Automated Analysis Approach for the Detection of High Survivable Ransomware

Yahye Abukar Ahmed[1], Barış Koçer[1*], Bander Ali Saleh Al-rimy[2]
[1]Department of Computer Engineering, Selcuk University,
Konya 42075, Turkey
[e-mails: yahye@simad.edu.so; bariskocer@gmail.com]
[2] Faculty of Computing, Universiti Teknologi Malaysia,
Johor Bahru 81310, Malaysia
[e-mail: bnder321@gmail.com]
*Corresponding author: Barış Koçer

*Received July 7, 2019; revised October 15, 2019; accepted November 17, 2019;
published May 31, 2020*

Abstract

Ransomware is malicious software that encrypts the user-related files and data and holds them to ransom. Such attacks have become one of the serious threats to cyberspace. The avoidance techniques that ransomware employs such as obfuscation and/or packing makes it difficult to analyze such programs statically. Although many ransomware detection studies have been conducted, they are limited to a small portion of the attack's characteristics. To this end, this paper proposed a framework for the behavioral-based dynamic analysis of high survivable ransomware (HSR) with integrated valuable feature sets. Term Frequency-Inverse document frequency (TF-IDF) was employed to select the most useful features from the analyzed samples. Support Vector Machine (SVM) and Artificial Neural Network (ANN) were utilized to develop and implement a machine learning-based detection model able to recognize certain behavioral traits of high survivable ransomware attacks. Experimental evaluation indicates that the proposed framework achieved an area under the ROC curve of 0.987 and a few false positive rates 0.007. The experimental results indicate that the proposed framework can detect high survivable ransomware in the early stage accurately.

Keywords: Ransomware, supervised machine learning, Support Vector Machine, Artificial Neural Network, Term Frequency-Inverse document frequency.

http://doi.org/10.3837/tiis.2020.05.021 **ISSN : 1976-7277**

1. Introduction

Regarding cyber-attacks caused by the malware, the most wide-spread and sophisticated destructive is the one motivated by the ransomware [17], which is malicious software that prevents users to access the data on their computer, typically by encrypting the user's essential files or blocking the victim's computer and demands payment [2,4]. User's data access is permitted again if the victim paid the requested ransom using the anonymous currency mechanisms like Bitcoin [1, 10, 19]. The malicious action of the ransomware starts by tricking the users to download the payload. After malicious intent has been delivered to the victim's machine, the infection begins, and the payload executes. The malware encrypts the most important user's files on the hard drives, removable drives and mapped network shares for extortion [16]. After the encryption, the ransomware displays a message that requires payment to restore the captured user's data [5, 10]. The next step is to register the decryption key with a particular user and make available when the ransom is paid; therefore, ransomware uses the command- and- control(C&C) server to establish communication with its creator [15].

Although the revolution of ransomware appeared at the end of the 1980s [20, 12] when the PC CYBORG also known as Aids Info Disk (AIDS) Trojan starts to calculate the number of times the machine has booted until a criterion number (90) reached. After that, the Trojan AIDS locks the critical user's files, hides all directory and encrypts the labels of the files on the drive C: [3, 51]. However, the sequence of successful attacks of ransomware has resulted to increase many new ransomware variants in the last few years; for instance, the WannaCry cyber threat has been reported in 99 countries, and over 75,000 attacks have been carried out on machines running the Windows operating system [11]. The motivation is the significant revenue of the extortion, for example, effective ransomware like CryptoWall version 3.0 earned an estimated $325 Million as extortion in the USA alone [7, 9]. A report released by FBI just in 2016 estimated that the losses of $1 billion caused by ransomware [6]. The victims of ransomware are not only limited to home users or individuals but also targets government networks, businesses and health services. It causes damage to financial losses or sensitive information that can lead to the disruption of daily operations [2, 12].

In this study, we analyze and detect high survivable ransomware using machine learning algorithms of different integrated feature set. According to [10, 18] definition of high survivability described, "The ransomware has the high survivability property if it can maintain control over a critical host resource RC. Therefore, it grants access to RC solely when it is needed, and such that if the ransomware is modified or removed, RC is rendered permanently inaccessible and the decryption process can be completed only by the command and control server (CC) key while the ransom is paid." This means the effect of the attack forces the host to be dependent on the ransomware creators unless the attacker interferes (provides decryption key) the resources are inaccessible [18]. In summary, this paper presents three main contributions:

- We propose a framework for describing dynamically monitored valuable features of high survivable ransomware (HSR) by conducting a behavioural-based analysis of HSR within sandbox in an isolated environment, through the Term Frequency-Inverse document frequency (TF-IDF), we have extracted the most relevant features that provide the best performance in detecting new ransomware on windows platforms.
- We have developed detection models for HSR utilizing supervised machine learning algorithms on an integrated number of prominent features. The proposed method achieves

high accuracy and less false positive rate for detecting HSR in the early phases of the attack.
- We have empirically validated the method with an extensive experimental evaluation to show the effectiveness of the proposed framework. We also tested the ability of the proposed method by comparing the experimental results against with previous work, other classifiers and VirusTotal.

The *remaining* of this paper is organized as follows; the second section will discuss the infection vectors of ransomware. In section 3, we describe the related works of ransomware detection. In section 4, we introduce our proposed framework. In section 5, we highlight the experimental result of the research. Finally, in section 6, the evaluation and results of the proposed approach for the detection of ransomware are also discussed.

2. Infection Vectors

Ransomware creators use a range of different sophisticated techniques to spread their malicious intents; in this section, we highlight the most common ransomware propagation method such as:

- ***Spam Emails:*** The primary infection vectors for ransomware is through malicious spam emails, where the victim is tricked [12]. Opening a phishing email is an insufficient method to execute ransomware, but attackers still need users to download or open malicious attachments that directly install the ransomware; another way of the phishing email to deliver ransomware is to click on malicious links within phishing emails that appear to be a legitimate email message, the 93% of phishing attacks is ransomware purpose [31, 33].
- ***Exploit kits:*** Another common method for spreading ransomware is a toolkit that automates the exploitation of software vulnerabilities for distributing malware [26]. Most often, hackers inject malicious code on a website that redirects the victim to a malicious site [30]. The exploit kit identifies vulnerabilities in browsers; if it is vulnerable, it can leverage it to download ransomware [28]. Some ransomware variant such as wannaCry ransomware propagated through a dropper component named as EternalBlue that identifies vulnerabilities in the Server Message Block (SMB) protocol, which enables ransomware to drop binary onto all unpatched, vulnerable windows machine [29, 30].
- ***Social Engineering (SE):*** is the art of manipulating, persuading, suggestion and deceiving people to gain access to a user's computer [32]. It is an easier method that plays into human nature's inclination to trust or to carry out actions that grant the ransomware creators to access the victim's machine.
- ***Malvertising:*** is a type of online malicious advertising method [21, 37] used by attackers to inject malicious advertisements into trusted websites with many visitors [33, 18]. Often, when the user opens the website, there is no need to click on the ad; loading malvertising page will connect to several different URLs that lead to ransomware infection [34].

3. Related Works

The idea of employing symmetric key cryptography in the cyber extortion started in 1989 when the AIDS Trojan has begun to infect machine through floppies [13, 51]. The use of public key cryptography for extortion was first introduced in [18]. They have presented how cryptography can be implemented in ransomware through Trojan. The authors also proposed countermeasures to monitor the access of the cryptographic tools. Nevertheless, this preventive approach is unable to detect the advanced ransomware variants. Kharraz et al. [7]

proposed a method to monitor the Master File Table (MFT) for activity and sorts of I/O Request Packets (IRP) of the file system to detect zero-day ransomware attacks. They suggested the mitigation strategy of employing the decoy technique to detect the maliciousness of the file. However, it is not clear whether the normal user would access the decoy resource before the attack occurs. Later work, they enhanced in [8] to introduce a new method called Unveil that is designed to detect the attack when ransomware tampers the user's data, typically by creating a fake user environment. This approach is able to protect the user's files. However, the victim should sacrifice some data before the UNVEIL identifies the attack. The detection of high survivable ransomware was first proposed by Ahmadian et al. [5], the authors implemented 2entFOX framework that extracts 20 static and dynamic features and their statistical possibilities. For classification case, they applied the Bayesian belief network to detect high survivable ransomware. However, the detection rate of this method was low due to the high dimensional feature space used. Microsoft's Cryptographic API (MS CAPI) calls were presented by the Young [23] explaining the method to encrypt the sensitive user's data and to produce the key by using MS CAPI with eight types of API calls. Similar work presented by palisse et al. [2], which is a detection mechanism that enables users to decrypt their files by getting the advantage of the weak chaining mode that used by some ransomware with cipher algorithm. Authors also propose another detection method based on the intercept calls used by Microsoft's Cryptographic API. However, the proposed countermeasures are insufficient to detect other types of ransomware that use Cipher Block Chaining (CBC) mode. In addition, the protection is implemented, after the files are encrypted.

Network and file integrity monitor called tripwire was presented by Ben22 [24] when critical system files are modified, it gives an alarm to the administrator. These monitors are based on witness files and simple hash comparisons, the LanmanServer operation is denied if the witness files are altered. However, this approach cannot guarantee the changed witness files are modified by a malicious program or a normal user. An alternative method [22] presented HelDroid, automated approach that classifies known and unknown mobile ransomware and scareware using a machine-learning method. Their approach is based on detecting threatening text associated with a ransom note and the "building blocks" that are typically needed to implement a mobile ransomware application.

Recently, machine-learning methods were addressed for classification of ransomware and benign application. Early detection of crypto-ransomware was introduced by Sgandurra et al. [38] to identify a set of characteristics of ransomware that captured in its early phases of ransomware at run-time, authors proposed EldeRan, a framework to observe some unique actions performed by ransomware to dynamically analyse features that support ransomware detection. Authors selected the informative binary features using Mutual information criteria and then applied a machine-learning algorithm like Regularized Logistic Regression classifier that achieved 96.3% detection rate with an area under the ROC curve of 0.995%. They assigned a threshold of 30 seconds for the sample to execute. However, setting a fixed time is not applicable to all ransomware samples, since some variants exhibit their malicious activities after human interaction or discovering the executing environment. Takeuchi et al. [52] proposed a scheme to detect ransomware using support vector machine (SVM) with API calls history. The samples are executed in a controlled environment to monitor the behaviour of the program. They deeply examined the sequences of API calls by creating a standardized vector representation of q-grams extracted from the output logs. The SVM-based scheme showed an accuracy of 97.48%. A work presented by Alhawi at al. [53] introduced the NetConverse, a supervised machine learning approach to detect ransomware using conversation-based network traffic features. They analysed 9 ransomware families, extracted 13 features using

TShark and feed 6 classifiers such as Bayes network (BN), Decision Tree (J48), K-Nearest Neighbors (IBK), Multi-Layer Perceptron, Random Forest and Logistic Model Tree. The highest accuracy performed Decision Tree (J48) classifier that showed 97.1% of detection rate with less positive. Qian and Bridges [39] used an automated method for the extraction of ransomware feature from the sandbox output logs, they analyzed WannaCry ransomware and two polymorphic samples in isolated environment. To rank the most significant ransomware features, term frequency-inverse document frequency (TF-IDF) is used to weight the 74 features from the generated behavioral logs and the top 43 informative features are selected to discriminate the malware from benign samples. They claim that the method can accurate extract features from the logs, and the TF-IDF approach provides deeply analysis of WannaCry malware than other extraction algorithms. Nevertheless, the number of used WannaCry samples cannot describe the characteristics of ransomware. Similarly, an interesting behavioural early detection framework is proposed in Bander et al. [14] to detect zero-day crypto-ransomware using machine learning techniques with data-centric and semantic features. The detection module of the framework contains behavioural and anomaly detection scheme to improve the accuracy of the detection in the early stage before the encryption is carried out. However, the proposed framework was not implemented empirically.

On the other hand, instead of using dynamic analysis, a recent work of Zhang et al. [50] investigated the opcode sequences feature for detection and classification of ransomware static analysis approach to map ransomware into families. Author extracted opcode from ransomware samples created *N-grams* sequences and calculated for each *N-grams* using term frequency-inverse document frequency (TF-IDF) to select the informative features between families. Then, applying five machine-learning algorithms achieved the best accuracy of 91.43%. Similarly, Poudyal at al. [54] extracted assembly and dll level of ransomware binaries statically to perform multi-level analysis, then cosine similarity is used to a measure the similarity between these binaries. Eight supervised machine learning classifiers are employed to classify ransomware and benign sample. The proposed a framework achieved an accuracy of 97.95% when both extracted features are combined. However, the sophisticated packing techniques used by newly emerged ransomware can easily evade the static analysis. Furthermore, is not efficient for early ransomware detection since there is no need to execute the malware samples during the static method, while some variants exhibit their malicious activities on the runtime [12].

4. The Proposed Framework

In this section, we present our proposed framework for identification and detection of high survivable ransomware. To make our methodology visual and understandable, we propose a methodology framework that consists of three main phases. Data collection and preparations phase that includes obtaining ransomware and benign dataset from a variety of sources, checking whether datasets are malware or not and vice versa and labelling the malware family using VirusTotal service. The second phase is to analyse samples using Cuckoo sandbox that generates JSON format report. The collected behavioural log files are passed to pre-processing tasks such as removal of duplicate files, file type identification, and parsing. The relevant features are extracted from the analysis file logs to get valuable feature sets. We have applied the term frequency and inverse document frequency (TF-IDF) algorithm for feature selection. (iii) Finally, supervised machine learning algorithms were implemented for the classification of ransomware and benign sample.

5. Experiments

In this section, we demonstrate the experimental design of the proposed framework and describe the method of the behavioural analysis approach in the sandbox. We also present the dimensionality reduction of the features for training and testing purpose. The model development and measuring the performance of the supervised machine learning algorithms are also discussed in the following sections.

5.1 Experimental Setup

To gain an in-depth behavioural analysis of ransomware requires executing samples in a controlled environment. Therefore, we built our malware analysis lab following the best practices suggested in [25, 27]. Cuckoo Sandbox is used, a well-known leading open source tool to automate malware analysis [27]. Ubuntu 16.04 LTS Desktop fully updated was our host operating system while installing Cuckoo sandbox. WindowsXp_server_Pack3 32bit was selected as a guest machine due to its weaker security protections that enable us to observe more ransomware behaviour. To perform the analysis in a secure, Virtual box machine was used with controlled access to the Internet -host-only adapter- to enable commands and controls (C&C) communication, and to prevent the spread of ransomware. Anti-virus, security updates, firewall, and user account control of windows XPSP3 guest were disabled to execute ransomware successfully. Some commonly third-party applications such as Microsoft Office, Acrobat Reader, Google Chrome and Mozilla Firefox was installed in the guest operating system. Python agent was also installed that runs inside the guest and acts as cross-platform for communication and the exchange of data between cuckoo and the guest OS. Finally, in Windows XP several normal user files inside directories (e.g., My Documents, My Pictures and Videos, valid browsing history) was created to observe the behaviour activities of ransomware.

5.2 Dataset Description

The data set for this study consists of ransomware and benign. We collected 1,254 ransomware samples of 14 different families from several sources such as VirusShare and VirusTotal, - which are publicly computer virus repositories on the net-, we crawled malware repositories and online forums that share samples. We also downloaded and collected 1308 benign applications that hosted from the most trustworthy sources such as software.informer, and system files located in the “System32” directory of a fresh installed Windows 7 Professional. To build a realistic dataset, we used in our experiments benign applications that have ransomware behaviour as shown in **Table 1**. The acquired samples are stored in separate files on both malware and benign group. To verify that the downloaded benign applications do not contain malicious components inside their payload, we double-checked the MD5 hash values from Virus Total service that has 57 common different antivirus software. To obtain the exact family name of ransomware is a very challenging task especially when you have a large number of malicious files. We applied Antivirus vendors’ labelling scheme in terms of the popularity of ransomware classes among Antivirus Engines. The general problem we encountered is mislabelling some samples by antivirus engines as specific ransomware family. Therefore, we parsed the labels by the set of AV engines that commonly used to assign malware labels using python script with a threshold value of 85 that aggregated the labels from the pool of AV in VirusTotal repository. We consider ransomware to be a specific family if 85% of AV engines described it as belonging to this family name.

5.3 Automatic Behavior Analysis

In this work, an automated dynamic analysis was used to analyse the samples in an isolated environment. Although the total original dataset was 2562, after removing samples that did not execute correctly, or cuckoo terminated the analysis because of the maximum timeout that we set samples to run, or those that many AV assigned different ransomware family names, 673 ransomwares from 14 distinct families and 742 benign samples were analysed. Every sample up to a range of 4 until 9 minutes were analysed to show to their malicious behaviour and to capture the execution traces of the samples using a cuckoo sandbox, while the ransomware sample is running on the host. Cuckoo monitored and recorded information in terms of the API calls, network traffic, changes of files and folders, processes and memory dumps. We used virtualization software to take a snapshot of the guest machine before the execution of each malware sample, after execution, the entire system was reverted to a previous clean state before the infection. Some ransomware samples wait for human interaction like mouse or keyboard event before executing their malicious activities, thus, we used a python script that performs basic user's activity such as browsing websites, clicking and deleting documents and folders on the desktop. During the analysis of the samples, we observe ransomware variants used both symmetric and asymmetric algorithms to encrypt the user's data. Crypto ransomware creates a randomly symmetric key with AES algorithm in victim's machine and then encrypts files along with that generated key. After encrypting the data, it encodes the secret key with asymmetric encryption. At the end of execution, the time taking ransomware samples to encrypt files is variety ranging from 27 seconds like Petya up to 2 minutes for CryptoWall.

Table 1. Distribution of malicious and benign files

Ransomware Class	**Samples**	**First Seen**	**Goodware Class**	**Application Name**	**Samples**
WannaCry	74	2017	Compression	Winzip, 7-zip, WinRAR, PeaZip, IZArc	225
Reveton	50	2012			
Torrent Locker	108	2012	Encryption	BitLocker, Disk Cryptor, VeraCrypt , TrueCrypt	172
Dirty Decrypt	51	2015			
CryptLocker	173	2013	Data Destruction	CBL Data Shredder, HDDErase , MHDD, PCDiskEraser, KillDisk , SDelete	401
Cerber	171	2016			
Trojan	82	2013			
Kollah	73	2014	Drivers Updater	Driver Booster, DriverPack Solution, DriveTheLife	230
Citroni	67	2015			
Pgpcoder	46	2015	Browsers	Chrome, Firefox, Opera ,Safari , Netscape, Internet Explorer	152
Kovter	23	2013			
Petya	89	2016	Multimedia tools	Canva, Animoto, Photopeach , Picasa, Livestream	182
CryptoWall	151	2014			
TeslaCrypt	96	2015	Others		96
Total Samples	**1254**				***1308***

5.4 Feature Extraction and Selection

Once the analysis is completed, cuckoo generated human-readable JavaScript Object Notation (JSON) report for each analysed malware sample. In this study, the most time

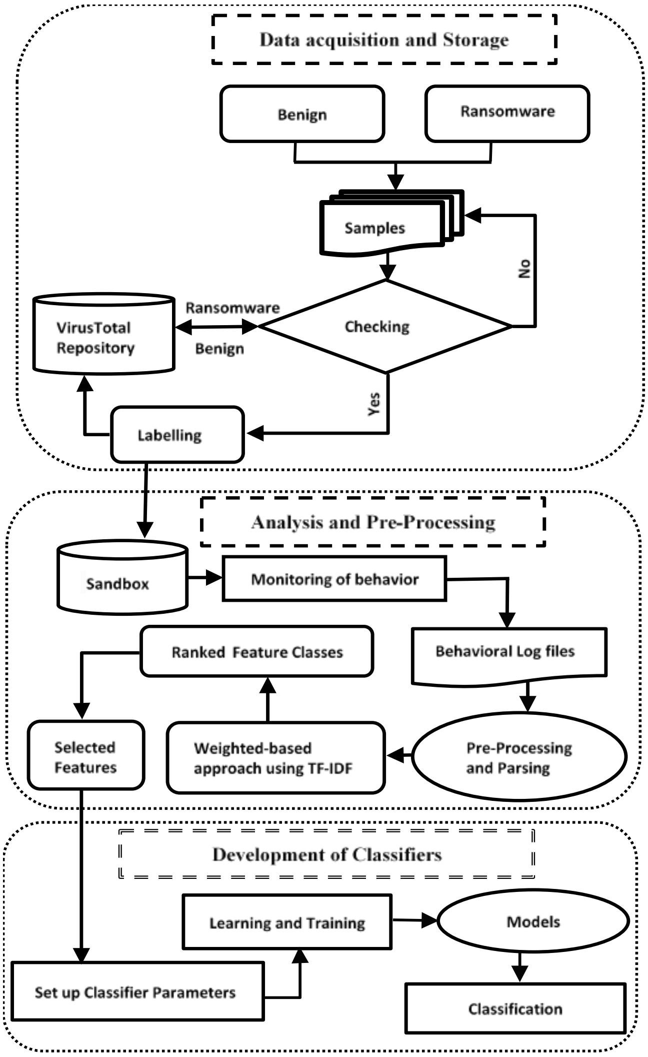

Fig. 1. Proposed framework architecture

dedicated to the extraction of the indicative and accurate behavioural features from JSON report, which is not an easy task. After we collected the results of the analysis, we need to retrieve the key elements from the JSON reports such as SHA1, MD5, and API, etc. The size of the report generated by the sandbox occupies hundreds of MBs, analysing and examining each report manually is experimentally infeasible, therefore, we build our own parsing algorithm to convert JSON formatted string representations to key-pair objects. The feature parsing reads the JSON files from all sandbox output reports and then parsed to get the required features to reduce the search space. The feature parser functions as structure to correlate a ransomware sample's feature calls into states. The parser maps the Registry paths, windows API calls, files Operation, Strings, Directories, Drops and libraries into seven different states. Every state represents the presence or the absence of that specific call for this feature. The Feature parser creates a matrix containing the feature and its states. For instance, in ransomware phases, when the attached completed, the ransomware deletes all the original victim's data while keeps the encrypted one, in this case, RegDelete method is used. So, the parser creates a matrix by investigating whether this specific registry key operation was performed or not. The total number of features was 13, 631 and this number of features is too large for processing and feeding to classification algorithm, therefore, we selected 3930 as prominent features as expressed in **Fig. 2**. Selecting the most relevant subset features from the original features can improve classifier performance and the accuracy of classification operation [41, 45, 40]; hence, the effective feature set was identified using term weight as the criterion of feature selection. We applied term frequency-inverse document frequency (TF-IDF) feature selection method for setting the weight to a term based on its inverse document frequency and evaluating how important feature is a document in the collection [36]. The purpose of using TF-IDF weighting is to eliminate those features that occur commonly in many vectors while giving more attention to features that are less frequent in the vectors. The formula for the TF-IDF expressed as follows:

$$\mathrm{W_i} = \mathrm{TF}(\omega_i, \mathrm{d}) \times \mathrm{IDF}(\omega_i) \quad (1)$$

Where W_i is the weighting scheme of word ω_i in document $d \in D$, and TF (ω_i, d) is the frequency of term of ω_i in document d, and IDF (Inverse Document Frequency) is then defined as:

$$IDF(\omega_i) = \log(\frac{|D|}{DF(\omega_i)}) \quad (2)$$

Where DF (ω_i) represents the appearance of $\underline{\omega_i}$ in a document D. After experiments and study of many technical reports, seven feature classes were extracted as shown in **Table 2** with a brief explanation.

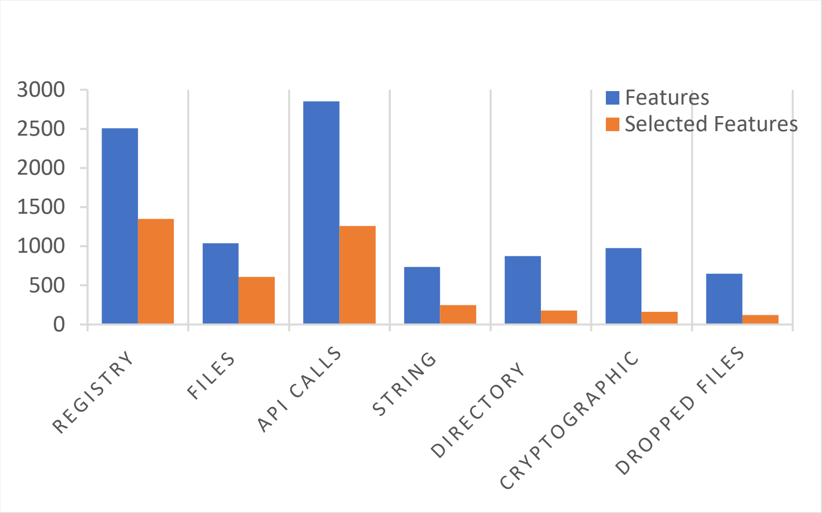

Fig. 2. Number of extracted and selected features

We observed in our experiments that the highest scores counted by TF-IDF are Registry Keys and API Stats. These are the two most indicative among all other feature classes. Dropped files feature are scaled down due to some normal operations frequently occur in the entire analysed log files.

Table 2. Weights of extracted feature classes using TF-IDF algorithm

#	Feature Classes	Analysis Type	TF-IDF Score	Feature Class Description
1	Registry Paths	Dynamic	2.682	Registry key operations such as registry keys opened, read, written and deleted
2	Windows API Calls	Dynamic	2.596	Windows API calls the traces of invocations of native functions and API calls
3	Files Operations	Dynamic	1.583	File operations such as read, open, write and delete operations
4	Printable String Information (PSI)	Static	1.452	İs a sequence of characters that provide hints about the functionality of a program
5	Directory Operation	Dynamic	1.420	Operations performed on a directory
6	Cryptographic Libraries	Dynamic	1.391	Contains and implements several popular cryptographic algorithms and standards.
7	Dropped Files	Dynamic	1.290	During installation application dropped set Extensions of files

5.5 Classification Methods

The last phase of this study is to distinguish malicious files from benign files utilizing supervised machine learning algorithms like Support vector machine (SVM) and Artificial Neural Network (ANN). since we have large number of features, linear classifier is considered to be a valuable option. In addition, SVMs has higher generalization capability than the other machine learning algorithms with regard to small training dataset. Through the detection phase, the unseen file can be divided into ransomware and benign files [42, 48]. We first applied Support Vector Machine (SVM) that sorts given data into one of two classes, The SVM makes a classification by finding a linear hyperplane separating given vectors into two given categories [43, 47]. A hyperplane is described as $\boldsymbol{w \cdot x_i + b = 0}$, where $\boldsymbol{x_i}$ is defined as a point on dimensional space, w is normal to the line where ***b*** is the bias weight that describes the interval between origin and the hyperplane. The separating line (hyperplane) is described as two classes in the form of:

$$w \cdot x_i + b \geq +1, \ \text{for}\, y_i = +1; \tag{3}$$
$$w \cdot x_i + b \leq -1, \ \text{for}\, y_i = -1. \tag{4}$$

The above two inequality equations (3) and (4) are merged into the form of a single inequality in equation (5) as follows.

$$y_i(w \cdot x_i + b) - 1 \geq 0 \tag{5}$$

The points of the two hyperplanes can be described as $\boldsymbol{w \cdot x_i + b = \pm 1}$. Thus, the normal form of SVM classification decision function is defined as follows [45], [46]:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{r} \alpha_i y_i K(x, x_i) + b\right), \tag{6}$$

Where α_i, $i = 1,..,r$ are Lagrange multipliers and $K(x, x_i)$ is defined as a kernel function. The magnitude of α_i determined by parameter C. SVM is powerful and easy to train, so it is no need for local optimal. The trade-off between the error rate and the classifier complexity can be managed simply [44].

This study, Multilayer Perceptron (MLP) Network is used with the backpropagation algorithm to train the model. Multilayer perceptron (MLP) is a hierarchal structure of many perceptron consists of many hidden layers in the network architecture [49]. The activation function used was the sigmoid, a real function sc: IR → (0, 1) defined by the expression (given by Eq. 7), and the input to the hidden node can be expressed as in equation 8:

$$s_c(\mathrm{x}) = \frac{1}{1 + e^{cx}} \tag{7}$$

$$sum = W_1 \, x_1 + W_2 \, x_2 + W_0 \, B_0 \tag{8}$$

The hidden nodes have a significant role in the network performance, so the above equation of the hyperplane can be combined into a form of a one-line equation:

$$X_2 = \frac{w_1}{-w_2} x_1 + \frac{w_0 B}{-w_2} \tag{9}$$

5.6 Performance Criteria

In this section, the classifier performance is evaluated using standard accuracy measurement. The best SVM and ANN models among the tested models are compared. The evaluations of these models based on their classification measurement such as True Positive Rate (TPR), is the case in which the proportion of positive samples, like ransomware that is identified correctly as shown in equation (10). False Positive Rate (FPR) is the case in which the

Algorithm 1: Extracting Features from JSON Report

```
Input: Set of JSON report path $J_R$ that contains a number of behavioral and static features $f$.
    Output: Parsed files
        Function ObtainFeature ($S_{data}$, states) {
            for process in json_data['behavior'] ['processes'] do
             if json_data_process is equal to states then
             set first_seen_temp=process_first_seen
                if first_seen is greater than first_seen_temp or equal to zero
                  set first_seen= first_seen_temp
                for features in json_data_process[F] do
                  if features [F] not in our_Dictionary_features then
                     set our_Dictionary_Features [F] and timestamps = f and
                     Feature_time
                     Our_Dictionary_ Features [F][count]=1
                  Else set our_Dictionary_ Features and timestamps= Features_F_time
                  and append_F
                  our_Dictionary_Features [F][count]+1, return first_seen,
                  our_Dictionary_Features [F]}
         Function Json_Files (JR, PR )
            for (JR, PR) in G_file() do //traverse the file names in a directory tree
            for i, name in enumerate [all_files] do
                   If name ends with ('json') then
                      Fname= initialize files that matches (name)
                      Open the json data ($J_R$ + $F_{name}$) as ($json_f$)
                      Set $S_{data}$= load ($json_f$)
                      Call the function of ObtainFeature ( $S_{data}$ ,true )
                   Print(PR)}
        In the main function {
            Input $J_R$ ← parse_directory//initialize the director to be parsed
            Input PR ← F   // set the place where the result will be stored
            Store the user's input in the ($J_R$, PR ) variables
            Json_Files( JR, PR )
              if name is equal to main then
                exit the system
```

proportion of negative instances wrongly identified as positive as shown in Equation (11). True Negative (TN): is the case in which samples are correctly classified as benign programs. False Negative (FN): is the quantity of number that are misclassified malicious programs.

$$TPR = \text{Sensitivity} = \frac{|TP|}{|TP|+|FN|} \tag{10}$$

$$FPR = Specificity = \frac{|FP|}{|FP|+|TN|} \tag{11}$$

The Total amount of accuracy is the ratio of properly identified samples, either negative or positive, divided by the total samples as defined in equation (12).

$$Total\ Accuracy = \frac{|TP|+|TN|}{|TP|+|FP|+|TN|+|FN|} \tag{12}$$

The total accuracy of the generated classifier determines the effectiveness and performance. Another method of identifying the performance of the classifier is the use of the ROC curve which is points of a plot that shows the trade-off between a classifier's FP rate and its TP rate.

6. Evaluations and Results

This section describes the evaluation and the performance of SVM and ANN across different experiments. Creating SVM and ANN models require to set various parameters and test individually to select the best model among them. In this study, SVM kernel functions such as the linear kernel, polynomial kernel and Radial basis function (RBF) are used. The parameters values for kernel functions can have an extreme effect of the model's performance and the generalization error. Therefore, we set the parameters of these kernel functions with the incremental regularization parameter $\boldsymbol{\lambda}$ and the cost parameter of $\boldsymbol{C}$. These selection parameters require an exhaustive repetitive search over the parameter space to find the best settings. On the other hand, user-defined neural network parameters such as hidden layers, momentum and learning rate were assigned values to test the network performance. Learning rate parameter is the specified user value that controls the step size when weights are iteratively adjusted. Hence, in this study, four values of learning rate are picked: 0.1, 0.3, 0.6 and 0.9. Finally, this study created many different models along with a variety of outputs. In the following subsection, three different experiments are conducted to train and test the classifiers.

6.1 Train-test splitting method

The purpose of this experiment is to evaluate the performance of the proposed integrated features by employing a train-test split method, which is dividing the whole data set into two subsets: training and testing data. first, we split randomly our dataset with a uniform distribution of 80: 20% ratio as training and testing respectively. The experimental results of ANN showed an accuracy of 0.958 with 0.101 false positive rates while SVM presented higher false positive of 0.109 compared to ANN and the accuracy of 0.932. The ROC curve of this experiment is presented in **Fig. 5**, and the **Table 3** shows the results of the FPR, TPR, AUC, precisions and the recalls and the accuracy of the classifier based on the training and testing splitting method.

6.2 10-Fold Cross-validation Method

In the train-test splitting method, once the data set is divided into a ratio that does not relevant each class of the experimental samples, the result of the holdout error rate will be inaccurate. To overcome this limitation, we applied the10-Fold cross-validation technique to prevent the overfitting problem and to estimate the effectiveness of our models. In this approach, the entire data set was randomly shuffled and divided into 10 equal-sized of subsets such that, each repetition (10-fold) we build our model with *10-1* folds of the data set for the evaluation of the trained model and the remaining one-fold constitutes for testing.

Table 3. Result of the train-test splitting method

	FP Rate	**TP Rate**	**Precision**	**Recall**	**AUC**	**Detection Rate**
SVM	0.109	0.853	0.923	0.926	0.904	0.932
MLP	0.101	0.956	0.945	0.951	0.965	0.958

Table 4. Result of the 10-fold cross validation method

	FP Rate	**TP Rate**	**Precision**	**Recall**	**AUC**	**Detection Rate**
SVM	0.035	0.962	0.945	0.942	0.982	0.952
MLP	0.036	0.982	0.931	0.932	0.971	0.945

In this experiment, we have evaluated the performance of classifiers using 10-fold cross-validation to train and test the algorithms. The results achieved by the classifiers in this experiment on the whole dataset were quite satisfactory. The best accuracy reached SVM by presenting 0.982 of AUC with less than 0.035 of false positive rate. It is important to examine the ability of the classifiers for the distinguishing the ransomware from benign samples, therefore, precision and recall are applied to both datasets and presents 0.945 and 0.942 respectively. SVM also shows fairly better accuracy of 0.952 comparing to MLP that shows 0.945 of detection rate and 0.036 of the false positive rates as presented in **Fig. 6** and **Table 4**. This indicates that SVM has super generalization ability and is quite tolerant for training the iteration of 10-fold set size.

6.3 Testing with selected subset features

The aim of this experiment is to evaluate how selected subset features can effectively contribute to the performance of the algorithms. Selection of subset feature eliminates the redundant and irrelevant features and reduces the dimensionality of the dataset. In this experiment, we divide our features into seven subset features by considering their importance and ranking based on the aforementioned feature selection algorithm presented in Section 5.4. We created the most prominent features as Top-N feature set: top20, top30, top40, top50, top60, top70, and top80. The experimental results demonstrated that ANN showed the highest accuracy of 98.79% when top30 of the feature set was used as training and testing. However, this classification accuracy had dramatically decreased to 95.63% when top20 of the feature set was used. On other hands, the best model of SVM presented an accuracy of 97.6% when top40 of the feature was applied for training the model. Although SVM performed ratio of 0.993 of TPR and 0.0371 of FPR, this indicates that SVM has a higher ratio of false positive rate compared to ANN. The experimental result implies the importance of considering the selection of different subset features. The following **Fig. 3** compares SVM and ANN classification accuracy across different subset features. By inspecting **Fig. 3**, both ANN and SVM had low classification accuracy when top80 of the feature set used to train and test the model. This indicates that more features do not improve the performance of the classifiers as **Table 5** shows the results of each classifier based on selected features.

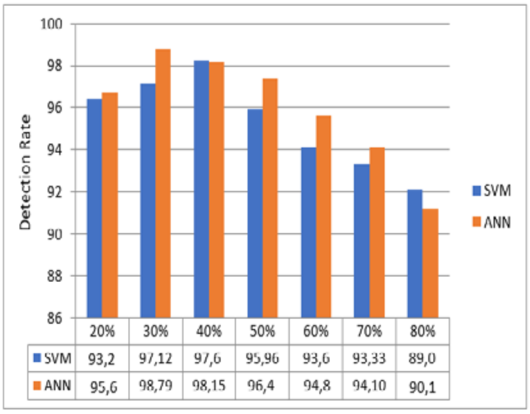

Fig. 3. Comparison of SVM and ANN classification accuracy

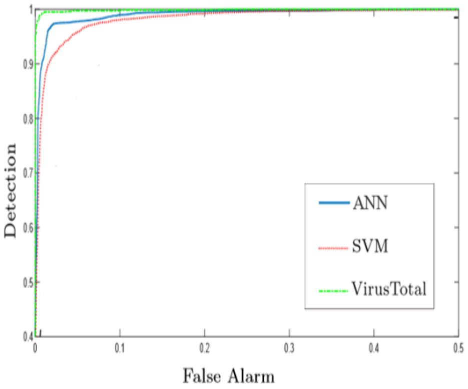

Fig. 4. Comparison of the proposed method to VirusTotal

6.4 Comparison with other Classifiers

In this experiment, we evaluate the performance of the proposed method with three classifiers such as K-Nearest Neighbor (KNN), Decision Tree and Random Forest (RF). We tested the most informative features selected by the aforementioned feature selection algorithm presented in Section 5.4. we explored the highest top features set with a testing 10-Fold cross validation techniques.

Table 7 compares the accuracy of each classifiers trained and tested with the selected top features. The ANN has performed the highest accuracy among all classifiers by presenting 0.986 of accuracy and less false positive rate. The RF classifier presented the lowest accuracy among the classifiers and shows 0.798 of accuracy, this poor performance is probably due to the overfitting problem during the training that the model is unable to generalize the new features in the testing phase. On other hands, we evaluated the performance of the classifiers in terms of values of the Area under Curve (AUC) as evaluation metrics in this experiment. The **Fig. 7** illustrates the AUC variations for the five classifiers. Three classifiers (ANN, SVM, and DT) have high AUC rate while other two classifiers (kNN and RF) have slightly lower AUC rate.

Table 7. Comparison of our proposed approach with other Classifiers

	FP Rate	TP Rate	Precision	Recall	AUC	Detection Rate
kNN	0.246	0.812	0.812	0.802	0.823	0.834
ANN	0.0261	0.986	0.981	0.979	0.983	0.986
SVM	0.016	0.986	0.976	0.971	0.973	0.979
RF	0.284	0.747	0.751	0.747	0.786	0.798
DT	0.061	0.945	0.945	0.944	0.946	0.952

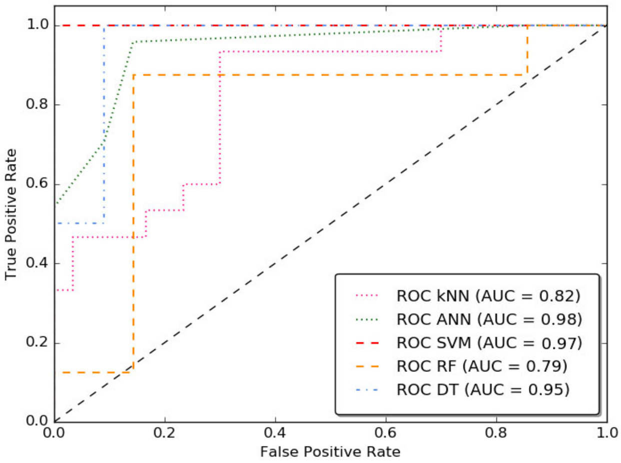

Fig. 7. Roc Curve Comparison of the proposed method with other classifiers

6.5 Comparison with AV Scanners

Although our proposed method shows a better result, we also need to examine the capabilities of detecting the variants of known and unknown ransomware. We measure our classifiers through effectiveness and accuracy criteria and compare the performance of our proposed method with anti-virus engines. For comparative benchmarks, we selected the five Anti-Virus (AV) scanners with the highest detection rate available at VIRUSTOTAL service. So, the detection performance of each AV scanners is evaluated using standard accuracy measurements such as True Positive Rate (TPR), False Positive Rate (FPR), ROC curve and the detection rate.

Table 5. Shows FPR, TPR, AUC and accuracy for SVM and ANN with different subset features

	Support Vector Machine				**Artificial Neural Network**			
	FP Rate	*TP Rate*	*AUC*	*Det. Rate*	*FP Rate*	*TP Rate*	*AUC*	*Det. Rate*
20	0.371	0.625	0.948	0.932	0.033	0.952	0.972	0.956
30	0.041	0.959	0.976	0.971	0.007	0.988	0.986	0.987
40	0.006	0.993	0.977	0.976	0.012	0.987	0.982	0.981
50	0.071	0.935	0.974	0.959	0.035	0.962	0.985	0.964
60	0.160	0.839	0.951	0.936	0.035	0.964	0.978	0.948
70	0.041	0.958	0.973	0.933	0.034	0.951	0.980	0.941
80	0.103	0.837	0.238	0.891	0.036	0.841	0.186	0.901

From the experimental results, we observe that ANN significantly outperformed the other approaches and showed 0.986 of AUC, it also presented TPR of 0.988 and FBR of 0.036, despite ANN's false positive rate is higher than 3 of the AVs as shown in **Table 6**.On other hands, the VirusTotal achieved the highest accuracy over the SVN and ANN algorithms with

Table 6. Comparison of our proposed approach with AVs

	TP Rate	FP Rate	AUC	Detection Accuracy
SVM	0.0993 ± 0.0837	0.0371 ± 0.006	0.977 ± 0.238	0.9760±0.8910
ANN	0.0988 ± 0.0841	0.0360 ± 0.007	0.986 ± 0.186	0.9870±0.9010
AV1	0.0203 ± 0.0079	0.0186 ± 0.0080	0.977 ± 0.238	0.989 ± 0.0273
AV2	0.0159 ± 0.0060	0.0166 ± 0.0048	0.986 ± 0.186	0.986 ± 0.0262
AV3	0.0274 ± 0.0082	0.0396 ± 0.0080	0.977 ± 0.238	0.9569 ± 0.0173
AV4	0.0205 ± 0.0079	0.0000 ± 0.0000	0.986 ± 0.186	0.9369 ± 0.0173
AV5	0.0101 ± 0.0079	0.0496 ± 0.0080	0.977 ± 0.238	0.9160 ± 0.0273

Table 8. Comparing the proposed approach with earlier work

Works	Classifier(s) Used	Detection Rate
Proposed work	SVM and ANN	0.986
Sgandurra et al [38], EldeRan Framework	Logistic Regression	0.963
Mahbub et al [55], RansHunt framework	SVM	0.971
Ahmadian et al. [5], 2entFOX Framework	Bayesian belief network	0.985
Alhawi at al. [53], NetConverse scheme	BN, J48, kNN, MLP, RF and LMT	0.971
Zhang et al. [50].	DT, RF, KNN, NB and GBDT	0.914
Poudyal at al. [54].	BN, LR, SVM, DT, RF and ADA	0.965

a detection rate of 0.989 and 0.986, though ANN classifier outperforms 3 out of 5 top selected AV scanners. This is due to the most common detection method used by the antivirus is the signature-based detection; which implies that VirusTotal AV engines already have a matched signature of these datasets. In addition to, some of our data set are publicly available for a longer period.

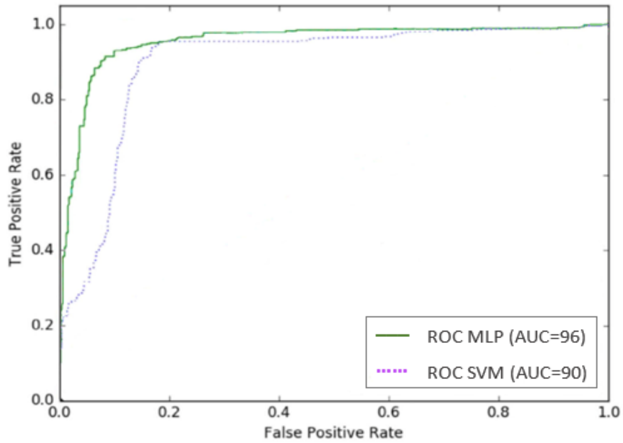

Fig. 5. ROC curve of the classifiers on train-test splitting method

The **Fig. 4** shows the comparison of the detection rate and false alarm of the VIRUSTOTAL and our method using both ANN and SVM. From the result of **Fig. 4**, it is obvious that the AUC of VIRUSTOTAL outperforms ANN and SVM algorithms, however, according to the false alarm criteria, the VirusTotal shows an average of 5.4% and is worse than ANN classifier that presents an average error rate of 2.3%. ANN provided a better accuracy regarding SVM with 4.4% error rate.

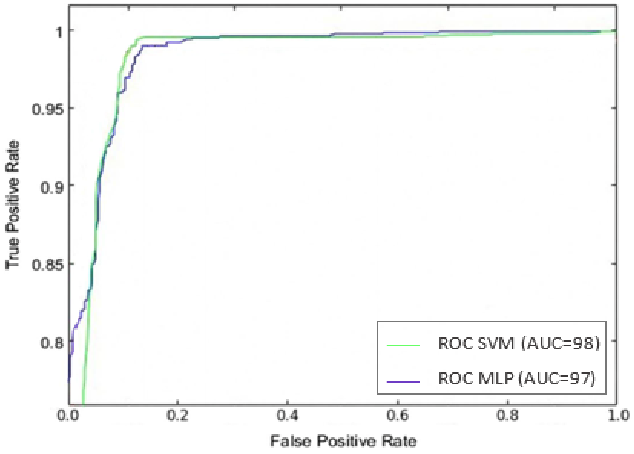

Fig. 6. ROC curve of the classifiers on the 10-Fold validation method

6.6 Comparison with previous work

The purpose of this section is to compare the performance of the proposed method with the previous similar works based on feature type and the classifier used. Sgandurra et al [38] proposed a machine learning based framework with integrated features to identify the characteristics of the ransomware in the earlier phases, authors analyzed and extracted seven different features dynamically, they applied Logistic Regression classifier that achieved 96.3% detection rate with an area under the ROC curve of 0.995%. Similar work was presented by Mahbub and Mahbubur [55] using integrated features from static and dynamic analysis with machine learning algorithm. Authors proposed RansHunt framework to detect ransomware using support vector machine (SVM) with unique features. They claimed the hybrid analysis method can early detect the new ransomware variants. The RansHunt framework achieved an accuracy of 97.10 with less positive rate. A work of Zhang et al. [50] employed the opcode sequences features with five machine-learning algorithms for detection of ransomware. The classifier showed an accuracy of 91.43%. The Table 8. summarizes the comparison result of our proposed method against with similar works.

7. Conclusion

In this paper, we proposed a behavioural malware detection framework for high survivable ransomware (HSR) using a machine-learning approach. We performed an automated dynamic behavioural analysis for 673 real-world ransomware samples that infect Windows platforms. We focused on the malicious behaviours of 14 newly emerged ransomware families. The activities performed by the malicious program is recorded in the sandbox in a controlled environment and obtained generated report of the samples as JSON format. Moreover, the key observation of this research is to investigate an integrated set of features that could indicate what the ransomware strain is actually doing on the system. The TF-IDF algorithm has shown to be an effective approach for ranking and weighting behavioural features. We suggested seven integrated informative feature classes that can reduce the time cost for training machine-learning algorithms. We developed a detection model for HSR by utilizing Support Vector Machine and Artificial Neural Network algorithms using integrated valuable features.

Three different experimental evaluation was conducted to measure the performance of the proposed method. Through our experimental results, the proposed approach has shown to be easy to train and test and achieved a detection accuracy of 98.7 with few false positive rates below 3%. To empirically evaluate the proposed approach, we compared the experimental results with previous work, other classifiers and the VirusTotal service.

References

[1] Ezhil, Kalaimannan, Sharon K. John, Theresa DuBose, and Anthony Pinto, "Influences on ransomware's evolution and predictions for the future challenges," *Cyber Security Technology*, vol. 1, no.1, pp. 23-31, November, 2017. Article (CrossRef Link)

[2] Aurélien Palisse, Hélène Le Bouder, Jean-Louis Lanet, Colas Le Guernic, and Axel Legay, "Ransomware and the legacy crypto API," in *Proc. of Int. Conf. on Risks and Security of Internet and Systems*, pp. 11-28, September 5, 2016.Article (CrossRef Link).

[3] Alexandre Gazet, "Comparative analysis of various ransomware virii," *Computer Virology*, vol. 6, no.1, pp.77-90, February, 2010. Article (CrossRef Link).

[4] Daniel Morato, Eduardo Berrueta, Eduardo Magaña, and Mikel Izal, "Ransomware early detection by the analysis of file sharing traffic," *Journal of Network and Computer Applications*, vol. 124, pp.14-32, September 21, 2018. Article (CrossRef Link).

[5] Mohammad Mehdi Ahmadian, and Hamid Reza Shahriari, "2entFOX: A framework for high survivable ransomwares detection," in *Proc. of 13th Int. Iranian Society of Cryptology Conf. on Information Security and Cryptology (ISCISC) IEEE*, pp. 79-84, September 7-8, 2016. Article (CrossRef Link).

[6] Chris Moore, "Detecting Ransomware with Honeypot Techniques," in *Proc. of 2016 Cybersecurity and Cyberforensics Conf. (CCC)*, pp. 77-81, August 2-4, 2016. Article (CrossRef Link).

[7] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *Proc. of Int. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 3-24, July 5, 2015. Article (CrossRef Link).

[8] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda, "UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware," in *Proc. of 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2017. Article (CrossRef Link).

[9] Ju-Seong Ko, Jeong-Seok Jo, Deuk-Hun Kim, Seul-Ki Choi, and Jin Kwak, "Real Time Android Ransomware Detection by Analyzed Android Applications," in *Proc. of 2019 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1-5, January 22-25, 2019. Article (CrossRef Link).

[10] Mohammad Mehdi Ahmadian, Hamid Reza Shahriari, and Seyed Mohammad Ghaffarian, "Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares," in *Proc. of 12th Int. Iranian Society of Cryptology Conf. on Information Security and Cryptology (ISCISC)*, pp. 79-84, September 8-10, 2015. Article (CrossRef Link).

[11] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Computers & Security*, vol. 74, pp.144-166, May, 2018. Article (CrossRef Link).

[12] Da-Yu K.A.O, Shou-Ching HSIAO, and T. S. O. Raylin, "Analyzing WannaCry Ransomware Considering the Weapons and Exploits," in *Proc. of 2019 21st International Conference on Advanced Communication Technology (ICACT)*, pp. 1098-1107, February 17-20, 2019. Article (CrossRef Link).

[13] Routa Moussaileb, Benjamin Bouget, Aurélien Palisse, Hélène Le Bouder, Nora Cuppens, and Jean-Louis Lanet, "Ransomware's early mitigation mechanisms," in *Proc. of the 13th International Conference on Availability, Reliability and Security, ACM*, pp. 1-10, August 27–30, 2018. Article (CrossRef Link).

[14] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainuddin Mohd Shaid, "A o-day aware crypto-ransomware early behavioral detection framework," in *Proc. of Int. Conf. of Reliable Information and Communication Technology*, pp.758-766, May 27, 2017. Article (CrossRef Link).

[15] LogRhythm, "The Ransomware Threat: A guide to detecting an attack before it's too late," 2016.

[16] Amin Kharraz, and Engin Kirda, "Redemption: Real-time protection against ransomware at end-hosts," in *Proc. of International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 98-119, October 12, 2017. Article (CrossRef Link).

[17] Habib ur Rehman, Eiad Yafi, Mohammed Nazir, and Khurram Mustafa, "Security Assurance Against Cybercrime Ransomware," in *Proc. of Int. Conf. on Intelligent Computing & Optimization*, pp. 21-34, October 4, 2018. Article (CrossRef Link).

[18] Adam Young and Moti Yung, "Cryptovirology: Extortion-based security threats and countermeasures," in *Proc. of IEEE Symposium on Security and Privacy*, pp. 129-140, May 6-8, 1996. Article (CrossRef Link).

[19] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler, "Cryptolock (and drop it): stopping ransomware attacks on user data," in *Proc. of IEEE 36th Int. Conf. on Distributed Computing Systems (ICDCS)*, pp. 303-312, June 27-30, 2016. Article (CrossRef Link).

[20] Manish Shukla, Sutapa Mondal, and Sachin Lodha, "POSTER: Locally Virtualized Environment for Mitigating Ransomware Threat," in *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security*, pp. 1784-1786, October 24 – 28, 2016. Article (CrossRef Link).

[21] Ahmad O Almashhadani, Mustafa Kaiiali, Sakir Sezer, and Philip O'Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware," *IEEE Access*, vol.7, pp. 47053-47067, March 27, 2019. Article (CrossRef Link).

[22] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Computing Surveys (CSUR)*, vol.44, no.2, pp.6, February, 2012. Article (CrossRef Link).

[23] Adam L Young, "Building a Cryptovirus Using Microsoft's Cryptographic API," in *Proc. of the Int. Conf. on Information Security*, pp. 389–401, September 20, 2005. Article (CrossRef Link).

[24] Ben22, "Cryptolocker-using PowerShell as a tripwire," 2014. Article (CrossRef Link).

[25] Nicolo Andronio, "Heldroid: fast and efficient linguistic based ransomware detection," 2015. Article (CrossRef Link).

[26] Kevin Corrigan, "Ransomware: A Growing Epidemic for Business," 2017. Article (CrossRef Link).

[27] Claudio Guarnieri, Alessandro Tanasi and Jurriaan Bremer, "What is Cuckoo?," 2014.

[28] Christian Rossow, Christian J. Dietrich, Chris Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos, and Maarten Van Steen, "Prudent practices for designing malware experiments: Status quo and outlook," in *Proc. of IEEE Symposium on Security and Privacy (SP)*, pp. 65–79, May, 2012. Article (CrossRef Link).

[29] Ross Brewer, "Ransomware attacks: detection, prevention and cure," *Network Security*, vol. 2016, no.9, pp. 5-9, September, 2016. Article (CrossRef Link).

[30] Ibrar Yaqoob, Ejaz Ahmed, Muhammad Habib ur Rehman, Abdelmuttlib Ibrahim Abdalla Ahmed, Mohammed Ali Al-garadi, Muhammad Imran, and Mohsen Guizani, "The rise of ransomware and emerging security challenges in the Internet of Things," *Computer Networks*, vol.129, pp.444-458, September 6, 2017. Article (CrossRef Link).

[31] Diksha Goel, and Ankit Kumar Jain, "Mobile phishing attacks and defence mechanisms: State of art and open research challenges," *Computers & Security*, vol. 73, pp. 519-544, December 20, 2017. Article (CrossRef Link).

[32] LogRhythm Labs, "A Technical Analysis of WannaCry Ransomware," 2017.

[33] Jeonghwan Lee, Jinwoo Lee, and Jiman Hong, "How to Make Efficient Decoy Files for Ransomware Detection?," in *Proc. of the Int. Conference on Research in Adaptive and Convergent Systems*, pp. 208-212, September, 2017. Article (CrossRef Link).

[34] Akashdeep Bhardwaj, Vinay Avasthi, Hanumat Sastry, and G.V.B. Subrahmanyam, "Ransomware digital extortion: A rising new age threat," *Indian Journal of Science and Technology*, vol. 9, no.14, pp.14, April, 2016. Article (CrossRef Link).

[35] Pablo L. Gallegos-Segovia, Jack F. Bravo-Torres, Víctor M. Larios-Rosillo, Paúl E. Vintimilla-Tapia, Iván F. Yuquilima-Albarado, and Juan D. Jara-Saltos, "Social engineering as an attack vector for ransomware," in *Proc. of CHILEAN Conf. on Electrical Electronics Engineering, Information and Communication Technologies (CHILECON)*, pp. 1-6, October 18-20, 2017. Article (CrossRef Link).

[36] Qian Chen, Sheikh Rabiul Islam, Henry Haswell, and Robert A. Bridges, "Automated Ransomware Behavior Analysis: Pattern Extraction and Early Detection," *arXiv preprint arXiv:1910.06469*, October 15, 2019. Article (CrossRef Link)

[37] Aditya K Sood and Richard J. Enbody, "Malvertising–exploiting web advertising," *Computer Fraud & Security*, vol. 2011, no. 4, pp.11-16, 2011. Article (CrossRef Link).

[38] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C. Lupu, "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection," *Computing Research Repository (CoRR)*, abs/ *1609.03020, arXiv.org E-print Archive, Cornell University, Ithaca, NY (USA)*, September, 2016. Article (CrossRef Link)

[39] Qian Chen and Robert A. Bridges, "Automated Behavioral Analysis of Malware A Case Study of WannaCry Ransomware," *Computing Research Repository (CoRR), Cornell University, Ithaca, NY (USA)*, September 25, 2017. Article (CrossRef Link).

[40] Carlos Cepeda, Dan Lo Chia Tien, and Pablo Ordónez, "Feature selection and improving classification performance for malware detection," in *Proc. of 2016 IEEE Int. conf. on on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, pp. 560-566, October 8-10, 2016. Article (CrossRef Link).

[41] Abhishek Singh, Baibhav Singh, and Hirosh Joseph, "Malware analysis," *Vulnerability Analysis and Defense for the Internet*, pp. 169-211, 2008. Article (CrossRef Link).

[42] Ulrich Bayer, Andreas Moser, Christopher Kruegel, and Engin Kirda, "Dynamic analysis of malicious code," *Computer Virology*, vol. 2, no.1, pp. 67-77, May, 2006. Article (CrossRef Link).

[43] Carsten Willems, Thorsten Holz, and Felix Freiling, "Toward automated dynamic malware analysis using cwsandbox," *IEEE Security & Privacy*, vol. 5, no. 2, April, 2007. Article (CrossRef Link).

[44] R.Veeramani and Nitin Rai, "Windows API based malware detection and framework analyses," in *Proc. of Int. conf. on networks and cyber security*, vol. 3, March, 2012.

[45] Eitan Menahem, Asaf Shabtai, Lior Rokach, and Yuval Elovici, "Improving malware detection by applying multi-inducer ensemble," *Computational Statistics & Data Analysis*, vol. 53, no.4, pp.1483-1494, October, 2009. Article (CrossRef Link).

[46] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter*, vol.11, no.1, pp.10-18, November, 2008. Article (CrossRef Link).

[47] Jan Luts, Fabian Ojeda, Raf Van de Plas, Bart De Moor, Sabine Van Huffel, and Johan AK Suykens, "A tutorial on support vector machine-based methods for classification problems in chemometrics," *Analytica Chimica Acta*, vol.665, no.2, pp.129-145, April, 2010. Article (CrossRef Link).

[48] Ajay Mathur and Giles M. Foody, "Crop classification by support vector machine with intelligently selected training data for an operational application," in *Proc. of Int. Journal of Remote Sensing*, vol. 29, no.8, pp. 2227-2240, April 1, 2008. Article (CrossRef Link).

[49] Xianfeng Song, Zheng Duan, and Xiaoguang Jiang, "Comparison of artificial neural networks and support vector machine classifiers for land cover classification in Northern China using a SPOT-5 HRG image," *Int. Journal of Remote Sensing*, vol.33, no.10, pp.3301-3320, Novermber, 2012. Article (CrossRef Link).

[50] Hanqi Zhang, Xi Xiao, Francesco Mercaldo, Shiguang Ni, Fabio Martinelli, and Arun Kumar Sangaiah, "Classification of ransomware families with machine learning based on N-gram of opcodes," *Future Generation Computer Systems*, vol. 90, pp. 211-221, January, 2019. Article (CrossRef Link)

[51] Nikolai Hampton, Zubair Baig, and Sherali Zeadally, "Ransomware behavioral analysis on windows platforms," *Information Security and Applications*, vol. 40, pp.44-51, June, 2018. Article (CrossRef Link)

[52] Yuki Takeuchi, Kazuya Sakai and Satoshi Fukumoto, "Detecting ransomware using support vector machines," in *Proc. of the 47th Int. Conf. on Parallel Processing Companion*, vol. 1, pp. 1-6, August, 2018. Article (CrossRef Link).

[53] Omar M. K. Alhawi, James Baldwin and Ali Dehghantanha, "Leveraging machine learning techniques for windows ransomware network traffic detection," *Cyber Threat Intelligence*, pp. 93-106, 2018. Article (CrossRef Link).

[54] Subash Poudyal, Kul Prasad Subedi and Dipankar Dasgupta, "A Framework for Analyzing Ransomware using Machine Learning," in *Proc. of IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1692-1699, November, 2018. Article (CrossRef Link).

[55] Md Mahbub Hasan and Md. Mahbubur Rahman, "Ranshunt: A support vector machines based ransomware analysis framework with integrated feature set," in *Proc. of 20th Int.Conf. of Computer and Information Technology (ICCIT)*, pp. 1-7,December, 2017. Article (CrossRef Link).

Yahye Abukar is a Ph.D. student in Computer Engineering from Konya Technical University in Turkey. He received his Master degree in Information Security from Universiti Teknologi Malaysia (UTM), Malaysia. He got B.Sc. degree in Information Technology from SIMAD University, Somalia. His research interests focus on Network Security, Malware analysis, Cryptography, and machine learning.

Barış Koçer received master degree and the PhD degree from Selçuk University Computer Engineering Department with a thesis on applying transfer learning methods to optimization problems in 2006 and 2010 respectively. He is currently a is Assoc. Prof. in Konya Technical University Computer Engineering Department and he is working on developing new optimization methods.

BANDER ALI SALEH AL-RIMY is a senior lecturer at UNITAR International University. He received the B.Sc. degree in computer engineering from the Faculty of Engineering, Sana'a University, Yemen, in 2003, the M.Sc. degree in information technology from OUM, Malaysia, in 2013, and the Ph.D. degree in computer science (information security) from the Faculty of Engineering, Universiti Teknologi Malaysia (UTM), in 2019. His research interests include but not limited to Malware, IDS, IoT, network security, and routing technologies. Dr. Al-Rimy was a recipient of several academic awards and recognitions including but not limited to the UTM Alumni Award, the UTM Best Postgraduate Student Award, the UTM Merit Award, the UTM Excellence Award, the OUM Distinction Award, and the Best Research Paper Award.