Web Application Testing Lab

Intern: **Radha Singh**

Assessment date: 20/2/2026

Environment: Kali VM (192.168.75.128) attacking Metasploitable2 VM (192.168.75.129)

1. Introduction

This Web Application Testing Lab was conducted to gain practical, hands-on experience in identifying and analyzing common web application vulnerabilities aligned with the **OWASP Top 10**. The objective was to simulate a real-world web security assessment against a deliberately vulnerable environment and to document findings in a structured and professional manner, similar to an industry�standard penetration testing report.

The target environment used for this assessment was a **Damn Vulnerable Web Application (DVWA)** virtual machine hosted within a controlled lab network. The testing focused on both **automated and manual techniques**, ensuring a comprehensive evaluation of application security posture.

2. Test Environment and Setup

• **Target Application:** DVWA

• **Target IP:** 192.168.175.129

• **Testing Scope:** Authentication and input-handling functionalities

• **Tools Used:**

o **Burp Suite** – Manual request interception and manipulation

o **sqlmap** – Automated SQL Injection testing

o **OWASP ZAP** – Passive and active vulnerability scanning

All testing activities were performed in a closed lab environment strictly for educational and ethical purposes.

3. Methodology

The assessment followed a structured web application testing methodology:

1. **Reconnaissance & Mapping**

o Identified application endpoints such as login pages and input forms.

o Mapped request/response flows using an intercepting proxy.

2. **Vulnerability Identification**

o Automated scans using OWASP ZAP to identify low-hanging vulnerabilities.

o Manual inspection of parameters and user inputs.

3. **Exploitation & Validation**

o sqlmap was used to confirm SQL injection flaws.

o Burp Suite was used to manually test XSS payload execution and session handling.

4. **Documentation**

o Findings were logged with severity ratings, affected URLs, and impact descriptions.

---

4. Vulnerability Findings

Test Log

Test ID	**Vulnerability**	**Severity**	**Target URL**
001	SQL Injection	Critical	http://192.168.75.128/login
002	Reflected XSS	Medium	http://192.168.75.129/form

4.2 Reflected Cross-Site Scripting (Medium)

A reflected XSS vulnerability was identified in a user input form where submitted data was reflected in the response without proper sanitization.

• **Tool Used:** Burp Suite (manual payload injection)

• **Impact:**

o Execution of malicious JavaScript in a victim's browser

o Potential session token theft

o Risk of phishing or user impersonation

Although less severe than SQL Injection, this flaw can be chained with social engineering attacks to compromise user accounts.

5. Manual Testing Insights

Manual testing using Burp Suite revealed weaknesses that automated tools alone may miss. By intercepting HTTP requests, parameters were modified to test:

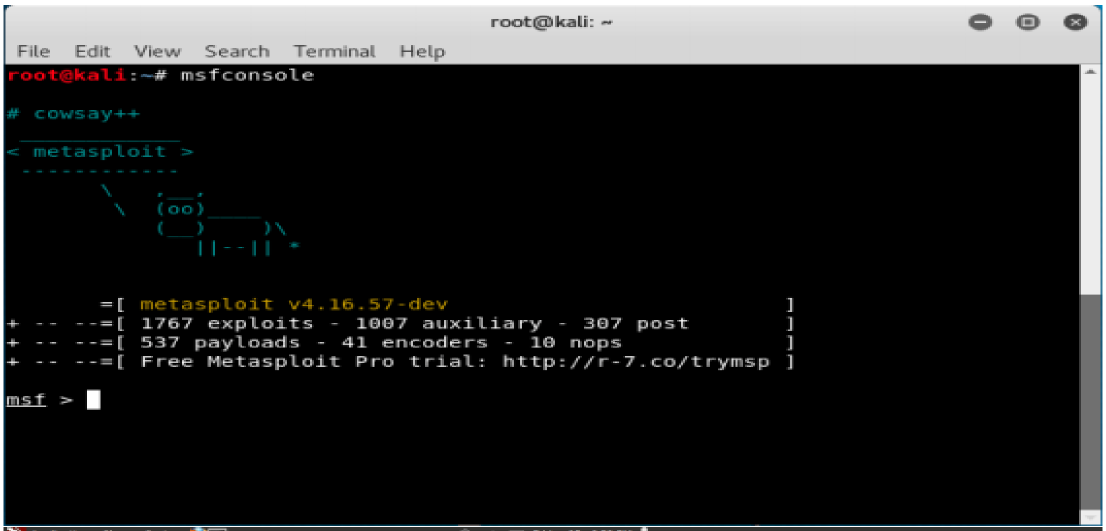
• Session token predictability

• Input validation weaknesses

• Authentication logic robustness

This step emphasized the importance of **human-driven testing** alongside automation.

Results:

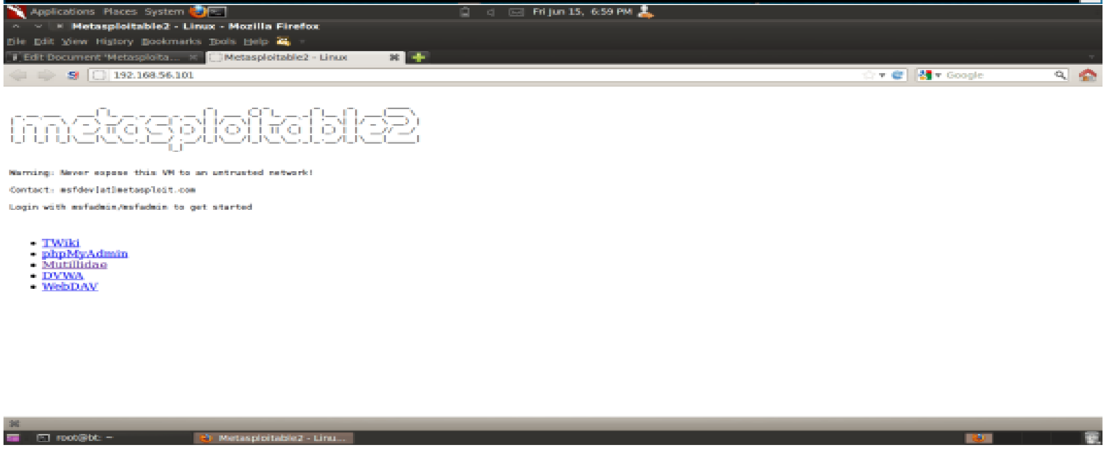


```
root@kali:~# msfconsole

# cowsay++
 ____________
< metasploit >
 ------------
       \   ,__,
        \  (oo)____
           (__)    )\
              ||--|| *

       =[ metasploit v4.16.57-dev                         ]
+ -- --=[ 1767 exploits - 1007 auxiliary - 307 post       ]
+ -- --=[ 537 payloads - 41 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > 
```


Applications Places System
Fri Jun 15, 6:59 PM
Metasploitable2 - Linux - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Edit Document 'Metasploita...
Metasploitable2 - Linux
192.168.56.101
Google
root@bt: ~
Metasploitable2 - Linu...

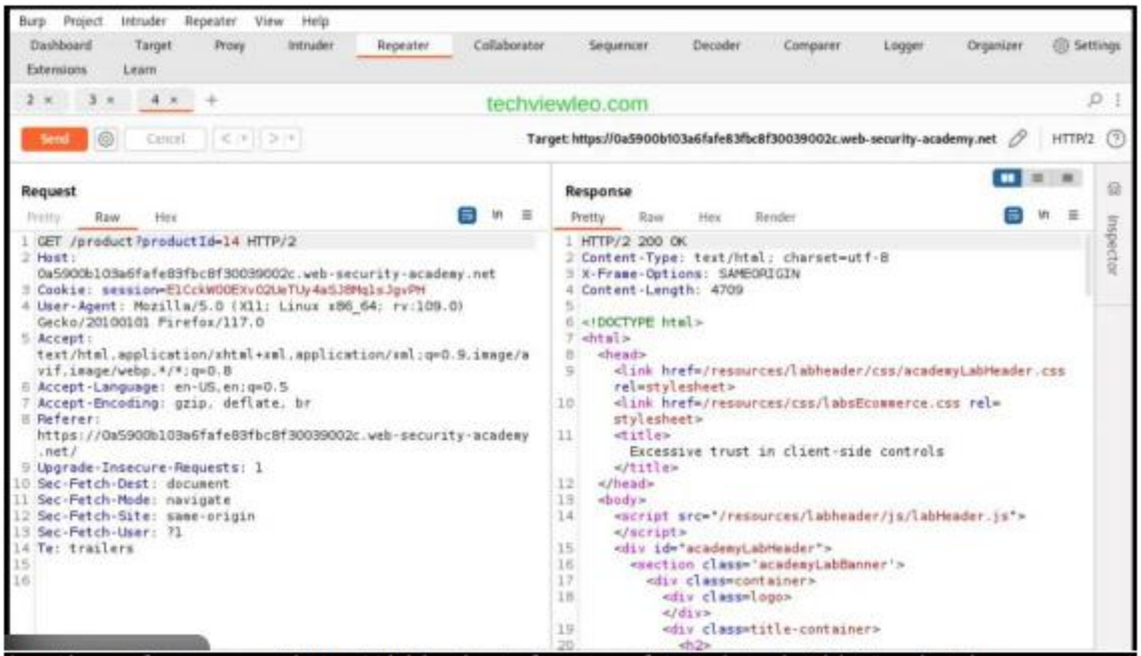
```
metasploitable2

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started
```

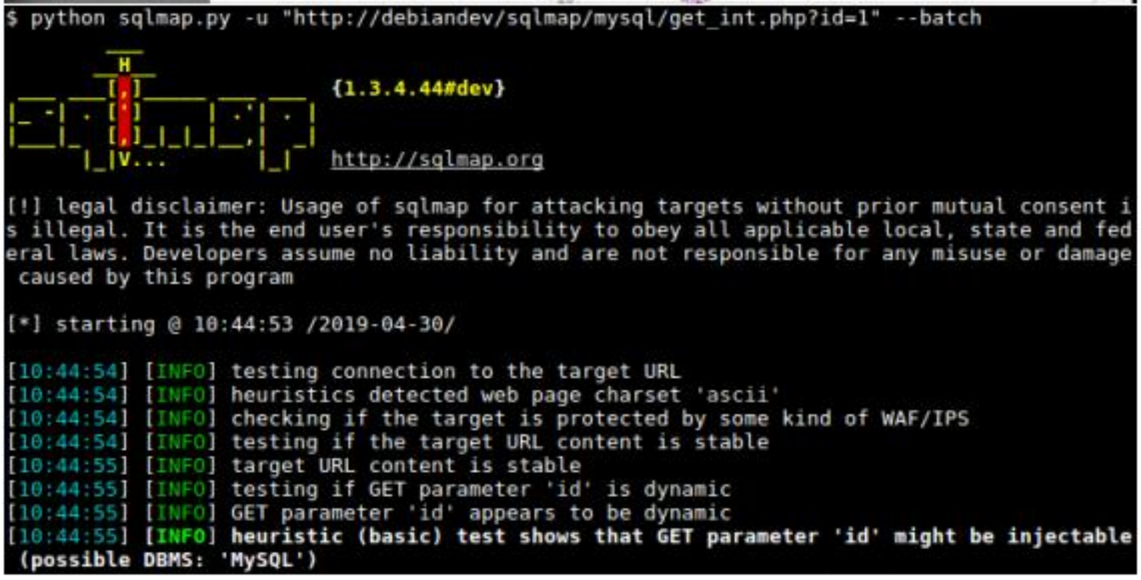
- TWiki
- phpMyAdmin
- Mutillidae
- DVWA
- WebDAV


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Settings
Extensions Learn
techviewleo.com
Send
Cancel
Target: https://0a5900b103a6fafe83fbc8f30039002c.web-security-academy.net
HTTP/2
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
Inspector

```
GET /product?productId=14 HTTP/2
Host:
0a5900b103a6fafe83fbc8f30039002c.web-security-academy.net
Cookie: session=E1CckWOOEXvO2UeTUy4aSJ8MqlsJgvPH
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/117.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer:
https://0a5900b103a6fafe83fbc8f30039002c.web-security-academy
.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 4709

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css
    rel=stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=
    stylesheet>
    <title>
      Excessive trust in client-side controls
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
```



```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.3.4.44#dev}
|_ -| . [']     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

6. Remediation Recommendations

- Implement prepared statements and parameterized queries.
- Apply strict input validation and output encoding.
- Use secure session management practices (HttpOnly, Secure flags).
- Regularly perform security testing during development cycles.

7. Conclusion

This lab successfully demonstrated how common OWASP Top 10 vulnerabilities manifest in real�world applications. By combining automated tools with manual testing techniques, critical security flaws were identified and validated. The exercise reinforced the importance of secure coding practices and continuous security testing in modern web application development.

8. Web Test Summary

The web application testing identified critical SQL Injection and medium-severity reflected XSS vulnerabilities in DVWA. Using Burp Suite, sqlmap, and OWASP ZAP, flaws in input validation and authentication were confirmed, highlighting the need for secure coding, proper sanitization, and regular security assessments.