Talos Vulnerability Report

TALOS-2021-1292

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

AUGUST 10, 2021

CVE NUMBER

CVE-2021-21829

Summary

A heap-based buffer overflow vulnerability exists in the XML Decompression EnumerationUncompressor::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 independantly. Within `EnumerationUncompressor::UncompressItem` a length provided by the compressed file is used as trusted input for a `memcpy`.

```
void UncompressItem(UncompressContainer *cont,char *dataptr,XMLOutput *output)
   // An item is decompressed by looking up the dictionary
{
   unsigned idx=cont->LoadUInt32();
   EnumDictItem *item=((EnumUncompressState *)dataptr)->itemarray+idx;

   output->characters((char *)item->dataptr,item->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

```
ASAN:DEADLYSIGNAL
=================================================================
==32355==ERROR: AddressSanitizer: SEGV on unknown address 0x6369656c (pc 0xb7caecc0 bp 0xbf9cf488 sp 0xbf9cf028 T0)
    #0 0xb7caecbf  /build/glibc-ViVLyQ/glibc-2.23/string/../sysdeps/i386/i686/multiarch/memcpy-ssse3.S:136
    #1 0x80f37da in __asan_memcpy (/home/fuzz/Desktop/xmill/unix/xdemill+0x80f37da)
    #2 0x8188a4b in Output::StoreData(char*, int) /home/fuzz/Desktop/xmill/./src/Output.hpp:198:10
    #3 0x8185fe1 in XMLOutput::characters(char*, int) /home/fuzz/Desktop/xmill/./src/XMLOutput.hpp:241:10
    #4 0x81a4e8b in EnumerationUncompressor::UncompressItem(UncompressContainer*, char*, XMLOutput*)
/home/fuzz/Desktop/xmill/./src/EnumCompress.cpp:361:7
    #5 0x819fb6a in DivSepUncompressor::UncompressItem(UncompressContainer*, char*, XMLOutput*)
/home/fuzz/Desktop/xmill/./src/DivCompress.cpp:453:10
    #6 0x819f022 in OrSepUncompressor::UncompressItem(UncompressContainer*, char*, XMLOutput*)
/home/fuzz/Desktop/xmill/./src/OrCompress.cpp:330:7
    #7 0x81870ad in UncompressContainerBlock::UncompressText(XMLOutput*) /home/fuzz/Desktop/xmill/./src/UnCompCont.hpp:180:7
    #8 0x81840f7 in DecodeTreeBlock(UncompressContainer*, UncompressContainer*, UncompressContainer*, XMLOutput*)
/home/fuzz/Desktop/xmill/./src/Decode.cpp:82:10
    #9 0x8197226 in Uncompress(char*, char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:854:7
    #10 0x8196c37 in HandleSingleFile(char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:248:10
    #11 0x8197482 in HandleFileArg(char*) /home/fuzz/Desktop/xmill/./src/Main.cpp:382:4
    #12 0x81976f5 in main /home/fuzz/Desktop/xmill/./src/Main.cpp:494:7
    #13 0xb7b9f646 in __libc_start_main /build/glibc-ViVLyQ/glibc-2.23/csu/../csu/libc-start.c:291
    #14 0x80664d3 in _start (/home/fuzz/Desktop/xmill/unix/xdemill+0x80664d3)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV /build/glibc-ViVLyQ/glibc-2.23/string/../sysdeps/i386/i686/multiarch/memcpy-ssse3.S:136
==32355==ABORTING
```

Timeline

2021-04-30 - Vendor Disclosure

2021-08-10 - Public Release

CREDIT

Discovered by Carl Hurd of Cisco Talos.

© 2022 Cisco Systems, Inc. and/or its affiliates. All rights reserved. View our Privacy Policy.