Mobile App Security Analyzer

CYSE3

Rupert Agoyaoy

500227341

Ajitesh Kuppa Sivakumar

500230219

Puneet Kaur

500231967

Shikshya Gautam

500227425

Jaspreet Kaur

500220252

Project Advisor

Suranjit Paul

April 6, 2025

Abstract

The Mobile App Security Analyzer project presents a comprehensive framework designed to assess and enhance the security posture of mobile applications across various platforms, including iOS and Android. As mobile applications increasingly handle sensitive user data, the significance of robust security measures has escalated, making them prime targets for cyber threats. This project aims to identify vulnerabilities, analyze potential threats, and provide actionable recommendations to fortify mobile app security. Through a detailed literature review, the advantages and disadvantages of mobile app security analyzers are explored, alongside a comparative analysis of various tools and methodologies, including static, dynamic, behavioral, and hybrid analysis techniques. Case studies illustrate real-world applications of security assessments, highlighting the importance of compliance with regulations such as GDPR and the need for continuous monitoring. The project culminates in a demonstration of selected tools, showcasing their capabilities and effectiveness in identifying security flaws. Ultimately, the Mobile App Security Analyzer serves as a vital resource for developers and organizations seeking to protect sensitive user information and maintain consumer trust in an increasingly digital landscape.

Keywords: Mobile App Security Analyzer, Vulnerabilities, Cyber threats, Security measures, Compliance.

Table of Contents

Table of Figures

List of Tables

CHAPTER 1
INTRODUCTION

Introduction

Project Overview

The Mobile App Security Analyzer is a tool for assessing iOS and Android app security, identifying vulnerabilities, and providing recommendations to enhance app protection in today's data-sensitive digital environment.

Significance of the Project

With mobile apps vital to daily life and thus prime cyber targets, the Mobile App Security Analyzer addresses critical data privacy and security concerns. Providing a comprehensive assessment, the project aims to strengthen mobile app security, protect user data, and maintain consumer trust, as robust security measures are essential in this vulnerable landscape.[1].

Importance of Mobile App Security

Mobile app security is crucial for several reasons:

Protection of Sensitive Data

Securing sensitive user data, like personal, financial, and health information, is crucial for mobile apps to prevent breaches and unauthorized access (Miller, 2017)[2].

Regulatory Compliance

Organizations must comply with data protection regulations like GDPR and HIPAA to avoid significant legal and financial penalties. (Stallings & Brown, 2019)[3].

[1] Kumar, R., & Singh, A. (2018). Mobile Application Security: A Comprehensive Guide to Mobile Security. Springer.
[2] Miller, M. (2017). Mobile App Security: A Comprehensive Guide to Protecting Your Mobile Applications. O'Reilly Media.
[3] Stallings, W., & Brown, L. (2019). Computer Security: Principles and Practice (4th ed.). Pearson.

Maintaining User Trust

Secure mobile apps build user trust, enhancing business reputation and fostering customer loyalty (Gollmann, 2011)[4].

Role of Security Analyzers

Security analyzers play a critical role in enhancing mobile app security by:

Identifying Vulnerabilities

They detect security weaknesses within mobile applications, allowing developers to address issues before they can be exploited (Sullivan, 2019)[5].

Conducting Penetration Testing

Security analyzers simulate real-world attacks to uncover exploitable vulnerabilities, providing organizations with insights into potential security gaps (Anderson, 2020)[6].

Generating Automated Reports

These tools produce detailed reports on identified vulnerabilities, their severity, and recommended remediation strategies, facilitating informed decision-making for developers and security professionals (Raghavan, 2018)[7].

Continuous Monitoring

Security analyzers support ongoing monitoring of mobile applications, alerting organizations to newly discovered vulnerabilities and emerging threats. This enables proactive security measures and timely updates to maintain a strong security posture.

[4] Gollmann, D. (2011). Computer Security (2nd ed.). Wiley.
[5] Sullivan, B. (2019). The Mobile Application Hacker's Handbook: A Guide to Mobile Application Security. Wiley.
[6] Anderson, R. (2020). Security Engineering: A Guide to Building Dependable Distributed Systems (3rd ed.). Wiley.
[7] Raghavan, S. (2018). Mobile Security: A Comprehensive Guide to Protecting Your Mobile Devices. Apress.

CHAPTER 2
LITERATURE REVIEW

Literature Review

Advantages of Mobile App Security Analyzers

Mobile app security analyzers offer several benefits that are critical for organizations looking to enhance their security posture:

Improved Security

Security analyzers identify vulnerabilities in mobile apps, strengthening security and reducing cyber risks in today's threat landscape (Kumar & Singh, 2018)[8].

Regulatory Compliance

Security analyzers automate compliance checks, ensuring adherence to regulations (e.g., GDPR, HIPAA), reducing legal risks, and enhancing user trust (Stallings & Brown, 2019)[9].

Risk Management

Security analyzers help organizations manage risks by detecting threats early, enabling timely remediation and reducing security incidents (Miller, 2017)[10].

Cost-Effectiveness

Automating security assessments saves time and reduces costs compared to manual testing, improving efficiency (Anderson, 2020)[11].

Enhanced User Trust

Security analyzers and a commitment to security boost user trust, increasing engagement and retention in mobile apps (Gollmann, 2011)[12].

[8] Kumar, R., & Singh, A. (2018). Mobile Application Security: A Comprehensive Guide to Mobile Security. Springer.

[9] Stallings, W., & Brown, L. (2019). Computer Security: Principles and Practice (4th ed.). Pearson.

[10] Miller, M. (2017). Mobile App Security: A Comprehensive Guide to Protecting Your Mobile Applications. O'Reilly Media.

[11] Anderson, R. (2020). Security Engineering: A Guide to Building Dependable Distributed Systems (3rd ed.). Wiley.

[12] Gollmann, D. (2011). Computer Security (2nd ed.). Wiley.

Disadvantages of Mobile App Security Analyzers

While mobile app security analyzers provide numerous benefits, there are also potential drawbacks that organizations should consider:

Cost

Security analyzer implementation and maintenance can be costly, straining budgets, especially for smaller businesses, requiring careful financial planning (Raghavan, 2018)[13].

Complexity

The integration of security analyzers into existing development workflows can introduce complexity, requiring additional training for developers and security teams to effectively utilize the tools (Sullivan, 2019)[14].

False Positives

False positives may identify non-existent or non-exploitable vulnerabilities, leading to unnecessary remediation efforts and resource allocation (Kumar & Singh, 2018)[15].

Limited Scope

Coverage limitations exist in some security analyzers, potentially leaving gaps in assessments. Organizations may need multiple tools for comprehensive protection (Miller, 2017)[16].

[13] Raghavan, S. (2018). Mobile Security: A Comprehensive Guide to Protecting Your Mobile Devices. Apress.

[14] Sullivan, B. (2019). The Mobile Application Hacker's Handbook: A Guide to Mobile Application Security. Wiley.

[15] Kumar, R., & Singh, A. (2018). Mobile Application Security: A Comprehensive Guide to Mobile Security. Springer.

[16] Miller, M. (2017). Mobile App Security: A Comprehensive Guide to Protecting Your Mobile Applications. O'Reilly Media.

Dependence on Tools

Automation alone is insufficient; human expertise and manual testing are crucial complements for robust security(Gollmann, 2011)[17].

Current Trends and Threats in Mobile App Security

Mobile applications are essential to digital ecosystems but introduce security risks, including data breaches, unauthorized access, and financial fraud.

Data Leakage and Privacy Risks

Data leakage, due to excessive and often non-consensual user data collection and sharing, is a major mobile app security concern.

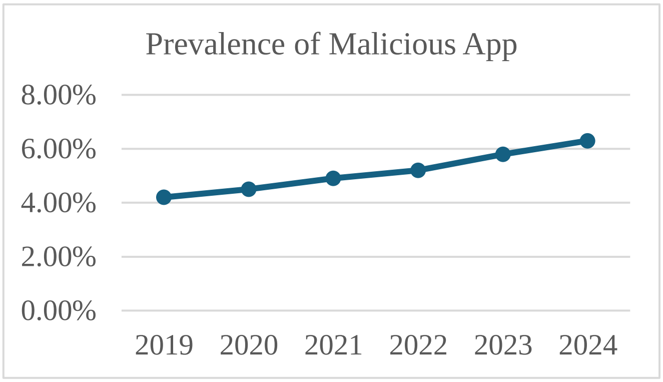


Figure 1Prevalence of Malicious App from 2019 to 2024

Prevalence of Malicious Apps

In 2024, 6.3% of smartphones had at least one malicious app, exposing users to spyware, credential theft, and unauthorized transactions (New York Post, 2024)[18].

Fitness Apps' Data Sharing

80% of fitness apps share user data, including health, location, and contacts, with some collecting 21+ data types, raising major privacy concerns. (News.com.au, 2024)[19].

[17] Gollmann, D. (2011). Computer Security (2nd ed.). Wiley.
[18] New York Post. (2024, November 27). Smartphone users warned to delete 15 dangerous predatory apps. https://nypost.com/2024/11/27/tech/smartphone-users-warned-to-delete-15-dangerous-predatory-apps/
[19] News.com.au. (2024, November 27). Fitness apps sucking up 21 different types of user data, study finds. https://www.news.com.au/technology/fitness-apps-sucking-up-21-different-types-of-user-data-study-finds/news-story/aaed2d7eee252536c90369aa917fd156

Travel Apps and Excessive Permissions

Travel apps request an average of 23 permissions, with 28% being unnecessary for their functionality, creating potential privacy risks and data overexposure (Daily Telegraph, 2024)[20].

Social Media Data Tracking

Popular social media apps collect up to 32 different types of user data, often tracking location, interactions, and personal preferences for targeted advertising (Lifewire, 2024)[21].

These findings indicate that many mobile applications prioritize data collection over user privacy, raising concerns about regulatory compliance and consumer trust.

Insecure Communication and Authentication Failures

Weak encryption protocols, poor SSL/TLS configurations, and inadequate authentication mechanisms leave mobile apps vulnerable to cyberattacks such as man-in-the-middle (MITM) attacks and unauthorized access.

SSL Misconfigurations

Between 16% to 30% of mobile apps have improper SSL configurations, making them vulnerable to interception and MITM attacks (Digital.ai, 2024)[22].

[20] Daily Telegraph. (2024, November 27). Travel apps are among the most privacy-unfriendly apps. https://www.dailytelegraph.com.au/lifestyle/travel-apps-are-among-the-most-privacy-unfriendly-apps/news-story/84e58a902c5a937b1b4038731d53b8bb

[21] Lifewire. (2024, November 27). Most invasive apps: The apps that collect the most user data. https://www.lifewire.com/most-invasive-apps-8716661

[22] Digital.ai. (2024). 2024 application security threat report. https://digital.ai/resource-center/ebooks/2024-application-security-threat-report/

Unauthorized Access via Stolen Devices

Cybercriminals have exploited weak authentication mechanisms to gain access to banking and financial apps after stealing devices, leading to fraudulent transactions and financial losses (Financial Times, 2024)[23].

These issues highlight the importance of enforcing strong authentication, such as multi-factor authentication (MFA) and biometric verification, to mitigate unauthorized access risks.

Third-Party SDK and Malware Risks

Many mobile applications rely on third-party software development kits (SDKs), which can introduce security vulnerabilities if not properly vetted.

Third-Party SDK Vulnerabilities

85% of mobile apps contain security vulnerabilities due to third-party SDKs, making them susceptible to supply chain attacks (Digital.ai, 2024)[24].

These findings emphasize the importance of thorough security vetting for third-party SDKs and the need for strong app review mechanisms to detect and remove malware-infected apps.

Reverse Engineering and App Vulnerabilities

Reverse engineering allows attackers to decompile mobile apps, analyze their code, and discover security flaws that can be exploited.

[23] Financial Times. (2024, November 27). Thieves access banking apps and conduct unauthorized transactions after stealing phones. https://www.ft.com/content/682171de-510b-47b8-9619-177fa4d7de6a

[24] Digital.ai. (2024). 2024 application security threat report. https://digital.ai/resource-center/ebooks/2024-application-security-threat-report/

Prevalence of App Vulnerabilities

75% of mobile applications contain at least one security flaw, making them highly susceptible to reverse engineering, tampering, and code injection attacks (Digital.ai, 2024)[25].

Developers must obfuscate and encrypt sensitive code to protect applications from reverse engineering and tampering.

Compliance and Consumer Expectations

With growing awareness of privacy risks, users are demanding stronger security measures in mobile applications.

Consumer Demand for Protection

A 2024 global survey found that 99.5% of consumers expect mobile apps to provide total security, covering data encryption, fraud prevention, and compliance with privacy laws (Appdome, 2024)[26].

Failure to meet these expectations can lead to a loss of consumer trust, reduced app adoption, and regulatory penalties.

Comparative Analysis of Tools

Methodology Comparison

Static, Dynamic, Behavioral, and Hybrid Analysis

[25] Digital.ai. (2024). 2024 application security threat report. https://digital.ai/resource-center/ebooks/2024-application-security-threat-report/

[26] Appdome. (2024). Global consumer security survey reveals highest demand for mobile app security in 4 years. https://www.appdome.com/press-release/global-consumer-security-survey-reveals-highest-demand-for-mobile-app-security-in-4-years/

Different analysis methodologies have unique use cases, benefits, and limitations. Some are ideal for early-stage development, while others excel with deployed applications. The table below details these distinctions.

Methodology	How It Works	Pros	Cons
Static Analysis	*Analyzes source code, bytecode, or binary files to detect vulnerabilities prior to execution.*	*Facilitates early security flaw detection, automated scanning, and quick results, useful for compliance auditing.*	*May generate false positives, does not identify runtime-specific vulnerabilities like session management issues.*
Dynamic Analysis	*Monitors application behavior in real-time to detect security flaws.*	*Effectively finds authentication flaws, injection vulnerabilities, and insecure network communications.*	*Requires execution, can be time-consuming, and may miss some execution paths.*
Behavioral Analysis	*Utilizes sandboxing to observe app behavior, including network activity, permissions usage, and system interactions.*	*Detects malicious activities, such as spyware, unauthorized data transmissions, and hidden backdoors.*	*Limited to detecting observable behaviors and may require extensive configuration for thorough results.*
Hybrid Analysis	*Combines static and dynamic techniques for comprehensive security testing.*	*More accurate and detailed detection covers both code-level and runtime security risks.*	*Higher resource consumption requires specialized tools and may be complex to implement effectively.*

Table 1 Static, Dynamic, Behavioral, and Hybrid Analysis

Proprietary vs. Open-Source Solutions

Security testing tools are either proprietary, offering support and advanced features, or open-source, providing flexibility and cost-effectiveness for small teams and researchers.

Aspect	Proprietary Tools	Open-Source Tools
Cost	*Often expensive, requiring licensing or subscription fees.*	*Free to use or minimal costs.*
Support	*It comes with professional support, regular updates, and dedicated customer service.*	*Community-driven support may have slower updates and issue resolutions.*
Customization	*Limited to the features provided by the vendor.*	*Highly customizable, allowing users to modify and extend capabilities as needed.*
Compliance	*Includes built-in compliance checks for GDPR, PCI-DSS, and other regulatory requirements.*	*Requires manual validation and compliance reporting.*
Scalability	*Well-suited for enterprises with large security teams and extensive testing needs.*	*More appropriate for small businesses, research labs, and security hobbyists.*
Ease of Use	*User-friendly interfaces with guided workflows and dashboards.*	*Often requires technical expertise to configure and use effectively.*

Table 2 Proprietary vs. Open-Source Solutions

Mobile App Security Analyzers

Tools & Unique Features

Mobile app security analyzers are categorized by their vulnerability detection methods, each with unique advantages and limitations. Choosing the right tool depends on specific project needs. Below is a detailed breakdown of these categories.

Tool	Category	Features	Cost
MobSF	*Static & Dynamic*	*Open-source, static & dynamic analysis, malware analysis, penetration testing, API testing.*	*Free*
SonarQube	*Static*	*Continuous code quality & security inspection, multi-language support.*	*Free (Community Edition); paid plans available*
Snyk	*Static*	*Identifies open-source library vulnerabilities, CI/CD integration.*	*Free tier; paid for advanced features*

CodeQL	*Static*	*Code analysis with query capabilities for vulnerability detection, multi-language support.*	*Free (open-source)*
Burp Suite	*Dynamic*	*Web application security testing, including dynamic analysis for mobile apps.*	*Free Community Edition; paid plans available*
OWASP ZAP	*Dynamic*	*Open-source web application security scanner for dynamic analysis.*	*Free (open-source)*
Drozer	*Dynamic*	*Android security testing framework, primarily for dynamic analysis.*	*Free (open-source)*

Table 3 Tools & Unique Features

This table lists mobile app security testing tools, their categories, features, and costs. Contact vendors for precise pricing, which varies by needs and size.

Diagram & Architectural Details

This diagram details a comprehensive APK/IPA mobile app analysis process, combining static and dynamic vulnerability assessments.

Submission and Validation

Analysis starts with file submission, then validation for security compliance.

Static Analysis

This phase involves decompiling the APK or extracting the IPA to perform various checks, including:

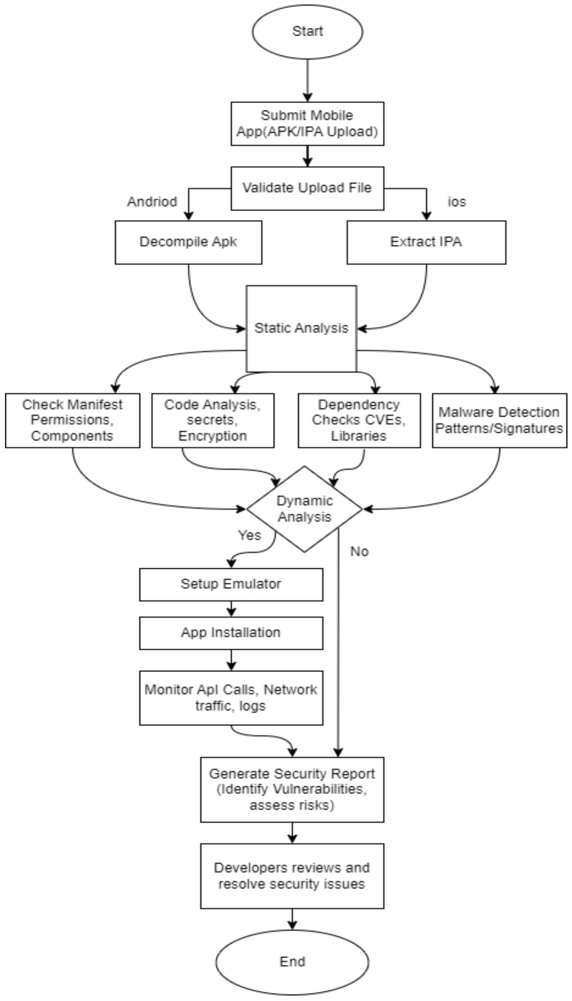

Start
Submit Mobile App(APK/IPA Upload)
Validate Upload File
Andriod
ios
Decompile Apk
Extract IPA
Static Analysis
Check Manifest Permissions, Components
Code Analysis, secrets, Encryption
Dependency Checks CVEs, Libraries
Malware Detection Patterns/Signatures
Dynamic Analysis
Yes
No
Setup Emulator
App Installation
Monitor Api Calls, Network traffic, logs
Generate Security Report (Identify Vulnerabilities, assess risks)
Developers reviews and resolve security issues
End

Figure 2 Mobile Security Analyzer Workflow

Manifest Permissions and Components

Reviewing application permissions to identify potential security risks.

Code Analysis

Assessing the code for secrets, encryption vulnerabilities, and other security flaws.

Dependency Checks

Verifying libraries and components against known vulnerabilities (CVE).

Malware Detection

Identifying any patterns or signatures indicative of malicious software.

Dynamic Analysis

After static analysis, the application undergoes dynamic testing, which includes:

Emulator Setup

Configuring an environment to run the application safely.

App Installation

Installing the app within the emulator for testing.

Monitoring

Observing API calls, network traffic, and logs to detect any unusual behaviors.

Security Reporting and Remediation

A comprehensive security report is generated, identifying vulnerabilities and assessing risks. Developers review the report to address security issues, strengthening the application against potential threats.

CHAPTER 3
CASE STUDIES

Case Studies

Case Study 1: Privacy and Security Analysis of Early-Deployed COVID-19 Contact Tracing Android Apps[27]

Context

During the COVID-19 pandemic, contact tracing apps were quickly deployed, raising security concerns like inconsistent privacy policies, excessive permissions, and GDPR compliance issues, with some apps collecting more data than disclosed.

Challenges

Excessive Data Collection

Many apps requested continuous location access and unnecessary permissions.

Weak Privacy Policies

Privacy disclosures often failed to accurately reflect actual data collection practices.

Security Vulnerabilities

Hardcoded credentials and improper encryption left data at risk.

GDPR Non-Compliance

Many apps failed to meet requirements for user consent, data minimization, and purpose limitation.

Impact

Erosion of Public Trust

Privacy concerns hindered user adoption, limiting contact tracing effectiveness.

[27] Khan, M. A., & Alzahrani, A. I. (2020). A privacy and security analysis of early-deployed COVID-19 contact tracing Android apps. *Empirical Software Engineering*, 25(6), 1-30. https://doi.org/10.1007/s10664-020-09934-4

Legal Risks

GDPR violations exposed governments to potential legal actions.

Data Breach Vulnerabilities

Poor security measures increased exposure to unauthorized access and exploitation.

Solutions

Implement Privacy-by-Design

Ensure security and privacy are built into app development from the start.

Stronger Regulatory Compliance

Align with GDPR and data protection laws through better consent mechanisms.

Enhanced Security Audits

Conduct pre-deployment vulnerability testing to fix security flaws.

Clearer Privacy Policies

Improve transparency by aligning privacy policies with actual data practices.

Methodology

Static Analysis (MobSF)

Scanned APK files for vulnerabilities, including insecure permissions and hardcoded secrets.

Dynamic Analysis

Monitored runtime behavior for unauthorized data transmissions and privacy violations.

Privacy Policy Review.

Compared stated privacy policies with actual data collection practices.

Tools Used

Mobile Security Framework (MobSF)

Static analysis for security flaws.

Manual Privacy Policy Review.

Evaluated legal and technical compliance.

Network Traffic Monitoring

Tracked unauthorized data transmissions.

Case Study 2: Security Assessment of Malaysian Mobile Applications[28]

Context

Malaysia's digital transformation has increased reliance on mobile apps for banking, e-commerce, and government services. With over 7% smartphone penetration, these apps handle sensitive data, but rapid development and legacy system integration have led to security vulnerabilities, highlighted by a 2022 data leak affecting 22.5 million Malaysians.

Challenges

Outdated Security Measures

30% of tested apps used deprecated cryptographic algorithms.

Insecure Data Storage

40% of apps stored user data in plaintext, making it accessible to attackers.

Unencrypted Network Transmissions

25% of apps transmitted sensitive data over HTTP instead of HTTPS.

[28] Zainuddin, N., & Rahman, A. (2023). Security assessment of Malaysian mobile applications: A detailed analysis. *Security and Privacy*, 6(1), e200. https://doi.org/10.1002/sec.200

Compliance Gaps

Many apps failed to meet Malaysia's Personal Data Protection Act (PDPA) and Bank Negara Malaysia (BNM) security guidelines.

Impact

Financial Fraud Risks

Weak security in banking apps increased exposure to unauthorized transactions and fraud.

Reputation Damage

Data leaks led to user distrust and reduced app adoption.

Legal Consequences

PDPA violations could result in fines up to RM500,000.

Solutions

Enforce Strong Encryption (AES-256, TLS 1.2+)

Strengthening security for both stored and transmitted data.

Mandate HTTPS with Certificate Pinning

Protect sensitive data from Man-in-the-Middle (MITM) attacks.

Introduce Security Certification

Require government and financial apps to pass a Security by Design certification.

Developer Training Programs

Conduct MDEC-sponsored workshops to improve cybersecurity awareness.

Methodology

A **three-phase** security assessment was conducted:

1. **Pre-runtime Phase (MobSF)**

   Static analysis for security flaws and regulatory compliance gaps.

2. **Runtime Phase (Burp Suite, Frida)**

   Dynamic testing to analyze network traffic and detect runtime attacks.

3. **Post-runtime Phase (SQLite Browser)**

   Forensic analysis of stored user data to detect security risks.

Tools Used

Mobile Security Framework (MobSF)

The tool identifies hardcoded AWS keys in fintech apps.

Burp Suite

Captured unencrypted network traffic from travel apps.

Frida

Bypassed OTP security checks in financial apps.

Drozer

Exposed vulnerabilities in Android Content Providers.

SQLite Browser

Recovered plaintext Personally Identifiable Information (PII) from shopping apps.

Case Study 3: Android Apps Vulnerability Detection[29]

Context

The Mobile JKN app, part of Indonesia's BPJS Kesehatan, provides access to healthcare services and personal medical data. However, critical flaws in cryptography, authentication, and runtime security have been identified, making user data vulnerable to unauthorized access.

Challenges

Weak Cryptography

The app used inadequate encryption methods, leaving patient data exposed.

Root and Debugger Bypass Risks

The app lacked root detection, allowing attackers to run it on modified devices.

Runtime Exploitation

Debugger tools could extract sensitive data and manipulate app behavior.

Impact

Patient Data Exposure

Personal health records were at risk of unauthorized access.

Compliance Violations

Security flaws could result in violations of Indonesia's data protection laws.

Loss of Public Trust

Weak security could undermine confidence in Indonesia's healthcare system.

[29] Sari, D. P., & Prabowo, H. (2022). Android apps vulnerability detection: A case study of the Mobile JKN application. *Journal of Information Security and Applications*, 68, 103-115. https://doi.org/10.1145/3556974

Solutions

Strengthening Cryptographic Methods

Implement AES-256 encryption for stored and transmitted data.

Enhance Root Detection

Prevent the app from running on rooted or jailbroken devices.

Implement Debugger Detection

Block unauthorized runtime analysis tools.

Methodology

Static Analysis (MobSF)

Examined APK files for vulnerabilities like hardcoded secrets, weak encryption, and SQL injection risks.

Dynamic Analysis (Frida)

Monitored the app’s runtime behavior to detect security bypass risks and debugger exploits.

Tools Used

Mobile Security Framework (MobSF)

Identified insecure storage and poor cryptographic practices.

Frida

Detected runtime security flaws and potential data extraction vulnerabilities.

CHAPTER 4
USER REQUIREMENTS, OUR PRODUCT, BUSINESS IMPACT

User Requirements

Secure APK Analysis via Web: Users need an accessible platform to analyze Android APKs without setting up or installing security tools locally.

Multi-Engine Comparison: Developers and security teams want to view and compare security insights from MetaDefender and VirusTotal in one place.

No-Code, Intuitive Interface: There's demand for a clean, easy-to-navigate UI that simplifies complex security reports and indicators.

Data Confidentiality: Users require confidence that their APK files are securely handled, with minimal exposure during scanning and reporting.

Actionable, Visual Reports: Stakeholders expect well-structured summaries that clearly highlight threats, permissions, and indicators for quick decision-making.

Our Solution

Web-Based APK Security Analyzer: A full-stack web application that allows users to upload APKs and instantly fetch threat intelligence from both VirusTotal and MetaDefender APIs.

Dynamic Comparative Dashboard: Presents side-by-side visualizations of scan results, including detected malware, flagged behaviors, and risk scores.

Automated Parsing & Categorization: Extracts and organizes API responses into clear categories — like file reputation, antivirus detections, and manifest metadata.

Minimal Manual Input: Users only upload their file; the system handles API calls, response formatting, and comparison logic autonomously.

Security-Centric Design: Prioritizes safe handling of files and secure API integration while remaining scalable for future feature expansion (e.g., CI/CD integration, additional tool support).

Business Impact

Enhanced Mobile App Security: Helps organizations identify threats in APKs before release, reducing the risk of malware distribution and reputational damage.

Time-Efficient Analysis: By aggregating multiple analysis engines into a single platform, it significantly reduces manual overhead and speeds up threat assessment.

Improved Decision-Making: Provides clear, comparative insights that enable faster and more informed decisions regarding app deployment or remediation.

Operational Transparency: Converts complex scan data into understandable summaries, improving communication between technical and non-technical stakeholders.

Foundation for Scalable Security Workflows: Offers a modular and extensible design that could be integrated into enterprise-level mobile security review pipelines in the future.

CHAPTER 5

TECHNICAL DETAILS & NON-TECHNICAL

Technical Details

Overview of Selected Tools

To comprehensively analyze mobile application security, Mobile Security Framework (MobSF), VirusTotal, and MetaDefender were used. These tools provide static analysis and malware/vulnerability scanning, offering a layered approach to identifying potential security issues.

Mobile Security Framework (MobSF)

MobSF, an open-source framework, automates security testing for Android and iOS, using static, dynamic, and malware analysis, with stronger dynamic analysis for Android, and offers user-friendly reporting.

VirusTotal

VirusTotal, a free online service, aggregates malware analysis from numerous antivirus engines and URL scanners, providing comprehensive threat detection and community-driven threat intelligence.

MetaDefender

MetaDefender, a security platform, employs multi-scanning with multiple antivirus engines and Deep CDR to neutralize threats, focusing on organizational security through vulnerability assessment and proactive threat prevention.

Features and Capabilities

Tool	Features
Mobile Security Framework (MobSF)	*Static Analysis: Scans source code, binaries (APK/IPA), and manifest files for issues like hardcoded secrets, insecure permissions, deprecated APIs, and OWASP Mobile Top 10 vulnerabilities (e.g., improper platform usage).*

	Dynamic Analysis (Android-focused): Executes apps in an emulator to monitor runtime behaviors, including file system interactions, network calls, and insecure data storage. Limited iOS dynamic support.
	Malware Analysis: Detects known malware signatures, suspicious activities, and risky permissions.
	API Security: Flags insecure API usage (e.g., unprotected intents in Android) and checks for unencrypted (HTTP) communication.
	Reporting: Generates consolidated reports (PDF, HTML) with risk ratings, code snippets, and remediation guidance.
VirusTotal	*Multiscanning: Aggregates results from numerous antivirus engines and website scanners to detect a wide range of malware threats.*
	URL Analysis: Scans URLs for malicious content, including phishing and malware distribution.
	File Analysis: Analyzes files of various types for malware signatures and suspicious behavior.
	Community Intelligence: Shares threat data with the security community, building a comprehensive threat database.
	API Access: Provides an API for integrating VirusTotal's scanning capabilities into other tools and workflows.
MetaDefender	*Multiscanning: Utilizes multiple antivirus engines for enhanced threat detection and prevention.*
	Deep CDR (Content Disarm and Reconstruction): Neutralizes potential threats by removing malicious elements from files, ensuring safe file usability.
	Vulnerability Assessment: Identifies and reports software vulnerabilities within files and systems.
	Proactive Threat Prevention: Focuses on preventing threats from entering organizational networks.
	File Sanitization: Cleans files of malware and other threats, allowing for safe sharing and use.

Table 4 Selected tools features and capabilities.

Tool	**Insights**
Mobile Security Framework (MobSF)	*Integrate with CI/CD pipelines (e.g., Jenkins) for automated scans.*
	Combine with Objection for iOS dynamic testing.
	Use VirusTotal API to enhance malware detection.
	Limited against obfuscated code (e.g., ProGuard)
VirusTotal	*Useful for quick, initial malware assessments of files and URLs.*
	API integration enhances automated threat intelligence workflows.

	Effectiveness depends on the quality and number of integrated antivirus engines.
	Provides limited context on the specific nature of detected threats beyond engine flags.
MetaDefender	*Deep CDR is effective for sanitizing files before sharing in secure environments.*
	Multiscanning offers a higher detection rate than single antivirus solutions.
	Valuable for organizations focused on preventing malware entry at network perimeters.
	Can add significant processing overhead due to thorough file analysis.

Table 5 Tools insights.

Infrastructure Diagram

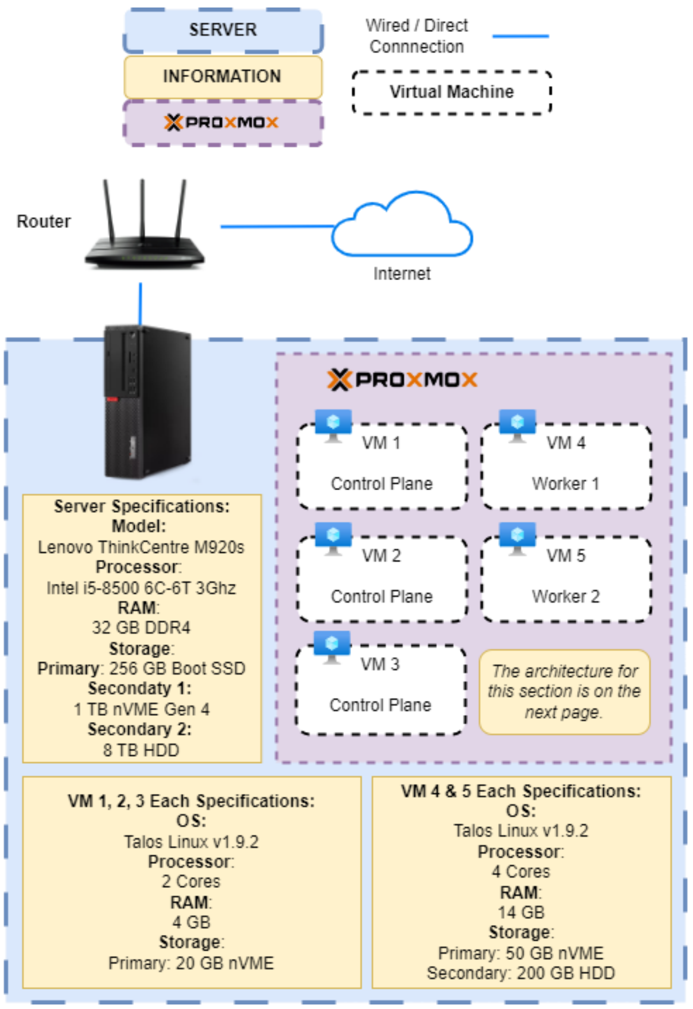


Figure 3 Infrastructure Diagram illustrates the system architecture.

Infrastructure Overview

A virtualized infrastructure using Proxmox VE simulates a scalable cluster environment for testing and deploying distributed systems, optimizing resource use and connectivity across control and worker nodes.

Wired Connectivity

Ensures stable, high-speed communication between the server and the home router, which acts as the primary gateway to external networks.

Host Server

Proxmox VE, an open-source platform, manages the entire virtualized environment on a single physical server, enabling efficient VM creation, management, and scaling.

Virtual Machine Configuration

The virtual machines (VMs) are categorized into two functional roles to simulate a cluster environment:

Control Plane Nodes

Three VMs manage the cluster's operations, including resource scheduling and orchestration.

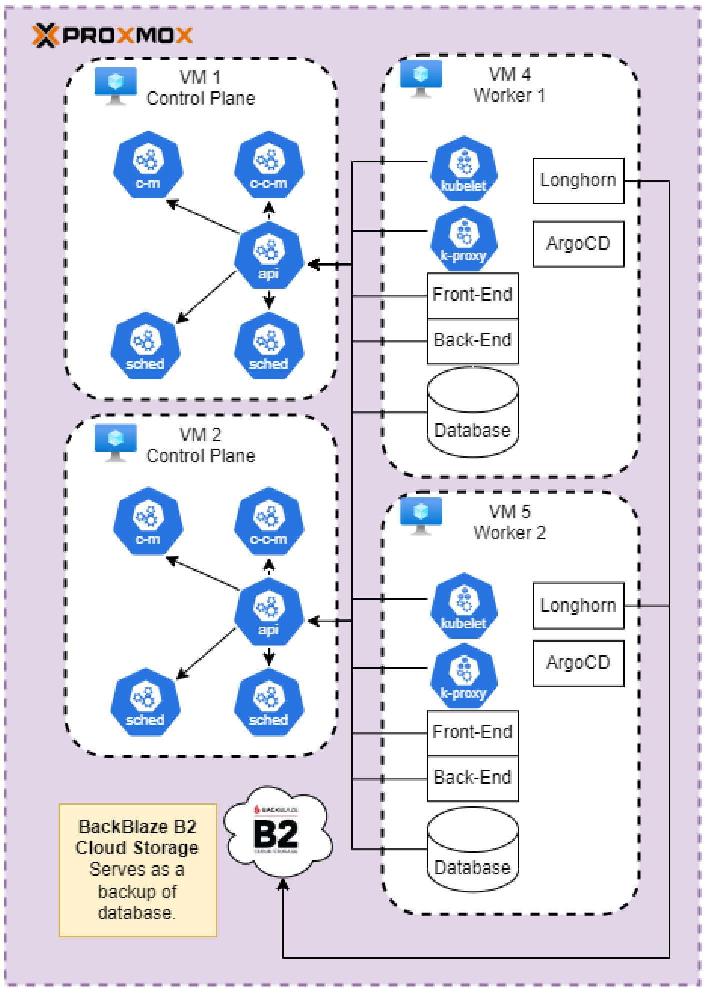


Figure 4 Kubernetes Architecture

Worker Nodes

Two VMs handle workload execution and perform operational tasks within the cluster.

All virtual machines operate on Talos Linux v1.9.2, a lightweight and secure operating system tailored for cluster-based deployments.

Control Plane Components: VM 1 & VM 2: Control Plane Nodes

c-m (Controller Manager)

Manages the various controllers for regulating the states of the different objects in the cluster.

api (API Server)

Serves as the front-end for the Kubernetes control plane. It handles requests from clients and communicates with other components.

sched (Scheduler)

Responsible for selecting which node an unscheduled pod will run on based on resource availability and other criteria.

Worker Nodes: VM 4 & VM 5: Worker Nodes

Kubelet

An agent that runs on each worker node, ensuring that containers are running in a pod.

k-proxy

Manages network traffic to the pods. Acts as a reverse proxy and load balancer, facilitating communication between clients and the services running in the pods.

Deployments and Services

Longhorn

A distributed block storage project for Kubernetes that provides high-availability storage.

ArgoCD

A declarative, GitOps continuous delivery tool for Kubernetes, automating the deployment process from Git repositories.

Application Configuration

Front-End and Back-End Services

Each worker node hosts applications divided into front-end and back-end components, connecting to a Database for persistent storage.

Backup Solution

BackBlaze B2 Cloud Storage

Identified in the diagram as the backup storage solution for the database, ensuring data durability and accessibility across different environments.

General Workflow

Control Plane Operations

The control plane manages the desired state of the cluster. It dynamically schedules pods onto worker nodes based on resource availability.

Worker Nodes Operations

Worker nodes execute the applications and services. They interact with the control plane through the kubelet and handle incoming HTTP requests via k-proxy.

Storage Management

Longhorn provides persistent storage for applications running on the worker nodes. BackBlaze B2 maintains regular backups of the database, enhancing data security.

Cutting-edge virtualization technologies establish a robust, scalable, and versatile cluster environment, optimizing resource allocation, networking, and security for distributed system exploration and application.

High-Level Architecture

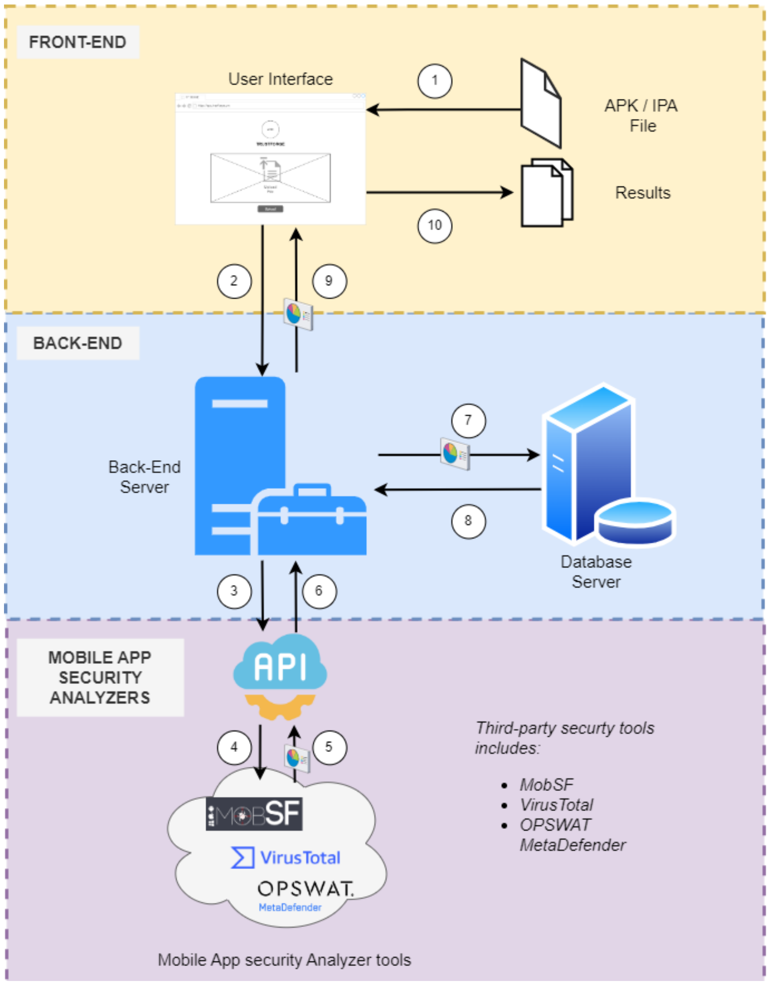


Figure 5 High-Level Architecture Diagram.

The architecture employs three layers—Front-End, Back-End, and Mobile App Security Analyzers—to provide a streamlined mobile app security analysis process.

Front-End Layer

The Front-End layer serves as the primary interface for users, enabling interaction with the system. Key components include:

User Interface (UI)

This interface allows users to upload mobile application files such as APK (Android) or IPA (iOS) files for security analysis (Step 1). Once the analysis is complete, the results, which include detailed reports on vulnerabilities and potential risks, are displayed back to the user (Step 10).

Back-End Layer

The Back-End layer acts as the system's core processing unit, managing communication between the user interface, analyzers, and database. Key processes include:

The process begins with the Back-End server receiving uploaded files from the Front-End (Step 2). Subsequently, it utilizes a robust API layer to delegate analysis tasks to the Mobile App Security Analyzers (Step 3).

Once the analyzers complete their tasks, the Back-End gathers and structures the generated results (Step 6). It then interacts with the Database Server to store these analysis results and retrieve historical data, when necessary (Step 7), which can also be used for comparison or further processing (Step 8).

Finally, the processed results are sent back to the Front-End for user visualization (Step 9). The Back-End ensures scalability, secure communication, and efficient processing to handle multiple analysis requests simultaneously.

Mobile App Security Analyzers

This layer comprises third-party security tools and frameworks specifically designed for in-depth mobile app security analysis. Key functionalities include:

Tools and Frameworks

Includes MobSF, VirusTotal, and MetaDefender. These tools collectively analyze the uploaded files for vulnerabilities such as insecure code practices, outdated libraries, or potential malware.

The API layer initiates and manages the security analysis with the tools (Step 4), which then generate and forward the results to the Back-End server (Step 5). This layer is highly modular, enabling the integration of additional tools as required to keep up with evolving security challenges.

Core Function Flow Chart

Core Function Flow

The primary function of the system is to analyze APK/IPA files for security and functionality purposes using a structured workflow. The following describes the process in detail:

The process initiates with a user upload of an APK or IPA file via the front-end, serving as analysis input. The back-end then validates the file and checks the database for prior analysis, avoiding duplicates. If a past record exists, the stored results are displayed, streamlining the workflow. If not, the system conducts a comprehensive analysis using integrated tools, including security scans, static/dynamic analysis, and functionality checks.

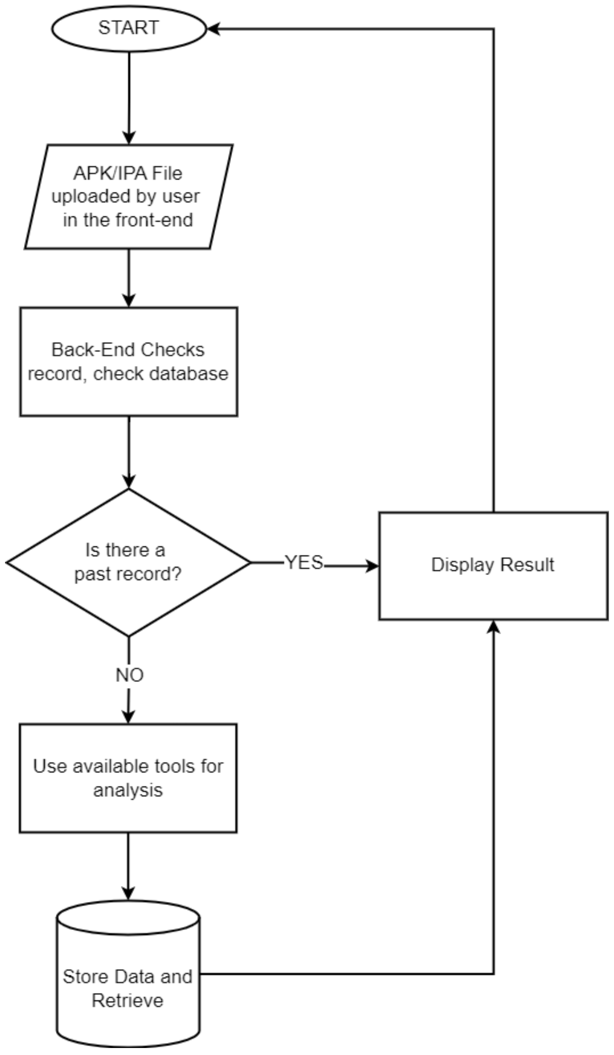

START
APK/IPA File uploaded by user in the front-end
Back-End Checks record, check database
Is there a past record?
YES
Display Result
NO
Use available tools for analysis
Store Data and Retrieve

Figure 6 Project Flow Chart depicting the process workflow.

The resulting data is stored for future reference, creating a file repository. Finally, the analysis results, either retrieved or newly generated, are displayed to the user through the front-end.

Benefits of the Workflow

The system is designed for efficiency, avoiding redundant analysis by checking for existing records, thus saving computational resources. Its modular design allows for scalability, enabling the integration of additional tools and functionalities. Furthermore, user convenience is prioritized by providing results in real-time, ensuring a seamless and efficient experience.

User Interface Wireframe

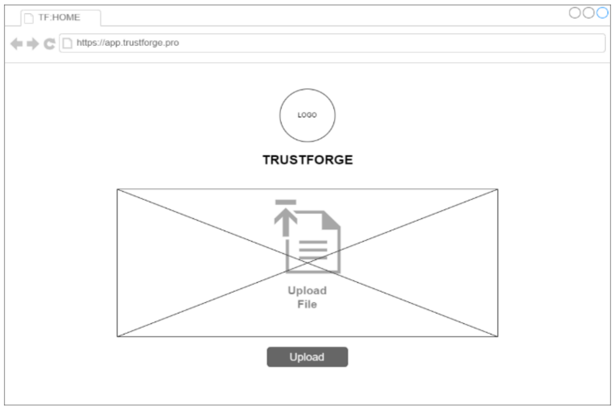


Figure 8 Wireframe Uploading APK / IPA file for

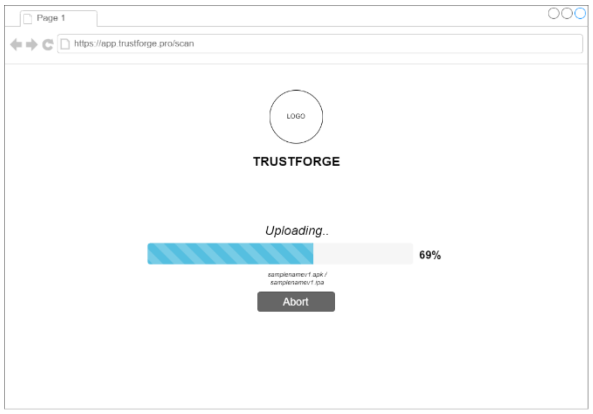


Figure 7 Wireframe Upload Page

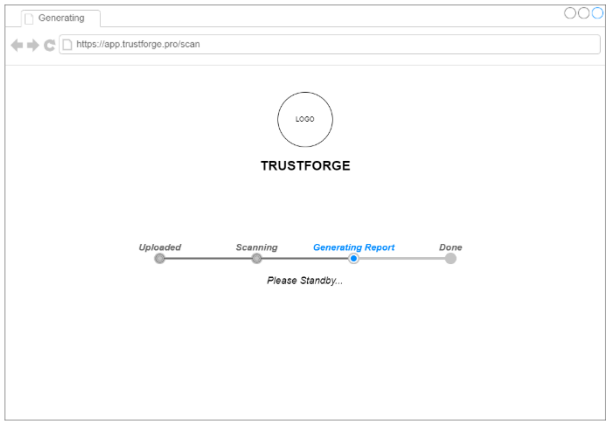


Figure 10 Wireframe The

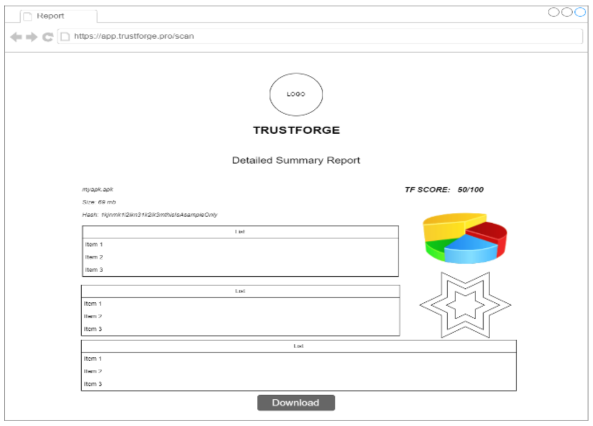


Figure 9 Wireframe Detailed Summary Report

Non-technical diagram

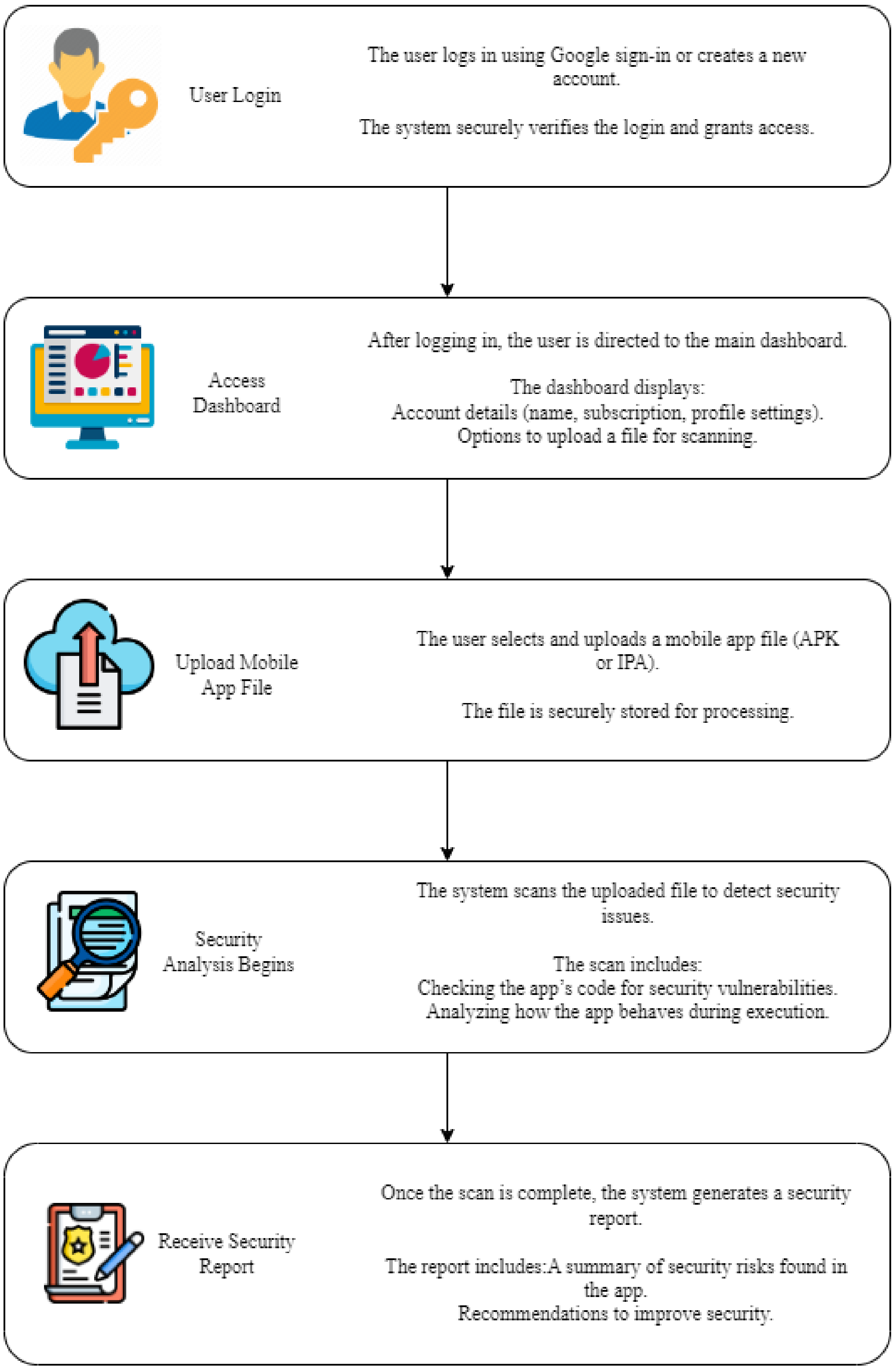

User Login
The user logs in using Google sign-in or creates a new account.
The system securely verifies the login and grants access.
Access Dashboard
After logging in, the user is directed to the main dashboard.
The dashboard displays:
Account details (name, subscription, profile settings).
Options to upload a file for scanning.
Upload Mobile App File
The user selects and uploads a mobile app file (APK or IPA).
The file is securely stored for processing.
Security Analysis Begins
The system scans the uploaded file to detect security issues.
The scan includes:
Checking the app's code for security vulnerabilities.
Analyzing how the app behaves during execution.
Receive Security Report
Once the scan is complete, the system generates a security report.
The report includes:A summary of security risks found in the app.
Recommendations to improve security.

Figure 11 Non-technical Diagram

CHAPTER 6
DEMONSTRATION

Demonstration

TrustForge Sign in Page

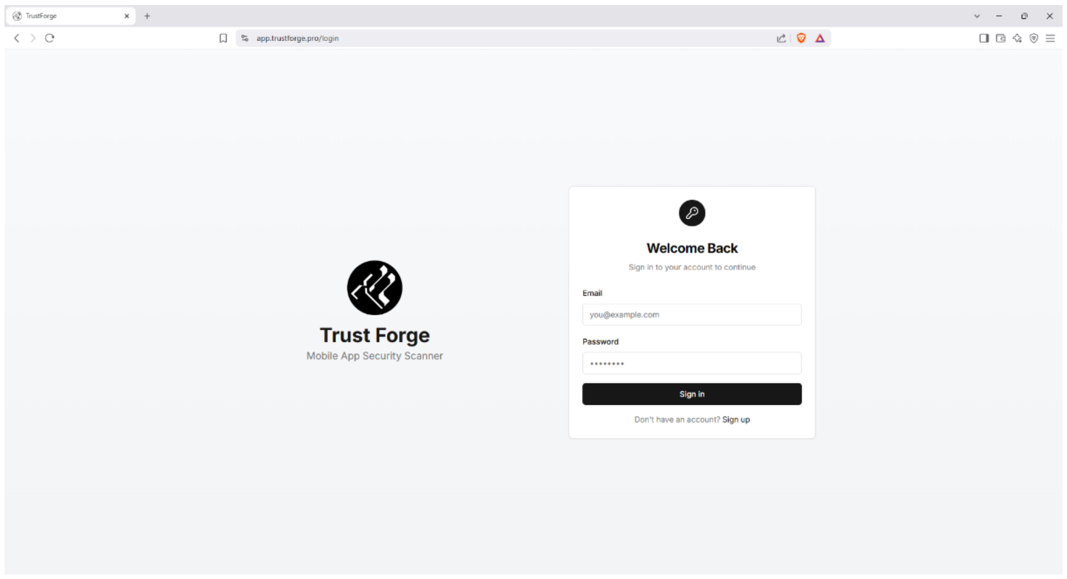


Figure 12 TrustForge Sign-in page

TrustForge dashboard

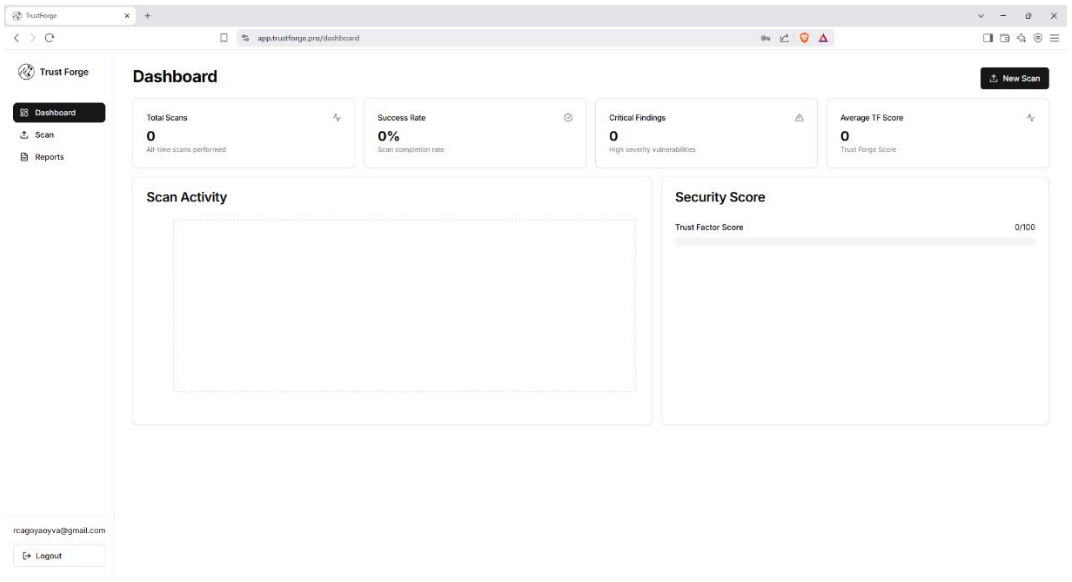
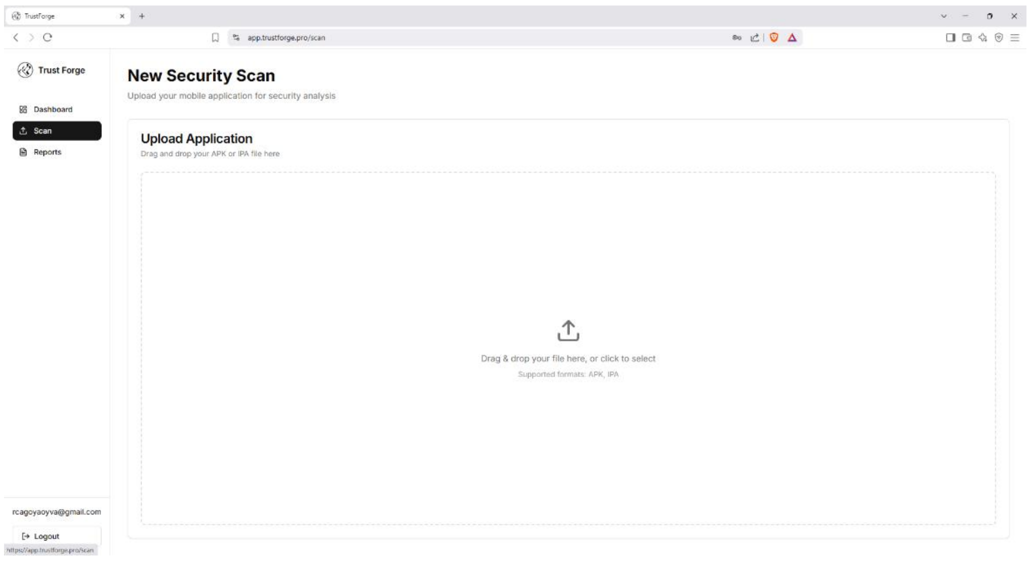


Figure 13 TrustForge Dashboard

TrustForge upload



New Security Scan

Upload your mobile application for security analysis

Upload Application

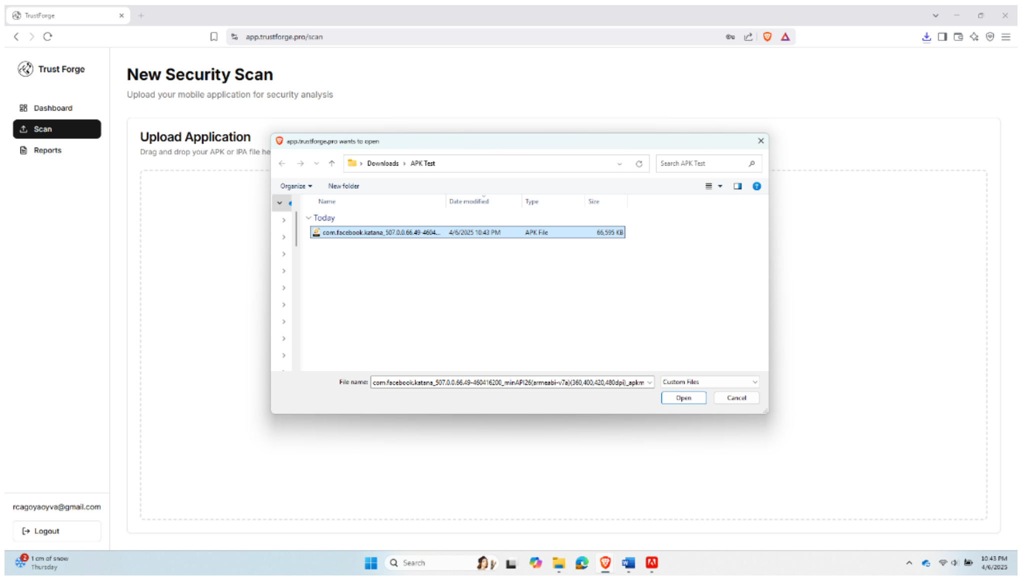
Drag and drop your APK or IPA file here

Drag & drop your file here, or click to select

Supported formats: APK, IPA

Figure 14 TrustForge Upload Page

TrustForge initiating scan



New Security Scan

Upload your mobile application for security analysis

Upload Application

Drag and drop your APK or IPA file he

app.trustforge.pro wants to open

Downloads > APK Test

Name	Date modified	Type	Size
Today			
com.facebook.katana_507.0.0.66.49-4604...	4/6/2025 10:43 PM	APK File	66,595 KB

File name: com.facebook.katana_507.0.0.66.49-460416200_minAPI26(armeabi-v7a)(360,400,420,480dpi)_apkm

Figure 15 TrustForge upload file

TrustForge Scan Process

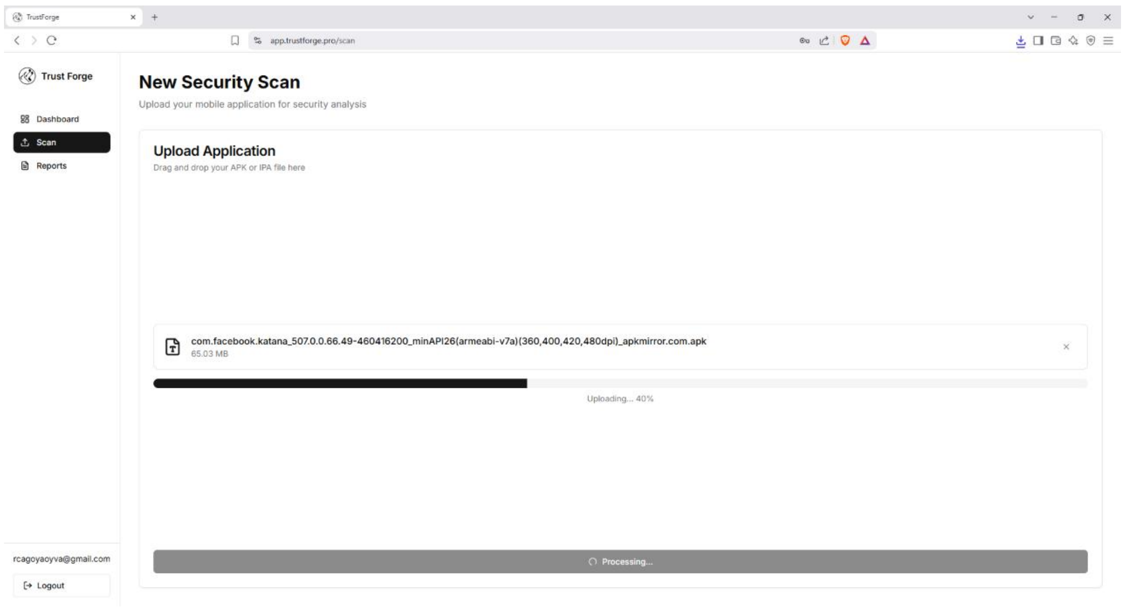
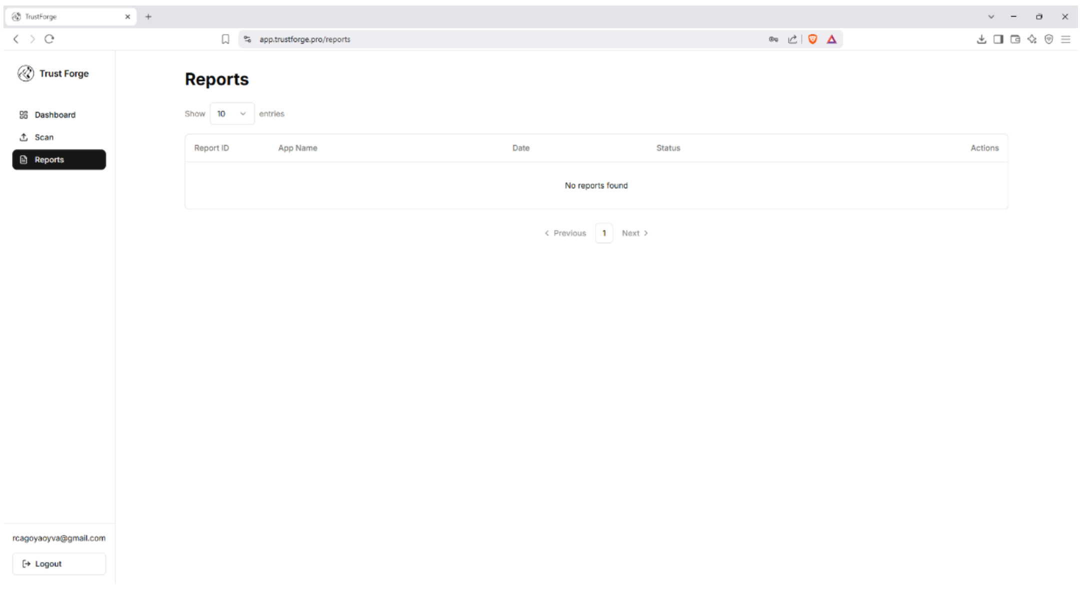

TrustForge
app.trustforge.pro/scan
Trust Forge
Dashboard
Scan
Reports
New Security Scan
Upload your mobile application for security analysis
Upload Application
Drag and drop your APK or IPA file here
com.facebook.katana_507.0.0.66.49-460416200_minAPI26(armeabi-v7a)(360,400,420,480dpi)_apkmirror.com.apk
65.03 MB
Uploading... 40%
Processing...
rcagoyaoyva@gmail.com
Logout

Figure 16 TrustForge processing the file

TrustForge reporting


TrustForge
app.trustforge.pro/reports
Trust Forge
Dashboard
Scan
Reports
Reports
Show 10 entries

Report ID	App Name	Date	Status	Actions
No reports found				

‹ Previous 1 Next ›
rcagoyaoyva@gmail.com
Logout

Figure 17 TrustForge reporting page

CHAPTER 7
CONCLUSION

Conclusion

In conclusion, this project developed the Mobile App Security Analyzer to meet the growing demand for enhanced mobile application security. The project successfully outlined a framework for assessing and improving app security across iOS and Android platforms. By exploring various analysis methodologies, evaluating security tools, and examining real-world case studies, the project provided a thorough understanding of the complexities of mobile app security. The emphasis on identifying vulnerabilities, mitigating threats, and adhering to data protection regulations highlights the project's contribution to safeguarding user data and fostering a secure mobile ecosystem. While demonstrating the capabilities of security tools, the project reinforces the ongoing need for vigilance and adaptation in the face of evolving cyber threats.

REFERENCES

References

Alshahrani, A. & Alzahrani, A. (2023). An Android applications vulnerability analysis using MobSF. ResearchGate. https://www.researchgate.net/publication/383024798_An_android_applications_vulnerability_analysis_using_MobSF

Alzahrani, A. & Alharthi, A. (2018). DDefender: Android application threat detection using static and dynamic analysis. IEEE Xplore. https://ieeexplore.ieee.org/document/8326293

Alzahrani, A. & Alharthi, A. (2019). RanDroid: Structural similarity approach for detecting ransomware applications in Android platform. In IEEE International Conference on Electro Information Technology (EIT) (pp. 0892 - 0897). IEEE. https://ieeexplore.ieee.org/document/8500161

Anderson, R. (2020). Security engineering: A guide to building dependable distributed systems (3rd ed.). Wiley.

Digital.ai. (2024). 2024 application security threat report. https://digital.ai/resource-center/ebooks/2024-application-security-threat-report/

Frida. (n.d.). Frida documentation. https://frida.re/docs/home/

Gollmann, D. (2011). Computer security (2nd ed.). Wiley.

IBM. (n.d.). IBM Security AppScan. https://www.ibm.com/security/application-security/appscan

IBM Security. (2024). Cost of a data breach report 2024. https://www.ibm.com/reports/data-breach

Immunio. (2020, September 15). Automated mobile app security testing with MobSF: An overview. Medium. https://medium.com/@immunio/automated-mobile-app-security-testing-with-mobsf-an-overview-d9b6493d2245

Khan, M. A. & Alzahrani, A. I. (2020). A privacy and security analysis of early-deployed COVID-19 contact tracing Android apps. Empirical Software Engineering, 25(6), 1-30. https://doi.org/10.1007/s10664-020-09934-4

Kumar, R. & Singh, A. (2018). Mobile application security: A comprehensive guide to mobile security. Springer.

Lifewire. (2024, November 27). Most invasive apps: The apps that collect the most user data. https://www.lifewire.com/most-invasive-apps-8716661

Miller, M. (2017). Mobile app security: A comprehensive guide to protecting your mobile applications. O'Reilly Media.

MobSF. (n.d.). Mobile Security Framework (MobSF). GitHub. https://github.com/MobSF/Mobile-Security-Framework-MobSF

NIST. (2018). NIST Special Publication 800-163: Vetting the security of mobile applications. National Institute of Standards and Technology. https://csrc.nist.gov/publications/detail/sp/800-163/final

OWASP. (n.d.). OWASP mobile security testing guide. https://owasp.org/www-project-mobile-security-testing-guide/

Ragh - Raghavan, S. (2018). Mobile security: A comprehensive guide to protecting your mobile devices. Apress.

Sari, D. P. & Prabowo, H. (2022). Android apps vulnerability detection: A case study of the Mobile JKN application. Journal of Information Security and Applications, 68, 103-115. https://doi.org/10.1145/3556974

Stallings, W. & Brown, L. (2019). Computer security: Principles and practice (4th ed.). Pearson.

Sullivan, B. (2019). The mobile application hacker's handbook: A guide to mobile application security. Wiley.

Veracode. (2021). The state of software security: Mobile application security. https://www.veracode.com/security/state-software-security-report

Zainuddin, N. & Rahman, A. (2023). Security assessment of Malaysian mobile applications: A detailed analysis. Security and Privacy, 6(1), e200. https://doi.org/10.1002/sec.200

APPENDIX

Appendix

- 30 second video link: https://youtu.be/bvlP0H8gKYI?feature=shared
- Group portfolio link: https://puneet1517kaur.wixsite.com/code-sanctum-1
- GitHub Project Front-end Link: https://github.com/ajitesh-kuppa-sivakumar/trustforge-fe
- GitHub Project Front-end Link: https://github.com/ajitesh-kuppa-sivakumar/trustforge-be