

Web Application Vulnerability Testing

Introduction

Web application vulnerability testing is the process of identifying security weaknesses in web applications that can be exploited by attackers. This practical focuses on understanding common OWASP Top 10 vulnerabilities using a deliberately vulnerable application. Tools like DVWA and Burp Suite are used to analyze, exploit, and observe insecure application behavior. The goal is to gain hands-on experience in web application security testing and mitigation.

Tools Used

- Kali Linux
 - Burp Suite Community Edition
 - Damn Vulnerable Web Application (DVWA)
 - Mozilla Firefox
-

OWASP Top 10

OWASP Top 10 represents the most critical web application security risks identified by the Open Web Application Security Project (OWASP). These vulnerabilities highlight common weaknesses in modern web applications that attackers frequently exploit.

Some common OWASP Top 10 vulnerabilities include:

- SQL Injection (SQLi)
- Cross-Site Scripting (XSS)
- Broken Authentication
- Security Misconfiguration
- Sensitive Data Exposure

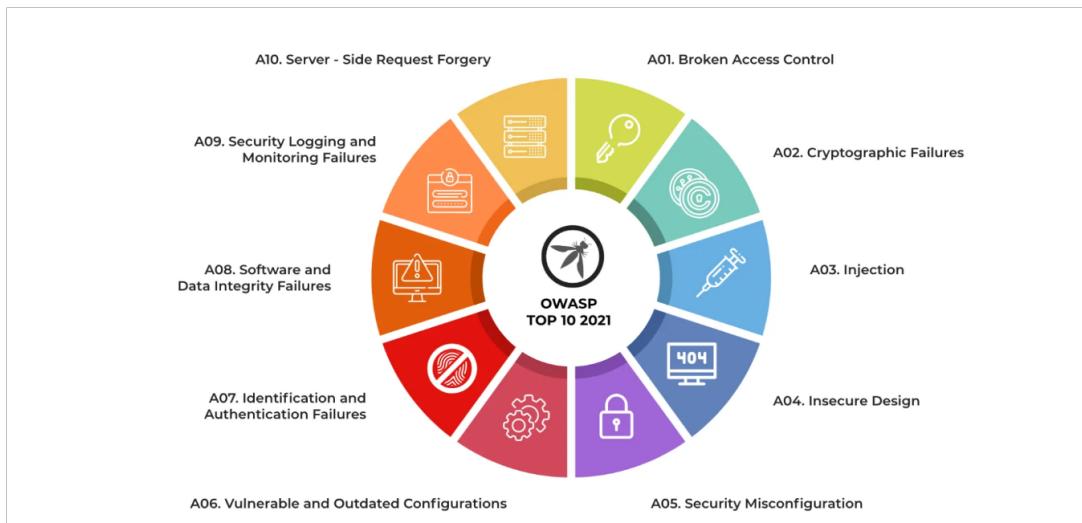


Figure 1: OWASP Top 10 Web Application Security Risks

Step 1: Setup Vulnerable Application (DVWA)

DVWA is installed and configured on Kali Linux using Apache and PHP. The application runs locally on the system.

URL: <http://localhost/dvwa>

```
(kali㉿kali)-[~/var/www/html]
└─$ cd dvwa/
└─$ ls
about.php  database  favicon.ico  logout.php  README.fa.md  README.md  README.zh.md  setup.php
CHANGELOG.md Dockerfile  hackable  phpinfo.php  README.fr.md  README.pl.md  robots.txt  tests
compose.yml  docs  index.php  php.ini  README.id.md  README.pt.md  SECURITY.md  vulnerability
config  dvwa  instructions.php  README.ar.md  README.it.md  README.tr.md  security.php
COPYING.txt  external  login.php  README.es.md  README.ko.md  README.vi.md  security.txt

(kali㉿kali)-[~/var/www/html/dvwa]
└─$ cd config/
└─$ ls
config.inc.php.dist

(kali㉿kali)-[~/var/www/html/dvwa/config]
└─$
```

Figure 2: DVWA Setup Page on Kali Linux

Step 2: Login to DVWA

The default credentials are used to log in to the DVWA dashboard.

Username: admin

Password: password



Figure 3: DVWA Login Page

Step 3: Configure Burp Suite Proxy

Burp Suite Community Edition is launched and the browser proxy is configured to intercept HTTP requests.

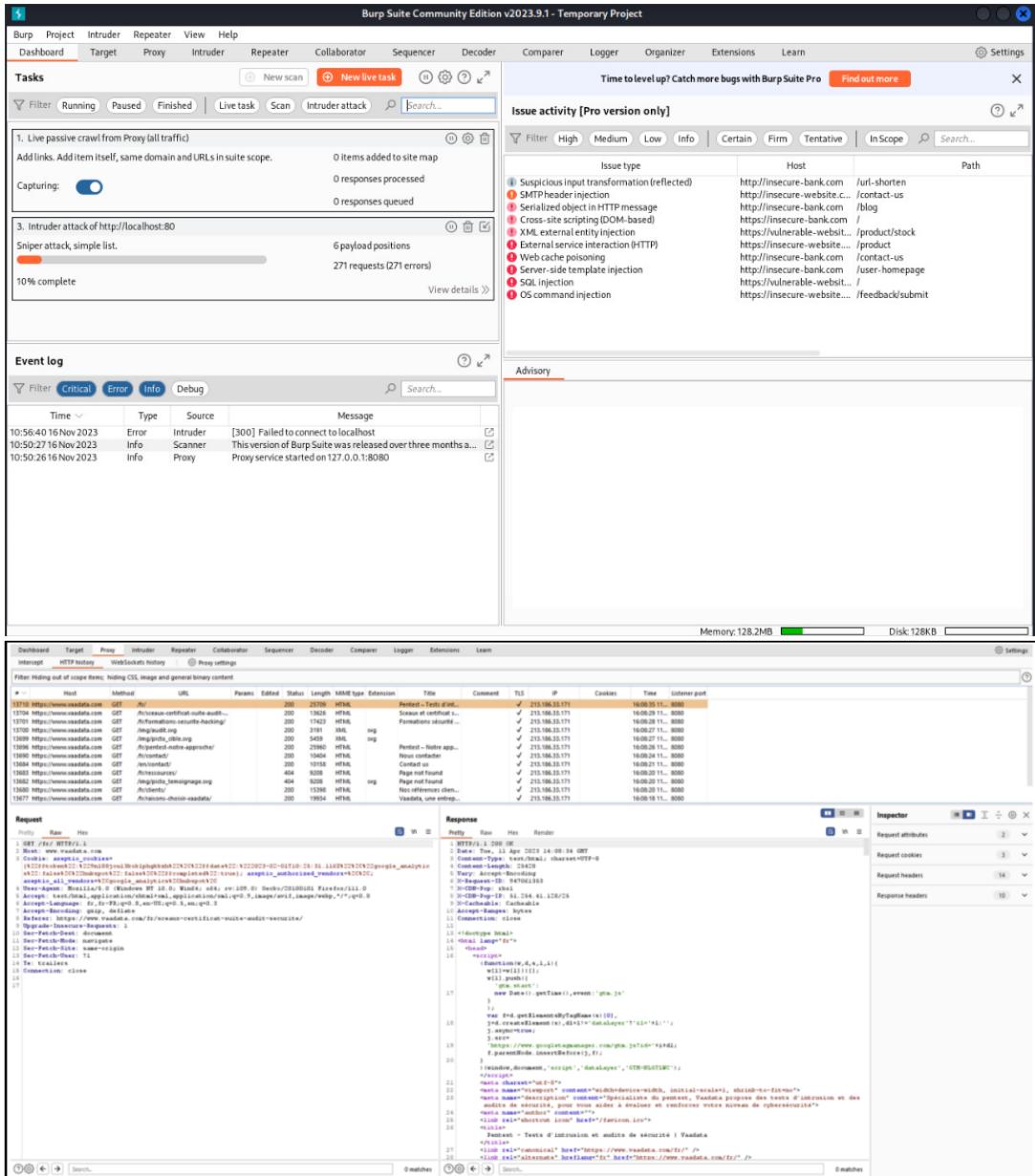


Figure 4: Burp Suite Proxy Intercept Enabled

Step 4: SQL Injection Testing

SQL Injection is tested using the DVWA SQL Injection module by inserting malicious SQL queries.

Sample Payload:

' OR '1'='1

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted in green), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. Below the sidebar, the status is shown as Username: admin, Security Level: low, and PHPIDS: disabled. At the bottom of the sidebar is a 'View Source' and 'View Help' link.

Vulnerability: SQL Injection

User ID: Submit

```

ID: admin ' OR 1=1--
First name: admin
Surname: admin

ID: admin ' OR 1=1--
First name: Gordon
Surname: Brown

ID: admin ' OR 1=1--
First name: Hack
Surname: Me

ID: admin ' OR 1=1--
First name: Pablo
Surname: Picasso

ID: admin ' OR 1=1--
First name: Bob
Surname: Smith
  
```

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_Injection
- <http://terruh.mavituna.com/sql-injection-cheatsheet-oku/>
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql_sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- <http://bobby-tables.com/>

Damn Vulnerable Web Application (DVWA) v1.10 "Development"

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq1 ttl=57 time=15.7 ms
64 bytes from 8.8.8.8: icmp_seq2 ttl=57 time=15.7 ms
64 bytes from 8.8.8.8: icmp_seq3 ttl=57 time=15.8 ms
64 bytes from 8.8.8.8: icmp_seq4 ttl=57 time=15.7 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 15.672/15.710/15.764/0.033 ms
  
```

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/in/>
- https://owasp.org/www-community/attacks/Command_Injection

Figure 5: SQL Injection Vulnerability in DVWA

Step 5: Cross-Site Scripting (XSS) Testing

XSS vulnerability is tested by injecting JavaScript code into user input fields.

Sample Payload:

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

The image shows two screenshots of the DVWA (Damn Vulnerable Web Application) interface, specifically the 'Reflected Cross Site Scripting (XSS)' section. Both screenshots show the same configuration and results, demonstrating a successful XSS exploit.

Screenshot 1 (Top): The user has injected the payload `<script>fetch("http://loc:3000")</script>` into the 'What's your name?' field. The 'Submit' button is visible. Below the form, under 'More Information', is a list of links related to XSS:

- [https://www_OWASP_org/index.php/Cross-site_Scripting_\(XSS\)](https://www_OWASP_org/index.php/Cross-site_Scripting_(XSS))
- https://www_OWASP_org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

Screenshot 2 (Bottom): The user has injected the payload `<script>alert(123456)</script>` into the 'What's your name?' field. The 'Submit' button is visible. Below the form, under 'More info', is a link to an external XSS example:

<http://ha.ckers.org/xss.html>

Both screenshots show the resulting output: a red alert box in the browser window displaying the message "Hello".

Figure 6: XSS Alert Box Triggered in Browser

Step 6: Observe Application Responses

The server responses are analyzed using Burp Suite to identify insecure behaviors and data leakage.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the 'Intercept' tab, a single request is listed:

Time	Type	Direction	Host	Method
09:42:32 3 Jul 2024	HTTP	→ Request	portswigger.net	GET

In the 'Request' pane, the raw HTTP request is displayed:

```
1 GET / HTTP/1.1
2 Host: portswigger.net
3 Cookie: stg_returning_visitor=Wed%2C%2022%20Nov%202023%2009:06:36%20GMT; t=HIRDfA0071UBE1C%2B180VA%3D%3D; AWSALBAPP-0=_remove_; AWSALBAPP-1=_remove_; AWSALBAPP-2=_remove_; AWSALBAPP-3=_remove_;
```

The 'HTTP history' tab shows a list of recent requests:

#	Host	Method	Params	URL	Edited	Status code	Length	MIME type	Extension	Title
46	https://portswigger.net	GET		/content/images/svg/arrow-youtube....		200	1901	XML	svg	
45	https://portswigger.net	GET		/content/images/patterns/dots-spac...		200	9553	XML	svg	
30	https://portswigger.net	GET		/content/images/logos/portswigger-r...		200	11444	XML	svg	
29	https://portswigger.net	GET		/content/images/logos/portswigger-...		200	7190	XML	svg	
27	https://portswigger.net	GET		/images/validate-your-certification.s...		200	35113	text	svg	
26	https://portswigger.net	GET		/images/research_email.svg		200	12301	text	svg	

The 'Response' pane shows the detailed response headers and body:

```
1 HTTP/2 200 OK
2 Date: Wed, 03 Jul 2024 08:49:07 GMT
3 Content-Type: image/svg+xml
4 Content-Length: 361
5 Server: Kestrel
6 Accept-Ranges: bytes
7 Cache-Control: must-revalidate, max-age=0
8 Etag: "1dacbb15796c669"
9 Last-Modified: Mon, 01 Jul 2024 12:22:30 GMT
10 Strict-Transport-Security: max-age=31536000; preload
11 X-Content-Type-Options: nosniff
12 X-Frame-Options: SAMEORIGIN
```

Figure 7: HTTP Response Analysis in Burp Suite

Step 7: Documentation of Vulnerabilities

Vulnerability	Payload Used	Impact
SQL Injection	' OR '1'='1	Unauthorized database access
Cross-Site Scripting	<script>alert()</script>	Execution of malicious scripts

Step 8: Suggested Mitigations

- Use prepared statements and parameterized queries

- Implement proper input validation and output encoding
 - Disable detailed error messages
 - Apply least privilege principle
 - Regular security testing and updates
-

Summary

This practical demonstrated how common web vulnerabilities such as SQL Injection and XSS can be identified using Burp Suite and DVWA. Proper security controls and secure coding practices are essential to protect web applications.
