

## Detecting Complex Viruses

Peter Ferrie, Frederic Perriot 2004-12-06

There are many metrics by which to measure the efficiency and effectiveness of an antivirus product and the response organization that is backing it. Some of the commonly used metrics today include the antivirus company's response time to new threats and well as the availability of proactive detection. But are these metrics enough?

The purpose of this paper is to examine the difficulties of detecting complex viruses, including polymorphic, metamorphic and entry-point obscuring viruses. Whether or not an anti-virus technology can detect these viruses can be a useful metric to consider when evaluating AV products.

In this article, we will show how complex viruses can offer an entirely different threat to organizations. It is important to step into the world of complex viruses by defining what a *metamorphic*, *polymorphic*, and *entry-point obscuring* virus is, understand when it is considered a real threat, and then see some real-life examples of complex viruses that have been discovered. This will lead into a discussion on the limitations of current anti-virus engine technology, and then finally, we will try to gauge the importance of detecting these complex viruses accurately, and in a timely fashion.

### Overview of complex viruses

At one time, the aggregate number of viruses a product detects was considered a useful and popular metric, but this has largely been abandoned in favor of other more useful and scientific measures. Today, an AV company's response time to new threats and the proactive detection that their product offers are both considered more important evaluation criteria. But these criteria often do not consider complex viruses, a different kind of threat. Detecting a *complex* virus means detecting a threat that is either inherently difficult to detect, or exposes engine limitations that make it difficult to detect. We will start with a few definitions.

A *polymorphic* virus is a virus that changes its appearance in host programs. For instance, it encrypts its body with a different key each time, and prepends a decryption routine to itself. The decryption routine (known as the "decryptor") is mutated randomly across virus instances, so as to be not easily recognizable.

A *metamorphic* virus, by comparison, is a virus that also changes its appearance in host programs, however it does so without necessarily depending on encryption. The difference in appearance comes from changes made by the virus to its own body. There are several techniques that can produce such an effect.

One of these morphing techniques used by metamorphic viruses is with the insertion and removal of "garbage" instructions. These are instructions that have no effect on the function of the virus, but simply take up space and which can make analysis more difficult when they appear in large quantities. Another technique is to change the basic encoding of instructions at the opcode level. That is, switching between two different opcodes that are functionally-equivalent.

Perhaps the most complex transformation of a metamorphic virus is the replacement of entire blocks of logic with functionally-equivalent blocks of logic. Consider the task of multiplying  $x$  by 3. One expression of this is " $3*x$ ". However, an alternative expression is to replace the single multiplication with a repeated addition instead: " $x+x+x$ ". Both expressions will result in the same answer, yet they look very different.

An *entry-point obscuring* ("EPO") virus is a virus that gets control from the host program in an indirect way, rather than straightforwardly through the main entry-point. Typically, it involves patching a variable location in the host program code, perhaps a function prologue or an API call sequence, and redirecting control flow to the virus code from there.

An inherently difficult virus could be a polymorphic Win32 virus whose appearance varies greatly between samples. Regardless of what technology is available to detect the virus, the first hurdle is to analyze and understand the way the virus works, and invent an algorithm capable of detecting all virus replicants. This can be a daunting task, even assuming the ability to write the detection as a standalone program in a language of one's choice.

## Determining the threat

Complex viruses do not represent a real threat until they are discovered outside of a laboratory and "*in the wild*". Herein lies the problem: the difficulty is in defining what it means for a virus to be "*in the wild*".

The industry definition of a virus "*in the wild*" is typically a virus that has been seen by at least

two independent submitters in at least two different regions. However, this definition overlooks the existence of localized outbreaks, in which one or more companies in a single region might be heavily infected. In that case, a virus might be considered "in the wild" based solely on the number of submissions, but this can be misleading if people submit the same virus sample repeatedly. This also overlooks the case of virus "seeding", in which a virus is placed in a public location, such as the Usenet newsgroups, in the hope that enough people will be tempted to run it -- but no one actually does.

The fact remains that many of the most complex viruses are not especially widespread. If a sample of this virus has not been submitted by a "sufficient" number of outsiders, in a short period of time, it may be considered a "zoo" virus with minimal widespread threat. However, it's important to remember that this level of threat can change at any time.

## Examples of "zoo" viruses

Examples of infamous "zoo" viruses include the complex Win32 viruses known as [W95/SK \(PDF document\)](#), [W95/Zmist \(PDF document\)](#), [W32/Simile \(PDF document\)](#), [W32/Efish \(PDF document\)](#) (from the [W32/Chiton family](#)), and [W95/Perenast](#). Just mention any of these names to an AV researcher and watch their terror-stricken face. [W32/Gobi \(PDF document\)](#) and [W32/Zelly](#) are two of the most recent such brain-teasers. Both are very polymorphic, employing multiple encryption layers and entry-point obscuring.

These examples are all worth a few days (and nights) of work at the least, taking into account reverse-engineering, replicating the virus, and writing the detection signature. It can help a researcher to start writing the detection as a standalone C program before integrating it into one's AV product.

## Limitations in AV engine technology

Unfortunately AV researchers do not have the luxury to write standalone programs from scratch to respond to new viruses. Instead they are constrained by a framework imposed by an AV product. The framework may be more or less flexible, and usually comes with a set of constraints that largely determine how efficient a response will be possible.

A comparatively simple virus affecting an emerging platform (say, Win64) may expose AV engine limitations that make it just as hard to detect as a tough Win32 polymorphic virus, in a subjective way -- depending on what AV engine technology is available to respond. Maybe the

affected file format is not parsed by the engine, or only incompletely supported. Emulation may or may not be available. These factors greatly influence the ability to detect the virus.

Some of the new viruses that affected the Win64 platform in 2004, and were relatively difficult to detect, included [W64/Rugrat \(PDF document\)](#) (IA64), [W64/Shruggle](#) (AMD64), plus some new viruses with MSIL infectors. The corresponding executable file formats are varied, and even the job of picking a simple search string for an immutable virus can turn into a contortionist's exercise if the underlying AV engine lacks support for these file formats.

Naturally, there is the fear of an inherently difficult virus affecting an esoteric or emerging platform like Win64. Such viruses do occasionally surface in zoo collections, to the delight of no one except a virus researcher. Two examples of these new viruses, both released in early 2004, are [MSIL/Impanate \(PDF\)](#) and [MSIL/Gastropod \(PDF document\)](#) - viruses for the Microsoft .NET framework. The first of these, MSIL/Impanate, is an EPO virus. It appends its code to a random method in the file, and rebuilds the host around it. The second of these, MSIL/Gastropod, is a metamorphic virus. Its appearance is altered by the virus intentionally adding and removing "garbage" instructions.

## The importance of detecting complex viruses

You may rightfully ask: why does it matter to detect such viruses, if they belong to "zoo" collections? Well, first of all, sometimes they do find their way into the wild. [W32/Toal](#), for instance, a difficult polymorphic worm, was discussed on an emergency virus mailing list after being spotted actively spreading. Some complex viruses currently registered as zoo samples spread aggressively enough that they would stand a chance to infect machines in the real world if some mischievous soul were to release them.

Moreover, even for purely zoo viruses unlikely to ever cause problems in the wild, the response (or lack thereof) of AV companies to such viruses can reveal a lot about limitations in the engine technology available, and perhaps the skill and dedication of the response teams. Some companies provide detection quickly, in a matter of hours or days, while some others finally ship a solution after months of work (or years in some extreme cases, like W95/Zmist!), and yet other companies simply give up.

Besides the speed of response, the quality of detections also varies greatly, as measured by the ability to detect all samples of a polymorphic virus for instance, and doing so with an acceptable

false-positive rate. What is an acceptable false-positive rate? While this varies from company to company, usually no more than a handful of false positives would be considered acceptable -- however, there are exceptions to this. One recent example, W32/Zelly, was allowed an enormous (up to 50%) false-negative rate by some anti-virus companies just to be among the first to detect it.

What if your AV company gives up on difficult zoo viruses? It certainly says something about either the flexibility of their technology, or the skill and dedication of their response team. What if tomorrow's Mydoom is heavily polymorphic? Will they be able to respond to it in a timely manner?

If you think it's an unlikely scenario, compare it to the following analogy: if you had to pick a surgeon, would you choose the one who carried out hundreds of successful open-heart surgeries, or the one who only ever did appendectomies? Even for an appendectomy, most would choose the first one.

## Conclusion

In this article we've looked at some of the difficulties in detecting complex viruses, by first discussing what they are and why they can be difficult to discover. We then looked at a few examples of "zoo" viruses and how they can uncover limitations in various AV engines. As we have seen, finding complex viruses can be another useful metric in determining which anti-virus technology is best suited to the needs of an organization -- in addition to other common metric such as response time to new threats, and how effective the pro-active detection offered really is.

### About the authors

[Peter Ferrie](#) and [Frederic Perriot](#) are senior researchers with Symantec's USA Security Response.

[Privacy Statement](#)

Copyright 2006, SecurityFocus