

AN INFORMATION RETRIEVAL APPROACH FOR MALWARE CLASSIFICATION BASED ON WINDOWS API CALLS

JULIA YU-CHIN CHENG, TZUNG-SHIAN TSAI, CHU-SING YANG

Institute of Computer and Communication Engineering, National Cheng Kung University, Taiwan

E-MAIL: {julia.yc.cheng, fh552001}@gmail.com, csyang@ee.ncku.edu.tw

Abstract:

Automated malware toolkits allow for easy generation of new malicious programs. These new executables carry similar malicious code and demonstrate similar malicious behavior on infected hosts. In order to speed up the efficiency of malware detection, discriminating a malware as known or a new species of malware has become a critical issue in the security industry. In this paper, we propose a new approach to precisely classify malicious executables by employing information retrieval theory. Dynamic analysis of a sample's sequence of Windows API function calls produces corresponding parameters and values which is used as input to a standard TF-IDF weighting scheme to identify malware families by their behavior characteristics. Irrelevance reduction is developed to filter out non-relevant features and improve accuracy of malware classification. Finally, a similarity measure is used to determine the most similar malware family to the tested samples.

Keywords:

Malware classification; Information retrieval; Windows API calls; Similarity measure; TF, IDF

1. Introduction

As the frequency of security threats increase and the number of incident rises, security organizations recognize that malware evolution is changing towards efficient malicious code generation as well as antivirus evasion tactics. In order to achieve fast generation and antivirus evasion, malware authors create toolkits for automated generation of new malware that carry malicious code and newer evasion techniques inside.

In these circumstances, the numbers of new malware samples grows exponentially but the capabilities and quantity of behavior analysis for signature generation cannot keep up. Accordingly, the large amount of new malware leads to failure of malware classification and precise identification by antivirus

software. According to a McAfee report, 133 million samples of malware was discover in 2012 [1].

For signature generation to update antivirus software, malware behavior analysis is used to understand the detail of what happen at infected host. Generally speaking, there are two kinds of behavior analysis methods: static analysis and dynamic analysis [2]. Static analysis inspects the assembly code binary determine retrieve code flow and sequential instructions to identify the malicious sample without actually executing the sample. The advantage of this approach is clarity of the code flow but the big disadvantage of static analysis is the ambiguous. Ambiguous results may arise when a malware coder uses evasion detection techniques such as packing, compressing or garbage instructions to self-modify the architecture of content flow [3]. Furthermore, pieces of malicious code triggered by specific values such as system date, specific mutex name or jump instructions exacerbate the intricate analysis of the ambiguous code flow.

The dynamic analysis method examines the actions performed by malware while it is being executed in a controlled and safe environment. This means dynamic analysis monitors run-time behavior to determine which function calls are intercepted sequentially, also called hooking, by a malware to determine an overview of malicious behavior in a virtual machine. These calls can be examined to identify characteristic actions such as creating a file or transmitting TCP packets. In the Windows operation system, Windows Application Programming Interface (API) function calls [11] and system calls provide different layers of functionality, for instance, networking, file handling, system services, etc. Take the API function *WriteFile* as an example: *WriteFile* opens a file handler and performs write operations to the handler. Thus, determining which Windows API function calls are hooked by malware is sufficient to identify behavior during execution. Dynamic analysis can overcome the shortcoming of static analysis on ambiguous and

self-modifying code and allow for logging vulnerable behavior is a runtime environment. The limitation of dynamic analysis is the inability to map comprehensive functionalities of malicious executables.

Malware classification algorithm are implemented by employing the results of dynamic behavior analysis or static code analysis. Current research studies on malware classification present two major analysis approaches to analyze and classify malware samples. One approach is to use op-code sequential blocks extracted from static analysis [4], [5] to differentiate malware families using binary code comparison. The other is to use API function calls and system calls from dynamic analysis tools to train the algorithm in order to predict and classify malware samples [6], [7], [8]. These studies only take specific function calls and strings triggered from specific function calls into consideration rather than the complete function call sequences hooked by the malware. This decrease the accuracy of malware classification if the characteristic function calls are not selected correctly.

In this paper, we aim to propose an approach to classify malware samples into known malware families by employing Information Retrieval (IR) theory [9]. These samples were collected from honeypots over four months almost all seem to be generated from malware toolkits because of similar behavioral patterns reflecting their malicious intention on the infected hosts. After sample collection, a dynamic analysis sandbox tool - Cuckoo Sandbox [10] was used to extract API function calls from the malware samples. API function calls consisting of parameters and the corresponding values of each malicious executable can be completely transformed into a document using the irrelevance reduction process, and then apply similar measure on the document to classify malware samples into the most relevant malware family.

The rest of the paper is organized as follows: Related work is presented in Section 2. Section 3 describes the proposed approach. The experiment setup and results are presented in Section 4. Conclusions follow in Section 5.

2. Related Work

Recently, researchers are interested in analyzing and classifying malware using the sequence of Windows API function calls to model malware behavior. Malware writers to perform malicious actions by hooking Windows API calls [11]. Therefore, the analysis of the API call sequence will give a complete understanding of malicious behavior. Initially, the API function calls from a known malware family are extracted to construct a training dataset for the family. The API calls of unclassified

samples are then compared with that of known malware families to determine the most relevant families. Many machine learning algorithm classifiers have been developed to achieve the goal of malware classification.

Rieck et al. [6] determined the behavior patterns of malware families and ranked for classification by applying the support vector machine (SVM) algorithm. For each malware family, specific strings of API calls are required to learn and classify the behavior patterns. From the experimental results, the accuracy of classification is lower if API call strings are not selected precisely for dividing hyperplane vectors or if the selected API call strings are highly similar among several families. To classify malware families, printable strings contained in each sample are input to various algorithms including Naive Bayes (NB), Detection Tree (DT), AdaBoost, SVM and Random Forest (RF) in [8]. These strings could be API function names, the corresponding values triggered by the API call, and meaningless or unknown strings. Without an efficient feature selection mechanism, these strings cause classification overhead. Veeramani and Rai [12] proposed statistical analysis to model malware behavior based on API calls. This proposed framework aimed to increase the accuracy rate of categorizing executables by using the Document Class Wise Frequency (DCFS) to measure the relevant API calls for classifying the executable as malicious or benign.

For current studies, profiling representative behavioral patterns regardless of algorithms is difficult if only specific API calls are selected for learning and classification. In this paper, the comprehensive API function calls sequence of each sample is collected as a document. TF-IDF weighting and similarity measure of documents by IR theory, which represents the comprehensive malicious behavior, are used to profile and retrieve the malware families.

3. Approach Description

For the goal of discrimination between families of malware, IR theory is used for obtaining informative features relevant to differentiate malware families in a collection of malicious executables. An IR-based malware classification model is defined as 4-tuple $[\mathcal{M}, \mathcal{Q}, \mathcal{T}, \mathcal{S}(q_i, m_j)]$ where $\mathcal{M}$ is a set of representations of executables of known families; $\mathcal{Q}$ represents malicious executables transformed into a set of queries $(q_0, q_1, \dots, q_k)$; $\mathcal{T}$ details how to profile malicious executables with the vector representation of behavior features $(t_0, t_1, \dots, t_r)$ and $\mathcal{S}(q_i, m_j)$ is a score for calculating a similarity measure function to find the most relevant malware family m_j to a query q_i . In this paper, the behavioral feature can

be regarded as the "term" in IR theory.

3.1. Malware Classification Model

Figure 1 illustrate the overall process of how malicious executables are profiles and classified. The model starts with the generation of malware behavior analysis using a dynamic analysis sandboxing tool for a malware corpus that has been pre-identified into families by antivirus software. The behavior features are the sequential windows API calls consisting of API function names, parameters and the corresponding value. Next, a behavioral features profiling process is used to model the behavior of malware samples by weighting the frequency and relevancy of terms with the representation of vectors. In order to improve the accuracy of family classification, "Irrelevance Reduction" is applied to ignore features that are marginally unrelated, redundant or noisy. For example the size and address of memory allocation appears on almost malware samples and is not an essential feature for family identification. In this profiling phase, terms weighting vectors of classified samples can represent the characteristics of malware families. Finally, the similarity measure score is applied to retrieve the most similar malware family by the "Family Retrieval" process. If no similar malware family is retrieved, the sample will be declared as unknown family for next re-profiling phase to be inspected and defined as a new malware family or an existing families when the behavioral feature is updated.

3.2. Malware Corpus

Malware executables were collected to assemble a malware corpus $\mathcal{M}$ using honeypot tools such as Dionaea [13] and [?]. This corpus contained only Portable Executable (PE) files executed on all MS 32-bit operation systems. The malware corpus was scanned using anti-virus software ClamAV [15] to determine their malware family and to split the malware corpus into known and unclassified samples. The known malware corpus $\mathcal{M}$ is retrieved to differentiate malware families ($m_0, m_1, \dots, m_n$) in the profiling phase while unclassified samples are classified in the retrieval phase to verify the proposed approach.

3.3. Behavior Analysis Generation

After the malware corpus was collected and scanned by antivirus software, the next step is to identify the malware behavior for each executable on runtime observation. A powerful open-source sandbox tool, Cuckoo Sandbox [10], automat-

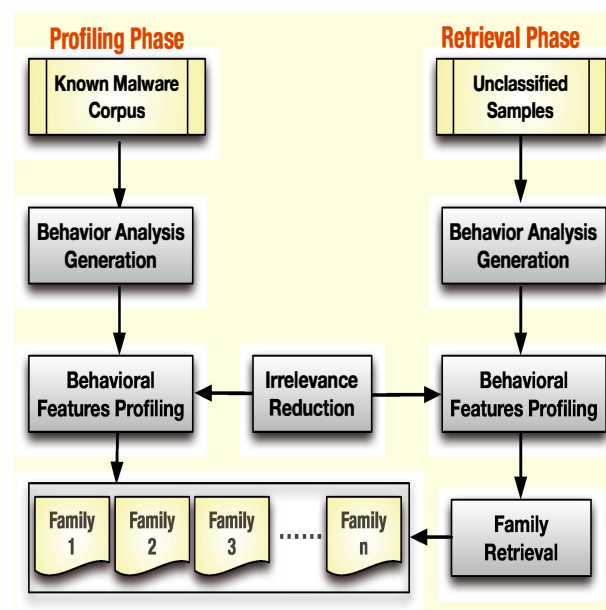


Figure 1. IR-based Malware Classification Model

ically monitors state changes performed by executing a malware in a controlled environment. Each API call contains the API function name, parameter and the corresponding parameter value. Compared with previous work [6], [8], [12], specific API function names and specific strings both are used to learn and classify malware families for machine learning algorithms. This results in specific behavior patterns to be analyzed heavily such as modification of file system, changes to the Windows registry and so on, but does not reflect the comprehensive behavioral characteristics. In this paper, we combine the API function name, its parameter and the corresponding value as a behavior feature, called a "term" in the IR domain, for malware classification. Figure 2 shows the behavior feature. The API function name, parameter and its corresponding value of each line are joined as a behavior feature.

3.4. Irrelevance Reduction

Irrelevance reduction on behavior feature is essential to decrease the dimensions for behavior profiling. Two stages are needed. The first stage is to measure the relevance of the function name to the malware family by *Mutual Information (MI)* shown as Equation (1):

No.	Function Name	Parameter	Values
1	VirtualAllocEx	flAllocationType	0x00003000
2	VirtualAllocEx	flProtect	0x00000040
3	LoadLibraryA	lpFileName	user32.dll
4	CreateFileW	lpFileName	C:\WINDOWS\system32\inet.exe
5	CreateFileW	lpFileName	C:\WINDOWS\system32\KERNEL32.dll
6	CreateFileW	lpFileName	C:\WINDOWS\system32\USER32.dll
7	CreateFileW	lpFileName	C:\WINDOWS\system32\ADVAPI32.dll
8	CreateFileW	lpFileName	C:\7ac1a1de4fa47da43eebca8107d
9	CreateFileW	lpFileName	C:\11656f015654902321c13605b232cc46
10	LoadLibraryA	lpFileName	C:\WINDOWS\System32\winrmr.dll
11	RegOpenKeyW	hkey	0x00001038
12	RegOpenKeyW	hkey	0x00002cc4
13	RegOpenKeyW	lpSubKey	Software\Microsoft\Windows NT\CurrentVersion\winlogon
14	CreateFileW	lpFileNam	C:\WINDOWS\Explorer.exe
15	CreateServiceA	lpServiceName	NET Service
16	CreateServiceA	dwServiceType	0x00000110
17	CreateServiceA	lpBinaryPathName	C:\WINDOWS\trkwsvc.exe
18	CreateFileW	lpFileName	\\.\p

Figure 2. Behavioral Features List

$$MI(F, M) = \sum_{f \in F} \sum_{m \in M} \log\left(\frac{p(f, m)}{p(f)p(m)}\right) \quad (1)$$

where F is the frequency of function names extracted from the function name column and M is the category of malware families. The function $p(f, m)$ is the joint probability distribution function of f and m , and $p(f)$ and $p(m)$ are the marginal probability distribution function of f and m . Rules are defined to remove irrelevant behavior features (R), to bypass the parameter and value and keep the function name only (B), and to substitute the specific pre-defined string for the string on the value field (S) on the second stage. Rules are given in Table 1.

For example, the lines 1-2 of behavioral features list will be removed from the list if rule 1 is applied to match the function name *VirtualAllocEx*. This API call initializes memory within the virtual address of a process and is unimportant for classification. Rule 6 is used to check for the features of line 5-7 of behavioral features list. The file *KERNEL32.dll*, *USER32.dll*, and *ADVAPI32.dll* in the value column belonging to benign all files is replaced with *benign.dll*. We keep *net.exe* on the line 4 because it is created by malware. Through irrelevance reduction, non-relevant features will be reduced and relevant features will be reserved for characterizing malware behavior.

3.5. Behavioral Feature Profiling

The well-known vector space $tf * idf$ has been adopted in this paper to profile behavior features with the vector of rep-

Table 1. Rules to Reduce Irrelevance

No.	Rule Action	Rule Description
1	(R): Reduce	Target: (R): Process Memory Function name is related to memory allocation or process memory
2	(R): Reduce	Target: (R): Registry Key Open the registry key or set the value to registry key
3	(R): Reduce	Target: (R): Device IO Control Open the registry key or set the value to registry key
4	(B): Bypass	Target: (R): File Pointer File pointer by read and write file operation points to input/output device.
5	(S): Substitute	Target: (R): Malware Name Malware name appears on the value field. The malware name is replaced with "infected" string.
6	(S): Substitute	Target: (R): Common dll files The benign dll files on C:\WINDOWS\system32*.dll appears on the values field. Replace the filename with "benign.dll"

resentation for pre-classified malware families and unclassified samples. A vector representing a malware sample is made up of weighted frequencies of behavior features. We denote w_{ij} to represent the weights of feature t_j in the malware families m_i and define as Equation (2):

$$w_{ij} = tf_{ij} * idf_j \quad (2)$$

where tf_{ij} is the occurrence frequency of the feature t_j in m_i , idf_j the inverse document frequency computed as $\log(n/df_j)$, with n indicating the number in the malware corpus and df_j is the number of samples containing the feature t_j .

3.6. Family Retrieval

The retrieval of the most similar family for a sample is the goal of this process. Once the query vector composed of weighted frequencies of behavioral feature is calculated, it is compared with the vector of each malicious executable of all malware families. In this way, the family retrieval process aims to measure how similar the query is to other executables from all malware families. The score $S_{max}(\mathcal{M}, q_k)$ is to find the most relevant family in $\mathcal{M}$ giving the query q_k . Hence, for

each m_i in $\mathcal{M}$ we calculate the score by applying the following Equation (3):

$$S(m_i, q_k) = \frac{\sum_{t_j \in q_k} w_{ij} * w_{kj}}{\sqrt{\sum_{t_j} (tf_{ij} * idf_j)^2} \sqrt{\sum_{t_j} (tf_{kj} * idf_j)^2}} \quad (3)$$

where w_{ij} and w_{kj} respectively represent the weights of feature t_j in the malware executable m_i and in the query q_k . $S_{max}(\mathcal{M}, q_k)$ is the maximum of $S(m_i, q_k)$

4. Experiment Results

To evaluate the performance and effectiveness of our proposed approach. malware samples were collected via Dionaea [13] and Thug [14]. Dionaea is a low-interaction honeypot to emulate vulnerable Windows network services for malware collection. Thug is low-interaction honeyclient, designed to detect exploit code and gather malware samples from malicious websites. After malware collection, antivirus engine ClamAV [15] is applied to identify their malware families. Ten well-known malware families are selected to compose a malware corpus. These families and the occurrence of each malware family are shown in Table 2.

Table 2. 10 well-known families

1: trojan.VB (23)
2: Trojan.Mybot (130)
3: Trojan.Vanbot (133)
4: Worm.Sasser (17)
5: Win.Trojan.Zbot (23)
6: Worm.Palevo (9)
7: Trojan.Gobot (35)
8: Trojan.Bredolab (10)
9: Worm.Padobot (37)
10: Worm.Korgo (8)

The malware corpus is randomly split into known samples for the profiling phase and unclassified samples for the retrieval phase using a 60% to 40% split. For each partition, behavior analysis reports are generated profiled as the vector of representation. Similarity measure equation is used to retrieve the most relevant family as discussed in Section 3.

In the first experiment we examine that irrelevance reduction is essential to filter behavioral features and raise the accuracy of malware classification. The number of behavioral features is

14285 without irrelevance reduction and **2420** with irrelevance reduction. In Figure 3 the per-family accuracy with and without irrelevance reduction (RR) is shown.

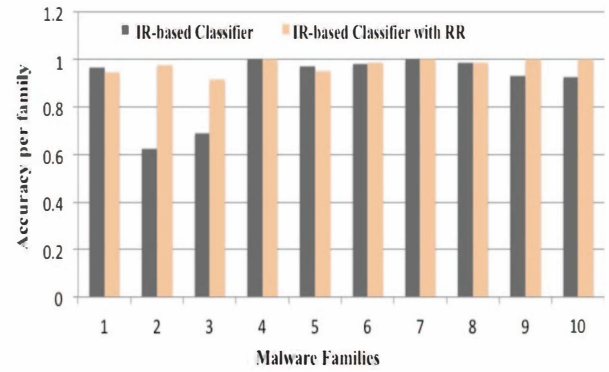


Figure 3. Per-family accuracy with and without RR

With irrelevance reduction, on average, 97% of unclassified binaries are correctly classified to their malware families. In particular, the accuracy of Trojan.Mybot and Trojan.VanBot is enhanced from 62.6% to 97% and from 69% to 91.4%. The enhanced accuracy of detection for these two malware families demonstrates the effectiveness of irrelevance reduction on the proposed IR-based malware classifier. Trojan.Mybot and Trojan.VanBot have many variants. The improved accuracy rate indicates that irrelevance reduction efficiently filters shared behavior characteristics to increase the performance of malware classification. Compared to Rieck et al. [6], using pre-defined specific API calls and strings to train a malware classifier for each family using the SVM algorithm gives an average accuracy rate of 88%. The characterizing of behavioral features in [6] could be unilateral and cause the low accuracy on specific families.

The second experiment presents the performance comparison of various algorithms. The same malware corpus split into training and testing partitions is used to examine various classifiers. Irrelevance reduction is also employed for the malware corpus. The common standard metrics measures the classification performance including 1) *Precision*- the proportion of predicts positively, 2) *Recall*- the proportion of measuring correct families that were assigned and 3) *Accuracy*- the proportion of decisions that were correct. Table 3 presents the average experimental results grouped according to classifiers.

We applied the same malware corpus to Random Forest, SVM, Nave Bayes and proposed IR-based classifiers for profil-

Table 3. Classification Performance Comparison on Various Classifiers

Classifier	Precision	Recall	Accuracy
Random Forest	0.85	0.825	0.83
SVM	0.511	0.634	0.633
Naive Bayes	0.818	0.806	0.806
Proposed Approach	0.907	0.802	0.977

ing and classified. It was surprising that SVM gives the weakest results, while the other classifiers are above 80% and our proposed approach achieves the best accuracy 97%. We infer noisy behavioral features which mislabeled families dramatically decrease the performance. Another possible reason is that we did not use the proper kernel function to construct hyperplanes for classification.

Our results show that the IR-based approach can be used to achieve high accuracy of malware classification. It also demonstrates that Windows API calls merging parameters and values with irrelevance reduction are useful for malware classification. In [8], various classifiers including Naive Bayes, SVM, Random Forest, Decision Tree and IB1 achieve over 92% accuracy while our experiment results do not present high accuracy. We conclude the major reasons are the effectiveness of modeling the behavioral characteristics of malware families and the dataset selection on malware families affect the results.

5. Conclusions

The main contribution of this work is an IR-based approach for automatic malware classification. The major ideas are to characterize malicious behavioral features according to their weights and to identify the most relevant malware families using similarity measure. Irrelevance reduction is also applied to remove unrelated features for increasing the performance of malware classification. From the experimental results, the proposed approach achieves the high accuracy of malware classification.

Finally, we examine the limitations of this approach. One drawback is that we rely on the efficacy of dynamic analysis tools. If the tool is not able to explore the API calls sequence triggered by malicious executables, the approach cannot analyze or classify the executables. Currently, some malwares combining evasion techniques to avoid execution in the sandboxing environment increase difficulty of malware classifica-

tion. The other weakness is the proposed method cannot classify new families that are not already profiled. This reserved to be resolved in future work.

References

- [1] McAfee Labs, McAfee Threats Report: Fourth Quarter 2012.
- [2] Egele, Manuel, et al. "A survey on automated dynamic malware-analysis techniques and tools." *ACM Computing Surveys (CSUR)* 44.2 (2012): 6.
- [3] Hu, Xin. *Large-Scale Malware Analysis, Detection, and Signature Generation*. Diss. The University of Michigan, 2011.
- [4] Santos, Igor, et al. "Idea: Opcode-sequence-based malware detection." *Engineering Secure Software and Systems*. Springer Berlin Heidelberg, 2010. 35-43.
- [5] Christodorescu, Mihai, Somesh Jha, and Christopher Kruegel. "Mining specifications of malicious behavior." *Proceedings of the 1st India software engineering conference*. ACM, 2008.
- [6] Rieck, Konrad, et al. "Learning and classification of malware behavior." *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer Berlin Heidelberg, 2008. 108-125.
- [7] Al Amro, Sulaiman, and Antonio Cau. "Behavioural api based virus analysis and detection." (2012).
- [8] Tian, Ronghua, et al. "An automated classification system based on the strings of trojan and virus families." *Malicious and Unwanted Software (MALWARE)*, 2009 4th International Conference on. IEEE, 2009.
- [9] Salton, Gerard, and Michael J. McGill. "Introduction to modern information retrieval." (1986).
- [10] Cuckoo Sandbox. Available from: <http://www.cuckoosandbox.org/>
- [11] Father, Holy. "Hooking Windows APITechnics of hooking API functions on Windows." *CodeBreakers J* 1.2 (2004).
- [12] Veeramani, R., and Nitin Rai. "Windows API based Malware Detection and Framework Analysis."
- [13] Dionaea-catch bugs. Available from: <http://dionaea.carnivore.it>
- [14] Thug-Python low-interaction honey client. Available from <https://github.com/buffer/thug>
- [15] ClamAV Antivirus. Available from: <http://www.clamav.net/lang/en/>