# Test Files and Product Evaluation: the Case for and against Malware Simulation

**David Harley (ESET & AMTSO)**
**Lysa Myers (West Coast Labs)**
**Eddy Willems (GDATA & EICAR)**

*This paper was presented at the AVAR (Association of anti Virus Asia Researchers 13th Conference in Bali in November 2010.*

## Abstract

Any researcher with the most modest public profile is used to being asked for virus samples. Traditionally, we've advocated the use of alternatives, especially the EICAR test file, to anyone who doesn't have access to malware through mainstream, trusted channels, as a way of simulating malware behaviour without the attendant risks of genuinely malicious behaviour. But is the EICAR file really suitable for the range of scenarios for which it is prescribed?

Of course, it's always been difficult for aspirant testers outside the mainstream circle of trust to tap into the sample repositories and exchange mechanisms that benefit the major testers. However, as the influence of AMTSO on testing-related issues has increased, it has resulted in a move away from static testing to some form of dynamic testing, it's become even more difficult for such testers to establish the connections in the industry that would make it easier for them to tap into the evolving sample and information exchanges that the big players in the industry are formulating, or the knowledge and experience that would enable them to explore more realistic alternative methodologies for trapping and validating samples.

Ironically, while dynamic testing offers, in the abstract, something closer to real-world testing, there's been increased and unanticipated use of the EICAR file in testing contexts for which it was not designed - or appropriate, being a classic survivor of the strictest form of signature detection.

This paper draws on our combined experience of AV research, testing, and EICAR directorship to look at the genesis and development of the EICAR test file, from the rationalization of product-specific installation test files, through virus/malware simulation software, through its re-specification in 2003, to its recent rebirth as a test tool. Most importantly, it discusses, with examples, the separation in functionality between its use as an installation check and when it is (and, more often, isn't) feasible to use it as a limited test tool, primarily as a check on detection functionality.

## Introduction

Any anti-malware researcher with even the most modest public profile is used to being asked two questions:

1. Don’t the anti-virus companies write all the viruses?
2. Will you give me some viruses to test my AV with? [1]

The first question has been answered time and time again, though perhaps not adequately, since it continues to be asked, but we’re not going to attempt to answer it yet again here, except in the very limited context of testing, where the pros and cons of malware creation and modification for evaluating detection capability has been considered at some length in an AMTSO paper [2]. While the creation of *simulated* malware may seem less contentious, modification of otherwise non-malicious test files may present unexpected problems that we’ll consider in due course.

Traditionally and in more general contexts, we’ve advocated the use of alternatives, especially the EICAR test file, to anyone who doesn’t have access to malware through mainstream, trusted channels, as a way of simulating malware behaviour without the attendant risks of genuinely malicious behaviour.

As the influence of AMTSO on testing-related issues has increased, it has resulted in a move away from static testing to some form of dynamic testing. While dynamic testing is, potentially, a better representation of the real threat landscape than any static test [3], it poses a number of difficulties for aspiring testers [4]. More precisely, while all detection testing is difficult to do properly, dynamic testing is an order of magnitude harder [3].

It has always been hard for aspirant testers to establish the connections in the industry that would make it easier for them to exchange static binary samples. It’s even harder to tap into the evolving sample and information exchanges that the big players in the industry are formulating, or the knowledge and experience that would enable them to explore more realistic alternative methodologies for trapping and validating samples, so a “harmless” executable that’s freely available may have particular attractions.

But is the EICAR file really suitable for the range of scenarios for which it is prescribed?

## Virus Simulation

Sarah Gordon [5] classified simulators as follows, and we see no particular reason to depart from that model:

- **Simulators for Education** – demonstration programs such as the Virus Simulation Suite, Virlab, and AVP. These, she suggested, raise awareness, but mislead by setting up false expectations as to behaviour. We wouldn’t argue with that: in fact, the observation seems more accurate than ever in an era where most malware, being intended to make profit rather than a visual demonstration of some amateur coder’s prowess, runs as inconspicuously as possible to reduce the likelihood of detection and removal.

- **Tests using Simulators** – in particular, Rosenthal's Virus Simulator. While the simulator marketed for many years by Doren Rosenthal is no longer a serious, live issue, the research community continues to regard it as both ethically and technically flawed. Even if we ignore the ethical issues [2] which most critics of the antivirus industry (and, by association, the Anti-Malware Testing Standards Organization) regard as a purely self-serving/self-protective objection [6], the technical objections raised by both Gordon [5] and Sambucci [7] in the 1990s are no less true today.

- The purpose of the EICAR Test File according to Gordon is to establish:
    - Whether AV is installed "correctly"
    - What happens when the AV finds a virus
    - Which messages are displayed
    - How it handles "custom warnings", batch files, and notifications to the system administrator over the network.

Gordon states that "the existence of the EICAR test file, we feel, obviates the need for simulated viruses used for testing purposes".

We see even less use for simulated malware in an age of malware glut, when virus labs are seeing tens and even hundreds of thousands of unique malicious binaries per day, but have expectations of what the EICAR file can reasonably be used for been set too high?

## The EICAR Test file

We like EICAR. [8] Certainly we like the organization – indeed, it was after the 2010 EICAR conference that we resolved to submit this paper – but also the test file that bears EICAR's name, though only in contexts where its use is appropriate.

### There's Testing, Then There's Testing

The EICAR test file was never particularly intended as a tool for testing in the sense of either comparative detection testing or testing for certification purposes [9], though correct detection of the test file is certainly a legitimate (if limited) test target. Apart from that, it's hard to see any useful purpose for it in a test intended to compare or evaluate detection testing, except to confirm that the scanner is active, as described below.

Rather, it's intended as an installation check (or test). Frankly, we regard the term "EICAR test file" as unfortunate in that "by default" we now think of testing in terms of comparative testing or certification. However, getting the world to refer to it as the "EICAR installation check file" is probably as hopeless as restoring the original and non-pejorative use of the term hacking.

When the EICAR file became almost universally supported for installation checking by mainstream antivirus companies, it replaced files with similar functionality created by individual vendors (such as S&S/Dr. Solomon's and Frisk) for use with their own products.

It must be admitted that as a simulated virus, the EICAR file is something of a failure, since it lacks all the characteristics that we normally associate with self-replicating malware. Most obviously, it doesn't replicate (!!!), though it can and has been described as simulating an overwriting virus [10]. Furthermore, it's hard to see how it could be described as malicious, and since all it does when it's executed is tell you what it is, it can't even be described accurately as a Trojan horse. It consists of the following string of characters – actually, the specification allows for a (very little) bit more than that, but we'll come to that.

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

If it executes (though active on-access antivirus *should* prevent it from executing), it prints the following string to the screen.

```
EICAR-STANDARD-ANTIVIRUS-TEST-FILE!
```

It doesn't tell you much beyond that, of course: only that your AV is awake. It doesn't, in itself, prove that AV is configured properly. It doesn't even prove it detects any real viruses. (We don't know if there's any fake/rogue AV out there that detects the EICAR test file, but it wouldn't surprise us, though we hope we're not giving some miscreant an idea...)

Unlike most of the other "simulations" listed by Gordon, the EICAR file was never intended to be a realistic malware simulation. In fact we wonder if simulated malware can ever behave like "real" malware in any sense that makes it useful in detection testing without actually being malware. But EICAR exhibits (as long as it meets the specification correctly) no malicious behaviour of its own. It simply aims to generate a response from security software that approximates to the response that real malware would generate. As we'll see from the section on the evolution of the EICAR file, it can't be described as closer than an approximation. While most mainstream products will identify it, and usually won't allow it to execute (often quarantining it or limiting access in other ways), their response is rarely exactly the same as that elicited by real malware. Even displayed messages are likely to differ from standard messages, reflecting the fact that most security researchers consider it inappropriate to flag a non-malicious file as malicious. Hence such messages as "EICAR test file not-a-virus detected". (That's not a real example, but it's not farfetched.)

## Strings Attached

While the EICAR file *has* been used to illustrate some principles of string searching in virus detection (actually, by one of the authors [11], because a text file is easier to use for teaching purposes than a binary), it's really only suitable for illustrating advanced detection techniques in a very simplistic fashion. This is because it's so atypical in another way. Whereas the anti-malware industry is highly focused on generic and proactive technologies (under the various but related umbrellas of heuristic analysis, behaviour analysis, sandboxing, emulation and so on), the EICAR test file is a classic survivor of the strictest form of signature detection, sometimes referred to as exact identification.

Exact Identification can be defined as the recognition of a virus when every section of the non-modifiable parts of the virus body is uniquely identified: in principle, the same applies to non-viral malware. In this case, identification is even more atypical in that:

1. The entire “sample” is an ASCII text string, even though (apart from the substring which is actually displayed on execution) it’s a rather goofy-looking string.
2. More crucially, the entire “sample” (appended whitespace characters apart) *is* the non-modifiable code.

To understand exactly why this is important, we first have to consider the history of the EICAR file.

## Evolution of the EICAR file

The EICAR test file was created by members of CARO [12] for EICAR in the early 1990s. At that time there was close cooperation between CARO, EICAR and the whole AV industry. CARO (Computer AntiVirus Researcher's Organization) is an informal group of individuals who have been working together since around 1990 across corporate and academic borders to study the whole of computer malware. CARO essentially superseded other less formalized groups of anti-virus professionals. CARO preceded EICAR, a more formal organization, but was founded by a similar set of people. Whereas CARO was always a technical group, EICAR also had a distinct legal and general security focus. Nowadays, the two groups operate largely independently of each other.

However, a historic joint project was the EICAR test file, created by CARO members and published by EICAR, in order to meet a perceived need for a simple means by which a helpdesk operator could read it over the telephone to an end user, to allow a check on whether his or her antivirus was working. [12]

### First Wave

The original definition in short is as follows:

> “The file is a legitimate DOS program, and produces sensible results when run (it prints the message "EICAR-STANDARD-ANTIVIRUS-TEST-FILE").
>
> It is also short and simple - in fact it consists entirely of printable ASCII characters, so that it can easily be created with a regular text editor. Any anti-virus product that supports the test file should detect it in any file providing that the file starts with the following 68 characters:
>
> X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
>
> To keep things simple, the file uses only upper case letters, digits and punctuation marks, and does not include spaces. The only thing to watch out for when typing in the test file is that the third character is the capital letter "O", not the digit zero.”

However, two interesting events during the years between the original specification and 2003 necessitated a rethink:

> (1) The first was a virus commonly known as Bat/Bwg.a@MM, an Internet worm which never appeared in The Wild (http://www.wildlist.org/faq.htm) but was generated by the construction kit Batch Worm Generator. The most interesting feature of this virus is that it was an attack on the EICAR test file, or rather made use of it as a stealth approach to infection. Bat/Bwg.a@MM starts with the EICAR string: when the worm is run, it generates a

"File not found" error but the execution continues. Many AV products misdetected this virus as the EICAR test file when it first appeared. While this was the first widely-known instance of malware impersonating the EICAR file, it wasn't actually the first problem deriving from a slight looseness in this initial definition. While the definition is entirely accurate on what the file actually is (or was at that time), it assumed a commonsense approach on the part of all AV vendors and therefore didn't go into detail on what the file *could not* or *should not* be. This led to such anomalies as detection of files being detected as the EICAR file because they contained the EICAR string, even where it didn't constitute the very first characters of the file.

(2) As a result of this malware, copious discussions ensued on various anti-virus forums. This exploitation of the original file created a great deal of rumour and speculation. It was even suggested that the file should be changed completely. As a result of such problems and ensuing discussion, each different vendor found different ways to adjust their detection so that real malware wasn't missed because the malware author had cunningly embedded the EICAR string into his creation.

## Second Wave

EICAR saw these problems and wanted to help the AV vendors by with a slight change to the formal definition so that a fully-regularized, correct, safe definition was available that would leave no doubt in the minds of either vendors or users as to what a fully conformant EICAR test file should look like.

So EICAR (Eddy Willems) came up with a first proposal for a change in the formal definition:

"The file is a legitimate DOS program, and produces sensible results when run (it prints the message "EICAR-STANDARD-ANTIVIRUS-TEST-FILE").

It is also short and simple - in fact, it consists entirely of printable ASCII characters, so that it can easily be created with a regular text editor. Any anti-virus product that supports the test file should detect it in any file providing that the file starts with the following 68 characters, and is exactly 68 bytes long]:

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

To keep things simple ...."

You remember that observation we made about exact identification earlier? However, this suggestion received somewhat mixed reactions:

- Some vendors and researchers agreed
- Some vendors didn't consider the definition sufficiently explicit or exact
- Some of them considered it premature because more clarification and discussion was needed.

Discussion started again within the forums, but also at meetings of the WildList Organisation and CARO. EICAR decided to change the definition again in such a way as would be mutually agreed by most vendors, so EICAR tried to bundle all the ideas that had been suggested into the final version of the definition, which was released at the beginning of 2003 and published on the EICAR website May 1 2003.

### Say Hello, Wave Goodbye

The definition now looks like this:

> "The file is a legitimate DOS program, and produces sensible results when run (it prints the message "EICAR-STANDARD-ANTIVIRUS-TEST-FILE").
>
> It is also short and simple - in fact, it consists entirely of printable ASCII characters, so that it can easily be created with a regular text editor. Any anti-virus product that supports the test file should detect it in any file providing that the file starts with the following 68 characters:
>
> X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
>
> The first 68 characters is the known string. It may be optionally appended by any combination of whitespace characters with the total file length not exceeding 128 characters. The only whitespace characters allowed are the space character, tab, LF, CR, CTRL-Z."

Reactions to the second change were more favourable [13], and most vendors or experts had no further suggestions. Nearly everybody discussed the problem internally and a consensus was found, and the discussions faded away, at least until EICAR 2010 conference – but more of that later.

### Summary Cum Laude

Thus, while the file was always essentially a 68-character string, the enhanced definition, as long as it is scrupulously followed, comes close to eradicating risk of false negatives (malware managing to execute because its presence is masked by the presence of the test string). It also lessens the risk of a special case of false positive: that is, when the EICAR file is detected when it shouldn't be (for example, embedded in a Word document as text). White space apart, the file was always intended to be an exact sequence of bytes.

However, this hasn't prevented the appearance of a number of ingenious but inappropriate ways to use it in ways that were never intended, many of which involve completely misreading or ignoring the formal specification.

## Fun with Dick, Jane and EICAR

Attractive though it is for aspiring testers to be able to test without the privileged access to mainstream sample repositories and exchange that's open to mainstream testers through their contacts with each other and the AV industry, mixing up EICAR detection with malware detection creates more potential problems than it solves.

There have been a number of attempts to prove some sort of point about the EICAR file itself, antivirus products and their effectiveness or the lack of it, or about AV and testing, most of which include some form of modification of the file itself.

Doren Rosenthal wrote some utilities that included virus simulation, and in fact made frequent comparisons between his tools and the EICAR file. He regarded EICAR (probably not without cause) as an intentional “spoiler” to deter sales of his utilities, which were disliked by AV researchers in general, for a number of reasons:

- They were based on the false premise that a product that detects a real virus should also detect a simulation of that virus. [10]
- The previous false premise is also based on the equally false premise that a virus signature is some unique footprint and that all security software uses the same signature. [14] There is, of course, no reason why random virus fragments should be detected as if they were a real virus, and even less reason for several vendors to use the same signature.
- The registered version of the utilities also included a real if relatively innocuous virus.
- The inappropriate use of Rosenthal’s programs by some testers in comparative testing [15] forced the AV industry to add detection of yet another non-virus to the impressive selection of garbage files, intendeds and other inappropriate objects it needs to detect if a product isn’t to be penalized for *not* detecting what it *shouldn’t need* to detect.

## Spycar

Spycar (http://www.spycar.org/) came out of a research project at Intelguardians Labs, as a result of collaboration between Ed Skoudis, Tom Liston and Mike Poor. It was intended to test anti-spyware programs by observing their response to certain behaviours commonly associated with Windows spyware, using tools that were not malicious. Like Rosenthal’s virus, they made no permanent changes to the system. Not everyone was convinced by the methodology, though the idea of checking behavioural detection rather than signature detection was by no means totally invalid. However, while the self-conscious “homage” to the EICAR name might lead you to expect a kind of EICAR file for spyware, the Spycar team nailed both the difference between the intent behind the two approaches and the functionality of the EICAR file rather well.

> “The EICAR file can be used to verify that your anti-virus tool is alive and running. Spycar tests behavior-based alerting and blocking. ... You’ve got a smoke detector, and you want to see if it is working. The EICAR file is like the big red test button on the smoke detector ... the smoke detector beeps, telling you that the battery is charged and everything seems to be working properly ... Spycar... mimics the behavior of a real fire (again, in a benign fashion) to see if your smoke detector is protecting you.”

Well, we might question “everything seems to be working properly”. To quote one of these authors [16]:

> “It's a (limited) check on whether the program is installed, but I'm not sure it's a measure of whether it's installed correctly. For instance, the fact that a scanner reports correctly that a file called EICAR.COM contains the EICAR string, doesn't tell you whether it will detect macro viruses, for example. In fact, if I wanted to be really picky, I'd have to say that it doesn't actually tell you anything except that the scanner detects the EICAR string in files with a particular extension. “

Still, the Spycar definitions illustrate quite clearly the difference between an installation check file and an attempt to create a type of tool that could, in principle, be of potential value in evaluating products.

In practice, Spycar is now of limited use because vendors can (and do!) write detections that are based on behaviour as well as static signatures. [17]

Other players have, however, somehow overlooked the restrictions imposed by the tight definition of what the EICAR file should be and the consequent limitations on how it should be detected.

## **"Having fun with the EICAR test file"**

This is a rough disassembly of the EICAR file from a series of antique cross-postings to alt.comp.virus, bugtraq, and ntbugtraq. [18]

```
01 0100 58 pop  ax
 02 0101 354F21 xor  ax,214Fh
 03 0104 50 push ax
 04 0105 254041 and  ax,4140h
 05 0108 50 push ax
 06 0109 5B pop  bx
 07 010A 345C xor  al,5Ch
 08 010C 50 push ax
 09 010D 5A pop  dx
 10 010E 58 pop  ax
 11 010F 353428 xor  ax,2834h
 12 0112 50 push ax
 13 0113 5E pop  si
 14 0114 2937 sub  [bx],si
 15 0116 43 inc  bx
 16 0117 43 inc  bx
 17 0118 2937 sub  [bx],si
 18 011A 7D24 jge  0140
 19 011C 45494341
522D5354
414E4441
52442D41
4E544956
49525553
2D544553
542D4649
4C452124 DB   "EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$"
 20 0140 48 dec  ax
 21 0141 2B482A sub  cx,[bx+si+2A]
```

The poster (using the handle "keepitsecret") went on to suggest that "using ESATF is a cool and legal way to learn how AVs do their job: we won't disassemble any AV or break any licence agreements in our tests! The nice part is to watch how heuristics work with a code in principle detected by its signature (somehow, a way to assess the limits of this method)....." (ESATF stands for EICAR Standard Antivirus Test File.)

First he zipped the file and ran some scanners against the resultant archive, with the following results from 9 different scanners (identification of the scanners has been removed, to ensure that no-one draws any inappropriate conclusions about products that are still in existence).

```
EICAR_Test.
EICAR-STANDARD-TEST-FILE.
Eicar_test_file.
EICAR_Test_File (exact).
EICAR-Test-File.
 EICAR Test String.
EICAR-AV-TEST-FILE.
EICAR_Test_File.
EICAR test file.
```

Then he started to play the bold hacker by changing three letters of the character string displayed by the file when allowed to execute. Instead of `"EICAR-STANDARD-ANTIVIRUS-TEST-FILE!"` the string now displayed would be `"EICAR-STANDING-ANTIVIRUS-TEST-FILE!"` This makes no essential difference to the code: in fact, it's a similar modification to the "variant" of Stoned that changed the substring "stoned" to "sanded." Apparently the experimenter believed that a savvy AV scanner should recognize it as essentially the same object.

However, these were the results (N/D is the poster's shorthand for Not Detected or a similar message):

```
EICAR_Test ( modified ).
N/D.
N/D.
EICAR_Test_File.unknown?
N/D.
N/D.
EICAR-AV-TEST-FILE.
N/D.
N/D.
```

And these were the poster's comments.

```
"Only three AVs are aware of the alteration! Are others using the
original ESATF string as signature? If so, it's not very clever
(should they learn about wildcard string? For the "fun", they could
have search for the EICAR? pattern!)...
```

Further suggestions from this chap:

Add a NOP instruction to the beginning of the file.

Add a JMP instruction to shoot straight past the code to the end of the file.

XOR and OR encryption

Add file search and replication.

As you might have expected, most of the products “tested” quite rightly ignored the modified versions: one or two detected it despite some modifications, and as the file bore less and less resemblance to the test file and more to the “Trivial” virus family, more products detected it generically as an overwriting virus. The poster, however, drew a number of unwarranted conclusions about the shortcomings of anti-malware technology: clearly he was unaware of the strict specification of the file, as already discussed (so we won’t revisit the counterarguments made by better-informed researchers at the time, though there’s a particularly good response by Vesselin Bontchev still available at http://marc.info/?l=ntbugtraq&m=105735579817004&w=2). Instead, let’s move on to a set of somewhat similar experiments that came to light at the iAWACS and EICAR conferences in Paris in 2010.

## Obscurity and the City of Light

A number of entries were made for the iAWACS 2010 PWN2KILL Contest [19] intended to demonstrate that “given fixed actual malware threats, the different existing antivirus software of unequal quality.” Well, we suppose that *all* comparative testing is based on that proposition, but there was also a clear intent to make a point about the shortcomings of security software. Testing was also a major theme of the EICAR 2010 conference [20] which immediately followed the iAWACS conference [21], promising [22] a “routine allowing testing of the requirement to use samples of viral code with respect to the current most widely encountered threats.” The most spectacular fruit of this exercise was the final presentation at EICAR, based on one of the PWN2KILL attacks. [23].

The attack described in this EICAR paper uses somewhat similar techniques to “keepitsecret’s” paper: no real malware was used, but only variations on the EICAR file. The paper concluded that the EICAR test file is not detected by a manual (on-demand) scan:

- When its bytes are changed
- When split into two parts
- When the EICAR string is incorporated into data
- When cryptographic or polymorphic techniques are used
- When characters are added to the file.

“The only case where the test file is detected is when the file remains intact into the document and also when the technique uses unchanged version of file.” No surprises there: if you’ve read this far, you know that this is exactly how we would anticipate the scanners responding to each of these scenarios, given the formal definition of the file.

Testing of the scanners under fire using on-access (realtime) scanning concluded that in general, “...the eicar file is detected at the execution when the file is created” notwithstanding two anomalous results “...in the case of temporal obfuscation....” and that the file is detected on-access even when encrypted. Again, this is what we would expect in general: most obfuscation of known (or unknown but proactively detectable) malcode is likely to defeat an on-demand scanner, but will be detected on execution (at which time the obfuscating “layer” is removed). As before, changing the first byte of the file or adding characters results in non-detection. While this is correct behaviour, the presenters took it to mean that detection had been bypassed (a false negative), and observed in their final conclusions that "...the detection scheme seems to be only limited to pattern matching." (More or less true, depending on your definition of pattern matching...)

As Willems observed [20], some of the conclusions here were flawed, being based on the erroneous presumption that general detection methodology can be deduced from the single case of EICAR detection, but the underlying theory seems appropriate (even laudable, in its determination to avoid reliance on purely malicious code), and might be very well suited to research using a more suitable match of tool and testing purpose. And as Harley commented [21], while the vendor community was understandably taken aback at being penalized for its scrupulous conformance to EICAR's own rigorous specification, it would be a pity to overlook essential messages about the need for scientific research to underpin pragmatic solutions.

## Conclusion

The EICAR test file has only the most limited application to testing, and it would probably be more appropriate to refer to it as the EICAR installation check file, or something similar.

- It's intended as an installation check, not for detection testing. It tells you nothing about how effectively it detects real malware, and has no place in a test intended to evaluate detection of real malware. In extreme cases, we've seen [24] instances where a single test has attempted to assess:
    - Recognition of the EICAR test file
    - Recognition of presumed In the Wild (ItW) malware
    - Recognition of presumed malware not known to be ItW
    - Recognition of presumed malware not expected to be known to the scanner

  At the very least, this indicates a muddled and therefore undependable methodology.
- Even as an installation check, it tells you practically nothing about whether the product is correctly installed and configured, only that it's functioning in some manner.
- Most scanners detect it, even on platforms where, as a DOS executable, it can't execute natively (for example under Mac OS without emulation). However, AV response is not standardized: that is, the way in which it is flagged and processed is not laid down in the EICAR specification. In fact, the exact way in which it's detected is not specified, either, though in order to take full account of the specification, the available detection techniques are limited to some form of exact identification.
- Because of the very tight specification, modification to the core executable will normally invalidate the test, though the use of some sort of wrapper may be acceptable. A scanning product may be flexible enough to recognize the EICAR file when harmless modifications are made, but that's a design decision that isn't really a suitable target. A product that fully supports the EICAR specification but is not flexible enough to recognize modifications to the core code is *not* behaving incorrectly, and may be more "correct" than a scanner that is

more flexible. It really depends on context: for instance, a scanner that flags EICAR and EICAR variants differently could be said to be behaving appropriately as long as it doesn't present opportunities for malware to "piggyback" the test file.

- It's possible to use the EICAR test file to test characteristics and issues that are related to detection but don't require the use of real malware samples. [25]. However, it would take a great deal of care and experience to use such techniques accurately in the context of a comparative test.
- It cannot be assumed that the way in which a scanner behaves when it detects the EICAR test file is identical to the way in which it will behave when it detects real malware.

Unfortunately, even the use of the EICAR file as an installation check can be problematical. One of the authors used to receive regular log summaries from an outsourcer indicating that an email-monitoring AV scanner was detecting multiple instances of the EICAR file (for multiple, read tens of thousands) every week. The report was insufficiently detailed to indicate how many of these reports were installation checks by the company to whom the service was outsourced, and how many were installation checks carried out by end sites. The author in question also regularly received complaints from end sites that they were unable to test their own site security correctly because the email scanner blocked some or all of their tests. It's difficult to see much value in installation checks in such volumes, except as a "here I am, doing my job" public relations exercise. [26].

A constant theme in the (mis-)use of the EICAR file in comparative testing has been the assumption that how the file is detected by a particular security product is a reflection of the way in which it detects real malware. It's hard to see how this could possibly be so, irrespective of the limitations imposed by the formal definition. Modern anti-malware has little to do with the simplistic signature scanning of the Good Old Days: multiple technologies need multiple (reactive and proactive) approaches to testing, and any test that draws conclusions based on single protection layers is in danger of misleading its audience.

EICAR has been described [10] as being useful for convincing management that you're earning your crust by installing working software, or as a demonstration to end users of the sort of message they might see if malware does reach their system (though as we've seen, that might only be an approximation. A paper by Randy Abrams [23] describes some techniques for extending the test file's capabilities such as wrapping it in nested zip files, or as an embedded OLE-2 object in Office documents. The EICAR file is fine for looking at these issues, which can be considered effectively without the use of real malware. It may also offer a limited means [10] of checking on:

- How your software is or could be deployed locally (and to a lesser extent, configured)
- Monitoring or demonstrating incident-handling procedures in the context of corporate security

As a tool for comparative evaluation, the limitations imposed by its formal definition, however, we see little use for it in its present form.

## References

[1] David Harley, alt.comp.virus FAQ, http://www.faqs.org/faqs/computer-virus/alt-faq/part4/

[2] AMTSO, “Issues Involved in the Creation of Samples for Testing” http://amtso.org/amtso---download---issues-involved-in-the-creation-of-samples-for-testing.html

[3] David Harley, “Antivirus Testing and AMTSO: Has Anything Changed?”, Computer Forensics Education & Training conference 2010, http://www.amtso.org/uploads/cfet2010-antivirus-testing-and-amtso.pdf

[4] Matt Garrad, Paul Jones, Lysa Myers and Michael Parsons, “Paradigm Shift - From Static To Realtime, A Progress Report”, EICAR 2010 Conference Proceedings, http://www.amtso.org/uploads/eicar2010-meyers-paradigmshift.pdf

[5] Sarah Gordon, “Are Good Virus Simulators Still a Bad Idea?”, Elsevier Advanced Technology 1995. http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6VJG-3WP2C4W-4&_user=10&_coverDate=09%2F30%2F1996&_rdoc=1&_fmt=high&_orig=search&_origin=search&_sort=d&_docanchor=&view=c&_searchStrId=1445369969&_rerunOrigin=google&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=c5cc64e0e6ad3d99669126cddfd51520&searchtype=a

[6] David Harley, “I’AMTSO Mysterioso and Malware Creation,” http://amtso.wordpress.com/2010/06/17/mysterious-amtso-and-malware-creation/

[7] Luca Sambucci, Virus Simulator Test. Italian Computer Antivirus Research Organization. 1994.

[8] CB Presidential Research Services, 1952 Presidential Campaign Slogans, http://www.presidentsusa.net/1952slogan.html, 2009

[9] David Harley and Andrew Lee, “Call of the WildList: Last Orders for Wildcore-Based Testing?”, Virus Bulletin 2010.

[10] David Harley, Robert Slade and Urs Gattiker, “Viruses Revealed”, Osborne 2001.

[11] David Harley, “How (Some) Antivirus Products Detect Malware”: presentation for schools, 2007.

[12] Padgett Peterson, personal communication

[13] Eddy Willems, “The Winds of Change: Updates to the EICAR Test File”, Virus Bulletin June 2003

[14] Vesselin Bontchev, “Analysis and Maintenance of a Clean Virus Library” http://www.people.frisk-software.com/~bontchev/papers/virlib.html

[15] Joe Wells et al., Open Letter 2000 http://cybersoft.com/v3/whitepapers/paper_details.php?content=cs008

[16] Andrew Lee and David Harley in “The AVIEN Malware Defense Guide” (ed. Harley), Syngress 2007

[17] Ed Skoudis, "What is Spycar?", http://searchsecurity.techtarget.com/expert/KnowledgebaseAnswer/0,289625,sid14_gci1286802,00.html

[18] "Having fun with the EICAR test file": http://archive.cert.uni-stuttgart.de/bugtraq/2003/06/msg00251.html; http://marc.info/?l=ntbugtraq&m=105733144930014&w=2]

[19] http://www.esiea-recherche.eu/iawacs_2010.html

[20] Eddy Willems, "EICAR 2010: Rainy Days in Paris", Virus Bulletin, June 2010

[21] David Harley, "PWN2KILL, EICAR and AV: Scientific and Pragmatic Research", Virus Bulletin, June 2010

[22] The Future of AV-Testing: EICAR position paper, http://www.eicar.org/information/infomaterial/eicar_positionpaper_web.pdf

[23] Jonathan Dechaux, Jean-Paul FIzaine, Romain Griveau and Kanza Jaafar, "New trends in Malware Sample-Independent AV Evaluation Techniques with Respect to Document Malware", 19th EICAR Annual Conference Proceedings 2010

[24] David Harley, "Untangling the Wheat from the Chaff in Comparative Anti Virus Reviews", http://www.eset.com/resources/white-papers/AV_comparative_guide.pdf

[25] Randy Abrams, "Giving the EICAR test file some teeth", Virus Bulletin 1999 Conference Proceedings

[26] http://www.secmeme.com/2010/08/frustrated-anti-virus.html