Multiple Feature-Based Recurrent Neural Network for Highly Accurate Ransomware Detection in Android Devices

Vyom Kulshreshtha[1,*], Deepak Motwani[2], Pankaj Sharma[3]
[1]PhD Scholar, Computer Science & Engineering, Amity University Madhya PradeshnGwalior, India
[2]Computer Science Engineering, Amity University Madhya Pradesh, Gwalior, India
[3]Computer Science Engineering, Eshan College of Engineering, Mathura, India
Emails: Vyom19@gmail.com; dmotwani@gwa.amity.edu; topankajsharma126@gmail.com

Abstract

Ransomware or crypto-ransomware is a big headache to digital media and transactions nowadays. Generally, Ransomware affects the operating system and transfers the valuable information and data stored in the system. Some ransomware attacks the system and corrupts the system file, making it useless to the user. Data encryption with a private key is also one of the attaching fashions of some types of ransomwares. Most ransomware attacks are reported in android operating system-based devices. The solution to ransomware is only the earlier identification of an attacked pattern in the operating system and removal of it. Artificial Intelligence (AI) plays a major role in various kinds of attack detection and classification processes. Machine learning (ML) technique can be used to train and classify the presence of ransomware in android-based devices. Various parameters, such as the characteristics of applications' permission access to various inputs of the devices. The data can be used to train the Recurrent Neural Network (RNN), the most popular and highly accurate ML module that performs a highly accurate classification process. The performance can be evaluated using various sensitivity evaluation metrics such as accuracy, sensitivity, specificity, and precision.

Keywords: Ransomware; Crypto Ransomware; Android Operating System; Data Encryption; Artificial Intelligence; Machine Learning; Recurrent Neural Network

1. Introduction

The Internet of Things (IoT) devices' use of embedded computer technology has grown exponentially, making system security an essential concern. Malware is a crucial hazard among the various security risks since it is very simple to create, build, and spread within the system(s). Malicious software, also referred to as "malware" is a programme or application created by an attacker to gain unauthorised access to the computing device(s) to carry out malicious activities like data theft, unauthorised access to sensitive information, and unauthorised manipulation of the stored information. Malware poses a significant concern, and in recent years, the number of malware strains has significantly increased. To reduce the malware threat as a result, a thorough and effective malware detection approach is required. [1].

This makes such assaults extremely damaging since they have the potential to wipe both private user and corporate sensitive data (which is frequently left without a backup). Symantec reports that the number of ransomware variants surged by 46% in 2017 because of significant outbreaks like Petya/NotPetya, which affected Ukrainian businesses. It is not unexpected to see that the same pattern also applied to mobile ransomware in 2017, when more than 42 000 copies were stopped. One of the most common types of malwares used in cyber-attacks is known as ransomware, which encrypts data on a victim's computer to make it inaccessible and keeps the data hostage until the victim pays the ransom, usually in the form of money. There are typically two main types of ransomwares: crypto ransomware and locker ransomware. [2].

By secretly searching the victim's computer for documents to encrypt, crypto ransomware gains access to the system. Only after paying a ransom and getting the attackers' decryption keys. Crypto-ransomware primarily targets user-generated files with certain extensions, such as.pdf, .jpg, and.doc files, which typically include important and private user data. Instead of encrypting the entire physical disk, this type of malware instead targets user-generated files. [3].

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025

Despite advancements in software and hardware security, traditional antivirus solutions struggle to keep up with the rapidly evolving ransomware landscape. Unlike other types of malwares, the impact of ransomware is irreversible. Even after neutralizing an attack, encrypted files remain locked until a decryption key is obtained [4].

There are several ways that ransomware may function, from merely locking the desktop to encrypting the entire computer's data. Ransomware has distinct behavioural traits from conventional malware. For instance, classic malware frequently strives for stealth to gather banking login details or keystrokes covertly. The entire purpose of a ransomware assault is to alert the victim that is infected publicly; hence, ransomware behaviour is in stark contrast to stealth. Ransomware attacks can disrupt a distributed infrastructure and prevent efficient communication across different data centres. These systems feature intricate corpus and algorithmic frameworks. These settings, i.e., data centres, house enormous quantities of data and allow ransom payments to prevent reputational risk and data exploitation [5].

Currently, a dangerous communication system that alerts users to the necessary permissions before each programme is loaded is used on Android platforms to protect against malware. Because this approach grants permits on an individual basis, it is only marginally successful. The user needs a lot of analytical expertise to tell malicious software from useful programmes. Since both good and bad applications need the same rights, this permission-based system is unable to discriminate between us. In general, permission-based approaches are created more for risk assessment than for the detection of malware [6].

The primary objectives of this work are stated below:

1. To improve at efficiently detecting ransomware.
2. To improve the performance of accuracy for security.
3. To improve the flexibility by applying all kinds of available ransomware datasets.

The remainder of the document is organised as follows; part II represents pertinent literature. The suggested system's architecture and a thorough description of each of its components were covered in depth in Section III. Results of the experiment are under Section IV. Section V deals with the conclusion at the end.

2. Literature Survey

Numerous studies have been conducted on identifying Android malware. In a variety of research studies, malware has been identified using certain characteristics of Android applications. We only present a few closely related current studies in deep learning-based Android malware.

Vinayakumar et al. proposed hollow and deep networks to detect and classify ransomware. Use the prevalence of application programming interface (API) calls to identify ransomware from benign and other families by defining their characteristics. Conducted numerous network parameter and structural studies to determine the multi-layer perceptron's (MLP) ideal design. With a learning rate between [0.01-0.5], each experiment is conducted for 500 epochs. The outcome from our data set is more encouraging in terms of separating ransomware from innocuous software and its relatives based on the executable file. MLP has the greatest accuracy of 1.0 for categorising malware as either benign or ransomware, and it has the highest accuracy of 0.98 for categorising ransomware. In addition, MLP outperformed the other traditional machine learning classifiers in identifying and categorising malware [7].

Jinsoo et al. proposed a two-stage mixed ransomware detection model, Markov, and Random Forest models. To identify the traits of ransomware, we first concentrate on the Windows API call sequence pattern and construct a Markov model. The remaining data is then used to create a Random Forest machine learning model, which helps to reduce both false positive (FPR) and false negative (FNR) error rates. We can attain a total accuracy of 97.3% using the two-stage mixed detection approach thanks to 4.8% FPR and 1.5% FNR. [8].

Firoz et al. proposed to detect if a program is a ransomware or goodware; DNAact-Ran first selects significant features. DNAact-Ra construction. BCS and MOGWO algorithms. It then creates the digital DNA Sequence for a few traits, and utilising the active learning principle, it categorises the instances as either goodware or ransomware. The DNAact-Ran suggested architectureIn three crucial parts of the procedure, the DNAact-Ran method identifies ransomware. DNA sequence generation, feature selection, and ransomware detection. DNAact-Run was tested on 582 ransomware and 942 goodware cases to assess the performance of precision, recall, f-measure, and accuracy to assess the efficacy of the suggested strategy. The assessment findings demonstrate that DNAact-Run can reliably and effectively anticipate and detect ransomware when compared to other techniques. [9].

Michele et al. proposed learning-based detection strategies that rely on System API information. These methods take use of the fact that ransomware assaults significantly rely on the System API to carry out their operations and make it possible to differentiate between ransomware, generic malware, and good ware. Tested three approaches—using System API data through packages, classes, and methods—and contrasted their effectiveness with other, more intricate state-of-the-art solutions. The findings demonstrated that systems based on System API could identify malware in general and ransomware

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025

with excellent accuracy, on par with systems using data that are more complicated. Additionally, the suggested methods demonstrated resistance to static obfuscation attempts and could precisely recognise unique samples in the environment. Finally, we created and launched R-PackDroid, a full-featured ransomware and malware detector, for the Android platform to ensure early on-device detection.) [10].

Kok et al. proposed a pre-encryption detection algorithm (PEDA) for detecting crypto-ransomware before the occurrence of any encryption. The PEDA has two distinct detection thresholds. A signature repository (SR) is the first, and it shows any signature matches with well-known ransomware. A learning algorithm (LA) is used at the second detection level to find both known and undiscovered crypto-ransomware. LA to train the prediction model using a machine learning method uses data from the application programme interface (API). LA is being assessed using both standard and non-traditional indicators to comprehend PEDA functioning. The true positive rate, accuracy, and precision are common measures that might give useful performance indications, but they are not thorough enough to evaluate the LA capabilities. Six additional measures have been suggested to offer further understanding. According to the findings, LA has succeeded in identifying crypto ransomware before the encryption was practical, and its performance is reliable with a significant net benefit [11].

Khan et al. presented a method based on Moving Target Defence approach. They have used multiple layers of MTD with an objective to alteration of the attack surface resulting in reducing attack success ratio. The layers were used to hide the existing extension of file to prevent attacks by the variants of ransomware [12].

The earlier ransomware detection techniques are insecure and computationally inefficient. New ransomware apps cannot be found using the earlier techniques. To achieve the very minimum level of security needed for ransomware detection, extensive training and validation are needed.

3. Proposed Method

A. Block diagram

Fig.1. shows the block diagram of proposed method. In this paper, we address the malware detection and family classification problem related to the IoT. The data taken from the Ransomeware dataset. The data are converted into matrix. Extracted features are statistical features, arithmetic features, pattern features from the converted matrix data. Then we cascaded all the features. Finally trained and testing using RNN network.

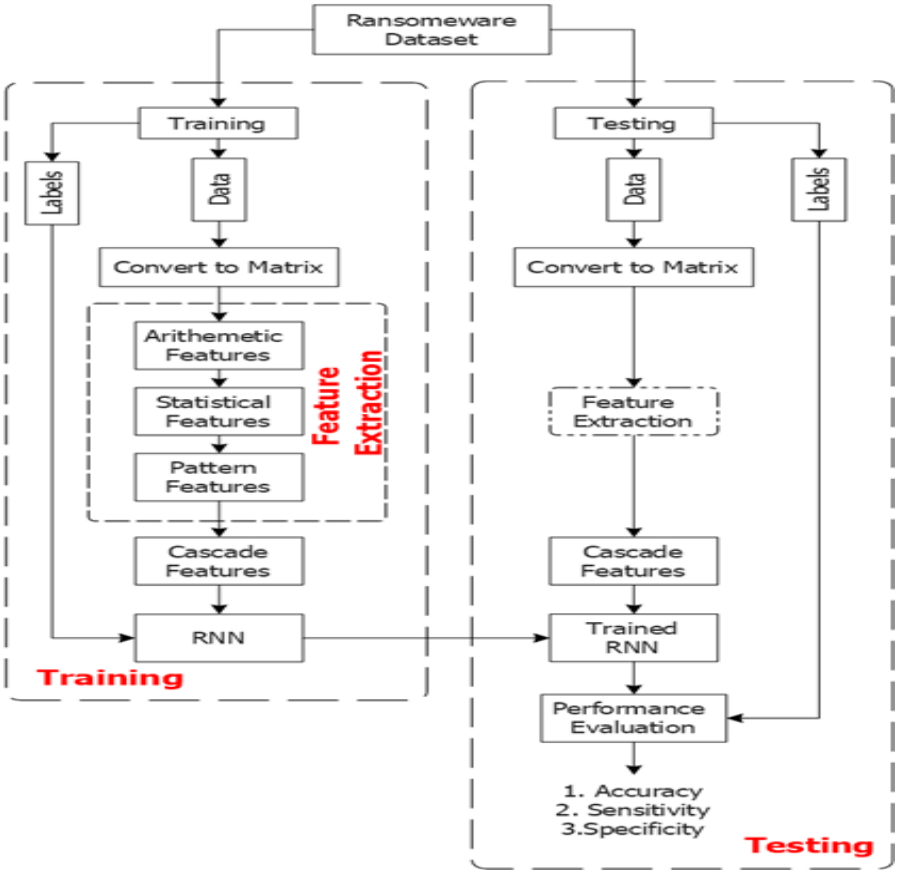

Figure 1. Block diagram of proposed method

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025

B. Pattern Features

Operation Code (Opcode)

One of the most utilised aspects for malware detection is opcode. An Opcode is a single instruction that the processor's (CPU) may carry out and that explains how an executable file behaves. An opcode in assembly language is a command like CALL, ADD, or MOV. Recognize dangerous code using Opcode sequences [13-15].

Strings

An executable file contains strings, which are collections of characters like "gayfgt" that are typically encoded in ASCII or Unicode (1 byte per character) formats. An executable file's printed strings can be used to extract useful data, such as an IP address, a URL to connect to, and other details, to check for maliciousness. Create a signature to categorise IoT malware [16-19].

ELF file header

A file format called ELF (Executable and Linkable format) has a wealth of intriguing data that may be utilised to identify malware. Create a signature to categorise IoT malware [20-21].

C. Statistical Features

Contrast

$$\sum_{i,j=0}^{level-1} P_{i,j}(i-j)^2 \tag{1}$$

Dissimilarity

$$\sum_{i,j=0}^{level-1} P_{i,j}|i-j| \tag{2}$$

Homogeneity

$$\sum_{i,j=0}^{level-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{3}$$

ASM

$$\sum_{i,j=0}^{level-1} {P_{i,j}}^2 \tag{4}$$

Energy

$$\sqrt{ASM} \tag{5}$$

Correlation

$$\sum_{i,j=0}^{level-1} P_{i,j}\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right] \tag{6}$$

Energy

$$EG = \sum_i^{N_G}\sum_j^{N_G}\{p(i,j)\}^2$$

Correlation (CO)

$$CO = \frac{\sum_i^{N_G}\sum_j^{N_G}(i-\mu_x)(j-\mu_y)\ p(i,j)}{\sigma_x\sigma_y} \tag{7}$$

Inverse Difference Moment (IDM)

$$idm = \sum_i^{N_G}\sum_j^{N_G}\frac{1}{1+(i-j)^2}\ p(i,j) \tag{8}$$

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025

D. RNN

Recurrent Neural Networks (RNNs) are a type of neural network that are well suited for sequence-based data such as time-series data or natural language processing. RNNs can be used in ransomware detection by analysing network traffic data and detecting abnormal behaviour that may indicate the presence of ransomware [22]. The RNN model can be trained on a dataset of network traffic to learn the patterns of normal behaviour, and then identify deviations from these patterns that could indicate the presence of ransomware. One approach to using RNNs for ransomware detection is to model network traffic as a time series of packets, with each packet representing a data point in the time series [23]. The RNN model can then be trained to predict the next packet in the time series based on the previous packets. Any deviation from the predicted packet could indicate the presence of ransomware. Ransomware often includes ransom notes or other messages that can be detected through NLP techniques, such as sentiment analysis or named entity recognition.

At each time step t, the RNN takes an input vector x_t and a hidden state vector h_{t-1} as input, and computes a new hidden state h_t and an output vector y_t as follows:

$$x_t = f(W\_\{hh\}h\_\{t-1\} + W\_\{xh\}x_t + b_h) \quad (9)$$

$$y\_t = g(W\_\{hy\}h_t + b_y) \quad (10)$$

Where h_t is the hidden state vector at time step t, which contains information from previous time steps. x_t is the input vector at time step t. W_{hh}, W_{xh}, and W_{hy} are weight matrices that connect the input and hidden layers and the hidden and output layers. b_h and b_y are bias vectors. f and g are activation functions applied element-wise to the vectors.

Overall, RNNs can be a powerful tool for ransomware detection when used in conjunction with other machine learning techniques and domain knowledge of network security. Table. 1. Shows the training parameters of RNN. The RNN network needs Relu activation function, 256x256 input size, L1, L2 dropout of regularization, batch size of 1024, and 50 of epochs.

Table 1: Training parameters of RNN

Parameters	RNN
Activation function	Relu
Input size	256x256
Regularization	L1, L2, dropout
Batch size	1024
Learning rate	1xe-4
Epochs	50
Optimizer	Adam
No. of units	128

Algorithm

1. Training the Proposed Model

When a terminal condition is not resolved, do

2. Examine a mini batch of data that includes both malicious and beneficial programmes.

3. Determine the generative RNN's outputs for malware.

4. Obtain the Gumbel-Softmax output of malicious software as well as the output of the replacement RNN on good programmes.

5. Get the victim RNN's outputs from the malicious examples and good programmes.

6: Reduce LS on both benign and malicious input by altering the weights of the replacement RNN.

7: Reduce LG on malware data by altering the weights of the generative RNN.

8: end though

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025

4. Results and Discussions

Implementation of the suggested approach uses a 16GB RAM and Intel Core i7-8750H processor. Python has been utilised by as a development environment for this work.

A. Dataset

This system was developed using the CICAndMal2017 Android malware dataset. Instead of using an emulator, the data is acquired by running both safe and malicious apps on a real smartphone. Adware, ransomware, scareware, and SMS malware are the four primary kinds of malware samples included in the CICAndMal2017 [24–26]. The total number of records in the dataset is 1509550. There are 460976 hostile ransomware intentions and 1048574 entries for benign ransomware purposes. Total 83 characteristics and a Tag (Attack, Normal) are present in the dataset sample. The total number of features is 84. As far as we are aware, our suggested ransomware detection method is the first to run on the CICAndMal2017 dataset and obtain performance metrics [26].

For training and testing purposes, we divided the data into a 70:30 distribution. 1207640 records, or 70% of the whole dataset, are made available for training. Of the total 301910 records, 30% are used for testing.

B. Performance Matrices

The classifier's accuracy is the gauge of an accurate prediction. It provides the capability of the overall classifier's performance. The definition of accuracy is,

$$Accuracy = \frac{\text{TP+ TN}}{\text{TP+ TN+ FP+FN}} \quad (11)$$

This crucial measure extracts elements from the android devices connected to malware. The gathered information aids in determining whether the information is connected to malware or non-malware traits.

$$Sensitivity = \frac{TP}{TP+FN} \quad (12)$$

From the obtained brain characteristics, this metric is utilised to identify the particular features of malware, which are then calculated as follows:

$$Specificity = \frac{TN}{TN + FP} \quad (13)$$

True Positive (TP): This occurs when the subject programme is recognised by the detection technique as malware, and it actually is malware.

True Negative (TN): This occurs when the subject programme is accurately identified by the detection technique as being ordinary software, not malware. Because the detection result was accurate and consistent with reality, the prior two outcomes are said to as true.

False Positive (FP): When a piece of software is mistakenly identified as malware when it is actually just regular software.

False Negative (FN): When the subject programme is malware, the detection algorithm wrongly classifies it as regular software.

Table 2 shows the performance of proposed method base on training and testing. 70/30 achieves 85.69 of accuracy, 86.96 of sensitivity, 88.96 of Specificity, 80/20 percentage highest performance achieved like 99.54 of accuracy, 99.85 of sensitivity, and 99.64 of specificity.

Table 2: Performance of proposed method base on training and testing

Methods	Accuracy	Sensitivity	Specificity
40/60	56.96	59.78	51.96
50/50	63.48	68.48	69.19
60/40	79.28	76.89	75.98
70/30	85.69	86.96	88.96
80/20	99.54	99.85	99.64

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025

Table 3. shows the comparison with previous classification methods. [8] method achieves 97.3 of accuracy, 95.2 of sensitivity, and 98.5 of specificity. [13] method achieves 95.7 of accuracy, [14] method returns 99.46 of accuracy, [15] method achieves 97.74 of accuracy, 99.85 of sensitivity, 85.7 of specificity. This work achieves 99.54 of accuracy, 99.85 of sensitivity, and 99.64 of specificity.

Table 3: Comparison with previous classification methods

Methods	Accuracy	Sensitivity	Specificity
[8]	97.3	95.2	98.5
[13]	95.7	-	-
[14]	99.46	-	-
[15]	97.74	99.5	85.7
This work	99.54	99.85	99.64

Fig.2 ROC curve of proposed method. It shows the 0.99 of true positive rate. Fig.3. shows the comparative performance of proposed method. This proposed method achieves highest performance of accuray.

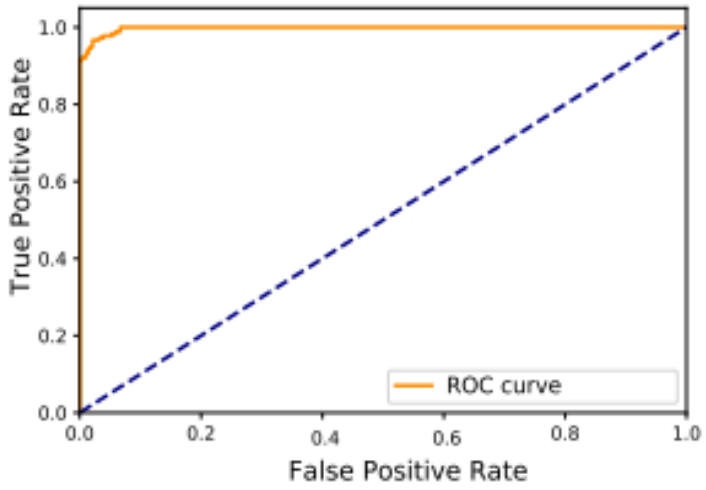

Figure 2. ROC curve of proposed method

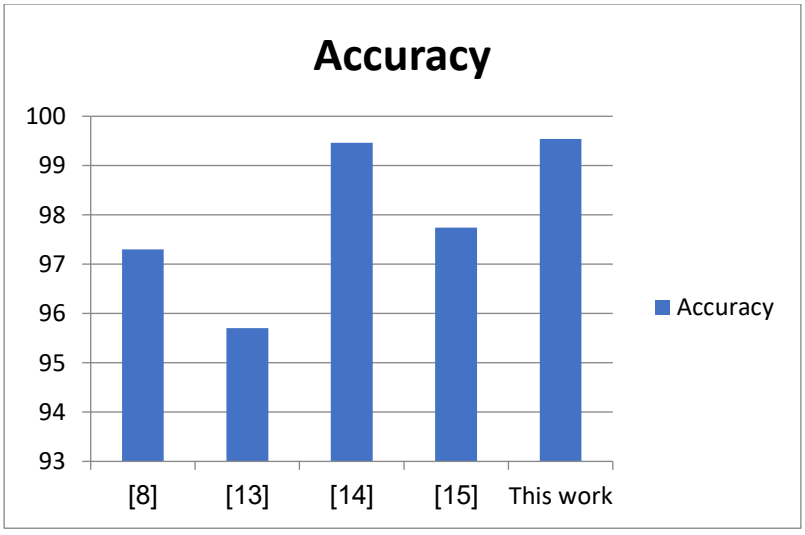

Figure 3. Comparative performance of proposed method

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025

5. Conclusion

In this paper, we offer a hybrid technique for the identification of stealthy malware that makes use of architectural (trace) and code attributes extracted using RNN and localised features extraction, respectively. The most effective RNN for malware detection is selected, and it is fed to an ML classifier in the RNN-based technique. Further sequence tagging was done to identify these unique patterns. The suggested technique successfully solves every sequence classification criterion, as mentioned in the study, which makes sequence classification a challenging endeavour. We use an RNN to extract and process the localised characteristics for sequence classification in order to achieve the greatest average accuracy of 99.54% when compared to stealthy malware. As a result, we draw the conclusion that the suggested technique is reliable and quick in finding both classic malware and stealthy malware.

Reference

[1] S. Alsoghyer and I. Almomani, "On the effectiveness of application permissions for Android ransomware detection," in *2020 6th Conference on Data Science and Machine Learning Applications (CDMA)*, 2020, pp. 94-99. IEEE.

[2] R. Agrawal, J. W. Stokes, K. Selvaraj, and M. Marinescu, "Attention in recurrent neural networks for ransomware detection," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3222-3226. IEEE.

[3] M. S. Kumar, J. Ben-Othman, and K. G. Srinivasagan, "An investigation on wannacry ransomware and its detection," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 1-6. IEEE.

[4] M. Musonda, A. Zimba, and M. Sinyinda, "Machine learning-based crypto ransomware detection model on Windows platforms," in *Proceedings of International Conference for ICT (ICICT)-Zambia*, vol. 5, no. 1, 2023, pp. 141-147.

[5] O. M. K. Alhawi, J. Baldwin, and A. Dehghantanha, "Leveraging machine learning techniques for Windows ransomware network traffic detection," in *Cyber Threat Intelligence*, Cham, Switzerland: Springer, 2018, pp. 93-106.

[6] S. R. B. Alvee et al., "Ransomware attack modeling and artificial intelligence-based ransomware detection for digital substations," in *2021 6th IEEE Workshop on the Electronic Grid (eGRID)*, 2021, pp. 01-05. IEEE.

[7] R. Vinayakumar, K. P. Soman, K. K. S. Velan, and S. Ganorkar, "Evaluating shallow and deep networks for ransomware detection and classification," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 259-265. IEEE.

[8] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-stage ransomware detection using dynamic analysis and machine learning techniques," *Wireless Personal Communications*, vol. 112, no. 4, pp. 2597-2609, 2020.

[9] F. Khan, C. Ncube, L. K. Ramasamy, S. Kadry, and Y. Nam, "A digital DNA sequencing engine for ransomware detection using machine learning," *IEEE Access*, vol. 8, pp. 119710-119719, 2020.

[10] M. Scalas, D. Maiorca, F. Mercaldo, C. A. Visaggio, F. Martinelli, and G. Giacinto, "On the effectiveness of system API-related information for Android ransomware detection," *Computers & Security*, vol. 86, pp. 168-182, 2019.

[11] S. H. Kok, A. Azween, and N. Z. Jhanjhi, "Evaluation metric for crypto-ransomware detection using machine learning," *Journal of Information Security and Applications*, vol. 55, p. 102646, 2020.

[12] M. M. Khan, M. F. Hyder, S. M. Khan, J. Arshad, and M. M. Khan, "Ransomware prevention using moving target defense-based approach," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 7, p. e7592, 2023.

[13] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "EfficientNet convolutional neural networks-based Android malware detection," *Computers & Security*, vol. 115, p. 102622, 2022.

[14] C. Li, K. Mills, D. Niu, R. Zhu, H. Zhang, and H. Kinawi, "Android malware detection based on factorization machine," *IEEE Access*, vol. 7, pp. 184008-184019, 2019.

[15] B. M. Khammas, "Ransomware detection using random forest technique," *ICT Express*, vol. 6, no. 4, pp. 325-331, 2020.

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025

[16] K. A. Alissa, D. H. Elkamchouchi, K. Tarmissi, A. Yafoz, R. Alsini, O. Alghushairy, A. Mohamed, and M. A. Duhayyim, "Dwarf mongoose optimization with machine-learning-driven ransomware detection in internet of things environment," *Applied Sciences*, vol. 12, no. 19, p. 9513, 2022.

[17] S. R. B. Alvee, B. Ahn, T. Kim, Y. Su, Y.-W. Youn, and M.-H. Ryu, "Ransomware attack modeling and artificial intelligence-based ransomware detection for digital substations," in *2021 6th IEEE Workshop on the Electronic Grid (eGRID)*, 2021, pp. 01-05. IEEE.

[18] Z. Abdullah, F. W. Muhadi, M. M. Saudi, I. R. A. Hamid, and C. F. M. Foozy, "Android ransomware detection based on dynamic obtained features," in *Recent Advances on Soft Computing and Data Mining: Proceedings of the Fourth International Conference on Soft Computing and Data Mining (SCDM 2020)*, Melaka, Malaysia, Jan. 22–23, 2020, pp. 121-129. Springer.

[19] F. Martinelli, F. Mercaldo, and A. Saracino, "Bridemaid: An hybrid tool for accurate detection of Android malware," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 899-901.

[20] I. Bibi, A. Akhunzada, J. Malik, G. Ahmed, and M. Raza, "An effective Android ransomware detection through multi-factor feature filtration and recurrent neural network," in *2019 UK/China Emerging Technologies (UCET)*, 2019, pp. 1-4. IEEE.

[21] S. Poudyal, D. Dasgupta, Z. Akhtar, and K. Gupta, "A multi-level ransomware detection framework using natural language processing and machine learning," in *14th International Conference on Malicious and Unwanted Software (MALCON)*, 2019.

[22] R. Almohaini, I. Almomani, and A. AlKhayer, "Hybrid-based analysis impact on ransomware detection for Android systems," *Applied Sciences*, vol. 11, no. 22, p. 10976, 2021.

[23] I. Baptista, S. Shiaeles, and N. Kolokotronis, "A novel malware detection system based on machine learning and binary visualization," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2019, pp. 1-6. IEEE.

[24] T. Lu, L. Zhang, S. Wang, and Q. Gong, "Ransomware detection based on V-detector negative selection algorithm," in *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, 2017, pp. 531-536. IEEE.

[25] Q. M. Yaseen, "The effect of the ransomware dataset age on the detection accuracy of machine learning models," *Information*, vol. 14, no. 3, p. 193, 2023.

[26] D. Su, J. Liu, X. Wang, and W. Wang, "Detecting Android locker-ransomware on Chinese social networks," *IEEE Access*, vol. 7, pp. 20381-20393, 2018.

DOI: https://doi.org/10.54216/FPA.190107
Received: October 14, 2024 Revised: January 03, 2025 Accepted: January 27, 2025