Android Ransomware Recognition with Feature Extraction for Improved Security System Detection

Roberto Vazquez-Ferrer, Doctoral Student
Polytechnic University of Puerto Rico
rjvf.007@gmail.com
Fabian Garcia-Romero, Student
Polytechnic University of Puerto Rico
fabiangarcia4409@gmail.com
Jonathan R. Sosa-Lugo, Student
Polytechnic University of Puerto Rico
sosa_139096@students.pupr.edu
Eduard Ramos-Flores, Master Student
Polytechnic University of Puerto Rico
ramos_43563@students.pupr.edu
Diamaris Gonzalez-Rodriguez, Student
Polytechnic University of Puerto Rico
gonzalez_121589@students.pupr.edu;
Alfredo Cruz. Ph.D.
Polytechnic University of Puerto Rico
alcruz@pupr.edu
Jeffrey L. Duffany. Ph.D.
Ana G. Mendez University
jeduffany@suagm.edu

Abstract

Ransomware is a notorious malware targeting end-users, governments, and business organizations. It has become a lucrative business for cybercriminals with revenues of millions of dollars and a severe threat to organizations with financial losses of billions of dollars. This article proposes using feature extraction for improved ransomware detection. This approach involves analyzing the behavior of an Android application and extracting specific characteristics or features indicative of malicious behavior using Deep Learning algorithms. For that reason, the researcher can accurately recognize the existence of ransomware on an Android device or Cloud system when the concept is applied. In addition, the decision tree was used to identify exotic patterns successfully. Thus, feature extraction increases the accuracy of ransomware detection systems, making it a promising solution for improving the security of Android devices against these threats. The methods proved to increase the accuracy of machine learning were Support Vector Machine (SVM), KNN, and Random Forest (RF). Also, it provided a classification between benign and malicious with 100% accuracy of machine learning. As well as successfully classifying Android Malware like Adware, Ransomware, SMS, and Scareware with 100% accuracy of machine learning.

Keywords: Deep Learning, Malware, Ransomware, Android, Malware Detection, Machine Learning, Ransomware Detection, Feature Extraction, Android Ransomware.

Introduction

Ransomware is already prevalent in networks and PCs but has recently become common in mobile devices and the Internet of Things (IoT). The increasing popularity of the Android operating system has led to a rise in malware attacks, including ransomware, which encrypts a victim's files and demands payment for their release. Advanced techniques for extracting features from Android malware are needed to detect and prevent such attacks effectively, which can be used to develop more robust systems for malware detection and prevention. Therefore, the problem statement is: How can we develop advanced techniques for feature extraction from Android malware to enhance the detection and prevention of ransomware attacks? In addition, there is no characteristic identification to distinguish each ransomware family utilizing Deep Learning in the Android smartphone application. Moreover, Cybercriminals continuously experiment with unique methods that propagate ransomware, consisting of social engineering assaults by phishing attempts (Beaman et al., 2021).

The researcher intends to explore deep learning to identify characteristics using Deep-Learning of Android malware in image format. Also, plan to use Keras Deep Learning and determine which Keras kernel provides high accuracy. Deep learning has serval applications in every domain due to the ability for decision-making and expansion in cybersecurity. Advanced attacks and threat detection became in less time due to these potential technologies (Alqahtani et al., 2019; Baek et al., 2021; Fernando et al., 2020; Urooj et al., 2022).

Definitions

Feature extraction (FE)- derives information from the feature set to build a new feature subspace. (Raschka & Mirjalili, 2019). Figure 1 shows the concept idea of feature extraction. In this case, the example is a house. When the house passes through the feature extraction algorithm, it produces an outcome consisting of vectors that separate or break knowledge pieces from the house.

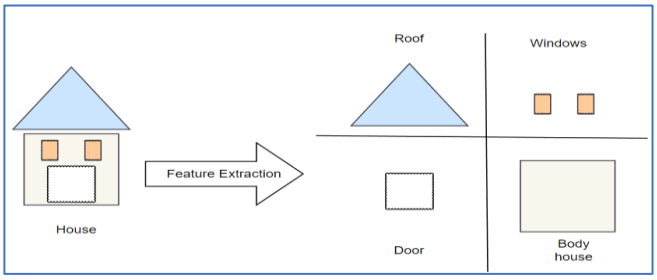


Figure 1: Feature Extraction Concept

Principal Component Analysis (PCA) - helps reduce the dimensionality of extensive datasets vectors and transforms them into orthogonal projection features in lower-dimensional space that contains the most relevant information from the raw datasets (Raschka & Mirjalili, 2019).

Confusion Matrix (CM) - helps us visualize the classifier's performance getting confused in discriminating between classes (Raschka & Mirjalili, 2019).

Recall parameter attempts to answer. What portion of actual positives identifies correctly? (Raschka & Mirjalili, 2019).

Precision parameter attempt to answer What proportion of identifications was correct? (Raschka & Mirjalili, 2019).

DenseNet-201 - is a convolutional neural network that is 201 layers deep. The pre-trained network can categorize images into object classifications. Consequently, the network has learned valuable feature representations for various images. The network also has an image input size of 224-by-224 (Huang et al., 2016).

Related Works

Daku et al. (2018) propose a behavioral-based classification model that extracts behavioral features from ransomware samples and uses machine learning algorithms to classify the samples into different families. The paper describes the methodology used to collect and preprocess the ransomware data and the feature extraction and selection process. The authors evaluate the performance of several machine learning algorithms, including decision trees, support vector machines, and random forests, on classifying 150 ransomware samples of 10 different family types. Also, the results show that the proposed model achieved high accuracy in classifying ransomware samples into their respective families. Finally, it demonstrates the effectiveness of using behavioral-based features for ransomware classification (Daku et al., 2018).

Cusack et al. (2018) developed the process for obtaining rich flow records, extracting flow features, and categorizing ransomware using RaftLib's high-performance and parallel architecture in conjunction with novel hardware-based flow generators. First, the research only uses the unencrypted components of HTTPS traffic for model construction since malware communication is shifting to HTTPS for delivery and control. Next, it creates a stream processor using five kernels to process rich flow records and extract high-level flow features for our random forest classifier. In addition, the research considerably reduced the number of Features in our initial feature set while still achieving a detection accuracy rate of over 87% and a high false negative rate of less than 10% (when they watch the communication between the infected computer and the C&C server)(Cusack et al., 2018).

A method for detecting malicious payloads in various file types was proposed by Baptista et al. (2019). The experimental results showed 91.7% and 94.1% detection accuracy achieved for ransomware in .pdf and .doc files, respectively. This proposal refers to a binary visualization for malicious payload analysis and detection. The malicious payload, the email attached, is the component of the attack that causes harm to the victim. With this information, we fear a different kind of vision on how to prevent or combat attacks. These files are the most used in our daily lives. This algorithm achieves an overall average detection rate of about 74%, with 12% false positives and 14% false negatives (Baptista et al., 2019).

Alqahtani et al. (2019), using machine learning, developed detection techniques for Android operating systems. Android OS has 69% of infections that have been a target of this system. When the user downloads an infected game, the malicious system could send text messages and steal money from the user's bank account. That is why spotting Malware in Google Play Store has become a common way to target many users.

Moreover, the researchers concluded that most surveyed papers used SVM and NB algorithms to detect malware. However, the highest accuracy rate was 100%, with a 0.00% false positive obtained by OneR and J48 algorithms, with their detection rate being 83% and 90%, respectively. In a critical area such as malware detection, we need an excellent Machine Algorithm to detect malware with high accuracy and detection rate (Alqahtani et al., 2019).

Baek et al. (2021) proposed using a two-hybrid malware detection state to protect IoT devices that define a mixed malware state. It combines several different attacks combined into one binary file. This method also aims to increase malware detection accuracy by reducing the false detection rate by 87% of accuracy. (Baek et al., 2021).

Methodology

The following methodology is suitable for our experiment.

A. Data Preprocessing:
    1. Convert the Android binary source code into grayscale images to make footprints of the malware family.
    2. Split the dataset into training, validation, and testing sets by using decision trees to identify exotic patterns in the training dataset that the deep learning model may not capture.
    3. Normalize the pixel values to make them fall within the range of 0 to 1.

B. Model Designing, Implementation, and Optimization:
    1. Use the Keras library for deep learning to design and implement a model.
    2. Train the model using the training dataset and evaluate it using the validation dataset without exotic patterns.

3. Train the model using the training dataset and evaluate it using the validation dataset with exotic patterns.
4. Select the best-performing model based on the validation accuracy and F1 score.

C. Deep-Feature Extraction:
1. Bypassing the SoftMax from Deep Learning selected
2. Evaluate the Deep-Feature Extraction model on the PCA analysis to see if it improves the classification.

D. Testing and Result Analysis on Machine Learning:
1. Test the final model on the testing dataset with Deep-Feature Extraction and evaluate its performance with machine learning like SVM, KNN, and random forest (RF).
2. Analyze the results and compare them with existing approaches to evaluate the effectiveness of the proposed approach by at least 95%.

Figure 2 shows Model 1, proposed by Cusark et al. (2018), which used the PCAP file to detect malware by using feature reduction/selection and then machine learning with the result of 84% (Cusack et al., 2018). Also, Model 2, proposed by Baek et al. (2021), used binary file passing to feature extraction using a cuckoo sandbox and then gave thorough feature selection and machine learning for classification with a result of 94% (Baek et al., 2021). Finally, the goal proposed by the research consists of Android binary datasets passed through image conversion. Deep-Learning to feature extraction and Machine Learning for classification with a result greater than 94%. The Android dataset used is from CICAndMal2017 and CICAndMal2019 (Lashkari et al., 2018; Taheri et al., 2019)

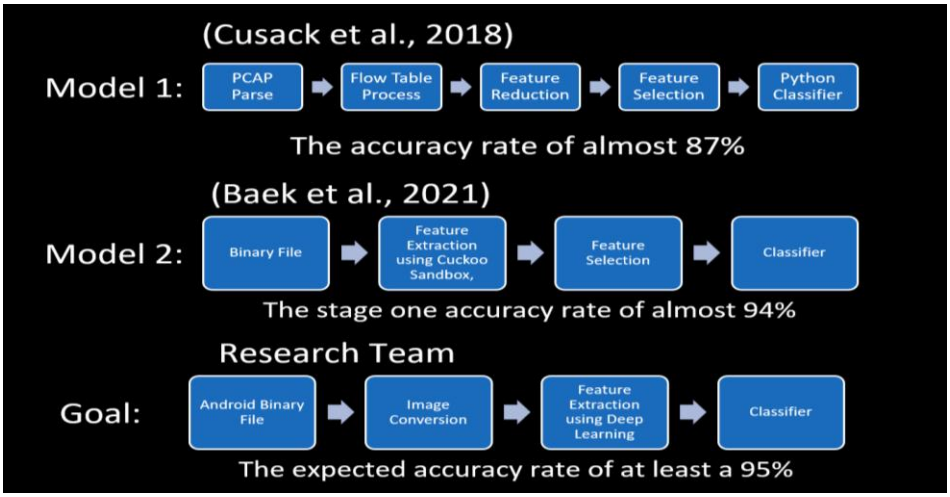


Figure 2: Experiment Strategy

Figure 3 shows the intention experiment level of Part 1, that classifier between Benign and Malware. As well, Part 2 that classifier between Scareware, SMS, Adware, and Ransomware.

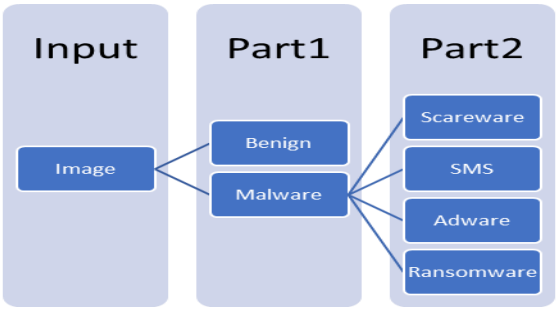


Figure 3: The System in Part Level Intention

Results

Experiment A: Data Preprocessing

Data Preprocessing Phase 1: Convert the Android binary source code into grayscale images to make footprints of the malware family. Figure 4 shows the source code to convert binary to image based on width, height, and location. The research team used the width 128 and height 128 values.

```
import numpy as np
from PIL import Image

def binaryToImage(filename, width, height, imageOutFolder):
    # Read file using numpy "fromfile()"
    with open(filename, mode='rb') as f:
        binaryTransform = np.fromfile(f,dtype=np.uint8,count=width*height).reshape(height,width)
    # Make into PIL Image and save
    PILimage = Image.fromarray(binaryTransform)
    PILimage.save(imageOutFolder)
    print('The file', imageOutFolder, 'saved.')
```

Figure 4: Binary to Image in Python

Figure 5 shows an example of binary-to-image transformation. Figure 5 (a) consists of an image of benign, and Figure 5 (b) consists of a ransomware image.

(a) (b)

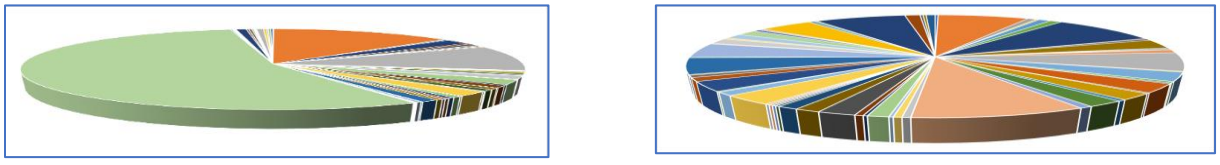
Figure 5: Binary to image example

Data Preprocessing Phase 2: Split the dataset into training, validation, and testing sets using decision trees to identify exotic patterns in the training dataset that the deep learning model may not capture.

The experiment consists of applying a decision tree to Android datasets. The intention is to identify the unique patterns in Android datasets. First, the dataset consists of two parts. The first part consists of Androids datasets between Benign and Malware, and the second part consists of the Android Malware dataset that includes Scareware, SMS, Adware, and Ransomware. The idea is to perform a decision tree from the Sklearn library from Python by using the classification from the datasets. When the decision tree finishes fitting the entire dataset, the following process consists of reorganizing the dataset via leaf position. Finally, the output consists of when the leaf contains a dataset size of more than four, selecting 80% of datasets for specific-leaf randomly.

The result shows in Figure 6 that the datasets contain exotic patterns by leaf tree, and the decision tree provided excellent work reorganizing the datasets. Figure 6 (a) represents Part 1 of the Benign and Malware datasets by leaf, while the Benign represents the green with more than 50% dataset. Figure 6 (b) illustrates Part 2, that the dataset-leaf ranks between Scareware, SMS, Adware, and Ransomware that do well distributed. In other words, the decision tree is crucial to determine trained datasets instead of using 80% of training concepts.

Part 1 Decision Tree Leaf **Part 2 Decision Tree Leaf**



(a) (b)

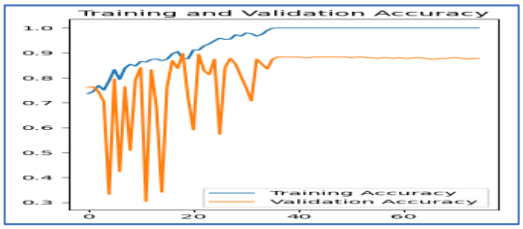
Figure 6: Decision Tree by Leaf

Summary: Part 1 and Part 2 contains exotic patterns that must be part of the training and validation datasets. The research team used 80% of each dataset leaf for balancing the dataset chosen.

Experiment B: Model Designing, Implementation, and Optimization. First, use the Keras library for deep learning to design and implement a model. Second, train the model using the training dataset and evaluate it using the validation dataset without/with exotic patterns. Finally, select the best-performing model based on the validation accuracy and F1 score.

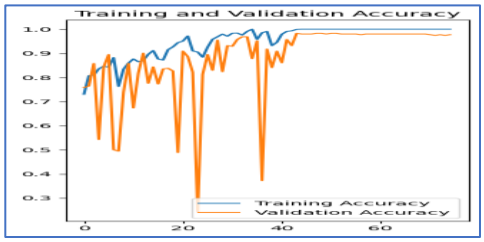
Part 1 shows the Deep Learning in Figure 7, which contains the training and validation accuracy result for Benign and Malware. Figure 7 (a) represents Deep Feature Extraction using raw datasets without exotic pattern recognition with an average of 88% of average validation accuracy. Figure 7 (b) illustrates Deep Feature Extraction using raw datasets with essential pattern recognition with greater than 97% validation accuracy.

Before



(a)

After

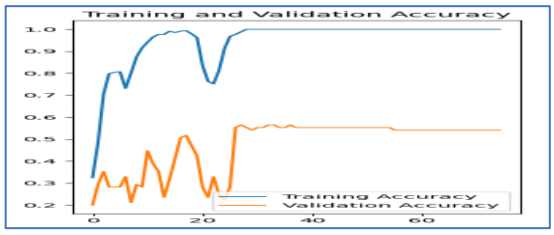


(b)

Figure 7: Deep Learning Training and Validation Accuracy for Benign and Malware in CUDA Core

Part 2 shows the Deep Learning in Figure 8, which contains the training and validation accuracy result for Scareware, SMS, Adware, and Ransomware. Figure 8 (a) represents Deep Feature Extraction using raw datasets without exotic pattern recognition with an average of 55% of average validation accuracy. Figure 8 (b) illustrates Deep Feature Extraction using raw datasets with essential pattern recognition with greater than 85 % validation accuracy.

Before



(a)

After

(b)

Figure 8: Deep Learning Training and Validation Accuracy for Scareware, SMS, Adware, and Ransomware in CUDA core

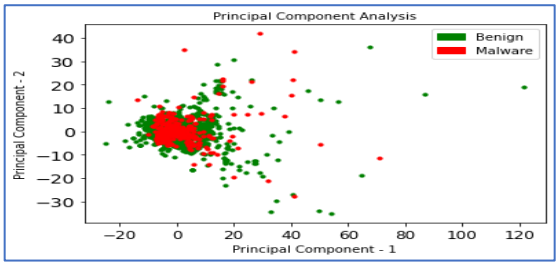
Experiment C: Deep Feature Extraction. The intention consists of bypassing the SoftMax from Deep Learning DenseNet201. Second, evaluate the Deep-Feature Extraction model on the PCA analysis to see if it improves the classification by observing visual clustering classification.

Figure 9 shows how to convert Deep Learning into Deep Feature Extraction. Each Deep learning has a SoftMax that classifies the label. In this case, when bypassing the SoftMax layer, the outcome consists of an output feature. Before bypassing the SoftMax layer, the research uses the dataset and trains them with Deep Learning. After completing the training, bypass the SoftMax layer to convert the system into Deep-Feature Extraction.

Figure 9: Deep Feature Extraction

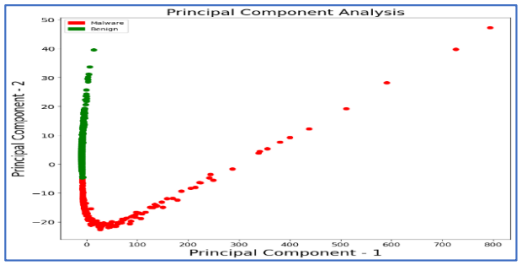
Part 1 results in Figure 10 show the principal component analysis for classification between Benign and Malware from DenseNet201 Deep-Feature extraction. Figure 10 (a) also represents the PCA from the raw datasets without the exotic pattern feature, which produces overlapping categories. Figure 10 (b) illustrates the PCA from the DenseNet201 Deep Feature extraction with the crucial pattern considered and the outcome with a beautiful cluster.

Before



(a)

After

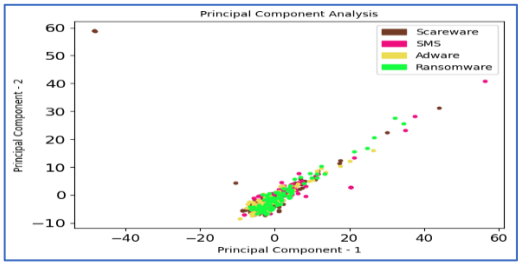


(b)

Figure 10: PCA for classification between Benign and Malware

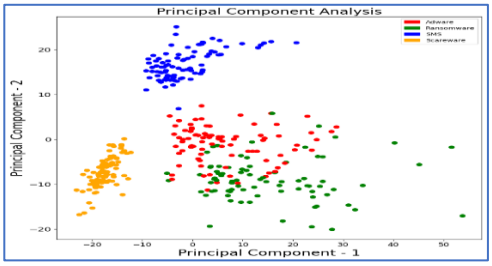
Part 2 results in Figure 11 show the principal component analysis for classifying Malware families. Figure 11 (a) represents the PCA from the raw datasets without the exotic pattern feature with overlapping categories. Figure 11 (b) illustrates the PCA from the DenseNet201 Deep Feature extraction with a crucial pattern considered and a beautiful outcome cluster.

Before



(a)

After



(b)

Figure 11: PCA for classification of Malware Family

Experiment D: Testing and Result Analysis on Machine Learning. First, test the final model on the testing dataset with Deep-Feature Extraction and evaluate its performance with machine learning like SVM, KNN, and random forest (RF). Finally, analyze the results and compare them with existing approaches to evaluate the effectiveness of the proposed approach by at least 95%. Figure 12 shows the intention to use the Deep Feature extraction outcome in machine learning like SVM, KNN, and RF.

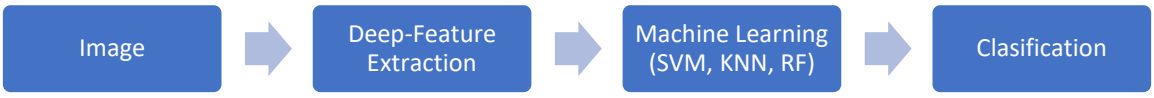
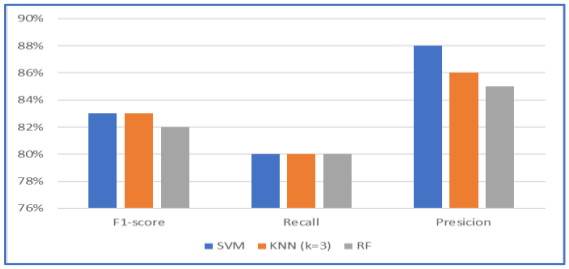


Figure 12: Model Propose

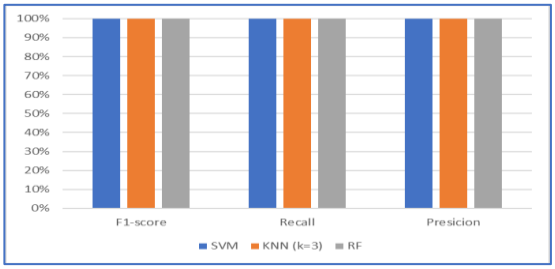
Part 1 score results shown in Figure 13 represent the score analysis between Benign and Malware machine learning using SVM, KNN, and Random Forest. Figure 13 (a) illustrates machine learning using only the raw datasets without exotic pattern features, and the score could be better. Moreover, after creating feature extraction with Deep Learning with essential patterns and using them in machine learning, the accuracy increments are superior in Figure 13 (b).

Before



(a)

After

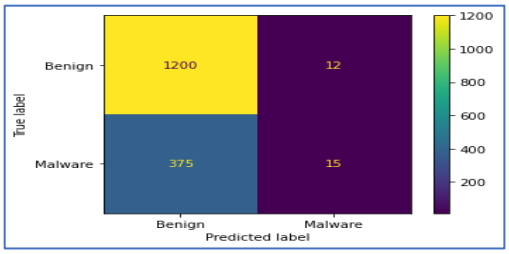


(b)

Figure 13: Score Analysis between Benign and Malware

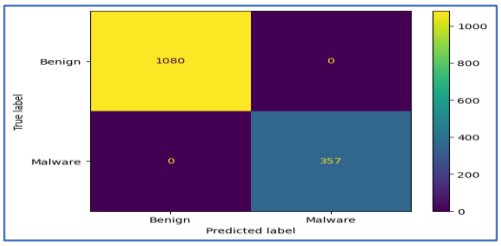
Part 1 confusion matrix (CM) result shown in Figure 14 represents CM analysis between Benign and Malware machine learning that produces the high score. Figure 14 (a) illustrates machine learning using only the raw datasets without exotic pattern features, and the confusion matrix reflects overlapping and is not excellent. Moreover, after creating feature extraction with Deep Learning with essential patterns and using them in machine learning, the confusion matrix is a beautiful heat map in Figure 14 (b).

Before



(a)

After

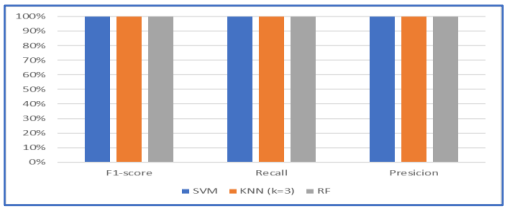


(b)

Figure 14: Confusion Matrix Analysis between Benign and Malware

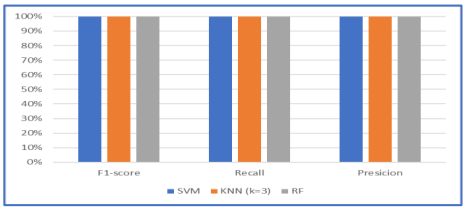
The Part 2 score results shown in Figure 15 represent the score analysis between Benign and Malware machine learning SVM, KNN, and Random Forest. Figure 15 (a) illustrates machine learning using only the raw datasets without exotic pattern features. Moreover, after creating feature extraction with Deep Learning with essential patterns and using them in machine learning, the accuracy increments are superior in Figure 15 (b).

Before



(a)

After

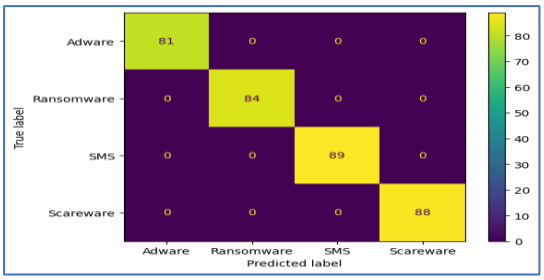


(b)

Figure 15: Score Analysis Malware

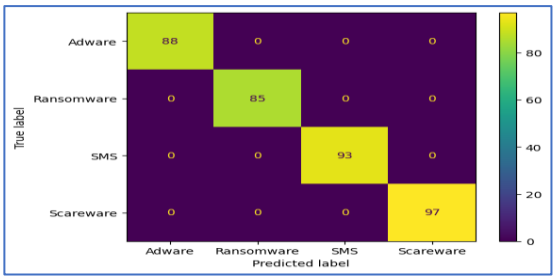
Part 2 confusion matrix (CM) result shown in Figure 16 represents the confusion matrix analysis of malware with machine learning that produces the high score. Figure 16 (a) illustrates machine learning using only the raw datasets without exotic pattern features, and the confusion matrix reflects no overlapping. Moreover, after creating feature extraction with Deep Learning with essential patterns and using them in machine learning, the confusion matrix is a beautiful heat map in Figure 16 (b).

Before



(a)

After



(b)

Figure 16: Confusion Matrix Analysis Malware

Conclusion

An innovative approach to classify Android malware, including ransomware, using feature extraction and machine learning is proposed in this paper. Also, the combination of Decision Tree and Deep Learning identifies important features for constructing Machine Learning like SVM, KNN, and RL. The crucial pattern in the dataset used for training and validation is determined by applying a Decision Tree. Deep feature extraction from the raw dataset uses a Convolutional Neural Network (CNN). A significant improvement in validation accuracy compared to existing methods is shown by our results. This paper provides a robust and efficient way for feature extraction and classification in Android malware detection.

In the case of benign and malicious classification, the Deep Feature Extraction provided 97% accuracy when using them for machine learning like SVM, KNN, and RL, providing a 100% accuracy, precision, and precision f1-score. Our system provided a superior percentage by comparing the result with Model 1 and Model 2 discussed previously, which produced 87%, and 94% accuracy.

For malware classification like Adware, Ransomware, SMS, and Scareware, the Deep Feature Extraction provides 87% accuracy when using them for machine learning like SVM, KNN, and RL a 100% accuracy, precision, and precision f1-score. Also, the PCA looks like a beautiful cluster between classifications. Our system provided a superior percentage by comparing the result with Model 1 and Model 2 discussed previously, which produced 87%, and 94% accuracy.

Finally, the decision tree is essential in identifying exotic patterns in the training dataset. Interestingly, other research does not use the decision tree based on leaf classification to separate the similar dataset by using the leaf location of the dataset.

Future Work

The research team aims to integrate our Android Ransomware identification feature extraction technology with Android cloud security systems and the Apple environment. We can recognize ransomware earlier by introducing feature extraction into cloud-based solutions for real-time application behavior analysis. Our model was tested using grayscale images from the Android database. Future work will include expanding testing with additional datasets and implementing our model in a cloud computing environment. It also intends to incorporate our feature extraction and ransomware recognition model with other security measures, such as firewalls, intrusion detection systems, and encryption techniques, to provide a comprehensive security solution.

Acknowledgment

The work supported is based upon this material by, or in part by, the National Centers of Academic Excellence in Cybersecurity (NCAE-C) under contract/award H98230-20-1-0411 and NSF CyberCorps(R) Scholarship for Service (SFS) 214638.

References

Alqahtani, E. J., Zagrouba, R., & Almuhaideb, A. (2019). *A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms. 2019 6th International Conference on Software Defined Systems, SDS 2019*, 110–117. https://doi.org/10.1109/SDS.2019.8768729

Baek, S., Jeon, J., Jeong, B., & Jeong, Y. S. (2021). Two-Stage Hybrid Malware Detection Using Deep Learning. *Human-Centric Computing and Information Sciences*, *11*. https://doi.org/10.22967/HCIS.2021.11.027

Baptista, I., Shiaeles, S., & Kolokotronis, N. (2019). *A Novel Malware Detection System Based On Machine Learning and Binary Visualization*.

Beaman, C., Barkworth, A., Akande, T. D., Hakak, S., & Khan, M. K. (2021). Ransomware: Recent advances, analysis, challenges, and future research directions. *Computers and Security*, *111*. https://doi.org/10.1016/j.cose.2021.102490

Cusack, G., Michel, O., & Keller, E. (2018). *Machine Learning-Based Detection of Ransomware using SDN. SDN-NFVSec 2018 - Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization, Co-Located with CODASPY 2018*, *2018-January*, 1–6. https://doi.org/10.1145/3180465.3180467

Daku, H., Zavarsky, P., & Malik, Y. (2018). Behavioral-Based Classification and Identification of Ransomware Variants Using Machine Learning. *Proceedings - 17th IEEE International Conference on Trust, Security, and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, 1560–1564. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00224

Fernando, D. W., Komninos, N., & Chen, T. (2020). A Study on the Evolution of Ransomware Detection Using Machine Learning and Deep Learning Techniques. *IoT*, *1*(2), 551–604. https://doi.org/10.3390/iot1020030

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2016). *Densely Connected Convolutional Networks*. http://arxiv.org/abs/1608.06993

Lashkari, A. H., Kadir, A. F. A., Taheri, L., & Ghorbani, A. A. (2018). Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification. *2018 International Carnahan Conference on Security Technology (ICCST)*, 1–7. https://doi.org/10.1109/CCST.2018.8585560

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2, 3rd Edition*. Packt Publishing. https://www.packtpub.com/product/python-machine-learning-third-edition/9781789955750

Taheri, L., Kadir, A. F. A., & Lashkari, A. H. (2019). Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls. *2019 International Carnahan Conference on Security Technology (ICCST)*, 1–8. https://doi.org/10.1109/CCST.2019.8888430

Urooj, U., Al-Rimy, B. A. S., Zainal, A., Ghaleb, F. A., & Rassam, M. A. (2022). Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. *Applied Sciences (Switzerland)*, *12*(1). https://doi.org/10.3390/app12010172