Talos Vulnerability Report

TALOS-2021-1290

# AT&T Labs Xmill XML decompression PlainTextUncompressor::UncompressItem heap-based buffer overflow vulnerability

AUGUST 10, 2021

### CVE NUMBER

CVE-2021-21825

### Summary

A heap-based buffer overflow vulnerability exists in the XML Decompression PlainTextUncompressor::UncompressItem functionality of AT&T Labs' Xmill 0.7. A specially crafted XMI file can lead to remote code execution. An attacker can provide a malicious file to trigger this vulnerability.

### Tested Versions

AT&T Labs Xmill 0.7
Schneider Electric EcoStruxure Control Expert 15

### Product URLs

None

### CVSSv3 Score

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

### CWE

CWE-122 - Heap-based Buffer Overflow

### Details

Xmill and Xdemill are utilities that are purpose-built for XML compression and decompression, respectively. These utilities claim to be roughly two times more efficient at compressing XML than other compression methods.

While this software is old, released in 1999, it can be found in modern software suites, such as Schneider Electric's EcoStruxure Control Expert.

During the `Uncompress` functionality of Xdemill, container blocks are decompressed independently. Within `PlainTextUncompressor::UncompressItem` a length provided by the compressed file is used as trusted input for a `memcpy`.

```
void UncompressItem(UncompressContainer *cont,char *dataptr,XMLOutput *output)
{
   unsigned long len=cont->LoadUInt32();

   output->characters((char *)cont->GetDataPtr(len),len);
}
```

Once the length is passed the `XMLOutput::characters` no additional checks are made to check the length of the input versus the provided length, and is passed to `Output::StoreData`.

```
void OUTPUT_STATIC characters(char *str,int len)
{
   switch(x.status)
   {
   case XMLOUTPUT_OPENATTRIB:
      StoreData(str,len);
      return;

   case XMLOUTPUT_OPENLABEL:
      StoreChar('>');

   case XMLOUTPUT_AFTERDATA:
   case XMLOUTPUT_AFTERENDLABEL:
   case XMLOUTPUT_INIT:
      StoreData(str,len);
   }
   x.status=XMLOUTPUT_AFTERDATA;
}
```

`Output::StoreData` passes the function input directly into `memcpy` (mymemcpy is a #define of memcpy) which allows a malicious user to overflow the provided heap-based buffer which can result in remote code execution.

```
void OUTPUT_STATIC StoreData(char *ptr,int len)
   // Stores the data at position 'ptr' of length 'len'
{
   while(bufsize-curpos<len)
   {
      mymemcpy(buf+curpos,ptr,bufsize-curpos);
      len-=bufsize-curpos;
      ptr+=bufsize-curpos;
      curpos=bufsize;
      Flush();
   }
   mymemcpy(buf+curpos,ptr,len);
   curpos+=len;
}
```

Crash Information

```
=================================================================
==5418==ERROR: AddressSanitizer: heap-buffer-overflow on address 0xb4d03478 at pc 0x080f3825 bp 0xbffe8ad8 sp 0xbffe86b0
READ of size 385 at 0xb4d03478 thread T0
    #0 0x80f3824 in __asan_memcpy (/home/fuzz/Desktop/xmill/unix/xdemill+0x80f3824)
    #1 0x8188d6e in Output::StoreData(char*, int) /home/fuzz/Desktop/xmill/./src/Output.hpp:204:7
    #2 0x8185fe1 in XMLOutput::characters(char*, int) /home/fuzz/Desktop/xmill/./src/XMLOutput.hpp:241:10
    #3 0x818e0ba in PlainTextUncompressor::UncompressItem(UncompressContainer*, char*, XMLOutput*)
/home/fuzz/Desktop/xmill/./src/StdCompress.cpp:148:7
    #4 0x81870ad in UncompressContainerBlock::UncompressText(XMLOutput*) /home/fuzz/Desktop/xmill/./src/UnCompCont.hpp:180:7
    #5 0x81840f7 in DecodeTreeBlock(UncompressContainer*, UncompressContainer*, UncompressContainer*, XMLOutput*)
/home/fuzz/Desktop/xmill/./src/Decode.cpp:82:10
    #6 0x8197226 in Uncompress(char*, char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:854:7
    #7 0x8196c37 in HandleSingleFile(char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:248:10
    #8 0x8197482 in HandleFileArg(char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:382:4
    #9 0x81976f5 in main /home/fuzz/Desktop/xmill/./src/Main.cpp:494:7
    #10 0xb7bd5646 in __libc_start_main /build/glibc-ViVLyQ/glibc-2.23/csu/../csu/libc-start.c:291
    #11 0x80664d3 in _start (/home/fuzz/Desktop/xmill/unix/xdemill+0x80664d3)

0xb4d03478 is located 0 bytes to the right of 6008-byte region [0xb4d01d00,0xb4d03478)
allocated by thread T0 here:
    #0 0x810a814 in __interceptor_malloc (/home/fuzz/Desktop/xmill/unix/xdemill+0x810a814)
    #1 0x819977e in SetMemoryAllocationSize(unsigned long) /home/fuzz/Desktop/xmill/./src/MemMan.hpp:119:40
    #2 0x8197cd3 in UncompressBlockHeader(Input*) /home/fuzz/Desktop/xmill/./src/Main.cpp:780:4
    #3 0x8197100 in Uncompress(char*, char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:840:10
    #4 0x8196c37 in HandleSingleFile(char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:248:10
    #5 0x8197482 in HandleFileArg(char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:382:4
    #6 0x81976f5 in main /home/fuzz/Desktop/xmill/./src/Main.cpp:494:7
    #7 0xb7bd5646 in __libc_start_main /build/glibc-ViVLyQ/glibc-2.23/csu/../csu/libc-start.c:291

SUMMARY: AddressSanitizer: heap-buffer-overflow (/home/fuzz/Desktop/xmill/unix/xdemill+0x80f3824) in __asan_memcpy
Shadow bytes around the buggy address:
  0x369a0630: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x369a0640: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x369a0650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x369a0660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x369a0670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x369a0680: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00[fa]
  0x369a0690: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x369a06a0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x369a06b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x369a06c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x369a06d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Heap right redzone:      fb
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack partial redzone:   f4
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==================================================================
Error in file '../fuzz/triage/fut.xmi':
Error while uncompressing container!
```

Timeline

2021-04-30 - Vendor Disclosure

2021-08-10 - Public Release

CREDIT

Discovered by Carl Hurd of Cisco Talos.