RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 07, Issue 03 March

http://dx.doi.org/10.47392/IRJASH.2025.015

Secure APK Installation Using Block Chain: A Decentralized Approach to Threat Prevention

Arthy Rajakumar[1], Monica[2], Gayathiri[3], Pon karthiga[4]
[1]Associate Professor, Dept. of IT, Kamaraj College of Engg. & Tech., Virudhunagar, Tamil nadu, India.
[2,3,4]UG Scholar, Dept. of IT, Kamaraj College of Engg. & Tech., Virudhunagar, Tamil nadu, India.
Emails: *arthyit@kamarajengg.edu.in[1], 22uit040@kamarajengg.edu.in[2], 22uit026@kamarajengg.edu.in[3], 22uit012@kamarajengg.edu.in[4]*

Article history
Received: 28 January 2025
Accepted: 11 February 2025
Published: 03 March 2025

Keywords:
Blockchain, APK security, smart contracts, threat detection, decentralized verification, machine learning, real-time monitoring, cybersecurity, crowdsourced intelligence, tamper-proof authentication.

Abstract
The rapid proliferation of mobile applications has heightened security concerns related to malware infections, data breaches, and unauthorized access. Traditional security mechanisms primarily rely on centralized verification models, which introduce vulnerabilities such as tampering, single points of failure, and slow response times to emerging threats. This paper introduces a blockchain-based approach for secure APK installation, incorporating immutable records, decentralized verification, and real-time threat detection. The proposed framework integrates machine learning for dynamic behavioral analysis and smart contracts to enforce security policies automatically. By leveraging blockchain, the system ensures the integrity of APKs, verifies developer authenticity, prevents tampering, and enables a collaborative threat intelligence network for real-time security updates. Furthermore, the decentralized nature of blockchain eliminates reliance on centralized authorities, providing a transparent, efficient, and tamper-proof APK verification mechanism. The focus will be on optimizing blockchain consensus mechanisms, advancing smart contract functionalities, and integrating AI-powered threat intelligence for dynamic cybersecurity.

1. Introduction

The widespread use of mobile applications has introduced significant security risks, necessitating robust verification mechanisms to ensure the integrity of APK installations. Cyber threats such as malware injections, phishing attacks, and unauthorized modifications of APK files have increasingly targeted mobile users. Conventional security mechanisms—such as antivirus scanning, application store verification, and signature-based detection—struggle to provide effective, real-time security. These traditional methods rely on centralized architectures that introduce single points of failure, making them vulnerable to targeted cyberattacks and tampering. Additionally, the lag between the discovery of new malware and the updating of security databases creates a window of opportunity for attackers to exploit vulnerabilities. Blockchain technology has emerged as a revolutionary solution by decentralizing security verification and ensuring data immutability. Unlike conventional security models, blockchain-based APK verification leverages a distributed ledger to store cryptographic hashes of verified applications, preventing unauthorized modifications and ensuring transparency. Smart contracts automate security enforcement by executing predefined policies, such as validating APK hashes, verifying developer credentials, and restricting installation of untrusted applications. Furthermore, machine learning algorithms enhance security by dynamically analyzing application behavior, detecting anomalies, and identifying zero-day threats before they can cause harm. This paper proposes a comprehensive blockchain-based security framework that integrates decentralized trust mechanisms, machine learning-powered threat detection, and automated smart contract policies to secure APK installations. By leveraging blockchain's immutable records, collaborative threat intelligence, and distributed validation mechanisms, the system ensures robust security and resilience against emerging cyber threats. The subsequent sections explore related research, system architecture, role of blockchain, advantages of decentralized security, and future directions for enhancing blockchain-driven APK verification.

2. Existing System

The integration of blockchain into cybersecurity has been extensively studied in recent years, particularly in securing mobile applications. Several researchers have explored blockchain's potential in addressing APK security challenges, including integrity verification, decentralized authentication, and automated threat mitigation. Moosavi and Taherdoost (2023) conducted a systematic review of blockchain applications in cybersecurity, emphasizing its role in decentralizing security verification. Their study highlights how blockchain eliminates single points of failure and ensures that security records remain tamper-proof, making it an ideal solution for securing APK installations. Verma and Ram (2024) explored blockchain's potential in data security, particularly its role in preventing unauthorized access and data tampering. Their research discusses how blockchain's immutable ledger strengthens APK verification by ensuring that only trusted versions of applications are installed, mitigating risks of malware-infected APKs. AlFaw, Elmedany, and Sharif (2022) analyzed security vulnerabilities within blockchain frameworks, identifying concerns such as 51% attacks and smart contract exploits. Their findings underscore the need for robust consensus mechanisms and continuous security audits, which can be integrated into blockchain-based APK verification systems to prevent unauthorized modifications. Osterrieder et al. (2024) examined the application of machine learning in identifying fraudulent blockchain activities. Their study suggests that combining blockchain with machine learning models significantly enhances APK security by detecting behavioral anomalies and identifying potential threats before they cause harm. Dwivedi et al. (2024) classified blockchain-based attacks and their corresponding mitigation strategies. Their research demonstrates how real-time monitoring and decentralized trust mechanisms improve blockchain's resilience against cyber threats, making it an effective tool for securing APK verification processes. Homayoun et al. (2019) proposed a blockchain-based framework for detecting malicious mobile applications in app stores. Their system leverages a dual blockchain structure to validate APK integrity and developer authenticity, ensuring that only verified applications are installed. The reviewed studies highlight blockchain's potential in enhancing mobile security through decentralized verification, immutable records, and smart contract automation. However, challenges such as scalability, computational efficiency, and integration with existing mobile security frameworks remain areas for further research. The proposed blockchain-based APK security system builds on these studies by integrating collaborative threat intelligence, machine learning-driven security analysis, and automated security enforcement mechanisms to create a robust, future-proof security framework for mobile applications. [1-5]

3. Proposed System

The methodology for securing APK installations through a blockchain-based decentralized approach involves several key steps and components. The process can be divided into phases, focusing on system design, implementation, integration of blockchain, machine learning, and smart contracts, and evaluation of security effectiveness. The overall system is explained in Figure 1.

3.1. APK Registration and Upload Process

3.1.1. APK Submission

Developers submit APK files for installation through a secure developer portal. The APK file's hash is calculated (using a hashing algorithm like SHA-256) and stored on the blockchain as part of a transaction. The hash serves as a unique fingerprint for the APK file. APK installations. By leveraging blockchain's immutable records, collaborative

3.1.2. Developer Verification

The developer's identity is verified through digital certificates or decentralized identity solutions (e.g., self-sovereign identity based on blockchain).This step ensures that only verified developers can upload APKs and establishes trust in the application's source. Figure 1 Shows Overall System Architecture [6-10]

3.1.3. Immutable Record Creation

Upon successful verification, an immutable record of the APK file, including its hash, metadata (e.g., version number, developer name), and any relevant security certifications, is created on the blockchain. The record is timestamped, ensuring that any changes to the APK file in the future can be easily detected

.

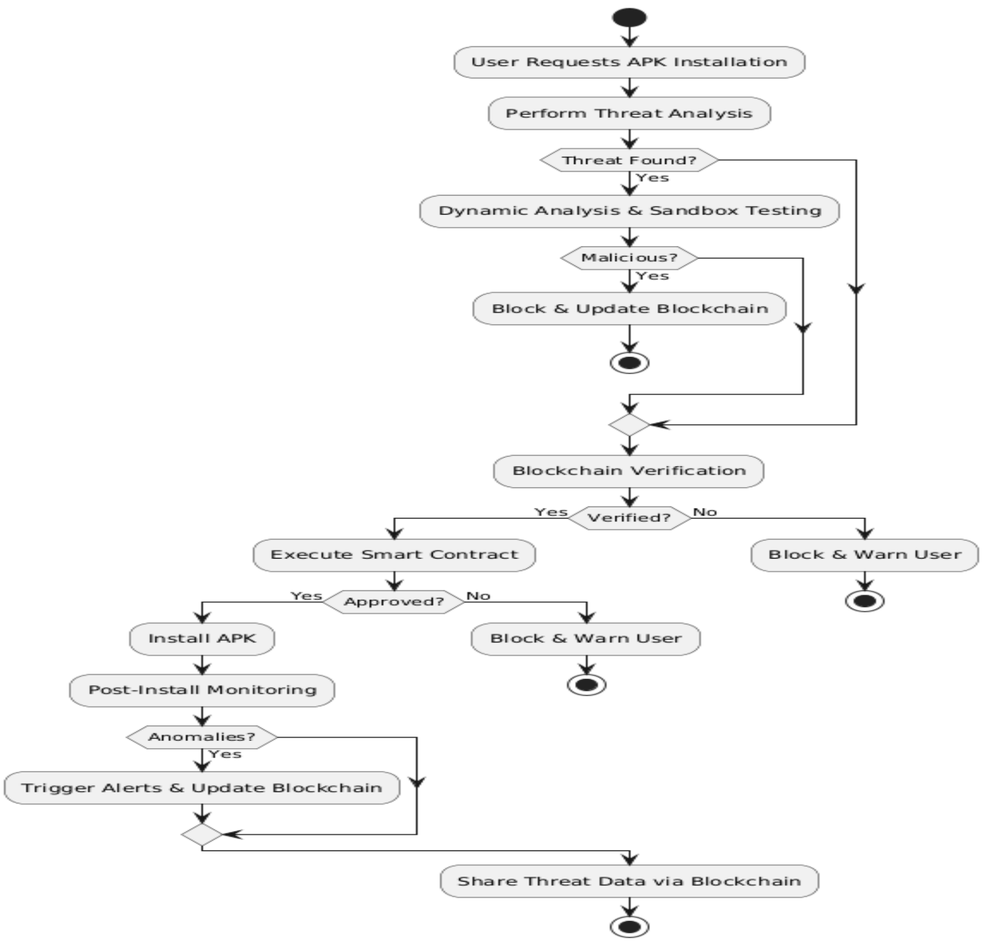


Figure 1 **Overall System Architecture**

3.2. System Design & Architecture

3.2.1. Block Chain Network Setup

Choose a suitable blockchain platform (e.g., Ethereum, Hyperledger, or a custom-built blockchain) to deploy a decentralized ledger for APK verification Implement a public or private blockchain network depending on the desired level of transparency and accessibility. Public blockchains ensure complete transparency, whereas private blockchains may be used for enterprise-specific applications. Define the roles and permissions for the network participants, including application developers, users, and security validators.

3.2.2. Smart Contract Development

Develop smart contracts that define security policies and validation rules for APK installations. These smart contracts can perform operations such as:

- Verifying the cryptographic hash of APKs against the blockchain ledger.
- Confirming the identity and reputation of the developer.
- Enforcing policies like restricting the installation of unsigned or tampered APK

files.

3.2.3. Decentralized Verification Nodes

Set up a decentralized network of validators that will verify the integrity and authenticity of APK files. These nodes can include developers, trusted security researchers, and other stakeholders. Each node will participate in a consensus process to verify the APK and record the result on the block chain. [11-15]

3.3. APK Installation and Real-time Threat Detection

3.3.1. User-side Verification

When a user attempts to install an APK, the user's device checks the APK's hash against the blockchain's stored hash. If the APK has been tampered with, the installation will be blocked. The device can query the blockchain to ensure the APK is from a trusted developer and has not been altered since its registration.

3.3.2. Machine Learning Integration for Dynamic Threat Detection

Implement machine learning models that analyze the APK's behavior once installed on a device, monitoring for any signs of malicious activity. The models can detect unusual behaviors such as unexpected resource consumption, unauthorized access to sensitive data, or abnormal network communication patterns. When a threat is detected, the system can trigger a smart contract that alerts users and automatically blocks the APK.

3.3.3. Real-time Threat Intelligence

Use blockchain's decentralized nature to share real-time threat intelligence among network participants. For example, if a new malware strain is detected in the system, it can be recorded and flagged on the blockchain. All nodes (validators) are instantly updated with this information, preventing further installations of the compromised APK.

3.4. Smart Contract Enforcement

3.4.1. Enforcing Security Policies

Develop specific smart contracts that enforce predefined security policies during APK installation, such as:

- Checking for compatibility with the device's OS version.
- Ensuring the APK has not been tampered with by verifying its hash.
- Blocking APKs that do not pass developer authentication or fail machine learning-based threat detection.

3.4.2. Automated Security Responses

In case of a detected security issue, the smart contract can automatically:

Flag the APK for review. Block further installations or remove any already-installed instances from devices. Alert the user and the developer about the detected issue.

3.5. Collaborative Threat Intelligence and Consensus

3.5.1. Consensus Mechanism

Utilize blockchain's consensus mechanism (such as Proof-of-Work or Proof-of-Stake) to ensure that only validated and trusted APKs are allowed on the platform. Consensus ensures that the decentralized network of validators agrees on the validity of an APK before it is registered or installed.

3.5.2. Crowdsourced Threat Intelligence

The decentralized nature of the system allows security researchers, developers, and other stakeholders to contribute and share threat intelligence. As new threats are identified, they are immediately recorded on the blockchain, providing all participants with up-to-date information and allowing the system to adapt dynamically to emerging threats.

3.6. Continuous Monitoring and Feedback Loop

3.6.1. Ongoing APK Behaviour Monitoring

Once the APK is installed on the user's device, continuous monitoring and analysis are carried out to detect potential security threats. Anomaly detection through machine learning models ensures real-time identification of unknown threats.

3.6.2. User Feedback Integration

User-reported feedback and ratings can be integrated into the block chain system. If multiple users report security issues related to a particular APK, the system can trigger an automatic review process.

3.6.3. Block Chain Analytics

Regular audits of the block chain can provide insights into APK installation trends, developer behaviour, and APK integrity, helping to identify patterns of malicious activity.

3.7. Evaluation and Performance Testing

3.7.1. Security Effectiveness

Perform security tests on the system to verify its ability to detect and prevent various threats, such as malware injections, phishing attacks, and unauthorized APK modifications. Compare the blockchain-based APK security system with traditional security mechanisms in terms of detection accuracy, response time, and user experience.

3.7.2. Performance Testing

Evaluate the performance of the blockchain network, including transaction speed, scalability, and system reliability. Ensure the blockchain can handle high volumes of APK verification requests in real-time. Test the integration of machine learning models and smart contracts under real-world conditions.

3.7.3. User Experience Assessment

Conduct user surveys and usability studies to assess the ease of use of the system and the effectiveness of the verification and installation process from the user perspective.

8.Future Work:

3.8. Advanced Consensus Mechanisms

Explore more efficient consensus algorithms, such as Proof-of-Authority or Delegated Proof-of-Stake, to improve scalability and reduce transaction costs.

3.8.1. AI-enhanced Threat Intelligence

Integrate advanced AI systems to provide predictive threat analysis and automate the discovery of zero-day vulnerabilities.

3.8.2. Smart Contract Optimization

Improve the flexibility and adaptability of smart contracts to respond to complex, evolving security requirements. This methodology outlines a robust, decentralized approach to APK security using block chain, machine learning, and smart contracts, offering enhanced protection against evolving cyber threats while ensuring the integrity and authenticity of mobile applications.

4. System Architecture and Workflow

The system architecture integrates multiple security layers, leveraging block chain, machine learning, and real-time monitoring for secure APK installations. When a user initiates an APK installation as shown in figure 2, the system triggers a security validation process. This ensures that the application undergoes rigorous security checks before execution. The multi-layered approach prevents unauthorized or harmful installations. Figure 2 shows Extraction of APK Metadata APK installations. By leveraging blockchain's immutable records, collaborative threat intelligence, and distributed

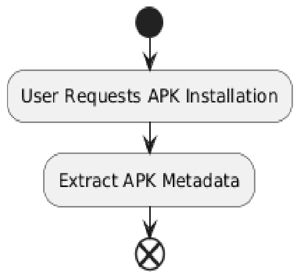


Figure 2 Extraction of APK Metadata

4.1. Initial Threat Analysis

Security tools such as Drozer and Quark scan the APK for vulnerabilities, suspicious permissions, and known malware signatures. This static analysis helps identify potential threats before execution. Any flagged APKs undergo further scrutiny. The figure 3 describes the initial threat analysis in detail.

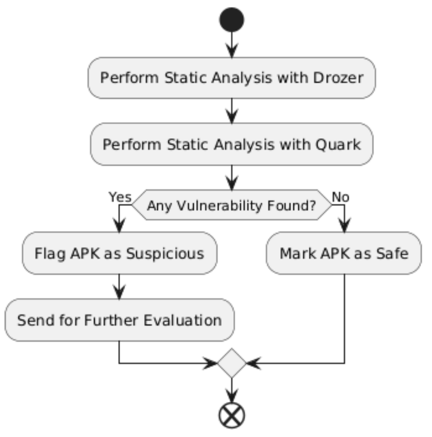

Perform Static Analysis with Drozer
Perform Static Analysis with Quark
Yes
Any Vulnerability Found?
No
Flag APK as Suspicious
Mark APK as Safe
Send for Further Evaluation

Figure 3 Initial Threat Analysis

4.2. Dynamic Behaviour Analysis

Machine learning models analyse the APK's runtime behaviour to detect hidden malware patterns. If anomalies are found, the system marks the APK for further evaluation. This proactive approach helps detect evolving threats. The figure 4 describes the initial threat analysis in detail.

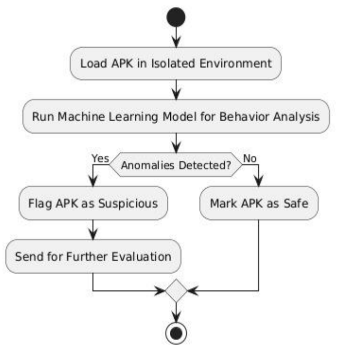

Load APK in Isolated Environment
Run Machine Learning Model for Behavior Analysis
Yes
Anomalies Detected?
No
Flag APK as Suspicious
Mark APK as Safe
Send for Further Evaluation

Figure 4 Dynamic Behaviour Analysis

4.3. Handling Unknown Threats

If an APK exhibits suspicious behavior, it is tested in a controlled sandbox environment. If deemed malicious, its details are added to the blockchain to warn other users. This step enhances adaptive security against emerging threats as shown in Figure 5. [16-20]

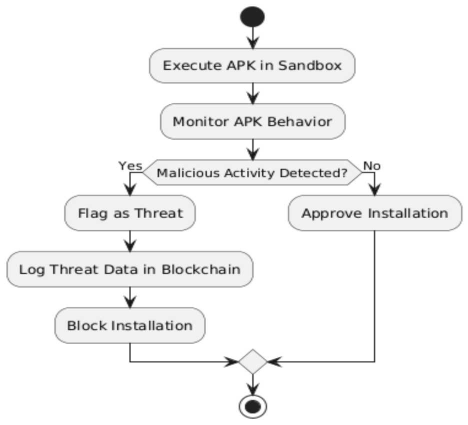


Figure 5 Handling Unknown Threats

4.4. Block Chain Verification Layer

The figure 6 shows how the blockchain verification layer works.

- APK Hash Verification: A cryptographic hash is generated and compared with verified hashes stored on the blockchain, ensuring file integrity.
- Developer and Version Authentication: Blockchain records validate the APK's source and ensure it is an official, updated version.
- Tamper-Proof Security: Immutable blockchain data prevents unauthorized modifications, securing APK metadata from manipulation.

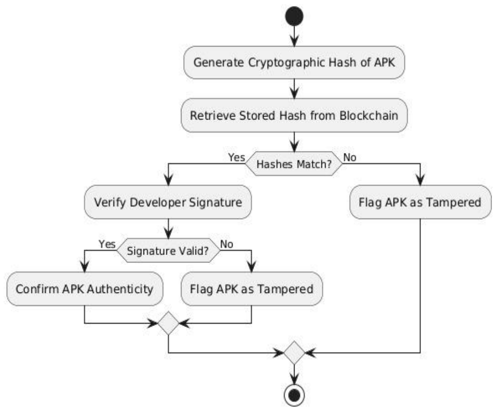


Figure 6 Block Chain Verification Layer

4.5. Smart Contract Implementation

Smart contracts automate the verification and security enforcement process (Figure 7):

- Automated Installation Approval: A smart contract checks the APK's hash, developer signature, and behavior reports before permitting installation.
- Enforcing Security Policies: Smart contracts define conditions under which an APK is allowed or blocked, ensuring decentralized, trustless decision-making.
- Dynamic Threat Response: If an APK is flagged as malicious, the smart contract can trigger immediate action, such as blocking its installation and updating the block chain threat intelligence.

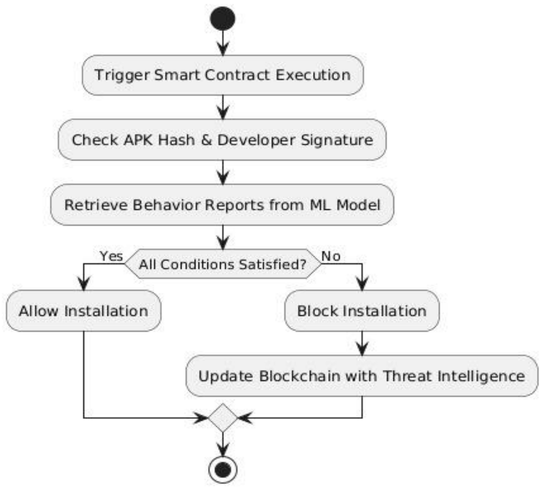

Trigger Smart Contract Execution
Check APK Hash & Developer Signature
Retrieve Behavior Reports from ML Model
Yes
All Conditions Satisfied?
No
Allow Installation
Block Installation
Update Blockchain with Threat Intelligence

Figure 7 Smart Contract Implementation

4.6. Outcome Decision

Based on all security checks and smart contract evaluations, the system either allows or blocks the APK installation. If threats are detected, the user is notified with a warning as shown in figure 8. Secure applications proceed to the final installation stage.

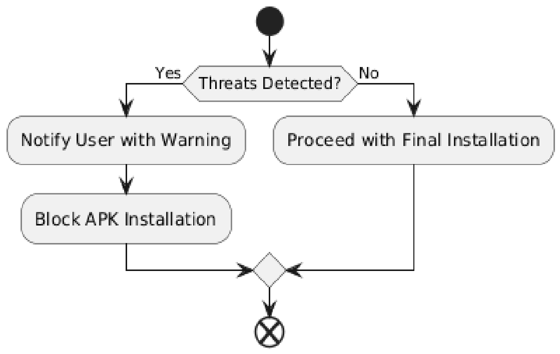


Figure 8 Outcome Decision

4.7. Post-Installation Monitoring

Even after installation, the system continuously monitors the application for abnormal behavior. If security risks emerge, alerts are triggered, and users are advised to take necessary actions as shown in Figure 9. [21-25]

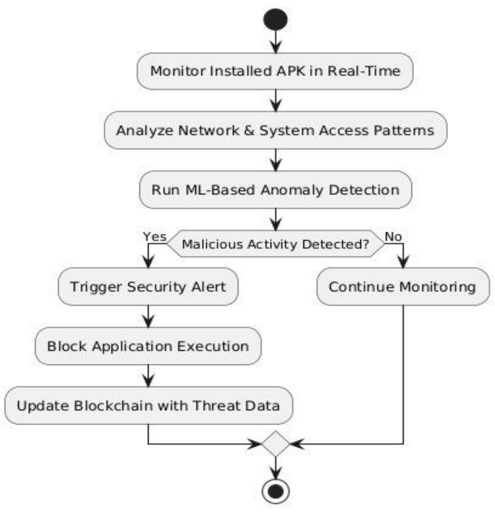

Monitor Installed APK in Real-Time
Analyze Network & System Access Patterns
Run ML-Based Anomaly Detection
Malicious Activity Detected?
Yes
No
Trigger Security Alert
Continue Monitoring
Block Application Execution
Update Blockchain with Threat Data

Figure 9 Post-Installation Monitoring

4.8. Crowdsourced Threat Intelligence

Threat intelligence data is shared across the block chain network, ensuring all users receive the latest security updates. Smart contracts facilitate automated updates and enforce compliance with security protocols. This decentralized approach strengthens overall protection against new malware threats.

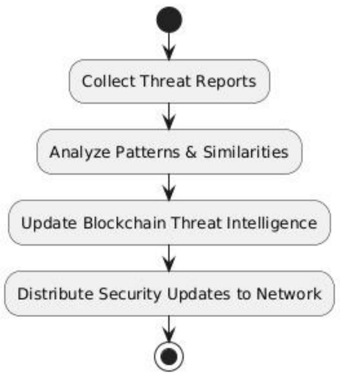
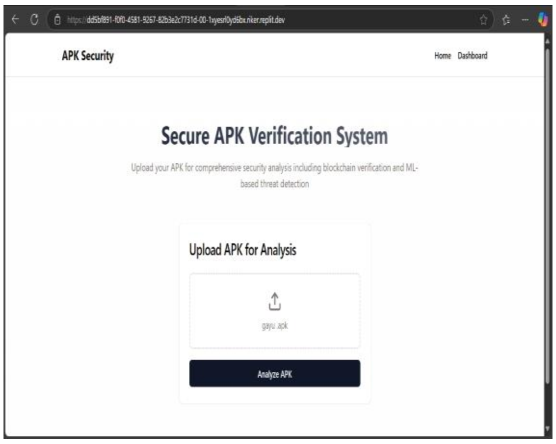

Collect Threat Reports
Analyze Patterns & Similarities
Update Blockchain Threat Intelligence
Distribute Security Updates to Network

Figure 10 Crowdsourced Threat Intelligence

5. Results and Discussion

The Secure APK Verification System analyzes APK files using blockchain verification and ML-based threat detection to ensure security. Server logs confirm API requests for APK analysis, showing consistent hashing, validating that files were not tampered with, and ensuring real-time threat monitoring. Logs also show key timestamps of Post and get requests, confirming the system's efficiency. gayu.apk was analyzed and found to be safe with a 0% threat score, verified via blockchain, and allowed for installation, requiring only internet and access_network_state permissions. fake.apk, however, had a 67% threat score, flagged as dangerous due to excessive permissions like Read_sms , send_sms , access_fine_location , and dangerous_permission. It failed blockchain verification, was detected using suspicious APIs, and was blocked from installation to prevent security risks. The security mechanism effectively prevents the installation of harmful apps while ensuring only trusted APKs can be installed.



Upload your APK for comprehensive security analysis including blockchain verification and ML-based threat detection

Figure 11 Upload APK

Figure 11 displays a Secure APK Verification System webpage where users upload APKs for security analysis, with gayu.apk selected for verification. The Secure APK Verification System confirms a successful APK upload and the start of analysis. [26-30]

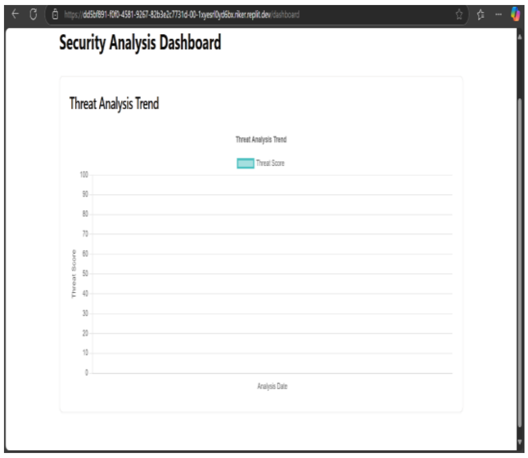
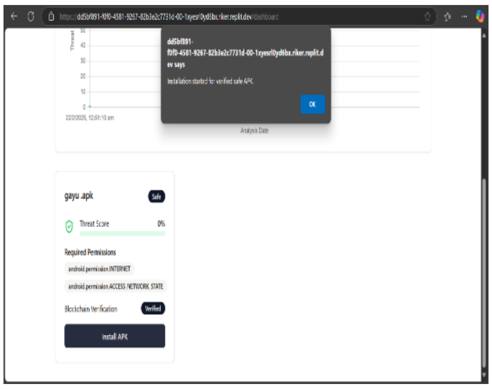


Figure 12 Dashboard

The Security Analysis Dashboard graph (Figure 12) shows a 0% threat score, confirming the APK is safe.



dd5bf891-f0f0-4581-9267-82b3e2c7731d-00-1xyesrl0yd6bx.riker.replit.dev says

Installation started for verified safe APK.

gayu .apk — Safe

Threat Score 0%

Required Permissions

android.permission.INTERNET

android.permission.ACCESS_NETWORK_STATE

Blockchain Verification — Verified

Figure 13 Threat Analysis

The Security Analysis Dashboard shows gayu.apk (0% threat) being installed after blockchain verification (Figure 13). The Secure APK Verification System shows fake.apk uploaded for analysis using blockchain and ML detection. The APK Security webpage confirms a successful APK upload and the start of analysis.

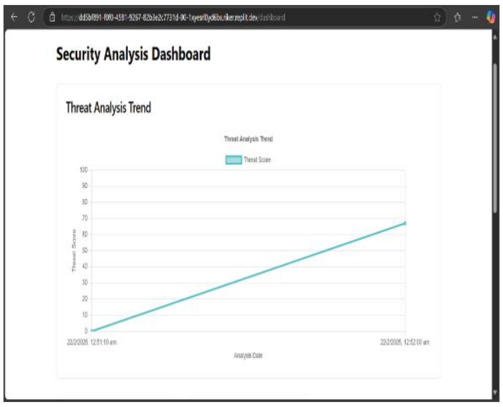
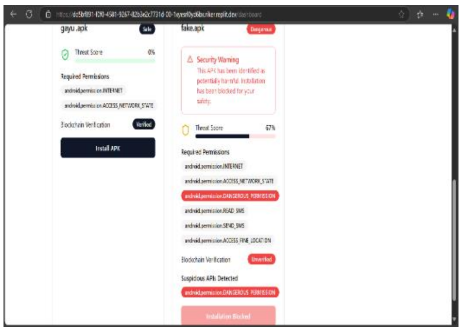


Figure 14 Threat Analysis Trend

The Security Analysis Dashboard graph (Figure 14) shows a rising threat score, indicating increasing APK risk. [31-33]



gayu .apk — Safe

Threat Score 0%

Required Permissions

android.permission.INTERNET

android.permission.ACCESS_NETWORK_STATE

Blockchain Verification — Verified

fake.apk — Dangerous

Security Warning

This APK has been identified as potentially harmful. Installation has been blocked for your safety.

Threat Score 67%

Required Permissions

android.permission.INTERNET

android.permission.ACCESS_NETWORK_STATE

android.permission.DANGEROUS_PERMISSION

android.permission.READ_SMS

android.permission.SEND_SMS

android.permission.ACCESS_FINE_LOCATION

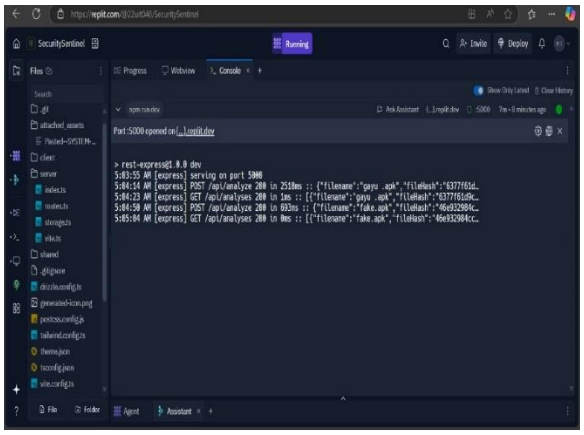
Blockchain Verification — Unverified

Suspicious APIs Detected

android.permission.DANGEROUS_PERMISSION

Figure 15 Threat Score

The Dashboard graph shows fake.apk with a 67% threat score, signaling potential harm (Figure 15).



Port :5000 opened on {...}.replit.dev

```
> rest-express@1.0.0 dev
5:03:55 AM [express] serving on port 5000
5:04:14 AM [express] POST /api/analyze 200 in 2518ms :: {"filename":"gayu .apk","fileHash":"6377f61d…
5:04:23 AM [express] GET /api/analyses 200 in 1ms :: [{"filename":"gayu .apk","fileHash":"6377f61d9c…
5:04:50 AM [express] POST /api/analyze 200 in 693ms :: {"filename":"fake.apk","fileHash":"46e932984c…
5:05:04 AM [express] GET /api/analyses 200 in 0ms :: [{"filename":"fake.apk","fileHash":"46e932984cc…
```

Figure 16 Log

The logs (Figure 16) show an Express.js server on port 5000 handling POST and GET requests for APK analysis, processing gayu.apk and fake.apk with their respective hashes.

Conclusion

The analysis of the two APK files, gayu.apk and fake.apk, reveals significant differences in their security status. gayu.apk is deemed safe with a 0% threat score. It requests only basic permissions like Internet and Access Network State, which are common for applications requiring network access. Additionally, it has passed block chain verification, confirming its authenticity and security. fake.apk, on the other hand, is identified as a potential security risk with a 67% threat score. It requests dangerous permissions such as READ_SMS, SEND_SMS, and ACCESS_FINE_LOCATION, which can compromise user privacy and security. The blockchain verification for this app is unverified, indicating a possible lack of authenticity or tampering. As a result, its installation has been blocked to prevent potential harm. Log Analysis: The logs show API requests for analyzing both APKs, with responses confirming their security status. The detection system effectively identified and flagged the risky APK, ensuring proactive security measures. The security system effectively differentiates between a safe and a potentially harmful APK. fake.apk is classified as dangerous due to its suspicious permissions, while gayu.apk is verified and secure. This demonstrates the importance of permission analysis and blockchain verification in preventing malicious app installations.

References

[1]. Zhang, L., & Liu, Q. (2024). Blockchain and Artificial Intelligence Integration for Enhancing IoT Security
[2]. Wang, J., et al. (2025). Security and Privacy Challenges in Blockchain Applications
[3]. Choo, K. R., et al. (2025). Advanced Forensic Investigation of Decentralized Blockchain Applications
[4]. Song, X., et al. (2025). Decentralized Finance (DeFi) Security: A Blockchain-Based Approach to Prevent Attacks
[5]. Moosavi, S., & Taherdoost, H. (2023). Blockchain Technology Application in Security: A Systematic Review
[6]. Verma, A., & Ram, S. (2024). Application of Blockchain Technology in Data Security
[7]. Osterrieder, J., et al. (2024). Enhancing Security in Blockchain Networks: Anomalies, Frauds, and Detection Techniques
[8]. Dehghantanha, A., et al. (2024). A Systematic Literature Review of Blockchain Cyber Security
[9]. Weng, S., et al. (2024). Blockchain-Enabled Secure Internet of Medical Things (IoMT) Systems: Security and Privacy Solutions
[10]. Li, T., et al. (2025). Leveraging Blockchain for Secure Data Sharing in Edge Computing Networks
[11]. Zhou, Y., & Yu, H. (2025). Blockchain-Enabled Supply Chain Management: Challenges, Solutions, and Applications
[12]. Xu, S., et al. (2024). Blockchain-Based Smart Contracts: A Comprehensive Survey of Security Vulnerabilities
[13]. Choo, K. R., et al. (2024).Internet of Things Security and Forensics: Challenges and Opportunities
[14]. Jia, X., & Zhao, H. (2025). A Blockchain-Based Approach for Secure and Transparent Voting Systems
[15]. Homayoun, S., et al. (2019). Blockchain framework for detecting malicious mobile apps.
[16]. Iqbal, M., & Matulevičius, R. (2019). Security risks in blockchain-based applications.
[17]. Alfandi, O., & Alshaher, A. (2023). Blockchain & ML integration for data privacy.
[18]. Zhang, L., & Liu, Q. (2024). Blockchain & AI for IoT security.
[19]. Wang, J., et al. (2025). Security & privacy challenges in blockchain.
[20]. Choo, K. R., et al. (2025). Forensic investigation of blockchain apps.
[21]. Song, X., et al. (2025). Blockchain security in DeFi applications.
[22]. Moosavi, S., & Taherdoost, H. (2023). Blockchain applications in security.
[23]. Verma, A., & Ram, S. (2024). Blockchain in data security.
[24]. Osterrieder, J., et al. (2024). Fraud detection in blockchain networks.
[25]. Dehghantanha, A., et al. (2024). Blockchain cybersecurity review.
[26]. Weng, S., et al. (2024). Blockchain for IoMT security & privacy.
[27]. Li, T., et al. (2025). Secure data sharing in edge computing.
[28]. Zhou, Y., & Yu, H. (2025). Blockchain in supply chain management.
[29]. Xu, S., et al. (2024). Smart contract security vulnerabilities.
[30]. Arxiv (2024). Blockchain for privacy applications.
[31]. Nature (2023). Blockchain for transactional data privacy.
[32]. ResearchGate (2023). Blockchain security in networks.
[33]. IEEE (2023). Blockchain for network security.