

CHAPTER 7

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.

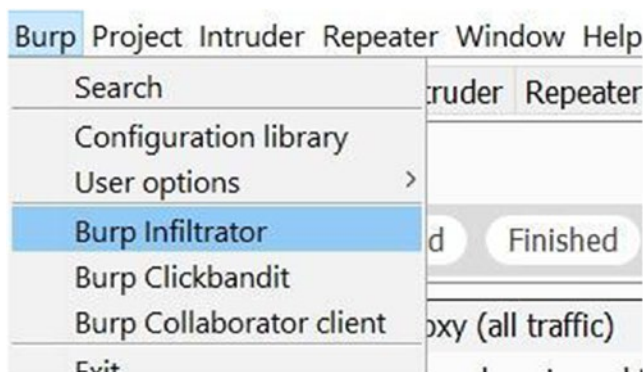


Figure 7-1. Navigating to the Burp Suite Infiltrator

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

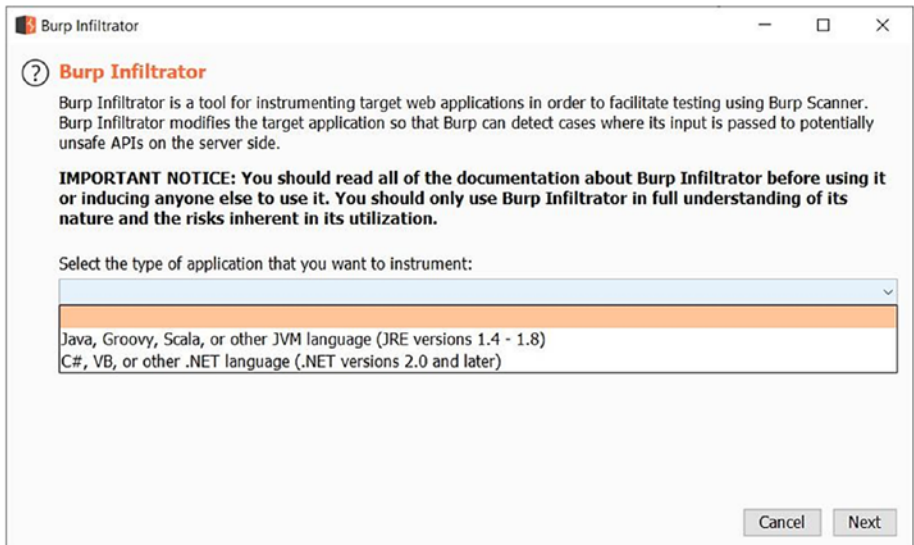


Figure 7-2. Generating the Infiltrator agent

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.

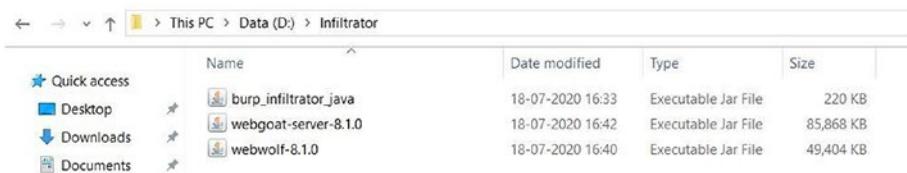
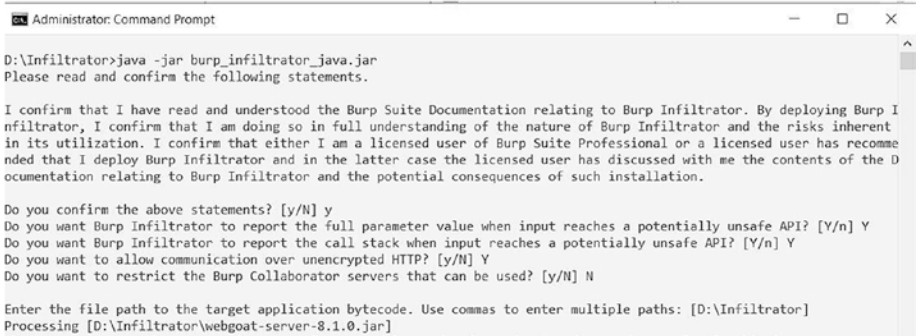


Figure 7-5. *Newly generated Infiltrator agent*

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.



```

Administrator: Command Prompt
D:\Infiltrator>java -jar burp_infiltrator_java.jar
Please read and confirm the following statements.

I confirm that I have read and understood the Burp Suite Documentation relating to Burp Infiltrator. By deploying Burp Infiltrator, I confirm that I am doing so in full understanding of the nature of Burp Infiltrator and the risks inherent in its utilization. I confirm that either I am a licensed user of Burp Suite Professional or a licensed user has recommended that I deploy Burp Infiltrator and in the latter case the licensed user has discussed with me the contents of the Documentation relating to Burp Infiltrator and the potential consequences of such installation.

Do you confirm the above statements? [y/N] y
Do you want Burp Infiltrator to report the full parameter value when input reaches a potentially unsafe API? [Y/n] Y
Do you want Burp Infiltrator to report the call stack when input reaches a potentially unsafe API? [Y/n] Y
Do you want to allow communication over unencrypted HTTP? [y/N] Y
Do you want to restrict the Burp Collaborator servers that can be used? [y/N] N

Enter the file path to the target application bytecode. Use commas to enter multiple paths: [D:\Infiltrator]
Processing [D:\Infiltrator\webgoat-server-3.1.0.jar]

```

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.

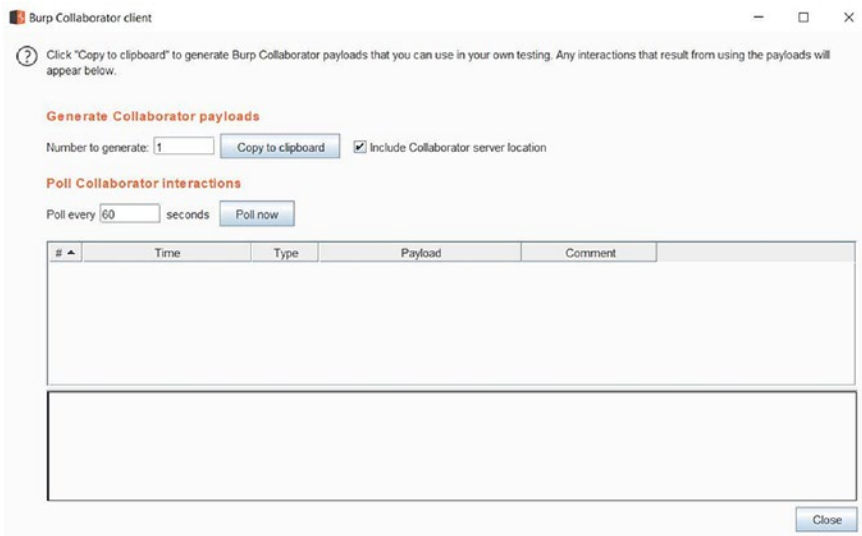


Figure 7-7. The Burp Suite Collaborator client

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

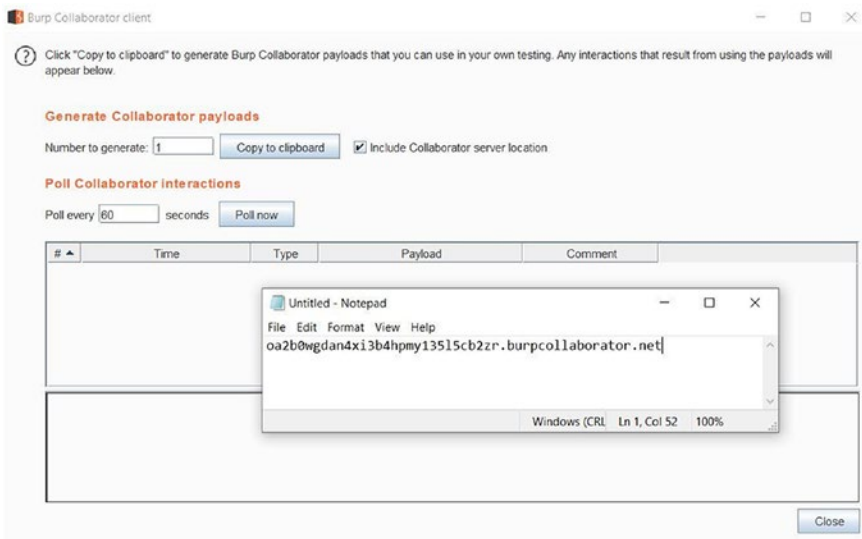


Figure 7-8. Configuring the Collaborator client

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.

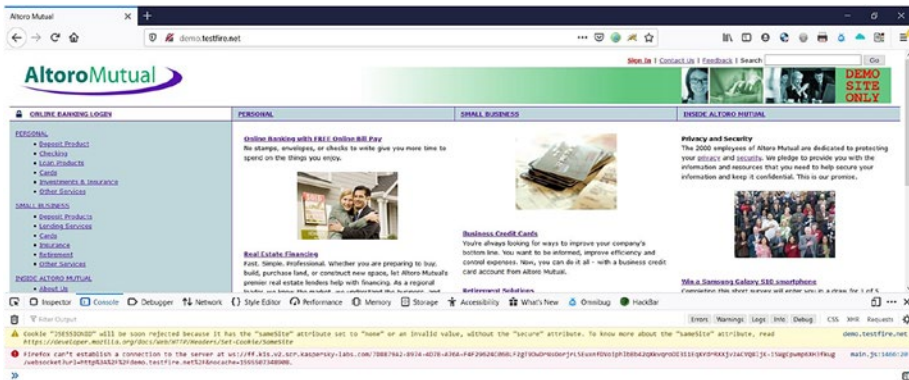


Figure 7-10. Target for Clickbandit

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

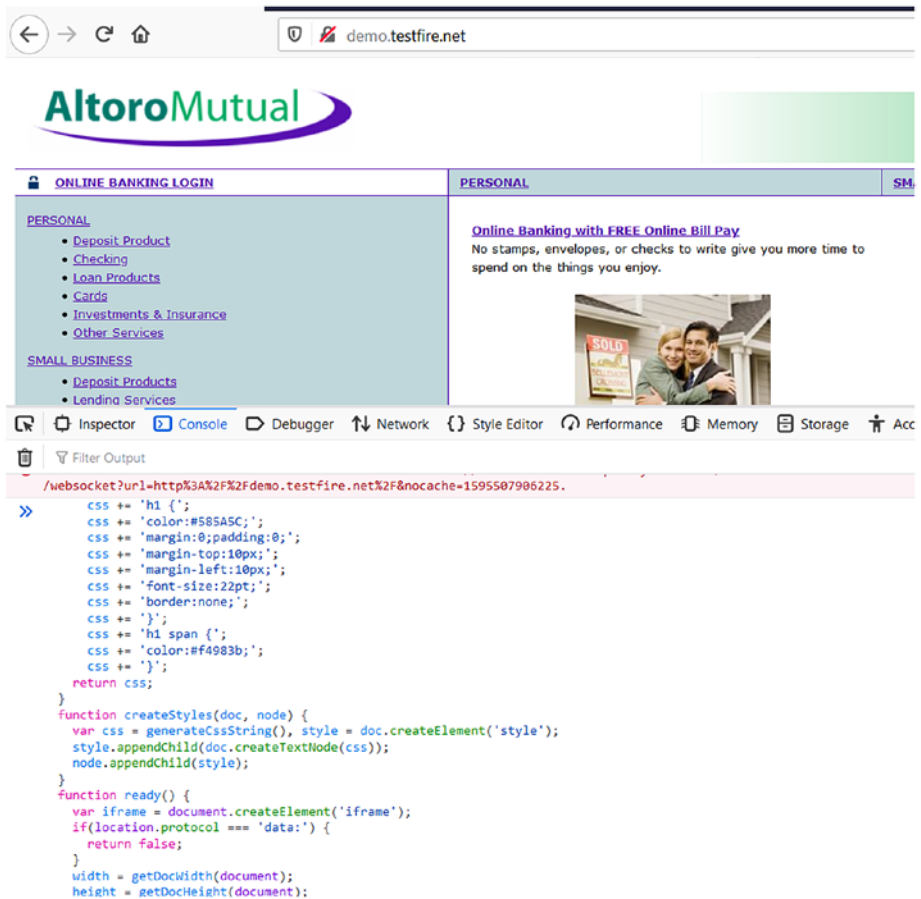


Figure 7-11. Copying the Clickbandit code in browser console

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.

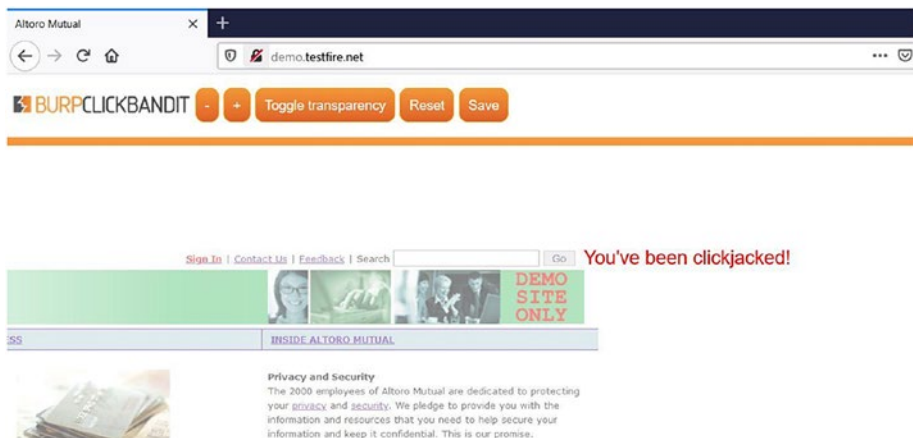


Figure 7-12. The Clickbandit UI

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.

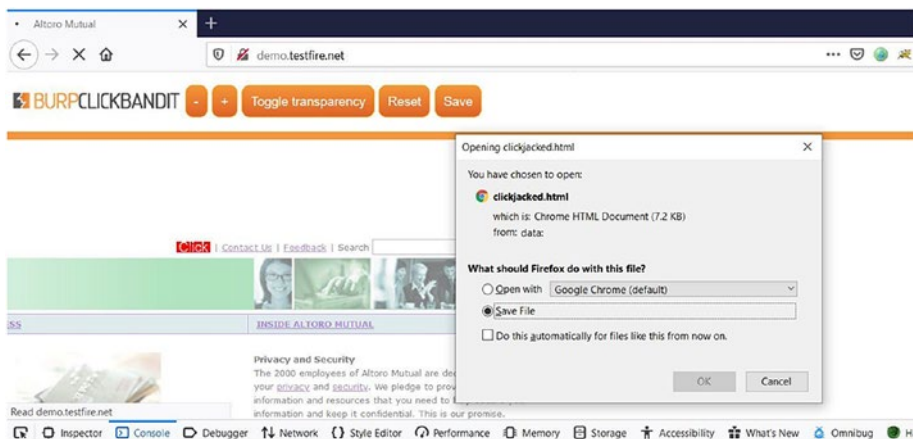


Figure 7-13. Saving the Clickbandit code

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

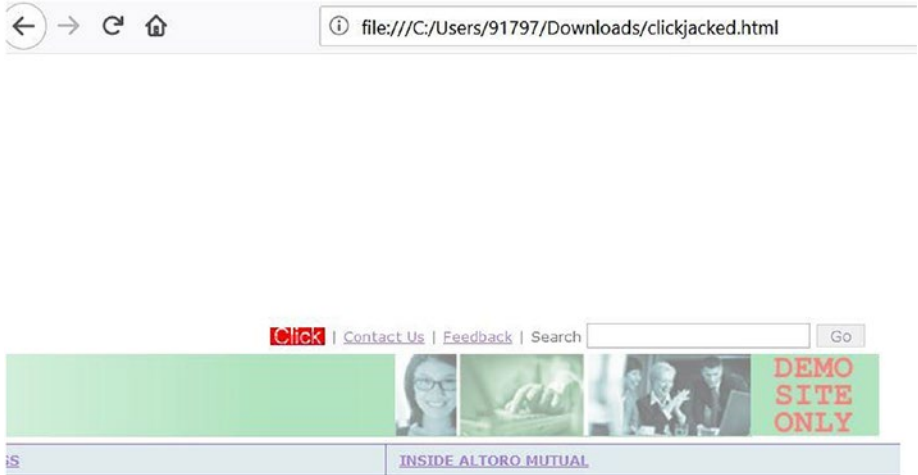


Figure 7-14. Executing the Clickbandit code

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.

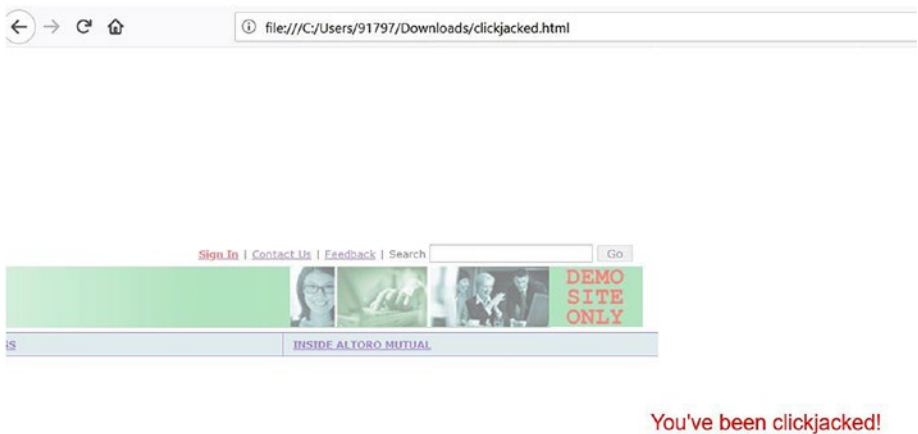


Figure 7-15. Executing the Clickbandit code

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.

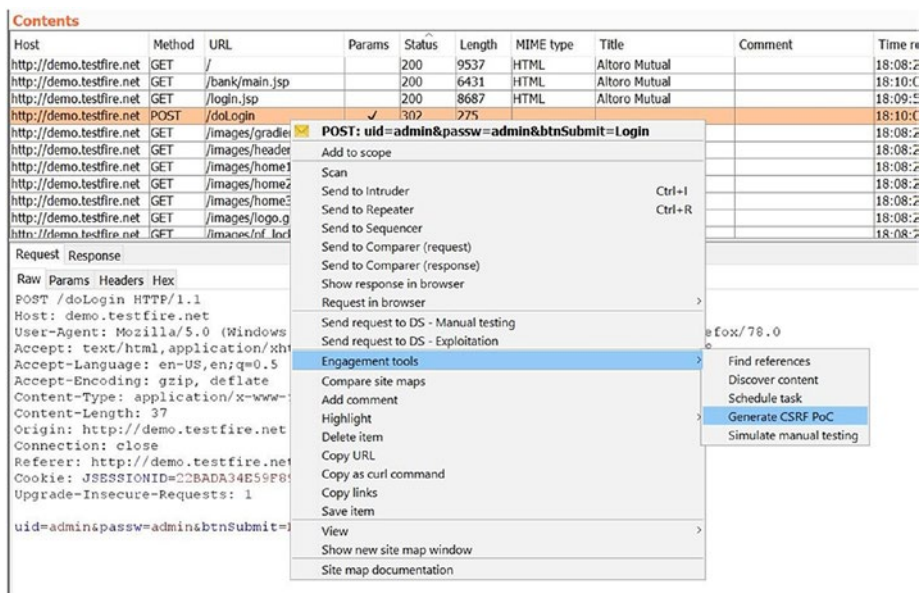


Figure 7-16. Sending request to CSRF PoC generator

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

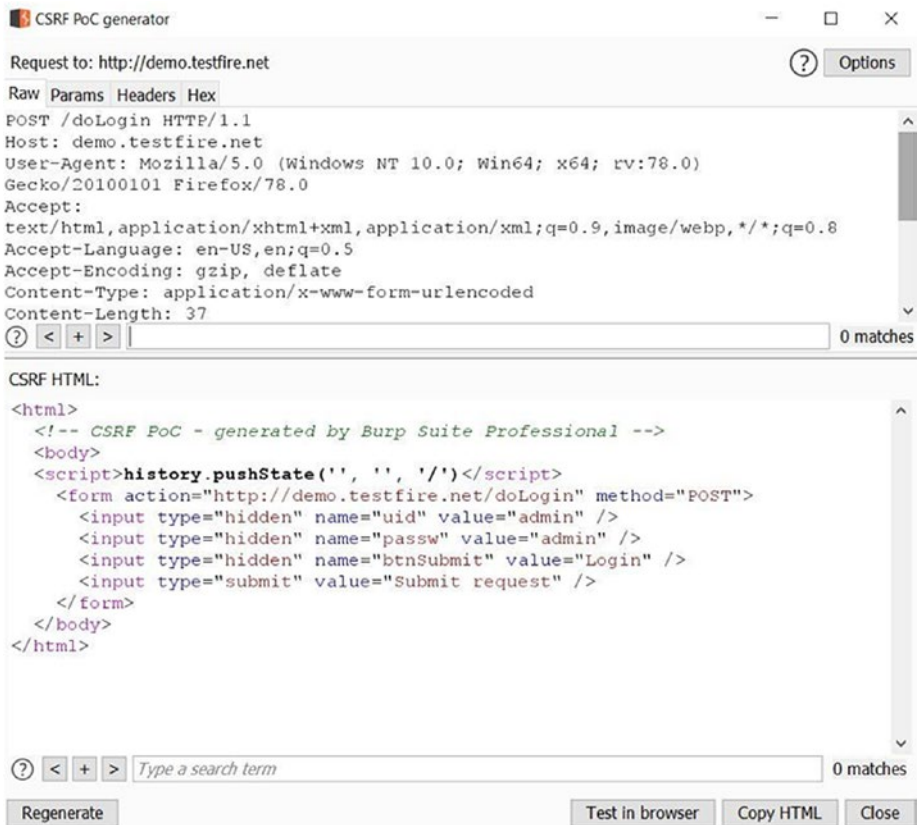


Figure 7-17. CSRF PoC generator

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.

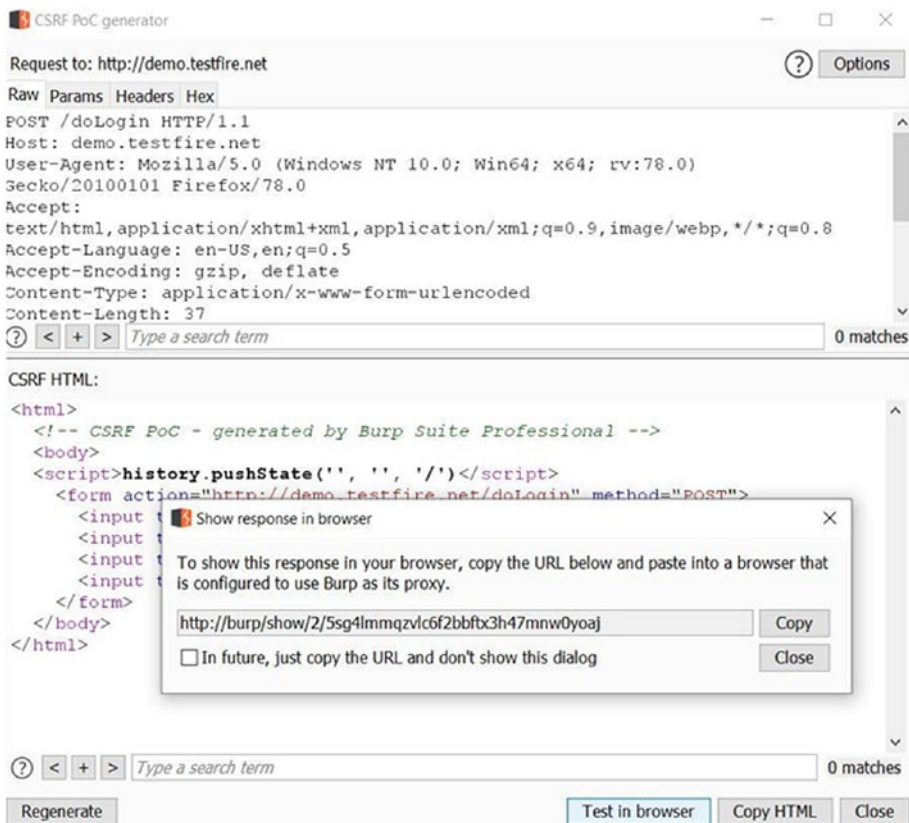


Figure 7-18. CSRF PoC generator

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

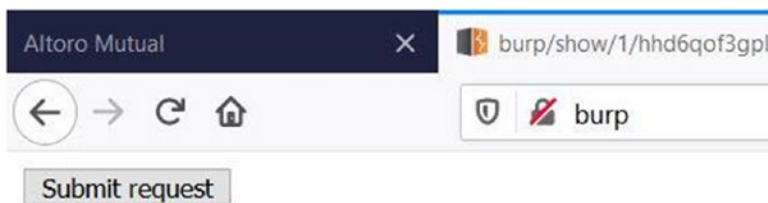


Figure 7-19. Verifying the CSRF PoC in browser

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

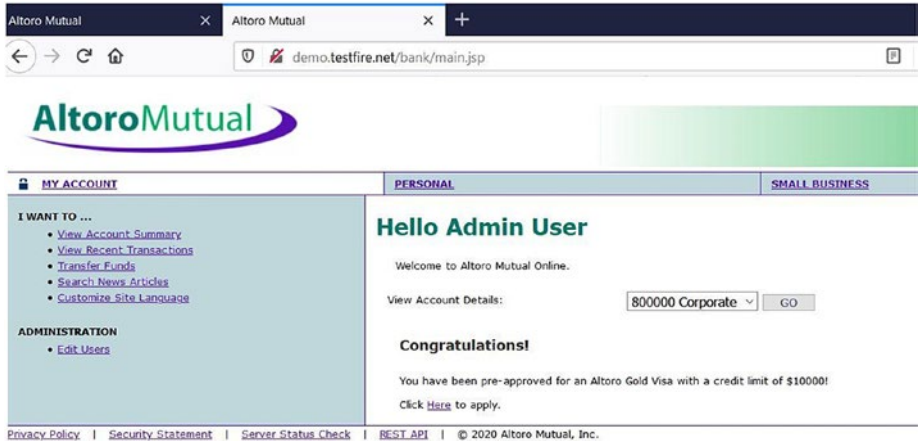


Figure 7-20. Verifying the CSRF PoC in browser

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.

Exercises

1. Use Infiltrator to instrument any of your target Java applications.
2. Find a vulnerable CSRF request and try to generate a proof-of-concept using the CSRF PoC generator.
3. Generate a clickjacking proof-of-concept code for your target web application.