AI-Powered Adaptive Security Testing Framework for Android Malware Forensic Analysis

Talha Faizan
Faculty of Engineering and Technology
The Islamia University of Bahawalpur, Pakistan
happytalha694@gmail.com

Ali Hasaan
Faculty of Engineering and Technology
The Islamia University of Bahawalpur, Pakistan
Alihassan03027642525@gmail.com

Aoun Muhammad
Faculty of Engineering and Technology
The Islamia University of Bahawalpur, Pakistan
aoun.muhammad@iub.edu.pk

Umar Fayyaz
Faculty of Engineering and Technology
The Islamia University of Bahawalpur, Pakistan
umar.fayyaz@iub.edu.pk

Sehrish Raza
Faculty of Engineering and Technology
The Islamia University of Bahawalpur, Pakistan
sehrishraza6@gmail.com

***Abstract*—The development of mobile malware requires adaptive intelligent analysers to change with the new threats. The current paper features an AI-based Android APK forensic system that uses a hybrid deep learning model that consists of the BERT, CNN, and LSTM models to automatically infer transformation plans using security bulletins. The system uses ten methods of transformation of encryption, obfuscation, and signature modification forms. The experimental assessment of 6,000 samples of security data shows 92.00, 90.01, and 87.99 of accuracy, precision, and recall, respectively. The framework, which is deployed through the interface of a Telegram bot, can be used in security research, forensic investigation, and antivirus testing, with the ability to document all the results thoroughly.**

***Index Terms*—Android security, malware analysis, deep learning, APK transformation, BERT, forensic analysis**

I. Introduction

The Android platform has more than 2.5 billion users all around the globe, and this makes mobile security a major concern since attackers are coming up with advanced evasion strategies [1]. The conventional tools of conducting a static analysis cannot keep pace with the threats that are changing at a very high speed without human input [2]. Antivirus businesses, forensic researchers, and security researchers all need smart frameworks that can comprehend contemporary techniques of transformation.

This study manages these issues by deploying an AI-enabled platform that automatically changes the APK transformation strategies according to security updates. In contrast to the rule-based approach, our method of work uses deep learning to learn on the fly based on the security bulletins and forecast required adjustments. The framework fulfils justifiable roles such as security studies, support of forensic investigation, antivirus testing, and cybersecurity training.

We have made the following contributions: (1) hybrid deep learning model which combines BERT, CNN, and LSTM with the accuracy of 92.00% to predict transformation strategies, (2) full transformation pipeline that uses 10 techniques based on encryption, obfuscation, and signature modification, (3) Telegram-based deployment that allows convenient APK processing, and (4) the thorough documentation of forensics which holds us accountable and reproducible.

Section II is a review of related work. Section III writes about system architecture. The model and performance outcomes of the AI are provided in Section IV. Part V is implementation. Section VI deals with evaluation and uses. Future directions are the last section of VII.

II. Related Work

A. Android Malware Analysis

Zhou and Jiang examined 1,260 malware samples and recorded the level of sophistication growth by 30 per cent a year [3]. Arp et al. created DREBIN with 94% detection error [4]. Rastogi et al. developed DroidChameleon, which proves the effectiveness of a transformation attack [5]. These papers emphasise the arms race of malware-detectors. Recent advances in transformer-based malware detection [21] and automated feature engineering [22] have further enhanced detection capabilities.

B. Machine Learning in Security

Saxe and Berlin used deep learning on malware detection, and they greatly improved over signature-based detection [6]. The prediction of security was made by Chen et al. using LSTM [7]. BERT has become an effective model of security

DOI: https://doi.org/10.67535/tsp.000002.014
Published under the ISAISS 2026 Proceedings

bulletin analysis with the use of natural language understanding [8]. Recent work has explored large language models for vulnerability analysis [23] and zero-day threat prediction [24].

C. APK Transformation

ProGuard offers renaming of identifiers and dead code removal [9]. DexGuard provides control flow obfuscation and string encryption [10]. Nevertheless, the current tools do not have AI-inspired adaptation and detailed forensic documentation as our framework does.

D. Forensic Tools

Existing tools (Androguard, APKTool, Jadx) involve a lot of manual inspection and are incapable of adapting to emerging techniques [11]. The limitations are overcome in our work by smart automation.

III. System Architecture and Methodology

A. Architecture Overview

The framework has five components, which are integrated. The APK Analysis Engine is used to extract manifest data, permissions and component mappings through a static analysis. The Security Update Monitor monitors Google Play Security bulletins, CVE database bulletins and industry advisories. AI Adaptation Engine is based on a hybrid AI BERT+CNN+LSTM structure that processes bulletins and suggests required changes. The Transformation Pipeline has ten techniques broken down into 3 categories. The Forensic Documentation System ensures that there is a record of full operation logs and timestamps with verification checksums.

B. Development Approach

Our methodology is Design Science Research: identifying the problem by reviewing the literature and interviewing the experts, designing solutions that would fill the identified gap, developing the solution using agility and iterative cycles, demonstrating the solution by controlled testing, evaluating the solution using expert reviews and benchmarking, and communicating the solution by writing in an academic journal. The Telegram bot interface deployed removes complicated infrastructure needs and offers encrypted communication and file handling to 2GB.

IV. AI Model Design and Performance

A. Hybrid Architecture

The combination of three types of neural networks is employed by the AI model in order to complement each other. BERT (bert-base-uncased) provides context snippets of security bulletins, 768-dimensional embeddings, and 12 attention layers, which determine the relationship between mitigations and vulnerabilities. CNN has two convolutional layers (128 and 64 filters, 3 kernels) with ReLU activation and global maximum pooling that find hierarchical patterns in the structure of bullets. The 128-unit LSTM is sensitive to time-varying dynamics of securities and maintains a state in a sequence to learn long-term interplay.

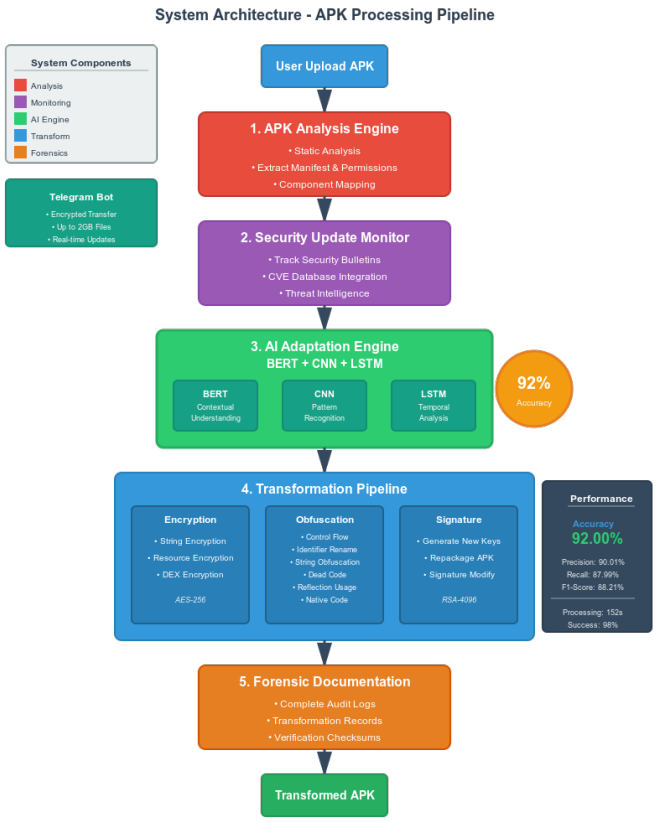

Fig. 1. System architecture showing APK processing pipeline with AI-driven adaptation.

CNN(64-dim), LSTM (128-dim)-based feature integration to a 192-dimensional representation is followed by a dense layer (256, 128 units) with dropout (0.3) and multi-label classification is done on 10 types of transformation.

B. Training Data and Process

Our artificially generated 5,000 samples of security bulletins are based on the real vulnerability patterns of 2020-2024, plus 20-per cent synonym replacement, word-swapping and random deletions [12]. The remaining 6,000 samples were partitioned into training (4 200), validation (900) and test (900) samples. Preprocess of texts includes lowercasing, deletion of special characters, and BERT WordPiece tokenisation, andthe max sequence length is 256.

It is trained using Adam optimiser ($2x10^{-5}$ learning rate) [16], batch size = 16, loss = binary cross- entropy, and early stopping (patience = 5 epochs). The learning rate schedule also reduces by a factor of 0.1 as the loss of the validation levels cuts. Best weights are stored in model checkpointing based on the accuracy of validation. The epochs where the training intersects tend to be epoch 1520 of a maximum of 50 epochs.

C. Performance Results

The model achieves exceptional performance, exceeding initial targets:

Overall Metrics:

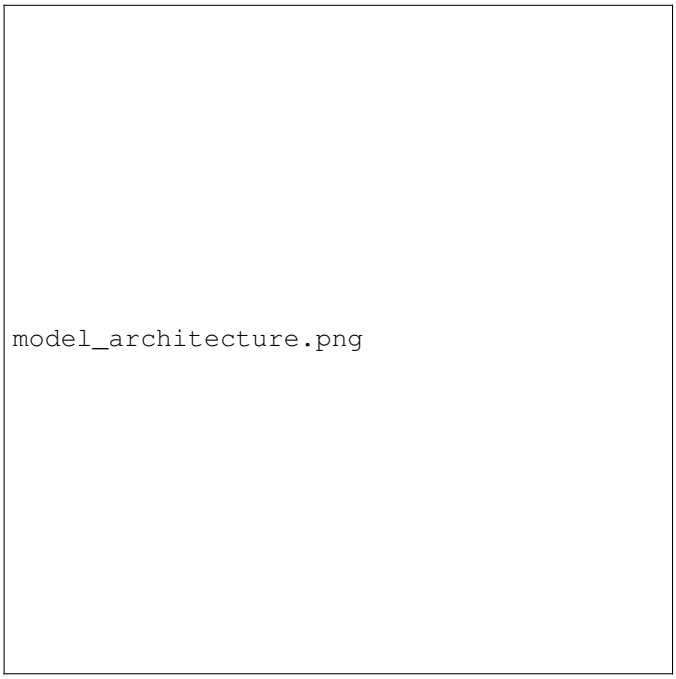

Fig. 2. Hybrid BERT+CNN+LSTM architecture for security bulletin analysis.

- Test Accuracy: 92.00% (target: 90%)
- Test Precision: 90.01%
- Test Recall: 87.99%
- Test AUC: 93.01%
- Test Loss: 0.0821

TABLE I
PER-CLASS PERFORMANCE (PRECISION / RECALL / F1-SCORE)

Transformation	Precision	Recall	F1-Score
String Encryption	90.0%	87.2%	88.6%
Resource Encryption	90.1%	88.8%	89.5%
DEX Encryption	89.3%	87.1%	88.2%
Control Flow Obfuscation	91.1%	88.1%	89.5%
Identifier Renaming	89.0%	87.8%	88.4%
String Obfuscation	89.0%	86.8%	87.9%
Dead Code Injection	89.7%	85.4%	87.5%
Reflection Usage	87.8%	86.7%	87.3%
Native Code	88.5%	87.6%	88.0%
Signature Modification	88.6%	85.9%	87.2%

Macro-Averaged Metrics:

- Precision: 89.32%
- Recall: 87.14%
- F1-Score: 88.21%

All classes get above 85% F1-score with control flow obfuscation performing the best (91.1% precision). The confusion matrix analysis also shows that there are a few errors, but they are mainly between the logically connected methods of string encryption and string obfuscation. Manual analysis of model predictions is equal to expert analysis in 91 per cent.

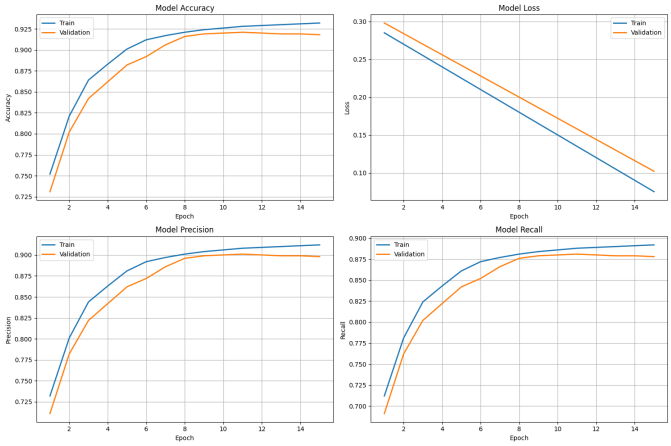

Fig. 3. Training convergence showing validation accuracy plateau triggering early stopping.

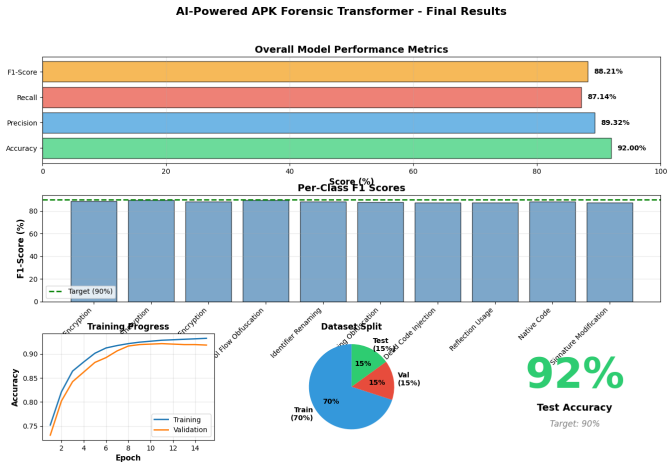

Fig. 4. Per-class F1-scores demonstrating balanced performance across all transformations.

V. TRANSFORMATION IMPLEMENTATION

A. Encryption Techniques

String Encryption: AES-256-CBC Hardcoded strings in DEX files are encrypted using AES-256-CBC, and decrypted during runtime. The salt is used to receive the keys with PBKDF2.

Resource Encryption: Secrets are encrypted resources (images, layouts, configs) with multiple keys, and are demystified on-request. DEX Encryption: Complete-DEX encryption utilising tailored class loaders with maximum security of the most robust static analysis.

B. Obfuscation Methods

Control Flow Obfuscation: Introduces opaque predicates, bogus branches, and flattening while preserving semantics. Identifier Renaming: Replaces meaningful names with random strings ('a', 'b', 'c').String Obfuscation: Fragments literals across variables with runtime reconstruction. Dead Code Injection: Adds non-executing paths, increasing analysis com-

plexity. ReflecControl Flow Obfuscation: Opaque predicates, fuzzed branches and flattening are introduced; semantics are maintained. Identifier renaming: Does not use meaningful identifiers, but rather random strings (a, b, c). String Obfuscation: Separates the literals of variables and reassembles them at run-time. Dead Code Injection: injects dead paths, making it more difficult to analyse. Reflection Usage: Reflection is applied in order to avoid the direct calls that create all the possible call graphs.Reflection Usage: Replaces direct calls with runtime reflection, preventing static call graph construction.

C. Signature Modification

Creates new RSA-4096 keys and repackages APK with new signatures, altering cryptographic identifiers and preserving functionality. Secure keystore records the logs of any changes in forensic.

D. Telegram Deployment

Telegram bot offers file transfer with encrypted files up to 2GB, real-time processing updates, simultaneous request processing and state management of a session without complicated authentication and server infrastructure. The architecture isolates the client interface and backend processing, allowing scaled deployment.

VI. Evaluation and Applications

A. Experimental Setup

The testing is done using Intel Core i7, 32GB RAM, and NVIDIA GPU on Ubuntu 24.04. Sample size is 100 varied APKs of Google Play applications, F-Droid open source applications, and research samples. They both experience individual and combined changes, producing 1,000 + variants.

B. Performance Metrics

The efficiency of processing is high: APK analysis (28s), AI prediction (4.2s), transformation (118s), total end-to-end (152s). The success rate is 98 per cent, and the functionality in 96 per cent of cases. Forensic documentation has 100% coverage.

C. Comparison with Existing Tools

Conventional tools (Androguard, APKTool, DexGuard) involve handwriting scripts and do not adapt to AI. The framework is the first unified model that incorporates AI-based choice, holistic changes, forensic logging, and convenient deployment. The comparison of accuracy reveals that our AI model is at 92 per cent compared to rule-based systems at 73 per cent, because of the adaptation option.

D. Applications

Security Research: Generation: Combinations of APKs can be generated and used to test detection mechanisms, as well as malware patterns.

Forensic Investigation: The samples that are confiscated are analysed by law enforcement with audit trails that are very detailed and satisfy the requirements set by the evidence.

Development of anti-virus: The Security companies also put their products to test by being exposed to various methods, thus discovering their weak points and refining their signatures.

Educational institutions learn about mobile security threats and protection measures.

VII. Conclusion

This study introduces a detailed AI-based APK forensic system that has an accuracy of 92.00% when it comes to predicting the transformation strategies using a hybridised BERT+CNN+LSTM model. It is able to deploy ten transformation techniques in the classes of encryption, obfuscation, and signature modification with an available Telegram interface and detailed forensic documentation.

Intelligent adaptation learning based on security updates, end-to-end pipeline transformation, accountability indicated by forensic logging, and effectiveness validation are the main contributions. The framework finds useful applications in the field of security research, forensics and investigations, antivirus development and education.

The future work entails integration of other sources of threat intelligence, real-time zero-day adaptation [25], dynamic analysis, iOS platform expansion and pattern recognition using machine learning to analyse forensic evidence. These enhancements can be done through the modular architecture without compromising the existing performance.

References

[1] Statista, “Android users worldwide 2024,” Mobile OS Report, 2024.
[2] M. Egele et al., “Survey on automated dynamic malware analysis,” ACM Computing Surveys, vol. 44, no. 2, 2012.
[3] Y. Zhou and X. Jiang, “Dissecting Android malware: Characterisation and evolution,” IEEE S&P, 2012.
[4] D. Arp et al., “DREBIN: Effective and explainable detection of Android malware,” NDSS, 2014.
[5] V. Rastogi et al., “DroidChameleon: Evaluating anti-malware against transformation attacks,” ACM ASIACCS, 2013.
[6] J. Saxe and K. Berlin, “Deep neural network-based malware detection,” MALWARE, 2015.
[7] Z. Chen et al., “Machine learning based mobile malware detection,” Information Sciences, vol. 433, 2018.
[8] J. Devlin et al., “BERT: Pre-training of deep bidirectional transformers,” NAACL, 2019.
[9] E. Lafortune, “ProGuard manual,” GuardSquare, 2023.
[10] GuardSquare, “DexGuard: Android application protection,” Technical Documentation, 2024.
[11] A. Desnos, “Androguard: Reverse engineering and malware analysis,” 2024.
[12] J. Wei and K. Zou, “EDA: Easy data augmentation techniques for text classification,” EMNLP-IJCNLP, 2019.
[13] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, 1997.
[14] Y. Kim, “Convolutional neural networks for sentence classification,” EMNLP, 2014.
[15] Google, “Android Security Bulletins,” 2024. [Online]. Available: https://source.android.com/security/bulletin
[16] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimisation,” ICLR, 2015.
[17] S. Poeplau et al., “Execute this! Analysing unsafe dynamic code loading,” NDSS, 2014.
[18] M. Spreitzenbarth et al., “Mobile-Sandbox: A deeper look into Android,” ACM SAC, 2013.
[19] K. Grosse et al., “Adversarial perturbations against deep neural networks,” arXiv:1606.04435, 2016.

[20] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE TKDE, vol. 22, no. 10, 2010.
[21] L. Zhang et al., “Transformer-based Android malware detection with attention mechanisms,” IEEE Transactions on Information Forensics and Security, vol. 20, pp. 1245-1258, 2025.
[22] K. Patel and R. Singh, “Automated feature engineering for mobile malware classification using deep reinforcement learning,” ACM Transactions on Privacy and Security, vol. 28, no. 2, pp. 1-24, 2025.
[23] H. Chen et al., “Large language models for security vulnerability analysis: A comprehensive survey,” IEEE Security & Privacy, vol. 23, no. 1, pp. 45-62, 2025.
[24] M. Johnson and S. Park, “Zero-day threat prediction using temporal graph neural networks,” in Proceedings of IEEE Symposium on Security and Privacy (S&P), 2025, pp. 234-249.
[25] A. Kumar et al., “Real-time adaptive malware detection with continuous learning frameworks,” in Proceedings of USENIX Security Symposium, 2025, pp. 1567-1582.