sustainability

MDPI

Article

A Weighted Minimum Redundancy Maximum Relevance Technique for Ransomware Early Detection in Industrial IoT

Yahye Abukar Ahmed [1], Shamsul Huda [2], Bander Ali Saleh Al-rimy [3,*], Nouf Alharbi [4], Faisal Saeed [5], Fuad A. Ghaleb [3] and Ismail Mohamed Ali [1]

[1] Faculty of Computing, SIMAD University, Mogadishu 801, Somalia; yahye@simad.edu.so (Y.A.A.); bnder321@gmail.com (I.M.A.)
[2] School of Information Technology, Deakin University, Burwood, Melbourne 3125, Australia; shamsul.huda@deakin.edu.au
[3] School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia (UTM), Johor Bahru 81310, Malaysia; abdulgaleel@utm.my
[4] College of Computer Science and Engineering, Taibah University, Al-Madinah P.O. Box 344, Saudi Arabia; nmoharbi@taibahu.edu.sa
[5] DAAI Research Group, Department of Computing and Data Science, School of Computing and Digital Technology, Birmingham City University, Birmingham B4 7XG, UK; alsamet.faisal@gmail.com
* Correspondence: bander@utm.my

Abstract: Ransomware attacks against Industrial Internet of Things (IIoT) have catastrophic consequences not only to the targeted infrastructure, but also the services provided to the public. By encrypting the operational data, the ransomware attacks can disrupt the normal operations, which represents a serious problem for industrial systems. Ransomware employs several avoidance techniques, such as packing, obfuscation, noise insertion, irrelevant and redundant system call injection, to deceive the security measures and make both static and dynamic analysis more difficult. In this paper, a Weighted minimum Redundancy maximum Relevance (WmRmR) technique was proposed for better feature significance estimation in the data captured during the early stages of ransomware attacks. The technique combines an enhanced mRMR (EmRmR) with the Term Frequency-Inverse Document Frequency (TF-IDF) so that it can filter out the runtime noisy behavior based on the weights calculated by the TF-IDF. The proposed technique has the capability to assess whether a feature in the relevant set is important or not. It has low-dimensional complexity and a smaller number of evaluations compared to the original mRmR method. The TF-IDF was used to evaluate the weights of the features generated by the EmRmR algorithm. Then, an inclusive entropy-based refinement method was used to decrease the size of the extracted data by identifying the system calls with strong behavioral indication. After extensive experimentation, the proposed technique has shown to be effective for ransomware early detection with low-complexity and few false-positive rates. To evaluate the proposed technique, we compared it with existing behavioral detection methods.

Keywords: crypto-ransomware; Industrial Internet of Things; enhanced maximum Relevance and minimum Redundancy; TF-IDF; supervised approach

Citation: Ahmed, Y.A.; Huda, S.; Al-rimy, B.A.S.; Alharbi, N.; Saeed, F.; Ghaleb, F.A.; Ali, I.M. A Weighted Minimum Redundancy Maximum Relevance Technique for Ransomware Early Detection in Industrial IoT. *Sustainability* **2022**, *14*, 1231. https://doi.org/10.3390/su14031231

Academic Editors: Mohsin Raza, Ghufran Ahmed, Muhammad Awais and Jawad Ahmad

Received: 27 November 2021
Accepted: 14 January 2022
Published: 21 January 2022

1. Introduction

Recently, the Industrial Internet of Things (IIoT) has witnessed on of the most devastating ransomware attacks that infected major companies such as the Colonial Pipeline, which disrupted the pipeline's operations and causing a serious fuel crisis in the Southeastern United States on 7 May 2021. This incident was one of the fastest-spreading computer-world attacks carried out by ransomware that encrypts and/or hijacks the data, making them inaccessible to the users [1–3]. After encrypting the victim's assets, the ransomware author demands a ransom for the restoration of the assets (user's data) into their original states [4–6]. If the victim paid the ransom to the attacker through the anonymous currency

mechanisms such as Bitcoin [1,3], the access to the encrypted assets is made available again [7,8]. Distribution of the crypto-ransomware can happen by a very large infection vector, including application and browser vulnerabilities, extraction of ZIP files, malicious payload, e.g., cryptoWall, JRE vulnerability, e.g., DMA locker, exploit kits, such as neutrino, eternal blue, eternal romance, etc.

The ransomware analysis is commonly divided by the static and dynamic approach. The most common detection method used by the antivirus is the signature-based detection; it is based on the characterization of the knowledge in its repository. When a new malicious piece is discovered, the anti-virus vendors need to catch its binary signature through analyzing the instructions of the executable code [9–11]. This signature technique is becoming more difficult and detectable, since all recent malicious applications have intention of stealth techniques to evade the detection [9,11]. Ransomware writers employ obfuscation technique for automatically modifying themselves to the unknown version to evade antivirus software detection [4]. This means that the signature-based detection becomes less efficient and reliable to the new unseen ransomware and easily escapes the detection by simple defense such as code obfuscation [6,11]. Many researchers employed the analysis of the ransomware executable binary files using a dynamic approach [12]. To extract the real behavior of the malicious file, samples are executed in controlled an environment such as sandbox [13–15]. In this approach, the changes of the file systems, such as registry modification, file deletion and encryptions, are monitored.

Many researchers are concerned about the detection of executable binary files using machine learning algorithms to test different methods [14]. The authors in [15] presented a method for detecting previously unseen polymorphic computer viruses. Their model was based on Support Vector Machine (SVM) algorithm using the system API to detect malicious codes. Although the sample dataset size was small, they showed the model's detection accuracy and performance by comparing the result of SVM with other learning algorithms. Similarly, Kolter and Maloof [16] introduced a text classification method that detected and identified malicious binary executable files. They compared machine learning classifiers, such as, naive Bayesian, decision tree, boosted decision tree and SVM, and the result showed that the boosted decision tree had a perfect performance. Another work of Singhal and Raul [17] proposed an advanced machine learning-based method for malware detection. Their module extracted API calls made by various normal and harmful executables and implemented an enterprise gateway level to act as a supplement anti-virus present on end-user computers.

Due to resource limitations, most ransomware detection solutions proposed to protect the IIoT focus on reducing the data dimensionality and extract a compact representation of attack patterns [18–21]. The common characteristic of these solutions is the reliance on the autoencoder to derive a reduced set of features for model training. The autoencoder transforms the high-dimensional data into another set with a lower dimension. However, this approach focuses on capturing more information regardless of its relevancy. Although several solutions have used variances of the minimum Redundancy Maximum Relevancy (mRMR) for ransomware early detection, they are not suitable for the IIoT environment as they assume that the APIs are noise-free, which does not hold for the IIoT environment where heterogeneous components are co-located and many of those API are not compatible with each other. Therefore, there is a need for models that can be trained by low-dimensional yet informative features.

In order to enhance the early detection of ransomware in Industrial IoT, this paper examines several machine learning techniques for classifying benign and maliciousness executables programs using API call features, and then presents a comparison of their effectiveness with the existing methods in a controlled environment. In summary, this paper presents three main contributions:

1. A filtering method for API call noisy reduction was proposed that reduces the size of the API call logs based on the indication of a malicious file.

2. A Weighted Enhanced maximum-Relevance and minimum-Redundancy (WEmRmR) technique that selects the most informative API calls with small number of computations. Unlike the original mRmR, the weighted mRmR avoids the limitation for ranking the features based on their importance in the collection. Therefore, we applied the term frequency-inverse document frequency (TF-IDF) method to evaluate the weights of these features to produce the proposed WEmRmR technique.
3. The performance of the proposed technique was evaluated on different datasets with several feature-sets. We also compared our proposed WEmRmR method with the original mRmR to measure the computational complexity and number of evaluations.

The remaining of this paper is organized as follows; the second section discusses the related works. Section 3 presents the proposed methodology. Section 4 presents the experimental result of the research. Finally, in Section 5, the conclusion of the research is presented.

2. Related Work

Recently, supervised and unsupervised machine learning methods were suggested to detect and classify the ransomware and benign application. Ahmed et al. [13] proposed a highly survivable ransomware (HSR) early detection approach with supervised machine learning classifiers. They employed behavioral-based analysis of HSR to extract the integrated features through Term Frequency-Inverse document frequency (TF-IDF). Their experimental results achieved high accuracy and less false-positive rate for detecting HSR in the early phases of the attack. The authors employed seven features to distinguish the ransomware from the benign executable files. However, there are some common features that share the malicious and benign application, which can lead to poor characterization of the ransomware behavior and caused more false-positive alarms. Dynamic behavior-based crypto-ransomware was introduced by Sgandurra et al. [8] to capture the characteristics of ransomware by proposing the EldeRan framework. The authors applied a machine learning algorithm such as the Regularized Logistic Regression classifier that achieved a 96.3% detection rate with an area under the ROC curve of 0.995%. However, identifying a fixed time for ransomware detection is not appropriate to all ransomware samples, since some variants display their malicious activities after human interaction [22,23]. Iglesias and Zseby [24] proposed mRmR, WMR SAM, and LASSO for a multi-stage feature reduction with 41 traffic features. In their work, they reduced and selected the most important features into 16 features. For classification purpose, different algorithms, such as Decision Tree (DT), k-Nearest Neighbor (kNN), Naïve Bayes (NB), Least Absolute Shrinkage and Selection Operator and the Least Angle Regression (LASSO-LAR), Artificial Neural Network (ANN) and Support Vector Machine (SVM), are employed with five-fold cross-validation. However, the complexity and the number of evaluations of mRmR generated features are clearly experimented in this work.

Similar to traditional systems, the IIoT are targeted by ransomware attacks, whose effect mostly is catastrophic as it disrupts the critical infrastructures and industrial operations. In their study, [18] investigated the likelihood that targeted ransomware attacks disturb the edge layer of IIoT. The attack vectors have been explored and both dynamic and static analysis were carried out. The study pinpointed the importance of the inclusion of kernel-related parameters in detecting the ransomware. However, no solution has been proposed for mitigating such attacks. In the study conducted by [19], the authors proposed a deep learning-based technique to extract the latent representation of ransomware attack patterns in IIoT. The solution relies on a deep autoencoder to reduce data dimensionality. However, one of the major limitations of the autoencoder is that it learns to capture as much information as possible rather than as much relevant information as possible. Consequently, the ability of the model to perceive the attack patterns is negatively affected. The high data dimensionality was also investigated [20], and a solution based on stacked variational autoencoder (VAE) was proposed. The data were augmented using the VAE and new artificial observations have been generated. However, the autoencoder sacrifices

the relevancy of the extracted features. The same approach has also been used by [21] as they built the detection model based on the Constructive Denoising Auto-Encoder (CDAE) coupled with the Convolutional Neural Network (DNN). The model, however, tries to preserve less information at the cost of losing the representativeness.

An enhanced mRMR feature selection technique was proposed by Al-rimy et al. [23], which relies on the Redundancy Coefficient Gradual Upweighting (RCGU) technique to capture the discriminative features from ransomware pre-encryption data. By evaluating each feature individually, the RCGU overcomes data insufficiency and provides robust features. Regarding the feature selection approach, filter methods are proposed for detection of a ransomware sample by selecting the informative runtime characteristics of ransomware. Huda et al. [1] employed a non-signature-based framework with API calls using hybrids of a support vector machine wrapper and a filter-based approach. Authors combined filters with the wrapper approach to identify the selective features. In the experimental results, the mRmR was used with API call's scores to feed SVM-based wrapper heuristics algorithm that reaches an accuracy of 94.362% with 291 APIs.

The supervised learning was also used to build ransomware detection models. These models are trained using data collected during ransomware analysis [25–27]. Such an analysis takes place either statically or dynamically [28]. For ransomware analysis, the feature selection is a main step that many of existing research works employ to introspect the latent characteristics of the malicious program [29–32]. For static analysis, ransomware's Portable Executable (PE) file is unpacked, and the source code is introspected to extract the malicious patterns. Several studies followed this approach [33–35]. Although static analysis is fast, safe and accurate in identifying previously known ransomware samples, this approach suffers from several flaws [36–38]. Particularly, static analysis is unable to deal with evasive strains that leverage obfuscation and packing techniques to change their structures and/or protect the code from being analyzed [38–43]. In addition, the static analysis is not suitable for crypto-ransomware early detection as the detection relies on runtime data which cannot be provided by the static analysis.

On the other hand, the dynamic analysis captures runtime data generated during the execution of ransomware samples [28]. Such data represent the behavioral aspect of the malicious software, hence can be used to detect the attacks [34,44]. Like static data, dynamic data can be introspected, and malicious patterns can be extracted [45]. Due to its efficacy for countering the sophisticated ransomware families that employ polymorphic techniques to deceive detection, the dynamic analysis gained popularity in the research community [23,41,42,46–50]. During the dynamic analysis, several types of data are collected, including API calls, PE contents, and file systems; memory; CPU and I/O statistics [2]. The collected data are used to extract several attack patterns by which detection models are built. However, the main drawback of ransomware dynamic analysis is the inclusion of many irrelevant data and noise due to the overlapping between runtime data generated by ransomware process and other applications running simultaneously in the system. As such, it is important to distinguish between the patterns pertaining to ransomware behavior and other applications.

3. Methodology

In this section, we present the research methodology of the proposed technique that can detect the ransomware in the earlier phases using supervised machine learning. We discuss the general architecture that contains four main steps, including data acquisition, analysis of data, feature extraction, selecting the informative features using Enhanced maximum-Relevance and minimum-Redundancy (EmRmR) ranked with the TF-IDF algorithm. Finally, supervised machine learning algorithms with an integrated number of prominent features are utilized.

3.1. Dataset Collection, Pre-Processing and Analysis

In this study, the dataset that contains benign and ransomware samples is employed. Figure 1 shows the workflow of the environmental dynamic sample execution. It consists of a dataset that combines benign and ransomware samples, ground truth represented by Virustotal, Cuckoo Sandbox, and API corpus. Similar to [19,21], the sandbox simulates the edge gateway for the IIoT system. Both dataset instances are in the format of Windows Portable Executable (PE) file binaries. We collected and acquired a total of 1500 unique samples both benign and malicious files. The benign samples of 450 executable files collected from the file systems in the "System32" directory of a fresh installation of Windows 64-bit environment. We also acquired a total of 1050 ransomware executable files from the publicly computer virus websites, such as VirusShare, Maltrieve and VirusTotal. These samples are representative of the real world crypto-ransomware that is gathered from 16 different families, such as Petya, Kovter, WannaCry, Cerber, Citroni, Reveton, Kollah, Torrent Locker, Dirty Decrypt, Crypt-Locker, Crypto-Wall, Trojan-Ransom, Tesla-Crypt, and Pgpcoder.

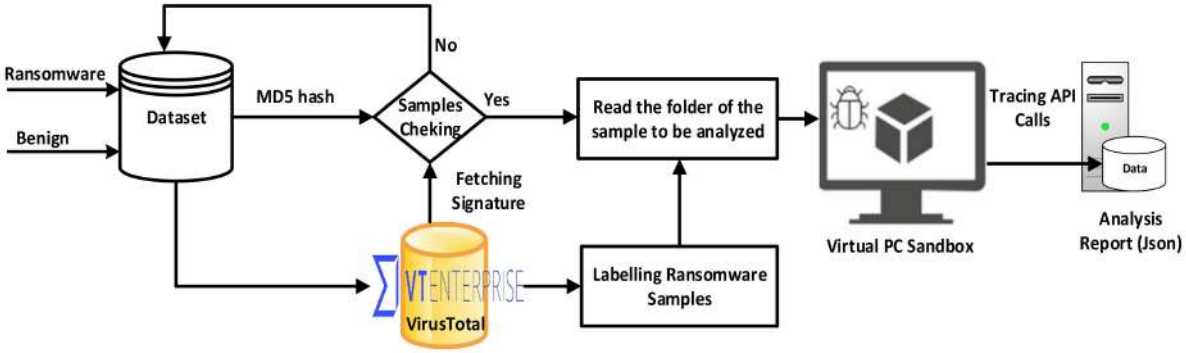

Figure 1. The Environmental Dynamic Sample Execution.

To determine the maliciousness of the files, we double-checked the MD5 hash values of the samples. To do this, we employed Virus Total service to compare the obtained hashes with 57 common different antivirus software. The collected samples contain redundant MD5 hash instances with other ransomware names that may lead to a poor detection of the model and increase the false-positive rate. To remove these redundancies, again, we rechecked the sample's hash through online Virus Total service. For ransomware family categorization, we employed antivirus vendors' labelling scheme to classify the sample's names. Once the data acquisition and labelling process are finished, the pre-processing tasks begin to clean the dataset such as file type identification, and removal of duplicate files.

Since ransomware writers disguise themselves by employing evasion techniques, such as obfuscation, compression, and encryption to subvert the static analysis approach, we used a dynamic analysis approach to gain valuable information related to ransomware behavior. In this method, malware instances are executed inside a guest virtual machine with a host-sandbox to expose the dynamic malicious activities of the samples. To capture the runtime characteristics of samples, we employed Cuckoo Sandbox, a free open-source tool for automation of malware analysis [25]. Cuckoo monitored and recorded information in terms of the API calls, network traffic, changes of files and folders, processes and memory dumps.

We installed Cuckoo sandbox in Ubuntu 16.04 LTS Desktop fully updated version as host operating system. The guest machine was WindowsXp_server_Pack3 32bit as it has less security protections that provides more ransomware activities. We executed every sample in the Sandbox up to a range of 4 until 9 min to exhibit its malicious behavior, but modern ransomware pauses its execution until human interaction such as a mouse event or a key is pressed. Therefore, we employed a python script that works under the Sandbox to do the normal user's activities, such as clicking, creating and deleting documents and

folders on the desktop. Once the sample execution is completed, the output of the sandbox is a human-readable file with extension of JavaScript Object Notation (JSON) format.

The framework for the ransomware detection-based supervised-approach with WEm-RmR is shown in Figure 2. The framework is composed of five components, namely data collection, sample extraction, API gathering, API refinement, and detection engine. During data collection, the ransomware and benign samples are collected to build the dataset and labels are added for each sample based on the decision of the Virustotal. During sample execution, both benign and ransomware samples are submitted to the sandbox for analysis. The APIs are captured and stored into trace files. Then, the refinement is conducted where the APIs are purified, and the failed ones are identified. During this phase, the features are extracted using the N-gram technique and then selected using the proposed EmRmR. The selected features are then used in the detection engine to train the model.

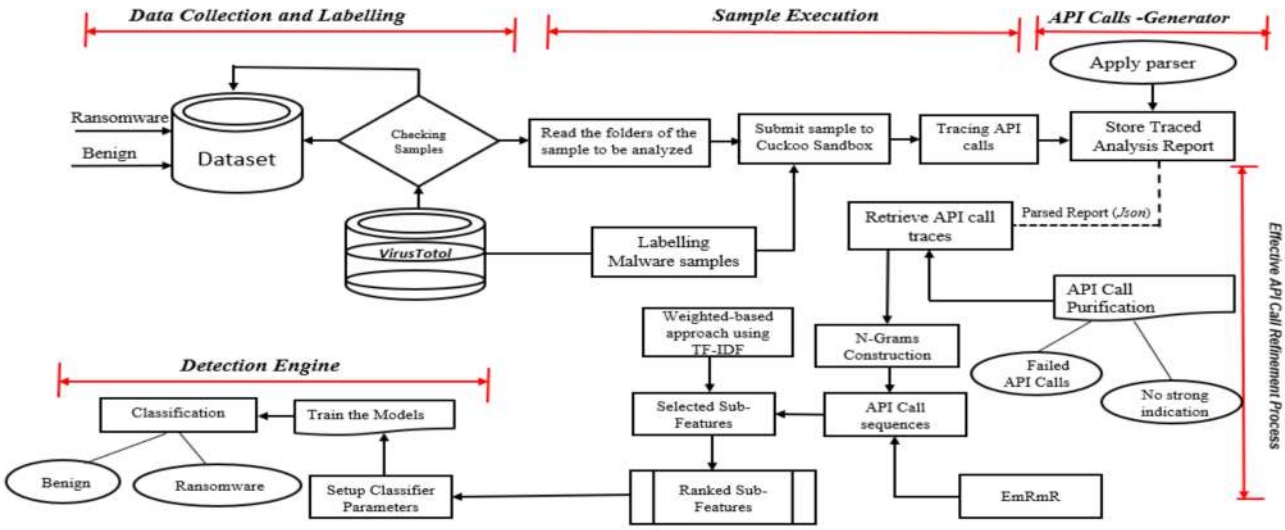

Figure 2. The general framework for Ransomware detection-based supervised-approach with WEmRmR.

3.2. Run-Time Feature Extraction

The behavioral JSON log files generated by the sandbox are then moved into the extraction phase to gain the valuable API calls. Table 1 shows samples of runtime API calls used by the ransomware during the execution. Although this output log file contains several groups of analyzed malware activities, we focused only the behavioral part that defines the runtime characteristics of the malicious samples. API Call features are generated from the malicious log files and fed into machine learning algorithms for detection purposes. These effective API Calls are considered to be the most promising approach for the detection and classification of ransomware by providing a valuable behavior of suspicious activities and the attack patterns [16]. To expose the malicious activities, a ransomware file requires a service from the operating system through an application program interface (API) call that represents the essential behavior of the malware [27].

The collected log files contain a huge number of runtime features that occupy hundreds of MBs of memory, so retrieving the most important elements from these JSON files manually is experimentally infeasible. Therefore, we employed a parsing algorithm to convert JSON-formatted string representations to the appropriate machine learning format objects as shown in Algorithm 1. The parsing algorithm reads the content of the output JSON file of every process and extracts the entire trace that consists of the API call parameters and its values. To reduce the size of the traces, the parameters and the return-values of the API calls were eliminated from the log files. The names of the related API calls were extracted to create a feature vector based on the process timestamp.

Algorithm 1: API Calls Feature Extraction

Input: Set path J_R that contains a static feature f.
Output: Extracted files

```
for process in json_data['behavior'] ['processes'] do
if json_data_process is equal to states then
set FST = process_first_seen
if FS is greater than FST or equal to zero
set first_seen = first_seen_temp
for features in json_data_process[F] do
if features [F] not in _Dict_features then
set _Dict[F] and timestamps = f and _time
Our_Dict [F][count] = 1
Else set _Dict [F] and timestamps = _F_timeappend_F
Dict [F][count]+1, return FS, Dict_ [F]}
Function ParsedJson_Files (JR, PR)
for (JR, PR) in G_file() do
for i, name in enumerate [all_files] do
If name ends with ('json') then
Fname = initialize files that matches (name)
Open the json data (JR+Fname) as (jsonf)
Set Sdata = load (jsonf)
GetFeature ( Sdata , true)
Print(PR)}
In the main function {
Input JR ← parse_directory
Input PR ← F
Store the user's input in the (JR, PR) variables
Parsed Json_Files(JR, PR)
    if name is equal to main then terminate
```

3.3. Weighted Enhanced Maximum Relevance and Minimum Redundancy (WEmRmR)

The output of the JSON files includes a huge amount of unrelated runtime API calls. The ransomware authors inject many API calls into the flow execution of the program and this leads to a noisy characteristic of the feature vector. The detection results of machine learning becomes poor due to the hard monitoring and analysis of the real runtime ransomware activities. Therefore, many researchers employed the Maximum Relevance and Minimum Redundancy (mRmR) algorithm that was suggested by Peng [31]. The mRmR is a popular filter method that is used to select the informative characteristics with high relevance of the class target, and reduces the redundancy features with a supplementary feature set [29]. The implementation of mRmR has been effectively employed in many different fields, such as video processing, microarray gene expression and data analysis [30]. In the mRmR approach, the maximum relevance (mR) selects the dynamic runtime features correlated to the characteristics of ransomware types without considering the relationships among features. If S is a set of ransomware features, $MI(f_i, C)$ is the mutual information between the features f_i and the malicious target class, the mR is calculated as:

$$Max\_Relevance(f_i, C) = \frac{1}{|S|} \sum_{f_i \in S} MI(f_i, C) \quad (1)$$

$$MI(f, f) = \sum_{x,y} p(x, y) log \frac{p(x, y)}{p(x)p(y)} \quad (2)$$

The maximum relevance (*mR*) describes the runtime characteristics related to the malicious file, but mR causes redundancies [18]. To address this replication issue in *mR*, the minimum Redundancy (*mR*) criterion is employed and illustrated as:

$$Min\_Redundance(f_i, C) = \frac{1}{|S|^2} \sum_{f_i, f_j \in S} MI(F_i, F_j) \tag{3}$$

Therefore, the combination of $Max\_Relevance(f_i, C)$ and $Min\_Redundance(f_i, C)$ into one function is described by mRMR using the mutual information difference $(mRMR\ MID)$ explained as follows:

$$MID = MAX_S \left[MI(f_i, \mathrm{C}) - \frac{1}{|S|} \sum_{f_i, f_j \in S} MI(F_i, F_j) \right] \tag{4}$$

The disadvantage of mRmR is the redundant calculations of mutual information (MI) among pairs of features, so, we proposed a slight version of the Enhanced (EmRmR) method that employs with an empty set. This approach identifies the most informative features $\{f_i \ldots\ldots f_q\}$ from the original feature set F and eliminates the noise. This repletion does not finish until all subset features are equal to q. Due to the repetition, the mRmR algorithm calculates the MID value on the $(n - q + 1)$ feature at each iteration.

The generated featurWes by the EmRmR are not ranked, therefore, we need to weight the selected features based on their importance in the document collection. We employed the term frequency-inverse document frequency (TF-IDF) method to weigh the features as expressed, as follows:

$$\mathrm{W_i} = \mathrm{TF}(\omega_i, \mathrm{d}) \times \mathrm{IDF}(\omega_i) \tag{5}$$

where $\mathrm{W_i}$ is the scoring method of word ω_i in the feature pooling d∈D, and TF (ω_i, d) is the frequency of term of ω_i in document d, and IDF (Inverse Document Frequency). Algorithm 2 presents the pseudocode of the proposed Weighted Enhanced maximum Relevance and minimum Redundancy (WEmRmR).

Algorithm 2: Weighted API Call Extracted using Enhanced maximum Relevance and minimum Redundancy (WEmRmR)

Input: Discretised data d, number of features in d is F, subset features $\{\mathrm{f_i} \ldots\ldots \mathrm{f_q}\} \in$ F , class C, number of features to select q, $\mathrm{S_f}$ selected features
Output: selected output features F

```
        Initialization
27.     S_f ← 0;
28.     for f_i ∈F do
29.             Relevance (S) ← Mutual_Info (f_i, C);
30.             Aggregate_ Redun = 0;
31.     end for
32.     S_f = Max (Relevance (S))
33.     l ∈ {s1 s2 s3 ..... s_t} ← [f_i ∈F | S]//To store the highest scorer
34.     for t1: q − 1 do
35.     size ← len (l ∈ {s1 s2 s3 ..... s_t})
36.     while ν ++ < size do
37.             for f_j∈F do
38.               Rel_f ← Relevance (S)
39.               Aggregate_ Redun = (k + 1 + MI(f_i, f_j + 1))
40.             end for
41.             R_f ← Rel_f -Aggregate_Redun;
42.     end for
43.     return selected feature subset {f_i ..... f_q};
```

Table 1. Runtime API Call behaviors.

API Calls	API Category
System Information	GetSystemWindowsDirectoryW, GetUserNameA, GetSystemInfo, GetSystemDirectoryA, GetSystemDirectoryA, GetNativeSystemInfo
Process and Thread	WriteProcessMemory, CreateRemoteThread, SetThreadContext, NtResumeThread
Registry	RegNotifyChangeKeyValue, RegCreateKeyExA, NtAllocateVirtualMemory
Cryptography	CryptGenKey, EncryptMessage, CryptEncrypt, CryptHashData, CryptCreateHash, CryptAcquireContextW
File Management	DeleteFileW, CreateFileW, FindNextFileW, WriteFile, CopyFileA

4. Experimental Results

In this section, we present the experimental results of the several machine learning algorithms, such as Logistic Regression (LR), k-Nearest Neighbor (kNN), Random Forest (RF), Decision Tree (DT), AdaBost (AD) and Support Vector Machine (SVM) on various feature dimensions generated by the Enhanced EmRmR and, then, we weighted the EmRmR. To evaluate the proposed detection scheme, the accuracy of the model, F-measure, precision and recalls, and the Area Under the Curve are depicted. The definitions of these performance metrics have been defined in Table 2. To reduce the dimensionality of the features, we selected the Top-k feature set: 30, 60, 90, 120 and 150.

Table 2. Performance Criteria.

Metrics	Description
True Positive (TP)	The ransomware samples that are predicted as ransomware
False Positive (FP)	The number of benign which are predicted as ransomware
True Negative (TN)	The samples that are correctly classified as benign programs
False Negative (FN)	The number of ransomware samples that are misclassified as benign programs
Accuracy	The ratio of properly identified samples, either malware or benign
Precession	Proportion of predicted ransomware which is real ransomware
Recall	Proportion of actual ransomware which is predicted ransomware
F-Measure	The balance between the precision and the recall on the selected subset features
ROC	Area Under the curve

4.1. Accuracy

The accuracy of the classifiers describes the performance of the trained and the tested models. To evaluate the accuracy, we experimented the sub-selected *k* features from the EmRmR and present the results in Tables 3 and 4. The accuracy variation among the classifiers ranging from 57.61% to 98.63 for the top-k feature sets: 30, 60, 90, 120, 150 and 180, with 10-fold cross-validation. Table 3 illustrates the classifiers' performance on the accuracy of the selected k features without ranking. The lowest accuracy (75.10–89.92%) among the selected k dimensions showed by the classifiers when k = 30 is employed to train and evaluate the detection model. This poor detection accuracy is probably due to the short ransomware behavior characteristic that is not important for describing the pattern of the malicious file from normal files. That is why some ransomware families tend to carry out the entire attack within a short time to inflict the maximum damage before the detection takes place. Such behavior is challenging as it does not give enough time to capture sufficient data needed for accurate detection.

Table 3. The accuracy of classifiers on EmRmR k features.

	kNN	LR	SVM	RF	DT	AD
K_30	0.853	0.899	0.773	0.751	0.865	0.782
K_60	0.674	0.958	0.912	0.861	0.949	0.949
K_90	0.769	0.956	0.986	0.853	0.974	0.917
K_120	0.798	0.965	0.853	0.926	0.899	0.912
K_150	0.752	0.782	0.892	0.847	0.909	0.881

Table 4. The accuracy of classifiers on WEmRmR features.

	kNN	LR	SVM	RF	AD	DT
WK_30	0.371	0.625	0.948	0.932	0.033	0.952
WK_60	0.041	0.959	0.976	0.971	0.007	0.988
WK_90	0.006	0.993	0.977	0.976	0.012	0.987
WK_120	0.071	0.935	0.974	0.959	0.035	0.962
WK_150	0.160	0.839	0.951	0.936	0.035	0.964

The highest accuracy (76.93–98.64%) was achieved by the classifiers when the top 90 k of the feature set was employed. However, when we expanded the k feature dimensionality to k = 150 and k = 180, the classifiers' detection has dramatically declined to (78.23–89.14%). This is because the effect of overfitting using high-dimensional data, which adversely affects the accuracy of the detection. Particularly, adding more features leads to feed the model with less relevant and noise input that confuses the model and makes it less sensitive to the actual attack patterns.

On the other hand, we also evaluated the accuracy of the classifiers based on the WEmRmR method on same k feature dimensionality as shown in Table 4. Overall, the results of this experiment are nearly similar to the previous one, but were quite satisfactory. Interestingly, the accuracy of the classifiers significantly outperformed compared the previous experiment when the $k = 30$, $k = 60$ and $k = 180$ is used. This is attributed to the ability of WEmRmR to filter out the noise, irrelevant as well as the redundant features, which contribute to improve the discriminative capability of the detection model.

4.2. F-Measure

In this section, we evaluated the F-measure to convey the balance between the precision and the recall on the selected subset features. This overcomes the problem of biasness towards the major class in imbalanced datasets as it is the case for a ransomware dataset used in this study. Figure 3 compares the F-measure of each classifier trained and evaluated with different $k - Features$ dimensions without ranked scheme. The highest F-measure values were achieved by the SVM (0.988), and DT (0.986) on a subset feature of $k = 90$, followed by the LR (0.964), RF (0.917) on $k = 120$. The classifier of AD and kNN showed a fairly good F-measure of 0.903 and 0.842 with $k = 90$ and $k = 30$ feature dimensions, respectively. We observed the lowest F-measure presented by the AD (0.554) when $k = 60$ of the feature set was employed to train and test the detection model. This is due to the lack of diverse features that AdaBoost needs to build the base estimators that introspect the data from different perspectives.

Figure 4 shows that the classifiers on differently weighted feature dimensions have good/better F-measure compared to the previous unranked feature dimension. From the result in Figure 4, it is obvious that the classifiers on $k = 120$ performed the highest with an F-measure ranging from 0.727 to 0.989. More specifically, the LR Classifier showed the highest F-measure of 0.989% on $k = 120$, but presented a poor F-measure on the weighted subset feature of $k = 150$. This is again the result of overfitting that high-dimensional data normally cause.

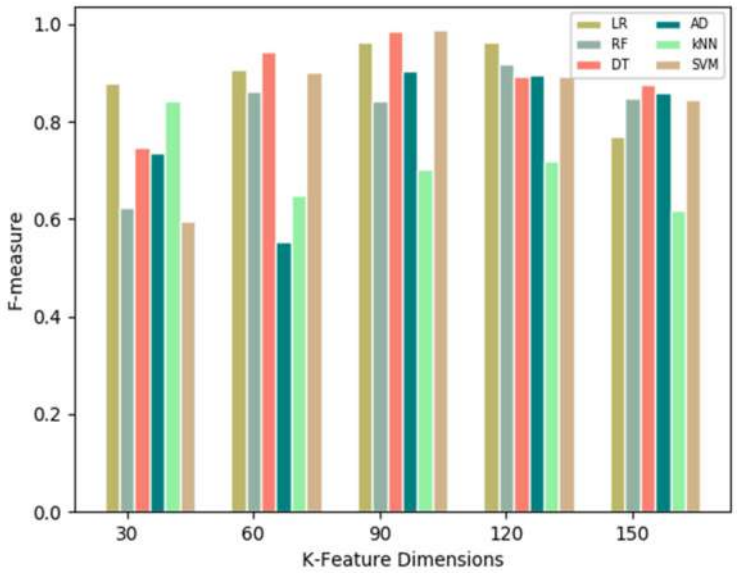

Figure 3. Classifier's F-measure with EmRmR k feature dimensions.

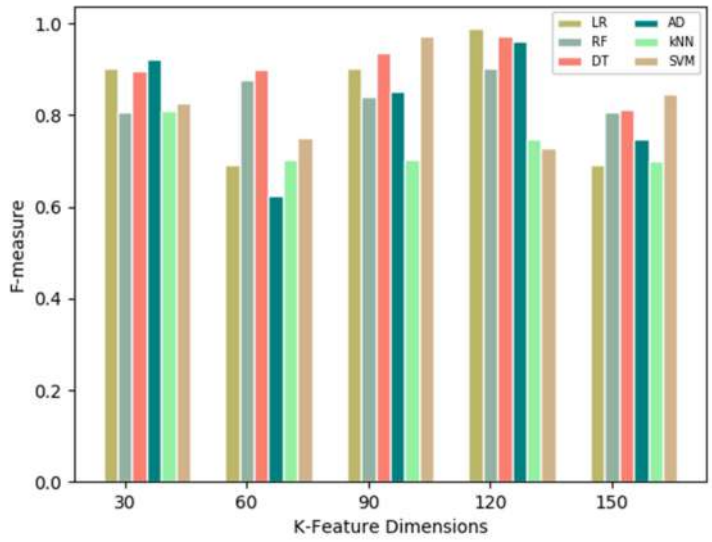

Figure 4. Classifiers' F-measure with WEmRmR k feature dimensions.

Likewise, although the size of the weighted k features is enlarged to 150, the classifiers showed the lowest F-measure; this reveals that the small size of weighted features does not indicate the runtime characteristics of the malicious file.

4.3. Area under the Curve

To measure the performance of the classifiers, we evaluated the Area Under the Curve (AUC), precision and recall on the sub-selected *k* features. Tables 5 and 6 demonstrate the AUC values, precision and recall of different classifiers on both weighted and unweighted $k-$feature sets. From Table 5 (A,B), the classifiers on k_90 and k_120 feature sets achieved the highest AUC values of all the k-feature dimensions, ranging from 0.996 to 0.724, with a lesser false-positive rate of 0.016. The lowest AUC values were obtained by the classifiers when k_30 feature sets were employed. This demonstrates that less k-selected attributes do not determine the dynamic characteristics of the malicious file. Moreover, the AUC values of the detection model increased, whereas the size of k was set to 120. This is attributed to the ability of the proposed feature selection technique to pick discriminative features and filter out the irrelevant redundant features.

Table 5. TPR, FPR, Precision, Recall, and AUC with EmRmR feature dimensions in different algorithms.

(A)	kNN					LR					SVM				
	K_30	**K_60**	**K_90**	**K_120**	**K_150**	**K_30**	**K_60**	**K_90**	**K_120**	**K_150**	**K_30**	**K_60**	**K_90**	**K_120**	**K_150**
TPR	0.812	0.647	0.73	0.747	0.751	0.893	0.958	0.978	0.972	0.753	0.76	0.903	0.986	0.862	0.893
FPR	0.246	0.381	0.285	0.284	0.453	0.049	0.049	0.021	0.027	0.205	0.45	0.145	0.016	0.053	0.185
Precision	0.812	0.653	0.687	0.701	0.605	0.864	0.892	0.953	0.958	0.749	0.581	0.903	0.989	0.874	0.832
Recall	0.813	0.647	0.696	0.707	0.615	0.869	0.896	0.958	0.957	0.756	0.583	0.904	0.979	0.879	0.831
AUC	0.834	0.614	0.732	0.782	0.623	0.914	0.921	0.976	0.951	0.772	0.601	0.879	0.996	0.916	0.859
(B)	**RF**					**DT**					**AD**				
	K_30	**K_60**	**K_90**	**K_120**	**K_150**	**K_30**	**K_60**	**K_90**	**K_120**	**K_150**	**K_30**	**K_60**	**K_90**	**K_120**	**K_150**
TPR	0.751	0.861	0.841	0.917	0.847	0.845	0.945	0.986	0.892	0.915	0.722	0.555	0.872	0.903	0.882
FPR	0.451	0.186	0.346	0.081	0.175	0.081	0.061	0.018	0.049	0.161	0.41	0.399	0.043	0.072	0.134
Precision	0.601	0.862	0.853	0.918	0.849	0.732	0.945	0.986	0.845	0.895	0.719	0.526	0.887	0.879	0.851
Recall	0.602	0.861	0.841	0.917	0.847	0.736	0.945	0.986	0.863	0.896	0.723	0.555	0.896	0.885	0.853
AUC	0.591	0.901	0.848	0.918	0.869	0.753	0.952	0.978	0.914	0.901	0.743	0.551	0.926	0.914	0.864

Table 6. TPR, FPR, Precision, Recall, and AUC with weighted EmRmR feature dimensions in different algorithms.

(A)	kNN					LR					SVM				
	WK_30	**WK_60**	**WK_90**	**WK_120**	**WK_150**	**WK_30**	**WK_60**	**WK_90**	**WK_120**	**WK_150**	**WK_30**	**WK_60**	**WK_90**	**WK_120**	**WK_150**
TPR	0.823	0.73	0.705	0.727	0.705	0.921	0.684	0.889	0.895	0.722	0.824	0.553	0.966	0.716	0.893
FPR	0.296	0.285	0.326	0.268	0.326	0.124	0.51	0.049	0.075	0.41	0.203	0.033	0.039	0.316	0.185
Precision	0.801	0.731	0.701	0.731	0.702	0.893	0.687	0.895	0.91	0.696	0.824	0.743	0.969	0.715	0.832
Recall	804	0.732	0.697	0.727	0.705	0.894	0.686	0.892	0.895	0.693	0.828	0.749	0.969	0.716	0.831
AUC	0.805	0.732	0.705	0.732	0.679	0.899	0.695	0.916	0.938	0.743	0.853	0.728	0.986	0.743	0.859
(B)	**RF**					**DT**					**AD**				
	K_30	**K_60**	**K_90**	**K_120**	**K_150**	**K_30**	**K_60**	**K_90**	**K_120**	**K_150**	**K_30**	**K_60**	**W_90**	**K_120**	**K_150**
TPR	0.806	0.891	0.841	0.903	0.806	0.914	0.903	0.946	0.647	0.861	0.925	0.751	0.852	0.535	0.845
FPR	0.216	0.049	0.146	0.07	0.216	0.132	0.145	0.036	0.381	0.136	0.126	0.451	0.049	0.439	0.08
Precision	0.812	0.869	0.851	0.889	0.812	0.132	0.903	0.936	0.653	0.805	0.923	0.621	0.841	0.612	0.722
Recall	0.806	0.871	0.841	0.876	0.806	0.886	0.903	0.936	0.647	0.801	0.921	0.613	0.835	0.555	0.733
AUC	0.794	0.914	0.848	0.914	0.794	0.906	0.879	0.938	0.614	0.826	0.914	0.591	0.856	0.534	0.753

On the other hand, we evaluated the AUC values of the classifiers on weighted k feature dimensions on their relevance and ranking by the TF-IDF-EmRmR. Overall, the results of AUC values, precision and recall improved significantly compared with the preceding experiment. Regarding to Table 6 (A,B), the highest accuracy of 0.998 to 0.732 presented by the classifiers with a low false-positive rate of 0.017 when the k_120 of the TF-IDF-EmRmR-based features were employed to train the detection accuracy. When we increased the number of the selected k feature dimension to K_150, the classifiers presented the lowest AUC values. This is due the fact that the large size of the k feature dimension does not differentiate the characteristics of the malicious file from the benign. It is interesting to note that the k_30 feature set generated by the TF-IDF-EmRmR has shown fairly better AUC values compared to the EmRmR features.

5. Comparing with Previous Works

In this section, as shown in Table 7, the proposed method is compared with earlier similar works against the detection performance using different kinds of ransomware. The aim is to present the effectiveness of the detection model of the supervised classifiers based on the performance criteria. Vinod et al. [32] proposed a dynamic-based method for detection of the obfuscated malicious files. This approach applied a hybrid method, such as Principal Component Analysis (PCA) and Minimum Redundancy and Maximum Relevance (mRmR) filter with various mnemonic *n*–grams, to produce informative dynamic behaviors. The generated features are then fed into several classification algorithms that showed a detection accuracy of 94.1% with a less false-positive rate. Similar work has been conducted by Iglesias and Zseby [24] who used a combined feature reduction approach for the network traffic. They employed *SAM, LASSO, WMR, and mRmR* selection techniques to reduce 41 traffic features into 16 predominant features. In their work, the classification algorithms presented an accuracy (0.27–95.48) with mRmR-generated features. This is attributed to the ability of the proposed feature selection technique to make the appropriate trade-off between relevancy and redundancy. Thus, the proposed WEmRmR can select the

most distinctive features while maintaining the low redundancy between candidate and already-selected features. This diversifies the detection model's input, which helps look into data from many perspectives and understand the data more precisely.

Table 7. Comparing the proposed method with previous works.

Works	Classifier(s) Used	Detection Rate
Proposed Method (WEmRmR)	kNN, LR, SVM, RF, AD and DT	0.993
Vinod et al. [32]	NB, SMO, IBK, J48, ADA and RF	0.941
Iglesias and Zseby [24]	DT, kNN, NB, LASSO-LAR, ANN and SVM	0.954
Sgandurra et al. [8]	Logistic Regression	0.963
Ye et al. [33]	OOA_Apriori, OOA_FP-Growth and OOA_Fast_FP-Growth	0.931

6. Conclusions

Dynamic behavioral ransomware detection on API call sequences with a supervised machine learning algorithm are proposed in this study. We used runtime analysis for 673 ransomware samples on Windows platforms. The study focused on the malicious behaviors of 16 newly emerged ransomware families. To evade the detection, ransomware authors employ packing and obfuscation methods that make the dynamic behavior analysis more difficult. The most promising method for ransomware detection is the dynamic analysis that determines the characteristics of the real ransomware behavior. This method uses API calls to distinguish the malicious activities done by the ransomware from the benign files. However, when ransomware executes, a huge amount of irrelevant and redundant API calls are also invoked. This will lead to poor capability of machine learning detection. To address this issue, this paper proposed a weighted enhanced maximum relevance and minimum redundancy (WEmRmR) filter that generates similar features as the EmRmR but better with ranking. To achieve this, the generated EmRmR features are then passed into the term frequency-inverse document frequency (TF-IDF) algorithm to rank the features based on their importance in the feature sets. The experimental results have shown the effectiveness of the proposed technique that obtains a high classification accuracy with few false-positive rates. To empirically evaluate the proposed technique (WEmRmR), we compared the experimental results with the previous similar works and the results showed the superiority of the WEmRmR technique. Future works can investigate the effectiveness of this technique on different datasets obtained from Industrial IoT. In addition, the application of other machine learning and deep learning methods can be investigated.

Author Contributions: Conceptualization, Y.A.A., S.H. and B.A.S.A.-r.; methodology, Y.A.A., S.H. and B.A.S.A.-r.; software, S.H., I.M.A. and B.A.S.A.-r.; validation, B.A.S.A.-r., F.A.G. and Y.A.A.; formal analysis, Y.A.A., B.A.S.A.-r. and I.M.A.; investigation, S.H. and B.A.S.A.-r.; resources, I.M.A.; data curation, Y.A.A.; writing—original draft preparation, Y.A.A.; writing—review and editing, B.A.S.A.-r., F.S., N.A. and F.A.G.; visualization, Y.A.A. and F.A.G.; supervision S.H., N.A. and F.S.; project administration, F.S. and B.A.S.A.-r.; funding acquisition, N.A. and F.S. All authors have read and agreed to the published version of the manuscript.

Funding: Not applicable.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

References

1. Shamsul, H.; Islam, R.; Abawajy, J.; Yearwood, J.; Hassan, M.M.; Fortino, G. A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection. *Future Gener. Comput. Syst.* **2018**, *83*, 193–207.
2. Urooj, U.; Maarof, M.A.B.; Al-rimy, B.A.S. A proposed Adaptive Pre-Encryption Crypto-Ransomware Early Detection Model. In Proceedings of the IEEE 2021 3rd International Cyber Resilience Conference (CRC), Langkawi Island, Malaysia, 29–31 January 2021; pp. 1–6.
3. Ahmadian, M.M.; Shahriari, H.R. 2entFOX: A framework for high survivable ransomwares detection. In Proceedings of the IEEE 13th International ISC Conference on Information Security and Cryptology (ISCISC), Tehran, Iran, 7–8 September 2016; pp. 79–84.
4. Urooj, U.; Al-rimy, B.A.S.; Zainal, A.; Ghaleb, F.A.; Rassam, M.A. Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. *Appl. Sci.* **2022**, *12*, 172. [CrossRef]
5. Kalaimannan, E.; Sharon, K.J.; Theresa, D.; Anthony, P. Influences on ransomware's evolution and predictions for the future challenges. *Cyber Secur. Technol.* **2017**, *1*, 23–31. [CrossRef]
6. Al-Rimy, B.A.S.; Maarof, M.A.; Shaid, S.Z.M. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Comput. Secur.* **2018**, *74*, 144–166. [CrossRef]
7. Al-Rimy, B.A.S.; Maarof, M.A.; Shaid, S.Z.M. Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Gener. Comput. Syst.* **2019**, *101*, 476–491. [CrossRef]
8. Daniele, S.; Luis, M.G.; Rabih, M.; Emil, C.L. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. *arXiv* **2016**, arXiv:1609.03020.
9. Aurélien, P.; Hélène, L.B.; Jean-Louis, L.; Colas, L.G.; Axel, L. Ransomware and the legacy crypto API. In *International Conference on Risks and Security of Internet and Systems*; Springer: Cham, Switzerland, 2016; pp. 11–28.
10. Ghaleb, F.A.; Maarof, M.A.; Zainal, A.; Al-rimy, B.A.S.; Alsaeedi, A.; Boulila, W. Ensemble-based hybrid context-aware misbehavior detection model for vehicular ad hoc network. *Remote Sens.* **2019**, *11*, 2852. [CrossRef]
11. Alexandre, G. Comparative analysis of various ransomware virii. *Comput. Virol.* **2010**, *6*, 77–90.
12. Adamov, A.; Carlsson, A. Reinforcement Learning for Anti-Ransomware Testing. In Proceedings of the 2020 IEEE East-WestDesign & Test Symposium (EWDTS), Varna, Bulgaria, 4–7 September 2020; pp. 1–5.
13. Yahye, A.A.; Koçer, B.; Huda, S.; Al-rimy, B.A.S.; Hassan, M.M. A system calls refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *J. Netw. Comput. Appl.* **2020**, *167*, 102753.
14. Zhang, B.Y.; Yin, J.P.; Hao, J.B.; Zhang, D.X.; Wang, S.L. Using Support Vector Machine to Detect Unknown Computer Viruses. *Int. J. Comput. Intell. Res.* **2007**, *2*, 100–104. [CrossRef]
15. Scaife, N.; Carter, H.; Traynor, P.; Butler, K.R. Cryptolock (and drop it): Stopping ransomware attacks on user data. In Proceedings of the IEEE 36th International Conference on Distributed Computing Systems (ICDCS), Nara, Japan, 27–30 June 2016; pp. 303–312.
16. Kolter, J.Z.; Maloof, M.A. Learning to detect and classify malicious executables in the Wild. *J. Mach. Learn. Res.* **2006**, *7*, 2721–2744.
17. Singhal, P.; Raul, N. Malware detection module using machine learning algorithms to assist in centralized security in enterprise networks. *arXiv* **2012**, arXiv:1205.3062. [CrossRef]
18. Al-Hawawreh, M.; den Hartog, F.; Sitnikova, E. Targeted ransomware: A new cyber threat to edge system of brownfield industrial Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 7137–7151. [CrossRef]
19. Al-Hawawreh, M.; Sitnikova, E. Leveraging deep learning models for ransomware detection in the industrial internet of things environment. In Proceedings of the IEEE 2019 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 12–14 November 2019; pp. 1–6.
20. Al-Hawawreh, M.; Sitnikova, E. Industrial Internet of Things based ransomware detection using stacked variational neural network. In Proceedings of the 3rd International Conference on Big Data and Internet of Things, Isfahan, Iran, 17–18 April 2019; pp. 126–130.
21. Al-Hawawreh, M.; Sitnikova, E.; Aboutorab, N. Asynchronous Peer-to-Peer Federated Capability-Based Targeted Ransomware Detection Model for Industrial IoT. *IEEE Access* **2021**, *9*, 148738–148755. [CrossRef]
22. Aboaoja, F.A.; Zainal, A.; Ghaleb, F.A.; Al-rimy, B.A.S. Toward an Ensemble Behavioral-based Early Evasive Malware Detection Framework. In Proceedings of the IEEE 2021 International Conference on Data Science and Its Applications (ICoDSA), Bandung, Indonesia, 6–7 October 2021; pp. 181–186.
23. Al-Rimy, B.A.S.; Maarof, M.A.; Alazab, M.; Shaid, S.Z.M.; Ghaleb, F.A.; Almalawi, A.; Ali, A.M.; Al-Hadhrami, T. Redundancy coefficient gradual up-weighting-based mutual information feature selection technique for crypto-ransomware early detection. *Future Gener. Comput. Syst.* **2021**, *115*, 641–658.
24. Iglesias, F.; Zseby, T. Analysis of network traffic features for anomaly detection. *Mach. Learn.* **2015**, *101*, 59–84. [CrossRef]
25. Melvin, A.A.R.; Kathrine, G.J.W. A Quest for Best: A Detailed Comparison Between Drakvuf-VMI-Based and Cuckoo Sandbox-Based Technique for Dynamic Malware Analysis. In *Intelligence in Big Data Technologies—Beyond the Hype*; Springer: Singapore, 2021; pp. 275–290.
26. Kolosnjaji, B.; Apostolis, Z.; George, W.; Claudia, E. Deep learning for classification of malware system call sequences. In *Australasian Joint Conference on Artificial Intelligence*; Springer: Cham, Switzerland, 2016; pp. 137–149.
27. Vinod, P.; Viswalakshmi, P. Empirical Evaluation of a System Call-Based Android Malware Detector. *Arab. J. Sci. Eng.* **2018**, *43*, 6751–6770. [CrossRef]

28. Olaimat, M.N.; Maarof, M.A.; Al-rimy, B.A.S. Ransomware Anti-Analysis and Evasion Techniques: A Survey and Research Directions. In Proceedings of the IEEE 2021 3rd International Cyber Resilience Conference (CRC), Langkawi Island, Malaysia, 29–31 January 2021; pp. 1–6.
29. Ramírez-Gallego, S.; Lastra, I.; Martínez-Rego, D.; Bolón-Canedo, V.; Benítez, J.M.; Herrera, F.; Alonso, B. Fast-mRMR: Fast Minimum Redundancy Maximum Relevance Algorithm for High-Dimensional Big Data. *Int. J. Intell. Syst.* **2017**, *32*, 134–152. [CrossRef]
30. Angulo, A.P.; Shin, K. Mrmr+ and Cfs+ feature selection algorithms for high-dimensional data. *Appl. Intell.* **2019**, *49*, 1954–1967. [CrossRef]
31. Peng, A.; Jean-Marc, A.R.; Chamseddine, T. Enhancing malware detection for Android systems using a system call filtering and abstraction process. *Secur. Commun. Netw.* **2015**, *8*, 1179–1192.
32. Vinod, P.; Vijay, L.; Singh, G.S.M. Reform: Relevant features for malware analysis. In Proceedings of the IEEE 2012 26th International Conference on Advanced Information Networking and Applications Workshops, Fukuoka, Japan, 26–29 March 2012; pp. 738–744.
33. Ye, Y.; Dingding, W.; Tao, L.; Dongyi, Y.; Qingshan, J. An intelligent PE-malware detection system based on association mining. *J. Comput. Virol.* **2008**, *4*, 323–334.
34. Andronio, N.; Zanero, S.; Maggi, F. HELDROID: Dissecting and detecting mobile ransomware. In *International Symposium on Recent Advances in Intrusion Detection*; Springer: Cham, Switzerland, 2015; Volume 9404, pp. 382–404.
35. Mercaldo, F.; Nardone, V.; Santone, A.; Visaggio, C.A. Ransomware Steals Your Phone. Formal Methods Rescue It. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*; Springer: Cham, Switzerland, 2016; pp. 212–221.
36. Zhang, H.Q.; Xiao, X.; Mercaldo, F.; Ni, S.G.; 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]
37. Zimba, A.; Wang, Z.S.; Chen, H.S. Multi-stage crypto ransomware attacks: A new emerging cyber threat to critical infrastructure and industrial control systems. *ICT Express* **2018**, *4*, 14–18. [CrossRef]
38. Ahmed, Y.A.; Kocer, B.; Al-rimy, B.A.S. Automated Analysis Approach for the Detection of High Survivable Ransomware. *KSII Trans. Internet Inf. Syst.* **2020**, *14*, 2236–2257.
39. Banescu, S.; Wuchner, T.; Salem, A.; Guggenmos, M.; Ochoa, M.; Pretschner, A. A framework for empirical evaluation of malware detection resilience against behavior obfuscation. In Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo, PR, USA, 20–22 October 2015; pp. 40–47.
40. Choudhary, S.P.; Vidyarthi, M.D. A Simple Method for Detection of Metamorphic Malware using Dynamic Analysis and Text Mining. *Proced. Comput. Sci.* **2015**, *54*, 265–270. [CrossRef]
41. 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]
42. Rhode, M.; Burnap, P.; Jones, K. Early-stage malware prediction using recurrent neural networks. *Comput. Secur.* **2018**, *77*, 578–594. [CrossRef]
43. Daku, H.; Zavarsky, P.; Malik, Y. Behavioral-Based Classification and Identification of Ransomware Variants Using Machine Learning. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 1560–1564.
44. Maseer, Z.K.; Yusof, R.; Mostafa, S.A.; Bahaman, N.; Musa, O.; Al-rimy, B.A.S. DeepIoT. IDS: Hybrid Deep Learning for Enhancing IoT Network Intrusion Detection. *CMC-Comput. Mater. Contin.* **2021**, *69*, 3945–3966. [CrossRef]
45. Alsoufi, M.A.; Razak, S.; Siraj, M.M.; Al-Rimy, B.A.; Ali, A.; Nasser, M.; Abdo, S. A Review of Anomaly Intrusion Detection Systems in IoT using Deep Learning Techniques. *Adv. Data Sci. Adapt. Anal.* **2021**, *72*, 2143001. [CrossRef]
46. Al-rimy, B.A.S.; Maarof, M.A.; Prasetyo, Y.A.; Shaid, S.Z.M.; Ariffin, A.F.M. Zero-day aware decision fusion-based model for crypto-ransomware early detection. *Int. J. Integr. Eng.* **2018**, *10*, 6. [CrossRef]
47. Al-Rimy, B.A.S.; Maarof, M.A.; Alazab, M.; Alsolami, F.; Shaid, S.Z.M.; Ghaleb, F.A.; Hadhrami, A.-T.; Ali, A.M. A pseudo feedback-based annotated TF-IDF technique for dynamic crypto-ransomware pre-encryption boundary delineation and features extraction. *IEEE Access* **2020**, *8*, 140586–140598. [CrossRef]
48. Darem, A.A.; Ghaleb, F.A.; Al-Hashmi, A.A.; Abawajy, J.H.; Alanazi, S.M.; Al-Rezami, A.Y. An Adaptive Behavioral-Based Incremental Batch Learning Malware Variants Detection Model Using Concept Drift Detection and Sequential Deep Learning. *IEEE Access* **2021**, *9*, 97180–97196. [CrossRef]
49. Cohen, A.; Nissim, N. Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory. *Expert Syst. Appl.* **2018**, *102*, 158–178. [CrossRef]
50. Gomez-Hernandez, J.A.; Alvarez-Gonzalez, L.; Garcia-Teodoro, P. R-Locker: Thwarting ransomware action through a honeyfile-based approach. *Comput. Secur.* **2018**, *73*, 389–398. [CrossRef]