Abysssec Research

1) Advisory information

Title	**: Microsoft Office Excel 2002 memory corruption vulnerability (0day)**
Version	**: Office Excel 2002(office xp)**
Discovery	**: http://www.abysssec.com**
Vendor	**: http://www.microsoft.com**
Impact	**: High**
Contact	**: shahin [at] abysssec.com , info [at] abysssec.com**
Twitter	**: @abysssec**
CVE	**: ZERO DAY**

2) Vulnerable version

3) Vulnerability information

Class

1- memory corruption (Integer issue)

Impact

Attackers can exploit this issue by enticing an unsuspecting user to open a specially crafted Excel ('.xls') file. Successful exploits can allow attackers to execute arbitrary code with the privileges of the user running the application.

Remotely Exploitable

Yes

Locally Exploitable

Yes

4) Vulnerabilities detail

HFPicture record consists of an integrated encryption of a picture contents that may be a MSODRAWING or MSODRAWINGGROUP record format. The fields of this record consist of the followings:

Offset	Name	Size	Contents
4	rt	2	Record type; this matches the BIFF rt in the first two bytes of the record; =0866h
6	grbitFrt	2	FRT flags; must be zero
8	(unused)	8	Must be zero
16	rgf	1	Bit flags, see description below.
15	rgb	var	An embedded encoding of the contents of the picture; May be in MSODRAWING or MSODRAWINGGROUP record format as indicated in rgf flags listed below.

The sub_305933A8 function is responsible for processing this record. rgb field is used for encryption. One of the functions called in the process of rgb is sub_30E2C12E from mso.dll module:

In a part of the function 4bytes of the rgb field is read and passed to the Ordinal578 (30B1C646) function:

```
.text:30E2AF61            mov    eax, [ebp+var_14]
.text:30E2AF64            cmp    eax, 1
.text:30E2AF67            jbe    loc_30F089A7
.text:30E2AF6D
.text:30E2AF6D loc_30E2AF6D:                  ; CODE XREF: sub_30E2C12E+DC87Cj
.text:30E2AF6D            push   0FFFFFFFFh
.text:30E2AF6F            push   eax
```

```
.text:30E2AF70          lea     eax, [edi+0F0h]
.text:30E2AF76          push    eax
.text:30E2AF77          call    Ordinal578
```

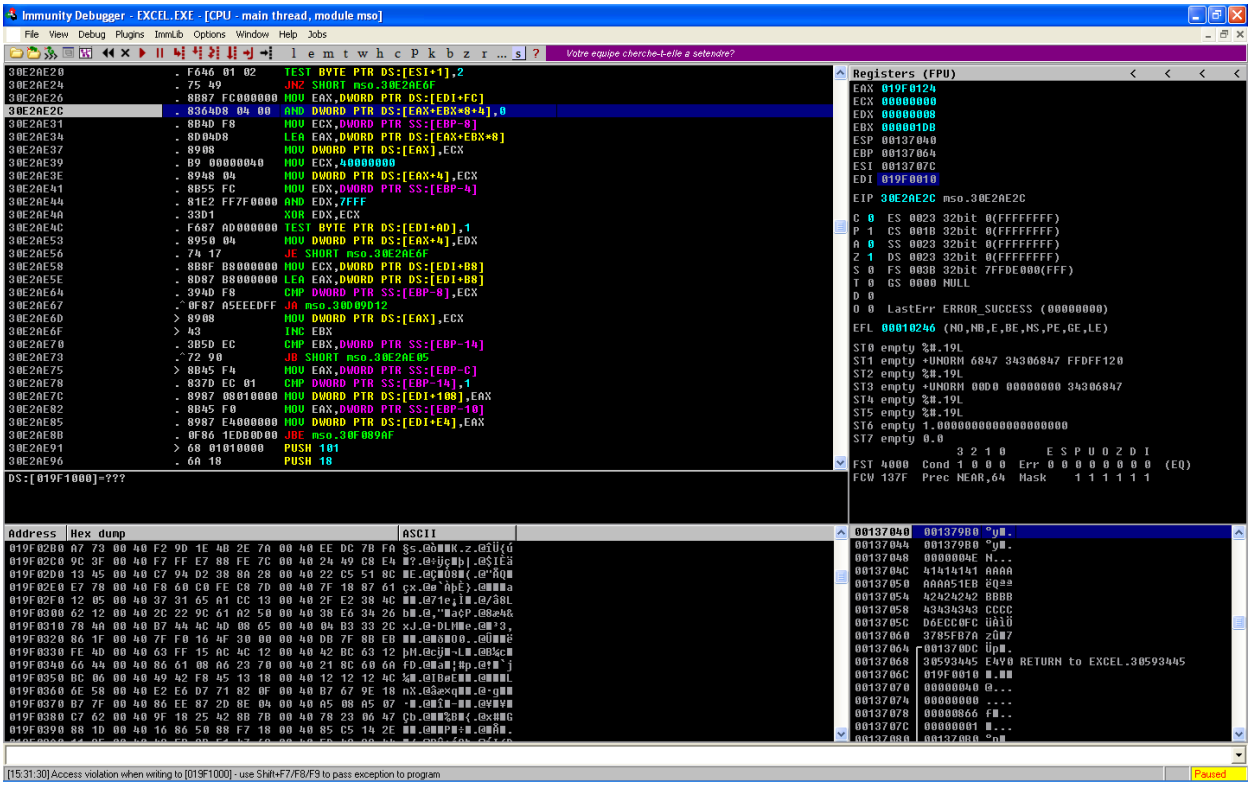
The flaw exists in the sub_30B1C646 function because it doesn't properly check it's argument and the argument is under our control. This vulnerability occurs because it considers the values as unsigned and compares it with this consideration although we have a signed number.

```
.text:30B1C651          mov     esi, [ebp+arg_4]
.text:30B1C654          movzx   eax, word ptr [ebx+2]
.text:30B1C658          cmp     eax, esi
.text:30B1C65A          push    edi
.text:30B1C65B          jl      loc_30B2468E
.text:30B1C661          cmp     [ebp+arg_8], 0
.text:30B1C665          jge     loc_30D3DAA2
.text:30B1C66B          movzx   edi, word ptr [ebx]
.text:30B1C66E          cmp     esi, edi
.text:30B1C670          jle     short loc_30B1C698

...

.text:30B1C698          push    1
.text:30B1C69A          mov     [ebx], si
.text:30B1C69D          pop     eax
.text:30B1C69E
.text:30B1C69E loc_30B1C69E:                    ; CODE XREF: Ordinal578+3BCC40j
.text:30B1C69E          pop     edi
.text:30B1C69F          pop     esi
.text:30B1C6A0          pop     ebx
.text:30B1C6A1          leave
.text:30B1C6A2          retn    0Ch
```

Next if this functions return 1 we enter to a loop that read bytes of excel values and copy them to a block of heap. But the point here is that the value of the loop is initialized by our negative number (or positive large number).

```
.text:30E2AE05          mov     ecx, [esi+2Ch]
.text:30E2AE08          push    8
.text:30E2AE0A          lea     edx, [ebp+var_8]
.text:30E2AE0D          call    sub_30E2CA0C
.text:30E2AE12          test    eax, eax
.text:30E2AE14          jz      loc_30F08A1E
.text:30E2AE1A          mov     eax, [esi+30h]
.text:30E2AE1D          add     dword ptr [eax], 8
.text:30E2AE20          test    byte ptr [esi+1], 2
.text:30E2AE24          jnz     short loc_30E2AE6F
.text:30E2AE26          mov     eax, [edi+0FCh]
.text:30E2AE2C          and     dword ptr [eax+ebx*8+4], 0    →    crash
.text:30E2AE31          mov     ecx, [ebp+var_8]
.text:30E2AE34          lea     eax, [eax+ebx*8]
.text:30E2AE37          mov     [eax], ecx
.text:30E2AE39          mov     ecx, 40000000h
.text:30E2AE3E          mov     [eax+4], ecx
.text:30E2AE41          mov     edx, [ebp+var_4]
.text:30E2AE44          and     edx, 7FFFh
.text:30E2AE4A          xor     edx, ecx
```

```
.text:30E2AE4C              test    byte ptr [edi+0ADh], 1
.text:30E2AE53              mov     [eax+4], edx
.text:30E2AE56              jz      short loc_30E2AE6F
.text:30E2AE58              mov     ecx, [edi+0B8h]
.text:30E2AE5E              lea     eax, [edi+0B8h]
.text:30E2AE64              cmp     [ebp+var_8], ecx
.text:30E2AE67              ja      loc_30D09D12
.text:30E2AE6D
.text:30E2AE6D loc_30E2AE6D:                        ; CODE XREF: sub_30E2C12E-122419j
.text:30E2AE6D              mov     [eax], ecx
.text:30E2AE6F
.text:30E2AE6F loc_30E2AE6F:                        ; CODE XREF: sub_30E2C12E-130Aj
.text:30E2AE6F                                      ; sub_30E2C12E-12D8j
.text:30E2AE6F              inc     ebx
.text:30E2AE70              cmp     ebx, [ebp+var_14]
.text:30E2AE73              jb      short loc_30E2AE05
```



To crash the program skip 38bytes from the beginning of the record then initialize 4byte. And it will crash based on your 4bytes value. To find the beginning of the record in the poc file search '66 08 4E 00' value in the hex editor. (866 is the identity for HFPicture record)

Hint: The value you overwrite should be a negative (large positive) number.