VIRUS ANALYSIS 2

You've Got M(1**)a(D)i(L+K)l

Peter Ferrie
Symantec Security Response, Australia

Encryption techniques have evolved over the years, from simple bit-flipping, through polymorphism, to metamorphism, and combinations of these have been used as well (for example, see *VB*, May 2002, p.4). All of these techniques have one thing in common: they are applied to the virus body. The alternative is to apply them to the thing that contains the virus body. This variant of the Chiton family, which the virus author calls W32/Junkmail, is one of those.

To the Manner Born

When Junkmail 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, Junkmail is aware of the techniques that are used against viruses that drop files, and will work around all of the counter-measures: 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/Gemini (see *VB*, September 2002, p.4) and W32/EfishNC (Junkmail is based very heavily on that virus). If the standalone copy is not running already, then Junkmail will run it now. The name of the dropped file is 'Exp*I*orer.exe'. Depending on the font, the uppercase 'i' will resemble a lowercase 'L', making the viral process difficult to see in the task list.

Hook, Line, Sinker

After dropping the standalone copy, Junkmail will alter the Registry in such a way that the virus is run whenever an application is launched. Junkmail 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. Additionally, the change makes removal more difficult because if the virus is removed before the Registry is restored, then applications cannot 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*, then the virus will place an undocumented value in an undocumented structure, which results in the task not being displayed in the task list. This mimics the actions of the undocumented 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.

Threads

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 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 entrypoint to point directly to the virus code. For files that 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 in 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 after every six hours, and will send a single email to the last address that the virus found while searching for files to infect. It is also here that the encryption is applied to the container, rather than the virus body. The virus sends itself using the MIME message format, as described in RFC 1521. While this should present no problems, it appears that a number of developers have overlooked one significant sentence: 'All header fields defined in this document, including MIME-Version, Content-type, etc., are subject to the general syntactic rules for header fields specified in RFC 822. In particular, all can include comments'. The result is that an email that would normally look like this:

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
 boundary=TFICLMGJ
```

can be altered to look like this:

```
M(F)IM(])E-(*/
*)V(y)e(7)r(*)s(U*0)i(*LZ)o(H)n(.):(l)
1(:*=).0
Content-Type: mul(26)t(fH*)ip(|*)a(***)rt(*)/
mi(/*j)x(8)e('M)d;
 (<|)bo(*,)u(1**)nda(D)r(L+K)y=TFICLMGJ
```

In case that wasn't bad enough, the virus contains an abundance of other tricks – the subject is chosen randomly from a list, or in some cases will contain only a random filename and no other text. The message body contains variable parts, so one message body could begin with:

```
I received this file from you yesterday
evening.

I think it was sent without you knowing by
the Badtrans trojan.

The filename was altered but it looked like
an important document inside.
```

while another could begin with:

```
I received this file from you yesterday
morning.

I think it was sent without you knowing by
the Sircam worm.

The filename was changed but it looked like
an important picture inside.
```

I've Been Framed

The attachment type is chosen randomly from a list. Some of the types are those that are vulnerable to the IFrame exploit that allows automatic execution of the attachment. There are 22 of these types. The other types are those that will display the CID instead of the filename when prompting the user to open or save the attachment. The CID has been named 'email' with this in mind – the person who views the message will see a prompt to Open or Save an attachment called 'email', and will likely select Open. There are four of these types.

The filename of the attachment is also 'email', followed by one or two suffixes, chosen randomly from lists. Some viruses use '.bat' as a suffix even though the file is binary, however Junkmail uses the suffix in the correct way – if .bat is chosen, the virus sends itself as a real .bat file. If .shs is chosen, the virus sends itself as an OLE2 file. Otherwise, the virus sends itself in the *Windows* PE file format.

Layer upon Layer

The .bat method is an interesting technical achievement. There are certain characters that are interpreted differently on *Windows 9x/ME* and *Windows NT/2000/XP*. The virus author is aware of this, and the .bat code is able to determine the *Windows* platform and allow for the differences. Following the platform determination is a line containing executable code composed entirely of printable characters. The technique is known as 'executable ASCII'. The code is only 217 bytes long, but it is able to decode a base64 attachment, write it to a file, then launch that file. The decoder itself is only 59 bytes long. The rest of the .bat file is the base64-encoded copy of the virus. If the .bat file is executed, it will determine the *Windows* platform, create a temporary file and write both the decoder and attachment there, then run the temporary file. The temporary file will decode and run the attachment, which will launch the virus.

The structure of the OLE2 file is not constant either, thanks to a feature of *Windows*. The file contains only the absolute minimum number of components required to run – one storage and one stream. When the file is executed, *Windows* will automatically create the 'missing' storages and streams and update the file structure, resulting in a file that could be several times its original size.

Conclusion

The RFCs are full of features that many people might, but very few people do, use. This can lead to complacency among developers, leading to loopholes, leading to Junkmail and those that will follow it. Engine developers need to re-read the RFCs and implement support for even the most obscure features because, as is demonstrated here, these unusual features can be used for unusual purposes.

W32/Junkmail

Alias:	W32/Chiton variant.
Type:	Memory-resident parasitic appender/inserter, slow mailer.
Infects:	*Windows* Portable Executable files.
Payload:	None.
Removal:	Delete infected files and restore them from backup.