An Ensemble-based Supervised Machine Learning Framework for Android Ransomware Detection

Shweta Sharma[1], Rama Krishna Challa[1], and Rakesh Kumar[2]
[1]Department of Computer Science and Engineering, National Institute of Technical Teachers Training and Research Chandigarh, India
[2]Department of Computer Science and Engineering, Central University of Haryana, India

Abstract: *With latest development in technology, the usage of smartphones to fulfill day-to-day requirements has been increased. The Android-based smartphones occupy the largest market share among other mobile operating systems. The hackers are continuously keeping an eye on Android-based smartphones by creating malicious apps housed with ransomware functionality for monetary purposes. Hackers lock the screen and/or encrypt the documents of the victim's Android based smartphones after performing ransomware attacks. Thus, in this paper, a framework has been proposed in which we (1) utilize novel features of Android ransomware, (2) reduce the dimensionality of the features, (3) employ an ensemble learning model to detect Android ransomware, and (4) perform a comparative analysis to calculate the computational time required by machine learning models to detect Android ransomware. Our proposed framework can efficiently detect both locker and crypto ransomware. The experimental results reveal that the proposed framework detects Android ransomware by achieving an accuracy of 99.67% with Random Forest ensemble model. After reducing the dimensionality of the features with principal component analysis technique; the Logistic Regression model took least time to execute on the Graphics Processing Unit (GPU) and Central Processing Unit (CPU) in 41 milliseconds and 50 milliseconds respectively.*

Keywords: *Smartphone security, android, ensemble learning, ransomware, and dimensionality reduction.*

Received February 20, 2021; accepted March 7, 2021
https://doi.org/10.34028/iajit/18/3A/5

1. Introduction

Besides food, clothing, and shelter; the Internet has emerged as the basic need for human beings over the past few years. The Internet can be accessed from personal computers to smartphone devices for sending emails, online shopping, online banking, interacting with family and friends through social media, and so on. Nowadays, smartphones with 12 GB RAM and 256 GB internal memory are available in the market to achieve the same task as a personal computer does [24].

However, out of all mobile Operating Systems (OS), Android OS dominates the market with approx. 75% global market share [24]. Android users install applications (commonly known as apps) in their smartphones which are written in JAVA language by the app developers [18]. Android has an open architecture [10]; thus the source code is available to the public and anybody can write and deploy the apps on third-party app stores such as APKPure.

Hackers take advantage of this open platform of Android OS to insert malicious code with ransomware functionality in genuine apps. Ransomware is a type of malware which locks the screen (locker ransomware) and/or encrypts the documents (crypto ransomware) on users' device [4]. Figure 1 shows the process of ransomware apps infecting Android devices after over-claiming permissions from users. After infecting the devices, hackers demand ransom from the victim in the form of crypto-currency to unlock the phone and decrypt the documents.

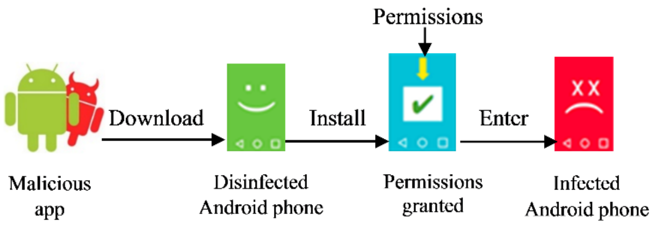

Figure 1. Ransomware targeting Android smart-phone.

The number of ransomware attacks on smart-phones is rapidly increasing as compared to other types of malware attacks. A total number of 4,339 ransomware samples have been detected during the first quarter of 2020 alone where the majority of ransomware are targeting the Android devices [24]. During this pandemic, there are several fake COVID-19 apps with ransomware functionality are available on the third-party app stores. For example, COVID19 Tracker and dubbed black rose lucy ransomware [24].

A lot of research work [2, 8, 27] has been done for the detection of generic Android malware using machine learning techniques. However, few researchers in the literature (discussed in the next section) worked on detection of Android ransomware. In this experimental work, we develop a framework to extract significant novel features of Android ransomware and perform detection using machine learning techniques.

Machine learning techniques require Graphics Processing Unit (GPU) for faster computation to train the dataset. Owing to the fact that the power cost of CPU to train the dataset is much lower than the GPU. Thus, CPU can be used in several situations where the time difference between CPU and GPU to train the dataset is relatively small. However, we perform a comparison of three different machine learning techniques (Logistic Regression, Neural Networks, and Support Vector Machine) to detect Android ransomware on two different platforms available on the cloud (i.e., CPU and GPU).

The contributions of this paper are:

a) Extracting novel features of Android ransomware,
b) Reducing the dimensionality of features.
c) Implementing an ensemble learning model to detect Android ransomware.
d) Finding best machine learning model for detection of Android ransomware.
e) Detect both locker and crypto ransomware.
f) Comparative analysis of the proposed framework on GPU and CPU to compute computational time.
g) Comparing the proposed framework with existing frameworks.

In the earlier version of the paper [23], the dimensionality reduction technique was not applied to the extracted features and individual machine learning models were employed to detect Android ransomware. In this paper, we implement Principal Component Analysis (PCA) technique to reduce the dimensions of the features which will eventually reduce the computation time required by machine learning models. An ensemble model (i.e., Random Forest) has also been implemented in this paper to further improve the accuracy rate of detecting Android ransomware. A significant reduction of computation time required by machine learning models after implementing the PCA technique has also been observed in this paper.

The rest of the paper is arranged as follows: section 2 presents the existing literature on Android ransomware detection. Section 3 presents the proposed framework for the detection of Android ransomware. Section 4 presents the experimental results of the proposed framework. Section 5 concludes the paper.

2. Literature Review

Saracino *et al.* [20] extracted system calls, installation of new apps, apps requesting admin privileges, apps producing large processes, apps running in the foreground, SMS, contact list, user activity to check whether a user is active or idle. They performed the detection of various malware such as Rootkits, Trojans, Ransomware, Spyware, and Botnet using k-Nearest Neighbors (kNN) classifier. Mercaldo *et al.* [15] extracted the JAVA Bytecode feature where each instruction was parsed and transformed to Calculus of Communicating Systems (CCS) by applying a transform operator. They used mu-calculus logic and applied Concurrency Workbench of New Century model for the detection of Android ransomware.

Maiorca *et al.* [14] extracted the Dalvikbytecode feature present in the dex files. They analyzed invoke-type instructions which belonged to system Application Program Interface (API) packages. They applied (RF) classifier on extracted features for the detection of Android ransomware. Gharib and Ghorbani [12] extracted text (encrypt, lock, porn, threat, monetary), images (nudity, logos), API methods, permissions, system and API calls sequence. They applied Support Vector Machine (SVM), RF, AdaBoost, Neural Networks, and Naive Bayes (NB) on extracted features for the detection of Android ransomware.

Ferrante *et al.* [11] extracted n-grams opcodes (n=2) present in the smali files. They monitored memory, system calls, network traffic logs, and CPU usage. They applied Decision Trees (DT), NB, and Logistic Regression (LR) on extracted features to detect ransomware apps.

Su *et al.* [26] extracted text, window properties, system commands, and permissions present in resource files, layout files, decompiled code, and manifest files respectively. They applied SVM, DT, RF, LR, and ensemble method on extracted features to detect ransomware apps. Scalas *et al.* [21] analyzed system API packages, classes, and methods present in the dex files. They applied RF classifier on extracted features for the detection of Android ransomware. Abdullah *et al.* [1] monitored system calls of Android OS and applied DT, NB, and RF to detect Android ransomware.

Asano *et al.* [6] compared the performance of GPU and CPU in the area of image processing where they found that due to small local memory in GPU; it can't execute machine learning models which use shared arrays. Panigrahi *et al.* [19] executed pattern matching algorithms (Bloom Filter and Wu-Manber) in GPU and CPU to detect Personal Computer (PC) malware where they found that the GPU takes less time to execute the pattern matching algorithms as compared to CPU.

Sharma *et al.* [23] extracted permissions, intents, text in users' native language from strings and images, locking, encryption, and encoding methods misused by Android ransomware. They applied LR, SVM, and Neural Networks on extracted features for the detection of Android ransomware. They found the computational time required by machine learning models on GPU and CPU to detect Android ransomware.

- *Discussion:* Sharma *et al.* [23] didn't implement any dimensionality reduction technique to reduce the required computation time to run the machine learning models on GPU and CPU. Also, Su *et al.* [26] worked only on detecting locker ransomware and ignored crypto ransomware detection. In literature, Asona *et al.* [6] Panigrahi *et al.* [19]

compared the performance of GPU and CPU in the area of image processing and PC malware. However, there is no comparative analysis of machine learning models on GPU and CPU to compute the computational time for detection of Android ransomware.

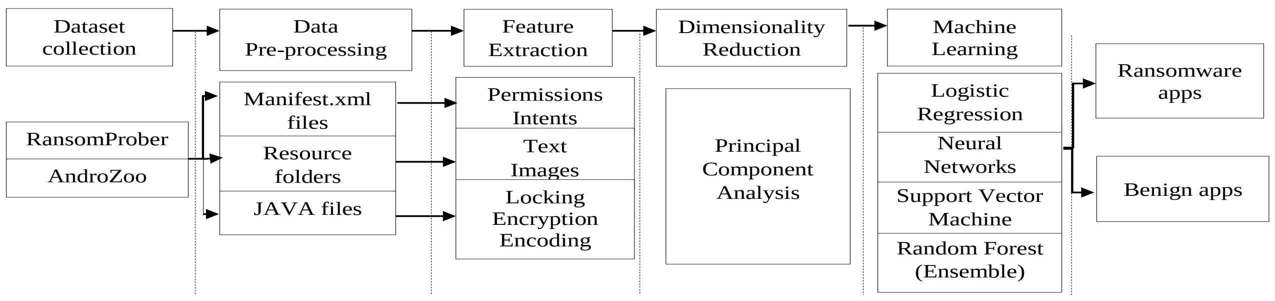

Figure 2. Flowchart of the proposed framework.

3. Proposed Framework

The proposed framework includes several steps as shown in Figure 2. These steps are discussed in the following:

- *Step* 1: Data Preprocessing

The datasets required for the experimental purpose have been collected from RansomProber [9] which contains malicious (ransomware) samples and AndroZoo [3] which contains benign samples. The raw dataset collected in the form of Android Package Kits (APKs) contains 2,721 ransomware samples and 2,000 benign samples [22]. The APK is a file format used by the Android OS to transmit and install apps on the smartphones. The following types of tools are used to perform reverse engineering to disassemble all APKs into a readable format:

- The *APKTool* is a command line tool for disassembling the APKs into Dalvik Executable (dex) files, AndroidManifest.xml files, resource folders, and smali files.
- The *dex2jar* tool is a command line tool for creating JAVA Archive (JAR) files from dex files.
- The *Java Decompiler* tool is a graphical utility for extracting the JAR files and parsing the JAVA source code.

After performing data pre-processing, there are total number of 2,076 ransomware apps and 2,000 benign apps left for analysis.

- *Step* 2: Feature Extraction

After performing data pre-processing in previous step, the objective of this step is to extract following features from ransomware apps:

- *Permissions*: all apps in Android OS must be permitted by the Permission system during installation. Android developers declare permissions in AndroidManifest.xml file by <uses‒permissions> tag. Thus, if an application wants access to contacts, microphone, location, or any other API, then Android OS will give a message to the user to either allow or deny the access. But malicious apps ask for unnecessary permissions from Android users to hijack the Android devices [13]. For example, ransomware illegally ask for KILL_BACKGROUND_PROCESSES permission to stop the antivirus process to prevent itself from being detected [25].
- *Intents*: intents are abstract objects in Android which are used by an app to make an action on another app (e.g., showing a map, taking a photo, sending a message). Intents are used as a glue to co-ordinate the interactions between activities, services, and broadcast receivers. Just like permissions, Android developers declare intents in AndroidManifest.xml file by <action> tag. But malicious apps illegally use Intents for obnoxious operations. For example, ransomware use DEVICE_ADMIN_ENABLED intent to gain device administrative rights [25].
- *Text*: the resource directory inside each Android app contains layout (for user interface) and values folder (string values in form of text). Android developers define text with <TextView> tag and labels of buttons with <string> tag. But ransomware misuse this directory to display text on the screen to threaten users after locking the device and display various steps to pay ransom via crypto-currency to unlock the phone and/or get the data back. The text can be displayed in users' native language. The Natural Language Processing (NLP) steps such as segmentation, Tokenization, Lemmatization, and stop words removal have been performed on extracted text in English, Chinese, and Russian language.
- *Images*: besides layout and values, the resource directory also contains drawable folder to store bitmap graphic files in different resolutions. Android developer use it to display static images in Android apps. But ransomware misuse this

directory to display fake images to threaten the users. Thus, Optical Character Recognition (OCR) technique has been used to extract text written in images. Just like string text, the NLP steps has been performed on extracted text from images in English, Chinese, and Russian language.

- *Locking*: android offers apps with locking functionality to let users' lock the screen. Android developers declare various methods in JAVA files to enable this functionality. But attackers misuse this functionality by writing malicious code of ransomware. Ransomware overrides these methods to lock the smartphone screen automatically after installation. For example, ransomware illegally use the onKeyUp, onKeyDown, TimerTask, and startActivity methods to prevent users' to escape lock screen.
- *Encryption*: android provides apps with encrypting functionality for users' to protect personal data. Android developers declare various methods in JAVA files to enable this functionality. But ransomware overrides these methods to encrypt users documents. For example, ransomware illegally use the Cipher.getInstance, Cipher.init, Cipher.doFinal, AesCrypt, CipherOutputStream methods to prevent users to get the data.
- *Encoding*: android provides character encoding functionality (UTF-8 and Base 64) in JAVA files such that the users' can read contents written in other languages than English. But ransomware overrides the Base64.encodeToString, Base64.decode, set Content Type, Url Encoded Form Entity, set Content Encoding JAVA methods to display threatening messages in users' native language.

Table 1 shows that total number of 1045 features of Android ransomware have been extracted in this step.

Table 1. Feature extraction of Android ransomware.

Files/Folders	Features	Total (Feature wise)
Android Manifest.xml	Permissions	330
	Intents	26
Values and Layout	Text	606
Drawable	Images	66
classes-dex2jar.jar.src	Locking	4
	Encryption	8
	Encoding	5
Total		1045

- *Step* 3: Dimensionality Reduction of Features

The features extracted in Step 2 are high-dimensional features. Thus, reducing the dimensions of the features will eventually reduce the computation time required by machine learning models. Therefore, PCA technique which is a dimensionality reduction technique is employed to reduce the higher dimensional features to low dimensional features. PCA converts the large number of features in a smaller set without losing any information [5].

As shown in Figure 3, approx. 90% variance is achieved by first 20 components which implies that 20 principal components cover maximum information as obtained by total number of original features (i.e., 1045) of Android ransomware.

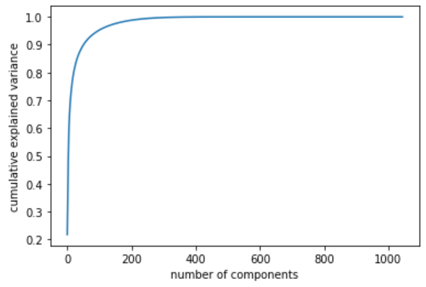

Figure 3. Cumulative variance achieved by number of principal components of Android ransomware.

PCA combines the features in such a way that the new features or principal components contain maximum amount of variance (information). For example, Figure 4 shows that 20 PCA features (or principal components) of Android ransomware covers all information as obtained by total number of original features of Android ransomware. Here, the first component covers the maximum information of Android ransomware, followed by the second component which covers the maximum remaining information of Android ransomware, and so on.

Figure 5 shows a 2-dimensional scatter plot of PCA components of Android ransomware. The scatter plot contains transformed features (i.e., combinations of original features) of Android ransomware which is obtained by PCA.

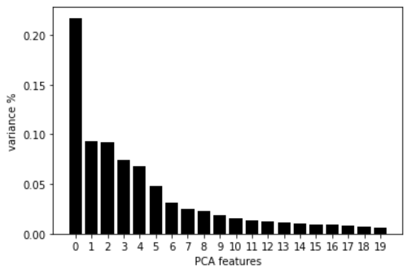

Figure 4. Percentage of variance by each PCA feature of Android ransomware.

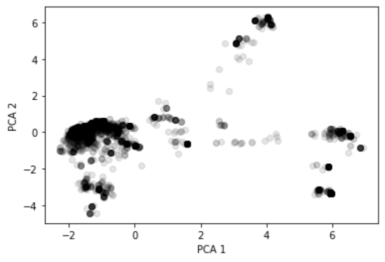

Figure 5. Scatter plot of PCA features of Android ransomware.

- *Step* 4: Detection of Android Ransomware

This is the final step in which supervised machine learning models are employed on extracted features to detect Android ransomware. The existing literature [11, 12, 26] reveals that classification models, namely, Logistic Regression (LR), Neural Networks, Support Vector Machine (SVM), and Ransom Forest (RF) have been used by the researchers to detect Android ransomware with high accuracy. These supervised models are employed and trained on significant features in our experimental work which are described in the following:

- *Logistic Regression*: this machine learning model is used when the outcome to predict is dichotomous (binary classification) [16]. For example, the goal of this experimental work is to classify whether an app is ransomware or benign based on multiple features. The output in LR is calculated as shown in Equation (1).

$$y = \frac{1}{1+e^{-z}}(W^T X + b) \tag{1}$$

Where:
W is n-dimensional vector, X is the input,
b is the real number, T is the transpose,
$\frac{1}{1+e^{-z}}$ is a sigmoid function

- *Artificial Neural Networks*: a neural network is a superset of LR with several network layers in which sigmoid function is used in the final layer.Neural networks contain artificial neurons (which are actually LR) to perform computations such as detection of ransomware apps. The first layer is the input layer in which the number of nodes is always equal to the number of features. The final layer is the output layer which tells whether the app is ransomware or benign. The layers present between input and output layers are hidden layers which are generally a black-box that uses association and activation functions and calculated values are then sent to the output layer. In this experiment, the Rectified Linear Unit (ReLU) activation function shown in Equation (2) is used in 5 hidden layers and Sigmoid function shown in Equation (3) is used in the final output layer.

$$\text{ReLU}=\max(0,x) \tag{2}$$

$$O = \frac{1}{1+e^{-z}} \tag{3}$$

Where:
X is the input to the neurons,
$\frac{1}{1+e^{-z}}$ is a sigmoid function

- *Support Vector Machine*: LR draws different decision boundaries near to the data points whereas SVM finds a maximum margin in the data points such that the distance between the hyperplane [7]. The maximum margin classifier can be calculated as shown in Equation (4):

$$minimize\ \frac{1}{2}||w||^2 where\ y_i(w.x_i + b) \geq 1, \forall i \tag{4}$$

Where:
x is the input,
w is the weight vector to the hyperplane,
$w.x_i + bis$ the output of a linear SVM

- *Random Forest*: in comparison to above implemented individual machine learning models (LR, Neural Networks, SVM); the ensemble model makes prediction by combining decisions of individual machine learning models. For example, RF is an ensemble machine learning model which performs Bootstrap Aggregation (Bagging) of different decisions trees to make predictions (such as classification of Android apps as ransomware or benign) [17].The branching of nodes in decision trees has been calculated using Entropy as shown in Equation (5):

$$Entropy = \sum_{i=1}^{c} -p_i * log_2(p_i) \tag{5}$$

Where:

p_i is the probability of an outcome

4. Experimental Results

The proposed framework discussed in the previous section is implemented on GPU and CPU with Python 3.6. The experiments for the detection of Android ransomware have been performed on GPU and CPU on Google Colaboratory which is a cloud service provided by Google. In our experimental work, 70% of the data has been used to train the model and 30% of data has been used for testing. These models are evaluated on four parameters, namely- Accuracy, Precision, Recall, and F-Score as shown in Equations (6), (7), (8), and (9) respectively.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{6}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{8}$$

$$F - score = 2.\frac{Precision*Recall}{Precision+Recall} \tag{9}$$

Where, *TP* is the True Positives values in which ransomware apps are correctly classified as ransomware. *TN* is the True Negatives values in which benign apps are correctly classified as benign. *FP* is the False Positives values in which the benign apps are wrongly classified as ransomware. *FN* is the False

Negatives values in which ransomware samples are wrongly classified as benign.

Table 2 shows the values of the performance metrics obtained after implementing the proposed framework to detect Android ransomware. The results show that all classification models employed in our research work effectively detect ransomware apps with a marginal difference. The RF model achieves the highest value of accuracy, precision, recall, and F-score to detect Android ransomware, followed by LR, Neural Networks and SVM model. The RF model gives best results because RF is an ensemble model which combines a set of decision trees to perform better than the individual machine learning models [17]. The LR model performs better than the Neural Network model for binary classification because neural networks require large amount of training data as compared LR to give better detection accuracy. Also, the SVM model performs best with unstructured and semi-structured data. However, in our experimental work, the data is stored in a structured manner with pre-defined independent variables.

Table 2. Experimental results of the proposed framework.

Machine Learning Model	TP	TN	FP	FN	Accuracy	Precision	Recall	F-Score
Random Forest (Ensemble Model)	634	585	0	4	0.9967	1.0	0.9937	0.9968
Logistic Regression	634	584	1	4	0.9959	0.9984	0.9937	0.9960
Neural Network	631	583	2	7	0.9926	0.9968	0.9890	0.9929
Support Vector Machine	631	582	3	7	0.9918	0.9952	0.9890	0.9921

4.1. Comparative Analysis of Proposed Framework on GPU and CPU

After measuring the performance metrics, this section calculates the computational time taken by classification models (LR, Neural Network, and SVM) to detect Android ransomware on GPU and CPU. The computational time taken by individual machine learning model to detect Android ransomware is shown in Figure 6 which shows that GPU takes less computation time as compared to CPU due to parallel processing. Also, the results show that the LR takes the least time (177 milliseconds in GPU, 235 milliseconds in CPU), followed by SVM (712 milliseconds in GPU, 803 milliseconds in CPU), and Neural Network model (5600 milliseconds in GPU, 6238 milliseconds in CPU). This reveals that the time difference to execute LR (58 milliseconds) and SVM model (91 milliseconds) on GPU and CPU is minimal. Thus, LR and SVM model can be executed on CPU to save the power cost to train the dataset. On the other side, the time difference to execute Neural Network model (638 milliseconds) on GPU and CPU is large because this model is more difficult to train due to back-propagation and number of hidden layers used. This suggests that the Neural Network model should be executed in GPU for faster computation.

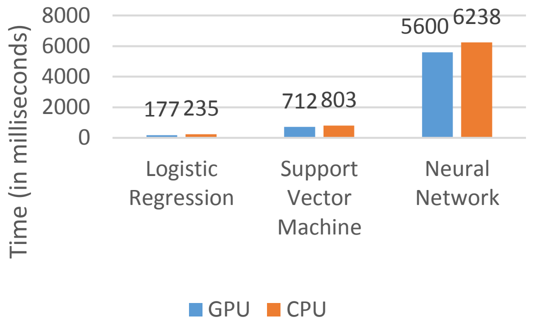

Figure 6. Computational time taken by machine learning models without PCA technique in GPU and CPU to detect Android ransomware.

After implementing PCA technique in individual machine learning model, the computational time required to detect Android ransomware has been significantly reduced. Figure 7 shows that after implementing PCA technique; LR takes the least time (41 milliseconds in GPU, 50 milliseconds in CPU), followed by SVM (42 milliseconds in GPU, 52 milliseconds in CPU), and Neural Network model (756 milliseconds in GPU, 1120 milliseconds in CPU). This reveals that the time difference to execute LR (9 milliseconds) and SVM model (10 milliseconds) on GPU and CPU is minimal. Thus, LR and SVM model can be executed on CPU to save the power cost to train the dataset. On the other side, the time difference to execute Neural Network model (364 milliseconds) on GPU and CPU is large because this model is more difficult to train due to back-propagation and number of hidden layers used. This suggests that the Neural Network model should be executed in GPU for faster computation.

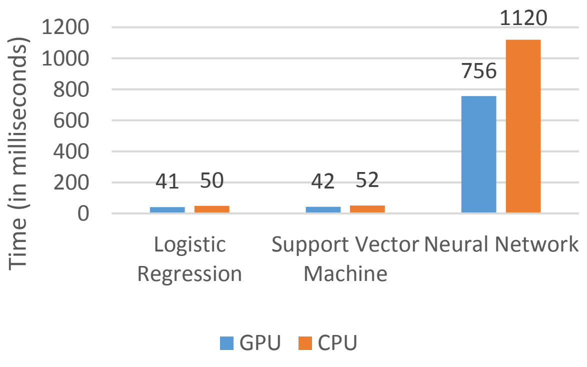

Figure 7. Computational time taken by machine learning models with PCA technique in GPU and CPU to detect Android ransomware.

4.2. Comparison of Proposed Framework with the Existing Frameworks

This section compares the accuracy of the best machine learning model (i.e., ensemble learning RF model) of the proposed framework with the existing frameworks to detect Android ransomware. Figure 8 shows that the proposed framework achieved the best accuracy (99.67%) to detect Android locker and crypto ransomware as compared to the existing Systemcall-based [1], DNA-DROID [12], API-based [21], and R-PackDroid [14] frameworks.

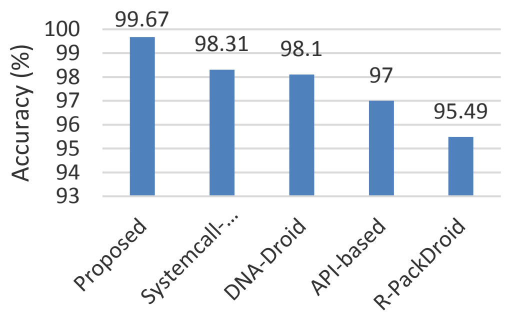

Figure 8. Comparison of the proposed framework with existing frameworks to detect Android ransomware.

5. Conclusions

This paper proposed a framework to classify Android ransomware and benign apps by using supervised machine learning models. The proposed framework extracted novel features by performing static analysis to recognize unknown ransomware apps. The proposed framework is implemented on GPU and CPU for comparative analysis of computational time taken by machine learning models to detect Android ransomware. The analysis revealed that the LR and SVM model can be implemented on CPU to save the power cost of GPU; but Neural Network model should be implemented on GPU for faster computation. The results showed that the proposed framework with ensemble RF model gave the best results with 99.67% accuracy as compared to the existing frameworks for the detection of locker and crypto Android ransomware.

The experimental results showed that the machine learning models with PCA technique took less computational time as compared to machine learning models without PCA technique. After implementing PCA technique; the LR model took minimum computation time (41 milliseconds in GPU, 50 milliseconds in CPU) and RF model delivered the best results as compared to other baseline models in terms of accuracy, precision, recall, and F-score to detect Android ransomware. The proposed framework can be deployed in an artificial intelligence based anti-malware to detect ransomware apps in Android based smartphones in real-time scenario.

References

[1] Abdullah Z., Muhadi F., Saudi M., Hamid I., and Foozy C., "Android Ransomware Detection Based on Dynamic Obtained Features," *in Proceedings of International Conference on Soft Computing and Data Mining*,Cham, pp. 121-129, 2020.

[2] Abuthawabeh M. and Mahmoud K., "Android Malware Detection and Categorization Based on Conversation-level Network Traffic Features," *in Proceedings of the International Arab Conference on Information Technology*, Al Ain, pp. 42-47, 2019.

[3] Allix K., Bissyandé T., Klein J., and Traon Y., "Androzoo: Collecting Millions of Android Apps for the Research Community," *in Proceedings of the International Conference on Mining Software Repositories*, Texas, pp. 468-471, 2016.

[4] Andronio N., Zanero S., and Maggi F., "Heldroid: Dissecting and Detecting Mobile Ransomware," *in Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, Kyoto, pp. 382-404, 2015.

[5] Arivudainambi D. and Visu P., "Malware Traffic Classification using Principal Component Analysis and Artificial Neural Network for Extreme Surveillance," *Computer Communications*, vol. 47, pp. 50-57, 2019.

[6] Asano S., Maruyama T., and Yamaguchi Y., "Performance Comparison of FPGA, GPU and CPU in Image Processing," *in Proceedings of the International Conference on Field Programmable Logic and Applications*, Czech Republic, pp. 126-131, 2009.

[7] Buczak A. and Guven E., "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1153-1176, 2016.

[8] Chakraborty T., Pierazzi F., and Subrahmanian V., "EC2 : Ensemble Clustering and Classification for Predicting Android Malware Families," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 262-277, 2017.

[9] Chen J., Wang C., Zhao Z., Chen K., Du R., and Ahn G., "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, 2018.

[10] Faruki P., Bharmal A., Laxmi V., Ganmoor V., Gaur M., Conti M., and Rajarajan M., "Android Security: A Survey of Issues, Malware Penetration, and Defenses," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 998-1022, 2015.

[11] Ferrante A., Malek M., Martinelli F., Mercaldo F., and Milosevic J., "Extinguishing Ransomware -A Hybrid Approach to Android Ransomware Detection," *in Proceedings of the International Symposium on Foundations and Practice of Security*, Nancy, pp. 242-258, 2017.

[12] Gharib A. and Ghorbani A., "DNA-Droid: A Real-Time Android Ransomware Detection Framework," *in Proceedings of the International*

Conference on Network and System Security, Cham, pp. 184-198, 2017.

[13] Kashefi I., Kassiri M., and Salleh M., "Preventing Collusion Attack in Android," *The International Arab Journal of Information Technology*, vol. 12, no. 6, pp. 719-727, 2015.

[14] Maiorca D., Mercaldo F., Giacinto G., Visaggio C., and Martinelli F., "R-PackDroid : API Package-Based Characterization and Detection of Mobile Ransomware," *in Proceedings of the International Symposium on Applied Computing*, Morocco, pp. 1718-1723, 2017.

[15] Mercaldo F., Nardone V., and Santone A., "Ransomware inside out," *in Proceedings of the International Conference on Availability, Reliability, and Security*, Salzburg, pp. 628-637, 2016.

[16] Milosevic N., Dehghantanha A., and Choo K., "Machine Learning Aided Android Malware Classification," *Computers and Electrical Engineering*, vol. 61, pp. 266-274, 2017.

[17] Muppavarapu V., Rajendran A., and Vasudevan S., "Phishing Detection using RDF and Random Forests," *The International Arab Journal of Information Technology*, vol. 15, no. 5 pp. 817-824, 2018.

[18] Nauman M. and Khan S., "Design and Implementation of a Fine-Grained Resource Usage Model for the Android Platform," *The International Arab Journal of Information Technology*, vol. 8, no. 4, pp. 440-448, 2011.

[19] Panigrahi C., Tiwari M., Pati B., and Prasath R., Malware Detection in Big Data Using Fast Pattern Matching: A Hadoop Based Comparison on GPU," *in Proceedings of the International Conference on Mining Intelligence and Knowledge Exploration*, Cham, pp. 407-416, 2014.

[20] Saracino A., Sgandurra D., Dini G., and Martinelli F., "Madam: Effective and Efficient Behavior-Based Android Malware Detection and Prevention," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 83-97, 2016.

[21] Scalas M. Maiorca D., Mercaldo F., Visaggio C., Martinelli F., and Giacinto G., "On the Effectiveness of System API-Related Information for Android Ransomware Detection," *Computers and Security*, vol. 86, pp. 168-182, 2019.

[22] Sharma S. Kumar N., Kumar R., and Krishna C., "The Paradox of Choice: Investigating Selection Strategies for Android Malware Datasets Using a Machine-learning Approach," *Communications of the Association for Information Systems*, vol. 46, no. 1, pp. 619-637, 2020.

[23] Sharma S., Krishna R., and Kumar R., "Android Ransomware Detection using Machine Learning Techniques: A Comparative Analysis on GPU and CPU," *in Proceedings ofInternational Arab Conference on Information Technology*, 6th of October city, pp. 1-6, 2020.

[24] Sharma S., Krishna C., and Kumar R., "A Survey on Analysis and Detection of Android Ransomware," *Concurrency and Computation: Practice and Experience*, pp.1-25, 2021.

[25] Sharma S., Kumar R., and Krishna C. R., "Ransom Analysis: The Evolution and Investigation of Android Ransomware" *in Proceedings of International Conference on IoT Inclusive Life*, Chandigarh, pp. 33-41, 2020.

[26] Su D., Liu J., Wang X., and Wang W., "Detecting Android Locker-Ransomware on Chinese Social Networks," *IEEE Access*, vol. 7, pp. 20381-20393, 2018.

[27] Wang S., Yan Q., Chen Z., Yang B., Zhao C., and Conti M., "Detecting Android Malware Leveraging Text Semantics of Network Flows," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1096-1109, 2018.

Shweta Sharma is pursuing Ph.D. from the Dept. of Computer Science and Engineering at NITTTR Chandigarh, India. She received M.Tech from Central University of Punjab, Bathinda. Her research areas include Smartphone Security, Malware Detection, and Machine Learning.

Rama Krishna Challa is a Professor in the Dept. of Computer Science and Engineering at NITTTR, Chandigarh, India. He received his Ph.D. from IIT Kharagpur and M.Tech. from CUSAT Cochin. His research areas include Wireless Communications and Networks,and Cyber Security.

Rakesh Kumar is an Associate Professor in the Dept. of Computer Science and Engineering at CUH, Mahendergarh, India. He received his PhD from NIT, Kurukshetra and M.Tech from GGSIPU. His research areas include Wireless Networks, Mobile Computing, and Cloud Computing.