

# VIRUS ANALYSIS 1

## YOU'VE GOT MORE M(1\*\*)A(D)I(L+K)L

Peter Ferrie

*Symantec Security Response, USA*

Another day, another exploit is disclosed. A little over two months later, a virus using the exploit is discovered. It seems that some virus writers do read *NTBugtraq*. There is a new member of the W32/Chiton family. The author of the virus calls this one 'W32/JunkHTMaiL', a variation of the name of the virus upon which it is based –W32/Junkmail (see *VB*, November 2002, p.10) – perhaps to draw attention to the self-executing HTML exploit which this virus uses to launch itself from email.

When JunkHTMaiL is started for the first time, it decompresses and drops a standalone executable file that contains only the virus code, using a 'fixed' (taking into account the variable name of the *Windows* directory) filename and directory.

As with the other viruses in the family, this virus is aware of the techniques that are used against viruses that drop files, and will work around all of the countermeasures: if a file exists already, then its read-only attribute (if any) will be removed, and the file will be deleted. If a directory exists instead, then it will be renamed to a random name. The structure of the dropped file is the same as that used by W32/Junkmail. If the standalone copy is not running already, then the virus will run it now. The name of the dropped file is 'ExpIor.exe'. Depending on the font, the uppercase 'i' may resemble a lowercase 'L', making the viral process difficult to identify in the task list.

### HOOK, LINE, SINKER

After dropping the standalone copy, the virus alters the Registry in such a way that the virus will be run whenever an application is launched.

The virus alters the 'Shell\Open\Command' keys for the 'com', 'exe', and 'pif' extensions in both the 'LocalMachine' and 'CurrentUser' hives. Both hives are altered because, in *Windows 2000* and *XP*, the Current User values override the Local Machine values. The three extensions are altered because they are all associated with applications. In addition, the change makes removal of the virus more difficult – if the virus is removed before the Registry is restored, then applications will not be able to be launched easily. Fortunately, some improvisation allows for ways around this problem.

If the computer is running *Windows NT/2000/XP*, then the virus will add itself as a service. The virus does not start the

service, perhaps because the standalone copy is running already, and *Windows* will perform that action anyway, when the computer is rebooted. If the computer is running *Windows 9x/Me*, the virus will place an undocumented value in an undocumented structure, with the result that the task is not displayed in the task list. This mimics the actions of the recently documented and already very well-known RegisterServiceProcess() API.

### IT TAKES TWO TO ARGUE

Whenever the standalone copy is executed, the virus will parse the command line to determine why it is running. The parsing is done in the platform-independent way that is favoured by the virus author – if the computer is running *Windows 9x/Me*, then the virus will use the ANSI APIs to examine characters; if the computer is running *Windows NT/2000/XP*, then the virus will use the Unicode APIs to examine characters. If there are arguments on the command line, then the virus assumes that it was launched via the Registry alteration, and will attempt to execute the application that is named in the first argument.

If there are no arguments on the command-line, then the virus assumes that it has been launched as the standalone copy, and will execute its main code. The main code begins by retrieving the addresses of the APIs that it requires and creating the threads that will allow the virus to perform several actions simultaneously.

The first thread runs once every hour. It will enumerate all drive letters from A: to Z:, looking for fixed and remote drives. If such a drive is found, then the virus will search in all subdirectories for files to infect. Files will be infected if they are *Windows* Portable Executable files for *Intel 386+* CPUs, and are not DLLs.

The method of infection is the same as for some of the other variants in the family – the virus will either append its data to the last section, or insert its data before the relocation table, and alter the entry point to point directly to the virus code. If files do not possess the infection criteria, the suffix of their name is checked against a list of files that might contain email addresses. The virus is interested in files whose suffix is 'asp', 'cfm', 'css', or 'jsp', or contains 'php' or 'htm'. If such a suffix is found, then the file is searched for a 'mailto:' string, and the email address that follows is saved for later.

The second thread runs once every two hours. It will enumerate the network shares and attempt to connect to them. If the connection succeeds, then the virus will search all subdirectories for files to infect.

The third thread also runs once every two hours. It will attempt to connect to random IP addresses. There are two

routines for this action, one for ANSI platforms, and one for Unicode platforms. If the connection succeeds, then the virus will search in all subdirectories for files to infect.

The fourth thread is the one from which the virus gains its name. It runs once every four hours, and will send a single email to the last address that the virus found while searching for files to infect. The virus sends itself using the MIME message format, as described in RFC 1521, and carries an attachment using the MHTML document format, as described in RFC 2557.

While this should present no problems, it appears that a number of developers have overlooked one significant aspect of the formal BNF of, for example, the content of the MIME-Version field. This is that the colon and digits, etc., are separate tokens, and that no white space is required. The formal BNF for the content of the MIME-Version field looks like this:

```
version := "MIME-Version" ":" 1*DIGIT "." 1*DIGIT
```

and a typical MIME-Version declaration looks like this:

```
MIME-Version: 1.0
```

However, when considering the tokens individually, the result is that these are equivalent:

```
MIME-Version      :      1 .      0
MIME-Version:1.0
```

with the obvious problems for those parsers that don't expect white space to appear, or that require at least one space after the colon. The virus attacks the second assumption, by removing the space in all cases.

## LAYER UPON LAYER

*Microsoft* introduced the 'Web Archive' format after the release of *Office 2000*. It is based on the MHTML standard, and resembles a MIME email file, complete with MIME-Version, a Content encoding field, and 'attachments'. An unfortunate consequence of this choice is that such files, when sent as email attachments, can be encoded recursively. Thus, the beginning of such a file might look like this:

```
MIME-Version:1.0
CONTENT-LOCATION:FILE:/// .EXE
CONTENT-TRANSFER-ENCODING:BASE64
```

However, after recursive encoding of the type implemented by the virus, it might look like this:

```
=4DI=6DE=2D=76=65=52s=49=6F=6E:1=28=43H=29.=30
=63ONT=45=4E=54=2D=4CO=63=61=54=69=4F=4E=3AFIL=65:/
/=2F=2E=45xe
=63=6Fn=74=45=6E=74-t=72=61=6E=53=46=45=72-
=65NcOd=49=6E=67=3A=62(=4B=7B=
)a=28=7B+=29s=28=45=29e6(=77Y)4
```

The top level is octet-encoding. It exists to support the sending of characters that are not within the acceptable ASCII range (i.e. foreign and reserved characters), however any character can be encoded with this method. If the octet-encoding is decoded, as will occur when an email program detaches the attachment, it might look like this:

```
MIME-veRsIon:1(CH).0
cONTENT-LOcAtIOn:FILE:/// .Exe
contEnt-tranSFER-eNcOdIng:b(K{)a({+)s(E)e6(wY)4
```

Now we see the case inversion and comment insertion that was first demonstrated by W32/Junkmail.

## NOT A BUG, BUT A FEATURE

An email sent by the virus will have an attachment called *Email.htm*. This is a Web Archive file that has a script appended to it. When a file is passed to *Internet Explorer (IE)*, *IE* will search a large amount of that file for HTML code. This is, according to *Microsoft*, by design.

Thus, a MHTML file with a script appended to it can have that script executed, even though the file does not begin with the '<HTML>' tag. The virus uses a script that requests *IE* to run the file that is located inside the same MHTML file. If the *IE* security settings allow the scripting of ActiveX controls that are not marked as safe, then the file will be launched without prompts, regardless of the zone in which it is executing. *Microsoft* has released a patch (MS03-014) which disables MHTML as a codebase source. The patch is described as applying to *Outlook Express*, however the file that does the work (*inetcomm.dll*) actually belongs to *Internet Explorer*.

## CONCLUSION

The world of security and the world of viruses have become intertwined over the years, and so far we have been fairly lucky that, despite the full disclosure of many exploits, very few have been used in viruses. The successful virus requires knowledge and luck, and while we can't defend against luck, we can see that too much knowledge can be a bad thing.

### W32/Chiton variant

Type:	Memory-resident parasitic appender/ inserter, slow mailer.
Infects:	<i>Windows</i> Portable Executable files.
Payload:	None.
Removal:	Delete infected files and restore them from backup.