Understanding behavioural detection of antivirus[1]

Authors:

Liang Soon Chai, MSc (Royal Holloway, 2015)

Lorenzo Cavallaro, ISG, Royal Holloway

Andrea Lanzi, Computer Science, University of Milan

Abstract

Antivirus products nowadays almost always come bundled with some form of proactive detection based on application behaviours. Many antivirus products in the market claim that they can detect and stop new and unknown threats with minimum use of traditional byte matching signatures. However, behavioural based detection is not the ultimate solution and can be circumvented.

There have been a few attempts in the past to evaluate the antivirus using disassembly techniques, forensic tools, live malware, and test applications that use techniques reverse-engineered from malware. It is tedious to analyse the disassembled codes of the various antivirus components and time consuming to use generic forensic tools to find the presence of antivirus hooks in the Windows operating system. Testing with live malware is dangerous and does not really help in understanding how the behavioural detection engines work. Testing the antivirus using techniques obtained from reverse-engineering of the malware is a lot of hard work.

In this project, we study the behavioural detection engines of the antivirus in both the 32-bit and 64-bit versions of Windows 7 by using specialised modules to identify the technologies that the antivirus are using to monitor application behaviours. Our analysis revealed that the 64-bit versions of the antivirus are generally weaker than their 32-bit versions. We also replayed the actions of both malware and legitimate applications from sandbox logs to the antivirus and was able to reveal what the antivirus are monitoring and what activities trigger a reaction without other interference factors like white-listing to reduce false-positive detections on legitimate applications and black-listing to use traditional byte-matching signatures to detect known malware.

Introduction

Most users rely on antivirus to protect their systems from malware or malicious software. For many years, antivirus are able to detect malware accurately using efficient signature matching algorithms. The situation changed when malware creators started to use a mixture of commercial and home-brewed packers to generate large amount of variants to overwhelm the antivirus vendors. Many of the samples were actually the same malware but using different packers. It became clear that it is inefficient to create different signatures for the same malware.

Antivirus vendors started to adopt behaviour based signatures to detect the different variants of the same malware. Regardless of how the malware change their appearance, they will have to perform

[1] This article is to be published online by Computer Weekly as part of the 2016 Royal Holloway information security thesis series. It is based on an MSc dissertation written as part of the MSc in Information Security at the ISG, Royal Holloway, University of London. The full MSc thesis is published on the ISG's website.

some activities in order to achieve malicious goals such as making themselves persistent in the infected systems or retrieving sensitive data from the infected machines. This enables the antivirus vendors to use behavioural signatures to detect the various kinds of malicious activities such as keystroke logging and communication with the command and control (C&C) servers of the malware.

Behavioural signatures are definitely more effective than the traditional byte matching signatures but they are also more prone to false positives as well. Many legitimate software in the past has been wrongly detected as malware just because they happen to exhibit some malware-like behaviours. Antivirus vendors have to use some forms of whitelisting technologies to help reduce the incidence of false positives. Many research institutions and companies setup free online sandbox services such as Anubis (http://anubis.iseclab.org) to allow the public to submit binaries and receive reports on the activities performed by the binaries. The idea is to gather the activity logs of applications that the public is using and then make use of them to come up with new and better algorithms to help improve the design of the behavioural based detection systems.

Behavioural based detection is not the ultimate solution and can be circumvented too. Malware creators are always trying to find new ways to achieve the same malicious goal. For example, there are many locations in the Windows registry that can be used by malware to gain persistency in the system. The malware will succeed if they managed to find one location in the Windows registry that the antivirus vendors are unaware of. Therefore, it is important to understand how the behavioural detection engines of the antivirus work and be able to determine their effectiveness against out-in-the-wild malware.

In this article, we investigate how the antivirus are monitoring the application behaviours and present a system that can make use of the activity logs collected by the Anubis sandbox to test the reactions of the antivirus products. We implemented a few modules to understand the technologies used for behavioural monitoring and an emulator to playback the actions found in the activity logs. We then evaluated them on real-world antivirus products. Our evaluation shows that it is possible to identify behaviours that will contribute to the antivirus detection without having to bother about issues related with white-listing of legitimate applications and black-listing of malware based on traditional byte matching signatures.

Technologies for Behavioural Monitoring

Any modern antivirus that provides decent real-time protection against unknown malware will have at least a system driver component that is able to access the Application Programming Interfaces (APIs) exported by the kernel modules and execute codes in kernel-land. The only user-land components are usually the graphics user interfaces and web browser protection modules. On systems prior to Windows Vista, antivirus vendors mainly hook the System Service Dispatch Table (SSDT) to monitor local activities (e.g. files, registry, processes, etc.) while network activities are monitored by hooking the Transport Driver Interface (TDI) driver and Network Driver Interface Specification (NDIS) driver. As Microsoft improved the capabilities of these APIs in Windows Vista and later versions of Windows, antivirus vendors are slowly shifting their hook-based monitors to use kernel notification callbacks and mini filter drivers. Antivirus vendors are also starting to use the Windows Filtering Platform (WFP) APIs to inspect network traffic.

System Service Dispatch Table. The System Service Dispatch Table (SSDT) contains information about two service dispatch tables in the Windows operating system which are used by the kernel's system service dispatcher to dispatch system calls. The first service dispatch table (NT SDT) contains

pointers to the core executive system calls implemented in the Windows NT kernel, ntoskrnl.exe while the second service dispatch table (Win32k SDT) contains pointers to the Windows USER and GDI system calls implemented in the Windows Subsystem, win32k.sys. Most of the antivirus are replacing entries in these two tables to intercept the system calls.

A module is created to check whether the antivirus products are hooking any of the functions from the two service dispatch tables. The module will extract clean copies of the service dispatch tables directly from the Windows NT kernel and Subsystem driver files and compare them against the live service dispatch tables to look for changes made by the antivirus products. The method to locate the clean NT service dispatch table is the same for both the 32-bit and 64-bit versions of Windows but the method to locate the clean Win32k service dispatch table is different. Validation of both the live dispatch tables is also different for the 32-bit and 64-bit versions of Windows.

Kernel Notification Callback. Kernel Notification Callbacks are actually API hooks introduced by Microsoft to solve the issue of antivirus hooking the SSDT. Antivirus can register callback routines with the Windows kernel using APIs to receive notifications whenever a new process or thread is created or destroyed and whenever a portable executable (PE) image is loaded into memory.

A module is created to check whether the antivirus products are installing any of the kernel notification callbacks that are used to monitor processes, threads, and loaded images. The Microsoft debugger, WinDbg (https://msdn.microsoft.com/en-us/library/windows/hardware/ff551063(v=vs.85).aspx), was used to disassemble the callback register APIs during investigation. The idea is to identify any symbols that can be used to locate the start of a data structure such as an array or list that is used by the Windows NT kernel to manage the callback routines. These symbols are not publicly exported by the Windows NT kernel and have to be obtained by adding an offset to some publicly exported symbols.

Filter Driver. Filter drivers are optional drivers that can extend the functionalities of a device or an existing driver. Antivirus typically employs filter drivers to validate changes made to the file system and registry, protect the keyboard and mouse device stacks, and filter network traffic.

A module has to be created to display the pre and post callback routines installed for filtering file and registry operations and the WFP callouts as there are no available tools. The DeviceTree tool by OSR Online (https://www.osronline.com/article.cfm?article=97) is used to search for any legacy file, keyboard, mouse, and TDI filter drivers while WinDbg has available commands to list the NDIS filter drivers. Minifilter drivers used to monitor file and registry operations are not visible in the DeviceTree tool. Although there are commands to view the list of minifilter drivers in WinDbg, there is no easy way to list the pre and post operation callback routines of each minifilter driver. WFP callout drivers like the minifilter and NDIS filter drivers are not visible in the DeviceTree tool. Unlike NDIS, there are no available commands in WinDbg to view the WFP callout drivers or WFP callouts.

Emulation with Anubis Dataset

The emulator is composed of two components which together provide the capability to playback the system calls to the antivirus in a controlled manner. The first component is a kernel-land driver whose main purpose is to intercept and redirect all file and registry modifications to a virtual storage area for selected user processes. The second component is a user-land application in charge of parsing logs created by the Anubis sandbox and replaying the system calls. Together, the emulator allows the antivirus to analyse the system calls before it intercepts and blocks all file and registry modifications to the Windows operating system.

The emulator currently only supports a subset of the Win32 APIs and system calls that appear in the Anubis logs and those that are monitored by the antivirus. Network APIs are also not supported at the point of writing. Due to the nature of some of Win32 APIs and system calls, not every Win32 APIs and system calls can be fully emulated. The Win32 APIs and system calls are dynamically loaded during runtime rather than statically linked during the compilation process to avoid static detection by the antivirus.

Evaluation and Results

Setup. Trial versions of five antivirus products were installed in VMware virtual machines running on 32-bit and 64-bit versions of Windows 7 Service Pack 1 and updated to their latest versions in July 2015:

- AVG Internet Security 2015
- Bitdefender Total Security 2015
- Kaspersky Internet Security 2015
- Norton Security 2015
- TrendMicro Maximum Security 2015

The antivirus products were first analysed using the modules we developed and some well-known tools to understand what technologies they are using to monitor the activities of the applications. The antivirus were then subjected to playback of Win32 APIs and system calls using our emulator and the activity logs taken from the Anubis dataset which consists of behaviours exhibited by a set of benign applications and a set of malware.

Experiment 1. Our first experiment involved using a combination of our modules and well-known tools to look for modifications made by the antivirus to the Windows operating system in order to implement their behaviour monitors.

In the 32-bit user-land, AVG and Bitdefender are observed to hook 32-bit processes while Kaspersky only hooks its own processes. Norton and Trend Micro are not using any forms of hooks to monitor application behaviours. In the 64-bit user-land, AVG and Bitdefender are hooking both the 32-bit and 64-bit processes while Kaspersky and Norton hook 32-bit processes running in the 64-bit Windows operating system. Trend Micro does not use any forms of hooks.

Antivirus	**32-bit**	**64-bit**	**Description**
AVG	Y	Y	Found only a few hooks in *kernel32.dll* and *ntdll.dll*
Bitdefender	Y	Y	Extensive amount of hooks found in *advapi32.dll*, *kernel32.dll*, *ntdll.dll*, *user32.dll*, and *ws2_32.dll*
Kaspersky	Y	Y	Hooks found in its own processes and also in 32-bit processes (mainly *ntdll.dll*) running in the 64-bit Windows operating system
Norton	N	Y	Hooks found in 32-bit processes (mainly *ntdll.dll*) running in the 64-bit Windows operating system
Trend Micro	N	N	-

Table 1 Summary of antivirus behavioural technologies used in Windows 7 (User-land)

In the kernel-land, all the five antivirus products are using SSDT hooks in the 32-bit Windows operating system while only Kaspersky is hooking the Win32k SDT in the 64-bit Windows operating system. All the five antivirus are observed to implement minifilter drivers for monitoring file and registry operations. Kaspersky is the only antivirus that installs keyboard and mouse filter drivers.

Norton and Trend Micro have switched to using WFP callouts for monitoring network activities while the rest of the antivirus are still using the older technologies such as TDI and NDIS filter drivers.

Antivirus	SSDT		Kernel Notification Callback	Filter Driver				
	32-bit	64-bit		File & Registry	Keyboard & Mouse	TDI	NDIS	WFP
AVG	Y	N	Y	Y	N	Y	Y	Y
Bitdefender	Y	N	Y	Y	N	N	Y	Y
Kaspersky	Y	Win32k SDT	Y	Y	Y	Y	Y	N
Norton	Y	N	Y	Y	N	N	N	Y
Trend Micro	Y	N	Y	Y	N	N	N	Y

Table 2 Summary of antivirus behavioural technologies used in Windows 7 (Kernel-land)

Experiment 2. In our second experiment, a total of 20 logs from the set of benign applications and 20 logs from the set of malware are randomly chosen and replayed using the 32-bit version of the playback application with the antivirus running in the 32-bit and 64-bit Windows operating system, followed by the 64-bit version of the playback application with the antivirus running in the 64-bit Windows operating system.

The idea is not to determine which antivirus product is superior but to understand what kind of behaviours (e.g. single or a sequence of actions) will trigger an alert. From the detection results, we are able to compile a list of behaviours that are considered malicious by the antivirus. Some of these behaviours are shown in Table 3. The symbol X in the table means the antivirus thinks the action is bad while the symbol X* means the antivirus thinks the action is bad but result is inconsistent in the 32-bit Windows operation system and the 64-bit Windows operating system. For example, the action of opening the file "C:\autoexec.bat" by the 32-bit application is detected by the antivirus in 32-bit version of Windows 7, but the same action by the same 32-bit application is not detected in the 64-bit version of Windows 7. The detection rates of the antivirus in the 64-bit Windows operating system are generally lower than the detection rates of the antivirus in the 32-bit Windows operating system.

Action(s)	32-bit	64-bit
Opening the file "C:\autoexec.bat"	X*	X
Copying itself to the "C:\Windows\" folder	X	
Copying itself to "C:\Windows\svchost.exe"	X	X
Writes a driver file to the "C:\Windows\System32\drivers" folder	X*	X
Opening a service using Service Control Manager (SCM)	X	
Copies itself or writes out a PE file to some location, creates a service via SCM pointing to that location, and starts the service using SCM	X*	
Creates the registry key HKLM\Software\Classes\CLSID\{4B1C1060-F0EB-0539-3C7E-3C28618388C1}\LocalServer32	X	
Creates the registry key HKLM\Software\Classes\CLSID\{6CBBC508-8B9A-11D5-EBA1-F78EEEEEE983}\InprocServer32	X	

Creates a key in the registry branch HKLM\System\CurrentControlSet\Services	X	
Writing into registry key HKLM\Software\Microsoft\Windows\CurrentVersion\Run	X	
Installing a hook procedure for monitoring keystroke messages via SetWindowsHookEx	X*	

Table 3 Examples of behaviours that trigger alerts from antivirus

Conclusions

In this article, we inspected real-world antivirus and discovered how they are monitoring the application behaviours in both the 32-bit and 64-bit versions of the Windows 7 operating systems. Modules were developed to check the SSDT, kernel notification callbacks, file and registry callbacks, and WFP callouts as there are no readily available tools or scripts. An emulator was also developed to replay the Win32 APIs and system calls found in the Anubis activity logs in the presence of the antivirus. This is done in order to simulate the behaviours of the sandboxed application to the antivirus without actually executing the application itself. Our results show that it is possible to identify behaviours which can be a Win32 API, a system call, or a sequence of Win32 APIs and/or system calls that are being flagged by the antivirus.

Biographies

Liang Soon Chai graduated in 2015 from Royal Holloway University of London with an MSc in Information Security. He is currently working as a security software engineer for Centre for Strategic Infocomm Technologies (CSIT), carrying out research into malware defence technologies and effectiveness of security software.

Lorenzo Cavallaro is a Senior Lecturer of Information Security in the Information Security Group (ISG) at Royal Holloway University of London. His research focuses largely on systems security. He has founded and is leading the recently-established Systems Security Research Lab (S2Lab) within the ISG, which focuses on devising novel techniques to protect systems from a broad range of threats, with the ultimate aim of building practical tools and providing security services to the community at large.

Andrea Lanzi is an assistant professor of Computer Science in the LaSER at University of Milan. His research mainly deals with host intrusion detection systems (HIDS), memory errors, reverse engineering, malware and forensic analysis. In recent years, he mainly studied the application of emulation/virtualization and compiler techniques for malware analysis and detection and worked on analysing large-scale security datasets to investigate the behaviour of current cyber threats.