Out-of-bounds memory access in DXF loader (path identification) #4037

Closed **eldstal** opened this issue on Jan 5 · 9 comments

eldstal commented on Jan 5 Contributor

Summary

A DXF-format drawing with particular (not necessarily malformed!) properties may cause an out-of-bounds memory access when imported using `import()`.

Vulnerable versions

- OpenSCAD Linux (commit `eedf370`)
- OpenSCAD Windows x64 (2021.01)

Steps to reproduce

Two Proof-of-concept files are provided. `oob_dxfdata_505` is larger, but more reliably reproduces the fault. `oob_dxfdata_k5` is my attempt at a minimal working example, but only reliably crashes the linux build from the latest commit on the `main` branch (i.e. it does not crash in the windows release). Both illustrate the same crash, so I expect the `_min` example to be more helpful in testing.

oob_dxfdata_505.zip, the larger messier p-o-c
oob_dxfdata_k5.zip, the smaller p-o-c

1. Unzip one of the provided proof of concept files
2. Open the `.scad` file in OpenSCAD
3. Render with F6
4. Observe the application crashing

Alternatively, for headless operation:

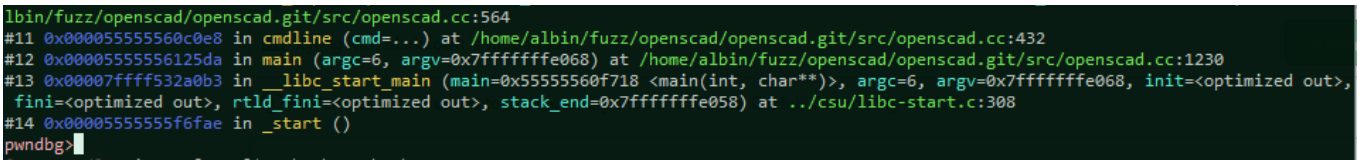
1. Unzip one of the provided proof of concept files
2. `openscad --export-format stl -o /dev/null oob_dxfdata_k5.scad`

3. Observe the segmentation fault

Screenshot

```
Program received signal SIGSEGV, Segmentation fault.                                                    [26/7200]
DxfData::DxfData (this=0x7fffffffc9f0, fn=0, fs=2, fa=12, filename="/home/albin/fuzz/openscad/pocs_manual/oob_dxfdata_k5.dxf", layername
="", xorigin=0, yorigin=0, scale=1) at /home/albin/fuzz/openscad/openscad.git/src/dxfdata.cc:470
470                                     if (grid.eq(ref_point[0], ref_point[1], this->points[lines[k].idx[0]][0], this->points[lines[k].
idx[0]][1])) {
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────────────[ REGISTERS ]─────────────────────────────────────────────
 RAX  0x555556a85528
 RBX  0x555556a21a60 —▸ 0x555556a24040 ◂— 0x1
 RCX  0x0
 RDX  0x8
 RDI  0x7fffffffc110 ◂— 0x555556a85520
 RSI  0x1
 R8   0x555556a21ab0 ◂— 0x500000004
 R9   0x555556a23fe0 —▸ 0x555556a23f20 —▸ 0x555556a21ad0 —▸ 0x555556a23f80 —▸ 0x555556a23f50 ◂— ...
 R10  0x1
 R11  0x7ffff54eebe0 (main_arena+96) —▸ 0x555556a38c10 ◂— 0x0
 R12  0x555556a208e0 —▸ 0x5555569c7c78 —▸ 0x555555b50bf8 (SourceFile::~SourceFile()) ◂— endbr64
 R13  0x1
 R14  0x0
 R15  0x0
 RBP  0x7fffffffc9a0 —▸ 0x7fffffffca80 —▸ 0x7fffffffcb10 —▸ 0x7fffffffcb40 —▸ 0x7fffffffcb80 ◂— ...
 RSP  0x7fffffffc170 ◂— 0x4014660000000000
 RIP  0x5555556fbe26 ◂— movsd  xmm5, qword ptr [rax]
──────────────────────────────────────────────[ DISASM ]───────────────────────────────────────────────
 ► 0x5555556fbe26    movsd  xmm5, qword ptr [rax]
   0x5555556fbe2a    movsd  qword ptr [rbp - 0x820], xmm5
   0x5555556fbe32    mov    eax, dword ptr [rbp - 0x760]
   0x5555556fbe38    movsxd rdx, eax
   0x5555556fbe3b    lea    rax, [rbp - 0x600]
   0x5555556fbe42    mov    rsi, rdx
   0x5555556fbe45    mov    rdi, rax
   0x5555556fbe48    call   0x5555556ff728                <0x5555556ff728>

   0x5555556fbe4d    mov    eax, dword ptr [rax]
   0x5555556fbe4f    movsxd rdx, eax
   0x5555556fbe52    mov    rax, qword ptr [rbp - 0x7d8]
───────────────────────────────────────────[ SOURCE (CODE) ]───────────────────────────────────────────
In file: /home/albin/fuzz/openscad/openscad.git/src/dxfdata.cc
   465                  enabled_lines.erase(current_line);
   466                  auto lv = grid.data(ref_point[0], ref_point[1]);
   467                  for (size_t ki = 0; ki < lv.size(); ++ki) {
   468                          int k = lv.at(ki);
   469                          if (lines[k].disabled) continue;
 ► 470                          if (grid.eq(ref_point[0], ref_point[1], this->points[lines[k].idx[0]][0], this->points[lines[k].idx[0]][
1])) {
   471                                  current_line = k;
   472                                  current_point = 0;
   473                                  goto found_next_line_in_open_path;
   474                          }
   475                          if (grid.eq(ref_point[0], ref_point[1], this->points[lines[k].idx[1]][0], this->points[lines[k].idx[1]][
1])) {
───────────────────────────────────────────────[ STACK ]───────────────────────────────────────────────
00:0000│ rsp 0x7fffffffc170 ◂— 0x4014660000000000
01:0008│     0x7fffffffc178 ◂— 0x4014660000000000
02:0010│     0x7fffffffc180 ◂— 0x4018660000000000
03:0018│     0x7fffffffc188 ◂— 0x3ff0000000000000
04:0020│     0x7fffffffc190 ◂— 0x0
05:0028│     0x7fffffffc198 ◂— 0x0
06:0030│     0x7fffffffc1a0 —▸ 0x555556a21c68 —▸ 0x555556a21c78 ◂— 0x0
07:0038│     0x7fffffffc1a8 —▸ 0x555556a21c48 —▸ 0x555556a22080 ◂— '/home/albin/fuzz/openscad/pocs_manual/oob_dxfdata_k5.dxf'
─────────────────────────────────────────────[ BACKTRACE ]─────────────────────────────────────────────
 ► f 0   0x5555556fbe26
   f 1   0x5555557f124a
   f 2   0x555555f36f49
   f 3   0x5555556192d5
   f 4   0x5555557f277d
   f 5   0x5555557f244d
   f 6   0x555555b38168
   f 7   0x555555b3822c
───────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> bt
#0  DxfData::DxfData (this=0x7fffffffc9f0, fn=0, fs=2, fa=12, filename="/home/albin/fuzz/openscad/pocs_manual/oob_dxfdata_k5.dxf", layer
name="", xorigin=0, yorigin=0, scale=1) at /home/albin/fuzz/openscad/openscad.git/src/dxfdata.cc:470
#1  0x00005555557f124a in ImportNode::createGeometry (this=0x555556a21c10) at /home/albin/fuzz/openscad/openscad.git/src/import.cc:198
#2  0x0000555555f36f49 in GeometryEvaluator::visit (this=0x7fffffffd130, state=..., node=...) at /home/albin/fuzz/openscad/openscad.git/
src/GeometryEvaluator.cc:607
#3  0x00005555556192d5 in NodeVisitor::visit (this=0x7fffffffd130, state=..., node=...) at /home/albin/fuzz/openscad/openscad.git/src/No
deVisitor.h:72
#4  0x00005555557f277d in BaseVisitable::acceptImpl<ImportNode> (state=..., node=..., visitor=...) at /home/albin/fuzz/openscad/openscad
.git/src/BaseVisitable.h:30
#5  0x00005555557f244d in ImportNode::accept (this=0x555556a21c10, state=..., visitor=...) at /home/albin/fuzz/openscad/openscad.git/src
/importnode.h:22
#6  0x0000555555b38168 in NodeVisitor::traverse (this=0x7fffffffd130, node=..., state=...) at /home/albin/fuzz/openscad/openscad.git/src
/NodeVisitor.cc:14
#7  0x0000555555b3822c in NodeVisitor::traverse (this=0x7fffffffd130, node=..., state=...) at /home/albin/fuzz/openscad/openscad.git/src
/NodeVisitor.cc:20
#8  0x0000555555b3822c in NodeVisitor::traverse (this=0x7fffffffd130, node=..., state=...) at /home/albin/fuzz/openscad/openscad.git/src
/NodeVisitor.cc:20
#9  0x0000555555f33c68 in GeometryEvaluator::evaluateGeometry (this=0x7fffffffd130, node=..., allownef=true) at /home/albin/fuzz/opensca
d/openscad.git/src/GeometryEvaluator.cc:63
#10 0x000055555560d585 in do_export (cmd=..., render_variables=..., curFormat=FileFormat::ASCIISTL, root_file=0x555556a208e0) at /home/a
```



Cause

I haven't quite been able to wrap my head around the DXF parser yet, but the OOB access occurs in `src/dxfdata.cc` on one of the lines 470, 475, 505, or 510 depending on the input file.

It appears to be related to the fact that if multiple line segments share points in common, they are merged into contiguous paths. As a part of this, The ADD_LINE macro manipulates the `grid.data<>` and `lines<>` structures. By creating lines either in the `ENTITIES` section of the DXF file or outside of it, an attacker is able to ensure that the `grid.data` entry created on line 108 points to an index in `lines` which never becomes valid.

On line 470 (and the others listed above), this value (`k`) is used as an index into `lines` , out of bounds.

Impact

It appears that the out-of-bounds data can only be read, and not written. An out-of-bounds read does not expose a security vulnerability on its own, but can be used to bypass automatic security features such as stack canaries and pointer encryption.

Proposed mitigation

The algorithm employed in `DxfData::DxfData` needs to be revised to prevent this type of aliasing. Without more insight into the DXF format and how it is used by OpenSCAD, I cannot say for sure where the actual flaw lies.

It seems likely that the `ADD_LINE` macro on lines 108-109 is the culprit, since it inserts values that will be used for indexing even though the values are not yet valid indices.

ChrisCoxArt commented on Jan 15 • edited ▾ Contributor

I see lots of warnings, but no crash on MacOS with the current development code.
I even tried with guard malloc, guard edges, and zombie (use after free) detection enabled.
Ah, but got it to fail with malloc_scribble enabled!
It's reading one entry over the end of the lines vector (size = 10, index = 10). Because of the way vectors double allocations (so this one has 16 values allocated), the address is valid, but the data is bogus, and the data is used as an index to another array, which then gets the access violation. Under normal debugging situations, the memory was probably zero-ed and leading to a valid index.

ChrisCoxArt commented on Jan 15 • edited ▾ Contributor

And there are at least 3 places in the code with the same fragility.
OK, probably best to skip any bad indices, and log a warning about the bad index in the DXF file.

I hope this doesn't break any import tests... Good, it doesn't break any of the tests. I still want to test this with more files, though.

ChrisCoxArt commented on Jan 16 • edited Contributor

I thought maybe we could catch the error earlier in the process (when adding lines), but so far I haven't figured out a way to do it.
This DXF code is more than a little fragile.

BTW - the second example actually hits the same sort of error in a different place, so thanks for providing both!

kintel commented on Jan 16 Member

OpenSCAD's DXF support is really a rather large piece of technical debt. At some point it's worth digging further into replacing it with an existing library, see https://github.com/openscad/openscad/wiki/Project%3A-Improve-DXF-import-and-export

ChrisCoxArt commented on Jan 16 Contributor

Ok, I have a fix that is safe, if not optimal. Trying to catch the error earlier runs into a lot of code that I cannot decipher easily (it's a huge state machine, with many inter-dependent arrays of data) -- so I cannot say that any changes there would be safe.

eldstal commented on Feb 4 Contributor Author

Thanks for taking a look at this issue! Has the fix been merged?

This vulnerability has been assigned CVE-2022-0496.

ChrisCoxArt commented on Feb 4 Contributor

No, the fix has not been merged. It's waiting on other pull requests to be merged (which have been sitting for 3 weeks).

ChrisCoxArt added a commit to ChrisCoxArt/openscad that referenced this issue on Feb 4

add safety to line lookups in DXF import, fixes openscad#4037 ... 6beaca5

ChrisCoxArt commented on Feb 4 — Contributor

Now it has been merged! yay!

t-paul commented on Feb 4 — Member

Hold the horses, it will be in a bit :)

t-paul closed this as completed in 770e323 on Feb 4

---

t-paul added a commit that referenced this issue on Feb 4

Merge pull request #4090 from openscad/dxf-updates ... ✓ 52ce0f7

t-paul added a commit that referenced this issue on Feb 5

CVE-2022-0496 Out-of-bounds memory access in DXF loader. ... 00a4692

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

4 participants