Abysssec Research

1) Advisory information

Title	**: Movie Maker Remote Code Execution (MS10-016)**
Version	**: moviemk.exe 2.1 (XP SP3)**
Analysis	**: http://www.abysssec.com**
Vendor	**: http://www.microsoft.com**
Impact	**: Ciritical**
Contact	**: shahin [at] abysssec.com , info [at] abysssec.com**
Twitter	**: @abysssec**
CVE	**: CVE-2010-0265**

2) Vulnerable version

Windows XP SP2,SP3
Windows Movie Maker 2.1

Windows Vista SP1,SP2 and x64 versions
Windows Movie Maker 2.6
Windows Movie Maker 6.0
Windows Movie Maker 6.1

3) Vulnerability information

Class

1- Heap overflow

Impact

Successfully exploiting this issue allows remote attackers to cause denial-of-service conditions or execute arbitrary code.

Remotely Exploitable

Yes

Locally Exploitable

Yes

4) Vulnerabilities detail

The vulnerable part starts at IsValidWMToolsStream function. In this function new is used to times for allocating space. In both cases, values of Size needed for allocating memory is read from .mswmm file.

In first case after calling 'new' function, ExtractData function of CDocManager is read to fill the content of allocated space. Content of this space is read from .mswmm file.

ExtractData function of CdocMnager class takes three arguments. First argument specifies a string that is to be called. Second argument is a pointer to a space that known string from the first arguments is copied to it. And third argument specifies length of data that should be read.

```
.text:011814C4          push    [ebp+bstrString] ; unsigned int
.text:011814C7          call    ??2@YAPAXI@Z    ; operator new(uint)
.text:011814CC          mov     ebx, eax
.text:011814CE          pop     ecx
.text:011814CF          mov     [ebp+psz], ebx
.text:011814D2          mov     [ebp+var_3C], ebx
.text:011814D5          mov     byte ptr [ebp+var_4], 1
.text:011814D9          push    [ebp+bstrString]
.text:011814DC          mov     ecx, esi
.text:011814DE          push    ebx
.text:011814DF          push    [ebp+var_30]
.text:011814E2          call    ?ExtractData@CDocManager@@QAEJPBGPAXJ@Z ;
CDocManager::ExtractData(ushort const *,void *,long)
.text:011814E7          mov     edi, eax
.text:011814E9          test    edi, edi
.text:011814EB          jge     short loc_1181503
```

Then in next steps 'new' function is used for the second time for allocating new space which then ExtractData function is called for copying "WmtoolsValid"value to the allocated space.

```
.text:01181540          push    [ebp+bstrString] ; unsigned int
.text:01181543          call    ??2@YAPAXI@Z    ; operator new(uint)
.text:01181548          pop     ecx
.text:01181549          mov     [ebp+var_14], eax ;  ebp-14h = pBuffer
.text:0118154C          mov     [ebp+var_40], eax
.text:0118154F          mov     byte ptr [ebp+var_4], 2
.text:01181553          push    [ebp+bstrString]
.text:01181556          mov     ecx, esi
.text:01181558          push    ebx
.text:01181559          push    edi
.text:0118155A          call    ?ExtractData@CDocManager@@QAEJPBGPAXJ@Z ; ExtractData(ushort const
*,void *,long)
.text:0118155F          mov     esi, eax
.text:01181561          test    esi, esi
.text:01181563          jge     short loc_118158A
```

If look at the above code carefully, you will notice second parameter of ExtractData function, is the pointer to the space that allocated by first call to 'new' function and here is the vulnerable point. Because the allocated space to

the second call is not used and instead the same space which is allocated by first call is used. As there is no bound checking on length of data that should be read from the file (third argument of ExtractData function), it is possible to copy longer data than size of the allocated space of first 'new' call by manipulating this value in file which cause an overflow.

This vulnerability can show itself in the first call after CDocManager::CSmartStream::Read function within the ExtractData function.

```
.text:01180C07          mov     esi, [ebp+arg_8]
.text:01180C0A          mov     ecx, [eax]
.text:01180C0C          push    edi
.text:01180C0D          lea     edx, [ebp+var_14]
.text:01180C10          push    edx
.text:01180C11          push    esi
.text:01180C12          push    [ebp+arg_4]
.text:01180C15          push    eax
.text:01180C16          call    dword ptr [ecx+0Ch]     ; CDocManager::CSmartStream::Read
.text:01180C19          mov     edi, eax
.text:01180C1B          test    edi, edi
.text:01180C1D          jge     short loc_1180C34
.text:01180C1F          or      [ebp+var_4], 0FFFFFFFFh
.text:01180C23          mov     eax, [ebp+var_10]
.text:01180C26          test    eax, eax
.text:01180C28          jz      short loc_1180C30
.text:01180C2A          mov     ecx, [eax]
.text:01180C2C          push    eax
.text:01180C2D          call    dword ptr [ecx+8]     ; Crash Point
```

Exploit:

Note: Our vulnerability is a Heap Overflow class and Movie Maker 2.1 software doesn't have DEP protection. As we mentioned earlier, this vulnerability can show his face in the first call (call [ecx+8]) after CDocManager::CSmartStream::Read function within ExtractData function. because this call instruction, calls one of the inputs of vtable and because our overflowed data can overwrite values of this vtable, so the program crashes.

By these backgrounds in mind, the only thing to execute shellcode is to manipulate values of vtable properly to execute our arbitrary address by calling [ecx+8].