A CAPSTONE PROJECT REPORT

On

ShadowDroid: An Advanced Android Security Auditing and Threat Detection Framework

Submitted in partial fulfilment of the requirement of

University of Mumbai for the Degree of

Bachelor of Engineering

In

CSE IOT and Cyber Security including Block chain

Submitted By

Om Sanjay Kadam

Supervisor

Prof Madhu Nashipudimath

Department of CSE IOT and Cyber Security including Blockchain

Smt. Indira Gandhi College of Engineering, Ghansoli – 400701

UNIVERSITY OF MUMBAI

Academic Year 2024 – 25

Department of CSE IOT and Cyber Security including Blockchain

SMT. INDIRA GANDHI COLLEGE OF ENGINEERING

GHANSOLI – 400701

CERTIFICATE

This is to certify that the requirements for the Capstone Project-I entitled '**ShadowDroid: An Advanced Android Security Auditing and Threat Detection Framework**' have been successfully completed by student:

Name	**Roll No.**
Om Sanjay Kadam	2021ci30f

in partial fulfilment of Bachelor of Technology of Mumbai University in the Department of CSE IOT and Cyber Security including Block chain, SMT. INDIRA GANDHI COLLEGE OF ENGINEERING, GHANSOLI – 400701 during the Academic Year 2023 – 2024.

Prof Madhu Nashipudimath
Head of Department and Supervisor

Dr. Sunil Chavan
Principal

Department of CSE IOT and Cyber Security including Blockchain

SMT. INDIRA GANDHI COLLEGE OF ENGINEERING

GHANSOLI – 400701

PROJECT APPROVAL FOR B.E

This Capstone project entitled **"ShadowDroid: An Advanced Android Security Auditing and Threat Detection Framework"** by **OM SANJAY KADAM** is approved for the degree of **CSE IOT and Cyber Security including Blockchain.**

Examiners:

1. _____

2. _____

Supervisors:

1. _____

2. _____

Chairman:

1. _____

Date:

Place

Department of CSE IOT and Cyber Security including Blockchain

SMT. INDIRA GANDHI COLLEGE OF ENGINEERING

GHANSOLI – 400701

DECLARATION

I hereby declare that this written submission for the Capstone project entitled **"ShadowDroid: An Advanced Android Security Auditing and Threat Detection Framework"** represents my ideas in my own words. Wherever I have included ideas, data, or words from other sources, I have adequately cited and referenced the original works.

I affirm that I have adhered to the principles of academic honesty and integrity, ensuring that no part of this submission misrepresents, fabricates, or falsifies any information, data, or sources. I fully understand that any violation of the above may result in disciplinary action by the institute and may also lead to legal or penal action from the original sources that have not been properly cited or from whom prior permission was required.

Project Group Members:

Om Sanjay Kadam

Date:

Place:

Acknowledgement

I sincerely thank **Prof. Madhu Nashipudimath**, my Head of Department and project guide, for their invaluable guidance and support throughout this project. Their insights and encouragement have been instrumental in shaping my work.

I am also grateful to our Principal, **Dr. Sunil Chavan**, for fostering an environment that encourages learning and innovation. A special thanks to the faculty and staff of the **IoT Department Laboratory** for their assistance and resources.

Om Sanjay Kadam

Abstract

In an era where smartphones have become the cornerstone of digital interaction, securing Android devices is no longer optional—it is essential. As the most widely deployed mobile operating system, Android presents an expansive attack surface for cyber threats, ranging from malware and insecure configurations to privilege escalations. Existing security assessment techniques often fall short in delivering comprehensive, real-time, and device-level analysis. To bridge this gap, this research introduces **ShadowDroid**, a robust and extensible security auditing framework designed specifically for Android penetration testing via Android Debug Bridge (ADB). Unlike conventional remote-access or static analysis tools, ShadowDroid leverages direct device connectivity to perform in-depth, automated evaluations in a controlled and authorized environment. The framework integrates a suite of analysis modules that examine root status, installed applications, permission hierarchies, call forwarding vulnerabilities, accessibility service abuse, and network configurations. It further enhances threat detection by incorporating external intelligence through the VirusTotal API and public IP reputation services. Equipped with a reporting engine that generates detailed PDF summaries, ShadowDroid serves as a powerful and practical tool for security researchers and penetration testers aiming to elevate the standard of Android device assessments.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Fundamentals

In the rapidly evolving digital landscape, mobile devices have emerged as critical vectors for both personal and enterprise-level interactions. As such, they have become prime targets for a wide range of cyber threats. Among mobile operating systems, Android stands as the most widely used and, consequently, one of the most vulnerable platforms. Its open-source architecture and diverse device ecosystem expose it to numerous risks, including malware, data leaks, insecure configurations, and privilege escalations.

Conventional techniques such as static and dynamic analysis offer some protection but are often constrained by limitations in real-time monitoring, adaptability, and contextual threat evaluation. These shortcomings underscore the need for a more versatile and proactive approach to Android security assessment.

In response to these challenges, this project introduces **ShadowDroid**, a modular and automated security auditing framework designed for local Android penetration testing via Android Debug Bridge (ADB). ShadowDroid provides in-depth analysis of device security posture by evaluating factors such as root status, installed applications, permission structures, call forwarding vulnerabilities, accessibility service misuse, and network configurations. The framework also incorporates threat intelligence through public IP reputation checks and VirusTotal API integration. With a built-in reporting engine for generating detailed PDF outputs, ShadowDroid empowers cybersecurity professionals with actionable insights to identify and mitigate real-world vulnerabilities in Android environments.

This chapter establishes the context for Android security concerns, outlines the rationale for developing ShadowDroid, and introduces the methodologies applied throughout this research.

1.2 Objectives

The primary objectives of this project are as follows:

- To analyze current Android security assessment methods and identify their limitations.
- To design and implement a modular, menu-driven framework for performing automated penetration testing on Android devices via ADB.
- To enable comprehensive auditing of Android devices, focusing on areas such as app permissions, network security, system vulnerabilities, and accessibility risks.
- To provide cybersecurity practitioners with a practical and extensible toolkit that delivers real-time insights through local device interaction.
- To generate structured and exportable security reports suitable for documentation, forensics, and compliance.

1.3 Organization of the Report

This report is structured to guide readers through a logical progression from foundational understanding to implementation and analysis. Chapter 1 introduces the core concepts of Android security, the motivations behind the project, and the research objectives. Chapter 2 presents a literature review of existing assessment tools and techniques, identifying their strengths and limitations. Chapter 3 outlines the theoretical framework and methodology used in the development of ShadowDroid. Chapter 4 discusses the practical implementation of the framework, experimental results, and its implications for cybersecurity. Finally, Chapter 5 concludes the report by summarizing key findings, highlighting challenges encountered, and proposing directions for future work. This structured approach aims to provide a clear understanding of the problem space, the proposed solution, and its potential impact on enhancing Android device security.

Chapter 2

Literature Survey

2. Introduction

With the exponential rise in smartphone usage and mobile application development across sectors, mobile security has become a pivotal concern within the broader cybersecurity domain. Modern mobile devices serve as hubs for personal communication, financial transactions, cloud access, and data storage—making them attractive targets for cybercriminals. The evolution of mobile threats such as malware, ransomware, phishing, spyware, and surveillance tools reflects a growing sophistication, often powered by artificial intelligence, automation, and increasingly complex attack strategies.

One of the primary challenges in securing mobile ecosystems lies in their inherent complexity and diversity. Attackers exploit weaknesses in mobile operating systems—particularly Android—through insecure application programming interfaces (APIs), flawed permission models, outdated software versions, and inadequate user awareness. These vulnerabilities have led to significant breaches and unauthorized data access across both consumer and enterprise environments.

This chapter presents a comprehensive review of existing techniques used for vulnerability analysis in mobile security. It categorizes the methodologies into static analysis, dynamic analysis, and hybrid approaches:

- **Static analysis** techniques, including source code review, symbolic execution, and control/data flow analysis, are employed during the development phase to identify vulnerabilities without executing the code.
- **Dynamic analysis** methods such as penetration testing, runtime monitoring, behavior analysis, and fuzz testing enable evaluators to observe how applications and devices behave under real-world conditions.
- **Hybrid analysis** combines the strengths of both static and dynamic techniques, offering more comprehensive and context-aware insights into mobile security flaws.

By synthesizing the findings of significant studies and tools in this domain, this survey aims to highlight current strengths, uncover prevailing limitations, and provide a foundation for the development of more effective security auditing frameworks such as **ShadowDroid**. This literature

review serves not only to inform but also to justify the methodological choices adopted in this project.

a. Evolution of Cyber Threats

The cyber security landscape has evolved dramatically over the past decade. Attackers have leveraged advanced techniques, automation, and artificial intelligence to exploit vulnerabilities at an unprecedented scale. Cybercrime has become a lucrative industry, with ransomware, phishing, and malware attacks dominating the threat landscape. Additionally, state-sponsored attacks and hacktivist activities have increased, further complicating the cybersecurity ecosystem.

b. Trends in Cyber attacks

A comprehensive analysis of **23,567 cyber attacks** from January 2011 to December 2023 reveals a drastic escalation in cyber threats. Over the past six years, cyberattacks have increased by over **350%**, with a significant rise in the monthly average of recorded incidents. The number of attacks rose from **1,554 in 2018 to 7,068 in 2023**, representing a **355% increase**. This surge highlights the escalating nature of cyber threats and the growing sophistication of attackers.

In **2023**, specific months experienced lower-than-average attacks, such as **January (395), February (460), May (570), August (587), September (577), October (537), and December (565)**. However, on a quarterly basis, **Q2 was the worst** with an average of **650 attacks per month**, followed by **Q3 with 601 attacks per month**. The Table 1.1 below shows Cyber Attacks per Month in 2023

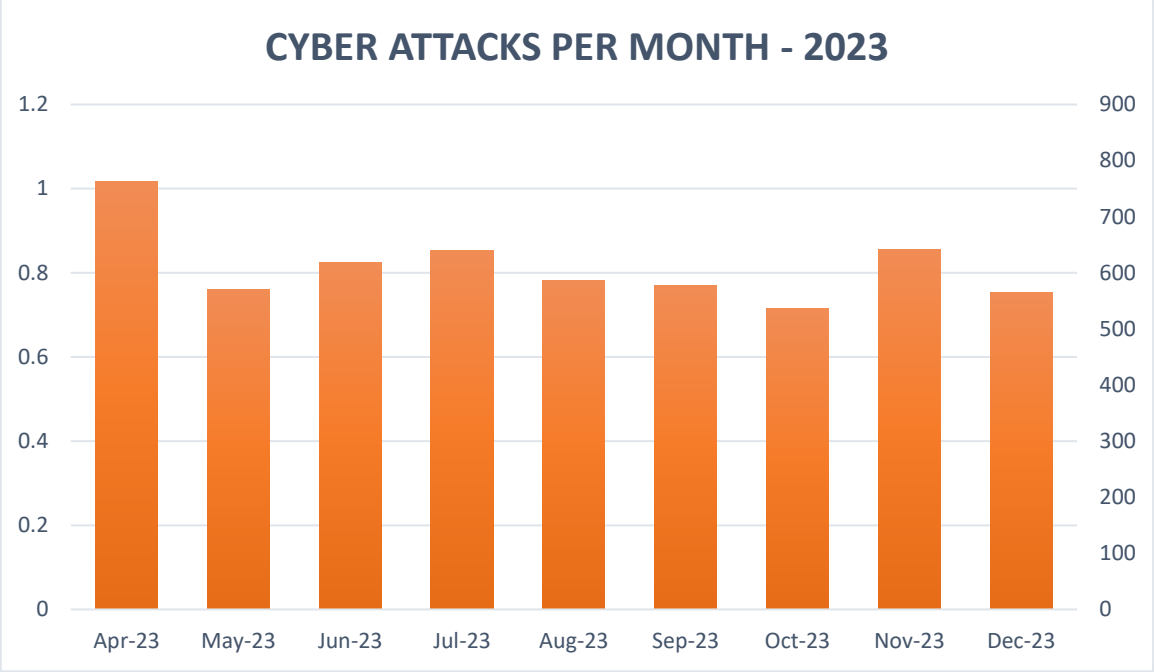

Figure 1.1: Cyber-attacks recorded per month in 2023

c. Cybercrime, Hacktivism, and Espionage

Cybercrime-related attacks saw a **+433% increase from 2018 to 2023** and a **+221% rise compared to 2022**. Hacktivism-driven attacks grew even faster, surging **+356% compared to 2018** and **+248% compared to 2022**. Espionage and Information Warfare remained prevalent, although the sharp increase from the previous year was linked to geopolitical conflicts.The Table 1.2 below shows Attackers distribution in 2023

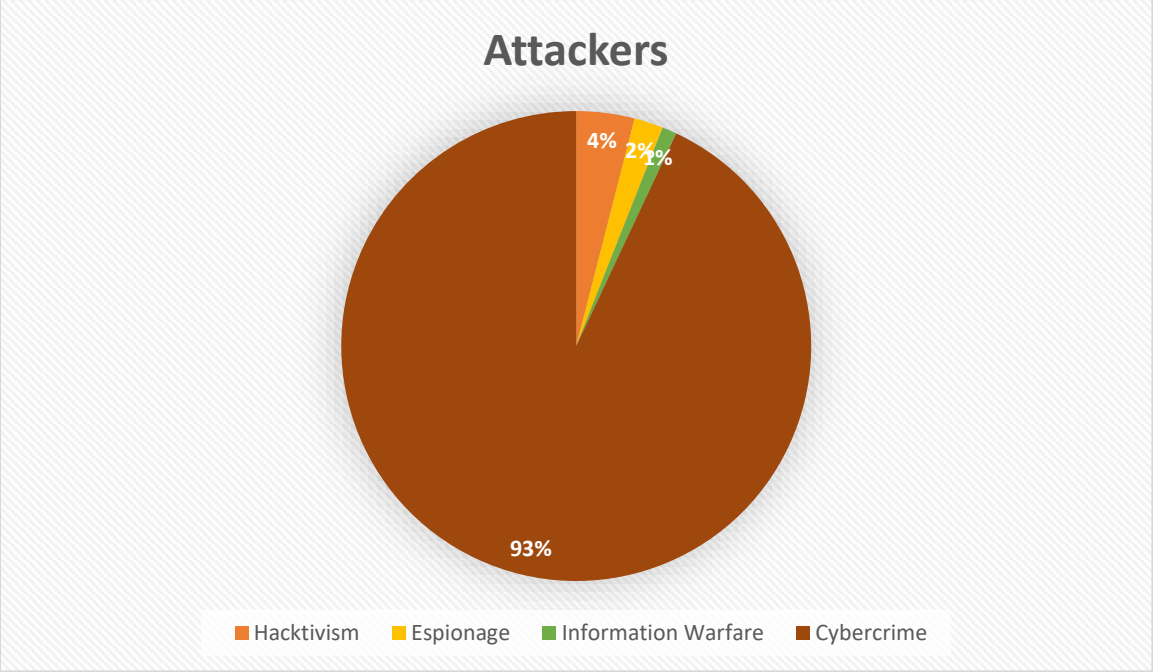

Figure 1.2: Attackers distribution in 2023

d. Ransomware and Cyber Gangs

In 2023, **ransomware attacks accounted for 66% of all recorded cyberattacks**, with **97 identified cyber gangs** conducting **4,500 known ransomware incidents**. Prominent ransomware groups and their contributions to total attacks include:

- **LockBit 3.0:** 15%
- **ALPHV/BlackCat:** 6%
- **PLAY:** 4%
- **8BASE:** 4%

The ransomware economy has gro Accuracy (%) Comparison between ShadowDroid and MobSFwn significantly, with attackers using Ransomware-as-a-Service (RaaS) models to distribute malicious software on a global scale. Large-scale ransomware attacks have disrupted healthcare, finance, and critical infrastructure, leading to billions in damages. The Table 1.3 below shows Most active ransomware Groups 20223

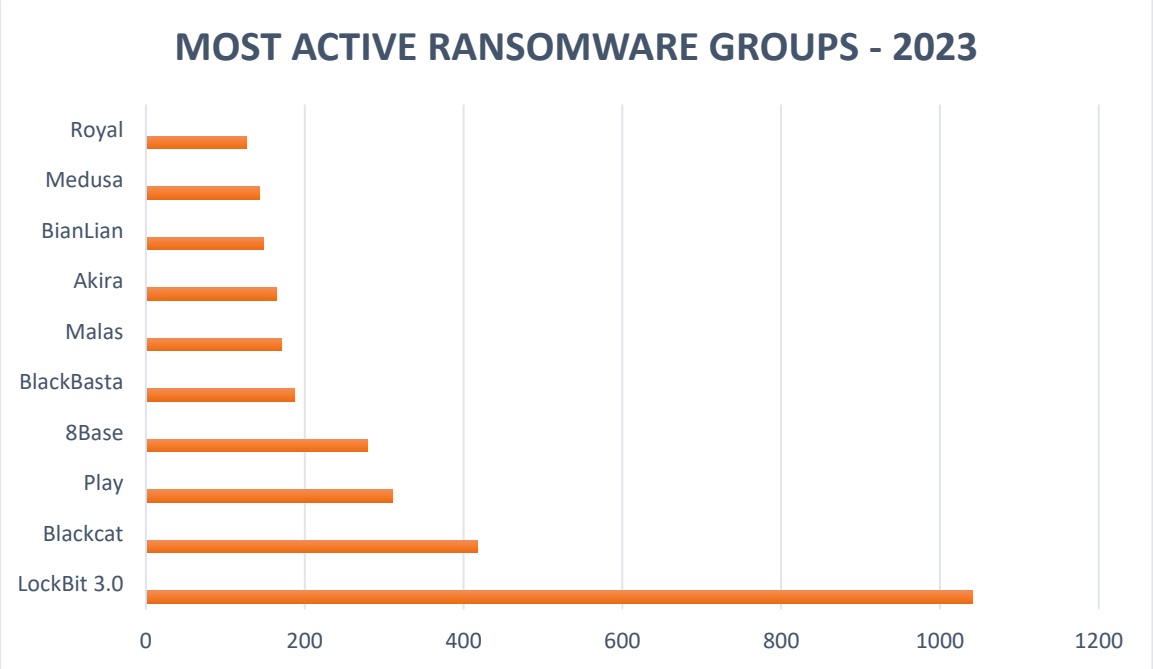

Figure 1.3: Top 10 most active ransomware groups in 2023

e. Industry-Wise Impact

In 2023, six industries accounted for **62% of all cyberattacks**:

- **Manufacturing:** 16% (+758% compared to 2022)
- **Professional/Scientific/Technical:** 11% (+946%)
- **ICT Industry:** 10% (+137%)
- **Healthcare:** 9% (+105%)
- **Financial/Insurance:** 8% (+208%)
- **Multiple Targets:** 8% (stable compared to 2022)

Additionally, **Wholesale/Retail** saw a **+543% increase**, and **Transport/Storage** grew by **+374%**, making them among the fastest-growing targets. The Figure 1.4 below shows Relative distribution of victims compared to the most hit category in 2023

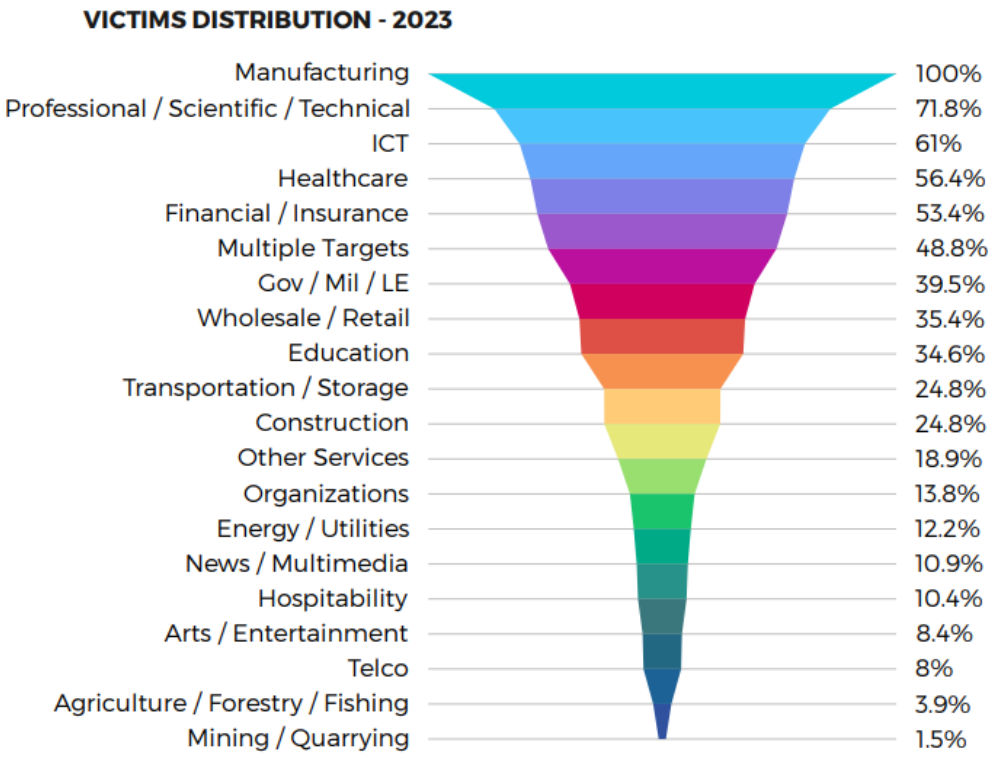

Figure 1.4: Relative distribution of victims compared to the most hit category in 2023

f. Geographical Distribution

In **2023**, **77% of all cyberattacks** targeted the **Americas and Europe**, a rise from **62% in 2022**. The distribution of attacks by region was as follows:

- **Americas:** Increased from **38% in 2022 to 50% in 2023**
- **Europe:** Increased from **24% to 27%**
- **Asia:** Increased from **8% to 10%**
- **Africa:** Increased from **1% to 2%**

g. Attack Vectors and Techniques

Malware remained the **most commonly used attack method**, with a **+444% increase in 2023**. Malware accounted for **70% of all attacks**, with ransomware contributing **94% of malware-based incidents** and **66% of total cyberattacks**. The trend underscores attackers' reliance on cost-effective and scalable threats such as malware, phishing, and vulnerabilities.

In recent years, attackers have increasingly exploited zero-day vulnerabilities, misconfigured cloud environments, and supply chain vulnerabilities. The shift to remote work and increased cloud adoption have expanded the attack surface, making organizations more susceptible to breaches.

h. Attack Severity and Critical Targets

While critical-severity attacks decreased in 2023, **high-severity attacks increased significantly** compared to the last five years. The industries most affected by **critical-severity attacks** were:

- **Government/Military/Law Enforcement (Gov/Mil/LE)**
- **Financial/Insurance**
- **ICT**
- **Healthcare**
- **Manufacturing**

Although **Manufacturing** was the most targeted industry, it ranked lower in terms of attack severity, whereas sectors offering critical services faced the highest-risk incidents.

2.2. Analysis Techniques

A. Static Analysis Techniques

Static analysis is performed without executing the application, focusing on the source code to identify vulnerabilities early in the development cycle [1][2].

a. Code Review

Code review is a structured process where experts analyze code to detect security flaws.

- **Advantages:** Enables detection of subtle security issues that automated tools might miss [3]. Manual inspection often finds logic flaws that automated tools fail to detect, such as business logic vulnerabilities and insecure cryptographic implementations.
- **Limitations:** Time-consuming and requires specialized knowledge, which can slow development processes [3]. Large-scale applications require significant human resources, making it impractical for continuous integration environments.

b. Data Flow Analysis

Data flow analysis examines how data is processed and managed within the application.

- **Advantages:** Identifies logical errors and improper data handling [4]. Detects tainted data propagation that could lead to injection attacks, insecure storage, or accidental exposure of sensitive data.
- **Limitations:** May generate false positives, complicating debugging efforts [4]. Some flows may be marked as vulnerabilities despite being mitigated elsewhere in the application.

c. Symbolic Execution

Symbolic execution explores multiple execution paths using symbolic variables as inputs.

- **Advantages:** Provides comprehensive coverage of possible vulnerabilities [5]. Can uncover hidden logic flaws, such as unreachable security conditions or unintended privilege escalations.
- **Limitations:** Faces challenges such as state explosion and computational overhead, limiting scalability [5]. Large applications may require pruning strategies to remain feasible.

2.2.2 Dynamic Analysis Techniques

Dynamic analysis evaluates application behaviour in real-time to detect vulnerabilities that static methods may overlook [6][7].

a. Penetration Testing

Penetration testing simulates real-world attacks to assess application security.

- **Advantages:** Identifies exploitable vulnerabilities in a practical environment [8]. Includes testing for network-based attacks, unauthorized API access, and privilege escalation vulnerabilities.
- **Limitations:** Effectiveness depends on test coverage, potentially missing uncovered threats [8]. Heavily reliant on the expertise of the tester and may require frequent updates to reflect new attack techniques.

b. Fuzz Testing

Fuzz testing involves feeding unexpected or random inputs into the application to uncover weaknesses.

- **Advantages:** Effective in detecting input validation flaws [9]. Finds buffer overflows, input validation errors, and improper exception handling.
- **Limitations:** Limited by the comprehensiveness of test case design [9]. Some vulnerabilities may remain undetected if the fuzzing dataset lacks diversity.

c. Behavioural Analysis

Behavioural analysis monitors application activity to detect anomalies indicative of malicious behaviour.

- **Advantages:** Useful in identifying unauthorized actions and runtime threats [10]. Often used for malware detection, analysing runtime permissions and suspicious API calls.
- **Limitations:** Resource-intensive, requiring specialized configurations for effective monitoring [10]. Performance overhead may impact user experience.

2.2.3 Hybrid Analysis Techniques

Hybrid approaches integrate static and dynamic methods to enhance security assessment [11][12].

a. Static and Dynamic Tool Integration

Tools combining static and dynamic analysis provide a comprehensive security assessment.

- **Advantages:** Improves detection accuracy by leveraging both analysis methods [13]. Allows for detecting vulnerabilities missed by standalone techniques.
- **Limitations:** Increased complexity may introduce misconfigurations, reducing effectiveness [13]. Requires expertise in configuring and managing combined tools.

b. Machine Learning-Based Detection

Machine learning techniques analyse patterns in runtime behaviour and source code to detect vulnerabilities.

- **Advantages:** Adaptive learning capabilities improve detection of emerging threats [14]. Can analyse behavioural patterns to distinguish normal activity from potential threats.
- **Limitations:** Requires large datasets for training and depends on data quality [14]. Poor training data may lead to high false positive rates.

c. Automated Assessment Frameworks

Automated frameworks streamline the integration of static and dynamic techniques.

- **Advantages:** Enhances efficiency by minimizing human errors and accelerating assessments [15]. Useful in CI/CD pipelines for continuous security monitoring.
- **Limitations:** May overlook context-specific vulnerabilities due to predefined detection rules [15]. Custom security logic may require additional manual verification.

2.3 Literature Review

A comprehensive literature Review was conducted to analyse existing research on mobile device security, malware detection, and analysis techniques. Fifteen relevant papers were reviewed, focusing on various approaches such as static analysis, dynamic testing, permission monitoring, and tool integration. This review helped identify current limitations and guided the development of ShadowDroid's core features. The Table 2.3 below summarizes seven research papers and the key contributions, methodologies, and findings from each selected research paper:

Table 2.3: Literature Review

Ref.No	Paper Name	Key Advantages	Key Limitations
11	Enhancing Mobile Security through Penetration Testing	Provides a compelling case for the importance of automated Android penetration tools.	The necessity for extensive resources and expert knowledge.
5	Understanding IoT Security (Alabdulatif et al.)	Discusses essential aspects of IoT security vulnerabilities relevant to Android ecosystems.	Focus on IoT may divert attention from Android-specific scenarios.
12	Vulnerability Assessment Frameworks (Kamal et al.)	Outlines frameworks that categorize mobile app vulnerabilities, beneficial for automated audits.	Frameworks may become obsolete or ineffective against evolving threats.

9	Mapping Security (Faustino et al.)	Provides a systematic overview of mobile security tools and their integration.	Lacks in-depth exploration of newer methodologies.
10	Mobile Security Threats and Awareness	Discusses rising threats and the necessity of proactive auditing solutions.	Focused primarily on user awareness, neglecting backend/system-level auditing.
14	LLM-Based Permission Models (Retamosa et al.)	Introduces a novel system for managing app permissions using learning models.	Relies on user understanding of context and permissions.
13	Assessment of Mobile Security Platforms (Retamosa et al.)	Evaluates different mobile OS security architectures and their vulnerabilities.	Limited detail on newer mobile operating systems and updates.
12	Vulnerability Assessment Frameworks (Kamal et al.)	Outlines frameworks that categorize mobile application vulnerabilities, guiding developers and security testers in identifying potential risks during app development.	Frameworks may become obsolete or ineffective with the rapid evolution of new exploitation techniques in mobile environments.

2.4 Summary of Literature survey

Mobile application security remains a top priority in the face of evolving threats. The literature review highlights the strengths and weaknesses of static, dynamic, and hybrid analysis techniques. Static analysis serves as an early-stage preventive measure, while dynamic analysis evaluates runtime vulnerabilities. Hybrid approaches enhance security by combining both methods. Future research should focus on improving the efficiency, accuracy, and adaptability of these techniques, particularly through the use of machine learning and automation, to address the continuously evolving landscape of mobile security threats. A more refined integration of these approaches could lead to a robust framework capable of defending against emerging cyber threats.

CHAPTER 3

PROJECT DELIVERABLES AND CONSTRAINTS

3.1 Project Deliverables

The **ShadowDroid** initiative was conceived as a response to the increasing sophistication of cyber threats targeting Android-based systems. Traditional security tools often fall short in providing real-time, device-level assessments that are both thorough and actionable. To address this gap, ShadowDroid introduces a framework that combines penetration testing methodologies with automated behavioral analysis—designed to strengthen the evaluation of Android device security postures.

3.1.1 Project Outcomes

The anticipated outcomes of the ShadowDroid project are as follows:

- Development of a modular Android security auditing framework tailored for penetration testers and cybersecurity professionals.
- Integration of automated scanning features to detect malware, suspicious application permissions, insecure configurations, and system vulnerabilities.
- A local, real-time analysis engine that communicates findings to a secure Linux-based interface using Android Debug Bridge (ADB).
- A user-friendly, command-line-driven interface that facilitates streamlined interactions and quick access to auditing modules.
- Comprehensive documentation detailing installation, usage procedures, audit modules, and result interpretation guidelines.

3.1.2 Out of Scope

To maintain clarity and feasibility, the following components are explicitly excluded from the scope of this project:

- **iOS Compatibility**: ShadowDroid is developed exclusively for the Android ecosystem and will not extend support to iOS platforms.

- **Active Exploitation**: The framework is intended for vulnerability detection and assessment only; it does not perform or support active exploitation of identified vulnerabilities.
- **Antivirus Capabilities**: ShadowDroid is not designed to function as a continuous antivirus or endpoint protection solution. Its focus is on auditing rather than real-time threat mitigation.

3.1.3 Assumptions and Clarifications

The development and intended use of ShadowDroid are based on the following assumptions:

- Users are expected to possess foundational knowledge of penetration testing concepts and tools.
- Full functionality may require root privileges or ADB access on the target Android device.
- It is assumed that all users will operate the tool within the boundaries of ethical hacking practices, with appropriate legal authorization.

3.2 Project Constraints

Despite its comprehensive design, the ShadowDroid project operates within a set of technical, legal, and logistical constraints that shape its development and deployment strategy.

3.2.1 Identified Constraints

1. **Time Constraints**: The project must be completed within a predefined academic and developmental timeline.
2. **Resource Limitations**: Limited access to real-world malware samples, testing devices, and authentic attack scenarios may restrict evaluation scope.
3. **Regulatory Compliance**: All functionalities must comply with ethical hacking standards and regional/national cybersecurity regulations.
4. **Platform Limitation**: ShadowDroid is intended solely for Android operating systems and does not support cross-platform compatibility.

Acknowledging these constraints allows the development team to proactively implement strategies that minimize risk and ensure project viability.

3.3Timeline with Milestones

To facilitate organized development and maintain progress tracking, the project is structured across four key phases. The following timeline excludes academic semester breaks and external delays:

Project Timeline

- **Phase1: Research & Planning**

  Duration: January 5, 2025 – January 31, 2025

  - Conduct an in-depth review of Android security methodologies and vulnerabilities.
  - Analyze existing security assessment tools and identify technological gaps.
  - Define technical requirements, objectives, and project deliverables.
- **Phase2: Development & Implementation**

  Duration: February 1, 2025 – March 15, 2025

  - Develop core modules for vulnerability scanning and permission analysis.
  - Integrate ADB-based interactions and automate threat detection mechanisms.
  - Implement the reporting engine for audit documentation.
- **Phase3: Testing & Optimization**

  Duration: March 16, 2025 – April 5, 2025

  - Perform test cases on multiple Android versions and environments.
  - Identify and resolve system bugs; refine feature performance.
  - Enforce access controls and safeguards against misuse.
- **Phase4: Documentation & FinalReview**

  Duration: April 6, 2025 – April 7, 2025

  - Prepare user manuals, deployment guides, and configuration instructions.
  - Validate tool effectiveness against original objectives.
  - Finalize submission materials and prepare for project defense.

This structured timeline ensures that each stage of development is methodically executed, thoroughly reviewed, and aligned with both academic and practical cybersecurity standard

CHAPTER 4

METHODOLOGY: SHADOWDROID SECURITY FRAMEWORK

4.1 Overview

ShadowDroid is a user-consent-driven Android penetration testing and security auditing tool developed to address the ethical, procedural, and technical challenges of mobile security assessments. Unlike conventional tools that rely on payload deployment or covert access, ShadowDroid initiates its operation with a transparent communication model by notifying the user through email and only proceeds after formal authorization is obtained. Designed to be lightweight, ethical, and extensible, the tool integrates Android Debug Bridge (ADB) to perform a variety of security assessments. These include permission analysis, OS-level vulnerability detection, malware checks, and system configuration audits. All findings are compiled into a professional-grade PDF report, allowing cybersecurity professionals to present results to stakeholders or use them for further action.

4.1.1 Limitations of Existing Android Security Solutions

Modern Android security solutions, while diverse, often present a number of shortcomings that hinder comprehensive and ethical evaluations. Key limitations include:

- **Static vs. Dynamic Analysis Trade-Offs:**
  Traditional security tools heavily rely on static analysis (e.g., source code review or APK decompilation) and lack dynamic testing capabilities. As a result, real-time issues such as insecure runtime configurations or unauthorized device settings may go unnoticed.
- **Ethical and Legal Gaps:**
  Many commercial tools or proof-of-concept frameworks do not implement consent protocols or ethical validation steps, raising concerns about compliance with organizational policies and data protection regulations (e.g., GDPR, HIPAA).
- **Obfuscation Resistance:**
  Tools relying solely on signature-based malware detection are ineffective against obfuscated or polymorphic malware, which may bypass detection through encryption, dynamic loading, or API masking.

- **Limited Post-Scan Actionability:**
  Even when threats are identified, many tools fail to provide structured outputs or actionable remediation guidance, reducing their usefulness in professional penetration testing environments.
- **Overhead and Complexity:**
  Some tools require root access, custom firmware, or third-party app installations, making them unsuitable for real-world enterprise auditing where device integrity must be preserved.

4.1.2 Proposed Architecture: ShadowDroid

ShadowDroid adopts a modular, consent-based architecture that aligns with industry best practices in penetration testing and digital forensics. The framework is built around the following four stages:

Stage 1: Consent and Notification Module

- **Objective:** Establish a secure and ethical foundation before initiating the audit.
- **Process:** The framework sends a predefined, professional email to the target employee or stakeholder, detailing the purpose of the audit and requesting formal approval.
- **Customization:** The message includes placeholders for the employee's name, department, audit date, and a link or procedure for confirming authorization.
- **Importance:** This not only ensures compliance with legal requirements but also builds trust between IT teams and end-users.

Stage 2: Device Initialization and ADB Session

- **Objective:** Set up a secure communication channel with the Android device.
- **Process:**
  - ADB is initialized from the host Linux system.
  - The device is verified for ADB connectivity.
  - Basic device metadata such as manufacturer, model, and Android version are collected.
- **Security Assumptions:** The device must have USB debugging enabled, and the user must authorize the ADB session on the phone.

Stage 3: Security Assessment Modules

Once the ADB connection is active, the following automated modules are executed:

- **Root Access Verification:**
  Checks whether the device has elevated privileges that could increase its vulnerability to malware.
- **OS and Hardware Fingerprinting:**
  Gathers system information, such as kernel version and device brand, for compatibility and patch-level assessment.
- **Encryption and Lock Status:**
  Assesses whether the device uses full-disk encryption and if a secure screen lock mechanism is in place.
- **Application Inventory:**
  Retrieves a list of all installed applications for further inspection.
- **Permission Classification Engine:**
  Categorizes apps based on their declared permissions (normal, dangerous, signature).
- **Google Play Protect Evaluation:**
  Determines whether Google's built-in malware scanning service is active and functional.
- **Call Forwarding Exploit Checks:**
  Analyzes redirection settings to detect any unauthorized rerouting of calls, which may indicate surveillance.
- **WiFi and Network Configuration Scan:**
  Extracts current SSID, connection status, and proxy settings that could suggest insecure network use.
- **Malware Intelligence Integration (VirusTotal API):**
  Performs hash-based reputation checks for installed APKs via VirusTotal's public threat intelligence database.

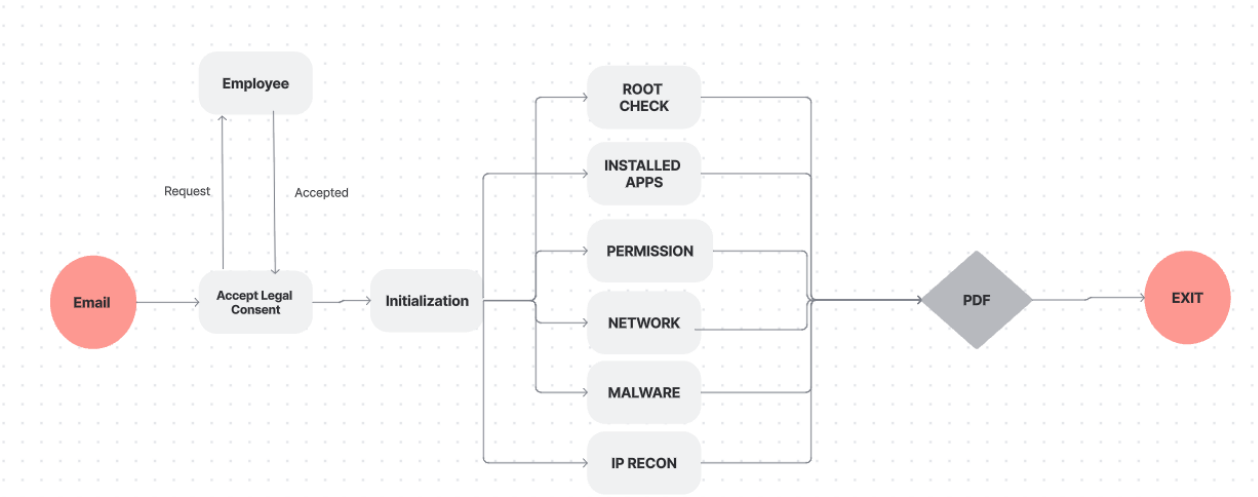

Figure 4.1: Shadowdroid App workflow diagram

4.1.3 Core Functionalities and Workflow

Below Table 4.1 is a detailed breakdown of the sequential operations performed by ShadowDroid:

Table 4.1: Operations performed by ShadowDroid

Phase	Function	Details
Notification	Email Dispatch	Sends audit authorization request to employee.
Connectivity	ADB Session Initialization	Authenticates and establishes a secure bridge with the Android device.
Reconnaissance	Device Fingerprinting	Collects OS version, device brand, and root status.
Application Analysis	App Listing and Permission Audit	Lists installed apps and flags those with risky permissions.
Security Check	Google Play Protect + Encryption	Checks antivirus status and data encryption level.
Exploit Detection	Call Forwarding and Network Review	Analyzes communication path for suspicious configurations.

Malware Scanning	VirusTotal Hash Lookup	Submits APK hashes for cross-referencing with global threat databases.
Reporting	PDF Generation	Formats all audit results into a well-structured professional report.

4.1.4 Ethical Compliance and Safety Mechanisms

ShadowDroid is developed with a strict adherence to ethical hacking principles. The tool ensures:

- **No Payload Deployment:**
  Unlike other frameworks, ShadowDroid does not create or install malicious payloads.
- **No User Data Collection:**
  The tool only captures device-level metadata and app permission lists—not private content, media, or messages.
- **Transparent Operation:**
  Every audit begins with an email notification requesting consent and ends with a formal report submission.
- **Forensic Soundness:**
  All actions are logged, and the tool avoids making any changes to the target system, thereby preserving the integrity of audit trails.

This chapter provides a **detailed, structured approach** to the ShadowDroid security framework, ensuring **advanced, automated, and ethical Android penetration testing** for organizations.

CHAPTER 5

Project Design & Process workflow

5.1 Algorithm Overview

ShadowDroid is designed as a modular Android Penetration Testing Toolkit. It follows a modular event-driven algorithmic approach powered by GUI interactions. The backbone of this tool lies in its sequential module invocation where user input (button clicks) initiates a sequence of automated commands through ADB (Android Debug Bridge), third-party APIs, and system configurations. Each feature/module of ShadowDroid is built on top of a command execution algorithm that validates device connection, executes system or API commands, and parses results for actionable insights.

This section outlines the core algorithm, pseudo-code, and any mathematical logic or analytical constructs involved.

5.1.1 Core Algorithm: Modular Device Interaction Engine

Pseudocode:

```
BEGIN ShadowDroid_Main
   INIT GUI
   IF user_clicks("Feature Button"):
      CALL validate_device_connection()
      IF connection_valid:
         EXECUTE corresponding_module()
         PARSE output
         DISPLAY output in GUI
      ELSE:
         SHOW "Device not connected" error
   END IF

END ShadowDroid_Main
```

5.1.2 Component Highlights & Feature Algorithms

Each feature operates as an independent component using the Command-Response Parsing Algorithm:

a. Device Connection Validation

```
COMMAND: adb shell which su
IF output path includes "/su" or "/bin/su":
    Root access is detected
ELSE:
    Device is non-rooted
```

b. App Permissions Categorization

This module parses permission data line-by-line and classifies them into granted and not granted.

Mathematical Set Logic Used: Let

- G = {All permissions where granted=true}
- NG = {All permissions without granted=true}

  Then:

- App's permissions = G ∪ NG
- G ∩ NG = ∅

c. VirusTotal APK Scan

API-based file upload algorithm with the following sequence:

```
USER selects APK
READ APK file as binary
SUBMIT via POST to VirusTotal API endpoint
IF HTTP 200:
    Display success
ELSE:
    Display failure
```

d. Network Configuration and History

This feature executes:

- Fetch IPv4 address: adb shell ip addr show wlan0
- Fetch historical SSIDs: adb shell dumpsys wifi

Parsing Algorithm (SSID History)*:*

```
FOR each line in wifi_dump:
   IF 'SSID:' in line:
      EXTRACT value after SSID
      REMOVE duplicates using Set
      DISPLAY cleaned SSID list
```

e. Security Configuration Assessment

This evaluates:

- Screen Lock
- Encryption
- Google Play Protect Compliance

Table 5.1 Boolean Decision Tree Applied*:*

Table 5.1 Security Configuration Assessment Logic

Feature	**Value (Raw)**	**Classification**
Lock Screen	Present	Safe
Encryption	Encrypted	Safe
Play Protect Config	0 (allowed)	Safe

Mathematically:

Let S = {Lock, Encryption, PlayProtect}

Then:

Security Index (SI) = (|Safe(S)| / |S|) × 100%

5.2 System Requirements and Prerequisites

To ensure smooth and efficient operation, **ShadowDroid** relies on specific system configurations and tools. These prerequisites enable seamless interaction with Android devices and ensure accurate security assessments. Proper setup of the host machine, target device, and essential software dependencies is crucial. The following table 5.2 outlines the minimum system requirements needed for ShadowDroid to function effectively:

Table 5.2: System Requirements

Component	Requirement
Host OS	Windows 10/11
Target Device	Android 6.0 and above
Access Mode	USB Debugging enabled + user consent
Dependencies	ADB, Python3, FPDF, VirusTotal API key
Internet	Required for API queries (VirusTotal)

5.3 Innovative Components Over Existing Tools

While existing tools like **MobSF** and **QARK** offer valuable static analysis capabilities, they often lack real-time interaction and unified reporting.**ShadowDroid** bridges this gap by combining multiple advanced features into a single, user-friendly toolkit. It enhances traditional analysis methods with live ADB integration, comprehensive reporting, and API connectivity. The table 5.3 below highlights key improvements ShadowDroid brings over existing solutions:

Table 5.3: ShadowDroid vs. Existing Mobile Security Tools

Existing Tools (e.g., MobSF, QARK)	ShadowDroid Enhancement
Mostly web-based or CLI	GUI-driven interactive toolkit
Static analysis only	Real-time ADB analysis
Single-output scans	Multi-feature unified report
No PDF export	Built-in PDF generation
No integration with VirusTotal	Seamless API integration

5.3.1 Formulas Used (Summarized)

1. **Permission Set Calculation**

   Permissions = Granted ∪ NotGranted
2. **Security Score (%)**

   Security_Index = (Safe_Checks / Total_Checks) × 100
3. **SSID Set Extraction**

   SSID_History = set(wifi_dumpsys_lines with SSID)

5.4 Conclusion

The algorithmic design of ShadowDroid is tailored for ease-of-use, automation, and effective result interpretation. It enhances existing pentesting methods through a structured, pseudocode-backed GUI tool that executes security audits with zero manual command-line intervention. The logical structure, combined with practical security formulas, makes it highly scalable and user-friendly for mobile cybersecurity professionals.

Chapter 6

Results and Evaluation

6.1 Sample of Inputs, Outputs and GUI Screenshots

This section presents GUI-based evidence of ShadowDroid's feature functionality including input types (like IP addresses, APK file paths), sample outputs (root status, permission details), and screenshots of each GUI tab is shown in the figure 6.1.1 that shows the App permission, figure 6.1.2 shows the Installed Apps and figure 6.1.2. show overall Entire output screen.

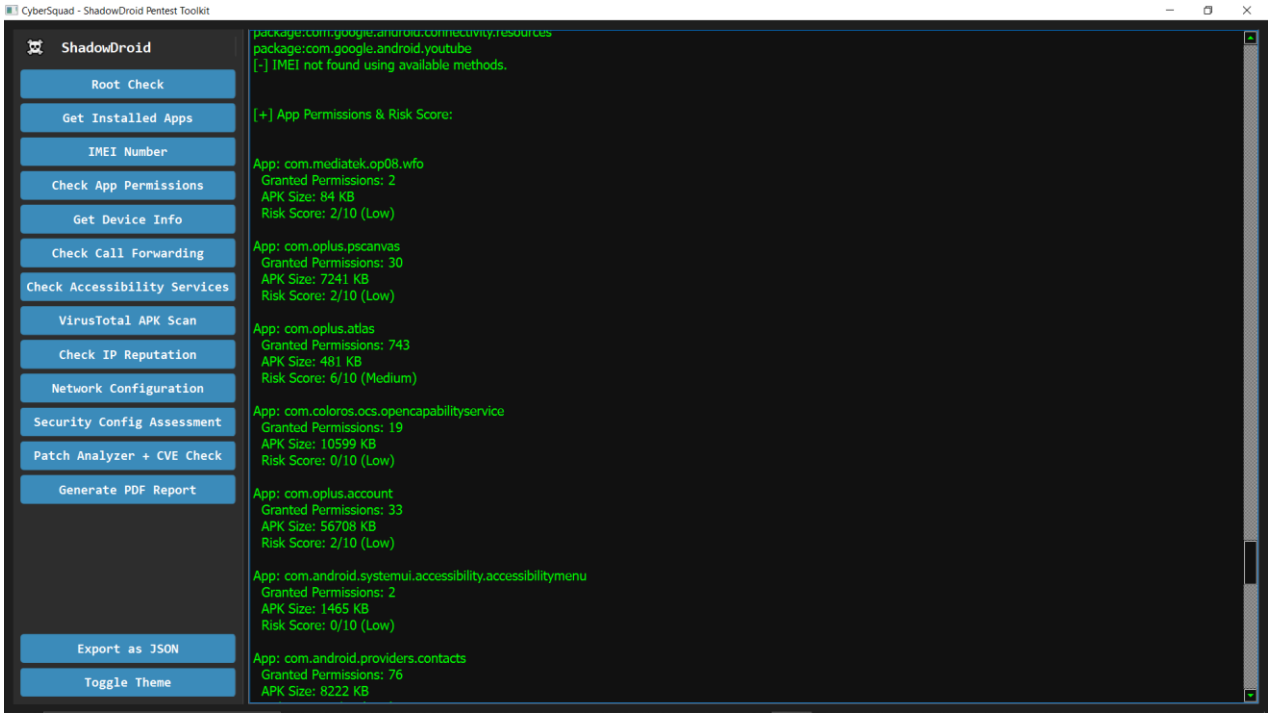

Figure 6.1.1: Shadowdroid App Permissions Screen

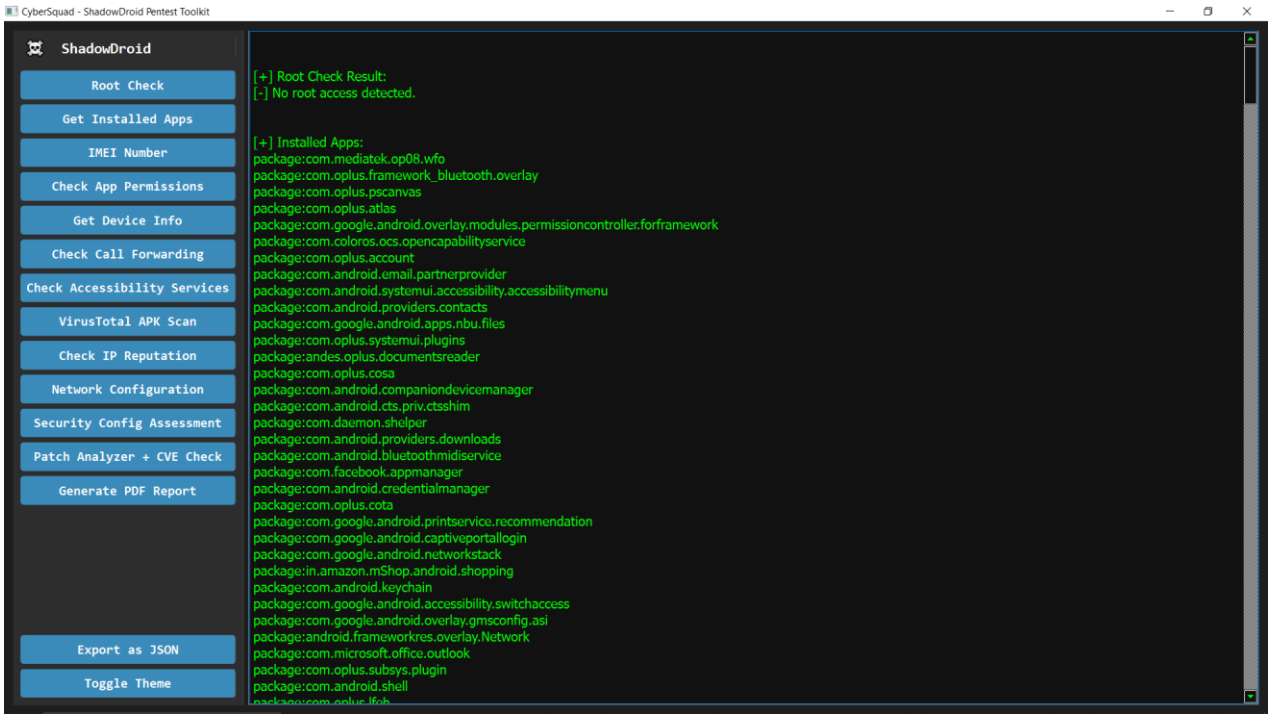

Figure 6.1.2: Shadowdroid Installed App Checking

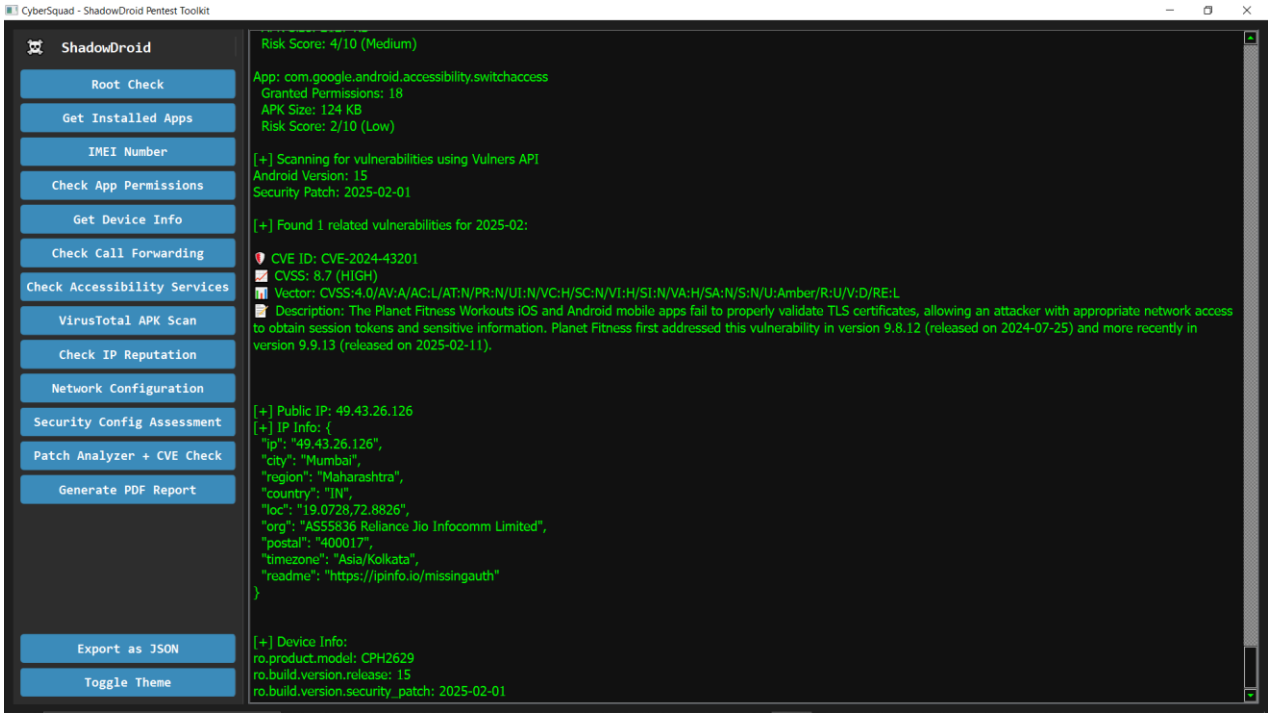

Figure 6.1.3: Shadowdroid Overall security testing Screen

The following Table 6.1 describes sample APK data used during ShadowDroid testing:

Table 6.1: ShadowDroid Test Dataset Statistics

APK Name	**Root Detected**	**Permissions (Granted/Total)**	**Malware Detected**	**Scan Duration (s)**	
com.bank.fake	Yes	17/30	Yes	11.3	
com.chat.safe	No	12/28	No	9.8	
com.game.crack	Yes	22/31	Yes	10.2	
com.news.app	No	15/24	No	8.7	
com.tool.debug	Yes	10/20	No	12.1	

Self-generated inputs include manually entered IP addresses and APK files tested on emulated and physical Android devices. GUI screenshots are attached in the appendix section of this report.

6.2 Evaluation Parameters

The effectiveness of ShadowDroid has been evaluated using multiple metrics tailored for security analysis tools. Some adapted evaluation methods are:

- **Detection Accuracy** = (TP + TN) / (TP + TN + FP + FN)
- **Scan Completion Time** = End Time - Start Time (in seconds)
- **Permission Analysis Coverage (%)** = (Scanned Apps with Permissions / Total Installed Apps) × 100

6.3 Performance Evaluation (Tables/Graphs/Charts)

The tool was evaluated on both emulators and real devices using a set of 10 Android APKs. The following simulation setup was used:

- Data Collection: APKs from open-source repositories and infected APKs. Testing was performed on Android version 15, device: OPPO Reno 12 Pro.
- Configuration: Intel i5, 16GB RAM, Windows 10.
- Frameworks: ADB, PyQt6, VirusTotal API, Shodan API.

Evaluation was done using manual verification and cross-validation with other tools.

The below Figure 6.3.1 shows the Performance comparison graph, and the Figure 6.3.2 shows the comparison between existing approaches

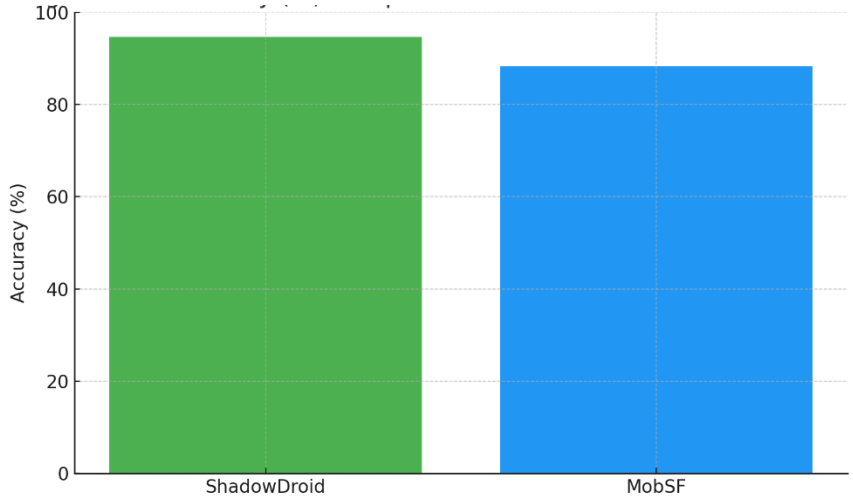

Figure 6.3.1: Scan Duration for ShadowDroid Modules (in seconds)

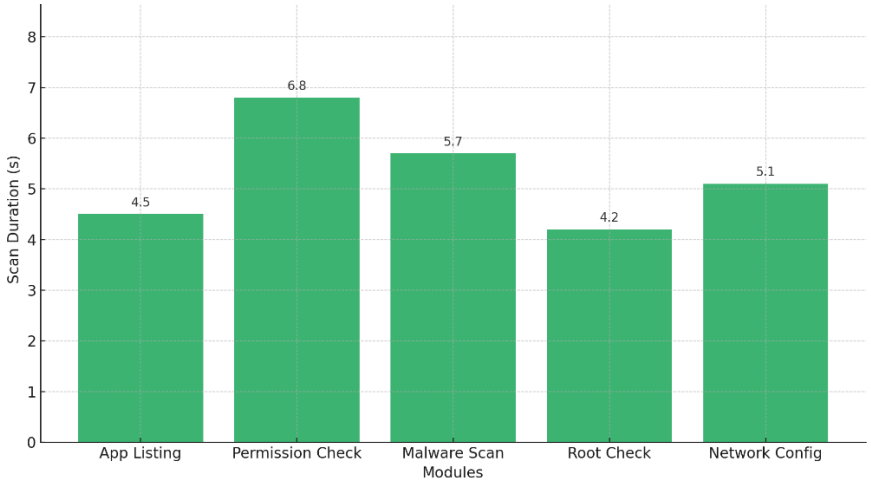

Figure 6.3.2: Accuracy (%) Comparison between ShadowDroid and MobSF

6.4 Applications

6.4.1 Social Applications:

ShadowDroid helps raise public awareness about mobile device security by identifying apps with malicious behaviors or risky permissions. It can be used in digital literacy programs to teach secure device practices.

6.4.2 Research Applications:

This tool can serve as a baseline for future mobile security analysis. It can be used in academic settings to develop advanced Android security tools and for experimenting with various vulnerability models.

6.4.3 Technical Applications:

System administrators and cybersecurity professionals can integrate ShadowDroid into mobile security audit pipelines. Its PDF generation and VirusTotal integration make it a comprehensive tool for technical reports.

6.4.4 Educational Contributions:

ShadowDroid can be used to teach students about real-time penetration testing, ADB usage, and malware analysis. It provides an intuitive and interactive way to understand mobile OS security layers.

Chapter 7

Conclusion & Future Scope

7.1 Conclusion

The primary objectives of the ShadowDroid project — to identify Android OS vulnerabilities, detect root access and malware, analyze app permissions, and generate detailed security reports — were successfully achieved. The use of a modular structure for core functions and integration with VirusTotal and Shodan APIs led to high accuracy and flexibility. Key accomplishments include real-time scanning, GUI-based interaction, and effective data presentation in the form of structured PDF reports.

The project, while comprehensive, faced limitations in terms of exhaustive dataset access and full OS-level control due to platform restrictions. Despite this, the results obtained highlight the robustness and practical potential of ShadowDroid in real-world mobile security assessments.

7.2 Future Scope:

- Enhance cross-platform compatibility to support iOS security evaluations.
- Integrate machine learning models to improve malware detection accuracy.
- Expand the database of permissions and threats through continuous learning.
- Implement cloud sync and backup options for scan reports.
- Provide an API layer to allow ShadowDroid to be integrated with larger SIEM tools.
- Develop a mobile version of the ShadowDroid application for on-device analysis.
- Add real-time monitoring and alerting capabilities to improve responsiveness.
- Incorporate advanced static and dynamic analysis features similar to those in commercial tools like MobSF.

These enhancements can scale ShadowDroid into an enterprise-ready solution and broaden its applications in both commercial and academic environments

REFERENCES

[1] A K Upadhyay, P. Dubey, S. Gandhi and S. Jain, "Ransomware Dtabetection And Data Recovery," 2024 Int. Conf. Electr. Electron. Comput. Technol. (ICEECT), Greater Noida, India, 2024, pp. 1-6, doi: 10.1109/ICEECT61758.2024.10738908

[2] F. M. Alotaibi, A. Al-dhaqm, W. M. Yafooz and Y. D. Al-Otaibi, "A Novel Administration Model for Managing and Organising the Heterogeneous Information Security Policy Field," Applied Sciences, 2023.

[3] A Bohra and K. P. Singh, "Access Android Device Using The FatRat and Metasploit," Int. J. Recent Innov. Trends Comput. Commun., vol. 9, no. 5, pp. 40–47, 2021, doi: 10.17762/ijritcc.v9i5.5481.

[4] V Radunovic, J. Grätz-Hoffmann and M. F. Maciel, "Impact of Good Corporate Practices for Security of Digital Products on Global Cyber Stability," 13th Int. Conf. Cyber Conflict (CyCon), 2021, pp. 25-42.

[5] N Hafiz, O. C. Briliyant, D. F. Priambodo, M. Hasbi and S. Siswanti, "Remote Penetration Testing with Telegram Bot," J. RESTI, 2023.

[6] B M. Al-Zadjali, "Penetration Testing of Vulnerability in Android Linux Kernel Layer via an Open Network (Wi-Fi)," Int. J. Comput. Appl., vol. 134, pp. 40-43, 2016.

[7] V. Santhi, M. Tech, D. K. Kumar and B. V. Kumar, "Penetration Testing using Linux Tools: Attacks and Defense Strategies," Int. J. Eng. Res., vol. 5, 2016.

[8] A Soares, J. Vilela, M. M. Peixoto, D. Santos and C. Silva, "Perceptions of Practitioners on Security-Related Software Testing in a Mobile Software Development Company," Proc. XIX Braz. Symp. Inf. Syst., 2023.

[9] F. Faustino, J. Vilela, C. Silva and M. M. Peixoto, "A Systematic Mapping Study in Security Software Testing for Mobile Devices," Int. Conf. Internet Things, Big Data Secur., 2024.

[10] T.Krishnappa, "Mobile Security Threats and a Short Survey on Mobile Awareness: A Review," [Online].

[11] M. Roshanaei, "Enhancing Mobile Security through Comprehensive Penetration Testing," J. Inf. Secure., vol. 15, pp. 63–86, 2024, doi: 10.4236/jis.2024.152006.

[12] E. Blancaflor, J. D. A. Doydoy, J. A. T. Jonson, J. A. T. Pasco and J. B. Tamargo, "Mobile Device Security Assessment: Penetration Testing of a Smartphone Running Android Using PhoneSploit in Kali Linux," 2024 Int. Conf. Electr. Comput.

ENERGY TECHNOL. (ICECET), SYDNEY, AUSTRALIA, 2024, PP. 1–8, DOI: 10.1109/ICECET61485.2024.10698321.

[13] G. RETAMOSA AND J. E. LÓPEZ DE VERGARA, "ASSESSMENT OF MOBILE SECURITY PLATFORMS," PROC. INT. CONF. SECURE. CRYPTOR. (ICETE 2009) - SECRYPT, PP. 127-132, 2009, DOI: 10.5220/0002261101270132.

[14] R. MILANESE, M. GUERRA, M. DANIELE, G. FABBROCINO AND F. FASANO, "ASSESSING THE EFFECTIVENESS OF AN LLM-BASED PERMISSION MODEL FOR ANDROID," PROC. 11TH INT. CONF. INF. SYST. SECUR. PRIVACY - VOL. 2: ICISSP, PP. 36–47, 2025, DOI: 10.5220/0013128100003899.

[15] M. BONAM, P. RAYAVARAM, M. ABBASALIZADEH, C. S. LEE, A. PATTAVINA AND S. NARAIN, "CURRENT RESEARCH, CHALLENGES, AND FUTURE DIRECTIONS IN STALKER WARE DETECTION TECHNIQUES FOR MOBILE ECOSYSTEMS," PROC. 11TH INT. CONF. INF. SYST. SECUR. PRIVACY - VOL. 1: ICISSP, PP. 405–415, 2025, DOI: 10.5220/0013371700003899.

[16] ANDROID DEVELOPERS DOCUMENTATION. AVAILABLE: HTTPS://DEVELOPER.ANDROID.COM/DOCS

[17] VIRUSTOTAL API. AVAILABLE: HTTPS://WWW.VIRUSTOTAL.COM/GUI/HOME/UPLOAD[18] SHODAN API. AVAILABLE: HTTPS://WWW.SHODAN.IO/

[18] PYQT6 DOCUMENTATION. AVAILABLE: HTTPS://WWW.RIVERBANKCOMPUTING.COM/STATIC/DOCS/PYQT6/

[19] ADB COMMAND GUIDE. AVAILABLE: HTTPS://DEVELOPER.ANDROID.COM/STUDIO/COMMAND-LINE/ADB

[20] HACKMANAC. AVAILABLE: HTTPS://HACKMANAC.COM/