ANALYSIS 2

CHAMBER OF HORRORS

Peter Ferrie
Symantec Security Response, USA

Amongst the glut of viruses that we see every day, sometimes there is one to surprise us. W32/Chamb is one of those: the first virus to infect compiled HTML (CHM) files parasitically.

WHAT A CHAMPION

Compiled HTML files are *Microsoft*'s way of packaging entire web pages – HTML pages, pictures, sounds, etc. – into a single file that can be transported and viewed offline. The environment for displaying the pages is replicated exactly, since they are passed to the browser by the viewing application. The problem is that the files (properly called 'streams' in this context) in the package are not written to disk prior to being rendered, so anti-malware software is out of luck if it does not support the CHM file format. At this point, it should be noted that the file format is both complex and undocumented, but we have reverse-engineered it. Let's have a look inside.

Compiled HTML files begin with the signature 'ITSF'. That signature stands for 'InfoTech Storage File', which is *Microsoft*'s name for the library that is used to read and write CHM files. Interestingly, when the name is shortened to 'IStorage', we get the name of the programming interface that is used to manage such files. More interestingly, the IStorage interface is the same as the one used by OLE2 files, and which dates back to 1992. The only difference between the OLE2 and CHM implementation is the introduction of the InfoTech Storage System (ITSS) DLL that handles the transparent compression and decompression of the data inside CHM files.

IT'SS LIIKE THISS

Apart from the signature, the ITSF header contains nothing of particular interest. Immediately following it are two directories, divided into two quadwords each. The first quadword in each directory contains the file offset of the data in that block; the second quadword in each directory contains the length of the data in that block.

The first directory block contains the file size, and a flag that is set when a CHM file is first created. The purpose of the flag is to indicate that the file is either a 'work in progress' (when set), or has been finalised (when clear) and no other modifications are allowed.

The second directory block begins with the signature 'ITSP'. It contains information about the number and size of the file list blocks, and the location of the indexes used to access the data quickly in the file list blocks.

The file list blocks follow immediately. They begin with the signature 'PMGL'. The PMGL blocks contain the list of stream names for the streams in the CHM file. There are two types of stream in CHM files: system-data streams and user-data streams. The system-data streams are recognisable because their names begin with two colon characters '::'. The user-data streams are recognisable because their names begin with the forward slash character '/'. The reason for the forward slash character is because these are pathnames. These pathnames are relative to the root directory, which in this case is contained within the CHM file. The stream names are stored in alphabetical order to allow for easy indexing. However, index blocks (which begin with the signature 'PMGI') are added only when there are multiple PMGL blocks.

There are two types of user-data stream: internal and external. The internal user-data streams are recognisable because their names begin with either a hash character '#' or a dollar sign '$'. Anything else is assumed to be an external user-data stream.

Additionally, each PMGL block contains the identity of the previous and next PMGL block, which means that the PMGL blocks can be reordered in peculiar ways, though this would need to be done manually.

CHAMPING AT THE BIT

Each stream name is followed by the dataspace index, the offset of the data relative to the start of the dataspace, and the size of the data. These values are encoded using a seven-bit continuation method: the eighth bit in each byte is used to specify that the value spans multiple bytes. The other seven bits form seven bits of the value, in big-endian format.

The location of the dataspace is found by searching within the stream names for the system-data stream called '::DataSpace/NameList'. After decoding the offset of the NameList, we reach a list of names in zero-terminated Unicode Pascal format (which seems extreme – either zero-terminated or Pascal format alone is sufficient to determine the length of the strings). Only two names should appear in the list: Uncompressed and MSCompressed.

The data in the 'Uncompressed' stream are simply stored. The data in the 'MSCompressed' stream are compressed with *Microsoft*'s LZX compression method, which is also one of the compression methods supported by the CAB file format. However, unlike in CAB format where each file is compressed individually, CHM files compress all of the streams as though they were a single block (a so-called 'solid' archive). While this can increase the compression ratio significantly, it can also increase the time required to extract individual items significantly. *Microsoft* compromised between these two characteristics, by breaking the single large block into smaller blocks of fixed size and compressing those individually. The information about these smaller blocks is stored in a 'reset table' (see below).

In order to decompress the data in the 'MSCompressed' stream, some additional streams must be retrieved first. One of those is the '::DataSpace/Storage/MSCompressed/ControlData' stream, which contains the information about the LZX compression parameters. The other two streams are '::DataSpace/Storage/MSCompressed/Transform/{7FC28940-9D31-11D0-9B27-00A0C91E9C7C}/InstanceData/ResetTable' and '::DataSpace/Storage/MSCompressed/Content'. The 'ResetTable' stream is used to control the periodical resetting of the decompression state. By resetting the decompression state periodically, it no longer becomes necessary to decompress the entire large block to reach an arbitrary file. The reset table allows one to begin the decompression at the nearest reset state prior to the required offset, which can make the decompression faster for some items. Finally, the 'Content' stream contains the compressed data.

As an aside, there is an interesting extension in the '::DataSpace/Storage/MSCompressed/Transform/List' stream. It appears that it was intended to provide support for customised decompression and/or decoding layers, but the stream data in existing CHM files are malformed – the stream contains only a partial GUID in Unicode character form, because the stream is too small to contain a complete GUID. Judging by the stream length, it was probably intended to hold an ASCII string and some small additional data.

CHARM OFFENSIVE

So what does all of this have to do with W32/Chamb? Actually, very little – since the virus makes use of the IStorage interface, all of these details are handled by the ITSS DLL, and all the virus has to do is call a few functions to perform the required actions, much as any other file infecting does for an ordinary file system.

In any case, the virus begins by searching the current directory for CHM files to infect. The infection marker is that the file has the read-only attribute set. Otherwise, the file is considered a candidate for infection.

If the virus finds a file to infect, it creates a new file called 'c' in the current directory, which is used as a temporary working file during the infection process. The temporary file is required because the ITSS DLL does not allow writing to a 'finalised' CHM file.

The virus enumerates all of the storages and streams in the file to infect, and writes each of them to the temporary file. Anything within the original file that is neither a storage nor a stream will be discarded during the infection process. The ITSS DLL decompresses the streams automatically as they are read, and compresses them as they are written.

For any stream whose name ends with '.HTM', the virus will append an object reference to a stream called '.exe'. Upon completion of the enumeration, the virus will add itself as the stream called '.exe', thus ensuring that it will be called whenever a page is viewed in the infected CHM file.

The ITSS DLL sorts the storage and stream names as they are added. The result is that even though the '.exe' stream is the last to be added, thanks to its name, it will be among the first in the PMGL blocks.

After adding the '.exe' stream, the virus will copy the 'c' file over the original file, set the file date and time stamps to those of the original file, and set the read-only attribute to mark the file as infected.

Upon completion of the file enumeration, the virus simply exits. The virus contains no payload, it is simply yet another proof of concept from a virus author who specialises in producing them.

THE CHASM OPENS WIDE

Compiled HTML files have been a favourite of malware authors for several years already, but until now only in static form. For the most part, they have been trojans that downloaded other malware, but at least one family of worms (W32.Blebla) used a CHM file in order to spread. Now that we have a parasitic virus for CHM files, the advice is the same as when the first WinHelp infectors appeared in 1999: don't press F1!

W32/Chamb

Type:	Parasitic direct-action infector.
Infects:	*Windows* CHM files.
Self-recognition:	Read-only attribute is set.
Payload:	None.
Removal:	Delete infected files and restore them from backups.