International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.2, Mar-Apr 2012 pp-547-551 ISSN: 2249-6645

Object Oriented Association (OOA) Mining Based Classification in Storage Cloud

V.Naresh Kumar[1] Ramesh nallamalli[2] Gorantla praveen[3]
I.Pavan Kumar[4] K Subrahamanyam[5]
M.Tech[CSE] M.Tech[CSE] M.Tech[CSE] M.Tech[CSE] Professor
[1,2,3,4,5]Dept of computer Science and Engineering, KL University, Vaddeswaram, Guntur (District), AP.

Abstract: Cloud Storage provides whatever amount of storage you require, on an immediate basis. It is persistent. It can be accessed in a variety of ways, both in the data center where the cloud is housed, as well as via the Internet. If you obtain this from an external provider, it is purchased on a pay as you go basis. You do not manage it, you use it, and the service provider manages it." Cloud systems should be geographically dispersed to reduce their vulnerability due to earthquakes and other catastrophes, which increase technical challenge on a great level of distributed data interpretability and mobility. Data interoperability is even more essential in the future as one component of a multi-faceted approach to many applications; many open challenges still remain such as cloud data security and the efficiency of query processing in the cloud. The proliferation of malware in recent years has presented a serious threat to the security of computer systems. Polymorphic computer viruses, which adopt obfuscation technique, are more complex and difficult than their original versions to detect, as well as new, previously unseen viruses, often making antivirus companies ineffective when using the classic signature-based virus detection technique. In this paper, we rest on the analysis of Win API calling sequences of PE files and propose a new approach for detecting polymorphic or even unknown malware in the Windows platform based on data mining technique, namely OOA Mining algorithm. Our approach rests on an analysis based on the Win API calling sequence that reflects the behavior of a piece of particular code. 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 King- Soft Corporation is performed to compare various malware detection approaches.

Keywords: OOA Mining, Windows API Sequence, PE File, Malware

I. Introduction

The proliferation of malware in recent years has presented a serious threat to the security of computer systems. Polymorphic computer viruses, which adopt obfuscation technique, are more complex and difficult than their original versions to detect, as well as new, previously unseen viruses, often making antivirus companies ineffective when using the classic signature-based virus detection technique. In this paper, we rest on the analysis of Win API calling sequences of PE files and propose a new approach for detecting polymorphic or even unknown malware in the Windows platform based on data mining technique, namely OOA Mining algorithm[21]. Our approach rests on an analysis based on the Win API calling sequence that reflects the behavior of a piece of particular code. The analysis is carried out directly on the PE code. It is achieved in three major steps: construct the API calling sequences for both training set and testing set; and then extract the OOA rules from the training set using OOA Mining algorithm; at last, detect the testing set according to the OOA rules created by the OOA rules generator. We implement a malware detection system, DMAV system, to evaluate the effectiveness of our proposed approach, mainly three major modules included: 1. PE Parser: considering that virus scanner is a speed sensitive application, and, in order to improve the system performance, we develop a PE parser to construct the API calling sequences of a PE file instead of using a third party disassembler. In advance, in order to make convenience for our DMAV system's further analysis, we also implement three other functions: function calls' extraction from a PE file's export table, the extraction of a PE file's section information and the disassembler of a PE file. 2. OOA Rules Generator: after extracting the Win API sequences of the training set by our PE parse, we store these Win API sequences into the database as signatures, then pass them through the OOA Rules generator to mine the rules satisfying the specific objective, and thus store such rules into the OOA Rules database. In addition, we implement three distinguishing algorithms for our OOA mining, and they are as follows: OOA_Apriori, OOA_FPgrowth, OOA_DMAV_FPgrowth. 3. Malware Detection Module: in order to determine whether a PE file in the testing set is a malware or not, we pass this PE file's API sequence constructed by our PE parser, together with each OOA Rule in OOA Rules database created by our OOA rule generator, through a malware detection module. In this paper, we make contribution to optimize the signature creation using OOA mining algorithm, and, the experiment results illustrate that, compared to the other two OOA mining algorithms, our OOA_DMAV_FPgrowth algorithm performs the highest

efficiency. Thus, encouraging experimental results demonstrate the robustness and intelligence of our DMAV system: compared with several popular anti-virus software's, our DMAV system can detect not only known viruses, but also polymorphic and new previously unseen malware effectively and efficiently.

II. 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

III. 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, Killonce, Sobig, and LoveGate all aim at PE les. The system consists of three major components: PE parser, OOA rule generator, and malware detection module, as illustrated in Figure 1.

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 les 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.

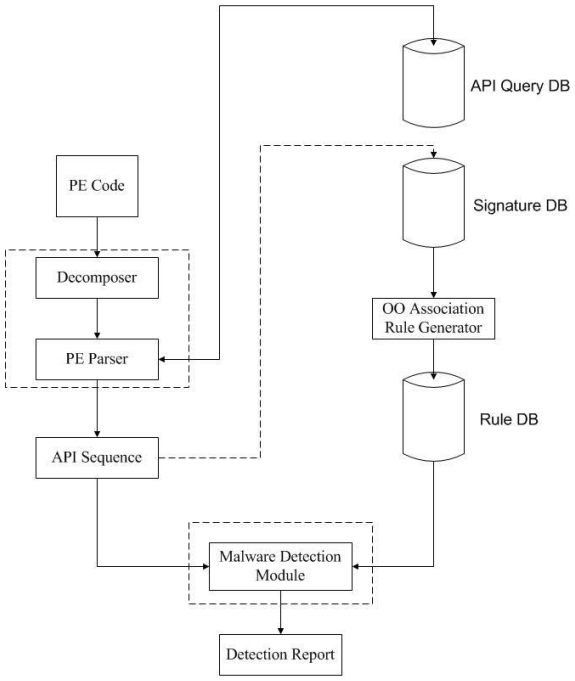

Figure 1. IMDS system architecture.

Then we use the API calls as the signatures of the PE les 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. 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.

IV. 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 predetermined 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 over fitting issues in classification and achieves excellent accuracy [2]. In our IMDS system, we adapted OOA mining techniques [14] to generate the rules.

A. OOA Definitions

In our IMDS system, the goal is to find out how a set of API calls supports the specific objectives:

$$Obj_1 = (Group = Malicious), and\ Obj_2 = (Group = Bening).$$

1) Definition1: (Support and confidence) $I = \{I_1, \cdots \cdots, I_2\}$ 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 returns the number of records in the dataset DB where holds.

2) Definition 2: (OOA frequent itemset) Given $mos\%$ as a user-specified minimum support. I is OOA frequent item set/pattern in DB if $os\% \geq mos\%$.

3) Definition 3: (OOA rule) Given $moc\%$ as a user-specified confidence. Let $I = \{I_1, \ldots \ldots I_m\}$ be an OOA frequent itemset. $I \rightarrow Obj(os\%, oc\%)$ is an OOA rule if $oc\% \geq moc\%$.

B. 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 FPtree and generating frequent patterns from the tree. But the structure of an OOA Fast FP-tree is different from that of an OOA FPtree 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].

C. 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)→ obj1=(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.

D. 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:

$$Obj_1 = (Group = Malicious), and\ Obj_2 = (Group = Bening).$$

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 _les 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.

V. DATA COLLECTION

As stated previously, we obtained 29580 Windows PE _les 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 _les of Windows 2000/NT and XP operating system.

VI. 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.

A. 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, 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.

B. 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.

1) Polymorphic Virus Detection

In this experiment, we sample 3000 malicious _les and 2000 benign _les 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.

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.

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 illustrate that our IMDS system achieves much higher efficiency than other scanners when being executed 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, the obtained results clearly shows that the false positives by using our IMDS system are much fewer than other scanners.

C. Comparisons of Different Classification Methods

In this set of experiments, we compare our system with Naïve 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], 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 as close to 1 as possible; and selecting a big without exceeding the maximum support in the data set. Then, in the experiment, we set to 0.294 and to 0.98. Ten-fold cross validation is used to evaluate the accuracy of each classifier.

VII. 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 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.

VIII. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for association rulemining. In *Proceedings of VLDB-94*, 1994.

[2] H. Cheng, X. Yan, J. Han, and C. Hsu. Discriminative frequentpattern analysis for effective classi_cation. In *ICDE-07*, 2007.

[3] M. Christodorescu and S. Jha. Static analysis of executables to detectmalicious patterns. In *Proceedings of the 12th USENIX SecuritySymposium*, 2003.

[4] M. Fan and C. Li. Mining frequent patterns in an fp-tree withoutconditional fp-tree generation. *Journal of Computer Research andDevelopment*, 40:1216.1222, 2003.

[5] J. Han and M. Kamber. *Data mining: Concepts and techniques, 2ndedition*. Morgan Kaufmann, 2006.

[6] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidategeneration. In *Proceedings of SIGMOD*, pages 1.12, May 2000.

[7] C. Hsu and C. Lin. A comparison of methods for multiclass supportvector machines. *IEEE Trans. Neural Networks*, 13:415.425, 2002.

[8] J. Kephart and W. Arnold. Automatic extraction of computer virussignatures. In *Proceedings of 4th Virus Bulletin InternationalConference*, pages 178.184, 1994.

[9] J. Kolter and M. Maloof. Learning to detect malicious executables inthe wild. In *Proceedings of KDD'04*, 2004.

[10] B. Liu, W. Hsu, and Y. Ma. Integrating classi_cation and associationrule mining. In *Proceedings of KDD'98*, 1998.

[11] H. Peng, F. Long, and C. Ding. Feature selection based on mutualinformation: Criteria of max-dependency, max-relevance, andmin-redundancy. *IEEE Trans. Pattern Analysis and MachineIntelligence*, 27, 2005.

[12] J. Rabek, R. Khazan, S. Lewandowski, and R. Cunningham.Detection of injected, dynamically generated, and obfuscatedmalicious code. In *Proceedings of the 2003 ACM workshop on Rapidmalcode*, pages 76.82, 2003.

[13] M. Schultz, E. Eskin, and E. Zadok. Data mining methods for detection of new malicious executables. In *Proceedings of IEEEInternational Conference on Data Mining*, 2001.

[14] Y. Shen, Q. Yang, and Z. Zhang. Objective-oriented utility-basedassociation mining. In *Proceedings of IEEE InternationalConference 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 AnnualComputer Security Applications Conference*, 2004.

[16] J. Swets and R. Pickett. *Evaluation of diagnostic system: Methodsfrom signal detection theory*. Acdemic 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 InternationalConference 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.

[21] Yanfang Ye*, Dingding Wang, Dongyi Ye, "Intelligent Malware Detection System" *in proceedings of Industrial and Government Track Short Paper*, 2007.