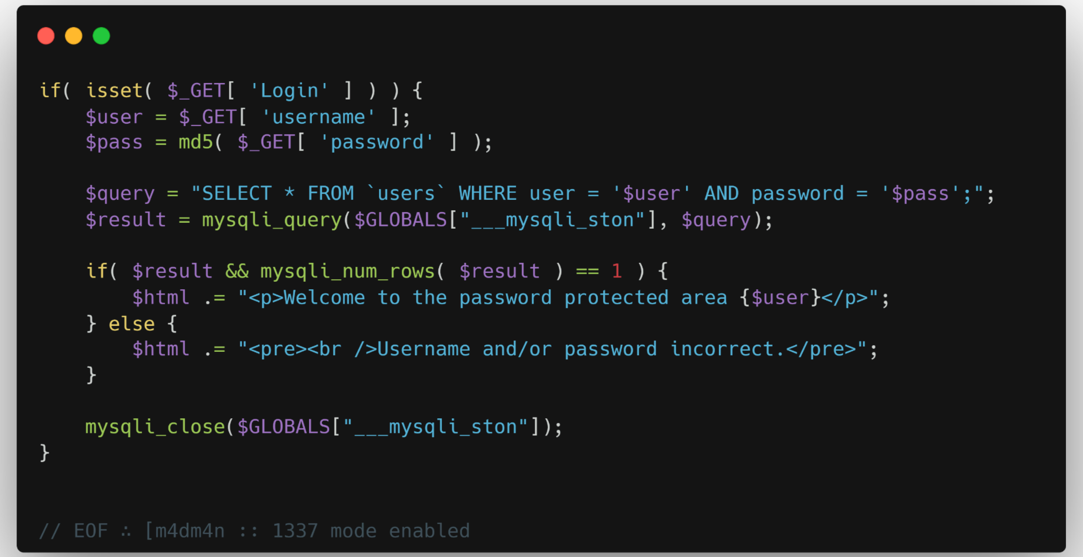
DVWA Brute Force (Low Security Level)

📌 Overview

In this lab, we analyze the insecure implementation of the login mechanism in Damn Vulnerable Web Application (DVWA) with the lowest security setting. We perform a full-stack vulnerability analysis, design a brute-force attack scenario, and conclude with security best practices for real-world defense

Source Code Analysis: Vulnerability Discovery



```php
if( isset( $_GET[ 'Login' ] ) ) {
    $user = $_GET[ 'username' ];
    $pass = md5( $_GET[ 'password' ] );

    $query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"], $query);

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        $html .= "<p>Welcome to the password protected area {$user}</p>";
    } else {
        $html .= "<pre><br />Username and/or password incorrect.</pre>";
    }

    mysqli_close($GLOBALS["___mysqli_ston"]);
}


// EOF ∴ [m4dm4n :: 1337 mode enabled
```

🔍 **Vulnerabilities Identified:**

🔒 Vulnerability	❌ Problem Description
GET Method for Login	Exposes credentials in URL, browser history, logs, referrer headers
No Rate Limiting or Lockout	Unlimited login attempts allowed
No Anti-CSRF Token	Allows CSRF-style credential stuffing
Weak MD5 Hashing	Fast, unsalted, precomputed hashes vulnerable to brute-force and rainbow tables
SQL Injection Risk	Direct string interpolation in SQL without proper parameterization
Feedback-Oriented Brute Force	Clear text reveals login success/failure, aiding automation
No Session Validation	No check for authenticated users, CSRF mitigation, or session hijacking defenses

---

2️⃣Exploitation Phase: Automating Brute-Force 🔨 Tool: Burp Suite Intruder
🔧 Setup:

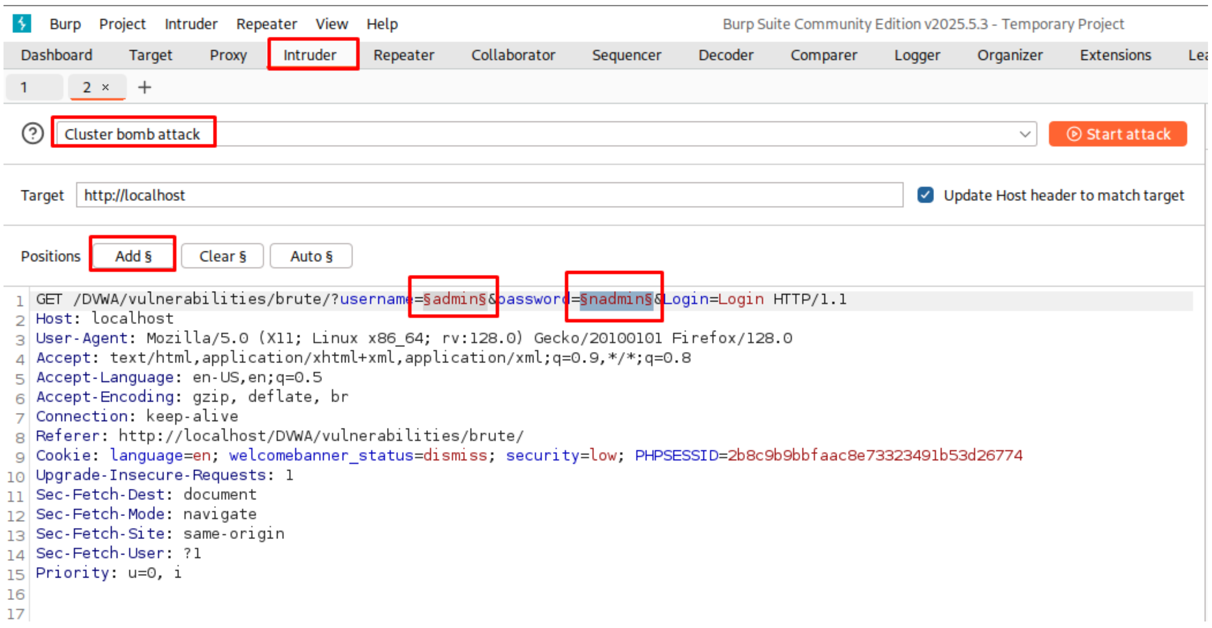
- Intercept a valid login request.
- Send to Intruder.
- Define payload position in either `username`, `password`, or both.

🔁 Payload:

- Use a wordlist such as `rockyou.txt`, SecLists, or custom usernames/passwords.

🎯 Success Indicator:

- Use response length/grep match and always follow redirects


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.5.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions
1 2 ×
Start attack
Add § Clear § Auto §

Cluster bomb attack

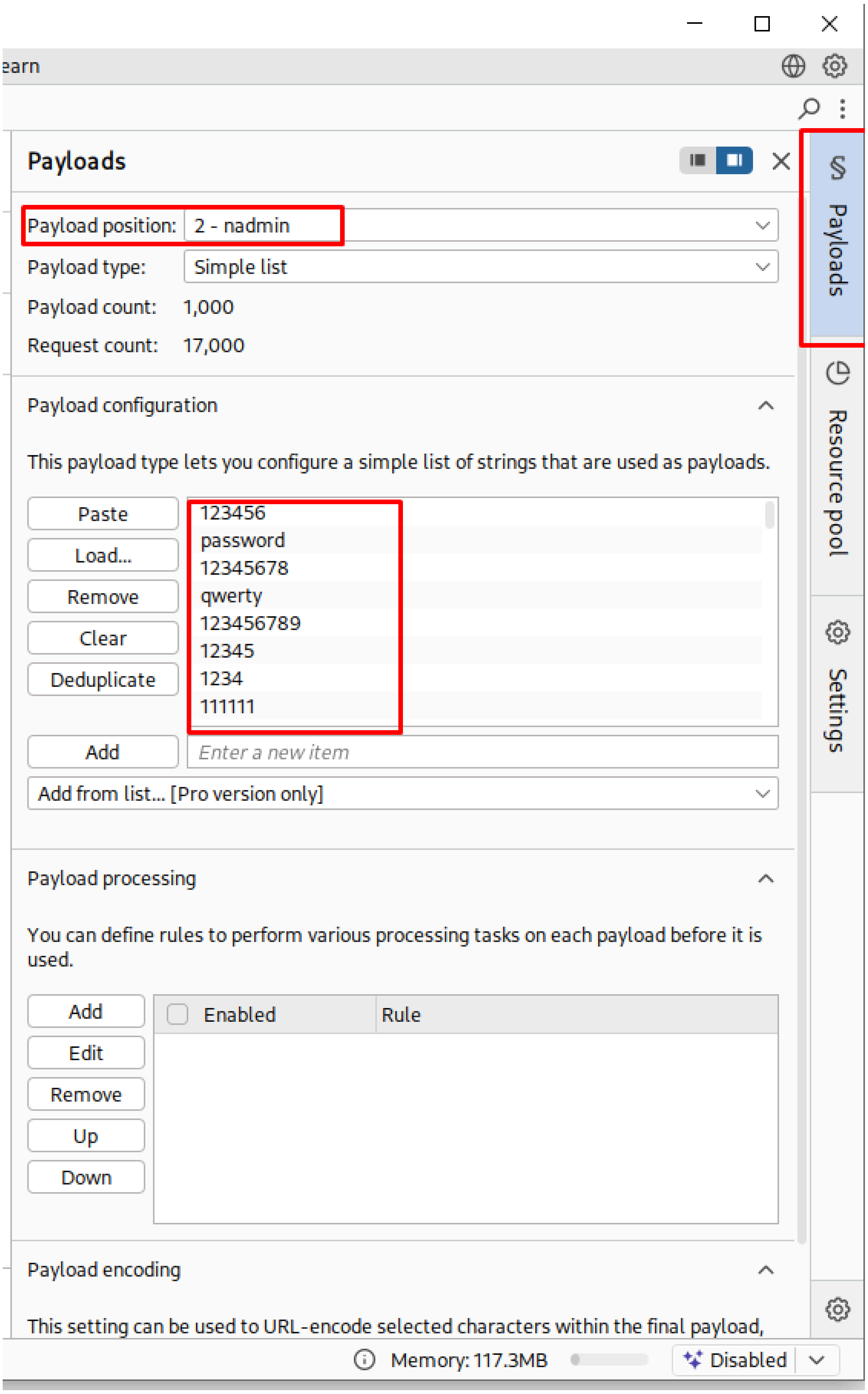
Target http://localhost

Update Host header to match target

Positions

```
GET /DVWA/vulnerabilities/brute/?username=§admin§&password=§nadmin§&Login=Login HTTP/1.1
Host: localhost
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://localhost/DVWA/vulnerabilities/brute/
Cookie: language=en; welcomebanner_status=dismiss; security=low; PHPSESSID=2b8c9b9bbfaac8e73323491b53d26774
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

Payload position: 2 - nadmin

Payload type: Simple list

Payload count: 1,000

Request count: 17,000

Payload configuration

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

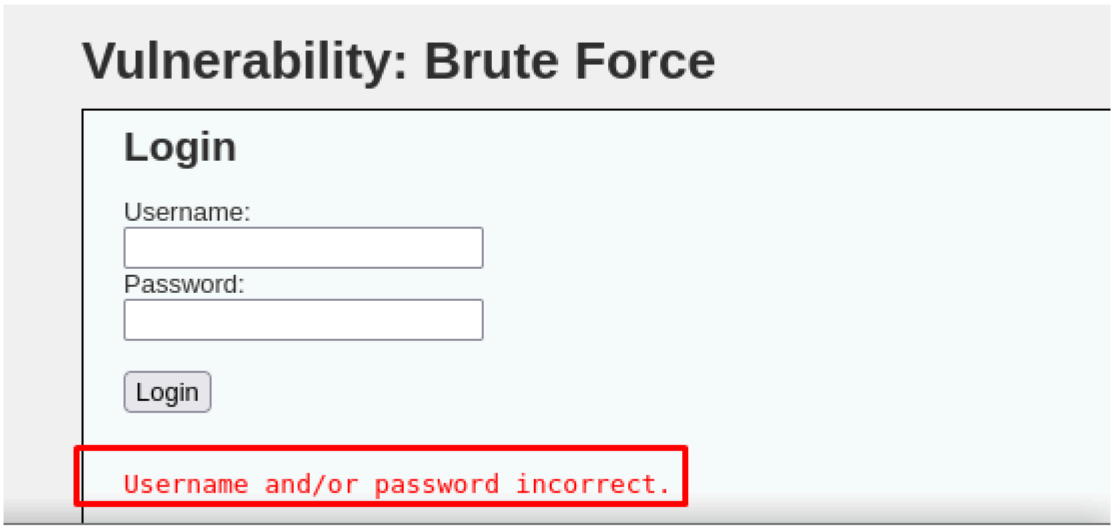
```
123456
password
12345678
qwerty
123456789
12345
1234
111111
```

Payload processing

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

Payload encoding

This setting can be used to URL-encode selected characters within the final payload,



Vulnerability: Brute Force

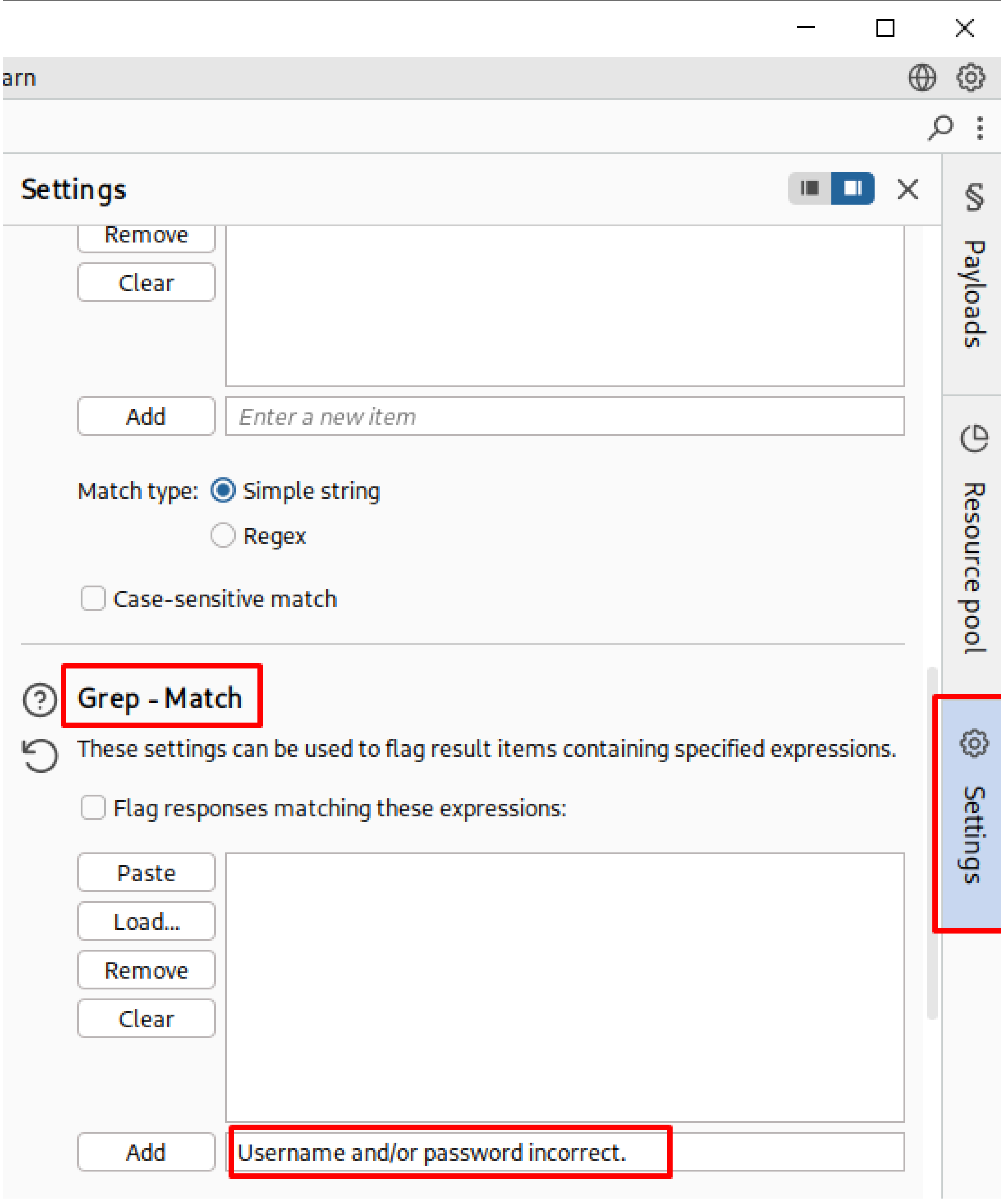
Login

Username:

Password:

Login

```
Username and/or password incorrect.
```



Settings

Remove
Clear
Add
Enter a new item

Match type: [✓ Simple string] Regex

☐ Case-sensitive match

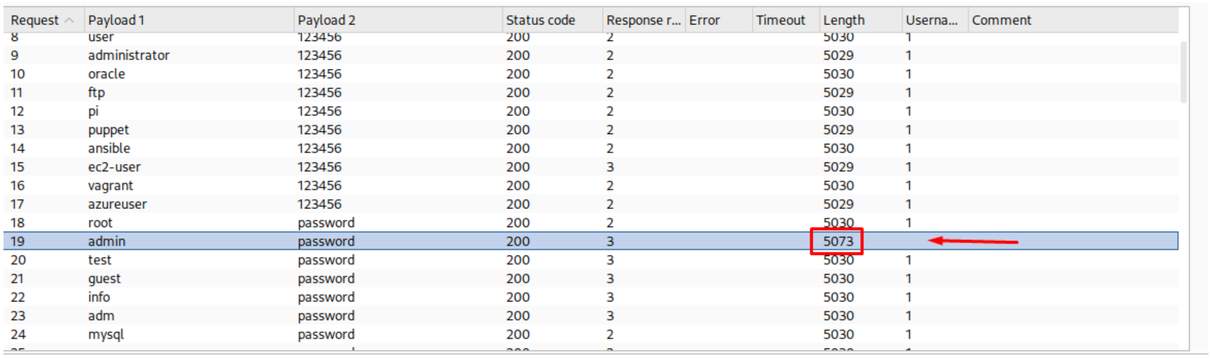
Grep - Match

These settings can be used to flag result items containing specified expressions.

☐ Flag responses matching these expressions:

Paste
Load...
Remove
Clear
Add

Username and/or password incorrect.



Request	Payload 1	Payload 2	Status code	Response r...	Error	Timeout	Length	Userna...	Comment
8	user	123456	200	2			5030	1	
9	administrator	123456	200	2			5029	1	
10	oracle	123456	200	2			5030	1	
11	ftp	123456	200	2			5029	1	
12	pi	123456	200	2			5030	1	
13	puppet	123456	200	2			5029	1	
14	ansible	123456	200	2			5030	1	
15	ec2-user	123456	200	3			5029	1	
16	vagrant	123456	200	2			5030	1	
17	azureuser	123456	200	2			5029	1	
18	root	password	200	2			5030	1	
19	admin	password	200	3			5073		
20	test	password	200	3			5030	1	
21	guest	password	200	3			5030	1	
22	info	password	200	3			5030	1	
23	adm	password	200	3			5030	1	
24	mysql	password	200	2			5030	1	

4️⃣Risk & Real-World Impact

🧠 Exploit Impact	🎯 Description
Credential Stuffing	Use leaked credentials from past breaches.
Account Takeover	Gain access to accounts due to password reuse.
Privilege Escalation	Admin account could be brute-forced.
Automated Recon for Bots	Attackers scan apps at scale for such login endpoints.

5️⃣🚧 Mitigation Measures (Secure Coding)

🚫 Vulnerability	✅ Mitigation
GET login method	Switch to POST with HTTPS only
No rate limiting	Use fail2ban, CAPTCHA, or exponential backoff
Weak hashing	Use password_hash() (bcrypt) with password_verify()
No input sanitization	Use prepared statements (mysqli_prepare)

No CSRF protection	Generate and validate CSRF token in session/form
No account lockout	Implement lockout mechanism after X failed attempts
Predictable feedback	Use generic error: "Invalid login credentials."

DVWA Brute Force (Medium Security Level)

📌 Overview The medium-level configuration improves input handling but remains vulnerable to brute-force and lacks secure login mechanisms.

🔍 **Vulnerabilities Identified:**

🔒 Vulnerability	❌ Problem Description
GET Method for Login	Still leaks credentials via URL
Weak MD5 Hashing	Still used
No Rate Limiting	Brute-force still possible
Predictable Error Feedback	Responses help identify correct credentials
No CSRF Protection	No anti-CSRF token enforcement
No Account Lockout	No limit on failed attempts

2️⃣Exploitation Phase: Automating Brute-Force 🔨 Tool: Burp Suite Intruder / wfuzz 🔧 Setup:

- Intercept GET request with sanitized input.
- Send to Intruder.

🔁 Payload:

- Use rockyou.txt or seclist targeted lists.

🎯 Success Indicator:

- Login success phrase or avatar image present.

4️⃣Risk & Real-World Impact

🧠 Exploit Impact	🎯 Description
Bot Attacks	Still feasible due to lack of CAPTCHA
Session Hijacking	No session validation, no secure cookies
Credential Enumeration	Error feedback leaks status

5️⃣🚧 Mitigation Measures (Secure Coding)

🚫 Vulnerability	✅ Mitigation
GET method	Use POST only
CSRF missing	Token-based protection required
MD5 hashing	Replace with bcrypt via password_hash()
Feedback messages	Standardize all errors

DVWA Brute Force (High Security Level)

📌 Overview The high-level security mode introduces CSRF tokens and some delay tactics but still lacks strong password hashing and session controls.

🔍 **Vulnerabilities Identified:**

🔒 Vulnerability	❌ Problem Description
MD5 Hashing	Still insecure
GET Method	Still used for login
Limited Anti-Automation	Delay is random, but not effective enough
Feedback Error Messaging	Still reveals login status
No Secure Cookie Flags	No HttpOnly, Secure, SameSite

2️⃣Exploitation Phase: Automating Brute-Force 🔨 Tool: Burp Suite Intruder (manual CSRF token refresh required) 🔧 Setup:

- Intercept login and extract CSRF token
- Inject into each request manually or automate token fetching

🔁 Payload:

- Use shortened dictionary
- Brute-force slower due to random `sleep()`

🎯 Success Indicator:

- Successful login message and avatar image

4️⃣Risk & Real-World Impact

🧠 Exploit Impact	🎯 Description
CSRF Bypass	Predictable token or re-used token may work
Credential Harvest	Manual attacks still possible
No Lockout	Still brute-forceable under slow rate

5️⃣🚧 Mitigation Measures (Secure Coding)

🚫 Vulnerability	✅ Mitigation
CSRF token handling	Bind tokens to sessions, rotate often
Weak hashing	Use password_hash() + salting
Delay logic	Use exponential backoff, CAPTCHA

DVWA Brute Force (Impossible Security Level)

📌 Overview The impossible level implements strong login security including POST method, CSRF tokens, account lockouts, and PDO with prepared statements.

🔍 **Vulnerabilities Identified:**

🔒 Vulnerability	❌ Problem Description
None exploitable via brute-force	Strong anti-automation and secure coding enforced

---

2️⃣Exploitation Phase: Automating Brute-Force 🔨 Tool: Attempt fails due to CSRF token enforcement, account lockout, and random delays 🔧 Setup:

- Difficult to extract valid CSRF token on every request
- Account locks after 3 failed attempts

🔁 **Payload:**

- Not feasible; blocked via logic

🎯 Success Indicator:

- Not reachable unless valid creds and token in time window

4️⃣**Risk & Real-World Impact**

🧠 Exploit Impact	🎯 Description
Minimal risk	Controls prevent bruteforce
Auditable	PDO and prepared statements used

5️⃣🚧 Mitigation Measures (Secure Coding)

✅ The implementation includes:

- POST login + HTTPS
- password_hash() and password_verify()
- CSRF token validated per session
- PDO prepared statements
- Login attempts tracking and lockout
- Random sleep and error feedback control
- Secure cookie flags (Secure, HttpOnly, SameSite)