http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (2): 380-384, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.380.384

Malware Detection Through Mining Symbol Table of Linux Executables

Jinrong Bai, Yanrong Yang, Shiguang Mu and Yu Ma
School of Information Technology and Engineering, Yuxi Normal University, 653100, China

Abstract: The popularity of Linux has been increasing over the years and many popular applications are available for Linux. Malware detection method is rarely studied in the Linux platform at present, the main analysis and detection methods still have a lot of limitations. System calls from symbol table which can reflect the behavior of program code pieces and carry semantic interpretations which can reflect an attacker's intent and goal. This study proposed a new malware detection method by mining system calls from symbol table of Linux executables. The experimental results show that the detection method proposed in this work achieves more than 98% accuracy for distinguishing between benign and malicious executables. Compared with other static approaches, the proposed method can identify known and unknown malware and is hard to evade by malware which applies the obfuscation and packing technique.

Key words: Information security, data mining, malware detection, symbol table, system call

INTRODUCTION

Linux is a free and open source operating system which have similar kernel with UNIX operating system. At present there are many different Linux releases which can be installed in all kinds of computer hardware equipment such as mobile phones, tablet PC, routers, video game console, the desktop computer and super computer. Linux is a leading operating system and the world's fastest supercomputers are based on Linux kernel including the 10 fastest (http://linux.wikia.com/wiki/Linux).

With the increasing development and application of Linux, more disruptive malware appeared in the Linux platform. Malware (malicious software) refers program or file harmful to user computer including computer viruses, worms, Trojans, spyware and so on. Malware can disrupt computer operation, gather sensitive information or gain unauthorized access to computer systems without user permission (McGraw and Morrisett, 2000). There are billions of host computers, network and website are attacked, compromised and tampered. Unfortunately, the current malware analysis and detection technology are insufficient and still has a lot of limitations. On the one hand, the attack methods and procedures become more and more complicated and cunning; on the other hand, the vulnerabilities existing in the information system increase quickly. Furthermore, the attackers present novel anti-detection and evasion techniques, which makes it mandatory for the defenders to update the existing defense techniques (Chen *et al*., 2008).

Now commercial anti-virus software deploys signature-based detection technology which can only identify known malware and cannot identify unknown malware. Malware signature is a series of binary string in the malware that uniquely identifies it. Those binary strings are extracted from the malware's machine code and raw file contents. Traditional malware signatures ineffectively identify self-mutating and modified variants in a strain of malware which change their appearance every time when they propagate (Cesare and Xiang, 2010). Because malware detection method is rarely studied in the Linux platform at present, the main analysis and detection methods still have a lot of limitations. System calls from symbol table which can reflect the behavior of program code pieces and carry semantic interpretations which can reflect an attacker's intent and goal (Ye *et al*., 2009). The system calls are good static features since they can not only parse the possible behaviors of a malicious executable but also capture the malware author's intent and goal. This study proposed a new malware detection method by mining system calls from symbol table of Linux executables. The goal of this study is to realize a new Linux malware detection method which can identify known and unknown malware and is hard to evade by malware which applies the obfuscation and packing technique.

In order to overcome the disadvantages of the widely used signature-based malware detection method, data mining and machine learning approaches are proposed for malware detection. The performance of these techniques critically depends on the set of features used to describe the executables and the classifier.

Schultz *et al*. (2001) proposed a data-mining framework that detects new, previously unseen malicious

Corresponding Author: Jinrong Bai, School of Information Technology and Engineering, Yuxi Normal University, 653100, China
Tel: +8613982275276 Fax: +8608772051657

executables accurately and automatically. They used three different features for detecting malicious executable files on the Windows platform. The first technique uses the DLLs, function calls and the number of different function calls used within each DLL as features. In the second technique, they used the consecutive printable characters in the binary as binary features (i.e., present or absent). The third technique uses two-byte byte sequences as binary features. Later on, Kolter and Maloof (2006) have improved the third technique by Schultz *et al.* (2001) by using 4 g as binary features.

Santos *et al.* (2011) proposed a semi-supervised learning technique employed for the detection of unknown malware. They represented the files using byte n-gram patterns, a well-known approach that has achieved good results with supervised machine learning. This methodology can reduce labeling efforts, while still maintaining a high rate of accuracy.

Shabtai *et al.* (2012) used the OpCode n-grams distribution which is extracted from the disassembled files. The OpCode n-gram patterns are used as features for the classification process. OpCode n-gram of various size representations and eight types of classifiers were evaluated in this study. Experimental results indicate that this methodology achieves a level of accuracy higher than 96%.

Shafiq *et al.* (2009) proposed PE-Miner that mines structural features-extracted from different sections of PE headers-of Windows executables and uses it to detect malware. Shahzad and Farooq (2012) proposed ELF-Miner, a malware detection framework that mines structural information-extracted from different sections of ELF headers-of Linux executables.

All those researches show the efforts and achievements in unknown malware classification and analysis.

OVERVIEW OF ELF FORMAT

A binary file typically contains various headers that describe the organization of the file and a number of sections which hold various information about the program such as instructions, data, read-only-data, symbol table, relocation tables and so on.

The ELF (Executable and Linking Format) was originally developed and published by UNIX System Laboratories (USL) as part of the Application Binary Interface (ABI). Now-a-days, the most common Linux file type in use is ELF format. ELF Object files participate in program linking (building a program) and program execution (running a program). For convenience and efficiency, the object file format provides parallel views of

Linking view	Execution view
ELF header	ELF header
Program header table	Program header table
Section i	Segment m
…	…
Section j	Segment n
Section header table	Section header table

Fig. 1: The ELF format

a file's contents, reflecting the differing needs of these activities. The different viewing perspectives are shown in Fig. 1. So, for the linking view, the sections and section header table are important, the program header table is optional. On the other hand, for the execution view, the segments and program header table are important and the section header table is optional.

An ELF header resides at the beginning and holds a ''road map'' describing the file's organization. A program header table, if present, tells the system how to create a process image. A section header table contains information describing the file's sections. Sections hold the bulk of object file information: instructions, data, symbol table, relocation information and so on. Each section occupies one contiguous (possibly empty) sequence of bytes within a file.

Sharable objects and dynamic executables usually have 2 distinct symbol table sections, one named ".symtab" and the other ".dynsym". The dynsym is a smaller version of the symtab that only contains global symbols. The dynsym is allocable and contains the symbols needed to support runtime operation. An object file's symbol table holds information needed to locate and relocate a program's symbolic definitions and references. Symbols manage symbolic names to functions and data. Debuggers use them to figure out what they are looking at and to provide you with a human readable view of that information.

SYSTEM FRAMEWORK

As shown in Fig. 2. proposed malware detection method is carried out with four major steps:

- **Step 1:** First developing a ELF parser to analyze ELF file
- **Step 2:** Performing feature extraction to select informative system calls related to different types of malware

- **Step 3:** Followed by using classification algorithms to construct the classifier
- **Step 4:** Finally applying classifier to classify a file as either benign or malicious executables

EXPERIMENT

Datasets: In this section, an overview of the datasets used in this research is presented. Datasets are divided into malware and benign executables. This study collected 756 benign ELF executables and 763 malware. The benign ELF files are collected from Linux operating system's directories/bin,/sbin and/usr/bin. Linux malware dataset is collected from VX heavens. The distribution of malware is in Table 1.

Feature extraction: This study predicted the behavior and functionality of an executable program by analyzing its usage of the system calls. The ELF executable includes a dynsym section that determines what system calls are imported by the executable. The information contained in the dynsym section is used by the loader at runtime to identify the addresses of the system calls so that whenever a system call is called, a jump to the system call code is executed. User-level malware programs require the invocation of system calls to interact with the OS in order to perform malicious actions. Therefore, analyzing and extracting malicious behaviors from these programs requires the identification of system calls invoked from within the code. Based on experiences of malware analysis, this research presented that the system calls could be better static features for malware detection.

Data mining techniques for malware detection usually starts with the first step of generating a feature set. The feature extraction block first does a validity check to confirm the validity of an ELF file. If the file is legitimate ELF executable, it extracts system calls from the dynsym section. This study dumped each executable file's dynsym section of the ELF format and counted all system calls that are used. Removing those system calls appeared less than 100 times, there are 246 different system calls left. This study computed information gain for each system call and selected the best 100 system calls for later classification according to the information gain measure. Compared with the feature extraction approach by running the program in a sandbox and focusing on its interaction with the operating system, static feature extraction method proposed in this work is easier and less as the file features, it is not easy for the malware to evade this detection method. This is because even if they generate the variants of the malware by recompiling or adopting obfuscation techniques such as polymorphism and metamorphism, it is impractical for them to modify all of the system calls in their programs.

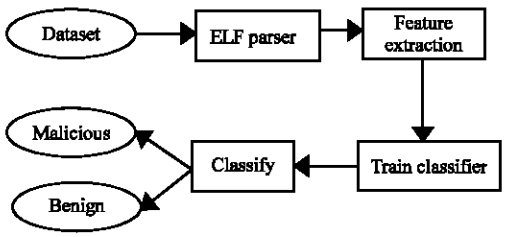

Fig. 2: The architecture of proposed method

Classification: Recently, data mining and machine learning technology is applied in the field of malware detection. It has been become the point study, because it can make use of data mining technology, dig meaningful patterns from a large number of malware and make use of machine learning technology to help summarize the identification knowledge of known malicious code.

The particular problem of labeling a program as malicious or benign is an example of the general classification problem. A number of classifiers have been built and shown to have very high accuracy rates. This study applied several classification learning methods, which are implemented in WEKA (Witten *et al*., 2011): J48 (decision tree) (Witten *et al*., 2011), IBk (*k*-nearest neighbor algorithm) (Witten *et al*., 2011) and Random Forest (Breiman, 2001). This work also employed ensemble methods-Boosting (Freund and Schapire, 1997) to improve performance of J48.

Evaluation description: Since this research have only a finite set of samples, evaluation involves splitting the available samples into a training set and a testing set. The standard 10 fold cross-validation processes is used in this research. This methodology helps in systematically evaluating the effectiveness of this study to detect previously unknown (i.e., zero-day) malicious ELF files.

For evaluation purposes, the following classical measures are usually employed. The True Positive Rate (TPR) measure is the rate of positive instances (i.e.,

Table 1: The distribution of ELF format malware

Malware categories	Backdoor	Rootkit	DoS	Flooder	Exploit	Worm	Trojan	Virus	Total
Quantity	100	70	81	89	166	61	63	133	763

malware files) classified correctly. False Positive Rate (FPR) is the rate of negative instances (i.e., benign files) misclassified. The Total Accuracy measures the number of absolutely correctly classified instances, either positive or negative, divided by the entire number of instances.

The ROC (Receiver Operating Characteristic) curve is a graph produced by plotting the fraction of TPR versus the fraction of FPR for a binary classifier as its discrimination threshold varies. This study also used the Area Under the ROC Curve (AUC) measure in the evaluation process. A higher AUC value means that the ROC graph is closer to the optimal threshold. ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution.

EXPERIMENTAL RESULTS

This study employed four classification algorithms (J48, Random Forest, AdboostM1(J48) and IBk) to train classifier. Detection accuracy of trained classifiers is about 98%. The experimental results are presented in Table 2. The ROC curves for these methods are shown in Fig. 3.

Table 2: Experimental results for four classification algorithms

Classification algorithm	TPR (%)	FPR (%)	Accuracy (%)	AUC
J48	99.7	4.5	97.7	0.987
Random Forest	99.4	3.1	98.2	0.985
AdboostM1(J48)	99.2	2.2	98.5	0.993
IBk	99.7	2.5	98.6	0.993

AUC: Area un ROC curve, TPR: True positive rate, FPR: False positive rate

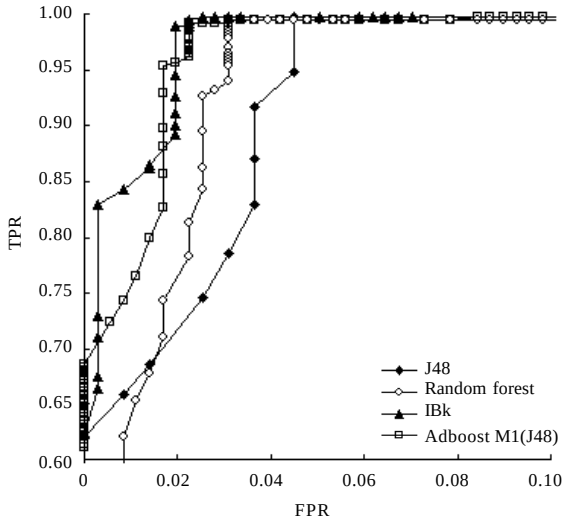

Fig. 3: ROC curves for four classification algorithms, TPR: True positive rate, FPR: False positive rate

As can be seen from Table 2, all classification algorithms detected above 99% of new malware and the relative performance of those methods is roughly the same. This work can conclude from Table 2 that IBk outperforms the rest of the data mining classifiers in terms of the detection accuracy. Table 2 shows the AUC results of all classification algorithms. The AUC results of all methods are above 0.985, all methods performed comparably and the AUC value of two methods AdboostM1 and IBk) is 0.993. Figure 3 shows the ROC curves of all methods. The ROC curve for IBk dominated all others and J48 did not performed as well as all others. To sum up, mining system calls from symbol table of Linux executables is a feasible way to identify known and unknown Linux malware.

CONCLUSIONS

This study has introduced a new malware detection method that mines system calls information extracted from symbol table of Linux executables. This work developed a malware detection system consisting of an ELF parser, feature extraction, classifier training and malware detector. Proposed method achieves more than 98% accuracy depending on the selected classifier. Compared with other static approaches, detection method proposed in this work is hard to evade by malware which applies the obfuscation and packing technique.

ACKNOWLEDGMENTS

This research was supported by the National 863 Plans Projects of China under Grant No. 2008AA01Z208.

REFERENCES

Breiman, L., 2001. Random forest. Mach. Learning., 45: 5-32.

Cesare, S. and Y. Xiang, 2010. Classification of malware using structured control flow. Proceedings of the 8th Australasian Symposium on Parallel and Distributed Computing, January 18-22, 2010, Australian Computer Society, Inc., Australia, pp: 61-70.

Chen, X., J. Andersen, Z.M. Mao, M. Bailey and J. Nazario, 2008. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. Proceedings of the 38th Annual IEEE International Conference on Dependable Systems and Networks, June 24-27, 2008, IEEE, USA., pp: 177-186.

Freund, Y. and R.E. Schapire, 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci., 55: 119-139.

Kolter, J.Z. and M.A. Maloof, 2006. Learning to detect malicious executables in the wild. J. Mach. Learnin. Res., 7: 2721-2744.

McGraw, G. and G. Morrisett, 2000. Attacking malicious code: A report to the infosec research council. IEEE Software, 17: 33-41.

Santos, I., J. Nieves, and P. G. Bringas, 2011. Semi-supervised learning for unknown malware detection. Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, April 6-8, 2011, Springer Spain, pp: 415-422.

Schultz, M., E. Eskin, E. Zadok and S.J. Stolfo, 2001. Data mining methods for detection of new malicious executables. Proceedings of the IEEE Symposium on Security and Privacy, May 14-16, IEEE Computer Society Washington, DC, USA., pp: 38-49.

Shabtai, A., R. Moskovitch, C. Feher, S. Dolev and Y. Elovici, 2012. Detecting unknown malicious code by applying classification techniques on OpCode patterns. Secur. Inf., Vol: 1 10.1186/2190-8532-1-1

Shafiq, M.Z., T.S. Momina, F. Mirza and M. Farooq, 2009. PE-Miner: Mining structural information to detect malicious executables in real time. Proceedings of the Recent Advances in Intrusion Detection, September 23-25, 2009, France, Springer, pp: 121-141.

Shahzad, F. and M. Farooq, 2012. Elf-miner: Using structural knowledge and data mining methods to detect new (linux) malicious executables. Knowl. Inform. Syst. J., 30: 589-612.

Witten, I.H., E. Frank and A.H. Mark, 2011. Data Mining: Practical Machine Learning Tools and Techniques. 3rd Edn., Morgan Kaufmann, San Francisco, CA.

Ye, Y., L. Chen, D. Wang, T. Li, Q. Jiang and M. Zhao, 2009. SBMDS: An interpretable string based malware detection system using SVM ensemble with bagging. J. Comput. Virol. J., 5: 283-293.