Proficient classification of packed and polymorphic malware Using malwise

A.THILAGAVATHI,
Student,M.E(C.S.E) IInd Year,
Shri Andal Alagar College of Engineering,
Mamandur
csethilak@gmail.com

MR.P.ELUMALAI
Assistant professor,
Dept of CSE,
A.R Engineering College,
Villupuram

Abstract

String based signatures have remained popular in commercial systems due to their high efficiency, but can be ineffective in detecting malware variants. To classify the packed and polymorphic malware, this paper proposes a novel system named Malwise. Classification is performed using a fast application level emulator to reverse the code packing transformation and three data mining algorithms to perform classification. We use real and synthetic malware to demonstrate the effectiveness and efficiency of Malwise. Polymorphism describes related malware sharing a common history of code. Code sharing among variants can be derived from autonomously self mutating malware, or manually copied by the malware creator to reuse previously authored code. Both static and dynamic analysis will be performed for effective classification.

Index Terms— Computer security, malware, structural classification, unpacking.

I. INTRODUCTION

Malware, short for malicious software, means a variety of forms of hostile, intrusive, or annoying software or program code. 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. In this project we describe malware variants with the umbrella term of polymorphism. We are the first to use the approach of structuring and decompilation to generate malware signatures. String based signatures have remained popular in commercial systems due to their high efficiency, but can be ineffective in detecting malware variants.

A. Existing Approaches and Motivation

Static analysis incorporating n-grams [3, 4], edit distances [5], API call sequences [10], and control flow [1-2] have been proposed to detect malware and their polymorphic variants. However, they are either ineffective or inefficient in classifying packed and polymorphic malware.

The malware's real content is frequently hidden using a code transformation known as packing [7]. Packing is not solely used by malware. Packing is also used in software protection schemes and file compression for legitimate software, yet the majority of malware also uses the code packing transformation.

Unpacking is a necessary component to perform static analysis and to reveal the hidden characteristics of malware. In the problem scope of unpacking, it can be seen that many instances of malware utilize identical or similar packers. Many of these packers are also public, and malware often employs the use of these public packers. Many instances of malware also employ modified versions of public packers. Being able to automatically unpack malware in any of these scenarios, in addition to unpacking novel samples, provides benefit in revealing the malware's real content – a

necessary component for static analysis and accurate classification.

For modern malware classification approaches, a system must be developed that is not only effective against polymorphic and packed malware, but that is also efficient. In this paper we present an effective and efficient system that employs dynamic and static analysis to automatically unpack and classify a malware instance as a variant, based on similarities of features.

B. Contributions

This paper makes the following contributions. First, we propose using a feature search method that focuses on selecting generic features that are applicable to different families of viruses. Second, we propose using three algorithms to classify nonspecific features for exact and approximate identification of flow graphs. Third, we propose and evaluate automated unpacking Using application level emulation that is equally capable of desktop Antivirus integration. The automated unpacked is capable of unpacking known samples and is also capable of unpacking unknown samples. Finally, we implement and evaluate our ideas in a novel prototype system called Malwise that performs automated unpacking and malware classification.

C. Structure of the Paper

The structure of this paper is as follows: Section2 describes related work in automated unpacking and malware classification; Section 3 refines the problem definition and our approach to the proposed malware classification system; Section 4 describes the design and implementation of our prototype Malwise system; Section 5 evaluates Malwise using real and synthetic malware samples; finally, Section 6 summarizes and concludes the paper.

II. RELATED WORK

A. Automated Unpacking

Automated unpacking relies on typical behavior seen in the majority of packed malware – hidden code is dynamically generated and then executed. The hidden code is naturally revealed in the process image during normal execution. Monitoring execution for the dynamic generation and execution of the malware's hidden code can be achieved through emulation [6]. Emulation provides a safe and isolated environment for malware analysis. The advantage of application level emulation over whole system emulation is significantly greater performance. Application level emulation for automated unpacking has had commercial interest [9] but has realized few academic publications evaluating its effectiveness and performance.

B. Polymorphic Malware Classification

A variation of n-grams, coined n-perms has been proposed [4] to describe malware characteristics and subsequently used in a classifier. An alternative approach is using the basic blocks of unpacked malware, classified u sing edit distances, inverted indexes and bloom filters [5]. The main disadvantage of these approaches is that minor changes to the malware source code can result in significant changes to the resulting byte stream after compilation. The novel set similarity search we perform enables the real-time classification of malware from a large database. No prior related research has performed in real-time. Additionally distinguishing our work is the proposed automated unpacking system, which is integrated into the flow graph based classification system.

C. The Difference between Malwise and Previous Work

Our research differs from previous flow graph classification research by using a novel approximate control flow graph matching algorithm employing structuring. No prior related research has performed in real-time. Additionally distinguishing our work is the proposed automated unpacking system, which is integrated into the flow graph based classification system.

III. PROBLEM DEFINITION AND OUR APPROACH

The problem of malware classification and variant detection is defined in this Section.

A. Problem Definition

A malware classification system is assumed to have advance access to a set of known malware. This is for construction of an initial malware database. The database is constructed by identifying invariant characteristics in each malware and generating an associated signature to be stored in the database. After database initialization, normal use of the system commences. The system has as input a previously unknown binary that is to be classified as being malicious or non malicious. The input binary and the initial malware binaries may have additionally undergone a code packing transformation to hinder static analysis. The classifier calculates similarities between the input binary and each malware in the database. If identified as a variant, the database may be updated to incorporate the potentially new set of generated signatures associated with that variant.

B. Our Approach

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. Our classifier is genuinely heuristic and does not rely on signatures. In experiments testing our method against that of leading research, our method achieved better performance. In both models the features selected and used by the classifier had comparable overall support within the dataset.

We also introduced an evaluation method for virus classifiers that tests more convincingly its ability to detect new viruses. Our method does not allow classifiers to use examples in training that are variants of viruses present in the test set.

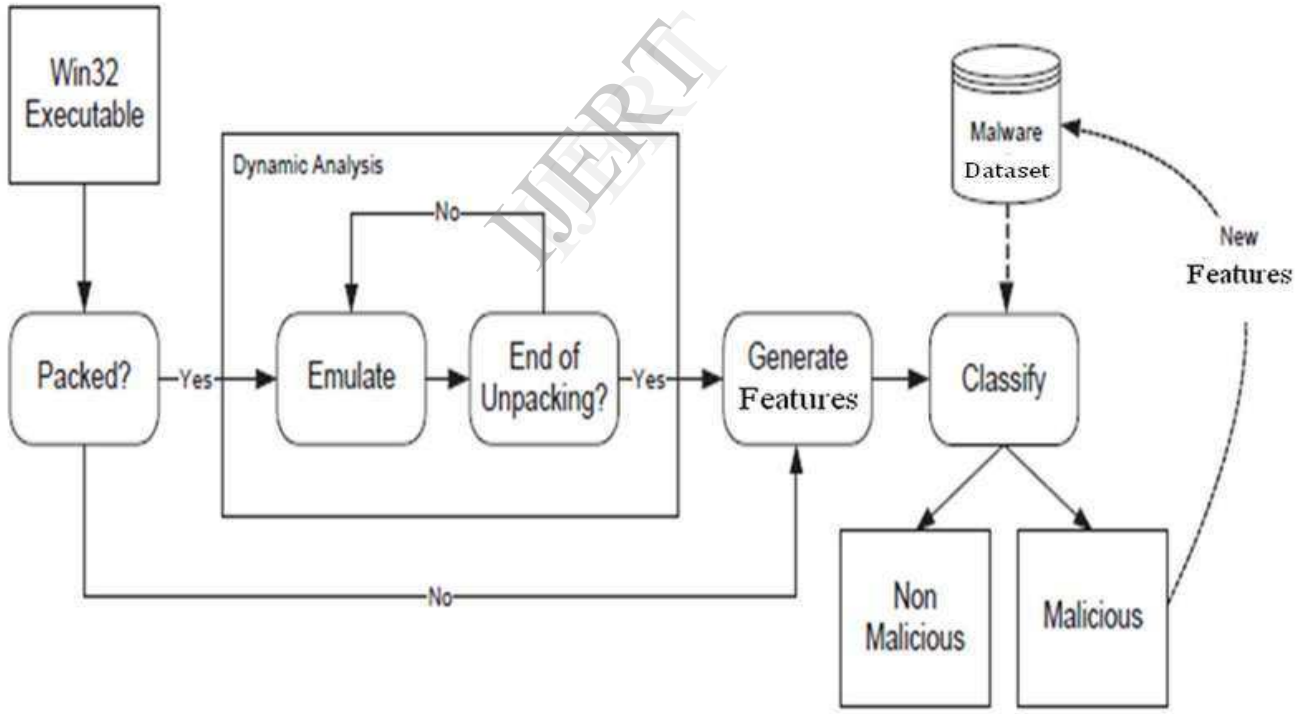

Fig. 1. Block diagram of the malware classification system

IV. SYSTEM DESIGN AND IMPLEMENTATION

A. Identifying Packed Binaries Using Entropy Analysis

Malwise performs an initial analysis on the input binary to determine if it has undergone a code packing transformation. Entropy analysis [8], is used to identify packed binaries. The entropy of a block of data describes the amount of information it contains. It is calculated as follows:

$$H(x) = -\sum_{i=1}^{N} \begin{cases} \mathrm{p(i)}\, log_2\, p(i), & \mathrm{p(i)} \neq 0 \\ 0, & \mathrm{p(i)} = 0 \end{cases}$$

Where *p(i)* is the probability of the *ith* unit of information in event *x*'s sequence of *N* symbols. For malware packing analysis, the unit of information is a byte value, *N* is 256, and an event is a block of data from the malware. Compressed and encrypted data have relatively high entropy. If the binary is identified as being packed, then the dynamic analysis to perform automated unpacking proceeds. If the binary is not packed, then the static analysis commences immediately.

B. Application Level Emulation

Automated unpacking requires malware execution to be simulated so that the malware may reveal its hidden code. The hidden code once revealed is then extracted from the process image. Application level emulation provides an alternate approach to whole system emulation for automated unpacking. Application level emulation simulates the instruction set architecture and system call interface.

C. Complexity Analysis

We assume a search complexity is $O(\log(N))$ for both global and local flowgraph databases. The runtime complexity of malware classification is on average $O(N\log(M))$ where M is the number of control flow graphs in the database, and N is the number of control flow graphs in the input binary. N is proportional to the input binary size and not more than several hundred in most cases. The worst case can be expected to have a runtime complexity of $O(N\log(M) + AN\log(N))$, where A is the number of similar malware to the input binary.

V. EVALUATION

Our method does not allow classifiers to use examples in training that are variants of viruses present in the test set. Our results show that our system, which uses family non-specific features, performs very well, while existing techniques for detecting previously unseen viruses perform significantly more poorly under our evaluation method.

VI. CONCLUSION

In this paper we proposed different algorithms to unpack malware using application level emulation. To detect the completion of unpacking, we proposed and evaluated the use of entropy analysis. It was shown that our system can effectively identify variants of malware in samples of real malware. In future work we 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. We would also like to explore retrospective testing. Retrospective testing would involve using a set of older viruses in the training set and a set of more recent ones in the test set. Finally, it was demonstrated the efficiency of unpacking and malware classification warrants Malwise as suitable for potential applications including desktop and Internet gateway and Antivirus systems.

VII. AUTHOR'S BIOGRAPHY

Thilagavathi (csethilak@gmail.com) is a M.E, IInd Year student at Shri Andal Alagar College of Engineering, Mamandur(SAACE), her area of interests includes Networking, Database Management, Software Engineering and her Co-Author Mr.P.Elumalai (cse.elumalai@gmail.com) is a Assistant Professor at A.R Engineering College,Villupuram ,his research interests includes Network Security, Adoc Networks, Data Mining, Web Services and SOA and working as a AP/CSE for more than 4 years.

REFERENCES

1. E. Carrera and G. Erdelyi, "Digital genome mapping–advanced binary malware analysis," in *Virus Bulletin Conference*, 2004, pp. 187-197.
2. I. Briones and A. Gomez, "Graphs, Entropy and Grid Computing: Automatic Comparison of Malware," in *Virus Bulletin Conference*, 2008, pp. 1-12.
3. J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 470-478.
4. M. E. Karim, A. Walenstein, A. Lakhotia, and L. Parida, "Malware phylogeny generation using permutations of code, "*Journal in Computer Virology*, vol. 1, pp. 13-23, 2005.
5. M. Gheorghescu, "An automated virus classification system," in *Virus Bulletin Conference*, 2005, pp. 294-300.
6. M. G. Kang, P. Poosankam, and H. Yin, "Renovo: A hidden code extractor for packed executables," in *Workshop on Recurring Malcode*, 2007, pp. 46-53.
7. P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee, "Polyunpack: Automating the hidden-code extraction of unpack-executing malware," in *Computer Security Applications Conference*, 2006, pp. 289-300.
8. R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," *IEEE Security and Privacy*, vol. 5, p. 40, 2007.
9. T. Graf, "Generic unpacking: How to handle modified or unknown PE compression engines," presented at the Virus Bulletin Conference, 2005.
10. Y. Ye, D. Wang, T. Li, and D. Ye, "IMDS: intelligent malware detection system," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007.