# Malware Detection using Windows Api Sequence and Machine Learning

Chandrasekar Ravi, R Manoharan
Department of Computer Science and Engineering,
Pondicherry Engineering College,
Pillaichavady, Puducherry - 605014, India

## ABSTRACT

Monitoring the behavior of program execution at run-time is widely used to differentiate benign and malicious processes executing in the host computer. Most of the existing run-time malware detection methods use the information available in Windows Application Programming Interface (API) calls. The proposed malware detection system uses the Windows API call sequence. A 3rd order Markov chain (i.e. 4-grams) is used to model the API calls. This composite feature set is provided as an input to the malware detection system to raise the final alarm. Association mining based classification is used because it yields higher detection accuracy than previous data mining based detection systems which employed Naive Bayes, Support Vector Machine and Decision Tree techniques. A minimal subset of API categories is monitored while maintaining high detection accuracy. The number of generated rules is reduced, by removing the redundant rules, to make the malware analysis efficient. The key novelty of the proposed malware detection system is the iterative learning process combined with the run-time monitoring of program execution behavior which makes this as a dynamic malware detection system. The performance of the proposed malware detection system is evaluated for accuracy of malware detection system and compared with the existing data mining based detection systems. It is inferred that the proposed malware detection system outperforms the existing malware detection systems.

## General Terms

Pattern Recognition, Security, Algorithms et. al.

## Keywords

Malware detection, Windows API calls, Machine learning.

## 1. INTRODUCTION

Malware which is also termed as malicious software enters system without the permission of user of the system [1]. The words 'malicious' and 'software' are merged to create the term Malware. Malware is a very big threat in day-to-day computing world. The volume and complexity of malware is increasing and evolving. More and more organizations are trying to address the problem of malware. But, the source of malware which is the websites distributing the malware is increasing at a very high rate and is getting out of control. Most of the malware enters the computer while downloading files from the Internet. Once the malicious software enters into the system, it scans for vulnerabilities present in the operating system and perform unintended and unauthorized actions on the file system thereby slowing down the performance of the system.

The characteristics of malware are as follows [2]. Malware can either act as standalone malicious software or combine with other malicious software to act as a larger unit. Any number of new and additional modules can be added to the malware programs and can still achieve good performance. Malware is available over the internet in large volumes and can easily infect as many hosts as possible simultaneously. Malware is user-friendly for attackers by providing them the capability to introduce more complex attacks beyond the attacker's skill level. Malware can neither be easily detected nor removed from the system and it bypasses most of the security measures undertaken by the user. Malware can escape even complex forms of authentications. A wide range of devices are affected by malware. Malware has become a part of the cyber attack system. Malware can now-a-days earn lot of income to the attacker by performing various criminal activities.

A Malware detector can be mathematically visualized as a function with domain and range. The domain is the set of executable programs and the range is either malicious or benign [3]. The detector verifies a program to decide whether the program is benign program or malicious program. The malware detector identifies the malware programs using the signatures of malware. The pattern of the machine code of a malicious program is called as signature. Antivirus software compare their database of virus signatures with the signature of the files on the hard disk and other media as well as within the memory of the system. The signature-based malware detection is further divided into Dynamic, Static and Hybrid signature-based detection [4].

The paper is organized as follows. Section 2 discusses the related work. The proposed architecture and the proposed algorithms are discussed in Section 3 and 4 respectively. Section 5 discusses the experimental results. Section 6 concludes the paper.

## 2. RELATED WORK

Faraz Ahmed, Haider Hameed, Zubair Shafiq, Muddassar Farooq proposed a tool [5]. The tool extracts Windows API calls sequence, provides it as input to standard machine learning algorithms and classifies the process as malware or benign. Their experimental results show that, analyzing the Windows API call sequence improves the classifier's detection accuracy. A minimal subset of API categories was identified using scalability analysis. High detection accuracy is achieved by monitoring only the minimal API categories.

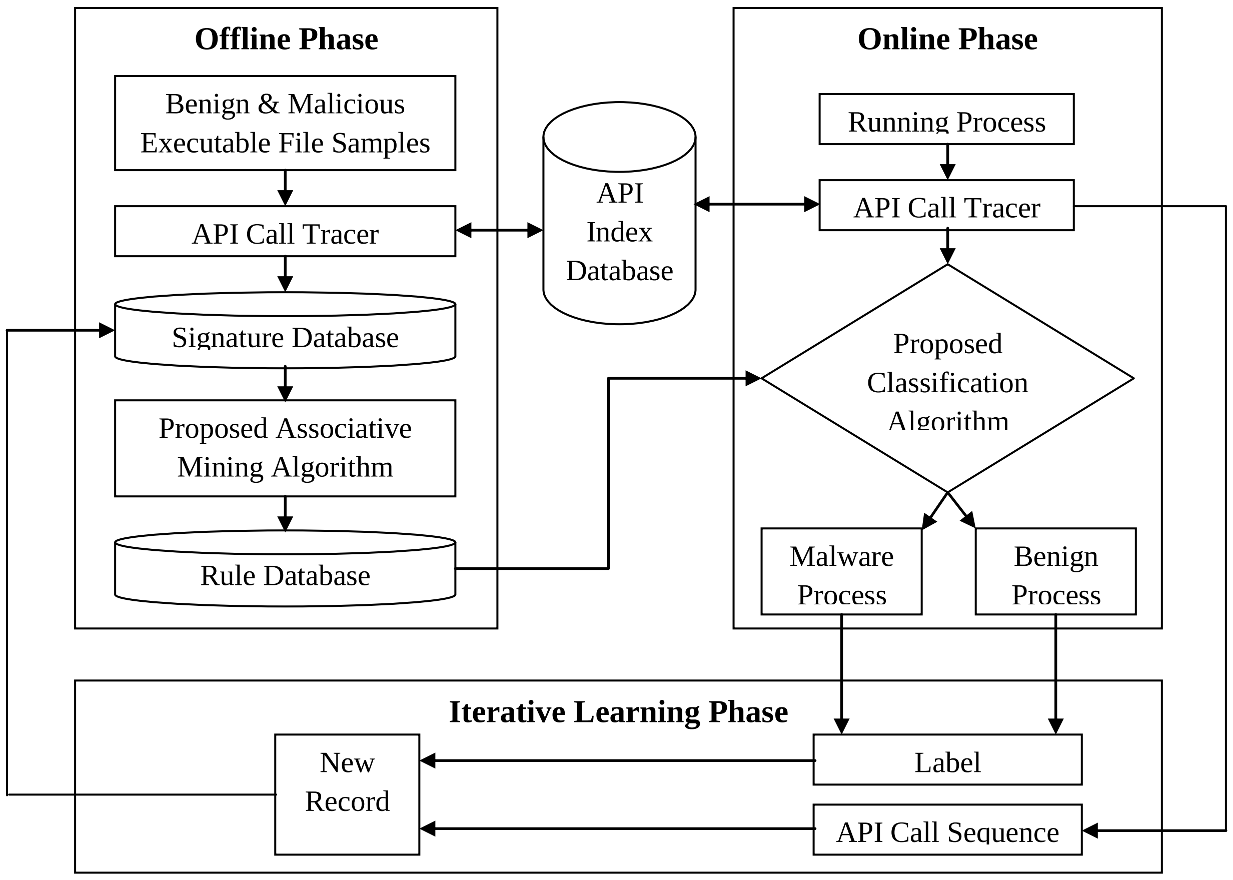

**Figure 1: Architecture of the proposed malware detection system**

Yi-Dong Shen, Zhong Zhang and Qiang Yang presented an approach [6], called Objective-Oriented utility-based Association (OOA) mining. The approach models association patterns that are related to a user's objective and utility. This approach focus on a user's objective and utility to measure the usefulness of association patterns, thus, OOA mining approach differs from the existing association mining approaches.

Yanfang Ye, Dingding Wang, Tao Li, Dongyi Ye and Qingshan Jiang developed the Intelligent Malware Detection System (IMDS) [7, 8] using Association mining based classification. They analyzed the Windows API call sequence called by executable files. A large collection of executable files obtained from KingSoft Corporation anti-virus laboratory was studied to compare various malware detection approaches. Their experimental results demonstrated that the accuracy and efficiency of IMDS system, using Association mining based classification, has higher performance than popular anti-virus software such as Norton Antivirus and McAfee Virus Scan, and previous data mining based detection systems employing Naive Bayes, Support Vector Machine (SVM) and Decision Tree techniques [9]. This approach has already been incorporated into the scanning tool of Anti-Virus software of KingSoft Corporation.

Yanfang Ye, Tao Li, Qingshan Jiang and Youyu Wang systematically evaluated the effects of the post processing techniques of associative classification in malware detection and proposed CIDCPF to detect the malware from the gray list. CIDCPF adapted the post processing techniques. The CIDCPF method has been integrated into IMDS system, and the new system is termed as CIMDS [10] system. Their experimental results demonstrates that the efficiency of detecting malware from the gray list of CIMDS system has higher performance than popular antivirus software tools, such as McAfee virus scan and Norton Antivirus, as well as previous data-mining-based detection systems.

In comparison, the key novelty of the proposed malware detection system is the iterative learning process combined with the run-time monitoring of program execution behavior which makes this as a dynamic malware detection system. Also, in the proposed malware detection system, Association mining based classification, based on the one proposed by Yi-Dong Shen et al., is used in the run-time monitoring of a program's API call sequence. A 3rd order Markov chain (i.e. 4-grams) is used to model the API call sequences. A minimal subset of API categories is monitored.

## 3. PROPOSED ARCHITECTURE

The architecture of proposed malware detection system is shown in Figure 1. The run-time monitoring of program execution behavior combined with an iterative learning phase makes the proposed malware detection system as a dynamic malware detection system. Association mining based classification is used in the run-time monitoring of a program's Windows API [11] call sequence. A 3rd order Markov chain (i.e. 4-grams) is used to model the API call sequences. Minimal set of API categories are monitored. The system consists of 3 phases: Offline, Online and Iterative learning phase.

The offline phase consists of the following components: dataset, API call tracer, API index database, signature database, rule generator and rule database. The dataset consists of the malware executable samples collected from various malware databases including VXHeavens [12] and benign executable samples collected from a freshly installed copy of Windows XP. The API call tracer traces the API call sequence of the process pertaining to the malware and benign executable samples of the dataset. Through the API index database, the API calls are represented using integer IDs. Thus the execution sequence generated by the API tracer is converted into integer sequence. The API call integer sequence and the corresponding label of all the samples in the dataset are stored in the signature database. The rule generator uses the proposed Association mining algorithm to generate classification rules which consists of the integer sequence (4-gram), support value, confidence value and the corresponding label. The rule which satisfy the support and confidence threshold is kept in the rule database. The support and confidence are defined as below.

*Support = [Count (I U {Class}, DB) / |DB|] x 100%*

*Confidence = [Count (I U {Class}, DB) / Count (I, DB)] x 100%*

Where,

I is the itemset

Count (I, DB) is the no. of records in the DB containing I (the itemset)

Count (I U {Class}, DB) is the no. of records in the DB in which (I U {Class}) holds

Class represents malware or benign

DB is the signature database

|DB| is the no. of records in DB

The online phase consists of the following components: the target process, API call tracer, API index database and the classifier. The target process is the process whose behavior has to be monitored at run time. The API call tracer traces the API call sequence of the target process. Through the API index database, the API calls are represented using integer IDs. Thus the execution sequence generated by the API tracer is converted into integer sequence. The classifier classifies the target process as malware or benign using the traced API call integer sequence and the rules generated in the offline phase.

In the iterative learning phase, the API call sequence and the classification label of the target process is iteratively added to the signature database after each classification to enhance the training model.

## 4. PROPOSED ALGORITHMS

The algorithms for tracing the API call sequence of a running process, mining rules and classification namely API Call Tracer algorithm, Rule Miner algorithm and Classifier algorithm respectively are given below.

API Call Tracer algorithm takes a running process and API index database as inputs and gives its called API function as an integer sequence as the output. This is done by hooking the API calls, monitoring the process and logging the API calls. The obtained sequence is converted into integer sequence by referring to the API index database and stored in signature database along with its corresponding label.

Rule Miner algorithm is used for mining frequent patterns and generating the rules. All the 4-grams of the integer API sequence of each training process is obtained and labeled with its corresponding class i.e. malware or benign. Each 4-gram with its label, support and confidence value represents a rule. Support and confidence of all the rules are calculated and rules satisfying support and confidence threshold is added in rule database.

The Classifier algorithm classifies the target process as malware or benign using rule database and API call sequence of target process. The above integer sequence is broken into 4-grams. Using the rule database, the confidence value of all 4-grams is substituted. The average confidence value of all 4-grams in malware class and benign class are calculated separately. Thus if the average confidence value of 4-grams in malware class is greater than that of benign class, the running process will be classified as malware else the running process will be classified as benign.

### 4.1 Call tracer algorithm

The algorithm for tracing API call sequence is as follows.

```
Input        : Process ‘Pr’, API index database
Output       : API call sequence ‘Seq’
Procedure    :
    Run ‘Pr’
    Hook the APIs and monitor ‘Pr’
    Using API index database, convert API call sequence of
    ‘Pr’ into integer sequence ‘seq’
    Return ‘seq’
```

### 4.2 Rule miner algorithm

The algorithm for mining rules is as follows.

```
Input        : API call sequence ‘Seq’
Output       : Classification rules
Procedure    :
    For all ‘seq’ in signature database
    Split ‘seq’ into 4-grams & label them with their
    corresponding class each forming a rule
    For all rules calculate support and confidence
    Add the rules satisfying support and confidence threshold
    in rule database
```

### 4.3 Classifier algorithm

The algorithm for classifying the process is as follows.

```
Input        : Rule database and API call sequence
(integer sequence) ‘seq’ of target process
Output       : ‘Pr’ is malware or benign
Procedure    :
    Break ‘seq’ into 4-grams and substitute confidence value
    of all 4-grams from rule database
    Calculate average confidence value of all 4-grams with
    label as malware and label as benign separately.
    IF the average confidence value of 4-grams in malware
    class is greater than that of benign class THEN the
    running process will be classified as malware, ELSE the
    running process will be classified as benign.
```

## 5. EXPERIMENTAL RESULTS

The proposed malware detection system is evaluated using accuracy of the malware detection system (ACY). Accuracy is defined as the ratio of sum of number of malicious process rightly classified as malicious and number of benign process rightly classified as benign to total number of process classified.

$$ACY = [(X + Y)/Z] * 100\ \%$$

Where,

X is the number of malicious process rightly classified as malicious

Y is the number of benign process rightly classified as benign

Z is the total number of process classified

The experiments are conducted using Windows XP operating system, Intel Pentium D 2.80GHz CPU and 1Gb of RAM. Windows executable files which are recognized as benign and malicious executables are collected. The malicious executables mainly consists of backdoors, worms, and Trojan horses collected from VXHeavens [12]. The benign executables are gathered from freshly installed copy of Windows XP operating system. The proposed algorithms are implemented using Visual C++. The API Call Tracer algorithm is implemented using the IAT hooking technique [13, 14, 15]. The dataset consists of 94 benign and 179 malware executables of which 68 benign and 95 malware executables were used to train the malware detection system and 26 benign and 84 malware executables were used to test the malware detection system. ACY of existing malware detection systems, which were calculated and tabulated in [10], are used for performance comparison against ACY of the proposed system. ACY of the proposed system and existing systems are shown in Table 1. Each value of ACY of the proposed system, shown in Table 1, is the average value obtained by 10 runs of each experiment with support value greater than 40% and confidence value greater than 95%.

**Table 1. Accuracy of different malware detection systems**

| **Malware detection systems** | **Accuracy (%)** | |
|---|---|---|
| | **Training set** | **Testing set** |
| Naïve Bayes | 50.9186 | 48.6926 |
| SVM | 68.2373 | 64.4357 |
| J4.8 | 55.7260 | 56.7716 |
| IMDS | 67.1230 | 64.1319 |
| CIMDS | 71.3484 | 67.5049 |
| Proposed system | 99.3800 | 90.0000 |

CIMDS, an existing malware detection system, outperforms the other existing systems. The ACY of the proposed malware detection system is 39% more than that of CIMDS using training set and 34% more than that of CIMDS using testing set.

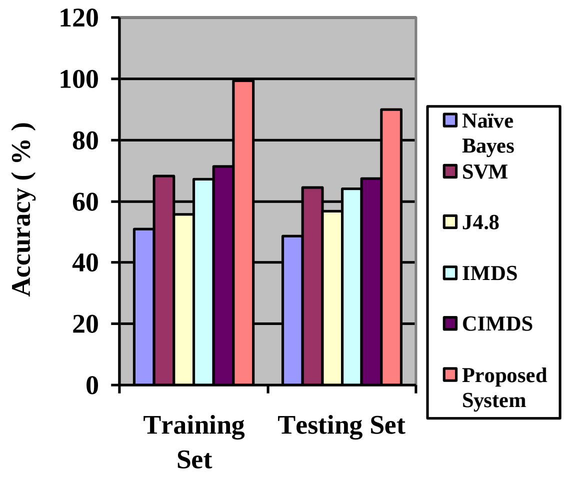

**Figure 2: ACY of different malware detection systems**

Figure 2, showing the ACY of different malware detection systems, illustrates that the proposed malware detection system outperforms various existing malware detection systems.

## 6. CONCLUSION

A malware detection system is proposed which uses Windows API call sequence. A 3rd order Markov chain (i.e. 4-grams) is used to model the API call. This composite feature set is provided as an input to the malware detection system to raise the final alarm. Association mining based classification is used because it yields higher detection accuracy than previous data mining based detection systems which employed Naive Bayes, Support Vector Machine (SVM) and Decision Tree techniques. A minimal subset of API categories is monitored whilst maintaining high detection accuracy. The number of generated rules is reduced, by removing the redundant rules, to make the malware analysis efficient. The iterative learning process combined with the run-time monitoring of program execution behavior makes this as a dynamic malware detection system. The performance of the proposed malware detection system is evaluated for accuracy of the malware detection system and compared with the existing data mining based detection systems. It is inferred that the proposed malware detection system outperforms the existing malware detection systems.

## 7. REFERENCES

[1] Rizwan Rehman, G. C. Hazarika and Gunadeep Chetia, “Malware Threats And Mitigation Strategies: A Survey”, Journal of Theoretical and Applied Information Technology, Vol. 29, No. 2, pp. 69-73, July 2011.

[2] OECD Ministerial Meeting Report, “Malicious Software (Malware): A Security Threat to the Internet Economy”, Korean Communication Commision, Final draft, May 2007.

[3] Vinod, P. Laxmi, V. and M. S. Gaur. 2009. Survey on Malware Detection Methods. In Proceedings of the Hacker 2009, pp. 74-79.

[4] Nwokedi Idika and Aditya P. Mathur. 2007. A Survey of Malware Detection Techniques. SERC Library.

[5] Faraz Ahmed, Haider Hameed, Zubair Shafiq M. and Muddassar Farooq. 2009. Using Spatio-Temporal Information in API Calls with Machine Learning Algorithms for Malware Detection. In Proceedings of the 2nd ACM Workshop on Artificial Intelligence and Security (AISec 2009), pp. 55-62.

[6] Yi-Dong Shen, Zhong Zhang and Qiang Yang. 2002. Objective-oriented utility-based association mining. In Proceedings of the IEEE International Conference on Data Mining (ICDM 2003), pp. 426-433.

[7] Yanfang Ye, Dingding Wang, Tao Li and Dongyi Ye. 2007. IMDS: Intelligent malware detection system. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07), pp. 1043–1047.

[8] Yanfang Ye, Dingding Wang, Tao Li, Dongyi Ye and Qingshan Jiang, "An intelligent pe-malware detection system based on association mining," Journal in Computer Virology, vol. 4, pp. 323–334, Feb. 2008.

[9] Jiawei Han and Micheline Kamber. 2006. Data mining: Concepts and Techniques, Morgan Kaufmann publishers: San Francisco, 2nd edition.

[10] Yanfang Ye, Tao Li, Qingshan Jiang and Youyu Wang. 2010. CIMDS: Adapting Postprocessing Techniques of Associative Classification for Malware Detection. In Proceedings of the IEEE Transactions On Systems, Man, And Cybernetics - Part C: Applications And Reviews, vol. 40, No. 3, pp. 298-307.

[11] Overview of the Windows API. Available at: http://msdn.microsoft.com/en-us/library/aa383723(VS.85).aspx.

[12] VX Heavens Virus Collection. Available at: http://vx.netlux.org/.

[13] IAT-Hooking-Revisited. Available at: http://www.autosectools.com/IAT-Hooking-Revisited.pdf.

[14] Understanding the Import Address Table. Available at: http://sandsprite.com/CodeStuff/Understanding_imports.html.

[15] IAT Function Hooking. Available at: http://sandsprite.com/CodeStuff/IAT_Hooking.html