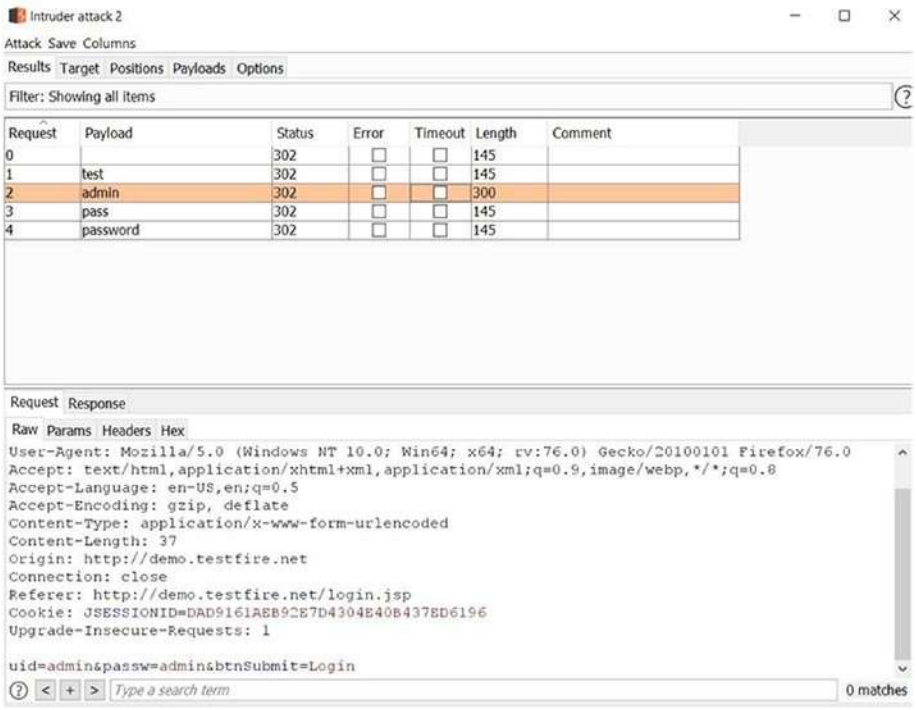


Figure 5-8. Intruder attack results

From Figure 5-8, we can see that Intruder sent five requests each with a different payload. Upon observing and comparing the content length, we can notice that for payload ‘admin’ the response was different. Hence it could be the password for the admin user we are trying to log in. We can then easily verify this by manually logging into the target application.

Options

The last part of Intruder is the ‘Options’ tab. We have already seen that Intruder works as a fuzzing tool or it can perform a brute force attack. This implies the Burp Suite engine would have to send a large number of requests, await the responses, and then process them based on a predefined ruleset. The ‘Request Engine’ option as shown in Figure 5-9 helps configure the number of parallel threads, number of retries on the network failure, and

pause before the retry duration. The values as shown in Figure 5-9 are default and preconfigured. However, depending on specific use cases, these values can be tailored accordingly.

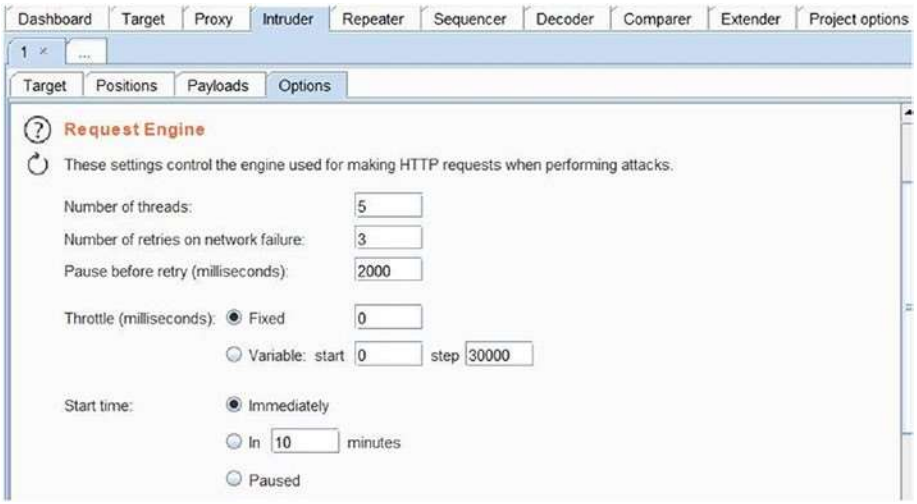


Figure 5-9. Intruder configuration options

Intruder sends a large number of requests to the target along with several permutations and combinations of payloads. The responses can be overwhelming to go through. This is where the ‘Grep Match’ feature comes in handy as shown in Figure 5-10. With this feature we can configure the Intruder engine to flag or highlight interesting responses having keywords like error, exception, illegal, fail, stack, access, directory, etc. If Intruder finds these keywords in any of the responses, they will be explicitly highlighted, making the analysis much easier.

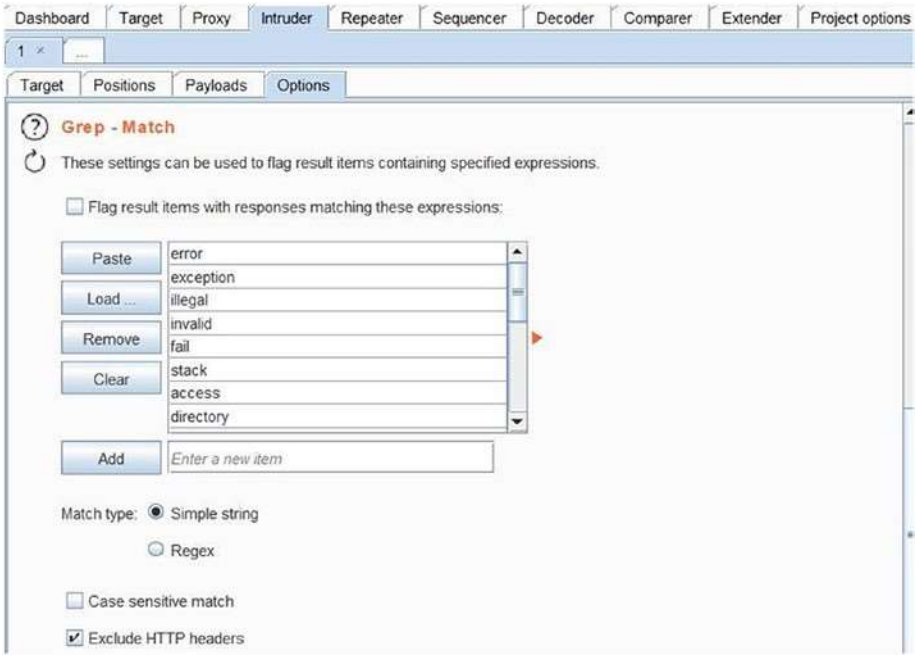


Figure 5-10. Extracting relevant data from Intruder results

Summary

In this chapter we learned about using the Intruder tool to perform fuzzing and brute force attacks. We started off the chapter by learning how to send requests to Intruder, configuring positions, payloads, and finally launching the attack and interpreting the results. We also saw some of the configurable options for Intruder.

In the next chapter we'll see some additional useful tools within the Burp Suite like Repeater, Comparer, Decoder, and Sequencer.

Repeater, Comparer, Decoder, and Sequencer

In the last chapter we learned about how Intruder can be used for fuzzing and performing brute force attacks. In this chapter we will look at some more Burp Suite tools like Repeater, Comparer, Decoder, and Sequencer.

Repeater

Repeater, as the name suggests, is a simple tool within Burp Suite that helps in replaying requests. We have already seen in previous chapters that when we browse an application through Burp Suite, a large number of requests are captured. Not all of the captured requests can be helpful for further testing or analysis. However, there could be a handful of requests with interesting parameters that are worth spending time on for further analysis. Repeater helps precisely in this scenario. If we find a particular request worth investigating further, we can simply send it to Repeater. Once in Repeater, we can play around with the request the way we want; tamper its headers, parameters etc.; and then send the request to the application and see how it responds. Repeater is a very simple yet powerful tool and has an easy interface as shown in Figure 6-1.

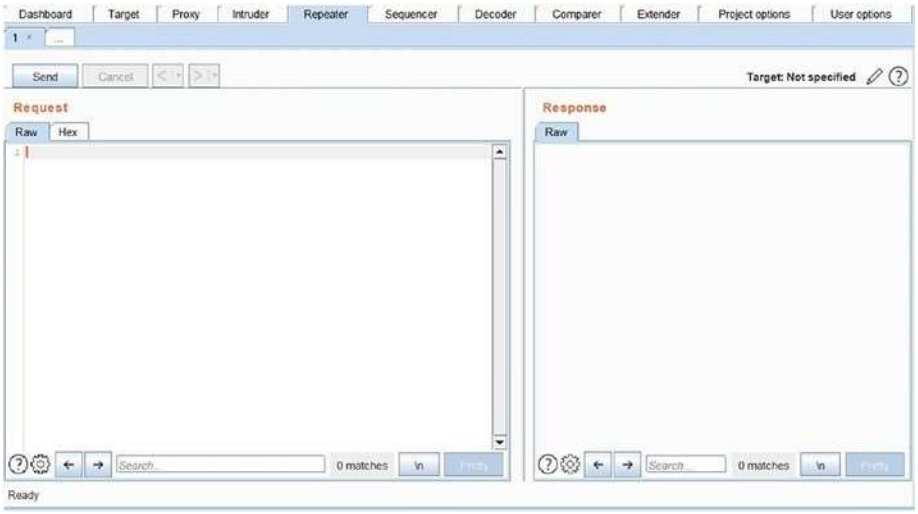


Figure 6-1. The Repeater console

The Repeater tab can be directly accessed in Burp Suite. By default, it opens up empty and we need to feed it with an appropriate HTTP request data. To

specify the target, click on the ‘edit’ icon next to ‘Target’ and a new window will pop up as shown in Figure 6-2, where we can enter Host and Port details we wish to interact with.

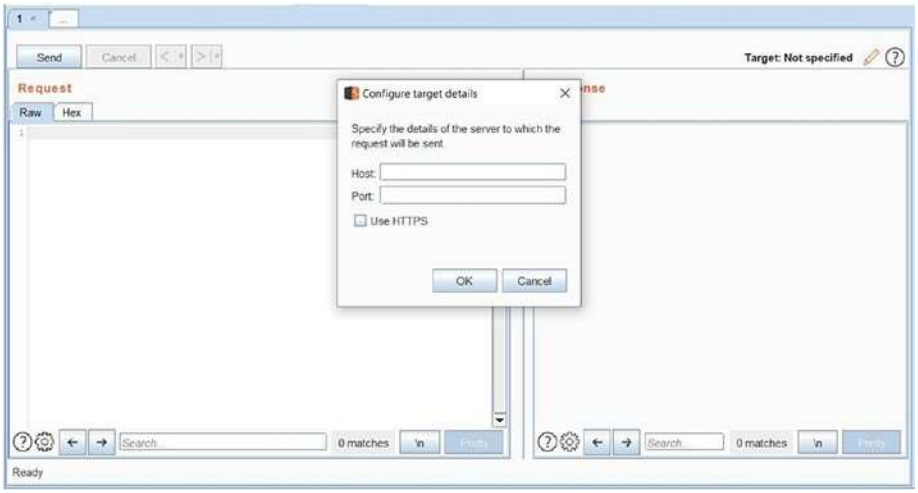


Figure 6-2. Configuring the target details in Repeater

The more convenient way of sending data to Repeater is as shown in Figure 6-3. Simply right-click any of the requests that you wish to send to Repeater for further analysis and click on ‘Send to Repeater’. Another alternative is to select the request to be sent to Repeater and press the hotkey ‘Ctrl + R’.

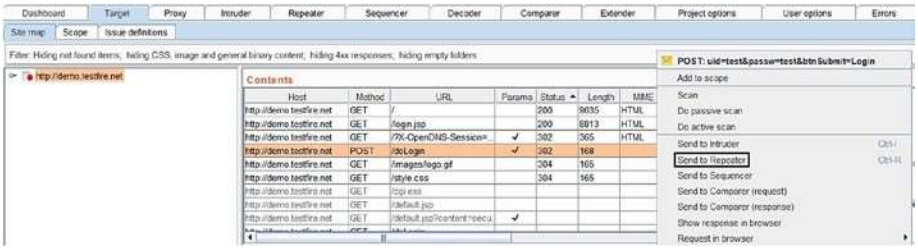


Figure 6-3. Sending HTTP request to Repeater

The request to be investigated is now sent to Repeater as shown in Figure 6-4. The request tab contains the HTTP request we selected and the response tab is blank as we haven’t sent the request yet.

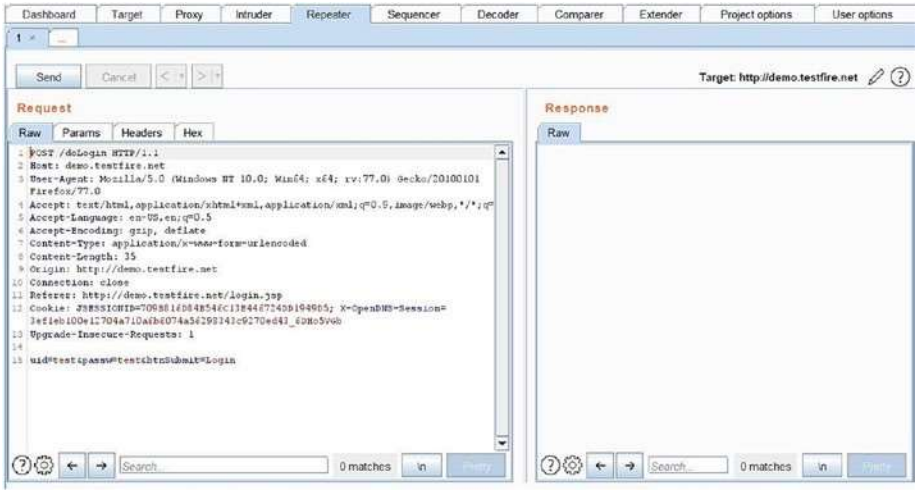


Figure 6-4. Raw HTTP request in Repeater

The Request window has several tabs within as shown in Figure 6-5. One of the tabs is 'Params', which lists all the parameters associated by default with the request we loaded. We can edit the existing parameters, add new parameters, or even delete any of the existing parameters. It is always interesting to see how the application responds to all of these parameter edits.

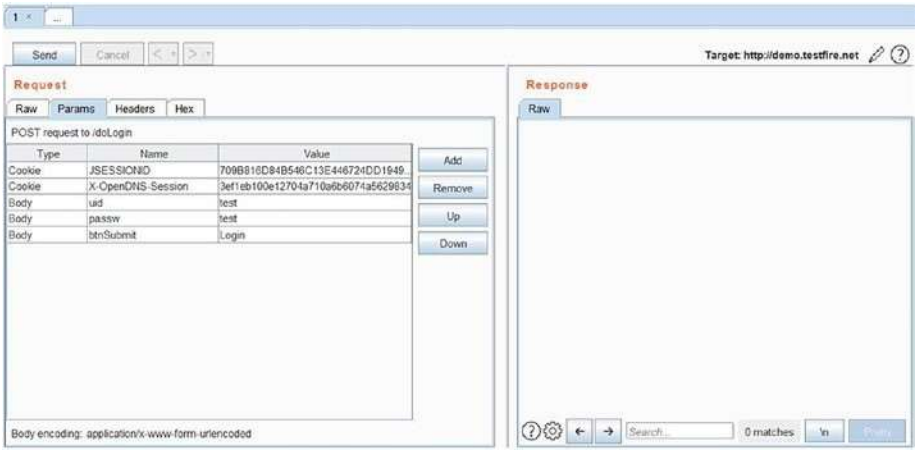


Figure 6-5. Parameter tab in Repeater

The next tab is the 'Headers' tab as shown in Figure 6-6, which lists all the default headers to be sent along with the request. Again, all the header values

are editable: we can edit the existing values, add new header fields, and remove any of the existing header fields as well.

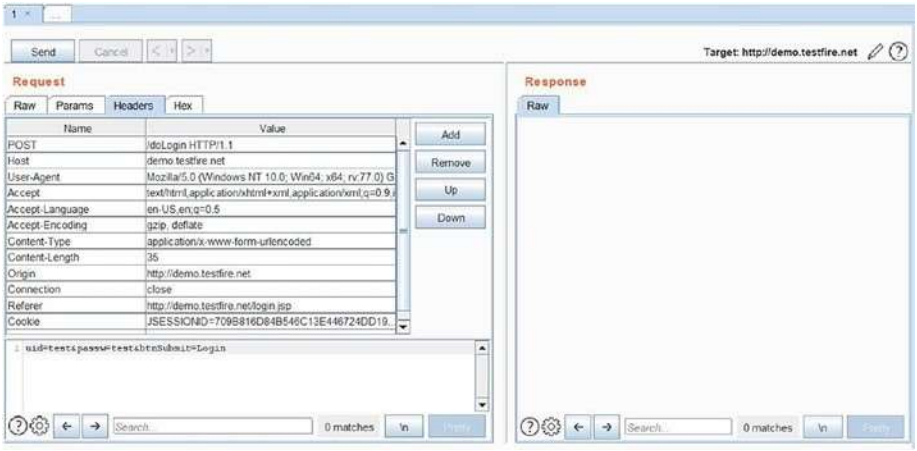


Figure 6-6. Headers tab in Repeater

Once we set the required parameters and header values, we can click on the 'Send' button as shown in Figure 6-7 and we get a response from the application, which is shown in the 'Response' tab.

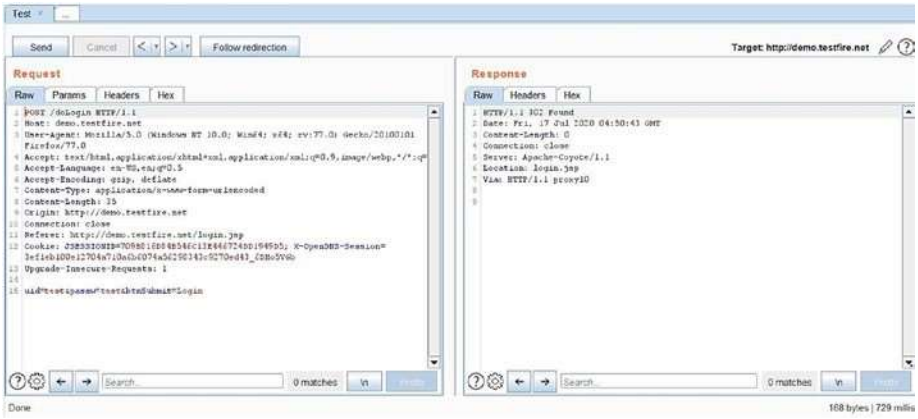


Figure 6-7. Response tab in Repeater

In this case the response received was HTTP status 302, which means the page is meant to redirect. We can click the 'Follow redirection' option and then get the final response as shown in Figure 6-8.

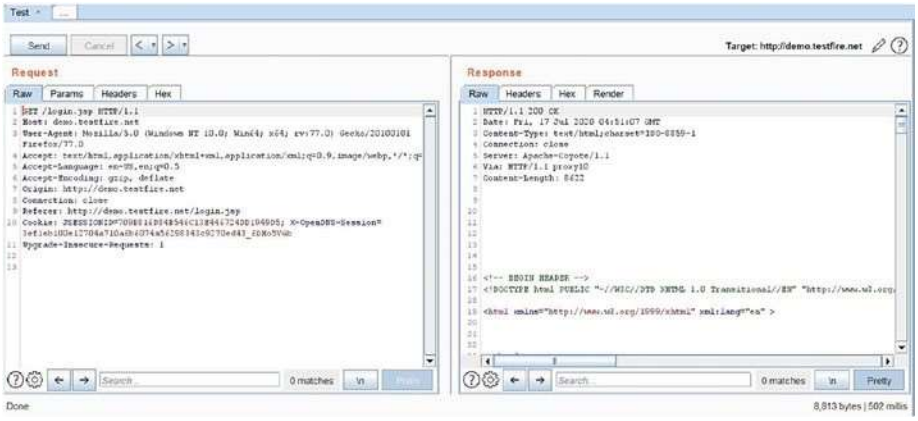


Figure 6-8. Response tab in Repeater

The response shown here is raw text and at times can be difficult to interpret. Hence, we can click the ‘Render’ tab within the ‘Response’ section to visually load the response as if we were seeing it in the browser, as shown in Figure 6-9.

Figure 6-9. Rendering the response in Repeater

Repeater also provides an option to view the response in the real browser. To do this, simply right-click on the response that you wish to see in the browser, and click on ‘Show response in browser’ as shown in Figure 6-10.

Figure 6-10. Viewing the response in the Browser

Burp Suite will pop up another window as shown in Figure 6-11, with a link that needs to be copied into the browser.

Figure 6-11. Show response in the Browser

Once the link generated by Burp Suite is copied in the browser, the response gets rendered accordingly.

Comparer

In the last section we got familiar with the Repeater tool. Once the request is sent to Repeater, there is wide scope to tamper the request parameter or header fields and send it across to the target application. The responses in each case may vary depending on what parameters or header fields that were set in the request. There could be multiple responses, each of them looking quite similar. This is where the Comparer tool comes in handy. Comparer simply compares content head-to-head and highlights differences if there are any. To send a response to Comparer, simply right- click the response and click on ‘Send to Comparer’ as shown in Figure 6- 12.

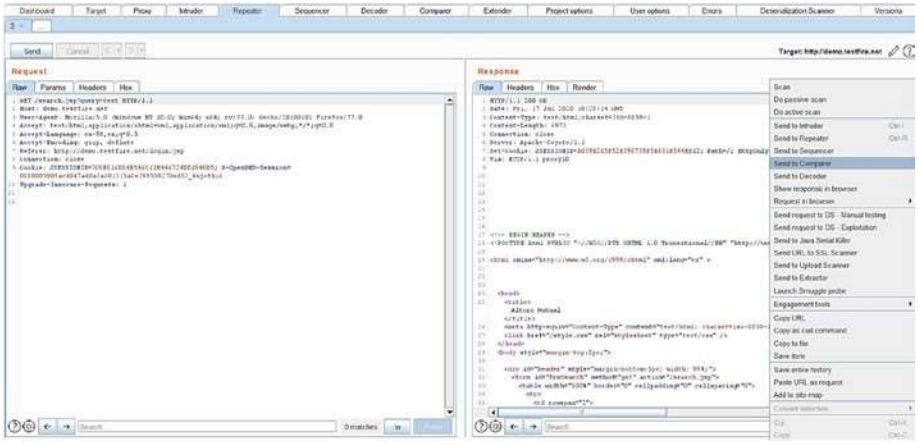


Figure 6-12. Sending Repeater response to the Comparer

Now as the Comparer needs at least two text blocks to compare, we send another response to Comparer as shown in Figure 6-13.

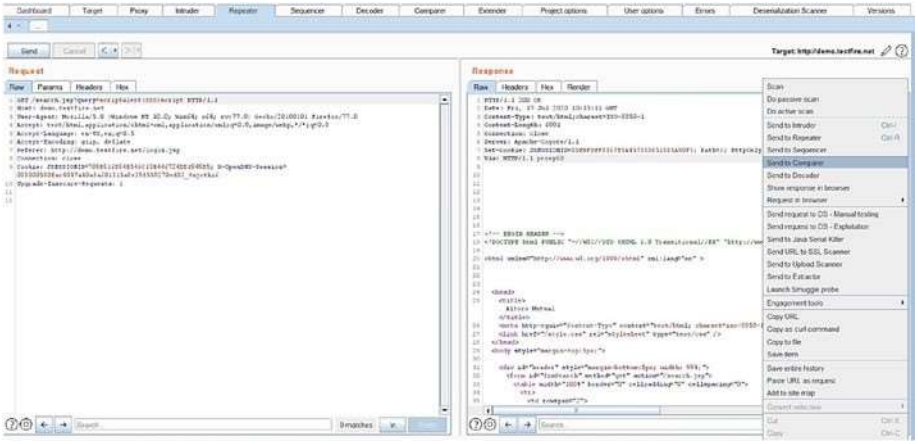


Figure 6-13. Sending Repeater response to the Comparer

We can now navigate to the Comparer tab as shown in Figure 6-14 and see both responses that we sent from Repeater earlier.

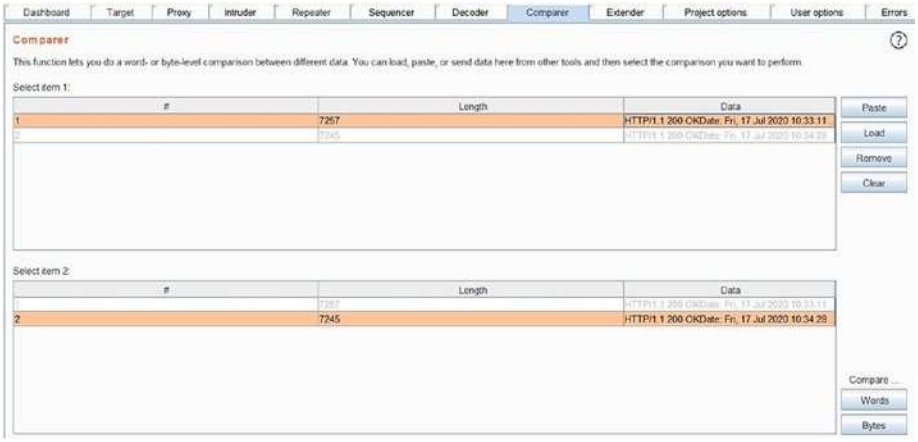


Figure 6-14. The Comparer console

Now since we need to find differences in words from both of the responses, we click on the 'Words' button and get a new window opened as shown in Figure 6-15.

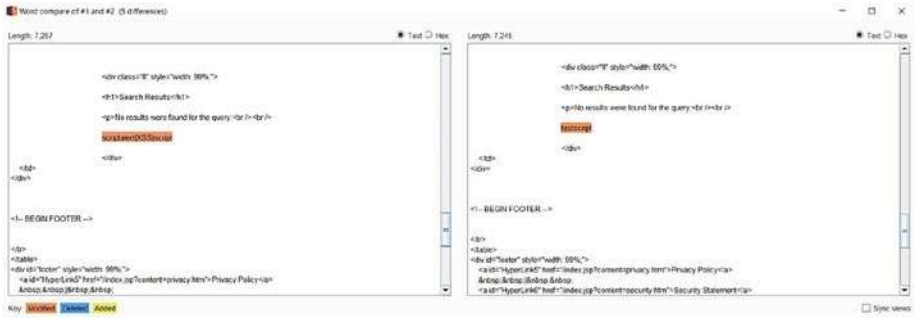


Figure 6-15. Comparing the responses in the Comparer

Comparer will now highlight the text changes in two of the parallel windows.

Decoder

Web applications commonly use various encoding schemes like Ascii, HTML, Base 64, etc. From a security testing perspective, it's very common to encounter such encoded strings during testing. Burp Suite Decoder is a simple utility that can encode or decode text in a format of a URL, HTML, Base 64, ASCII hHx, Hex, Octal, Binary, and Gzip. Simply navigate to the Decoder tab as shown in Figure 6-16 and enter the text that needs to be decoded.

Figure 6-16. Decoding Base 64 data

In this case we entered a Base 64 encoded value and then clicked on ‘Decode as’ Base 64 to get the decoded output in the next window as admin:admin.

We can also use this tool to encode any plain text as shown in Figure 6- 17.

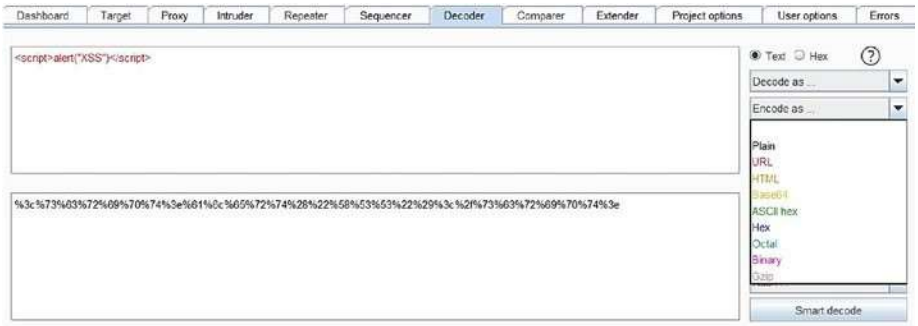


Figure 6-17. Encoding data using the Burp Suite Decoder

We simply entered the plain text <script>alert(“XSS”)</script> and then clicked on ‘Encode as’ the URL to get the encoded output in the next window.

Sequencer

Web applications depend a lot on tokens, session IDs, or other such unique and random identifiers. From a security perspective, it is important to test the randomness or uniqueness of these tokens and identifiers. If the tokens aren’t strong and random enough, then attackers can easily brute force them and get unauthorized access.

Burp Suite Sequencer is a tool that helps us test the strength of application tokens. We can send any request to Sequencer just by right- clicking the request and clicking on 'Send to Sequencer' as shown in Figure 6-18.

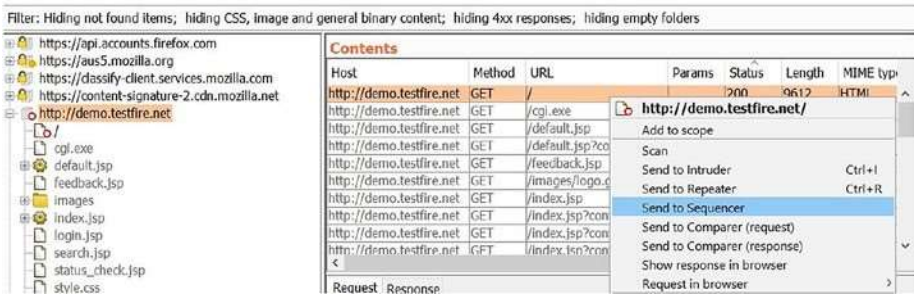


Figure 6-18. Sending HTTP request to Sequencer

We can now navigate to the Sequencer tab as shown in Figure 6-19.

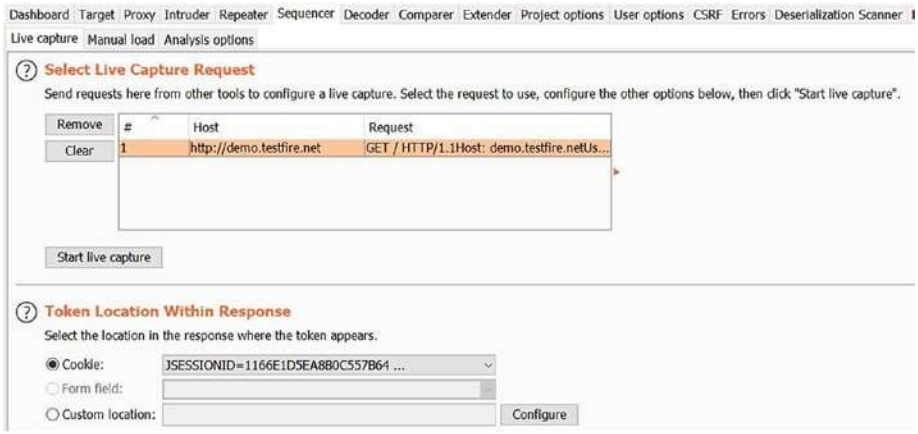


Figure 6-19. The Burp Suite Sequencer

We can clearly see the Sequencer has automatically parsed the request and selected the JSESSIONID token present in the cookie. This token would now be analyzed for its strength and randomness. To start the test, simply click on 'Start live capture' and a new window will open up as shown in Figure 6-20.

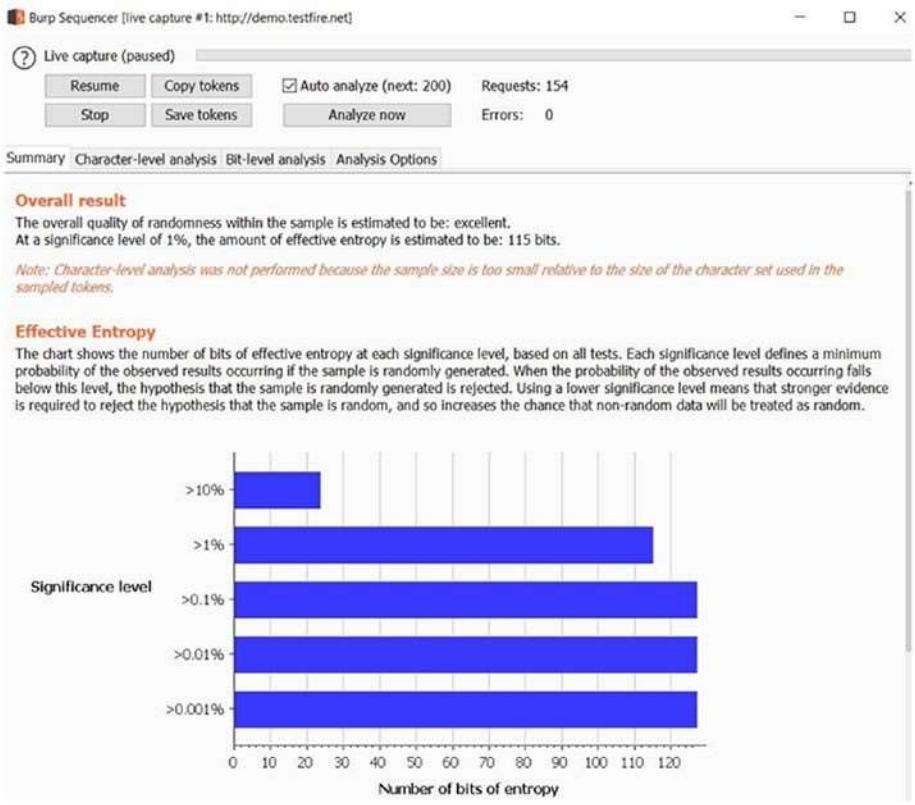


Figure 6-20. Session ID analysis using Sequencer

The live capture starts and we can pause or resume it any time we wish. However, it's important to note that to effectively analyze the token strength, a sample size of at least 100 should be considered. Once the capture is complete, Sequencer shows us the result, which was found to be "excellent" in this case. So, the token JSESSIONID is strong, unique, random, and hence safe to use.

Sequencer also allows us to load a sample of tokens manually and then analyze them. To do this, simply navigate to the 'Manual load' tab in Sequencer as shown in Figure 6-21.

Repeater, Comparer, Decoder, and Sequencer

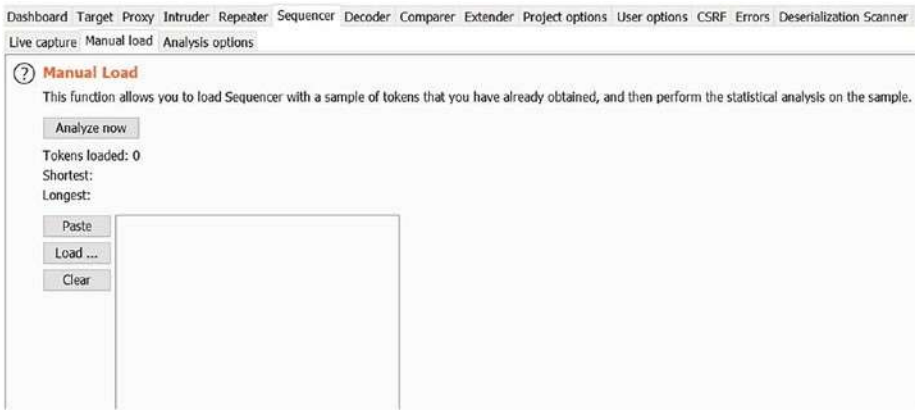


Figure 6-21. Manually loading tokens for analysis

We can now copy and paste a sample of tokens and then analyze them accordingly.

Summary

In this chapter we saw how to tamper and replay requests using Repeater, then perform a head-to-head comparison of responses using Comparer. We then learned about the Decoder tool, which helps encode and decode text in various formats. Lastly, we got familiar with the Sequencer tool, which can be used to assess the effectiveness of tokens.

In the next chapter, we'll learn about some additional useful tools within the Burp Suite like Infiltrator, Collaborator, Clickbandit, and CSRF PoC Generator.

Repeater, Comparer, Decoder, and Sequencer

Infiltrator, Collaborator, Clickbandit, and CSRF PoC Generator

In the last chapter we looked at some Burp Suite tools like Repeater, Sequencer, Decoder, and Comparer. In this chapter we will continue to explore more useful tools like Infiltrator, Collaborator, Clickbandit, and CSRF PoC (proof-of-concept) generator.

Infiltrator

Burp Suite Infiltrator is a tool that instruments the target web application so that the vulnerability detection by the Burp Suite scanner becomes more efficient and accurate. Infiltrator makes irreversible changes in the code and essentially hooks into the target application. This way, it helps the Burp Suite scanner get more visibility into the application code and potentially detect unsafe calls and functions.

As the Infiltrator makes irreversible changes to the target application code, it is advisable to run it only for a test instance and not on a production instance. Currently, the Infiltrator is supported if the target application is using any of the following technologies:

Java

Groovy

Scala

Other JVM language (JRE versions 1.4 - 1.8)

C#

Visual Basic

Other .Net language (.Net versions greater than 2.0)

To get started with the Infiltrator, click on the Burp menu and then select 'Burp Infiltrator' as shown in Figure 7-1.

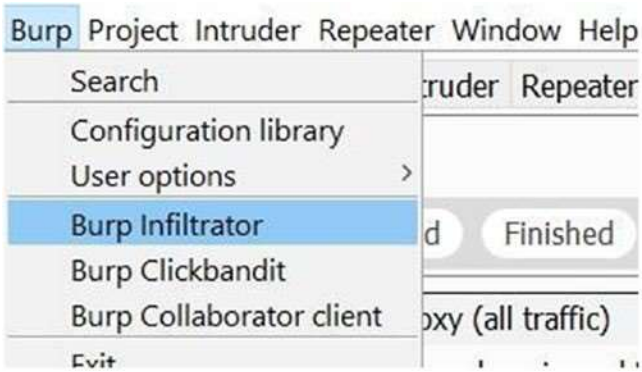


Figure 7-1. Navigating to the Burp Suite Infiltrator

A new window will pop up as shown in Figure 7-2. This wizard will help us generate the Infiltrator agent.

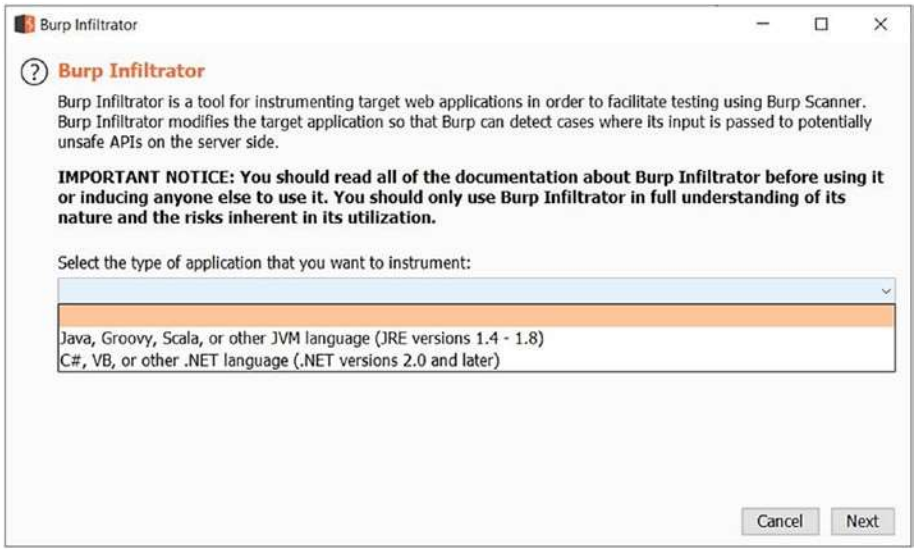


Figure 7-2. Generating the Infiltrator agent

We need to select the technology our application is using like Java or .NET and click on Next. Then the wizard will ask the location where we wish to save the Infiltrator agent, as shown in Figure 7-3.

Figure 7-3. Generating the Infiltrator agent

Next, the wizard will simply generate the Infiltrator agent and save it to the location we selected earlier, as shown in Figure 7-4.

Figure 7-4. Generating the Infiltrator agent

The important thing to note here is the Infiltrator agent should be in the same directory where the target application is located as shown in Figure 7-5.

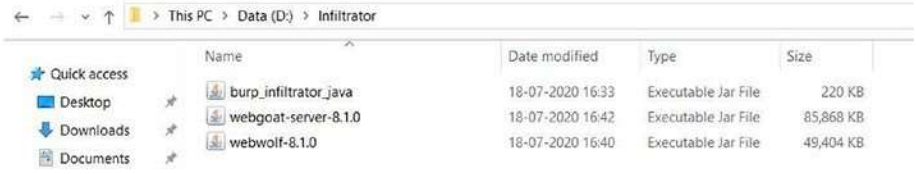


Figure 7-5. Newly generated Infiltrator agent

Now that both the Infiltrator agent and the target application are in the same directory, we can open a command prompt and type command ‘java -jar burp_ infiltrator.jar’ as shown in Figure 7-6.

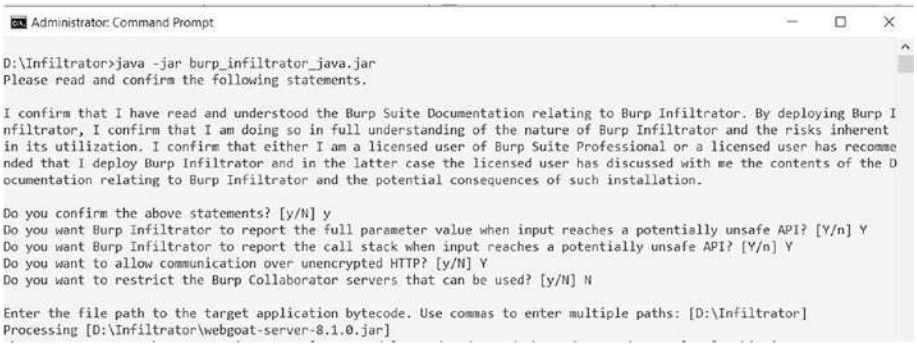


Figure 7-6. Executing the Infiltrator agent

The Infiltrator will now run and modify the Java applications in the directory. This is a one-time procedure, and the application needs to restart once the patched code is available. Burp Infiltrator also makes use of Collaborator, which we will be seeing in the next section.

Collaborator

Collaborator is a tool provided by Burp Suite that helps in attacks like

Server-Side Request Forgery (SSRF) or any of the out-of-band attacks. The Burp Suite Collaborator service helps by generating random payloads in the form of hostnames. These payloads can then be used as part of requests in various attack scenarios. If the attack is successful, then an interaction occurs between the target application server and the Burp Collaborator server. Then using the Burp Collaborator client, we can poll and check if any such interactions have happened.

To get started with the Burp Collaborator, simply click on the Burp menu and click “Burp Collaborator client’. A new window will pop up as shown in Figure 7-7.

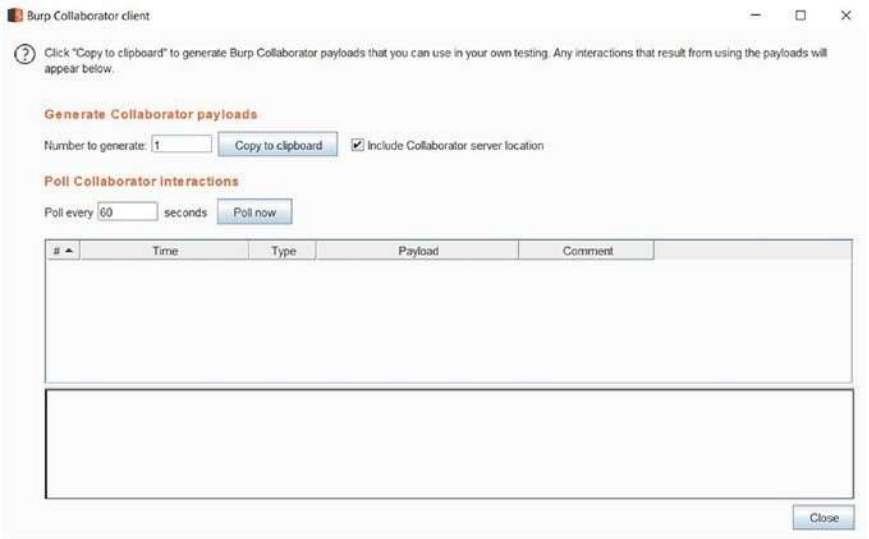


Figure 7-7. The Burp Suite Collaborator client

Now click on the ‘Copy to clipboard’ button and paste its value in the notepad as shown in Figure 7-8.

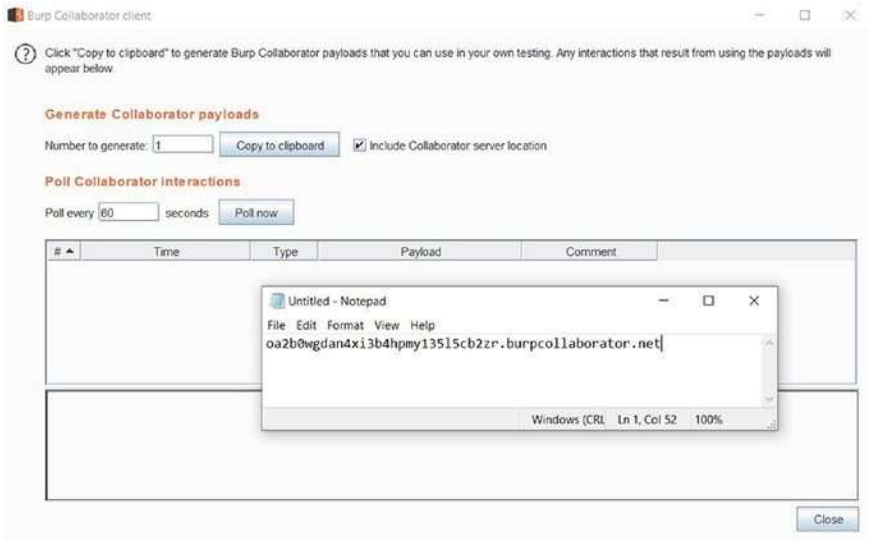


Figure 7-8. Configuring the Collaborator client

The random value generated by the Burp Collaborator can now be used in payloads in requests sent as part of an attack. The Burp Collaborator client automatically polls the Collaborator server after every 60 seconds to check if there has been any interaction. This duration can be customized or you can simply click on the ‘Poll now’ button to manually check for Collaborator interactions.

Clickbandit

Clickjacking is one of the very common attacks on web applications. Using clickjacking, the attacker tries to trick the user into clicking something different than what the user sees visually. If successful, the attacker can get access to confidential information. Clickjacking is also known as a UI redressal attack, as the attacker tries the deceptive technique of creating a fake UI and then tricks the victim into executing malicious actions or events.

Burp Suite offers a utility called ‘Clickbandit’ that significantly simplifies the process of generating Proof-of-Concept for an application that is vulnerable to Clickjacking.

To get started with the Clickbandit tool, simply go to the Burp menu and click on ‘Burp Clickbandit.’ A new window will pop up as shown in Figure 7-9.

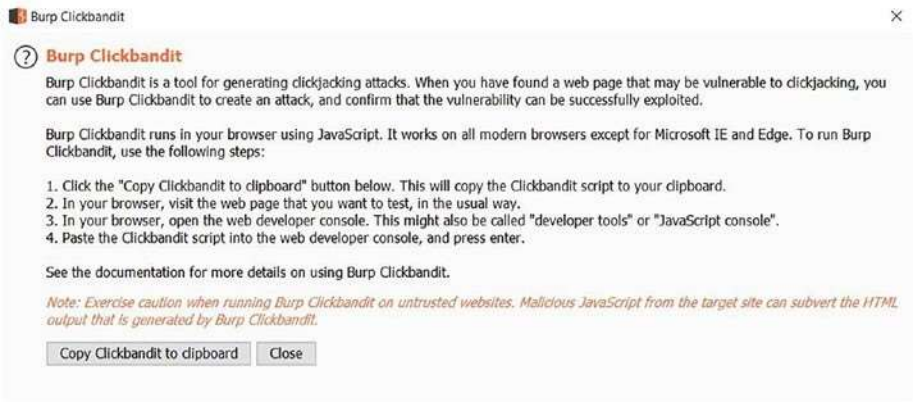


Figure 7-9. The Burp Suite Clickbandit tool

This window has steps listed that we need to follow in order to generate the Clickjacking Proof-of-Concept. The first step is to click on the ‘Copy Clickbandit to clipboard’ button. The next step is to open the browser and press function key F12 to go into the browser console as shown in Figure 7-10.

Figure 7-10. Target for Clickbandit

To proceed further, we need to paste the Clickbandit code into this browser console, which we copied earlier as shown in Figure 7-11.

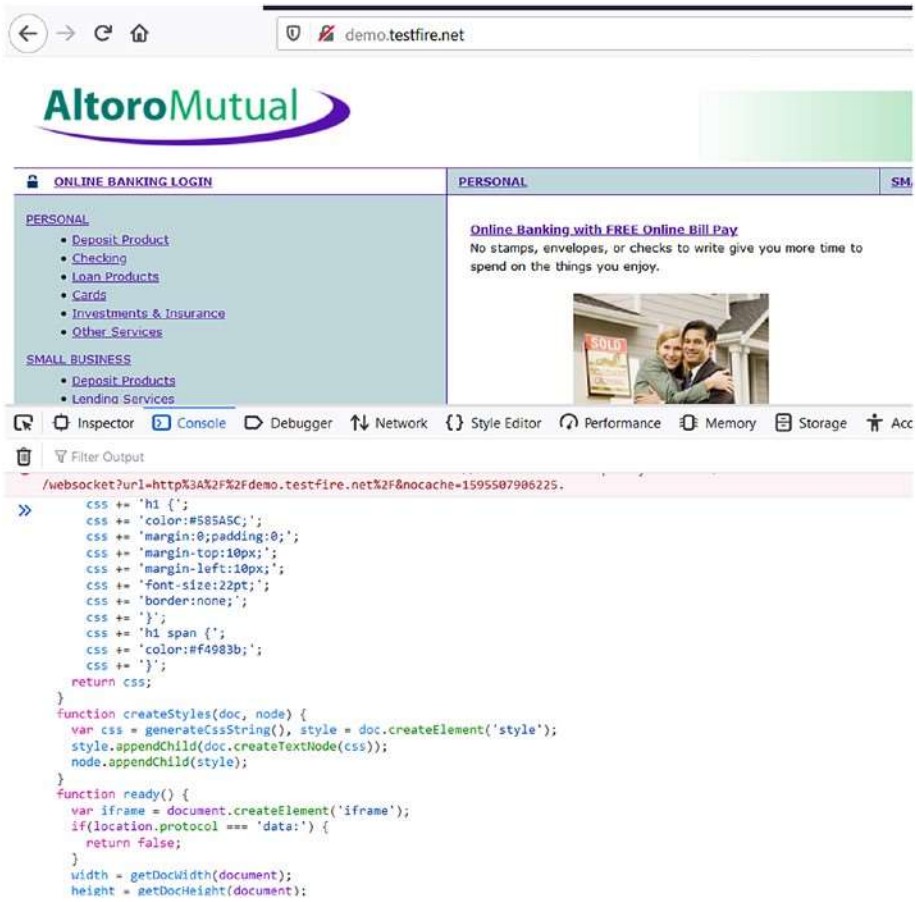


Figure 7-11. Copying the Clickbandit code in browser console

Once the code is copied into the browser console, simply press Enter and you'll notice the Burp Clickbandit UI appears on top of the page as shown in Figure 7-12.

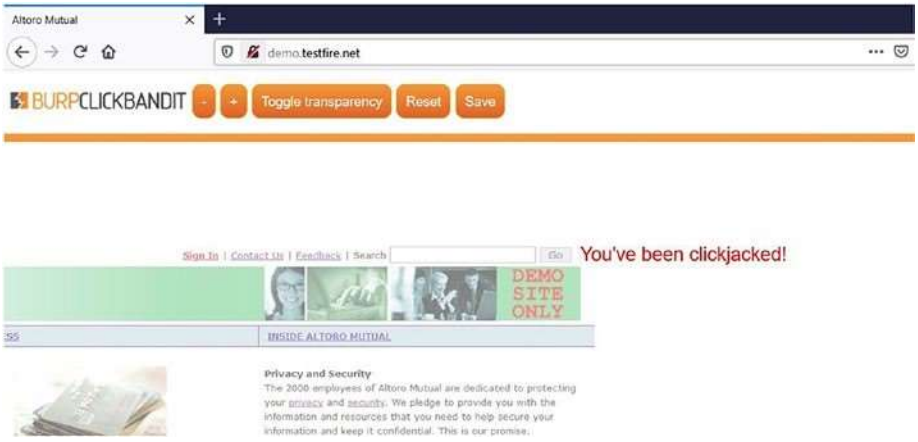


Figure 7-12. The Clickbandit UI

Now we need to perform and record the actions that we wish to include as part of the Clickjacking attack. Once all the required actions are done, click on the save button and you will be able to save a file named ‘clickjacked.html’ as shown in Figure 7-13.

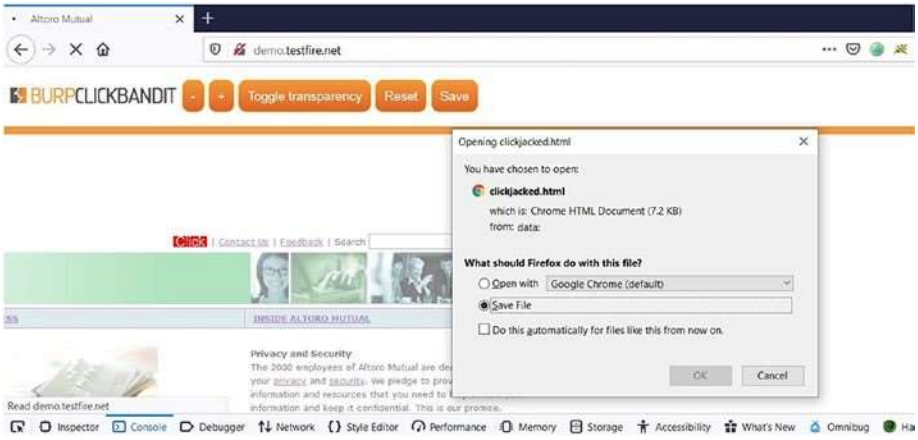


Figure 7-13. Saving the Clickbandit code

You can now open the file ‘clickjacked.html’ separately in the browser as shown in Figure 7-14.

Figure 7-14. Executing the Clickbandit code

You'll notice that the actions you captured earlier are now being replayed, and if you click, then you get a message 'You've been clickjacked!' as shown in Figure 7-15.

Figure 7-15. Executing the Clickbandit code

CSRF

Cross-Site Request Forgery, commonly known as CSRF, is another type of attack on web applications that exploits session management flaws to trick

the victim into performing unwanted actions. Burp Suite has a utility that makes it very easy to generate Proof-of-Concept for CSRF vulnerability.

We first need to identify and confirm the request for which we wish to generate the CSRF Proof-of-Concept code. Once the request is finalized, simply right-click the request, go to 'Engagement tools', and click on 'Generate CSRF PoC' as shown in Figure 7-16.

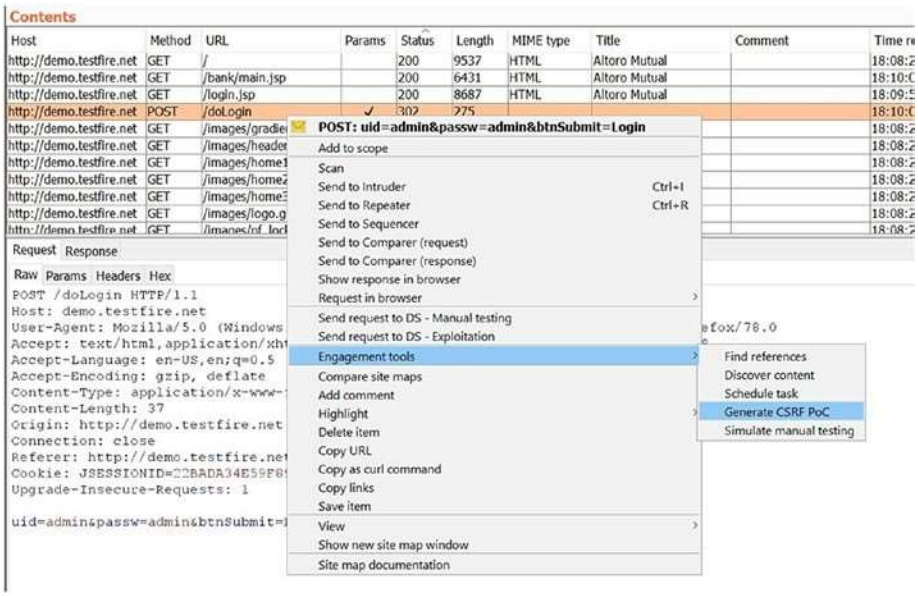


Figure 7-16. Sending request to CSRF PoC generator

Now, a new window will pop up as shown in Figure 7-17, which has the POST request along with the CSRF code.

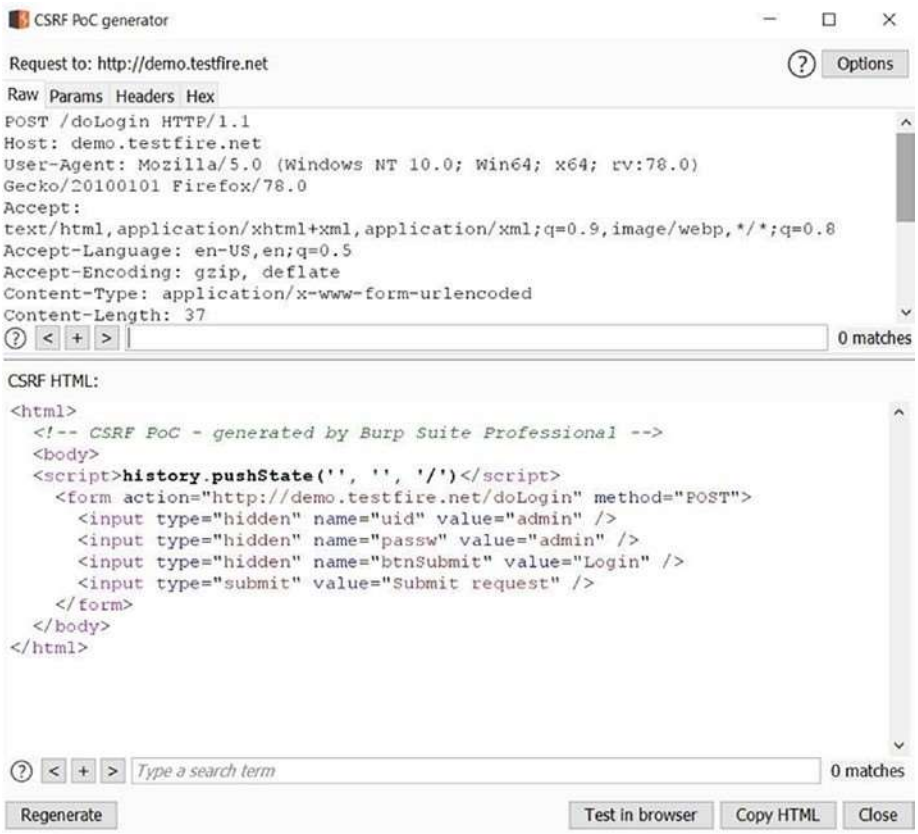


Figure 7-17. CSRF PoC generator

It is now easy to modify the CSRF code as required and then we can either directly test it in the browser or generate a separate HTML file. To test the CSRF code in the browser, click on the 'Test in browser' button, and a new window will pop up as shown in Figure 7-18.

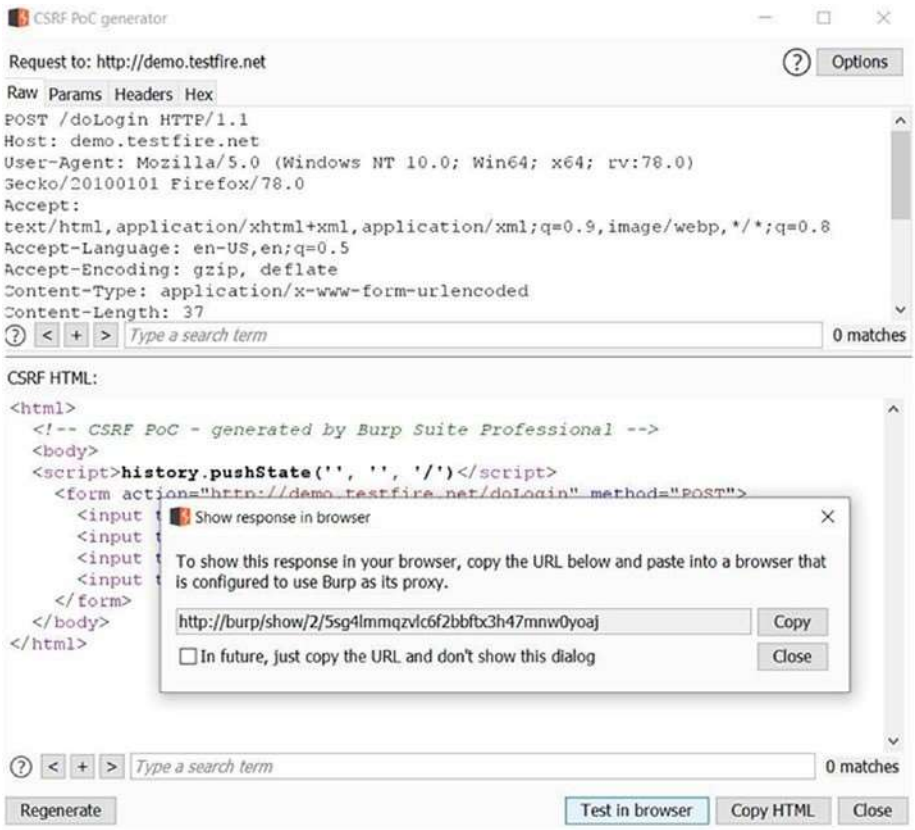


Figure 7-18. CSRF PoC generator

Now click on the 'Copy' button, open the browser, and paste into the address bar as shown in Figure 7-19.

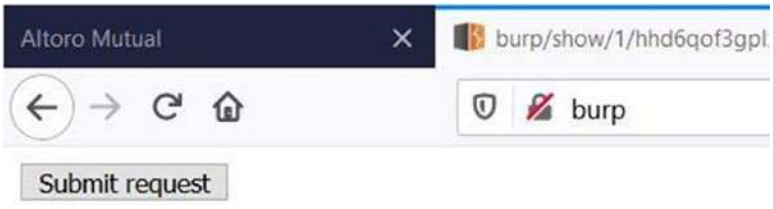


Figure 7-19. Verifying the CSRF PoC in browser

Now click on the button 'Submit request', and the CSRF code will get executed as shown in Figure 7-20.

Figure 7-20. Verifying the CSRF PoC in browser

Summary

In this chapter we learned about using Intruder for instrumenting applications and increasing detection capabilities of the Burp Scanner. Then we saw the Burp Collaborator, which can be effectively used in out- of- band attacks like SSRF. We then looked at the Clickbandit tool that helps generate proof-of-concept code for applications vulnerable to clickjacking; and lastly, we glanced through the CSRF PoC generator, which helps us quickly generate and test proof-of-concept code for Cross-Site Request Forgery attacks.

In the next chapter, we'll see the automated scanning and reporting capabilities of the Burp Suite.

Infiltrator, Collaborator, Clickbandit, and Csrf poc Generator

Scanner and Reporting

In the last chapter, we learned about various tools like Infiltrator, Collaborator, Clickbandit, and CSRF PoC generator. In this chapter, we'll explore the features and capabilities of the Burp Suite scanner for automated vulnerability detection.

Scan Types

So far throughout the book, we have seen several capabilities of Burp Suite that are useful for manual testing. However, Burp Suite also provides a web application vulnerability scanner that automates the process of finding vulnerabilities. This is indeed a very feature-rich scanner and is capable of detecting potential web vulnerabilities.

The Burp Suite offers two types of scans: Passive Scan and Active Scan. The passive scan runs in the background, by default, while we browse an application through Burp Suite. A passive scan simply monitors the traffic and tries to list vulnerabilities like missing security flags in cookies, missing security headers, traffic being sent over unencrypted communication channels, etc. Thus, the passive scanner doesn't attempt to inject any payloads into any of the insertion points, but rather just highlight vulnerabilities that can be found only by passively monitoring ongoing requests and responses. The active scan goes a step further tries to insert payloads into insertion points and check if parameters are vulnerable. Active scanning is a more intense technique; however, it does a better job in finding vulnerabilities that a passive scanner may never find. We'll now look further into the details of performing an active scan using Burp Suite.

Crawl and Audit

Active scanning is usually a two-step process. The first step involves crawling or spidering the application and the second step involves attacking the parameters with payloads. The Burp Suite scanner offers two options: either crawl and audit or just crawl. To start a new audit, go to the Dashboard tab and click on 'New scan' as shown in Figure 8-1.

Figure 8-1. New scan task

A new window will pop up as shown in Figure 8-2. We select the option ‘Crawl and audit’. Next, we need to specify the target URL that we wish to scan. In this case, we enter the target URL as ‘demo.testfire.net’.

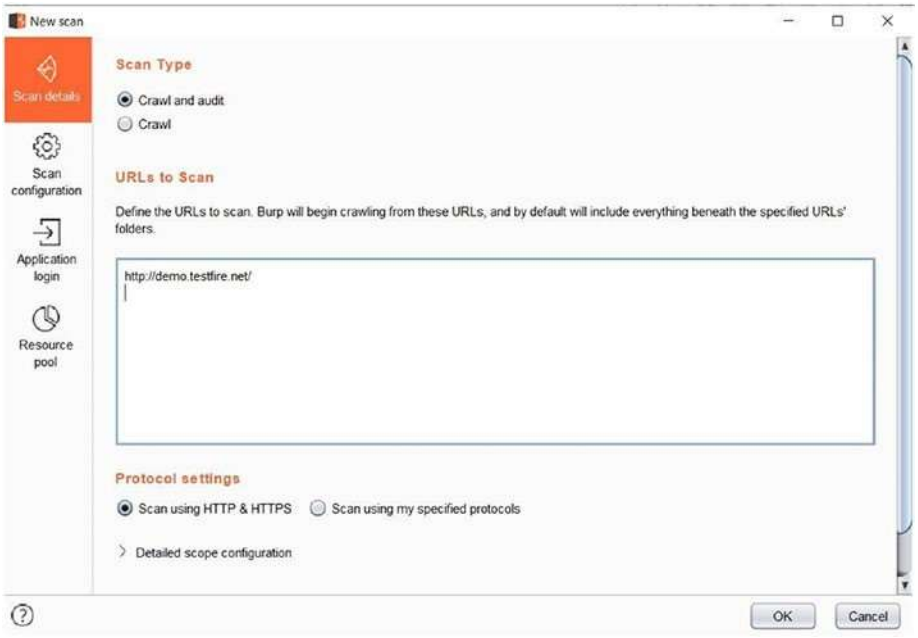


Figure 8-2. New scan configuration

Next is the scoping section as shown in Figure 8-3. The URL's that we wish to be part of the audit need to be specified under the 'Included URL prefixes' tab, and if there are any particular URLs that we don't want to be included in the audit, then those need to be explicitly added under 'Excluded URL prefixes.'

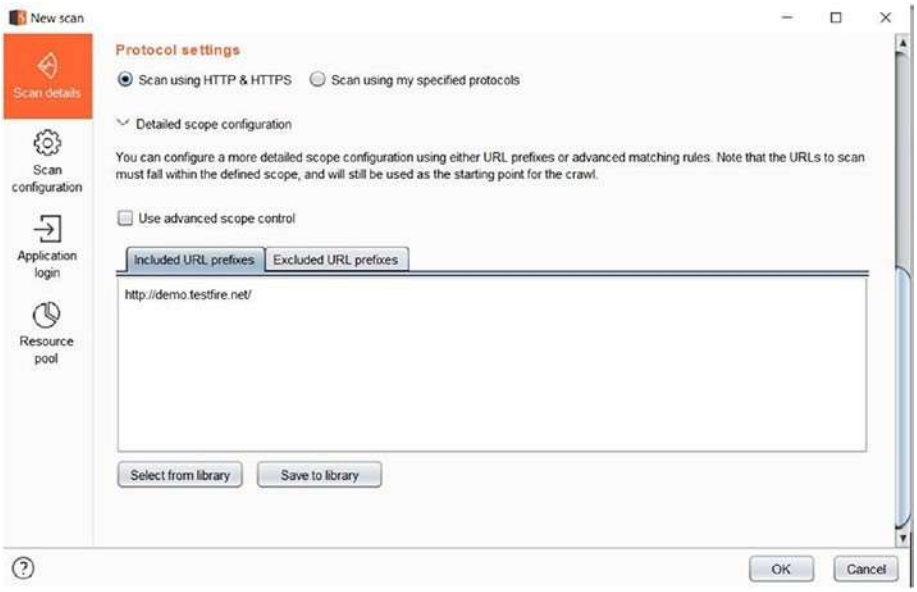


Figure 8-3. New scan configuration

Now that we have configured the target URL, we just need to click on 'OK' and the crawl and audit activity starts as shown in Figure 8-4. However, it's important to note that this activity will start with a default scan configuration.

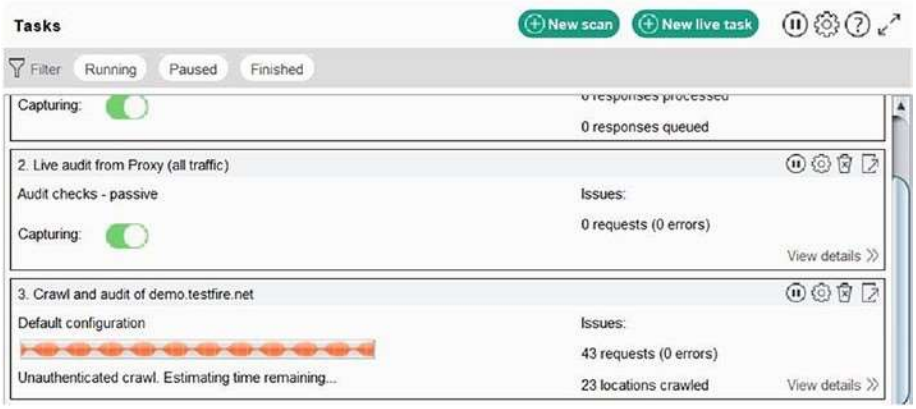


Figure 8-4. Scan tasks in progress

In the next section, we'll be looking at customizing the scan configuration.

Scan Configuration

In the previous section, we configured and initiated a crawl and audit task on a target URL but with default configuration settings. In this section, we'll take a look at how the scan configuration can be tailored to suit our needs. To customize the scan configuration, click on the 'Scan configuration' tab as shown in Figure 8-5.

Figure 8-5. Scan configuration

The scan configuration allows us to customize crawl settings as well as the audit settings. We'll first go through the crawl configuration settings. Click on the 'New' button and select 'Crawl'. A new window will pop up as shown in Figure 8-6.

Figure 8-6. Crawl optimization settings

The crawl optimization configuration allows us to set the maximum link depth up to which we wish to crawl along with the crawl strategy, which is set to normal by default. We can change the crawling strategy to fast by selecting it through the drop-down menu, depending on the particular scan scenario.

Next, we can configure the crawl time limits as shown in Figure 8-7. If the target application is large and complex, then it might take a lot of time for crawling. We can set a limit to this by defining the maximum time that we wish to spend on crawling the application. We can also limit the crawl by the number of locations discovered or the maximum number of requests made during the crawl function.

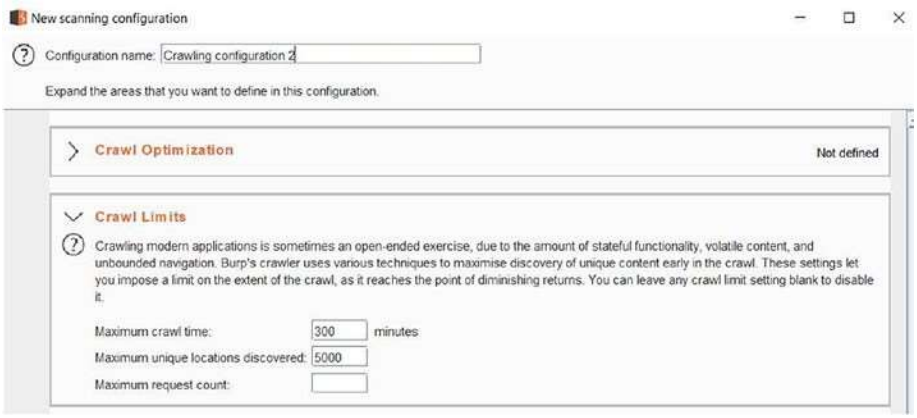


Figure 8-7. Crawl limit configuration

The next configuration setting is related to the login functions as shown in Figure 8-8. It might happen that the target application has a login function. In such a case, Burp Suite will even try to register a new test user.

Figure 8-8. Login function configuration

The next configuration setting is related to handling application errors during a crawl function as shown in Figure 8-9. There could be multiple reasons behind application errors, like an authentication failure, network problems, etc. This configuration setting tells the Burp Suite scanner to pause the crawl and audit function if there are a certain number of consecutive application errors.

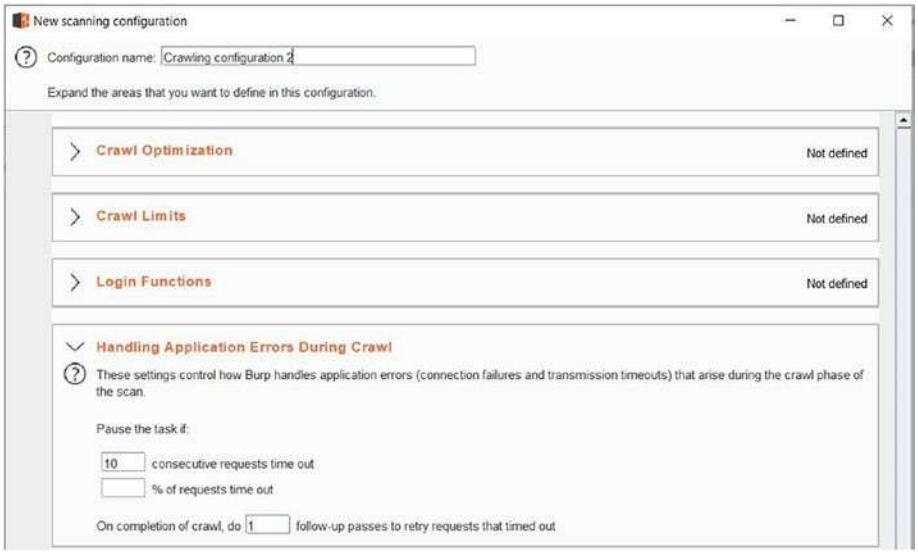


Figure 8-9. Configuring application errors during Crawl

The next set of configuration settings are miscellaneous as shown in Figure 8-10. This includes settings on whether we want the Burp Suite scanner to automatically submit forms or if we wish to customize the user- agent if we want to fetch robots.txt and the sitemap, etc.

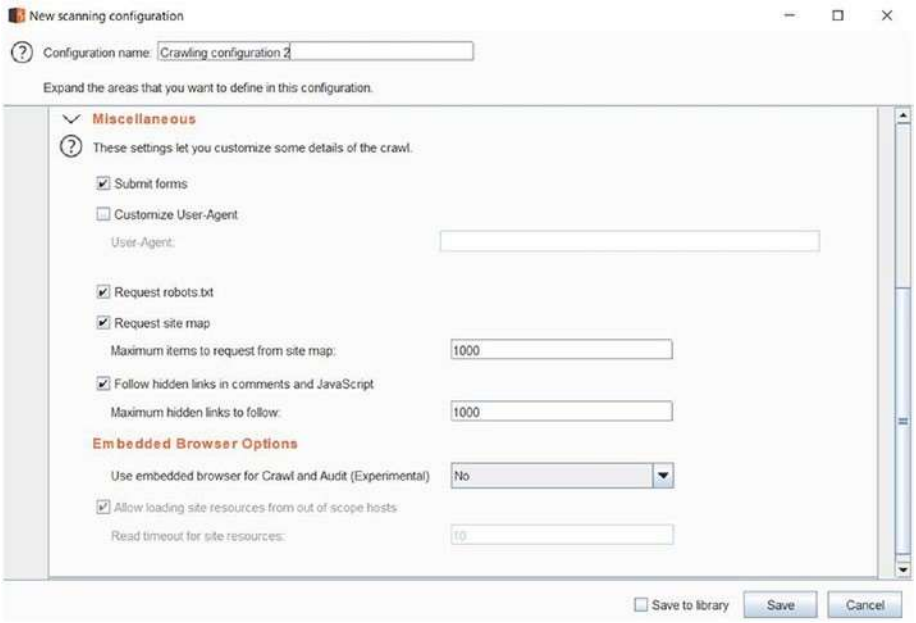


Figure 8-10. Miscellaneous crawl configuration

The next set of configuration settings are related to the audit function. To start with, the first audit configuration setting is 'Audit Optimization' as shown in Figure 8-11. This setting allows us to configure audit speed and accuracy. The audit speed can be set to either fast, normal, or thorough, while the audit accuracy can be set to normal or to minimize false positives.

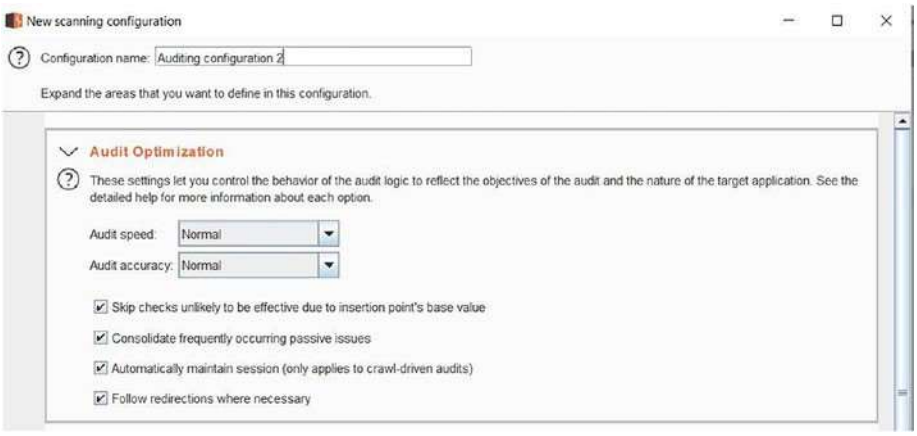


Figure 8-11. Audit optimization configuration

The next audit configuration setting is related to the type of issues reported as shown in Figure 8-12. The Burp Suite scanner detects a variety of issues. However, during a particular test scenario, it might so happen that only a particular type of issue needs to be tested. In such a case, it won't be worth spending time on testing all other types of issues. Hence this configuration setting allows us to customize the type of issues that we want to be tested during the scan.

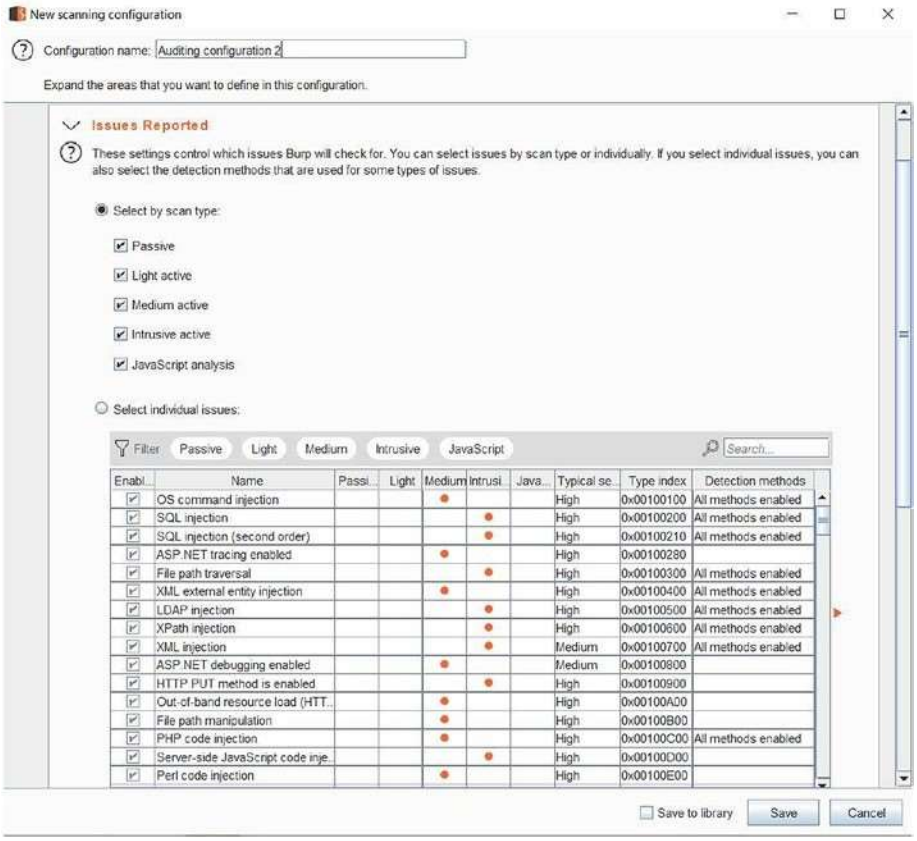


Figure 8-12. Type of issues to be detected during an audit

The next audit configuration setting is related to handling application errors during the audit function, as shown in Figure 8-13. We have already seen a

similar configuration setting for the crawl function. This setting helps configure the number of failures after which the audit task would be paused.

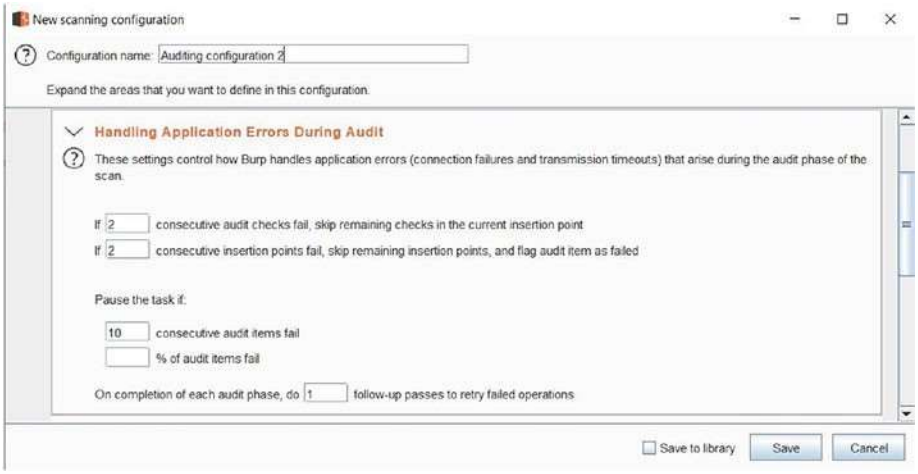


Figure 8-13. Configuring application errors during Audit

The next audit configuration setting is related to the type of insertion points that we want the Burp Suite scanner to attack during the audit function as shown in Figure 8-14. Selecting all the types of insertion points will increase the possibility of finding more vulnerabilities, but at the same time it will also take longer to finish the audit.

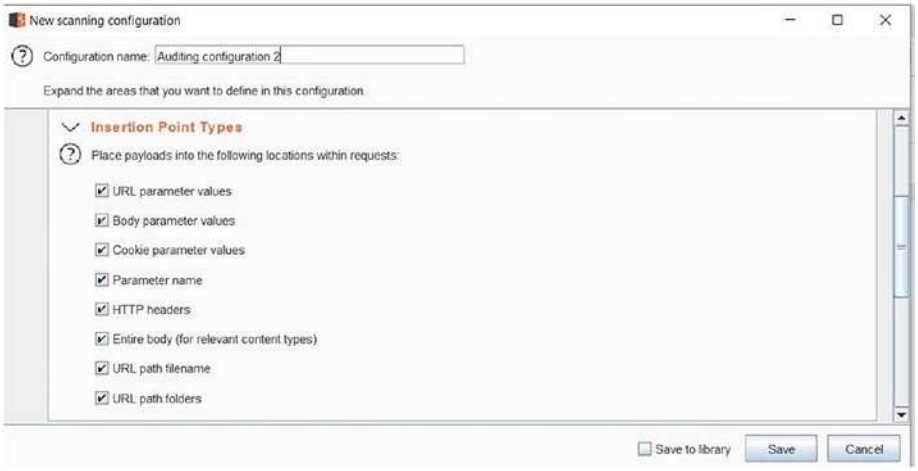


Figure 8-14. Configuring insertion point types

All the crawl and audit scan configuration settings we saw so far are set to optimal values by default. We can quickly trigger a new crawl and audit task using the default scan configuration. However, depending on particular scan scenarios, it might be required for you to customize the scan configuration settings.

Application Login

The next important scan configuration setting is configuring the 'Application login' as shown in Figure 8-15. While scanning the target application, we may come across certain pages that do not require authentication, while there could be a few pages that can be accessed only after authentication. If we want the Burp Suite scanner to audit the pages behind authentication as well, then we need to provide credentials.

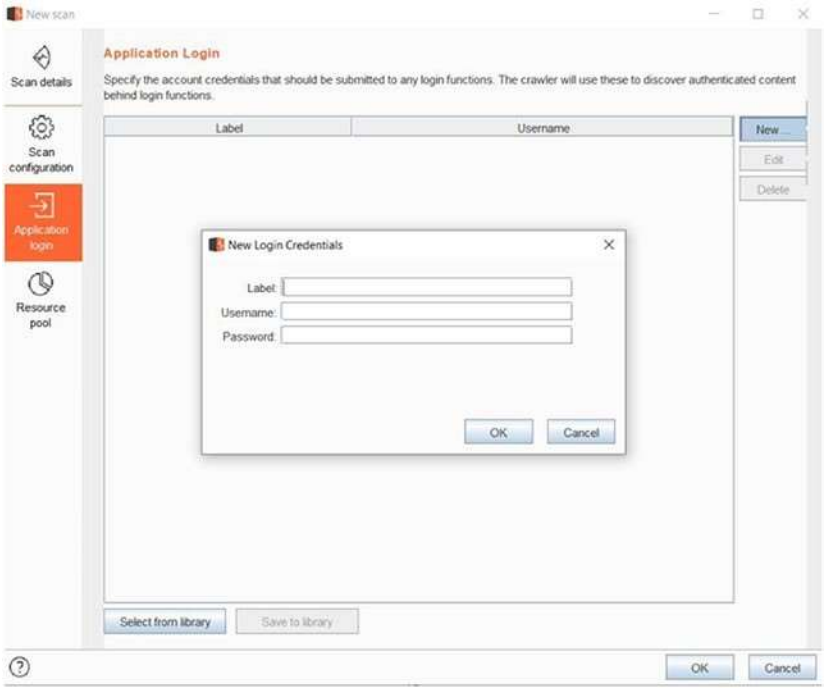


Figure 8-15. Configuring application login

Credentials can be added by simply clicking on the 'New' button and providing the required username and password.

Resource Pools

The last scan configuration option is “Resource Pool” as shown in Figure 8-16. The resource pool helps define the system resources that will be used across multiple tasks. By default, the resource pool is created, which allows for a maximum of 10 concurrent requests. We can leave this to default unless we want to do multitasking within the same Burp Suite project.

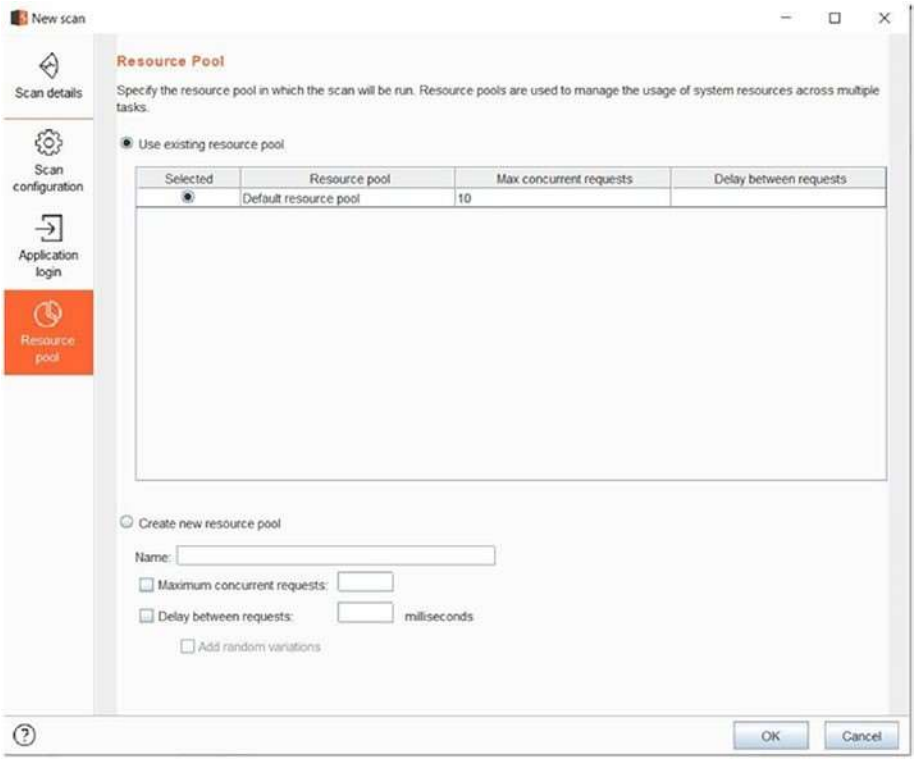


Figure 8-16. Configuring resource pools

Reporting

Reporting the issues in a presentable format is as important as finding them. Burp Suite offers an excellent reporting feature that helps us generate a report in the required format with all relevant extract about the vulnerability. The report, once generated, can be shared with relevant stakeholders for further action.

Once the crawl and audit task is complete, all the issues that were found during the scan are listed in the 'Issue activity' pane as shown in Figure 8-17. We now simply need to select the issues that we wish to be part of the report. To do this, simply right-click the issue that needs to be reported and click 'Report issue.'

Figure 8-17. Exporting issues to report

The Burp Suite reporting wizard will now ask us about the format of the report we wish to have as shown in Figure 8-18. Currently, the Burp Suite supports generating reports in HTML or XML formats.

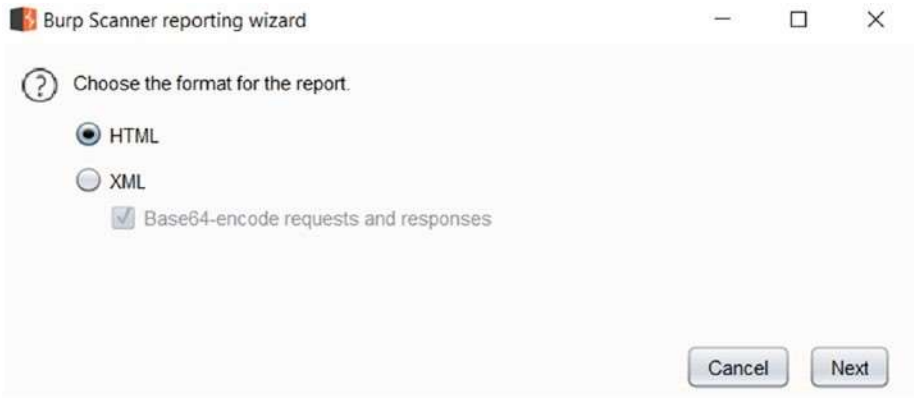


Figure 8-18. Selecting format for the report

Next, we need to select which details about the issue are required in the report as shown in Figure 8-19.

Figure 8-19. Selecting type of details to be included in the report

W then need to select whether we want full HTTP requests and responses for the reported issues or only the relevant extracts as shown in Figure 8-20.

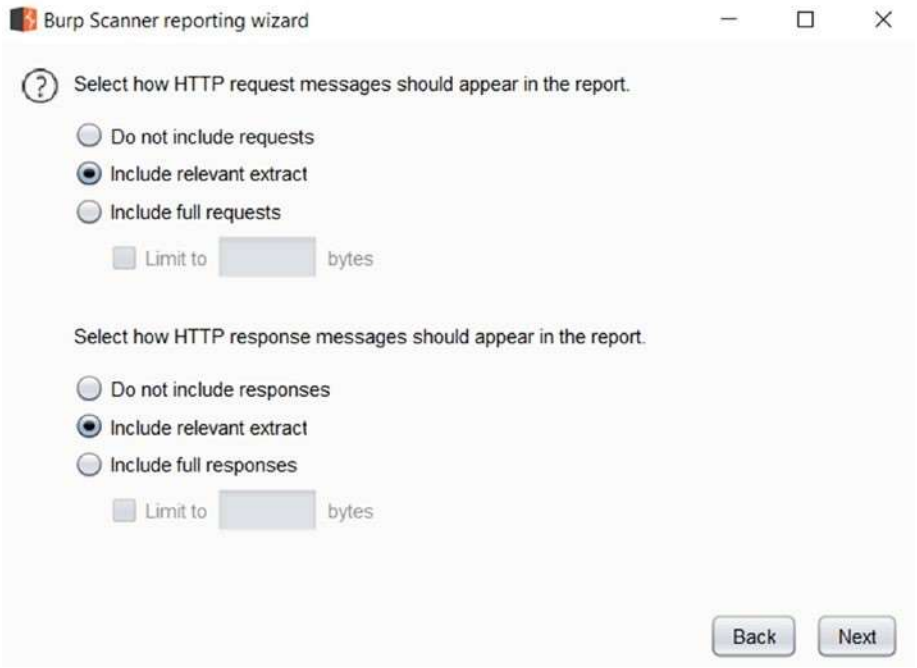


Figure 8-20. Selecting requests and response formats for report

Lastly, we need to select the name and location where we want the report to be generated along with the title of the report as shown in Figure 8-21.

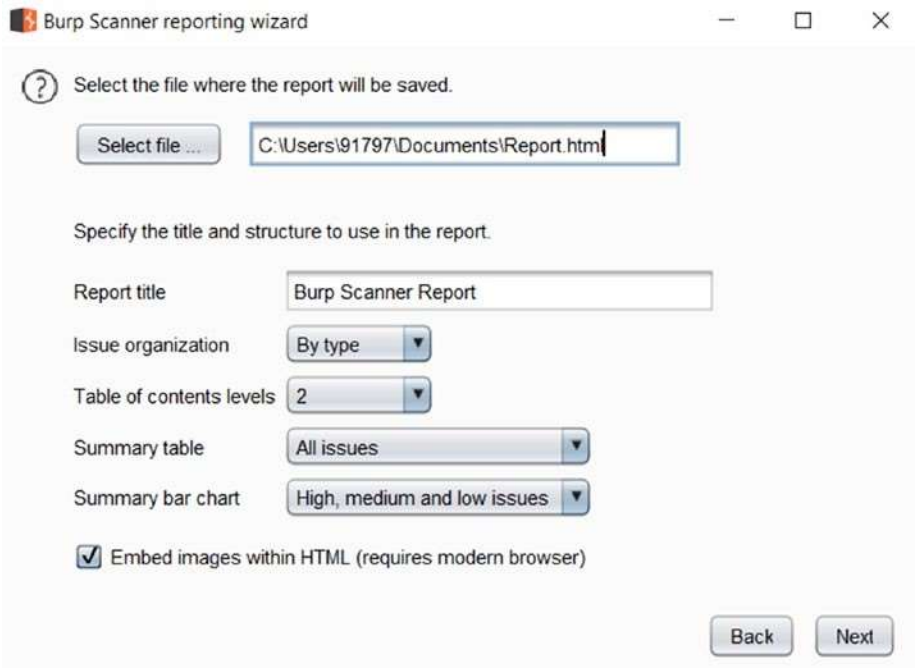


Figure 8-21. Configuring location where the report will be saved

Now the Burp Suite reporting wizard will generate a vulnerability report as shown in Figure 8-22.

Figure 8-22. Generating the report

The generated report can them be viewed in any of the browsers as shown in Figure 8-23. The report shows a summary of findings based on confidence as well as severity.

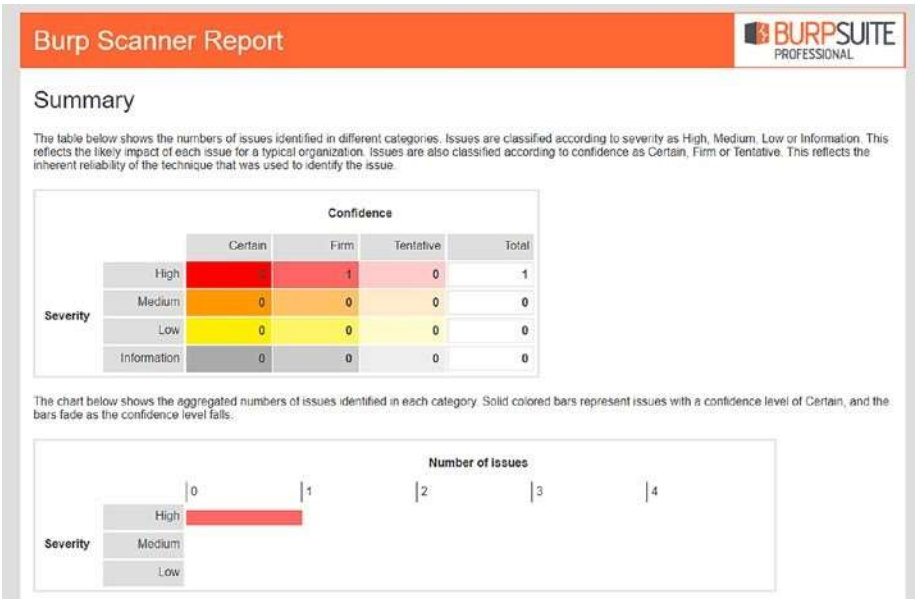


Figure 8-23. Viewing the report in the browser

The report also shows vulnerability in detail along with the relevant request and response as shown in Figure 8-24.

Scanner and reporting

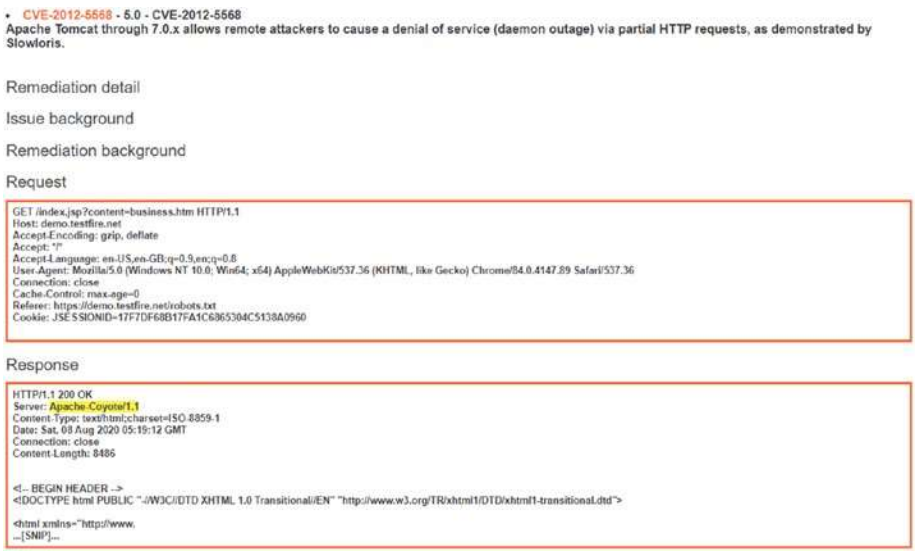


Figure 8-24. Vulnerability details in the report

Summary

In this chapter, we learned about the Burp Suite scanner and how it can be customized to effectively find application vulnerabilities in an automated manner.

In the next chapter we'll see how to use the Burp Suite Extender to install additional plugins and enhance the capabilities.

Extending Burp Suite

In the last chapter, we learned about the Burp Suite scanner, which effectively helps in automating vulnerability detection. In this chapter, we'll be exploring the Burp Suite extender feature through which we can further enhance the capabilities of Burp Suite.

Burp Suite Extensions

So far, throughout the book, we have seen various capabilities of Burp Suite for manual as well as automated vulnerability detection. We have explored various tools and utilities within Burp Suite that can be leveraged for specific tasks.

Burp Suite has now evolved more like a platform that is flexible enough to accommodate external functionalities and utilities. As we have already seen, Burp Suite does provide numerous capabilities out of the box.

However, these capabilities can be extended further using extensions. The Burp Suite Extensions come in various forms as below:

Default extensions – These extensions are listed by default, out of the box in any of the Burp Suite setups, and can be installed through the Burp Suite Extender.

Pro extensions – These are extensions that can be installed and run only on the Burp Suite Professional Edition.

Regular extensions – These are extensions that can be installed and run on the Burp Suite Community Edition as well as the Professional Edition.

Other extensions – Burp Suite has opened up APIs that developers can write with new custom extensions. Such extensions are not part of the official extension store but need to be downloaded and installed manually.

BApp Store

The easiest way to install an extension in Burp Suite is through the BApp Store. To access the BApp Store, simply navigate to Extender ➤ BApp Store as shown in Figure 9-1.

Figure 9-1. BApp Store

The BApp Store has a very easy-to-use interface with two panes as shown in Figure 9-2.

extending Burp Suite

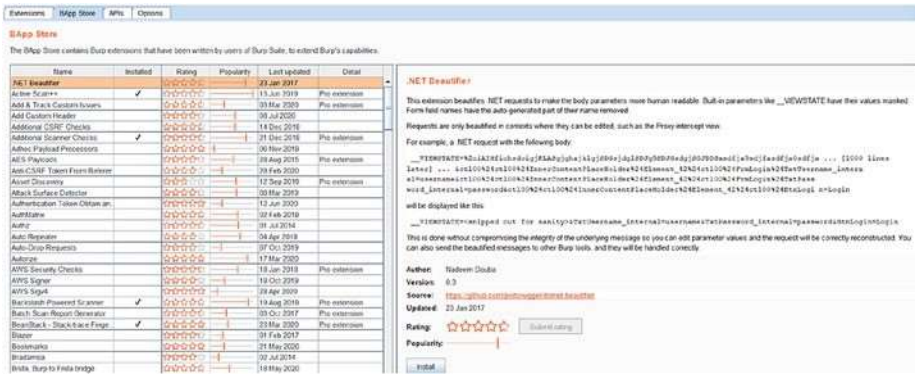


Figure 9-2. BApp Store

The left pane lists all the available extensions along with the following information:

Name of the extension,

Whether it is currently installed or not,

Rating of the extension,

The popularity of the extension,

Date when the extension was last updated,

Whether the extension is only available for use with the Burp Suite Professional Edition or if it can be used in the Community Edition as well.

The right pane details out information for any of the extensions we select from the left pane. This includes information like the following:

Details on what the extension is about and how it can be used

Author

Extension version

Source

Date when the extension was last updated

Rating and popularity of the extension

Install button to install and add the extension to current Burp Suite setup

It is important to note that new extensions keep on getting added to the BApp Store on a regular basis. To ensure the list of extensions is the latest one, simply click on the 'Refresh list' button as shown in Figure 9-3.

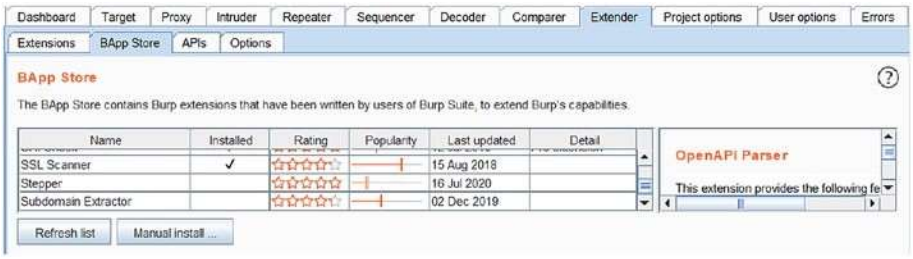


Figure 9-3. Browsing through extensions in BApp Store

Some of the useful extensions from the BApp Store are as follows:

Active Scan++ – This extension is developed to further enhance the Burp Suite's passive and active scanning capabilities.

Additional Scanner Checks – This extension adds a few more checks to a passive scanner like DOM-based XSS etc.

CSRF Scanner – This extension helps passively scan for Cross-Site Request Forgery (CSRF) vulnerabilities.

Discover Reverse Tabnabbing – This extension searches the HTML code for possible Tabnabbing vulnerabilities.

Error Message Checks – This extension helps passively detect any error or exception messages that may contain sensitive information like stack traces.

Headers Analyzer – This extension passively checks the response headers and flags all missing security headers like X-XSS-Protection, X-Frame-Options, and many more.

HTML5 Auditor – This extension checks if any of the potentially unsafe HTML5 functions have been used like storing sensitive data on client-side storage, client geolocation, etc.

J2EEScan – This extension helps improve test coverage for J2EE applications as well as adds additional test cases.

Java Deserialization Scanner – This extension adds to the Burp Suite ability to detect Java Deserialization vulnerabilities.

JavaScript Security – This extension further adds several passive checks related to JavaScript security like DOM issues, Cross-Origin Resource Sharing (CORS), etc.

Retire.js – This extension passively monitors the traffic and detects the use of any vulnerable third-party library along with necessary CVE details.

SameSite Reporter – This extension checks if the SameSite attribute has been set in cookies or not.

Software Version Reporter – This extension passively parses the traffic and reports all the software version details. This information can further help in application enumeration.

Upload Scanner – This extension adds capabilities to Burp Suite to detect file upload functionality and related vulnerabilities.

Web Cache Deception Scanner – This extension scans the application for the presence of any Web Cache Deception vulnerability.

CSP Auditor – This extension scans the response headers and checks if Content Security Policy (CSP) has been configured correctly or not.

CVSS Calculator – This extension facilitates scoring vulnerabilities using CVSS methodology from within Burp Suite.

Manual Installation

In the previous section, we saw how we could browse through, select, and install extensions using the BApp Store. Not all extensions that are written are available in the BApp Store. There could be extensions written by individual authors,

published on different websites like GitHub, etc. In such a scenario where the extension is not present in the BApp Store, we need to download it separately and install it manually. To install extensions manually, navigate to the ‘Extensions’ tab within Extender as shown in Figure 9-4.

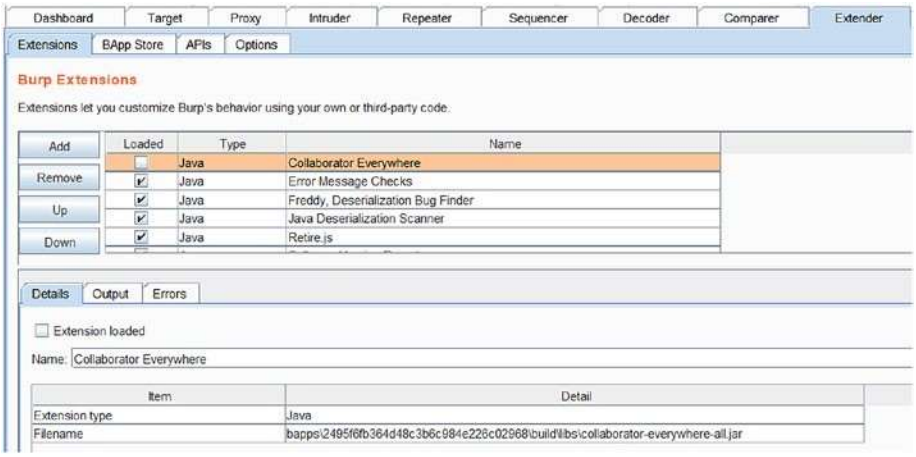


Figure 9-4. Adding extensions manually

Burp Suite accepts the installation of third-party extensions with the following formats as shown in Figure 9-5.

- Java
- Python
- Ruby

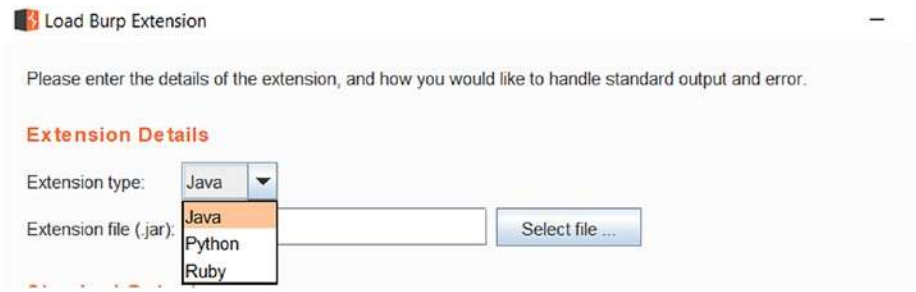


Figure 9-5. Selecting type of the extension

The other options include whether we wish to show the output and errors after extension installation on the console or if we wish to save it to a file as shown in Figure 9-6.

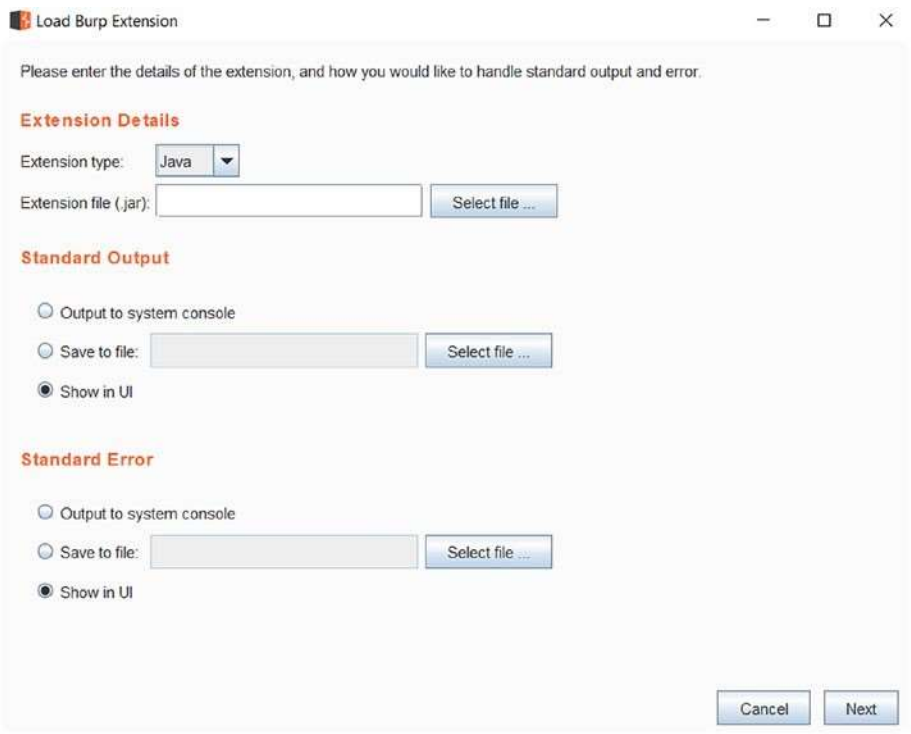


Figure 9-6. Adding extensions manually

To install an extension, select the extension type (Java / Python / Ruby) and then simply browse and select the location where the extension is located on disk as shown in Figure 9-7.

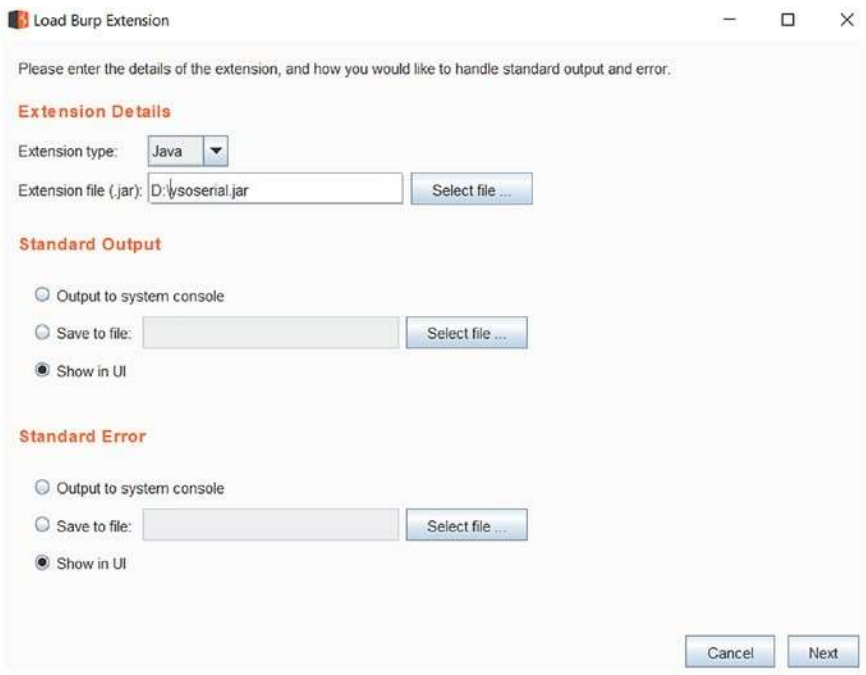


Figure 9-7. Selecting the extension file

If the extension installation gets completed successfully, a message is displayed as shown in Figure 9-8.

Figure 9-8. Loading extensions manually

Settings

Now that we have seen how to install extensions either through the BApp Store or manually, let's go through some additional settings related to extensions.

- Settings can be accessed by navigating to 'Extender ➤ Options' as shown in Figure 9-9. The first two settings define if you want to automatically reload extensions when you start the Burp Suite and if you wish to automatically update the extensions on Startup.

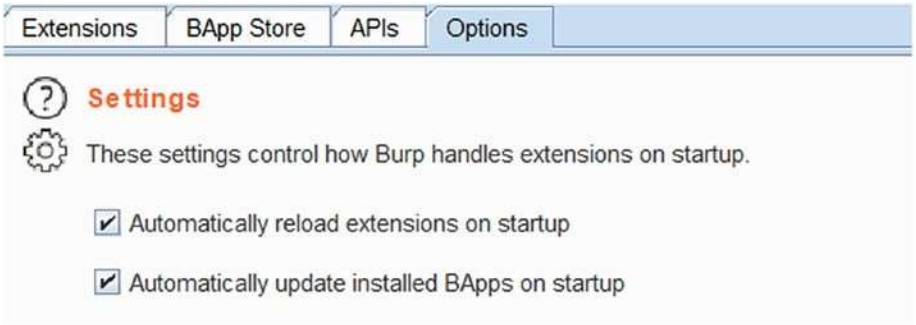


Figure 9-9. BApp Store options

The next setting is related to the Java environment. Most of the extensions are written in Java. To ensure these extensions run correctly, it might be required to provide a path to any additional library dependencies as shown in Figure 9-10.

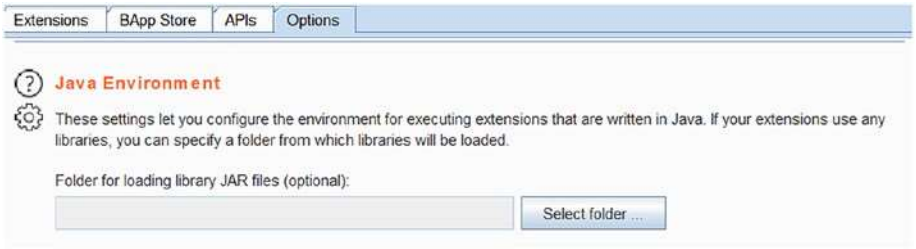


Figure 9-10. Configuring the Java environment

The next setting is related to setting up the Python environment. Some extensions require a Python interpreter implemented in Java called Jython.

Jython can be downloaded and installed from https://www.jython.

org/download. Once installed, you need to update the path to the Jython installation directory as shown in Figure 9-11.

Figure 9-11. Configuring the Python environment

The last setting is related to setting up the Ruby environment. As we have seen earlier, Burp Suite supports extensions written in Ruby as well; hence we need to specify the path to the Ruby interpreter as shown in Figure 9-12. In order to run a Burp Suite extension written in Ruby, JRuby needs to be downloaded and installed from https://www.jruby.org/ download.

Figure 9-12. Configuring the Ruby environment

Other Useful Extensions

Earlier in this chapter, we already saw some useful extensions available in the BApp Store. As the Burp Suite gives Application Programming Interface (APIs) to developers, it is easy to write and develop custom extensions as required.

Here are some additional useful extensions that can be manually installed to Burp Suite.

sometime – This extension can be downloaded from https://github.com/linkedin/sometime. This extension passively monitors the traffic to check if the application is vulnerable to the Same Origin Method Execution.

burp-suite-gwt-scan – This extension can be downloaded from https://github.com/augustd/ burp-suite-gwt-scan - This extension helps automatically identify insertion points for GWT (Google Web Toolkit) requests when sending them to the active Scanner or Burp Intruder.

Admin panel finder – This extension can be downloaded from https://github.com/moeinfatehi/

Admin-Panel_Finder -This extension assists in the

enumeration of infrastructure and application Admin Interfaces that might have been left open by mistake.

Pwnback – This extension can be downloaded from https://github.com/P3GLEG/PwnBack. This extension helps to retrieve old and archived versions of the application if present. It can be useful to compare the old and current versions of the application to check the changes and associated vulnerabilities.

Minesweeper – This extension can be downloaded from https://github.com/codingo/Minesweeper

-This extension helps detect scripts being loaded from over 23000+ malicious cryptocurrency mining domains (cryptojacking).

For an additional and comprehensive list of the Burp Suite extensions, refer to https://github.com/snoopysecurity/awesome-burp- extensions.

APIs

Throughout this chapter, we have seen the use of Extender to add and install new extensions that significantly improve the Burp Suite capabilities. Burp

Suite offers another useful feature in the form of the Application Programming Interface (API). Using these APIs it is possible to write our own extensions. The list of available APIs and detailed guidance on their usage is available under the 'Extender ➤ APIs' tab as shown in Figure 9-13.

extending Burp Suite

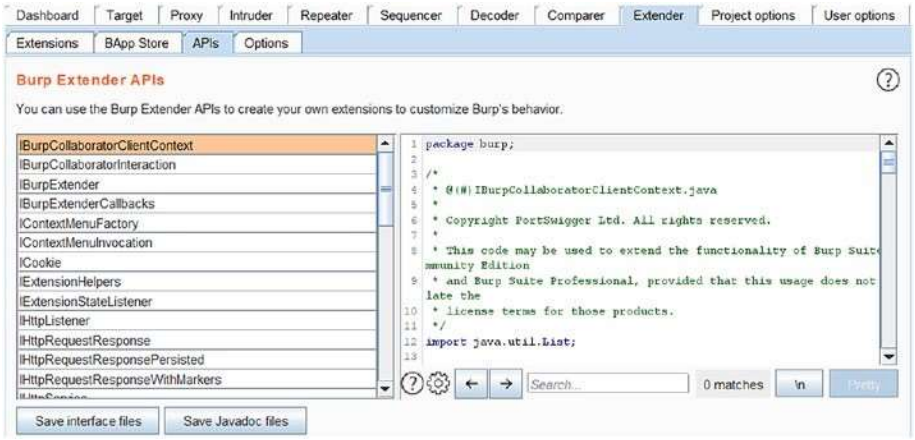


Figure 9-13. The Burp Suite Extender APIs

Summary

In this chapter, we got familiar with the Burp Suite extender, which allows enhancing the Burp Suite capabilities through external extensions. We explored the BApp Store, which has a list of many useful extensions and we also learned to install an extension manually in case it's not present in the BApp Store. Last, we listed a few additional extensions apart from those officially present in the BApp Store.

In the next chapter we'll see how we can leverage the Burp Suite capabilities to test mobile applications and APIs. We'll also see the complete workflow for testing an application using the Burp Suite.

Testing Mobile Apps and APIs with Burp Suite

In the last chapter, we learned about the Burp Suite extender feature, which allows enhancing the Burp Suite's capabilities through third-party extensions. In this final chapter, we'll glance through how Burp Suite can be used to test APIs and mobile applications as well. We'll conclude with a quick summary of the workflow for testing any web application using Burp Suite.

API Security Testing with Burp Suite

Throughout this book, we have been learning about various capabilities of Burp Suite, which can be used for Web Application security testing. However, today's

modern applications are more interoperable and interconnected. This is achieved through the use of Application Programming Interfaces (APIs).

Exposing APIs significantly helps in automating tasks; however, at the same time, it does introduce security risks as well if not implemented securely. While most of the web application vulnerabilities apply to APIs, there are a few vulnerabilities that are specific to APIs. OWASP has published the Top 10 vulnerability list for API, which can be found at https://owasp.org/www-project-api-security/.

The approach for security testing of APIs through Burp Suite is very similar to the regular web applications that we have seen so far in the book. As APIs communicate over HTTP/HTTPS protocol, the traffic can be intercepted and tampered in Burp Suite just like any other regular web application request and response. For performing security testing on APIs using Burp Suite, we can use one of the following approaches:

Crawl the application in a regular way and figure out the endpoints belonging to APIs. Once the API endpoints are identified, the corresponding requests can be sent to the Repeater or Intruder for further testing.

Many times, the APIs are invoked through the User Interface (UI) functionalities in the application. In such a case, you can simply create a new 'Crawl and Audit' task in the Burp Suite scanner and ensure all scanner checks and tasks are complete.

There could be a set of APIs that are not directly invoked from any of the UIs. Such APIs are often tested manually using tools like Postman. We can easily integrate Postman with Burp Suite to capture all required API traffic. Once the required API requests and responses are in Burp Suite, it is just a matter of testing them further using Repeater or Intruder as necessary.

We'll now see how we can integrate Postman with Burp Suite. Postman is a popular tool used for manual API testing. It can be downloaded from https://www.postman.com/downloads/. The Postman application interface is as shown in Figure 10-1.

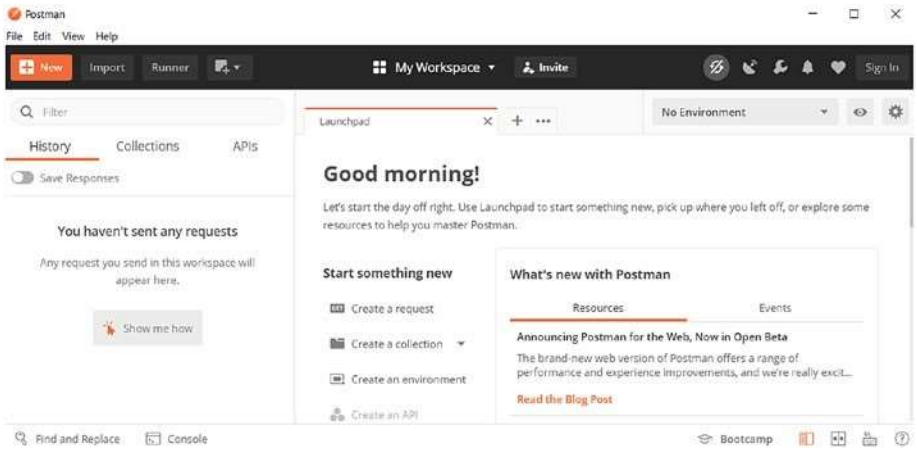


Figure 10-1. The Postman tool

In order to configure Postman to work along with Burp Suite, click on the 'Settings' option in the upper right corner as shown in Figure 10-2.

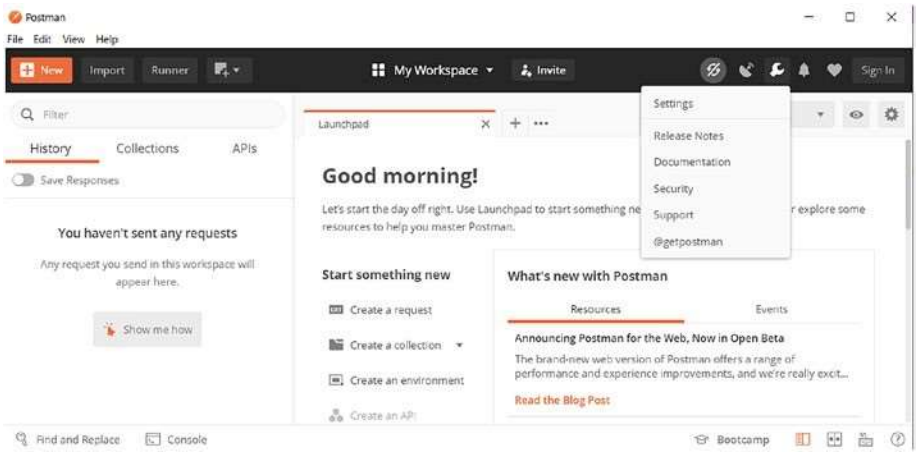


Figure 10-2. Navigating to the settings in the Postman tool

A new Settings window will open up as shown in Figure 10-3. Now further navigate to the proxy tab and select 'Add a custom proxy configuration' and enter the proxy server address as that of the machine running the Burp Suite (usually localhost or 127.0.0.1) and port as 8080 or the one on which we want the Burp Suite proxy listener to be active on.

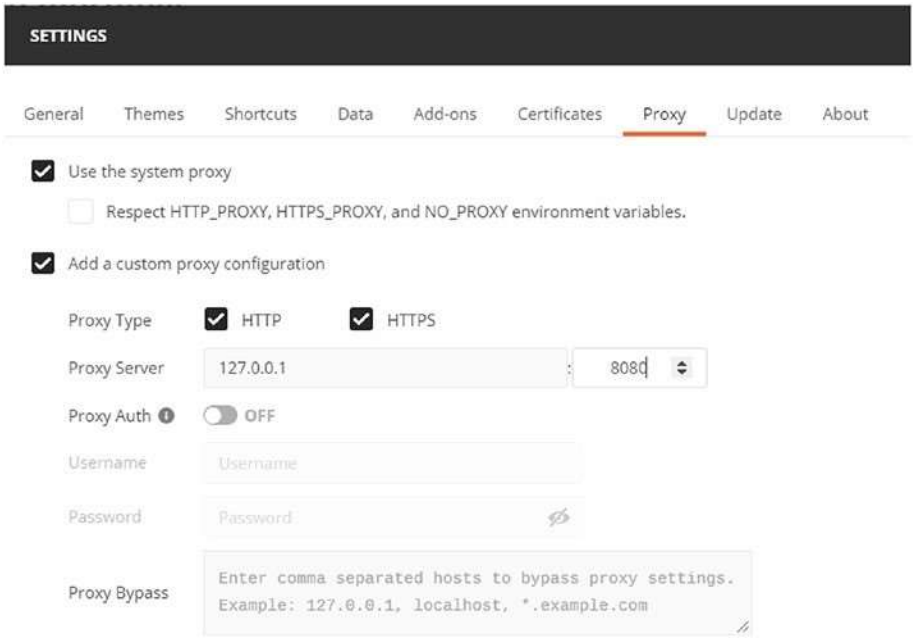


Figure 10-3. Configuring the proxy in the Postman tool

Now that we are done with the configuration on the Postman side, we need to ensure that the right proxy configuration is also done on the Burp Suite Side. To ensure the correct proxy is configured in the Burp Suite, navigate to the Proxy tab and Options as shown in Figure 10-4, and ensure the IP address and port number match to what was configured earlier in Postman.

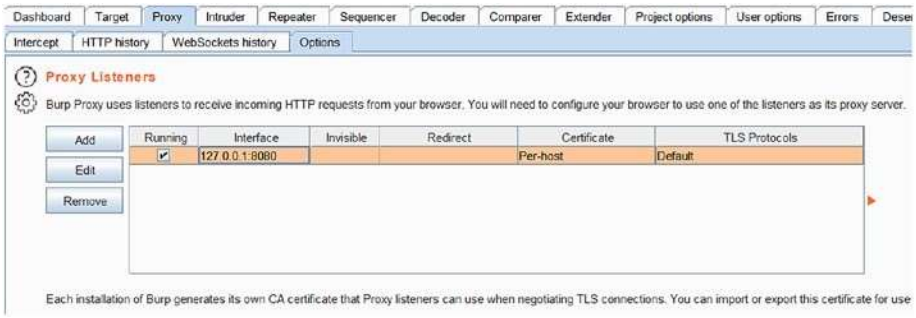


Figure 10-4. Setting up the Burp Suite proxy listener

Once both the Postman and Burp Suite are configured to work together, all the traffic generated from Postman will now pass through Burp Suite. This is very similar to how we configured our browsers to work along with Burp Suite.

Now that we have seen how to configure the Postman to work along with Burp Suite, we'll see how the reverse works; that is, how do we export data from Burp Suite into the Postman for selective testing?

While the Burp Suite Repeater and Intruder serve most of the purposes for performing security testing of an API, there could be a need to test a particular API in the Postman interface. In such a case, it is possible to export the API request from Burp Suite into the Postman tool.

For exporting an API request from Burp Suite to the Postman, we would need to install an extension called 'Postman Integration'. Simply navigate to the Extender tab and open the 'BApp Store' and install the 'Postman Integration' extension as shown in Figure 10-5.

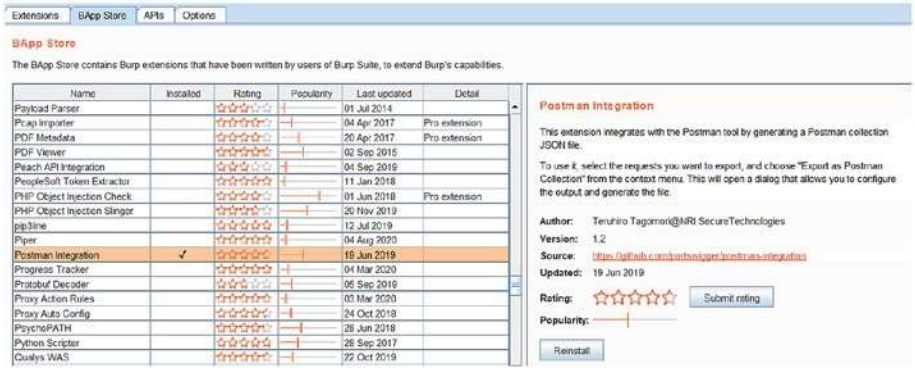


Figure 10-5. Installing the Postman Integration extension in the Burp Suite through BApp Store

Once the extension is installed:

Go to the Target tab,

Select the request you want to export to Postman,

Right-click on the request to be exported,

Select option ‘Export as Postman Collection’ as shown in Figure 10-6.

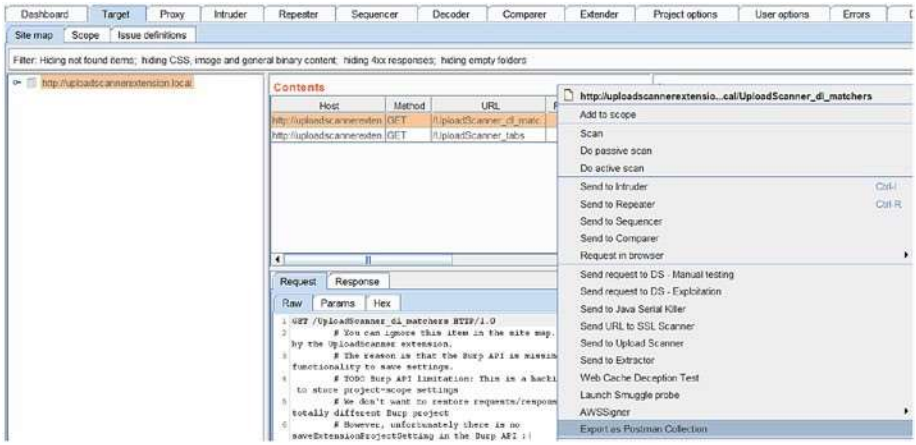


Figure 10-6. Exporting requests in the Burp Suite as Postman collection

A new window will pop up, as shown in Figure 10-7. Enter the required Collection Name and Folder name and click on the Export button. The collection would then be exported and saved to the specified location.

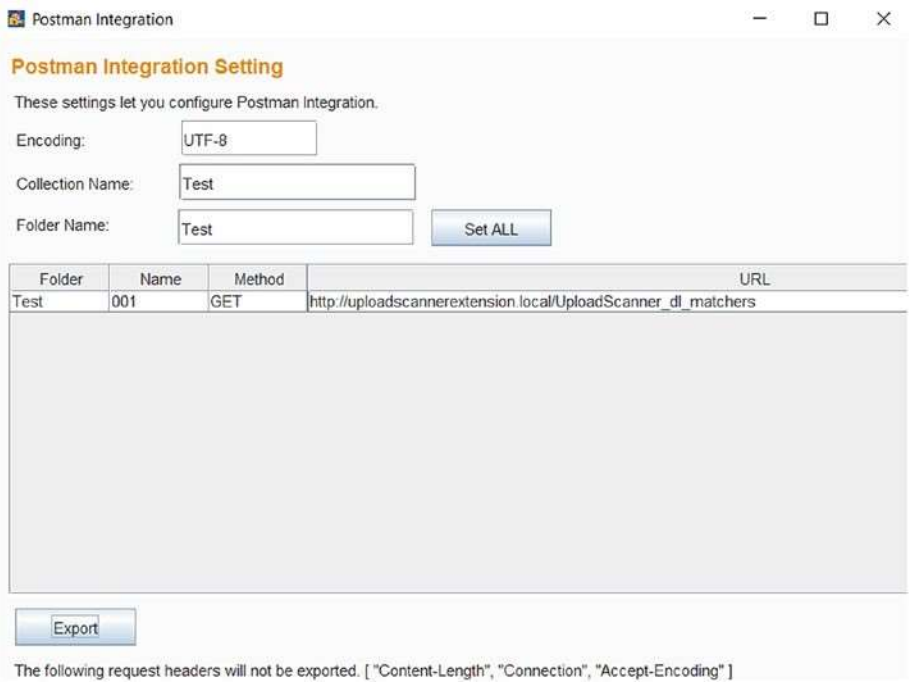


Figure 10-7. Configuring the Postman integration settings

Next, open the Postman application and click on Import, as shown in Figure 10-8. Click on choose files and select the file that we exported from Burp Suite earlier.

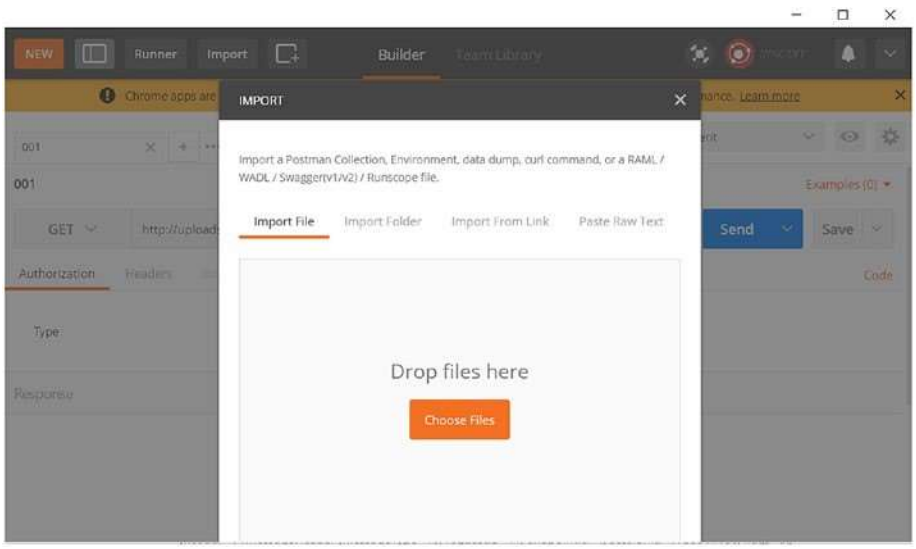


Figure 10-8. Importing the collection into the Postman tool

We can now see the API request exported from the Burp Suite into the Postman application, as shown in Figure 10-9.

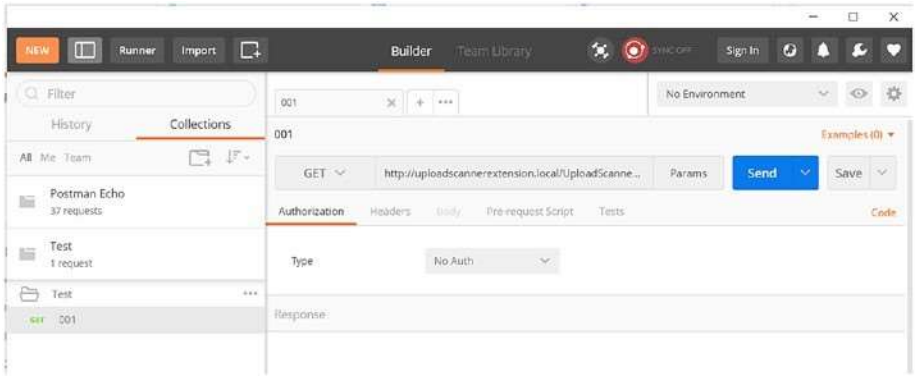


Figure 10-9. Collection imported in the Postman tool

Mobile Application Security Testing with Burp Suite

In the previous section, we saw how Burp Suite could be configured along with the Postman to perform security testing on APIs. In this section, we'll

have an overview of how we could leverage the Burp Suite capabilities for performing security testing of Mobile Applications.

Before we get into further details about Mobile Application security testing, it is important to understand the fact that the Burp Suite literally acts and serves as an HTTP proxy. This means we can effectively use Burp Suite for performing security testing on any device or application that interacts over HTTP or HTTPS protocol.

Mobile applications are no different; they use the same HTTP / HTTPS protocol for communication; hence that traffic can be routed through the Burp Suite just like any other regular web application. Figure 10-10 shows a mobile application client (the equivalent of the browser on PC), which is passing all the traffic through Burp Suite to the Application Server.

Figure 10-10. Connecting mobile application to the Burp Suite

We'll now see how we can configure Burp Suite to work along with the mobile application. First, we need to ensure that the correct Burp Suite proxy is set to listen on all interfaces. To do so, navigate to the Proxy ➤ Options tab, as shown in Figure 10-11.

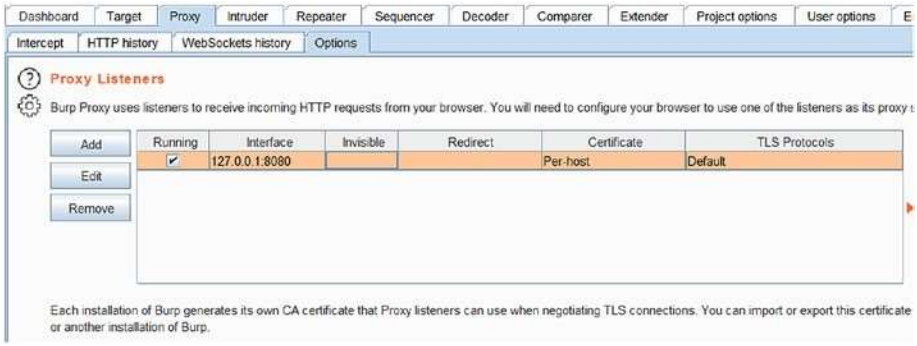


Figure 10-11. Setting up the Burp Suite proxy listener

Now click on the ‘Add’ button, and a new window will pop up as shown in Figure 10-12. Configure the port number and select ‘Bind to address’ as ‘All interfaces’ and click ‘OK’.

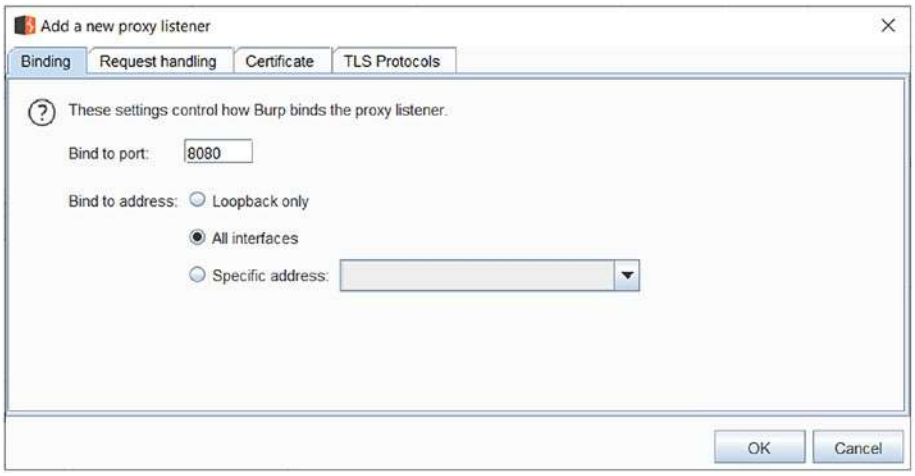


Figure 10-12. Setting up the Burp Suite proxy listener

Now notice the proxy listener section as shown in Figure 10-13, which lists the interface as ‘*:8080’.

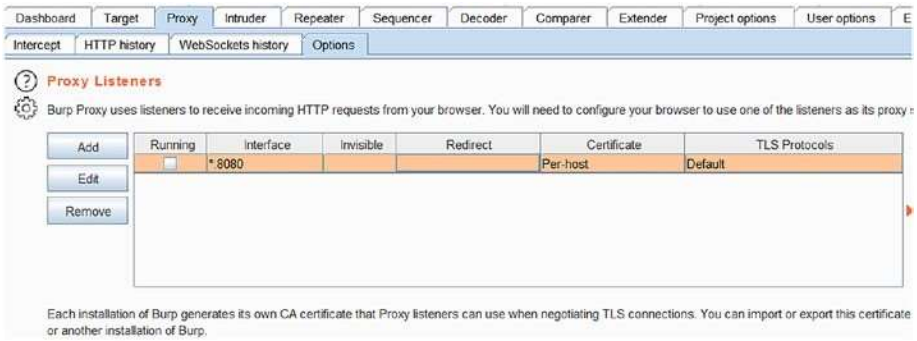


Figure 10-13. Setting up the Burp Suite proxy listener

Now that we have configured the Burp Suite proxy to listen on port 8080 on all available interfaces, we’ll move ahead to the mobile device configuration.

It is important to note that in order to configure the mobile device to work along with Burp Suite, both the system running the Burp Suite and the mobile

device need to be in the same logical network. The simplest way to achieve this is by connecting the mobile device and the system running Burp Suite to the same Wireless Access Point. Once the mobile device and the system running Burp Suite are connected to the same network, we need to configure the network settings on mobile to use Burp Suite as a proxy.

To configure the network proxy on the mobile device, go to the Wireless Settings and select the Wireless Network that you are connected to, as shown in Figure 10-14.

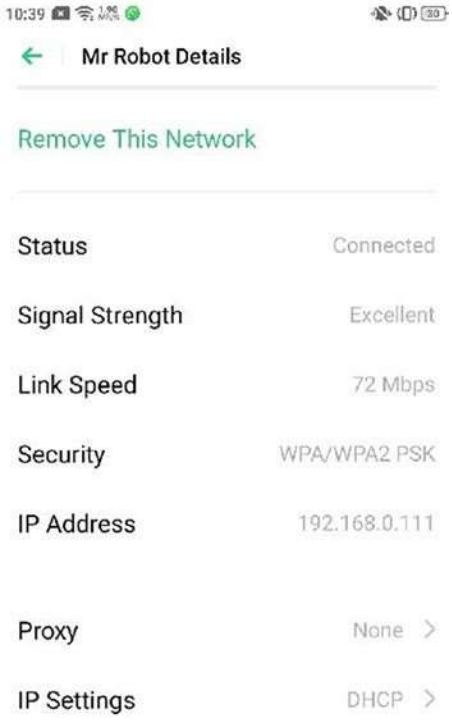


Figure 10-14. Configuring network proxy on mobile device

Now click on the Proxy option, and a new configuration window will open as shown in Figure 10-15.

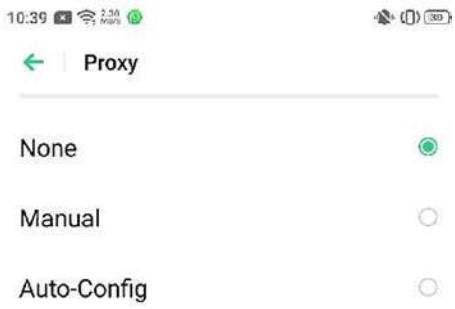


Figure 10-15. Configuring network proxy on mobile device

By default, the network proxy is set to ‘None’. We need to change this to ‘Manual’, as shown in Figure 10-16.

Figure 10-16. Configuring network proxy on mobile device

Now that we have changed the Proxy type to manual, we need to enter the IP address of the system where Burp Suite is running along with the port number on which the Burp Suite proxy service is listening to, as shown in Figure 10-17.

Figure 10-17. Configuring network proxy on mobile device

Once the manual proxy has been configured, all the traffic originating from the mobile application would now be routed through Burp Suite. Once the requests are in Burp Suite, they can be tampered with Repeater or Intruder just like any other regular HTTP request.

It’s important to note two points with regard to performing security testing on mobile applications using Burp Suite:

testing Mobile apps and apis with burp suite

The Burp Suite can only help to execute manual security tests on the mobile application and, to a certain extent, perform dynamic application security testing.

The exact process of configuring the network proxy on mobile devices varies as per the type and version of the operating system they run on. However, at a high level, the process would be similar to what we discussed in this section.

Security Testing Workflow with Burp Suite

Throughout the book, we have seen all aspects of Burp Suite and its capabilities. We have seen various tools and utilities that are provided out of the box as well as the use of third-party extensions, which significantly enhance the Burp Suite capabilities.

Now, as we are at the end of the book, it would be worth summarizing the workflow or approach for effectively using Burp Suite to test the security of web applications.

Following is the phased approach that one can follow for effective use of Burp Suite:

Get the right setup and configuration – Before we actually begin using Burp Suite, it is important that it's set up and configured correctly.

Make sure the right edition and the latest version of Burp Suite are being used.

Configure the browser proxy settings to work along with Burp Suite.

Install the Burp Suite CA certificate into the browser.

Configure the platform authentication, upstream proxy, and socks proxy as required.

Review and define the Burp Suite HotKeys.

Ensure automatic project backup is enabled and configured correctly.

Ensure project options like Timeouts, Hostname resolution,

Out of Scope Requests, Redirections, TLS Configuration, Session Handling Rules, Cookie Jar, and Macros are configured as needed.

Make sure the proxy listener is configured and running properly.

Ensure all the required extensions are installed and loaded.

Check that the target application can be patched using the Burp Suite Infiltrator.

Crawl and understand the application – Once the Burp Suite is appropriately configured, it is important to crawl, surf, and browse the target application to know more about it.

Use the crawl function in the scanner to browse the application.

Make use of the content discovery feature.

Manually browse through the critical workflows.

Carefully observe the target tab and monitor the HTTP requests.

Find out and highlight interesting requests with parameters.

Use the analyze target feature to get an overview of the application scope.

testing Mobile apps and apis with burp suite

Use engagement tools to find comments, scripts, and references.

Monitor the issues reported by the passive scans.

Attack the application – Now that we have done enough reconnaissance, it's time to attack selective application functionalities.

Run an audit task using the Burp Suite scanner.

Find out the interesting requests, especially the one with parameters, and send the request to the Repeater for further investigation.

Tamper and play around with the request, parameters, headers, and body using the repeater.

If a request and a parameter need to be tested against bulk payloads, then make use of Intruder.

Use the comparer to analyze and interpret differences in various responses.

Use a sequencer to test the strength of tokens.

Use the decoder for encoding or decoding any of the parameter values as required.

Test for the Clickjacking vulnerabilities using the Clickbandit tool.

Generate proof of concept for Cross-Site Request Forgery attack using the CSRF PoC generator.

Make use of the Burp Suite collaborator to effectively detect

out-of- the band vulnerabilities like XML External Entity Injection (XXE) and Server-Side Request Forgery (SSRF)

Monitor the issues pane under the target tab for all vulnerabilities found.

Select the required vulnerabilities and export them into an HTML report.

Summary

In this chapter, we saw how we could leverage Burp Suite capabilities for performing security testing on APIs as well as mobile applications. We also summarized the workflow that can be followed to make the best use of Burp Suite for web application security testing.