International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 1, Jan-Feb 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Demystifying Android Malware Detection with Explainable AI

Miss. Shruti Bodke, Miss. Prachi Patil, Miss. Tanvi Patil, Prof. Prachi Dhanawat
Department of Information Technology,
Usha Mittal Institute of Technology,SNDT University

----------**********----------

Abstract:

Machine learning–based techniques are widely regarded as effective solutions for detecting Android malware and have shown strong performance by utilizing commonly adopted features. However, in real-world applications, most machine learning models only provide a simple classification result, such as labeling an application as malicious or benign. In practice, security analysts and other stakeholders are more concerned with understanding the reasons behind such classifications. This challenge belongs to the field of interpretable machine learning, particularly within the domain of mobile malware detection. Although several interpretability techniques have been proposed in other artificial intelligence research areas, there has been limited work focusing on explaining why an Android application is identified as malware and addressing the domain-specific difficulties involved.

To address this limitation, this paper introduces a new interpretable machine learning framework, called XMal, which is capable of both accurately classifying malware and providing meaningful explanations for its decisions. The first stage of XMal employs a multi-layer perceptron combined with an attention mechanism to highlight the most influential features contributing to the classification outcome. The second stage automatically generates natural language explanations that describe the primary malicious behaviors found within applications. The proposed approach is evaluated through human studies and quantitative analysis, and is further compared with existing interpretable methods such as Drebin and LIME. The results demonstrate that XMal can more precisely uncover malicious behaviors and can also explain the causes of misclassifications. This study provides valuable insights into interpretable machine learning through the lens of Android malware detection and analysis.

Keywords: **Android Malware Detection, Interpretable Machine Learning, Malware Classification, Attention Mechanism, Security Analysis**

----------**********----------

I. INTRODUCTION

With the rapid growth of mobile technology, Android applications have become an essential part of daily life, storing sensitive information such as banking details, personal messages, and private data. As the popularity of Android devices has increased, cyber attackers have shifted their focus toward mobile platforms and now actively target users through malicious applications. This has made Android malware a serious security threat in recent years.

Several approaches have been proposed to detect Android malware. Traditional signature-based techniques depend on continuously updated malware databases and are ineffective against newly emerging threats. Behavior-based methods rely on predefined malicious patterns and therefore struggle to recognize unknown attacks. Data flow–based techniques mainly focus on identifying data leakage but cannot capture all malicious activities. More recently, researchers have adopted machine learning algorithms such as k-nearest neighbors, support vector machines, random forests, and gradient boosting to classify Android applications. These models commonly use features like permissions and API calls and have achieved high detection accuracy. Deep learning techniques, including convolutional and

recurrent neural networks, have further improved performance.

Despite their success, these machine learning–based approaches only provide a simple classification result, indicating whether an application is malicious or benign. Such limited output is insufficient in many real-world scenarios. For example, app store administrators need to understand the exact harmful behaviors of an application before deciding whether to remove it. Security analysts must manually inspect malware to identify its actions, which is time-consuming and difficult when dealing with large datasets. Moreover, understanding how a model makes its decisions is crucial for building trust in automated systems and defending against adversarial attacks that attempt to manipulate classifiers.

To overcome these limitations, it is necessary not only to detect malware accurately but also to explain the reasoning behind each classification. However, existing interpretability techniques either provide explanations that are too generic or fail to consider the relationships between different features. Therefore, a specialized and domain-aware interpretability approach is required for Android malware detection.

In this work, we propose XMal, an interpretable machine learning framework that integrates an attention mechanism with a neural classifier to identify influential features and automatically generate human-readable descriptions of malicious behaviors. This approach aims to improve both the accuracy and transparency of Android malware detection systems.

II.RESEARCH BACKGROUND

Machine learning has become a powerful tool for classification tasks, but many models operate as black boxes, offering little insight into how predictions are made. Interpretability refers to the ability of a model to present its decisions in a form that humans can easily understand. To address this issue, several model-agnostic techniques such as LIME and LEMNA have been proposed to approximate complex models using simpler ones. In addition, interpretability studies in text classification and image recognition have attempted to trace predictions back to meaningful components such as words or image regions.

In the context of Android malware detection, applications are analyzed using features extracted from permissions, intents, and API calls. Security analysts typically inspect configuration files and source code to identify suspicious operations and map them to malicious behaviors. Some existing systems, such as Drebin, attempt to interpret classifications by examining feature weights from linear models. However, these methods often rely on global model parameters rather than sample-specific explanations, leading to inaccurate interpretations. Gradient-based methods approximate complex models but introduce unavoidable bias.

Attention mechanisms have recently gained popularity in deep learning due to their ability to highlight important parts of the input. Originally developed for neural machine translation, attention assigns different weights to input elements based on their relevance to the output. This not only improves accuracy but also provides interpretability by indicating which components influence the prediction most strongly. Attention has been successfully applied in computer vision and natural language processing to explain model behavior.

However, traditional attention mechanisms operate on vector-based inputs and cannot be directly applied to the scalar features used in malware detection. Furthermore, existing interpretability techniques often ignore correlations between features, such as the relationship between permissions and API calls. These correlations are crucial in understanding malicious behaviors.

Motivated by these challenges, this study introduces a customized attention-based framework designed specifically for Android malware analysis. By learning the relationships among features and assigning meaningful weights, the proposed approach aims to produce accurate classifications along with clear explanations of malicious activities. This background establishes the foundation for developing an interpretable and

reliable malware detection system tailored to the Android platform.

III. LITERATURE SURVEY

Android malware detection has been widely studied due to the increasing security threats on mobile platforms. Early research mainly focused on traditional signature-based detection techniques, where known malware patterns are stored in databases and matched against new applications. Although this method is simple and fast, it fails to detect new or modified malware variants, making it ineffective against evolving threats.

To overcome this limitation, behavior-based detection approaches were introduced. These methods analyze the runtime actions of applications and compare them with predefined malicious behavior patterns. While such techniques are capable of identifying unknown malware, they depend heavily on manually defined behavior rules and require extensive expert knowledge. Moreover, they may miss sophisticated attacks that disguise their behavior.

Another line of research adopted data flow analysis to identify privacy leakage and unauthorized data transmission. These methods track the movement of sensitive information within applications and detect suspicious flows. Although effective for identifying data leakage, they cannot comprehensively capture all types of malicious activities.

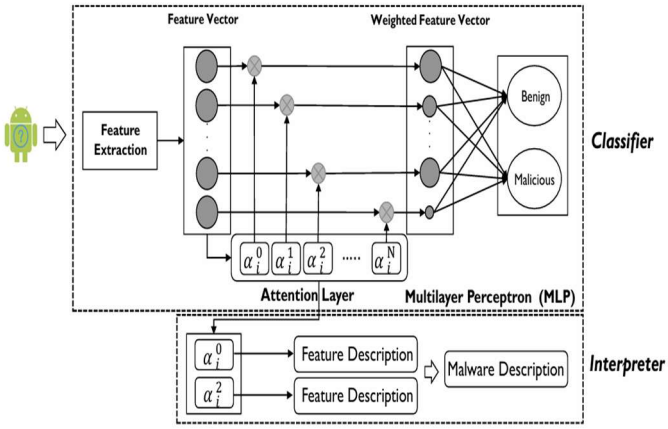
With the advancement of machine learning, many researchers proposed classification-based approaches for Android malware detection. Algorithms such as k-nearest neighbors, support vector machines, random forests, and gradient boosting have been widely used. These methods typically rely on features like permissions, API calls, and intents. Later, deep learning models such as convolutional neural networks and recurrent neural networks further improved detection accuracy by automatically learning complex feature representations.

Despite achieving high accuracy, most of these machine learning models operate as black boxes and only provide binary outputs without explaining the reasons behind their decisions. Some interpretability methods, such as LIME and gradient-based techniques, attempt to approximate the internal behavior of classifiers. However, these methods often ignore feature correlations and produce explanations that do not accurately reflect real malicious behaviors.

The Drebin system introduced a linear classification model that identifies important features based on global model weights. However, this approach generates the same explanation for all samples and does not consider individual application characteristics. Other interpretability studies in text and image domains have shown that attention mechanisms can effectively highlight important inputs and improve transparency.

Based on these observations, there is a clear need for an interpretable Android malware detection system that can both classify applications accurately and provide meaningful, behavior-oriented explanations. This requirement motivates the development of the proposed XMal framework.

IV.PROPOSE SYSTEM DESIGN



The proposed system, named XMal, is designed to perform two main tasks:

- Accurate malware classification

- Automatic generation of human-readable explanations
- The system consists of two major components:
- Classifier Component
- Interpreter Component

Overall Architecture of the Proposed System

1. The workflow of XMal can be summarized as follows:
2. APK files are collected as input.
3. Permissions and API calls are extracted as features.
4. A customized attention-based neural classifier predicts whether the app is malicious.
5. The attention layer identifies the most influential features.
6. These features are mapped to semantic meanings.
7. A behavior description is automatically generated.

Classifier Component

The classifier component is responsible for detecting whether an Android application is malicious. The system extracts two major types of features: permissions and API calls, which are commonly used in malware analysis and provide meaningful semantic information.

Since not all features are useful, a feature selection process is applied to retain only the most discriminative ones. The final feature vector represents the presence or absence of selected permissions and APIs in each application.

A customized neural network is then trained using these feature vectors. The model consists of two layers:

An attention layer, which assigns importance weights to each feature.

A multi-layer perceptron, which performs the final classification.

The attention layer captures the correlation between different features and highlights those that contribute most strongly to the prediction. This allows the model to identify the key factors influencing the classification decision.

Interpreter Component:

The interpreter component explains why an application is classified as malware. It performs the following steps:

- Selects the top influential features based on attention weights.
- Maps each selected feature to its functional meaning using Android developer documentation.
- Groups similar features into semantic categories.
- Applies predefined ordering rules to arrange behaviors logically.
- Generates a natural language description summarizing the malicious activities.
- For example:
- Features related to boot events indicate automatic startup.
- Network APIs indicate internet communication.
- Telephony APIs indicate data collection.
- These semantics are combined to form a complete behavior description such as: “Launches on system startup, collects device information, and transmits data to a remote server.”

Advantages of the Proposed System:

The proposed system offers several benefits:

- Provides both classification and explanation
- Considers feature correlations through attention mechanism
- Generates human-readable behavior descriptions

- Improves trust and transparency in malware detection
- Helps analysts understand malicious activities quickly

Explainable Prediction Using Large Language Models:

After an APK file is submitted by a user, the proposed system performs static analysis to extract relevant features such as permissions and API calls. These features are processed by multiple machine learning classifiers to determine whether the application is malicious or benign. While traditional malware detection systems typically provide only a binary classification result, the proposed framework places strong emphasis on interpretability and transparency.

To enhance explainability, the system integrates a Large Language Model (LLM)–based explanation module that generates human-readable justifications for each prediction. The classifier outputs, together with the most influential features identified during the decision process, are provided as structured inputs to the language model. Based on this information, the model produces a concise natural language explanation describing the primary factors that contributed to the classification outcome.

The explanation module converts low-level technical indicators, such as suspicious permission requests or abnormal API usage patterns, into clear and understandable descriptions of application behavior. This enables users to comprehend not only *what* decision was made, but also *why* the application was classified as malicious or benign.

Existing malware analysis platforms, such as multi-engine scanning services, primarily focus on aggregating detection results and do not provide meaningful, user-oriented explanations for their decisions. In contrast, the proposed system augments malware detection with an automated interpretation layer, improving transparency and supporting informed decision-making.

By combining machine learning–based classification with LLM-driven explanation generation, the proposed framework offers an interpretable and user-centric solution for Android malware detection. This approach enhances trust in automated security systems and supports practical deployment in real-world security analysis environments.

V.Technologies Used

The proposed XMal system utilizes a combination of machine learning models, feature engineering techniques, and natural language processing methods to achieve accurate and interpretable Android malware detection.

The primary features used in the system are permissions and API calls, which are extracted from Android application package (APK) files. Permissions provide information about the access rights requested by an application, while API calls reveal the internal operations performed by the program. These two feature types are widely adopted in mobile security research because they are easy to extract and carry meaningful semantic information.

For classification, a multi-layer perceptron (MLP) neural network is employed as the main prediction model. The MLP consists of an input layer, one or more hidden layers, and an output layer. It is trained using labeled datasets of benign and malicious applications to learn complex patterns in feature combinations.

To improve both accuracy and interpretability, a customized attention mechanism is integrated into the neural network. The attention layer assigns importance weights to each feature, allowing the system to focus on the most relevant permissions and API calls during classification. This mechanism also enables the identification of key features responsible for each prediction.

For explanation generation, the system relies on semantic mapping using official Android developer documentation. Each important feature is mapped to its corresponding functional behavior, such as network communication, data access, or system control. Rule-based ordering

techniques are then applied to organize these behaviors logically.

The final explanation is produced using natural language generation, which converts technical features into readable descriptions that clearly describe the malicious activities detected in an application.

Overall, the system integrates:

Android feature extraction tools
Neural networks (MLP)
Attention mechanisms
Semantic analysis
Rule-based interpretation
Natural language generation

VI.CONCLUSION

In this study, an interpretable Android malware detection framework named XMal has been proposed to address the limitations of traditional machine learning–based security systems. While existing models achieve high classification accuracy, they generally fail to provide explanations for their decisions, making them unsuitable for many practical security applications.

The proposed system combines an attention-based neural classifier with an automatic interpretation module. The attention mechanism highlights the most influential features, while the interpreter translates these features into meaningful descriptions of malicious behaviors. This approach allows the system to not only determine whether an application is harmful but also explain the reasons behind the classification.

Experimental results and human evaluations demonstrate that XMal can accurately identify malicious applications and generate reliable explanations that closely match expert analysis. Compared with existing interpretable methods such as Drebin and LIME, the proposed framework provides more precise behavior identification and clearer explanations.

The study shows that integrating interpretability into malware detection improves transparency, trust, and usability for security analysts and app store administrators. In future work, the framework can be extended to support dynamic analysis, handle new malware families, and adapt to evolving attack techniques.

Overall, XMal represents a promising step toward building intelligent, explainable, and reliable mobile security systems.

REFERENCES

[1] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., & Rieck, K. “Drebin: Effective and Explainable Detection of Android Malware in Your Pocket.” Proceedings of the Network and Distributed System Security Symposium (NDSS), 2014.

[2] Ribeiro, M. T., Singh, S., & Guestrin, C. “Why Should I Trust You? Explaining the Predictions of Any Classifier.” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.

[3] Witten, I. H., Frank, E., & Hall, M. A. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2011.

[4] Hochreiter, S., & Schmidhuber, J. “Long Short-Term Memory.” Neural Computation, 1997.

[5] Bahdanau, D., Cho, K., & Bengio, Y. “Neural Machine Translation by Jointly Learning to Align and Translate.” International Conference on Learning Representations (ICLR), 2015.

[6] Android Developers. “Android API Reference Documentation.” Google Inc.

