VIRUS ANALYSIS 1

Sharpei Behaviour

Péter Ször
Symantec Security Response, USA

In a recent article in *Virus Bulletin* (see *VB* March 2002, p.6) I described the internal details of W32/Donut. In my article I pointed out that, contrary to some reports, W32/Donut was *not* 'the first C# virus'. In fact, Donut contained no functional C# code.

Shortly after the appearance of Donut, however, we received another virus – W32.HLLP.Sharpei@mm, written by Gigabyte. Unlike Donut, this virus does contain functional C# code and is the first *real* C# virus.

Infection Trivia

Having stated 'it is not a trivial task to write a C# virus', I was asked recently by a member of AVIEN (Anti-Virus Information Exchange Network) whether the virus writer had beaten my expectations. Indeed, I had not expected the first C# virus to appear so quickly.

However, I still find it non-trivial to infect *.NET* files with code written in C# that will infect other *.NET* files at their C# entry point. Donut demonstrated how simple it can be to for a virus to infect a *.NET* file at its 32-bit entry point.

Infection techniques that do not pay attention to the *.NET* file format can be considered trivial. The HLLP method (High Level Language Prepender – the HLLP definition might differ from company to company but this is how we have always used it) is an example of such an easy technique. This simple direct action infection is carried out by a C# component of the virus code.

The virus works as a mass-mailer even when the *.NET* framework is not installed. This makes it a current problem rather than a future threat, although the VBS script that is used for the mass mailing can be detected by today's heuristics as well as script blocking techniques.

You Have Mail

W32.HLLP.Sharpei@mm arrives as an email message with the subject '*Important: Windows update*'. The body of the email contains the text: '*Hey, at work we are applying this update because it makes Windows over 50% faster and more secure. I thought I should forward it as you may like it.*' The email arrives with an attachment named 'Ms02-010.exe'.

The actual format of the virus is a regular 32-bit PE file. I imagine that the assembly portion would have been created by the NGVCK kit and then altered with new code. The size of the first generation sample is 12,288 bytes. This executable contains a VBS script as well as a *.NET* PE file which is written in C# and contains MSIL code.

When the attachment is executed, the virus makes a copy of itself as C:\Ms02-010.exe. It drops a VBS file with the name 'Sharp.vbs', which performs the mass-mailing routine, sending the message as described.

Finally, Sharp.vbs deletes itself. Once the messages have been sent successfully, they are deleted from the *Outlook* Sent folder. As a result, you will not see any evidence of the messages in *Outlook*. This is an attempt by the virus to hide its activity.

If Mscoree.dll is found in the \System folder, the virus creates 'Cs.exe' in the \Windows folder, then executes it. Sharpei makes the assumption that this library is present only when the *Microsoft .NET* Framework is installed. Cs.exe is a 7680-byte *.NET* executable that is written in C# and runs only in the *.NET* Framework.

Finally, Ms02-010.exe creates the HKLM\Software\Sharp key in the registry and sets this to point to the executed infected file. Thus, in the first execution, this might point to 'C:\mymail\ms02-010.exe'. This string is used later on as a reference from Cs.exe (the *.NET* component) to the executed attachment or infected application.

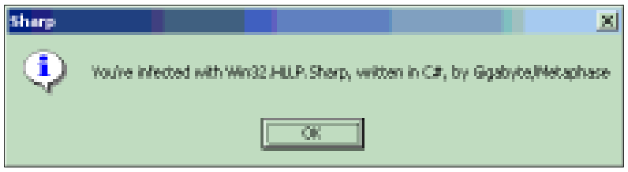
Cs.exe in Action

This is the portion of the code written in C# that implements buggy direct action prepender virus logic. It works properly for first-generation infections.

(In order to understand this code it is necessary to have IDA or the ildasm.exe utility to produce an output containing MSIL code. Reading MSIL will be a new skill for virus researchers to learn – at first glance the stack machine code is a little confusing to someone who reads assembly language, which uses registers for most tasks.)

First, the special '.cctor()' constructor function sets a local variable to the 'Sharp' key in the registry, referencing the executed infected application. Initially, this points to the 12,288-byte attachment, but later on it will point to the infected file with the host application appended to the virus. Basically this is a confusing way to pass a parameter to Cs.exe.

Next, the .entrypoint method takes over and uses System.Environment::GetFolderPath to determine the path of the Startup folder. Then a short sharp.vbs file is created in Startup folder with a message box which states 'You're infected with W32.HLLP.Sharp, written in C#, by Gigabyte/Metaphase'. The message box, as shown in the


Sharp

You're infected with Win32.HLLP.Sharp, written in C#, by Gigabyte/Metaphase

picture above, will be displayed when the machine is rebooted. Then the virus figures out the path to the Windows and the Program Files folders respectively.

FileSearch

Next, Sharpei calls its own 'FileSearch' function four times in order to look into three subdirectories of the Program Files folder and the Windows directory. Sharpei searches in these folders for files with the '.exe' extension.

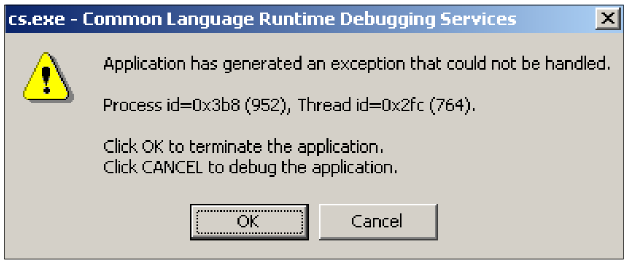
FileSearch is the infection function. First it reads the checksum field of the MZ header (0x12 file offset) to see if the marker 'g' is present. This marker is placed in the first generation sample.

If the marker is found the virus attempts to delete the file 'hostcopy.exe' (which is used during the infection) and searches for the next file to infect.

If the marker is not found the virus clears the attributes of the file and makes a copy of it as 'hostcopy.exe'. It uses the System.IO.File::Copy function with the third parameter set to 1, which means that the target will be overwritten should it exist.

Then the virus uses the variable that was set to the 'Sharp' key to copy the referenced file over the victim file. Finally, 'hostcopy.exe' is appended to the file.

After every infection 'hostcopy.exe' is deleted (it is only a temporary file). I should note that during test infection of the most recent version of the *.NET* Framework we experienced a number of error messages displayed by the Framework, as shown below.


cs.exe - Common Language Runtime Debugging Services

Application has generated an exception that could not be handled.

Process id=0x3b8 (952), Thread id=0x2fc (764).

Click OK to terminate the application.
Click CANCEL to debug the application.

When all four directories have been searched for infected files the virus attempts to check whether there is a host application within the executed application. The virus reads the full content of the host from position 0x3000 (12288) of the virus and moves that content to a new temporary file of its own called 'temp.exe'.

Routine Bugs

This routine suffers from a couple of bugs. First, the virus checks whether the executed file is 'MS02-010.exe' and if this is the case it attempts to delete 'temp.exe'. However, finally it will attempt to execute 'temp.exe' even if it is not there. It uses System.Diagnostics.Process.Start to execute the host temp file.

Whenever a pre-infected application is executed the new target will have a copy of the virus with a previous host appended to it and the new host appended again and so on. Thus each generation will become longer and longer and in many cases Sharpei will not be able to execute the proper host application.

Unfortunately this also means that it is impossible to restore the proper host application with AV repair. This is because the virus pays attention only to the file extension, rather than the file format.

It would be difficult to find the last host in the infected file in all cases. Thus I would recommend deletion for removal of the virus rather than repair. Otherwise the system would contain repaired files with confusing host content.

Dog Tag

Was it wise to name the virus after a dog breed? Well, it seemed a logical name to me. Gigabyte, the author of the virus, wanted to it to be called 'Sharp'. Originally, the Chinese Shar-Pei breeders cultivated the excess folds of skin on the Shar-Pei dog to give the breed an advantage in dogfights. In this virus the C# (C sharp) code was placed under a 'skin' (the 32-bit and VBS layer) to give the virus an advantage.

Conclusion

W32.HLLP.Sharpei@mm demonstrates that it is relatively simple to write a prepender virus in C#.

Administrators need to make themselves familiar with the *.NET* Framework security settings and should not leave the default configurations installed. Since the *.NET* security config files are found in the Windows folder it is important to make sure that the OS level security is used properly as well, otherwise viruses might change the settings one way or another.

Name: W32.HLLP.Sharpei   mm

Alias:	W32.HLLP.Sharp.
Type:	Uses VBS file to mass mail, direct action virus under *.NET* Framework.
Size:	12,288 bytes.
Payload:	Displays a message box on system start.
Repair:	Delete infected files and restore from backup.