A study of anti-virus' response to unknown threats

Christophe Devine <devine(@t)bob.cat>

Nicolas Richaud <nicolas.richaud(@t)lab.b-care.net>

Abstract

This study presents the evaluation of twelve anti-virus products with regards to programs not known from the signature files that show different kinds of malicious behavior. In practical terms, a set of twenty-one tests implementing various actions were developed; they cover key-logging, injection of code into other processes, network evasion, rootkit-like behaviour and exploitation of software vulnerabilities. The test programs were then run against each anti-virus program, and results were collected and consolidated. It was shown that all products tested here show deficiencies in at least one area, and some in all areas. For example, eleven anti-virus programs out of twelve still do not detect one code injection technique, which has been known for more than five years. Programs that spy on the user, such as recording the microphone, are not detected at all. Finally, this study provides recommendations to anti-virus vendors to enhance the capabilities of their products to detect malware, and improve safeguards against known attack techniques.

Introduction

Detection of malicious programs has traditionnaly relied on signature-based analysis. This method has the advantage of providing, in most cases, precise identification of the threat and relieves the user from the burden of making an informed decision. However, signatures may prove inadequate in several situations:

- When a new malware is being released in the wild, a small window of time exists between the first infections and the release of updated signature files; the number of computers that become infected will then be correlated to propagation speed of the malware [1].
- Targeted attacks exploiting "0-day" vulnerabilities that launch custom malicious code will thwart signature-based analysis. Although not widespread, a few targeted attacks have been identified in the past (for example, see [2]).
- Detecting a full range of known malware programs is a complex problem, as anti-virus are constrained by CPU resources; a perfect detection rate is not feasible [3]. Furthermore, users expect to be able to perform tasks without being hindered by their anti-virus program.

A new trend which has recently emerged is black-box detection of malware activity based on their behaviour, as exemplified by in the works of [4] and [5]. This method has the advantage of being able to detect malware in a more proactive fashion, at the cost of generating an higher number of false positives.

Rather than focusing on theoritical aspects of behavioural detection, this study concentrates on single test cases, each showing one particular method for performing a malicious action. Most surveys of anti-virus products only tested their signature-based detection engines; nevertheless, a few studies similar to this one do exist (such as [6]).

Methodology

Selection of anti-virus programs to be tested

The choice of products to be tested was based on their popularity, in order to cover the largest installed based as possible. Furthermore, time constraints would not have allowed testing the full range of all available anti-virus programs. Considering no recent and freely available study of anti-virus market share could be found, we relied on three denominators to make our decision:

- Download statistics for the Anti-virus section of the Softpedia website [7],
- Google number of results for the query “download [name of anti-virus X]”,
- The anti-virus vendor had to provide a free evaluation version of his product.

An initial list of thirty-eight products was retrieved from Virustotal [8]. This list was then narrowed down to twelve products chosen for subsequent testing, and are shown as follows:

Product name	**Version tested**
avast! professional edition	4.8.1296
AVG Internet Security	8.0.200
Avira Premium Security Suite	8.2.0.252
BitDefender Total Security 2009	12.0.11.2
ESET Smart Security (NOD32)	3.0.672.0
F-Secure Internet Security 2009	9.00 build 149
Kaspersky Anti-Virus For Windows Workstations	6.0.3.837
McAfee Total Protection 2009	13.0.218
Norton 360 Version 2.0	2.5.0.5
Panda Internet Security 2009	14.00.00
Sophos Anti-Virus & Client Firewall	7.6.2
Trend Micro Internet Security Pro	17.0.1305

Table 1: List of evaluated anti-virus programs

A Windows XP operating system (english version) with SP3 integrated was installed inside a VMware virtual machine. Two accounts were created, one with administrative rights (named “localadmin”), and another without (“localuser”). No additional patches or configuration changes were applied. A snapshot of the virtual machine state was then made, which served as the install base as well as the control subject.

Then, each anti-virus was installed as a leaf of the snapshot made previously, and fully updated to the latest version of the signatures. After this step, access to the internet was removed by switching the network adapter from “NAT” to “Host-only” mode, to ensure the tests could be reproduced identically for all anti-virus programs. It is important to note the anti-virus programs

were left in their default configuration. In a few cases, the user was asked about the type of network he was connected to; we always chose the most restrictive setting ("public network", "internet", etc.).

The installation phase was conducted between the 10th and 12th of December 2008.

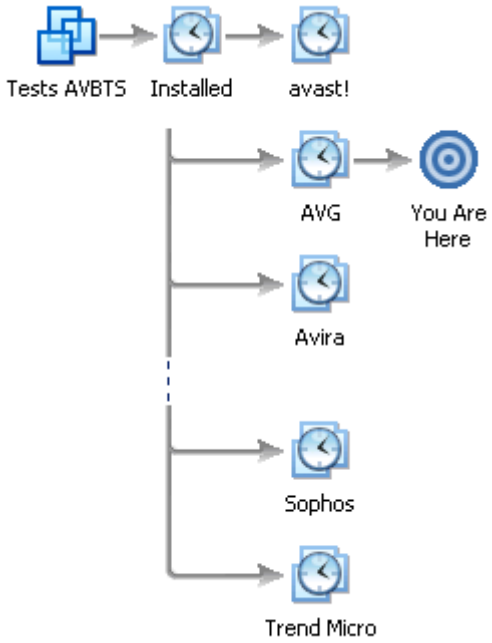

Figure 1: VMware snapshot tree

Selection of the tests to be performed

Tests to be run were selected as being able to represent a wide range of malicious behaviors that may be found "in the wild". For this purpose, research articles documenting specific malware were consulted (notably [9] and [10]), as well as "hacking" tutorials available on the Internet.

Identified malicious behaviors were implemented as series of single tests. Each test only contained the strictest number of operations required for the action to be completed successfully (for example, capturing keystrokes). After a test was run, the virtual machine was reset to the current snapshot, to prevent unwanted interaction between tests.

Three checks were added in each test program to prevent accidental execution outside the virtual machine:

- A warning message box is shown and allows cancelling the operation,
- The current computer name is checked against the expected computer name,
- The address of the Interrupt Descriptor Table is checked to verify the program is running inside VMware [11].

Some tests did require administrative privileges and are shown with [A] in from of them. Tests, which have been run from a non-privileged user account, are shown with [U].

The testing phase was conducted between the 14th and 19th of December 2008. Final tests were performed between the 7th and 9th of January 2009.

Limits of this study

- The tests only cover the evaluation versions of aforementioned anti-virus products and may generate different results from the paying versions.
- This study focused on HIPS-like (on-the-fly) detection of malicious behavior. Henceforth, it may not be relevant to the scanning capabilities of said products.
- This set of tests is limited and does not cover typical methods for malware to become persistent across reboots (such as the adding or modification of registry keys).
- Each test program was run for a limited amount of time (typically, one minute).

Test results

Keyloggers

This series of tests includes six keylogging techniques, three of which can be run from user space and do not require administration privileges. Others require the loading of a kernel driver, and were originally developed by Thomas Sabono [12] for the purpose of testing anti-rootkit programs.

- [U] testA01: The GetRawInputData() API was introduced in Windows XP to access input devices at a low level, mainly for DirectX-enabled games. This function was documented in 2008 on the Firewall Leak Tester [6] web site.
- [U] testA02 installs a WH_KEYBOARD_LL windows hook to capture all keyboard events (contrary to the WH_KEYBOARD hook, it does not inject a DLL into other processes).
- [U] testA03: The GetAsyncKeyState() API allows querying the state of the keyboard asynchronously.
- [A] testA11 hooks the keyboard driver's IRJ_MJ_READ function.
- [A] testA12 hooks the keyboard driver's Interrupt Service Request.
- [A] testA13 installs a "chained" device driver which places itself between the keyboard driver and upper level input device drivers.

The tests were run for one minute, during which keys were entered. The output of each sample was then checked.

Product name	**testA01**	**testA02**	**testA03**	**testA11**	**testA12**	**testA13**
avast!	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
AVG	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
Avira	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
BitDefender	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
ESET	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.

Product name	testA01	testA02	testA03	testA11	testA12	testA13
F-Secure	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
Kaspersky	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
McAfee	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
Norton	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
Panda	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.
Sophos	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	User alerted; keys logged.	User alerted; keys logged.	User alerted; keys logged.
Trend Micro	No alert; keys logged.	Program blocked; user alerted and prompted for action.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.	No alert; keys logged.

Table 2: Results of testing keyloggers

- Sophos warned the user about the loading of a kernel driver (rule “HIPS-/RegMod-013”), but did not prevent it from loading. It copied the driver in quarantine and recommended the user to send the sample to Sophos labs.
- Trend Micro detected and blocked the WH_KEYBOARD_LL hook. However the message shown was incorrect: it identified the threat at “Program Library Injection”.
- Kaspersky did not detect the loading of a malicious kernel driver, but warned the user four times when the program DbgView was run to inspect the driver’s output.
- The default configuration of Kaspersky anti-virus leaves the rules for detecting windows hooks and keyloggers unchecked. When both rules are enabled, Kaspesky detects and blocks testA02 and testA03.

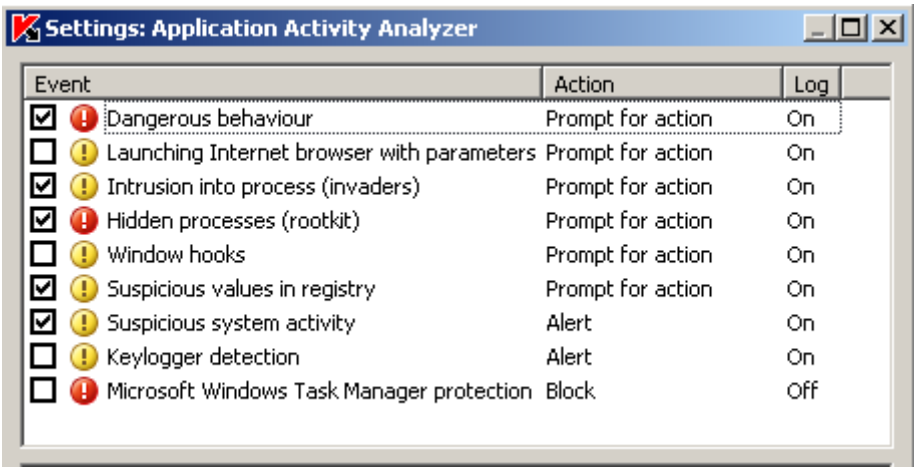

Figure 2: Kaspersky's "Proactive Defense" default configuration screen

Code injection and network access

This series of tests stresses the capabilities of anti-virus programs to prevent the unauthorized hijacking of a process by another, as well as attempts to access the network (for example, to upload gathered information or send spam).

- [A] testA21 installs a service running with SYSTEM privileges. It can operate as a server, listening on incoming connections on port 12345 and offering the client a CMD shell. It may also operate in the opposite direction, by initiating an outgoing connection.
- [U] testA22 starts Internet Explorer and attempts to inject its DLL into the target process using the QueueUserAPC() API. Then an outgoing connection on port 8080 is initiated; if successful, a CMD shell is attached.
- [A] testA23 is a passive network monitor; it captures HTTP traffic, using a RAW socket (this method does not require the loading of a separate driver).
- [U] testA31 tries to inject its DLL into Notepad using the CreateRemoteThread() API.
- [U] testA32 tries to inject its DLL into interactive processes with a WH_KEYBOARD windows hook.

It should be noted that both testA22 and testA31 do not require the use of WriteProcessMemory(). Instead, the string "I32.dll" is searched inside the target executable, and the directory containing the DLL is added to the user's PATH environment variable.

Product name	**testA21 (bind shell)**	**testA21 (reverse connect)**	**testA22**	**testA23**	**testA31**	**testA32**
avast!	No alert; but incoming connection blocked.	No alert; shell connected successfully.	No alert; shell connected successfully.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.

Product name	**testA21 (bind shell)**	**testA21 (reverse connect)**	**testA22**	**testA23**	**testA31**	**testA32**
AVG	Incoming connection detected and blocked; user alerted and prompted for action.	Outgoing connection detected and blocked; user alerted and prompted for action.	No alert; shell connected successfully.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.
Avira	Listening socket blocked; user alerted and prompted for action.	Outgoing connection detected and blocked; user alerted and prompted for action.	No alert; shell connected successfully.	Access to raw sockets blocked; user alerted and prompted for action.	Program blocked; user alerted and prompted for action.	No alert; DLL injected.
BitDefender	Listening socket blocked; user alerted and prompted for action.	Outgoing connection detected and blocked; user alerted and prompted for action.	No alert; shell connected successfully.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.
ESET	No alert; but incoming connections blocked.	No alert; shell connected successfully.	No alert; shell connected successfully.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.
F-Secure	Listening socket blocked; user alerted and prompted for action.	No alert; shell connected successfully.	No alert; shell connected successfully.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.
Kaspersky	No alert; but incoming connection blocked.	CMD shell execution blocked; user alerted and prompted for action.	CMD shell execution blocked; user alerted and prompted for action.	No alert; packets captured.	Program blocked; user alerted and prompted for action.	No alert; DLL injected.
McAfee	Listening socket blocked; user alerted and prompted for action.	No alert; shell connected successfully.	No alert; shell connected successfully.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.

Product name	testA21 (bind shell)	testA21 (reverse connect)	testA22	testA23	testA31	testA32
Norton	No alert; shell connected successfully.	No alert; shell connected successfully.	No alert; shell connected successfully.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.
Panda	Incoming connection detected and blocked; user alerted and prompted for action.	No alert; shell connected successfully.	No alert; shell connected successfully.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.
Sophos	Listening socket blocked; user alerted and prompted for action.	Outgoing connection detected and blocked; user alerted and prompted for action.	Access to network blocked; user alerted and prompted for action.	No alert; packets captured.	No alert; DLL injected.	No alert; DLL injected.
Trend Micro	Listening socket blocked; user alerted and prompted for action.	Outgoing connection detected and blocked; user alerted and prompted for action.	Attempt to execute Internet Explorer blocked; user alerted and prompted for action.	No alert; packets captured.	Program blocked; user alerted and prompted for action.	Program blocked; user alerted and prompted for action.

Table 3: Results of testing code injection and network access

- Surprisingly, Norton 360 allowed incoming connections on TCP port 12345, even though other ports such as SMB (TCP 139, 445) were blocked.
- Most firewalls could be bypassed by injecting a DLL with the QueueUserAPC() API. Nonetheless, Sophos detected a connection attempt was made from a DLL inside IEXPLORE.EXE, and Trend Micro directly blocked the launching of Internet Explorer.
- Once again, Kaspersky's default configuration prevented it from detecting the WH_KEYBOARD windows hook. It did detect, however, the redirection of the CMD shell's input/output handles and warned the user of a possible hacking attempt.
- Apart from Trend Micro, the WH_KEYBOARD hook was not detected. This allowed injecting a DLL in several programs, including the anti-virus' main GUI component.

User-mode and kernel-mode malicious activities

This section contains three tests covering various user-monitoring activities, developed at Thales in 2008 by Jean-Jamil Khalifé during his internship. It also features another set of techniques representative of classic rootkit behavior.

- [U] testA41 captures the contents of the clipboard repeatedly using the GetClipboardData() API. After one minute, the results of the capture are examined.
- [U] testA42 records surrounding sounds from the microphone present in the laptop used for the tests. For this purpose, a set of sound APIs are used (waveInAddBuffer, etc.). After recording for one minute, the resulting audio file is listened to.
- [U] testA43 captures the screen every three seconds using the BitBlt() API for one minute. The screenshots are then examined.
- [A] testA51 installs a simple backdoor by accessing \Device\PhysicalMemory (as described in [13]), and patches the SeAcessCheck() kernel function following [14]. After this is done, an attempt to terminate the spoolsv.exe service is done under a normal user account. This action is normally denied, but will be allowed if the backdoor functions properly.
- [A] testA52 opens \\.\PhysicalDrive0 and injects its code in the Master Boot Record (MBR). The new boot sector is largely based on eEye's BootRoot [15], but patches instead NTLDR's checksum verification code, then SeAccessCheck(). After rebooting, the same check (terminating spoolsv.exe under a non-privileged account) is done.
- [A] testA53 installs a kernel driver and hooks ZwQueryDirectoryFile() by modifying the System Service Dispatch Table (SSTD); the code is based on the implementation provided in [16]. It hides any file beginning with the word "AVBTS".

Product name	**testA41**	**testA42**	**testA43**	**testA51**	**testA52**	**testA53**
avast!	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
AVG	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
Avira	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	Detected as TR/Dropper.GEN	No alert; MBR modified; backdoor functional.	No alert; file hidden.

Product name	testA41	testA42	testA43	testA51	testA52	testA53
BitDefender	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
ESET	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
F-Secure	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
Kaspersky	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	Program blocked; user alerted and prompted for action.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
McAfee	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
Norton	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
Panda	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.
Sophos	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	User alerted; file hidden.
Trend Micro	No alert; clipboard contents captured.	No alert; microphone recorded.	No alert; screen captured.	No alert; RAM modified; backdoor functional.	No alert; MBR modified; backdoor functional.	No alert; file hidden.

Table 4: Results of testing user-mode and kernel-mode malicious activities

- Kaspersky blocked the attempt to access physical memory, whereas Avira detected this sample as “TR/Dropper.GEN” and blocked its execution.
- Similarly to kernel-mode keylogger tests, Sophos detected the loading of a kernel driver.
- All other tests were not detected.

Exploitation of vulnerabilities

Finally, this series of tests covers the exploitation of three relatively recent vulnerabilities.

- testA61 exploits the MS08-067 vulnerability, made public in October 2008 [17]. A buffer overflow in the Computer Browser service allows gaining full control over the target machine. Metasploit 3 [18] was used to perform the attack; for the purpose of this test, the firewall component of the anti-virus was disabled.
- [U] testA62 exploits the util.printf() buffer overflow vulnerability in Adobe’s Acrobat Reader, made public in November 2008 [19]. In this test, version 8.1.2 of Acrobat was exploited with a modified version of the PDF file posted on the milw0rm.com website.
- [U] testA63 exploits a stack overflow in VLC version lesser or equal to 0.9.4, made public in November 2008 [20]. Similarly, the malicious MPEG file was downloaded from milw0rm.

Product name	**testA61**	**testA62**	**testA63**
avast!	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.
AVG	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.
Avira	No alert; vulnerability exploitation successful.	Detected as HTML/Shellcode.Gen; user alerted and prompted for action.	No alert; vulnerability exploitation successful.
BitDefender	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.
ESET	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.
F-Secure	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.
Kaspersky	CMD shell execution blocked; user alerted and prompted for action.	No alert; vulnerability exploitation successful.	No alert; but exploit failed silently.

Product name	testA61	testA62	testA63
McAfee	Program blocked; user alerted and prompted for action.	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.
Norton	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.
Panda	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.
Sophos	User alerted; but vulnerability exploitation successful.	Detected as Troj/PDFJs-B and quarantined; user alerted.	No alert; vulnerability exploitation successful.
Trend Micro	No alert; vulnerability exploitation successful.	No alert; vulnerability exploitation successful.	Program blocked; user alerted and prompted for action.

Table 5: Results of testing the exploitation of vulnerabilities

- Kaspersky, McAfee and Sophos were able to detect the execution of Metasploit's windows/shell_bind_tcp shellcode, but reacted differently. Kaspersky and McAfee blocked the shellcode, whereas Sophos let it run.
- Avira and Sophos detected the presence of a malicious JavaScript, even though the JavaScript code in the original exploit was rewritten to avoid signature-based detection.
- Trend Micro warned the user about "Shell Modification" activity from within vlc.exe. This activity was flagged as having a low risk (this is the same generic warning as for testA31):

Figure 3: Trend Micro's warning after exploiting the VLC vulnerability

Conclusion and future work

One main disadvantage of our testing methodology was the requirement to perform all tests by hand. This made running the test suite against the panel of anti-virus programs very time-consuming. A possible evolution will be to run each test automatically using a predefined script; this poses the problem of detecting if the malicious action completed successfully, as well as detecting if the anti-virus picked up the threat.

It may be tempting to add an increasing number of tests in the future. Those may not be pertinent however, as malware authors will generally use the simplest method not detected by anti-virus programs. Why use advanced code injection techniques when a classic windows hook remains undetected? As such, this study hopes to raise the bar for malware authors, by encouraging companies that produce anti-malware products to take into account the different techniques presented in this study.

Adding new behavioural patterns will of course pose the problem of false positives; this may be mitigated using whitelisting, as well as providing users with correct and informative alert messages.

Finally, it is to be hoped the problems and lost revenue caused by malware will loose relevance as more secure computing architecture come forward, such as those base on sandboxed virtual machine (Java, Flash…) and more fine-grained access control. In this regard, the addition of UAC in Windows Vista, however flawed it may be [21], is a step in the right direction.

References

[1] Zesheng Chen, Chuanyi Ji: An Information-Theoretical View of Network-Aware Malware Attacks. CoRR abs/0805.0802: (2008)

[2] “The Microsoft Security Response Center (MSRC) : Update on Microsoft Excel Vulnerability”, as retrieved from http://blogs.technet.com/msrc/archive/2006/06/17/436860.aspx

[3] Shobha Venkataraman, Avrim Blum, Dawn Song: Limits of Learning-based Signature Generation with Adversaries. Proceedings of the 15th Annual Network and Distributed Systems Security Symposium (2008)

[4] Eric Filiol, Grégoire Jacob, Mickaël Le Liard: Evaluation methodology and theoretical model for antiviral behavioural detection strategies. Journal in Computer Virology 3(1): 23-37 (2007)

[5] Sébastien Josse: Rootkit detection from outside the Matrix. Journal in Computer Virology 3(2): 113-123 (2007)

[6] Guillaume Kaddouch: Firewall Leak Tester, http://www.firewallleaktester.com/

[7] http://www.softpedia.com/get/Antivirus/

[8] http://www.virustotal.com/sobre.html

[9] Heng Yin, Zhenkai Liang, Dawn Song: HookFinder: Identifying and Understanding Malware Hooking Behaviors. Proceedings of ISOC NDSS 2008.

[10] Jamie Butler and Kris Kendal: Blackout: What Really Happened. Black Hat USA 2008

[11] Joanna Rutkowska: Red Pill... or how to detect VMM using (almost) one CPU instruction, retrieved from http://www.invisiblethings.org/papers/redpill.html

[12] Thomas Sabono: La fiabilité des logiciels anti-rookits Windows 32 bits. SSTIC 2007

[13] “crazyload”: Playing with Windows /dev/(k)mem. Phrack 59 (2002)

[14] Greg Hoglund: A real NT Rootkit, patching the NT Kernel. Phrack 55 (1999)

[15] Derek Soeder and Ryan Permeh: eEye BootRoot. Black Hat USA 2005.

[16] Greg Hoglund, James Butler: Rootkits: Subverting the Windows Kernel. Addison Wesley, ISBN 0-321-29431-9 (2006)

[17] Vulnerability in Server Service Could Allow Remote Code Execution, as retrieved from http://www.microsoft.com/technet/security/Bulletin/MS08-067.mspx

[18] http://www.metasploit.com/framework/download/

[19] CVE-2008-1104: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-2992

[20] CVE-2008-4654: http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-4654

[21] Robert Paveza: User-Prompted Elevation of Unintended Code in Windows Vista, as retrieved from http://www.robpaveza.net/VistaUACExploit/UACExploitWhitepaper.pdf