MOAUB

Abysssec Research

1) Advisory information

Title	: Microsoft Excel OBJ Record Stack Overflow
Version	: Excell 2002 sp3
Discovery	: http://www.abysssec.com
Vendor	: http://www.microsoft.com
Impact	: Critical
Contact	: shahin [at] abysssec.com , info [at] abysssec.com
Twitter	: @abysssec
CVE	: CVE-2010-0822

2) Vulnerable version

Microsoft Open XML File Format Converter for Mac 0
Microsoft Office 2008 for Mac 0
Microsoft Office 2004 for Mac 0
Microsoft Excel 2002 SP3
+ Microsoft Office XP SP3
Microsoft Excel 2002 SP2
+ Microsoft Office XP SP2
- Microsoft Windows 2000 Professional SP3
- Microsoft Windows 2000 Professional SP2
- Microsoft Windows 2000 Professional SP1
- Microsoft Windows 2000 Professional
- Microsoft Windows 98
- Microsoft Windows 98SE
- Microsoft Windows ME
- Microsoft Windows NT Workstation 4.0 SP6a
- Microsoft Windows NT Workstation 4.0 SP6
- Microsoft Windows NT Workstation 4.0 SP5
- Microsoft Windows NT Workstation 4.0 SP4
- Microsoft Windows NT Workstation 4.0 SP3

- Microsoft Windows NT Workstation 4.0 SP2
- Microsoft Windows NT Workstation 4.0 SP1
- Microsoft Windows NT Workstation 4.0
- Microsoft Windows XP Home SP1
- Microsoft Windows XP Home
- Microsoft Windows XP Professional SP1
- Microsoft Windows XP Professional
Microsoft Excel 2002 SP1
+ Microsoft Office XP SP1
- Microsoft Windows 2000 Advanced Server SP2
- Microsoft Windows 2000 Advanced Server SP1
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Datacenter Server SP2
- Microsoft Windows 2000 Datacenter Server SP1
- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Professional SP2
- Microsoft Windows 2000 Professional SP1
- Microsoft Windows 2000 Professional
- Microsoft Windows 2000 Server SP2
- Microsoft Windows 2000 Server SP1
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Terminal Services SP2
- Microsoft Windows 2000 Terminal Services SP1
- Microsoft Windows 2000 Terminal Services
- Microsoft Windows 98
- Microsoft Windows 98SE
- Microsoft Windows ME
- Microsoft Windows NT Enterprise Server 4.0 SP6a
- Microsoft Windows NT Enterprise Server 4.0 SP6
- Microsoft Windows NT Enterprise Server 4.0 SP5
- Microsoft Windows NT Enterprise Server 4.0 SP4
- Microsoft Windows NT Enterprise Server 4.0 SP3
- Microsoft Windows NT Enterprise Server 4.0 SP2
- Microsoft Windows NT Enterprise Server 4.0 SP1
- Microsoft Windows NT Enterprise Server 4.0
- Microsoft Windows NT Server 4.0 SP6a
- Microsoft Windows NT Server 4.0 SP6
- Microsoft Windows NT Server 4.0 SP5
- Microsoft Windows NT Server 4.0 SP4
- Microsoft Windows NT Server 4.0 SP3
- Microsoft Windows NT Server 4.0 SP2
- Microsoft Windows NT Server 4.0 SP1
- Microsoft Windows NT Server 4.0
- Microsoft Windows NT Terminal Server 4.0 SP6
- Microsoft Windows NT Terminal Server 4.0 SP5
- Microsoft Windows NT Terminal Server 4.0 SP4
- Microsoft Windows NT Terminal Server 4.0 SP3
- Microsoft Windows NT Terminal Server 4.0 SP2

- Microsoft Windows NT Terminal Server 4.0 SP1
- Microsoft Windows NT Terminal Server 4.0
- Microsoft Windows NT Workstation 4.0 SP6a
- Microsoft Windows NT Workstation 4.0 SP6
- Microsoft Windows NT Workstation 4.0 SP5
- Microsoft Windows NT Workstation 4.0 SP4
- Microsoft Windows NT Workstation 4.0 SP3
- Microsoft Windows NT Workstation 4.0 SP2
- Microsoft Windows NT Workstation 4.0 SP1
- Microsoft Windows NT Workstation 4.0
- Microsoft Windows XP Home
- Microsoft Windows XP Professional

Microsoft Excel 2002
+ Microsoft Office XP
- Microsoft Windows 2000 Professional SP2
- Microsoft Windows 2000 Professional SP1
- Microsoft Windows 2000 Professional
- Microsoft Windows 95 SR2
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows 98SE
- Microsoft Windows ME
- Microsoft Windows NT 4.0 SP6a
- Microsoft Windows NT 4.0 SP5
- Microsoft Windows NT 4.0 SP4
- Microsoft Windows NT 4.0 SP3
- Microsoft Windows NT 4.0 SP2
- Microsoft Windows NT 4.0 SP1
- Microsoft Windows NT 4.0

Avaya Messaging Application Server MM 3.1
Avaya Messaging Application Server MM 3.0
Avaya Messaging Application Server MM 2.0
Avaya Messaging Application Server MM 1.1
Avaya Messaging Application Server 5
Avaya Messaging Application Server 4
Avaya Messaging Application Server 0
Avaya Meeting Exchange - Webportal 0
Avaya Meeting Exchange - Web Conferencing Server 0
Avaya Meeting Exchange - Streaming Server 0
Avaya Meeting Exchange - Recording Server 0
Avaya Meeting Exchange - Client Registration Server 0

3) Vulnerability information

Class

1- Code execution

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

The OBJ record is equal to graphic objects and control objects like Line, Rectangular, CheckBox control and … in excel. OBJ record has various types that type of each record is distinguished by subrecords of the OBJ record. Structure of the subrecord is the same as record structure in the BIFF files. It means first 2bytes is the identity for subrecord. And next 2byes specify the length and bytes after that are data. Various subrecord are:

Subrecord	**Number**	**Description**
ftEnd	00h	End of OBJ record
(Reserved)	01h	
(Reserved)	02h	
(Reserved)	03h	
ftMacro	04h	Fmla-style macro
ftButton	05h	Command button
ftGmo	06h	Group marker
ftCf	07h	Clipboard format
ftPioGrbit	08h	Picture option flags
ftPictFmla	09h	Picture fmla-style macro
ftCbls	0Ah	Check box link
ftRbo	0Bh	Radio button
ftSbs	0Ch	Scroll bar
ftNts	0Dh	Note structure
ftSbsFmla	0Eh	Scroll bar fmla-style macro
ftGboData	0Fh	Group box data
ftEdoData	10h	Edit control data
ftRboData	11h	Radio button data
ftCblsData	12h	Check box data

ftLbsData	13h	List box data
ftCblsFmla	14h	Check box link fmla-style macro
ftCmo	15h	Common object data

Always the first subrecord is ftCmo and the last one is ftEnd. Here are the fields of ftCmo:

Offset	Field Name	Size	Contents
0	ft	2	=ftCmo (15h)
2	cb	2	Length of ftCmo data
4	ot	2	Object type (see following table)
6	id	2	Object ID number
8	grbit	2	Option flags (see following table)
14	(Reserved)	12	Reserved; must be 0 (zero)

sub_30164E23 function is responsible for processing this record. the vulnerability we are going to show you is not exists in this function. This function store values related to subrecord into the buffer. In the next functions subrecord section is processed. One of the functions that process values subrecord fields is sub_3012FABC. This function process ftCmo fields:

```
.text:3012FAC8          mov    edi, [ebp+arg_0]
.text:3012FACB          xor    esi, esi
.text:3012FACD          cmp    dword_307E1FB4, esi
.text:3012FAD3             mov    ebx, [edi+6]
.text:3012FAD6          mov    [ebp+var_4], esi
.text:3012FAD9          mov    [ebp+var_4C], esi
.text:3012FADC          mov    [ebp+var_48], esi
.text:3012FADF          mov    [ebp+var_44], esi
.text:3012FAE2          mov    [ebp+var_40], esi
.text:3012FAE5          ja     loc_30274818
.text:3012FAEB          cmp    dword_307DB7A4, esi
.text:3012FAF1          jnz    short loc_3012FAFB
.text:3012FAF3          cmp    ebx, esi
.text:3012FAF5             jnz    loc_30127293
  ...
.text:30127293             push   dword ptr [ebx+4]
.text:30127296             call   sub_30127263
```

First line pointer to some buffer containing the content of ftCmo subrecord is copied to the edi register. Then in next steps, sixth offset from this buffer is copied to ebx register. If you pay attention to the ftCmo structure, you will notice that from this offset 12bytes is reserved. So the result which copied to the ebx is the first 4bytes of this reserved value.

Now if you follow the code you notice that we can have a jump to address 30127293 which in this address value of ebx register (can be initialize by us) plus 4 is passed as an argument to the sub_30127263 function. in fact this point is the vulnerable point because of no check on this field.

In sub_30127263 function only argument (which we have specified) is added to 10 and is passed to the MSO_804 function.

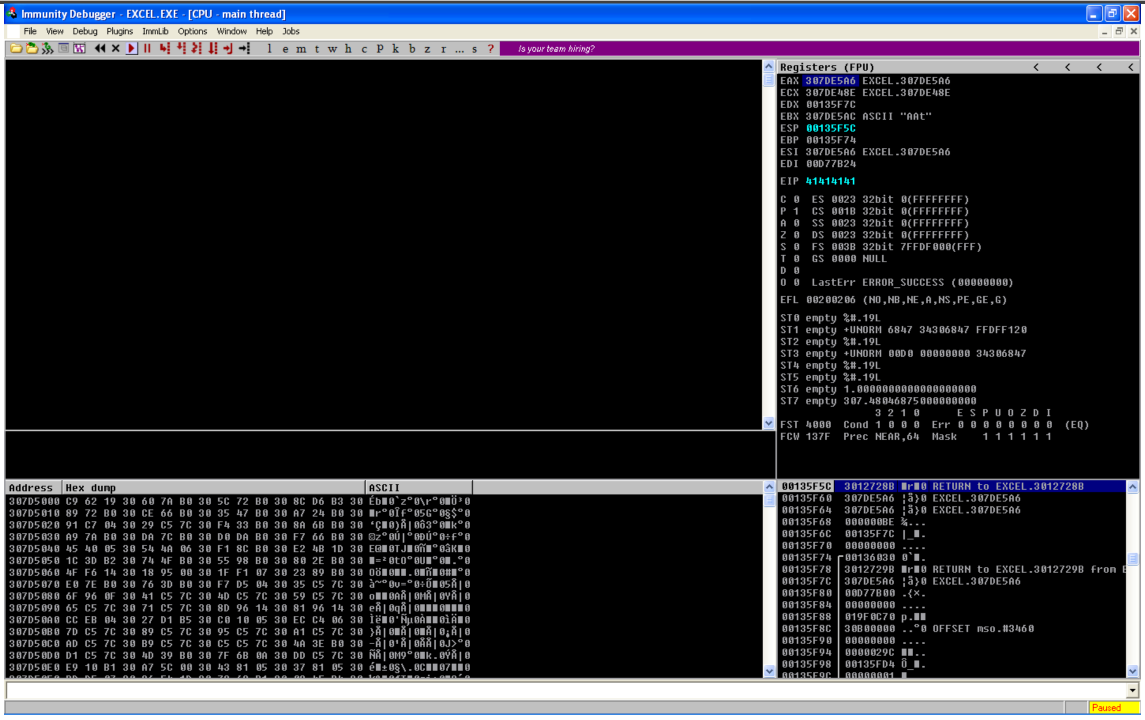
```
.text:30127263          push    ebp
.text:30127264          mov     ebp, esp
.text:30127266          mov     eax, [ebp+arg_0]
.text:30127269          push    esi
.text:3012726A          mov     esi, [eax+0Ah]
.text:3012726D          push    esi
.text:3012726E          call    MSO_804 [307D538C]
...
```

The only task that is performed in MSO_804 function is incrementing its argument by 60 and return its contents.

```
30E27FB0 #804     PUSH EBP
30E27FB1          MOV EBP,ESP
30E27FB3          MOV EAX,DWORD PTR SS:[EBP+8]
30E27FB6          TEST EAX,EAX
30E27FB8          JE mso.30C7A572
30E27FBE          MOV EAX,DWORD PTR DS:[EAX+3C]
30E27FC1          POP EBP
30E27FC2          RETN 4
```

Back to the sub_30127263 function , after executing the MSO_804 contents of return value of this function (under our control) is stored in ecx register and a little bit more content of some offset from this register is called.

```
...
.text:30127274          test    eax, eax
.text:30127276          jz      short loc_3012728E
.text:30127278          mov     ecx, [eax]
.text:3012727A          lea     edx, [ebp+arg_0]
.text:3012727D          push    edx
.text:3012727E          push    0BEh
.text:30127283          push    esi
.text:30127284          push    eax
.text:30127285          call    dword ptr [ecx+11Ch]
```

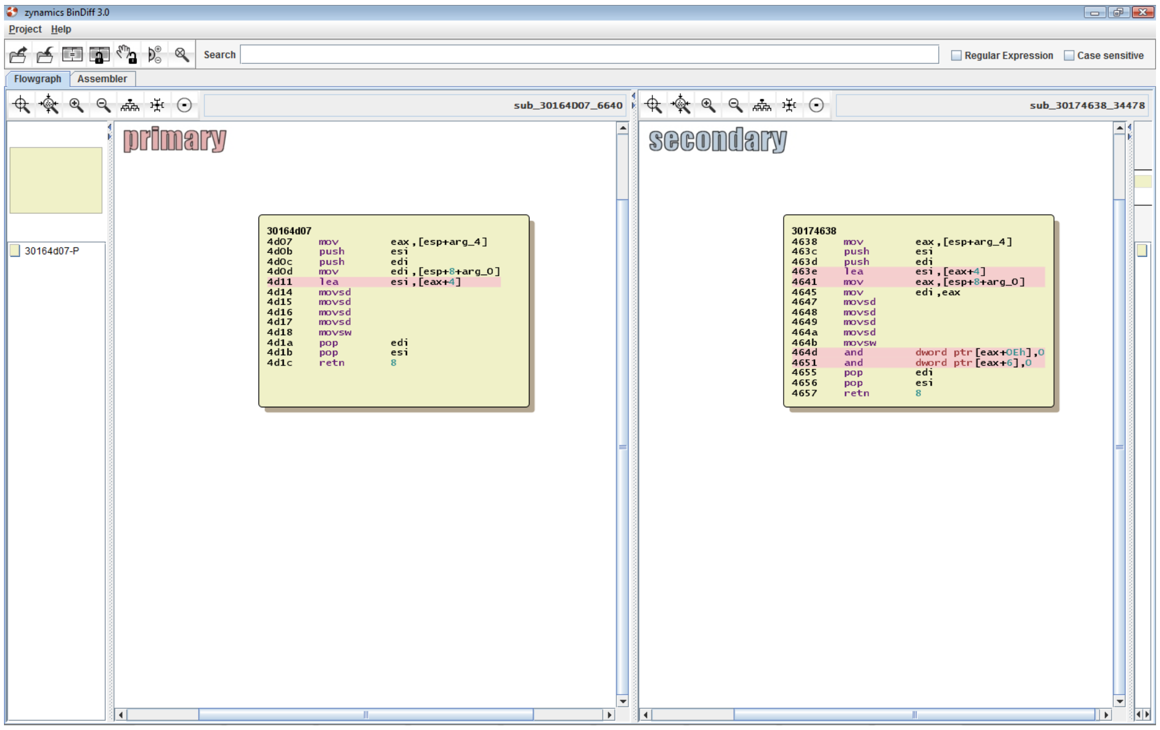

Immunity Debugger - EXCEL.EXE - [CPU - main thread]

```
Registers (FPU)
EAX 307DE5A6 EXCEL.307DE5A6
ECX 307DE48E EXCEL.307DE48E
EDX 00135F7C
EBX 307DE5AC ASCII "AAt"
ESP 00135F5C
EBP 00135F74
ESI 307DE5A6 EXCEL.307DE5A6
EDI 00D77B24

EIP 41414141

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00200206 (NO,NB,NE,A,NS,PE,GE,G)

ST0 empty %#.19L
ST1 empty +UNORM 6847 34306847 FFDFF120
ST2 empty %#.19L
ST3 empty +UNORM 00D0 00000000 34306847
ST4 empty %#.19L
ST5 empty %#.19L
ST6 empty 1.0000000000000000000
ST7 empty 307.48046875000000000
               3 2 1 0      E S P U O Z D I
FST 4000  Cond 1 0 0 0  Err 0 0 0 0 0 0 0 0  (EQ)
FCW 137F  Prec NEAR,64  Mask    1 1 1 1 1 1
```

```
Address  Hex dump                                         ASCII
307D5000 C9 62 19 30 60 7A B0 30 5C 72 B0 30 8C D6 B3 30 Éb■0`z°0\r°0■Ö³0
307D5010 89 72 B0 30 CE 66 B0 30 35 47 B0 30 A7 24 B0 30 ■r°0Îf°05G°0§$°0
307D5020 91 C7 04 30 29 C5 7C 30 F4 33 B0 30 8A 6B B0 30 ‘Ç■0)Å|0ô3°0■k°0
307D5030 A9 7A B0 30 DA 7C B0 30 D0 DA B0 30 F7 66 B0 30 ©z°0Ú|°0ÐÚ°0÷f°0
307D5040 45 40 05 30 54 4A 06 30 F1 8C B0 30 E2 4B 1D 30 E@■0TJ■0ñ■°0âK■0
307D5050 1C 3D B2 30 74 4F B0 30 55 98 B0 30 80 2E B0 30 ■=²0tO°0U■°0■.°0
307D5060 4F F6 14 30 18 95 00 30 1F F1 07 30 23 89 B0 30 Oö■0■■.0■ñ■0#■°0
307D5070 E0 7E B0 30 76 3D B0 30 F7 D5 04 30 35 C5 7C 30 à~°0v=°0÷Õ■05Å|0
307D5080 6F 96 0F 30 41 C5 7C 30 4D C5 7C 30 59 C5 7C 30 o■■0AÅ|0MÅ|0YÅ|0
307D5090 65 C5 7C 30 71 C5 7C 30 8D 96 14 30 81 96 14 30 eÅ|0qÅ|0■■■0■■■0
307D50A0 CC EB 04 30 27 D1 B5 30 C0 10 05 30 EC C4 06 30 Ìë■0'Ñµ0À■■0ìÄ■0
307D50B0 7D C5 7C 30 89 C5 7C 30 95 C5 7C 30 A1 C5 7C 30 }Å|0■Å|0■Å|0¡Å|0
307D50C0 AD C5 7C 30 B9 C5 7C 30 C5 C5 7C 30 4A 3E B0 30 Å|0¹Å|0ÅÅ|0J>°0
307D50D0 D1 C5 7C 30 4D 39 B0 30 7F 6B 0A 30 DD C5 7C 30 ÑÅ|0M9°0■k.0ÝÅ|0
307D50E0 E9 10 B1 30 A7 5C 00 30 43 81 05 30 37 81 05 30 é■±0§\.0C■■07■■0
```

```
00135F5C  3012728B ■r■0 RETURN to EXCEL.3012728B
00135F60  307DE5A6 ¦å}0 EXCEL.307DE5A6
00135F64  307DE5A6 ¦å}0 EXCEL.307DE5A6
00135F68  000000BE ¾...
00135F6C  00135F7C |_■.
00135F70  00000000 ....
00135F74 ┌00136030 0`■.
00135F78 │3012729B ■r■0 RETURN to EXCEL.3012729B from E
00135F7C │307DE5A6 ¦å}0 EXCEL.307DE5A6
00135F80 │00D77B00 .{×.
00135F84 │00000000 ....
00135F88 │019F0C70 p.■■
00135F8C │30B00000 ..°0 OFFSET mso.#3460
00135F90 │00000000 ....
00135F94 │0000029C ■■..
00135F98 │00135FD4 Ô_■.
```

Here you see a comparison between vulnerable and patched version of Excel xp sp3. sub_30164D0 function store content of the ftCmo subrecord in to a buffer. As you see in the patched version firs\t 4bytes of the 12bytes reserved value is set zero.



```
30164d07
4d07    mov     eax,[esp+arg_4]
4d0b    push    esi
4d0c    push    edi
4d0d    mov     edi,[esp+8+arg_0]
4d11    lea     esi,[eax+4]
4d14    movsd
4d15    movsd
4d16    movsd
4d17    movsd
4d18    movsw
4d1a    pop     edi
4d1b    pop     esi
4d1c    retn    8
```

```
30174638
4638    mov     eax,[esp+arg_4]
463c    push    esi
463d    push    edi
463e    lea     esi,[eax+4]
4641    mov     eax,[esp+8+arg_0]
4645    mov     edi,eax
4647    movsd
4648    movsd
4649    movsd
464a    movsd
464b    movsw
464d    and     dword ptr [eax+0Eh],0
4651    and     dword ptr [eax+6],0
4655    pop     edi
4656    pop     esi
4657    retn    8
```

Exploit

we can change EIP value to our arbitrary value. The only thing we should perform is to change some of the values in excel to point the program executing call [ecx+11c] instruction. And because we have value of ecx we can control the execution flow.

On the other hand some of the registers points to our data in excel file so it is simple to set EIP to some call reg value and place our shellcode in a location of the file which that register points.