VIRUS ANALYSIS

LOOK AT THAT ESCARGOT

Peter Ferrie
Symantec Security Response, USA

In 2003 I wrote 'A recompiling virus like W95/Anxiety, but without needing the source code, combined with an inserting virus like W95/ZMist, but without rebuilding the file manually ... The beast is unleashed' (see *VB*, April 2003, p.5). Now, hot on the heels of MSIL/Impanate (see *VB*, November 2004, p.6), which introduced inserting viruses for the *.NET* platform, comes MSIL/Gastropod, which brings the full set of techniques one step closer.

DO NOT PASS 'GO'

Gastropod begins by calling a method named 'Go', which calls another method, which calls two other methods, which … The virus appears to have been written by someone who has been introduced to classes in C#, and has embraced them with such enthusiasm that every few lines of code are candidates for a method. It is rather like the saying, 'when all you have is a hammer, everything looks like a nail'.

This virus searches in the current directory and all parent directories for files whose suffix is 'exe'. For each file it finds, Gastropod checks if a file exists with the same name, but preceded by an '_' (underscore). If such a file does not exist, then the virus attempts to rename the original file to this name. The renamed file will be used as a backup in case the infection fails for some reason, however the virus will delete the file if the infection is successful.

There are several bugs in the handling of files, often resulting in the virus exiting with errors such as that the '_[filename]' file cannot be found, or that a file cannot be created because it exists already.

DUE PROCESS

Gastropod was named 'Snail' by its author, because of the slow speed with which it executes. Of course, this is not helped by the poor coding style of the virus author, but an additional slowing factor is the way in which the virus detects if a file is infected already. The infection marker can be found by the virus only by opening each file, and examining each method corresponding to each type that is stored within each module. The file is considered to be infected if any type and length of a method matches that of the virus itself.

If the file is not infected already, then the virus will construct a list of types, marking which of those can be renamed safely. Fortunately, the virus will rename only its own classes and methods. If the current method being examined is the entrypoint method, then the virus will place its classes at the top of the list, before adding those of the host.

GARBAGE MAN

Once the list is complete, the virus disassembles itself and the host code, using the well-known ILReader.dll utility, which is carried by the virus. The virus appends this file to the host during replication, which accounts for nearly 60kb of the total infection size.

The disassembly is used by the virus to insert garbage instructions throughout the code of itself and the host. The garbage instructions are either a NOP (with 20 per cent chance), or a LDLOC and POP combination (with 10 per cent chance, and only if the current method contains local variables).

If the LDLOC instruction is used, then the virus will choose the index randomly from the local variables of the current method. The virus also removes these instructions if they are found during the disassembly process.

It is unclear why the virus detects and removes those instructions, but one possible explanation is that it is an artifact from the development of the metamorphic engine, prior to the addition of the virus body – since legitimate programs usually do not contain such instruction sequences. If that is the case, Gastropod shares similarities with W95/ZMist (see *VB*, March 2001, p.6), which contained a routine that inserted redundant JMP instructions into the host, without adding the virus body.

UNEXCEPTIONAL

The virus is aware of the exception handling mechanisms in MSIL, and handles them mostly correctly – however not everything goes according to plan.

There are a number of bugs in the parsing, resulting in always duplicating 'endfinally' instructions, and producing unreachable (and, in some cases, incorrect) 'leave' instructions. These are things that would have been obvious

to a tester, since they are visible in the sample that the virus author released.

This also makes it extremely difficult to restore a file to its state prior to infection. To complicate things further, the virus extracts the managed resources from the host, and stores them externally, in a file whose name is the same as the host, with '.resources' appended. Finally, if the host contained unmanaged resources, the virus will move the resources to the end of the file, and place them in a newly created section that is always named '.rsrc'.

If the entrypoint method ends with a 'ret' instruction, then the virus inserts code to abort the thread. This prevents the process from hanging around in memory, instead of terminating, and waiting for the virus code to complete, which would be suspicious to some users. A similar problem exists for Win32 viruses that hook the ExitProcess() API of a host.

After disassembling the virus code and the host code, the virus renames its classes and methods. The renaming is done using 6–15 letters, with a 12.5 per cent chance of an upper case letter. The virus avoids renaming the 'Go' method though, since the way in which the virus inserts the code into the host's Main method results in a hard-coded method name.

CONCLUSION

The acceptance of the *.NET* platform has been slow so far, but it is increasing, and the complexity of viruses for that platform has already progressed much further, in a much shorter time, than was the case for earlier *Windows* platforms.

The full potential of the *.NET* platform has not yet been realised by virus writers, but it is clear that they are working hard to reach that goal. We have received the warning, and our anti-virus engines must be prepared. For some companies, the *.NET* platform might be the next OLE2.

MSIL/Gastropod

Size:	77828 bytes.
Type:	Direct action, parasitic inserter.
Infects:	Microsoft *.NET* files.
Payload:	None.
Removal:	Delete infected files and restore them from backup.