VIRUS ANALYSIS 1

Looking a Bagift-Horse in the Mouth

Peter Ferrie and Frédéric Perriot
Symantec Security Response, USA

W32/Bagif is a polymorphically encrypted, entry point-obscuring, anti-heuristic, memory resident, parasitic infector of *Windows* Portable Executable files that are not DLLs. It replicates across mapped drives and shared directories on local area networks, and it appears to be based on the code of several existing viruses. In the same way that the author of W95/Bistro had his signature changed in the copy of the virus that was released, it is very likely that the author of W32/Bagif is not the one named in the code.

What Virus is That?

As an anti-heuristic device, files infected with Bagif do not have their entry point altered. Instead, the virus will search for the first call or jump to the ExitProcess() API, and replace the instruction with a transfer of control to near the end of the code section, where the virus will place itself. The technique is very similar to that used by W32/Simile (see *VB*, May 2002, p.4). Additionally, no section attributes are altered, so after infection files look very much as they did beforehand.

When a file infected with Bagif is executed for the first time, and if the virus gains control via execution of the replaced instruction, the virus executes the polymorphic decryptor. The decryptor has characteristics that allow it to be identified immediately as produced by the KME-32 (Kewl Mutation Engine). KME-32 is the engine used by several other viruses, including W95/MTXII, W32/Toal and W95/Zexam. Analysis of Bagif's code allows the engine to be identified as version 5.52, which was released on the first day of 2002.

How Can I Run Thee?

The decryptor uses the floating point unit to perform the decryption, which is an effective attack against the CPU emulators in some anti-virus products. The decryptor places a small (216 bytes) routine on the stack, and then runs this routine.

The routine searches in memory for KERNEL32.DLL and retrieves the API addresses for two functions: GlobalAlloc() and GetModuleHandleA(). The function names are stored as checksums, however the checksum algorithm is the simple checksum used by Delphi applications, among others, rather than the more common CRC32. It is probable that the algorithm was chosen for its smaller size, and considered acceptable despite the increased risk of non-unique checksums. Once the API addresses have been retrieved, the routine allocates memory in which to place the virus body, then decrypts the virus body directly into this memory.

The use of dynamically allocated memory is the method by which an encrypted virus can run from files without altering the section attributes, and the small 'first stage' routine reduces the chance of stack overflow.

Let Me Count the Ways

Once the virus body gains control, it checks whether another copy of it is already running. If no other copies are running, the virus decompresses and creates a file called 'backup.gif', in the directory used for storing temporary files. The compressor that is used is aPLib, a favourite among virus writers.

After the file is written, the virus will infect the file, execute it, then exit, leaving the dropped file as the one that remains running in memory. Whenever the dropped file is executed, it will copy itself to the Windows\System directory as 'ntloader.exe', and to the current user's Startup directory (if it can be found by querying HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Startup in the registry) as 'win32s.exe'.

After copying itself, the virus will install itself as the application that handles requests to run .exe files, by changing the exefile Open key in the Registry (HKCR\exefile\shell\open\command). If the dropped file was executed as a result of the change to the exefile Open key, there is a 50 per cent chance that the virus will infect the file being executed.

We Interrupt this Program

When the dropped file is executed for the first time it will register itself as a service process, if the undocumented RegisterServiceProcess() API is available (it exists in *Windows 9x/Me*), then create a thread that spreads the virus. The first part of the thread enumerates network disk resources; the second part of the thread enumerates drive letters. Intriguingly, the virus body contains the names of some APIs that can be used for email and/or backdoor effects, but there is no reference to these APIs in the virus code. Perhaps the virus author had not completed the routines before the virus was released.

Share and Enjoy

The spreading of this virus across the network is achieved using a method that is very similar to that used by W32/Magistr. Bagif begins by changing to the directory

that it has found on the network, then attempting to create a file, using a random name and extension, to determine whether the directory is writeable. If the file can be created, then the virus will guess at the name of the *Windows* directory and try to change to that directory. If the change is successful, the virus will copy itself as 'tsoc32.exe' and alter the WIN.INI in that directory to run the copied file whenever *Windows* is started. The virus uses the WritePrivateProfileString() API to do this, because that API allows the path of the .INI file to be specified.

Randumb

The random number generator that is used is very similar to the one in *Microsoft Visual C++*. It has a short period, but it is also small. Random number generators are very common in viruses, and range from the very simple (calling the GetTickCount() API repeatedly) to the very complex (the Mersenne Twister in W32/Chiton). The algorithm that is chosen is often a compromise between randomness, period length, and code size.

The spreading across drive letters is done backwards, from Z until the drive letter that contains the Windows\System directory. For each local and mapped network drive, the virus will check whether the directory is writeable, as described above. If the directory is writeable, the virus will scan recursively into directories, but only to a depth of four subdirectories. For each file that is found, there is a 50 per cent chance that it will be skipped explicitly. This behaviour is identical to that of W32/Simile.

The virus looks for files with extensions EXE or SCR, but whose name does not begin with 'EXPL' or 'UNRE', or whose name is 'HL'. These names would match files such as Explorer, and the game files for *Unreal* and *Half-Life*, all of which are self-checking.

Are You My Type?

Additionally, the file size must be between 4kb and 2Mb, and the file must be executable and not a DLL. The check for the CPU type has its origins in misinformation about the allowed values. The standard allows only for a value of 0x14C in the field, corresponding to Intel 386+ CPUs, but documents exist suggesting that the values 0x14D, 0x14E, and 0x14F, exist, corresponding to Intel 486, 586, and 686, CPUs. In fact, no value for Intel x86 CPUs, other than 0x14C, is supported by *Windows*.

The virus attempts to match the first few characters of every section name against a list of 15 names that the virus carries. Files are avoided if they contain unrecognised section names. After these checks have been made, the virus looks for its infection marker.

Files are considered infected if the difference between the third byte of the COFF Date/Time stamp field and the XOR of the low two bytes is less than eight. Files must also import the ExitProcess() API from KERNEL32.DLL, since the virus requires this function for its EPO implementation, however only the first eight characters are matched in the import name and the dll name, so an API called 'ExitProcrastinator()', for example, in a file called 'KERNEL32R0X', would be accepted too.

If the file imports ExitProcess(), the attributes of the first section are checked, unless the file to be infected is the dropped file. The checking of the first section appears to be a bug, since the number of sections is retrieved, but never used. It is likely that the last section was the one that was intended to be checked, which also matches the behaviour of W32/Simile. The contents of the first section are checked for 'virus-like' strings (a check for 'MZ' followed within 128 bytes by 'PE'). Files are avoided if they contain these strings.

I'm Attached to You

If a file passes all of these checks, the virus will allocate a buffer in which to place itself and the decryptor. The virus begins by filling the buffer with between 255 and 512 bytes of random values, then the virus body is encrypted and placed immediately after these values. The encryption is weak and the original key can be restored very easily. At this point, the virus will decompress and run the KME-32 code, to create a new decryptor for the 'first-stage' routine.

Once the decryptor has been created, the virus will append the buffer to the first section in the file, then update the physical and virtual locations of all of the following sections, as well as fixing the imports, resources, exports, and relocations.

While parsing the relocation table, the virus removes the relocation pointing to the instruction that was altered. This is necessary because the new instruction does not require relocation. If the image base is the *Windows* default value (0x400000), and the relocation section is the last one in the file, then there is a 50 per cent chance that the virus will remove the relocation section completely. If a checksum existed before, then the virus will calculate a new one.

Conclusion

W32/Bagif is an interesting sum of other viruses' parts, with some neat optimisations, but we have seen it all before.

W32/Bagif	
Type:	Polymorphic, EPO, memory-resident, parasitic.
Infects:	PE .EXE and .SCR files.
Self-recognition:	Magic value in PE header of files.
Removal:	Delete infected files and restore from backups.