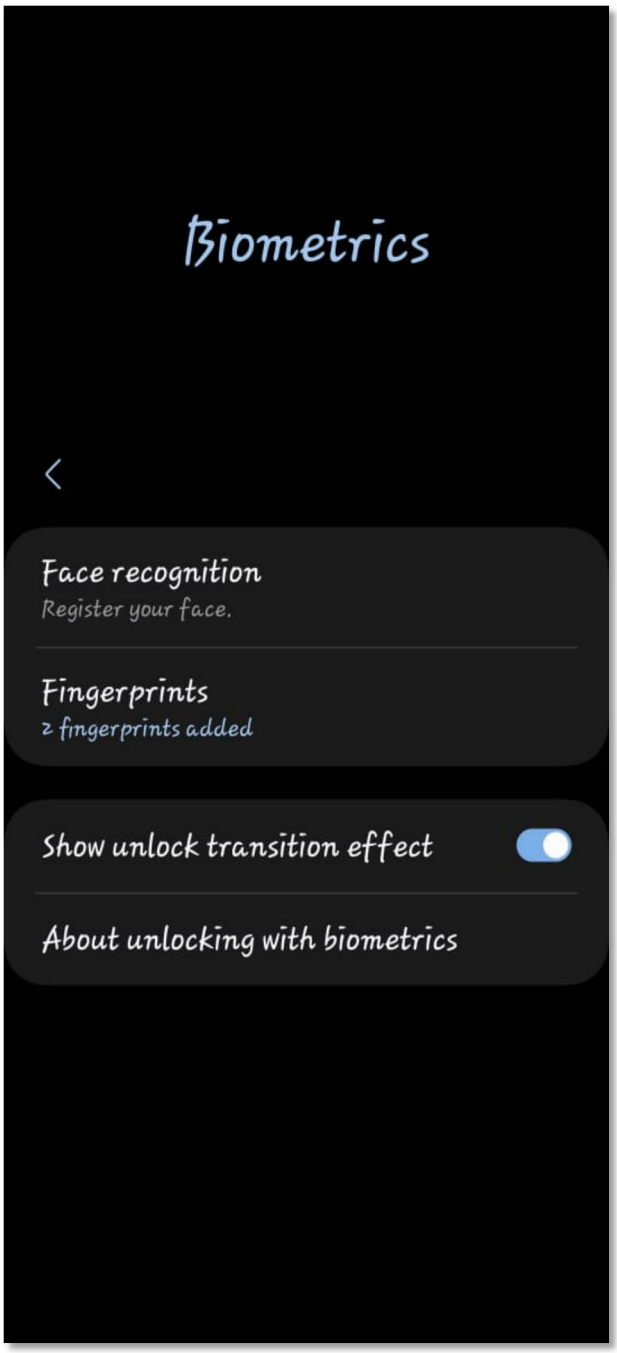
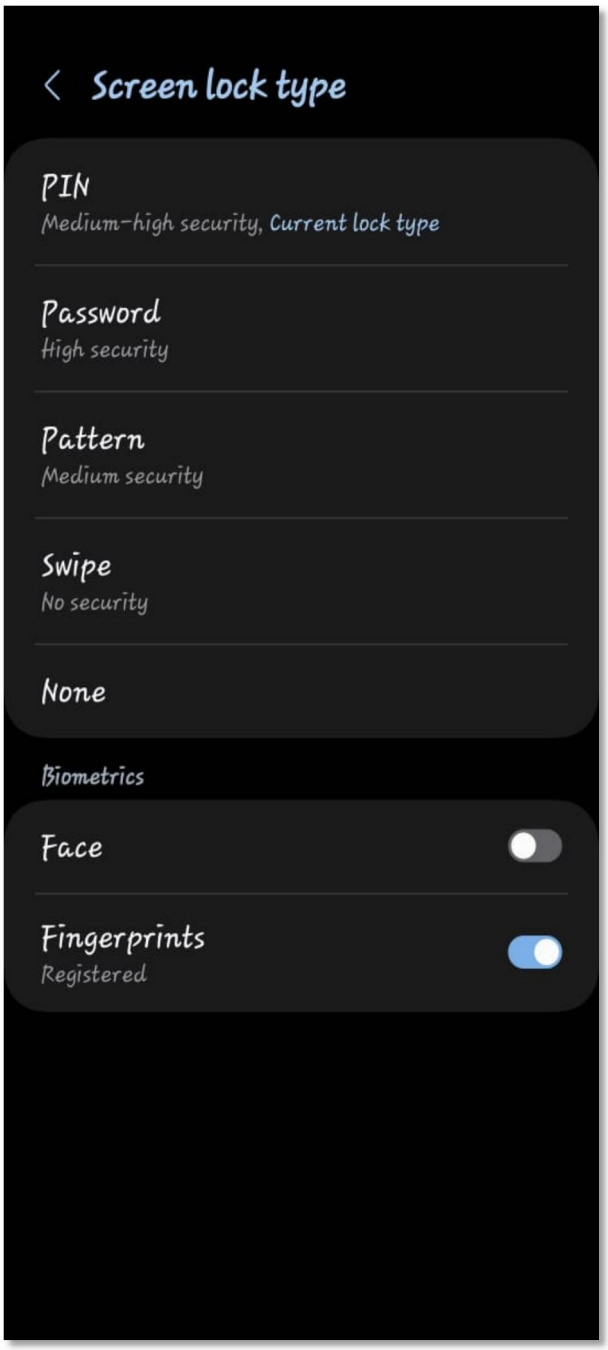
PRACTICAL 1

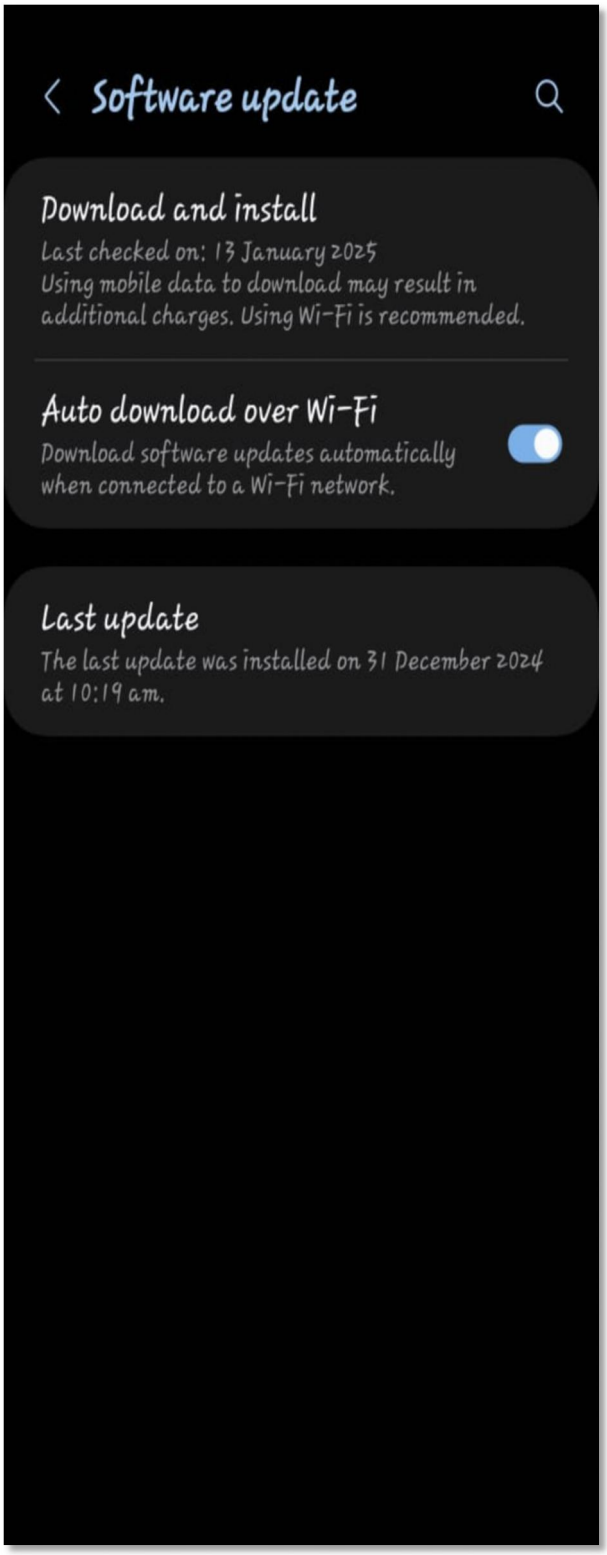
AIM: Setting up a secure mobile device configuration and hardening the OS and applications.

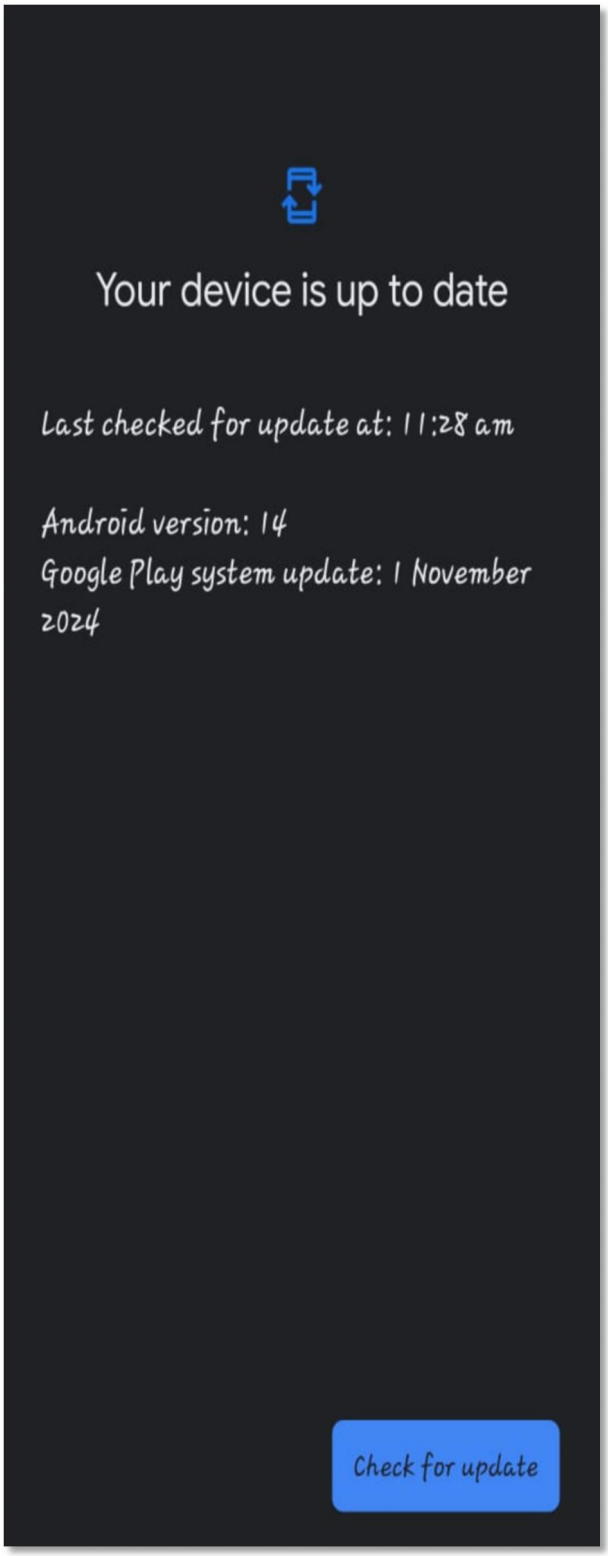
1. Use strong passwords: Use a strong password or PIN that's at least eight characters long and contains alphanumeric characters. You can also use biometric features like fingerprint authenticators.



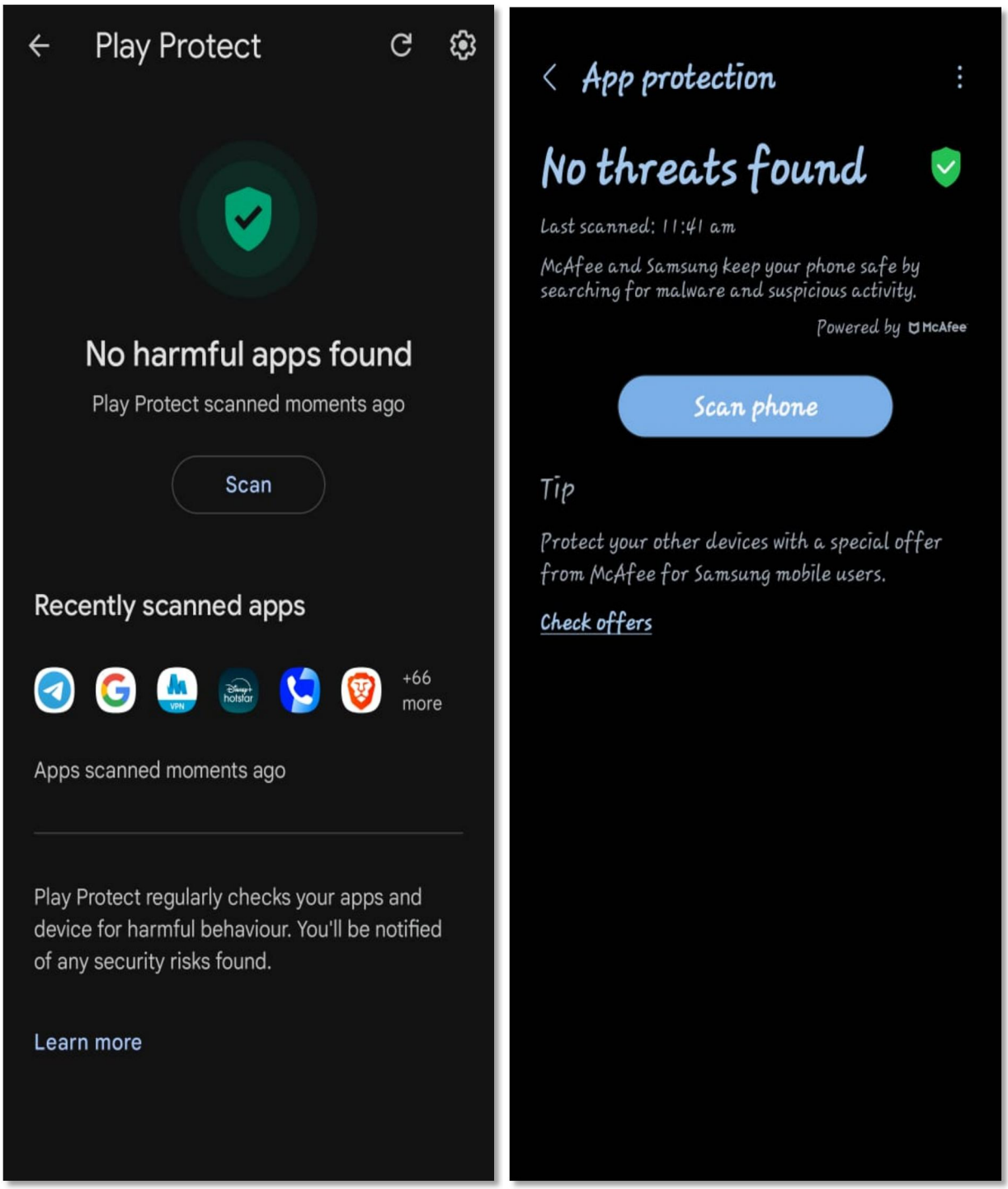


2. **Enable automatic updates:** Configure your device to automatically update your operating system and applications.

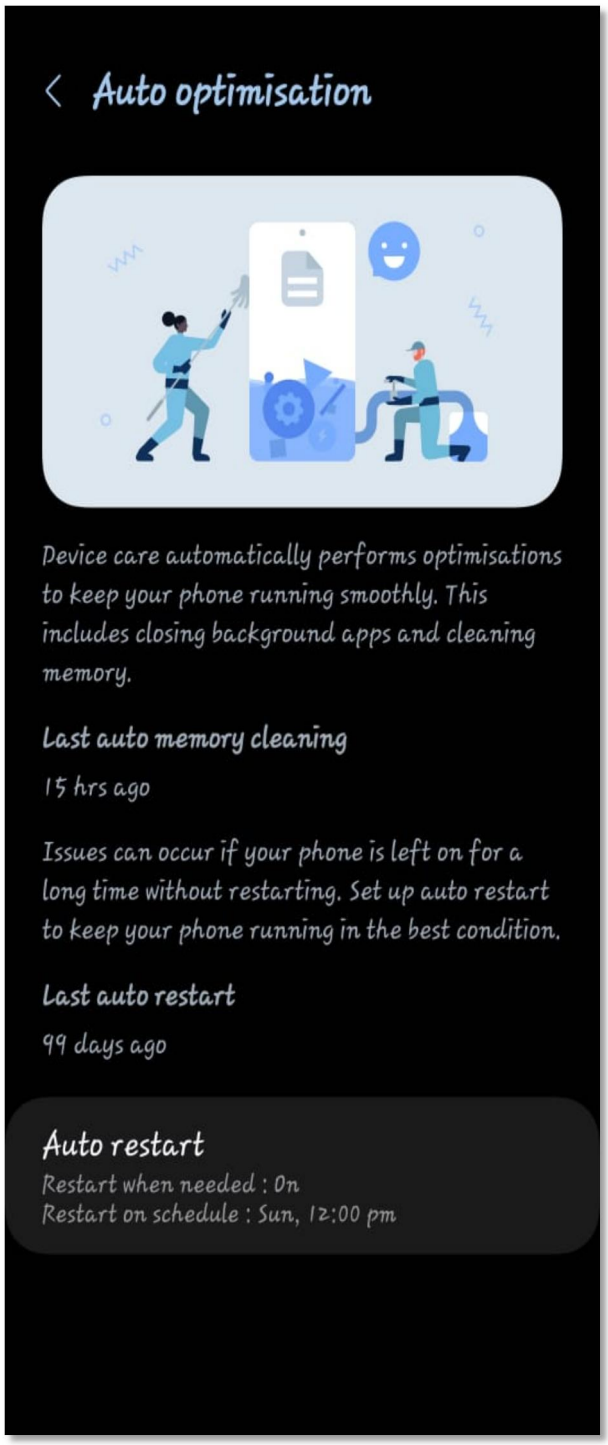




3. Enable Scan and App Protection: Scan the device and apps regularly and enable automatic scanning.



4. Enable Auto optimization: Enable auto optimization into mobile device for device care automatically performs optimizations to keep your phone running smoothly. This includes closing background apps and cleaning memory.



Conclusion:

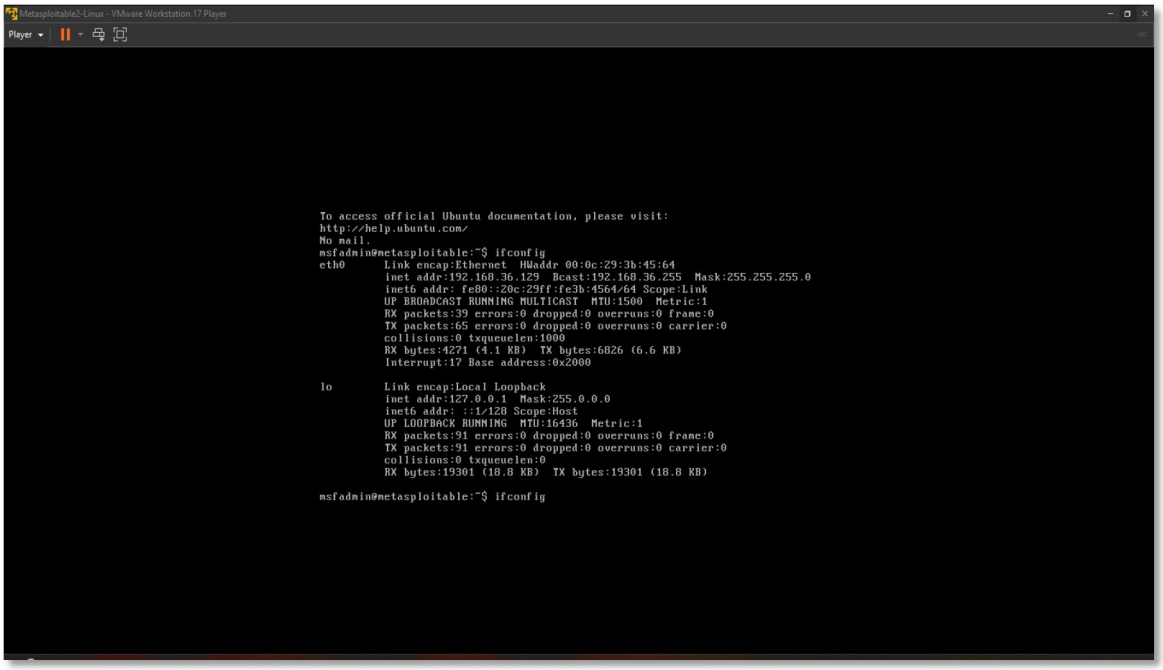
Setting up a secure mobile device configuration and hardening the operating system (OS) and applications are essential steps in protecting sensitive data and maintaining the integrity of mobile environments. By implementing security best practices such as enforcing strong authentication, regularly updating software, minimizing app permissions, encrypting data, and disabling unnecessary services organizations and individuals can significantly reduce the attack surface of mobile devices.

PRACTICAL 2

AIM: Conduct vulnerability assessments and penetration testing of mobile and IoT devices.

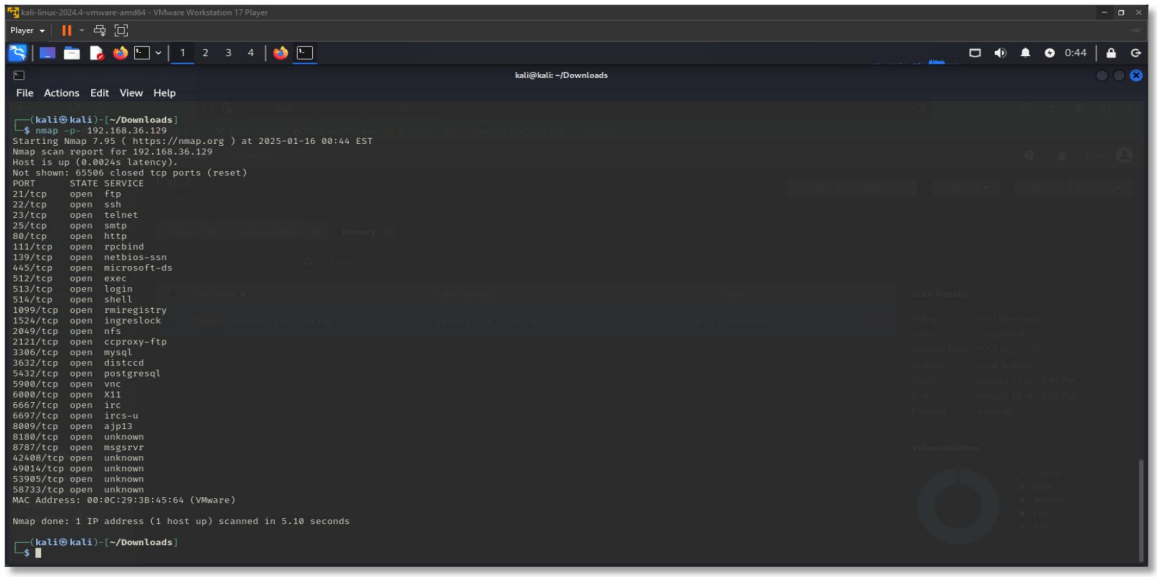
Step-1:

Find your IP address in metasploitable and scan that IP address in Nessus to scan that IP address to find the vulnerabilities.



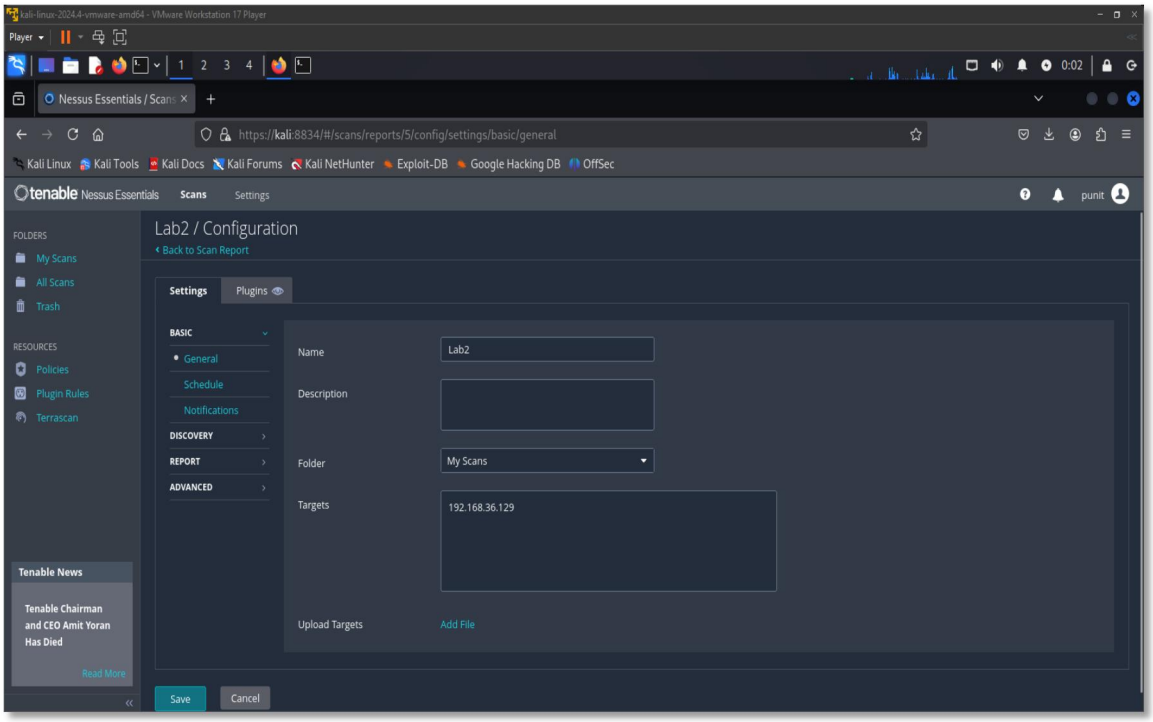
Step-2:

Scan Target IP of metasploitable Using Nmap to see the open ports.



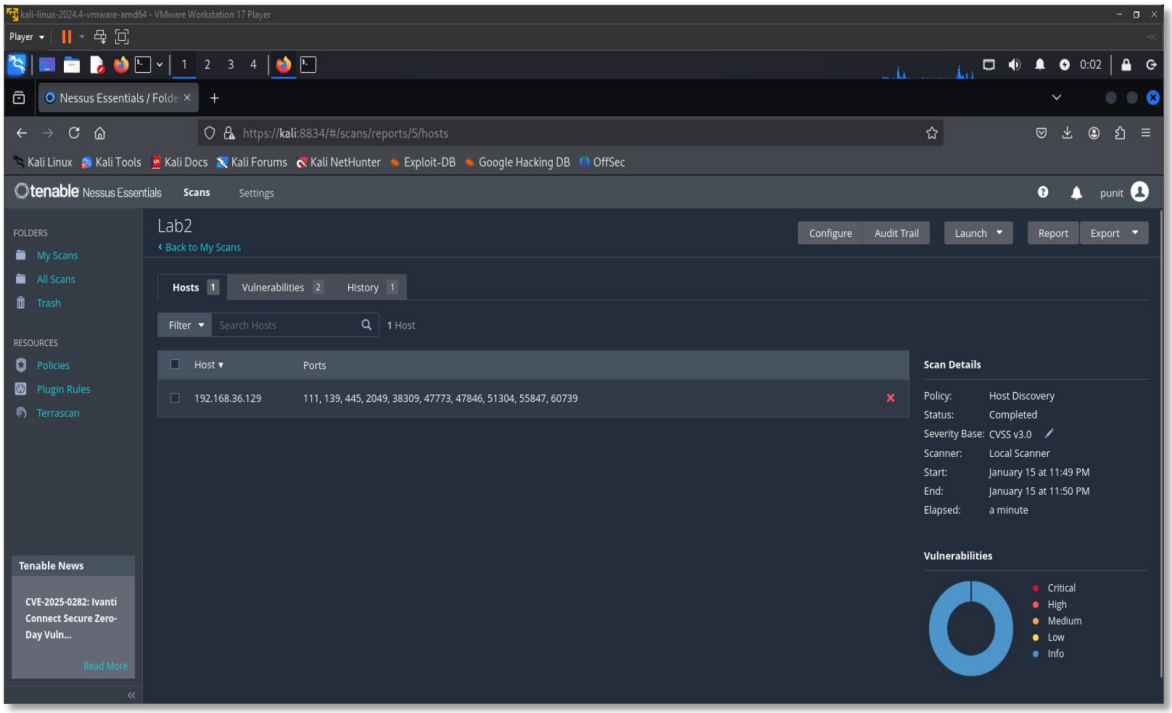
Step-3:

In Nessus Go to Host Discovery and enter the target IP that you want to scan and launch to find Vulnerabilities.



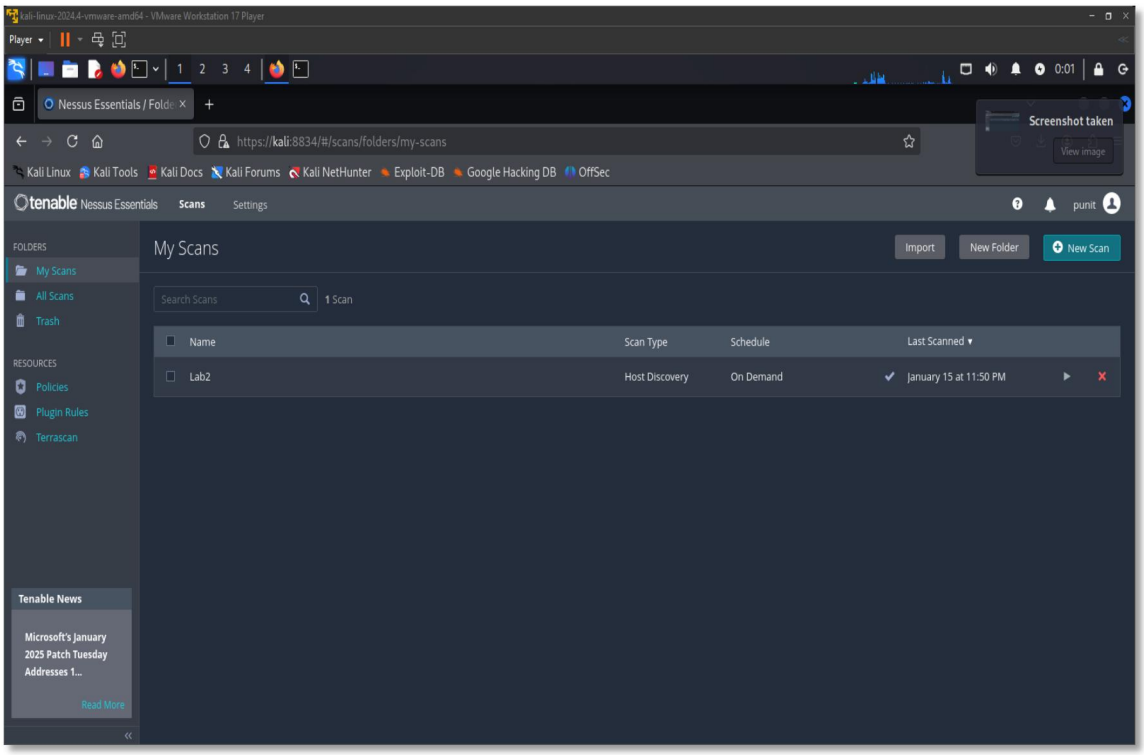
Step-4:

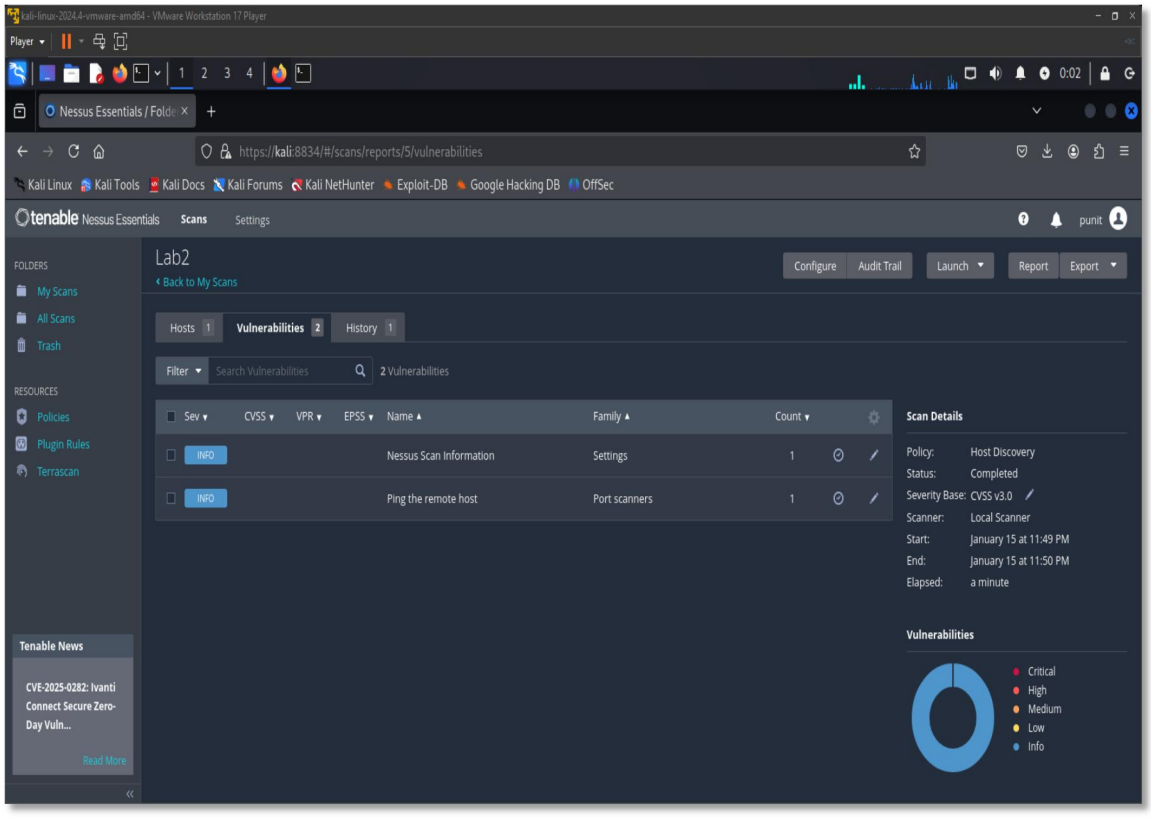
After that it will start to scan and after the completion it will show the vulnerabilities that have in the target IP or hosts.

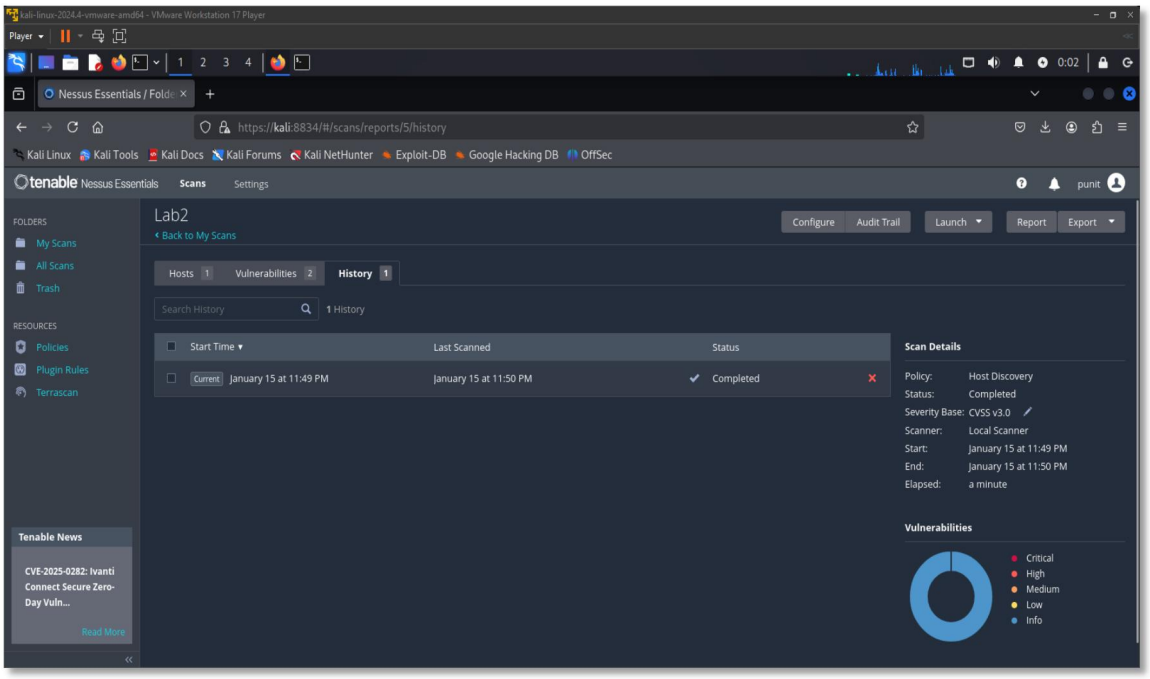


Step-5:

Now you are able to see the vulnerabilities that are scanned on that host or target IP and you are able to see the info of that vulnerabilities.







Conclusion:

Conducting vulnerability assessments and penetration testing on mobile and IoT devices is essential to identifying and mitigating security risks in today's interconnected world. These assessments help uncover vulnerabilities in device firmware, applications, network communications, and access controls that could be exploited by attackers. By proactively testing and analyzing potential attack surfaces, organizations can strengthen their overall security posture, ensure compliance with industry standards, and protect sensitive user data. As mobile and IoT ecosystems continue to evolve, regular testing and adaptive security measures remain critical to safeguarding both users and infrastructure.

PRACTICAL 3

AIM: Analyzing network traffic on mobile and IoT devices to identify potential security risks.

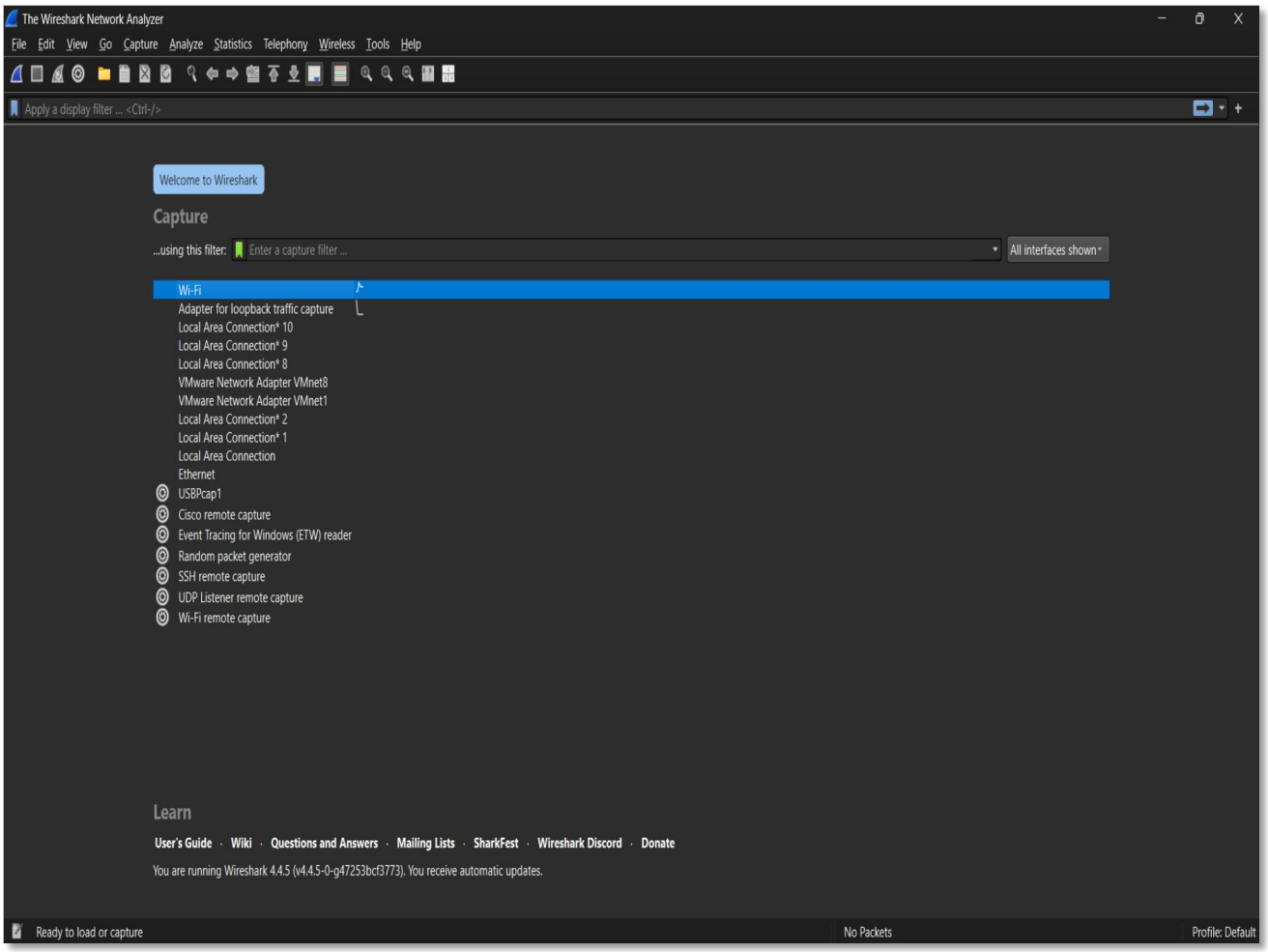
To understand how to capture and analyze network traffic from mobile devices (and simulate IoT traffic) to detect potential security threats.

Tools Used:

Wireshark (for packet capture and analysis)
Android phone (connected to same Wi-Fi network)
Windows/Linux PC (running Wireshark)
No real IoT devices used – IoT traffic is simulated

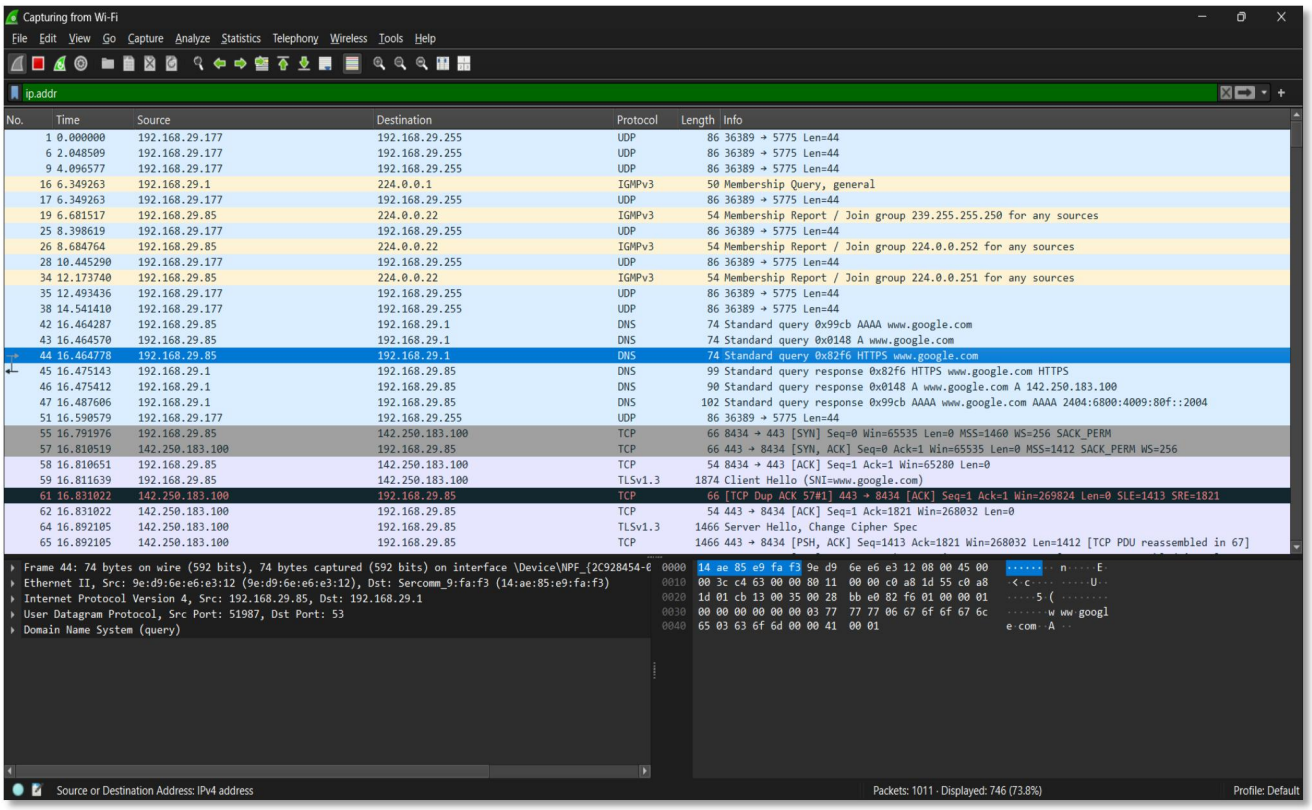
Step 1:

Choose Interface To Analysis:



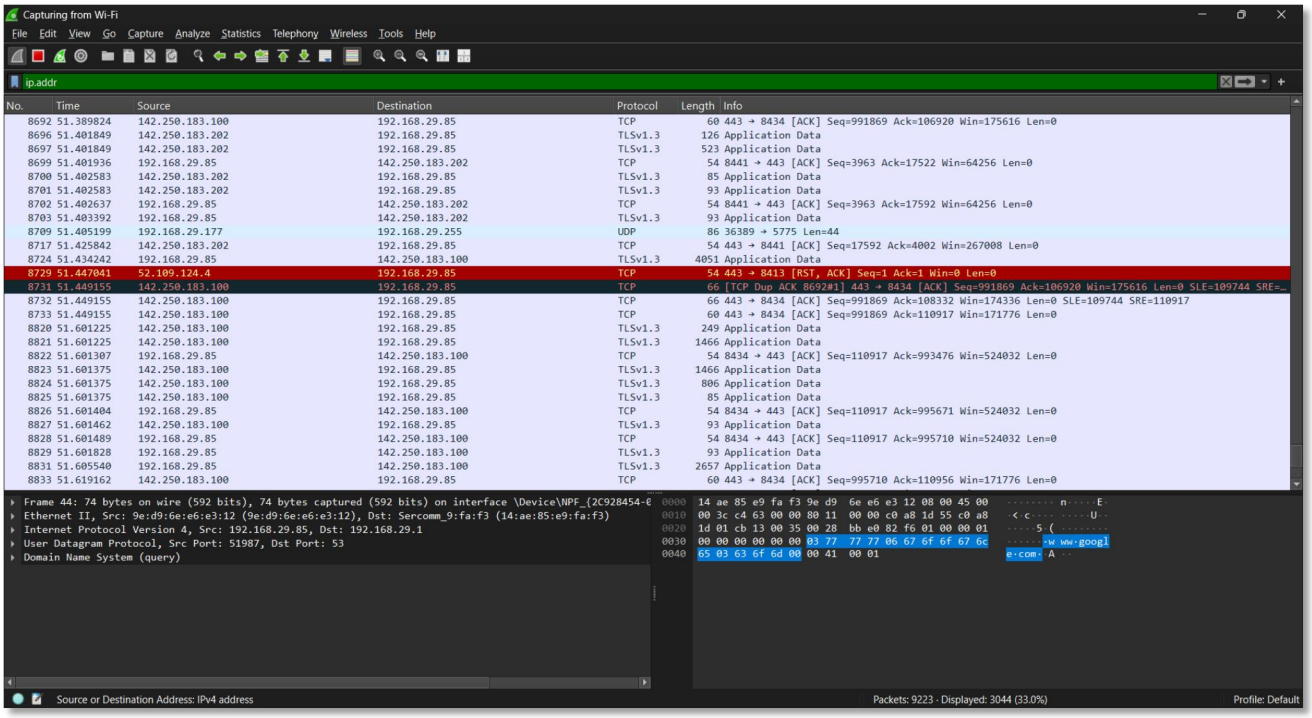
Step 2:

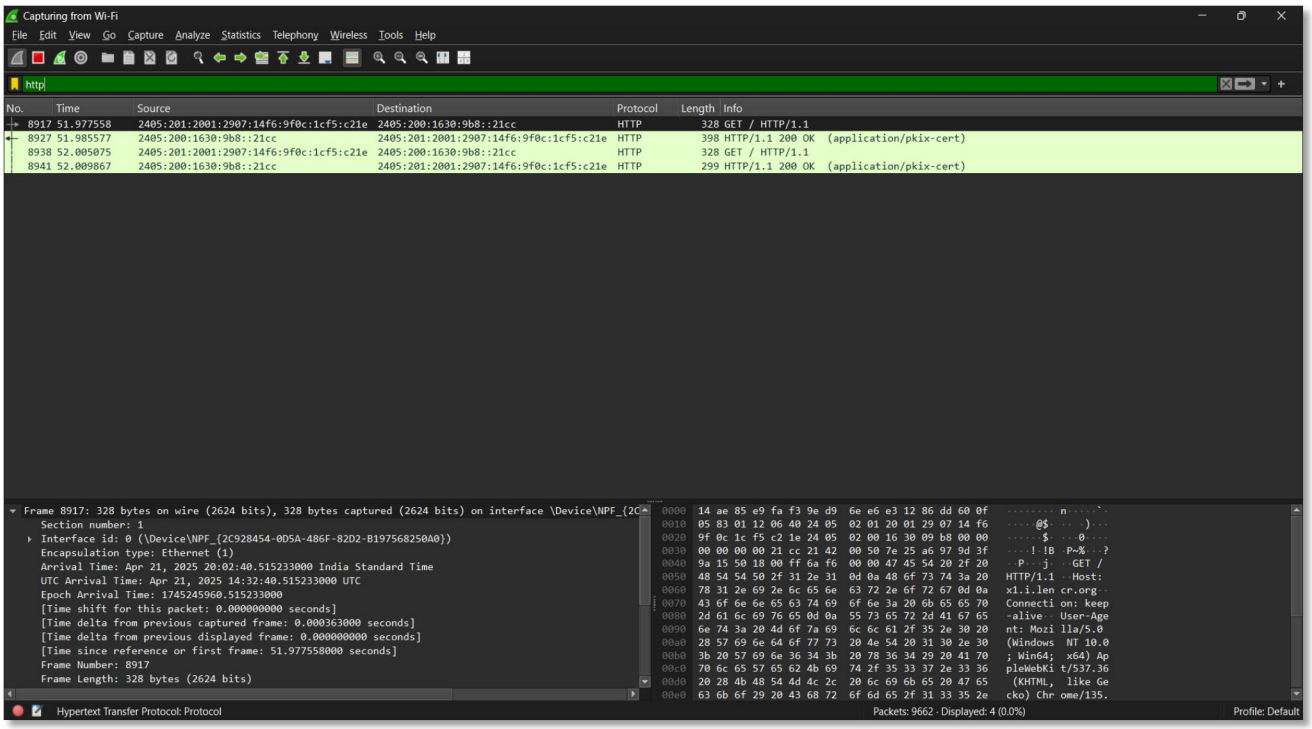
Analyze The Traffic:



Step 3:

Check The Malicious IP's:





Conclusion:

Analyzing network traffic on mobile and IoT devices is critical in identifying and mitigating potential security risks in today's interconnected world. These devices often operate with limited security measures and are frequent targets for malicious actors due to their ubiquitous presence and constant connectivity. Through detailed traffic analysis, unusual patterns, unauthorized data transfers, and potential breaches can be detected early, allowing for proactive defense strategies. Implementing robust traffic monitoring tools and adopting machine learning techniques can significantly enhance threat detection accuracy.

PRACTICAL 4

AIM: Performing static and dynamic analysis of mobile applications to detect vulnerabilities and potential malware.

MobSF (Mobile Security Framework) is an open-source, automated security testing tool designed for static, dynamic, and malware analysis of mobile applications (Android, iOS, and Windows). It helps security researchers and developers identify vulnerabilities, misconfigurations, and privacy risks in mobile apps.

Key Features:

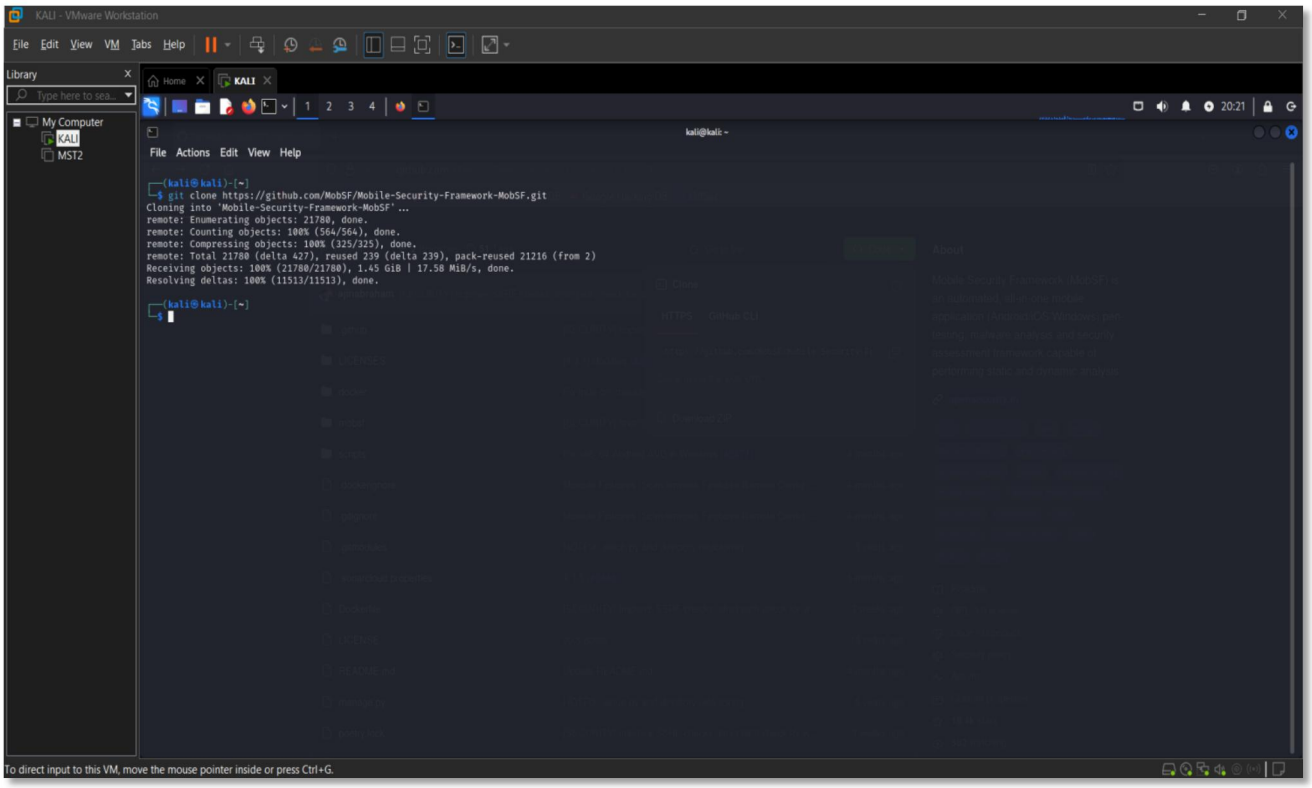
Static Analysis: Decompiles and analyzes APKs/IPAs to find security flaws without execution.
Dynamic Analysis: Runs the app in a sandbox to detect runtime threats.
Malware Analysis: Scans for malicious code and behavior.
API Monitoring: Detects insecure API calls and data leaks.

What is MobSF?

Mobile Security Framework (MobSF) is an open-source tool used for static, dynamic, and malware analysis of mobile applications (Android, iOS, and Windows). It helps security researchers and developers identify vulnerabilities in mobile apps.

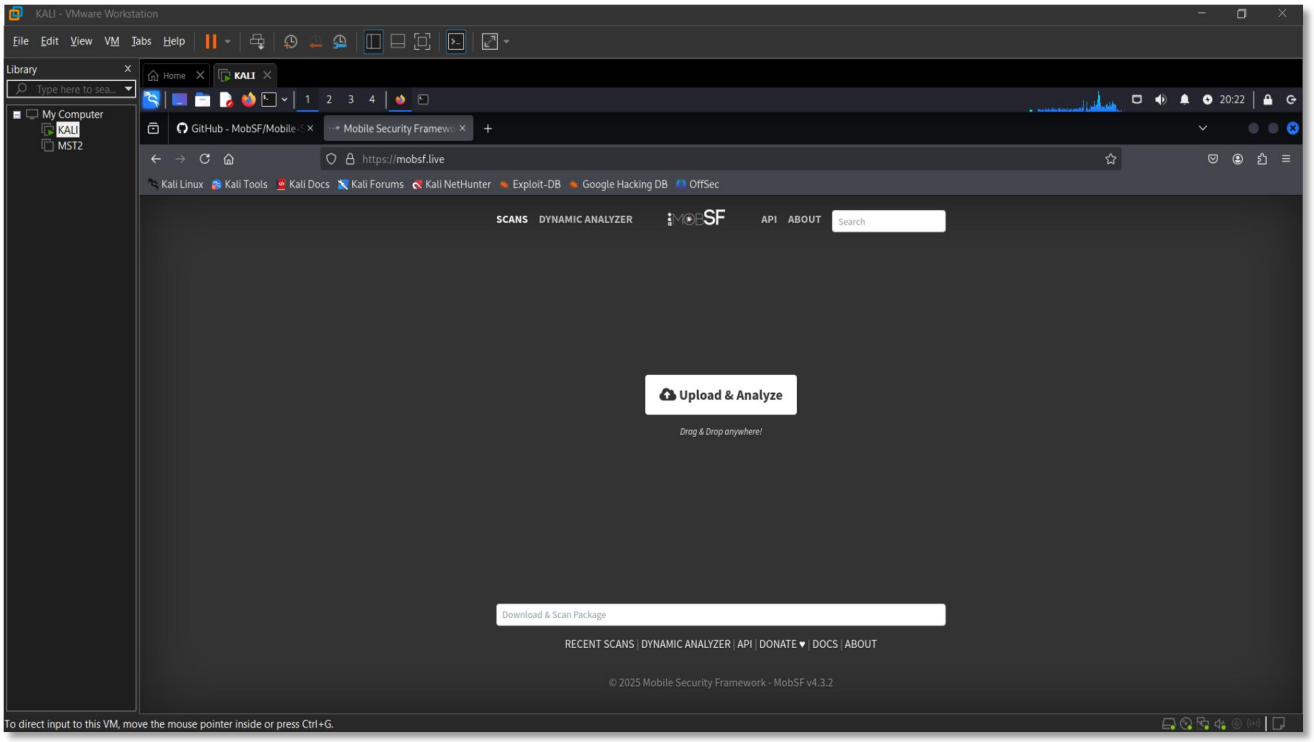
Step 1:

Git Clone The MobSF into Your System.



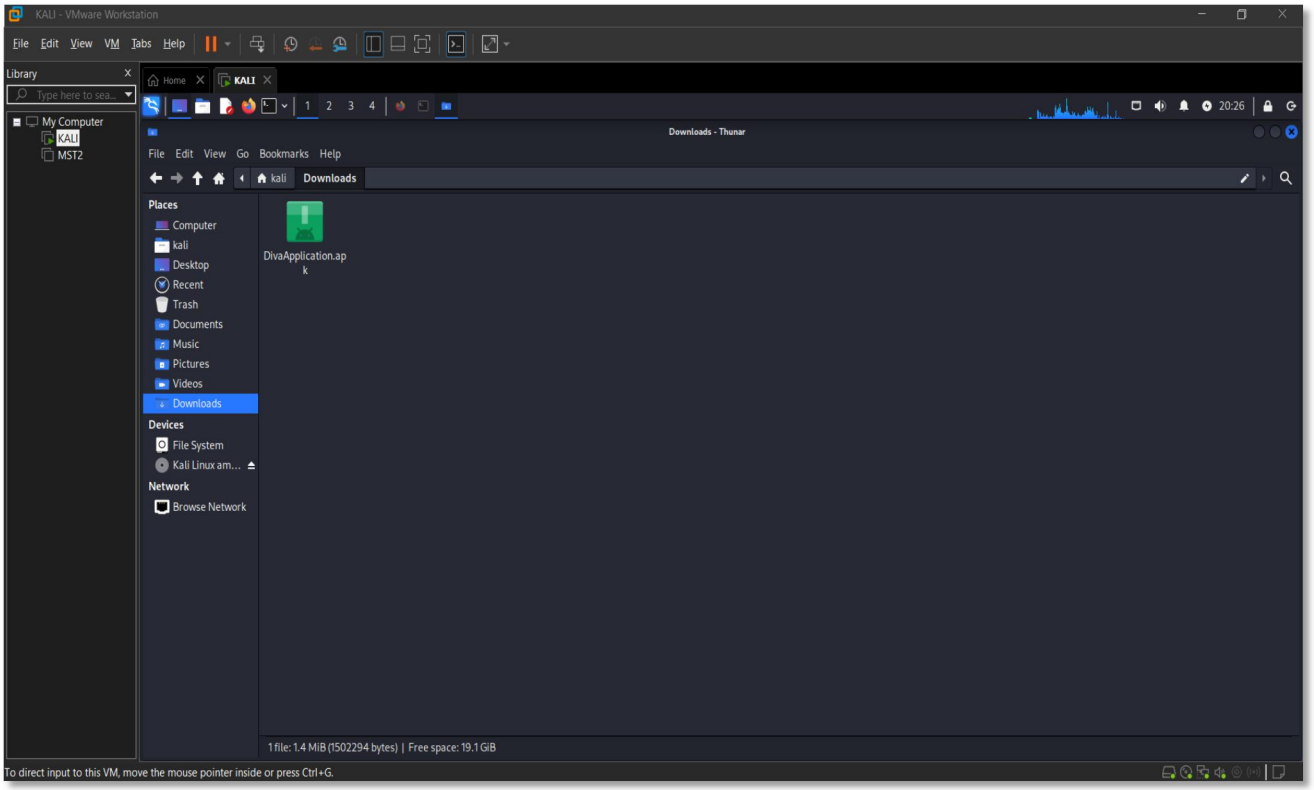
Step 2:

Open The MobSF in Browser.



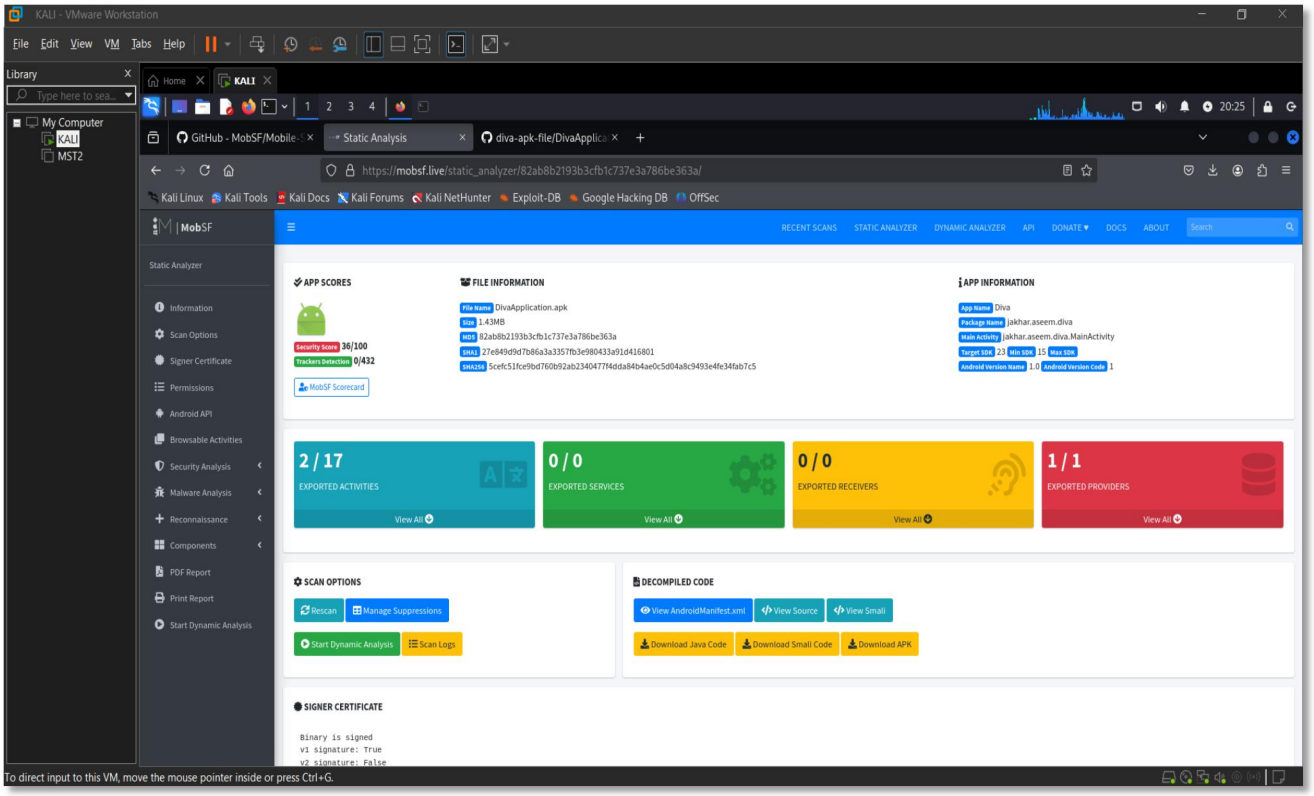
Step 3:

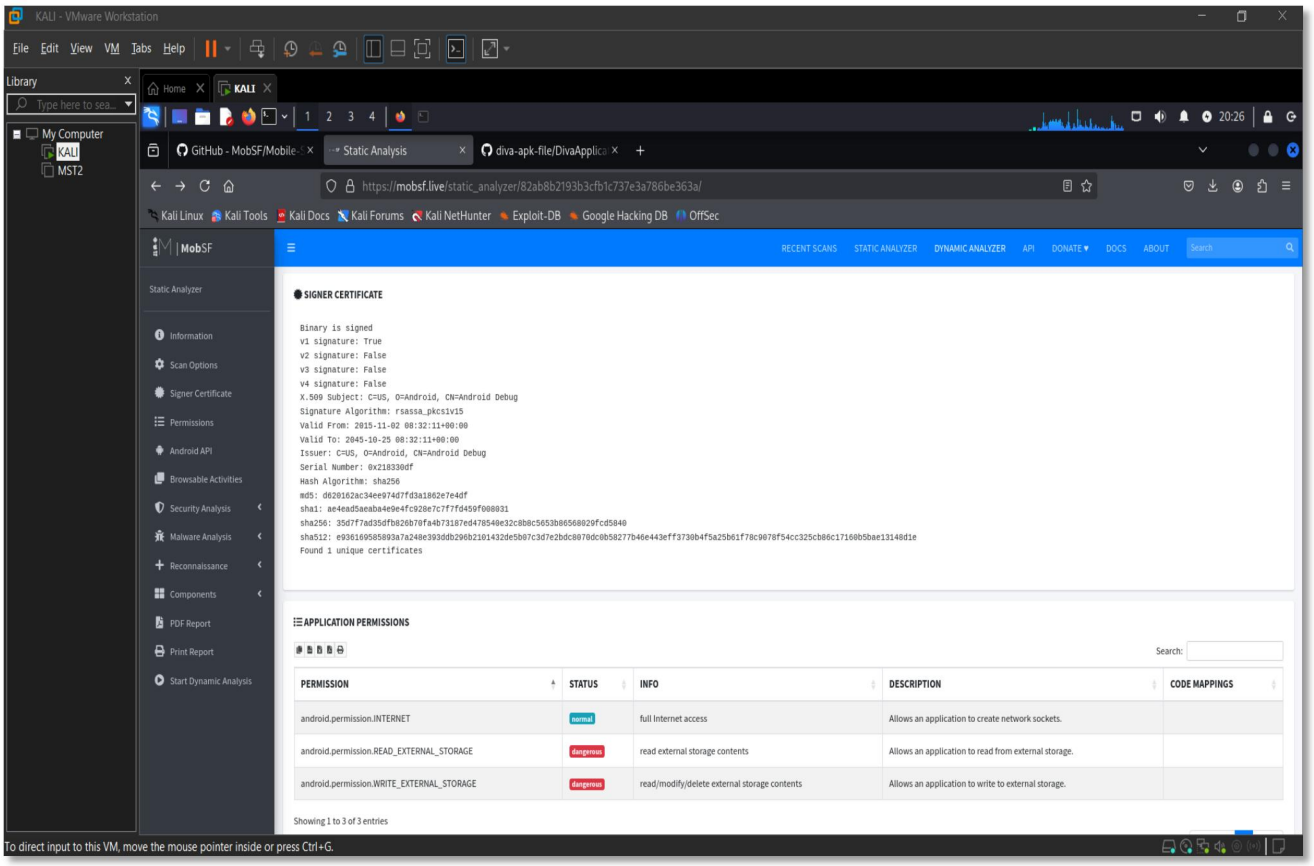
Select The APK File To Anaysis.



Step 4:

Analyze The Report.





Conclusion:

The use of Mobile Security Framework (MobSF) for performing both static and dynamic analysis proves to be an effective approach for identifying vulnerabilities and detecting potential malware in mobile applications. MobSF's comprehensive suite of features enables efficient scanning of APKs, IPAs, and source code, providing developers and security analysts with detailed insights into security flaws, privacy issues, and suspicious behaviors.

PRACTICAL 5

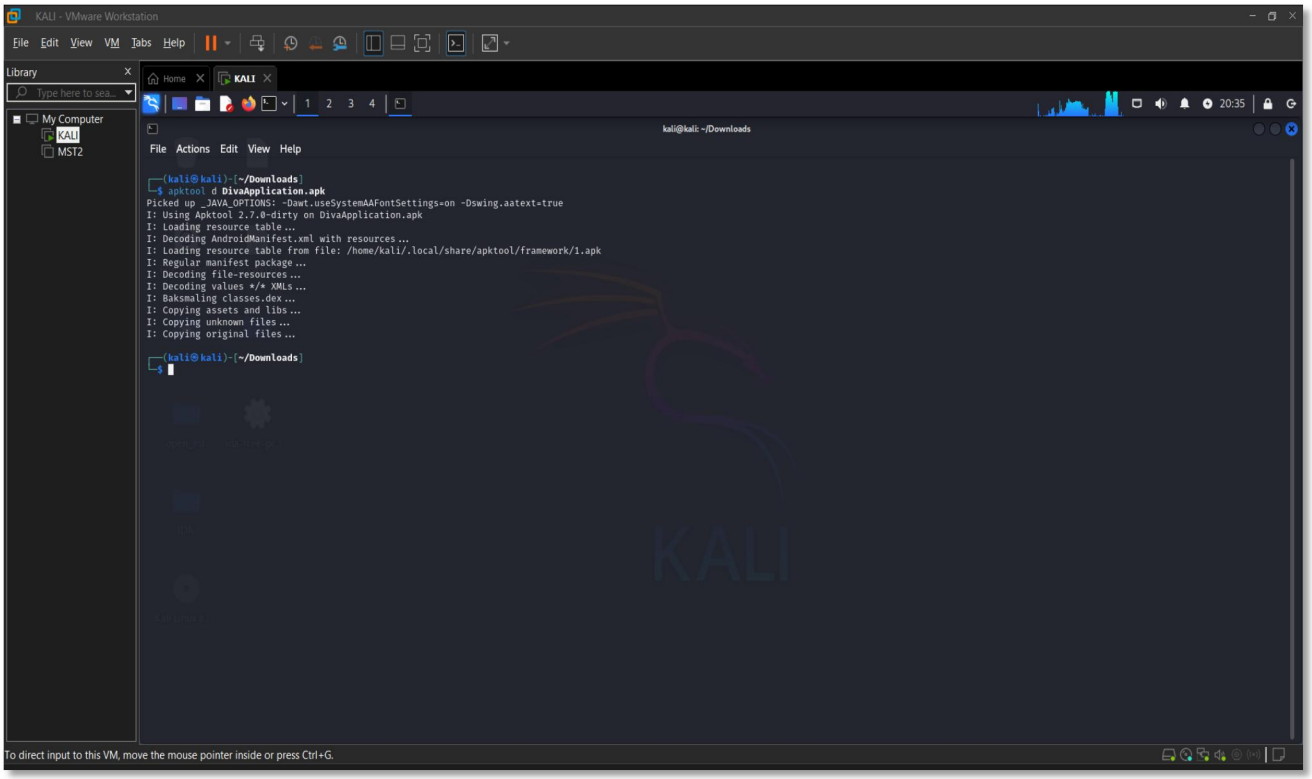
AIM: Decompiling apk by using apktoolkit.

Description:

Apktool is a powerful open-source tool that allows users to decompile and reverse engineer Android application packages (APKs). It decodes resources to nearly original form and disassembles the application's DEX (Dalvik Executable) files into Smali code. This is especially useful for understanding how an app works internally, identifying potential security vulnerabilities, and analyzing malware.

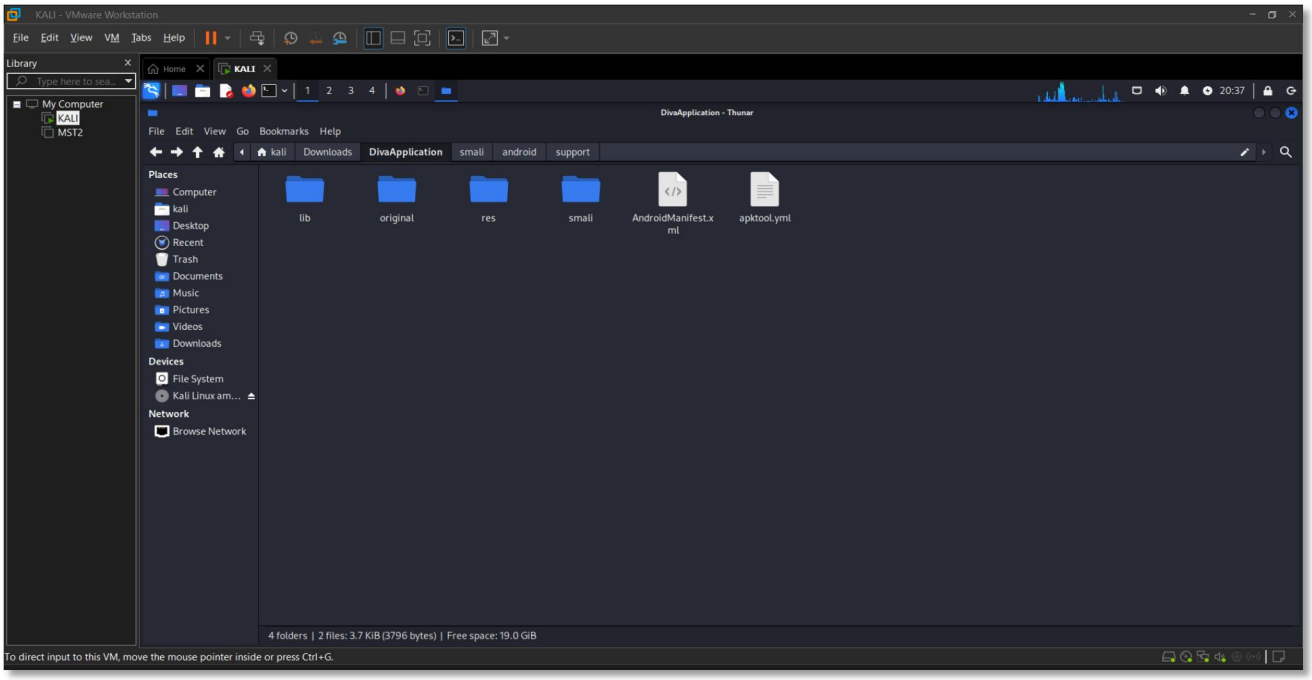
Step 1:

Install apktool and Use The apktool d <APK File>



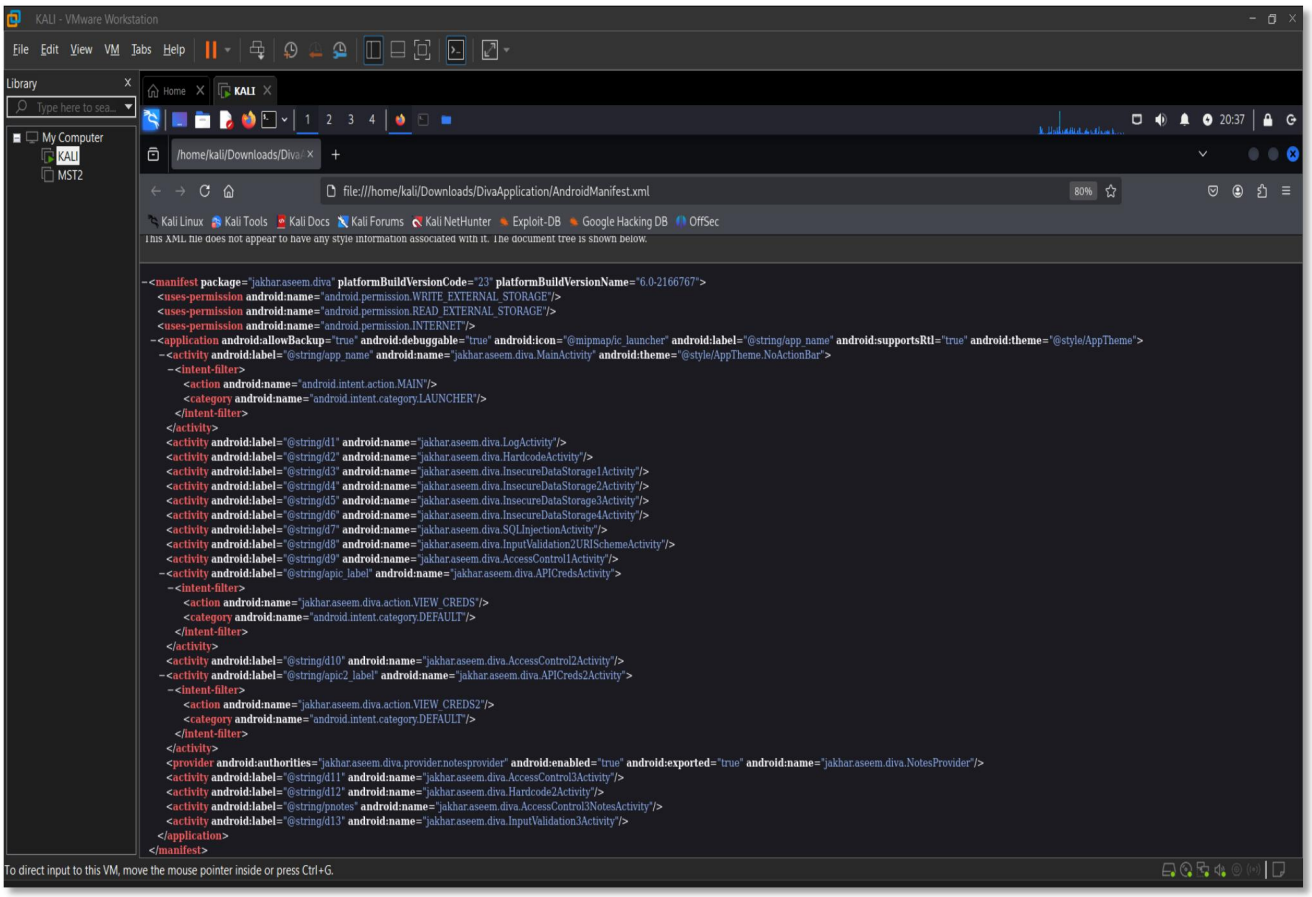
Step 2:

It Will Create The Decompile File.



Step 3:

Analyze The Decompile Code of The Given File.



Conclusion:

Decompiling APK files using Apktool is a vital technique in mobile app security analysis, penetration testing, and malware research. Apktool allows deep inspection of an app's structure, resources, and bytecode-level operations through its Smali output. While it does not convert code to high-level Java, it enables thorough investigation of app behavior and logic.

This method is invaluable for identifying vulnerabilities, hidden permissions, insecure components, or malicious code embedded within Android apps. When combined with other tools (like JADX for Java code, MobSF, etc.), Apktool plays a crucial role in static analysis and reverse engineering workflows.

PRACTICAL 6

AIM: Adb installation and connecting with emulator.

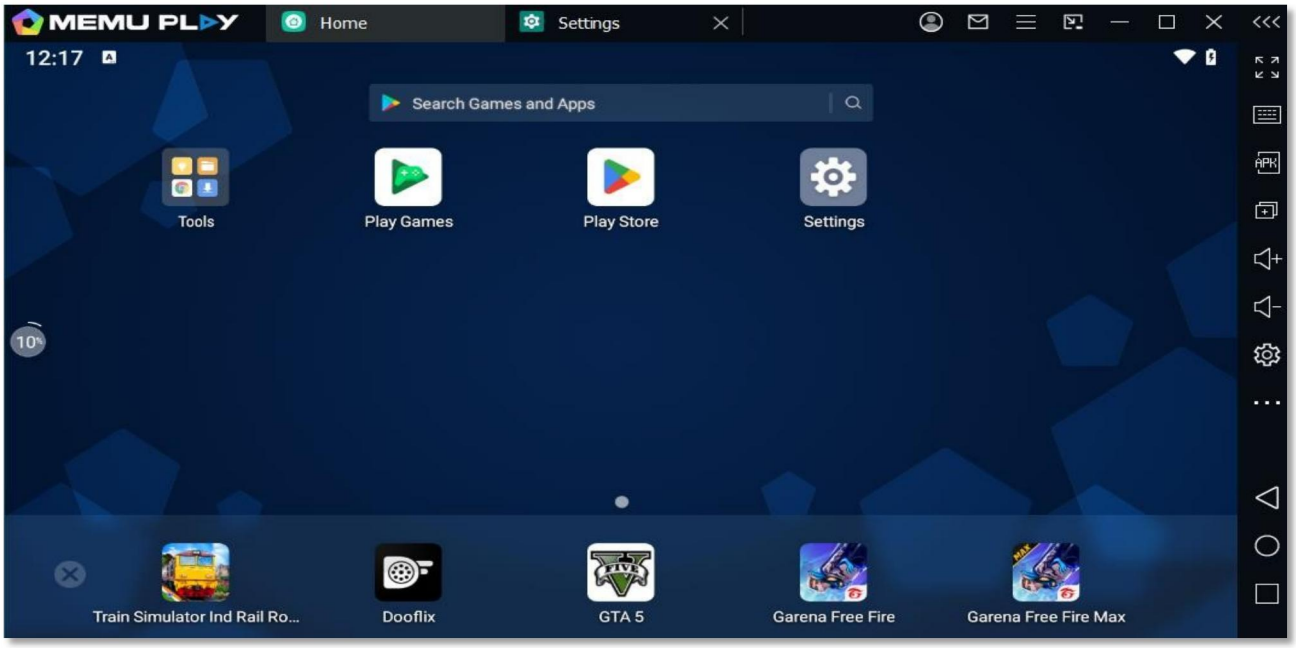
Description:

ADB (Android Debug Bridge) is a versatile command-line tool that allows you to communicate with a device (physical or emulator). It's part of the Android SDK Platform-Tools and is essential for debugging Android apps, transferring files, running shell commands, and more.

Using ADB with an emulator allows developers to simulate Android devices on their computers for app testing without needing physical hardware.

```
PS C:\Users\ACER\Downloads\platform-tools-latest-windows\platform-tools> adb connect 127.0.0.1:21503
already connected to 127.0.0.1:21503
PS C:\Users\ACER\Downloads\platform-tools-latest-windows\platform-tools> adb devices
List of devices attached
127.0.0.1:21503 device

PS C:\Users\ACER\Downloads\platform-tools-latest-windows\platform-tools>
```



Conclusion:

ADB is an essential tool for Android development and testing. Once installed, it allows seamless interaction with emulators and devices. Connecting to an emulator is simple and helps developers test their applications across different virtual devices efficiently. Mastering ADB can greatly improve your development workflow.

PRACTICAL 7

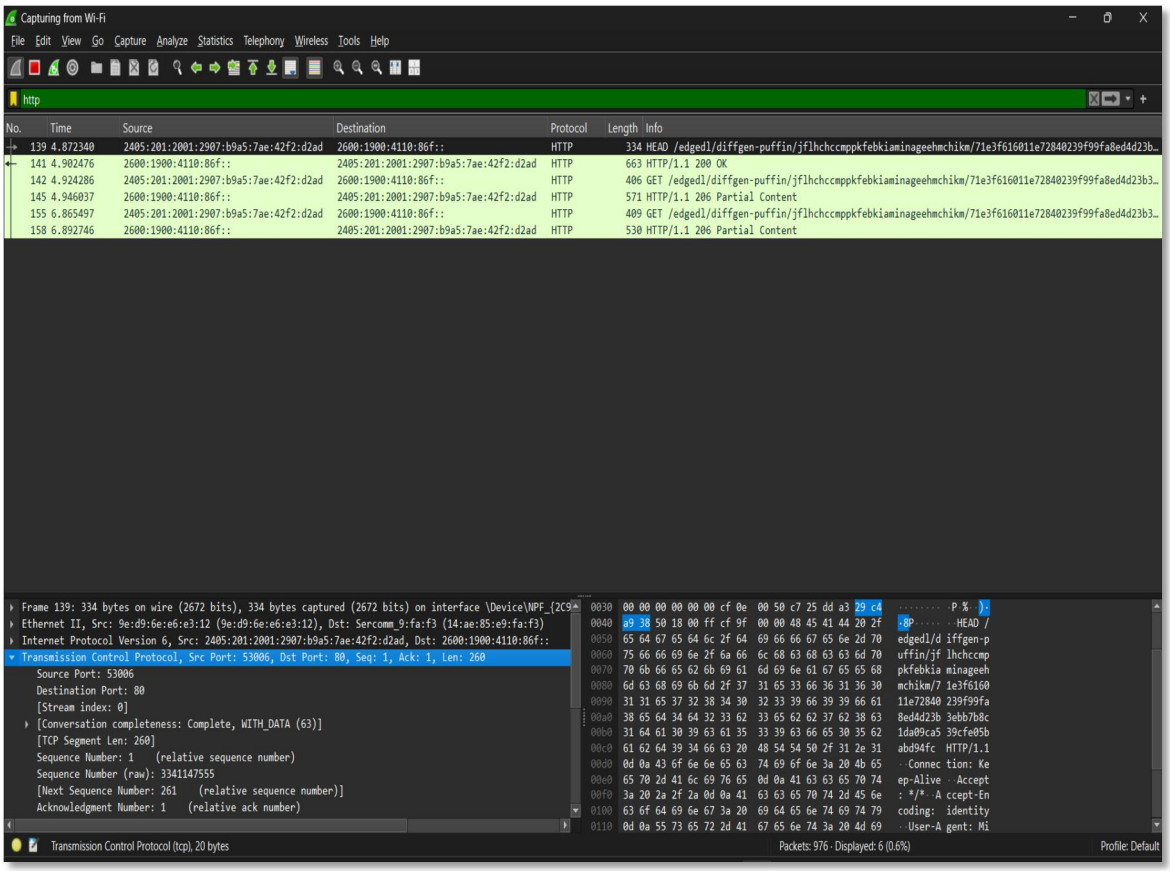
AIM: Using tools like Wireshark and tcpdump to capture and analyze mobile and IoT network traffic.

Introduction:

Wireshark and tcpdump are two of the most widely used tools for network traffic analysis. Wireshark, known for its powerful graphical interface and comprehensive packet analysis capabilities, is often favored for detailed, in-depth inspections of network data. Tcpdump, a command-line tool, is lightweight and highly effective for quick, real-time packet captures, making it ideal for mobile and IoT network environments where resources may be limited or where rapid diagnostics are needed.

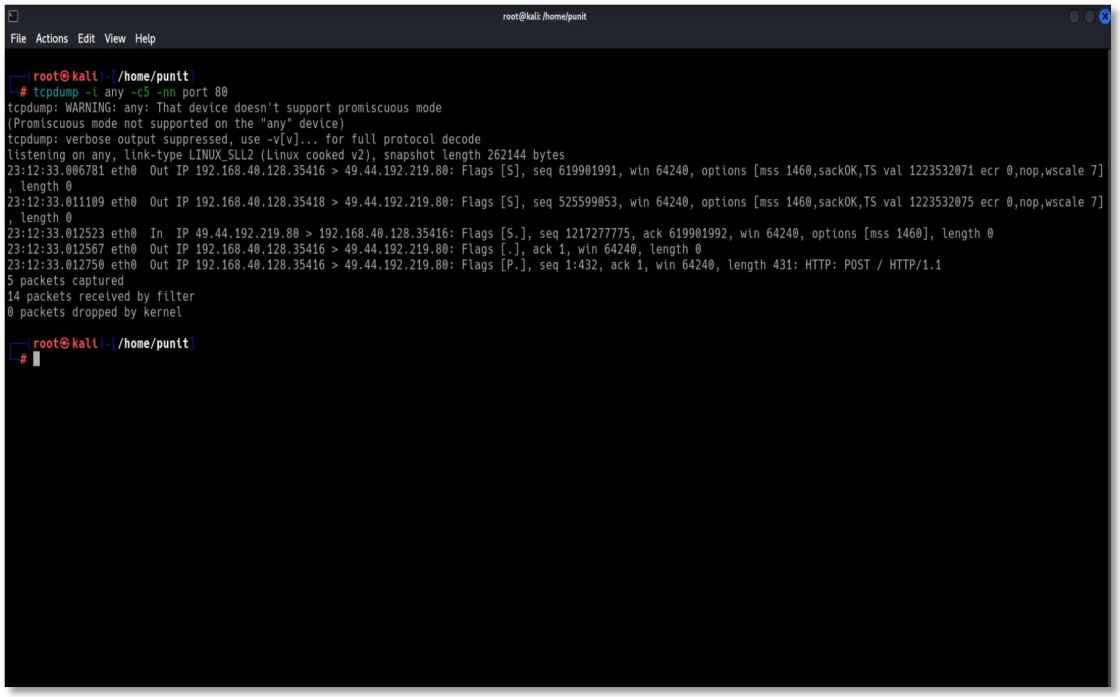
By capturing and analyzing network traffic from mobile and IoT devices, these tools provide valuable insights into the data flow, protocols, and potential vulnerabilities in the communication between devices and servers. They are indispensable in ensuring that these networks run smoothly, efficiently, and securely. This analysis is especially crucial as the volume of connected devices continues to rise, and the complexity of mobile and IoT networks grows.

WireShark Capturing:

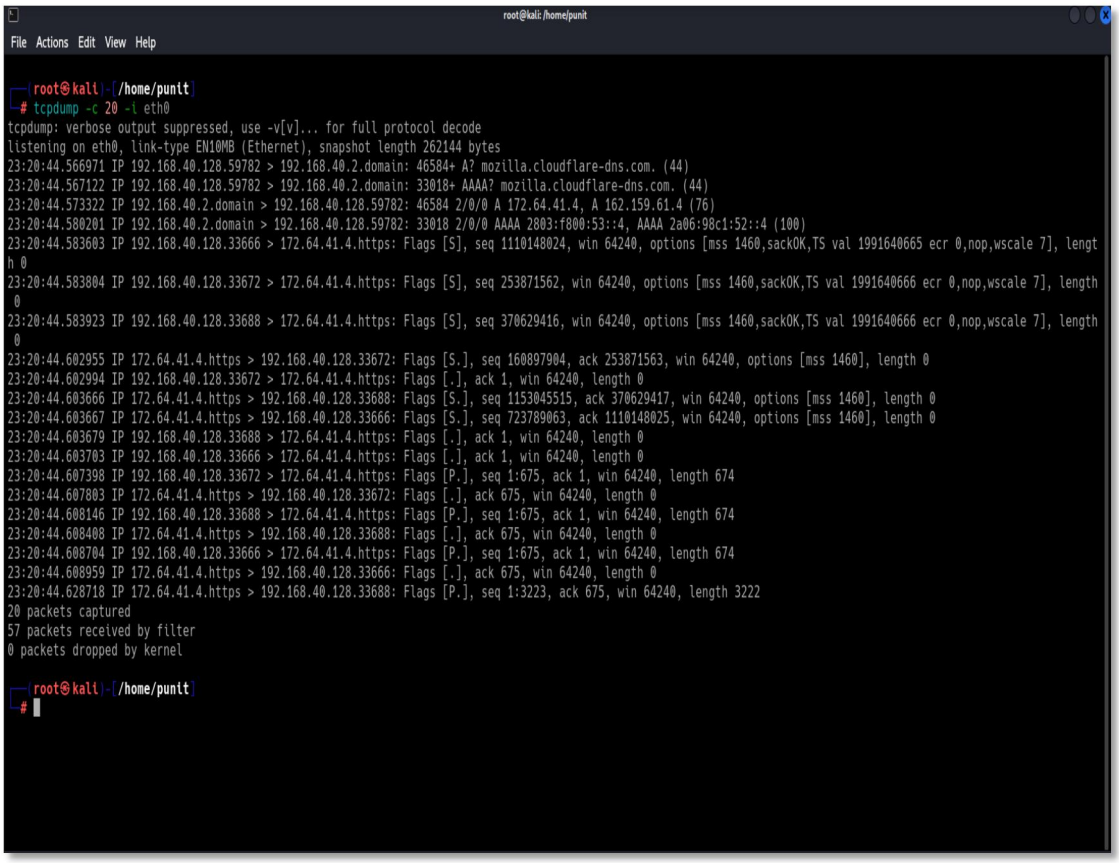


Tcpdump Capturing:

tcpdump -i any -c5 -nn port 80:



tcp -c 20 -I eth0:



Conclusion:

Using tools like Wireshark and tcpdump to capture and analyze mobile and IoT network traffic offers significant advantages for network administrators, security professionals, and developers. These tools allow for the detailed examination of the data exchanged between devices and servers, helping to diagnose issues, monitor performance, and ensure network security.

PRACTICAL 8

AIM: Re-compiling the apk using apktool kit.

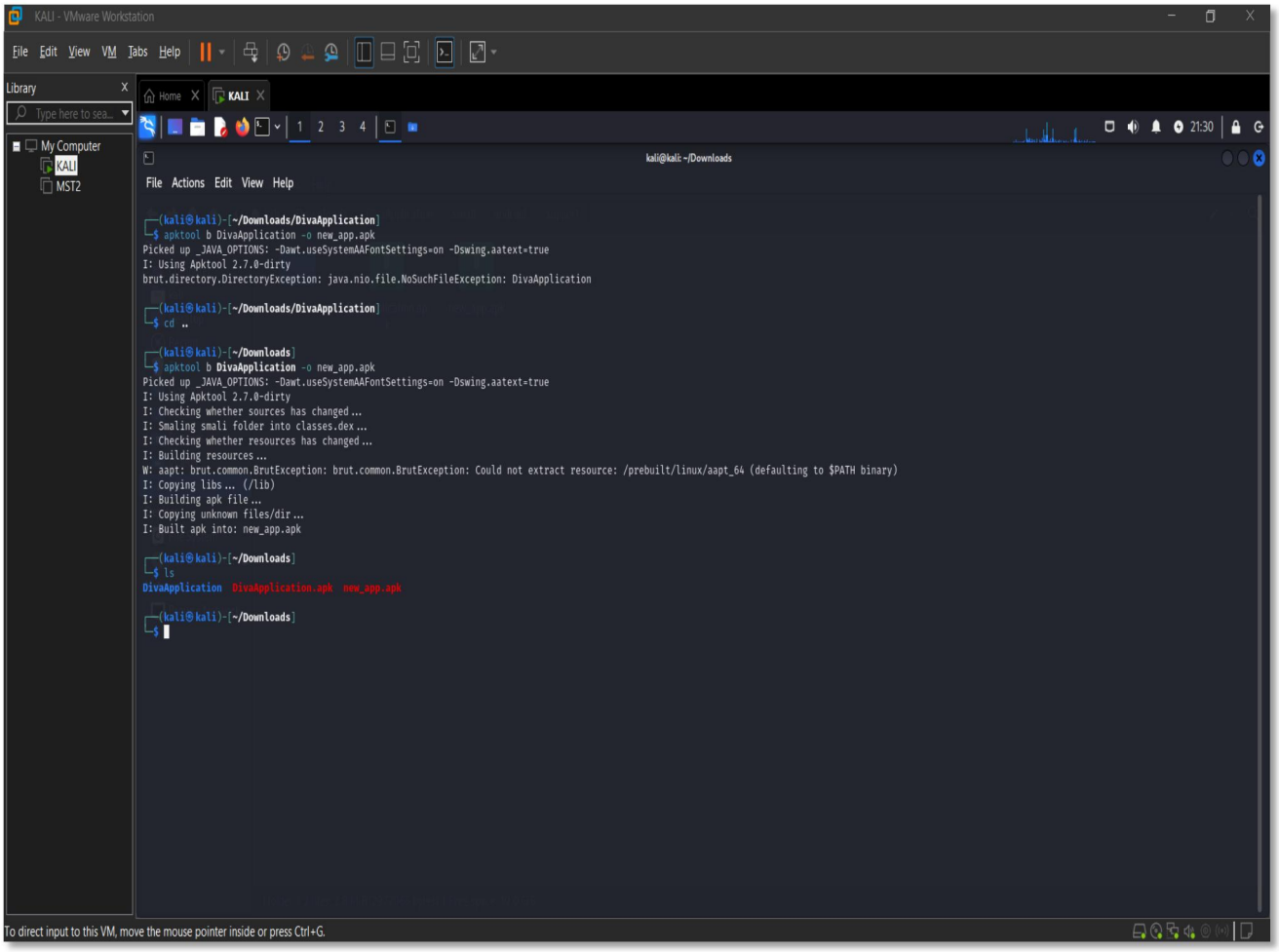
Description:

Apktool is a powerful reverse engineering tool for Android applications, allowing developers or security researchers to decompile and recompile APK (Android Package) files. Re-compiling refers to the process of rebuilding the APK after making modifications to its resources or code.

Step 1:

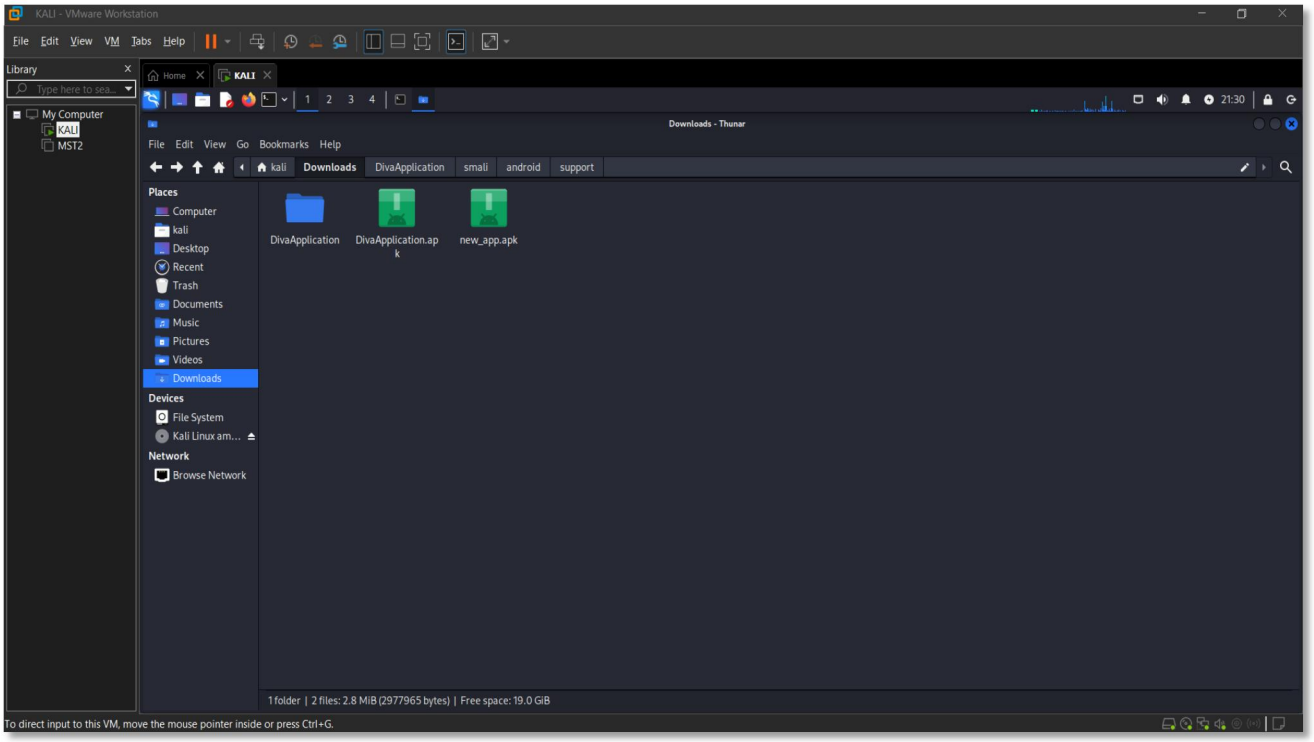
Recompile The APK Using apktool:

apktool b <Your_app> -o <New_app.apk>



Step 2:

Check The Recompiled APK.



Conclusion:

Re-compiling an APK using Apktool is a critical step in Android reverse engineering workflows. It allows for in-depth customization, debugging, or analysis of an application. Whether you're patching an app, testing security vulnerabilities, or modifying UI elements, Apktool provides a reliable way to reconstruct the APK after changes. However, it's important to handle this process ethically and ensure that you have the right to modify and redistribute the application, especially when dealing with proprietary software.

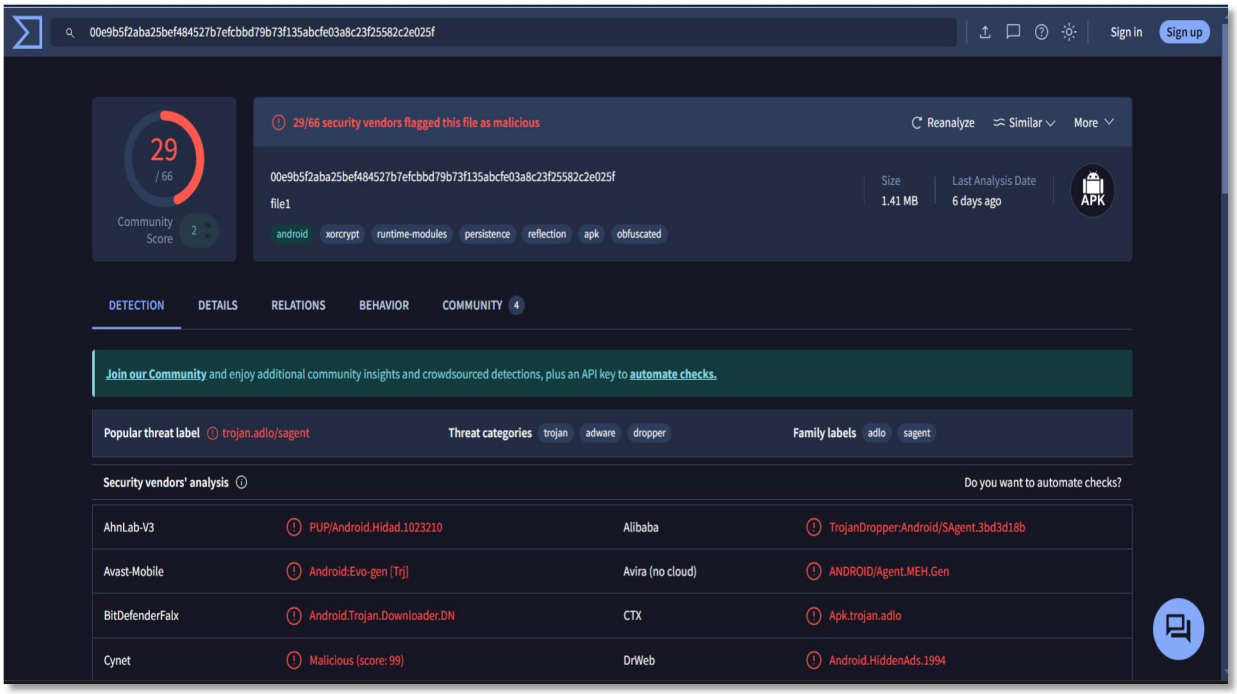
PRACTICAL 9

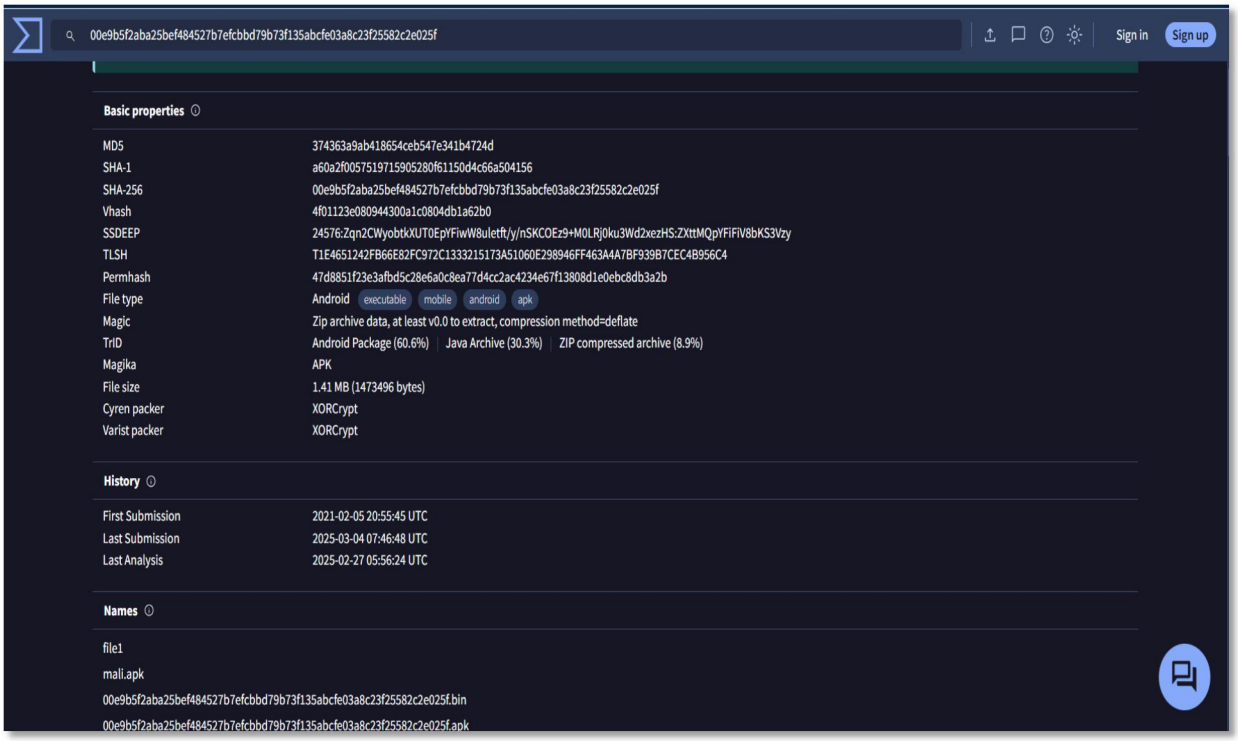
AIM: Analyzing mobile and IoT malware samples using tools like VirusTotal and Yara.

VirusTotal:

SHA-256 (file hash):

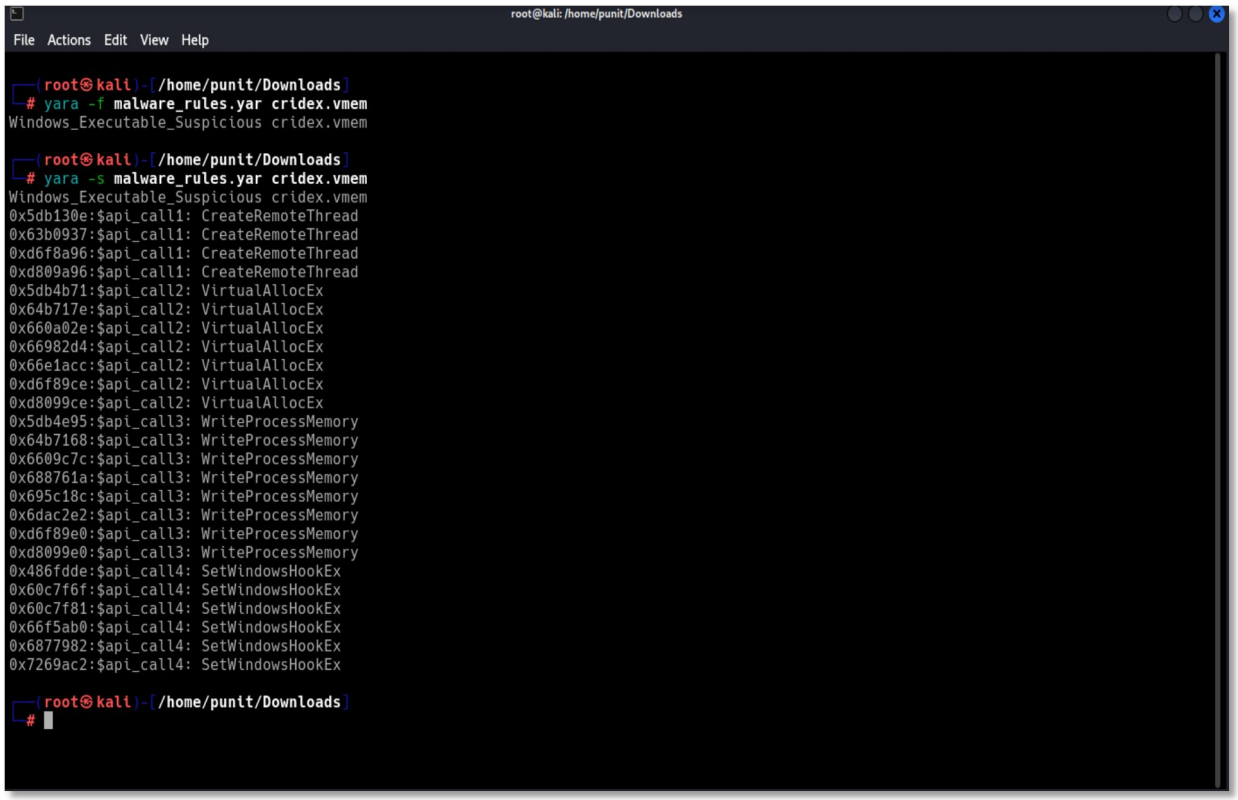
00e9b5f2aba25bef484527b7efcbbd79b73f135abcfe03a8c23f25582c2e025f



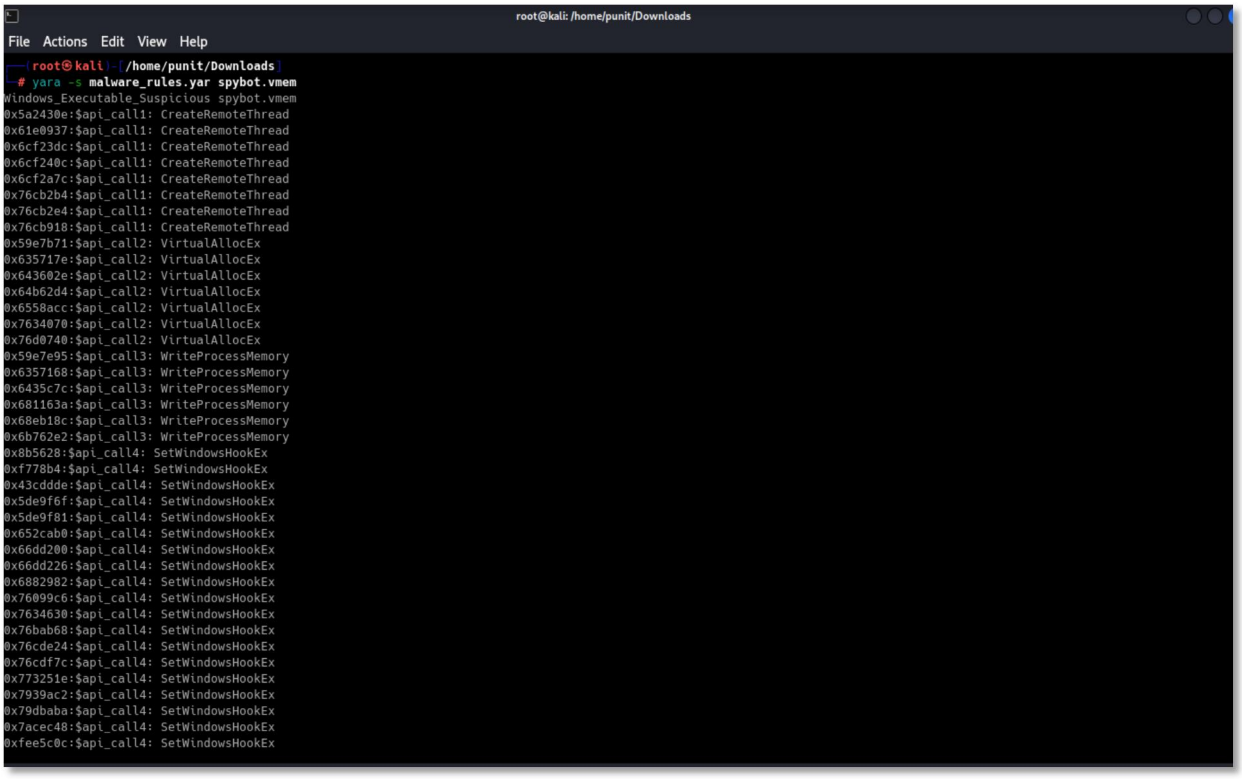


Yara Tool: Scan Using Custom Rules(malware_rules.yar)

cridex.vmem (Malware Sample):



Spybot.vmem (Malware Sample):



Conclusion:

Analyzing mobile and IoT malware samples using tools like VirusTotal and YARA significantly enhances threat detection and malware classification capabilities. VirusTotal provides quick insights by aggregating scan results from multiple antivirus engines, while YARA allows for the creation of custom rules to identify specific malware families or behaviors. Together, these tools streamline the analysis process, enabling security researchers and analysts to efficiently detect, categorize, and understand malicious software targeting mobile and IoT environments. As cyber threats continue to evolve, integrating such tools into malware analysis workflows is essential for maintaining robust cybersecurity defenses.

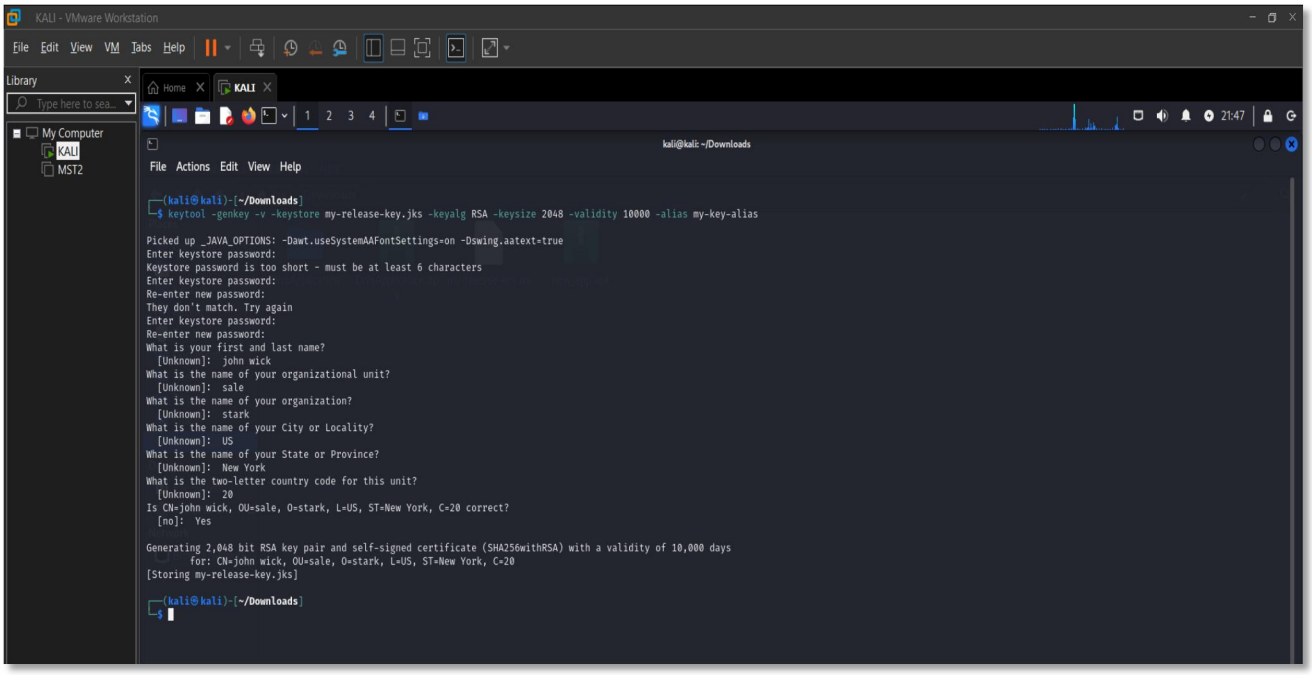
PRACTICAL 10

AIM: application signing by using the apksigner.

apksigner is a command-line tool provided by Android SDK that allows you to sign APK files so they can be installed on devices. Signing is mandatory for all Android applications.

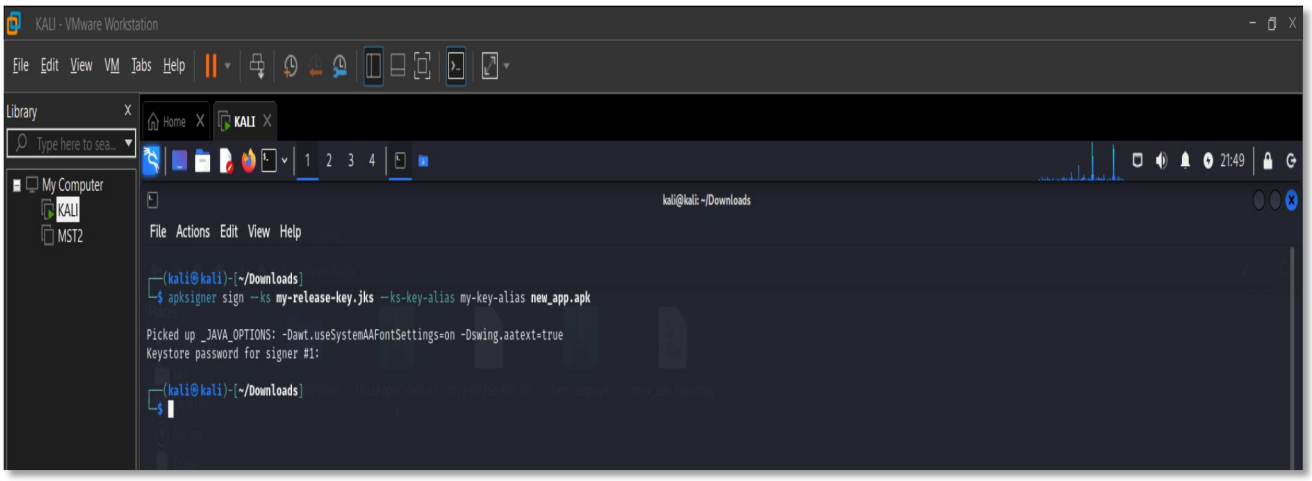
Step 1: Generate a Keystore Using:

keytool -genkey -v -keystore my-release-key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias my-key-alias



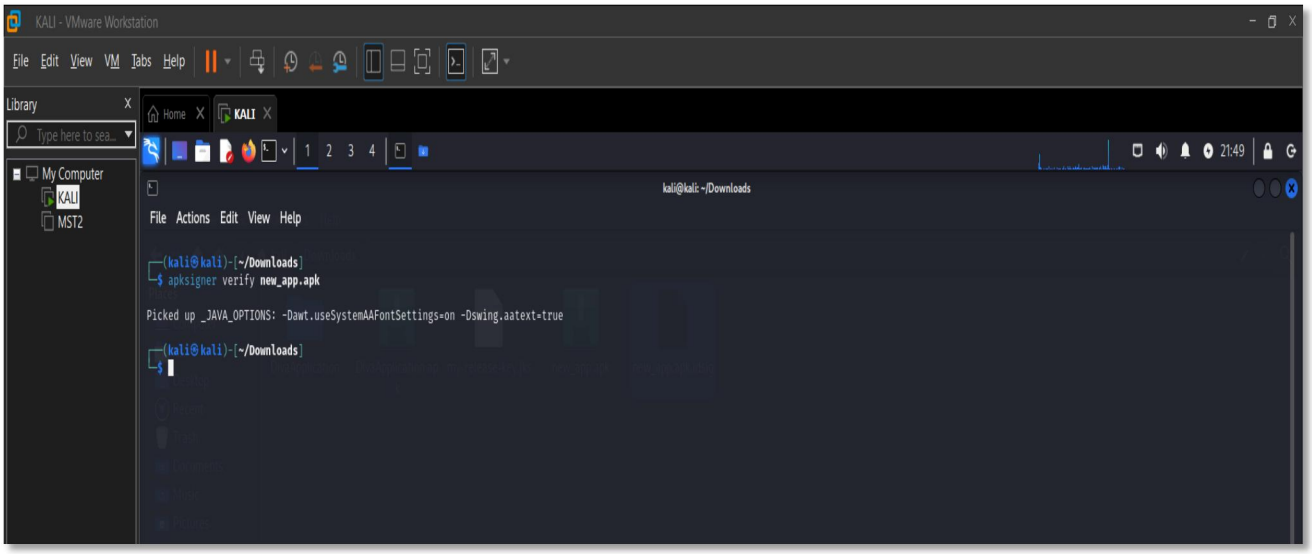
Step 2: Sign the APK Using apksigner.

apksigner sign --ks my-release-key.jks --ks-key-alias my-key-alias path/to/your-app.apk



Step 3: Verify the APK Signature.

apksigner verify path/to/your-app.apk



Conclusion:

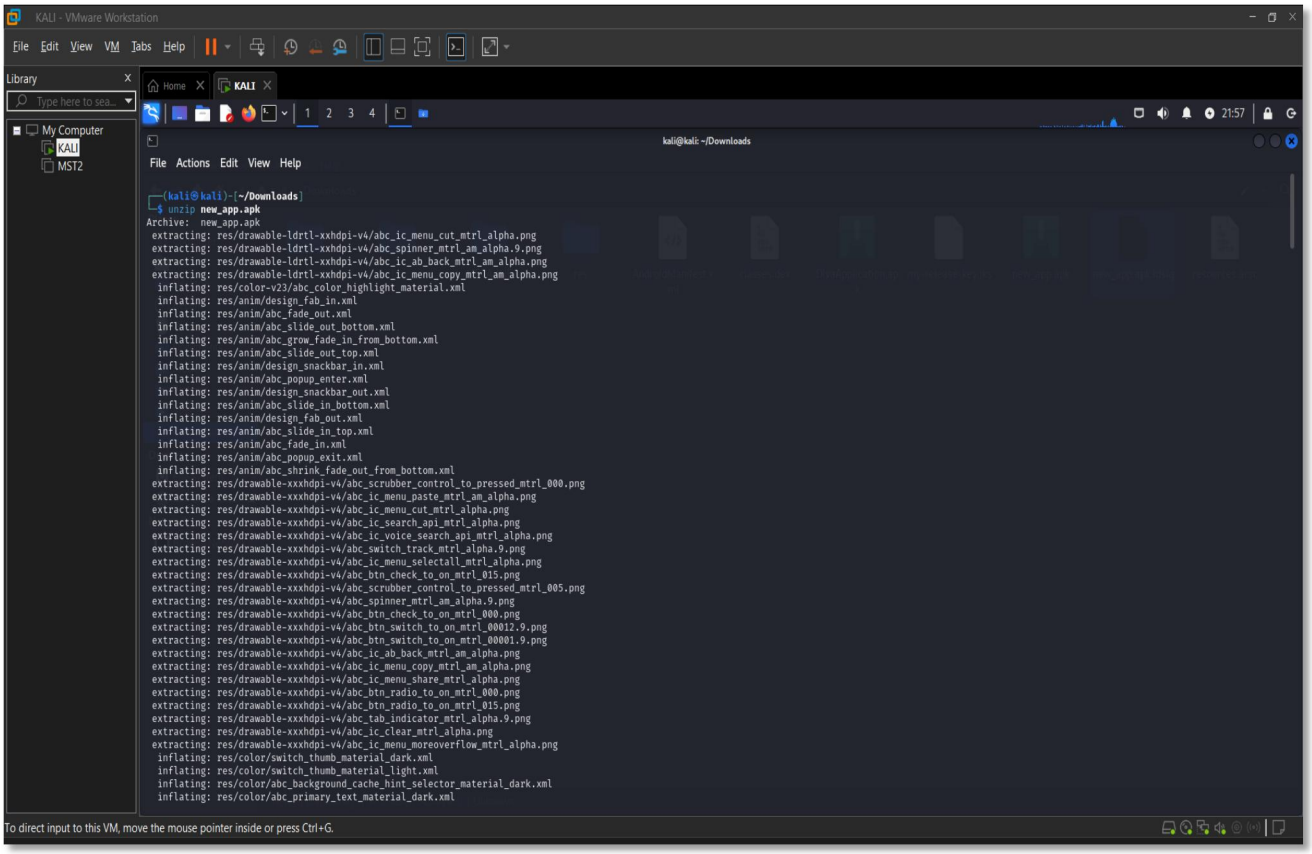
Signing an APK using apksigner is a crucial step in the Android app development and deployment process. It ensures the authenticity and integrity of the application, allowing it to be installed on Android devices and distributed via app stores. By using a secure keystore and the apksigner tool, developers can protect their apps from tampering and build trust with users. Properly signed APKs not only meet Android's security requirements but also support updates and long-term maintenance of applications.

PRACTICAL 11

AIM: converting apk into jar file.

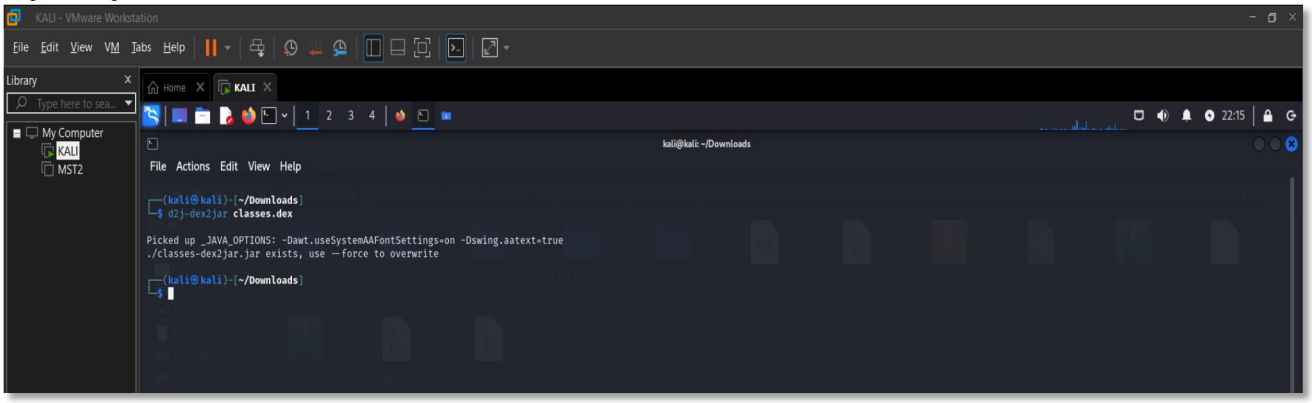
Converting an APK file into a JAR file is often done during Android reverse engineering to analyze the Java code. This process typically involves decompiling the APK's DEX (Dalvik Executable) files into a JAR (Java Archive) so that the code can be read using tools like JD-GUI or CFR.

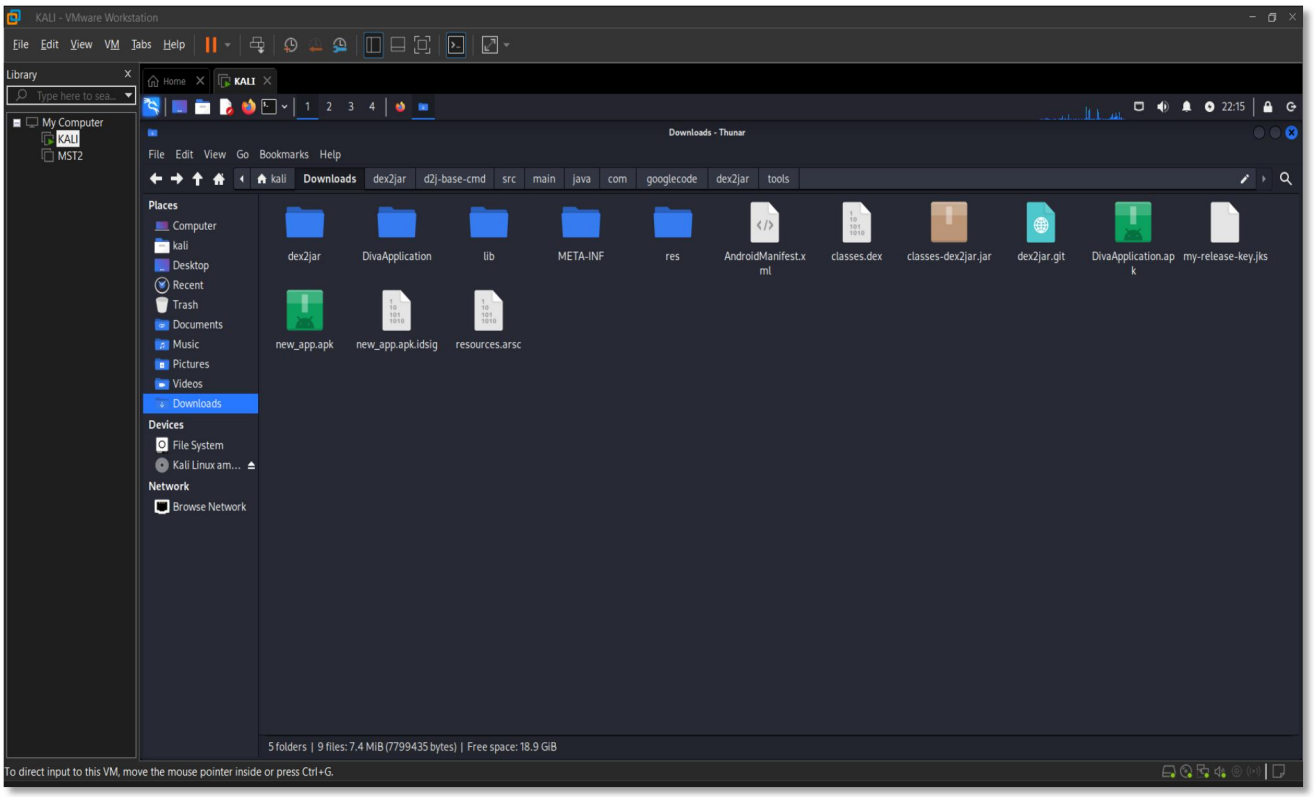
Step 1: Extract classes.dex from the APK.



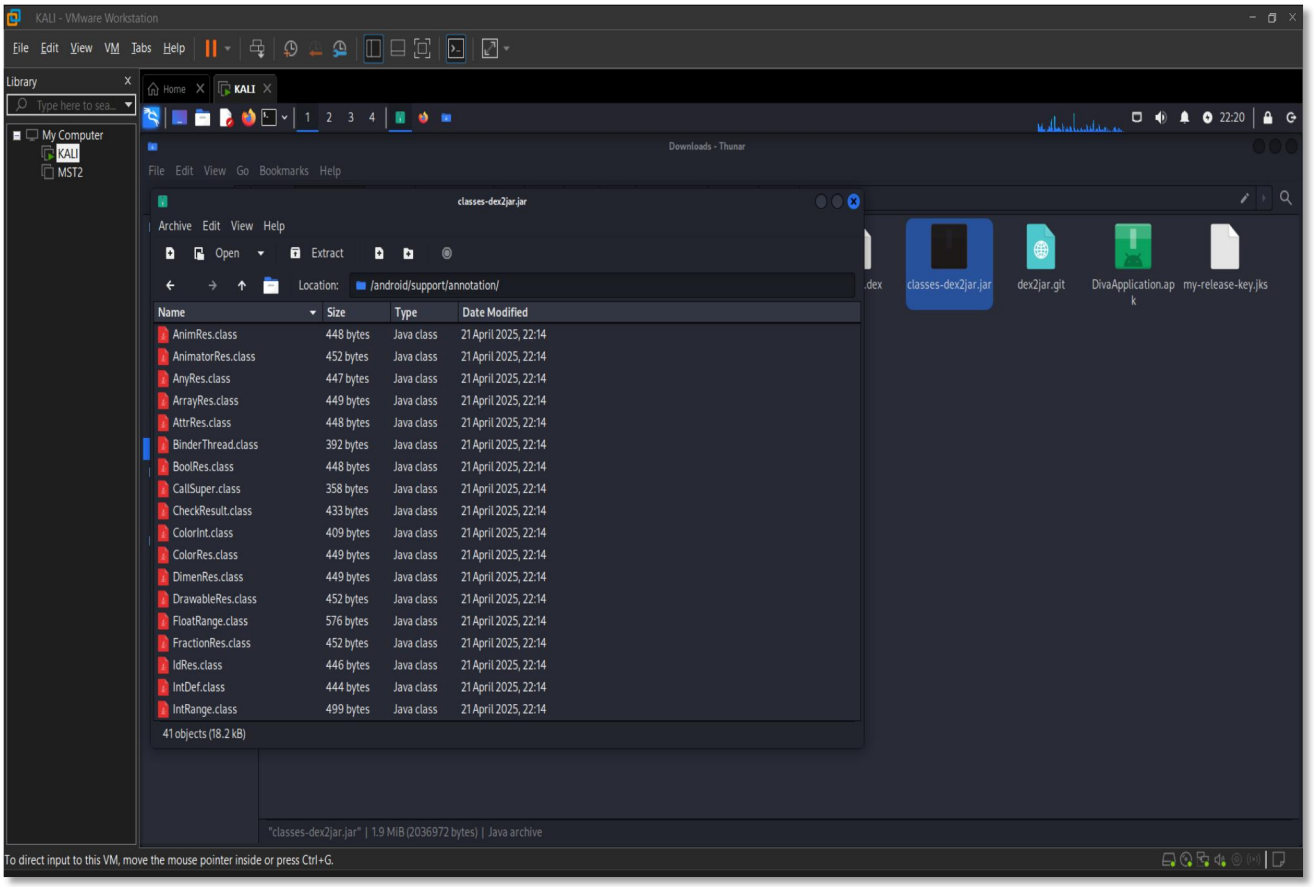
Step 2: Convert DEX to JAR using dex2jar.

d2j-dex2jar classes.dex





Step 3: View or Analyze the JAR File.



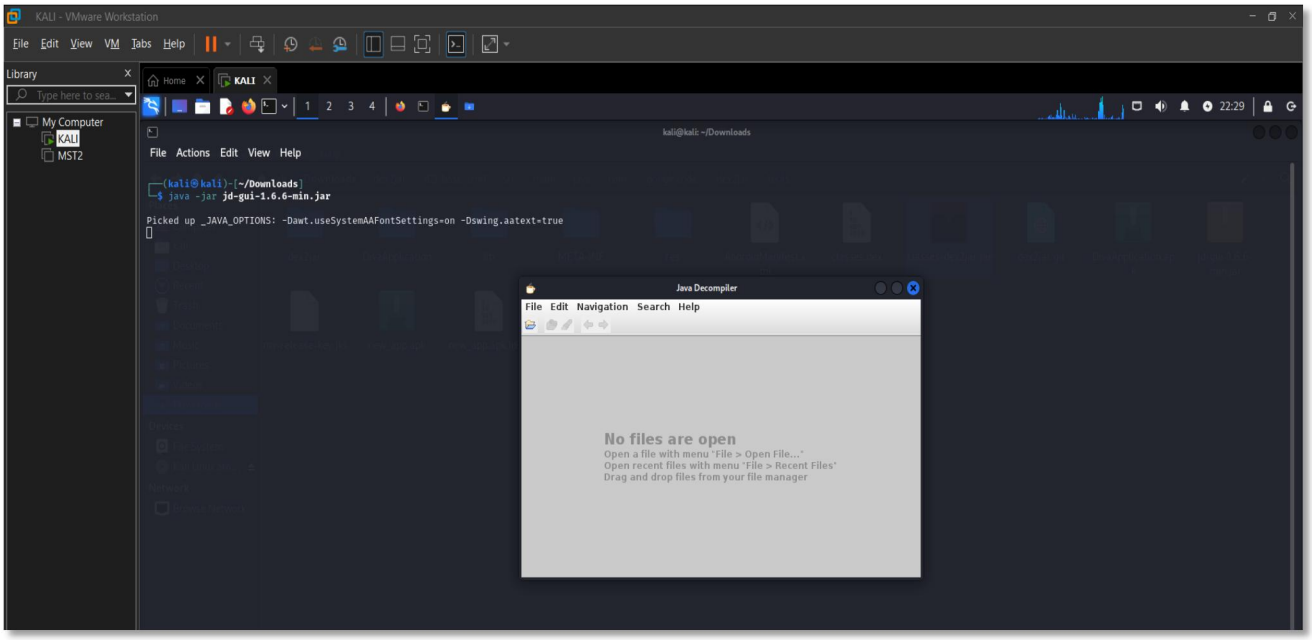
Conclusion:

Converting an APK into a JAR file is a key step in Android reverse engineering, allowing developers and security analysts to inspect the underlying Java code of an application. By extracting the classes.dex file from the APK and using tools like dex2jar, the DEX bytecode can be transformed into a JAR archive, which can then be decompiled into human-readable source code. This process is especially useful for analyzing app behavior, identifying vulnerabilities, or understanding app logic. However, it should always be performed ethically and within legal boundaries, respecting intellectual property rights.

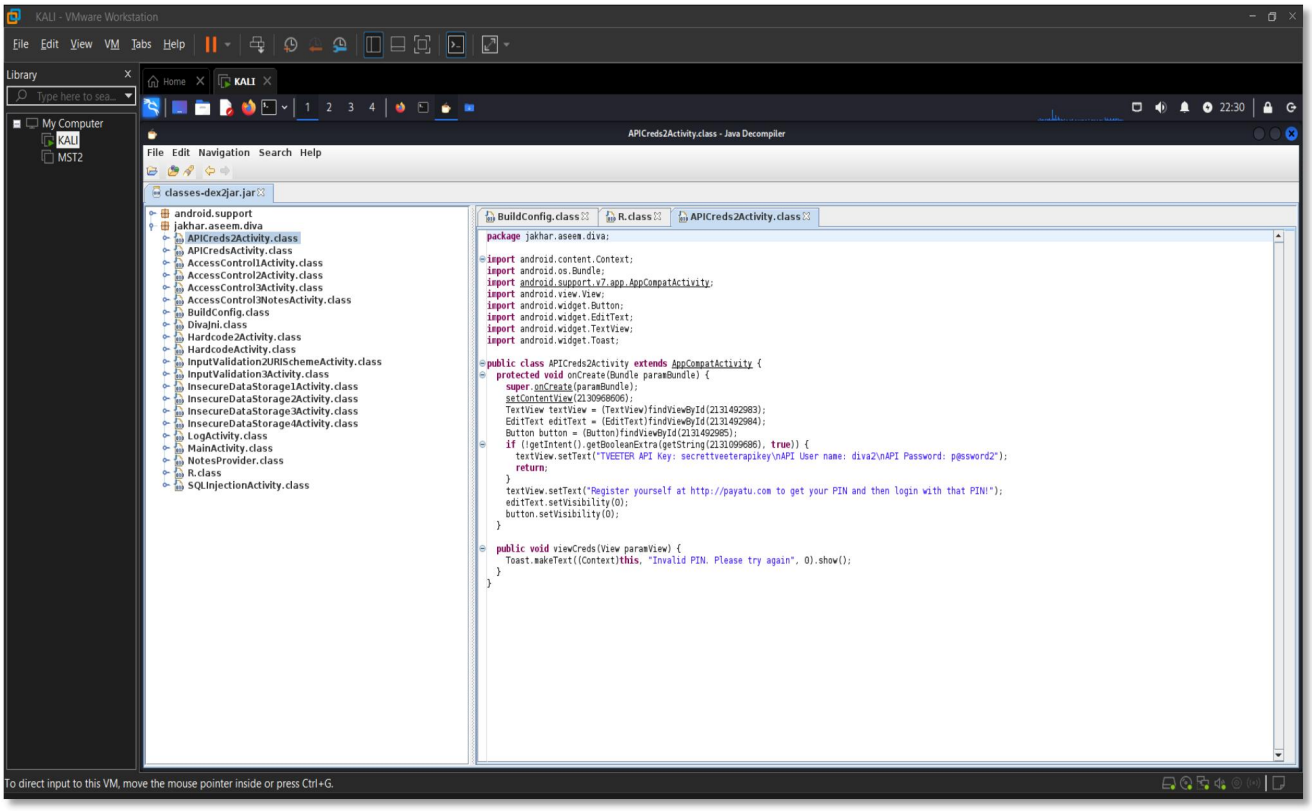
PRACTICAL 12

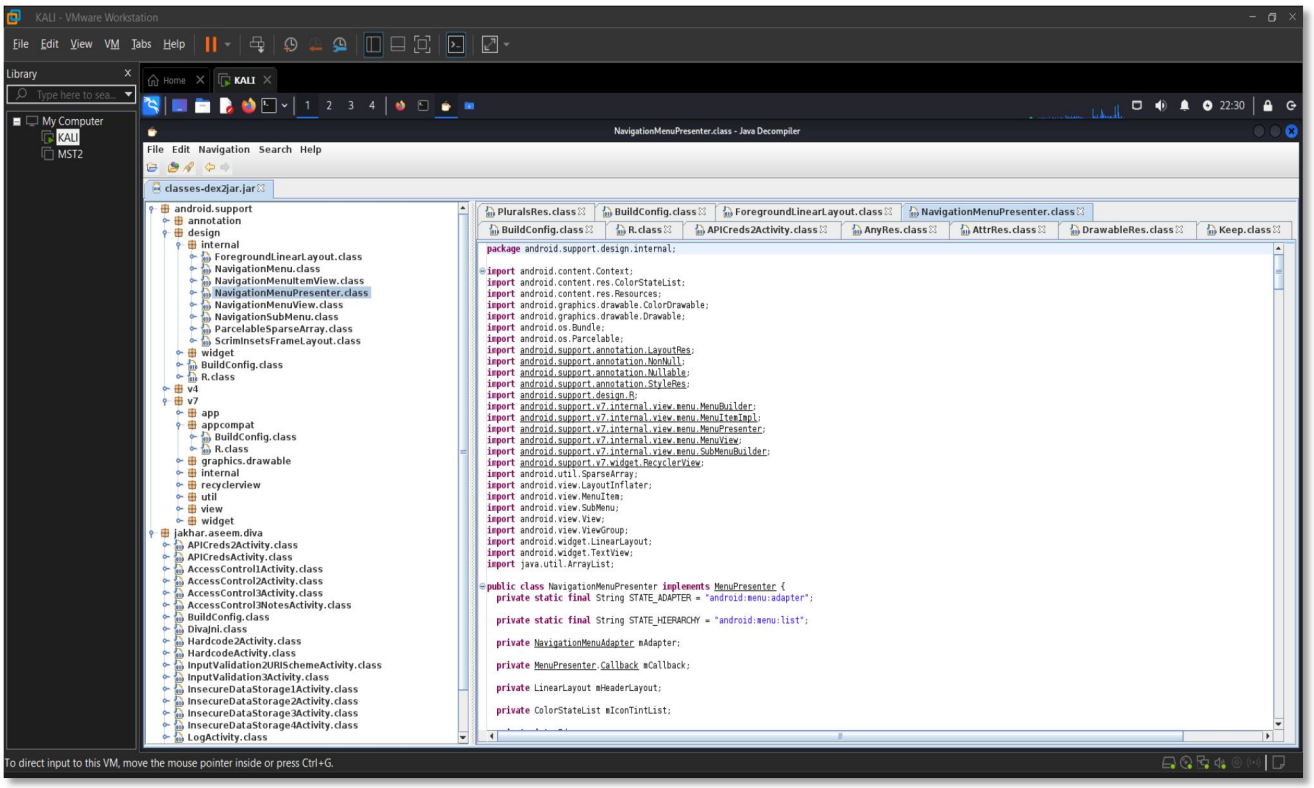
AIM: Opening jar file in jdx GUI and find hardcore strings.

Step 1: Run JD-GUI.



Step 2: Open the JAR File.





Conclusion:

Using JD-GUI to open a JAR file provides a straightforward way to decompile and analyze Java code from Android applications. This is especially useful for identifying hardcoded strings such as API keys, URLs, credentials, and other sensitive information that may pose security risks. By examining these strings, security analysts and developers can uncover potential vulnerabilities, ensure better code practices, and enhance application security. However, such analysis should always be conducted ethically and with proper authorization to respect legal and privacy boundaries.

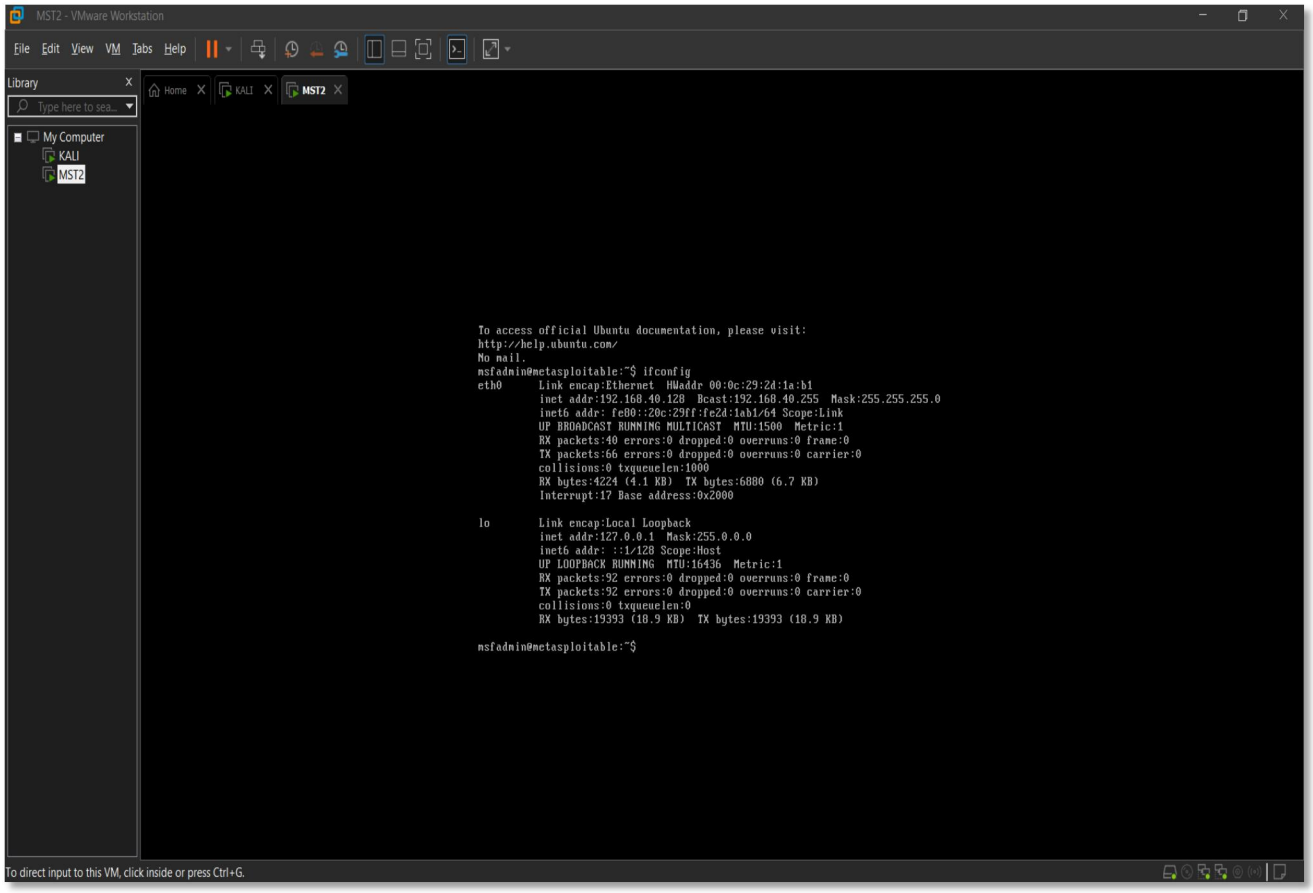
PRACTICAL 13

AIM: Conducting security audits and assessments of mobile and IoT environments using tools like Nmap and Metasploit.

Description:

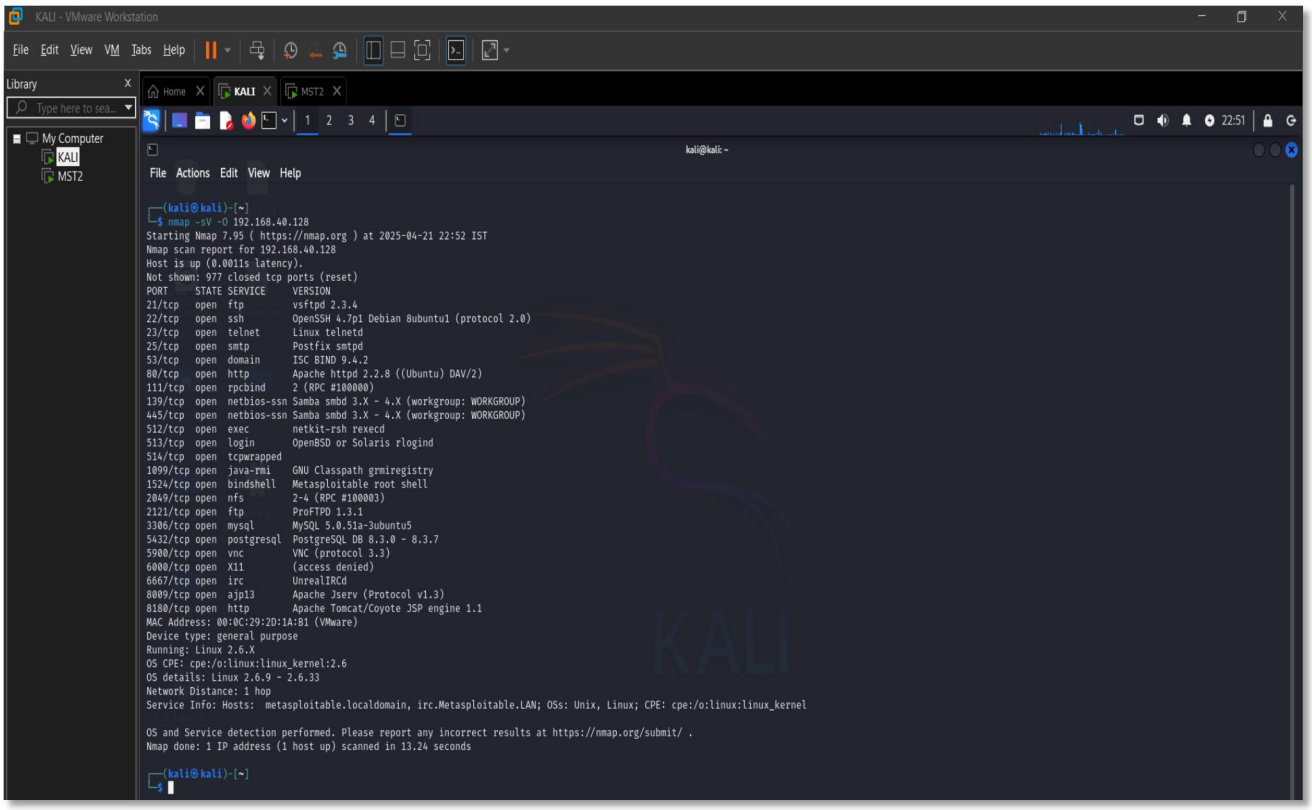
Mobile and Internet of Things (IoT) devices often have unique vulnerabilities due to their lightweight operating systems, always-on connectivity, and limited security configurations. Security audits and assessments are essential to identify potential risks, misconfigurations, and vulnerabilities that could be exploited. Tools like Nmap (for network scanning) and Metasploit (for exploitation and vulnerability testing) are commonly used to perform these assessments.

Step 1: ifconfig in Metasploitable.



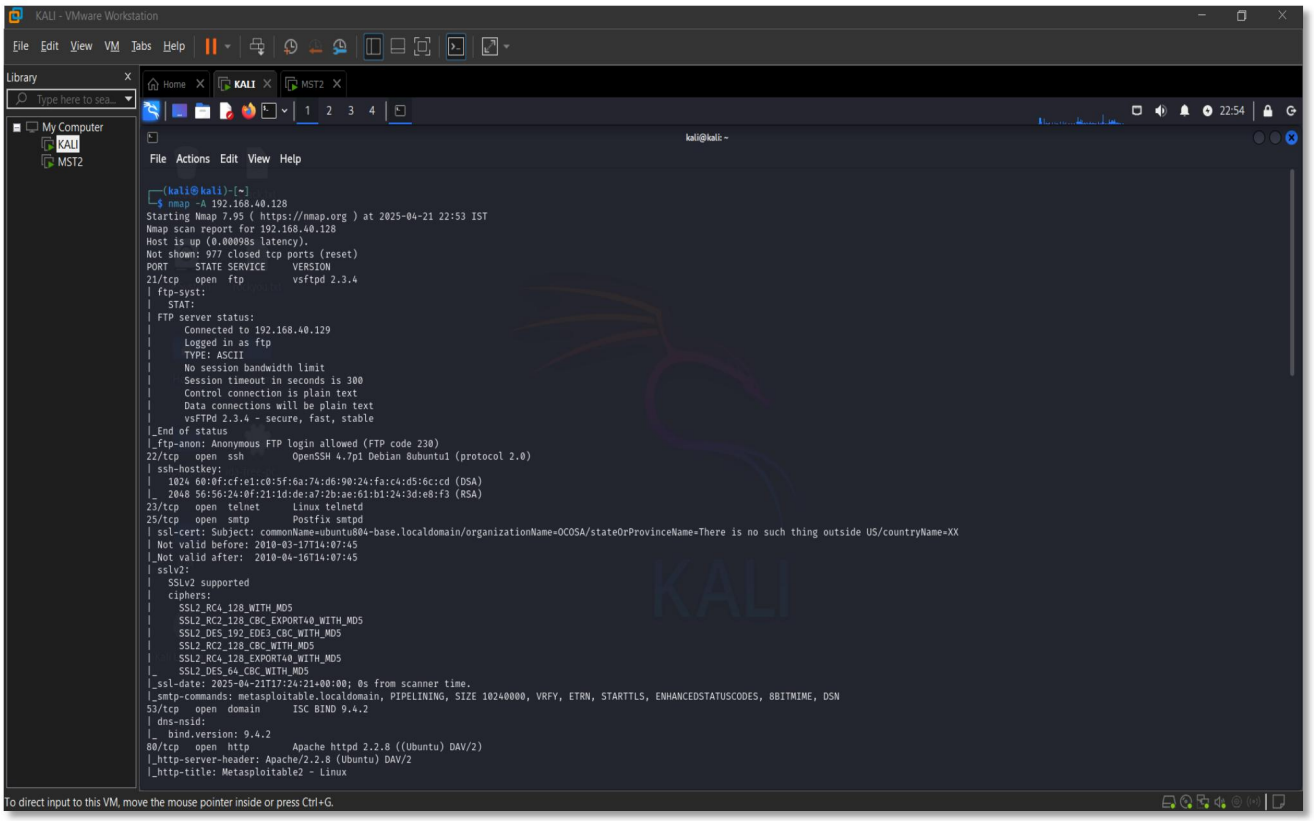
Step 2: Scan open ports and services:

-sV -O



Detect OS and possible device types:

-A



Conclusion:

Conducting security audits and assessments of mobile and IoT environments helps identify and mitigate potential attack vectors. Using tools like Nmap for discovery and vulnerability analysis, and Metasploit for controlled exploitation, you can gain a comprehensive understanding of the security posture. This proactive approach is crucial in securing environments with increasingly interconnected and diverse device ecosystems.

PRACTICAL 14

AIM: Implementing disaster recovery and incident response plans for mobile and IoT environments.

Task 1: Create a Basic Disaster Recovery Plan (DRP):

1. Critical Assets

In a mobile and IoT environment, critical assets include:

- **Mobile Devices**: Smartphones and tablets used by employees or consumers.
- **IoT Sensors**: Devices such as smart thermostats, cameras, or medical equipment that transmit data.
- **Cloud Storage**: Data stored in cloud services for backup, synchronization, and sharing.
- **Network Infrastructure**: Routers, switches, firewalls, and other components necessary for maintaining connectivity and data flow.

2. Backup Strategies:

- **Local Storage**: Store backups on external hard drives, NAS (Network-Attached Storage), or other local devices to ensure quick recovery.
- **Cloud-Based Backup**: Use services such as Google Drive, AWS, Azure, or others to store encrypted backups off-site. Ensure backups are automated and scheduled for regular intervals (e.g., daily, weekly).
- **Automated Backup Schedule**: Set up a schedule for automatic backups. For mobile and IoT environments, ensure that the backup includes:
    - System configurations.
    - Application settings and data.
    - Logs and historical data from IoT sensors.

3. Recovery Time Objective (RTO) & Recovery Point Objective (RPO):

- **RTO (Recovery Time Objective)**: Define the maximum acceptable downtime for each critical asset. For example:
    - Mobile devices: 4 hours.
    - IoT devices: 12 hours (depending on the device and its critical role).
    - Cloud storage or network infrastructure: 1 hour.
- **RPO (Recovery Point Objective)**: Define the maximum acceptable data loss. For example:
    - Mobile devices: 1 hour.
    - IoT devices: 12 hours.
    - Cloud storage: 30 minutes.

4. Steps to Restore Services After an Attack:

- **Identify Affected Systems**: Analyze logs from devices, cloud services, and network monitoring tools to identify the systems or devices affected by the attack.
- **Restore from the Most Recent Clean Backup**: Once affected systems are identified, restore the most recent clean backup that has been verified to be free of malware or corruption.
- **Verify System Integrity Before Reconnecting**: After restoration, check the integrity of the systems to ensure that they are fully functional and secure before reconnecting to the network.

Task 2: Simulate a Security Incident and Recovery Process:

1. File Recovery Using Tools

- **Windows (Recuva)**:
    - **Step 1**: Download and install Recuva from the official website.
    - **Step 2**: Launch Recuva and select the type of file you wish to recover (e.g., documents, photos).
    - **Step 3**: Scan the drive where the deleted file was located.
    - **Step 4**: Select the files to restore from the recovery list and choose a safe location for restored files.

- **Linux (extundelete)**:
    - **Step 1**: Install extundelete using the package manager (e.g., sudo apt install extundelete).
    - **Step 2**: Unmount the drive containing the lost file (e.g., umount /dev/sdX).
    - **Step 3**: Run extundelete with the command: sudo extundelete /dev/sdX --restore-file <path-to-file>.
    - **Step 4**: Review the restored file to verify its integrity.

2. Backup Restoration:

- **Step 1**: Access your cloud or external backup storage.
- **Step 2**: Select the most recent backup prior to the incident and start the restoration process.
- **Step 3**: Monitor the progress of the restoration.
- **Step 4**: After restoration, verify the data integrity to ensure no corruption or incomplete recovery.

3. Verification:

- **Step 1**: Open and inspect the restored data to confirm it is complete and intact.
- **Step 2**: Test any critical applications or IoT devices that use this data to confirm full functionality.
- **Step 3**: Run diagnostics to ensure no remnants of the issue remain.

Task 3: Perform a Tabletop Exercise on Incident Response:

Scenario: A mobile device is infected with malware.

1. Identify the Issue:

- **Analyze Logs for Suspicious Activity**: Use mobile device management (MDM) software or log aggregation tools to review device logs for abnormal behavior, such as unauthorized app installations or strange network traffic.
- **Check Security Alerts for Malware Indicators**: Review security software alerts, antivirus software logs, or cloud service security logs for known malware signatures.

2. Contain the Attack:

- **Disconnect the Affected Device from the Network**: Disconnect the device from Wi-Fi and cellular data to prevent the malware from spreading.
- **Block the Malicious Application or Service**: Use MDM or security tools to isolate or block the malicious application. Disable any services associated with the malware.

3. Recover the System:

- **Reinstall the OS or Factory Reset the Device**: Perform a full factory reset to remove any residual malware. Alternatively, reinstall the mobile OS to restore a clean state.
- **Restore Data from a Clean Backup**: After resetting the device, restore data from a previously clean backup, ensuring that it predates the infection.
- **Verify that the Malware Is Completely Removed**: Perform a full device scan with up-to-date antivirus software to ensure that no malware persists after restoration.

Conclusion:

Implementing robust disaster recovery and incident response plans in mobile and IoT environments is no longer optional—it's a necessity. These ecosystems are inherently dynamic, widely distributed, and vulnerable to a wide range of cyber threats and physical disruptions. Effective DR and IR strategies must account for the unique characteristics of mobile devices and IoT systems, such as limited processing power, intermittent connectivity, and diverse operating platforms. Organizations must adopt proactive approaches, including real-time monitoring, automated threat detection, and decentralized response mechanisms.

PRACTICAL 15

AIM: Investigating and responding to security incidents on mobile and IoT devices and networks.

1. Initial Detection & Identification:

When you first suspect or detect an incident (e.g., through an alert or user complaint), verify the issue:

- **Example Scenario:** You receive an alert from your Mobile Device Management (MDM) system that an employee's mobile device is showing unusual network activity.
- **Practical Step:** Use your MDM or monitoring tools to check:
  - The device's recent activity (e.g., abnormal data consumption, unauthorized app installation).
  - Whether the device is jailbroken or rooted.
  - Any signs of malware or unauthorized access.

- **Practical Step:** Log into the device's management interface (if accessible) and check:
  - Network logs for unusual traffic or communication with external IPs.
  - The firmware version and ensure it's up-to-date (outdated firmware can be a big security risk).
  - Check for unauthorized devices connected to your IoT network.

2. **Containment:**

Once you've verified there's a security incident, contain the threat to prevent it from spreading further.

- **For Mobile Devices:**

If the device is infected, disconnect it from your network immediately (e.g., turn off Wi-Fi and data, or use your

  - MDM to lock the device and remove access to sensitive data).
  - If you suspect a malicious app, forcefully uninstall it using MDM or via device settings.
- **For IoT Devices:**

  - If you detect unusual activity on an IoT device (like a camera or sensor), disconnect the device from your network to prevent it from sending data out or being used for attacks like botnets.

3. Investigation & Analysis:

Now that the threat is contained, start investigating the cause and scope of the incident.

For Mobile Devices:

- Use mobile forensic tools (e.g., Cellebrite, Oxygen Forensics) to extract logs and data from the device. You want to look for:
  - Evidence of malware or unauthorized apps.
  - Suspicious network connections or data exfiltration.
- Check event logs for any system or app errors that coincide with the suspicious behavior.
- Review installed apps—are there any apps that shouldn't be there or that have excessive permissions?

For IoT Devices:

- Analyze network traffic using tools like Wireshark to identify unusual patterns, like sudden spikes in data or communication with external IP addresses.
- Check device logs for any signs of unauthorized access or misconfigurations.
- If the IoT device is managed by a third-party vendor, reach out to them for help in tracing the issue.

4. Eradication:

After identifying the root cause, eradicate the threat by removing any malicious components.

For Mobile Devices:

- Remove malware: If malware was detected, use mobile security software to clean the device.
- Revert to a known good backup: If the device was severely compromised, restoring to a clean, secure backup may be necessary.
- Update device software to patch any vulnerabilities that may have been exploited.

For IoT Devices:

- Update firmware: Make sure that the device is running the latest, most secure firmware version.

- Reconfigure device settings to ensure it's secure and doesn't have any vulnerabilities (e.g., default passwords).
- If needed, replace the device entirely if the compromise is severe or if the device cannot be properly secured.

5. Recovery:

Once you've removed the threat, focus on bringing systems back online safely.

For Mobile Devices:

- Reconnect the device to your network only after ensuring it's secure and malware-free.
- Monitor the device's behavior closely for any signs of recurrence.
- Make sure the employee is re-educated on device security practices, like avoiding suspicious apps and links.

For IoT Devices:

- Reintroduce the IoT device to the network gradually, ensuring it's properly configured and isolated from sensitive systems.
- Test the device to make sure it's functioning properly and not causing issues on the network.
- Continuously monitor the device to detect any new suspicious activity.

6. Post-Incident Review and Reporting:

After the incident is resolved, it's time to document everything and improve your security posture for the future.

For Mobile Devices:

- Conduct a root cause analysis: What exactly led to the compromise? Was it a user error, outdated software, or a vulnerability?
- Update your mobile security policies: Perhaps introduce stronger access controls, a more aggressive app vetting process, or regular device audits.
- Report the incident: If it involved sensitive data, notify the necessary stakeholders (e.g., affected users, internal teams, and possibly regulatory bodies).

For IoT Devices:

- Review network traffic logs and ensure no other devices are compromised.
- Update IoT security protocols: Consider implementing better segmentation for your IoT devices, stronger device authentication, and stricter firmware update policies.
- Educate teams on best practices for securing IoT devices, such as using secure configurations, avoiding default passwords, and performing regular vulnerability assessments.

7. Preventative Measures:

To reduce the likelihood of future incidents, adopt some proactive security measures:

Mobile Devices:

- Endpoint protection: Ensure that all devices are using security solutions that include antivirus and device encryption.
- Regular software updates: Make sure that all mobile devices are kept up-to-date with the latest security patches.
- Training: Provide training for employees on how to recognize phishing attempts and secure their devices.

IoT Devices:

- Network segmentation: Isolate IoT devices on a separate network to prevent lateral movement if one device is compromised.
- Strong authentication: Use strong passwords and multi-factor authentication for device access wherever possible.
- Regular audits: Schedule periodic reviews of your IoT devices to ensure that they are secure and up-to-date.

Conclusion:

Investigating and responding to security incidents on mobile and IoT devices and networks is critical in today's hyper-connected world. These devices often operate with limited security features, making them attractive targets for cyberattacks. Effective incident response requires a proactive, layered approach that includes real-time monitoring, threat intelligence, and device-specific forensics. Given the diversity and scale of IoT and mobile ecosystems, collaboration between manufacturers, service providers, and security teams is essential. Strengthening detection capabilities, ensuring timely patching, and implementing strong authentication can significantly mitigate risks. As the threat landscape continues to evolve, so too must the tools, strategies, and policies used to secure mobile and IoT infrastructures.