Chapter 7
Using Executable Slicing to Improve Rogue Software Detection Algorithms

Jan Durand
Louisiana Tech University, USA

Nicholas Kraft
University of Alabama, USA

Juan Flores
Louisiana Tech University, USA

Randy Smith
University of Alabama, USA

Travis Atkison
Louisiana Tech University, USA

ABSTRACT

This paper describes a research effort to use executable slicing as a pre-processing aid to improve the prediction performance of rogue software detection. The prediction technique used here is an information retrieval classifier known as cosine similarity that can be used to detect previously unknown, known or variances of known rogue software by applying the feature extraction technique of randomized projection. This paper provides direction in answering the question of is it possible to only use portions or subsets, known as slices, of an application to make a prediction on whether or not the software contents are rogue. This research extracts sections or slices from potentially rogue applications and uses these slices instead of the entire application to make a prediction. Results show promise when applying randomized projections to cosine similarity for the predictions, with as much as a 4% increase in prediction performance and a five-fold decrease in processing time when compared to using the entire application.

1. INTRODUCTION

With today's market globalization of software development and the proliferation of malicious attackers, it is becoming almost impossible to have any trust in the software that is loaded onto our systems. Rogue applications, or applications in which code has been added, modified or removed with the intent of causing harm or subverting a system's intended function (McGraw & Morrisett, 2000), are becoming more and more prevalent. To combat these infiltrations, consumers, as well

DOI: 10.4018/978-1-4666-2482-5.ch007

as corporations, are turning to anti-virus software products, which contain virus detection engines. Though very good at what they do, virus detection engines rely on a database of signatures to detect known rogue applications. Signature based systems inherently limit the detection of new and previously unknown types of rogue attacks. To that end there have been several research attempts to overcome these limitations. In one of these attempts (Atkison, 2009) we have shown the value of using randomized projection algorithms in detecting malicious applications.

The purpose of this paper is to provide methods and techniques to overcome the limitations inherent in the signature-based systems mentioned above. Through this research effort, we will provide a methodology for detecting rouge applications by enhancing the random projection, dimensionality reduction concept by using executable slicing. Executable slicing is a strategic method of compartmentalizing applications, and is used as a pre-processor to the algorithm. It will be shown that by adding this pre-processing step a significant gain in accuracy as well as in precision and recall can be achieved.

The following section provides a background description of previous methods that involve static analysis, information retrieval and randomized projection. In Section 3, the experimental design of this work is discussed including software and data used. In Section 4, results achieved are described. Finally, in Section 5 the conclusion and future directions are presented.

2. BACKGROUND

Developing effective potential solutions to the malicious software detection problem is an important direction in host security research. There have been few research papers, (Kang, Poosankam, & Yin, 2007; Perdisci, Lanzi, & Lee, 2008) are good examples, that pose the option of executable slicing while looking at malicious detection. Though their focus is directed toward packed executables, the focus of this paper is to show that statically analyzing sections or slices of an executable will improve prediction rates of non-packed, standalone executables. It is important to understand the methods and techniques that are used for these predictions. Since the randomized projection technique in this solution is used in conjunction with an information retrieval prediction algorithm we will include a small background on information retrieval as well as static analysis.

2.1. Static Analysis

Static analysis, sometimes referred to as static program analysis or static code analysis, is the examination of the source or object code of an application in order to identify patterns that indicate potential design errors and/or security threats (Food and Drug Administration [FDA], 2010). This analysis approach eliminates the need to execute an application in order to determine its behavior, contrary to its counter-part dynamic analysis, thus avoiding the potential compromise of the host system.

Static analysis has proven to be a very useful tool in detecting undesirable or vulnerable code in applications. There have been several research efforts such as (Bergeron, et al., 2001; Bergeron, Debbabi, Erhioui, & Ktari, 1999; Christodorescu & Jha, 2003; FDA, 2010; Jovanovic, Kruegel, & Kirda, 2006) that have incorporated the use of static analysis to detect malicious code in executable files.

Christodorescu et al. (2003) presented a static analysis framework for identifying malicious code patterns in executables and implemented SAFE, a static analyzer for executables. In their research, they show that SAFE is resilient to common obfuscation transformations on malicious code while three popular anti-virus scanners were susceptible to these attacks (Christodorescu & Jha, 2003).

Bergeron et al. (1999, 2001) present a three-step approach for detecting malicious code in applica-