FEATURE SERIES

Macro Viruses – Part 1

Dr Igor Muttik
AVERT Labs, UK

Macro viruses appeared about four years ago and are now the most prevalent in the field. Their number is growing very quickly (currently about 5,000). The macro virus category is developing swiftly and many new terms and notions are invented constantly, so it might be difficult to keep up to date with them. It is easy to get lost in words like 'mating', 'remnants', 'downconversion' until you know what they actually mean.

This series gives an insight into the environment in which macro viruses live (OLE2 files), summarizes the main features of macro viruses and of their host applications, explains currently used terminology and provides a basic knowledge of how macro viruses operate.

What is a Macro?

Many applications provide the functionality to create macros. A macro is a series of commands to perform some application-specific task. Macros are designed to make life easier, for example by performing everyday tasks like text formatting or calculations in spreadsheets.

Macros can be saved as a series of keystrokes (the application records which keys you press). They can also be written in special macro languages (usually based on real programming languages like C and BASIC). Modern applications combine both approaches and their advanced macro languages are as complex as general purpose programming languages. When the language allows the modification of files it becomes possible to create macros that copy themselves from one file to another. Such self-replicating macros are called macro viruses.

A Brief History

Many software packages have a macro language – perhaps the very first well-known and widespread one was the *Lotus 123* spreadsheet. It was proved long ago that it is possible for *Lotus 123* to write a self-replicating macro (a virus) which will be able to travel from one file to another. However, viruses have never been a problem for *Lotus 123* as its macro language is rather simple and access to files can only be performed via menus. So, a virus for *Lotus 123* would be extremely obvious – you would literally see the infection process right on your screen.

In December 1994, the researcher Joel McNamara wrote the first real macro virus for demonstration purposes. It was called DMV (Document Macro Virus). In fact, there were two viruses written – DMV for *WinWord* and DMV for *Excel*. The samples were used to demonstrate the possibility of macro viruses on these platforms. The first field macro virus – WM/Concept – appeared in the summer of 1995 and soon became the most widespread virus ever.

Platforms and Applications Supporting Macros

Most macro viruses are written for *Microsoft WinWord* and *Excel*. Viruses for *PowerPoint 97* also exist, even in the wild (PP97M/Tristate). However, there are also experimental macro viruses for *AmiPro* (Green_Stripe), *CorelDRAW* (CSC/CSV, etc.), *Access 97* (AccessiV, etc.) and several multi-partite viruses which infect executable files and *WinWord* documents (Anarchy.6093, Heathen).

Macro viruses can work on any machine carrying, say, *WinWord* – be it a PC, a Macintosh or a DEC Alpha computer. Macro hosting applications are able to work under many operating systems – *Windows 3*, *Windows 95*, *Windows NT*, MacOS, *SoftWindows*, etc. There are certain differences in implementations of the macro languages on different machines (OS support is usually slightly different, especially for the filesystem objects) but nevertheless, many macro viruses can spread successfully on very different types of computers and operating systems.

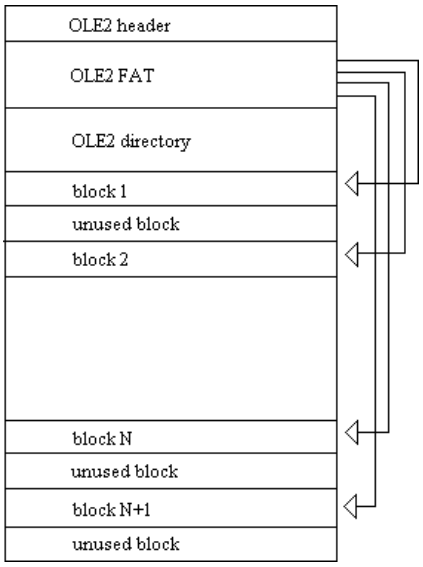
OLE2

Files produced by *Microsoft* applications (DOC documents, XLS spreadsheets, PPT presentations created by all versions of *WinWord* above 6.0, and all versions of *Excel* and *PowerPoint*) are stored in so-called OLE2 files (note, *MS-Access* files are not OLE2). OLE stands for Object Linking and Embedding. It is just a standard describing a file structure that is able to store many different streams within one file. An OLE2 file is a file system within a file.

OLE2 files contain a special signature at the beginning (D0 CF 11 E0 – which stands for DOCFILE), the FAT (File Allocation Table), and a directory just like a normal DOS disk. Space inside an OLE2 file is allocated in blocks referenced from the OLE2 FAT.

The access to OLE2 files is supposed to be gained through APIs provided by OLE2.DLL and OLE32.DLL. These DLL files support all necessary functionality to work with OLE2 files (like add/delete/modify stream, open/update an OLE2 file, etc.) The OLE2 technology is being licensed to other software producers, so many vendors are now supporting this format.

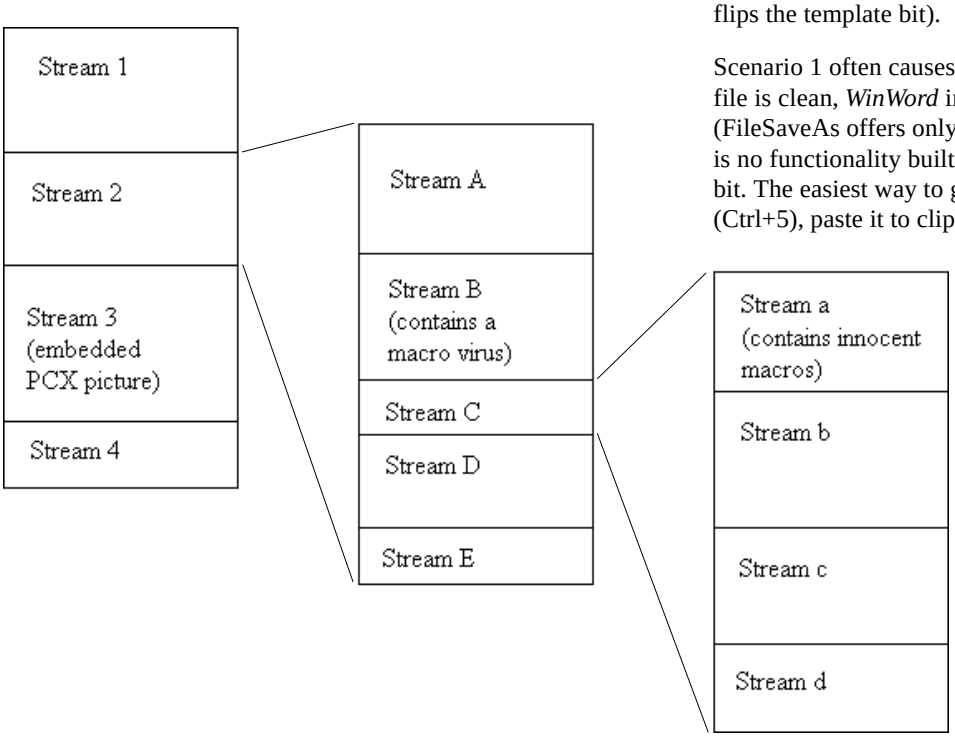
The flexibility of the OLE2 format allows the storage of many not necessarily related items (they are called streams) inside just one file. For example, the first stream of an OLE2 file may hold the text, the second another OLE2 file, the third an embedded picture, etc, see diagram.



As you can see, OLE2 files may have unused blocks. They usually contain information from previous saves of the file or just some random data from memory (even the important private data you may not want to be included in any file!).

A very common situation is when, say, an XLS file has a lot of strings reading 'laroux' inside but they are all in the unused space. This situation frequently causes users to panic while the file appears to be clean with no macros, let alone infected by the Laroux virus. To avoid files having unused clusters uncheck 'Allow fast saves' in Tools/ Options/Save.

In the following example the first OLE2 file has an embedded OLE2 file as Stream 2 and a PCX file as Stream 3. The embedded OLE2 storage contains a virus in Stream B and another embedded OLE2 storage as Stream C. This third OLE2 contains some macros in Stream a. This multi-level structure is all held within one real file which is organized as a file system on each level (because embedded objects are also in OLE2 format).



In an OLE2 file it is possible to embed an XLS into a PPT file (or an EXE into DOC, or a DOC into PPT). In this way, tree-like structures within OLE2 may be created. If the embedded object is being double-clicked on, its contents are activated and macros (if any) may be executed.

A further complication is that to save space, *PowerPoint* stores embedded OLE2 files in compressed form. So a PPT file is an ordinary OLE2 file but it can have another compressed OLE2 embedded. To be able to scan for macro viruses inside the OLE2 files on all levels of embedding scanners usually use their own OLE2 parsing and decompressing engine. That allows the scanning of *WinWord* documents directly, even without support of *Windows'* OLE2.DLL and OLE32.DLL. Decent scanners are able to scan OLE2 files even under DOS and *NT* on a *Novell* server, a Unix machine, Macintosh, DEC Alpha, Sun, etc.

Template bit, DOC/DOT

WinWord 6/7 documents have a special bit inside which says whether the current document contains anything but text. *WinWord 6/7* does not look for macros if the template bit is zero. Normally, DOC files have this bit reset (zero) and templates (DOT) set to 1. However, the bit itself is not linked to the file extension (and on Macintosh there are no fixed extensions for files).

So, it is possible to have: 1 – a file with no macros and the template bit set (this normally does not happen but can happen when all macros are removed from the DOC file), 2 – a file with macros (e.g. virus) and the template bit set (this is a normally infected file), or 3 – a file with macros (e.g. virus) and the template bit reset (this means the virus is inactive, or 'dormant' – it will not infect until somebody flips the template bit).

Scenario 1 often causes the user confusion. Even when a file is clean, *WinWord* insists on saving it as a template (FileSaveAs offers only 'Document Template' type). There is no functionality built into *WinWord* to clear the template bit. The easiest way to get rid of it is to select whole text (Ctrl+5), paste it to clipboard (Ctrl+C), close the file (Ctrl+W), start new file (File/New) and paste the text back (Ctrl+V). Now FileSaveAs will work fine.

Office 97 and *Office 2000* ignore the template bit and check for the presence of macro storage. However, it is possible to have an *Office 97* file with empty storage (i.e. no macros). Then an *Office* application would display the macro warning box even for a file with no macros whatsoever.

[*Next month, the second instalment covers WordBasic,VBA, up/down conversion and polymorphism. Ed.*]