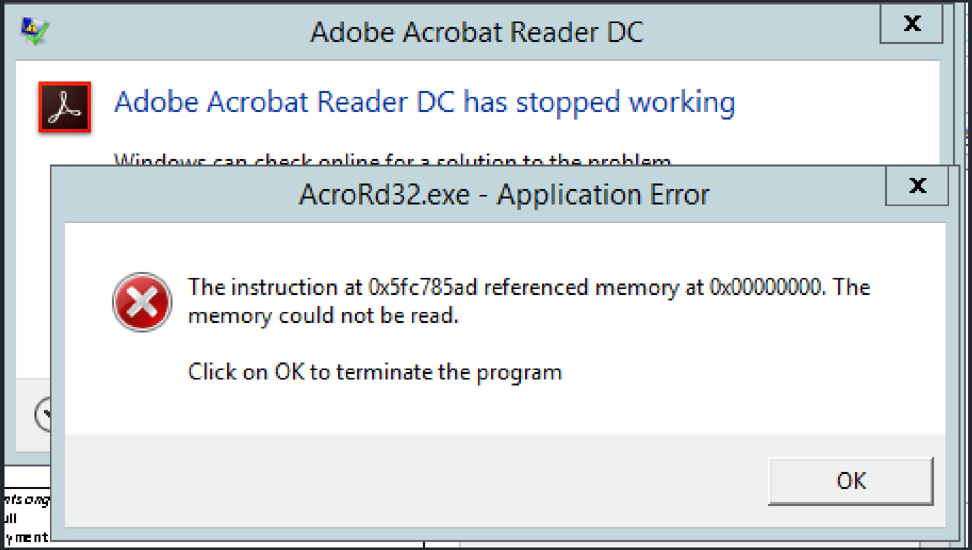
[Learning to Fuzz]

(For the $$$)



Adobe Acrobat Reader DC

Adobe Acrobat Reader DC has stopped working

Windows can check online for a solution to the problem

AcroRd32.exe - Application Error

The instruction at 0x5fc785ad referenced memory at 0x00000000. The memory could not be read.

Click on OK to terminate the program

Learning to Fuzz~$: whoami

$: Y1 NUS Student (BComp, CS)

$: Started playing CTFs in June of my last year of high school and subsequently learned how to do infosec-related research (~6 months of CTF/infosec research then 2 years of brain rot)

$: Interned at STARLabs (Oct 20 – Feb 21): CVE-2021-33760

$: Not a smart guy

Learning to Fuzz~$: Agenda

0x0: What is Fuzzing?

0x1: Tools

0x2: The Fuzzer and the Harness

0x3: Building the Harness

0x4: Testing the Harness

0x5: Running the Fuzzer

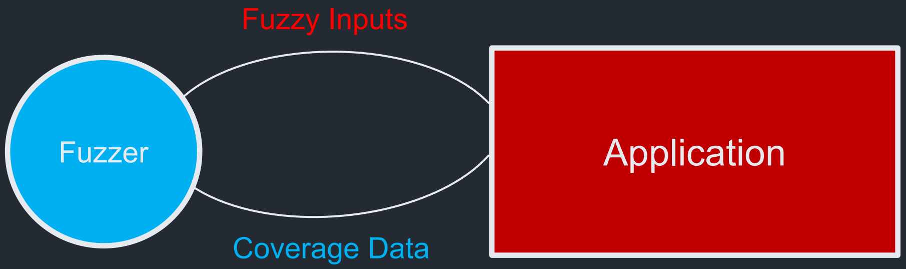
0x6: CVE-2021-33760

Learning to Fuzz~$: What is Fuzzing?

$: Using edge cases to find more edge cases
  ~: Try to execute as much of the code as possible
  ~: Systematically break every part of it

$: Find crashes with past crashes (or base cases)

$: Mutate > Test > Record Crash (if any) > Repeat



Learning to Fuzz~$: Tools

- WinDBG/WinDBG Preview (For Windows)
- GDB + Plugins (For Linux)
- Source Code if any (Chromium source etc.)
- Decompiler like IDA/Ghidra
- Visual Studio (For Windows apps)
- Any IDE you like with (usually) GCC/G++

Learning to Fuzz~$: The Fuzzer and The Harness

> We use the fuzzer to fuzz the application

> We use a harness to “activate” the library we wish to target

> Fuzzer and harness must work together

> Fuzzer runs the harness with base inputs (“start points” to mutate from)

Learning to Fuzz~$: The Fuzzer and The Harness

> Popular fuzzers exist: Peach Fuzzer, American Fuzzy Loop, etc…

> WinAFL: https://github.com/googleprojectzero/winafl

> Fuzzers can execute applications thousands of times per second!

Learning to Fuzz~$: The Fuzzer and The Harness

> Applications are big…

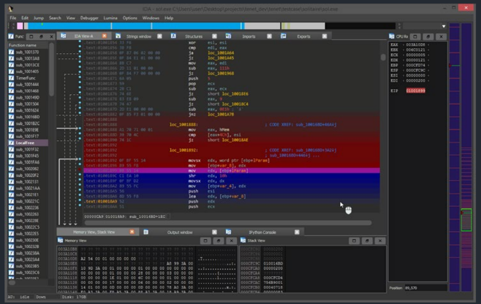
Adobe

Microsoft

> Per-execution cycle is slow

> We are not always interested in the whole application, just the specific library

Learning to Fuzz~$: Building the Harness



```
Function name
sub_1001370
sub_10013A8
sub_10013CE
sub_1001405
TimerFunc
sub_1001444
sub_1001468
sub_100149D
sub_1001504
sub_1001624
sub_10016BD
sub_1001B2C
sub_1001E9E
sub_1001F17
LocalFree
sub_1001F32
sub_1001F45
sub_1001FA6
sub_1002082
sub_10020F2
sub_1002131
sub_10021AA
sub_10021E1
sub_100221C
sub_1002236
sub_1002263
sub_100228E
sub_10022C5
sub_10022E5
sub_100230E
sub_100232B
sub_100238A
sub_10023A4
sub_10023B5
sub_10023C6
```

```
.text:01001854 33 F6               xor     esi, esi
.text:01001856 3B F8               cmp     edi, eax
.text:01001858 0F 87 06 02 00 00   ja      loc_1001A64
.text:0100185E 0F 84 E1 01 00 00   jz      loc_1001A45
.text:01001864 8B C7               mov     eax, edi
.text:01001866 2D 11 01 00 00      sub     eax, 111h
.text:0100186B 0F 84 F7 00 00 00   jz      loc_1001968
.text:01001871 6A 05               push    5
.text:01001873 59                  pop     ecx
.text:01001874 2B C1               sub     eax, ecx
.text:01001876 74 6E               jz      short loc_10018E6
.text:01001878 83 E8 09            sub     eax, 9
.text:0100187B 74 47               jz      short loc_10018C4
.text:0100187D 2D E1 00 00 00      sub     eax, 0E1h ; 'á'
.text:01001882 0F 85 F3 01 00 00   jnz     loc_1001A7B
.text:01001888
.text:01001888                 loc_1001888:                  ; CODE XREF: sub_10016BD+46A↑j
.text:01001888 A1 70 71 00 01      mov     eax, hMem
.text:0100188D 39 70 4C            cmp     [eax+4Ch], esi
.text:01001890 74 1C               jz      short loc_10018AE
.text:01001892
.text:01001892                 loc_1001892:                  ; CODE XREF: sub_10016BD+3A2↑j
.text:01001892                                               ; sub_10016BD+446↑j ...
.text:01001892 0F BF 55 14         movsx   edx, word ptr [ebp+lParam]
.text:01001896 89 55 F8            mov     [ebp+var_8], edx
.text:01001899 8B 55 14            mov     edx, [ebp+lParam]
.text:0100189C C1 EA 10            shr     edx, 10h
.text:0100189F 0F BF D2            movsx   edx, dx
.text:010018A2 89 55 FC            mov     [ebp+var_4], edx
.text:010018A5 56                  push    esi
.text:010018A6 8D 55 F8            lea     edx, [ebp+var_8]
.text:010018A9 52                  push    edx
.text:010018AA 51                  push    ecx
00000CA9 010018A9: sub_10016BD+1EC
```

```
EAX 003A10D8
EBX 006C0121
ECX 00000005
EDX 00000121
EBP 000CFD74
ESP 000CFC9C
ESI 00000000
EDI 00000200

EIP 01001899
```

```
003A10D8 A2 54 00 01 00 00 00 00                          .T......
003A10E8                                  A8 99 3A 00             ..:.
003A10F8 10 9D 3A 00 01 00 00 00 01 00 00 00 00 00 00 00 ..:.............
003A1108 00 00 00 00 01 00 00 00 28 00 00 00 03 00 00 00 ........(.......
003A1118 00 00 00 00 1E 01 00 00 6C 00 00 00 01 00 00 00 ........l.......
003A1128 00 00 00 00 17 00 00 00 06 00 00 00 02 00 00 00 ................
003A1138 14 01 00 00 0D 00 00 00 0D 00 00 00 78 A0 3A 00 ............x.:.
```

```
000CFC9C 010016BD
000CFCA0 00000200
000CFCA8 000CFCD4
000CFCAC 754B9001
000CFCB0 00040718
```

> Decompile the library

> Find out what it does

> Run debugger to see what happens during runtime

> Replicate execution cycle without replicating the whole application

Learning to Fuzz~$: Building the Harness

sxe	:	First-chance handling
bp	:	Set breakpoint
bm	:	Set symbol breakpoint
k	:	View callstack (function calls)
dc	:	Display double word values in given range
dps	:	Display memory in given range
pt	:	Step until next return
g	:	Continue
p	:	Step

Learning to Fuzz~$: Building the Harness

(e)ax: Primary accumulator (return value/input value)
(e)sp: Stack pointer

bx: Base register
cx: Count register
dx: Data register

ip: Instruction pointer
bp: Base pointer

Learning to Fuzz~$: Building the Harness

- Adobe JP2K Library: **JP2KLib.dll**

- Time-Travel Debugging is extremely useful

- If you want to try it out:
  1. Disable PageHeap on Acrobat DC (Google)
  2. Run Acrobat DC in WinDBG Preview
  3. Drag a sample JP2K file into Acrobat DC

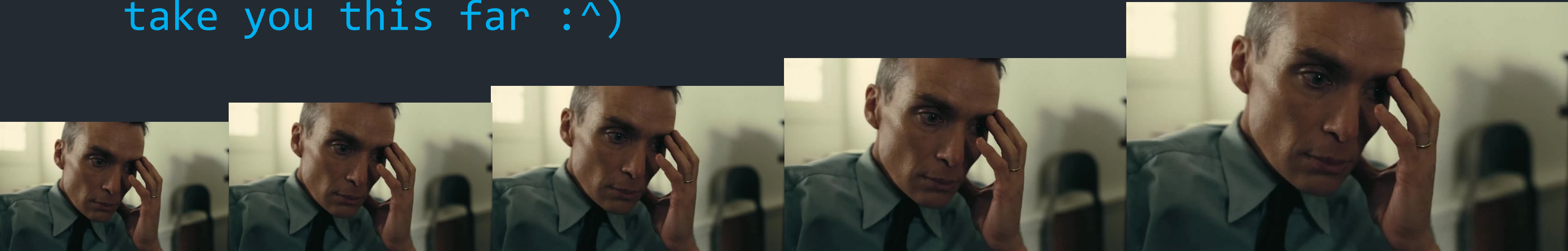
- Demo

Learning to Fuzz~$: Testing the Harness

> This is just like building an application: debug, debug and debug even more.

> Test your harness with in-app debugging as well (logging etc.)

> Test your harness in the debugger! Theory can only take you this far :^)



Learning to Fuzz~$: Running the Fuzzer

> We will make use of DynamoRIO for dynamic instrumentation -> maps library coverage

> Higher coverage = higher chance of finding crash

> We will watch for stability, coverage and executions/s and try to maximize all of them

Learning to Fuzz~$: Evaluation

> Optimizations (achieve similar coverage with less function calls etc.)

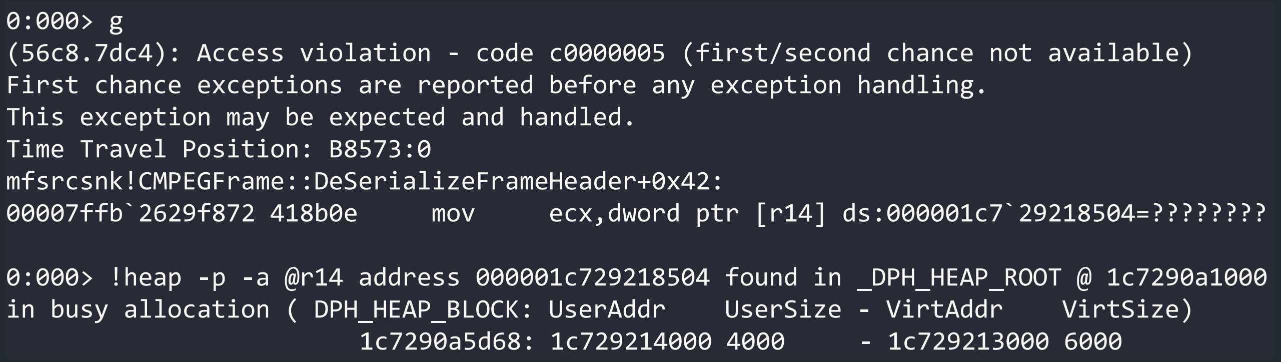
> Further reverse engineering

> Analyse your crashes -> 90% of the time they will be bogus crashes due to measures like sandboxing, exception handling etc.

> A good base input is as important as a good harness

Learning to Fuzz~$: CVE-2021-33760

> Integer underflow leading to OOB-read in Windows Explorer (IPropertyStore parsing)



```
0:000> g
(56c8.7dc4): Access violation - code c0000005 (first/second chance not available)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
Time Travel Position: B8573:0
mfsrcsnk!CMPEGFrame::DeSerializeFrameHeader+0x42:
00007ffb`2629f872 418b0e     mov     ecx,dword ptr [r14] ds:000001c7`29218504=????????

0:000> !heap -p -a @r14 address 000001c729218504 found in _DPH_HEAP_ROOT @ 1c7290a1000
in busy allocation ( DPH_HEAP_BLOCK: UserAddr    UserSize - VirtAddr    VirtSize)
                        1c7290a5d68: 1c729214000 4000     - 1c729213000 6000
```

Learning to Fuzz~$: CVE-2021-33760

CMP3MediaSourcePlugin::ParseHeader() -> for parsing MP3 header

CMP3MediaSourcePlugin::DoScanForFrameHeader() is called when parsing header and stores offset = 0x2282.

```
LABEL_29:
    LODWORD(v34) = offset;
    remaining_size -= offset; // 0x00000000000022e6 - 0x0000000000002282 =
                                 0x0000000000000064
    buf += offset;            // 0x000001c729214000 + 0x0000000000002282 =
                                 0x000001c729216282
    goto LABEL_30;
}
```

Learning to Fuzz~$: CVE-2021-33760

CMP3MediaSourcePlugin::DoReadFirstFrameBody() is called, then CMPEGFrame::DeSerializeFrameBody() is called with the same arguments.

```
// buf=000001c729216282, remaining_size=0000000000000064, &offset=0000003fdc7ce060

hr = CMP3MediaSourcePlugin::DoReadFirstFrameBody(MPEGFrame, buf, remaining_size, &offset);
=====================================================================================================
0:000> k
 # Child-SP          RetAddr               Call Site
00 0000003f`dc7cdee8 00007ffb`2629f789     mfsrcsnk!CMPEGFrame::DeSerializeFrameBody
01 0000003f`dc7cdef0 00007ffb`2629aaa1     mfsrcsnk!CMP3MediaSourcePlugin::ReadMPEGFrameBody+0x49
02 0000003f`dc7cdf60 00007ffb`2629e9ce     mfsrcsnk!CMP3MediaSourcePlugin::DoReadFirstFrameBody+0x41

0:000> r rcx, rdx, r8, r9
rcx=000001c72921bea0 rdx=000001c729216282 r8=0000000000000064 r9=0000003fdc7ce060
```

Learning to Fuzz~$: CVE-2021-33760

Within CMPEGFrame::DeSerializeFrameBody(), its internal check fails as the remaining size 0x64 is less than the required size

```
if ( body_tag == 'ofnI' ) {
    LODWORD(required_size) = required_size + 0x74;
    if ( remaining_size < required_size ) // required_size = 0x74
        goto LABEL_22;
}

LABEL_22:
    CallStackScopeTrace::~CallStackScopeTrace(v13);
    return hr; //returns HRESULT 0
}
```

Offset is used in calculation again! Integer underflow occurs.

```
LODWORD(v34) = offset;
remaining_size -= offset; // 0x0000000000000064 - 0x0000000000002282 = 0x00000000ffffdde2
buf += offset;            // 0x000001c729216282 + 0x0000000000002282 = 000001c729218504
```

Learning to Fuzz~$: CVE-2021-33760

At CMPEGFrame::DeSerializeFrameHeader+0x39 (mfsrcsnk.dll+0xf869), a check is performed. Since remaining_size contains a large value, the < check is not passed. As the code executes to this stage and it tries to access the invalid pointer stored in buf, an OOB read occurs.

```
if ( remaining_size < 4 ) {
    ... // Irrelevant Code
}

v10 = *buf; // OOB Read
```

Learning to Fuzz~$: Afterword

> Fuzzing is not as easy as you think!

> Requires understanding of code execution and lots of debugging

> I am as new as everyone at this, do read up more and don’t take my word as gospel

> Blog: https://ultimatehg.github.io

[End]