New issue

Jump to bottom

Serializing long double variables leaks uninitialized memory #625

Open **guidovranken** opened this issue on Mar 2, 2020 · 4 comments

guidovranken commented on Mar 2, 2020

Serializing the C/C++ native type `long double` stores uninitialized data into the serialized form.

Compile and run the following program with valgrind to observe this.

```
#include <cereal/archives/binary.hpp>
#include <stdio.h>
#include <sstream>
#include <string>

int main(void)
{
    std::stringstream ss;

    {
        cereal::BinaryOutputArchive archive(ss);
        long double v = 123;
        archive(v);
    }

    {
        const auto s = ss.str();
        FILE* fp = fopen("/dev/null", "wb");
        fwrite(s.data(), s.size(), 1, fp);
        fclose(fp);
    }

    return 0;
}
```

It is apparently an inherent trait of the `long double` type that even an initialized variable leaves some of its raw storage uninitialized.

This gives valgrind errors when compiled with both gcc and clang:

```
#include <stdio.h>
int main(void)
{
    long double v = 0;
    for (int i = 0; i < sizeof(v); i++) {
        printf("%02X", *(((unsigned char*)&v)+i));
    }
}
```

stephentyrone commented on Mar 4, 2020 • edited

`long double` is an 80-bit (10-byte) format on most x86 platforms, but it is either 4- or 8-byte aligned, which means that it has 2 or 6 padding bytes. The compiler does not need to write to those padding bytes, and accessing their contents is unspecified[1], even if you have written to them.

[1] it's *at least* unspecified. C2x says that accessing padding of integers, unions and structures is unspecified, but doesn't say anything about floating-point values, so it may be formally undefined. I haven't dug into the C++ standard on this one.

👍 2

guidovranken commented on Mar 4, 2020 Author

Thank you **@stephentyrone** . Do you know of any portable way to extract the relevant (eg. non-padding) number of bytes of a long double? Something like a `sizeof()` which returns 10 for a long double?

stephentyrone commented on Mar 4, 2020

I don't have a fully general method.

The following will work correctly for any system you are likely to encounter that uses IEEE 754 floating-point types, but is not perfectly general.

```
assert(LDBL_RADIX == 2);
size_t longDoubleValueBytes = (LDBL_MANT_DIG == 64 ? 10 : sizeof(long double));
```

ffontaine commented on Apr 3, 2020

FYI, this issue has been assigned the following CVE number: CVE-2020-11104

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants