Crowdstrike C++ NULL Pointer Dereferencing in C++

Kieren Nicôlas Lovell[a,1]

[a] *Pembroke College, University of Cambridge*

Abstract—This article provides an analysis of a recent incident involving CrowdStrike, where a system crash was caused by a null pointer dereference in C++. We explain what a null pointer error is, how it can lead to a system failure, and why such failures are critical when they occur in system drivers loaded at the start of the Windows boot process.

Keywords—*CrowdStrike, C++, Null Pointer, Dereferencing*

Contents

1. Overview of the Incident

Recently, CrowdStrike pushed a forced update that caused widespread system crashes. The root cause was traced to a null pointer dereference in the system driver code. This led to memory access violations, which, in turn, caused the operating system to crash with a blue screen of death (BSOD) [3].

2. Null Pointer Errors in C++

In C++, a null pointer is a special value (typically 0x0) used to indicate that the pointer does not point to a valid object. Dereferencing a null pointer, that is, trying to access the object to which it points, leads to undefined behaviour and typically results in a crash [1].

2.1. Example of Null Pointer Check

Here is a simple example of how programmers should check for null pointers in C++:

```cpp
string* p = get_name();

if (p == NULL) {
    print("Could not get name");
}
```

Code 1. Null Pointer Check

In this example, `get_name()` returns a pointer to a string. Before using this pointer, we check if it is null. If it is, we handle the error appropriately.

3. Crash Analysis

In the CrowdStrike incident, the crash was caused by accessing a null pointer without checking its validity. The memory address attempted to be read was `0x9c`, or 156 in decimal. This indicates that a member of an object was being accessed without verifying whether the object itself was valid.

3.1. Faulty code example

Let's look at this example:

```cpp
struct Obj {
    int a;
    int b;
};

Obj* obj = nullptr;

// Incorrect usage without null check
// This will cause a crash because 'obj' is null
print(obj->a);
```

Code 2. Faulty Null Pointer Dereference

We see here that `obj` is a null pointer. We are trying to access `obj->a` without checking to see if `obj` is null, and as a result we end up with a crash.

4. System Drivers and Boot Process

The system drivers are crucial components of the operating system and often loaded during the boot process. They have privileged access to the system and any failure in these drivers can lead to critical system errors.

4.1. Impact of Null Pointer Dereference in System Drivers

When a null pointer dereference occurs in a system driver, the consequences are severe. Unlike regular applications, which can be terminated without affecting the entire system, a failure in a system driver typically forces the operating system to crash to prevent further damage.

This is exactly what this C++ NULL Pointer update did in this CrowdStrike update, rendering all machines inoperative and upon reboot, crashed immediately and needed the .sys file to be removed locally, after IT users input Bitlocker keys. We can see this from the system error dumps on affected systems.

5. Error Dump Analysis

Now that we have gone through the above points, we can now analyse the crash dump, highlighting the null value causing the issue.

```
EXCEPTION_RECORD:  fffff8d01d83ec28 -- (.exr 0xfffff8d01d83ec28)
ExceptionAddress: fffff8021d735a1 (csagent+0x00000000000e35a1)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 0000000000000000
   Parameter[1]: 000000000000009c

Attempt to read from address 000000000000009c

CONTEXT:  fffff8d01d83ef60 -- (.cxr 0xfffff8d01d83ef60)
rax=fffff8d01d3f2b0 rbx=0000000000000003
rcx=fffff8d01d3f280 rdx=ffff9a81b596605c
```

```
rsi=fffff8d01d3f200 rdi=fffff8d01d83ef60
rbp=fffff8d01d83f1c0 rsp=fffff8d01d83f0d0
r8=0000000000000000 r9=0000000000000000
r10=0000000000000000 r11=0000000000000000
r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
rip=fffff8021d735a1  cs=0010 ss=0018 ds=002b
es=002b fs=0053 gs=002b efl=00050206
```

Code 3. Windows 10 Error Dump Extract

6. Context and Impact

The CrowdStrike-Microsoft outage is considered one of the largest IT crashes in history. It affected multiple sectors worldwide, including financial sectors (stock markets, banks, and NBFCs), public transport, aviation, corporate operations, media broadcasting, and hospitality.

6.1. Incident Details

CrowdStrike CEO George Kurtz clarified that the issue was not a cyberattack, but a defect in a single content update for Microsoft Windows hosts. This defect caused a negative interaction between CrowdStrike's "Falcon Sensor" software and the Microsoft operating system, leading to system crashes.

The outage primarily affected computers running Windows, while Mac and Linux hosts were unaffected. The update conflict caused the system to crash and display the blue screen of death (BSOD). The underlying issue was tied to Microsoft's Azure cloud platform and CrowdStrike's software.

6.2. Impacted Services

- **Financial Sector**: Traders and bankers faced operational disruptions, with a significant impact on stock markets and banking operations globally.
- **Aviation**: Airports and airlines experienced delays and cancellations, with manual check-ins and halted flights in some cases.
- **Health Systems**: Medical procedures were cancelled, and handwritten records were used in place of digital systems.
- **Media and Broadcasting**: Broadcasting services like Sky News and Australia's ABC experienced outages and operated at minimal capacity.

7. Conclusion

The CrowdStrike incident highlights the importance of proper null pointer checks in C++ programming, especially in system-critical code, such as drivers. Modern programming practices and languages that enforce memory safety, such as Rust, can help prevent such errors [2]. Robust update policies and rollback mechanisms are essential to mitigate the impact of faulty updates, because in C++ these errors are very easily made.

However, the rollback method would not be effective in this scenario because the faulty driver is loaded very low in the system stack, specifically during the Windows Startup process. Since the driver is critical for system initialisation, any failure at this level prevents the operating system from booting properly. This makes it impossible to apply a rollback or any other recovery method that relies on the system being operational.

The only way to resolve this issue is by manual intervention, such as booting into a recovery environment and manually removing the faulty driver.

This was possibly the worst possible collision of different failures. A really complex code mistake that was not tested effectively and was deployed to production, and because of the nature of what the module did and as a part of the early startup process, there was no way to roll back remotely or in an automated fashion from the CrowdStrike dashboard.

It will be a long week for a few IT operations teams. Good luck.

References

[1] R. Lischner, *C++ In a Nutshell: A Desktop Quick Reference.* " O'Reilly Media, Inc.", 2003.

[2] M. Noseda, F. Frei, A. Rüst, and S. Künzli, "Rust for secure iot applications: Why c is getting rusty," in *Embedded World Conference, Nuremberg, Germany, 21-23 June 2022*, WEKA, 2022.

[3] A. Satariano, P. Mozur, K. Conger, and S. Frenkel, "CrowdStrike-Microsoft Outage: What Caused the IT Meltdown," *N.Y. Times*, Jul. 2024, ISSN: 0362-4331. [Online]. Available: https://www.nytimes.com/2024/07/19/business/microsoft-outage-cause-azure-crowdstrike.html.