A Framework for the Static Analysis of Malware focusing on Signal Processing Techniques

Sascha Zeisberger and Barry Irwin
Department of Computer Science and Information Systems
Rhodes University, P.O. Box 94, Grahamstown 6140
Tel: +27 46 6038291, Fax: +27 46 6361915
email: g07z3446@campus.ru.ac.za, b.irwin@ru.ac.za

Abstract- The information gathered through conventional static analysis of malicious binaries has become increasingly limited. This is due to the rate at which new malware is being created as well as the increasingly complex methods employed to obfuscating these binaries. This paper discusses the development of a framework to analyse malware using signal processing techniques, the initial iteration of which focuses on common audio processing techniques such as Fourier transforms. The aim of this research is to identify characteristics of malware and the encryption methods used to obfuscate malware. This is achieved through the analysis of their binary structure, potentially providing an additional metric for autonomously fingerprinting malware.

Index Terms— Malware Analysis, Static Analysis, Frameworks

I. INTRODUCTION

Malware is a type of software that performs some malicious activity on a system, often deploying itself on a system without a user's consent. Malware varies in functionality and intent, often infiltrating user data or providing unauthorised access to a system. The theoretical rate at which malware may spread is increasing, as demonstrated by the flash worm [1]. It is becoming difficult to combat malware due to an increasing complexity in their structure [2].

Current anti-malware solutions require input from manual sources, requiring security specialists to analyse a specific sample of malware before being able to effectively combat it. This overhead results in the demand for automated solutions, which is being partially fulfilled but, due to the deterministic behaviour of these applications, malware developers are able to circumvent these measures [3].

This research presents a framework that is used to test alternative metrics with which to detect malware. Section II briefly discusses malware analysis characteristics and approaches. Section III discusses the implementation of the framework that demonstrates the usage of three components to visual a malware sample, and includes an initial experiment to measure the effectiveness of the components used to facilitate this approach to malware analysis.

II. MALWARE ANALYSIS

Malware analysis focuses on understanding the behaviours and characteristics of malware in order to effectively combat it [4]. Malware comes in a variety of forms, including viruses, trojans and worms, all with the goal to facilitate access to restricted information [4]. Blacklist-based applications, such as Anti-Virus, often require a new sample of malware to be reported and analysed, followed by its addition to a blacklist and then distributing the updated blacklist when combatting new malware threats [5]. As a result, it can be concluded that malware analysis is a critical process in combatting malware.

Malware analysis may be classified into two categories; Dynamic and Static analysis [5].

Dynamic analysis of malware is the process of executing a malware sample and monitoring its behaviour during runtime. This allows an analyst to compile specifications based on its interactions with the system [5]. While this is effective, it requires monitoring of host systems and is difficult to implement since it requires an analysis of runtime interactions [4].

Static analysis of malware is an analysis that is performed in a non-runtime environment, commonly undertaken with the use of un-packers and disassemblers to expose code fragments [4]. This technique is becoming less effective as a means of detecting malware due to the increase in the number of new malware variants being distributed [1]. For the purpose of this research and due to its simplicity in automation compared to dynamic analysis, only static analysis is used.

To further complicate analysis, malware creators attempt to mask functionality by encrypting malicious code [5]. Analysis tools are available that provide functionality that allow for the autonomous unpacking of encrypted malware, but there are exceptions where manual intervention is required [5].

III. PROPOSED SOLUTION

A simple framework has been developed to facilitate the identification of malware, focusing on signal processing techniques. The goal of the framework is to evaluate new detection techniques and create metrics with which to identify malware.

A. Framework Structure

The framework analyses binaries at the byte level, interpreting the data as an audio signal. Thereafter the data is passed through two main modules; data manipulation and data visualisation.

A Discrete Fourier Transform (DFT) takes data from a time-domain, similar to audio data, and identifies specific frequencies in that data [6, 7]. While it is computationally uneconomical to calculate DFTs, a computationally viable implementation, called the Fast Fourier Transform (FFT), is provided in the NumPy python library [8]. This processing

technique has been implemented in the framework and is used to convert the malware data into a pseudo frequency domain. The reason for using Fouriers is to attempt to bypass obfuscation by extracting the features of the compiled code, such as decryption methods included in encrypted code, working on the assumption that malware creators use similar methods of compiling and encrypting malware variants. The Fourier data is finally passed to the visualisation module.

The R statistical computing application [9] in conjunction with RPy2 [10] was used to autonomously generate the visualisations of the Fourier results.

Heat maps are matrix-based visualisations, where each coordinate within the heat map has a colour "temperature" based on a value within the associated matrix. This has the advantage of representing a dataset in a condensed, two dimensional space. Figure 1 shows a heat map comparison between two samples of malware, where the columns distinguish the different malware samples and rows distinguish where the sample was taken from, where the top row represent the top-right of the heat maps and the bottom row the bottom-right of the heat maps.

A two dimensional matrix is used as the input for the R heat map function. For this investigation, the imaginary part of the FFT function was ignored and the real data was organised into rows, each row containing 512 elements. The regions used in Figure 1 represents a small fraction of the heat maps and were chosen due to the patterns being immediately visible. In a set of 603 malware samples, a random sample of 10 binaries was selected for an analysis, and four of these have thus far displayed a similar pattern as seen in Figure 1.

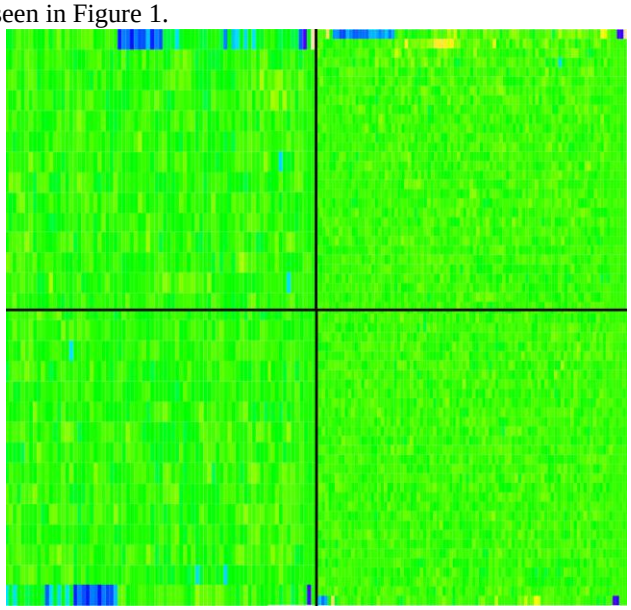

Figure 1 Heat map of a Fourier analysis performed on Malware

While promising, this is inconclusive of the FFT and heat map combination working for several reasons;

- The origins of the malware samples have yet to be determined. It may be a coincidence that a pattern occurred.
- The magnification of the sample heat maps had to be manipulated to provide a side-by-side analysis of the pattern. This means that while the pattern is similar, comparing the data on a one-to-one basis will yield no results.

The investigation would need to be conducted across the entire malware sample compared to a sample of legitimate binaries before any conclusion may be drawn.

B. Limitations

As this is an early implementation of the framework, there are several limitations. The current FFT functionality provided by NumPy in combination with the test system used, it is unable to handle binaries in excess of 3MB as this results in memory errors. In addition to this, RPy2's memory usage would increase with the size of the input Fourier data, eventually using the maximum available memory in the test system. This presented issues with larger datasets as the RPy2 library would eventually crash, citing memory issues.

As demonstrated in Figure 1, heat maps may not be the most effective method of visualising the Fourier data since the datasets are large. This adds difficulty in comparing two heat maps as the generated heat maps are too large for a side-by-side comparison.

IV. Conclusion

While the initial combination of Fouriers and heat maps has yet to provide any meaningful results, it has provided insight as to a possible design consideration for a final framework to evaluate signal processing techniques as an alternative form of malware analysis. It is concluded that the exploratory framework has sufficiently motivated for further research to be conducted as to the effectiveness of this malware analysis technique.

V. References

[1] (2012, June) Flame and Stuxnet makers 'co-operated' on code. [Online]. Available: http://www.bbc.com/news/technology-18393985

[2] (2012, June) AV-Test Malware Statistics. [Online]. Available: http://www.av-test.org/en/statistics/malware/

[3] U. Bayer, C. Kruegel, E. Kirda, "TTAnalyze: A Tool for Analyzing Malware", 2006.

[4] A. Honig and M. Sokorski, *Practical Malware Analysis*. William Pollock, 2012.

[5] S. Adai, B. Hartstein, M. Ligh, M. Richard, *Malware Analyst's Cookbook*. Wiley Publishing, 2011

[6] (2011, April) QtMultimedia in action: a spectrum analyser. [Online]. Available: http://labs.qt.nokia.com/2010/05/18/qtmultimedia-in-action-a-spectrum-analyser/ accessed

[7] (May, 2012) Mathematics of the Discrete Fourier Transform. [Online]. Available: https://ccrma.stanford.edu/~jos/mdft/DFT_Definition.html

[8] (2012, June) NumPy. [Online]. Available: http://numpy.scipy.org/

[9] (2012, May) The R Project for Statistical Computing. [Online]. Available: http://www.r-project.org/

[10] (2012, June) RPy2. [Online]. http://rpy.sourceforge.net/rpy2.html

Sascha Zeisberger received both his Computer Science undergraduate degree and his Honours Degree from Rhodes University in 2010 and 2011 respectively and is presently studying towards his Master of Science degree at the same institution. His research interests include Information Security and mobile development.