VIRUS ANALYSIS

LET THEM EAT BRIOCHE

Peter Ferrie
Symantec Security Response, USA

In 2003 I wrote: 'A virus using the manual reconstruction technique seems unlikely, since the underlying structures in *.NET* are extremely complex and contain many interdependencies' (see *VB*, April 2003, p.5). However, in 2004 we received one that did it: MSIL/Impanate.

Written by the virus writer known as 'roy g biv', a specialist in proof-of-concept viruses (most recently, the first 64-bit viruses on the *Win64* platform: W64/Rugrat on *IA64*, [see *VB*, June 2004, p.4] and W64/Shruggle on *AMD64*), Impanate is the first known parasitic, entry point obscuring appender for the *.NET* platform.

SIGN OF THE TIMES

Impanate searches in the current directory for files which do not contain a zero in the Second field of the LastWriteTime field. Impanate sets the Second field to zero in every file it examines, which serves both as an infection marker and as a means to avoid re-examining uninfectable files.

The use of the timestamp field is a speed optimization method, since it can be queried without incurring the performance penalty of opening the file. In addition, the LastWriteTime field is the only time field that is never changed when a file is copied to another location.

FILTRATION DEVICE

As with all viruses produced by this virus writer, files are infected only if they pass a strict set of filters. The conditions include that the file must be a character-mode or GUI application for the *.NET* framework, that the file is not a DLL, that the file contains no digital certificates, and that it has no bytes outside the image.

The virus avoids files that contain StrongNameSignatures or VTableFixups. StrongNameSignatures are used for digital signing of *.NET* files, so it is clear why the virus avoids files which contain them. However, it is not clear why the virus avoids VTableFixups.

The virus avoids files whose last section is writable, because the virus wants to place its code in the last section of the host, but the *.NET* framework will not allow code to execute from within a writable section.

In addition, the virus supports both 32-bit and 64-bit files, and will infect them both correctly, using a tiny piece of code trickery.

SLIPSTREAM

The virus parses the Metadata root header manually, searching for the streams that it requires. The streams are named '#~', '#Strings' and '#Blob'. The streams may appear in any order – most tools produce a constant order – but the virus will reorder them when it infects a file.

The virus is also aware of several undocumented characteristics of the *.NET* file format, including the extra data fields that can appear in the header and the flags that control the size of the stream references.

The '#~' stream contains information that is of interest to the virus. Specifically, the virus requires that the host contains 16-bit references to the '#Blob', '#GUID' and '#Strings' streams, which make the '#~' stream easier to parse, and that the host contains the following elements: TypeRefs, MemberRefs, StandAloneSigs, AssemblyRefs, Assemblies and Methods.

The virus parses the stream manually to find the TypeRefs, MemberRefs, StandAloneSigs, AssemblyRefs and Methods. The virus is not interested in the Assemblies as such, but simply requires that some are present.

SOME ASSEMBLY REQUIRED

The TypeRefs contain pointers into the '#Blob' stream of the descriptions (the types of parameters to be passed, if any, and the type of the return value, if any) of the library functions used by the host. The virus appends its own TypeRefs to those of the host and updates the references in the '#Blob' stream.

The MemberRefs contain pointers into the '#Strings' stream of the names of the library functions and properties used by the host. The virus appends its own MemberRefs to those of the host and appends the MemberRef names to the '#Strings' stream.

The StandAloneSigs contain the number and type of variables in each Method. The virus chooses randomly from the StandAloneSigs of the host, duplicates one of them and appends the StandAloneSigs of the virus to it.

The AssemblyRefs contain pointers into the '#Strings' stream of the names of external assemblies that contain the functions used by the host. The virus requires two particular assemblies to be referenced in order to replicate.

The first assembly the virus requires is 'mscorlib', which is the assembly that contains many core functions, and which is roughly equivalent to 'kernel32' for *Windows* applications.

The second assembly the virus requires is 'System', which the virus uses to access the process memory, in order to copy the virus code to a local buffer, for modification prior to placing it in the host.

The virus does not alter the AssemblyRefs collection, perhaps because it would mean updating each method of the host, resulting in many changes to the file.

METHOD ACTOR

The Methods contain the host code. The virus finds the first method that uses the StandAloneSigs that the virus chose earlier, and which supports the use of local variables. The virus also requires that the method contains no exception handling information. The most likely reason for this is that the process of updating the exception handling information is extremely complicated.

Having found a suitable method, the virus duplicates it, then appends the virus code to it. After the host method has run it would normally return to the caller; now, the virus will begin to execute at that time, before returning to the caller.

After appending the virus code, the virus parses it manually to update the references to the local variables and functions. The virus contains code to calculate the length of each instruction in the MSIL instruction set, and it knows which instructions need to be processed specially.

After updating the code, the virus updates the size of the last section and the host image size, and recalculates the file checksum, if required.

EXPECT THE UNEXPECTED

And so it comes to pass that, in the hands of a skilled programmer, the unlikely can became the ordinary. At least I didn't say that it could not be done because it was too difficult – anything is possible for those who have enough patience.

MSIL/Impanate

Size:	7539 bytes.
Type:	Direct action, parasitic, entry point obscuring appender.
Infects:	Microsoft *.NET* files.
Payload:	None.
Removal:	Delete infected files and restore them from backup.