Behavior Based Malware Analysis and Detection

DI Clemens Kolbitsch
Registration Number 0126605

The dissertation has been reviewed by:
Priv.-Doz. DI Dr. Christopher Krügel
and
Priv.-Doz. DI Dr. Engin Kirda

Malware is one of the most serious security threats on the Internet today. In fact, most Internet problems such as spam emails and denial of service attacks have malware as their underlying cause. That is, computers that are infected with malware are often networked together to form botnets, and many attacks are launched using these malicious, attacker-controlled networks.

With the increasing significance of malware in Internet attacks, much research has concentrated on developing techniques to mitigate malicious code. Unfortunately, current host-based detection approaches (i.e., anti-virus software) suffer from ineffective detection models. These models concentrate on the features of a specific malware instance, and are often easily evadable by obfuscation or polymorphism. Also, detectors that check for the presence of a sequence of system calls exhibited by a malware instance can be evaded by system call reordering. In order to address the shortcomings of ineffective models, several dynamic detection approaches have been proposed that aim to identify the behavior exhibited by a malware family. Although promising, these approaches are unfortunately too slow to be used as real-time detectors on the end host, and they often require cumbersome virtual machine technology.

In a first part of this thesis, we propose a novel malware detection approach that is both effective and efficient, and thus, can be used to replace or complement traditional anti-virus software at the end host. Our approach first analyzes a malware program in a controlled environment to build a model that characterizes its behavior. Such models describe the information flows between the system calls essential to the malware’s mission, and therefore, cannot be easily evadable by simple obfuscation or polymorphic techniques. Then, we extract the program slices responsible for such information flows. For detection, we execute these slices to match our models against the runtime behavior of an unknown program. Our experiments show that our approach can effectively detect running malicious code on an end user’s host with a small overhead.

Another important component in the fight against malicious software is the analysis of malware samples: Only if an analyst understands the behavior of a given sample, she can design appropriate countermeasures. Manual approaches are frequently used to analyze certain key algorithms, such as downloading of encoded updates, or generating new DNS domains for command and control purposes.

In a second part, we present a novel approach to automatically extract, from a given binary executable, the algorithm related to a certain activity of the sample. We isolate and extract these instructions and generate a so-called gadget, i.e., a stand-alone component that encapsulates a specific...
behavior. We make sure that a gadget can autonomously perform a specific task by including all relevant code and data into the gadget so that it can be executed in a self-contained fashion.

Gadgets are useful entities in analyzing malicious software: In particular, they are valuable for practitioners, as understanding a certain activity that is embedded in a binary sample (e.g., the update function) is still largely a manual and complex task. Our evaluation with several real-world samples demonstrates that our approach is versatile and useful in practice.

Both systems, our malware detection technique and HASTEN alike, heavily rely on dynamic analysis of a sample. However, the past has show that whenever an anti-malware solution becomes popular, malware authors promptly react and modify their programs to evade these defense mechanisms. For example, recently, malware authors have increasingly started to create malicious code that can evade dynamic analysis.

Thus, in a last part, we concentrate on evasion techniques that target these analysis systems. One recent form of evasion is stalling code. Stalling code is typically executed before any malicious behavior. The attacker’s aim is to delay the execution of such activity long enough so that an automated dynamic analysis system fails to extract the interesting behavior. This work presents the first approach to detect and mitigate malicious stalling code, and to ensure forward progress within the amount of time allocated for the analysis of a sample.

We built a prototype implementation, HASTEN, for our dynamic analysis systems ANUBIS. Experimental results show that our system works well in practice, and that it is able to detect additional malicious behavior in real-world malware samples.