VIRUS ANALYSIS

CRISS-CROSS

Peter Ferrie
Symantec Security Response, USA

Cross-infector viruses demonstrate the flexibility of certain file formats. While some of these viruses have clearly been written to maximise their replication potential (e.g. {W32/Linux}/Peelf, which infected 32-bit *Windows* and *Linux* files, or the member of the W32/Chiton family that infected both 32-bit and 64-bit *Windows* files), most seem to have been written simply to show that it can be done. With the release of issue 6 of the *RRLF* zine in July (published on http://www.rrlf-zine.de.vu/), we received three new cross-infectors, each for a different set of file formats.

MONAD... NO MAD

The first of these viruses is one of a set that captured the media's attention. The virus (which we did not name) is part of a set of viruses for the forthcoming *Microsoft Shell*, or *MSH*, also known as 'Monad'. (Interestingly, one use of the word monad is to signify 'the One' – perhaps the developers at *Microsoft* thought they wouldn't be taken seriously if they called it 'Neo'.)

Originally, *MSH* was expected to ship with the forthcoming *Windows Vista*, but it has since been dropped from the initial feature set – which makes rather a non-event of any viruses written for it.

The virus in question attempted to infect .BAT, .MSH and .CMD files. However, due to a bug which should have been obvious during testing, the .BAT and .CMD forms do not prepend the virus code to the target file, as the virus author would like. Instead, the virus code replaces the target file entirely.

The bug is caused by the virus using 'copy <a>+<b> <b>'. Since <b> is not read before the copy begins, <a> replaces it entirely. To prepend via a copy requires two operations: one to copy the combination to another file, the other to copy that file over the original.

The .MSH code does work as intended and correctly infects all three file types. However, none of the replication types will infect a file whose read-only attribute is set.

Since .BAT and .CMD files differ only in their extension, the virus is really only a two-platform cross-infector. The .BAT/.CMD form of the virus is able to work by relying on the fact that the *Windows* command interpreter will consider lines that are not valid batch commands to be references to external files. Since (presumably) those files do not exist, the *Windows* command interpreter will display error messages instead, but it will then continue to interpret the file.

YOU HAVE NO NEW MESSAGES

The virus attempts to hide its activities by switching off message printing, but some messages (such as those produced by the copy operations) cannot be suppressed, except on DOS, *Windows 9x* and *Windows ME*. It seems that the virus author tried to hide those too, by clearing the screen, but the command to clear the screen appears before the copy operations, so the messages remain on the screen after the replication has completed.

The .MSH part of the virus is able to work by relying on the fact that, as in JScript, an end of line character is not considered to be a delimiter if it appears between the tokens of a statement. Thus a statement can span several lines without causing an error. This is in contrast to VBScript, for example, where each statement must appear entirely on a single line (although multiple statements can appear on the same line, delimited by the ':' character). The several lines in this case form the .BAT/.CMD replication code, after which comes the .MSH replication code.

SEEK AND YE SHALL FIND

The replication from .BAT and .CMD files is achieved by extracting those lines that contain a keyword (the name that the virus author gave it), which appears in every line of the code. These lines are placed into another file, which is then supposed to be prepended to the target files, but as described above, that part simply overwrites the file instead.

In addition, a line of .BAT code that is never executed would have caused some unexpected behaviour if it had been executed. The most obvious effect would be that during subsequent replications, the screen would no longer be cleared at all.

The replication from .MSH files is achieved by searching for files whose extension matches any of the three target extensions, then finding the last of those files which begin with the keyword. Having found such a file, the virus searches again for files whose extension matches any of the three target extensions. The virus then prepends its code to any file that is not already infected, by copying a fixed number of lines of code.

VBJSCRIPT

The second and third new cross-infector viruses are written by a different author. The second, {VBS/JS}/Cada, infects both VBScript and JScript files by appending the virus code to the target file. It is able to work by relying firstly on the fact that in VBScript the 'rem' command causes the rest of the line to be ignored, while in JScript 'rem' is considered to be an acceptable name for a variable, so the virus assigns a value to it. After that, the entire JScript virus appears on a single line.

Secondly, the virus relies on the fact that in JScript, the '/*' and '*/' symbols constitute a pair that bound a multi-line comment. Thus, at the end of the JScript code, the '/*' symbol appears to begin the comment, followed by the VBScript code on the next line.

Finally, the virus relies on the 'rem' command to cause the rest of the line to be ignored by VBScript, followed immediately by the '*/' symbol to end the JScript comment.

In between, the code searches for all JS and VBS files that can be found in the current directory, and infects any files that are not infected already. The infection marker is the string 'rem=1'. However, since the virus performs no tokenisation of its own, it will consider a file to be infected even if it contains that string as part of a longer string (e.g. 'members_of_harem=1').

OPEN OFFICE

The last of the viruses is a set of four variants that form the {O97M/VBS/JS}/Macar family. They infect the *Microsoft Office* applications *Word*, *Excel*, *PowerPoint*, *Access*, *Project* and *Visio*. The .B and .C variants also infect VBScript files. The .D variant infects JScript files instead of VBScript files. The most interesting thing about the .A and .B variants of this virus is that, unlike typical cross-infectors, which execute different code depending on the file type, this code is exactly the same for all of the *Office* applications and for VBScript.

Infected *Office* documents for *Word*, *Excel*, *Project* and *Visio* execute their macro code automatically, via an auto-macro. Infected *PowerPoint* and *Access* documents require some user interaction to execute: the *PowerPoint* macro runs when the user clicks on a slide during a slideshow (which is made possible by the presence of a transparent AutoShape, which covers the entire slide), and the *Access* macro runs whenever a user opens the first form in the database.

The replication method for .A and .B is unusual – the virus exports its code to a file, reads back that file, removes everything but the virus code (*Office* applications add additional text when exporting macros), then adds what remains to other files. However, this method avoids the blank-line insertion problem that some macro viruses encounter.

TRUST ME

The Visual Basic Object Model was extended in *Office 2000* to prevent macros from accessing themselves, which means that a virus can no longer export its code to a file, then import the file to other documents. The .C and .D variants of Macar work around this limitation by creating a macro that carries the whole virus code and writes it to disk. The dropped code is then executed by the macro. Since the external file performs the replication, no reference to the macro code itself is required. This is also an effective anti-heuristic device, at least from the perspective of the macro platform, since the macro does not replicate, although the external file that runs is highly suspicious.

The script begins by setting the VBA security settings for the chosen application to the lowest level. It knows how to adjust the settings for all *Microsoft Office* versions, from *Office 97* up to the as-yet-unreleased *Office '12'* (the virus author guessed the names of the registry values correctly). The virus works in all the pre-release version of the *Office '12'* applications, with the exception of *PowerPoint*.

GET TO THE POINT

One of the more surprising behaviours, from the user's point of view, is the occasional visible launching of *PowerPoint*. Macar uses *OLE Automation* to infect documents, which is done by running the application, and scripting the actions to take. Thus, whenever Macar decides to infect a *PowerPoint* document, *PowerPoint* is launched (if it was not running already). The reason the launching of the program is visible is because *PowerPoint* does not allow its main window to be hidden, unlike the other target *Office* applications. *Visio* also behaves in an unusual manner – the splash screen is visible, but the main window is not.

Another surprising behaviour is that of *Project* which, once it appears in the Task List, never goes away. This occurs when Macar decides to infect *Project* documents, because *Project*, along with *PowerPoint*, does not allow multiple copies of itself to be running at the same time.

CONCLUSION

Cross-infectors present some interesting technical hurdles for virus writers and, to a degree, for anti-virus writers too (since the target platforms must be identified and replication on those platforms is required for correct naming – the appearance of the sample can also differ there in significant ways, which can affect the detection).

While virus writers' time is best spent doing entirely non-viral things, whatever slows them down is the next best thing.