A REVIEW ON SVM TRAINING PHASE REDUCTION FOR MALWARE DETECTION

Abhijeet J. Choudhari
Dept. of Computer Science & Engineering.
G.H.R.A.E.T., Nagpur.

Vikrant Chole
Dept. of Computer Science & Engineering.
G.H.R.A.E.T., Nagpur.

Abstract-With the increasing number of malwares, proper malware detection technique is a serious issue. Malware detection needs to differentiate between malware codes and the benign software. This survey provides an overview of different malware detection techniques proposed by different researchers, along with their advantages and limitations. When the malware family is known , the traditional signature based method and static analysis is used. However, unknown malware detection is an important issue to deal with. Efficiency of some of the techniques are focused in this paper.

Keywords :- Malware Detection , SVM , N-gram analysis , benign softwares .

I. INTRODUCTION

Recent years have shown a massive growth in malware with many computer systems are vulnerable to and infected with malware and malicious software. The amount and variety of malware increases day by day with the use of internet and along with this the malware writers use code obfuscation techniques to disguise the already known security threat from classic syntactic malware detectors.

To deal with this problem, several malware detection approaches are studied. The signature detection and monitoring suspected code for known security vulnerabilities are becoming ineffective and intractable. In response to this, researchers need to develop and implement effective intrusion detection techniques. There are many ways to mitigate malware infection and spread. Tools such as anti-virus and anti-spyware are able to identify and block malware based on its behavior or static features. A static feature may be a rule or a signature that uniquely identifies a malware or malware group. While the tools mitigating malware may vary, at their core there must be some classification method to distinguish malware files from benign files. In recent years many researchers have been using machine learning techniques to produce a binary classifier that is able to distinguish malware from benign files.

The crucial issue is the ability to detect the unknown malwares. The traditional statical methods to deal with unknown malwares are becoming ineffective. Several studies have been investigated in the direction of detecting unknown malwares based on its binary codes. Some of the methods use machine learning techniques. The runtime classification of malware codes and benign software is important. Efficient analysis and classification techniques have been used by many researchers.

One such approach is the Support Vector Machines (SVMs) which have attracted much attention as a new classification technique with good generalization ability. The basic idea of SVMs is to map input vectors into a high-dimensional feature space and linearly separate the feature vectors with an optimal hyperplane in terms of margins. Kernel functions are the root of support vector machine, since if we properly choose kernel function can make a lot of difference in classification problems.

II. LITERATURE REVIEW

A lot of work has been done in the detection of malicious opcode using both the static analysis and the dynamic analysis methods. Malware detection research can also be categorized in the way the information is processed after it is captured. The following Section gives a review of work done by various researchers.

The calls made to the operating system by the binary executables are used to determine whether it is malicious or not. The malicious programmers hide this information by replacing the call instruction by two *push* instruction and one *ret* instruction. After the ret instruction, the first push pushes the address of instruction and the second push pushes *ret*. The code is obfuscated by spreading these three instructions and then each instruction is splitted into multiple instructions. Lakhotia et al. [1] presented the use of abstract interpretation to detect the normal call/ret convention violations. An abstract stack relates every element of stack to the instruction, which pushes the element. It supports the disassembly of obfuscated code and detects obfuscations related to stack operations. This abstract stack graph technique is effective in finding the call/retn obfuscations. However , it is a partial solution , since it is confined to only few stack related operations such as *push , pop, call* and *ret.* this technique did not handle the condition when push and pop are splitted into multiple instructions and does not model the memory locations and register.

Bilar [2] compared the opcode distributions of malicious and benign software using static analysis method. It was then used to detect and differentiate polymorphic and metamorphic malwares. The polymorphic malware contains decryption routines which decrypt the encrypted constant parts of the malware body. The metamorphic malware generally do not use encryption but mutate their body in the next subsequent generations. Bilar extracted the opcodes and performed statical analysis. Bilar's findings showed that the most frequent opcodes (push, pop, mov) were the low predictors of malware while the rarer opcodes were stronger predictors of malware.

The traditional signature based malware detection method is inefficient to detect unknown malwares. Santos [4] presented the use of n-gram analysis to detect the unknown or new viruses. A collective classification approach is used which classifies a set of labeled and unlabelled instances. The classifier achieves a high performance using labeled set to detect unknown malwares.

Seker et al. [5] used Finite State Automaton for intrusion detection in program behavior. Along with the system call sequences, FSA also captured the program's branching and looping structure. The FSA approach performs more accurate detection of execution of unusual sections of code. They found that the FSA approach is quick learning, it gives low false positive rate and has less space and runtime requirements.

Santos et al. [7] implemented a technique for malware detection based on frequency of appearance of opcode sequence. Afterwards , they mine the relevance of each opcode , and then weigh each opcode sequence frequency. Mining opcode relevance gives more accurate detection of malware variants and is used to weigh the similarity function and also reduces the irrelevant opcode noise. The Santos [7] findings showed that when n=1, the comparison of malware families and their variants have a high degree of similarity. Thus using n=1 is not appropriate. The comparison results for n=2 gives more distributed degree of frequency. Therefore this method is able to detect high number of malware variants, by selecting a proper threshold of similarity ratio.

The supervised learning has a limited use because it needs a large number of malicious code and benign software first needs to be labeled. Santos et al [8] in its further research used a single class learning to detect new malware families. It examines the frequencies of appearance of opcode sequences to build the classifier. This uses only one set of labeled instances of a particular single class either malware or legitimate software. Santos [8] finds showed that it is better to label benign software instead of malware class when we can only label a less number of benign software , however if we can label a huge amount of malware class , then it is efficient to label the malware instances. Hence this research shows that single class learning does not require large amount of data and thus the cost of unknown malware detection can be reduced.

Shabtai et al. [10] in his research represented the inspected files using opcode n-gram patterns which are obtained from the executable files after debugging which provide a meaning representation of the code. The n-gram analysis is performed by varying n from 1 to 6. Shabtai findings showed that the when n-gram of size 2 is used it gives highest accuracy and G-means value and lowest False Positive Rate (FPR). This shows that opcode of sequence 2 is more representative than single opcodes , however the longer sequences decreases the accuracy. The opcodes generated are used as features for classification. The main goal for classification is to detect the unknown malwares. While using binary classification , the imbalance problem occurs when the portion of classes are not equal. A chronological evaluation is presented for frequent need of updating the training set. The evaluation methodology achieves a higher rate of accuracy which improves the overall performance as the training set is updated. Random Forest Algorithm [17] a complex classifier induces many decision trees and then combine the results of all tree and the boosted decision tree [18] which generates more accurate classifier.

Moskovitch [11] also investigated the detection of unknown malware using n-gram sequences. However , most of the studies use byte sequence n gram binary codes of executables , while Moskovitch used the Operation Codes (opcodes) generated by dissembling the executables.

Song et al. [12] presented a quantitative analysis of the strength and limitations of shellcode polymorphism its effect on the current intrusion detection technique. Modeling the classes of self-modifying code is untraceable by previously known methods for both statical and string signatures. The research improves the polymorphic techniques and creates proof - of –concepts. It determines if malcode itself has any distinguishing features that support the construction and use of exploit statical models. The technique combines two polymorphic engines CLET [15] and ADMmutatae [16] The CLET engine ciphers the shellcode and the ADMmutate hides the CLET's decoder. This combination makes shellcode not only impossible to model but also allows to exploit instance to blend in with normal network traffic. This findings shows that the polymorphism is significantly replacing signature based methods.

Sr. No.	Technique	Advantages	Disadvantage	Remarks
1.	Abstract Stack Graph technique	Supports the disassembly of obfuscated code & is effective in finding the call/retn obfuscations.	Provides only few stack related operations.	Provides a partial solution.
2.	Opcode comparison using static analysis method.	Useful in detecting polymorphic and metamorphic malwares.	It is a static analysis method.	Frequent opcodes were low predictors of malware while rarer ones were stronger .
3.	n-gram analysis	achieves a high performance using labeled set to detect unknown malwares.	When n increases, size increases largely for n>2.	Detects unknown malwares.
4.	Finite State Automaton for intrusion detection	Performs more accurate detection of execution of unusual sections of code.		Quick learning, low false positive rate .
5.	Malware detection based on frequency of appearance of opcode sequence.	gives more accurate detection of malware variants and removes irrelevant noise.	Does not gives accurate results for n=1.	Detect high number of malware variants, by selecting a proper threshold of similarity ratio.
6.	Quantitative analysis of shellcode polymorphism	Improves polymorphic techniques and creates proof-of-concepts .		Polymorphic techniques replaces signature based models.

Table 1. Comparison of some malware detection techniques.

III. CONCLUSION

In this survey, many different malware detection techniques have been studied. These techniques can be differentiated for the known malwares and for unknown malwares. For the known malware vulnarities, the traditional methods can be used . However for the unknown malwares, these classical signature based detection and statical analysis have been proved inefficient. In such a condition, the newly developed techniques need to be implemented, where rigorous classification and analysis is required, which is achieved using SVM or n-gram analysis. The techniques studied above like abstract stack graph or polymorphic methods efficient to detect unknown malwares, with some drawbacks. Two or more techniques can be used in combination for higher efficiency.

IV. REFERENCES

[1] A. Lakhotia, E. U. Kumar, and M. Venable, “A method for detecting obfuscated calls in malicious binaries,” *IEEE Trans. Software Eng.*, vol. 31, no. 11, pp. 955–968, Nov. 2005.

[2] D. Bilar, "Opcodes as predictor for malware," *Int. J. Electron. Security Digital Forensics*, vol. 1, no. 2, pp. 156–168, 2007.

[3] D. Bilar, "Callgraph properties of executables and generative mechanisms," *AI Commun., Special Issue on Network Anal. in Natural Sci.and Eng.*, vol. 20, no. 4, pp. 231–243, 2007.

[4] I. Santos, Y. K. Penya, J. Devesa, and P. G. Garcia, "N-grams-based file signatures for Malware Detection," *S3Lab, Deusto Technological Found.*, 2009 [Online].Available: pgbg@technologico.deusto.es

[5] R. Sekar, M. Bendre, D. Bollineni, and Bollineni, R. Needham and M. Abadi, Eds., "A fast automaton-based method for detecting anomalous program behaviors," in *Proc. 2001 IEEE Symp. Security and Privacy,IEEE Comput. Soc.*, Los Alamitos, CA, USA, 2001, pp. 144–155.

[6] W. L. K. Wang, S. Stolfo, and B. Herzog, "Fileprints: Identifying file types by n-gram analysis," in *Proc. 6th IEEE Inform. Assurance Workshop*,Jun. 2005, pp. : 64–71.

[7] I. Santos, F. Brezo, J. Nieves, Y. K. Penya, B. Sanz, C. Laorden, and P. G. Bringas, "Opcode –sequence –based malware detection," in *Proc. 2nd Int. Symp. Eng. Secure Software and Syst. (ESSoS)*, Pisa, Italy, Feb. 3–4, 2010, vol. LNCS 5965, pp. 35–43.

[8] I. Santos, F. Brezo, B. Sanz, C. Laorden, and Y. P. G. Bringas,"Using opcode sequences in single-class learning to detect unknown malware," *IET Inform. Security*, vol. 5, no. 4, pp. 220–227, 2011.

[9] I. Santos, F. Brezo, X. Ugarte-Pedrero, and Y. P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inform. Sci.*, 2011 [Online]. Available: http://dx.doi.org/10.1016/j.ins.2011.08.020

[10] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting unknown malicious code by applying classification techniques on Opcode patterns," *Security Informatics*, vol. 1, pp. 1–22, 2012.

[11] R. Moskovitch, C. Feher, N. Tzachar, E. Berger,M.Gitelman, S.Dolev,and Y. Elovici, "Unknown malcode detection using opcode representation," in *Proc. 1st Eur Conf. Intell. and Security Informatics (EuroISI08)*, 2008, pp. 204–215.

[12] Y. Song, M. Locasto, and A. Stavro, "On the infeasibility of modeling polymorphic shellcode," in *Proc. ACM Conf. Computer and Commun. Security*, 2007, pp. 541–551.

[13] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang, "Effective and efficient malware detection at the end host," in *Proc. 18th Usenix Security Symp.*, 2009, pp. 351–366.

[14] Christodorescu M., Jha S., "Static analysis ofexecutables to detect malicious patterns",Proceedings of the 12th USENIX SecuritySymposium, Washington (DC), August 2003, pp.169–186

[15] Detristan, T., Ulenspiegel, T., Malcom, Y., And Von Underduk, M. S. Polymorphic Shellcode Engine Using Spectrum Analysis. Phrack 11, 61-9 (2003).

[16] B. E. Bernhard, G. M. Isabelle, and V. N. Vladimir, H. Haussler, Ed.,"A training algorithm for optimal margin classifiers," in *Proc. 5th Ann.ACM Workshop on COLT ACM Press*, Pittsburgh, PA, USA, 1992, pp.144–152

[17] Kam HT: Random Decision Forest. Proc of the 3rd International Conference on ocument Analysis and Recognition 1995, 278-282.

[18] Freund Y, Schapire RE: A brief introduction to boosting. International Joint Conference on Artificial Intelligence Morgan Kaufmann Publishers Inc; 1999, 1401-1406.