

IMDS: Intelligent Malware Detection System

Yanfang Ye*
 Department of Computer
 Science
 Xiamen University
 Xiamen, 361005, P.R.China
 yeyanfang@yahoo.com.cn

Dingding Wang, Tao Li
 School of Computer Science
 Florida International University
 Miami, FL 33199, USA
 dwang003,taoli@cs.fiu.edu

Dongyi Ye
 College of Math and Computer
 Science
 Fuzhou University
 Fuzhou, 350002, P.R.China
 yiedy@fzu.edu.cn

ABSTRACT

The proliferation of malware has presented a serious threat to the security of computer systems. Traditional signature-based anti-virus systems fail to detect polymorphic and new, previously unseen malicious executables. In this paper, resting on the analysis of Windows API execution sequences called by PE files, we develop the Intelligent Malware Detection System (IMDS) using Objective-Oriented Association (OOA) mining based classification. IMDS is an integrated system consisting of three major modules: PE parser, OOA rule generator, and rule based classifier. An OOA_Fast_FP-Growth algorithm is adapted to efficiently generate OOA rules for classification. A comprehensive experimental study on a large collection of PE files obtained from the anti-virus laboratory of KingSoft Corporation is performed to compare various malware detection approaches. Promising experimental results demonstrate that the accuracy and efficiency of our IMDS system outperform popular anti-virus software such as Norton AntiVirus and McAfee VirusScan, as well as previous data mining based detection systems which employed Naive Bayes, Support Vector Machine (SVM) and Decision Tree techniques.

Categories and Subject Descriptors

I.2 [Artificial Intelligence]: Learning

General Terms

Algorithms, Experimentation, Security

Keywords

Malware, PE file, Windows API sequence, OOA mining

1. INTRODUCTION

Malicious executables are programs designed to infiltrate or damage a computer system without the owner's consent, which have become a serious threat to the security of computer systems. New,

*The work was done while the author was working in the Anti-virus laboratory at KingSoft Corporation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'07, August 12-15, 2007, San Jose, California, USA.
 Copyright 2007 ACM 978-1-59593-609-7/07/0008 ...\$5.00.

previously unseen viruses and polymorphic viruses which adopt obfuscation techniques are more complex and difficult to detect. Currently, most widely-used malware detection software uses signature-based and heuristic-based algorithms to recognize threats. Signatures are short strings of bytes which are unique to the programs. They can be used to identify particular viruses in executable files, boot records, or memory with small error rate [8]. However, signature based method is expensive and slow. In addition, they are not effectively against modified and unknown malicious executables. Heuristic-based recognition tends to provide protection against new and unknown threats. However, this kind of virus detection is usually inefficient and inaccurate.

Our efforts for detecting polymorphic and new, previously unseen malicious executables lead to the Intelligent Malware Detection System (IMDS), which applies Objective-Oriented Association (OOA) mining based classification [10, 14]. The IMDS rests on the analysis of Windows Application Programming Interface (API) execution sequences which reflect the behavior of program code pieces. The associations among the APIs capture the underlying semantics for the data are essential to malware detection. Our malware detection is carried out directly on Windows Portable Executable (PE) code with three major steps: (1) first constructing the API execution sequences by developing a PE parser; (2) followed by extracting OOA rules using OOA_Fast_FP-Growth algorithm; (3) and finally conducting classification based on the association rules generated in the second step. So far, we have gathered 29580 executables, of which 12214 are referred to as benign executables and 17366 are malicious ones. These executables called 12964 APIs in total. The collected data in our work is significantly larger than those used in previous studies on data mining for malware detection [13, 9, 18]. The experimental results illustrate that the accuracy and efficiency of our IMDS outperform popular anti-virus software such as Norton AntiVirus and McAfee VirusScan, as well as previous systems using data mining approaches such as Naive Bayes, SVM and Decision Tree. Our approach for malware detection has been incorporated into the scanning tool of KingSoft's AntiVirus software.

2. RELATED WORK

Besides the traditional signature-based malware detection methods, there is some work to improve the signature-based detection [15, 3, 12] and also a few attempts to apply data mining and machine learning techniques to detect new malicious executables.

Sung et al. [15] developed a signature based malware detection system called SAVE (Static Analyzer of Vicious Executables) which emphasized on detecting polymorphic malware. The basic idea of this approach is to extract the signatures from the original malware with the hypothesis that all versions of the same malware

share a common core signature. Schultz et al. [13] applied Naive Bayes method to detect previously unknown malicious code. Decision Tree was studied in [18, 9]. Kolter et al. [9] gathered 1971 benign executables and 1651 malicious executables in Windows PE format, and examined the performance of different classifiers such as Naive Bayes, support vector machine (SVM) and Decision Tree using 10-fold cross validation and plotting ROC curves [16]. Their results also showed that the ROC curve of the Decision Tree method dominated all others.

Different from earlier studies, our work is based on a large collection of malicious executables collected at KingSoft Anti-Virus Laboratory. In addition, we apply OOA mining technique to extract the characterizing frequent patterns to achieve accurate malware detection since frequent patterns found by association mining carry the underlying semantics of the data.

3. THE SYSTEM ARCHITECTURE

Our IMDS system is performed directly on Windows PE code. PE is designed as a common file format for all flavors of Windows operating system, and PE viruses are in the majority of the viruses rising in recent years. Some famous viruses such as CIH, CodeRed, CodeBlue, Nimda, Sircam, Killonice, Sobig, and LoveGate all aim at PE files. The system consists of three major components: PE parser, OOA rule generator, and malware detection module, as illustrated in Figure 1.

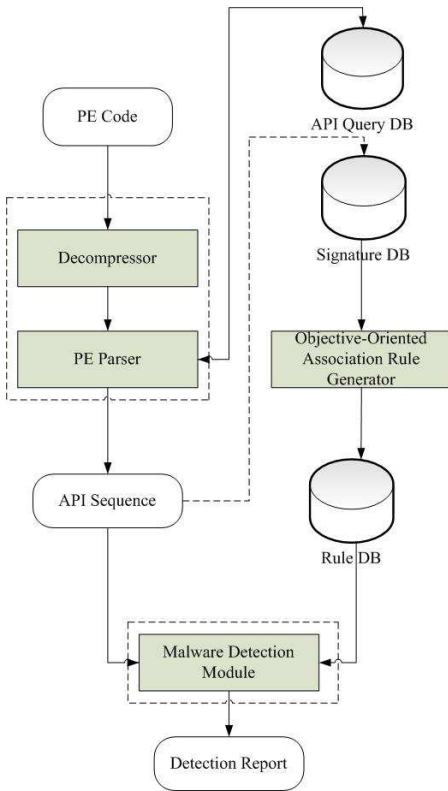


Figure 1: IMDS system architecture

The functionality of the PE parser is to generate the Windows API execution sequence for each benign/malicious executable. Since a virus scanner is usually a speed sensitive application, in order to improve the system performance, we developed a PE parser to construct the API execution sequences of PE files instead of using a third party disassembler. If a PE file is previously compressed by a

third party binary compress toll, it needs to be decompressed before being passed to the PE parser. Through the API query database, the API execution sequence generated by the PE parser can be converted to a group of 32-bit global IDs which represents the static execution sequence of the corresponding API functions. Then we use the API calls as the signatures of the PE files and store them in the signature database, which contains 6 fields: record ID, PE file name, file type (“0” represents benign file while “1” is for malicious file), called API sequence name, called API ID, and the total number of called API functions, as shown in Figure 2. After that, an OOA mining algorithm is applied to generate class association rules which are recorded in the rule database. To finally determine whether a PE file is malicious or not, we pass the selected API calls together with the rules generated to the malware detection module to perform the association rule based classification.

Figure 2: Samples in the signature database

4. CLASSIFICATION BASED ON OOA MINING

Both classification and association mining play important roles in data mining techniques. Classification is “the task of learning a target function that maps each feature set to one of the predefined class labels” [17]. For association rule mining, there is no pre-determined target. Given a set of transactions in the database, all the rules that satisfied the support and confidence thresholds will be discovered [1]. As a matter of fact, classification and association rule mining can be integrated to association rule based classification [10, 2]. This technique utilizes the properties of frequent patterns to solve the scalability and overfitting issues in classification and achieves excellent accuracy [2]. In our IMDS system, we adapted OOA mining techniques [14] to generate the rules.

4.1 OOA Definitions

In our IMDS system, the goal is to find out which set of API calls supports the specific objectives: $Obj_1 = (Group = Malicious)$, and $Obj_2 = (Group = Benign)$.

- **Definition 1 (Support and confidence)** Let $I = \{I_1, \dots, I_m\}$ be an itemset and $I \rightarrow Obj(os\%, oc\%)$ be an association rule in OOA mining. The support and confidence of the rule are defined as:

$$os\% = supp(I, Obj) = \frac{count(I \cup \{Obj\}, DB)}{|DB|} \times 100\%$$

$$oc\% = conf(I, Obj) = \frac{count(I \cup \{Obj\}, DB)}{count(I, DB)} \times 100\%$$

where the function $count(I \cup \{Obj\}, DB)$ returns the number of records in the dataset DB where $I \cup \{Obj\}$ holds.

- **Definition 2 (OOA frequent itemset)** Given $mos\%$ as a user-specified minimum support. I is an OOA frequent itemset/pattern in DB if $os\% \geq mos\%$.
- **Definition 3 (OOA rule)** Given $moc\%$ as a user-specified confidence. Let $I = \{I_1, \dots, I_m\}$ be an OOA frequent itemset. $I \rightarrow Obj(os\%, oc\%)$ is an OOA rule if $oc\% \geq moc\%$.

4.2 OOA_Fast_FP-Growth Algorithm

Although Apriori algorithm can be extended to OOA mining, it requires many iterations to generate all of the frequent itemsets before generating the association rules. An alternative OOA mining algorithm called OOA_FP-Growth is designed based on FP-Growth algorithm [6, 5]. In general, OOA_FP-Growth algorithm is much faster than OOA_Apriori for mining frequent itemsets. However, when the minimum support is small, OOA_FP-Growth generates a huge number of conditional FP-trees recursively, which is time and space consuming. Our malware detection relies on finding frequent patterns from large collections of data, therefore, the efficiency is an essential issue to our system. In our IMDS system, we extend a modified FP-Growth algorithm proposed in [4] to conduct the OOA mining. This algorithm greatly reduces the costs of processing time and memory space, and we call it OOA_Fast_FP-Growth algorithm.

Similar to OOA_FP-Growth algorithm, there are also two steps in OOA_Fast_FP-Growth algorithm: constructing an OOA_Fast_FP-tree and generating frequent patterns from the tree. But the structure of an OOA_Fast_FP-tree is different from that of an OOA_FP-tree in the following way: (1) The paths of an OOA_Fast_FP-tree are directed, and there is no path from the root to leaves. Thus, fewer pointers are needed and less memory space is required. (2) In an OOA_FP-tree, each node is the name of an item, but in an OOA_Fast_FP-tree, each node is the sequence number of an item, which is determined by the support count of the item. The detailed description can be referenced in [4].

4.3 An Illustrating Example

At the beginning of this section, we state that frequent patterns are essential to accurate classification. To demonstrate the effectiveness of the frequent patterns, we show an example rule generated by OOA_Fast_FP-Growth algorithm. We sample 5611 records from our signature database, of which 3394 records are malicious executables and 2217 records are benign executables. One of the rules we generated is:

$(2230, 398, 145, 138, 115, 77) \rightarrow Obj_1 = (Group = Malicious)$
 $(os = 0.296739, oc = 0.993437)$,

where os and oc represent the support and confidence, respectively. After converting the API IDs to API names via our API query database, this rule becomes:

$(KERNEL32.DLL, OpenProcess; CopyFileA; CloseHandle; GetVersionExA; GetModuleFileNameA; WriteFile;) \rightarrow$
 $Obj_1 = (Group = Malicious) (os = 0.296739, oc = 0.993437)$.

After analyzing the API sequence in this rule, we know that the program actually executes the following functions: (i) returns a handle to an existing process object; (ii) copies an existing file to a new file; (iii) closes the handle of the open object; (iv) obtains extended information about the version of the currently running operating system; (v) retrieves the complete path of the file that contains the specified module of current process; and (vi) writes data to the file. with the os and oc values, we know that this sequence of API appears in 1665 malware, while only in 11 benign files. Obviously, it is one of the essential rules for determining whether an executable

is malicious or not. In the experiments section, we perform a comprehensive experimental study to evaluate the efficiency of different OOA mining algorithms.

4.4 Associative Classifier

For OOA rule generation, we use the OOA_Fast_FP-Growth algorithm to obtain all the association rules with certain support and confidence thresholds, and two objectives: $Obj_1 = (Group = Malicious)$, $Obj_2 = (Group = Benign)$. Then we apply the technique of classification based on association rules (CBA) to build a CBA classifier [10] as our malware detection module. The CBA classifier is built on rules with high support and confidence and uses the association between frequent patterns and the type of files for prediction. So, our malware detection module takes the input of generated OOA rules and outputs the prediction of whether an executable is malicious or not.

5. DATA COLLECTION

As stated previously, we obtained 29580 Windows PE files of which 12214 were recognized as benign executables while 17366 were malicious executables. The malicious executables mainly consisted of backdoors, worms, and Trojan horses and all of them were provided by the Anti-virus laboratory of KingSoft Corporation. The benign executables were gathered from the system files of Windows 2000/NT and XP operating system.

6. EXPERIMENTAL RESULTS AND ANALYSIS

We conduct three sets of experiments using our collected data. In the first set of experiments, we evaluate the efficiency of different OOA mining algorithms. The second set of experiments is to compare the abilities to detect polymorphic and unknown malware of our IMDS system with current widely-used anti-virus software. The efficiency and false positives by using different scanners have also been examined. Finally, we compare our IMDS system with other classification based methods. All the experiments are conducted under the environment of Windows 2000 operating system plus Intel P4 1Ghz CPU and 1Gb of RAM.

6.1 Evaluation of Different OOA Mining Algorithms

In the first set of experiments, we implement OOA_Apriori, OOA_FP-Growth, and OOA_Fast_FP-Growth algorithms under Microsoft Visual C++ environment. We sample 5611 records from our signature database, which includes 3394 records of malicious executables and 2217 records of benign executables. By using different support and confidence thresholds, we compare the efficiency of the three algorithms. The results are shown in Table 3 and Figure 3. From Table 3, we observe that the time complexity increases exponentially as the minimum support threshold decreases. However, it shows obviously that the OOA_Fast_FP-Growth algorithm is much more efficient than the other two algorithms, and it even doubles the speed of performing OOA_FP-Growth algorithm. Figure 3 presents a clearer graphical view of the results.

6.2 Comparisons of Different Anti-virus Scanners

In this section, we examine the abilities of detecting polymorphic malware and unknown malware of our system in comparison with some of the popular software tools such as Norton AntiVirus 2006, Dr.Web, McAfee VirusScan and Kaspersky Anti-Virus. The efficiency and the number of false positives are also evaluated.

Experiment	1	2	3	4
mos	0.355	0.35	0.34	0.294
moc	0.9	0.95	0.9	0.98
OOA_Apriori	25	260.5	∞	∞
OOA_FP-Growth	8	16.14	60.5	280.2
OOA_Fast_FP-Growth	4.1	7.99	28.8	143.5

Table 1: Running time of different OOA mining algorithms (min). mos and moc represent the minimum support and minimum confidence for each experiment.

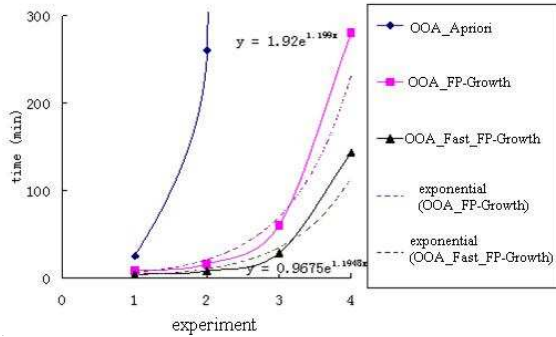


Figure 3: Comparison on the efficiency of different OOA mining algorithms

6.2.1 Polymorphic Virus Detection

In this experiment, we sample 3000 malicious files and 2000 benign files in the training data set, then we use 1500 malware and 500 benign executables as the test data set. Several recent Win32 PE viruses are included in the test data set for analysis such as Lovedoor, My doom, Blaster, and Beagle. For each virus, we apply the obfuscation techniques described in [15] to create a set of polymorphic versions. Then we compare our system with current most widely-used anti-virus software. The results shown in Table 4 demonstrate that our IMDS system achieves better accuracy than other software in polymorphic malware detection.

6.2.2 Unknown Malware Detection

In order to examine the ability of identifying new and previously unknown malware of our IMDS system, in the test data set, we use 1000 malware analyzed by the experts in KingSoft Anti-virus laboratory, while the signatures of the malware have not been recorded into the virus signature database. Comparing with other anti-virus software, our IMDS system performs most accurate detection. The results are listed in Table 5.

6.2.3 System Efficiency and False Positives

In malware detection, a false positive occurs when the scanner marks a benign file as a malicious one by error. False positives can be costly nuisances due to the waste of time and resources to deal with those falsely reported files. In this set of experiments, in order to examine the system efficiency and the number of false positives of the IMDS system, we sample 2000 executables in the test data set, which contains 500 malicious executables and 1500 benign ones.

First, we compare the efficiency of our system with different scanners including the scanner named “SAVE” [15, 20] described in related work and some widely-used anti-virus software.

The results in Figure 4 illustrate that our IMDS system achieves much higher efficiency than other scanners when being executed

Software	N	M	D	K	IMDS
Beagle	✓	✓	✓	✓	✓
Beagle V1	✓	✓	×	✓	✓
Beagle V2	✓	×	×	✓	✓
Beagle V3	×	×	×	✓	✓
Beagle V4	✓	✓	×	×	✓
Blaster	✓	✓	✓	✓	✓
Blaster V1	✓	✓	✓	✓	✓
Blaster V2	×	×	×	×	✓
Lovedoor	✓	✓	✓	✓	✓
Lovedoor V1	×	×	✓	×	✓
Lovedoor V2	×	×	×	×	✓
Lovedoor V3	×	✓	×	✓	✓
Mydoom	✓	✓	✓	✓	✓
Mydoom V1	×	×	×	×	✓
Mydoom V1	×	×	×	?	✓

Table 2: Polymorphic malware detection. Remark: N - Norton AntiVirus, M - MacAfee, D - Dr.Web, K - Kaspersky. In the table, “✓” indicates successful detection, “×” indicates failure to detect, and “?” represents only an “alert”; all the scanners used are of most current and updated version.

Software	N	M	D	K	IMDS
malware1	✓	✓	✓	×	✓
malware2	×	×	✓	✓	✓
malware3	✓	×	×	×	✓
malware4	×	×	×	×	×
malware5	×	✓	×	×	✓
malware6	×	×	✓	×	✓
malware7	✓	×	×	×	✓
malware8	×	×	×	×	✓
malware9	×	✓	✓	×	✓
malware10	×	×	×	×	✓
malware11	×	×	×	✓	✓
malware12	×	✓	×	✓	✓
⋮	⋮	⋮	⋮	⋮	⋮
malware1000	✓	×	×	×	✓
Stat.	633	753	620	780	920
Ratio.	63.3%	75.3%	62%	78%	92%

Table 3: Unknown malware detection

in the same environment. The number of false positives by using different scanners are also examined. By scanning 1500 benign executables whose signatures have not been recorded in the signature database, we obtain the results shown in Figure 5. Figure 5 clearly shows that the false positives by using our IMDS system are much fewer than other scanners.

6.3 Comparisons of Different Classification Methods

In this set of experiments, we compare our system with Naive Bayes, Support Vector Machine (SVM) and Decision Tree methods. We randomly select 2843 executables from our data collection, in which 1207 files are benign and 1636 executables are malicious. Then we convert the transactional sample data in our signature database into a relational table, in which each column corresponds to an API and each row is an executable. This transformation makes it easy to apply feature selection methods and other classification approaches.

First, we rank each API using Max-Relevance algorithm [11],

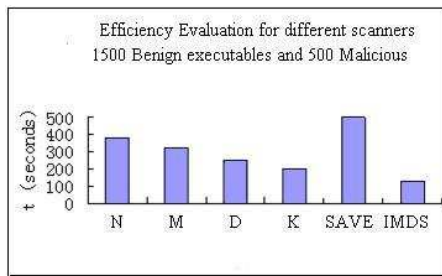


Figure 4: Efficiency of different scanners

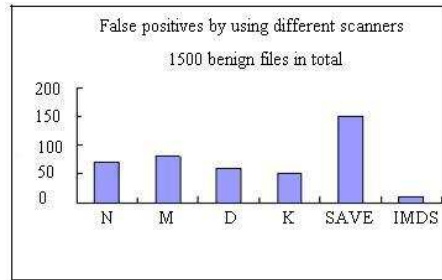


Figure 5: False Positives by using different scanners

and then choose top 500 API calls as the features for later classification. In the experiments, we use the Naive Bayes classifier and J4.8 version of Decision Tree implemented in WEKA [19], and also the SVM implemented in LIBSVM package [7]. For the OOA_Fast_FP-Growth mining, we select thresholds based on two criteria: setting *moc* as close to 1 as possible; and selecting a big *mos* without exceeding the maximum support in the data set. Then, in the experiment, we set *mos* to 0.294 and *moc* to 0.98. Ten-fold cross validation is used to evaluate the accuracy of each classifier. Results shown in Table 6 indicate our IMDS system achieve most accurate malware detection.

Algorithms	TP	TN	FP	FN	DR	ACY
Naive Bayes	1340	1044	163	296	81.91%	83.86%
SVM	1585	989	218	51	96.88%	90.54%
J4.8	1574	1027	609	62	96.21%	91.49%
IMDS	1590	1056	151	46	97.19%	93.07%

Table 4: Results by using different classifiers. TP, TN, FP, FN, DR, and ACY refer to True Positive, True Negative, False Positive, False Negative, Detection Rate, and Accuracy, respectively.

From the comparison, we observe that our IMDS outperforms other classification methods in both detection rate and accuracy. This is because “frequent patterns are of statistical significance and a classifier with frequent pattern analysis is generally effective to test datasets” [2].

7. CONCLUSIONS

In this paper, we describe our research effort on malware detection based on window API sequence. In summary, our main contributions are: (1) We develop an integrated IMDS system based on analysis of Windows API execution sequences. The system consists of three components: PE parser, rule generator and classifier; (2) We adapt existing association based classification techniques

to improve the system effectiveness and efficiency; (3) We evaluate our system on a large collection of executables including 12214 benign samples and 17366 malicious ones; (4) We provide a comprehensive experimental study on various anti-virus software as well as various data mining techniques for malware detection using our data collection; (5) Our system has been already incorporated into the scanning tool of KingSoft’s AntiVirus software.

Acknowledgments

The work of Tao Li is partially supported by NSF IIS-0546280. The authors would also like to thank the members in the Anti-virus laboratory at KingSoft Corporation for their helpful discussions and suggestions.

8. REFERENCES

- [1] R. Agrawal and R. Srikant. Fast algorithms for association rule mining. In *Proceedings of VLDB-94*, 1994.
- [2] H. Cheng, X. Yan, J. Han, and C. Hsu. Discriminative frequent pattern analysis for effective classification. In *ICDE-07*, 2007.
- [3] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *Proceedings of the 12th USENIX Security Symposium*, 2003.
- [4] M. Fan and C. Li. Mining frequent patterns in an fp-tree without conditional fp-tree generation. *Journal of Computer Research and Development*, 40:1216–1222, 2003.
- [5] J. Han and M. Kamber. *Data mining: Concepts and techniques, 2nd edition*. Morgan Kaufmann, 2006.
- [6] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of SIGMOD*, pages 1–12, May 2000.
- [7] C. Hsu and C. Lin. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Networks*, 13:415–425, 2002.
- [8] J. Kephart and W. Arnold. Automatic extraction of computer virus signatures. In *Proceedings of 4th Virus Bulletin International Conference*, pages 178–184, 1994.
- [9] J. Kolter and M. Maloof. Learning to detect malicious executables in the wild. In *Proceedings of KDD’04*, 2004.
- [10] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of KDD’98*, 1998.
- [11] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27, 2005.
- [12] J. Rabek, R. Khazan, S. Lewandowski, and R. Cunningham. Detection of injected, dynamically generated, and obfuscated malicious code. In *Proceedings of the 2003 ACM workshop on Rapid malware*, pages 76–82, 2003.
- [13] M. Schultz, E. Eskin, and E. Zadok. Data mining methods for detection of new malicious executables. In *Proceedings of IEEE International Conference on Data Mining*, 2001.
- [14] Y. Shen, Q. Yang, and Z. Zhang. Objective-oriented utility-based association mining. In *Proceedings of IEEE International Conference on Data Mining*, 2002.
- [15] A. Sung, J. Xu, P. Chavez, and S. Mukkamala. Static analyzer of vicious executables (save). In *Proceedings of the 20th Annual Computer Security Applications Conference*, 2004.
- [16] J. Swets and R. Pickett. *Evaluation of diagnostic system: Methods from signal detection theory*. Academic Press, 1982.
- [17] P. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Addison Wesley, 2005.
- [18] J. Wang, P. Deng, Y. Fan, L. Jaw, and Y. Liu. Virus detection using data mining techniques. In *Proceedings of IEEE International Conference on Data Mining*, 2003.
- [19] H. Witten and E. Frank. *Data mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2005.
- [20] J. Xu, A. Sung, P. Chavez, and S. Mukkamala. Polymorphic malicious executable sanner by api sequence analysis. In *Proceedings of the International Conference on Hybrid Intelligent Systems*, 2004.