Talos Vulnerability Report

TALOS-2020-1032

# BIMx Desktop Viewer Resource Parsing Integer Overflow Vulnerability

NOVEMBER 6, 2020

## CVE NUMBER

CVE-2020-6099

## SUMMARY

An exploitable code execution vulnerability exists in the file format parsing functionality of Graphisoft BIMx Desktop Viewer 2019.2.2328. A specially crafted file can cause a heap buffer overflow resulting in a code execution. An attacker can provide a malicious file to trigger this vulnerability.

## CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

Graphisoft BIMx Desktop Viewer 2019.2.2328

## PRODUCT URLS

BIMx Desktop Viewer - https://www.graphisoft.com/downloads/bimx/bimx_desktop.html

## CVSSV3 SCORE

8.8 - CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

## CWE

CWE-680 - Integer Overflow to Buffer Overflow

## DETAILS

BIMx Desktop Viewer allows for models created by Graphisoft ArchiCad to be shared and viewed by anyone. With Desktop Viewer, clients can view their prospective models without the need of having to install the entire suite of tools needed to create the model itself.

The modules used in this vulnerability are below:

```
00007ff77c5f0000 00007ff77c818000   BIMx       (deferred)
    Image path: BIMx.exe
    Image name: BIMx.exe
    Timestamp:        Wed Jun  5 08:09:29 2019 (5CF7BF09)
    CheckSum:         00000000
    ImageSize:        00228000
    File version:     2019.2.2328.0
    Product version:  2019.2.2328.0
```

The BIMx file format is composed of a variety of resource files which are read and written to disk before processed. To begin processing a given resource file, a 520 byte chunk is read from the file.

```
bimx+55c20
.text:0000000000055C20  mov  r9, r14                ; Input File Stream
.text:0000000000055C23  mov  edx, 208h              ; Number of elements to read
.text:0000000000055C28  mov  r8d, 1                 ; Size of each elements
.text:0000000000055C2E  lea  rcx, [rbp+3F0h+var_470] ; Output buffer
.text:0000000000055C32  call cs:fread               ; Call fread
```

This file chunk contains the name of the resource along with the number of bytes contained in this resource. This chunk looks like the following struct:

```
struct ResourceHeader {
    name: [u8; 512],
    offset_to_struct: u32,
    length_of_data: u32,
}
```

The application then allocates enough memory to fill with the resource bytes. Along with the resource bytes, themselves, the allocation can also contain an attribute or note of what the allocation is for.

```
bimx+78572
.text:0000000000078572  inc     rbx                   ; Increment note pointer
.text:0000000000078575  cmp     byte ptr [rdx+rbx], 0  ; Check if we found the end of the note
.text:0000000000078579  jnz     short loc_78572       ; Continue incrementing
```

The application then calculates the length of the note and then adds that to the found number of bytes for this resource for the final allocation.

```
bimx+78589
.text:0000000000078589  lea     eax, [rbx+6]
.text:000000000007858C  cdq
.text:000000000007858D  and     edx, 0Fh
.text:0000000000078590  lea     edi, [rdx+rax]
.text:0000000000078593  sar     edi, 4
.text:0000000000078596  inc     edi
.text:0000000000078598  shl     edi, 4
.text:000000000007859B  lea     ecx, [rdi+r15]  ; Final add of resource bytes and note length
.text:000000000007859F  movsxd  rcx, ecx
.text:00000000000785A2  call    cs:__imp_malloc
```

Assuming there is no problem with the allocation, the entire allocation is set to 0 and then filled with the resource bytes.

```
bimx+785b0
.text:00000000000785B0                movsxd  rdi, edi        ;
.text:00000000000785B3                mov     edx, 0AAh       ; Memset the note bytes to 0xaa
.text:00000000000785B8                mov     r8, rdi         ; Allocation note size
.text:00000000000785BB                mov     [rsp+38h+arg_0], rbp
.text:00000000000785C0                mov     rcx, r14        ; void *
.text:00000000000785C3                call    memset
.text:00000000000785C8                lea     rbp, [rdi+r14]  ; Address after the allocation note
.text:00000000000785CC                mov     r8, r15         ; Number of resource bytes
.text:00000000000785CF                mov     rcx, rbp
.text:00000000000785D2                xor     edx, edx        ; Fill with 0
.text:00000000000785D4                call    memset
.text:00000000000785D9                movzx   edi, bl
.text:00000000000785DC                mov     rdx, rsi        ; Allocation note
.text:00000000000785DF                mov     r8d, edi        ; Allocation note length
.text:00000000000785E2                mov     rcx, r14        ; New allocation
.text:00000000000785E5                call    memcpy
```

It is possible for an attacker to overflow the malloc allocation size after adding the resource length. This will result in an allocation that is smaller than requested. Since the allocation is smaller than expected, the memset and the memcpy cause an out of bounds write on a heap buffer, potentially resulting in a code execution.

### Crash Information

(1bd9c.1bda4): Access violation - code c0000005 (first chance) First chance exceptions are reported before any exception handling. This exception may be expected and handled. *** WARNING: Unable to verify checksum for BIMx.exe VCRUNTIME140!memset_repmovs+0x9: 00007ffb`d8f91689 f3aa rep stos byte ptr [rdi]

### TIMELINE

2020-03-26 - Initial contact

2020-03-31 - Vendor disclosure

2020-06-30 - 90 day notice

2020-07-03 - Vendor advised reports filtered as spam

2020-07-07 - Issued copy of reports & vendor confirmed

2020-07-28 - Vendor advised new version will address issue mid September

2020-09-15 - Follow up with vendor; no response

2020-11-06 - Public Release

### CREDIT

Discovered by Cory Duplantis of Cisco Talos.

---