# GIVING THE EICAR TEST FILE SOME TEETH

*Randy Abrams*

Microsoft Corporation, One Microsoft Way, Bldg 12N/1285, Redmond, WA 98052-6399, USA

There are situations that warrant the use of live viruses. *Virus Bulletin*, CheckMark, the *ICSA* and other review and certification bodies have a legitimate need to use live viruses in their processes. There are also situations where the use of live viruses is unwarranted. Specifically, live viruses should not be used when safer and equally effective methods can be used to obtain the required information. I will endeavour to show you how you can enhance your anti-virus product selection processes, learn more about your anti-virus scanner's capabilities, and verify your anti-virus product installation and configuration – all without the use of a single computer virus.

The miracle tool is a mere 68-character file. This file will not infect or otherwise wreak havoc on a computer, even if improperly used or deployed at home without adult supervision. The tool is none other than the *EICAR* test file.

Here is the *EICAR* test file:

**X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H***

The *EICAR* test file was created to provide users with a means to verify that their anti-virus product is installed correctly. *EICAR* is actually a functioning COM file and when run in DOS displays the message: EICAR-STANDARD-ANTIVIRUS-TEST-FILE!

Many companies that made up the anti-virus industry agreed to search for this string when performing a scan. If the string is found, typically at the beginning of a file, the scanner will react just as if it has come across a real, live virus. The specification found on the *EICAR* Web site states that these 68 characters should be the first 68 characters in the file. Most virus scanners will ignore this string if there are any other characters preceding it. I know of one scanner that will detect it with only the first 30 characters of the test file, and depending upon the configuration of the scanner, it does not seem to care if the *Windows NT* Operating System Kernel is in front of the string!

From now on I will refer to the *EICAR* test file simply as *EICAR*. *EICAR* is actually the *European Institute For Computer Anti-Virus Research*, but within the context of this presentation, referring to the test file as *EICAR* should not cause any confusion.

Before we go too far, it is important to understand what *EICAR* is not, and what it will not do. This is important because as you use *EICAR* for purposes other than which it was probably intended, you do not want to be deceived into believing you are learning something which you are not.

*EICAR* is not a virus, it will not replicate. *EICAR* cannot give you any meaningful data with respect to how effective a virus scanner is at detecting, mis-detecting, or disinfecting viruses. Throughout the years there have been discussions about *EICAR* detection with respect to the specification. It is not precisely detailed by the *EICAR* specification as to whether or not detecting the *EICAR* string in the middle of a file is a false positive. Some respected anti-virus researchers will argue vehemently that it is a false positive if there are one or more characters in front of the *EICAR* string and a product detects it as *EICAR*. Even if we accept this point of view as fact, there have been no studies, that I am aware of, that have established a correlation between products which false positive on *EICAR* and a higher overall false positive rate than scanners which only detect *EICAR* when it is the first 68 characters in the file. *EICAR* detection, wrong or right, does not tell you how likely a product is to false positive.

An *EICAR*-compliant virus scanner with four-year-old virus signature files can detect *EICAR*. The same scanner will probably miss every macro virus in existence. The ability to detect *EICAR* cannot be interpreted as a sign of poor, fair, good, or excellent detection capability. *EICAR* does not tell you how current your virus signature files are. The only chronological information you can glean from *EICAR* detection is that the scanner's signature files are newer than the existence of the *EICAR* test string.

There is one additional *caveat*. At least one product uses individual files for different types of virus detection. Macro virus signatures are in one file, Trojan signatures are in another file, standard virus signatures are in another file, and *EICAR* is in its own file. For this product, and any product like it, detection of the *EICAR* does not meaningfully indicate that any of the other signature files exist, or are active. It is important to keep this in mind when interpreting the results of an *EICAR* test.

Detection of the *EICAR* string proves only that the virus scanner under test is currently configured so that it can detect the string in the specific file under test. All other information must be inferred. For this reason it is important to construct your tests carefully and to limit conclusions to those that are supported by the test. For example, if I create an *EICAR* file, which I name EICAR.LZH, and the scanner under test detects the string, there are two conclusions that are warranted, and at least one which would be erroneous. The two valid conclusions are that the scanner is *EICAR*-compliant and the scanner is configured to scan files with an LZH extension. Even this conclusion concerning the extension is contingent upon my scanning a directory, as opposed to selecting the specific file for scanning. The erroneous conclusion would be that the scanner could detect viruses in files compressed with LZH compression. In this scenario I have not tested the scanner's ability to handle compression. To test for this I would need to compress the file with LZH and then test the scanner against the compressed file.

So, I am supposed to be giving *EICAR* some teeth and so far all I have done is stripped it to its gums. Let us start giving the *EICAR* test file some teeth.

Perhaps the most common use of *EICAR* is to point your scanner at the file and click scan. If your scanner does not detect it then there are four possibilities that immediately come to mind. First, maybe the string is not the first 68 characters in the file. Second, perhaps your scanner is not *EICAR*-compliant. I am truly delighted that you like *MSAV*, but it really is time to buy a newer scanner. There is no law that I know of that says a virus scanner has to detect *EICAR*; however I am not aware of any current virus scanner that does not detect *EICAR*. The third possibility is that your scanner may not have been installed properly; *EICAR* was meant to demonstrate this situation. Finally, if you are running multiple scanners on your machine, the on-access component of another scanner may be intercepting the on-demand scan of the product you are testing. This situation is usually pretty obvious: a scanner other than the scanner you are testing will probably report the file as being 'infected'.

If your scanner detected the file, you have a pretty good idea that your scanner is functional.

Now that we have verified our on-demand scanner is installed and working, the next use that most people find for *EICAR* is to see if their on-access scanner is working properly. I would suppose most people who take this step tend to try either to copy a test file or to open the test file. The on-access scanner triggers and all is well with the world. For many people this is where the *EICAR* test file is discarded. It is little wonder that the file is held in such low esteem. I do not believe most people ever stop to think about how to unlock *EICAR*'s true potential. Let us start with scanner configuration and testing and work our way into anti-virus product evaluation.

Just a few years ago we could tell our scanners to look at files with .COM and .EXE extensions and know we had virtually all of the file-infecting viruses covered. Macro viruses aside, this is no longer the case. In the March 1999 issue of the *Virus Bulletin* more than one product missed a 100% award because they did not scan files with .SCR extensions by default. Many people believe that by saving a *Word* document with an RTF extension they are safe from macro viruses, but do not realize that if their *Word* environment is contaminated with a macro virus such as CAP, no matter what extension or save as type is chosen an infected document template (.DOT) is saved. Is your virus scanner set to scan files with RTF extensions? What about .TXT or .HTM extensions? There are a variety of *Excel* extensions as well. Your scanner may be set to scan .XLS, but what about .XLA and .XLT? A *PowerPoint* presentation can house a macro virus. Does your scanner check .PPT and .PPS files? Leaving the macro arena, are you covered for .SCR? How about .VXD? There is a long list of extensions to look for. Different products have different defaults and there is no guarantee that the next version of your product will have the same defaults as your current version. To make it easier to know how my scanner is set up, I have put together an *EICAR* text suite. No one says the *EICAR* test file must be named EICAR.COM. Name it EICAR.DOC, EICAR.RTF, MYFILE.ZIP, or XY.ZAB. You get the idea. When you put your test files in a directory and scan them you find out if your scanner is actually checking the extensions that are important to you.

Let us move on to other configuration tests. Does your scanner check inside compressed files? Do you know which ones? Are you sure? You can zip up an *EICAR* file and test your scanner to see if it is scanning inside a zipped file. Note that this is different from scanning an *EICAR* named EICAR.ZIP. In the former case we are actually testing the scanner's ability to search inside a zipped file. There are a couple of reasons to perform this type of testing. To begin with, it is useful to know precisely how your scanner is configured. An additional reason to test this is because both scanners and compression packages change. Sometime in 1997 or 1998 a tool that creates ZIP files was modified. The result was that if a ZIP file was created on a floppy disk the file's header was different from if it was created on a hard disk. This caused some scanners to fail to detect infected files inside of the compressed ZIP files created on floppy disks. *EICAR* reveals cases such as this. Not all scanners are created equally, some support more compression types than others. By having *EICAR* samples that are compressed with a variety of different products, you can quickly and easily determine if your scanner handles these compression types. This is probably going to be most important to you if your company decides to license a compression tool such as WinRAR, PKZIP, InstallShield, or any of the variety of tools out there. If you are creating compressed files in-house it is probably a good idea to know if these files serve as lead linings against your virus scanner's x-ray vision. Do not assume that if your scanner can handle a single compressed file it can handle the same compression type when the file is broken up into many pieces. Some scanners can search inside files using *Microsoft*'s cabinet format, but only if it is a single cabinet file.

For example, if I have four-megabytes of files I wish to compress, I can choose to create several cabinet files of 1.44 megabytes each. This allows me to transport the files on floppy disks. I can select files and try to create individual cabinet files that are 1.44 megabytes each, or I can allow the tool to make 1.44-megabyte cabinet files. As the compression tool adds files, when it reaches 1.44 megabytes and it is in the middle of a file, it will close the current cabinet file and start the next cabinet with the second half of the file from the previous cabinet. When the latter is done, all bets are off for anti-virus detection in the cabinet files. I have not yet encountered a scanner that can scan inside these spanned cabinet files. Do not take my word for it; this is a perfect application for *EICAR* testing.

While we are on the topic of compressed files let us touch on recursive compression, where a compressed file is housed inside another compressed file. Perhaps a zipped file is included in a cabinet file. It is not uncommon for *Microsoft* files to have four or more layers of recursive compression with mixed compression formats. I am sure we are not the only company to package files in this manner. How does your scanner handle these files?

Embedded objects are close cousins of compressed files. Just as when you compress an infected file the virus signature string a scanner searches for has been altered, when you embed an object in *Word*, *Excel*, or *PowerPoint*, the signature is altered as well. By embedding *EICAR* in an *Office* document you can determine your scanner's ability to search for infected embedded objects in these types of files. A scanner that can detect the *EICAR* test file in a *Word* document may not be able to detect it in a PowerPoint file. When *Office 2000* went into beta, I used *EICAR* to learn which products had added the ability to scan inside the *Office 2000* file formats. One significant limitation of *EICAR* is that it is not a valid macro. Even if it was a valid macro command, the macro command is not the first series of characters in a macro or in the document. You will not be able to use *EICAR* to tell if your scanner can detect macros in an *Office* document, but I suspect it is unlikely that a product will be able to detect embedded objects in an *Office* document, but not macros. The reverse, however, is not a logical conclusion. Products were detecting an infected *PowerPoint* document prior to being able to detect an embedded object in a *PowerPoint* presentation.

This is an area of testing that can get quite complex. Depending on what you want to test, you can include a zipped file in a cabinet file, and embed that in a *Word* document. From there the *Word* document can be compressed with RAR and that file can be embedded in an *Excel* file. There is a myriad of possibilities. Why would anyone do this sort of testing? Some people want to know how far their email or firewall scanner can go. When there are configuration options for the number of layers of compression to scan through a person might want to know how much time each additional layer adds to the overhead of the scanner.

Let us leave the field of compression and go to context menu handling. The context menu is the choice of actions you are presented with when you right-click on a file, directory, or other object in *Windows*. Your right-click options will be different depending on the context that is appropriate for the object you have right-clicked on. When you right-click on a directory to scan it, what are you scanning? I have used at least one scanner that has hard coded settings for the context scan. I can configure the scanner to scan everything, but if I use the context menu to initiate a scan it will not scan all files. Do you know if your scanner exhibits this behaviour? *EICAR* is a safe, easy way to test for this behaviour.

While we are learning more about parts of our scanner, let us return to the on-access scanner. File open, file close, and file copy are some of the actions that an on-access scanner can be designed to scan. Not all on-access scanners are designed to monitor all of these operations. You can learn which actions your on-access scanner monitors with the *EICAR* test file. I have an on-access scanner that triggers if I even access a directory an infected file resides in. This same scanner will allow me to create an *EICAR* without detecting it. This on-access scanner does not monitor when a file closes. Of course this also means a dropper could be written that writes an old virus to disk without it being detected at the time it is created.

There is an area of testing that is of interest to me. It is not compression. It is not configuration. I will refer to it as file-system anomalies. In early 1999 I inadvertently found a bug in a product. If I had a directory with an extension, but no prefix, one of my scanners would not scan the directory. My *EICAR* suite now includes a directory that has an extension, but no prefix. If this can trip up a scanner once, it may potentially arise again in the same, or even in a different product. Another file-system test I run tests a scanner's ability to open files with double-byte characters in the file name. Double-byte languages are typically, if not exclusively, Asian languages. The Asian languages have so many characters that not all of them can be represented by one byte of data. If you recall, there are 256 unique characters that can be represented by one byte. Languages such as Chinese, Japanese, and Korean have thousands of characters. In order to be able to display all these characters the operating system must use two bytes of data to represent each character, hence the name double-byte. An operating system designed to handle double-byte characters, such as Chinese *Windows NT* has no problem with Chinese characters. I can install an English version of a virus scanner on a Chinese operating system and it will scan files that contain double-byte characters in their file names quite adeptly. The problem arises when my co-worker in China has a file with double-byte characters in the file name and copies this file to my English *Windows NT* machine. Some scanners will have no problems with this file, but others are unable to open this file or files like this to scan. Another case to examine is when a file has multiple extensions. If you have a file named 'This.xyz.com', will your scanner determine that it is a file of the '.XYZ' type and does not need to be scanned, or will it look to the final .COM extension? You could use a live virus to perform this test, but *EICAR* will safely provide you the answer to this question. *EICAR* is a safe way to discover how your scanner handles files or directories with unusual characteristics.

Let us leave the arena of detection altogether now and find out exactly how the features of your anti-virus product work. When you schedule a scan you typically relinquish live process verification and rely on logs to tell you what has happened. In some cases it may be unclear what really happened. Were zero infected files reported because there were not any or because your scheduled scan did not scan all of the targets you wanted it to? With an *EICAR* test suite you can schedule a scan against a known set and verify the results. If your scheduled scan includes email notification you can verify the email portion is properly configured and performing as expected. If the detection settings for a scheduled scan are made independently of the settings for the on-demand, or on-access scanner, you can use an *EICAR* set to verify your settings. If network alerting is selected, you can find out who is being alerted. In beta testing one product, I chose to have the 'administrator' notified of infections. Perhaps it was a bit arrogant of me to assume I was the administrator the scanner had in mind. As it turned out, the scanner had a field day with *Microsoft'*s network. I was not the first person it found logged on as administrator, and the person it did find was really surprised when a virus alert popped up on his desktop.

Perhaps you think that your on-access scanner will prevent an infected file from being copied onto a protected server, but actually the file is copied and then moved or copied and renamed. *EICAR* is a perfect tool to use for testing these behaviours. There is absolutely no need to put even the most innocuous of viruses on a machine to determine these behaviour characteristics; *EICAR* can provide you with this data.

Many of you have firewalls or email scanners. There are a variety of settings and conditions that warrant testing. You can email or ftp an *EICAR* file or test suite to determine the reactions and characteristics of your gateway scanner. If you wish to train some of your employees to be able to configure their scanners correctly, *EICAR* is a suitable tool to assist you.

Each time the scanner is updated or upgraded there is potential for something to break. When one product was adding support for *Office 2000* files I discovered that a different type of detection was not working properly. When I received the fix for this problem I found that the *Office 2000* detection no longer worked. An *EICAR* test suite can be a quick, safe, and effective way to run a variety of diagnostics each time you update or upgrade your anti-virus software.

This leaves product evaluation. This is the aspect of anti-virus at which *EICAR* may be both the least effective and the most efficient. You cannot tell a thing about how many viruses a scanner can detect with an *EICAR* suite. Of course you will not find this information by testing against a limited zoo either. If you have the resources, and I do not think many of us do, to test against a truly comprehensive set in a scientific manner, this may be your information for detection capability. If you must rely on the results of other organizations for the detection component of your evaluation you can use a thorough *EICAR* suite for additional information.

Once you have compiled an *EICAR* suite that you can use to measure the criteria that are important to you, it becomes a much easier, and safer task to test the products against your criteria. It becomes a simple operation to throw the scanner at a test-bed on any machine in your facility and log what the product can do. If the ability to deal with specific compression types is required, make it part of your test-set. If embedded objects are required, add them to your suite. If ZIP files must be scanned as they arrive in email, *EICAR* will reveal this capability as well.

It is important to note that your entire *EICAR* suite can be used on a PC that is connected to your network. *EICAR* testing does not come with the security overhead mandated when live viruses are used.

*EICAR* has two basic limitations. First, detection testing of live virus samples cannot be accomplished with the *EICAR* file. The second limitation is your imagination. *EICAR* is clearly capable of much more than telling you if your scanner can detect the *EICAR* test file.

Going forward, I would ask the anti-virus industry to add two additional *EICAR*-type tests. The first is boot sector testing. It is not unreasonable for users to wish to verify that boot sector scanning is functional. One anti-virus product was made available for public download and could not detect boot sector viruses on floppy disks that contained no files. Another product would not detect boot sector viruses if its on-access module was disabled. The anti-virus industry will continue to mandate the use of live boot sector viruses until there is a test that end users can perform to verify boot sector detection without the use of infectious code. The second test would be an *EICAR* macro. This could be something as simple as a line of code reading Variable1=X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*.

Again, this represents an opportunity for the anti-virus community to decrease the users' dependency on live viruses for testing. I have heard members of the anti-virus community speak out against these types of test file. The only reason I have heard stated against these two test types is that these types of test file are unneeded. I disagree and if other users would also like to have safe tests for these areas that the *EICAR* test file cannot address, I encourage you to contact your anti-virus vendors. Ask them for either a test file to check macro detection and boot sector detection, or live boot sector and macro samples to test that their product is performing correctly. You are not likely to receive live samples from your anti-virus vendor, but the industry should be able to agree upon boot and macro tests and implement them.