SOFTWARE – PRACTICE AND EXPERIENCE
Softw. Pract. Exper. (2013)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.2197

Antivirus security: naked during updates

Byungho Min, Vijay Varadharajan*,†, Udaya Tupakula and Michael Hitchens

Information and Networked Systems Security Research, Faculty of Science, Macquarie University, Sydney, Australia

SUMMARY

The security of modern computer systems heavily depends on security tools, especially on antivirus software solutions. In the anti-malware research community, development of techniques for evading detection by antivirus software is an active research area. This has led to malware that can bypass or subvert antivirus software. The common strategies deployed include the use of obfuscated code and staged malware whose first instance (usually installer such as dropper and downloader) is not detected by the antivirus software. Increasingly, most of the modern malware are staged ones in order for them to be not detected by antivirus solutions at the early stage of intrusion. The installers then determine the method for further intrusion including antivirus bypassing techniques. Some malware target boot and/or shutdown time when antivirus software may be inactive so that they can perform their malicious activities. However, there can be another time frame where antivirus solutions may be inactive, namely, *during the time of update*. All antivirus software share a unique characteristic that they must be updated at a very high frequency to provide up-to-date protection of their system. In this paper, we suggest a novel attack vector that targets antivirus updates and show practical examples of how a system and antivirus software itself can be compromised during the update of antivirus software. Local privilege escalation using this vulnerability is also described. We have investigated this design vulnerability with several of the major antivirus software products such as Avira, AVG, McAfee, Microsoft, and Symantec and found that they are vulnerable to this new attack vector. The paper also discusses possible solutions that can be used to mitigate the attack in the existing versions of the antivirus software as well as in the future ones. Copyright © 2013 John Wiley & Sons, Ltd.

Received 1 November 2012; Revised 3 February 2013; Accepted 22 March 2013

KEY WORDS: security; antivirus; malware; vulnerability; code execution; local privilege escalation; denial of service

1. INTRODUCTION

Nowadays, the attackers face two main hurdles when they try to compromise a computer system as follows: (i) obtaining temporary control over a target system; and (ii) maintaining or extending the control so that a malware can achieve its goal, such as cyber espionage. The first hurdle is usually tackled with a client-side attack using known or zero-day exploits or social engineering such as phishing and offline deception. Traditionally, the first step of finding a zero-day vulnerability and bypassing operating system-level security mechanisms (such as data execution prevention and address space layout randomization) is considered to be harder than the second one. The second step, which involves installing and operating a malware on a system without being detected by security tools (such as antivirus, firewall, intrusion detection systems and intrusion prevention systems), is usually believed to be relatively easier, using well-known evasion techniques such as polymorphism and metamorphism [1–4].

*Correspondence to: Vijay Varadharajan, Information and Networked Systems Security Research, Faculty of Science, Macquarie University, Sydney, Australia.

†E-mail: vijay.varadharajan@mq.edu.au

However, both obstacles are getting harder for attackers to overcome, partly because security tools have improved incorporating behavioural engines to detect malicious control flow such as Dynamic Link Library (DLL) injection. In particular, the second step has become much harder than it used to be because antivirus software systems now use both signatures and behavioural characteristics to detect malware. Many antivirus software systems perform static and/or dynamic analysis and a few even incorporate simulation, emulation, or sandboxed execution environment to determine dynamically the maliciousness of a binary. Furthermore, efforts on detecting obfuscated malware are continuously improving [5–10]. Thus, it is becoming increasingly challenging for attackers to bypass the detection of security tools to achieve their goals. In fact, executing and installing a malware to permanently compromise a system has become harder than most social engineering attacks. The highest hurdle for the attackers is implementing a malware that is both functional and still undetectable.

To overcome these hurdles, attackers have resorted to staged malware. Only a dropper or a downloader is executed at the first compromise; then the malware checks the target environment and modifies its behaviour depending on the state of the environment (such as enabling malicious modules or using bypassing techniques). Hence, it has become very important for security tools to detect not only the first exploit/malware execution but also every step of a staged malware in order to counteract the threats. This requires security tools to be *always* active while the computer system is running. If the antivirus functionality stops for any reason, even if it is for a very short period, a staged malware can exploit this period to permanently compromise the system.

In this paper, we propose a novel attack vector that attackers can apply to their staged malware. All antivirus software needs to be updated frequently to protect their system with up-to-date definitions and engines. If the antivirus software is partly or totally deactivated during an update, then this is an important design vulnerability that can be exploited by the attackers. In our work, we have discovered that such deactivation happens during update with well-known antivirus software such as Avira Antivirus product lines, AVG product lines, McAfee Total Protection, Microsoft Security Essential (MSSE) and Symantec Norton Internet Security. We have found and exploited this design vulnerability in these five product lines to nullify their effectiveness on a target computer system. A practical and detailed example of how an antivirus solution can be compromised during the update is explained with Avira; we have chosen Avira as it was the hardest one to compromise (from the perspective of offensive technique) amongst the five products that we have analysed. We have also provided brief descriptions of the other four product lines as well. In short, a malware installer (the first stage of a staged malware) monitors or triggers the update of the antivirus software, waits for the system to be unarmed due to the design vulnerability and performs arbitrary actions that are normally detected by the antivirus software, including the subversion of the antivirus itself. We show that local privilege escalation from 'a user to system' is also possible using this vulnerability. In other words, a staged malware can perform anything that is required for further intrusion and permanent installation (e.g. privilege escalation and kernel-mode rootkit), as the probability of detection decreases to 0. After the update, the malware takes permanent control of the target system and is able to do anything that it wants to without being detected.

The rest of this paper is structured as follows. We introduce the concept of staged malware in Section 2. General information on antivirus software systems is given in Section 3. The design vulnerability we have found is illustrated in Section 4 with the list of vulnerable product lines. A full explanation of the vulnerability with the antivirus software product, Avira, is given in Section 5. We describe several real-world attack examples exploiting this vulnerability in Sections 6 and 7. Possible solutions to overcome this vulnerability are suggested in Section 8. Related works and conclusions are given in Sections 9 and 10, respectively.

2. STAGED MALWARE

Most of well-known modern malware are staged ones. Figure 1 shows the basic operation of a staged malware. They use a downloader and/or a dropper as their initial program that will be executed first on target systems. For instance, Zeus, SpyEye and Citadel have been most notorious botnet malware and all of them use dropper or installer or downloader [11, 12]. Most widely spread malware such as

DOI: 10.1002/spe

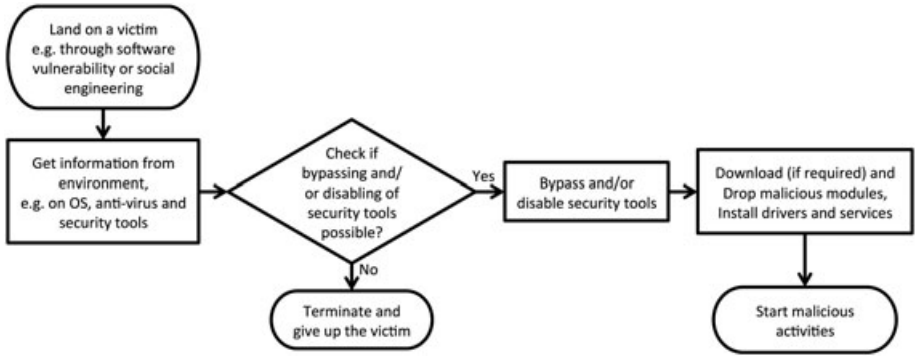

Figure 1. Staged malware.

Conficker (also known as Downadup), Ramnit and Sality, also use staged techniques to compromise target systems [13–15]; Sality, Ramnit and Conficker are still the most active and blocked malware in November 2011 [16]. Lastly, sophisticated and highly targeted malware such as Stuxnet, Duqu, Flame and Gauss, usually use more than two stages when compromising target systems [17–20].

A dropper/installer/downloader is executed (not installed) through the first attack vector such as browser zero-day payload or USB. This initial component normally checks the target environment for any antivirus software installed, version of OS and system configurations. Then, it determines if it can perform any malicious activities on the environment and if possible installs other components on the target system. If it concludes that the environment is not appropriate to perform any malicious activities, then it terminates itself. Without this stage, some or all of malware components and/or its activities would be detected by the antivirus software. Then information about the malware can be shared among antivirus corporations, and signatures for the malware can be created, resulting in the death of the malware. However, if the malware first checks the target environment and detects any antivirus software (say AV1) that detects its components or its behaviour, then it may choose to give up the installation and terminate its execution. However, this malware can still compromise other systems that do not have the antivirus software AV1 and will remain undetected for any length of time while performing its malicious activities. This checking procedure is a core part of modern malware, as all of the malware mentioned earlier use this type of strategy [11–15, 19, 20]. In the first stage, they essentially check the target environment to see that there exists any antivirus system and if it can be bypassed; then, they install some or all of the components of the malware on the target victim system or otherwise terminate the installation. In some cases, the dropper may install only a portion of its malware so as to make such malware components operate without being detected by the security tool installed on the victim system. This strategy can be thought as 'intrusion-in-depth' in the sense that even though a malware cannot perform its full functionality due to the limitation of the target environment, it can still perform other tasks such as install itself as a driver, communicate with C&C server and install rootkit module. In other words, the malware can continue intruding the target system even if some of its parts fail to function.

The staged approach at least gives two benefits to the attackers. First, it allows attackers to minimise the risk of being detected by security tools. Second, it minimises the exposed portion and hides the full functionality of the malware at the early stage of intrusion. Security researchers cannot reveal the functionalities of a malware when they can obtain only a dropper or downloader, because it might not include any meaningful functionality of the real malware.

As the malware research evolves, it is important for security tools to detect not only the first exploit or execution but also every level of the staged malware. One such effort of the security community is to make the security tools to be always active, as long as the system is running [21, 22]. If some or all of the functionalities (signature and behavioural-based detection, realtime protection and rootkit detection) fail to be active for any reason, even for a very short period, the security protection provided by the tool can be totally nullified, because the staged malware may detect such condition and exploit it. Actually, there are malwares that aim boot or shutdown time, for example, bootkit, so that it can compromise the system before the security tool begins or after the tool terminates [23–26].

DOI: 10.1002/spe

3. ANTIVIRUS SOLUTIONS

Almost all the users install at least one security tool on their system to protect it from compromise. The most widely used security tools on a commodity PC are antivirus and firewall solutions. Antivirus software is usually responsible for the detection, prevention and removal of malware on a single computer system. Also, most users install only one antivirus product on their system due to the degradation of system performance and the conflicts and incompatibilities between security tools [27]. In summary, the security of commodity PC systems heavily depends on the effectiveness of the antivirus software installed on them. This is one of the unique characteristics of antivirus software, which other purpose software do not have. The consequence is that if the antivirus software can be subverted, then the security of the system can collapse.

Another unique characteristic of antivirus software is the need for high frequency of updates. As many new malware appear on a daily basis, installing an antivirus software is not sufficient to block the attacks. The software must be updated on a regular basis; especially, signatures (also known as definitions) should be updated at least once a day.

For example, default configurations for major antivirus solutions are as follows: Avira commercial products check for an update every 2 h [28]; McAfee and Avast free products look for an update every 4 h [29,30]; Symantec products checks for an update every hour [31], and Symantec's certified definitions are released three times on a weekday and one during weekend [32]; Sophos Endpoint Security and Control is even configured to check for an update every 10 min [33]. In addition, antivirus tools update their engines and their programmes as well. Engine updates occur much more frequently than program updates, because an updated program is usually released annually as a new version. A typical antivirus update process can be depicted as shown in Figure 2.

On update, every antivirus software downloads and applies the latest signatures and engines. To do so, antivirus solutions usually create several new processes. Also, many antivirus software perform re-initialisation or restart some components in order to apply the latest updates. As updates occur frequently on a regular basis, any flaw related to updates may lead to a critical problem on the system. We will elaborate on this issue in Section 4.

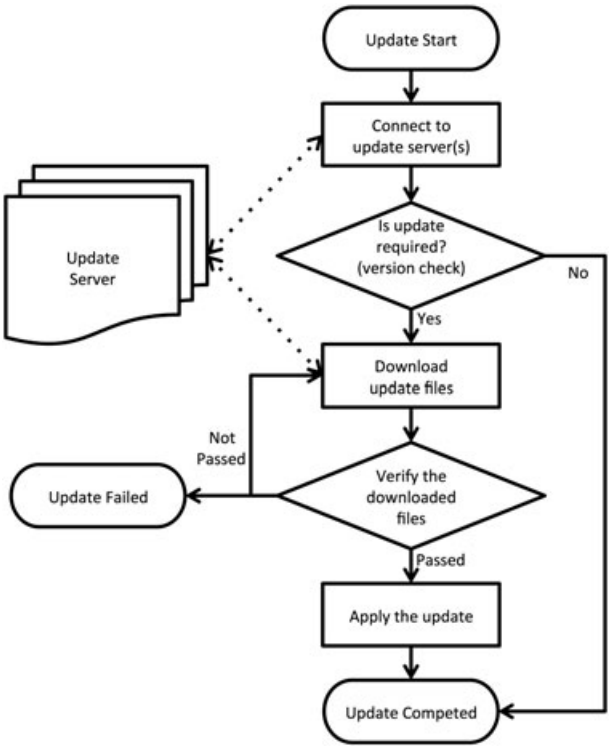

Figure 2. Typical update process of antivirus software.

DOI: 10.1002/spe

4. OBSERVATIONS ON VULNERABILITY OF ANTIVIRUS SOFTWARE UPDATE

In general, it is hard to bypass the detection and protection mechanisms of an antivirus software while it is working. Thus, most antivirus solutions start as early as possible during a system boot, minimising the period that the system is exposed to malware threats without any protection. For instance, Microsoft introduced Secure Boot [21] and Early Launch Anti-Malware [22] in an effort to minimise such time gaps. Also, most antivirus solutions including Avira, Avast, Symantec, Kaspersky, ESET, AVG and Trend Micro install one or more system drivers, as they are loaded prior to Windows services and user applications. Furthermore, it is even harder for malware to penetrate an antivirus software after the tool becomes active and starts monitoring. This is why some malware aims at boot or shutdown time so that they can compromise the system and/or the antivirus software before the software begins or after it terminates.

However, there can be another point in time during which an antivirus software may turn off its functionalities, namely, during the time of update. Updates of antivirus software happen frequently, and in most cases, users can also trigger or perform an update. Therefore, if there is any short period during the update when an antivirus solution becomes inactive, a malware can target this time frame and compromise the system and the antivirus software. In some cases, it is even possible for the malware to trigger or perform an update on behalf of a user and exploit this time period. This type of vulnerability provides the attacker with another way to leverage their attacks. This opportunity for attacks could turn out to be even better than targeting the boot or shutdown time. First, every malware that is designed to become active prior to the antivirus solution[‡] should be installed before it can be activated on a target system. As most modern antivirus solutions prevent these rootkits from being installed, bypassing antivirus products and installing such malware still remains as one of the major obstacles for the attackers. As described in Section 6.2, an update-related vulnerability can be used to install rootkits on the target system. Second, performing malicious activities at shutdown time can be hard if the malware is running as a user process, because almost all the antivirus solutions utilise kernel-mode driver to protect the system (and themselves) from compromising including unauthorised alteration. These kernel-mode drivers live longer than common user processes at shutdown time. Therefore, the malware should be installed as a kernel-mode driver or a service so that it can wait until the antivirus solution's drivers are unloaded on the target system to perform the malicious activities. Again, installing a driver or a service from unknown third parties is blocked by most antivirus software, causing this type of malware that is hard to implement. However, once installed, for instance, by partially bypassing some defences of the antivirus, the malware can perform normally prohibited activities at shutdown time after the antivirus is totally unloaded from the system although there may not be enough time to carry out all the functions. In contrast, much more time consuming activities can be performed if the antivirus itself is subverted during an update on a running system (not a shutting-down system). In addition, compared with boot and shutdown time, attackers can choose the attack time if the update of an antivirus software can be triggered by users. Furthermore, in some cases (usually with laptops or office computers or servers), updates may happen more frequently than reboot, which gives attackers more chances to compromise the target system.

Let us now look at how a malware can detect whether some or all modules of an antivirus software are inactive during an update. As described earlier, almost all the antivirus software use services and kernel-mode drivers, which are usually the core of the protection they provide. Therefore, a malware can monitor the status of the antivirus software modules. A single service or driver's temporary out-of-service can lead to a total compromise of the antivirus software and the system. We examined this vulnerability with the top antivirus software products. Note that the top 10 antivirus vendors have a share of something like 90% of the whole market [34], and the top 11 products account for 70% of worldwide market share (see Table I), although there are over 50 antivirus solutions available worldwide. As shown in Table II, five out of the 10 vendors' product lines turned out to be susceptible

[‡]This type of malware includes master boot record (MBR), BIOS, and Extensible Firmware Interface (EFI) rootkits.

DOI: 10.1002/spe

Table I. Worldwide antivirus vendor and product market share [34].

AV vendor	Market share (%)	AV product	Market share (%)
Avast	17.50	avast! Free Antivirus	13.80
Microsoft	13.90	Microsoft Security Essentials	13.60
Avira	12.10	Avira Free Antivirus	10.00
ESET	10.60	ESET NOD32 Antivirus	7.10
Symantec	10.20	AVG antivirus Free Edition	6.60
AVG	9.10	McAfee VirusScan	3.70
Kaspersky	6.20	ESET Smart Security	3.70
McAfee	4.50	Norton AntiVirus	3.50
Trend Micro	3.00	Kaspersky Internet Security	3.40
Panda	2.30	Norton Internet Security	3.20
other	10.60	AVG antivirus	2.50
		Other	29.00

AV, antivirus.

Table II. Tested antivirus solutions and their results.

Vendor	Product	Affected or not
Avast	avast! Free Antivirus 7	Seems not vulnerable
Microsoft	**Microsoft Security Essentials 4**	**Vulnerable**
ESET	ESET Smart Security 5	Seems not vulnerable
Symantec	**Symantec Norton Internet Security 2013**	**Vulnerable**
AVG	**AVG AntiVirus Free and Commercial 2012 and 2013**	**Vulnerable**
Avira	**Avira Antivirus Free, Premium and Internet Security 2012 and 2013**	**Vulnerable**
Kaspersky	Kaspersky Internet Security 2013	Seems not vulnerable
McAfee	**McAfee Total Protection 2013**	**Vulnerable**
Panda	Panda Internet Security 2013	Seems not vulnerable
Trend Micro	Trend Micro Maximum Security 2013	Seems not vulnerable

to this particular vulnerability as follows: Avira, AVG, McAfee, Microsoft and Symantec.[§] In some cases such as Avira, only one module became inactive; however, it was possible to unload other modules using this tiny crack, allowing a malware to compromise completely the system and the antivirus software. In other cases, such as AVG, McAfee, Microsoft and Symantec, no additional task was required to subvert the antivirus products and the system. Although the details on this vulnerability and its exploit differ among the five antivirus product lines, in this paper, we only describe the Avira's case because (i) the fundamental concept of this vulnerability is identical among all the vulnerable software; (ii) Avira was the only product among our five, which was selected as one of the five best antivirus solutions in 2011 [35]; (iii) Avira was the most difficult one to compromise; and (iv) a full explanation on Avira is sufficient to demonstrate this type of vulnerability. Attacks on the other four vulnerable antivirus software are briefly given in Section 7.

5. DETAILED EXPLANATION–VULNERABILITY WITH AVIRA

In order to understand the vulnerability and attack, it is necessary to have a basic understanding of the Avira antivirus software. Hence, in this section, first, we describe briefly the standing of Avira in the antivirus industry, and give a simplified overview of its architecture and update procedure, followed by a detailed explanation of Avira's vulnerability.

[§]Notice that other products are marked as 'seems not vulnerable'. This is because temporary unload of services and/or drivers may not happen with every update. As far as we have investigated, some updates cause the restart of services and the reload of kernel-mode drivers, and the ratio of such updates vary depending upon the antivirus solutions (from rare product updates to relatively frequent definition and engine updates). We have observed the behaviours of the antivirus solutions' updates for about a month, and the products marked as 'seems not vulnerable' did not restart or reload their modules.

DOI: 10.1002/spe

Table III. The number of common vulnerabilities and exposures entries for top antivirus vendors.

	Number of CVEs	
Vendor	1999–August 2012	2010–August 2012
Symantec	317	74
McAfee	118	48
Kaspersky	17	16
ESET	18	12
Trend Micro	14	12
AVG	18	8
Microsoft Security Essentials	6	6
Avira	17	5
Avast	13	2
Panda	24	1
Total	562	184

CVE, Common vulnerabilities and exposures.

Table IV. The number of common vulnerabilities and exposures entries for major software vendors.

	Number of CVEs	
Vendor	1999–August 2012	2010–August 2012
Apple	1841	752
Microsoft	2903	694
Oracle	1406	674
IBM	1403	481
SUN	1355	262
Total	8908	2863

CVE, Common vulnerabilities and exposures.

5.1. Avira antivirus

Avira is one of the largest antivirus vendors with nearly 10% worldwide market share [34] (See Table I). Avira's free antivirus solution was ranked in the third place in terms of worldwide market share in September 2012 [34]. It was also selected as one of the top five antivirus products in 2011 by AV-Comparatives, along with Bitdefender, ESET, F-Secure and Kaspersky [35]; it got the top score (ADV+) in on-demand, retrospective,[¶] and performance tests. Avira's excellent ability to detect zero-day attacks using the heuristic engine has also been shown in [36]; only nine out of 22 could detect a zero-day proof of concept that used a notorious Java vulnerability in 2012 (CVE-2012-4681).

Additionally, the number of CVE[‖] entries for security solutions support the fact that Avira is a highly reliable software. Table III summarises the number of CVEs for market leading security software vendors. Compared with Table IV, which lists the number of CVEs for major software companies, security software are more secure than operating systems and application software. Avira is ranked in the third place in terms of recent (last 3 years) CVE entries and appears in the fourth place in terms of all time CVE entries, which supports its reliability. In general, as the complexity of software increases, there is a greater probability for that software to have more vulnerabilities. Therefore, if one antivirus software is significantly simpler than another, then the former is likely to have lesser number of vulnerabilities compared with the latter. This is reflected in Table III where we see some similarities between groups of antivirus products such as one group comprising Kaspersky, ESET and Trend Micro and another consisting of MSSE, Avira and Avast.

[¶]Retrospective test shows how good the static/offline heuristic detection of an antivirus product [35].
[‖]Common vulnerabilities and exposures is a dictionary of publicly known information security vulnerabilities and exposures, launched in 1999 and maintained by MITRE corporation [37].

DOI: 10.1002/spe

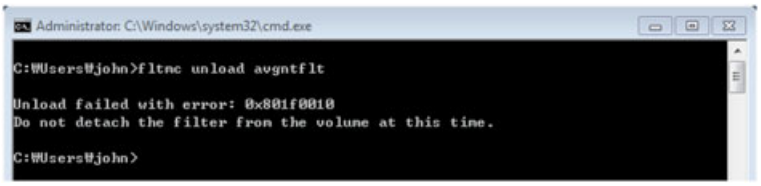

Figure 3. Driver protection: Avira's kernel-mode drivers cannot be unloaded.

5.2. Simplified architecture of Avira

Like many other major antivirus software that are currently available, Avira Antivirus Free software consists of user-level processes, Windows services and kernel-mode drivers.

- Windows services (NT Authority\SYSTEM)
  - Avira Realtime Protection service (avguard.exe)
  - Avira Scheduler service (sched.exe)
  - Avira Web Web Protection (avwebgrd.exe)
- User-mode processes
  - Avira System Tray Tool (avgnt.exe)
  - Avira Control Center (avcenter.exe)
- Kernel-mode drivers
  - Avira mini-filter driver (avgntflt.sys)
  - Avira Security Enhancement Driver (avipbb.sys)
  - Avira Manager Driver (avkmgr.sys)
  - Avira Snapshot Driver (ssmdrv.sys)

In addition, a few more processes such as update.exe, avnotify.exe and avwsc.exe are temporarily created during the update process.

Among these modules, Realtime Protection service is crucial in Avira's protection mechanism, as it provides realtime protection not only to the system (e.g. on-access detection of malware) but also to itself (self-protection such as prevention of unauthorised alteration on Avira-related files). In particular, unloading the kernel-mode drivers and the filter driver is blocked by Realtime Protection service as shown in Figure 3. Furthermore, a user cannot stop, pause or restart the service, as the service ignores such requests. Also, it is not possible to terminate or kill the processes associated with the service (blocked by the driver).[**]

Kernel-mode drivers further strengthen Avira's self-protection. One of them locks Avira-related registry keys so that they cannot be modified by users or malware (see Figure 4). Avira's program folder is protected by the filter driver so that even a user with administrator privilege cannot add any file and delete or modify its files (see Figure 5). Furthermore, the processes of Avira are protected by the kernel-driver in such a way that even the administrative user cannot kill them. Last but not least, Avira uses files in FAILSAFE folder if some of its files in its installation folder are corrupted. To sum up, the protection architecture of Avira is hard to defeat.

5.3. Update procedure of Avira

As mentioned earlier, every antivirus software must stay up-to-date with the latest signatures and engines. Avira is no exception. Normally, there are three ways for carrying out the updates. First, the update of the definitions occurs automatically on a daily basis. Next, a user can trigger an automatic update via a menu or a command line. And last, a user can download the latest definitions from the Internet and manually update Avira antivirus with the downloaded definitions.

[**]In contrast, some antivirus software such as Sophos End Point Security does not protect its services, thereby enabling a user or a malware to start/stop/restart the services. In this case, it is possible for a malware to stop the service, perform any malicious activity and then start the service. Although a notification pops up on the lower-right side of the screen, it can be easily ignored by the user because the period for which the service stops can be very short, such as 1 s.

DOI: 10.1002/spe

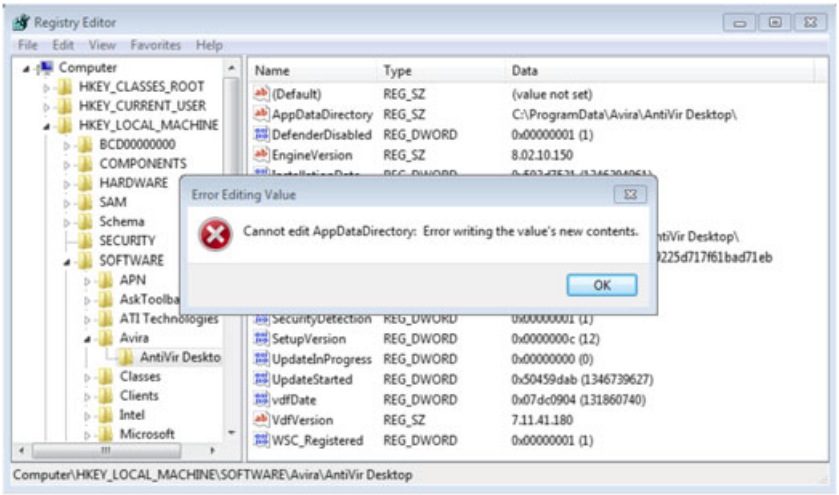

Figure 4. Registry protection: Avira's registry keys cannot be added/modified/deleted.

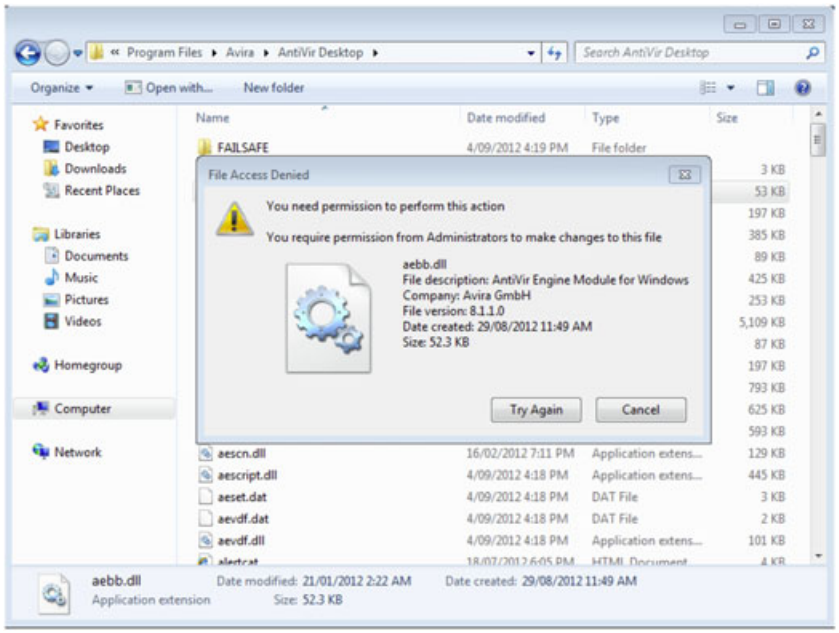

Figure 5. File protection: adding to a new file or deleting/modifying existing files in Avira's installation folder drivers is blocked.

Once an automatic update is started, Avira first checks the current definition and engine versions to determine whether an update is indeed required or not. If so, it downloads the latest definition and engine files from a dedicated server and checks them and then installs them. While updating, Avira keeps logging all relevant events so that a user or an administrator can infer possible reasons if the update fails. These log files are located in C:\ProgramData\Avira\AntiVir Desktop\LOGFILES by default. Also, the following registry keys represent update-related information including the last update time (in UNIX time) and the currently installed definition version

- HKLM\SOFTWARE\Avira\AntiVir Desktop\UpdateStarted
- HKLM\SOFTWARE\Avira\AntiVir Desktop\LastUpdate
- HKLM\SOFTWARE\Avira\AntiVir Desktop\vdfDate
- HKLM\SOFTWARE\Avira\AntiVir Desktop\VdfVersion

DOI: 10.1002/spe

- HKLM\SOFTWARE\Avira\AntiVir Desktop\UpdateInProgress

Once an update has started, the value of UpdateInProgress key becomes 1, and a few registry keys are temporarily created and then deleted during the update.

5.4. Vulnerability

During an update, Avira's Realtime Protection service has to be re-initialised. This is not an issue as the service remains active during the re-initialisation process. However, when some of its engines (of the type ae[DESCRIPTION].dll) are updated, the service must be restarted, not re-initialised. Figures 6 and 7 show the change of Realtime Protection service's PID during an update. The fact that the service has been terminated and restarted during an update can also be verified in the Windows system log or Avira's update log file. This restarting usually takes a few seconds on a commodity personal computer (about 7 s in Figure 8) and during this period, the system becomes almost 'naked' without any protection. Any known exploit can be executed without being detected. Zero-day exploits that are usually detected by the Avira's heuristic engine can be run. Furthermore, unloading Avira's filter driver becomes possible during this period, which is normally blocked by the Realtime Protection service. This in turn enables the attacker to replace any binary located in the installation folder with that of the attacker's. That is, Avira's Realtime Protection service binary or DLL files that the service will load can be replaced with malicious ones so that attacker's binary can be loaded and executed when the service restarts.

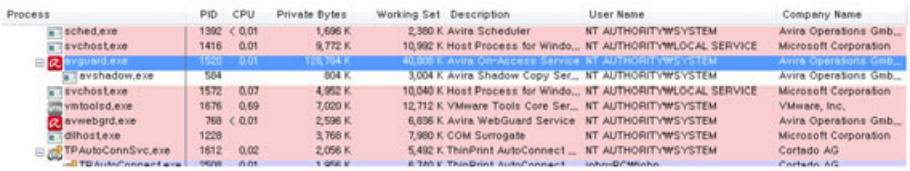

Figure 6. PID of Realtime Protection service (avguard.exe) is 1520 before an update.

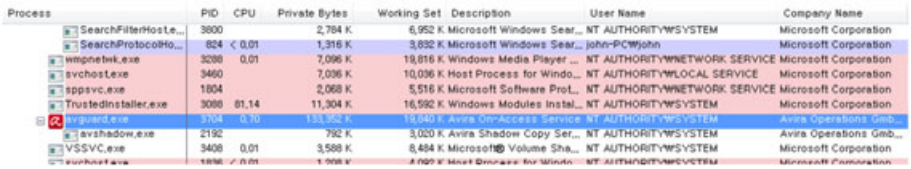

Figure 7. PID of Realtime Protection service (avguard.exe) is 3704 after the update.

Figure 8. Avira's update log file shows the fact that the service has been restarted.

DOI: 10.1002/spe

6. ATTACKS ON AVIRA

In this section, we describe three attack cases against Avira antivirus software exploiting the vulnerability described earlier in Section 5. Additional attack scenarios can also be envisaged.

We have implemented all these three attack cases. Our experimental environment consisted of a virtual machine with Windows 7 Ultimate. VMWare Fusion 5 was used as virtual machine monitor (VMM), and the Avira antivirus used was Avira Free Antivirus 2012.

6.1. Attack 1: file replacement to compromise Avira and the operating system

In this case, a typical real-world attack example can be summarised as follows: suppose that a user inserts an infected USB drive to his/her machine. A dropper is executed through the autorun feature of Windows. It is not detected by Avira software because the dropper is not performing any suspicious activity at this moment. From this point on, the dropper waits until an update happens, or the dropper itself can actively trigger an update on behalf of a user, as triggering an update is not detected by Avira antivirus. We have experimented with both these options. When an update occurs, the system became unarmed at least for a few seconds in the case of Avira due to the vulnerability. It is during this time that Avira's DLL file was automatically replaced by the dropper. Then, the replaced DLL (malware) was then automatically loaded by Avira, and the malware obtained SYSTEM privilege on the target system in the context of Avira.

Following is a detailed explanation on the exploitation of Avira's design vulnerability. Recall that a staged malware is used in this scenario.

1. An attacker attacks a victim system and succeeds to run a malware installer (first stage); any technique, such as social engineering and a client side attack, can be used. The attacker can even use any computer including computers in public kiosks as victims, for example, collecting information that other users have entered into public computers, building a botnet for DDoS or constructing an attack proxy system, which makes tracing the source of an attack hard to detect.
2. The malware installer performs only the actions that are permitted under any antivirus software's realtime protection. In this particular example, it checks the following:
   - OS version.
   - Privilege of current user.
   - Antivirus solution installed on the system.
   - Version of currently active signatures and engines.
3. With Avira installed on the target system, the installer triggers an update; alternatively, it may just wait for an update to be started by the Avira's scheduler service.
4. When the update begins, the installer monitors the status of Aviras Realtime Protection service.
5. Once the service is deactivated during the update, the installer performs the required actions that are normally blocked or prevented by Aviras Realtime Protection service. In this implementation example, the installer's ultimate goal was to replace Avira's sqlite3.dll with a malicious one (second stage) so as to subvert both Avira and the system. It performed the following tasks:
   (a) For privilege escalation, it dropped and executed any known or zero-day exploit that is normally detected by Avira. Notice that this local privilege escalation (from guest to SYSTEM) is required only once. After this file replacement process, the malware obtained SYSTEM privilege on the target machine.
   (b) Unloaded Avira's filter driver that is normally protected by the service.
   (c) Dropped the real payload (fabricated sqlite3.dll) and replaced the original file in Avira's installation folder with the malicious one. This file operation is shown in Figure 9.
6. The installer then deleted itself as a clean-up process to erase its existence; alternatively, the payload may delete the installer.
7. As the filter driver has been unloaded, it was then restored, even though Realtime Protection service automatically loads and attaches the filter driver when it restarts. The reason for the restoration is that the service's restart triggered by Avira after an update proceeded to

DOI: 10.1002/spe

Figure 9. Successful file operation on the installation folder.

some extent and then failed if the driver remains unloaded; of course, the installer can manually restart the service after the first restart by Avira fails. But still, the better solution is to restore the driver, because the service restart by Avira then succeeds. Interestingly enough, we observed that even if the start of the service was triggered and failed, it is logged as successfully started in the Avira's update log file, which is good from the attacker's point of view.

8. On restarting, Realtime Protection service loaded the malicious sqlite3.dll, which provides full SQLite functionalities, and became active without any problem. However, once loaded by the service, the malicious sqlite3.dll obtains SYSTEM privilege on the target machine. In other words, this attack allows the malware to escalate its privilege from a user to SYSTEM, which means user access control on Windows becomes ineffective. Also, almost any malicious activity was then possible, as it is loaded and executed in the context of Avira's Realtime Protection service. Furthermore, the DLL can perform file operations on the installation folder, even while the filter driver is loaded; this allowed the attacker to update the malicious DLL. The result of this file replacement and loading operations is shown in Figure 10. The original sqlite3.dll (sqlite3_ori.dll, 389KB) has been replaced with the malicious version (sqlite3.dll, 612 KB), and the fabricated DLL has been loaded by the service (window on the left side). Here, the original DLL was not removed to show its replacement. After becoming a part of Avira, the malware is able to modify Avira's memory area. Then, it is possible to make Avira look completely normal (with the tray icon's umbrella open) but totally ineffective.

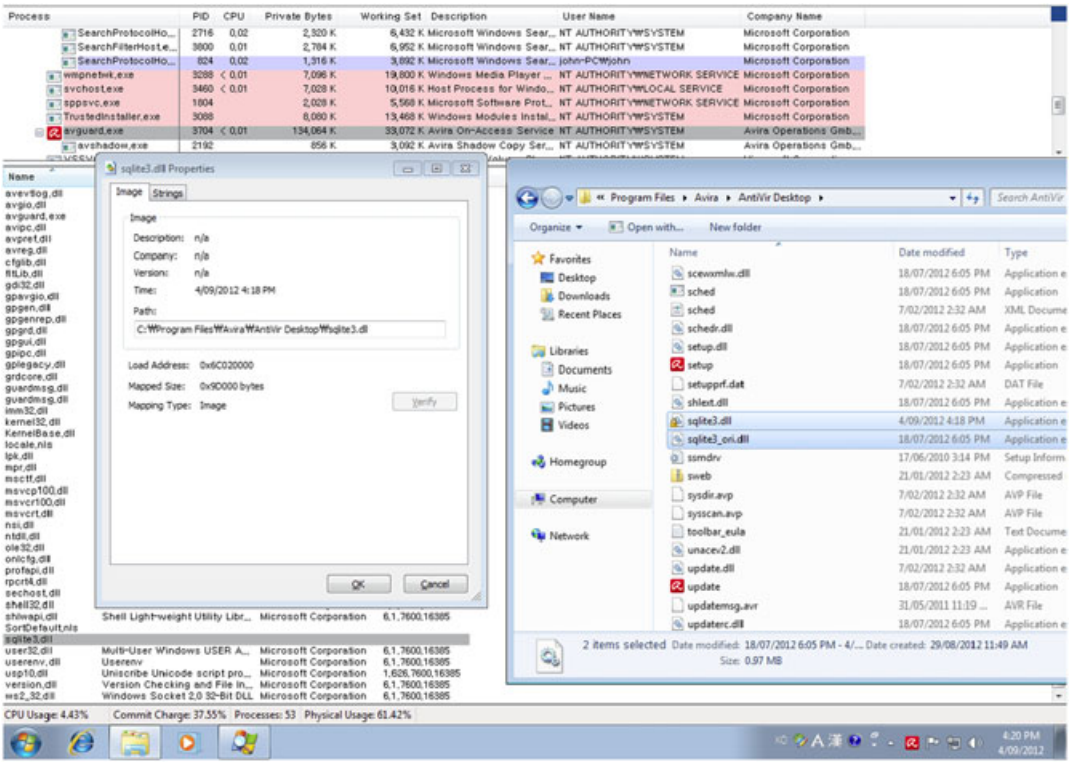

Figure 10. sqlite3.dll: replaced with malicious one and loaded by Avira's service.

DOI: 10.1002/spe

Instead of DLL replacement, EXE infection, modification and replacement are also possible. In this case, the installer adds its malicious payload to the service binary so that the payload can run whenever the service starts. In addition, an attacker may temporarily unload Avira's kernel-mode driver and substitute one of driver-related files so that he/she can gain kernel privilege, which will be the worst case.

6.2. Attack 2: arbitrary binary execution to compromise the operating system

In step 5, in the previous attack scenario, although Realtime Protection service is inactive, an attacker has several other options. Among them, dropping and executing a binary is one of the simplest malicious actions. Using this, we were able to install different types of malware including rootkits, trojan horses, backdoor and spyware.

This attack might seem trivial and featureless. But the importance of this attack lies in the difference between signature and behaviour detection techniques. Many evasion techniques using obfuscated code [1–4, 38–41] aim to bypass antivirus software's detection engine. The engine may employ traditional file signatures and/or heuristic (program-logic) signatures [42]. No matter what the engine is, once a malware bypassed it, the security of a system depends largely on the behavioural detection engine of the antivirus software. Behavioural detection engine monitors all the processes running on the system [††] and makes the decision as to whether a process is malicious or not based on its behaviour. At the same time, false positive rate should be very low so as not to annoy users with false alarms. However, in contrast to signature-based engines that use obvious footprints of malware, there can be many behaviours that can be either suspicious or legitimate depending on the context. This makes it really hard to implement a reliable behavioural detection engine (heuristic signature engines share similar difficulty). For this reason, obfuscated malware is still prevalent. In this context, a malware installer can execute any binary that is detected by Avira's file and heuristic signature engine but whose behaviours are not detected by Avira's behavioural engine, called Avira AntiVir ProActiv. Even the simplest form of malware that is not obfuscated at all can be dropped and executed without being detected.

We can take this attack one step further. In contrast to obfuscated malware whose actions can be detected by antivirus software, any malware that is detected by both signature (file and heuristic) and behavioural engines can be executed in this attack because Avira's Realtime Protection service is simply paused. In other words, malware do not have to bypass Avira, as it is deactivated. Let us consider an MBR rootkit installer executable. This installer is detected by Avira's signature engine, and its installation process is detected by Avira's behavioural engine. But a malware installer that is not detected by antivirus software can embed this rootkit installer module. Then, it drops and executes the module while Avira's Realtime Protection service is inactive. As a result, an MBR rootkit is successfully installed on the target system. Since then, the malware's processes and network activities become invisible to the operating system, thus undetectable by the antivirus software. Even though an actual experiment on this type of attack is not given here, we believe that it would not be hard to find a malware sample that is detected by both detection engines, but the behaviours of its installed (real) malware is not.

6.3. Attack 3: denial of service (DoS) on Avira

The final practical attack example that we present in this paper is a DoS attack on Avira. At the fifth step in the first attack scenario, the attacker replaces sqlite3.dll (or any other DLL the service loads) with a non-functional file. As shown in Figure 11, the service can no longer restart even if the user manually tries to start it. As a consequence, not only Realtime Protection service but also the other two services (Scheduler and Web Protection) fail to start after rebooting, thus making Avira nearly useless; even an update fails to start when the user clicks the 'Start update' button (see Figure 12).

[††]Trusted or predefined processes may be excluded from monitoring depending on the antivirus solutions.

DOI: 10.1002/spe

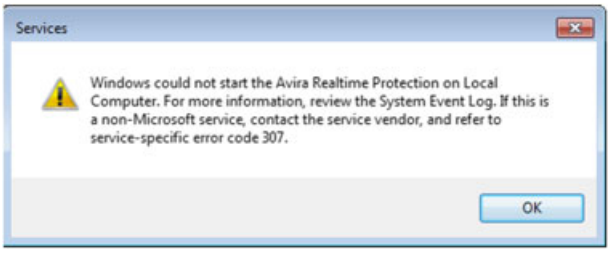

Figure 11. sqlite3.dll is replaced with a non-functional file and Avira's service cannot start.

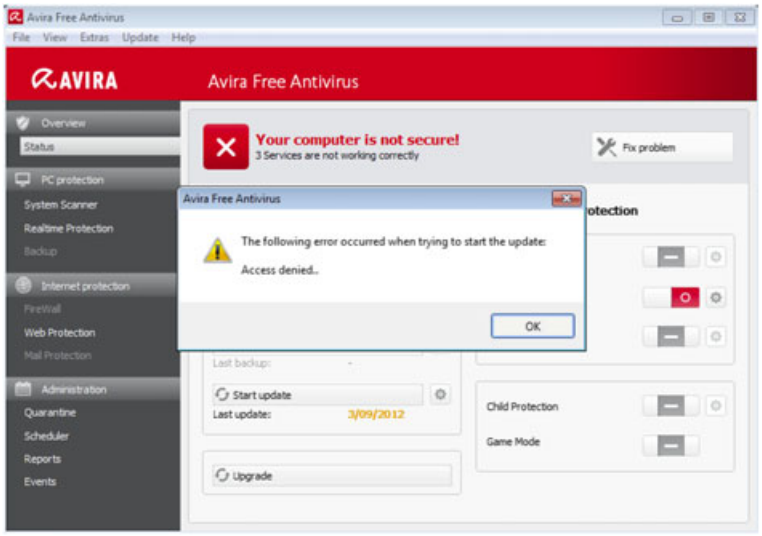

Figure 12. Update fails to start after denial of service attack.

Denial of Service attack can also be performed with a simple file copy. For instance, we placed a non-functional version of system DLL, for example, mpr.dll and version.dll, in the installation directory. Not only the Realtime Protection service but also user interface processes such as avgnt.exe (Avira System Tray Tool) and avcenter.exe (Avira Control Center) failed to start, making Avira entirely ineffective. After trying a few times, in the end, the user might think that there is a problem with Avira and decide to remove it; if this is carried out, it opens up an even bigger opportunity for the malware. Even when the user re-installs Avira or any other security tools, the system is exposed to threats during the time of removal and re-installation.

7. ATTACKS ON OTHER ANTIVIRUS SOFTWARE: AVG, MCAFEE, MICROSOFT AND SYMANTEC

As mentioned in Section 4, products from AVG, McAfee, Microsoft and Symantec are also vulnerable to the type of attacks mentioned earlier. We have also experimented with these antivirus software products. In this section, we present the screenshots corresponding to Avira's first attack case: loading malicious DLL in the context of antivirus software, thus subverting antivirus software itself and the operating system. Note that normally, service restart cannot be triggered by a user, and the frequency of update that leads to service restart and/or kernel-mode driver reloading is different in different antivirus products.

Our experimentation shows that AVG, McAfee and Microsoft are relatively easy targets. In these three cases, at least one service (AVGIDSAgent in the case of AVG Antivirus 2012, seven services in the case of McAfee Total Protection 2013 and Microsoft Anti-malware Service in the case of MSSE 4) restarts during some updates, and file replacement can be performed during this restart

DOI: 10.1002/spe

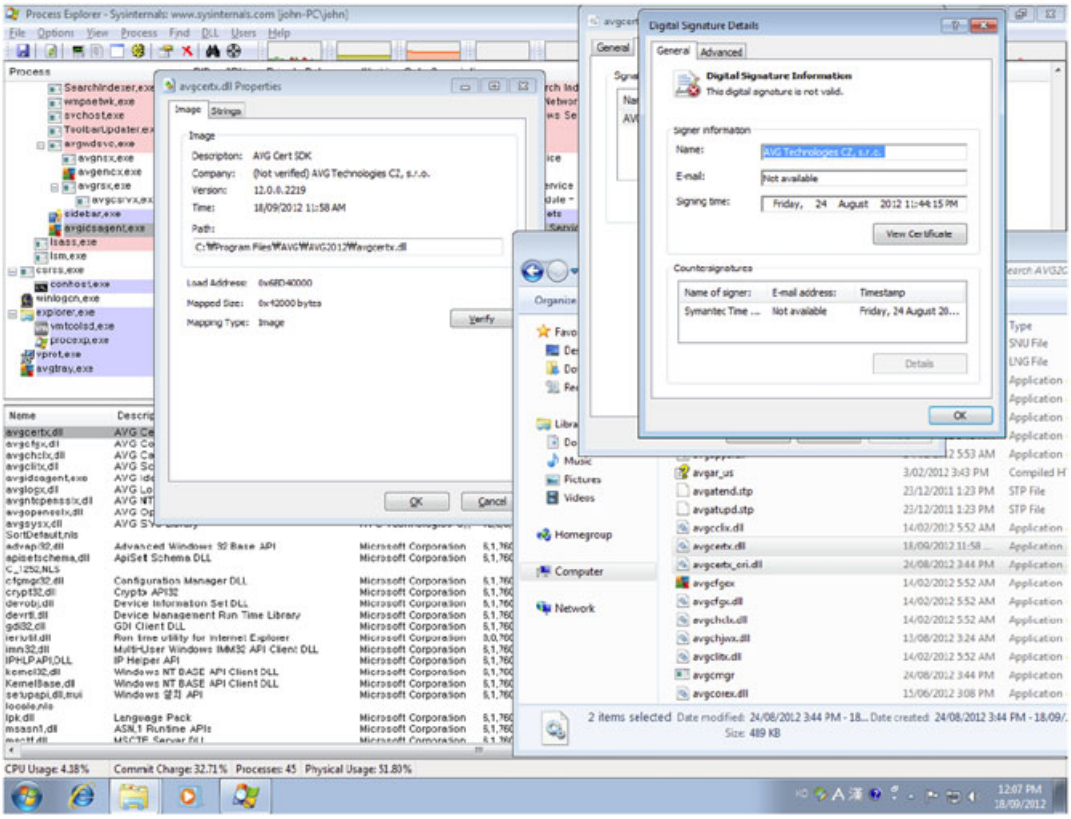

Figure 13. avgcertx.dll: replaced with malicious one and loaded by AVG's service.

without any further action, such as additional service stop or kernel-mode driver unloading. We have implemented loading of a malicious DLL in the context of these products, which are shown in Figures 13–15.

Symantec's Norton Internet Security 2013[‡‡] is a little bit harder to exploit, because it verifies the certificates of the files that it loads. Therefore, even if an attacker successfully replaced its DLL, the malicious file would not be loaded by the service of Norton Internet Security 2013. However, it turned out that it does not check other vendors' files. In particular, we found that the NIS service loaded Microsoft C Runtime Library (msvcp100.dll and msvcr100.dll), and these files were loaded by the service even if they are not properly signed as shown in Figure 16.

We also observed that other attacks are possible; for instance, DoS on these products can be carried out by replacing a file with a non-functional one. Interestingly, in the case of Symantec's Norton Internet Security 2013, DoS is even easier, as it verifies the file's certificate. Even if the replaced file is functionally identical to the original one, it is not loaded by the service as long as it does not have a valid certificate. As an example, if the verification on a core file, ccSvc.dll, has failed, update never finishes. If the annoyed user reboots the system, not only the service but also other service and processes including the one responsible for user interface cannot run, making the product totally ineffective on the system.

8. POSSIBLE SOLUTIONS

There can be a trivial solution, which involves fixing the design vulnerability and preventing the service from restarting during any update. However, such a product-level patch is usually more feasible

[‡‡]Notice that this product asks the user whether to apply the patch now or later. The restart of the Norton Internet Security's service occurs only when the user clicks 'apply now' button; otherwise, patch is applied on next reboot.

DOI: 10.1002/spe

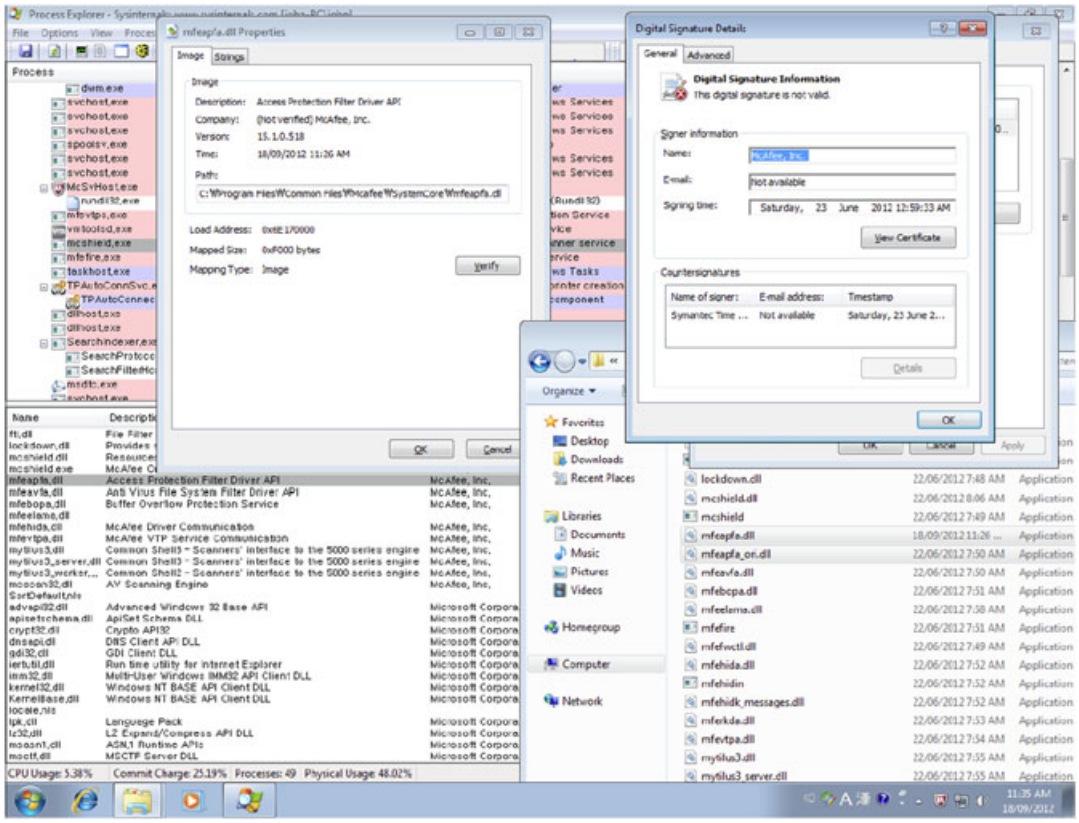

Figure 14. mefapfa.dll: replaced with malicious one and loaded by McAfee's service.

with the release of a new version. In this section, we suggest solutions that can be applied to the current version for closing the security vulnerability described in Section 4; therefore, we assume that service restart is a necessary evil during an update.

The root cause of the vulnerability lies in the update procedure. All the attacks including kernel-mode driver unload, file replacement, and arbitrary binary execution are possible as one or more services are temporarily deactivated during the update. Therefore, we can solve this problem by making these services remain activated at all time (even during an update) by creating temporary instances of these services. For example, Avira's update process may create and start a temporary service with a random name, for example, Realtime Protection[XXXXXXXX], which is functionally equivalent to Realtime Protection service. This temporary service uses not-yet-updated configuration data, engine and definitions. Then, the original service is stopped, and the newly downloaded engines and signatures are applied. As the final step, Realtime Protection service is started with the updated data, and the temporary service is stopped and deleted. If such a procedure is followed, then the period where the Realtime Protection service becomes inactive is eliminated, thus removing the vulnerability. As a result, the attacks described in 6 are not possible. This procedure is shown in Figure 17.

Another measure that can be used is similar to the one being used in MSSE. On every update, MSSE deletes its existing kernel-mode driver named in the form of MpKsl[XXXXXXXX], for example, MpKsl8cbe80ce, and installs a new one later. In the case of MSSE, it seems that the creation of a new driver may be delayed until the next system scan, and hence, there can be a relatively long time frame where an update-related driver does not exist. Because the driver appears to be

DOI: 10.1002/spe

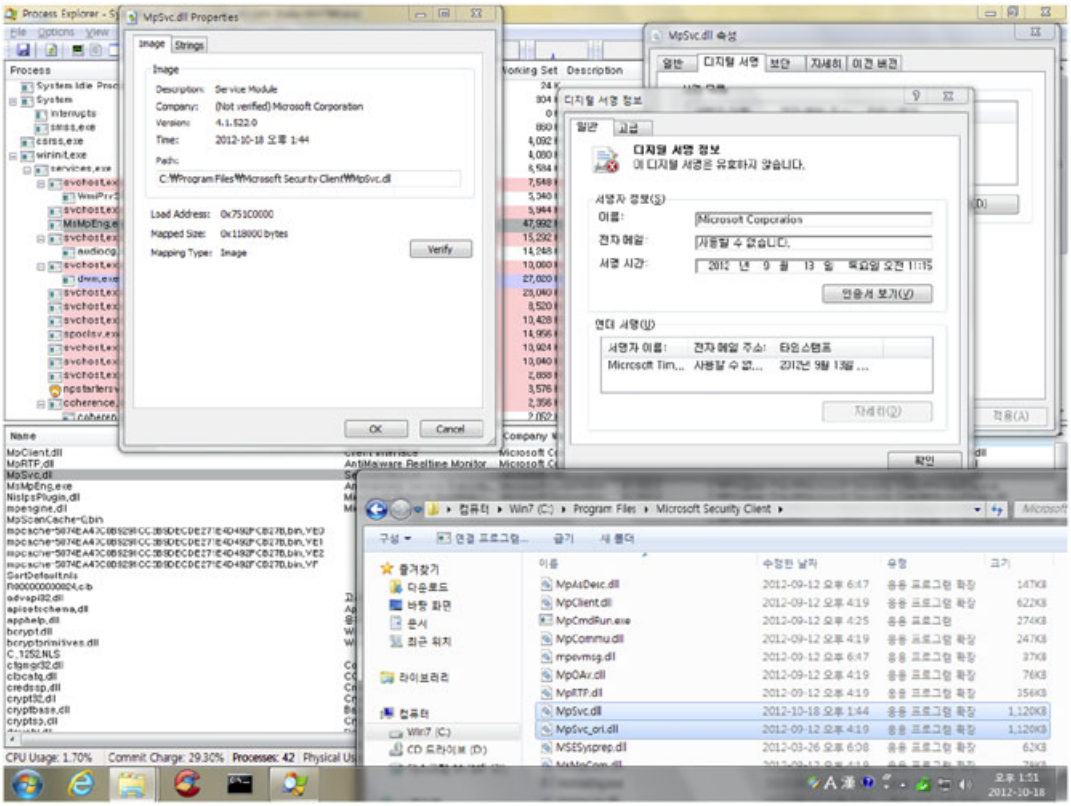

Figure 15. MpSvc.dll: replaced with malicious one and loaded by MSSE 4's service. Although the system language is Korean, one can verify that MpSVC.dll with an invalid signature has been loaded by the service.

related only to the update procedure and not to the impact on the functionality of MSSE, such a time frame might not be a big problem for MSSE.[§§]

Another additional defensive measure that can be taken is as follows. The update process can validate the certificates of all the executables and modules (not only the newly downloaded ones but also existing ones) before a service or a driver temporarily stops. If any validation fails, the update should be cancelled, and the file that fails to be validated must be restored with a valid one; and the restoration should be performed in a reliable way. If not, DoS attack might be possible by blocking the restoration process after the file replacement. This measure can further prevent EXE or DLL file replacement and placement attacks. Even though an attacker somehow succeeds in replacing or placing a binary that is supposed to be loaded by the service, the malicious binary cannot pass the validation unless it has a valid certificate. It would be better if the validation process requires the binaries to have valid certificates from the vendor itself and Microsoft, because a stolen but still valid certificate can be used by malware to pass such validation checks and bypass security tools. We have witnessed such cases in Stuxnet [17] and Duqu [18]. Recently, there was even a case that criminals used fake company data to obtain a certificate from a Certificate Authority (CA) [43]. If the validation process only accepts valid certificates from the vendor and Microsoft, even a malicious binary signed with an abused certificate cannot pass the validation and thus fail to subvert the antivirus solution and/or the system. To break this measure, an attacker has to gain a valid certificate from the vendor and Microsoft, which can be difficult. Also, such a validation mechanism can be easily implemented in the current version, because almost all the binaries of all antivirus software products that we have investigated are already signed with their vendor's certificates. However, we

[§§]As we did not perform any detailed investigation on MSSE, we cannot exclude the possibility that the driver has something with the functionality of MSSE in any way.

DOI: 10.1002/spe

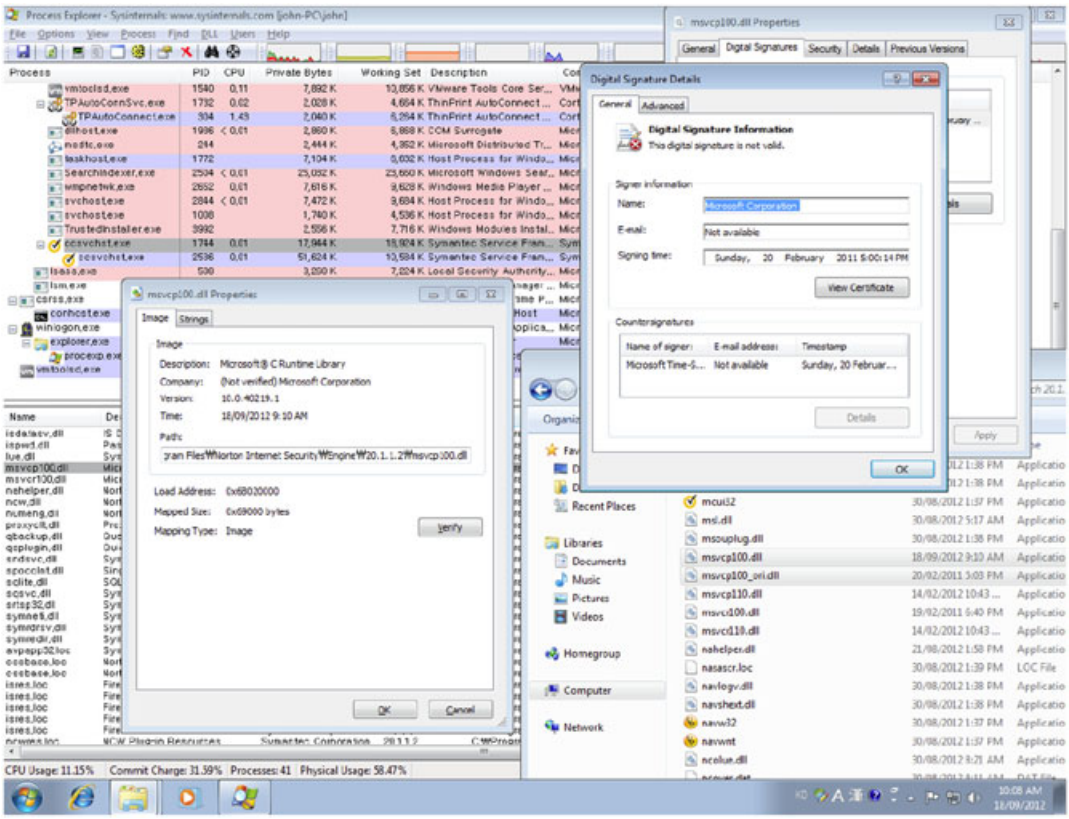

Figure 16. msvcp100.dll: replaced with malicious one and loaded by NIS 2013's service.

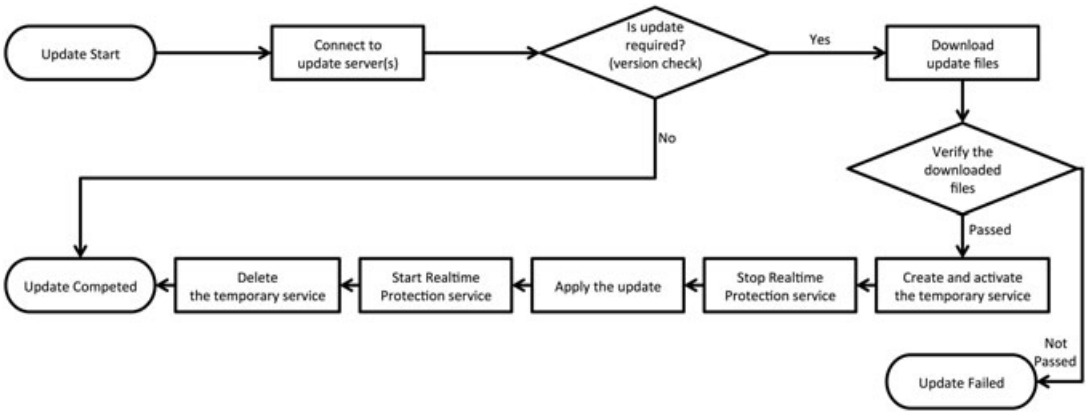

Figure 17. Secure update procedure for Avira.

should point out that this mitigation method may not be able to completely block DoS attacks. If the attacker can replace or place a file in the installation folder, the attacker can use a file that has valid certificate but does not provide any relevant functionality. For example, the attacker can copy a valid Microsoft-signed system DLL that resides in the Windows system folder to the antivirus software's installation folder and renames it to the DLL the antivirus tries to use, for example, sqlite3.dll. By default, the current folder is searched prior to the Windows system folder in Windows applications [44], and the copied DLL is loaded by the antivirus software. Although the file is signed with a valid Microsoft certificate, thus passing the validation process, services or other processes of the antivirus solution that use this DLL will fail to start because it does not provide desired functionalities.

DOI: 10.1002/spe

Therefore, this measure should not be thought of as a complete mitigation. It should be used with the aforementioned update solution[¶¶]

We strongly recommend that both security measures be implemented, as it is always good to realise the concept of security-in-depth or defence-in-depth, even within a single computer system or a single software product.

If the system under consideration is a virtual machine (VM)-based system, then the VMM can provide an additional security layer to the system and can act as a security tool during updates. This can help to resolve one of the root causes of the problem, namely, the security of a modern computer system heavily depends on the effectiveness of the antivirus solution(s) installed on it. The VMM can easily detect the start of an update by monitoring the VM's network activity. Once an update is detected, the VMM may enforce certain security policies so that the state of the VM cannot be altered except for the processes involved in the update. For instance, VMM may pause all other processes including the malware installer, or block any process, file and registry operations of other processes. As a result, the malware installer cannot perform any malicious activity during an update and therefore fails to compromise the system and the antivirus software.

In addition, the VMM can be used to provide additional security features. For example, users can use a specific VM for a specific purpose. That is, there can be one VM for carrying out sensitive transactions such as Internet banking, another VM for say playing computer games and even another one for general web browsing. However in this case, it may be that a VM has not been used for a relatively long time, thus resulting in an out-dated system. This issue can be addressed by the VMM checking the update status of a VM before it is started or resumed. Prior to allowing the use of a VM, VMM updates the VM. Such an update will need to include operating system updates, antivirus update and other mainstream software updates such as flash, JRE and Adobe reader.

9. RELATED WORK

In anti-malware research community, research on evading detection of antivirus software is an important and active research area. The ultimate goal is cheating and bypassing the detection engine of antivirus software by using obfuscation techniques including polymorphism, metamorphism and packing. General description of these three techniques can be found in [1, 2, 4]. Metamorphic malware can reprogram itself. In general, self-reprogramming is performed within the malware prior to its propagation so as to make its child malware look different from itself. Specific example of metamorphism can be found in [1, 4], and detection techniques are discussed in [6–8, 10]. Polymorphic malware is generated by polymorphic engines that usually use mutation to change the appearance of input program (malware) while preserving its semantics. Various defensive techniques against polymorphism are suggested in [5, 9]. Packers are used to make variations of a malware using techniques such as compression and encryption. Many packers are available including UPX, Themida, ASPack, PECompact and UPACK. Oberheide *et al.* proposed an online automated packing service, called PolyPack, that uses an array of packers and antivirus engines to achieve optimal evasion of antivirus solutions [3]. Besides these three evasion techniques that are typically used with executables, file obfuscation techniques for other file formats have also been proposed. Alvarez and Zoller used archive files to attack file processing mechanism of antivirus solutions and showed practical attacks such as antivirus bypass, DoS and code execution [40, 41]. Porst suggested an evasion technique that generates script-embedded PDF files that are not recognised by antivirus software but parsed by Adobe reader without a problem [39]. Jana and Shmatikov [38] suggested two antivirus evasion techniques that use various file formats. They showed even a simple and unobfuscated malware can bypass almost all antivirus software's detection engine. These evasion techniques normally deal with the case on how a malware or a malicious file (a dropper/downloader in the case of staged malware) can be executed without being detected by antivirus software. One exception to this is [40], which covers general software vulnerabilities as well as detection engine evasion. In contrast,

[¶¶]Also, it is important to use strong hashing and signing algorithms when creating certificates to avoid motivated attackers from forging certificates. The notorious Flame malware exploited MD5 hashing algorithm (weak collision resistant) and succeeded in making seemingly valid certificates [45].

DOI: 10.1002/spe

our work discusses how an antivirus software can be totally bypassed (including signature-based and behavioural-based detection engines) and subverted, as well as how the operating system can be completely compromised, after a malware or malicious file is delivered to a target system.

A few types of rootkits have their aim in common with our work in the sense that they also target the period when antivirus solutions are inactive. Rootkit is a kind of malware that hides the existence of malicious processes so that they can maintain control over the target system without being detected. In-depth survey and detection techniques of rootkits can be found in [46–48]. MBR (the first sector of a disk) rootkit is designed to replace the MBR with one of their own. As a result, it runs before the operating system starts up [49]. Kleissner showed how an MBR rootkit can patch Windows operating system before it starts so that unsigned drivers can be loaded [26]. Other examples include BIOS [25] and EFI rootkits [23, 24]. They infect BIOS and EFI rom code, respectively, so as to get control over the target system at the very early stage of system boot. Although antivirus software tries to detect rootkits, any malware including rootkits can be installed by a dropper or a downloader once the malware compromises the antivirus software and the system as described in Section 6.2.

There have also been other works analysing the effectiveness of antivirus solutions. Some suggest that antivirus solutions are useless as they cannot detect unknown threats and whereas others suggest that users must install one to protect themselves from known and unknown threats [50]. Sukwong *et al.* [42] analysed six commercial antivirus products and showed that even behavioural engines cannot detect all forms of malware. Josse [51] suggested an approach that can help users to choose an antivirus software with a certain degree of confidence. Oberheide *et al.* proposed the idea of antivirus as an in-cloud service and implemented CloudAV that uses 10 antivirus engines and two behavioural detection ones [52]. Gashi *et al.* also supported the benefits of antivirus diversity [53, 54]. The strategy of leveraging multiple antivirus software via network service can be useful for certain circumstances such as file scans. However, as we mentioned earlier, many staged malware are using an initial installer that is not detected by almost all antivirus software; then, it may directly inject malicious code into some process or use fileless-style malware [55].

Antivirus solutions are indeed a kind of software, and like any other software, they can have typical implementation vulnerabilities such as buffer overflow. Aforementioned antivirus attack using archive files [40, 41] is an example. This attack exploits antivirus software's memory corruption vulnerability and executes arbitrary code. Other instances of antivirus software's traditional implementation vulnerabilities can be found in [56, 57]. Finally, Al-Saleh and Crandall [58] suggested an interesting method for remotely probing how latest the signature database of antivirus software is and showed the result on ClamAV antivirus.

Finally, there is a framework called Evilgrade that allows attackers to inject fake updates into vulnerable software [59]. Then, the injected malicious update files can be executed. This framework requires the attacker to be able to make traffic redirection, which can be performed in various ways such as DNS tampering, DNS Cache Poisoning, ARP spoofing, Wi-Fi Access Point impersonation and Dynamic host configuration protocol (DHCP) hijacking. Evilgrade exploits the lack of verification and/or validation on the downloaded update files; this is different from our work that exploits a design vulnerability.

10. CONCLUDING REMARKS

In this paper, we propose a novel attack vector that exploits the unique characteristics of antivirus software, namely, the role of antivirus software in the provision of system security and the high frequency of updates that are needed. There is no limitation on the range of vulnerability related to updates. It can be a traditional implementation flaw, a fundamental design mistake or a logic fault. If such a vulnerability were found in an antivirus software, the system as well as the antivirus software could be totally compromised. We showed that this type of vulnerability indeed exists in real-world antivirus products from Avira, AVG, McAfee, Microsoft and Symantec. We have also demonstrated how such a vulnerability can be exploited for arbitrary operations and local privilege escalation. Defensive measures that can mitigate such vulnerability have also been discussed.

DOI: 10.1002/spe

The gate to a new attack vector has just been opened, and we believe that this design vulnerability related to update procedure may be present in other antivirus software. There could also be other types of vulnerability related to update procedure; for example, investigation of the re-initialisation of Realtime Protection service might reveal another vulnerability that can be triggered on every update.

However, it is important to stress that it is absolutely essential that at least one antivirus software must be present on every computer system, even if the antivirus software is somewhat imperfect. The larger the software, the greater the probability that any software will have a bug, and antivirus software is no exception. However, it is also true that antivirus and other security tools raise the bar for the attackers to breach the security of a system. Furthermore, it is important to note that there is considerable research happening in the anti-malware research community, which are aimed at improving detection techniques, especially behavioural-based ones, which will help to better counteract the attacks in the future [42].

REFERENCES

1. Murad K, Shirazi S, Zikria Y, Ikram N. Evading virus detection using code obfuscation. *Future Generation Information Technology* 2010; **6485**:394–401.
2. O'Kane P, Sezer S, McLaughlin K. Obfuscation: the hidden malware. *IEEE Security & Privacy* 2011; **9**(5):41–47.
3. Oberheide J, Bailey M, Jahanian F. PolyPack: an automated online packing service for optimal antivirus evasion. *Proceedings of the 3rd USENIX Conference on Offensive Technologies*, Montreal, Canada, 2009; 99.
4. Rad BB, Masrom M, Ibrahim S. Camouflage in malware: from encryption to metamorphism. *IJCSNS International Journal of Computer Science and Network Security* 2012; **12**(8):74–83.
5. Newsome J, Karp B, Song D. Polygraph: automatically generating signatures for polymorphic worms. *IEEE Symposium on Security and Privacy 2005*, CA, USA, 2005; 226–241.
6. Lee J, Jeong K, Lee H. Detecting metamorphic malwares using code graphs. *Presented at the SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing*, Sierre, Switzerland, 2010; 1970–1977.
7. Lin D, Stamp M. Hunting for undetectable metamorphic viruses. *Journal in Computer Virology* 2011; **7**(3):201–214.
8. Wong W, Stamp M. Hunting for metamorphic engines. *Journal in Computer Virology* 2006; **2**(3):211–229.
9. Xu J, Sung AH, Mukkamala S, Liu Q. Obfuscated malicious executable scanner. *Journal of Research and Practice in Information Technology* 2007; **39**(3):181–198.
10. Szor P, Ferrie P. Hunting for metamorphic. *Virus Bulletin Conference*, Prague, Czech, 2001.
11. AhnLab Analysis Team. Malware analysis: citadel, AhnLab ASEC, (AhnLab Security Emergency response Center), December 2012.
12. IOActive. Reversal and analysis of Zeus and SpyEye Banking Trojans, 2012. White paper, IOActive.
13. Symantec Security Response Team. *The Downadup Codex: A Comprehensive Guide to the Threats Mechanics Edition 2.0*, 2009. White paper, Symantec Security Response.
14. Microsoft malware protection center, "Win32/Ramnit", 2011. Available from: http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32%2fRamnit [last accessed 2 February 2013].
15. Falliere N. Sality: story of a peer- to-peer viral network, 2011. White paper, Symantec Security Response.
16. Symantec Intelligence. Symantec intelligence report: December 2012 Symantec, 2012.
17. Falliere N, Murchu LO, Chien E. W32.Stuxnet dossier, 2011.
18. Chien E, Murchu LO, Falliere N. W32.Duqu The precursor to the next Stuxnet, White paper, Symantec Corp., Security Response, November 2011.
19. Anity labs, analysis report on flame worm samples, July 2012. Anity Labs.
20. Kaspersky lab, Gauss: abnormal distribution, August 2012. Kaspersky Lab.
21. Design, develop, and certify hardware for Windows 8. Available from: http://msdn.microsoft.com/en-us/library/windows/hardware/br259114.aspx [last accessed 2 February 2013].
22. Early launch anti-malware. Available from: http://msdn.microsoft.com/en-us/library/windows/hardware/br259096 [last accessed 2 February 2013].
23. Heasman J. Hacking the extensible firmware interface, Black Hat USA, 2007.
24. Loukas, K. (Snare), De Mysteriis Dom Jobsivs: Mac EFI Rootkits, Black Hat USA, 2012.
25. Sirag H, Bondugula N, Gupta R. Advanced persistent attacks: BIOS rootkit-Mebromi, 2011.
26. Kleissner P. Stoned bootkit, Black Hat USA, 2009.
27. Steve, is it good to have multiple antivirus programs on my PC? zolexpc.com, 10-Jun.-2010. [Online]. Available from: http://zolexpc.com/blog/?p=147 [last accessed 2 February 2013].
28. Avira support. Available from: http://www.avira.com/en/support-for-home-knowledgebase-detail/kbid/1081 [last accessed 2 February 2013].
29. McAfee communities. Available from: https://community.mcafee.com/ [last accessed 2 February 2013].
30. Avast forum. Available from: http://forum.avast.com/index.php?topic=106560.msg848307#msg848307 [last accessed 2 February 2013].

DOI: 10.1002/spe

31. Symantec support. Available from: http://us.norton.com/support/ [last accessed 2 February 2013].
32. Symantec connect community. Available from: http://www.symantec.com/connect/forums/how-often-does-new-definition-file-come-out [last accessed 2 February 2013].
33. Sophos endpoint security and control upgrade guide. Available from: http://www.sophos.com/en-us/medialibrary/PDFs/documentation/sesc_102_ugeng.pdf [last accessed 2 February 2013].
34. OPSWAT. Market share report, September 2012. opswat.com, Sep-2012. [Online]. Available from: http://www.opswat.com/about/media/reports/antivirus-september-2012 [last accessed 2 February 2013].
35. AV-comparatives, summary report 2011, December 2011. AV-Comparatives.
36. djwm, Only 9 of 22 virus scanners block Java exploit. h-online.com, 31-Aug.-2012. [Online]. Available from: http://www.h-online.com/security/news/item/Only-9-of-22-virus-scanners-block-Java-exploit-1696462.html [last accessed 2 February 2013].
37. CVE. Available from: http://cve.mitre.org [last accessed 2 February 2013].
38. Jana S, Shmatikov V. Abusing file processing in malware detectors for fun and profit. *IEEE Symposium on Security and Privacy 2012*, CA, USA, 2012; 80–94.
39. Porst S. *How to Really Obfuscate your PDF Malware*, 2010. ReCon.
40. Alvarez S, Zoller T. The death of AV defense in depth ?- revisiting antivirus software, 2008. CanSecWest.
41. Alvarez S. Antivirus (In)security, CCC (Chaos Communication Camp), 2007.
42. Sukwong O, Kim H, Hoe J. Commercial antivirus software effectiveness: an empirical study. *Computer* 2011; **44**(3):63–70.
43. Assolini F. Brazilian Trojan bankers now digitally signed. securelist.com, 03-Sep.-2012. [Online]. Available from: http://www.securelist.com/en/blog/208193825/Brazilian_Trojan_bankers_now_digitally_signed [last accessed 2 February 2013].
44. MSDN. Search path used by Windows to locate a DLL. Available from: http://msdn.microsoft.com/en-us/library/7d83bc18(v=vs.80).aspx [last accessed 2 February 2013].
45. Sotirov A. Analyzing the MD5 collision in flame, June 2012. SummerC0n.
46. Bravo P, Garca DF. Rootkits Survey, 2011.
47. Bravo P, Garca DF. Proactive detection of kernel-mode rootkits. *Sixth International Conference on Availability, Reliability and Security (ARES)*, Vienna, Austria, 2011; 515–520.
48. Kapoor A, Mathur R. Predicting the future of stealth attacks. *Virus Bulletin Conference*, Barcelona, Spain, 2011.
49. Corrons L, Hoskins D. MBR Trojans: Exploring MBR rootkits, Network Security, 2008.
50. Potter B, Day G. The effectiveness of anti-malware tools. *Computer Fraud & Security* 2009; **2009**(3):12–13.
51. Josse S. How to assess the effectiveness of your antivirus? *Journal in Computer Virology* 2006; **2**(1):51–65.
52. Oberheide J, Cooke E, Jahanian F. CloudAV: N-version antivirus in the network cloud. *Proceedings of the 17th Conference on Security Symposium*, CA, USA, 2008; 91–106.
53. Gashi I, Stankovic V, Leita C, Thonnard O. An experimental study of diversity with off-the-shelf antivirus engines. *Eighth IEEE International Symposium on Network Computing and Applications (NCA 2009)*, MA, USA, 2009; 4–11.
54. Bishop P, Bloomfield R, Gashi I, Stankovic V. Diversity for security: a study with off-the-shelf antivirus engines. *IEEE 22nd International Symposium on Software Reliability Engineering (ISSRE) 2011*, London, UK, 2011; 11–19.
55. Golovanov S. A unique fileless bot attacks news site visitors. securelist.com, 16-Mar.-2012. [Online]. Available from: http://www.securelist.com/en/blog/687/A_unique_fileless_bot_attacks_news_site_visitors [last accessed 2 February 2013].
56. Wheeler A, Mehta N. Owning antivirus. *Black Hat Europe Conference*, Amsterdam, Netherlands, 2005.
57. Xue F. Attacking antivirus. *Black Hat Europe Conference*, Amsterdam, Netherlands, 2008.
58. Al-Saleh MI, Crandall JR. Application-level reconnaissance: timing channel attacks against antivirus software. *4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, MA, USA, 2011.
59. Evilgrade. Available from: http://code.google.com/p/isr-evilgrade/ [last accessed 2 February 2013].

DOI: 10.1002/spe