

Project 1 : DVWA Security Level Comparison Project

Objective: The object of project is to Compare how DVWA behaves at different security levels (Low, Medium, High, Impossible)

Tasks: We have picked 2 vulnerabilities

Sql injections-When user input (e.g., from a login form or URL parameter) is directly concatenated into an SQL query, attackers may inject malicious SQL code that the database executes.

Using payload:- 'OR 1=1-- with low Security

The screenshot shows the DVWA SQL Injection (Low) page. On the left, a sidebar lists various attack types. The 'SQL Injection' option is highlighted in green. The main content area has a 'User ID:' input field containing '1 OR 1=1--'. A 'Submit' button is present. Below the input field, the page displays five sets of data, each showing an ID, first name, and surname. All IDs are '1 OR 1=1--', and all names are the same as the ones listed in the sidebar. At the bottom, the status bar shows 'Username: admin' and 'Security Level: Security Level: low'. There are 'View Source' and 'View Help' links at the bottom right.

User ID: Submit

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

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

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

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

ID: ' 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/>

Username: admin
Security Level: Security Level: low

[View Source](#) [View Help](#)

Using payload: - 'OR 1=1-- with medium Security

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

Filter Bypass

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript Attacks

Authorisation Bypass

Open HTTP Redirect

Cryptography

API

DVWA Security

PHP Info

About

Logout

User ID:

ID: 1
First name: admin
Surname: admin

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/>

Username: admin
Security Level: **Security Level: medium**

[View Source](#) [View History](#)

Using payload: - 'OR 1=1-- with High Security

The screenshot shows a web application interface for a security test. On the left, a sidebar lists various security vulnerabilities: Command Injection, 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 Attacks, Authorisation Bypass, Open HTTP Redirect, Cryptography, API, DVWA Security, PHP Info, About, and Logout.

The main content area displays the results of a SQL injection exploit. It shows four rows of data, each with a red background and white text:

ID	First name	Surname
ID: ' OR 1=1-- -	First name: Gordon	Surname: Brown
ID: ' OR 1=1-- -	First name: Hack	Surname: Me
ID: ' OR 1=1-- -	First name: Pablo	Surname: Picasso

Below this, another row of data is shown:

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

A modal window titled "SQL Injection Session Input :: Damn Vulnerable Web Application" is open. It contains a form with a URL field set to "http://127.0.0.1/DVWA/vulnerabilities/sqli/session-input.php#", a "Session ID" input field containing "' OR 1=1-- -", and a "Submit" button. Below the form is a "More Information" section with a list of links:

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netscout.com/research/papers/white_papers/white_papers_00000000000000000000000000000000.pdf
- https://owasp.org/www-project-top-ten/2017/A1-SQL_Injection
- <https://bobby-tables.com>

At the bottom of the main page, there are status messages: "Username: admin", "Security Level: Security Level: high" (highlighted in yellow), and "Locale: en". There are also "View Source" and "View Help" buttons.

Cross-site scripting: - Cross-Site Scripting (XSS) is a web security vulnerability that occurs when an application allows untrusted input to be included in web pages without proper validation or output encoding. This enables attackers to inject malicious scripts that are executed in the victim's browser.

Using payload: - <script>alert("XSS")</script> with low Security

The screenshot shows the DVWA application interface. On the left is a vertical menu bar with various security modules listed. The 'XSS (Reflected)' module is currently selected and highlighted in green. The main content area has a title 'Vulnerability: Reflected Cross Site Scripting (XSS)'. Below the title is a form field labeled 'What's your name?' containing the value 'Hello'. To the right of the field is a 'Submit' button. The DVWA logo is visible at the top right of the page.

- 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)
- XSS (Reflected)**
- XSS (Stored)
- CSP Bypass
- JavaScript Attacks
- Authorisation Bypass
- Open HTTP Redirect
- Cryptography
- API
- DVWA Security
- PHP Info
- About
- Logout

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

Hello

Using payload: - <script>alert("XSS")</script> with medium Security

Shiv Kumar

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)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript Attacks
Authorisation Bypass
Open HTTP Redirect
Cryptography
API
DVWA Security
PHP Info
About
Logout

What's your name? Submit

Hello alert("XSS")

More Information

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <https://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

Username: admin
Security Level: Security Level: medium

[View Source](#) [View Help](#)



Using payload: - <script>alert("XSS")</script> with High Security

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello >

More Information

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <https://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript Attacks

Authorisation Bypass

Open HTTP Redirect

Cryptography

API

DVWA Security

PHP Info

About

Logout

Username: admin
Security Level: Security Level: high
Locale: en
SQLi DB: mysql

[View Source](#) [View Help](#)

Outcome of above project task: -

This project provided valuable insight into how web vulnerabilities appear in real-world applications and how they can be mitigated through secure coding practices. Testing identical payloads across multiple security levels demonstrated that effective security does not rely on a single solution, but rather on the implementation of layered controls.

DVWA proved to be a useful environment for gaining hands-on experience with both attack techniques and defensive measures. Through this comparison, practical understanding of SQL Injection, Cross-Site Scripting (XSS), and secure application development was significantly enhanced.

Shiv Kumar

Project 2: Mini Pentest Project: DVWA + Another App

Objective: Is to Perform a controlled mini penetration test

Tasks: Set up DVWA and one more vulnerable app (Juice Shop, Mutillidae, bWAPP).

Vulnerable app Setup done- for bWAPP



Perform recon scanning using nmap on bWapp ip

```
(shiv㉿kali)-[~] bWAPP v2.2
└$ nmap -sV 127.0.0.1
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-17 20:00 IST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000090s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.7 ((Ubuntu))

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.57 seconds
```

Using Directory enumeration using DIRB

http://127.0.0.1:42001

```
(shiv㉿kali)-[~] $ dirb http://127.0.0.1:42001
-----
DIRB v2.22 / Portal /
By The Dark Raver
-----
bWAPP, or a buggy web application, is a free and open source deliberately insecure web
START_TIME: Wed Dec 17 21:07:42 2025 and students to discover and to prevent web vuln
URL_BASE: http://127.0.0.1:42001/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
----- Which bug do you want to hack today? -----
----- bWAPP v2.2 -----
GENERATED WORDS: 4612
    HTML Injection - Reflected (GET)
---- Scanning URL: http://127.0.0.1:42001/ ----
    HTML Injection - Reflected (Current URL)
(!) FATAL: Too many errors connecting to host
    (Possible cause: COULDNT_CONNECT)
    LDAP injection (Search)
    Mail Header Injection (SMTP)
-----
END_TIME: Wed Dec 17 21:07:42 2025
DOWNLOADED: 0 - FOUND: 0
```

Using nikto -h 127.0.0.1

```
(shiv㉿kali)-[~] $ nikto -h 127.0.0.1
- Nikto v2.5.0
-----
+ Target IP: 127.0.0.1
+ Target Hostname: 127.0.0.1
+ Target Port: 80
+ Start Time: 2025-12-17 21:15:59 (GMT5.5)
----- [MISSING]
+ Server: Apache/2.4.7 (Ubuntu)
+/: Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.14.
+/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+/: 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-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page / redirects to: portal.php
+ /passwords/: Directory indexing found.
+ /robots.txt: Entry '/passwords/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /documents/: Directory indexing found.
+ /robots.txt: Entry '/documents/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /images/: Directory indexing found.
+ /robots.txt: Entry '/images/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /robots.txt: Entry '/admin/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /robots.txt: contains 5 entries which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /login.php: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /web.config: ASP config file is accessible.
+ //etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
```

```
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /admin/: This might be interesting.
+ /apps/: Directory indexing found.
+ /apps/: This might be interesting.
+ /db/: Directory indexing found.
+ /db/: This might be interesting.
+ /passwords/: This might be interesting.
+ /stylesheets/: Directory indexing found.
+ /stylesheets/: This might be interesting.
+ /admin/index.php: This might be interesting: has been seen in web logs from an unknown scanner.
+ /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information. See: CWE-552
+ /admin/phpinfo.php: Output from the phpinfo() function was found.
+ /admin/phpinfo.php: Immobilier allows phpinfo() to be run. See: https://vulners.com/osvdb/OSVDB:35877
+ /config.inc: DotBr 0.1 configuration file includes usernames and passwords. See: OSVDB-5092
+ /update.php: Cookie admin created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /update.php: Cookie movie_genre created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /update.php: Cookie secret created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /update.php: Cookie top_security created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /update.php: Cookie top_security_nossal created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /update.php: Cookie top_security_ssl created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /install.php: install.php file found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /login.php: Admin login page/section found.
+ /test.php: This might be interesting.
+ /wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wp-includes/js/tinymce/themes/modern/MeuhY.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-includes/js/tinymce/themes/modern/MeuhY.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /assets/mobirise/css/meta.php?filesrc=: A PHP backdoor file manager was found.
+ /login.cgi?cli=aa%20aa%27cat%20/etc/hosts: Some D-Link router remote command execution.
+ /shell?cat+/etc/hosts: A backdoor was identified.
+ 8885 requests: 0 error(s) and 49 item(s) reported on remote host
+ End Time: 2025-12-17 21:16:29 (GMT5.5) (30 seconds)
-----
+ 1 host(s) tested
```

Finding of Vulnerability using DVWA

SQL Injection

Using payload: - 'OR 1=1-- with low Security

The screenshot shows the DVWA application interface. On the left, a sidebar lists various attack types: Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insertion, SQL Injection (highlighted in green), SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript Attacks, Authorisation Bypass, Open HTTP Redirect, Cryptography, API, DVWA Security, PHP Info, About, and Logout. At the bottom of the sidebar, it says 'Username: admin' and 'Security Level: Security Level: low'. The main content area has a 'User ID:' input field and a 'Submit' button. Below the input field, the results of the SQL injection are displayed in red text:
ID: ' OR 1=1--
First name: admin
Surname: admin

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

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

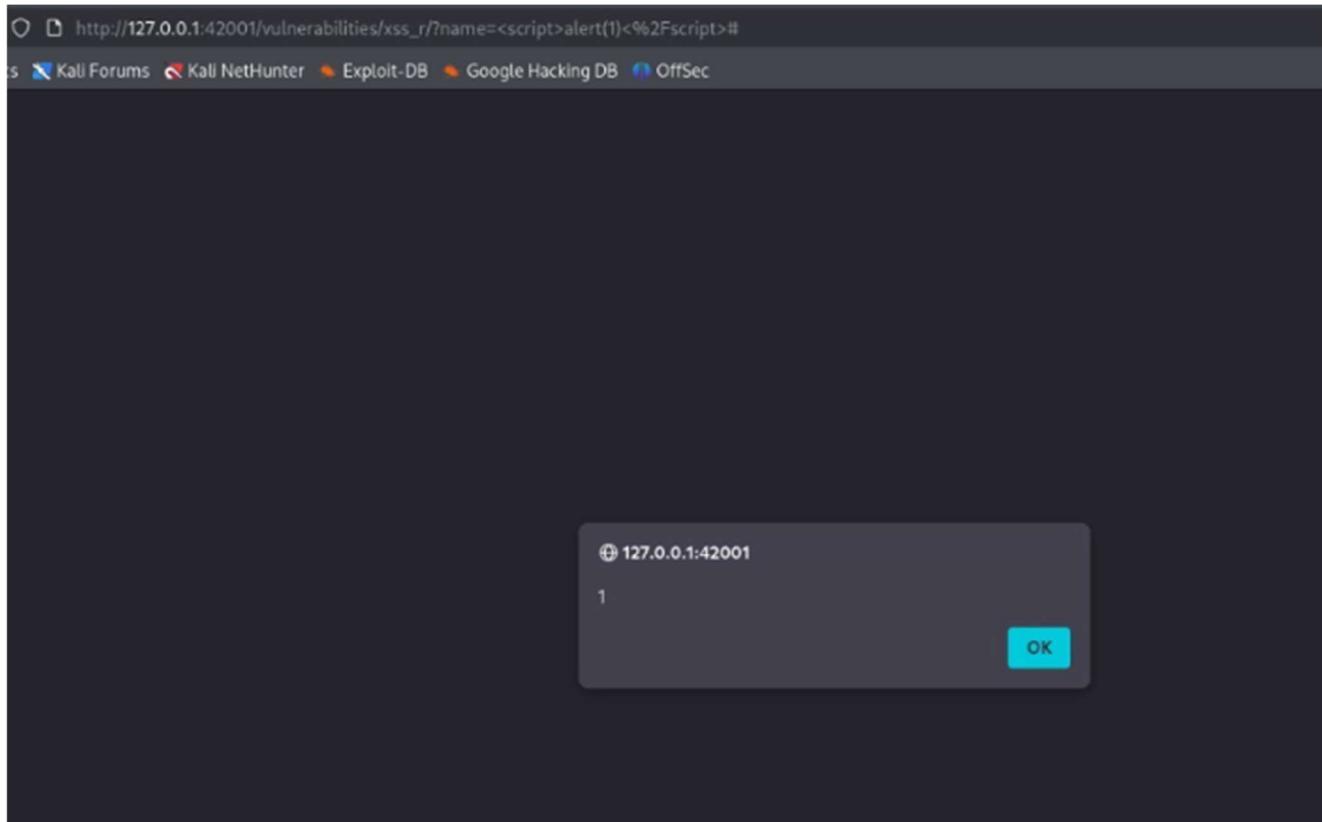
More Information

- 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/>

[View Source](#) [View Help](#)

Cross-Site Scripting

```
<script>alert("1")</script>
```



Executive Summary

This mini penetration testing exercise evaluated two deliberately vulnerable web applications, DVWA and bWAPP, within a controlled laboratory environment. The assessment focused on identifying common and critical web application vulnerabilities, including SQL Injection and Cross-Site Scripting (XSS).

Testing confirmed the presence of Cross-Site Scripting vulnerabilities in DVWA, demonstrating the security risks associated with improper input validation and output handling. These findings highlight how easily client-side attacks can be introduced when secure coding controls are insufficient.

For bWAPP, several high-risk security weaknesses were discovered during reconnaissance and vulnerability scanning activities. These included unnecessary service exposure and the use of outdated software components. Overall, the results emphasize the need for strong secure development practices, system hardening, and continuous security testing to minimize potential attack vectors.

Remediations

- Utilize parameterized queries and prepared statements to ensure safe interaction with databases
- Enforce strict validation and sanitization for all user-provided input
- Apply appropriate output encoding techniques to mitigate Cross-Site Scripting (XSS) risks
- Strengthen authentication mechanisms and securely manage user sessions
- Remove or disable unnecessary services and restrict access to unused network ports
- Keep all software components up to date by applying patches and updates regularly
- Align development and security practices with the OWASP Top 10 guidelines

Outcome of above project task: -

This project offered hands-on experience with penetration testing methods within a simulated real-world laboratory setup. By evaluating applications hosted across multiple platforms, it enhanced understanding of web application security issues as well as network-level attack vectors.

Integrating manual assessment techniques with automated security tools helped develop stronger technical expertise, improved problem-solving abilities, and strengthened skills in producing clear and professional security reports.

Project 3: Automated vs. Manual Testing Project

Objective: Analyze differences between automated scanners and manual testing

Tasks: Run Nikto, OWASP ZAP, sqlmap on DVWA.

Using Nikto Scan on DVWA

```
shiv@kali:~$ nikto -h 127.0.0.1
- Nikto v2.5.0
+ Target IP: 127.0.0.1
+ Target Hostname: 127.0.0.1
+ Target Port: 80
+ Start Time: 2025-12-18 11:42:00 (GMT5.5)

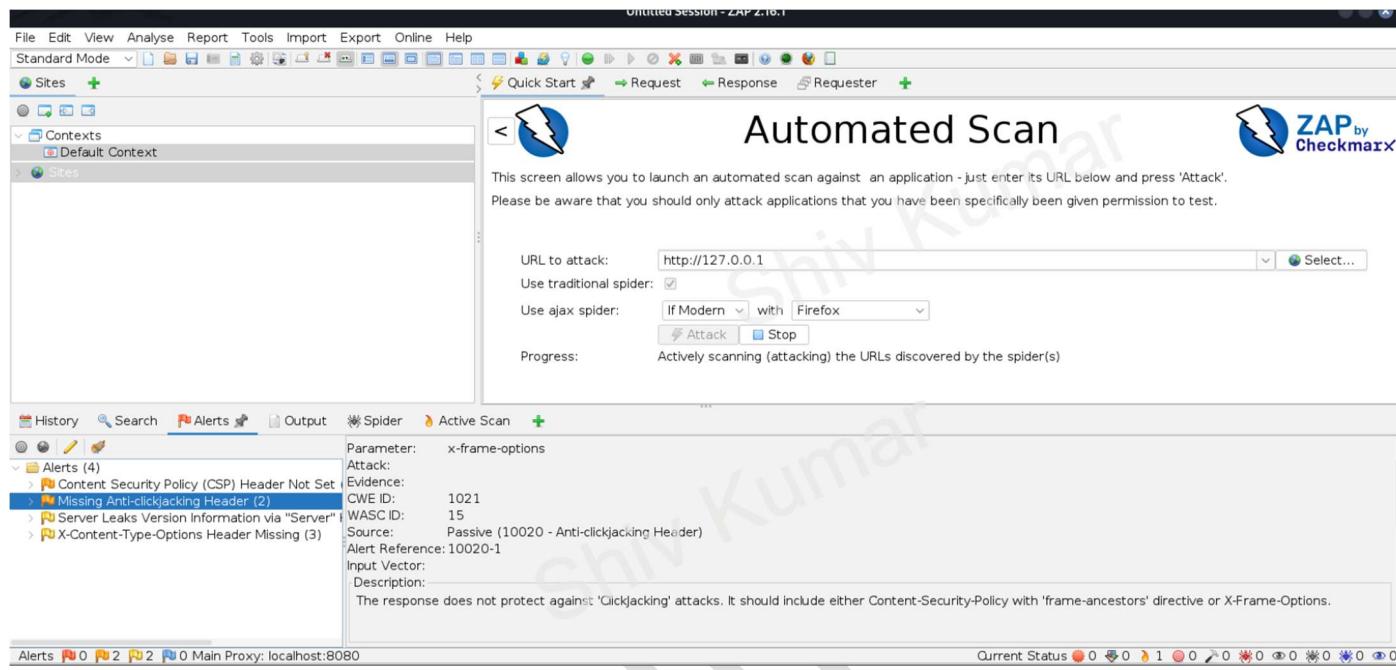
+ Server: Apache/2.4.65 (Debian)
+ : The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /: Server may leak inodes via ETags, header found with file /, inode: 29cf, size: 643cd0b9be8af, mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ OPTIONS: Allowed HTTP Methods: POST, OPTIONS, HEAD, GET .
+ //etc/hosts: The server install allows reading of any system file by adding an extra '/' to the URL.
+ /server-status: This reveals Apache information. Comment out appropriate line in the Apache conf file or restrict access to allowed sources. See: OSVDB-561
+ /wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-content/themes/twentyeleven/images/headers/server.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-includes/Requests/Utility/content-post.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wp-includes/js/tiny_mce/themes/modern/Meuh_y.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /wordpress/wp-includes/is/tinymce/themes/modern/Meuh_y.php?filesrc=/etc/hosts: A PHP backdoor file manager was found.
+ /assets/mobirise/css/meta.php?filesrc=: A PHP backdoor file manager was found.
+ /login.cgi?cli=aa%20aa%27cat%20/etc/: Some D-Link router remote command execution.
+ /shell?cat*/etc/: A backdoor was identified.
+ 8074 requests: 0 error(s) and 15 item(s) reported on remote host
+ End Time: 2025-12-18 11:42:33 (GMT5.5) (33 seconds)

+ 1 host(s) tested
*****
```

Using zap on DVWA

The screenshot shows the OWASP ZAP 2.16.1 interface. The main window title is "Untitled Session - ZAP 2.16.1". The top menu bar includes File, Edit, View, Analyse, Report, Tools, Import, Export, Online, and Help. The toolbar contains various icons for file operations, analysis, and reporting. On the left, there's a sidebar with tabs for "Sites", "Contexts", and "Spiders". The "Default Context" is selected under "Contexts". The main central area has a title "Automated Scan" with a lightning bolt icon. Below it, a message says: "This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'! Please be aware that you should only attack applications that you have been specifically given permission to test." There are input fields for "URL to attack" (set to "http://127.0.0.1") and "Spider Type" (set to "If Modern with Firefox"). Buttons for "Attack" and "Stop" are present. A progress bar at the bottom indicates "Actively scanning (attacking) the URLs discovered by the spider(s)". At the bottom of the interface, there's a summary of the current session: "New Scan Progress: 0: http://127.0.0.1", "Current Scans: 1", "Num Requests: 279", "New Alerts: 0", and "Export". Below this, a table titled "Sent Messages" lists network traffic details such as ID, Req. Timestamp, Resp. Timestamp, Method, URL, Code, Reason, RTT, Size, and Response Headers.

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size	Resp. Header	Size	Resp. Body
332	18/12/25, 11:51:46 am	18/12/25, 11:51:46 am	GET	http://127.0.0.1/usr/share/doc	404	Not Found	7 ms	161 bytes		271 bytes	
333	18/12/25, 11:51:46 am	18/12/25, 11:51:46 am	GET	http://127.0.0.1/usr/share/doc/apache2	404	Not Found	7 ms	161 bytes		271 bytes	
334	18/12/25, 11:51:46 am	18/12/25, 11:51:46 am	GET	http://127.0.0.1/var	404	Not Found	6 ms	161 bytes		271 bytes	
335	18/12/25, 11:51:46 am	18/12/25, 11:51:46 am	GET	http://127.0.0.1/var/www	404	Not Found	13 ms	161 bytes		271 bytes	
336	18/12/25, 11:51:46 am	18/12/25, 11:51:46 am	GET	http://127.0.0.1/var/www/html	404	Not Found	8 ms	161 bytes		271 bytes	
337	18/12/25, 11:51:46 am	18/12/25, 11:51:46 am	GET	http://127.0.0.1/etc	404	Not Found	38 ms	161 bytes		271 bytes	
338	18/12/25, 11:51:47 am	18/12/25, 11:51:47 am	GET	http://127.0.0.1/etc/apache2	404	Not Found	10 ms	161 bytes		271 bytes	



Using SQLMap on DVWA

```
(shiv@kali)-[~]
$ sqlmap -u "127.0.0.1" --cookie="security=low";
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 12:13:38 /2025-12-18/
[12:13:39] [INFO] testing connection to the target URL
[12:13:39] [INFO] checking if the target is protected by some kind of WAF/IPS
[12:13:39] [INFO] testing if the target URL content is stable
[12:13:39] [INFO] target URL content is stable
[12:13:39] [CRITICAL] no parameter(s) found for testing in the provided data (e.g. GET parameter 'id' in 'www.site.com/index.php?id=1'). You are advised to rerun with '--crawl=2'

[*] ending @ 12:13:39 /2025-12-18/
(shiv@kali)-[~]
```

Manually testing of the pages using Burp Suite

Using Burp suite on DVWA IP

The screenshot shows the Burp Suite interface with the following details:

- Header Bar:** Burp Project Intruder Repeater View Help | Burp Suite Community Edition v2025.10.7 - Temporary Project
- Toolbar:** Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
- Sub-Toolbar:** Intercept HTTP history WebSockets history Match and replace | Proxy settings
- Request Bar:** Intercept on → Forward | Drop Request to http://127.0.0.1:42001 | Open browser | ?
- Table Headers:** Time Type Direction Method URL Status code Len
- Table Data:** 15:30:00 ... HTTP → Request GET http://127.0.0.1:42001/vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit=Submit
- Request Panel (Left):** Shows a raw request message:

```
1 GET /vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit=Submit HTTP/1.1
2 Host: 127.0.0.1:42001
3 sec-ch-ua: "Chromium";v="143", "Not A(Brand";v="24"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Accept-Language: en-GB,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0
9 Safari/537.36
10 Accept:
11 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: http://127.0.0.1:42001/vulnerabilities/sqli/
17 Accept-Encoding: gzip, deflate, br
18 Cookie: PHPSESSID=0e178d564e7d48abf8ebb8ef9a16b886; security=low
19 Connection: keep-alive
```
- Inspector Panel (Right):** Shows request attributes and query parameters:

Name	Value
Protocol	HTTP/1
Method	GET
Path	/vulnerabilities/sqli/

Name	Value
id	' OR '1'='1
Submit	Submit

Compare: Which issues automation caught, which only manual testing revealed.

Automated scanning tools rapidly detected well-known security weaknesses, including absent security headers, SQL injection points, and basic cross-site scripting issues. These tools improved efficiency by reducing assessment time and covering a large portion of the application. In contrast, manual security testing provided a more in-depth analysis, enabling accurate payload creation and confirmation of whether vulnerabilities could actually be exploited. Certain problems, such as business logic errors and manipulation of parameters, were more effectively uncovered through hands-on testing. While automated solutions occasionally generated inaccurate or misleading results, manual testing demanded greater expertise and significantly more time.

The “Strengths and Weaknesses of Automated Scanners.”

Advantages:

- Perform scans quickly with minimal effort
- Examine a wide range of application components
- Helpful during the early stages of security assessment
- User-friendly, even for those with limited experience

Limitations:

- May report issues that are not actual vulnerabilities
- Lack insight into business logic and workflow behavior
- Unable to completely confirm the real-world impact of exploits
- Often fail to detect advanced or multi-step security flaws

Project 4: Build Your Own Vulnerable Web App Module

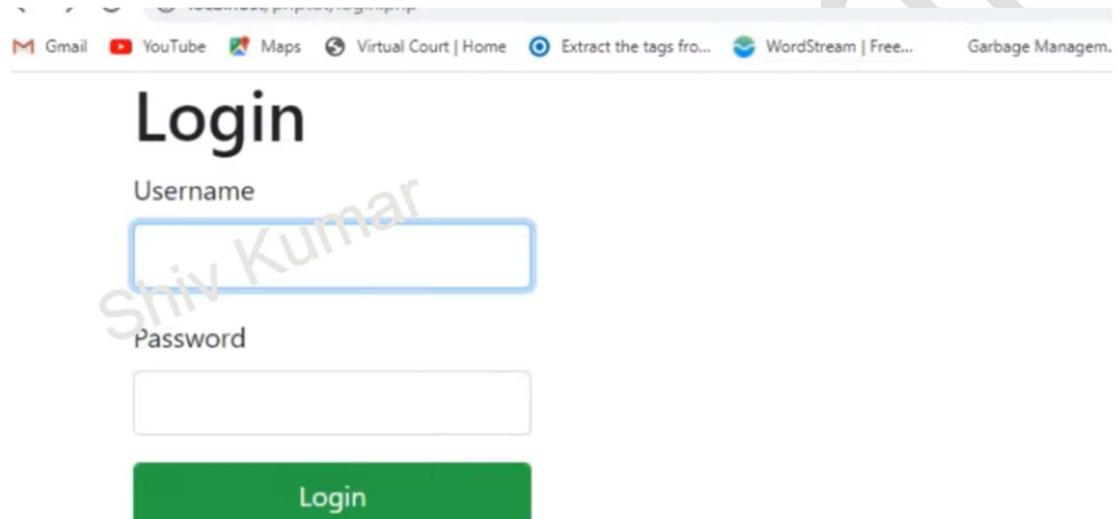
Objective: Understand insecure coding practices.

Tasks: Create a simple PHP page vulnerable to one attack (SQLi or XSS).

We have created: A login page without input validation.

We are using Vulnerable PHP

File name using= login.php



Vulnerable PHP Code

PHP File

```
1 <?php
2     include 'conn.php';
3     if(isset($_POST['login'])){
4
5         $user = $_POST['user'];
6         $pass = $_POST['pass'];
7
8         $sql = "SELECT username,password FROM register WHERE username='$user' and password = '$pass'";
9         $check = mysqli_query($conn,$sql);
10        if(mysqli_num_rows($check)>0){
11            echo "Login success";
12        }else{
13            echo "Incorrect details";
14        }
15    }
16 ?>
```

We are providing payload 'or '1' ='1

The screenshot shows a web browser window with a 'Login' form. The 'Username' field contains the payload "' or '1' ='1". The 'Password' field contains a series of asterisks ('*****'). A green 'Login' button is at the bottom. The browser's address bar shows the URL 'Virtual Court | Home'. The page title is 'Login'.

Once we click on login button the result is, we are able to login by using above payload

Gmail YouTube Maps Virtual Court | Home Extract the tags from... WordStream | Free... Garbage Managem... My Drive - Google... if-else vs switch - ja... Search
SELECT username,password FROM register WHERE username=" or '1' ='1' and password = " or '1' ='1'Login success

Now we are applying fix for above vulnerability

Solution: -We are using escape string in our code to fix the vulnerability

```
1 <?php
2     include 'conn.php';
3     if(isset($_POST['login'])){
4
5         $user = mysqli_real_escape_string($_POST['user']);
6         $pass = mysqli_real_escape_string($_POST['pass']);
7
8         $sql = "SELECT username,password FROM register WHERE username='$user' and
9               password = '$pass' ";
10        $check = mysqli_query($conn,$sql);
11        if(mysqli_num_rows($check)>0){
12            echo "Login success";
13        }else{
14            echo "Incorrect details";
15        }
16    ?>
```

We are trying payload again to login-

[YouTube](#) [Maps](#) [Virtual Court | Home](#) [Extract the tags from...](#) [Word2red](#)

Login

Username

' or '1'='1

Password

.....

Once we click on login button the result is, we are now not able to login by using above payload

SELECT username,password FROM register WHERE username='\ or \'1\' = 1\' and password = '\ or \'1\' = 1\' Incorrect details

Conclusion from above project activity

This project illustrated how poor coding practices can lead to serious security weaknesses in web applications. It was shown that even a basic login page can be exposed to SQL Injection attacks when user input is not handled correctly.

However, after applying prepared statements, the vulnerability was fully eliminated. This demonstrates the critical role of integrating secure coding techniques throughout the development process, rather than depending only on security testing after deployment.

Project 5: Research Project: Common Vulnerabilities Across Real Web Apps

Objective: Link DVWA learning to real-world CVEs.

Tasks: Pick one DVWA vulnerability (File Upload).

Real World Scenario Case 1: CVE-2018-7600 — Drupal File Upload Leading to Remote Code Execution

Description of above CVE:- CVE-2018-7600 is a real-world vulnerability in the **Drupal CMS** where improper handling of user input and file uploads allowed attackers to execute malicious code on the server.

This is very similar to the **DVWA File Upload vulnerability**, where the application does not properly restrict uploaded files.

How Attackers Exploited It

- Drupal allowed attackers to upload malicious files through a vulnerable form.
- The uploaded file contained executable code (such as a PHP script).
- Once uploaded, the attacker accessed the file through the browser.
- The server executed the malicious file, giving the attacker control.

The screenshot shows a web browser displaying information about CVE-2018-7600. The URL is https://www.cvedetails.com/cve/CVE-2018-7600/. The page title is "Metasploit modules for CVE-2018-7600".

Drupal Drupalgeddon 2 Forms API Property Injection

Disclosure Date: 2018-03-28 First seen: 2020-04-26

exploit/unix/webapp/drupal_drupalgeddon2

This module exploits a Drupal property injection in the Forms API. Drupal 6.x, < 7.58, 8.2.x, < 8.3.9, < 8.4.6, and < 8.5.1 are vulnerable. Authors: - Jasper Mattsson - a2u - Nixawk - FireFart - wvu <wvu@metasploit.com>

[More information](#)

CVSS scores for CVE-2018-7600

Base Score	Base Severity	CVSS Vector	Exploitability Score	Impact Score	Score Source	First Seen
7.5	HIGH	AV:N/AC:L/Au:N/C:P/I:P/A:P	10.0	6.4	NIST	
9.8	CRITICAL	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	3.9	5.9	NIST	
9.8	CRITICAL	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	3.9	5.9	134c704f-9b21-4f2e-91b3-4a467353bcc0	2025-02-07
9.8	CRITICAL	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	3.9	5.9	NIST	2025-01-27

 Detected Vulns
 IP Search
▼ Other
 Metasploit Modules
 CWE Definitions
 CAPEC Definitions
 Articles
 Blog

CVE-2018-7600 is in the CISA Known Exploited Vulnerabilities Catalog

 This issue is known to have been leveraged as part of a ransomware campaign.

CISA vulnerability name:
Drupal Core Remote Code Execution Vulnerability

CISA required action:
Apply updates per vendor instructions.

CISA description:
Drupal Core contains a remote code execution vulnerability that could allow an attacker to exploit multiple attack vectors on a Drupal site, resulting in complete site compromise.

Notes:
<https://nvd.nist.gov/vuln/detail/CVE-2018-7600>

Added on 2021-11-03 Action due date 2022-05-03

Exploit prediction scoring system (EPSS) score for CVE-2018-7600

[EPSS FAQ](#)

 94.49% Probability of exploitation activity in the next 30 days [EPSS Score History](#)

 ~ 100 % Percentile, the proportion of vulnerabilities that are scored at or less

Impact

- Attackers gained **remote code execution**.
- Full compromise of affected websites.
- Data theft, website defacement, and malware installation.

How It Was Fixed

- Drupal released a security patch.
- File uploads were restricted to safe file types only.
- Better server-side validation was added.
- Uploaded files were prevented from being executed.

Real world scenario case-2: CVE-2017-12615 — Apache Tomcat File Upload Vulnerability

Description: - CVE-2017-12615 is a vulnerability in **Apache Tomcat** where improper handling of uploaded files allowed attackers to upload and execute malicious **JSP files**.

This is very similar to DVWA's File Upload issue, where file type validation is missing or weak.

Both **DVWA File Upload** and **CVE-2017-12615 (Apache Tomcat)** suffer from the same core security issue: **improper handling of uploaded files**.

DVWA File Upload

- DVWA allows users to upload files without properly checking the file type.
- An attacker can upload a malicious file (e.g., shell.php).
- When the uploaded file is accessed, the server executes it.
- This gives the attacker control over the application.

CVE-2017-12615 (Apache Tomcat)

- Apache Tomcat allowed attackers to upload malicious JSP files.
- The server did not correctly restrict file uploads or execution.
- When the uploaded JSP file was accessed, it executed on the server.
- This resulted in remote code execution.

How Attackers Exploited It

- Tomcat allowed file uploads using the HTTP PUT method.
- Attackers uploaded a malicious file such as shell.jsp.
- Once uploaded, the attacker accessed the file through a browser.
- The server executed the malicious JSP file.

CVE-2017-12615 is in the CISA Known Exploited Vulnerabilities Catalog

 This issue is known to have been leveraged as part of a ransomware campaign.

CISA vulnerability name:

Apache Tomcat on Windows Remote Code Execution Vulnerability

CISA required action:

Apply updates per vendor instructions.

CISA description:

When running Apache Tomcat on Windows with HTTP PUTs enabled, it is possible to upload a JSP file to the server via a specially crafted request. This JSP could then be requested and any code it contained would be executed by the server.

Notes:

<https://nvd.nist.gov/vuln/detail/CVE-2017-12615>

Added on 2022-03-25 Action due date 2022-04-15

Exploit prediction scoring system (EPSS) score for CVE-2017-12615

[EPSS FAQ](#)

94.34%

Probability of exploitation activity in the next 30 days

[EPSS Score History](#)

~ 100
%

Percentile, the proportion of vulnerabilities that are scored at or less

Exploit prediction scoring system (EPSS) score for CVE-2017-12615

94.34% Probability of exploitation activity in the next 30 days [EPSS Score History](#)

~ 100 % Percentile, the proportion of vulnerabilities that are scored at or less

CVSS scores for CVE-2017-12615

Base Score	Base Severity	CVSS Vector	Exploitability Score	Impact Score	Score Source	First Seen
6.8	MEDIUM	AV:N/AC:M/Au:N/C:P/I:A:P	8.6	6.4	NIST	
8.1	HIGH	CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	2.2	5.9	NIST	
8.1	HIGH	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	2.2	5.9	134c704f-9b21-4f2e-91b3-4a467353bcc0	2025-02-06
8.1	HIGH	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	2.2	5.9	NIST	2024-07-16

Impact

- Remote Code Execution (RCE).
- Full control over the web server.
- Website defacement, data theft, or malware installation.

How It Was Fixed

- Apache released a security update.
- File upload permissions were restricted.
- Execution of uploaded files was disabled.
- Unsafe HTTP methods were blocked.

Conclusion from above project activity

The findings of this study show that the security flaws explored through DVWA closely mirror those found in real-world web applications. In particular, file upload weaknesses have been responsible for serious security events, such as unauthorized server access and exposure of sensitive data.

Practicing these attack scenarios in a controlled laboratory setting allows security practitioners to better identify potential risks and apply effective defenses before similar exploits can occur in production environments.