An Improved Pre-Exploitation Detection Model for Android Malware Attacks

Hamad Saleh A. Al Besher
Applied College, Najran University, Saudi Arabia | Faculty of Computing, Universiti Teknologi Malaysia, Malaysia
hsalbeshir@nu.edu.sa (corresponding author)

Mohd Fo'ad Bin Rohani
Faculty of Computing, Universiti Teknologi Malaysia, Malaysia
foad@utm.my

Bander Ali Saleh Al-rimy
School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, United Kingdom
bander.al-rimy@port.ac.uk

Received: 27 April 2024 | Revised: 17 June 2024 | Accepted: 25 June 2024

ABSTRACT

This paper presents an innovative approach to the early detection of Android malware, focusing on a dynamic pre-exploitation phase identification system. Traditional methods often rely on static thresholding to delineate the pre-exploitation phase of malware attacks, which can be insufficient due to the diverse behaviors exhibited by various malware families. This study introduces the Dynamic Pre-exploitation Boundary Definition and Feature Extraction (DPED-FE) system to address these limitations, which utilizes entropy for change detection, thus enabling more accurate and timely identification of potential threats before they reach the exploitation phase. A comprehensive analysis of the system's methodology is provided, including the use of vector space models with Kullback-Leibler divergence for dynamic boundary detection and advanced feature extraction techniques such as Weighted Term Frequency-Inverse Document Frequency (WF-IDF) to enhance its predictive capabilities. The experimental results demonstrate the superior performance of DPED-FE compared to traditional methods, highlighting its effectiveness in real-world scenarios.

Keywords-malware; android; pre-exploitation; machine learning, TF-IDF

I. INTRODUCTION

The widespread adoption of mobile technologies and online services has led to numerous cybersecurity challenges that hinder their seamless integration into various aspects of users' lives and businesses. Malicious software, also known as malware, poses a significant threat to the confidentiality, integrity, and availability of data in mobile systems and applications [1]. Mobile malware can be classified into five main types: viruses, worms, trojans, spyware, and ransomware. These malware categories target various platforms, such as mobile devices [2-5]. The history of mobile malware originates from the widespread use of mobile devices, particularly with the rise of smartphones. Subsequently, mobile malware has emerged as a substantial threat to the availability of data and services, affecting both individuals and businesses [6, 7]. Mobile malware enables attackers to gain access to vast resources and personal data [8]. The widespread adoption of Android OS has increased the lure of Android malware, which explains its prevalence in the threat landscape [9-12].

Android malware attacks are specifically designed to compromise personal and business data [13]. By 2016, approximately 13 million instances of Android malware had been identified. The number of malware instances increased to 26.6 million in 2018 as a result of new technologies that allow profitable exploits [14]. Furthermore, according to [15], the number of trojans specifically designed to attack mobile banking applications increased to 53,947. Moreover, the utilization of the dark web, a platform where pilfered data, weaknesses in systems, source code for malicious software, and tools for developing such software are exchanged, serves as an incentive for attackers to generate additional Android malware. While it is possible to use traditional PC solutions for Android devices, it is important to note that the fundamental principles of mobile security are different due to the unique computing systems and resource allocations of these devices.

Although mobile devices have made advances in terms of computing power and capabilities, their limited resources, such as battery life, restrict the ability to perform analysis on the device itself.

Early detection of Android malware is crucial to protecting users' data. This implies that it is crucial to detect these threats during the pre-exploitation phase before any harm. During this phase, the malware is installed on the victim's device and starts searching for the specific files it aims to attack. Several studies have tackled this issue using data collected before an attack occurs to develop machine learning-based detection methods [16]. These studies define the pre-exploitation phase by using either a specific duration of time or a predetermined set of API calls. To successfully detect Android malware before it can compromise a system, it is crucial to accurately identify attacks promptly [6, 9, 17-19]. To accomplish this objective, the detection solution should prioritize the pre-compromise phase. However, identifying Android malware in the pre-compromise phase poses a significant challenge [20]. Challenges in early detection arise from the ambiguous definition of the boundary between the pre-compromise phase, the lack of sufficient early attack patterns, and the complexity of high-dimensional data [21-24]. For accurate early detection, it is crucial to have a sufficient amount of data containing various attack patterns to train the model [25].

Several studies have examined the effectiveness of using the initial data collected during the early stages of malware attacks [16, 17], using fixed thresholding to delineate the initial stage. Two forms of thresholding have been used, specifically time-based and API-based. The time-based approach establishes a predetermined duration for the pre-exploitation phase, where data are gathered from the active process. Similarly, the API-based approach establishes a collection of APIs as the demarcation point that distinguishes malware attacks in the pre-exploitation phase from the subsequent phases. However, the use of fixed thresholding assumes that all malware attacks exhibit identical attack behavior that can be easily detected. However, this assumption does not hold for many Android malware families. The onset of exploitation can vary even between members of the same family due to the polymorphism and metamorphism techniques employed by malware to avoid detection [22, 23, 26]. This makes it difficult to accurately determine the boundary between the pre-exploitation phase and the data requirements for extracting relevant features and training accurate detection models.

The dynamic identification of pre-exploitation phases in malware is a critical aspect of early detection systems. Previous studies, such as [27], introduced a pseudo feedback-based annotated Term Frequency-Inverse Document Frequency (TF-IDF) technique for dynamic crypto-ransomware pre-encryption boundary delineation and feature extraction. Although this approach provides a solid foundation, it focuses primarily on ransomware and uses static thresholding for boundary detection. This study extends this approach by introducing the Dynamic Pre-exploitation Boundary Definition and Feature Extraction (DPED-FE) system, which uses entropy for change detection. This approach enables more accurate and timely identification of potential threats across various types of Android malware before they reach the exploitation phase. To this end, this is a novel entropy-based approach to dynamically determine the pre-exploitation boundary in ransomware detection. The contributions of this study are threefold:

- Dynamic Entropy Integration: Integrate entropy into the calculation of feedback, dynamically adjusting to the unpredictability of observed features. This approach enhances the detection of pre-exploitation activities by accurately identifying transitions in behavior that precede ransomware attacks.
- Enhanced Rocchio Feedback Mechanism: By incorporating entropy into the Rocchio Feedback equation, the traditional feedback mechanism is refined to account for the dynamic nature of malware behavior. This modification allows for more precise differentiation between relevant and irrelevant document vectors, improving the accuracy of the detection system.
- Comprehensive Evaluation: Extensive experiments were conducted using real-world datasets to validate the effectiveness of the proposed method. The results demonstrate significant improvements in the early detection of ransomware activities, reducing false positives and enhancing overall system robustness.

II. RELATED WORKS

Detection during the early stages of a malware attack is crucial to preventing the full execution of malicious activities. In the context of ransomware, which is a type of malware that encrypts files and demands a ransom for decryption, studies have focused on developing techniques to detect it in the pre-encryption stage, where the encryption process has not yet been completed [27-31]. Early detection is essential to stop the encryption process and prevent data loss or system compromise [28]. Various models and algorithms have been proposed to enhance the accuracy and performance of pre-encryption ransomware detection. For instance, eMIFS is a normalized hyperbolic ransomware deterrence model that utilizes the hyperbolic tangent function to individually evaluate features, improving the representation of feature relevance and redundancy [29]. Additionally, a pseudo feedback-based annotated TF-IDF technique has been introduced for dynamic crypto-ransomware pre-encryption boundary delineation and feature extraction, addressing the challenge of limited data availability in the early stages of attacks [27].

Prevention mechanisms, such as pre-encryption detection algorithms, play a vital role in disrupting ransomware operations before they can fully encrypt files and demand ransom [30]. Advanced detection methods, such as those that focus on ransomware anti-analysis tactics, contribute to the early identification of evolving threats in the dynamic cybersecurity landscape [31]. Moreover, the integration of machine learning algorithms and cryptographic techniques has shown promise in improving pre-encryption ransomware detection. Models such as RAPPER (Ransomware Prevention via Performance Counters) leverage performance counters to detect ransomware threats early, particularly by identifying patterns associated with public-key cryptosystems used in encryption processes [20]. By combining machine learning with cryptographic algorithms, researchers aim to analyze attack patterns and enhance detection capabilities [32]. Furthermore, some studies have explored the use of behavioral analysis, entropy estimation, and file entropy analysis to detect ransomware in the pre-encryption stage. Behavioral feature analysis provides insights into ransomware actions during the destruction phase, helping to develop robust detection systems [33]. File entropy analysis coupled with machine learning offers a high detection rate with low false positives and false negatives, improving the overall effectiveness of ransomware detection mechanisms [34].

In [27], a pseudo feedback-based annotated TF-IDF technique was proposed to delineate the pre-encryption boundaries of crypto-ransomware. This approach relies on static thresholding and focuses on ransomware-specific features. In contrast, the proposed DPED-FE system employs a dynamic boundary definition using entropy for change detection, which allows it to adapt to different malware behaviors. Additionally, the proposed feature extraction method incorporates advanced techniques such as Weighted Term Frequency-Inverse Document Frequency (WF-IDF), enhancing the model's predictive capabilities.

III. METHODOLOGY

Unlike the fixed thresholding used in previous studies, such as [27], the proposed DPED-FE scheme employs entropy for change detection to dynamically identify the transition from pre-exploitation to exploitation phases. This approach ensures the accurate collection of pre-exploitation data regardless of the varied time required for each sample to initiate actual sabotage. Furthermore, the improved data vectorization for feature extraction leverages the Rocchio Relevance Feedback (RRF) and WF-IDF techniques, providing a more robust framework for early detection. This allows for the detection of the phase change exhibited by malware during the transition from pre-exploitation to exploitation. The fundamental justification is that the malware must undergo certain crucial procedures to prepare for actual harm [27, 28]. Therefore, this transition could indicate the imminent onset of an exploitation process. This enables the convenient and accurate gathering of pre-exploitation data, irrespective of the differing time needed for each sample to carry out the actual sabotage. By utilizing these data, it is possible to extract representative pre-exploitation attack characteristics, which can then be used to develop more powerful and accurate early detection systems. The DPED-FE system consists of two essential elements: Dynamic Pre-exploitation Boundary Definition (DPED) and Features Extraction (FE) techniques. The DPBD defines the limits of the initial phase of Android malware attacks, while FE extracts important features from this phase to train the early detection model.

A. Raw Data Collection

To obtain up-to-date information from the malware samples in the collection, each sample was transmitted to the sandbox device for dynamic analysis [29]. Upon submission of a sample, the guest machine sandbox agent attaches itself to the process initiated by that sample, capturing all runtime information, including API calls, network traffic, file operations, and cryptographic operations. The data are stored in a trace file specifically assigned to that sample. These files serve as the foundation for constructing the dataset from which attributes are extracted and chosen. After each run, the guest machine was reverted to its initial uninfected state to ensure that the subsequent samples were not tainted by previous ones. Only API calls and their corresponding file operations, encryption, and network activities were preserved in the runtime data collected from each trace file, while all other data were discarded. These data are used to establish the pre-exploitation boundary and extract features.

B. Dynamic Boundary Definition Technique for Pre-Exploitation Data Extraction

The DEBD technique was developed and utilized by coupling the vector space model with Kullback-Leibler divergence (relative entropy) to accurately define the boundary of the pre-exploitation phase. As a result, the pre-exploitation boundary vector was created, which includes all the data related to exploitation used by Android malware during their attacks. This vector defines the boundary of the malware's pre-exploitation phase by identifying a behavioral change that occurs when transitioning from the pre-exploitation to the exploitation phase in the lifecycle of the malware. The fundamental concept at play here is that a significant alteration in malware behavior serves as an indication of an imminent act of sabotage.

The utilization of DEBD deviates substantially from the static approach employed in previous studies. The system monitors the start of exploitation for each specific case, accurately determining the boundary of the pre-exploitation phase by analyzing the sudden shift in malware behavior. DEBD initiates the process of creating the boundary vector. Subsequently, the data for each occurrence are divided into multiple dynamic windows, and relative entropy is then computed for every window. The Euclidean distance is used to measure the distance between adjacent windows. If the distance exceeds a specific threshold, it is considered a transition point that may indicate a shift in behavior. The threshold is determined by calculating the average of the vector values. Any distance that exceeds this average is classified as a boundary point. Subsequently, the collection of important points is formed as a boundary vector. Using this vector as input, a model is trained and subsequently employed to detect behavioral changes.

This approach integrates entropy to dynamically determine the pre-exploitation boundary in malware detection. Let $X = \{x_1, x_2, \ldots, x_n\}$ represent the set of monitored features over time, where x_i is the value of the i^{th} feature at a given time. Entropy $H(X)$ measures the uncertainty or randomness within the feature set X. For a discrete random variable, entropy is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) log p(x_i) \qquad (1)$$

where $p(x_i)$ is the probability of occurrence of x_i. To dynamically detect changes in feature behavior, entropy is calculated over a sliding window. Let W_t be the window of size

k at time t, containing the values $\{x_{t-k+1}, x_{t-k+2}, \dots, x_t\}$. The entropy of the window W_t is given by:

$$H(W_t) = \sum_{i=t-k+1}^{t} p(x_i) \log p(x_i) \quad (2)$$

To detect significant changes, the difference in entropy between consecutive windows is calculated:

$$\Delta H_t = H(W_t) - H(W_{t-1}) \quad (3)$$

A threshold θ is set to identify significant changes. If $|\Delta H_t|$ exceeds θ, it indicates a potential boundary that can be expressed as a boundary detected at t if $|\Delta H_t| > \theta$. The time t at which the boundary is detected marks the transition from the pre-exploitation to the exploitation phase. The pre-exploitation boundary t_b is defined as:

$$t_b = \{t \mid |\Delta H_t| > \theta\} \quad (4)$$

For a given feature set X observed over time, entropy $H(X)$ quantifies the uncertainty or randomness in the data. In the context of malware detection, higher entropy indicates greater unpredictability, which may signal the transition to exploitation behavior.

To dynamically monitor changes, entropy is calculated over a sliding window of size k. By examining the difference in entropy ΔH_t between consecutive windows, significant changes in the data behavior can be identified. A significant change in entropy, exceeding a predefined threshold θ, suggests a transition from pre-exploitation to exploitation. By detecting significant changes in entropy, the pre-exploitation boundary t_b can be dynamically determined. This approach ensures that the detection system adapts to different malware behaviors and accurately identifies the transition point. This system uses entropy to dynamically define the pre-exploitation boundary by monitoring changes in the randomness of the observed features. By calculating entropy over sliding windows and identifying significant changes, the system can adapt to various malware behaviors, ensuring timely and accurate detection of the pre-exploitation phase. This method improves the robustness of the detection mechanism and addresses the limitations of static threshold-based approaches.

In the beginning, a process of splitting the data into moving windows is initiated to create the boundary vector V_b, in accordance with [35]. Then, V_b is employed to segregate the data in each trace file into various distinct sets: $S_1, S_2, \dots, S_n$. The set whose boundary is higher than the threshold is considered a border set, and denoted as S_b. The border set S_b together with all preceding sets $(S_1, S_2, \dots, S_{b-1})$ are then fused into one set called pre-exploitation dataset, from which the features will be extracted.

C. *Improved Data Vectorization for Extracting Pre-Exploitation Features*

The pre-exploitation boundary is the specific point at which Android malware initiates the actual harm by utilizing a series of system calls and libraries. These activities encompass not only the explicit APIs but also other tasks, such as file operations and encryption, that are present within the S_b subset. This particular subset is the specific area where the elements of the pre-exploitation boundary vector can be found. However, not all data in S_b are connected to the malicious intentions of the malware, thus it is not suitable to treat them all equally. Engaging in such behavior may result in their detection early on in the trace file, consequently excluding a significant portion of the pre-exploitation data and depriving the model of crucial attack patterns. The RRF technique is employed to calculate the weight of each feature in the subset and only retain those with weights surpassing a predetermined threshold, thereby eliminating unrelated data. This technique differs from the original RRF, which evaluates the relevance of query terms using:

$$R_f = \frac{1}{n} \sum_r d_j - \frac{1}{N-n} \quad (1)$$

where R_f denotes the feedback vector, N represents the size of the S_b, and n signifies the number of relevant trace files. Additionally, $\sum_r d_j$ is the set of relevant elements, while $\sum_{ir} d_j$ is the set of irrelevant elements.

The RRF technique uses the TF-IDF approach to create two vectors, known as the relevant and irrelevant vectors. The pertinent vector is created by implementing the TF-IDF on the subset S_b. This vector is considered relevant, as it is derived from the data within the pre-exploitation set, encompassing all the data collected from the beginning until the moment the actual sabotage commences. On the other hand, the irrelevant vector is created by applying TF-IDF to both the post-exploitation data $\{ S_p$ such that $S_p = S_t \cap S_b \}$, where S_t represents the entire data collected from the beginning to the end of the attack, and combining the subsets. Following the creation of these vectors, the enhanced Rocchio technique is applied, formulated as:

$$R_f = R_b + \frac{1}{N} \left(\sum_r d_j - \sum_{ir} d_j\right) \quad (5)$$

In this context, R_f represents the feedback vector, and R_b is the original or initial vector. The relevant set $\sum_r d_j$ corresponds to the S_b subset in this particular case, while the irrelevant set $\sum_{ir} d_j j$ corresponds to S_p.

D. *A Weighted Term Frequency-Inverse Document Frequency Technique for Extracting Pre-Exploitation Features*

As previously mentioned, a difficulty arises when using the traditional TF-IDF method to calculate the IDF term for pre-exploitation data. More precisely, a particular characteristic may have a low Document Frequency (DF) value when only considering the pre-exploitation data, but a high DF value when taking into account the entire dataset. In this situation, the TF-IDF value of that particular feature will be increased in the pre-exploitation data but decreased across the entire dataset. Consequently, the feature would be given excessive significance or emphasis solely based on the data before it is exploited, falsely implying that it is a feature specific to attacks. Actually, this feature may have broad application for all data and should thus be given a lower TF-IDF score.

These versatile features provide only a small amount of predictive information and could be used by malicious software to make things more confusing. Therefore, it is appropriate to impose penalties on them, as emphasized in [36]. The TF-IDF technique addresses this issue by utilizing annotations to highlight the APIs that are more relevant to the pre-exploitation phase, while excluding general-purpose features that are less applicable to this particular stage. Equation (6) presents the standard formula for calculating TF-IDF.

$$w\left(api_k^j\right) = tf\left(api_k^j\right) \cdot \log \frac{N}{idf(api_k)} \quad (6)$$

where api_k represents the k^{th} feature and $tf\left(api_k^j\right)$ denotes the term frequency, which indicates how frequently the k^{th} feature is observed by the malware instance r_j within the subset. On the other hand, $idf(api_k)$ represents the calculation of inverse document frequency. It determines the number of occurrences of malware instances r_j within the subset known as the k^{th} feature, at least once. Here, N represents the total number of malware instances within the subset. Before computing the value of $w\left(api_k^j\right)$, each $tf\left(api_k^j\right)$ undergoes normalization to prevent TF-IDF from giving excessive preference to uninformative APIs in lengthy trace files compared to the more insightful ones found in shorter trace files. The normalization process is executed based on (7), where the term frequency of each instance was divided by the length of $tf\left(api_k^j\right)$ (the trace file of malware instance r_j). Therefore, the normalized TF-IDF is calculated according to (8).

$$tf'\left(x_k^j\right) = \frac{tf(x_k^j)}{length(tr_j)} \quad (7)$$

$$w'\left(x_k^j\right) = tf'\left(x_k^j\right) \cdot log \frac{N}{idf(x_k)} \quad (8)$$

In contrast to the traditional TF-IDF approach, WF-IDF distinguishes the features that are involved in the pre-exploitation phase from those that are activated during and after exploitation. WF-IDF can determine if a particular feature is commonly used, even if it has a high document frequency weight in the pre-encryption data. The WF-IDF process begins by assigning tags to each feature in the primary dataset, based on its position in the trace file in relation to the pre-exploitation boundary R_b. Utilizing the boundary vector R_b, every feature is examined in relation to R_b to determine whether it's situated before or after the boundary. Specifically, the features in the original dataset are reviewed sequentially and labeled as b until any R_b entry is encountered. Following the discovery of an R_b entry, the annotation switches to p for all subsequent features.

ALGORITHM 1: WEIGHTED TF-IDF (WF-IDF) TECHNIQUE

```
Input: $TR = \{tr_1, tr_2, \ldots\ldots, tr_m\}$; $TR$ denotes a corpus of
trace files of the of $m$ programs (benign and
malware); $tr_k$ is the trace file of the $k^{th}$ instance
in the corpus $TR$.

Output: $Data_{pre}$ the pre-exploitation dataset.

1: Begin
2: For each $tr_k$ in $TR$:
3:   For each $API$ in $tr_k$:
4:     While $any(V_b)$ is not encountered:
5:       $API_{current} \leftarrow API$
6:       $Annotate(API_{current}, pre)$
7:       $API_{current}++$
8: $F \leftarrow ngram(TR)$
9: For each API-gram($api_k$) in $tr_k$:
10:   $w(api_k) = atf(api_k) \cdot \log \frac{N}{idf(api_k)}$
11:$Data_{pre} \leftarrow aTF-IDF(F)$
12:End
```

As shown in (4), WF-IDF penalizes the common APIs that are called by most or all instances. These APIs are considered general-purpose APIs that do not add information about the target type. Therefore, the denominator of the equation increases according to the number of instances that contain the feature f_k, decreasing the value of $w(f_j^k)$. The pre-encryption WF-IDF weights are stored in a vector that represents the pre-encryption phase of the malware lifecycle. Algorithm 1 shows the pseudocode of the feature extraction using the WF-IDF technique.

IV. EXPERIMENTAL RESULTS

A. Experimental Environment Setup

The experiments were carried out in a controlled environment developed in Cuckoo Sandbox. Both malware and normal applications were included in the analysis. The malware applications were downloaded from virusshare.com, a popular and publicly accessible malware repository. Android malware and benign application files were executed separately. For each program, the data was stored in a trace file unique to that particular instance. Trace files contain the runtime utilization of the program being analyzed, including API, file operations, and network traffic. Following each execution, the guest machine was restored to its initial, unmodified state. The extracted data underwent several preprocessing steps to prepare for modeling. Those steps include noise removal, vectorization, normalization, and imputation. The proposed techniques were developed using a variety of Python libraries, including Sklearn, Pandas, and Numpy.

B. Experimental Results of Dynamic Pre-Exploitation Boundary Definition Technique

To assess the accuracy of the DPED technique in detecting the pre-encryption boundary, an event-based method, proposed in [9], was employed to identify cryptographic activities within the pre-exploitation data. The percentage of malware instances that started cryptography-related tasks before the identified limit was determined. The same method was applied to datasets generated using the time-based thresholding techniques proposed in [9, 17, 21]. To evaluate the efficiency of the DPED method, the findings of [9, 17, 21] were used, as shown in Figure 1. The x-axis depicts the different thresholding techniques, while the y-axis indicates the occurrences that exceeded the limit. Comparative analysis revealed that the DPED technique demonstrated superior accuracy in identifying the pre-exploitation boundaries compared to the other studies. It only failed to detect 8% of malware samples, while the alternative approaches missed 18%, 29%, and 35% of the samples, respectively.

Figure 2 shows a comparative analysis of various studies utilizing logistic regression, including the proposed method and [9, 16, 21]. The proposed method demonstrated the highest accuracy of 0.88, closely followed by [16] with 0.85, while [9] and [21] achieved 0.83. The proposed method achieved the highest F1 score (0.92), while [21] achieved the lowest F1 score (0.87). The proposed method exhibits the highest level of precision (0.9), while the other methods were relatively close but lower. It should be noted that the proposed method and [16] exhibited the highest recall values, 0.94 and 0.93, respectively, indicating a better ability to identify all pertinent instances. The proposed method demonstrated the highest ROC_AUC (0.8), which is a measure of the model's capacity to distinguish between classes. In contrast, the other studies exhibited a lower level of robustness in this regard. In general, the proposed method consistently achieved better results than the others in terms of various metrics, indicating that it may be a more efficient approach for this specific problem. However, the other three studies demonstrated varying levels of effectiveness in different aspects of performance.

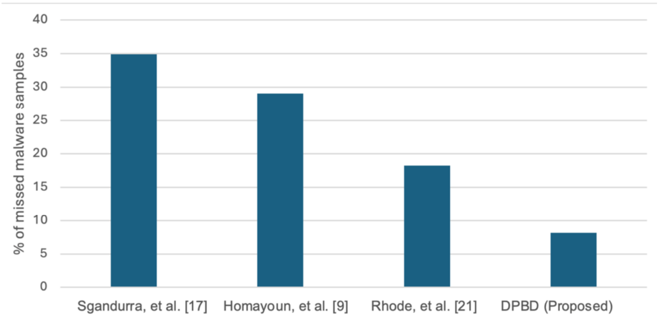

Fig. 1. Comparison of pre-exploitation boundary error between the proposed and related works.

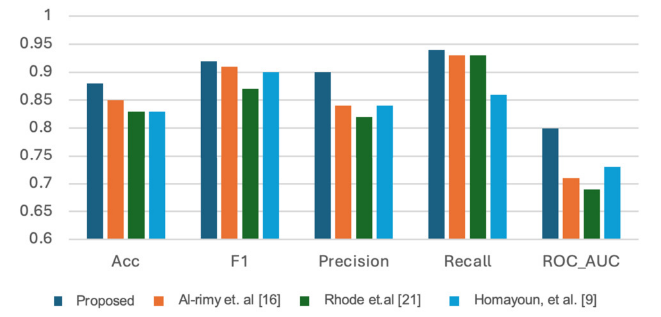

Fig. 2. Classification accuracy comparison between the dataset extracted using proposed DPED and datasets of the related works using LR.

Figure 3 presents a comparison by employing Support Vector Machines (SVM) in the proposed approach and [9, 16, 21]. The proposed method achieved higher accuracy (0.93) and F1 score (0.948). The other studies also showed similar but slightly lower accuracy scores. The proposed method achieved the highest precision of 0.916 and an impressive recall of 0.99, indicating high sensitivity and excellent ability to retrieve all relevant instances. In [16] and [21], significant recall was achieved, although to a lesser extent. The proposed method scored the highest ROC_AUC (0.855), indicating its superior discriminatory ability. In general, the proposed method consistently demonstrated strong and reliable performance, surpassing the other studies in all measurements.

Figure 4 shows a comparison using Deep Belief Networks (DBN). The proposed method demonstrated superior performance in all categories, surpassing [9], [16], and [21]. The F1 score of the proposed method was 0.959, indicating a balanced performance between false positives and negatives, and was closely followed by the other methods. It also demonstrated its strength in accurately identifying positive instances, as evidenced by its highest precision (0.942) and recall (0.974). Finally, the proposed method achieved the highest ROC_AUC score (0.893), indicating its superior ability to effectively distinguish between positive and negative classes. The proposed method consistently outperformed the other studies in all metrics, demonstrating its robustness and efficiency when using DBN. The other studies demonstrate comparable results but do not achieve the same level of effectiveness.

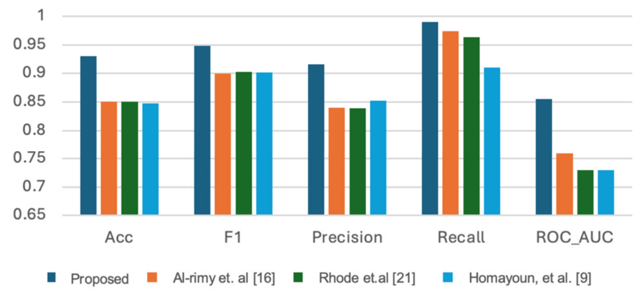

Fig. 3. Classification accuracy between the dataset extracted using the proposed DPED and datasets of the other studies using SVM.

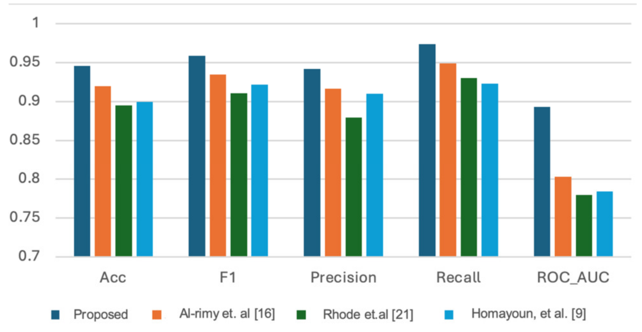

Fig. 4. Classification accuracy comparison in the dataset extracted using the proposed DPED and datasets of related works using DBN.

Figure 5 compares the same models using Convolutional Neural Networks (CNN). The proposed method achieved 0.931 accuracy, surpassing that of [16] (0.901), [21] (0.889), and [9] (0.906). Once again, the proposed method outperformed the others in terms of F1 score, achieving 0.949 and demonstrating a well-balanced performance in both precision and recall. The model in [16] scored 0.917, the one in [21] scored 0.912, and the one in [9] scored 0.91. The proposed method achieved 0.937 precision, while the other methods ranged from 0.874 to 0.893. The proposed method demonstrated exceptional performance in identifying all relevant instances, with a recall value of 0.966. In contrast, the other studies exhibited results within the range of 0.91 to 0.93. It also achieved the highest ROC_AUC score of 0.87, indicating superior discriminative ability between classes. The other methods had ROC_AUC scores ranging from 0.76 to 0.833. Hence, using CNN, the proposed method demonstrated superiority in all evaluated metrics, showcasing its effectiveness and resilience.

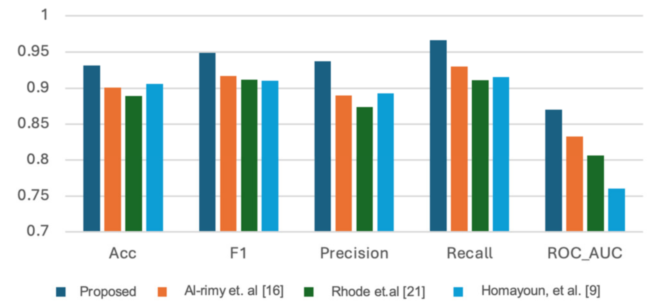

Fig. 5. Classification accuracy comparison between the dataset extracted using the proposed DPED and datasets of the related works using CNN.

Figure 6 compares the performance of various models using a Multilayer Perceptron (MLP). Curiously, the proposed method did not excel in all categories, indicating a more intricate range of performance. It achieved the highest level of accuracy (0.916), followed by [16] (0.895), [21] (0.879), and [9] (0.853). The F1 score provides a more detailed and nuanced evaluation, where the model in [16] achieved the highest score of 0.938, surpassing the proposed method (0.932). The models in [21] and [9] scored 0.9 and 0.884, respectively. Therefore, the model in [16] demonstrated an improved equilibrium between precision and recall. The proposed method demonstrated the highest level of precision, scoring 0.9, closely followed by [16] (0.892). This metric highlights the capacity of the proposed method to accurately detect positive instances. Surprisingly, the method in [16] had the highest recall score (0.986), indicating that it was highly effective in capturing all relevant instances. The proposed method followed, achieving 0.947, while the others fell behind. The ROC_AUC scores for the proposed method (0.842) and [16] (0.837) are closely matched, suggesting that they have a similar ability to distinguish between classes. Although the proposed method demonstrates strong accuracy, precision, and ROC_AUC, the performance analysis reveals a more nuanced landscape, with the method in [16] surpassing it in terms of F1 score and recall. This emphasizes the importance of thoroughly contemplating the particular assessment criteria that are most relevant to a given application or context.

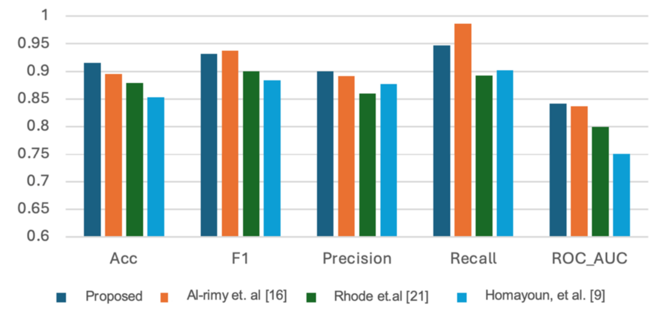

Fig. 6. Classification accuracy comparison in the dataset extracted using the proposed DPED and the datasets of the related works using MLP.

V. CONCLUSION

This study introduced a novel entropy-based approach for dynamically determining the pre-exploitation boundary in ransomware detection on Android devices, addressing the limitations of traditional detection techniques. By incorporating entropy into the Rocchio feedback mechanism, a dynamic feedback process was developed, which adjusts based on the unpredictability of observed features, significantly improving the detection of pre-exploitation activities. The proposed method calculates entropy over sliding windows and monitors significant changes, ensuring timely and accurate detection of ransomware behaviors. A comprehensive evaluation using real-world datasets demonstrated the effectiveness of the proposed approach, showing notable improvements in early detection and reduction of false positives. The integration of entropy provides a robust framework for dynamically refining feedback, enhancing the precision and adaptability of the detection system. This work lays the groundwork for future research and development in malware detection, with implications for protecting against evolving cyber threats. Its contributions highlight the potential for further refinement and application to other types of malware and the integration of machine-learning techniques to enhance the system's accuracy and adaptability. By advancing the field of Android malware detection, the proposed entropy-based approach offers a promising direction for improving cybersecurity in the rapidly evolving landscape of mobile threats.

REFERENCES

[1] N. Ye, Y. Zhang, and C. M. Borror, "Robustness of the Markov-chain model for cyber-attack detection," *IEEE Transactions on Reliability*, vol. 53, no. 1, pp. 116–123, Mar. 2004, https://doi.org/10.1109/TR.2004.823851.

[2] I. Yaqoob *et al.*, "The rise of ransomware and emerging security challenges in the Internet of Things," *Computer Networks*, vol. 129, pp. 444–458, Dec. 2017, https://doi.org/10.1016/j.comnet.2017.09.003.

[3] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du, and G. J. Ahn, "Uncovering the Face of Android Ransomware: Characterization and Real-Time Detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1286–1300, Feb. 2018, https://doi.org/10.1109/TIFS.2017.2787905.

[4] A. Azmoodeh, A. Dehghantanha, M. Conti, and K. K. R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 4, pp. 1141–1152, Aug. 2018, https://doi.org/10.1007/s12652-017-0558-5.

[5] S. Demesie Yalew, G. Q. Maguire, S. Haridi, and M. Correia, "Hail to the Thief: Protecting data from mobile ransomware with ransomsafedroid," in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, Oct. 2017, pp. 1–8, https://doi.org/10.1109/NCA.2017.8171377.

[6] J. A. Gómez-Hernández, L. Álvarez-González, and P. García-Teodoro, "R-Locker: Thwarting ransomware action through a honeyfile-based approach," *Computers & Security*, vol. 73, pp. 389–398, Mar. 2018, https://doi.org/10.1016/j.cose.2017.11.019.

[7] M. A. Azad, F. Riaz, A. Aftab, S. K. J. Rizvi, J. Arshad, and H. F. Atlam, "DEEPSEL: A novel feature selection for early identification of malware in mobile applications," *Future Generation Computer Systems*, vol. 129, pp. 54–63, Apr. 2022, https://doi.org/10.1016/j.future.2021.10.029.

[8] N. Caporusso, S. Chea, and R. Abukhaled, "A Game-Theoretical Model of Ransomware," in *Advances in Human Factors in Cybersecurity*, Orlando, FL, USA, 2019, pp. 69–78, https://doi.org/10.1007/978-3-319-94782-2_7.

[9] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 341–351, Apr. 2020, https://doi.org/10.1109/TETC.2017.2756908.

[10] G. Cusack, O. Michel, and E. Keller, "Machine Learning-Based Detection of Ransomware Using SDN," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, Temple, AZ, USA, Nov. 2018, https://doi.org/10.1145/3180465.3180467.

[11] D. Y. Kao and S.-C. Hsiao, "The dynamic analysis of WannaCry ransomware," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, Chuncheon, Korea (South), Feb. 2018, pp. 159–166, https://doi.org/10.23919/ICACT.2018.8323682.

[12] N. Hampton, Z. Baig, and S. Zeadally, "Ransomware behavioural analysis on windows platforms," *Journal of Information Security and Applications*, vol. 40, pp. 44–51, Jun. 2018, https://doi.org/10.1016/j.jisa.2018.02.008.

[13] A. Cohen and N. Nissim, "Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory," *Expert Systems with Applications*, vol. 102, pp. 158–178, Jul. 2018, https://doi.org/10.1016/j.eswa.2018.02.039.

[14] "Development of new Android malware worldwide from June 2016 to March 2020," *Statista*. https://www.statista.com/statistics/680705/global-android-malware-volume/.

[15] T. Shishkova, "IT threat evolution in Q1 2022. Mobile statistics," May 27, 2022. https://securelist.com/it-threat-evolution-in-q1-2022-mobile-statistics/106589/.

[16] B. A. S. Al-Rimy *et al.*, "A Pseudo Feedback-Based Annotated TF-IDF Technique for Dynamic Crypto-Ransomware Pre-Encryption Boundary Delineation and Features Extraction," *IEEE Access*, vol. 8, pp. 140586–140598, 2020, https://doi.org/10.1109/ACCESS.2020.3012674.

[17] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection." arXiv, Sep. 10, 2016, https://doi.org/10.48550/arXiv.1609.03020.

[18] S. Homayoun *et al.*, "Deep dive into ransomware threat hunting and intelligence at fog layer," *Future Generation Computer Systems*, vol. 90, no. Jan 19, Jul. 2018, https://doi.org/10.1016/j.future.2018.07.045.

[19] X. Zhang *et al.*, "An Early Detection of Android Malware Using System Calls based Machine Learning Model," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, Vienna, Austria, Aug. 2022, https://doi.org/10.1145/3538969.3544413.

[20] M. Alam, S. Sinha, S. Bhattacharya, S. Dutta, D. Mukhopadhyay, and A. Chattopadhyay, "RAPPER: Ransomware Prevention via Performance Counters." arXiv, Apr. 03, 2020, https://doi.org/10.48550/arXiv.2004.01712.

[21] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Computers & Security*, vol. 77, pp. 578–594, Aug. 2018, https://doi.org/10.1016/j.cose.2018.05.010.

[22] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-Based Online Malware Detection: Towards Efficient Real-Time Protection Against Malware," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 289–302, Oct. 2016, https://doi.org/10.1109/TIFS.2015.2491300.

[23] N. Nissim, Y. Lapidot, A. Cohen, and Y. Elovici, "Trusted system-calls analysis methodology aimed at detection of compromised virtual machines using sequential mining," *Knowledge-Based Systems*, vol. 153, pp. 147–175, Aug. 2018, https://doi.org/10.1016/j.knosys.2018.04.033.

[24] D. Morato, E. Berrueta, E. Magaña, and M. Izal, "Ransomware early detection by the analysis of file sharing traffic," *Journal of Network and Computer Applications*, vol. 124, pp. 14–32, Dec. 2018, https://doi.org/10.1016/j.jnca.2018.09.013.

[25] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection," *Future Generation Computer Systems*, vol. 101, pp. 476–491, Dec. 2019, https://doi.org/10.1016/j.future.2019.06.005.

[26] B. A. S. Al-rimy *et al.*, "Redundancy Coefficient Gradual Up-weighting-based Mutual Information Feature Selection technique for Crypto-ransomware early detection," *Future Generation Computer Systems*, vol. 115, pp. 641–658, Feb. 2021, https://doi.org/10.1016/j.future.2020.10.002.

[27] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware," presented at the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 2016, pp. 757–772.

[28] W. Z. A. Zakaria, N. M. K. M. Alta, M. F. Abdollah, O. Abdollah, and S. M. M. Yassin, "Early Detection of Windows Cryptographic Ransomware Based on Pre-Attack API Calls Features and Machine Learning," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 39, no. 2, pp. 110–131, Feb. 2024, https://doi.org/10.37934/araset.39.2.110131.

[29] A. Alqahtani and F. T. Sheldon, "eMIFS: A Normalized Hyperbolic Ransomware Deterrence Model Yielding Greater Accuracy and Overall Performance," *Sensors*, vol. 24, no. 6, Jan. 2024, Art. no. 1728, https://doi.org/10.3390/s24061728.

[30] S. H. Kok, A. Abdullah, N. Z. Jhanjhi, and M. Supramaniam, "Prevention of Crypto-Ransomware Using a Pre-Encryption Detection Algorithm," *Computers*, vol. 8, no. 4, Dec. 2019, Art. no. 79, https://doi.org/10.3390/computers8040079.

[31] M. A. Aftab and D. Q. Shafi, "Advanced Ransomware Detection: Unveiling Anti-Analysis Tactics through Enhanced Temporal Data Correlation." Research Square, Mar. 07, 2024, https://doi.org/10.21203/rs.3.rs-4019125/v1.

[32] N. Niture, "Machine Learning and Cryptographic Algorithms -- Analysis and Design in Ransomware and Vulnerabilities Detection." TechRxiv, Oct. 29, 2020, https://doi.org/10.36227/techrxiv.13146866.v1.

[33] C. C. Moreira, J. Claudomiro de Souza de Sales, and D. C. Moreira, "Understanding Ransomware Actions Through Behavioral Feature Analysis," *Journal of Communication and Information Systems*, vol. 37, no. 1, pp. 61–76, Mar. 2022, https://doi.org/10.14209/jcis.2022.7.

[34] K. Lee, S. Y. Lee, and K. Yim, "Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems," *IEEE Access*, vol. 7, pp. 110205–110215, 2019, https://doi.org/10.1109/ACCESS.2019.2931136.

[35] N. Suditu and F. Fleuret, "Iterative relevance feedback with adaptive exploration/exploitation trade-off," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, Maui, HI, USA, Jul. 2012, pp. 1323–1331, https://doi.org/10.1145/2396761.2398435.

[36] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, "Classification of ransomware families with machine learning based on N-gram of opcodes," *Future Generation Computer Systems*, vol. 90, pp. 211–221, Jan. 2019, https://doi.org/10.1016/j.future.2018.07.052.