binamuse
security research

Autodesk AutoCAD

DWG-AC1021 Heap Corruption

Mar 2013

AutoCAD is a software for computer-aided design (CAD) and technical drawing in 2D/3D, being one of the worlds leading CAD design tools. It is developed and sold by Autodesk, Inc.

AutoCad is vulnerable to an arbitrary pointer dereference vulnerability, which can be exploited by malicious remote attackers to compromise a user's system.

This issue is due to AutoCad's failure to properly bounds-check data in a DWG file before using it to index and copy heap memory values. This can be exploited to execute arbitrary code by opening a specially crafted DWG file, version AC1021.

This version was the native fileformat of AutoCAD Release 2007. New versions of the format emerged but AC1021 is still supported in modern AutoCADs for backward compatibility.

Joshep J. Cortez Sanchez
<joshep@binamuse.com>

Felipe Andres Manzano
<feliam@binamuse.com>

1 Description

Title	AutoCAD DWG-AC1021 Memory Corruption
Product	Autodesk AutoCAD
Version	G.55.0.0
Homepage	http://usa.autodesk.com/autocad/
Binary affected	acdb19.dll
Binary MD5	[b654128bed7e19ca6a46f8df755e6b8a]
Advisory	http://www.binamuse.com/advisories/BINA-20130724.txt

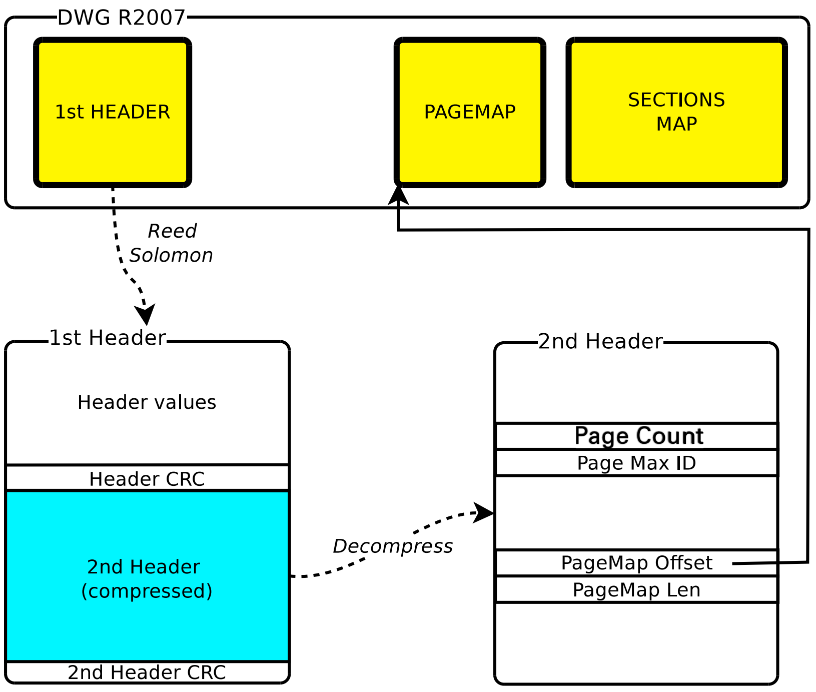
2 The DWG R2007 file format

The R2007 dwg format has sections and pages. There are system sections and data sections. The system sections contain information about where the data sections and their pages are in the file.

The system sections are built based in two main data structures: a *first header* and a *second header*. In addition, there are two important sections in the file structure, the *page map* and the *section map*.

Each one of this sections should be decoded using Reed Solomon algorithm and also could be compressed with a proprietary algorithm (which we will ignore).

The file structure looks like this:



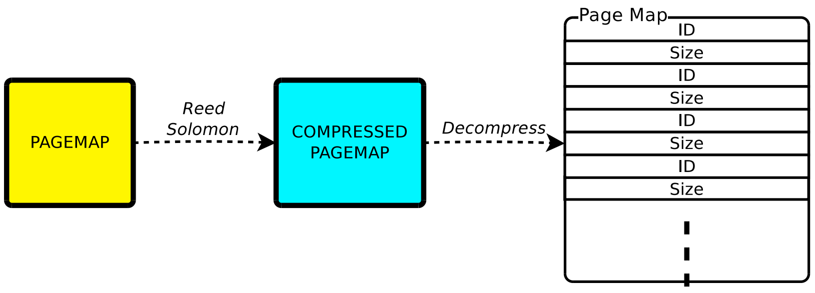
The DWG R2007 also known as AC1021 is well documented by the reversing effort of opendesign.

For more detail on this please check `http://opendesign.com`

3 Vulnerability Details

Not surprisingly AutoCAD starts by parsing the *1st header*. Among other things it reads the size and location of the *2nd header*. Then from the *second header* it reads the position in the file where the *page map* is stored, the number of pages present in the file (`number-of-pages`) and the maximum id (`maxid`) a page shall have. The *page map* is stored in a single system section page and it is composed by tuples *(Id, Size)* where the *Id* is the page number. After the loading of the *page map*, begins the processing of the *section map* that eventually will load all the objects present in the draw.

Graphically the data representing the page map on the file looks like this:



When each **PageMap** node is read two data structures are updated, a double linked list of page map nodes called **PMapList** and an `id` indexed array of node pointers called **PMapArray**. A quick description of this three entities follows.

3.1 The page map nodes

In memory the structure holding a *page map* node has the following fields:

acc: Accumulator of the field size (64 bits)

size: Size of the page. (64 bits)

id: Number of the page. (64 bits)

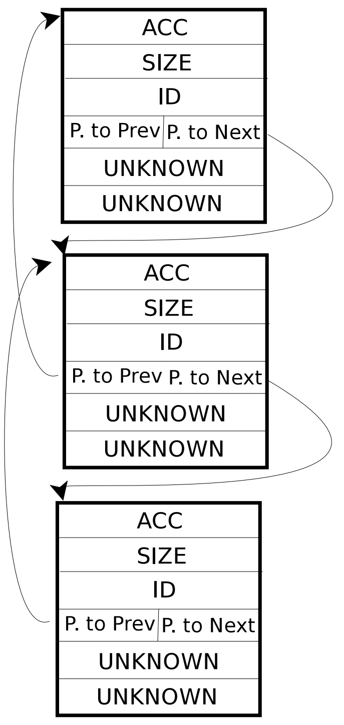
prev: Address in memory of the previous node. (32 bits)

next: Address in memory of the next node.(32 bits)

unknown: there are 2 unknown fields of 64 bits (Not used).

3.2 The page map linked list

And they are linked in the **PMapList** that looks like this:



3.3 The page map array

PMapArray is an array of node pointers maintained for quick access of the *page map* nodes. It maps the id to the actual page map node. It's size is `maxid` as declared in the *2nd header*.

When each *page map* node is created its address is stored in the corresponding `id` position of this array without checking its boundaries. Thus, enabling an arbitrary heap offset overwrite with a pointer to a recently created node.

4 Exploitation details

While parsing an AC1021 dwg file AutoCAD saves all the processed data in a complex structure **R2007Parse**. This structure is `0x5F98` bytes of size and heavily used and updated during the parsing of the file. All data read from the dwg is accessible from this structure.

Assuming that the library `acdb19.dll` is loaded at address `0x60000000`(it varies), **R2007Parse** mega structure is allocated in function `0x60167FD0`.

```
Address    Hex dump          Command
60167FD0    55               PUSH EBP
60167FD1    8BEC             MOV EBP,ESP
60167FD3    83EC 14          SUB ESP,14
...
60167FEB    68 985F0000      PUSH 5F98
60167FF0    8B45 F8          MOV EAX,DWORD PTR SS:[EBP-8]
60167FF3    8B08             MOV ECX,DWORD PTR DS:[EAX]
60167FF5    51               PUSH ECX
60167FF6    FF15 88801E64    CALL DWORD PTR DS:[<&acHeapAlloc>]
60167FFC    8945 F0          MOV DWORD PTR SS:[EBP-10],EAX
60167FFF    837D F0 00       CMP DWORD PTR SS:[EBP-10],0
...
6016803A    8BE5             MOV ESP,EBP
6016803C    5D               POP EBP
6016803D    C3               RETN
```

PMapArray it is allocated here:

```
CPU Disasm
6012C6E4    8985 3CFFFFFF    MOV DWORD PTR SS:[EBP-0C4],EAX
6012C6EA    8B8D C4F8FFFF    MOV ECX,DWORD PTR SS:[EBP-73C]
6012C6F0    8B51 40          MOV EDX,DWORD PTR DS:[ECX+40]
6012C6F3    8995 40FFFFFF    MOV DWORD PTR SS:[EBP-0C0],EDX
6012C6F9    8B85 3CFFFFFF    MOV EAX,DWORD PTR SS:[EBP-0C4]
6012C6FF    ;>>>> the alloc size(page max id) is pushed here
6012C6FF    50               PUSH EAX
6012C700    8B8D 40FFFFFF    MOV ECX,DWORD PTR SS:[EBP-0C0]
6012C706    8B11             MOV EDX,DWORD PTR DS:[ECX]
6012C708    52               PUSH EDX
6012C709    FF15 88801E64    CALL DWORD PTR DS:[<&acHeapAlloc>] ; ALLOC PMAPARRAY
6012C70F    8985 44FFFFFF    MOV DWORD PTR SS:[EBP-0BC],EAX
6012C715    83BD 44FFFFFF 0 CMP DWORD PTR SS:[EBP-0BC],0
6012C71C    74 18            JE SHORT 635FC736
```

As we can go off the **PMapArray** memory limits it would be interesting to have both **R2007Parse** and **PMapArray** allocated one near the other and at stable offset.

4.1 Ordering the memory

By experimentation we've obtained a set of values for `maxid` and `number-of-pages` that locate **R2007Parse** structure and the **PMapArray** in the same memory area. Also as we can load as many page map nodes as we need, they will be eventually be allocated one contiguously after the other in a LFH (see `http://illmatics.com/Understanding_the_LFH.pdf`).

With this layout it is possible to overwrite pointers and data in the big and complex **R2007Parse** structure. We can reach **R2007Parse** from **PMapArray + ID * 4** for some **ID**.

4.2 Controlling the bug

The function that fails to validate the id (index in **PMapArray**) is also in the acdb19.dll library and can be found at 0x63600C80. This function is called after the complete raw *page map* section is read from the file to memory. Its disassembly follows:

```
Address     Hex dump          Command
60130C80     55               PUSH EBP
60130C81     8BEC             MOV EBP,ESP
60130C83     81EC A8000000    SUB ESP,0A8
60130C89     56               PUSH ESI
...
60130F22     6A 04            PUSH 4
60130F24     8B55 B0          MOV EDX,DWORD PTR SS:[EBP−50]
60130F27     52               PUSH ED
60130F28     8B45 AC          MOV EAX,DWORD PTR SS:[EBP−54]
60130F2B     50               PUSH EAX
60130F2C     E8 FFB33000      CALL 6390C330
60130F31     8B8D 68FFFFFF    MOV ECX,DWORD PTR SS:[EBP−98]
60130F37     8B51 54          MOV EDX,DWORD PTR DS:[ECX+54]
; ARRAY BASE
60130F3A     8B4D DC          MOV ECX,DWORD PTR SS:[EBP−24]
60130F3D     890C10           MOV DWORD PTR DS:[EDX+EAX],ECX
; INDEX OVERFLOW
60130F40     8B95 68FFFFFF    MOV EDX,DWORD PTR SS:[EBP−98]
60130F46     8B42 24          MOV EAX,DWORD PTR DS:[EDX+24]
60130F49     83C0 01          ADD EAX,1
60130F4C     8B8D 68FFFFFF    MOV ECX,DWORD PTR SS:[EBP−98]
60130F52     8941 24          MOV DWORD PTR DS:[ECX+24],EAX
60130F55   ^ E9 C1FDFFFF      JMP 63600D1B
....
6013115B     B0 01            MOV AL,1
6013115D     5E               POP ESI
6013115E     8BE5             MOV ESP,EBP
60131160     5D               POP EBP
60131161     C2 0C00          RETN 0C
```

For each page map on the file this function allocates and initializes a node that represents it (0x63600D71). Once each page map node is created and filled with data from the file it is linked accordingly at the **PMapList** list. And a pointer to it is added to **PMapArray** at the fully controlled index id here:

```
60130F3D    890C10        MOV DWORD PTR DS:[EDX+EAX],ECX     ; INDEX OVERFLOW
```

EDX is the **PMapArray** address, EAX is the controlled array index and ECX is the address of the recently created page-map node. The id is blindly expected to be a valid page map index but its value is not validated and it is used to access(and write to) **PMapArray**.

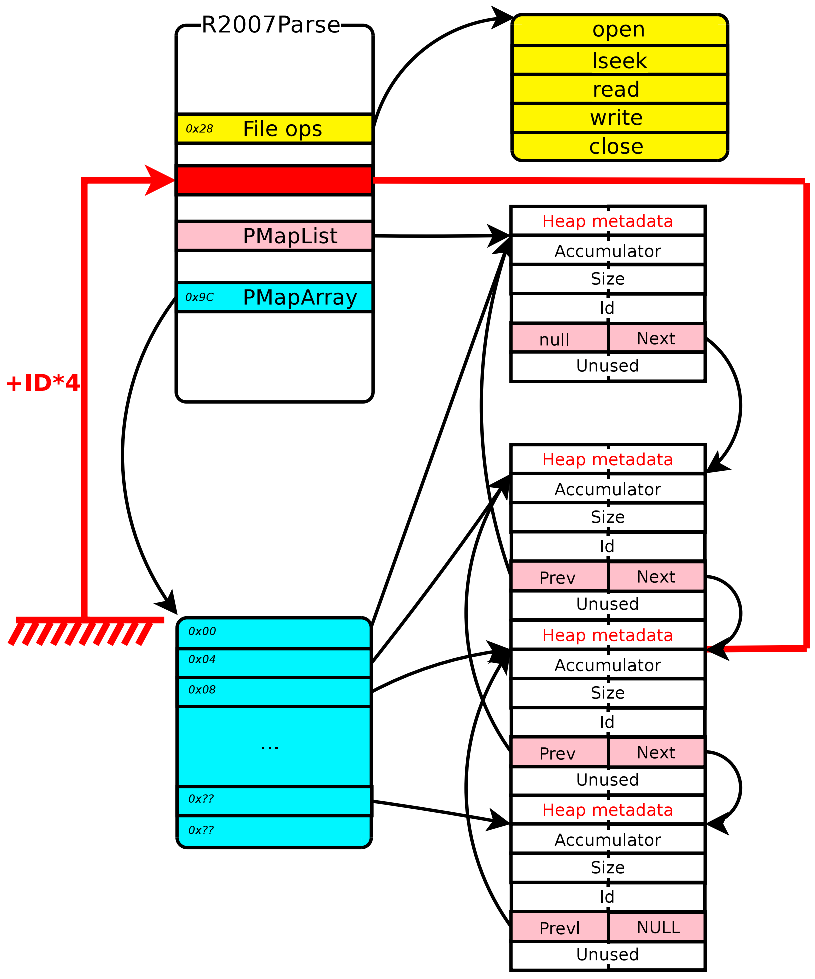
Thus, we can overwrite data in the heap memory relative to **PMapArray**. Specially we'll be able to modify the **R2007Parse** structure and eventually change the flow of execution.

4.3 Changing R2007Parse structure

R2007Parse holds the parsing state at every moment so changing variables there modify greatly the behavior of the following code.

We have determined empirically that when AuctoCAD reads a dwg file with maxid = 360448 and number-of-pages = 2048 it will have both **R2007Parse** and **PMapArray** in the same memory area and separated by a fixed offset: 0x271C0 (The offset varies upon ACAD versions)

Also the nodes starting at the 400-nth, are found to be in contiguous memory one after the other. The next diagram shows how **R2007Parse**, **PMapArray** and the bunch of consecutive nodes layout in memory:



Once **R2007Parse** and **PMapArray** are in the same heap and at a fixed offset of each other, we can overwrite any value in **R2007Parse** with a pointer to a node in the **PMapList**. Note that most of the data in a node is controlled.

The first field to overwrite is **R2007Parse**+0x28. The pointer at this offset is used to control further IO operations on the file. If this field is not zero, the function used to read the data from the file will work in an alternate mode. We will overwrite the +0x28 field forcing this alternate mode and the use of an overloaded READ function that will dereference from the file operations. This can be used to gain control of the execution.

The following function is used to read data from the file, including all sections (header, page map, sections map, etc) and among other things.

```
6012409b  8B95 7CFFFFFF    MOV EDX,DWORD PTR SS:[EBP−84]
601240a1  8995 F4FEFFFF    MOV DWORD PTR SS:[EBP−10C],EDX
601240a7  8B85 F4FEFFFF    MOV EAX,DWORD PTR SS:[EBP−10C]
601240ad  8945 98          MOV DWORD PTR SS:[EBP−68],EAX
601240b0  8B4D 98          MOV ECX,DWORD PTR SS:[EBP−68]
601240b3  894D 9C          MOV DWORD PTR SS:[EBP−64],ECX
601240b6  8B55 A4          MOV EDX,DWORD PTR SS:[EBP−5C]
;EBP−5C is f R2007Parse+8
601240b9  837A 20 00       CMP DWORD PTR DS:[EDX+20],0
<<<<*1*
601240bd  74 32            JE SHORT acdb18.65C0B531
601240bf  8B45 98          MOV EAX,DWORD PTR SS:[EBP−68]
601240c2  50               PUSH EAX
601240c3  8B4D 80          MOV ECX,DWORD PTR SS:[EBP−80]
601240c6  51               PUSH ECX
601240c7  8B55 A4          MOV EDX,DWORD PTR SS:[EBP−5C]
601240ca  8B42 20          MOV EAX,DWORD PTR DS:[EDX+20]
601240cd  8B4D A4          MOV ECX,DWORD PTR SS:[EBP−5C]
601240d0  8B49 20          MOV ECX,DWORD PTR DS:[ECX+20]
601240d3  8B10             MOV EDX,DWORD PTR DS:[EAX]
601240d5  8B42 04          MOV EAX,DWORD PTR DS:[EDX+4]
601240d8  FFD0             CALL EAX
<<<<*2*
601240da  8945 A0          MOV DWORD PTR SS:[EBP−60],EAX
601240dd  837D A0 00       CMP DWORD PTR SS:[EBP−60],0
601240e1  74 0C            JE SHORT acdb18.65C0B52F
601240e3  8B4D A4          MOV ECX,DWORD PTR SS:[EBP−5C]
601240e6  C641 1C 01       MOV BYTE PTR DS:[ECX+1C],1
601240ea  E9 86000000      JMP acdb18.65C0B5B5
601240ef  EB 4D            JMP SHORT acdb18.65C0B57E
601240f1  6A 00            PUSH 0
601240f3  8D55 9C          LEA EDX,DWORD PTR SS:[EBP−64]
601240f6  52               PUSH EDX
601240f7  8B45 98          MOV EAX,DWORD PTR SS:[EBP−68]
601240fa  50               PUSH EAX
601240fb  8B4D 80          MOV ECX,DWORD PTR SS:[EBP−80]
601240fe  51               PUSH ECX
601240ff  8B55 A4          MOV EDX,DWORD PTR SS:[EBP−5C]
60124102  8B02             MOV EAX,DWORD PTR DS:[EDX]
60124104  50               PUSH EAX
60124105  FF15 2C1E7866    CALL DWORD PTR DS:[<&ReadFile>]
<<<< *3*
...
```

Normally, the pointer at offset `+0x28` is NULL (*1*) and the function uses the normal `ReadFile` function(*3*).

After overwriting `RParse2007+0x28` with a valid address, the condition at *1* is met and *2* will be executed. To avoid any memory exceptions before de call is executed, we must overwrite other fields with valid addresses, ie. overwrite fields in **PMapList**. Finally `EAX` will be controlled and the ROP may starts.

The *ROP* chain and the *shellcode* are encoded in the page map of the file using the accumulator and `size` fields. For Autocad 2013 the *ROP* is based in the module `adApplicationFrame.dll` which is fixed in 0x10000000.

If everything went OK the *PoC* shall run a calculator.

5 Notes

We found this bug in `acdb19.dll` functions, but almost the same functionality appear in the module `acsigncore19.dll`. This module is part of the shell extension that handles the digital signature icons and is installed by default and in every update. Opening a folder containing a crafted dwg file in the windows explorer will crash it. The exploit wont work in the windows explorer process as it is configured to use ROP gadgets found only in the acad process. Though it will work when a link to a dwg file is opened from the web browser or when the victim browse for the file using the open menu of Autocad.

To find the interesting code bits in other dll versions use this table:

ALLOC R2007Parse	`68 98 5f 00 00`
ALLOC PMapArray	`8B 8D 40 FF FF FF 8B 11 52 FF 15`
MEM CORRUPTION	`8B 51 54 8B 4D DC 89 0C 10`

6 Usage

An exploit generator for this target is provided, `ACADR2007A2012Exploit.py`.

```
Usage: ACADR2007A2012Exploit.py [options]

Autodesk AutoCAD DWG-2007 Arbitrary Heap Offset Write

Options:
  -h, --help            show this help message and exit
  --verbose             For debugging
  --payload=PAYLOAD     Metasploit payload. Ex. 'windows/exec CMD=calc.exe '
  --output=OUTPUTFILE   Filename of the generated exploit
  --doc                 Print detailed documentation
```

7 References

Original Advisory: http://images.autodesk.com/adsk/files/Autodesk_AutoCAD_Code_ Execution_-Vulnerability_Hotfix_Readme.pdf
Blog Post: http://blog.binamuse.com/2013/07/autocad-dwg-ac1021-heap-corruption.html