INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

An Approach to Malware Detection Using Error Back Propagation Networks

Krish Godiawala
Final Year Student, Department of Computer Engineering
K. J. Somaiya College of Engineering, Mumbai, India
Devashish Vohra
Final Year Student, Department of Computer Engineering
K. J. Somaiya College of Engineering, Mumbai, India
Karan Mirani
Final Year Student, Department of Computer Engineering
K. J. Somaiya College of Engineering, Mumbai, India
Nimish Lakhani
Final Year Student, Department of Computer Engineering
K. J. Somaiya College of Engineering, Mumbai, India

Abstract:
With new malware being created every day the onus is on the researchers to identify their unique signatures to detect them. This puts systems to risk against these unknown malware for a considerable amount of time. Also with the advent of polymorphic and metamorphic viruses the job of the researchers is even more arduous. Thus in this paper we propose the extraction of application programming interface (API) calls from malware subcategories. Also as each malware has its own infection mechanism API calls differ. We propose the use of Neural networks for classifying executables based on their relevant API calls.

1. Introduction

The proliferation of malware in recent years has presented a serious threat to the computer system. Malware which is also known as malicious software, malicious code or malcode enters the system without the permission of the user. It contains the code for performing illegal activities in the system, affecting the integrity and functionalities of the file and causing damage to the system. Malware is a big threat in the day to day computing world. New malware variants are discovered at an alarmingly high rate, some malware families featuring tens of thousands of currently known variants. The need for security is in fact a response to the increasing number of attacks led against information systems.

There two main techniques used for malware detection, namely, signature-based detection and anomaly-based (behavioral) detection [1]. Existing antivirus (AV) products provide detection techniques which are based on signatures that have been collected from previous seen viruses and then added to an AV database [2]. These products match the stored signatures to the files which are seen as suspicious and then detect whether it is infected by virus or not. Thus, the new viruses will not be detected by traditional detection systems unless this new virus is received by the antivirus company and the virus signature is stored in its own database. Signature based detection suffers from two hindrances; first, high false positive (identifying benign files as malware) and second, high false negative (fail to detect malware). Therefore, in this research we focus on anomaly based detection using feature extraction such as application programming interface (API) calls and code obfuscation features [3]. They do not rely on a database of signatures, but instead concentrate on the behavior of the system.

Our principle approach behind this is to observe the normal behavior of the file, and any deviation from it will be classified as an intrusion which will be further tested by the neural network to find out whether the file is infected by a virus or not. The success of these techniques depends on what features should be learnt in the training phase to discriminate malware and benign accurately. We have used an analysis tool IDA pro debugger to disassemble the binary file to extract the features of the PE executable based on their API calls [4].

2. Background

In Windows, the Win32 API is a collection of services provided by helper DLLs that reside in user space, while the native APIs are services provided by the kernel. It is very important to understand the API calling system and its features in order to trace its behavior and predict any changes in it [5].

2.1. Windows Application Programming Interface system (API)

The core of the Windows O/S consists of the application programming interface (API). The user applications in the Windows OS are based on this API calling functions. These functions are stored in dynamic link libraries (DLLs) in order to gain access to the system resources, the registry and the processes. A Win32 API call is normally called from a process running at the user level, then the called API will be handled by the system and converted to its equivalent function, known as a native API call, which will be understood by the kernel of the OS. As the API calls can go in deep levels in the system, so the analysis of it might lead to understanding the behavior of the file. This can help the intruder to gain the access of the system and exploit the power of windows. Hence, malware users make use of API calls to perform malicious actions. The applications use these native API calls to provide the requested service. Our approach extracts these patterns and the system files used by the application to know its general behavior. Different features related to the calls like create and modifying some files, extracting information from the files and changing the system values can lead us to a malware being attached to this process [6]. Then we perform statistical analysis of the behavior known to classify it into any malware class.

2.2. Malware Types

Malware which is also termed as malicious software enters system without the permission of user of the system. The words 'malicious' and 'software' are merged to create the term Malware. The various types are:

- Virus: A computer virus is a program, a block of executable code, which attaches itself to another program in order to reproduce itself without the knowledge of User. It needs a host program to cause harm. There are different types of computer viruses for example boot sector viruses, parasitic viruses, polymorphic viruses and metamorphic viruses.
- Worms: A computer worm does not require any host program and replicates itself by executing its own code independent of any other program. In general, virus attempt to spread through programs/files on single computer while worm spread throughout a network aiming to infect other connected computers. Well-known examples of worms are Code Red.
- Trojan horse: A Trojan horse is a software program that appears to be very useful. It performs the desirable function for the users as stated but secretly performs some unauthorized actions like stealing information or harm the integrity of the system. Once installed in the victim’s computer, it secretly enables the attacker to access all or part of victim’s computer resources.
- Spyware: Spyware is a type of malware that can be installed on computers and which collects information about the users without their knowledge.
- Trapdoor or Backdoor: Trapdoor or Backdoor is a method of bypassing normal authentication procedure i.e. it allows unauthorized access to a system.
- Logic Bomb: A logic bomb or time bomb is a piece of code that detonates or sets off when specific condition is triggered. The condition may be a day, date, time, a particular 'if loop', time interval, or count.
- Rabbit: Rabbit is a malware that creates many instances of itself in order to exhaust the system resources.

3. Proposed Architecture

Here we explain the framework for the application. During implementation we make use of IDA Pro which is used to disassemble the Portable Executable (PE) files. We also use the Mysql Database and Matlab to run the Neural Network.

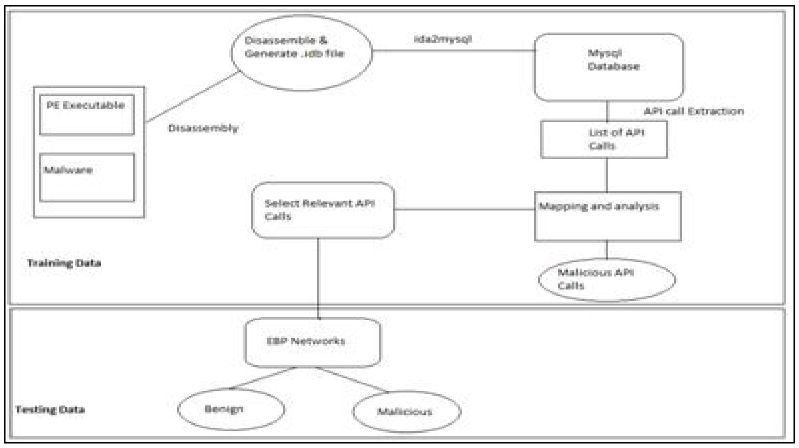

Figure 1: Architecture for API extraction and Training of Neural Network

3.1. Step 1: Disassemble the File

In this step we first disassemble the file using Interactive Dissembler (IDA) pro [10]. This creates a .idb file. We used a python script to store this idb file in the Mysql Database [7] [8]. The Mysql Database contains the following tables (Address comments, Address reference, Basic blocks, call graph, control_flow_graph, data, expression substitution, expression tree, functions, instructions, metainformation, modules, operand expressions, operand strings, operand tuple, and sections) for each executable. The function table contains list of each API called made by the application.

3.2. Step 2: Mapping and Analysis

We have compared the list of all API calls made by the application with the list of API calls given below. We have implemented our own logic to feed the above and create the target input set and target output set for the neural Network. We have written our own Java Program to implement the above functionality. [9]

Step	Virus Category	API Function Calls
First	Find to infect	Find First Stream, Find First File Transacted, Find First Stream Transacted W, Find First Stream W, Find Close, Find Next File, Find First Name, Find First File Ex, Find First File, Find First Name W, Find Next File Name, Find Next File Name W, Find First Name Transacted W, Find Next Stream W, Find Next Stream.
Second	Get information	Get File Attributes Ex, Get File Attributes Transacted, Get File Attributes, Get File Information By Handle, Get File Bandwidth Reservation, Get Compressed File Size Transacted, Get File Information By Handle Ex, Get Compressed File Size, Get Binary Type, Get File Size Ex, Get File Size, Get File Type, Get Temp File Name, Get Temp path, Get Final Path Name By Handle, Get Long Path Name Transacted Get Full Path Name Transacted, Get Full Path Name; Get Long Path Name, Get Short Path Name.
Third	Read and/or copy	Read File, Read File W, Open File, Open File Byld, Reopen File, Create Hard Link Transacted, Create Hard Link, Create Symbolic Link Transacted, Create Symbolic Link, Copy File Ex , Copy File, Create File W, Create File, Copy File Transacted, Create File Transacted.
Fourth	Write and/or delete	Replace File, Write File, Delete File Transacted, Delete File W, Delete File, Close Handle.
Fifth	Set information	Set File Information By Handle, Set File Valid Data, Set File Bandwidth Reservation, Set File Short Name, Set File Attributes Transacted, Set File Apis To OEM, Set File Attributes, Set File Apis To ANSI

Table 1: API calls to monitor

3.3. Step 3: Error Back Propagation Network

To introduce the learning component we have used an Error Back Propagation (EBP) neural Network. The target input set and target output set were given to the network while training and the network was tested using an unknown binary. This neural network is implemented using a Matlab script. Another network that could have been used was Self Organizing Feature Mapping (SOFM), however the number of neurons required while constructing a multilayer perceptron network is much lesser than SOFM [11]. This is the most important factor that needs to be considered for training and response time of the neural network which seems to be the best using a perception network.

4. Conclusion

In this the behavior of a program is extracted using API calls. These were matched up to potentially threatening API calls as referenced and fed to a Neural Network as training data. Finally an unknown binary was tested on this same apparatus.

5. Acknowledgement

This research was conducted at K.J. Somaiya College of Engineering (KJSCE) at Vidyavihar, Mumbai. We would like to thank our project guide, Prof. Nirmala Shinde, the project committee as well as our Departmental head for their support throughout the project. We would like to give special thanks to Prof. Swati Mali for taking an interest in out project.

6. References

1. Xiao-bin Wang, Guang-yuan Yang and Yi-chao Li, "Review on the application of Artificial Intelligence in Antivirus Detection System", University of Electronic Science and Technology of China, CIS 2008.
2. VX Heavens, http://vx.netlux.org
3. Windows API Functions, MSDN, http://msdn.microsoft.com/
4. Hex-Rays SA, IDA Pro, http://www.hex- rays.com/idapro/, 2008.
5. API Monitor. http://www.rohitab.com/apimonitor, 2011.
6. Chandrasekar Ravi and R Manoharan, "Malware Detection using Windows Api Sequence and Machine Learning", Volume 43– No.17, April 2012.
7. Idapython, http:// code.google.com/p/idapython/, 2009.
8. ZynamicsBinNavi, http://www.zynamics.com/binnavi.
9. Sulaiman Al amro and Antonio Cau, "Behavioural API based Virus Analysis and Detection", Vol. 10, No. 5, 2012.
10. IDA Pro Disassemble .Data Rescue, “An Advanced Interactive Multiprocessor Disassembler,” http://www.datarescue.com, 2011.
11. AnastasiaDoumas, Konstantinos Mavroudakis, DimitrisGritzalis and SokratisKatsikas, "Design of a neural network for recognition and classification of computer viruses", Computers and security, vol.14, No 5