MALWARE ANALYSIS

XXX RACTED

Peter Ferrie
Microsoft, USA

We have reached the last in the collection of viruses created by the writer 'fakemnded' in the EOF-rRlf-DoomRiderz virus zine (see also *VB* September 2008, p.4, and October 2008, p.4), although it is not the last entry in the series from the virus zine itself. This one is called W32/Exract.

STUPID IS AS STUPID DOES

The virus begins by getting the address of the IsDebuggerPresent() API and then calling it. If a debugger is present the virus exits. The virus also checks for alterations within the code that has just run. This is probably intended to detect breakpoints, but in normal circumstances, there wouldn't be any breakpoints left at that point. The routine also contains some dead code, which would have called the IsDebuggerPresent() API, and checked once again for alterations. It is fortunate, in a way, that the code doesn't run because, given the way in which the code is structured, the second check would always fail, and the virus code would crash shortly afterwards.

It is possible that the dead code is intended as a decoy, to tempt someone into placing a breakpoint there, which would lead to the virus being able to run freely. However, experience suggests that it is best not to assume something smart where something stupid is more likely.

After some further checks for alterations, including one in a location that has already been checked, we see some familiar code.

I'M A LOCAL

The virus stores the selector of the local descriptor table onto the stack, and then reads four bytes and checks if the result is non-zero. The result should always be non-zero because the location on the stack holds the previous stack frame when the process started, which is always an address above the 64 KB boundary. As a result, the top half of the stack frame will remain untouched and non-zero.

This might be an anti-emulator trick for an emulator that stores four bytes instead of two. However, it seems more likely that what the virus author had in mind was to read only two bytes and detect whether the local descriptor table (LDT) is in use, but had to reverse the condition because of the extra bytes that the virus reads. The use of the LDT is a characteristic of virtual machines such as *VMware* and *VirtualPC*, along with *Norman*'s *SandBox*.

IT'S PAYBACK

The virus carries two payloads. The first triggers on the 12th of any month. At that time, the virus displays a message box whose title is 'Sorry Unable to extract the file!' with the message body:

```
Error 617573 :Shareware period has been elapsed!

For more info search for 'Fakedminded' on google ,and
play warcraft too!
```

The author spelled his name correctly this time. I did as suggested, and searched for 'fakeminded' on *Google*. Funnily enough, none of the returned pages belonged to him.

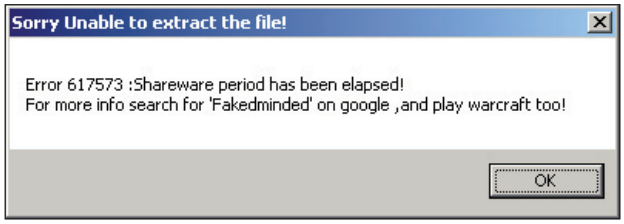
OPEN SESAME

The second payload triggers on the 4th of October. At that time, the virus attempts to drop a file called 'kloka.vbs' into the root directory of the C: drive. This action is disallowed by default under *Windows Vista*.

The virus attempts to create the registry key 'HKEY_CLASSES_ROOT\.sy64', and to set its default value to 'DOS1234'. The virus also attempts to create the registry key 'HKEY_CLASSES_ROOT\DOS1234\shell\open\command' and to set the default value to point to the 'kloka.vbs' file. However, the creation of registry keys in that location is disallowed by default under *Windows Vista*. There is also a bug in the registry code, which appears twice: if the registry value cannot be set, then the virus does not close the registry handle. The result is a handle leak if an error occurs.

The idea of that registry modification is to register a new suffix. Thereafter, executing a file whose suffix is '.sy64' will cause the script file to run. The virus attempts to produce this effect automatically, by creating a file called 'sysvb.sy64' directly in the Start Menu at 'C:\Documents And Settings\All Users\Start Menu\Programs\Startup' and 'C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup'. This action is also disallowed by default under *Windows Vista*.

If the 'kloka.vbs' file were to be executed, it would attempt to access the *Kaspersky Lab* website (www.kaspersky.com) once every ten seconds, in an infinite loop.



BUT WAIT, THERE'S MORE

Now we reach the main part of the virus. The virus opens its own file, requests the file size (which is not constant and might be very large, see below), and then allocates some memory to hold a copy of the entire file. There is a bug in this code, which is that the memory is never freed. Amazingly, the virus is really interested only in the file offset of the original end of the file.

The virus checks if its filename ends with 'bye'. There is a bug here, which is that the comparison is case-sensitive. If the filename does end with 'bye', then the virus attempts to delete the file 'C:\Program Files\Common Files\hushabye.exe'. However, the deletion of the file in that location is disallowed by default under *Windows Vista*.

The virus attempts to create the registry key 'HKEY_CLASSES_ROOT\.err64', and to set its default value to 'Coconest'. The virus also attempts to create the registry key 'HKEY_CLASSES_ROOT\Coconest\shell\open\command' and to set the default value to point to the 'C:\Program Files\Common Files\hushabye.exe' file. As before, the creation of registry keys in that location is disallowed by default under *Windows Vista*. The bugs that result in a possible handle leak (if an error occurs) are also present here.

Thereafter, executing a file whose suffix is '.err64' will cause the exe file to run. The virus attempts to produce this effect automatically, by creating a file called 'sysCheckup.err64' in the Start Menu, in the same location as 'sysvb.sy64'. Once again, this action is disallowed by default under *Windows Vista*. The virus then attempts to copy itself as 'hushabye.exe' to the 'Common Files' directory. This action is also disallowed by default under *Windows Vista*.

CLIP GO THE SHEARS

Once the installation is complete, and if the filename ends with 'bye', then the virus opens the clipboard and saves the 'handle' that is returned. In fact, what is returned is not a handle, but a flag that indicates success or failure. The virus queries the clipboard for a list of files that are currently being copied. If such a list exists, then the virus calculates the length of the clipboard data using a very poorly coded routine. Instead of performing a wcschr() to parse the Unicode characters correctly, the virus performs a byte-level step while comparing words in memory. This can lead to early termination if particular characters are found in the string, such as the Tibetan syllable 'Om'. Meditate on that, grasshopper.

The virus finds the last string in the list, calculates the length in bytes of that string, and then allocates that length. There is a bug in this routine, which is that the buffer is never freed. This can quickly become a problem. Since the code executes in a loop, and if the clipboard is not used for a while, then the same string will be seen repeatedly. This will cause further memory allocations, and eventually exhaust the system resources.

STRING THEORY

The virus copies the string to the newly allocated buffer, and converts it from Unicode to ASCII at the same time. A minor bug exists here, which is that the virus uses the byte count as a character count while copying the string. This results in the virus writing twice as much data as necessary, but is not a problem because the buffer is large enough to hold all of the data.

The virus also allocates a buffer to hold a copy of the string. There is a bug in this routine, which is that the buffer is never freed. The virus copies the string to this buffer, and then examines the copied string. If the string contains only a directory name, then the virus will skip the infection. Otherwise, the virus switches to the directory that contains the file, and then examines the filename. If the filename ends with '-packed.exe', then the virus will also skip the infection. This is the 'infection' marker.

INFECTIOUS GROOVES

The virus opens the file to infect, requests the file size, and then allocates some memory to hold a copy of the entire file. There is a bug in this code, which is that the memory is never freed. The virus reads the whole file, and 'encrypts' it (using just a simple XOR with the letter 'X'). Then the virus opens its own file, requests the file size, and then allocates some memory to hold a copy of the entire file. This is despite the fact that another copy of the virus already exists in memory. There is also a bug in this code, which is that the memory is never freed.

The virus creates new the file '<file>-packed.exe', where '<file>' is the name of the file to infect, and then writes the virus body and the encrypted file to it.

The virus converts the pathname to Unicode, and then allocates memory to hold the Unicode string. There is a bug in this routine, which is that the buffer is not freed. The virus constructs a new 'list' of files, which contains only one entry, and then empties the clipboard of all data before assigning the list. This code could be considered to contain two bugs. The first is that all of the data in the clipboard is discarded, instead of only the data of the file list

type (which can simply be replaced without emptying the clipboard at all). The second bug is that the original file list is discarded, leaving only one file to be copied.

CLOSED FOR THE DAY

The virus calls the CloseHandle() API for the 'handle' that was returned by the OpenClipboard() API. Fortunately for the virus author, the handle is treated as invalid by *Windows* and the request is ignored, rather than causing an error. The virus then closes the clipboard using the correct API.

At this point, either no file list exists, or the infection completed successfully. The virus calls the CloseHandle() API (again), for the 'handle' that was returned by the OpenClipboard() API, and also closes the clipboard (again) using the correct API. The virus sleeps for one second, and then resumes from the top of the function where the clipboard is opened again. Such a short delay is a serious bug. The clipboard is a unique resource, so no other applications can use it while the virus has it open. This produces a race condition for users who are trying to copy items. Sometimes it will work, and most times it won't.

-OOPS

If the filename does not end with 'bye', then the virus searches within the pathname for the '-' character. This is supposed to find the '-packed.exe' files, but it has the buggy behaviour of also finding directories that contain the '-' character. This bug affects the first generation code, such as when it is run from the 'EOF-DR-RRLF' directory. If the wrong file is executed, then the virus will decrypt data beyond the end of the buffer and crash.

However, if the '<file>-packed.exe' file is executed, then the virus will decrypt the appended data, create a new '<file>', and then display a message box stating 'File has been Exracted' [*sic*].

The virus does not run the original file. At this point it runs the installation code, as above, that begins by attempting to delete the 'hushabye.exe'. Finally, once installation is complete, the virus exits.