LASTLINE WHITEPAPER

In-Depth Analysis of Malware
Abstract
Malware analysis is the process of determining the purpose and functionality of a given malware sample (such as a virus, worm, or Trojan horse). This process is a necessary step to be able to develop effective detection techniques for malicious code. In addition, it is an important prerequisite for the development of removal tools that can thoroughly delete malware from an infected machine. Traditionally, malware analysis has been a manual process that is tedious and time-intensive. Unfortunately, the number of samples that need to be analyzed by security vendors on a daily basis is constantly increasing. This clearly reveals the need for tools that automate and simplify parts of the analysis process.

In a well-known, pioneering scientific paper presented by the researchers Ulrich Bayer, Christopher Kruegel and Engin Kirda at the 15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR) in Hamburg, Germany, in April 2006, the authors presented TTAnalyze, a tool for dynamically analyzing the behavior of Windows executables. Two of co-authors of this paper, Dr. Kruegel and Dr. Kirda are co-founders of Lastline. In this whitepaper, we give an executive summary of that scientific work and use excerpts from the paper.

For a more detailed overview of the study, the reader is referred to the full paper: http://www.lastline.com/papers/eicar06_ttanalyze.pdf.

In TTAnalyze, the binary is run in an emulated operating system environment and its (security-relevant) actions are monitored. In particular, the authors record the Windows native system calls and Windows API functions that the program invokes. One important feature of the system is that it does not modify the program that it executes (e.g., through API call hooking or breakpoints), making it more difficult to detect by malicious code. Also, the tool runs binaries in an unmodified Windows environment, which leads to excellent emulation accuracy. These factors make TTAnalyze an ideal tool for quickly getting an understanding of the behavior of an unknown malware. Lastline has built novel, improved tools that are conceptually similar to TTAnalyze, but are more effective and superior in practice.

Introduction
Malware, which is a generic term to denote all kinds of unwanted software (e.g., viruses, worms, or Trojan horses), poses a major security threat to computer users. Unfortunately, the problem of malicious code is likely to grow in the future as malware writing is quickly turning into a profitable business. Malware authors can sell their creations to miscreants, who use the malicious code to compromise large numbers of machines that can then be abused as platforms to launch denial-of-service attacks or as spam relays. Another
indication of the significance of the problem is that even people without any special interest in computers are aware of current attacks such as Flame and Stuxnet. This is because security incidents affect millions of users and regularly make the headlines of mainstream news sources.

The most important line of defense against malicious code are virus scanners. These scanners typically rely on a database of descriptions, or signatures, that characterize known malware instances. Whenever an unknown malware sample is found in the wild, it is usually necessary to update the signature database accordingly so that the novel malware piece can be detected by the scan engine. To this end, it is of paramount importance to be able to quickly analyze an unknown malware sample and understand its behavior and effect on the system. In addition, the knowledge about the functionality of malware is important for removal. That is, to be able to cleanly remove a piece of malware from an infected machine, it is usually not enough to delete the binary itself. It is also necessary to remove the residues left behind by the malicious code (such as unwanted registry entries, services, or processes) and undo changes made to legitimate files. All these actions require a detailed understanding of the malicious code and its behavior.

The traditional approach to analyze the behavior of an unknown program is to execute the binary in a restricted environment and observe its actions. The restricted environment is often a debugger, used by a human analyst to step through the code in order to understand its functionality. Unfortunately, anti-virus companies receive up to two hundred thousand new malware samples each day. Clearly, the analysis of these malware samples cannot be performed completely manually. Hence, automated solutions are necessary.

One way to automate the analysis process is to execute the binary in a virtual machine or a simulated operating system environment. While the program is running, its interaction with the operating system (e.g., the native system calls or Windows API calls it invokes) can be recorded and later presented to an analyst. This approach relieves a human analyst from the tedious task of having to manually go through each single malware sample that is received. Of course, it might still be the case that human analysis is desirable after the automatic process. However, the initial results at least provides details about the program’s actions that then help to guide the analyst’s search.

Current approaches for automatic analysis suffer from a number of shortcomings. One problem is that malicious code is often equipped with detection routines that check for the presence of a virtual machine or a simulated OS environment. When such an environment is detected, the malware modifies its behavior and the analysis delivers incorrect results. Malware also checks for software (and even hardware) breakpoints to detect if the program is run in a debugger. This requires that the analysis environment is invisible to the malicious
code. Another problem is when the analysis environment does not monitor the complete interaction with the system. When this happens, the malicious code could evade analysis. This might be possible because there exist thousands of Windows API calls, often with arguments that are composed of complex data structures. Furthermore, the malicious code could also interact directly with the operating system via native system calls. Thus, the analysis environment has to be comprehensive and cover all aspects of the interaction of a program with its environment.

In their paper, Bayer, Kruegel and Kirda describe TTAnalyze, a tool that automates the process of analyzing malware to allow a human analyst to quickly get a basic understanding of the actions of an unknown executable. Running a binary under TTAnalyse results in the generation of a report that contains information to give the human analyst a very good impression about the purpose and the functionality of the analyzed sample. This report includes detailed data about modifications made to the Windows registry and to the file system, information about interactions with the Windows Service Manager and other processes, as well as a complete log of all generated network traffic.

**System Design**

TTAnalyse is a tool for analyzing Windows executables (more precisely, files conforming to the portable executable (PE) file format. To this end, the program under analysis is executed inside a PC emulation environment and relevant Windows API and native system calls are logged.

TTAnalyse uses Qemu, an open-source PC emulator written by Fabrice Bellard, as its emulator component. Qemu is a fast PC emulator that properly handles self-modifying code. To achieve high execution speed, Qemu employs an emulation technique called dynamic translation. Dynamic translation works in terms of basic blocks, where a basic block is a sequence of one or more instructions that ends with a jump instruction or an instruction modifying the static CPU state in a way that cannot be deduced at translation time. The idea is to first translate a basic block, then execute it, and finally translate the next basic block (if a translation of this block is not already available). The reason is that it is more efficient to translate several instructions at once rather than only a single one.

Of course, Qemu could not be used in the system without modification. First, it had to be transformed from a stand-alone executable into a Windows shared library (DLL), whose exported functions can be used by TTAnalyse. Second, Qemu’s translation process was modified such that a callback routine into the analysis framework is invoked before every basic block that is executed on the virtual processor. This allows the authors to tightly monitor the process under analysis.
Before a dynamic analysis run is performed, the modified PC emulator boots from a virtual hard disk, which has Windows XP (with Service Pack 2) installed. The lengthy Windows boot-process is avoided by starting Qemu from a snapshot file, which represents the state of the PC system after the operating system has started.

The analysis process is started by executing the (malware-)program in the emulated Windows environment and monitoring its actions. In particular, the analysis focuses on which operating system services are requested by the binary (i.e., which system calls are invoked). Every action that involves communication with the environment (e.g., accessing the file system, sending a packet over the network, or launching another program) requires a Windows user mode process to make use of an appropriate operating system service. There is no way for a process to directly interact with a physical device, which also includes physical memory. The reason for this stems from the design of modern operating systems, which prohibit direct hardware access so that multiple processes can run concurrently without interfering with each other. Thus, it is reasonable to monitor the system services that a process requests in order to analyze its behavior.

Conclusions
Because of the window of vulnerability that exists between the appearance of a new malware and the point where an appropriate signature is provided by anti-virus companies, every new malware poses a serious threat to computer systems. This whitepaper described TTAnalyze, a system to analyze the behavior of an unknown program by executing the code in an emulated environment. The goal of the analysis process is to gain a quick understanding of the actions performed by malicious code with the general aim of reducing the window of vulnerability. To this end, the tool records the invocation of security-relevant operating system functions (both Windows API functions and native kernel calls). Because the sample program is executed completely in software on a virtual processor, TTAnalyze can tightly monitor the process without requiring any modifications to its code. This allows the system to easily handle self-modifying code and code integrity checks, two features commonly observed in malware. Furthermore, the emulated system presents itself to running processes exactly like a real system. This makes it more difficult for malware to detect the analysis environment when comparing this solution to virtual machine or debugging environments. Finally, TTAnalyze uses a complete and unmodified version of Windows as the underlying operating system in which the unknown program is started. Thus, TTAnalyze provides a perfectly accurate environment for malicious code.

Lastline has built novel and improved tools that are conceptually similar to TTAnalyze, but that are much more efficient and effective in practice.
About Lastline

Lastline is a technology pioneer dedicated to stopping advanced malware, zero-day attacks, drive-by downloads and sophisticated Advanced Persistent Threats. Lastline’s flexible Previct platform provides high-resolution analysis and protection; the required network security foundational layer capable of providing exacting security legacy APT, IPS, AV and next generation firewalls simply cannot see. The Santa Barbara based company is dedicated to providing the most accurate malware detection and defense available to our customers.