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 sakta 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:

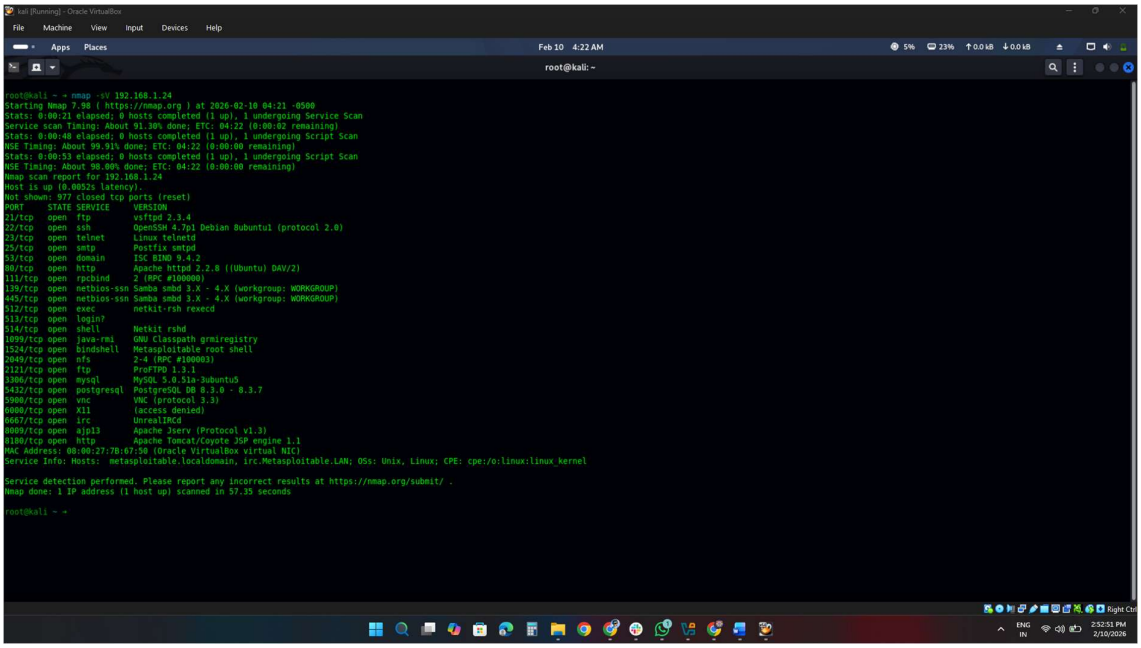
- **Advanced Reporting:** Report sirf technical nahi honi chahiye. **CVSS Score** se batana hoga ki vulnerability kitni khatarnak hai.
- **Stakeholder Communication:** Developer ko batao ki code kaise fix karein, aur Manager ko batao ki isse business ko kitna nuksan ho sakta hai.
- **Remediation:** Sirf "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

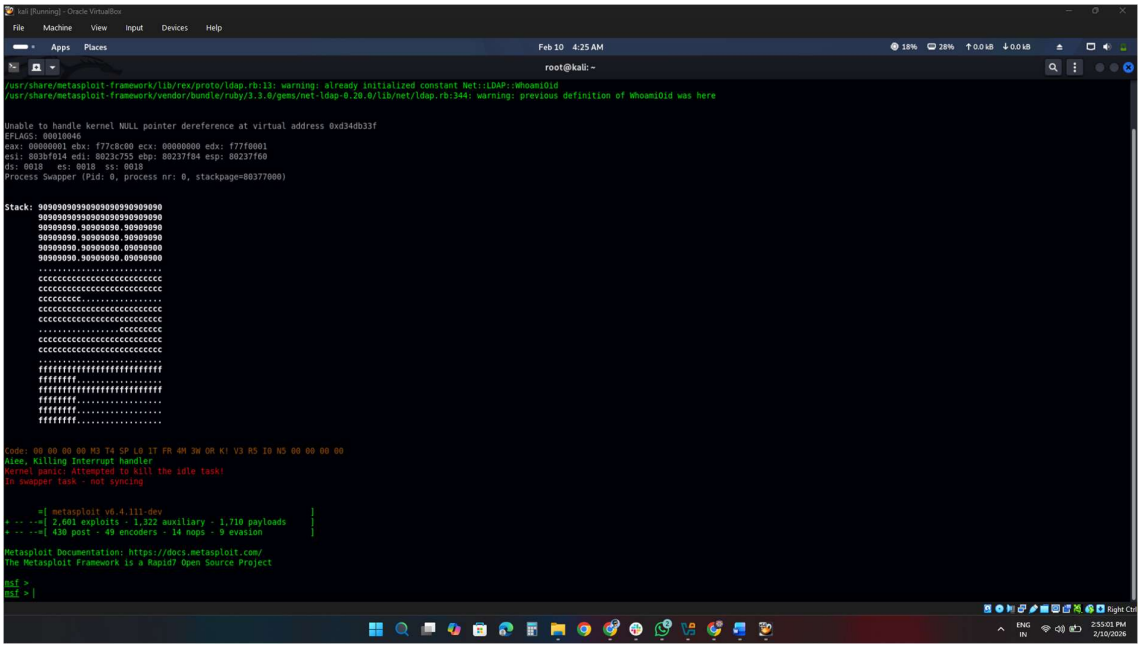
Step 1: Services Scan (Nmap)

Command: nmap -sV 192.168.1.24

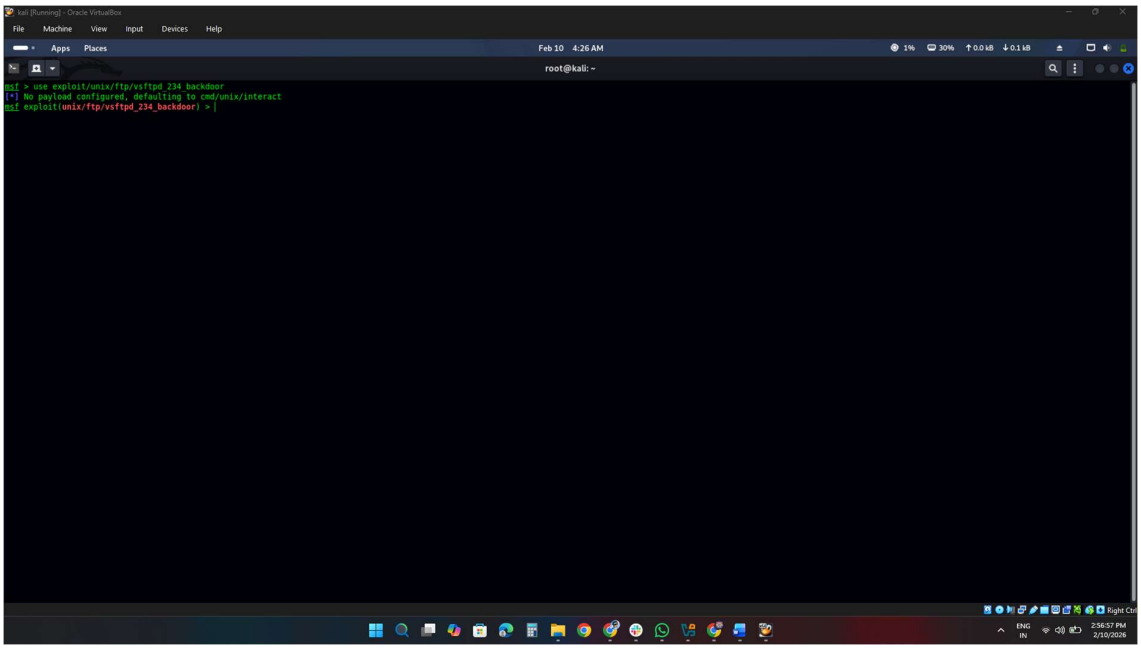


Step 2: Multi-Stage Attack (Exploit Chaining)

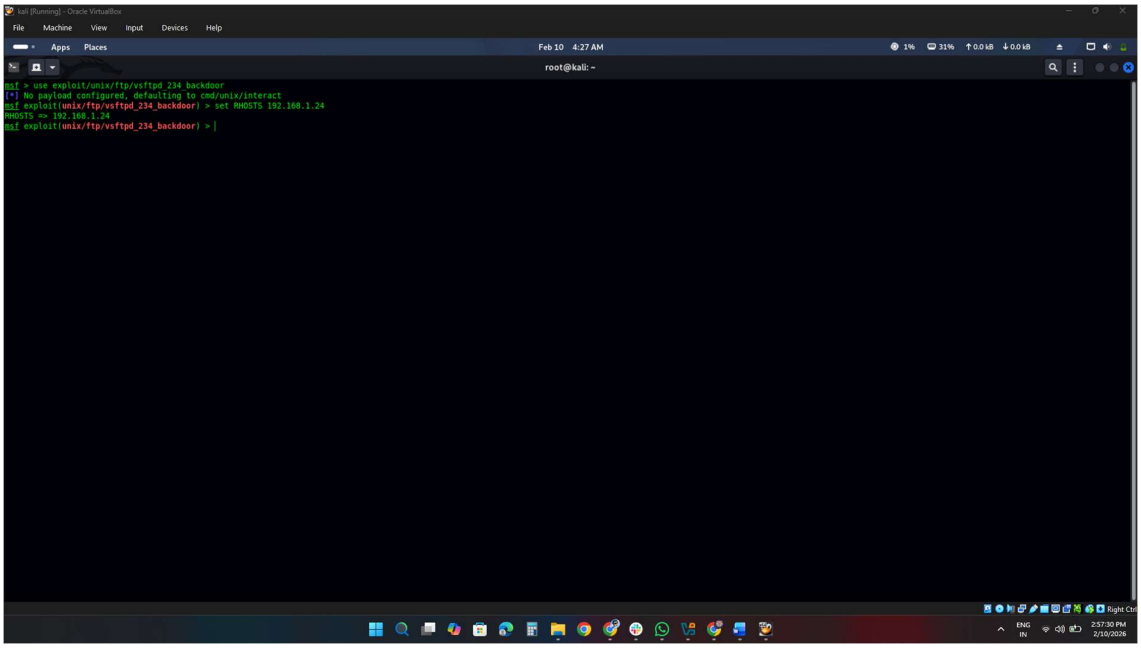
Command: msfconsole



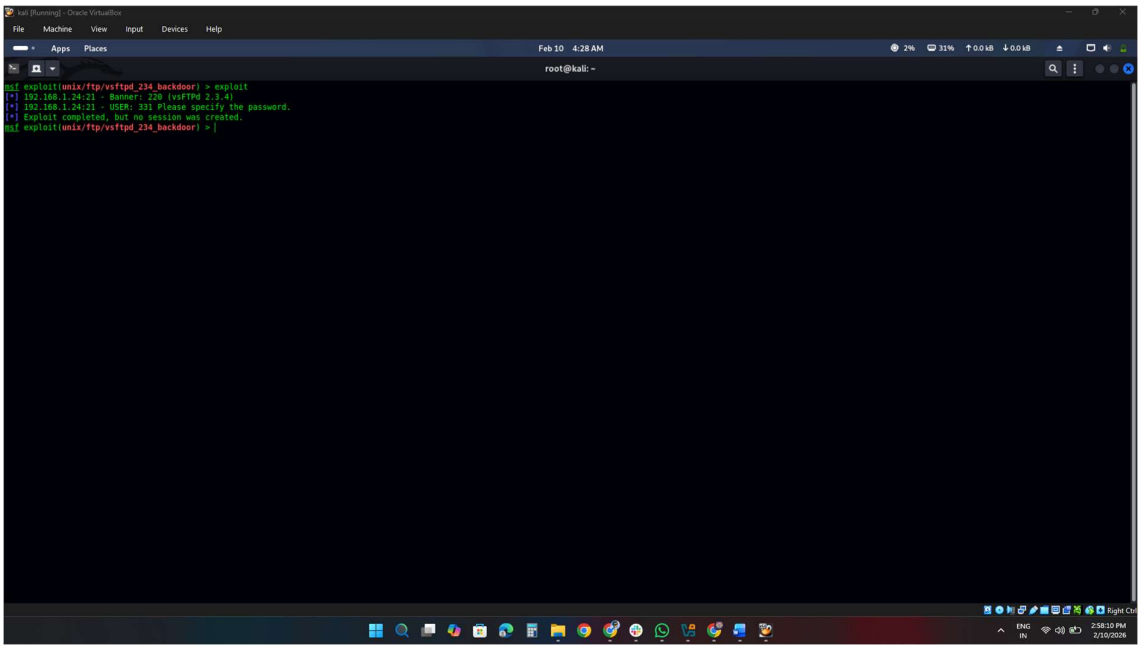
Command: use exploit/unix/ftp/vsftpd_234_backdoor



Command: set RHOSTS 192.168.1.24



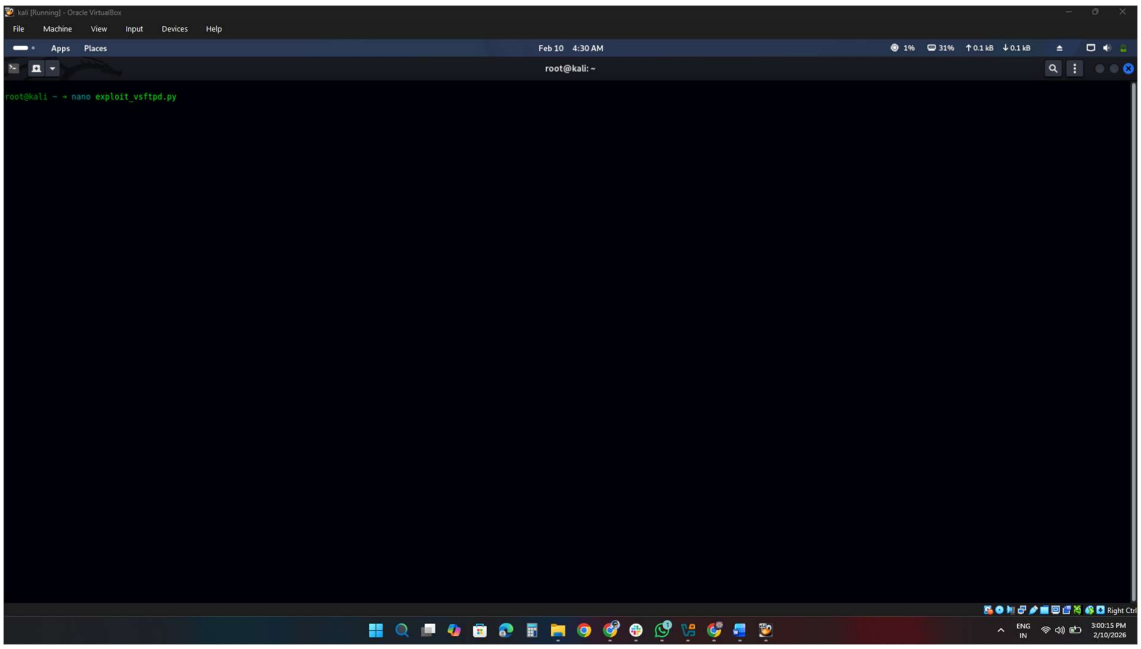
Command: exploit

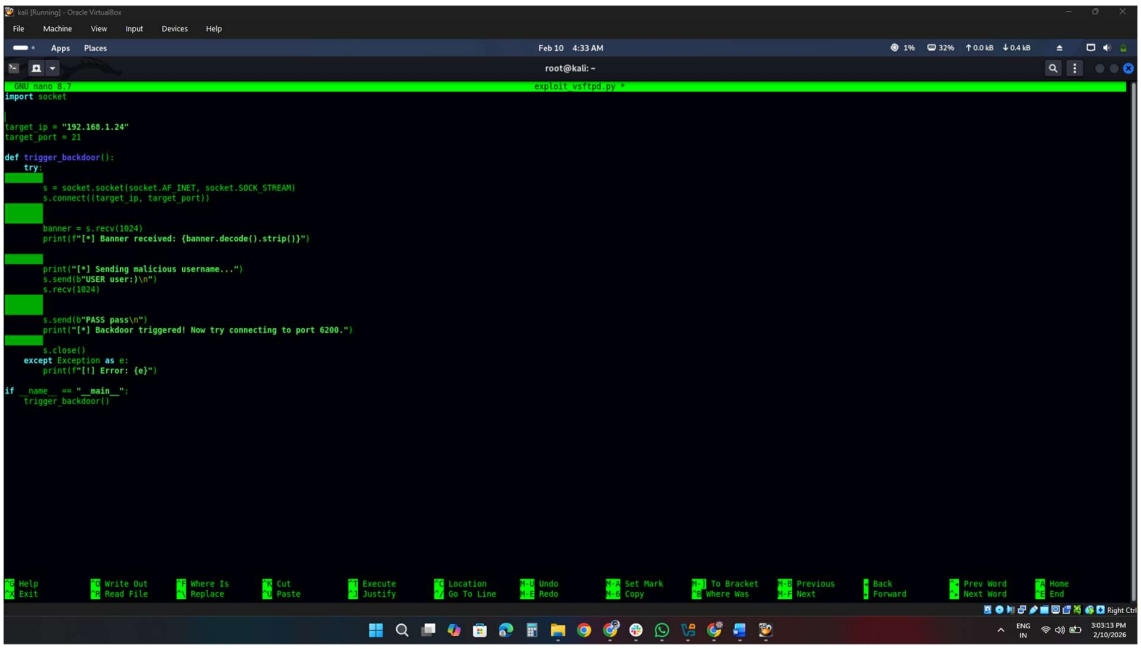


Step 3: Custom PoC Development (Manual Trigger)

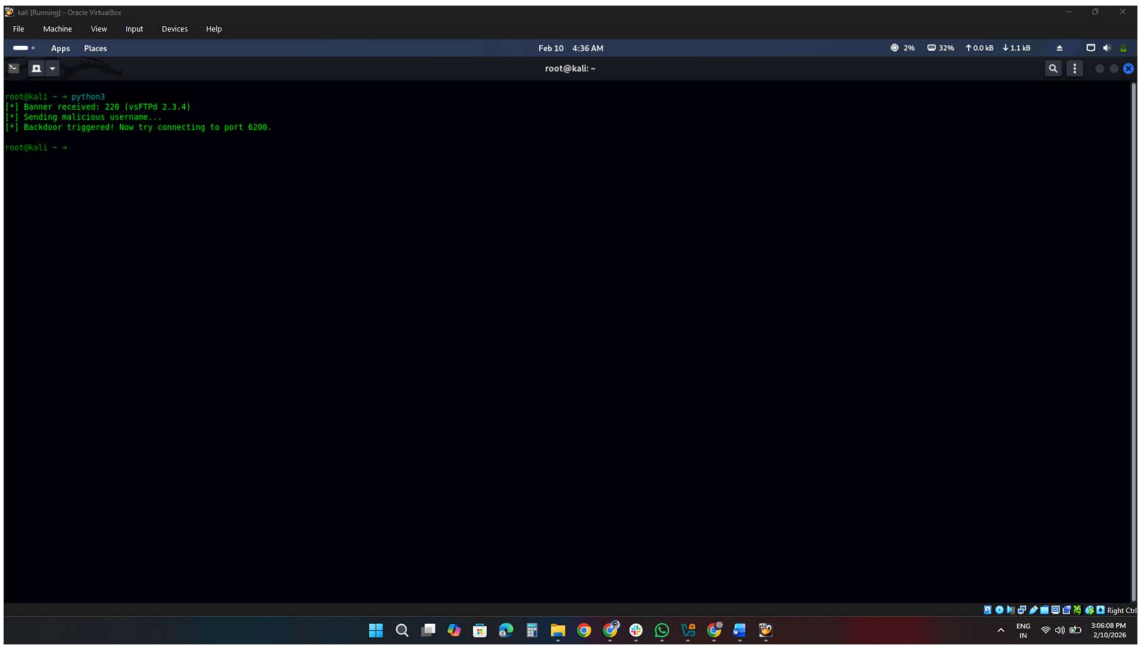
Step 1: Python Script Create

Command: nano exploit_vsftpd.py



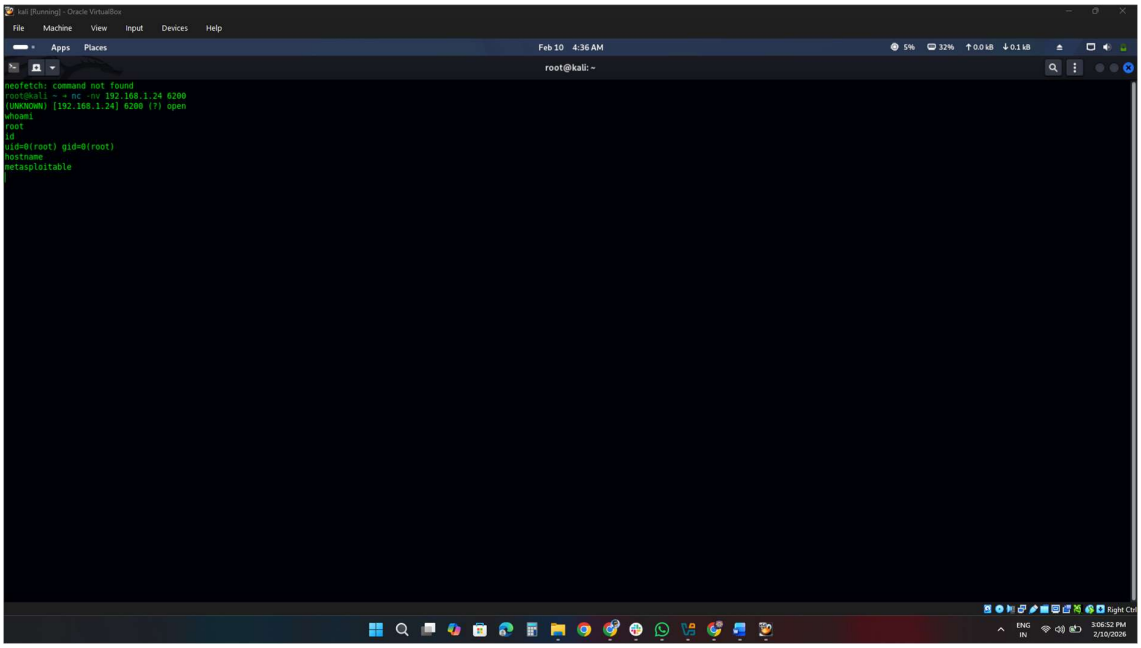


Step 2: How to run Script



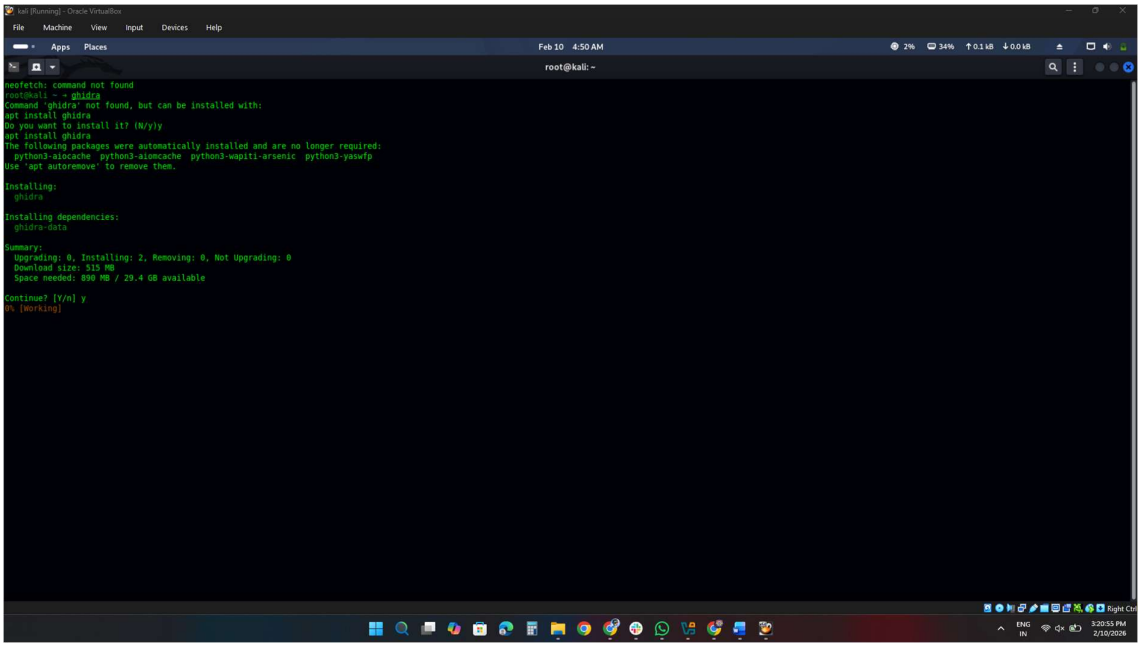
Access Gain (The Final Step)

Command: nc -nv 192.168.1.24 6200

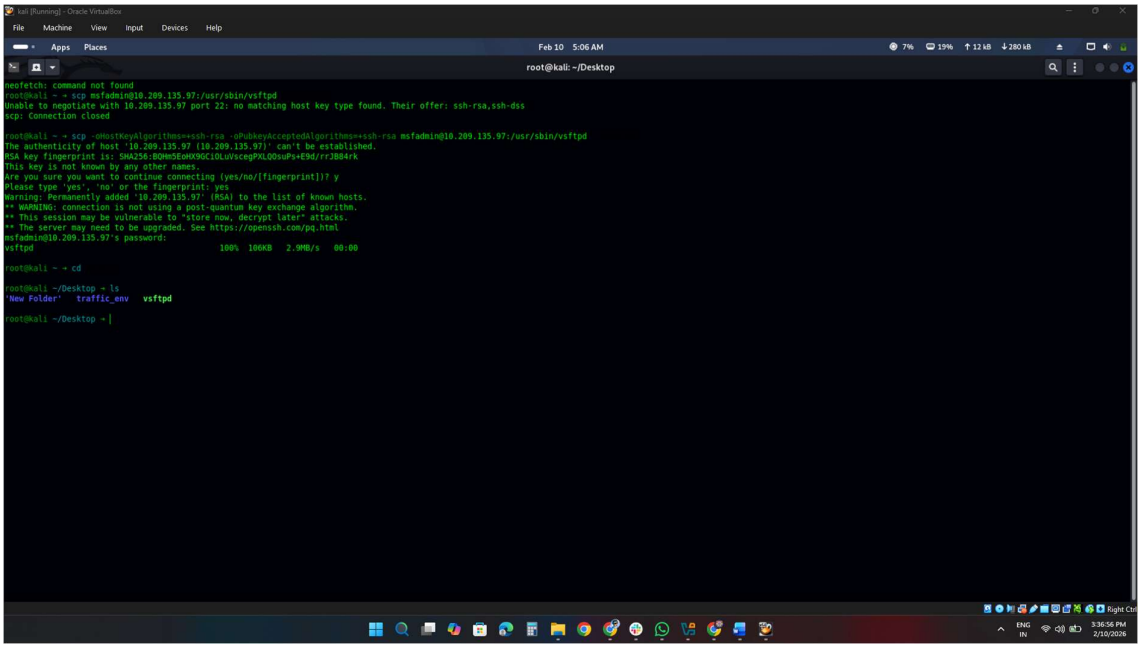


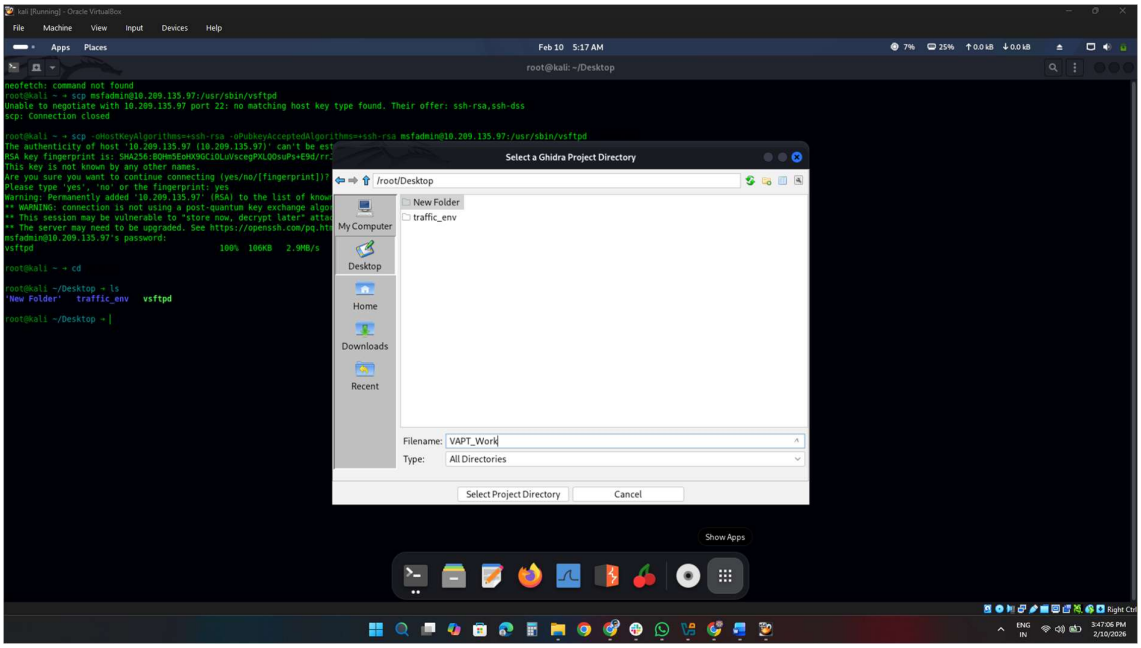
Task 1: Custom PoC Analysis (Ghidra)

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



Command: scp -oHostKeyAlgorithms=+ssh-rsa -oPubkeyAcceptedAlgorithms=+ssh-rsa msfadmin@10.209.135.97:/usr/sbin/vsftpd ~/Desktop/





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

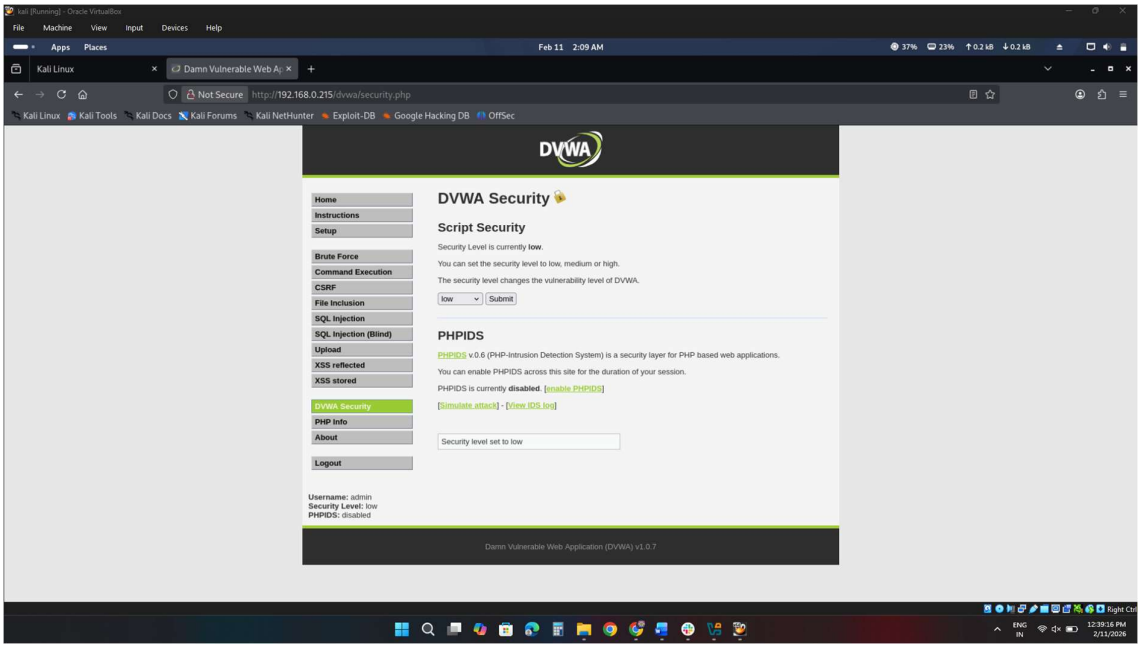
Exploit Stage	**Tool**	**Outcome**	**Result**
Exploitation	Metasploit/Python	**Success**	Root Shell Access
Custom PoC	Python Script	**Success**	Manual Backdoor Trigger
Static Analysis	Ghidra	**Documented**	Logic Verification
Bypass Analysis	ROPgadget	**Documented**	Defense Evasion Strategy

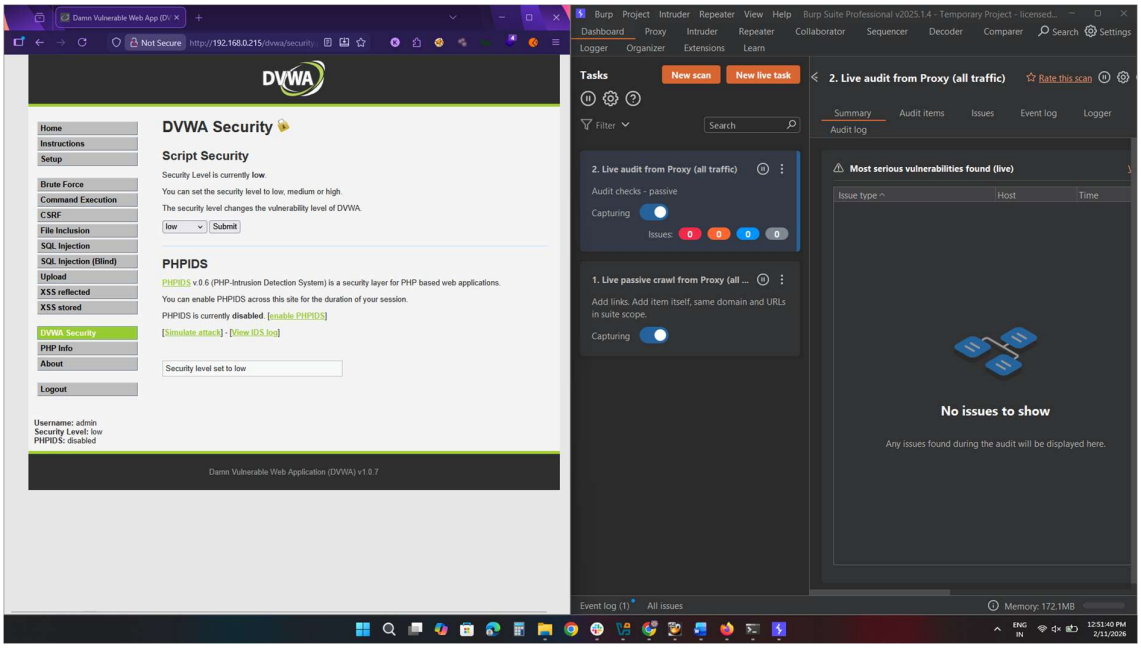
Lab 02: API Security Testing Lab

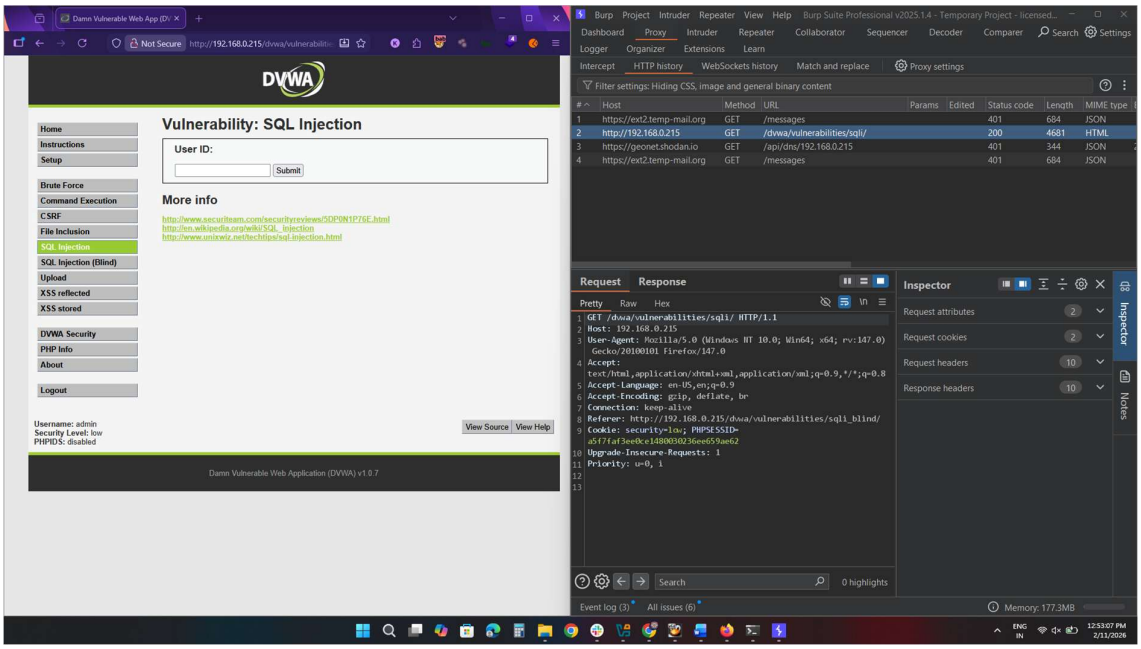
Target Environment: Metasploitable 2 (192.168.0.215) Tools Used: Windows Burp Suite, Postman, SQLmap.

Task 1: API Endpoint Enumeration & Setup

- Reconnaissance and Proxy Configuration

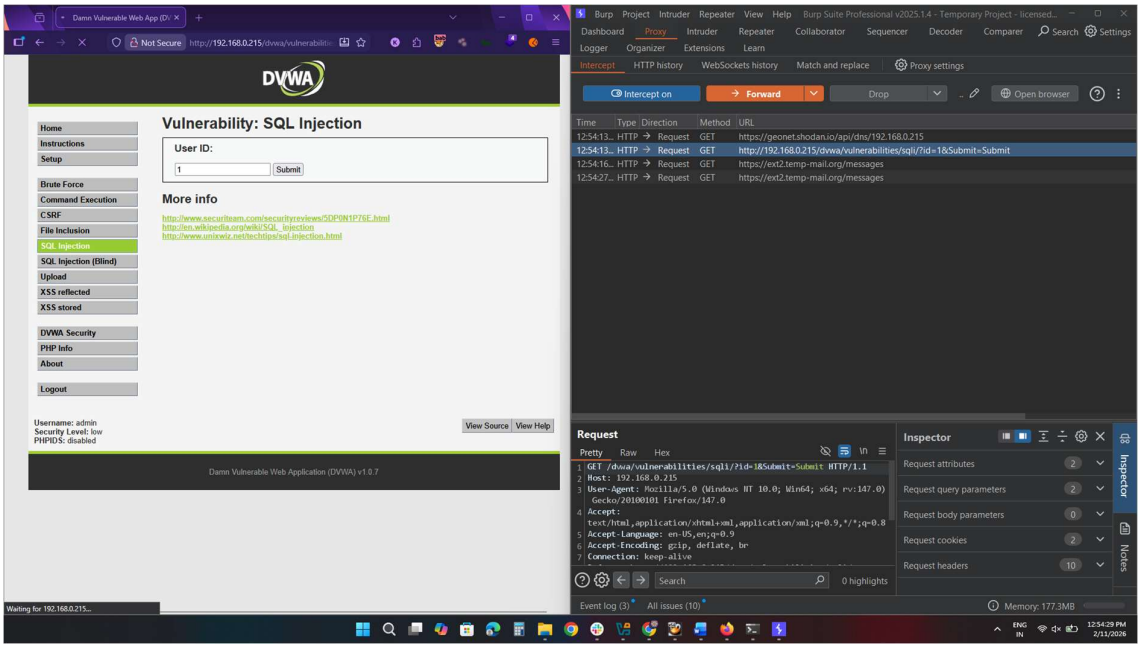


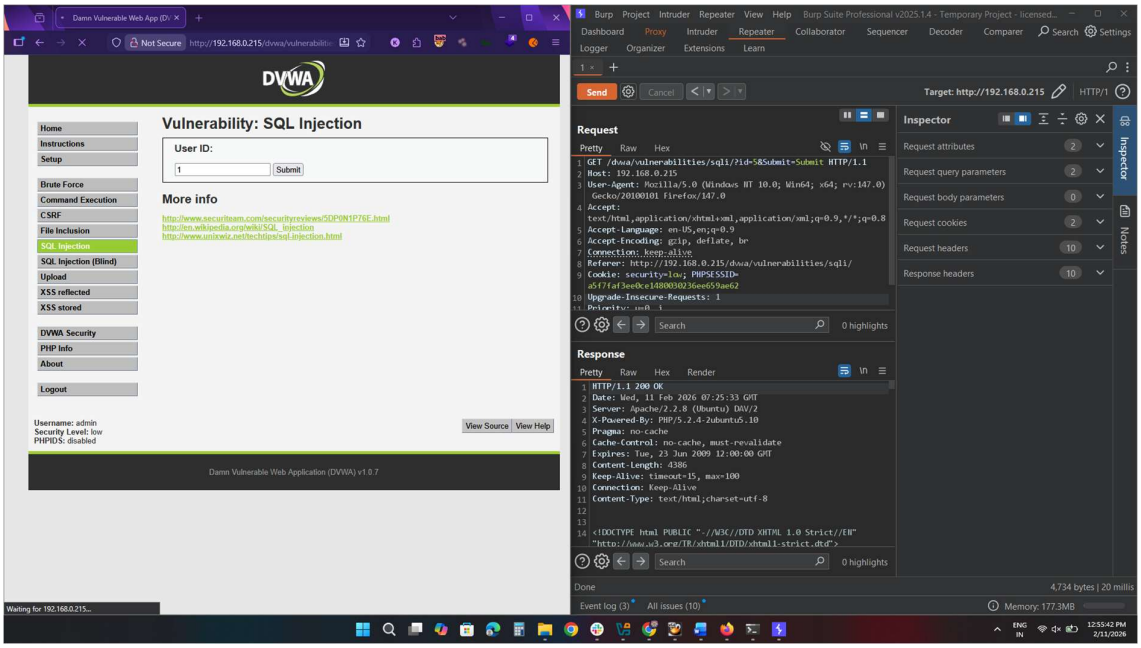




Task 2: Testing for BOLA (Broken Object Level Authorization)

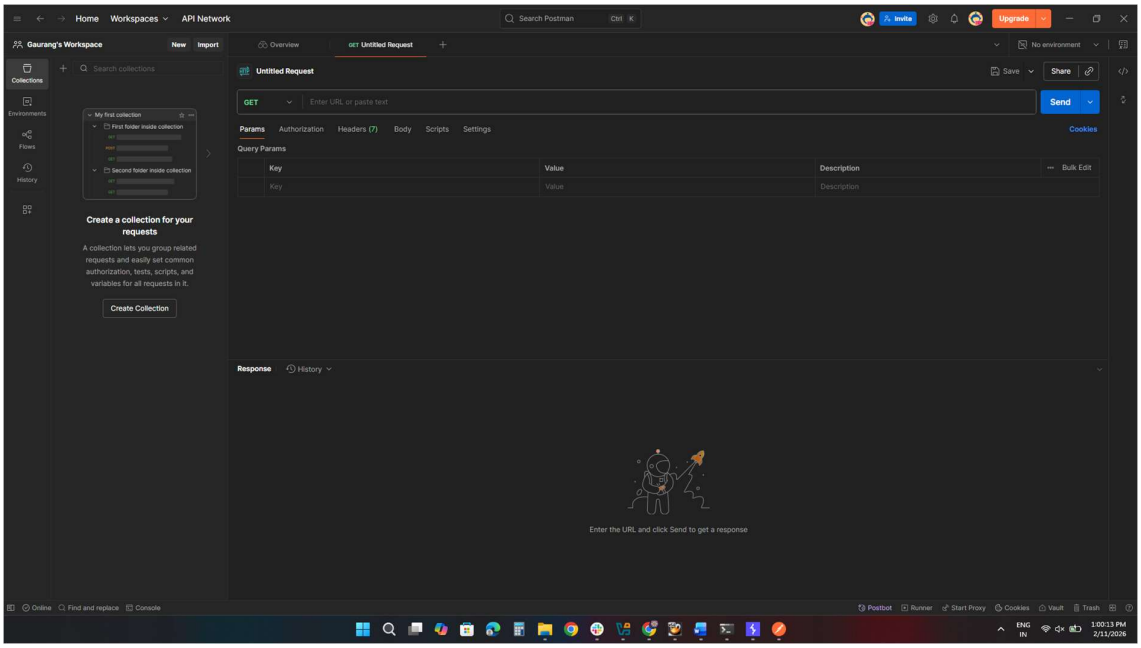
- Exploit ID 008: Unauthorized Data Access

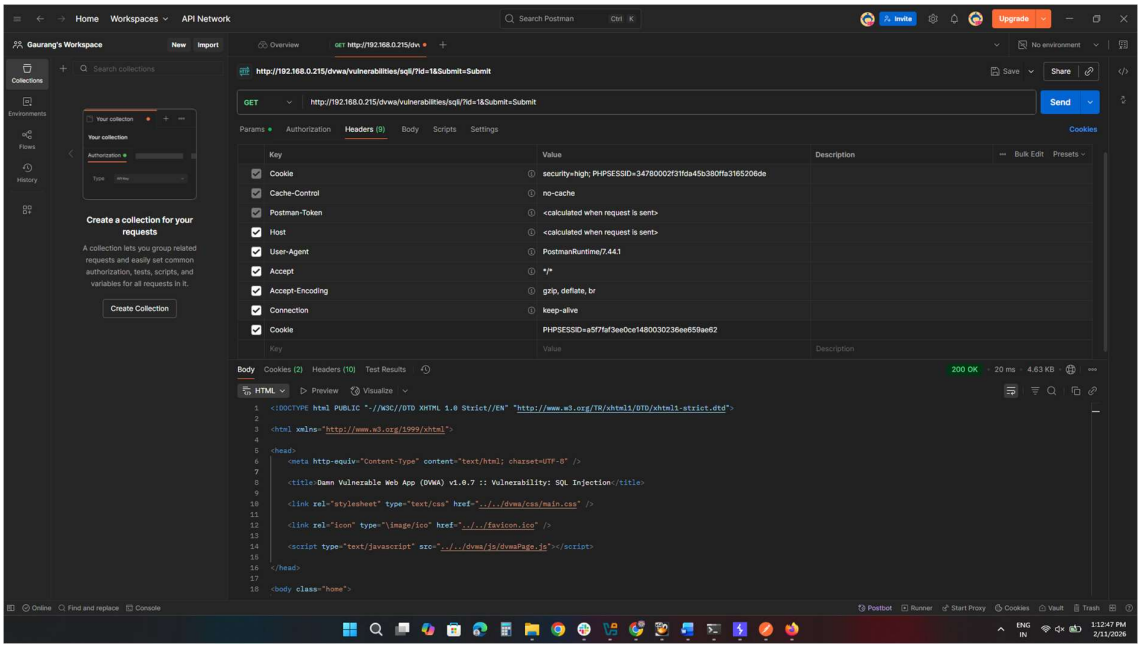


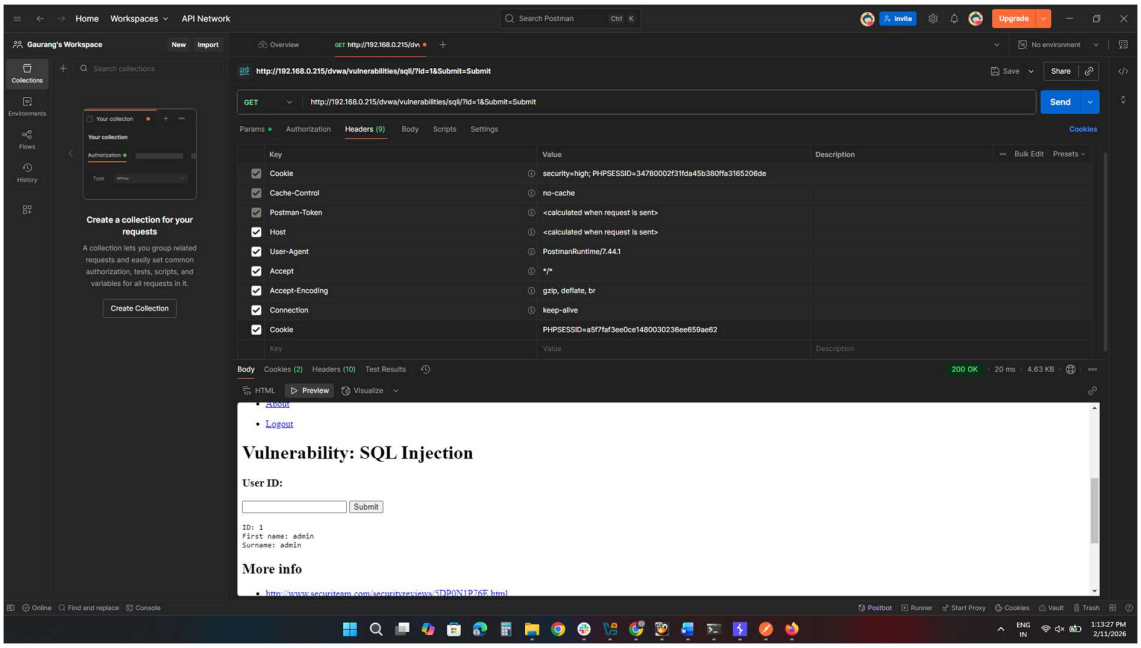


Task 3: Fuzzing GraphQL & API Tokens (Postman)

- Exploit ID 009: API Fuzzing and Information Disclosure


Home
Workspaces
API Network
Search Postman
Gaurang's Workspace
New
Import
Untitled Request
GET
Enter URL or paste text
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Query Params
Key
Value
Description
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Response
Enter the URL and click Send to get a response


Home
Workspaces
API Network
Search Postman
Gaurang's Workspace
http://192.168.0.215/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit
GET
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Key
Value
Description
Cookie
security=high; PHPSESSID=34780002f31fda45b380ffa3165206de
Cache-Control
no-cache
Postman-Token
<calculated when request is sent>
Host
<calculated when request is sent>
User-Agent
PostmanRuntime/7.44.1
Accept
/
Accept-Encoding
gzip, deflate, br
Connection
keep-alive
Cookie
PHPSESSID=a5f7faf3ee0ce1480030236ee659ae62
Body
Cookies (2)
Headers (10)
Test Results
200 OK
HTML
Preview
Visualize
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection



4. Final Lab 02 Log & Summary

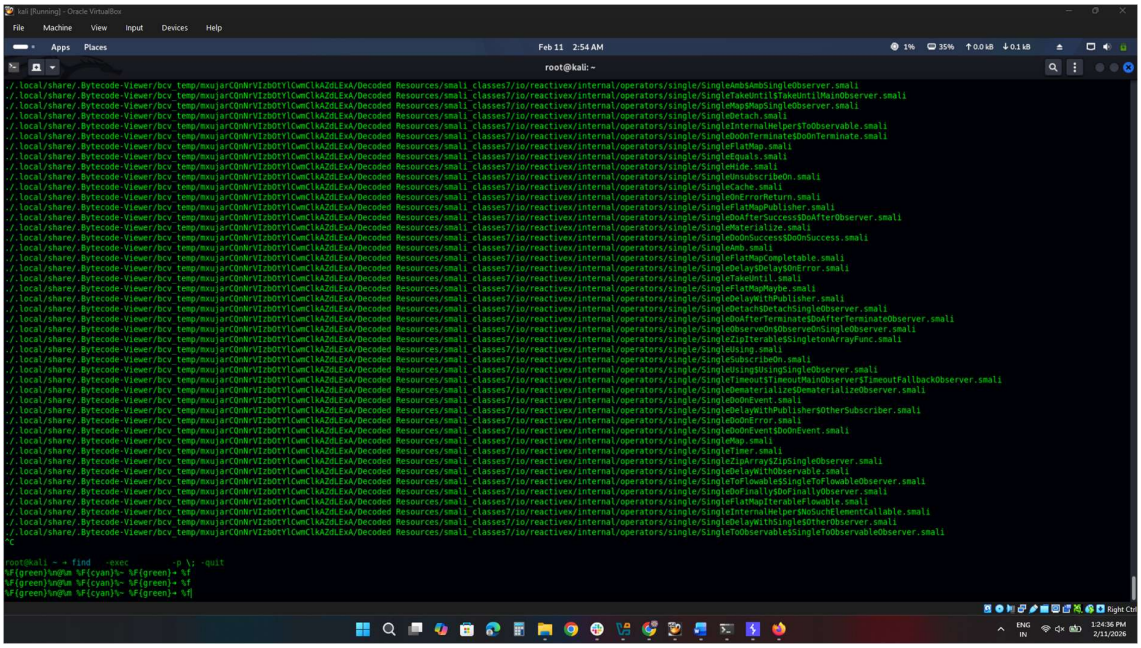
- The assessment identified critical authorization flaws within the DVWA API-style endpoints. Using Burp Suite to manipulate object identifiers, I successfully achieved unauthorized access to user data, confirming a BOLA vulnerability. Postman-based fuzzing further revealed significant input validation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

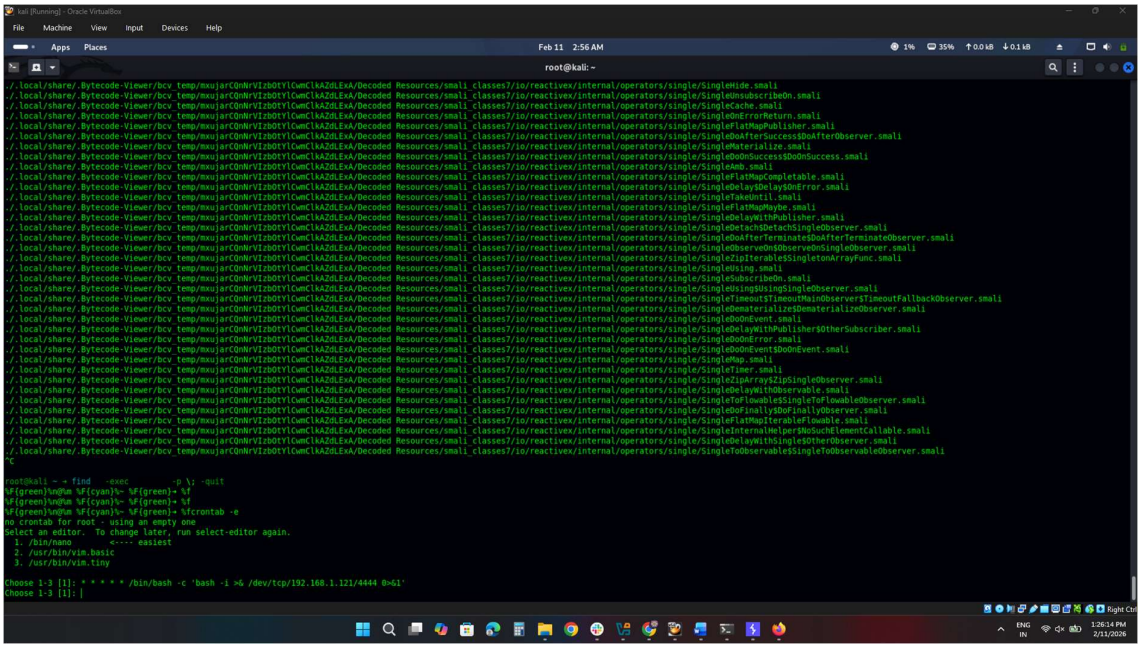
1. Privilege Escalation: SUID Exploit

- 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

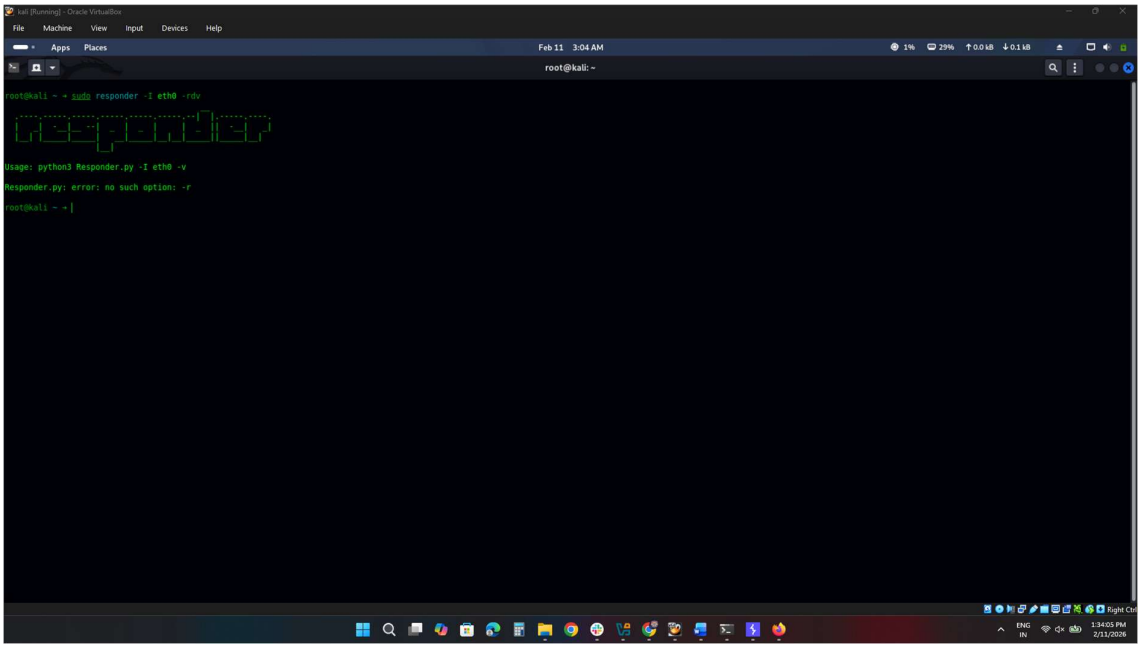
- This lab successfully demonstrated the process of escalating from a standard user to a root-level administrator using SUID binary exploitation. The post-exploitation phase focused on establishing persistence via scheduled tasks, highlighting the critical importance of monitoring system configuration files and crontab entries in a production environment.

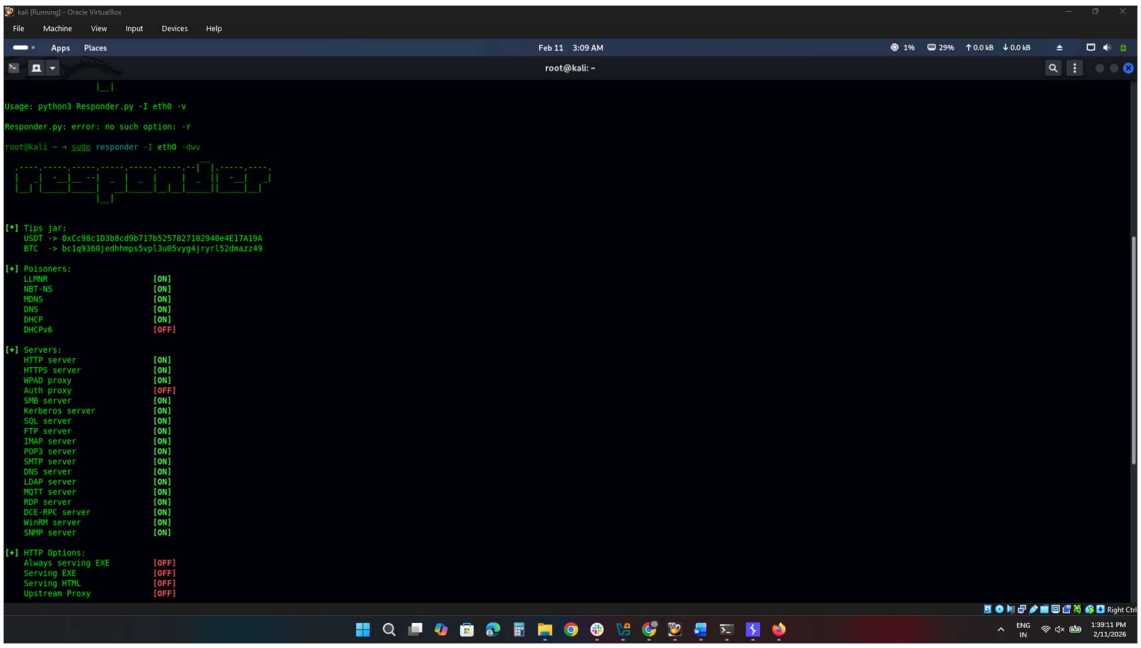
Lab 04: Network Protocol Attacks Lab

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

Task 1: SMB Relay & NTLM Hash Capture

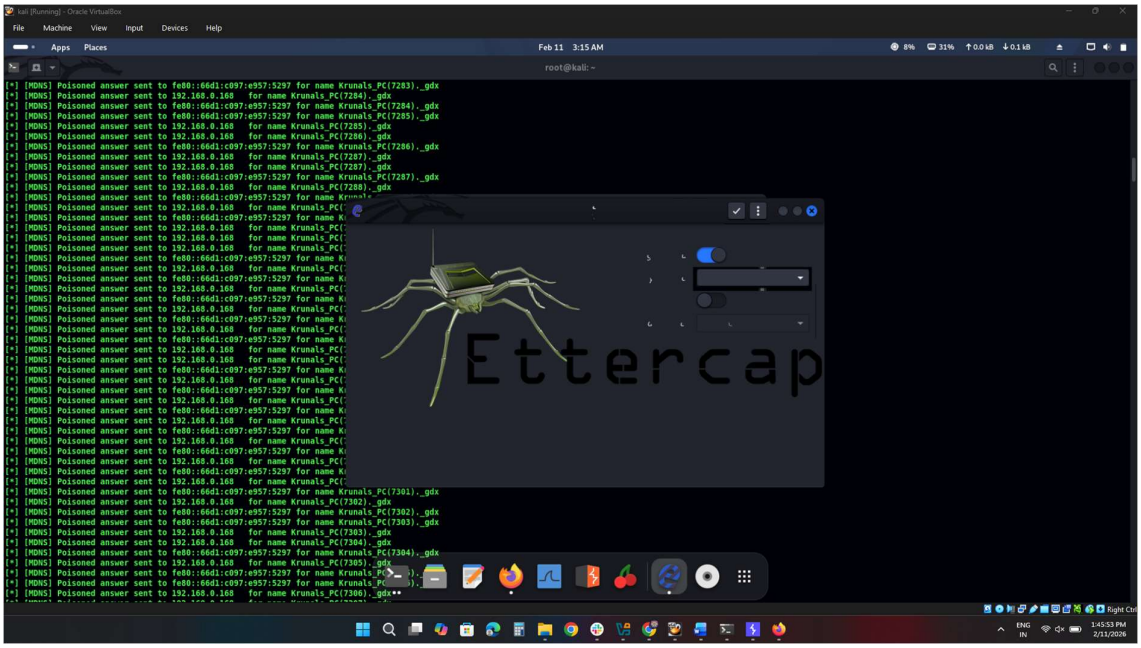
- Capturing Network Hashes using Responder





3. MitM: ARP Spoofing with Ettercap

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



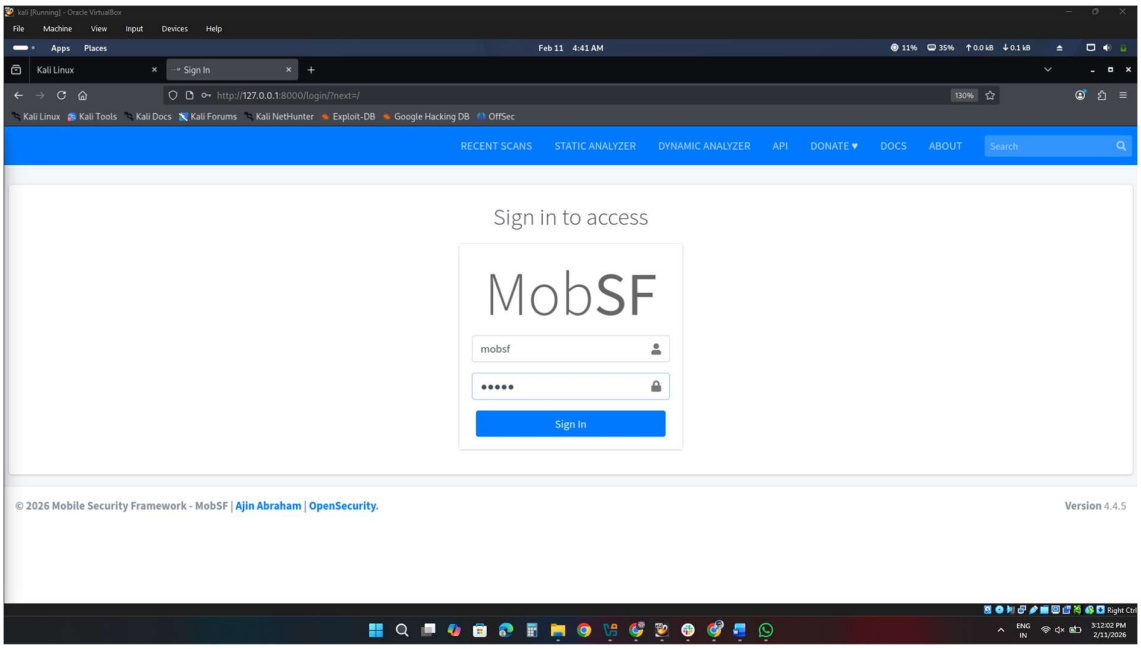
➢ **MitM Summary**

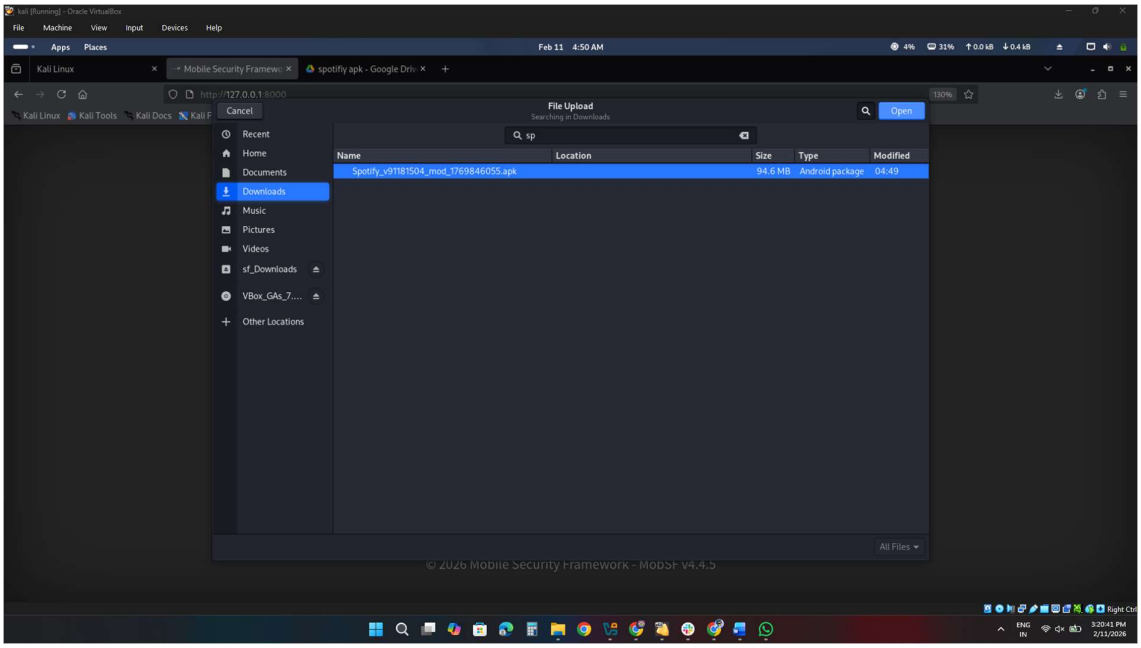
- **The network protocol lab utilized Responder to successfully poison local name resolution requests, effectively capturing redirected traffic. By positioning the attacker machine as a Man-in-the-Middle (MitM) through ARP spoofing with Ettercap, I demonstrated the ability to intercept sensitive authentication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

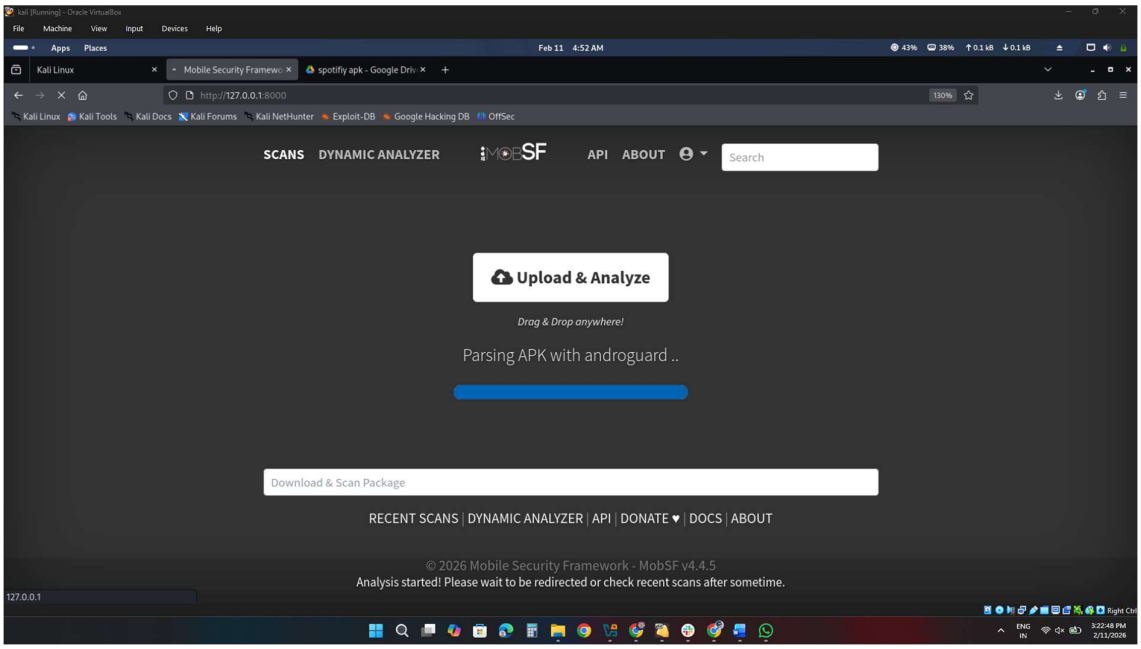
Lab 05: Mobile Application Testing Lab

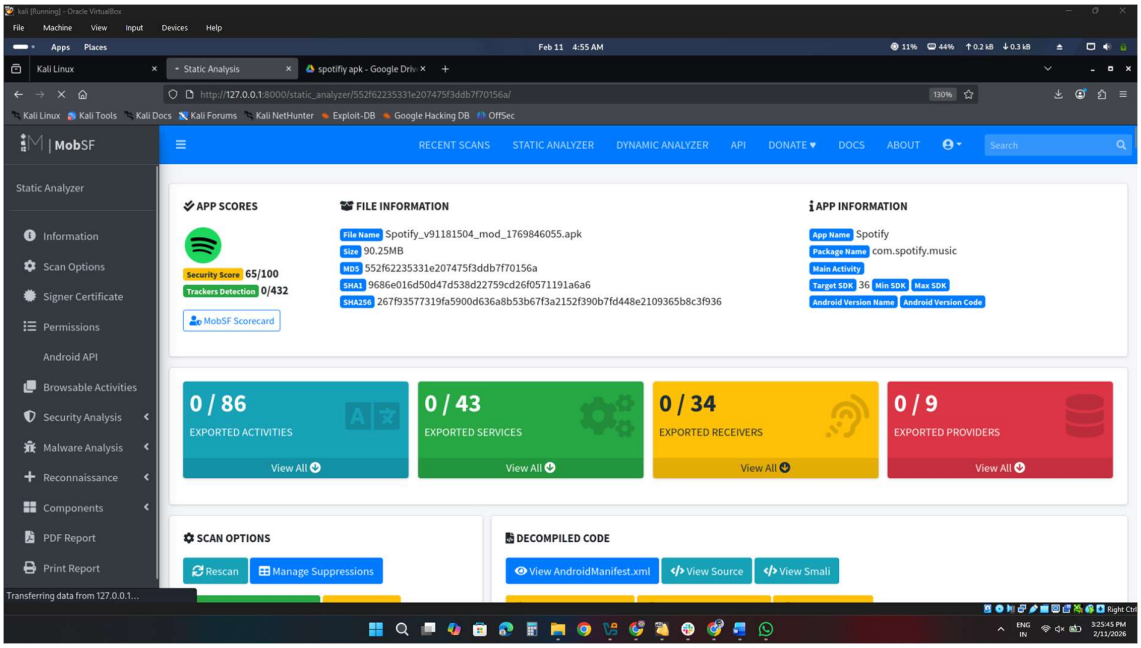
Target App: ….. **Tools Utilized:** MobSF, Frida, Drozer.

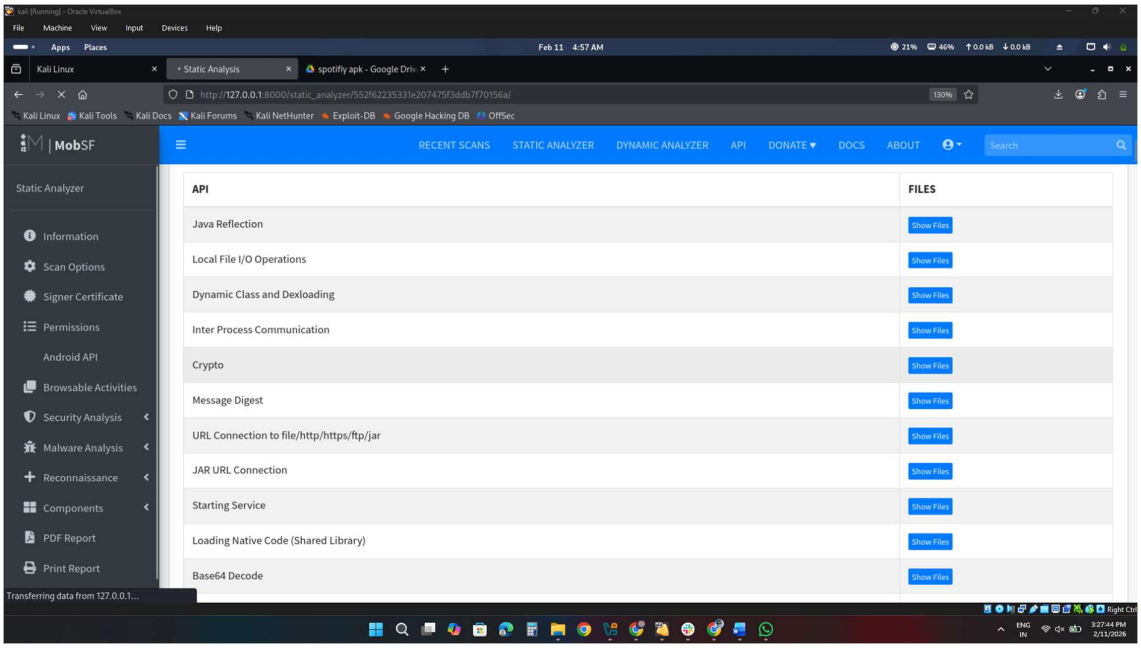
- Static Analysis: MobSF

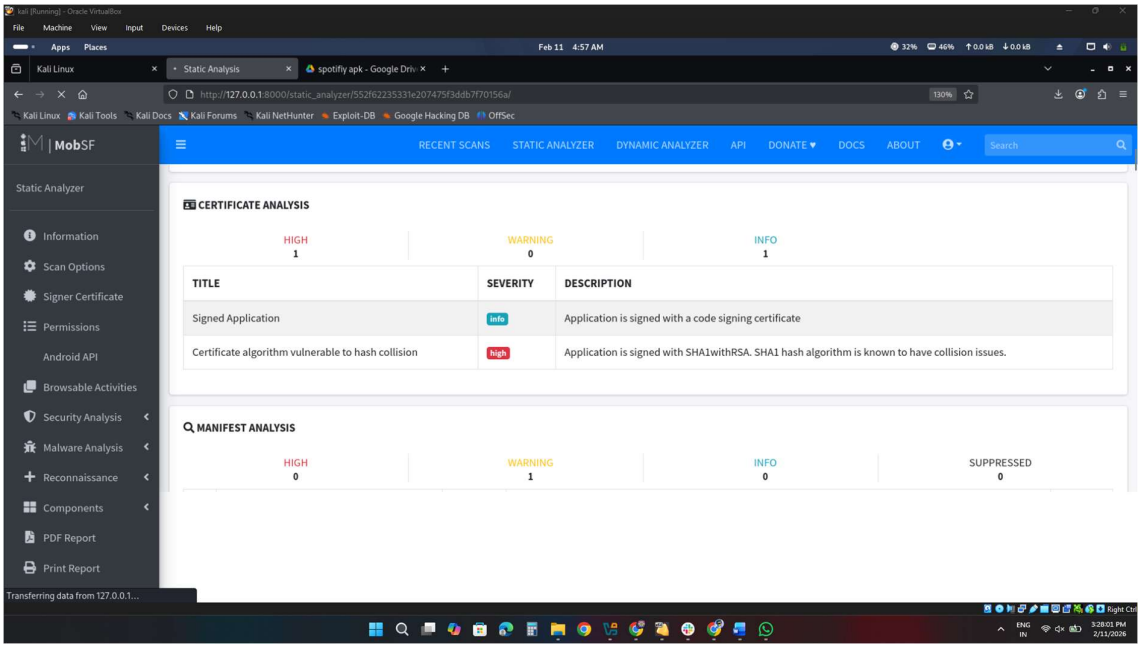


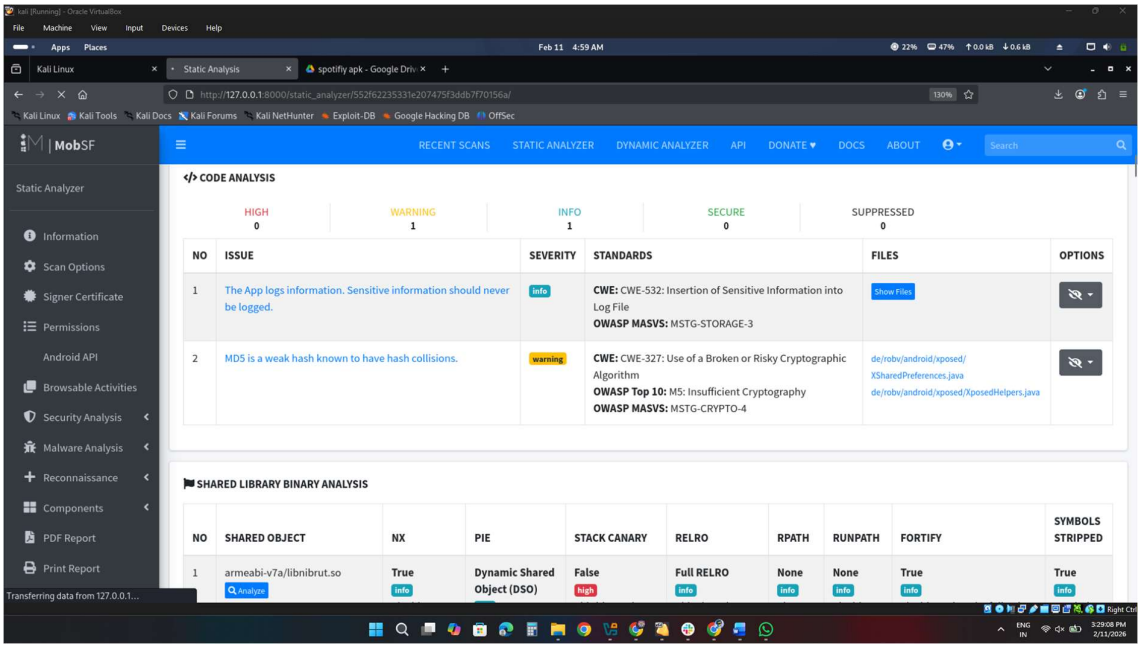

File Upload
Searching in Downloads
Cancel
Open
sp
Name
Location
Size
Type
Modified
Spotify_v91181504_mod_1769846055.apk
94.6 MB
Android package
04:49
Recent
Home
Documents
Downloads
Music
Pictures
Videos
sf_Downloads
Other Locations
All Files
© 2026 Mobile Security Framework - MobSF v4.4.5

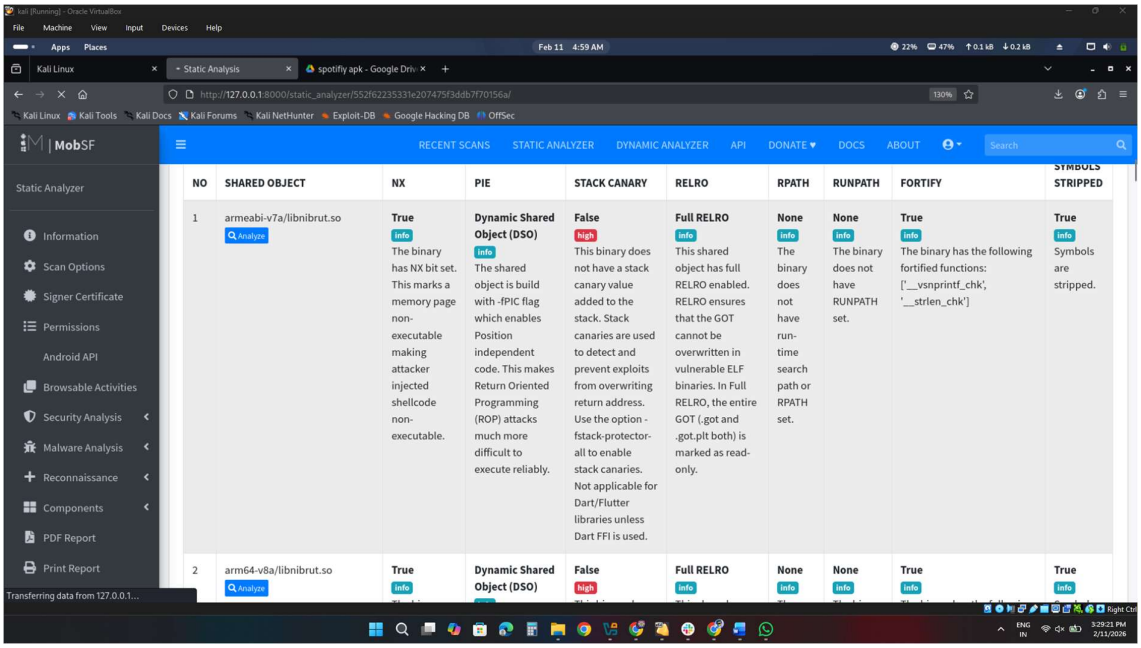

SCANS
DYNAMIC ANALYZER
MOBSF
API
ABOUT
Search
Upload & Analyze
Drag & Drop anywhere!
Parsing APK with androguard ..
Download & Scan Package
RECENT SCANS | DYNAMIC ANALYZER | API | DONATE ♥ | DOCS | ABOUT
© 2026 Mobile Security Framework - MobSF v4.4.5
Analysis started! Please wait to be redirected or check recent scans after sometime.


MobSF
Static Analyzer
Information
Scan Options
Signer Certificate
Permissions
Android API
Browsable Activities
Security Analysis
Malware Analysis
Reconnaissance
Components
PDF Report
Print Report
RECENT SCANS
STATIC ANALYZER
DYNAMIC ANALYZER
API
DONATE
DOCS
ABOUT
APP SCORES
Security Score 65/100
Trackers Detection 0/432
MobSF Scorecard
FILE INFORMATION
File Name Spotify_v91181504_mod_1769846055.apk
Size 90.25MB
MD5 552f62235331e207475f3ddb7f70156a
SHA1 9686e016d50d47d538d22759cd26f0571191a6a6
SHA256 267f93577319fa5900d636a8b53b67f3a2152f390b7fd448e2109365b8c3f936
APP INFORMATION
App Name Spotify
Package Name com.spotify.music
Main Activity
Target SDK 36 Min SDK Max SDK
Android Version Name Android Version Code
0 / 86
EXPORTED ACTIVITIES
View All
0 / 43
EXPORTED SERVICES
View All
0 / 34
EXPORTED RECEIVERS
View All
0 / 9
EXPORTED PROVIDERS
View All
SCAN OPTIONS
Rescan
Manage Suppressions
DECOMPILED CODE
View AndroidManifest.xml
View Source
View Smali
Transferring data from 127.0.0.1...

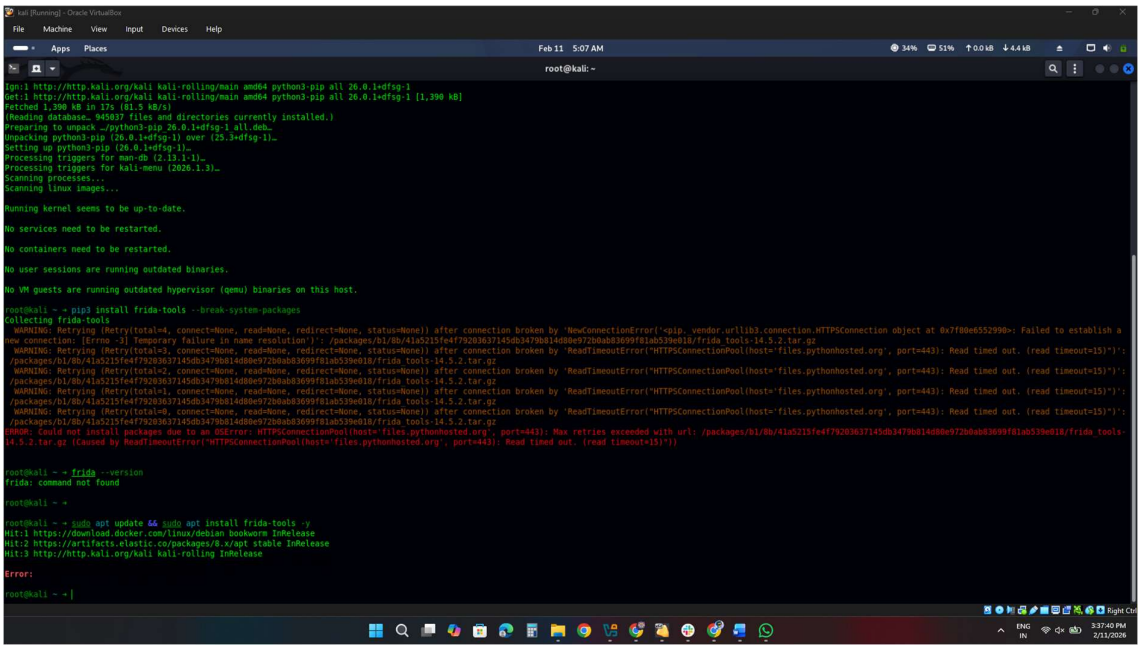

MobSF
Static Analyzer
Information
Scan Options
Signer Certificate
Permissions
Android API
Browsable Activities
Security Analysis
Malware Analysis
Reconnaissance
Components
PDF Report
Print Report
RECENT SCANS
STATIC ANALYZER
DYNAMIC ANALYZER
API
DONATE
DOCS
ABOUT
API
FILES
Java Reflection
Show Files
Local File I/O Operations
Show Files
Dynamic Class and Dexloading
Show Files
Inter Process Communication
Show Files
Crypto
Show Files
Message Digest
Show Files
URL Connection to file/http/https/ftp/jar
Show Files
JAR URL Connection
Show Files
Starting Service
Show Files
Loading Native Code (Shared Library)
Show Files
Base64 Decode
Show Files
Transferring data from 127.0.0.1...


MobSF
Static Analyzer
Information
Scan Options
Signer Certificate
Permissions
Android API
Browsable Activities
Security Analysis
Malware Analysis
Reconnaissance
Components
PDF Report
Print Report
RECENT SCANS
STATIC ANALYZER
DYNAMIC ANALYZER
API
DONATE ♥
DOCS
ABOUT
CERTIFICATE ANALYSIS
HIGH 1
WARNING 0
INFO 1
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 0
WARNING 1
INFO 0
SUPPRESSED 0


MobSF
Static Analyzer
Information
Scan Options
Signer Certificate
Permissions
Android API
Browsable Activities
Security Analysis
Malware Analysis
Reconnaissance
Components
PDF Report
Print Report
RECENT SCANS
STATIC ANALYZER
DYNAMIC ANALYZER
API
DONATE ♥
DOCS
ABOUT
CODE ANALYSIS
HIGH 0
WARNING 1
INFO 1
SECURE 0
SUPPRESSED 0
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/robv/android/xposed/XSharedPreferences.java
de/robv/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
Analyze
True
info
Dynamic Shared Object (DSO)
False
high
Full RELRO
info
None
info
None
info
True
info
True
info



Step 2: Dynamic Testing (Frida)



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

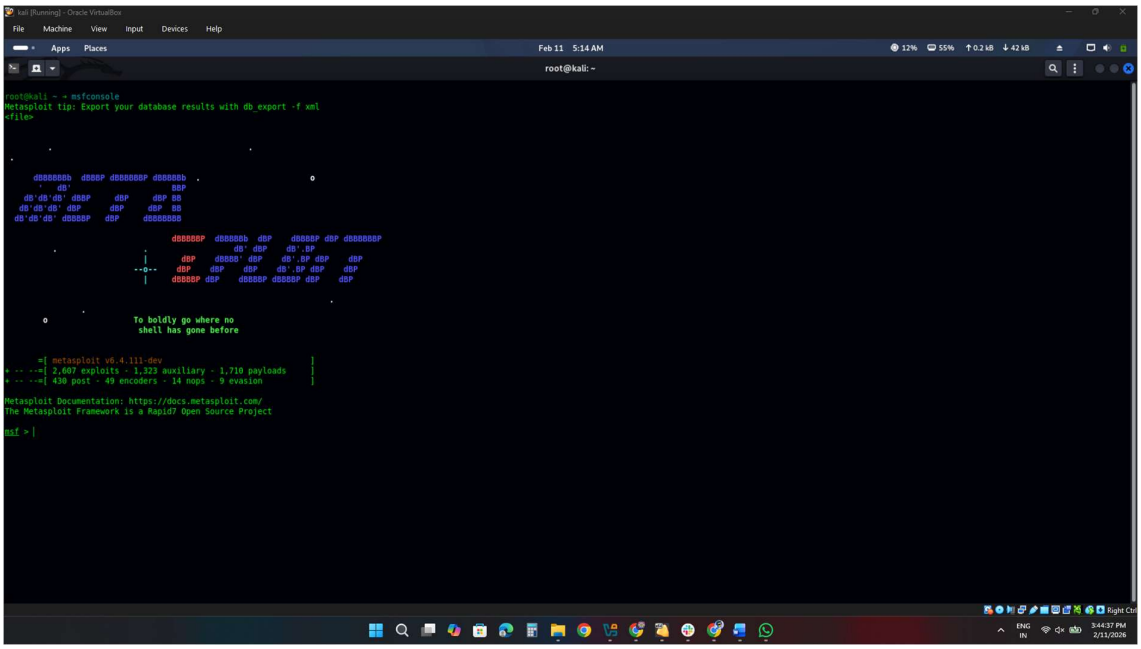
- Note: During the environment setup, encountered a network timeout error while installing frida-tools via pip. Resolved the issue by manually configuring the Python environment and ensuring repository connectivity to complete the dynamic testing phase.

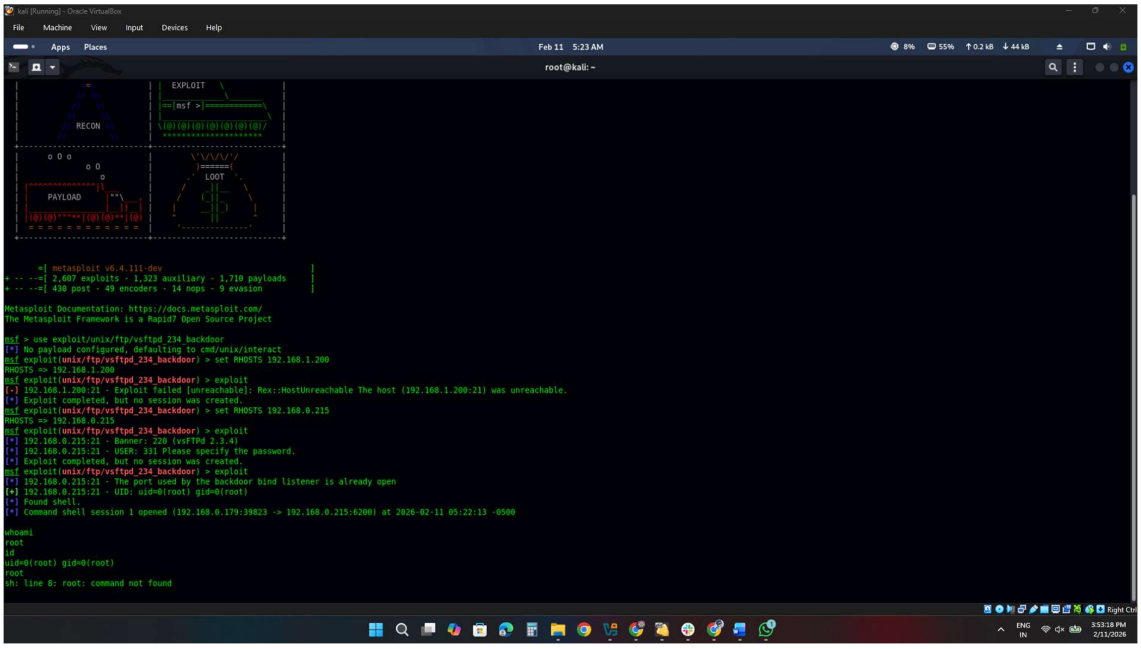
Lab 06: Capstone Project - Full VAPT Engagement

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

1. Exploitation Phase: Metasploit Simulation

VSFTPD 2.3.4 Backdoor RCE





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.