Attacking Antivirus

Feng Xue

Technical Lead, Nevis Labs.

Nevis Networks, Inc.

ABSTRACT

Antivirus solutions are now a common component of computer systems. However, security issues pertaining to the antivirus software itself have not captured enough attentions of antivirus vendors and computer users.

This paper discusses why antivirus software is vulnerable to various attacks and why its security is so critical. It examines the tools and techniques, especially fuzzing techniques, used by attackers to expose vulnerabilities in antivirus solutions. It also looks at the ways in which attackers exploit these vulnerabilities

The paper thus aims to raise levels of consciousness about the security of the security product.

KEYWORDS: Antivirus, Audit, Exploitation, Fuzzing, Security product

1. INTRODUCTION

According to the U.S. national vulnerability database [1], as shown in Figure 1, 165 vulnerabilities have been reported in antivirus software in the past 4 years.

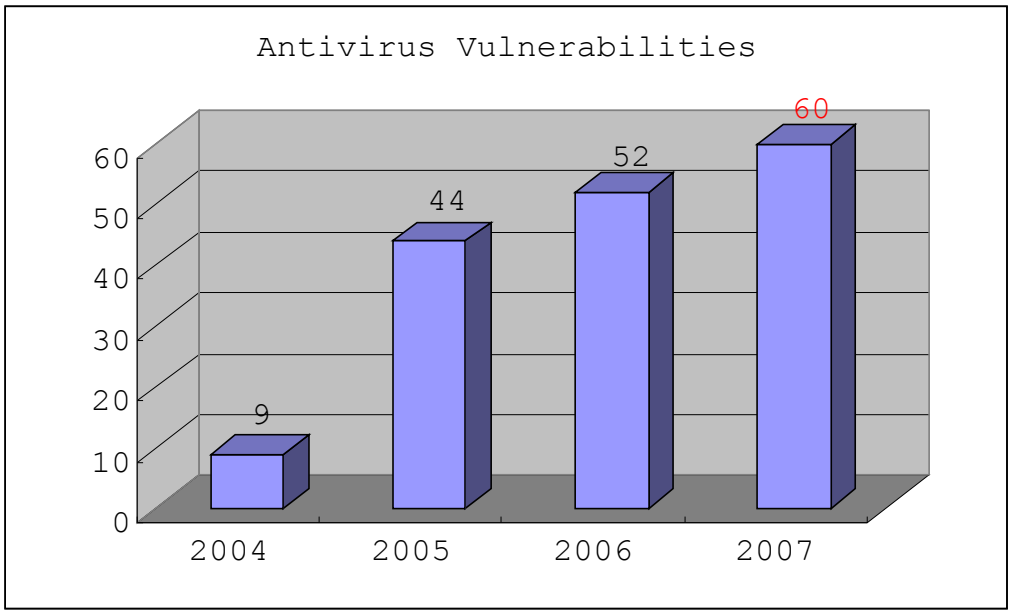


Figure 1 National Vulnerability Database

Thus, it is clear that antivirus software can be targeted just likes other components or services of computer systems.

Section 2 discusses why antivirus software is vulnerable to attacks. Section 3 discusses the techniques used to unravel vulnerabilities of antivirus software -- source code audits for example, or reverse engineering and fuzzing. Exploitation techniques will be examined in Section 4.

.

2. WHAT MAKES ANTIVIRUS A PERFECT TARGET

2.1 People have complete faith in it

The use of antivirus software has become something of an act of faith. People seem to feel more safe not with a more secure operating system, or with the latest patch, but with some antivirus software installed in their systems.

A recent study [3] shows that 81 per cent of all computer users have antivirus software installed on their computers. Quite clearly, antivirus software is a must-have for most users.

The questions are: Is that enough? Is such blind faith justified? What if attackers attack the antivirus software itself instead of the operating system?
Now that would turn the game on its head, wouldn't it?

Consider an average user, who gets some files (executables, documents, media, etc.), the installed antivirus on his computer will scan the incoming files automatically (The user may manually scan it if it looks suspicious). And with this the antivirus would serves the security gate for incoming files. Figure 2

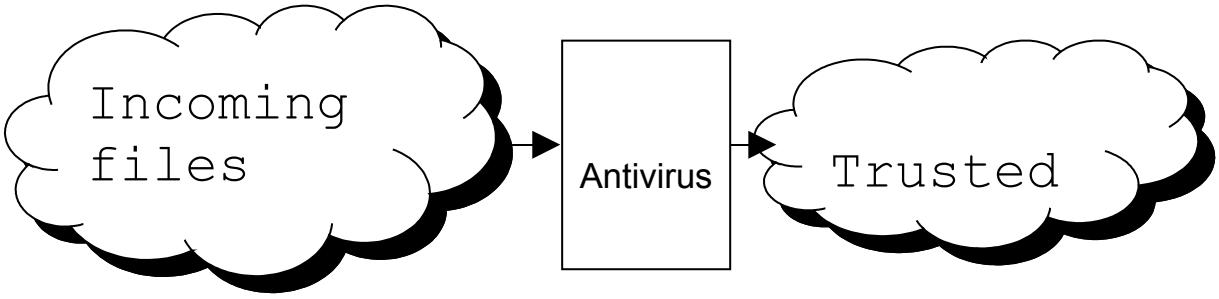


Figure 2: AV Antivirus serves the security gate for incoming files.

What he or she does not know is that many antivirus solutions developed in the past, were developed without holistic security in mind. Developers would assume that non-trusted files were safely being scanned by their software. But what if those very files hurt their solution software itself?

The threat to antivirus security is thus helped along by two things:

- The user's blind acceptance of the antivirus as a silver bullet.
- And the overconfidence of antivirus vendors in their software's immunity against all files.

2.2 Antivirus processes are error-prone

Antivirus software is one of the most complicated applications. It has to deal with hundreds of file types and formats:

- executables (exe, dll, msi, com, pif, cpl, elf, ocx, sys, scr, etc);
- documents (doc, xls, ppt, pdf, rtf, chm, hlp, etc);
- compressed archives (arj, arc, cab, tar, zip, rar, z, zoo, lha, lzh, ace, iso, etc);
- executable packers (upx, fsg, mew, nspack, wwpack, aspack, etc);

- media files (jpg, gif, swf, mp3, rm, wmv, avi, wmf, etc),

Each of these formats can be quite complex. Hence, it is extremely difficult for antivirus software process all these format appropriately.

This is amply clear in recent research into antivirus vulnerabilities. It reveals that most vulnerabilities exist in the following two components:

- Executable decompression [4].
- Data decompression [5].

Antivirus software will try to decompress the compressed executable and data before processing them.

The problem with the decompression of executables and data is that both the processes are highly complicated. The antivirus makes complex calculations, allocates memory, and extracts data according to the calculation. Any mistake in these throws open the door for vulnerabilities.

These points may be compelling enough for attackers to target antivirus solutions but how do they go about finding chinks in the antivirus armor?

3. FINGING VULNERABILITIES OF ANTIVIRUS

Basically there are four kinds of vulnerabilities seen in antivirus software:

- Local Privilege Escalation
- ActiveX-related
- Engine-based
- Management (Administrative) interface

The methodologies of auditing antivirus processes for each of the above are quite different from one another.

3.1 Auditing the Local privilege escalation issues

3.1.1 Weak DACL

Auditing weak DACL issues is one of the easiest things, sometimes it can be done manually without the help of any tools.

To check whether an antivirus installation directory (files) is vulnerable to this issue, you just need to right click on the directory (files) and navigate to the “security” tag. If the “Everyone” group has a “full control” permission, then probably it’s a brand new local root vulnerability!

Checking the ACL of services is a little bit different from the installation directory. It can be done by using sc.exe [6] from Microsoft, and the specific approach goes like this:
Login as a non-privilege user, and run the following commands

```
C:\sc config "antivirus service" binpath= D:\attack\attack.exe
```

If error #5 does not appear, Congratulations!
“Antivirus service” is the name of the service the attacker want to exploit, and

binpath is a binary file (could be a Trojan, rootkit, or anything) under the attacker's control.

After running this command, the attacker might have successfully changed the binary path of a Windows service, which will result in the attacker's own binary being launched as an elevated privilege (usually SYSTEM privilege).

3.1.2 Driver IOCTL handler

Security researchers have uncovered numerous problems of drivers in 2007.

A major problem is the driver IOCTL handler issue. The problem is caused by insufficient address space verification within IOCTL handlers of device drivers.

Auditing the driver IOCTL problem can be done easily through fuzzing. Several Win32 IOCTL fuzzers are public available, such as ioctlizer [7].

Kartoffel from ReverseMode [8] is also a great tool which aims for testing the security and reliability of the drivers.

So what need to be done is to fuzz the drivers installed by antivirus software and investigate every BSOD (blue screen of death) carefully.

3.2 Auditing the ActiveX issues

Auditing ActiveX issues of antivirus software is no different from auditing them for other applications. It can be done either through fuzzing or through manual auditing:

- Fuzzing

  ActiveX-based vulnerabilities became more prevalent in 2007 than ever. This was partially caused by the prevalence of ActiveX fuzzers.
  Two popular tools in this area are AxMan [9] and ComRadier [10].
  AxMan is more powerful, while ComRadier is more user-friendly.
  After installing antivirus software, the specific ActiveX control can be fuzzed by either choosing a single CLSID or specifying a directory.
- Manual auditing

  While fuzzing can uncover lots of memory corruption problems, a manual audit can reveal some other interesting vulnerabilities at the design level. The unsafe-method issues for example.

  The Swiss-army knives of manual audits are Oleview [11], FileMon, RegMon, TcpView, [12] and Wireshark [13].

  OleView provides a higher-level view of information contained in the registry, and it features tree controls with friendly names. it can be used to check if a control is marked as safe or not, and enumerate the methods provided.

  FileMon and RegMon are also extremely useful. They can be used to check what kind of file operations and registry operations are happening when certain ActiveX controls are initiated, or when certain methods are called.

Is ActiveX control trying to read/write some files/registry keys specified by user controlled parameters?

Tcpview can tell whether an ActiveX has some network activities.
Does this ActiveX control listen to any TCP/UDP port?
Dose it try to connect to some IP address specified by parameters?

It's recommended to keep Wireshark running during the whole process.
It can tell whether an ActiveX control is trying to connect to some website, IP address, or whether it is trying to download some files from somewhere else and run them, or whether it is trying to upload some files from your computers to somewhere else.

3.3 Auditing antivirus software engine

The engine is the most complex component of antivirus software thus auditing the engine would be tough.

Basically there are three ways of auditing:

- Source code audit

  As the name suggests, to audit the source code, auditors need access to the source code of the antivirus software.

  However, for security researchers, most source code is unavailable, except ClamAV [14].

  ClamAV is a very good target for a source code audit. A quick glance at the vulnerability database shows that there are 49 single CVE matches for ClamAV.

  Needless to say, a source code audit is a time consuming job.
  There are several mature tools for source audits, including FlawFinder, RATS , ITS4, SPLINT, CodeScan, and Coverity.

  It can also be done by manually navigating source code and looking for dangerous APIs, integer calculations, memory operations, and logical errors.

- Reverse engineering

  Since most commercial antivirus solutions are closed-source, source-code audit are almost impossible for researchers.

  Reverse engineering is one of the best choices. Researchers can analyze the assembly code directly and look for potential vulnerabilities.

  While reverse engineering an antivirus software engine, the target should be focused on the component responsible for parsing all kinds of file formats.

  These components are usually implemented as independent plug-ins, here are two examples:
  Kaspersky: Arj.ppl base64.ppl cab.ppl lha.ppl rar.ppl, etc

Bitdefender: arc.xmd arj.xmd bzip2.xmd cab.xmd docfile.xmd, etc

Just like other reverse engineering jobs, auditing antivirus engines through reverse engineering is extremely time-consuming. The good news is that Hex-rays [15] (a plug-in for IDA pro) makes decompiling much easier! Reading the assembly can be as easy as reading source code.

Alex Wheeler has done a very good job in this area. For more information please check his Blackhat presentation “0wning Anti-Virus” [16]

- Fuzzing

Fuzzing is an amazing technique, and it has accelerated software security a lot in the past few years.

Researchers have already presented fuzzing media players, fuzzing server applications, fuzzing web browsers, and published lots of fuzzers. However, few people had ever talked about fuzzing antivirus software.

There is only one fuzzer for antivirus software published to the best of my knowledge. That is vxfuzz [17] published by Tavis Ormandy.
n.runs once mentioned that they had an in-house fuzzer named Fuzzer-Framework v1.0 [18].

Since most of the engine-based vulnerabilities exist in the decompression process, fuzzing antivirus engine means fuzzing various decompressed data and executables, this makes it much easier than fuzzing anything else, from my point of view, here is what is needed for the fuzzing:

A. A big hard disk
   This is because antivirus software has to deal with hundreds of different file formats, it’s better to fuzz all the formats supported. That is why a big hard disk is needed to store all files generated (by the fuzzer).

B. Debugger
   Choose a debugger (windbg is recommended for this job). Ollydbg and ImmunityDebugger are both good choices.
   The usage of the debugger will be introduced later.

C. Fuzzer
   Actually, implementing an antivirus fuzzer is quite easy. The fuzzer does not need to handle exceptions and launch the target application again and again. This is the major difference between fuzzing other applications (Web browsers, Media players, etc) and fuzzing antivirus software.

   Building a fuzzer by using a script language (either python or perl) is quick and fun. However, if there are hundreds and thousands of files to be generated, C would be perfect from the performance point of view.

   Actually, a fuzzer for antivirus software is just a file generator. For example, to generate files, a very simple antivirus fuzzer only needs to read samples, replace

samples from bytes to bytes with fuzzing strings, and then save them in the directory specified.

Choosing fuzzing strings is usually based on the experience of the designer. The fuzzing string should contain some magic value which might cause faults (integer overflow, stack-based overflow, etc), such as 0xFFFFFFFF, 0x7FFFFFFF, 0x0000, 0x00000000, 'B'*256, 0xCCCCCCCC.

One thing might need to pay attention to is the CRC checksum. For file formats such as RAR and ZIP, the antivirus software might check the CRC of the file (or a certain part) first, and if it does not match, no further process will be done.

Custom CRC function can be implemented within the fuzzer for certain file formats.

D. Good samples

Good samples are important to fuzzing. As antivirus software processes so many file formats, auditors need to collect dozens even hundreds of samples.

Try to Google out "filetype: extentions".
The samples could even be created manually. Thus, the auditor needs to collect all the software needed -- WinRAR, PowerISO, MakeCAB, and various PE packers (UPX, FSG, ASPack, etc).
.

Once that's ready, there are four steps to follow.

A. **Create test cases**

This can be done by using the antivirus fuzzer (file generator). The test cases generated should be saved on a specified directory (the big hard disk).

B. **Download and install antivirus software**

Download and install the antivirus software which is to be fuzzed.
Antivirus vendors usually provide a trial version on their websites, auditors can download and install it.
Do not forget to take a snapshot after installation.

C. **Scan!**

Launch the scan against test cases.
Do not forget to attach the favorite debugger to the scanning process of the antivirus software.
If it's difficult to determine which process it is, launch a scan and check the CPU usage to find out.

D. **Get some sleep**

The scan may take hours or even days, depending on how many test cases have been created.
Get some sleep. On waking up, check if there are any exceptions appearing on the debugger.
Each exception needs to be analyzed in depth and figure out whether it is exploitable.

3.4 Auditing management interface

- Client/Server management

  Most C/S-based management protocols are proprietary, which means, no RFC or documents are available. It will be difficult to understand what the Client and Server are talking about by capturing packets. The traffic might look quite random or might be encrypted in some way.

  Fuzzing is a good choice in this situation. Spike [19] and Sulley [20] are two great fuzzing frameworks.
  For more information, please check the reference.

- Web interface

  Since most Web servers for management are developed in-house by antivirus vendors, they may not be well audited and analyzed.
  Fuzzing is always useful and worth trying. There are lots of web fuzzers publicly available, such as webfuzz [21], Spike and Sulley.

By auditing antivirus software, the author has discovered several vulnerabilities in popular antivirus programs. Most of these were done through fuzzing.

AhnLab AV Remote Kernel Memory Corruption TrendMicro AV UUE Decoding Format String Vulnerability Avast! AV TGZ Parsing Heap Corruption NOD32 Heap Overflow (unpublished, 0day at the time of the writing)

A future rise of antivirus vulnerabilities in the future is expected.

4. EXPLOITING ANTIVIRUS

The techniques employed by attackers to exploit antivirus vulnerabilities vary from case to case.

4.1 Local Privilege Escalation

Local privilege escalation problems faced by antivirus software are no different from those faced by other software. The problems can be categorized as follows:

4.1.1 Weak DACL

The weak DACL problem has occurred in both the installation directory and installed services.

As far as the installation directory is concerned, vulnerabilities exist in the Access Control List (ACL) settings which will be applied during installation. When antivirus software gives “Full Control” permission to the “Everyone” group, anyone can modify installed files. Due to the fact that almost every antivirus software runs some system services, attackers are able to simply replace an installed service file with their own malicious code (Trojan or rootkit) which can later be executed with SYSTEM privileges.

This problem has been faced by almost all antivirus vendors, including but not limited to McAfee, Symantec, TrendMicro, VBA32, Panda, PC Tools, CA eTrust, ZoneAlarm, AVG, BitDefender, Avast!, and Kaspersky.

Case: CVE-2005-1107 *McAfee Internet Security Suite 2005 Insecure File Permission Vulnerability* [22]
McAfee Internet Security Suite 2005 uses insecure default ACLs for installed files, which allows local users to gain privileges or disable protection by modifying certain files.

As far as installed services are concerned, vulnerabilities are usually caused by the fact that the SERVICE_CHANGE_CONFIG permission is assigned to "Everyone". Attackers can exploit such a vulnerability to gain escalated privileges by changing the associated program. The attack can be achieved by using the SC.exe, which is published by Microsoft.

Here is an example:

```
C:\sc stop "vulnerable antivirus service"
C:\sc config "vulnerable antivirus service" binpath= D:\attack\attack.exe
C:\sc start "vulnerable antivirus service"
```

The weak DACL problems have become rare in the past few years.

4.1.2 Driver IOCTL Handler issues

There's a dramatic rise in driver IOCTL handler issues in the past two years. Security researchers and hackers moved to this problem in 2006, and plenty of vulnerabilities were quickly uncovered in security products, especially in antivirus software and personal firewalls.

Driver IOCTL handler issues are usually caused by insufficient address space verification within IOCTL handlers of device drivers installed by the antivirus software. Attackers can take advantage of this by overwriting arbitrary memory and then executing arbitrary code with kernel privilege.

Security researchers have successfully demonstrated how to reliably exploit these issues either by hooking some rarely used system call or by adding a call gate in the GDT (Global Descriptor Table). [23]

Cases:
CVE-2007-3673 *Symantec AntiVirus symtdi.sys Local Privilege Escalation Vulnerability* [24]
Symantec symtdi.sys before 7.0.0 allows local users to gain privileges via a crafted Interrupt Request Packet (Irp) in an IOCTL 0x83022323 request to \\symTDI\, which results in memory overwrite.

CVE-2007-0856 *Trend Micro Products IOCTL Handler Privilege Escalation* [25]
TmComm.sys 1.5.0.1052 assigns Everyone write permission for the \\.\TmComm DOS device interface, which allows local users to access privileged IOCTLs and execute arbitrary code or overwrite arbitrary memory in the kernel context.

CVE-2006-4927 *Symantec AntiVirus IOCTL Kernel Privilege Escalation Vulnerability* [26]

NAVENG.SYS and NAVEX15.SYS device drivers 20061.3.0.12 and later allow local users to gain privileges by overwriting critical system addresses using a crafted Irp to the IOCTL functions (1) 0x222AD3, (2) 0x222AD7, and (3) 0x222ADB.

CVE-2007-3777 *AVG Antivirus AVG7CORE.SYS IOCTL Handler Privilege Escalation* [27]
avg7core.sys 7.5.0.444 in Grisoft AVG Anti-Virus 7.5.448 and Free Edition 7.5.446, provides an internal function that copies data to an arbitrary address, which allows local users to gain privileges via arbitrary address arguments to a function provided by the 0x5348E004 IOCTL for the generic DeviceIoControl handler.

4.1.3 Race condition
The race condition vulnerability usually exists in antivirus software on the Linux/Unix platform. There are two cases:
CVE-2007-6595 *Clam AntiVirus Race Condition Vulnerability* [28]
ClamAV 0.92 allows local users to overwrite arbitrary files via a symlink attack on (1) temporary files in the cli_gentempfd function in libclamav/others.c or on (2) .ascii files in sigtool, when utf16-decode is enabled.

CVE-2004-0217 *Symantec AntiVirus Scan Engine For Red Hat Linux Insecure Temporary File Vulnerabilities* [29]
The LiveUpdate capability (liveupdate.sh) in Symantec AntiVirus Scan Engine 4.0 and 4.3 for Red Hat Linux allows local users to create or append to arbitrary files via a symlink attack on /tmp/LiveUpdate.log.

These vulnerabilities exist because temporary files are created in an unsafe manner. Attackers can exploit them by creating a symbolic link (from a critical file on the system to the temporary filename) to cause antivirus software to overwrite the symlinked file. This allows attackers to gain elevated access to the system.

4.1.4 Other problems
There have also been several other problems in the past. Here are some examples:

- Symantec *LiveUpdate*
  Symantec Liveupdate has a long history of being bothered by local privilege escalation problems. There have been nearly five vulnerabilities of Symantec Liveupdate thus far: SYM04-018, CVE-2003-0994, CVE-2005-2759, CVE-2006-1836, CVE-2004-0217. The problems vary from an un-trusted search path to Window launched as a SYSTEM privilege.

- CVE-2006-3223 CA Antivirus Scan Job Description Format String Vulnerability. [30]
  By creating a specially crafted description (containing a format string) of a scan job, attackers would be able to execute arbitrary command under SYSTEM privileges.

- CVE-2007-3800 Symantec AntiVirus Corporate Edition Local Privilege Escalation Vulnerability [31]
  Symantec Real-Time scanner did not properly drop privileges while displaying the notification window which contains information of threat found on a system.

This vulnerability allows attackers easily escalate their privilege.

4.2 ActiveX

ActiveX based vulnerabilities became more prevalent in 2007 than ever. ActiveX vulnerabilities have been published every day. Antivirus software is no exception.

ActiveX controls are usually installed during antivirus production installation, or when people try to use antivirus vendors' "Free online scan" services. Sometimes, it is also installed by download managers.

Basically there are two kinds of ActiveX problems:

- Insecure Method: A design error

  This problem is very common in antivirus software because it generally includes the functionalities of file operations (create, delete, and execute) and network-based operations (upload and download).

  Here are two examples:

  A. CVE-2006-3976 *CA eTrust AntiVirus WebScan Automatic Update Code Execution Vulnerability* [32]

     According to Matthew Murphy, who identified the problem:

     > A specific flaw exists during the automatic update process for the WebScan ActiveX component. WebScan allows the initializing web page to specify the location that the component will use to download and install updates through the 'SigUpdatePathFTP' parameter (and potentially the 'SigUpdatePathHTTP' parameter). It downloads the 'filelist.txt' manifest and acquires any update files it lists. There is no verification performed by WebScan to assure the authenticity of the information in the file list or the files themselves.

  B. CVE-2007-1112 Kaspersky Antivirus ActiveX Control Unsafe Method Vulnerability [33]

     This is another typical case of the insecure method vulnerability, which allows attackers to steal files from computers with the Kaspersky antivirus software installed.

     There's a method called StartUploading(). As the name suggests, attackers can specify which file to steal, and where to upload. [33]

```
Function StartUploading (
      ByVal strFilePath As String ,
      ByVal strFTPAddress As String ,
      ByVal strFTPUploadPath As String
   ) As Long
```

- Memory corruption

  The memory corruption problem of ActiveX controls in antivirus software is no different from that faced by other applications. Attackers construct malicious input (usually an over-long string) and pass it to vulnerable calls or methods as a parameter. Memory corruption will then occur, including typical stack-based

overflow, heap overflow, or some other memory modification issues.

Antivirus software has a bad record on this front too. A quick search reveals that Symantec, Authentium, RAV products have been vulnerable to these problems.

To exploit ActiveX-based vulnerabilities, attackers just need to create a specially crafted HTML file, host it on websites under their control, and then convince victims to visit it. Once victims with vulnerable ActiveX controls (vulnerable software) installed on their computers visit malicious websites, their computers are compromised.

4.3 Engine-related issues

The engine is the most complicated part of antivirus software and therefore most vulnerabilities exist in it.

However, exploiting antivirus engine-based vulnerabilities is also the most complicated and interesting part because they can be exploited in many ways. And actually, such exploitation is limited only by your imagination!

Because the engine is responsible for parsing hundreds of different file formats, it's very hard for it to make everything correct. That's why most engine-based vulnerabilities exist in file format parsing.

Basically, there are three kinds of vulnerabilities:

- Memory corruption
  This is the most dangerous problem of engine problems because it usually results in a full-system compromise.

  If the code of the engine was developed without security in mind, it would probably create a good chance for attackers to carefully craft files (executable, compression package, audio, document, etc) to induce a memory-corruption condition. While parsing these files, stack-based overflows, heap overflows, or other memory access/ modification problems would occur.

  Almost all mainstream antivirus products have encountered this problem in the past and many of them have faced it several times over.

  Alex Wheeler [34] and n.runs [35] are both great contributors to memory corruption-based vulnerabilities of antivirus software engine.

- Denial of Service
  There are basically two kinds of denial of service:
  A. CPU DoS problem
  A specially crafted file may be able to get the antivirus engine to run into an infinite loop, thus making the CPU usage reach a very high level, which is usually 100 per cent.

  Do you still remember the infamous ZIP bomb? [36]

The following hex dump (Figure 3) is the header of a CHM file, while scanning this CHM file with NOD32, the CPU usage will stay at 100 per cent.

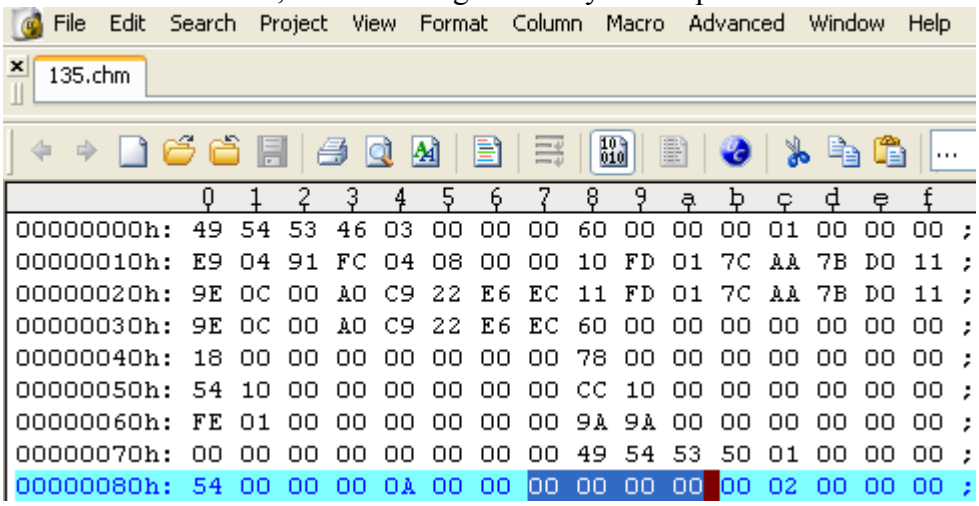


Figure 3: Denial of service POC for NOD32 (0day)

B. Disk space DoS problem

While processing a small file (less than 1KB for instance), antivirus software may eat up to 4GB disk space. This problem usually exists in the process when the antivirus engine is decompressing files. This is because the antivirus engine usually totally relies on the value read from files to allocate the disk space for corresponding decompressed files. And this value is probably manipulated by attackers.

A vivid example is the following ARJ file. While scanning this file, NOD32 will create a file NOD2.tmp which is 4Gb at:
C:\Documents and Settings\sowhat\Local Settings\Temp\NOD2.tmp

HEX dump: Figure 4

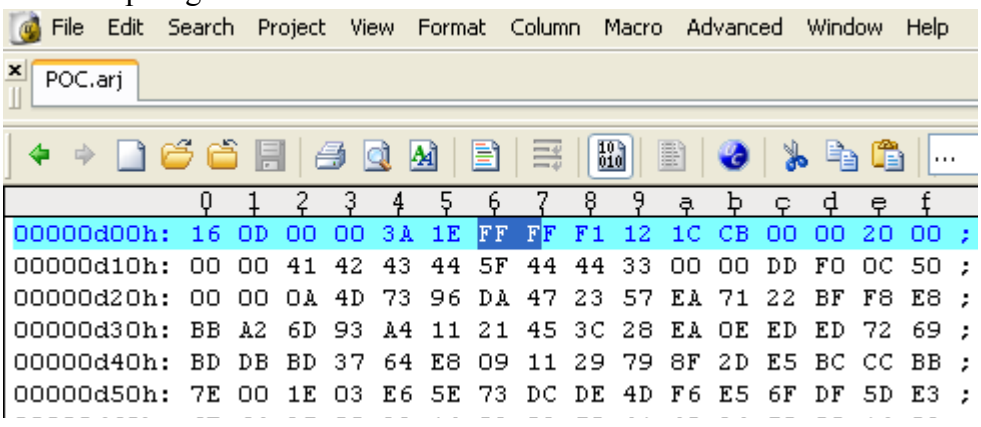


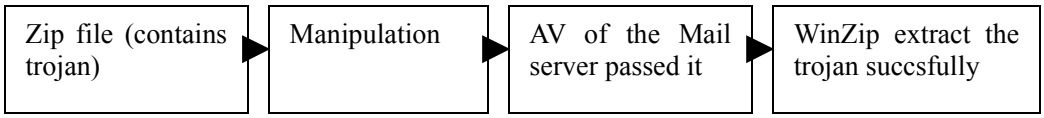
Figure 4: Denial of service POC for NOD32 (0day)

The severity of both CPU- and Disk-space-based denial of service may be considered to be low for desktop users. However, what if the same condition appears on a mail server with an antivirus engine scanning outgoing and incoming emails (attachments)? A malicious email will be able to leave the mail server in a very unstable condition, or completely out of services.

- Detection bypass

Detection bypass is a problem of relatively low severity. It is more important for server-side antivirus software than for desktop antivirus software.

Such attacks usually happen in the following situation:



Attackers will manipulate a zip file which contains a Trojan, and send it as an attachment of an email to victims.
After the manipulation, the antivirus engine of the mail server is unable to parse this zip file and will probably identify it as a legal file to let it pass. However, when victims get the email, the zip utility (WinZip for instance) would be able to process this zip file and extract the Trojan successfully.
Combined with some social engineering techniques, the victims may get compromised by executing this Trojan.

The engine based vulnerabilities can be exploited through various ways, including, but not limited to:

- Mail server
  Since most mail servers have configured some antivirus scan engines to scan outgoing and incoming emails, it creates a fantastic condition for attackers who have an exploit for the corresponding antivirus scan engine.

  Corporate networks may be armed with firewalls, IPS/IDS, antivirus and other security applications (software/hardware). Even though there are only two services open to attackers, one is web server of their website, and the other is the mail server. Attackers would be able to hack into the LAN by simply sending an email (with a malicious attachment) to the mail server.

  Here is a possible process:
  a. Manually search for the possible email addresses on Google, or use some email collection scripts such as emailcollect_v1.3.py [37]
  b. Send emails to collected addresses
  c. After emails reach mail server, they will be scanned by antivirus engine automatically. The scan engine will then be compromised because of vulnerabilities those emails (attachments) exploit, this will further result into a completely compromise of the mail server.

  Some of the mail servers install an antivirus scan engine by default, for example, IMail installs the Bitdefender antivirus.

  Kerio mail server fixed a "*possible buffer overflow in Visnetic anti-virus plug-in*" in the version 6.5.0. [38]

  The advantages of this kind of attack are,
  A. Attackers do not need to know any specific details of the internal LAN.
  B. The recipients do not need to open the malicious emails. They even don't need not receive them by using the email client.

  Real world case:

There's a real world incident reported recently [39], the attacker took advantage of the antivirus engine and compromised the mail server.

To summarize, Faas M. Mathiasen, a CISSP from Denmark reported that they noticed some odd pattern emerging from their mail servers, an important amount of data left their network over the mail server.

Their mail server is a fully patch Exchange 2007 with antivirus software installed. He is curious if there are any zero day exploit exists in the Exchange 2007, however, it turns out that:
"*Somebody using a"spoofed" email address send this file to a publicly disclosed email address and as soon as the scanner touched the file it triggered...* ***I thought I had watched a movie***".

- Email (client side)
  If attackers target individuals who have installed some vulnerable antivirus software, email might be a good choice. Here are the steps of the attack:
  A. Attackers send an email, which takes advantage of the antivirus engine vulnerability.
  B. The victim's computer will be compromised as soon as the email reaches his/her desktop, if the auto-protect of antivirus software is turned on.
  C. Even if the auto-protect is turned off, there are still possibilities that the victim would manually scan the attachment, especially when it looks really suspicious.

  Such an attack would most probably be launched as a limited attack.
  It is pretty similar to phishing attacks.

- Web
  It is not only possible to exploit ActiveX-based vulnerabilities over the Web, it is possible to exploit antivirus engine vulnerabilities too.

  To exploit the vulnerability of antivirus engine over the Web, IFRAME tag and .WMF file extensions would be very helpful.

  A typical attack scenario may be as follows:
  A. Attackers rename the exploit (say exploit.zip, which takes advantage of a ZIP parsing vulnerability of the antivirus) to exploit.wmf
  B. Hold a webpage which contains

     ```
     <iframe src = exploit.wmf>
     ```

  C. Convince victims to visit this webpage.
  D. While victims are browsing webpages, exploit.wmf would be downloaded onto the victims' computers automatically, without any user interaction.
  E. If the auto-protect of the antivirus is on, the antivirus engine would parse *exploit.wmf* automatically, and then possibly get compromised immediately.
  F. If the auto-protect is turned off, there would be still some more chances for attackers. Exploit.wmf is stored in the cache directory of the Web browser, and when a scheduled system scan (or manual scan) is launched, the antivirus engine might be shot.

- P2P/IM
  Exploiting vulnerabilities of an antivirus engine through P2P/IM is also possible. Files sent by friends over IM would be scanned automatically on receiving. Thus the antivirus software would be compromised right after the scanning.

4.4 Management

Most antivirus software has some management components for administrative purposes. These management components usually act as C/S mode. The server listens to some TCP/IP port and waits for connections, and the client actively makes outbound connections to the server.

- Client/Server management
  The client/server management components are usually in the proprietary protocol, which is developed by the antivirus vendor and can only be understood by themselves.

  A very good real-world case is CVE-2006-2630 [40], Symantec Antivirus Management Remote Stack Buffer Overflow.
  The remote management interface for Symantec Antivirus and Symantec Client Security is typically enabled and listens on TCP port 2967 by default. By sending a specially-crafted COM_FORWARD_LOG command, attackers would be able to trigger a typical stack-based buffer overflow and run arbitrary code under SYSTEM privilege.

  This vulnerability was later exploited by a variation of the infamous Spybot worm (W32.Spybot.ACYR, W32.Spybot.AMTE [40]).

- License management
  Some antivirus software includes a license management component for the license administrative purpose.

  The CA license software is a fantastic example. There were six vulnerabilities reported in it 2005 by iDefense Labs. [41]

- Web interface
  Web interface is another component for administrative purposes.

  In this case, the vulnerabilities can either be memory corruption issues caused by implementation errors, or security bypass issues caused by design errors.

  For both of them, the Symantec scan engine is a great reference, CVE-2005-2758 [42] for the first one and CVE-2006-0230 [43] for the second.

  CVE-2005-2758 Symantec Antivirus Scan Engine administrative interface Integer overflow [42]
  Integer signedness error in the administrative interface for Symantec AntiVirus Scan Engine 4.0 and 4.3 allows remote attackers to execute arbitrary code via crafted HTTP headers with negative values, which lead to a heap-based buffer

overflow.

CVE-2006-0230 Symantec Scan Engine Authentication Fundamental Design Error [43]
Symantec Scan Engine 5.0.0.24, and possibly other versions before 5.1.0.7, uses a client-side check to verify a password, which allows remote attackers to gain administrator privileges via a modified client that sends certain XML requests.

Exploiting Web-interface based vulnerabilities is the same as exploiting vulnerabilities of other Web servers (Apache, IIS), because most of them are light Web servers developed by antivirus vendors themselves.

5. CONCLUSION

In this paper, we have examined the techniques of finding vulnerabilities in antivirus software as well as the exploitation techniques.

We are not implying that antivirus is useless. Nor are we suggesting a replacement product. We only want to draw attention to the fact that the vulnerabilities of antivirus software are being a real threat.

Here are few words for antivirus vendors and end users:

For Vendors

Antivirus software puts too much trust on input files (files being scanned). Antivirus software should be developed and reviewed with the security in mind.

- Follow the SDL (Security Development Lifecycle)
- Audit your own products first.
- Fuzzing is incredibly effective
  - Fuzz before release. Fuzz and fix bugs before releasing them to the public
  - Fuzz after release. As the fuzzing techniques improve very fast.
- Follow Microsoft, Mozilla and others
  - Bulletin. Publish a bulletin and ask users to update ASAP.
  - Credit. Give credit to researchers.

For End Users

As stated earlier, end users have been putting too much faith in antivirus solutions, and have ignored the fact that antivirus software itself can be compromised.

People have scanned everything (applications, archives, documentation) suspicious in the past.
But now, it’s better to think twice before doing so.

6. FUTURE WORK

The security of antivirus software draws our attention to security products such as firewalls, IPS, IDS, and others.

Security products are supposed to protect users, what if they fail? What if they just open a new door for attackers in your system?

Future work shall focus on this aspect of 'security' products.

7. ACKNOWLEDGMENTS

The author is grateful to the following people for their insight, feedback, and technical opinions: Chi Zhang, Becky, Sanjay Pendse, Derin Mellor, Khushboo Shah, LinLin Zhao, Neeraj Thakar, Gary Kinghorn, Xin Ouyang, and Zheng Ren.

8. REFERENCES

[1] National Vulnerability Database
http://nvd.nist.gov/home.cfm
[2] Secunia 2007 Report
http://secunia.com/gfx/SECUNIA_2007_Report.pdf
[3] Internet Security Threats Will Affect U.S. Consumers' Holiday Shopping Online
http://www.bsacybersafety.com/news/2005-Holiday-Online-Shopping.cfm
[4] Executable compression
http://en.wikipedia.org/wiki/Executable_compression
[5] Data compression
http://en.wikipedia.org/wiki/Data_compression
[6] Sc.exe
http://technet2.microsoft.com/WindowsServer/en/library/0a658e97-51d5-4109-b461-a474c799964e1033.mspx
[7] ioctlizer
http://code.google.com/p/ioctlizer/
[8] Kartoffel
http://kartoffel.reversemode.com
[9] AxMan ActiveX Fuzzer
http://www.metasploit.com/users/hdm/tools/axman/
[10] COMRaider
http://labs.idefense.com/software/fuzzing.php#more_comraider
[11] Oleview
http://msdn2.microsoft.com/en-us/library/ms693754(VS.85).aspx
[12] Windows Sysinternals
http://technet.microsoft.com/en-us/sysinternals/default.aspx
[13] Wireshark
http://www.wireshark.org/
[14] ClamAV
http://www.clamav.net/
[15] Hex-rays
http://www.hex-rays.com
[16] 0wning Anti-Virus
http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-wheeler.pdf
[17] vxfuzz
http://my.opera.com/taviso/

[18] Antivirus (in)security
http://events.ccc.de/camp/2007/Fahrplan/attachments/1324-AntivirusInSecuritySergioshadownAlvarez.pdf
[19] Spike
http://www.immunitysec.com/resources-freesoftware.shtml
[20] Sulley
http://code.google.com/p/sulley/
[21] Webfuzz
http://www.fuzzing.org/wp-content/WebFuzz.zip
[22] CVE-2005-1107
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-1107
[23] Symantec Local Privilege Escalation Vulnerability Exploit
http://www.whitecell.org/list.php?id=50
[24] CVE-2007-3673
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2007-3673
[25] CVE-2007-0856
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-0856
[26] CVE-2006-4927
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-4927
[27] CVE-2007-3777
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-3777
[28] CVE-2007-6595
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6595
[29] CVE-2007-3777
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-0217
[30] CVE-2006-3223
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3223
[31] CVE-2007-3800
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-3800
[32] CVE-2006-3976
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3976
[33] CVE-2007-1112
http://www.zerodayinitiative.com/advisories/ZDI-07-014.html
[34] Alex Wheeler
http://secunia.com/search/?search=Alex+Wheeler+antivirus&w=0
[35] n.runs
http://www.nruns.com/parsing-engines-advisories.php
[36] Zip bomb
http://en.wikipedia.org/wiki/Zip_bomb
[37] emailcollect_v1.3.py
http://www.darkc0de.com/misc/emailcollect_v1.3.py
[38] Kerio Mail server 6.5.0 release
http://www.kerio.com/kms_history.html
[39] Possible Mail server compromise?
http://www.securityfocus.com/archive/75/488038/30/0/threaded
[40] Symantec AntiVirus Remote Stack Buffer Overflow Vulnerability
http://www.securityfocus.com/bid/18107/references
[41] CA License Software Multiple Buffer Overflow Vulnerabilities
http://secunia.com/advisories/14438/

[42] CVE-2005-2758
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2758
[43] CVE-2006-0230
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-0230