ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Detection of Packed and Polymorphic Malware Using Malwise

Senthil Arasu. R[1]
M.Tech Student
Department of Computer Science and Engineering
PRIST University
Pondicherry - India.

Bharathi. R[2]
Assistant professor
Department of Computer Science and Engineering
PRIST University
Pondicherry - India.

Abstract: ***Malware is a pervasive problem in distributed computer and network systems. Malware variants often have distinct byte level representations while in principal belong to the same family of malware. The byte level content is different because small changes to the malware source code can result in significantly different compiled object code. Malware variants with the umbrella term of polymorphism, using the approach of structuring and decompilation to generate malware signatures. Employing both dynamic and static analysis to classify malware. Entropy analysis initially determines if the binary has undergone a code packing transformation. If packed, dynamic analysis employing application level emulation reveals the hidden code using entropy analysis to detect when unpacking is complete. Static analysis then identifies characteristics, building signatures for control flow graphs in each procedure. The similarities between the set of control flow graphs and those in a malware database accumulate to establish a measure of similarity. A similarity search is performed on the malware database to find similar objects to the query. Additionally, a more effective approximate flow graph matching algorithm is proposed that uses the decompilation technique of structuring to generate string based signatures amenable to the string edit distance, using real and synthetic malware to demonstrate the effectiveness and efficiency of Malwise.***

Keywords: Computer security, malware, control flow, structural classification, and structured control flow, unpacking

I. INTRODUCTION

1. Network Security

Network security starts with authenticating the user, commonly with a username and a password. Since this requires just one detail authenticating the user name i.e. the password, which is something the user 'knows' this is sometimes termed one-factor authentication. With two-factor authentication, something the user 'has' is also used with three-factor authentication, something the user 'is' is also used (e.g. a fingerprint or retinal scan).

Once authenticated, a firewall enforces access policies such as what services are allowed to be accessed by the network users. Though effective to prevent unauthorized access, this component may fail to check potentially harmful content such as computer worms or Trojans being transmitted over the network. Anti-virus software or an intrusion prevention system (IPS) help detect and inhibit the action of such malware.

An anomaly-based intrusion detection system may also monitor the network and traffic for unexpected (i.e. suspicious) content or behavior and other anomalies to protect resources, e.g. from denial of service attacks or an employee accessing files at strange times. Individual events occurring on the network may be logged for audit purposes and for later high-level analysis.

2. Malwares

Malware, short for malicious (or malevolent) software, is software used or created by attackers to disrupt computer operation, gather sensitive information, or gain access to private computer systems. It can appear in the form of code, scripts, active content, and other software. 'Malware' is a general term used to refer to a variety of forms of hostile or intrusive software.

Malware includes computer viruses, worms, Trojan horses, spyware, adware, and other malicious programs. In law, malware is sometimes known as a computer contaminant, as in the legal codes of several U.S. states. Malware is not the same as defective software, which is software that has a legitimate purpose but contains harmful bugs that were not corrected before release. However, some malware is disguised as genuine software, and may come from an official company website. An example of this is software used for harmless purposes that is packed with additional tracking software that gathers marketing statistics.

Malware has caused the rise in use of protective software types such as antivirus, anti-malware, and firewalls. Each of these is commonly used by personal users and corporate networks in order to stop the unauthorized access by other computer users, as well as the automated spread of malicious scripts and software.

3. Network Layer or Packet Filters

Network layer firewalls, also called packet filters, operate at a relatively low level of the TCP/IP protocol stack, not allowing packets to pass through the firewall unless they match the established rule set. The firewall administrator may define the rules; or default rules may apply. The term "packet filter" originated in the context of BSD operating systems.

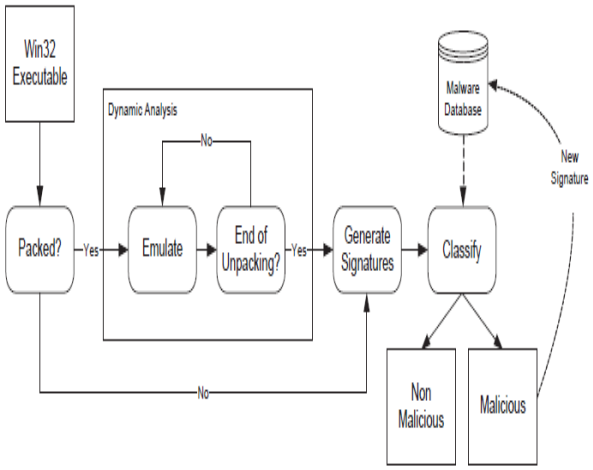
Network layer firewalls generally fall into two sub-categories, stateful and stateless. Stateful firewalls maintain context about active sessions, and use that "state information" to speed packet processing. Any existing network connection can be described by several properties, including source and destination IP address, UDP or TCP ports, and the current stage of the connection's lifetime.

If a packet does not match an existing connection, it will be evaluated according to the rule set for new connections. If a packet matches an existing connection based on comparison with the firewall's state table, it will be allowed to pass without further processing

II. PROPOSED SYSTEM

Our approach employs both dynamic and static analysis to classify malware. Entropy analysis initially determines if the binary has undergone a code packing transformation. If packed, dynamic analysis employing application level emulation reveals the hidden code using entropy analysis to detect when unpacking is complete. If not, then Static analysis then identifies characteristics, building signatures for control flow graphs in each procedure. The similarities between the set of control flow graphs and those in a malware database accumulate to establish a measure of similarity. A similarity search is performed on the malware database to find similar objects to the query. Two approaches are employed to generate and compare flow graph signatures. Two flow graph matching methods are used to achieve the goal of either effectiveness or efficiency. Exact Matching is an ordering of the nodes in the control flow graph is used to generate a string based signature or graph invariant of the flow graph. Approximate Matching is the control flow graph is structured in this approach. Structuring is the process of decompiling unstructured control flow into higher level, source code like constructs including structured conditions and iteration.

III. ARCHITECTURE DIAGRAM



IV. MODULE DESCRIPTION

A. Data Collection

Our data set consists of 100 binaries out of which 90 are benign and 10 are spyware binaries. The benign files were collected from Download.com, which certifies the files to be free from spyware. The spyware files were downloaded from the links provided by SpywareGuide.com. This hosts information about different types of spyware and other types of malicious software.

B. Byte Sequence Generation

We have opted to use byte sequences as data set features in our experiment. These byte sequences represent fragments of machine code from an executable file. We use xxd, which is a UNIX-based utility for generating hexadecimal dumps of the binary files. From these hexadecimal dumps we may then extract byte sequences, in terms of *n*-grams of different sizes.

C. Feature Extraction

The output from the parsing is further subjected to feature extraction. We extract the features by using following approaches, the Common Feature-based Extraction (CFBE) and Frequency-based Feature Extraction. The occurrence of a feature and the frequency of a feature. Both methods are used to obtain Reduced Feature Sets (RFSs) which are then used to generate the ARFF files.

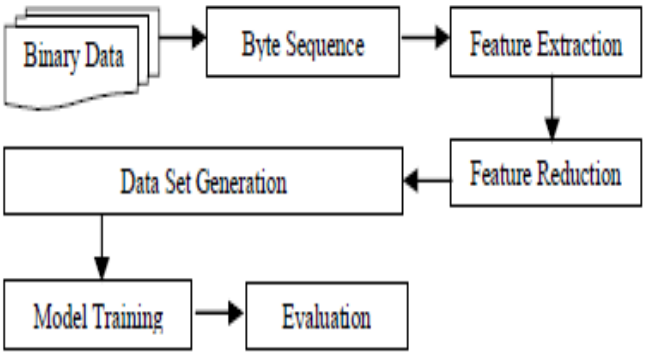
D. Dataset Generation

Two ARFF databases based on frequency and common features were generated. All input attributes in the data set are represented by Booleans. These ranges are represented by either 1 or 0.

E. Classification

A Naive Bayes classifier is a probabilistic classifier based on Bayes theorem with independence assumptions, i.e., the different features in the data set are assumed not to be dependent of each other. This of course, is seldom true for real-life applications. Nevertheless, the algorithm has shown good performance for a wide variety of complex problems. J48 is a decision tree-based learning algorithm. During classification, it adopts a top-down approach and traverses a tree for classification of any instance. Moreover, Random Forest is an ensemble learner. In this ensemble, a collection of decision trees are generated to obtain a model that may give better predictions than a single decision tree.

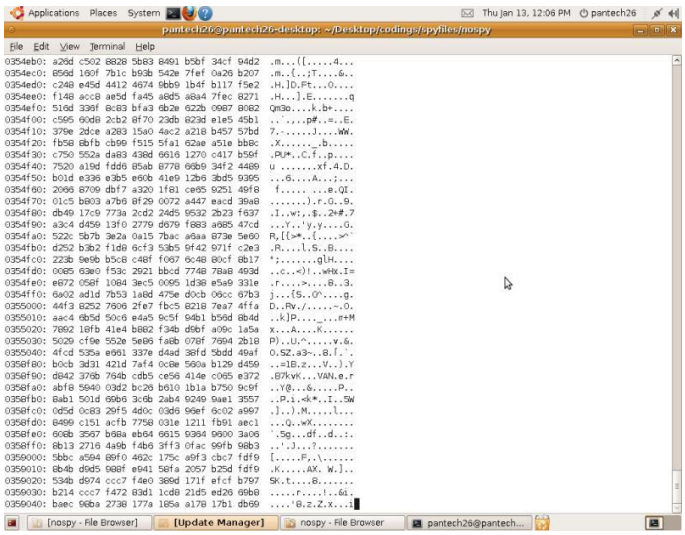
F. PROCESS



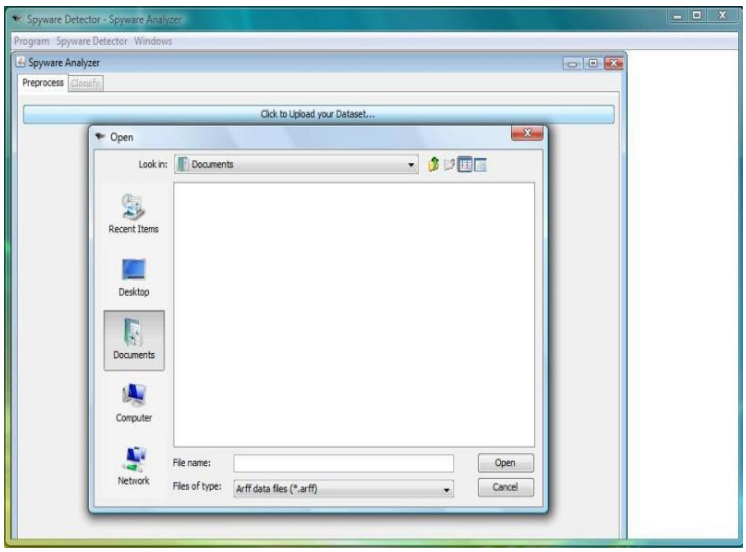
V. SCREEN SHOTS

Our approach employs both dynamic and static analysis to classify malware. Entropy analysis initially determines if the binary has undergone a code packing transformation. If packed, dynamic analysis employing application level emulation reveals the hidden code using entropy analysis to detect when unpacking is complete. If not, then Static analysis then identifies characteristics, building signatures for control flow graphs in each procedure. The similarities between the set of control flow graphs and those in a malware database accumulate to establish a measure of similarity.

G. SPYWARE FILES



H. SPYWARE DETECTOR



VI. Future Enhancement

Feature search method that focuses on selecting generic features that are applicable to different families of viruses. This ensured that our classifier is genuinely heuristic and does not rely on signatures. In future work this project propose focusing on reducing the false positive rate, by using a larger number of benign files, or by training our classifier using a cost matrix and setting a higher cost to misclassifying negative examples. Retrospective testing is a recent evaluation methodology for virus detection systems. In both models the features selected and used by the classifier had comparable overall support within the dataset. This indicates that our feature search method produced features that were more useful in detecting new unseen viruses. Our method does not allow classifiers to use examples in training that are variants of viruses present in the test set. This denies them an unfair advantage that they would not have in real world conditions

VII. Conclusion

Malware can be classified according to similarity in its flow graphs. This analysis is made more challenging by packed malware. Proposed different algorithms to unpack malware using application level emulation, also proposed performing malware classification using either the edit distance between structured control flow graphs, or the estimation of isomorphism between control flow graphs. Implemented and evaluated these approaches in a fully functional system, named Malwise.

References

1. Arnold. W. C and Kephart. J. O, (1994),"Automatic extraction of computer virus signatures," in 4th Virus Bulletin International Conference, pp. 178-184.
2. Briones .I and Gomez.A, (2008), "Graphs, Entropy and Grid Computing Automatic Comparison of Malware," in Virus Bulletin Conference, pp.1-12.
3. Bonfante.G, Kaczmarek.M, and Marion. J.Y ,(2008), "Morphological Detection of Malware," in International Conference on Malicious and Unwanted Software, IEEE, USA, pp. 1-8
4. Carrera .E and Erdelyi.G ,(2004), "Digital genome mapping– advanced binary malware analysis," in Virus Bulletin Conference, pp. 187-197.
5. Gerald. R. T and Lori. A. F ,(2007), "Polymorphic malware detection and identification via context-free grammar homomorphism," Bell Labs Technical Journal, vol. 12, pp. 139-147.
6. Gheorghescu .M ,(2005),"An automated virus classification system," in Virus Bulletin Conference, pp. 294-300.
7. Hamrock. J and Lyda .R ,(2007), "Using entropy analysis to find encrypted and packed malware," IEEE Security and Privacy, vol. 5, p. 40.
8. Karim .M. E , Lakhotia .A and Parida .L, Walenstein .A ,(2005), "Malware phylogeny generation using permutations of code," Journal in Computer Virology, vol. 1, pp. 13-23.
9. Kolter.J.Z and Maloof.M.A ,(2004),"Learning to detect malicious executables in the wild," in International Conference on Knowledge Discovery and Data Mining, pp. 470-478.

Web References

10. http://www.fsecure.com/en_EMEA/aboutus/pressroom/news/2007/fs_news_20071204_1_eng.html.
11. http://research.pandasecurity.com/archive/Mal_2800_ware_2900_format ion-statistics.aspx.
12. http://www.offensivecomputing.net.
13. http://alliance.mwcollect.org.
14. http://www.zynamics.com/vxclass.htm.

Author(s) Profile

Mr. SENTHIL ARASU.R, Received The B.E Degree In Computer Science And Engineering. Presently Pursuing Final Year M.TECH CSE, In PRIST University, Puducherry Campus, Puducherry, India in 2012 and 2014.

Ms. BHARATHI.R, Received The M.Tech In Computer Science And Engineering. Presently she is A Working Assistant Professor in Computer Science and Engineering at PRIST University, Puducherry Campus, and Puducherry, India.

