Scripting AV Signature File Updates and Testing

By Randy Abrams and Andreas Marx
With Mary Landesman as contributing editor

About the authors:

Randy Abrams
Operations Manager
Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, USA
http://www.microsoft.com
randyab@microsoft.com

Randy Abrams has worked for Microsoft since 1993. Over the past 7 years, Randy has been responsible for the virus scanning systems used in the final stages of the software release process to ensure clean code and to detect potential false positives. Past AVAR conferences have seen Randy present on topics such about testing anti-virus scanners and the use of multi-scanner systems.

Andreas Marx
CEO
AV-Test GmbH, Klewitzstr. 7, D-39112 Magdeburg, Germany
http://www.av-test.org
amarx@av-test.de

Andreas Marx has been involved in the antivirus industry since 1991. A WildList reporter since 1999, Andreas is currently the CEO of AV-Test.org where he has developed advanced methodologies for the testing of antivirus and security software. Andreas is also actively involved in virus analysis and security consultancy. In 2002 he earned his BSc in Business Information Systems from the Otto-von-Guericke-University Magdeburg and is currently working on his MSc.

Mary Landesman
Antivirus Guide
About, Inc., 249 West 17th Street, New York, NY 10011, USA
antivirus.guide@about.com

Mary Landesman is a security consultant and guide for the antivirus.about.com website. Her work has been published in various industry publications, including Virus Bulletin and Network Security magazine. She also performs spyware testing and is a contributing author to PC World magazine.

Abstract:

Multi-scanner systems have become increasingly prevalent. Anti-virus companies, test and certification organizations, software vendors, and corporate IT professionals are among the users and developers of systems that employ multiple anti-virus products. One of the challenges facing all of these users is the need to keep the anti-virus products current. The process of automatically obtaining updates requires verification of the signature files. Downloads may fail due to network timeouts, signature files may be corrupted, or new signatures may have defects preventing the detection of entire classes of viruses.

This paper will share techniques used by the authors to obtain and test anti-virus signatures files in existing multi-scanner systems. Functional examples include the use of Windows batch files techniques and may include utilities that are freely available to the user. The techniques presented are not claimed to be the best method, but represent methods currently in use. It is hoped that this presentation will encourage more active sharing of effective and best signature retrieval and testing practices between users of multi-scanners systems.

Why automate updates?

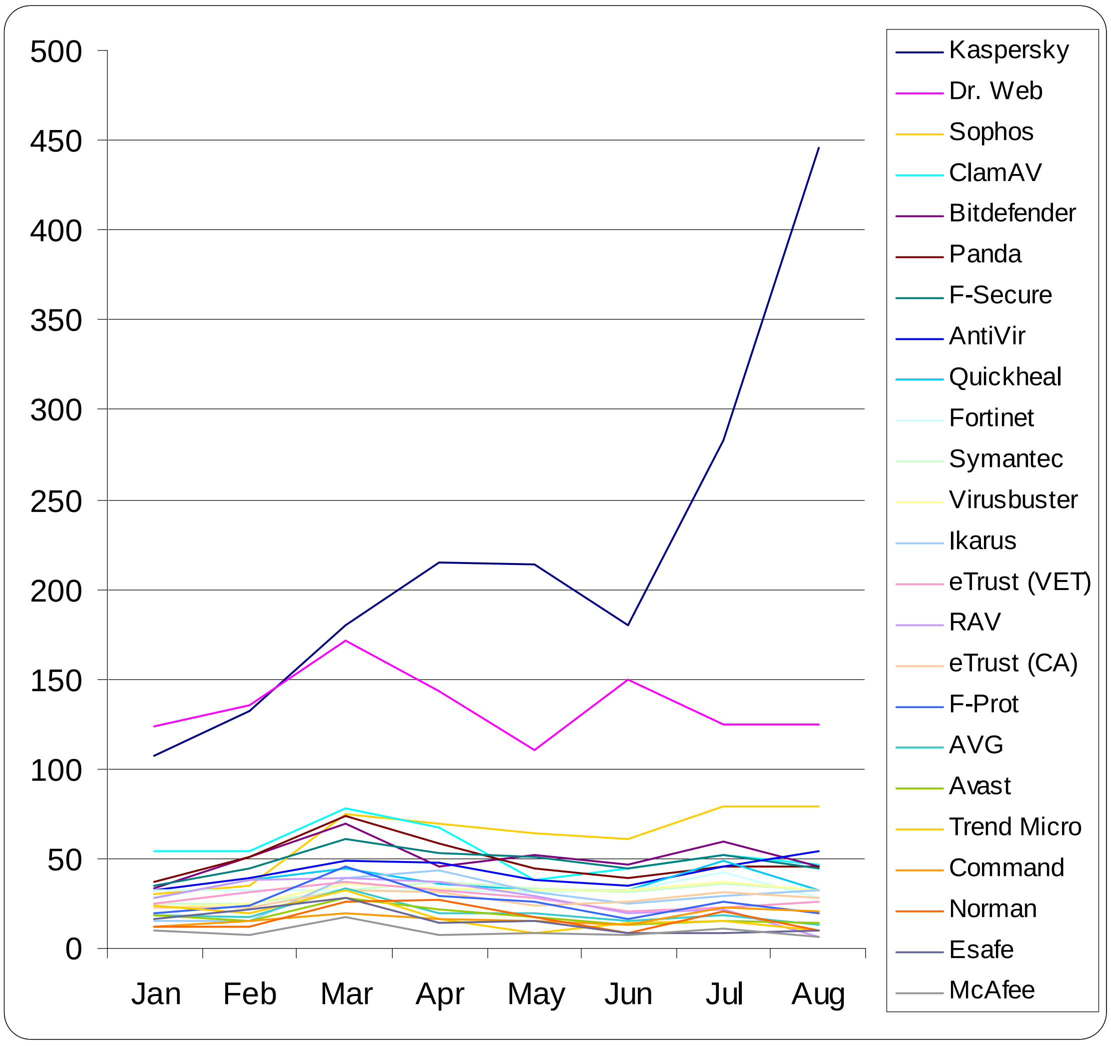
Both Andreas Marx, CEO of an independent anti-virus test organization, and Randy Abrams, a corporate user, have hundreds of terabytes of scanning experience with multi-scanner systems. Andreas has performed a wide variety of testing for anti-virus companies, magazines, and for the benefit of users in general. Randy specified and administered the processes used by Microsoft to ensure that the code that is shipped contains no viruses. Despite the different uses of the multi-scanner systems, both Andreas and Randy turned to scripting to enhance and automate their processes.

Almost every anti-virus product includes a mechanism to update itself. In corporations where desktop clients are not allowed to connect to the Internet, centralized corporate servers use built-in functionality provided by management tools included with the anti-virus product to obtain updates that are then provided for the desktop clients. The anti-virus companies even test their signature files prior to public release and many of the scanners test the signatures for corruption or tampering when the scanner is started. Why, with all of these advanced tools would anyone need to use scripts to obtain signature files manually?

- **Hardware constraints.** There are situations where the machines running the anti-virus products cannot be permitted to use the built-in updating systems. Physical isolation from the Internet, installation of multiple anti-virus products on one machine, and timing issues are all potential reasons why updating methods other than those provided by the manufacturer may be required.
- **Software release services.** Software developers must not only ensure their shipped code is virus-free, but also that there is minimal risk of false positives which can cause alarm and confusion among users.
- **Scan repositories.** Internet downloads and new media on CDs or DVDs may be subjected to multiple vendor scanners prior to installation or deployment. This may be to minimize the risk of a new threat or to overcome deficiencies that may be present in a single scanner (for example, not all scanners are really able to accurately scan inside InstallShield and MSI installation files).
- **Gateway/quarantine scanning.** It is common to employ policy management at the email gateway and quarantine policy violators, (e.g. blocking *.exe results in all .exe attachments being subjected to quarantine). New threats often spread via email, and response times dictate that one scanner may be able to detect what another cannot. With a multi-scanner system, you can check for possible viruses in the blocked files.
- **Antivirus vendors** may employ various scanners to check the names of the other tools (in order to sync the virus names and variant letters).
- **Layering.** Scanning with multiple vendor antivirus protection, if well chosen, can help bolster protection in otherwise weak areas. For example, one scanner may be particularly strong with detecting backdoors, keyloggers, and Trojans, while another may excel at ferreting out new Win32 worms (e.g. four eyes are better than one).
- **Quality assurance.** QA of virus signatures before deployment (e.g. if you have different antivirus engines on your gateways, firewalls, desktop and server systems)
- **Specialized testing**. Some kind of detection or outbreak response time testing (such as that performed by AV-Test.org)

Obviously, the number of reasons one may wish to employ multiple scanners -- and hence require a means of more effectively automating the updates of the selected scanners -- are as varied as the organizations who employ them.

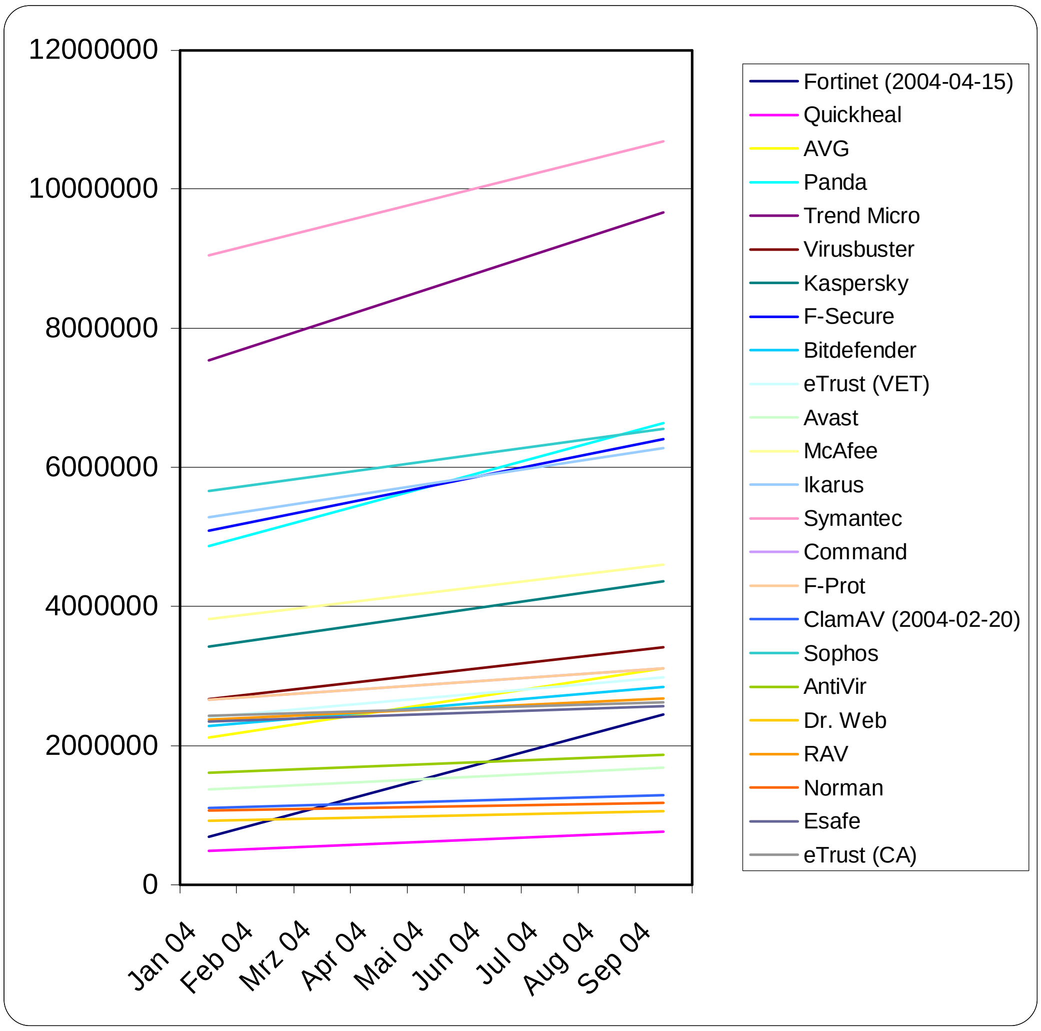
Frequency and number of updates



As seen in the chart above, frequency of updates can vary significantly between vendors. For the period from January 2004 through August 2004, McAfee released the fewest updates, averaging 9.6 updates per month with a total of 77 updates for the eight-month period. Conversely, Kaspersky Labs released the most updates: 1757 total or an average of 219.6 per month for the same period.

The following chart depicts the growth rate of the actual size of the definition updates (only virus signature files, without engine or program updates). On average, the total updates increased by 24.3% over the eight-month test period. The size of the definition files and projected growth rates are important considerations when defining the hardware requirements of the test system, to ensure adequate storage for the accumulated update files.

Note: The first time we have included ClamAV in this comparison was 2004-02-20 and we have started to track Fortinet updates as of 2004-04-15.



Obtaining the signatures

There are a variety of issues to work out when automating the downloading of virus signatures. At the top level there are FTP and HTTP downloads. These two types of methods are further divided into the categories of servers that require accounts and those that do not. After this level there are differences in approaches that the anti-virus vendors take to the construction of their updates.

- On the simple end of the spectrum we have products like McAfee and Trend that have download sites with zip files that are easily obtained, decompressed, and the files copied to the appropriate locations on the scanning machines.
- In the middle of the spectrum are products such as Kaspersky and Sophos that require different techniques depending upon the time of the month or year.

- On the difficult end of the spectrum are products that only update through the use of executable delivery mechanisms, often packaged using formats that the scanners themselves do not understand well enough to scan prior to executing.

While ftp downloads are possible, and not difficult using built in functionality provided with any Windows NT based operating system, there are some external tools that are required or simply make life easier when managing multiple anti-virus products. These tools include WGET, a hash or CRC program, and an unzip program. WGET is a utility that simplifies both FTP and HTTP downloads, with or without account names and passwords.

An example of a simple signature download using a batch file follows. *Italics* are used to indicate comments that are not part of the script.

-------------------------McAfee Update Batch File---
Using the for command we are going to parse the date and time environment variables to use in the files names of our downloads. This enables easy archiving!

```
for /f "tokens=1-4 delims=/ " %%i in ("%date%") do set mydate=%%j-%%k-%%l-
for /f "tokens=1-4 delims=:. " %%i in ("%time%") do set mytime=%%i_%%j_%%k-

C:
cd \download\Archives\$mcafee\update\
```

McAfee has weekly updates, as well as hourly beta updates. This section checks for the released (weekly) updates. We are first going to download a readme text file because it contains the version of the weekly dat file. Next we use the for command and findstr to parse the text file and obtain the information we are looking for.

```
::weekly  the :: at the beginning of the lone makes it  a comment.
wget ftp://ftp.nai.com/pub/datfiles/english/readme.txt
for /f "tokens=4 delims=: " %%i in ('findstr /i /c:"- DAT version:" readme.txt') do (
  if not exist dat-%%i.zip wget  ftp://ftp.nai.com//pub/datfiles/english/dat-%%i.zip
)
del readme.txt
```

This section obtains the beta dat files. This also uses errorlevel checking and a counter to retry on failure

```
Set dailycount=0
:daily
wget http://download.nai.com/products/mcafee-avert/daily_dats/DAILYDAT.ZIP
if  %errorlevel%==0 goto :gethash
set /a dailycount=%dailycount%+1
if %dailycount% GTR 3 del dailydat.z* and exit
```
use if /? At a command prompt under NT for complete syntax of the if command
```
goto :daily

:gethash
```

We're going to hash the file and set an environment variable to the value of the hash.

```
for /f %%i in ('md5 dailydat.zip') do set newmcafee=%%i
```

Next we compare hashes of last update to current update (old mcafee.bat sets an environment variable to the value of the last new download).

```
if exist oldmcafee.bat call oldmcafee.bat
```

If the hashes are the same we delete the new download and exit.

```
if /i ["%oldmcafee%"]==["%newmcafee%"] del dailydat.zip & exit
```

If the hashes are not the same delete oldmcafee.bat and make a new oldmcafee.bat file.

```
if /i not ["%oldmcafee%"]==["%newmcafee%"] (
del oldmcafee.bat & echo (set oldmcafee=%newmcafee%) >oldmcafee.bat
)
if /i not ["%oldmcafee%"]==["%newmcafee%"] (
ren dailydat.zip %mydate%%mytime%dailydat.zip && echo newfiles>newfiles.txt
)
exit
```

-----------------------------End McAfee Update Batch File--

McAfee updates are relatively easy to automate. Kaspersky Labs employs a structure that allows for updates that contain a small amount of data, as frequently as several times a day, a bit larger signature set every week, and a full signature replacement 4 times a year. This technique is efficient, but requires sets of files to be perfectly synchronized. It is very easy to download everything every time, but this is not efficient for those who pay for their bandwidth or have slow connections. The next example demonstrates some scripting techniques available using the Windows command shell.

The key to minimizing data transfer is only downloading what is needed. In order to do this one must determine what files are required and what files are already present. The file AVP.KLB, which is included in the Kaspersky updates, provides the means to accomplish both parts of making the determination of what is needed. AVP.KLB has a list of all of the signature files that are needed as well as a corresponding digital signature for each file. By comparing the list of file names to the actual files already on the system, and then comparing the digital signatures in the list to the digital signatures of the files on the system we are left with a list of files that are needed because they do not exist or because the digital signature has changed -- This indicates a new file. If any of the files already on the system are not in the AVP.KLB then they are no longer needed on the system.

-------------------------Kaspersky Update Batch File---
Of course, we want to ensure that we are in the correct directory to begin with.

```
C:
cd \KAV
```

*Initializing variables is not generally required in a batch file, but it is a good programming technique. In this example, if an error occurred and the batch file was restarted in the same command prompt, failure to initialize the variables could result in failure. Any commands in batch files can be enclosed in curved brackets. When using the "**SET**" command the curved brackets prevent blank spaces from being included in the variable value. Multiple commands can be written on the same line by joining them with the "&" symbol.*

```
(set avc_loop=) & (set klb_loop=) & (set newavc=) & (set oldavc=) & (set success=false)
```

In addition to initializing variables it is important to use new versions of files so the next section includes obsolete files are no longer present.

```
if exist c:\kav\temp\avp.klb del c:\kav\temp\avp.klb
if exist c:\kav\temp\newklb.txt del c:\kav\temp\newklb.txt
if exist c:\kav\temp\newavcfiles.txt del c:\kav\temp\newavcfiles.txt
if exist c:\kav\temp\avp.set del c:\kav\temp\avp.set

cd c:\kav\temp
```

Get_File is a subroutine wrapper for a 3 try download system with error checking. The Windows NT/2000/XP/2003 command shell allows for calling subroutines in batch files.

```
call :Get_File avp.klb
call :Get_File avp.set
```

What files have changed? We'll find the AVC names and locate their signatures in the avp.klb. Findstr is used with the "/c" switch to generate a list of the signature files with their hashes.

```
findstr /c:".avc" avp.klb >>c:\kav\temp\newklb.txt
```

The list of files and signatures needs to be parsed so as to generate a list of filenames and a list of digital signatures. This will require parsing. When parsing files the "=" character is problematic. The "=" character cannot be used as a delimiter in the for command. The scripting to work around this problem is fairly complex, so to make the job a bit easier an external tool (rep.exe) is used to replace the "=" character with a comma.

```
C:\UTIL\REP.EXE = , c:\kav\temp\newklb.txt
```

The previous line changes entries in the AVP.KLB file that look like this:

```
0=kernel.avc,0XLSznpdl71fB300e7Uwj1TdKc+yC8rcaj94jH6EUL9UuOmxXyKkBjNdD+,8445,09/09/2004
```

And writes the line below to a file called "NEWKLB.TXT".

```
0,kernel.avc,0XLSznpdl71fB300e7Uwj1TdKc+yC8rcaj94jH6EUL9UuOmxXyKkBjNdD+,8445,09/09/2004
```

Another subroutine is used to compare the signature name and corresponding digital signature value to the current signature's value. The subroutine is called and the name and digital signature are passed to the subroutine as arguments .The arguments "%%b" and "%%c" correspond to the second and third tokens in each line in the file "NEWKLB.TXT".

```
for /f "tokens=1-5 delims=," %%a in (c:\kav\temp\newklb.txt) do call :compare1 %%b %%c
```

The subroutine will generate a list of the files that need to be downloaded. If no such list has been generated the next line will take us to the end of the batch file where some final steps are performed. The double "&&" tells the command shell to only perform the next command on the line if the previous command succeeded.

```
if not exist c:\kav\temp\newavcfiles.txt (set success=TRUE) && goto :finalsteps
```

We need to download new files if we are here. The AVP.set file is needed for the scanner to properly function. Once again the "Get_File" subroutine is invoked to download the list of files that are required. The "/f" switch tells the for command how to process the portion between the curved brackets. In this case the file is parsed line by line.

```
move avp.set C:\KAV\newavcs
CD C:\KAV\newavcs
for /f %%i in (c:\kav\temp\newavcfiles.txt) do call :Get_File %%i
```

The "workingset" directory will be used to store our entire current signature set. The first time the batch file is run all signatures will be required and will end up in a directory called "oldavcs" as well as in "workingset".

```
del /q C:\KAV\workingset
move /y *.avc C:\KAV\oldavcs
```

The following "FOR" command could be written on one line, but it would word wrap in this text. The command shell allows for this command to be written on multiple lines. The command that follows adds files we need and already have from the "oldavcs" folder into the "workingset" folder.

```
for /f "tokens=1-2 delims=," %%i in (c:\kav\temp\newklb.txt) do (
if not exist C:\KAV\workingset\%%j copy C:\KAV\oldavcs\%%j C:\KAV\workingset
)

if exist c:\kav\temp\oldklb.txt del c:\kav\temp\oldklb.txt
copy c:\kav\temp\newklb.txt c:\kav\temp\oldklb.txt

copy /y avp.set C:\KAV\workingset
del /q avp.set
(set success=TRUE)
goto :finalsteps
```

The arguments passed to the subroutine are the name of the file (%1) and the digital signature associated with the file (%2).

```
::----SUBROUTINE COMPARE1----
:compare1
set newavc=%2
```

The "IF" command can also be written on multiple lines. The line above sets an environment variable to the value of a file's digital signature. The next section sets another variable to the value of the digital signature for the copy of the file that is currently on the system. If the values are the same the variables are reinitialized and the "goto :EOF" causes the subroutine to exit and return to the line following the call to the subroutine. If the values are different then the file name is written to a text file that will become the list of files that must be downloaded.

```
if exist c:\kav\temp\oldklb.txt (
for /f "tokens=1-3 delims=," %%i in ('findstr /c:"%1" c:\kav\temp\oldklb.txt') do (
(set oldavc=%%k)
)
)
if ["%newavc%"]==["%oldavc%"] (set newavc=) & (set oldavc=) & goto :eof
::We only get here is the .AVC file is different so we generate a list of files we will need.
echo %1 >>c:\kav\temp\newavcfiles.txt & (set newavc=) & (set oldavc=)
goto :eof
::----SUBROUTINE COMPARE1----
```

The get file subroutine uses an external utility called sleep.exe. A simple Visual Basic Script can also be used for this functionality. This routine deletes a file that we want to download if it already exists before we try to download it. The purpose of the loop is that an on access scanner can have the file open on the initial access.

```
::----Begin SUBROUTINE Get_File----
:Get_File
(set avc_loop=0)
(set del_loop=0)
:loop1
(set /a del_loop=%del_loop%+1)
if exist %1 del %1
if not ["%errorlevel%"]==["0"] (
```

The first time through this loop the value of del_loop is 1, the next time it is 2, etc.

```
if %del_loop% GTR 2 echo Can't delete %1 >>C:\KAV\update\download_problem.txt && goto
:finalsteps
goto :loop1
)
(set /a avc_loop=%avc_loop%+1)
wget -aC:\KAV\kavwget.log -t1 -w3 -T360 ftp://ftp.kaspersky.com/updates_x/%1
if ["%errorlevel%"]==["0"] goto :eof
```

```
sleep 30
if %klb_loop% LEQ3 goto :loop1
echo %Date% - %time% - Error on %1 download >> C:\KAV\update\download_problem.txt
goto :finalsteps
::----End SUBROUTINE Get_File----

:finalsteps
if /i not ["%success%"]==["True"] echo %date% - %time% >>C:\KAV\results\fail.txt
exit
```
-------------------------------------Kaspersky Update Batch File-----------------------------------

Why test the signature files?

Many anti-virus companies provide beta signatures that have not completed normal testing. These signatures are provided to those who are willing to accept the risk of signatures that have not finished quality testing in exchange for faster detection. However, even when the signature files that have completed quality testing by the anti-virus companies are used, testing can be prudent. As with all software products, the signature files may not work. Such problems may be due to:

- Corrupted virus signatures (e.g. broken archives, interrupted downloads), so the files are not useable (e.g., the "transfer process" not working properly)
- Download of "old" signatures due to problems with the synchronization process of the different download servers within the antivirus company
- Bad or corrupt updates (e.g., the "creation process" was erroneous)

Statistics provided by AV-Test.org at the Virus Bulletin 2004 conference indicated that of 37,000 updates obtained, approximately 7,000 were non-functioning.

Testing the signature files
Checking the integrity and validity of signatures files begins with a few trivial steps and gradually progresses in complexity.

1. Check whether the size of the files you've downloaded matches the usual update size. For example, if the usual download size if 5.5 MB, but the file you've downloaded is significantly smaller, it could be invalid. Of course, it might also be that the internal structure of the signature files has been optimized or changed. However, an update with zero bytes or less than 1 KB in size is definitely "bad".
2. Unzip an archive you've downloaded and determine whether there are any errors (i.e. any error messages or returned error codes?). If all files inside of the archive can be unzipped without problems, the transfer process appears to have been sound.
3. In some cases (i.e. Symantec), the update is stored inside of a self-extracting EXE archive. If so, you can check whether the file is really an EXE file (first bytes are "MZ"), the size (no Symantec update should be smaller than 5 MB, for example) and after you've started it, you can check for the return codes (everything alright?)
   Note: Due to the fact that you've downloaded an executable file, it might not be a bad idea to scan this EXE for viruses with other scanners first before execution. One would expect,

that the updates are clean (non-infected), but nobody can always be 100% sure that everything is OK! Though it's rather unlikely that an antivirus company will be infected, criminals can possibly spoof websites with low-level attacks (e.g. ARP poisoning) against you and you will download “their” update now instead of the antivirus company's one. This means, you will execute their code now... (Digital signatures might help to avoid such problems, but almost no antivirus updates carry a real digital certificate at the moment!)

4. After this, start the command-line scanner to see if the definitions are loaded properly. For this, you should check a few non-malware files where you know that the exit code should be 0 (e.g. simply scan the folder where the current antivirus scanner is located, e.g. “fooscan”) **à** and if the exit code is 0, you know this part is working properly.
5. You should also look at the version number of the def files (it should be easy to gather these ones from the report files) -- if they are less than the ones you have used for the last scan, something seems to be wrong. For example, the update servers of the antivirus company is out-of-sync and in one moment you access an old file and a few moments later, you see a new file and just a few seconds later, the old file again (this happens quite often!)
6. Next, you should test if the exit code works well in case of virus found, too. For this, you should scan an eicar.com, for example, to see if the scanner reports it properly (check the report file for the virus name) and the exit (error) code matches the usual one. This way, we have performed two checks: if the virus name matches the old one (e.g. if “EICAR_TEST” is still called “EICAR_TEST” by the program -- think about possible log file format changes!) and if the return code is still the same **à** if one of these two conditions are not fulfilled, something is wrong and should be reported!
7. If the AV update passes these checks, we know that they seem to work properly already, but this might not be enough, because there can still be (non-) detection problems!

Testing with live viruses
Real viruses (at best, renamed and/or packed inside of archives to avoid accidental infection) might be used for the second part of the test process to see if the different essential parts of the AV engine are working properly. However, it should be considered only when other methods (i.e. consulting qualified testing centers, EICAR, etc.) have been exhausted. Certainly a case can be made that the use of multiple scanners negates the need for live virus testing. In such cases, the odds of detection are much better for detecting a known virus if a scanner has a problem with a particular type of virus than if only one scanner is being used.

Assuming a need to test with live viruses has been established, it then becomes prudent to select the specific types of malware best suited to ferret out problems with detection that are most likely to occur. To accomplish this, it may be suitable to include some malware samples that are seldom seen today but which were common 5-10 years back. This is useful because few antivirus companies test these older samples carefully and thus errors are much more likely to occur.

Examples for such a test set:

- Tremor and One_Half as DOS viruses that are polymorphic (thus, the 16 bit code emulator will be tested)
- Jerusalem and Tai-Pan as DOS viruses that are static (thus, the plain signature scanner for 16 bit code will be inspected)

- Office 95 macro viruses such as XM/Laroux and WM/NOP (to test the Office 95 macro routines of the scanners which are different from the ones Office 97 and above are using)
- W97M/Melissa and X97M/Barisada (to test the Office 97+ macro engines)
- Win32/CIH and Win32/Lovesong (to test the plain Win32 scanning routines for “real” viruses)
- Win32/Magistr and Win32/Fono (to test the Win32 code emulator)
- Win32/Bagle and Win32/Netsky (to test the Win32 scan routines for worms which are usually a bit different from the ones used by AV software to detect file-infecting viruses)
- VBS/Loveletter and VBS/VBSWG (to test the text/script parsing routines of the AV engines)
- You may want to include some more special files, e.g. a malicious JPEG file which contains the MS04-028 exploit, to even test more internal program structures of the AV tools

Carefully selecting the samples allows you to test with a much smaller collection. In the example test set above, many different routines inside of the scanner have to be used in order to detect all of these malware thus no “big” virus collection is needed.

The parsing routines of your scripts can also be tested at the same time; hence you will know what the results should look like and if they are different (e.g. because of changes in the file structure of the log file, name changes or non-detection of a few files).

For those truly cautious, it could make sense to test both the last and the current signature from a given product at the same time, allowing you to see subtle differences (e.g. new detections or changed virus names) easily and minimize the risk of a “bad” signature updates. (Of course, the risk of non-detections should already be low, if you are using multiple scanners from different antivirus companies, because if one fails, you still have several other working ones).

Note: If you think the newly downloaded definition files won't work properly, simply continue to use the older definition files previous downloaded. It's also a good idea to store all files (regardless, if corrupted or not) inside of a file archive (some kind of database). This makes it easier to reproduce possible problems, provides an historical marker of detection capabilities, etc.

Setting up a scan system

The system should be well-protected (e.g. behind a firewall) and only be accessible by named persons -- think about the fact that viruses (whether quarantined or for testing) may be stored on this system, too -- and nobody from the outside should be able to access these files. Furthermore, if you check your new software releases using this system, third parties should be unable to “steal” the code (software piracy).

The multi-scanner system has to be connected to the Internet in some ways or you won't be able to update the scanners. It may be a good idea to have one system running for the update process only (which has Internet connection to some websites) and a second system (or even more) for the scan process that does not have any kind of Internet access.

The PC downloading the signature updates should have real-time or on-access scanning enabled. If this PC also is testing the signatures prior to deployment then you will need to programmatically disable the on-access service during the test. You may want to store the virus samples in a password protected zip and extract the files when testing, deleting the extracted files at the end of the test so as not to leave executable viruses laying around unprotected.

For the PC that does the scanning and has no Internet access, it is not advisable to leave a file share open on it. Instead, pull the files from a known location and leave the actual scanner as locked down as possible.

It's also good idea to generate as many logs as possible (for debugging, checking for errors etc.) and of course, in order to certify that the scan process was working properly (e.g. all files were scanned by all programs). For this, a tool which is able to generate MD5sums of all files should be included in the AV scan process, too: first, it generates a log of all files which should be scanned, including their MD5sums (and maybe date/timestamps, plus size) and in the second part, the scanner log files are included (maybe with the option to list all scan files to be on the safe side). This way, the admin of the multi-scanner system can proof that their system was working properly and the user can see if really all files were scanned which should be scanned and for third parties this process is much more transparent, too: the company can easily proof that to the best of their knowledge, the software they are using or releasing was not infected.

Scripting the testing of signatures is fairly straightforward as long as the anti-virus product meets the following criteria:

- The scanner must be able to be launched from the command line.
- The scanner must automatically exit at the end of the scan.
- The scanner must return a unique error code for a clean scan and a distinctly different return code for a scan that indicates a virus.

Scanners that return distinct error codes when conditions arise that are not normal, but are not the indication of an infection either, provide much better feedback to the tester. An example of such a scanner would be Norman Virus Control. If the scanner from Norman determines that a file it has scanned is corrupted, but not infected, then a unique error code is returned. This is extremely valuable information. If known good files are scanned and the indication of a corrupt file is returned then the test has indicated a problem with the signatures that may not have otherwise been indicated.

Most scanners unfortunately simply report a clean result if an empty directory is scanned. There are very few that return a unique error code for the case where nothing was scanned. Some scanners will return the same error code if a minor error occurs as if there were no errors at all. A minor error may be indicative of a major problem. To test completely the log files would need to be parsed. The methods and examples of log file parsing are a subject that could fill several 40-minute presentations.

The following is a minimal example of testing the signature files. The command lines are simplified for demonstration purposes and are not intended to represent the proper configuration for any specific environment. Only the EICAR test file is used in this example. The EICAR test file is the best way to balance test validity with safety and ease of use. There are meaningful tests that can be performed with live viruses provided the tester has knowledge of the proper handling of malware and properly isolates and safeguards the environment.

It will be assumed that the dat files were downloaded to c:\scanners\mcafee\archives, will be stored at c:\scanners\mcafee\test while being tested, and that when ready to deploy the files will be placed at c:\scanners\mcafee\update.

---------------Begin McAfee Dat Test------------

First we'll test the zip package to see if it is ok.

```
wzunzip -t c:\scanners\mcafee\archives\dailydat.zip
if not "%errorlevel%"=="0" set problem=bad_zip_file & goto :abort
```

Next we make sure the test folder is clean and extract the dat files.

```
Del q c:\scanners\mcafee\test\*.*
wzunzip c:\mcafee\archives\dailydat.zip  c:\scanners\mcafee\test
if not "%errorlevel%"=="0" set problem=bad_zip_file & goto :abort
```

This routine deletes a file that we want to

```
Copy /y c:\scanners\mcafee\test\*.* c:\mcafee

C:\mcafee\scan /sub /secure /unzip /report c:\mcafee\logs\cln.log c:\test\cln\*.*
If not %errorlevel%==0 set problem=FP & goto :abort
C:\mcafee\scan /sub /secure /unzip /report c:\mcafee\logs\vir.log c:\test\dirty\*.*
If not %errorlevel%==13 set problem=no_detect goto :abort
C:\mcafee\scan /sub /secure /unzip /report c:\mcafee\logs\vir2.log c:\test\dirty c:\test\cln\
If not %errorlevel%==13 set problem=no_detect goto :abort
```

If we are here it worked

```
del c:\scanners\mcafee\update\*.dat
copy c:\scanners\mcafee\test\*.dat c:\scanners\mcafee\update\*.dat
exit

:abort
```

Here we decide how we want to deal with the failure. The %problem% variable will tell us why there was a failure. Command line email programs can be used to send notifications, or a log that it monitored regularly can be used. There are also other options. For now we will simply write a log.

```
Echo %date% %time% - %problem% >>c:\scanners\logs\Mcafee_Update.log
Exit
```

---------------End McAfee Dat Test------------

Conclusion

Response time testing performed by AV-Test.org during the eight-month period from January 2004 to August 2004 of specific high-profile threats indicates an average response time of 10 hours. Some products were much quicker (for example both BitDefender and Kaspersky had average response times of less than 4 hours) where others were much slower (both McAfee and Symantec had average response times of over 14 and 16 hours respectively). Properly selected, multiple scanners increase the odds that one or more of the scanners in use will be updated more quickly and thus afford better protection during the critical hours of outbreak.

Likewise, a particular scanner may not be equipped to deal with a certain type of threat. For example, detection for the recent JPEG processing exploit still has not been added by all antivirus vendors; there are similar discrepancies when dealing with certain types of archive or compression formats.

By properly scripting the update process and effectively testing both the integrity and detection of the selected scanners, the use of multiple scanners is quite manageable. In return for the initial investment of proper preparation, more robust and well-rounded protection can be achieved efficiently.