



Malware Detection Method Based on File and Registry Operations Using Machine Learning

 Ömer Aslan¹,  Erdal Akın²

¹Corresponding Author; 17 Eylül University, Department of Software Engineering, Bandırma/Balıkesir Turkey; omer.aslan.bisoft@gmail.com

²Bitlis Eren University, Department of Computer Engineering, Bitlis, Turkey; erdalakin1985@hotmail.com

Received 28 December 2021; Revised 18 April 2022; Accepted 25 May 2022; Published online 31 August 2022

Abstract

Malware (Malicious Software) is any software which performs malicious activities on computer-based systems without the user's consent. The number, severity, and complexity of malware have been increasing recently. The detection of malware becomes challenging because new malware variants are using obfuscation techniques to hide themselves from the malware detection systems. In this paper, a new behavioral-based malware detection method is proposed based on file-registry operations. When malware features are generated, only the operations which are performed on specific file and registry locations are considered. The file-registry operations are divided into five groups: autostart file locations, temporary file locations, specific system file locations, autostart registry locations, and DLLs (Dynamic link libraries) related registry locations. Based on the file-registry operations and where they performed, the malware features are generated. These features are seen in malware samples with high frequencies, while rarely seen in benign samples. The proposed method is tested on malware and benign samples in a virtual environment, and a dataset is created. Well-known machine learning algorithms including C4.5 (J48), RF (Random Forest), SLR (Simple Logistic Regression), AdaBoost (Adaptive Boosting), SMO (Sequential Minimal Optimization), and KNN (K-Nearest Neighbors) are used for classification. In the best case, we obtained 98.8% true positive rate, 0% false positive rate, 100% precision and 99.05% accuracy which is quite high when compared with leading methods in the literature.

Keywords: Cybersecurity, malware detection, behavior-based detection, file-registry behaviors, machine learning

1. Introduction

In simple terms, malware can be defined as a set of symbols which performs undesirable changes to the computer hardware as well as operating system resources. There are various types of malware including virus, worm, rootkit, Trojan horse, backdoor, spyware, and so on. The number, complexity and damage of malware to the world economy is increasing everyday. According to scientific reports, the cost of cyber-based attacks to the world economy is estimated in trillions of dollars, and most of these damages come from malware.

To protect the computer based systems from malware, malware needs to be detected before entering the victim system or during the infection. Thus, malware samples need to be examined by using relevant tools. There are two common ways to analyze the malware: static and dynamic analysis [1]. In static analysis, the malware samples are analyzed without running the actual code. Program structures, used strings, imported and exported functions are obtained during the static analysis. However, in dynamic analysis, the codes of malware are performed under the protected environment (Virtual machines or sandboxes), and execution traces which represent the behaviors of the malware are collected. After the malware execution traces are collected, the features are generated. There are several approaches to detect malware which use static and dynamic analysis, namely, signature-based, behavioral-based, heuristic-based, model checking-based, deep learning-based, cloud-based, and mobile and IoT (Internet of Things)-based [2]. The names of the approaches vary according to the platform on which the detection algorithm is proposed and the method used.

At the beginning, the signature-based detection approach was widely used. However, due to the increasing complexity of malicious software in recent years, it has been insufficient to detect new

generation malware [3]. Therefore, over time, researchers have developed behavioral-based, heuristic-based and model checking-based detection approaches. In these approaches, models are created by determining the behaviors or static features of malware samples, and malicious software is detected by using these models [2]. These approaches can detect some malware that has not been seen before. Furthermore, deep learning-, cloud-, mobile- and IoT-based detection approaches have also been used especially in the last few years.

In this paper, a behavioral-based malware detection approach is used which identifies the specific file and registry operations. We divided file-registry operations into five groups including autostart file locations, temporary file locations, specific system file locations, autostart registry locations, and DLLs (Dynamic link libraries) related registry locations. After behaviors are created by using specified locations, the features and their frequencies are calculated. The most significant features are selected by using information gain measures. After features are selected, the machine learning classifiers including C4.5 (J48 version), RF (Random Forest), SLR (Simple Logistic Regression), AdaBoost (Adaptive Boosting), SMO (Sequential Minimal Optimization), and KNN (K-Nearest Neighbors) are used for classification.

In practice, current malware detectors cannot effectively recognize the zero-day malware variants. The proposed approach decreased the deficiencies that current studies have on malicious software detection and improved the performances. In the proposed method, only specific file directories and registry locations are considered when detecting malware. This is because the majority of malware strains show similar behaviors on specific file and registry locations, which cannot be seen on benign samples. To increase the model performance while decreasing the time complexity of the proposed method, the same behaviors on different instances of the same resources match into the same feature, but the frequency of the feature is increased.

The rest of the paper is organized as follows: Section 2 provides background information about malware types, malware analysis, and detection processes. Section 3 reviewed the literature studies on malware detection methods. A proposed method is presented in section 4, and an implementation is given in section section 5. Results and discussion is explained in section 6 and conclusion is given in section 7.

2. Background Information

In this section, definition and types of malware as well as malware analysis process, and detection approaches are explained in detail.

2.1 Malware Definitions and Types of Malware

Any program which performs malicious payloads on victim machines can be defined as malware. There are several malware variants created everyday including virus, worm, Trojan Horse, rootkit, backdoor, ransomware, and many more. The detection of those malware variants becomes challenging because one malware type can present other malware features. Besides, for complicated and targeted attacks, various malware strains work together to increase the impact of the cyber attacks. In addition, new devices and technologies are connected to computer networks everyday, and most of those devices and technologies have several vulnerabilities for hackers to exploit. Code obfuscation techniques are hiding malicious codes from the malware detection system [2]. All of these reasons make the malware detection process more difficult. To effectively detect the different malware variants, the main definitions and features that malware present needs to be examined deeply. Thus, the well-known malware variants are explained shortly as follows:

- **Virus:** It is a malicious software which injects its code into other files to reproduce. During the propagation, it changes its appearance to become undetectable by antivirus scanners [4].
- **Worm:** Unlike the virus, worms do not need other programs to reproduce, instead worms use computer networks to spread. It identifies the vulnerable machines on the network and then copies

itself into those vulnerable systems [5]. Most worms open backdoors in the victim system as well as enable unauthorized access. They delete their own files to hide themselves in the infected system.

- **Trojan Horse:** It appears to be useful benign software, however, it contains malicious code blocks. The attachment program needs to be performed to infect the victim system. It can create backdoors, cause unauthorized access, and reveal sensitive information to the third parties.
- **Rootkit:** A set of programs that conceal its existence from the operating system. It is a kind of man in the middle attack because it intercept and change the communications between the interfaces and several os components [6]. Rootkits are generally used by hackers to hide their existence in the system and provide root level access privileges. Rootkits are also combined with other malware strains in order to perform more sophisticated attacks. It is almost impossible to detect kernel rootkits since they run in kernel mode.
- **Backdoor:** It is a program that bypasses the traditional security mechanisms and opens the system to remote access for attackers [4]. The created backdoors are used by hackers and other malicious softwares to launch more complicated cyber attacks. Backdoors are mostly installed on victim systems by using Worms and Trojan horses.
- **Ransomware:** Ransomware is one of the most destructive types of malware which is designed to prevent or limit access to a computer system [7] until some amount of ransom is paid as a cryptocurrency. Recently, the number and impact of ransomware attacks on big companies have been increasing rapidly.

2.2 Malware Analysis and Detection Processes

To effectively detect malware from celanware, malware needs to be analyzed by relevant tools automatically. During the analysis process, program structures or behaviors are obtained. Malware analysis process divided into two subcategories: static and dynamic analysis. In static analysis, the content of the malware is analyzed without performing the actual codes, on the other hand, in dynamic analysis, the code of the malware is analyzed under dynamic analysis tools and malicious activities are collected [1]. There are a variety of static and dynamic tools including BinText, Md5deep, PEiD, PEview, IDA Pro, API Monitor, Regshot, Process Explorer, Process Monitor, Wireshark and Sandboxes. After malware execution traces are collected by relevant tools, feature generation as well as selection process take place. During the feature creation and selection processes, data mining (DM) and machine learning (ML) techniques are used. Furthermore, ML classifiers are used for the detection process as well. The relationship between malware analysis method, detection approaches, and machine learning techniques can be seen in figure 1.

There are several malware detection approaches which mostly use ML-based detection techniques. The name of the malware detection approaches change on the feature generation and selection processes as well as the platform that is used. The malware detection approaches are broadly divided into signature-based, heuristic-based, behavioral-based and model checking-based detection, and also can be divided further as deep learning-based, cloud-based, mobile devices-based, and IoT-based detection [2]. In each approach, the feature extraction method is different from one another. One detection approach certainly cannot be said to work better than the others, as each approach has its own advantages and disadvantages. The way the approach is used, the feature extraction and classification method affect its success. Traditional malware variants can be detected with high accuracy using the signature-based approach, but signature-based approach cannot demonstrate the same success when detecting new malware strains. A considerable amount of unknown malware samples are detected using behavioral-, heuristic-, and model-based detection approaches. However, they are insufficient to detect some new generation malware. Similarly, deep learning-, cloud- and mobile and IoT-based approaches fall short of detecting malicious software that constantly changes itself [2]. Different approaches can be combined to detect more complex as well as unknown malware. In this study, dynamic analysis is used for malware feature creation, and behavioral-based approach is used for detection.

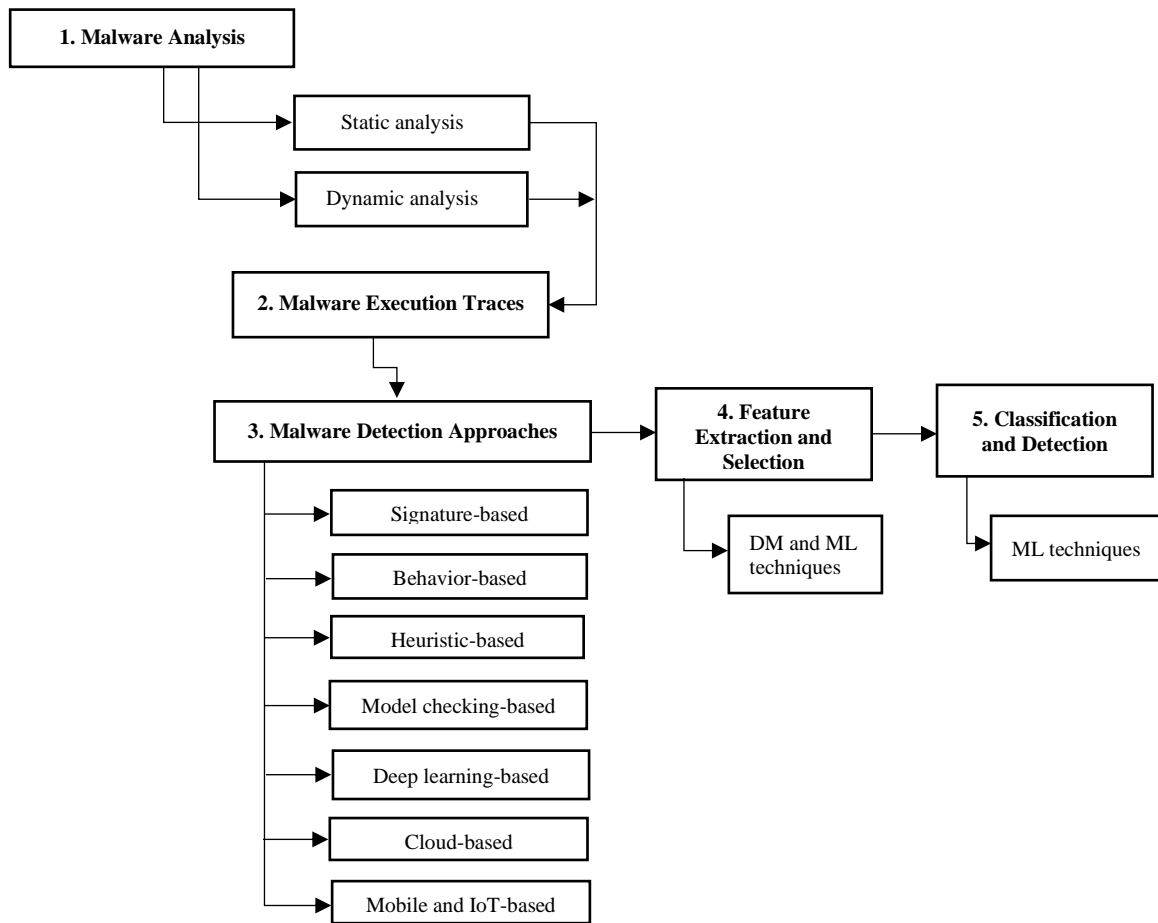


Figure 1 Malware analysis methods and detection approaches

3. Leading Methods in the Literature

There are several scientific papers written about malware detection. In each paper, the method used to distinguish malware from benign is different from one another. In this section, related studies are summarized based on proposed methods as well as measured performances.

Luckett *et al.* proposed a neural network based rootkit detection by using system call timing behaviors [8]. They collected system calls as well as corresponding execution times from the infected and uninfected system. They used KBeast rootkit for analysis. For training and detection, the zero is assigned for infected, 1 is assigned for uninfected. MATLAB is used to evaluate the collected features. They used two neural network architectures, namely, a static feedforward architecture, and recurrent nonlinear auto-regressive architecture, for experiments. According to the authors, the best experiment results obtained when combining a radial base and hyperbolic tangent transfer functions.

A static malware analysis technique, which uses static tools such as Bintext, PEID, PEview, MD5deep, Dependency walker, and IDA Pro as well as antivirus scanners, suggested in our previous study [9]. In the proposed method, collected malware and benign samples were analyzed under the static analysis tools with domain experts. For analyzed samples, important strings, reverse compiling results, hashes, and imported and exported functions were used to separate malware from the cleanware. The test case was performed on different versions of Windows virtual machines. As stated in the paper that antivirus scanners were fast and preferable for known malware, while static analysis tools performed better for zero-day malware

A malware detection method, which uses virtual memory access patterns was proposed by Xu *et al.* [10]. The study utilized hardware-assisted technique which monitored and classified memory access

patterns by using ML. According to the authors, malware may modify the control flow or data structures, which leaves fingerprint traits on program memory accesses. They used three markers including system calls, function calls, and the complete program run to collect the memory accesses. After the execution traces were collected, the feature selection was performed to specify the most relevant features. Finally, LR (Logistic Regression), SVM (Support Vector Machines) and RF classifiers were used for classification. As stated in the paper, the analysis rootkits were detected with 99% DR and less than 5% FPR.

Rosli *et al.* proposed a behavior-based detection method which uses registry data [11]. The suggested behavioral-based method used *k*-means clustering technique to separate malware based on the registry activities. According to the authors, the unsupervised clustering techniques are crucial in order to group the similar malware behaviors. In the proposed method, first, malware properties were extracted and chosen from the computer registry. Then, the *k*-means clustering detection model was used to separate suspicious behaviors from the normal ones. The experiment test results indicated that the proposed method clustered the normal and suspicious behaviors with more than 90% accuracy.

Bahador *et al.* presented a hardware level method which uses behavioral signatures to distinguish malware from the benign ones [12]. The proposed method used performance counter traces in order to detect and disable the malicious program samples immediately at the beginning of the execution. The paper stated that each behavior signature consists of a few number of singular values obtained from the hardware performance counter traces of known malware variants. When two performance counter traces are similar, the corresponding values should be proportional to each other. The collected malware and benign samples were analyzed under virtual machines. The test results indicated that the proposed method achieved the performance of 95.19%, 89.96%, and 92.50% for precision, recall, and f-measure, respectively.

Zhang *et al.* proposed a malware detection method which uses behavior chains [13]. Initially, the proposed method monitored the behavior points by using API calls. Then, the behavior chain was constructed by using the calling sequence of those behavior points. In the end, a LSTM (Long short-term memory) was used to specify malicious behaviors from the behavior chains. For experiment, 54.324 malware and 53.361 benign samples were collected and tested on Windows operating systems. The proposed method achieved 98.64% accuracy with less than 2% FPR.

The malware detection method for Industrial Internet of Things (IIoT) based on the behavior graph is proposed by Sun *et al.* [14]. API calls were obtained by running the malware and benign samples under Cuckoo sandbox. Then, parameter normalization and n-gram were applied to decrease the number of API traces. CBG (classified behavior graph) was created by selected APIs. The CBGs are the original program features which are generated for each sample. Similar behaviors were removed by using CNSG (crucial n-order subgraph). Then, CBGs were optimized to the key CBGs. Finally, NB, SVM, and AdaBoost classifiers were applied to key CBGs to separate malware from the benign samples. According to the papers, the proposed method achieved a high detection accuracy for tested malware and benign samples.

Azeez *et al.* proposed an ensemble learning approach to detect Windows PE malware strains [15]. The proposed approach consisted of fully connected and CNNs (Convolutional neural networks) which used the ExtraTrees classifier as a meta-learner. The base phase classification was performed by using a stacked ensemble of fully-connected and one-dimensional CNNs. In this phase for end-stage classification, machine learning (ML) classifiers were applied. Fifteen ML algorithms were used for meta-learner, and five ML algorithms including RF, NB, DT (decision tree), AdaBoost, and gradient boost were used for comparison. For experiments, Windows PE files were used. As stated in the paper, the more satisfactory results were obtained when an ensemble of seven neural networks as well as the ExtraTrees classifier for final-phase was used.

Rey *et al.* presented a federated learning at detecting malicious software samples in IoT devices [16]. They applied supervised as well as unsupervised federated methods to recognize malware variants for IoT devices. For this purpose, N-BaIoT dataset which consists of many real IoT devices' network traffic that were affected by malicious software were used. In addition, the obtained performances were

compared with participants which train a model locally (does not share the data) and participants which train a model globally (share data with the central). As mentioned in the paper, using the divergent data increases the performance of the federated models. To test the strength of the federated methods, adversarial attacks were used. According to the test results, the basic federated learning models are prone to adversarial attacks. However, their federated model is more robust to adversarial attacks. The robustness of the used federated learning model against adversarial attacks should be improved in the future.

A dynamic malware detection method which used IP (Internet protocol) reputation and ML techniques was explained by Usman *et al.* [17]. The suggested method computed the malicious behaviors of IP addresses' at run time. For this purpose, big data forensic was used to calculate the risk score of IP addresses. They used weighted risk score to identify the re-attempt of the causing damage, and confidence level to specify the degree of maliciousness. Paper stated that the experiments were carried out with a few ML classifiers including NB, SVM, MBK (mini batch k-means), DT, and best results were obtained when DT was used. The false alarm rate was high in the proposed method, in the future the false alarm rate should be reduced to improve the model performance.

Vu *et al.* suggested CNN-on-matrix technique to classify Android malware variants [18]. Each Android application was used as an image. First, for each application, the adjacency matrix was constructed. Then, constructed matrices were used as input images, and given to the CNN (Convolutional Neural Network) model. Finally, CNN model recognized each image sample as malware or benign. In this phase, the family of each malware apps were specified as well. According to the paper, the proposed approach could detect the Android apps with 94.3% when the drebin dataset was used and 97% accuracy is obtained for different malware families. The proposed method is limited with Android apps, and needs to be extended to support other mobile platforms.

In the related works, several malware detection methods were reviewed based on the suggested methods, used ML techniques and measured performances. Most of the studies used static and dynamic tools to analyze the malware. After the malware analysis stage was completed, data mining (DM) and ML techniques were applied to create malware features from the execution traces. Most of the papers were performed on specific malware variants or only a few malware samples which cannot be generalized to detect all malware types. In practice, current malware detection methods cannot detect unknown malware variants efficiently as well. The proposed method decreased the deficiencies that current studies have on malware detection and improved the detection and accuracy rates. Even though some of the existing studies results are close to the proposed method performances, the time complexity of the proposed method is lower. Besides, the proposed method can detect high percentage of the zero-day malware as well as different variants of malware including virus, worm, rootkit, ransomware, and obfuscated malware.

4. Proposed Method

We proposed a malware detection method which can be categorized in a behavioral-based approach that uses file and registry operations. In general, most of the malware performs operations on specific file directories and registry locations. This is because the majority of malware samples display similar behaviors on specific file and registry locations, which cannot be seen on benign samples. The proposed malware detection architecture seen in figure 2.

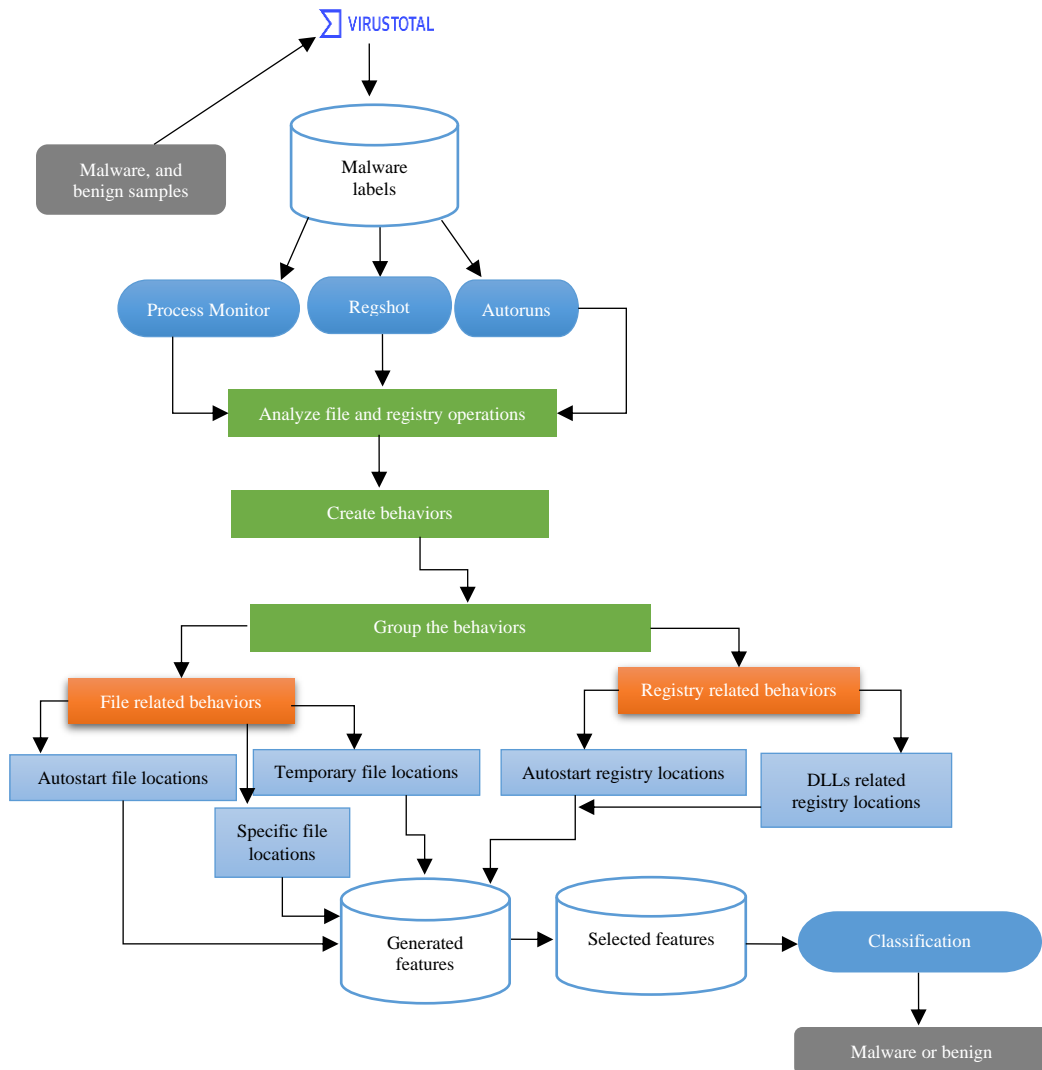


Figure 2 Malware detection architecture

Instead of all file and registry related operations, we only consider the file and registry operations which can rarely be seen in benign samples, however, mostly seen in malware samples. For instance, most of the malware types perform read and write operations to spread or inject itself into system processes or commonly used DLLs. In addition, most of the malware variants perform on specific folder locations including temporary file locations, as well as specific file and registry automatic startup (autostart) locations. When execution traces are collected following file-registry locations are considered:

1. Autostart file locations:
 - Shell:startup
 - Shell:common startup
 - %appdata%\Microsoft\Windows\Start Menu\Programs\Startup
 - C:\Users\USERNAME\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
 - C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
2. Temporary file locations:
 - %system%\Windows\Temp
 - C:\Windows\Temp
 - %userprofiles%\AppData\Local\Temp
3. Specific system file locations:
 - \Windows\System32

4. Autostart registry locations:

HKLM\Software\Microsoft\Windows\Currentversion\Run
 HKLM \Software\Microsoft\Windows\Currentversion\Runonce
 HKEY_Local_Machine\Software\Microsoft\Windows\Currentversion\Run
 HKCU\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
 HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run
 HKCU\Control Panel\Desktop\Scrnsave.exe
 HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices
 HKLM\ Software \Wow6432Node\Microsoft\Active Setup\Installed
 HKLM\ Software \Microsoft\Windows\CurrentVersion\Explorer\ShellServiceObjects
 HKLM\ Software \Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
 HKLM\ Software\ Microsoft\Windows NT\ CurrentVersion\Winlogon\UserIni
 HKLM\Software\Microsoft\Windows NT\CurrentVersion\Drivers32

5. DLLs related registry locations:

HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs
 HKLM\Software\Microsoft\Windows NT\ CurrentVersion\ Windows\AppInit_DLLs

After the execution traces are collected based upon specific file-registry locations, behaviors and features are created. When creating behaviors from the file-registry operations, one or more operations can create behaviors on the same file and registry instance. When creating features from the behaviors, ten consecutive orders are placed and behaviors are grouped based on file and registry operations. The same behaviors on different instances of the same resources match into the same feature, but the frequency of the feature is increased. Even if the behaviors are performed on different resources including file and registry, they can create features when relation is observed based on used locations. After the feature generation process is completed, the frequency of each feature is computed.

When the feature generation process is completed, most significant features are selected by using information gain. The information gain selects the features with maximum gain which decrease the information needed for the next split. Most of the time, if the dataset is properly constructed, the information gain is chosen for the most significant features in the dataset. We can compute the information gain as follows:

$$\text{Information gain}(A) = \text{Information}(D) - \text{Information}_A(D) \quad (1)$$

$$\text{Information}(D) = -\sum_{j=1}^v p_j \log_2(p_j) \quad (2)$$

$$\text{Information}_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \log_2\left(\frac{|D_j|}{|D|}\right) \quad (3)$$

$\text{Information}(D)$ shows the average amount of information needed to identify the class labels in the dataset, while $\text{Information}_A(D)$ indicates the amount of information needed after each partitioning during classification for features. After the most significant features are selected by using information gain feature selection criteria, the learning and testing phases are performed. For classification, well-known ML algorithms are used including C4.5 (J48), RF, SLR, AdaBoost, SMO, and KNN. We present the effectiveness of our proposed method by applying and comparing the outcomes of these ML algorithms.

5. Implementation

For the experiments, Windows 10 was used as a host machine with Oracle VirtualBox installed. Windows 7 and Windows 8 are installed on Oracle VirtualBox as guest machines. The collected malware and benign samples are performed on virtual machines Windows 7, 8, and 10. The malware samples are collected from several sources including ViruSign, Malshare, and Tekdefense [19, 20, 21]. The collected malware samples are from different malware types such as virus, worm, rootkit, backdoor, ransomware, spyware, and so on. The collected malware samples are labeled by using VirusTotal.

VirusTotal is a website which contains several antivirus scanners. The tested benign files are different system and third party' softwares. After each malware sample performed, the execution traces of file-registry operations are collected by using Process Monitor as well as Regshot, and Autoruns. Each time, the clean version of the guest machine is used. The collected execution file-registry traces are analyzed by using Python language in Windows 10 environment.

Totally, 582 malware and 300 benign samples are tested. After the file-registry traces are collected, behaviors are formed. Only five types of file and registry operations are used for behavior creation: autostart file locations, temporary file locations, specific system file locations, autostart registry locations, and DLLs related registry locations. After behaviors are created, features and their frequencies are computed. During the feature creation, ten consecutive orders are used. Then, most significant features are chosen by using information gain measures. That way, the most distinctive file and registry based features are generated. To correctly classify the most distinctive features, which are mostly seen in malware but rarely seen in benign samples, ML classifiers including C4.5 (J48), RF, SLR, AdaBoost, SMO, and KNN are used.

Our dataset consists of different types of malware including virus, worm, rootkit, backdoor, Trojan, ransomware and packed malware. To label the malware samples, the VirusTotal is used which contains several antivirus scanners. To identify each malware sample, the kind of file signature MD5 (message-digest algorithm) hashes are used. For each feature, the frequency of the properties is written into the dataset. If the related feature is not given, 0 is written as a frequency.

To measure the feasibility and efficiency of the proposed method: TPR, FPR, precision, and accuracy are used on our created dataset. Confusion matrix is used to calculate these values (Table 1).

Table 1 Confusion Matrix

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

In the confusion matrix, TP shows the number of malware samples correctly labeled as malware, TN shows the number of benign samples correctly labeled as benign, FP represents the number of benign samples being accidentally labeled as malware, and FN represents the number of malware mistakenly labeled as benign. These values are used to compute the TPR, FPR, precision, and accuracy as follows:

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

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

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) \quad (6)$$

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}) \quad (7)$$

6. Results and Discussion

This section shows the experiment results and evaluates the model performances. Learning and testing phases are performed on classifiers by using 10-fold cross-validation and holdout 75%, 25% split. The results are summarized in table 2, figure 3, table 3, and table 4.

Table 2 presents the various ML algorithms' performances on created malware dataset. The performances of classifiers based on TPR, FPR, and precision metrics are quite high. For example, when RF is selected as a classifier TPR, FPR, and precision measured as 98.6%, 0%, and 100%, respectively. In the same way, AdaBoost performance measures as 98.8%, 3.7%, and 98.1%, respectively. Similar performances are obtained when SLR, and J48 are used. The performances of KNN and SMO are lower

than other classifiers. Our performance results present that the proposed method can effectively separate malware from benign samples.

Table 2 Proposed method performances on different ML classifiers

Classifier	TPR (%)	FPR (%)	Precision
RF	98.6	0	100
AdaBoost	98.8	3.7	98.1
SLR	98.1	0.9	99.5
J48	98.1	1.8	99
KNN	92.6	4.2	97.9
SMO	88	13.3	92.8

Figure 3 shows the accuracy results on several ML algorithms. As it can be seen from the figure 3 that the best accuracies are obtained in order of RF, SLR, J48, AdaBoost, KNN, and SMO. Since, information gain is used as a feature selection, RF (99.05% accuracy), SLR (98.42% accuracy), J48 (98.11% accuracy), and AdaBoost (97.95% accuracy) classifiers performed pretty well. However, SMO classifier accuracy is measured as 87.52% which is lower than other classifiers. When we use correlation coefficients for the feature selection process, the performance of the SMO is increased up to a certain degree.

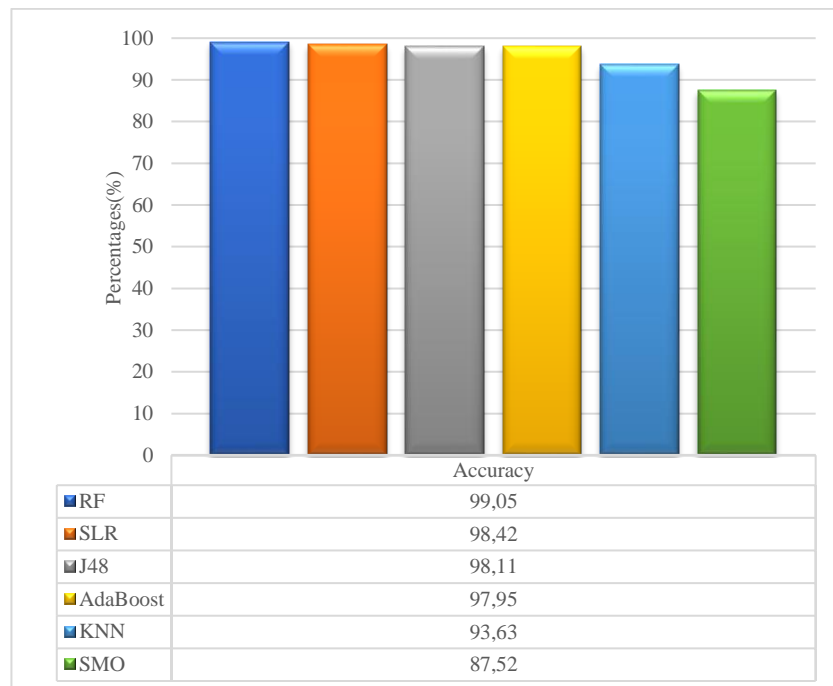


Figure 3 Various ML Classifiers accuracy on created dataset

Our dataset consists of 45 features. The table 3 shows the 27 file-registry related properties (which are generally seen in autostart file locations, temporary file locations, specific system file locations, autostart registry locations, and DLLs related registry locations) mostly seen in malware files with high frequency and rarely seen in cleanware. Even if a few of the listed features may be seen in cleanware, the frequency of those properties are very low when compared to the frequency of properties counted in malware samples. CreateFileReadFile, ReadFileWriteFile, WriteFile, SetBasicInformationFile, CreateFileWriteFile, CreateFileMapping, QueryStandardInformationFile, QuerySecurityFile, RegOpenKey, RegQueryValue, RegSetInfoKey, RegCreateKey, RegSetInfoKeyRegEnumKey, and RegDeleteValue properties are mostly seen in malware samples with high frequencies in substantial file and registry locations.

Table 3 The list of file and registry related properties which mostly seen in malware seldomly seen in cleanware files (The order of the features are not preserved)

ReadFileLoadImage
ReadFileWriteFile
WriteFileCreateFile
WriteFile
CreateFileSetBasicInformationFile
SetBasicInformationFile
CreateFileWriteFile
WriteFileCreateFileMapping
CreateFileReadFile
QueryBasicInformationFileReadFile
QueryBasicInformationFileCreateFileMapping
CreateFileMapping
CreateFileMappingLoadImage
LoadImageReadFile
QueryBasicInformationFileQueryDirectory
CreateFileMappingCreateFile
QueryStandardInformationFile
QuerySecurityFile
RegOpenKey
RegQueryValue
RegSetInfoKey
RegSetInfoKeyRegQueryKey
RegCreateKey
RegSetValue
RegQueryKeyRegSetInfoKey
RegSetInfoKeyRegEnumKey
RegDeleteValue

Table 4 shows the comparison results of the proposed method against the state-of-the-art methods in the literature. The proposed method generated remarkable results among the other methods. For instance, the proposed behavioral-based file-registry operations performance was measured as 99.05% when RF was used as a classification algorithm (Table 4). On the other hand, system call timing behaviors performance was 82.8% and memory access patterns performance was 88.4%. The best performance obtained from the sequence of behavioral points (behavioral chains) by 98.64% which was still lower than the proposed method performance (99.05%). Besides, the proposed method's feature space is much lower than the other methods that are mentioned in table 4.

Table 4. Shows the proposed method performances against leading methods in the literature

Paper	Year	Feature Representation	ML Algorithm	Performance(%)
Lockett <i>et al.</i> [8]	2016	System call timing behaviors	Neural Network	82.8
Xu <i>et al.</i> [10]	2017	Memory access patterns	RF	88.4
Rosli <i>et al.</i> [11]	2019	Behavior-based registry data	K-Means	90
Bahador <i>et al.</i> [12]	2019	Behavioral signatures on performance counter traces	RF	82.65
Zhang <i>et al.</i> [13]	2020	Sequence of behavioral points	LSTM	98.64
Sun <i>et al.</i> [14]	2021	Classified behavior graphs	SVM	97
Proposed Method	2022	Behavioral-based file-registry operations	RF	99.05

7. Conclusion

The number, severity, and complexity of malware have been increasing rapidly. To protect the computer based systems from malware, malware needs to be detected whenever it infect the victim system. However, the detection of malware becomes harder since new malware variants are more intelligent and use obfuscation techniques to hide themselves from the malware detection systems.

This paper proposed a new behavioral-based malware detection method based on file-registry operations. Only the operations which are performed on specific file and registry locations are considered during the features generation. These features are seen frequently in malware samples rarely

seen in benign samples. Hence, most of the malware variants are detected with our approach. After features are generated, information gain is used for feature selection. Finally, several machine learning classifiers including RF, J48, AdaBoost, SLR, SMO, and KNN are used for classification. The test case is performed on Windows virtual machines 7, 8, and 10. Proposed method could effectively detect the malware with high percentages. For instance, 98.8% true positive rate, 0% false positive rate, and 99.05% accuracy are obtained to distinguish malware from cleanware. As a future study, we aim to analyze more malware and benign samples, and also examine network related features by using Wireshark.

References

- [1] Ö. Aslan, R. Samet, "Investigation of possibilities to detect malware using existing tools," *IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)* pp. 1277-1284, October 2017.
- [2] Ö. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, 8, 6249-6271, 2020.
- [3] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, 8(1), 1-22, 2018.
- [4] Ö. Aslan, R. Samet and Ö.Ö. Tanrıöver, "Using a Subtractive Center Behavioral Model to Detect Malware," *Security and Communication Networks*, 2020.
- [5] J. Nazari, "Defense and Detection Strategies against Internet Worms," Artech House, 2004.
- [6] S. Sparks and J. Butler. "Shadow walker: Raising the bar for rootkit detection," *Black Hat Japan*, 11(63), 504-533, 2005.
- [7] K. Savage, P. Coogan, and H. Lau, "The evolution of ransomware," Symantec report, August 2015, available at: <https://its.fsu.edu/sites/g/files/imported/storage/images/information-security-and-privacy-office/the-evolution-of-ransomware.pdf>.
- [8] P. Lockett, J. T. McDonald and J. Dawson, "Neural network analysis of system call timing for rootkit detection," *Cybersecurity Symposium (CYBERSEC)* (pp. 1-6), April 2016.
- [9] Ö. Aslan, "Performance comparison of static malware analysis tools versus antivirus scanners to detect malware," In *International Multidisciplinary Studies Congress (IMSC)*, 2017.
- [10] Z. Xu, S. Ray, P. Subramanyan and S. Malik. "Malware detection using machine learning based analysis of virtual memory access patterns," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017 (pp. 169-174), March 2017.
- [11] N.A. Rosli, W. Yassin, M. A. Faizal and S. R. Selamat. "Clustering Analysis for Malware Behavior Detection using Registry Data," *International Journal of Advanced Computer Science and Applications (IJACSA)*, 10, 12, 2019.
- [12] M. B. Bahador, M. Abadi and A. Tajoddin, "HLMD: a signature-based approach to hardware-level behavioral malware detection and classification," *The Journal of Supercomputing*, 75(8), 5551-5582, 2019.
- [13] H. Zhang, W. Zhang, Z. Lv, A. K. Sangaiah, T. Huang and N. Chilamkurti. MALDC: "A depth detection method for malware based on behavior chains," *World Wide Web*, 23(2), 991-1010, 2020.
- [14] Y. Sun, A. K. Bashir, U. Tariq and F. Xiao, "Effective malware detection scheme based on classified behavior graph in IIoT," *Ad Hoc Networks*, 102558, 2021.
- [15] N. A. Azeez, O. E. Odufuwa, S. Misra, J. Oluranti and R. Damaševičius, "Windows PE malware detection using ensemble learning," In *Informatics*, (Vol. 8, No. 1, p. 10). Multidisciplinary Digital Publishing Institute, 2021.
- [16] V. Rey, P. M. Sánchez, A. H. Celdrán and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, 108693, 2022.

- [17] N. Usman, S. Usman, F. Khan, M. A. Jan, A. Sajid, M. Alazab and P. Watters, "Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics," *Future Generation Computer Systems*, 118, 124-141, 2021.
- [18] L. N. Vu, and S. Jung, "AdMat: A CNN-on-matrix approach to Android malware detection and classification," *IEEE Access*, 9, 39680-39694, 2021.
- [19] [Online]. Available: <https://www.virusign.com/> [Accessed in November 2021].
- [20] [Online]. Available: <https://malshare.com/> [Accessed in November 2021].
- [21] [Online]. Available: <http://www.tekdefense.com/> [Accessed in November 2021].