



Tavis Ormandy @tavis0

Jul 20, 2024 · 9 tweets · [tavis0/status/1814762302337654829](#)

This strange tweet got >25k retweets. The author sounds confident, and he uses lots of hex and jargon. There are red flags though... like what's up with the DEI stuff, and who says "stack trace dump"? Let's take a closer look... 1/n



This is actually a screenshot of !analyze -v output, I think the author conflated "stack trace" and "minidump". Regardless, he only looks at the decoded exception record and concludes "it was a NULL pointer"...? 🤔 2/n

It is a plausible explanation, 0x9c is not NULL, but dereferencing near-NULL addresses can have the same root cause. He explains that the code was reading a field at offset 156 from a NULL object pointer. 3/n

Now let's assume the following:

```
Obj* obj = NULL;
```

Then the address of:
 obj is 0
 obj->a is 0 + 4
 obj->b is 0 + 8

So if I do this on a NULL pointer:

```
print(obj->a);
```

The program stack dump like what you'll see above. It will cannot read value 0x000000004

Well, except... we can see in his screenshot that MSVC generated `mov r9d, [r8]`? That's really odd... I spend half my life looking at MSVC output, and I would expect to see `mov r9d, [r8+0x9c]`, so what's up with that? 4/n

```

00 -- (.cxi 0xffffffff0d18d1e68)
=0000000000000000 rcx=0000000000000000
=ffff9a81b596f9a4 rdi=ffff9a81b596a05c
=ffffb0d18d3ee60 rbp=ffffb0d18d3ef60
=0000000000000000 r10=0000000000000000
=ffffb0d18d3ef28 r13=ffffb0d18d3f0d0
=0000000000000004
pi nz na po nc
b es=002b fs=0053 gs=002b efl=00050206
mov r9d,dword ptr [r8] ds:002b:00000000'00
(xbad)
(xnifs)

```

Maybe I'm wrong, let's test it in godbolt . Nope, the code doesn't match! The code is either more complicated, or his hypothesis is incorrect. There is a way to check, he could type ``u`` (unassemble) into kd and examine the surrounding code. 5/n



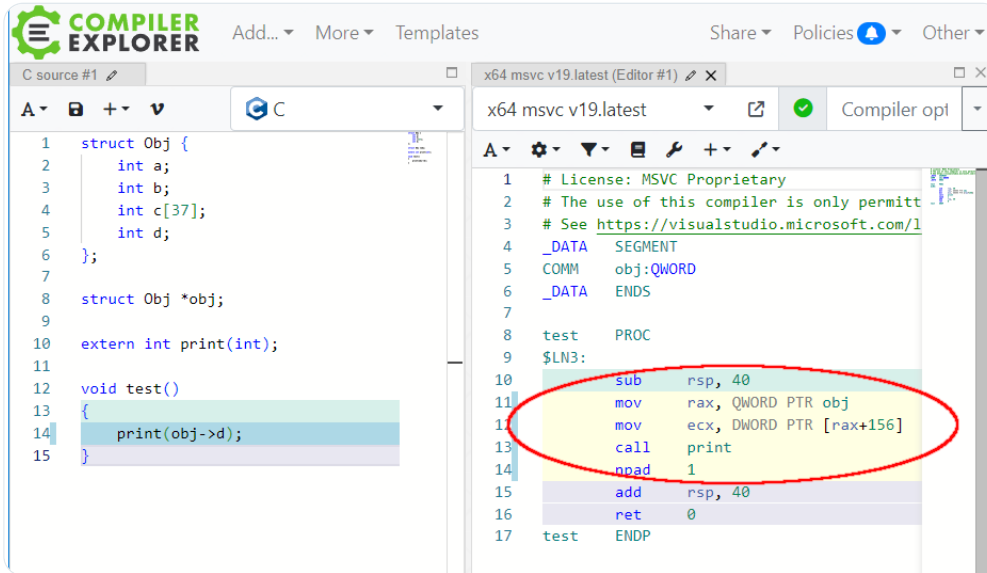
Compiler Explorer - C (x64 msvc v19.latest)

```

struct Obj { int a; int b; int c[37]; int d; }; struct Obj *obj; extern int print(int); void
test() { print(obj->d); }

```

<https://godbolt.org/z/sdz4PGxxo>



He didn't, but we still can! His version of the faulting module has the bytes 45 8b 08 at csagent+0xe35a1, I found that version in VT, and had a look. In fact, there *is* a NULL check (test r8, r8; jz) immediately before the dereference, so his theory is probably *wrong* 🍷 6/n

```

00E359C
00E359C          loc_1400E359C:          ; CODE XREF
00E359C 4D 85 C0          test    r8, r8
00E359F 74 07            jz     short loc_1400E35A8
00E35A1 45 8B 08        mov    r9d, [r8]

```

This code is reading pointers from a table in a loop, and some are invalid. Perhaps an error parsing a configuration file left some entries uninitialized, and one just happened to be 0x9c? It's just a theory, but at least mine fits the facts 😊 7/n

Here is the same crash seen by Patrick, except he saw the entry 0xffff9c8e00000008a, nowhere near NULL! If this is uninitialized data, perhaps it was okay during testing and that's why CS didn't catch it 🤖 8/n

<https://twitter.com/patrickwardle/status/1814343502886477857>

It's amusing to me that Patrick (who actually knows what he's doing) realizes this is complicated, so hedged his analysis with "(initial) details"... but this guy just rocks up with "I'm a professional!!!" and gets 25k retweets 😊 9/9

...