std::shared_ptr serialization asymmetry (depends on memory layout) #636

Closed **guidovranken** opened this issue on Mar 27, 2020 · 4 comments · Fixed by #667

Labels bug documentation

guidovranken commented on Mar 27, 2020

Cereal employs caching of `std::shared_ptr` values, using the raw pointer as a unique identifier. This becomes problematic if an `std::shared_ptr` variable goes out of scope and is freed, and a new `std::shared_ptr` is allocated at the same address. Serialization fidelity thereby becomes dependent upon memory layout.

```
#include <cereal/archives/binary.hpp>
#include <cereal/types/memory.hpp>

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

    {
        cereal::BinaryOutputArchive archiveOut(ss);
        {
            std::shared_ptr<bool> v = std::make_shared<bool>(true);
            archiveOut(v);
            printf("v is %p\n", v.get());
            printf("serialized: %d\n", *v);
        }
        {
            std::shared_ptr<bool> v = std::make_shared<bool>(false);
            archiveOut(v);
            printf("v is %p\n", v.get());
            printf("serialized: %d\n", *v);
        }
    }

    {
        cereal::BinaryInputArchive archiveIn(ss);
        std::shared_ptr<bool> v1, v2;
        archiveIn(v1);
        printf("deserialized: %d\n", *v1);
        archiveIn(v2);
        printf("deserialized: %d\n", *v2);
    }
    return 0;
}
```

Output is:

```
v is 0x5578c0144ec0
serialized: 1
v is 0x5578c0144ec0
serialized: 0
deserialized: 1
deserialized: 1
```

The input is (true, false) but the output is (true, true).

👍 3

ffontaine commented on Apr 3, 2020

This issue has been assigned the following CVE number: CVE-2020-11105

Tsubashi mentioned this issue on Aug 11, 2020

Cereal is highlighted as insecure (vulnerable module), refer to CVE-2020-11105 #651

Open

InBetweenNames commented on Oct 18, 2020 Contributor

Yeah unfortunately, the control block of `std::shared_ptr` doesn't encode a unique identifier that can be retrieved to prevent this from happening. Allocators also aren't required to always return unique addresses either, nor is there a way to enforce that. It seems like a hotfix should be released that specifically disables `std::shared_ptr` overloads provided by Cereal in favour of having users define their own, as I'm not sure this is something that can be safely handled at the library level. Serializing without caching would invalidate invariants from many-to-one relationships that could surprise users of `std::shared_ptr` upon deserialization, too.

Alternatively, one could document that correct usage of `std::shared_ptr` in Cereal requires all serialization to be done after all `std::shared_ptrs` are created and before any of them are destroyed. E.g., all serialization happens at exactly one point in time in the program. For programs that uses serialization to store and load global state upon startup and shutdown, this would probably be okay to use.

Cereal already provides a mechanism to disable or override it's own handling of STL types, so the current implementation could be made opt-in rather than opt-out for safety.

serpedon mentioned this issue on Dec 19, 2020

CVE-2020-11105: Store a copy of each serialized shared_ptr within the archive to prevent the shared_ptr to be freed to early. #667

Merged

serpedon commented on Dec 19, 2020 Contributor

Either I am overlooking something or the fix of this problem is quite straight forward, see patch proposal in linked pr #667.

My line of though was the following:
As written already above, correct usage of `std::shared_ptr` in Cereal requires that the `shared_ptr` is still valid at the point when all serialization occurs, usually at the end of the lifetime of the archive. It was suggested to document this constraint to the user, but since we are already dealing with smart pointers, I though, hey, let's implement this constraint by storing our own copy of the `std::shared_ptr`.

Am I right and it is this easy, or am I overlooking something?

InBetweenNames commented on Dec 19, 2020 Contributor

I think this makes sense, it should be documented so that users understand how it will affect the lifetime of their smart pointers, but the approach should fix the CVE. I'm kicking myself for not thinking of it sooner!

AzothAmmo added bug documentation labels on Dec 21, 2020

AzothAmmo closed this as completed in #667 on Feb 1, 2021

Assignees

No one assigned

Labels

bug documentation

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

CVE-2020-11105: Store a copy of each serialized shared_ptr within the archive to prevent the shared_ptr to be freed to early.
serpedon/cereal

5 participants