FEATURE SERIES

Macro Viruses – Part 2

Dr Igor Muttik
AVERT Labs, UK

Most of the old macro viruses work under and were written for the *WinWord 6.0* macro language which is known as WordBasic. The commonness of the *Word 6.0* environment at the time enabled *WinWord* macro viruses to become widespread. WordBasic is based on the good old BASIC programming language but has many (hundreds) of extensions (for example to deal with documents: edit, replace string, obtain the name of current document, open new window, move cursor, etc.).

WinWord 7.0 was included in *Office 95* and it also used WordBasic. The appearance of macro viruses forced *Microsoft* to release a version with some sort of protection. *WinWord 7.0a* was the first *MS* application to have the built-in anti-virus warning mechanism which is now present in almost all *Office* applications.

VBA3, VBA5, VBA6 and Excel Formulae

Excel 5.0 had a macro language called VBA3 (Visual Basic for Applications v.3). This was used as a prototype for VBA5 – the macro language for *Office 97* applications. VBA6 is the one used in *Office 2000*. For the list of new features and differences see *VB*, August 1999, p.13.

Old *Excel 4.0* had formula macros. They were kept for compatibility in all later *Excel* versions. There are also field viruses (e.g.XF/Paix) that use this macro language. Formula macros are written in an *Excel*-specific language that has nothing in common with either WordBasic or any VBA flavour and they live in the *Excel* spreadsheet's cells.

Visual Basic for Applications

In January 1997 *Microsoft* unveiled *Office 97* – it was a complete rewrite, no longer using WordBasic. In *Office 97* all applications use the same macro language – VBA5 (Visual Basic for Applications). *WinWord 8.0* (*WinWord* in *Office 97*) has an ability to convert (recompile) old macros into this new language. Many viruses can be recompiled this way resulting in completely different viruses (some of them non-viable as the convertor success rate was estimated by *Microsoft* at about 90%).

Further, *Microsoft* put some sort of detection of the most common viruses into the convertor to prevent their recompilation (so that, for example very common viruses like WM/Concept.A, WM/Wazzu.A and WM/Npad.A are not converted). Unfortunately, these precautions were not made in *Office 97* beta releases and several viruses were upconverted to the new format by the beta software.

Another feature of *Word 97* is that it produces a warning (like *WinWord 7.0a*) if somebody is trying to load a document containing macros. It displays a dialog box saying 'The document you are opening contains macros or customizations. Some macros may contain viruses that could harm your computer.' and presents three options:

1) Disable Macros (default)

2) Enable Macros

3) Do Not Open

This warning, however, can be turned off so that it will never appear again.

VBA5 is a far more complex language than WordBasic (in fact, it even includes WordBasic as a subset of its commands) and its data is stored in a file in a much more complex way. Macros written in VBA5/6 are represented in OLE2 files by two different entities – there is a compiled macro body and also compressed macro text (both are usually present in OLE2 files with macros). When macro text gets modified the macro body is recompiled from it.

Usually, both instances of a macro contain the same information (in simple terms, one is used by the VB editor, another by the VB interpreter). However, in the case of corruption this may be not true – for example, even if macro text is missing the compiled body could still be executed. Different scanners may choose to detect macro viruses in either form (compiled body or compressed text). This explains why different scanners may produce incoherent results on some corrupted samples (having, say, one form missing or damaged).

Under *Office 97* all major applications use the same macro language. That means cross-application viruses are possible (see *VB*, October 1998, p.9). What is more, files with PPT extensions (*PowerPoint 97*) can now have macros (which previous incarnations of *PowerPoint* did not have at all). Naturally, the first *PowerPoint* viruses appeared after *PowerPoint 97* had been released.

Upconverting and Downconverting

Excel 97 has an ability to save spreadsheets in old *Excel 5.0* format (i.e. in VBA3). So viruses in VBA5 format can be 'downconverted' back to VBA3 format. It is even possible to have both VBA3 and VBA5 incarnations of macros in a single spreadsheet file, recognizable by both old and new *Excels*. Downconverted viruses can be upconverted again, resulting in exactly the same virus body. However, it is known that the virus' formatting (for example, spaces, tabs and empty lines) does change. This is why, in the proper identification of *Excel* viruses, these variable parts should be ignored.

Following the release of *Office 2000*, macros written for *Office 97* (VBA5) could be upconverted to *Office 2000* (VBA6) and downconverted back to *Office 97*. Any anti-virus scanner should be capable of automatically dealing with all differences which appear as a result of multiple up/down-conversions.

Many viruses do not survive this (the convertor adds an empty line for every downconversion which breaks most common viruses). However, even among broken viruses a scanner should be able to find and clean the non-viable, damaged viruses.

VBA6 macros do not differ much from VBA5 ones. In fact, VBA6 includes VBA5 as a subset. Macros written for *Office 97* usually work under *Office 2000* while the opposite is not always true (i.e. *Office 2000* is downwardly compatible with *Office 97*). Decent scanners are not able to distinguish between *Office 97* and *Office 2000* incarnations of the same virus, performing internal mapping to cover up/down-conversions automatically.

Typical Life-cycle of a Macro Virus

The life cycle of a great majority of *WinWord* macro viruses is as follows. A macro virus in a document which is being loaded gets control (for example via so-called auto macros, those which get executed automatically at certain moments; such macros are – AutoOpen, AutoClose, etc). The corresponding macro copies all the viral macros to the global template (NORMAL.DOT on a PC). NORMAL.DOT is used automatically when *WinWord* starts. It contains user settings (fonts, etc) and shortcuts (key redefinitions) and it can also contain macros.

If NORMAL.DOT contains the AutoExec macro it will be executed when *WinWord* is started. If NORMAL.DOT contains AutoClose it will be executed every time any document is closed.

Macro viruses do not necessarily have to infect the global template (NORMAL.DOT). Some infect files directly, searching for a victim on a disk and infecting it that way. WM/Snickers and WM/Ordo use the MRU list (most-recently-used list at the bottom of the File menu usually consisting of four items) to get the names of files to infect. Others (like the W97M/Groov family and WM/Eraser) drop their own template in *WinWord's* template directory and may or may not also infect NORMAL.DOT. This additional template (if it is registered as an add-in) will work in exactly the same way as NORMAL.DOT and *WinWord* will pick it up automatically.

In many ways, *Excel* infectors are similar to *Word* infectors. However, instead of infecting NORMAL.DOT, viruses written for *Excel* usually drop a new startup file in the XLSTART folder. *Excel* automatically picks up such dropped files when it starts. The most common name is PERSONAL.XLS (which is *Excel's* default name for a startup file; Laroux.A uses it). However, the name can be anything – Laroux.e uses PLDT.XLS. *Excel* does not bother to check the extension so many viruses drop a file with no extension like 'BOOK1.' (e.g. O97M/Tristate.A).

Auto Macros

Most macro viruses operate using auto macros. *Word 6.0* had just a few of them (AutoOpen, AutoClose, AutoExec, AutoExit and AutoNew). VBA5/6 have many more. Apart from the old ones kept for compatibility, there are also 'event handlers' (Document_Open, Document_Close, Workbook_Open, Workbook_BeforeClose, etc.).

Event handlers should be put in special 'class' modules to be able to work (hence the name of the first W97M virus to use event handlers – W97M/Class). Event handlers exist only in VBA5/6 and are not present in VBA3 (*Excel 5.0*). That is why many *Excel 97* viruses (X97M/Hopper, for example) cannot be downconverted to *Excel 5.0* (following downconversion there are no modules in the file).

In fact, probably every single field virus makes use of the 'Auto' macros. Viruses which do not use them do exist but their chances of spreading are simply below the threshold that enables them to survive in the wild.

Menu Items Interception and Key Shortcuts

It is easy to modify the functionality of any *Office* application by associating its menu item with a macro. For example, many viruses have the macro called FileSaveAs (in VBA it would be a function with the same name and it can be defined in any module). If this menu item is activated by a user it is the macro which gets control, pretending to be a real menu option while it copies additional viral macros to the destination file. Viruses can also remove and modify menu items (many remove the Tools/Macro item to make it impossible to check for the presence of viral macros) using the Tools/Customize functionality.

Macro viruses can attach a macro to a particular keyboard key. For example, WM/Gangsterz and WM/DLK1.a link their viral macros to frequently used keys (like space, 'e', 'a') and activate when this key is pressed. This is one of the ways macro viruses can avoid using auto-macros or menu-linked macros to get control.

Polymorphic Macro Viruses

There are many known polymorphic macro viruses in existence at the time of writing. A few examples include WM/FutureN, WM/Outlaw, WM/Slow, WM/Minimorph, W97M/Class.a and W97M/STP.

They all use *WinWord's* editing abilities to modify their own macros (like replace function) before copying them. This has the effect of making the virus body variable. Another approach to hiding parts of the virus is to use document variables which are stored in a file (for example, a WordBasic program can assign a string variable A$ and then save it in a file along with macros). Such variables can contain various bits and pieces of viral code/data which are used by viral macros.

Stealth and Encryption

Stealth for macro viruses involves some measures to prevent the easy viewing of a virus' source code. Normally, the host program is capable of displaying the source code of any macro or module. To prevent easy viewing, some viruses remove the Tools/Macro, File/Templates/Organizer and Visual Basic Editor menu items. Some viruses present the user with artificial, empty dialog boxes instead of real ones or produce fake errors.

Macros can be encrypted. Encrypted macros are simply stored in scrambled form (however, note that the encryption of macros does not affect the text in a document – it still is easily readable). So, there are encrypted and not encrypted macro viruses. When an encrypted macro is shown in Tools/Macro the option to edit is not available. The encryption is easy to overcome – the key to decrypt macros is in the file, so it is not a problem to scan encrypted macros for viruses. Macro encryption is also called 'macros are read-only' because the macro editor does not allow the editing of encrypted macros.

Macros encrypted in *Word 6.0* cannot be converted to either *Office 97* or *Office 2000* because the latter insists on protection being enabled on modules' projects, not individual macros (under both applications several functions/ macros can be defined in one module).

Office 97/2000 can also use 'read-only' macros (or, as *Office* puts it, 'lock project for viewing'). Such macros have a special flag set and cannot be edited in the Visual Basic Editor. Macros, however, are not actually encrypted in any way and macro bodies are easily accessible by any tool except the built-in macro editor.

Password Protection

Entire *WinWord 6.0/7.0* documents can be password-protected. This means that the whole file is scrambled and access to the text and macros is not possible without deciphering the file. The password is needed to access the macros and check them for viruses. However, the protection is weak and there are many shareware and freeware *WinWord* password crackers around. Many contemporary scanners are able to do on-the-fly cracking of password protected documents to check them for viruses.

Under *Office 97/2000* macros are not password-protected so scanning for viruses is not only possible but easy, even when the text is protected by a password. Repair, however, might be difficult as the area protected by the password may contain references to viral macro bodies. Furthermore, encryption in *Office 97/2000* is far more advanced and cannot be cracked on-the-fly.

Corruptions and Manual Editions

WinWord 6.0 has buggy routines that are responsible for macro copying. As was discovered by the author, any I/O error during macro copying produces unpredicted results on the destination macro. For example, if macro virus was in a document on a floppy disk and the disk was removed when the macro copying was being performed, parts of the written copy will be corrupted without any warnings.

In fact, there are about 200 different variants of WM/Npad around and all of them (except the original virus) are the result of the natural corruption described above. Between 1997 and 1998 this natural corruption was the main source of macro viruses because many of them are able to replicate even if they are seriously corrupted (the WordBasic statement 'On Error Goto Next' helps a lot).

In *Office 97* corruptions are very rare so they are not responsible for creating new viruses. However, the availability of the Visual Basic Editor (say, via Alt+F11) makes it very easy to modify the source of any virus manually (if it is not a 'read-only' macro). Users' modifications to field viruses are the most common source of new variants because the vast majority of manual modifications are perfectly able to travel even when the user is attempting to 'disable' the virus.

Remnants

Some macro viruses have just one macro (like WM/Wazzu, W97M/Ethan, WM/MDMA). However, many macro viruses consist of multiple macros (for example, WM/Rapi, WM/Concept and W97M/Aleja). It could and does happen that some of the macros belonging to a virus go missing.

This might happen because somebody deleted alien macros using the Tools/Macro/Delete command or some sloppy anti-virus tool that did not perform the disinfection correctly or because *WinWord* hung in the middle of the macro copying operation. Whatever the reason, we get a document with the remnants of the original macro virus. Remnants occur very frequently in *Word 6.0* viruses.

In most cases, remnants no longer constitute a viable virus because some part of the original is missing. However, in some cases remnants can still be viral. Say, WM/DZT has two macros (AutoOpen, FileSaveAs in documents) – what is left can replicate if either of the two is missing.

Under *Office 97/2000* remnants are rare. The reason is that VBA5/VBA6 allows several functions (like AutoOpen) and event handlers (like Document_Close) to be present in one single module. So, there is no necessity for a virus writer to place a virus in more than one module. *Office 97* viruses spanning several modules do exist but they are rare. That is why viable remnants of native W97M and X97M viruses have yet to be reported.

[*Next month's final instalment covers mating, devolving, naming and prevalence. Ed.*]