



Article

Prevention of Crypto-Ransomware Using a Pre-Encryption Detection Algorithm

S. H. Kok ^{1,*} , Azween Abdullah ^{1,*} , N. Z. JhanJhi ^{1,*}  and Mahadevan Supramaniam ² ¹ School of Computer and IT (SoCIT), Taylor's University, Subang Jaya 47500, Selangor, Malaysia² Research and Innovation Management Center, SEGi University, Petaling Jaya 47810, Selangor, Malaysia; mahadevans@segi.edu.my

* Correspondence: koksimoong@sd.taylors.edu.my (S.H.K.); azween.abdullah@taylors.edu.my (A.A.); noorzaman.jhanjhi@taylors.edu.my (N.Z.J.)

Received: 23 August 2019; Accepted: 20 September 2019; Published: 1 November 2019



Abstract: Ransomware is a relatively new type of intrusion attack, and is made with the objective of extorting a ransom from its victim. There are several types of ransomware attacks, but the present paper focuses only upon the crypto-ransomware, because it makes data unrecoverable once the victim's files have been encrypted. Therefore, in this research, it was proposed that machine learning is used to detect crypto-ransomware before it starts its encryption function, or at the pre-encryption stage. Successful detection at this stage is crucial to enable the attack to be stopped from achieving its objective. Once the victim was aware of the presence of crypto-ransomware, valuable data and files can be backed up to another location, and then an attempt can be made to clean the ransomware with minimum risk. Therefore we proposed a pre-encryption detection algorithm (PEDA) that consisted of two phases. In, PEDA-Phase-I, a Windows application programming interface (API) generated by a suspicious program would be captured and analyzed using the learning algorithm (LA). The LA can determine whether the suspicious program was a crypto-ransomware or not, through API pattern recognition. This approach was used to ensure the most comprehensive detection of both known and unknown crypto-ransomware, but it may have a high false positive rate (FPR). If the prediction was a crypto-ransomware, PEDA would generate a signature of the suspicious program, and store it in the signature repository, which was in Phase-II. In PEDA-Phase-II, the signature repository allows the detection of crypto-ransomware at a much earlier stage, which was at the pre-execution stage through the signature matching method. This method can only detect known crypto-ransomware, and although very rigid, it was accurate and fast. The two phases in PEDA formed two layers of early detection for crypto-ransomware to ensure zero files lost to the user. However in this research, we focused upon Phase-I, which was the LA. Based on our results, the LA had the lowest FPR of 1.56% compared to Naive Bayes (NB), Random Forest (RF), Ensemble (NB and RF) and EldeRan (a machine learning approach to analyze and classify ransomware). Low FPR indicates that LA has a low probability of predicting goodware wrongly.

Keywords: crypto; encryption; machine learning; ransomware; intrusion detection

1. Introduction

Encryption is a process of encoding a message from a readable form to an unreadable form, in order to protect its content from unauthorized access. This method has been widely used in network security to ensure that only the intended recipient can access its content. It was crucial to encrypt the data, especially during its transit in the network, because the data can easily be stolen. However, it is now proven that encryption is a double edged sword.

At one edge, it was an excellent technique for cybersecurity to achieve its objectives of data confidentiality, data integrity, data authenticity and non-repudiation. On the other, it was used by cyber-criminals to lock its victim's files and then to extort ransom. This misuse can be found in an intrusion attack called crypto-ransomware.

Ransomware was a relatively new type of intrusion attack, with the objective of extorting a ransom from its victim, hence its name ransomware. There are primarily three distinct types of ransomware, as shown in Figure 1. The first type is called scareware. This type of ransomware does not post any real danger to its victim. It mainly tries to scare its victim into paying the ransom. One of the techniques used by scareware is imitating an authority that found some wrongdoing of the victim. It will demand a ransom in order to avoid legal prosecution. Another slightly different scareware threatens to expose the victim's wrongdoing to the victim's family and friends, therefore another name for it was called leakware.

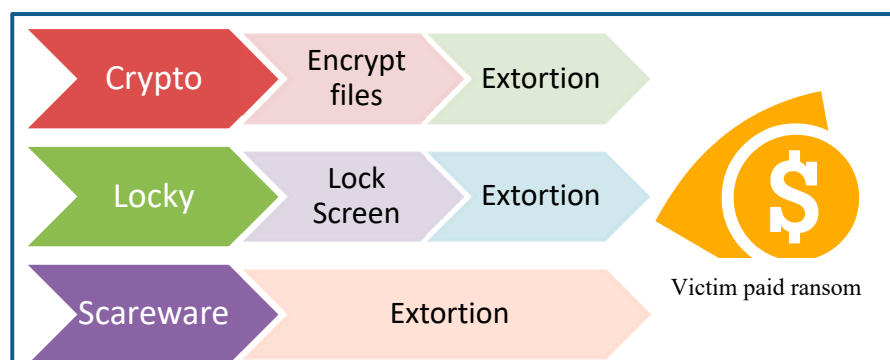


Figure 1. Three types of Ransomware.

The second type of ransomware is called locker-ransomware. This type of ransomware locks the victim's system by displaying a login page. The victim will need to pay a ransom in order to obtain the password in order to unlock the system. Locker-ransomware is considered to be mildly dangerous, because the attack can often be resolved by restarting the system in a Safe Mode. The third type of ransomware is called crypto-ransomware [1]. This type of ransomware is considered to be very dangerous, because it will encrypt the victim's files, rendering it impossible to be accessed without a valid decryption key [2]. Decrypting using a brute-force method could take thousands of years, depending on the strength of the encryption method used. In a positive light, this distinct characteristic of crypto-ransomware can be used as an important indication for its early detection, at the pre-encryption stage.

Therefore, this research will concentrate on addressing the danger posed by crypto-ransomware. Our proposed solution is PEDDA, which aims to detect crypto-ransomware before the ransomware starts to encrypt the files. This is important to ensure that the victim does not lose any files due to the encryption. If the victim has not lost any files, the victim does not need to pay any ransom, rendering the crypto-ransomware attack a failure.

2. Literature Review

In the past, the malware became increasingly frequent, causing a higher degree of damage to the victim [3]. One type of malware that had gained considerable attention was called ransomware, or more specifically crypto-ransomware. This type of malware has attracted significant interests from cybercriminals, because of several success stories that had world-wide impacts, such as Wannacry, Petya and NotPetya [4]. Due to the high interest, numerous new ransomware have been created, and existing ransomware have been improved with new variants. This has allowed ransomware to by-pass antivirus that uses the signature-based detection method that cannot detect zero-day attacks [5].

In addition, ransomware creation has been made easy with Ransomware-as-a-Service (Raas), which was available in the Dark Web [6]. Successful attacks by ransomware possess a major challenge to cyber security; this was especially important when most devices are connected via the Internet of Things (IoT) [7]. The risk and impact were higher for medical IoT devices [8].

Randep was the mapping of ransomware behavior using a Windows application programming interface (API) to determine the stages of a ransomware attack. In general, a ransomware attack can be segregated into three phases. The first phase is the stealth operations, the second phase is the suspicious activities, and the third phase is the obvious actions. Understanding ransomware behavior at every stage would allow the development of a mitigation plan at the desired stage [9]. There are generally two approaches to malware analysis. First, the approach uses static analysis that analyzes the malware's source code. The second approach uses dynamic analysis [10] that analyze the malware's actions after execution. In the first approach, the static analysis approach, one method is to transform the opcode (operation code) sequences into N-gram sequences [11]. The generated N-gram sequences will be the features for the machine learning model training. Subsequently, the trained predictive model can offer an accuracy of 91.43% [12].

Another method that uses static analysis is called RanDroid. This method aims to find a threatening message in either the text form or image form from the application code. The reason for this is because the objective of a ransomware is to demand a ransom from its victim. Thus, it must have a threatening message embedded in its code [13]. However, the threatening message could be another payload after the encryption of the victim's files. A portable executable header can also be extracted in static analysis, to determine malicious software from benign software. The team in [14] suggested that using both raw and derived value produced the best accuracy. Talos utilizes a formal method that uses application code to build ransomware rules for prediction, and was able to achieve a high accuracy of 99% [15]. However, this method was tested only for the Android platform. In the second approach, which uses dynamic analysis, one method was to capture all the processes performed by the ransomware in a sandbox. One of the most notable actions of crypto-ransomware was the encryption operation, which generated a high repetition of file system activities that can be tracked by the monitoring I/O operation [16–19].

Encryption is an important characteristic for crypto-ransomware, which uses the encryption method to lock a victim's file until the ransom was paid. However, encryption operation can also happen in legitimate ways, such as to protect the privacy and confidentiality of the user. The researcher in [20] has suggested determining the frequency threshold of the encryption operation in order to differentiate between the legitimate and illegitimate use of encryption. CryptoDrop suggested analyzing file activity based on encryption activity, but this method poses an average loss of 10 files [21]. RansomWall is a multi-layer system, with the last layer using Machine Learning to predict the outcome from features collected in the static analysis, dynamic analysis and trap layers. The method produces a detection rate of 98.25% and near-zero false positives using a Gradient Tree Boosting algorithm [22]. AntiBotics is an application authentication-based file access control. Ransomware needs to have access to files in order to perform any operation. Denying this access right can obstruct ransomware from capturing the file [23].

Song et al. [24] suggested monitoring processes and specific file directories to determine anomalies using statistical means. Process monitoring includes processor usage, memory usage and I/O rates. When an anomaly is detected, it will get confirmation from the user to remove the suspected ransomware. Another researcher suggested creating a trap called a honeyfile. The trap actually has two functions, the first is to attract an attack so that it will not attack the real files. The second function is to analyze the attack. This will allow the trap setter to understand the threat better, and allow it to create a further mitigation plan [25].

Table 1 shows the data used by previous researchers in ransomware research. Most of the researchers focus on file operation to determine the presence of ransomware, because the encryption process involves reading, modifying and deleting files. However, this approach allows several files to

be encrypted prior to detection. Other than file operation, API and I/O (input and output operation) are the two most used data.

Table 1. Summary of data used by researcher.

No.	Data	Count
1	File	8
2	API	3
3	I/O	3
4	Registry	2
5	Network	2
6	Opcode	1
7	Process	1
8	Text	1
9	Image	1
10	PE header	1
11	Android	2

During data collection, some researchers do not have a cut-off point, while others had employed time as cut-off points, such as 10 s, 30 s and 1 min. In this case, time cannot ensure that a ransomware has reached the same stage of attack, as it is dependent upon the ransomware itself, the system hardware and system configuration.

In view of this, we proposed that our detection system should concentrate on API to detect an encryption operation. API is an important system call to the operating system kernel for almost all of the system operations, in that all the programs must interact. In addition, once we know the relevant APIs, we can detect and pause the operation. This allows us to have control and opportunity to analyze the requesting program to determine whether it is a ransomware or otherwise. If it is ransomware, the program can be terminated; if it is a goodware, then the program can continue its operation. Therefore, our aim is to detect ransomware before encryption to ensure zero file loss.

3. Proposed Pre-Encryption Detection Algorithm (PEDA)

To understand the PEDA, we must first understand the behavior and characteristics of crypto-ransomware. Crypto-ransomware infection can be divided into five stages: Entry, command and control (C&C), search, encrypt and extort. This is illustrated in Figure 2. At the entry stage, the crypto-ransomware will self-propagate and complete its setup processes. More advanced crypto-ransomware will have additional tasks, such as payload persistence, restrict system restore, stealth mode, environment mapping and privilege elevation [26,27].

During the command and control (C&C) stage, the crypto-ransomware will try to establish a communication link to its C&C center, which is often managed by the crypto-ransomware creator or campaign manager. Advanced crypto-ransomware may employ communication masking in order to avoid detection. Once the communication has been established, it will send information about the victim's system, and in return it will obtain the encryption key. The next stage is the searching stage, where the crypto-ransomware will find all of the targeted file of interest. The file of interest consists of files that are considered to be important to the victim, so much so that the victim is (assumed to be) willing to pay the ransom for it. When the search has been completed, crypto-ransomware will start to encrypt the file using the encryption key from C&C. Once this stage started, the encrypted files are considered to be gone, unless the victim is willing to pay the ransom. The last stage of the crypto-ransomware is to display the extortion message, demanding ransom for the release of the decryption key.

In this research, we have proposed an algorithm, PEDA, that will detect the crypto-ransomware before the encryption process starts, as shown in Figure 3, in order to avoid valuable files from being

held for ransom. PEDAs consists of two phases, Phase-I is the learning algorithm, and Phase-II is the signature repository algorithm. Figure 4 shows the pseudo-code for PEDAs.

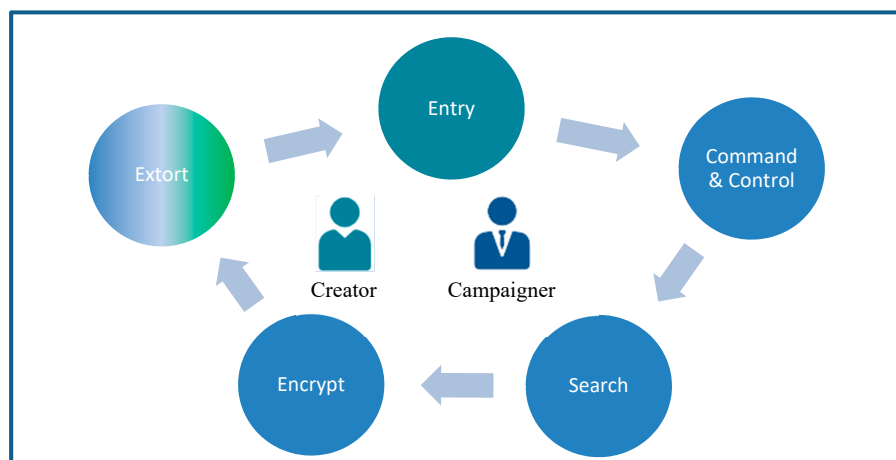


Figure 2. A Pre-encryption Detection Algorithm (PEDAs) Model.

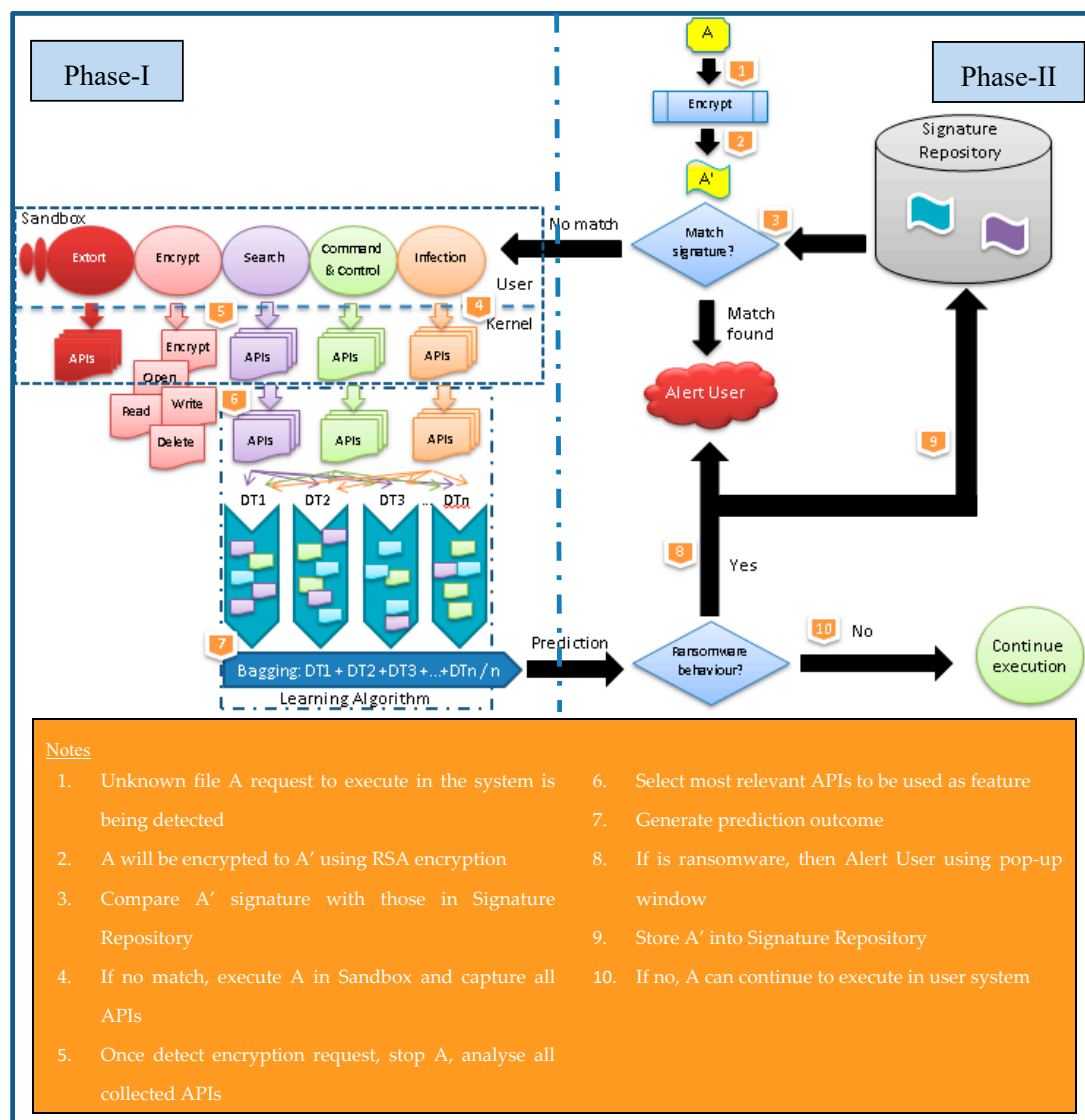


Figure 3. Proposed PEDAs Model.

Pseudo-Code for PEDAs

1. *Sample executes in sandbox*
2. *Check if API is for encryption*
3. *If yes, write API and time in text file with comma delimiter*
4. *Kill sample execution*
5. *Save text file with filename of system date and time*
6. *Open text file*
7. *Generate template API and set all count = 0*
8. *Check if API match with template API*
9. | *If yes, change API count = 1 in a new text file with comma delimiter*
10. | *Find next API*
11. | *If yes continue to step 8*
12. | *If no, close original text file*
13. | *Save new text file with filename same as original text file with "p" at the back*
14. *If no, write API and time in text file with comma delimiter, then continue to step 2.*
15. *Input file with "p" into predictive model*
16. *Get output from predictive model*
17. *If output is "ransomware", alert user with pop-up window*
18. *If output is "goodware", sample can execute in user system.*

Figure 4. Pseudo Code for PEDAs.**3.1. Phase-I: Learning Algorithm (LA)**

An unknown file will be injected into a sandbox that has been created using Cuckoo Sandbox. This tool will extract the API or system call, which happens in the kernel, at every stage of an executable file. Once the API for encryption has been detected, it will stop the execution. All the collected APIs will then be analyzed using the Learning Algorithm (LA). This LA consists of two stages: The first stage is discretization, and the second stage is prediction. Discretization helps to change the numerical value to a binary value that can simplify and enhance the classification process in the prediction stage. Each unique API has been considered as one feature. Therefore, we expect the number of APIs to be in the hundreds, which is considered to be a very high-dimensional data, and prone to over-fitting.

The prediction stage consists of a homogenous ensemble of decision trees. It will randomly select a feature based on \sqrt{F} , where F is the number of features. The randomly selected features will be used to build the decision tree, where the node is split based on the Gini Index $Gini(t) = 1 - \sum_{i=1}^N P(C_i|t)^2$ as shown in (1). Each outcome predicted by the decision tree will be bagged to produce the final prediction at this stage. This method can help to overcome over-fitting and noisy data [28].

$$Gini(t) = 1 - \sum_{i=1}^N P(C_i|t)^2 \quad (1)$$

where t is a condition, N the number of classes, and C_i is the i th class label.

In PEDAs implementation, the prediction stage must first be trained using a labeled dataset to build its prediction model. Then, the fixed prediction model will be used for prediction. We do not use a dynamic learning algorithm, because it may be subjected to a poisoning attack, as stated by Chen et al. [29]. Once this model has successfully been established, it will be able to determine whether the unknown file is a crypto-ransomware or not.

If the unknown file is found to be not a crypto-ransomware, it will be allowed to continue its execution. Otherwise, it will alert the user, and a notification is sent to the Signature Repository

3.2. Phase-II: Signature Repository

In Phase-II, once the unknown file is determined to be a crypto-ransomware, the file's signature will be stored in an SQL database, which is also called the Signature Repository (SR). A signature is produced by encrypting the file using the Rivest–Shamir–Adleman (RSA) method with no file extension. The purpose of creating the Signature Repository is to allow a significantly earlier detection of known crypto-ransomware. The process flow is shown in Figure 3.

When a file requests to be executed, it will be detected by PEDAs. First, the file will be encrypted using the RSA method, and a signature of the file will be created. This signature will then be compared with known crypto-ransomware in the Signature Repository using SQL queries. If a match can be found, then the user will be alerted using a pop-up window, and the file will not be allowed to be executed. If there is no match, the file will continue to be processed in Phase-I. If Phase-I also determines that it is not a crypto-ransomware, the file will be allowed to be executed in the user system. However, if Phase-I determines that the file is a crypto-ransomware, then the created signature will be stored in the Signature Repository, and the user will be alerted with a pop-up window.

It can be concluded that our proposed solution provides two layers of detection: Pre-execution (Phase-II) and pre-encryption (Phase-I) detection. Pre-execution detection uses a signature matching of known crypto-ransomware and it is fast and accurate, but it can be too rigid, and cannot detect new variants of the crypto-ransomware. The pre-encryption detection uses behavior matching that provides a higher probability to detect small variants of unknown crypto-ransomware. Both methods can complement each other to provide efficient and effective protection from crypto-ransomware.

4. Results and Discussion

The first comparison is between PEDAs-Phase-I, labeled as PEDA, and three other learning algorithms. These algorithms are Random Forest (RF), Naïve Bayes (NB) and Ensemble. This Ensemble algorithm is an ensemble of both RF and NB. Figure 5 shows the performance of these algorithms based on four evaluation metrics. Based on the result, it can be observed that PEDAs-Phase-I performs the best with the highest AUC of 0.9930, lowest FPR of 0.0156 and lowest test error of 0.0295. A high AUC value shows that the algorithm can distinguish well between the positive and negative outcomes. Low FPR shows that it is less likely to produce a false alarm, whereas, a low test error value shows that the algorithm can satisfactorily predict true positive and true negative. The ensemble has the highest detection rate of 0.9872, which shows that it can correctly predict a positive outcome. However, the ensemble algorithm performs the worst based on other metrics. Overall, the RF comes in second behind PEDAs-Phase-I, therefore RF is selected for enhancement with a feature selection in the next experiment.

In the second experiment, the performance of the PEDAs-Phase-I algorithm is compared to RF, and RF with 2 feature selection algorithms, to improve the performance of RF. The result is shown in Figure 6 below. Based on the result, Correlation-based Feature Selection (CFS) helps to improve the RF to achieve the highest detection rate of 0.9647, while it still fails to improve other metrics due to the high number of false negative predictions. Meanwhile, when using Principal Component Analysis (PCA) as a feature selection for RF, it performs worse than RF alone in all of the measured metrics.

In the third experiment, PEDAs-Phase-I is compared to the EldeRan algorithm. The result is shown in Figures 7 and 8. Based on the results, it can be observed that PEDAs-Phase-I and EldeRan perform very closely with each other. EldeRan performs better with the AUC of 0.9949, and the detection rate of 0.9634 as shown in Figure 7. Meanwhile, Figure 8 shows that EldeRan has a lower test error of 0.0238, but PEDAs-Phase-I has the lower FPR of 0.0156, which shows that it is less likely to produce a false alarm.

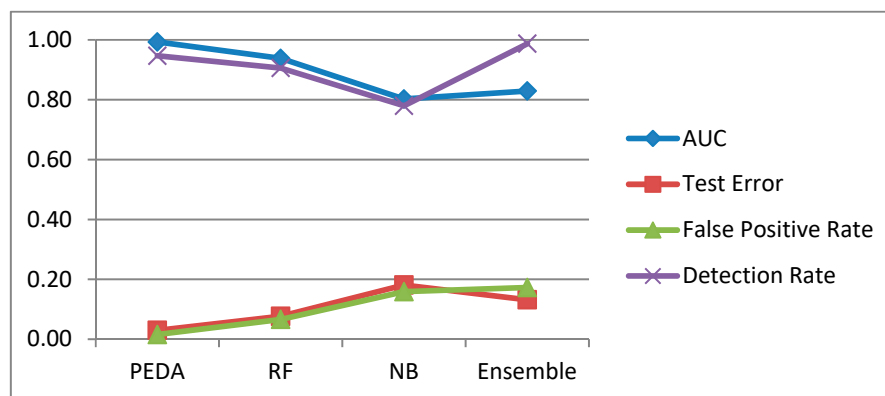


Figure 5. Comparison of (PEDA), EldeRan, (RF), (NB) and (RF and NB).

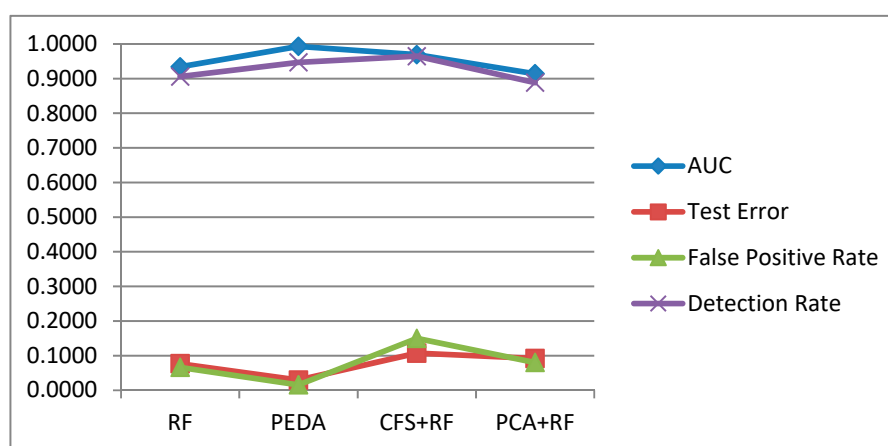


Figure 6. Comparison of RF, PEDA, Correlation-based Feature Selection + Random Forest (CFS+RF) and Principal Component Analysis + Random Forest (PCA+RF).

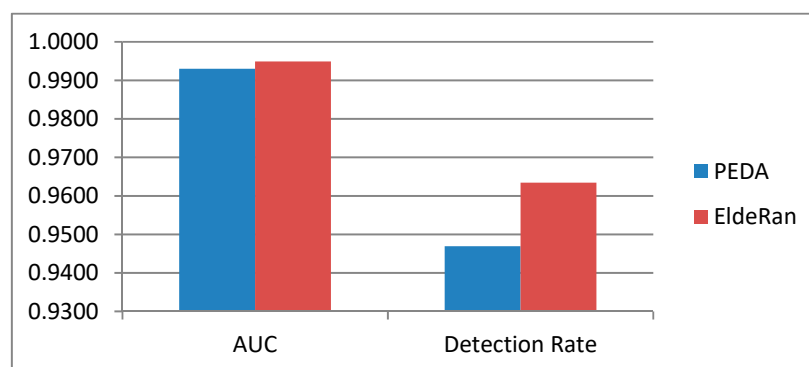


Figure 7. Comparison of PEDA and EldeRan (AUC and Detection Rate).

Based on the result, we can be convinced that PEDA-Phase-I performed better when compared to RF and NB. Even with feature selection in RF, the overall performance of PEDA-Phase-I is better. This research also shows that feature selection may improve or reduce the performance of the learning algorithm. Therefore, proper matching must be performed to obtain the desired result.



Figure 8. Comparison of PEDA and EldeRan (Test Error and false positive rate (FPR)).

4.1. Discussion

As mentioned in the Result section, we found that EldeRan and PEDA-Phase-I performed very closely with each other. However, there are a few important distinctions between the experiment setups of both algorithms. EldeRan used the whole Resilient Information System Security (RISS) dataset, which consists of all five categories of features with a total of 30,967 features. Meanwhile, PEDA-Phase-I used one category (API) with 232 features. Therefore, it shows that PEDA-Phase-I can achieve similar results with a considerably less number of features, and API is an important feature in ransomware detection.

EldeRan is a good model for ransomware detection, but may result in some data loss due to encryption, which can be clearly seen from the inclusion of the dropped files category in the RISS dataset. Files will be dropped or deleted after ransomware has encrypted the file to avoid the user from accessing it, and thus capturing the file for ransom.

4.2. Methodology

In our experiment, we had used a dataset that was created by RISS from Imperial College, London, in 2016. This dataset was selected because it has an API data for 10 ransomware families and a good selection of goodware. The dataset was created using a dynamic analysis approach for 582 samples of ransomware and 942 samples of goodware [30].

Detailed distributions of the samples are shown in Table 2. The data are captured in five main categories with 30,067 features. Detailed distribution of the category is shown in Table 3. However, for our research, we will only concentrate on one category, which is the system call category. The system call category has 232 features, and the dataset has been labeled with ransomware or goodware. This experiment was performed using a laptop with an Intel i5 processor, 8 Gb RAM and Win 7 (64-bit). The tools for the machine learning at this stage were Weka (Waikato Univesity, Hamilton, New Zealand) and RStudio.

First we checked for missing data, and none was found. Therefore we proceeded to train the pre-encryption detection algorithm (PEDA)-Phase-I, which is the Learning Algorithm (LA), using the dataset. First, the dataset was divided into a ratio of 8:2. Eighty per cent of the dataset will be used for training to establish the prediction algorithm. Once the prediction model has been established, we then evaluated its performance by feeding the remaining 20% of the dataset.

To gauge the performance of the LA model, we compared it with three other algorithms. The random forest (RF) and Naïve Bayes (NB) algorithms were selected based on our references [31,32]. The third algorithm was the Ensemble (of RF and NB), which was selected using majority voting [33,34]. The R code is as shown in Figure 9 below.

Table 2. Sample distribution in the Resilient Information Systems Security (RISS) dataset.

Sample	ID	Quantity
Goodware	0	942
Critroni'	1	50
CryptLocker'	2	107
CryptoWall'	3	46
KOLLAH'	4	25
Kovter'	5	64
Locker'	6	97
MATSNU'	7	59
PGPCODER'	8	4
Reveton'	9	90
TeslaCrypt'	10	6
Trojan-Ransom'	11	34

Table 3. Categories of features in the RISS dataset.

No.	Category	Quantity
1	API	232
2	Registration Key	346
3	Dropped File	6622
4	Files and Directory Operation	7500
5	Embedded String	16,267
	Total	30,967

```

#Naive Bayes
library("klaR")
mod_NB <- train(V1 ~ ., data=training, method="nb")
pred = predict(mod_NB, newdata=testing)
confusionMatrix(pred,testing[, "V1"])

#Random Forest
library("randomForest")
mod_RF <- train(V1 ~ ., data=training, method="rf")
pred2 = predict(mod_RF, newdata=testing)
confusionMatrix(pred2,testing[, "V1"])

#Ensemble Majority Voting (2 algo)
pred_majority<-as.factor(ifelse(pred=='1' & pred2=='1','1','0'))
confusionMatrix(pred_majority,testing[, "V1"])

```

Figure 9. Code in R for Confusion Matrix for 3 algorithms.

Next, we compared LA with an improvement of the best performing algorithm from the first comparison using feature selection. Two feature selection algorithms were chosen. The first algorithm was the Principal Component Analysis (PCA) that calculates the eigenvalue to find the eigenvector from the dataset. Features that have a low eigenvalue were dropped from the dataset, thus removing features that were less important to the outcome. Second, the pre-processing algorithm was the Correlation-based Feature Selection (CFS). This algorithm selects a feature that has high correlation to the outcome, but low correlation between other features.

After that we also compared the LA with the EldeRan algorithm [30]. Details of both algorithms, LA and EldeRan, are shown in Figure 10. Figure 10A shows the PEDAs-Phase-I (LA) because we used RISS dataset that cannot distinguish between APIs before and after encryption, which is required by PEDAs. However, in this research, for PEDAs-Phase-I, we had used only API data from the RISS dataset. Another important distinction for PEDAs-Phase-I was the specification for crypto-ransomware, while RISS had both crypto-ransomware and locky-ransomware.

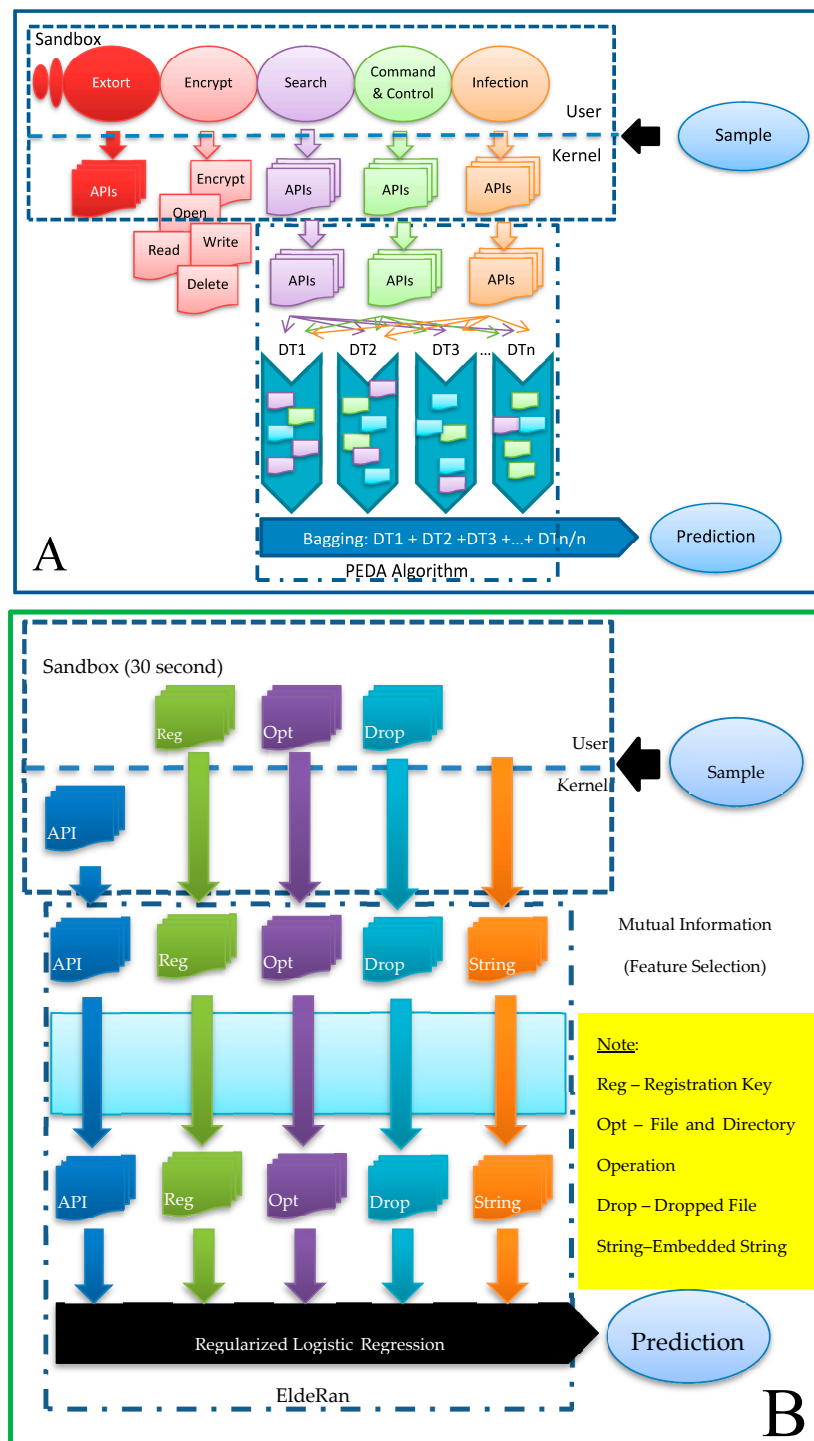


Figure 10. Comparison between A-partial PEDAs and B-EldeRan Algorithms.

Figure 10B is the EldeRan model that uses the whole RISS dataset that includes API, the registration key, file and directory operation, dropped file and embedded string. The dropped file data indicates that the RISS dataset contains data after the encryption stage, because the ransomware will drop the file only after the encryption had been completed. In the EldeRan model, first, the data go through feature selection using mutual information. Next, the selected features will be trained and tested using regularized logistic regression to predict the binary classification.

The last algorithm we compared with was the EldeRan algorithm [30], and the algorithm comparison is shown in Figure 10. Figure 10A is stated as a partial PEDA, because we used the RISS dataset that cannot distinguish between API before encryption and API after encryption, which is required by PEDA. However, the PEDA model had only used API data from the RISS dataset. Another important distinction for the PEDA model is the specification for crypto-ransomware, while RISS has both crypto-ransomware and locky-ransomware. Figure 10B is the EldeRan model that uses the whole RISS dataset that includes API, the registration key, file and directory operation, dropped file and embedded string. Dropped file data indicates that the RISS dataset contains data after the encryption stage, because ransomware will only drop the file after encryption had been accomplished. In the EldeRan model, first the data will go through feature selection using Mutual Information. After that, the selected features will be trained and tested using Regularized Logistic Regression to predict the binary classification.

For the evaluation metric, we compared four metrics according to the research by Sgandurra et al. [30]. These are areas under the curve of the receiver operating characteristic (AUC ROC), Test Error, Detection Rate and False Positive Rate (FPR). These metrics are based on a confusion matrix [35] as shown in Table 4.

Table 4. Confusion Matrix.

		Predicted Class	
		Negative (Normal)	Positive (Attack)
Actual Class	Negative (Normal)	True Negative (TN)	False Positive (FP)
	Positive (Attack)	False Negative (FN)	True Positive (TP)

Test Error is as shown in Equation (3) $Test\ Error = 1 - \frac{TP+TN}{TP+TN+FP+FN}$. It also means $(1 - Accuracy)$. Accuracy is the ratio of all accurate predictions, $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

$$Test\ Error = 1 - \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

where (referring to Table 4);

TP – True Positive

TN – True Negative

FP – False Positive

FN – False Negative

The Detection Rate is also called the True Positive Rate (TPR). The formula is as shown in Equation (4) $TPR = \frac{TP}{TP+FN}$. It is the ratio of the correctly predicted positive outcome over the actual positive outcome.

$$TPR = \frac{TP}{TP + FN} \quad (4)$$

The False Positive Rate (FPR) is shown in Equation (5) $FPR = \frac{FP}{FP+TN}$. It is the ratio of the wrongly predicted positive outcome over the actual negative outcome.

$$FPR = \frac{FP}{FP + TN} \quad (5)$$

ROC is the curve of FPR versus TPR, therefore the AUC of ROC calculates the prediction algorithm's ability to distinguish between positive and negative outcomes.

5. Conclusions and Future Work

Comparing LA with other learning algorithms such as RF and NB, LA has better performance in all the calculated metrics, namely AUC, test error, FPR and detection rate. Even when compared to an Ensemble of RF and NB, and RF with feature selection, LA's overall performance is still better.

Therefore, we can conclude that PEDAs-Phase-I or LA has achieved its objective as an effective prediction model to detect crypto-ransomware using only API data. LA consisted of an integration of the discretization of the data and the ensemble of multiple Decision Trees. Discretization helps to reduce the complexity of the data, and ensemble reduces over-fitting of high dimensional data. These two approaches have allowed LA to achieve a good result. LA's performance is also comparable to EldeRan, but with considerably simple and focused data of choice. EldeRan uses API, registration key, file and directory operation, dropped file and embedded string, totaling 30,067 features. In addition, LA is found to produce the lowest False Positive Rate of 1.56%, which means that it has a very low probability of determining a goodware wrongly. This is important to ensure that the user receives the minimum false alarm, reducing the stress of alerting user wrongfully. Therefore, we can conclude that PEDAs-Phase-I or LA can be a great tool to detect and stop any crypto-ransomware attack.

The PEDAs model aims to detect ransomware at the pre-encryption stage, which is crucial in stopping the need to pay ransom. However, it can only be implemented using a new dataset with API from the pre-encryption stage. This limitation can be considered as an objective for future research. In addition, PEDAs-Phase-II aims to store all detected ransomware's signature to produce a Signature Repository. This storage function is important to stop the ransomware at an earlier stage or the pre-execution stage, thus achieving zero damage from a known ransomware. Signature detection is rigid, but can produce accurate and fast detection.

Author Contributions: Funding acquisition, M.S.; Investigation, S.H.K.; Supervision, A.A. and N.Z.J.; Visualization, S.H.K.; Writing—original draft, S.H.K.; Writing—review & editing, A.A., N.Z.J. and M.S.

Funding: This work was supported by Taylor's University, 47500 Subang Jaya, Selangor, Malaysia, through its TAYLOR'S PhD SCHOLARSHIP Program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tailor, J.P.; Patel, A.D. A Comprehensive Survey: Ransomware Attacks Prevention, Monitoring and Damage Control. *Int. J. Res. Sci. Innov.* **2017**, *4*, 2321–2705.
2. Askarifar, S.; Rahman, N.A.A.; Osman, H. A review of latest wannacry ransomware: Actions and preventions. *J. Eng. Sci. Technol.* **2018**, *13*, 24–33.
3. Mathur, A.; Idika, N. *A Survey of Malware Detection Techniques*; Department of Computer Science, Purdue University: West Lafayette, IN, USA, 2007.
4. Shakir, H.; Jaber, A.N. A Short Review for Ransomware: Pros and Cons. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*; Springer: Cham, Switzerland, 2018.
5. Celiktas, B.; Karacuha, E. The Ransomware Detection and Prevention Tool Design by Using Signature and Anomaly Based Detection Methods. Ph.D. Thesis, Istanbul Technical University, Istanbul, Turkey, 2018.
6. Alhawi, O.M.K.; Baldwin, J.; Dehghantanha, A. Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection. *Adv. Inf. Secur.* **2018**, *70*, 1–11.

7. Homayoun, S.; Dehghantanha, A.; Ahmadzadeh, M.; Hashemi, S.; Khayami, R.; Choo, K.K.R.; Newton, D.E. DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer. *Future Gener. Comput. Syst.* **2019**, *90*, 94–104. [\[CrossRef\]](#)
8. Branche, P.O. Ransomware: An Analysis of the Current and Future Threat Ransomware Presents. Ph.D. Thesis, Utica College, Utica, NY, USA, 2017.
9. Hull, G.; John, H.; Arief, B. Ransomware deployment methods and analysis: Views from a predictive model and human responses. *Crime Sci.* **2019**, *8*, 2. [\[CrossRef\]](#)
10. Torres, P.E.P.; Yoo, S.G. Detecting and neutralizing encrypting Ransomware attacks by using machine-learning techniques: A literature review. *Int. J. Appl. Eng. Res.* **2017**, *12*, 7902–7911.
11. Pektaş, A.; Acarman, T. Classification of malware families based on runtime behaviors. *J. Inf. Secur. Appl.* **2017**, *37*, 91–100. [\[CrossRef\]](#)
12. Zhang, H.; Xiao, X.; Mercaldo, F.; Ni, S.; Martinelli, F.; Sangaiah, A.K. Classification of ransomware families with machine learning based on N-gram of opcodes. *Future Gener. Comput. Syst.* **2019**, *90*, 211–221. [\[CrossRef\]](#)
13. Alzahrani, A.; Alshehri, A.; Alshahrani, H.; Alharthi, R.; Fu, H.; Liu, A.; Zhu, Y. Randroid: Structural Similarity Approach for Detecting Ransomware Applications in Android Platform. In Proceedings of the 2018 IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3–5 May 2018; pp. 892–897.
14. Kumar, A.; Kuppusamy, K.S.; Aghila, G. A learning model to detect maliciousness of portable executable using integrated feature set. *J. King Saud Univ. Comput. Inf. Sci.* **2016**, *31*, 252–265. [\[CrossRef\]](#)
15. Cimitile, A.; Mercaldo, F.; Nardone, V.; Santone, A.; Visaggio, C.A. Talos: No more ransomware victims with formal methods. *Int. J. Inf. Secur.* **2018**, *17*, 719–738. [\[CrossRef\]](#)
16. Surati, S.B.; Prajapati, G.I. A Review on Ransomware Detection & Prevention. *IJRS* **2017**, *IV*, 86–91.
17. Kardile, A.B. Crypto Ransomware Analysis and Detection Using Process Monitor. Ph.D. Thesis, The University of Texas at Arlington, Arlington, TX, USA, 2017.
18. Kharraz, A.; Robertson, W. Techniques and Solutions for Addressing Ransomware Attacks. Ph.D. Thesis, Northeastern University, Boston, MA, USA, 2017.
19. Monika; Zavorsky, P.; Lindskog, D. Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization. *Procedia Comput. Sci.* **2016**, *94*, 465–472. [\[CrossRef\]](#)
20. Mulders, D.A.C. Network Based Ransomware Detection on the Samba Protocol. Master's Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2017.
21. Scaife, N.; Carter, H.; Traynor, P.; Butler, K.R.B. CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data. In Proceedings of the 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), Nara, Japan, 27–30 June 2016; pp. 303–312.
22. Shaukat, S.K.; Ribeiro, V.J. RansomWall: A Layered Defense System against Cryptographic Ransomware Attacks using Machine Learning. In Proceedings of the 2018 10th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, India, 3–7 January 2018.
23. Ami, O.; Elovici, Y.; Hendler, D. Ransomware Prevention Using Application Authentication-Based File Access Control. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing, Pau, France, 9–13 April 2018.
24. Song, S.; Kim, B.; Lee, S. The Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform. *Mob. Inf. Syst.* **2016**, *2016*, 2946735. [\[CrossRef\]](#)
25. Gómez-Hernández, J.A.; Álvarez-González, L.; García-Teodoro, P. R-Locker: Thwarting ransomware action through a honeyfile-based approach. *Comput. Secur.* **2018**, *73*, 389–398. [\[CrossRef\]](#)
26. Tandon, A.; Nayyar, A. *A Comprehensive Survey on Ransomware Attack: A Growing Havoc Cyberthreat*; Springer: Singapore, 2018.
27. Kok, S.H.; Abdullah, A.; Jhanjhi, N.Z.; Supramaniam, M. Ransomware, Threat and Detection Techniques: A Review. *Int. J. Comput. Sci. Netw. Secur.* **2019**, *19*, 136–146.
28. Fawagreh, K.; Gaber, M.M.; Elyan, E. Random forests: From early developments to recent advancements. *Syst. Sci. Control Eng.* **2014**, *2*, 602–609. [\[CrossRef\]](#)
29. Chen, S.; Xue, M.; Fan, L.; Hao, S.; Xu, L.; Zhu, H.; Li, B. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *Comput. Secur.* **2018**, *73*, 326–344. [\[CrossRef\]](#)

30. Sgandurra, D.; Muñoz-González, L.; Mohsen, R.; Lupu, E.C. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and Use for Detection. *arXiv* **2016**, arXiv:1609.03020v1.
31. Kok, S.H.; Abdullah, A.; Supramaniam, M.; Pillai, T.R.; Hashem, I.A.T. A Comparison of Various Machine Learning Algorithms in a Distributed Denial of Service Intrusion. *Int. J. Eng. Res. Technol.* **2019**, *12*, 1–7.
32. Lim, M.; Abdullah, A.; Jhanjhi, N.Z. Performance optimization of criminal network hidden link prediction model with deep reinforcement learning. *J. King Saud Univ. Comput. Inf. Sci.* **2019**. [[CrossRef](#)]
33. Olaniran, O.R.; Abdullah, M.A.A.B. BayesRandomForest: An R Implementation of Bayesian Random Forest for Regression Analysis of High-Dimensional Data. In *Proceedings of the Third International Conference on Computing, Mathematics and Statistics (iCMS2017)*; Springer: Singapore, 2018; Volume 1, pp. 95–102.
34. Taddy, M.; Chen, C.-S.; Yu, J.; Wyle, M. Bayesian and Empirical Bayesian Forests. *arXiv* **2015**, arXiv:1502.02312v2.
35. Kok, S.H.; Abdullah, A.; Jhanjhi, N.Z.; Supramaniam, M. A Review of Intrusion Detection System using Machine Learning Approach. *Int. J. Eng. Res. Technol.* **2019**, *12*, 9–16.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).