Find Me If You Can!

How to Locate a DLL's Unexported Functions

Oryan De Paz

Hello!

Oryan De Paz

Security Researcher @ Akamai Technologies

Former Low-Level Researcher & Developer @ Symantec

- ♡ Windows Internals
- ♡ Reverse Engineering
- ♡ Learning new things

@OryanDP

Attackers Point of View

* Not my nails :)

Let's Hack!

Let's Hack!

ProcessA

```
LoadLibrary(MyDll.dll);
```

Let's Hack!

ProcessA

```
LoadLibrary(MyDll.dll);
```

MyDll.dll

Let's Hack!

Let's Hack!

ProcessA

```
LoadLibrary(MaliciousDll.dll);
```

Let's Hack!

ProcessA

```
LoadLibrary(MaliciousDll.dll);
```

Let's Hack!

ProcessA

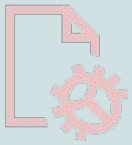
```
MyLoadLib(MaliciousDll.dll);
```

Let's Hack!

ProcessA

```
MyLoadLib(MaliciousDll.dll);
```

MaliciousDll.dll



Reflective Loader

Reflective Loading

- A library injection technique
- The library is responsible for loading itself by implementing a **minimal** Portable Executable (PE) file loader.
- Allocate and execute payloads **directly within the memory of the process**

Blackbone:
Open Source Reflective Loader

Blackbone

Windows memory hacking library

Features

- x86 and x64 support

Process interaction

- Manage PEB32/PEB64
- Manage process through WOW64 barrier

Process Memory

- Allocate and free virtual memory
- Change memory protection
- Read/Write virtual memory

Process modules

Blackbone:
Open Source Reflective Loader

```
void OSFillPatterns(std::unordered_map<ptr_t*,OffsetData>& patterns,
                    SymbolData& result)
{
    GetPattern_LdrpHandleTlsData(patterns, result);
    GetPattern_RtlInsertInvertedFunctionTable(patterns, result);
    GetPattern_RtlpInsertInvertedFunctionTableEntry(patterns, result);
    GetPattern_LdrProtectMrdata(patterns, result);
    GetPattern_LdrpFindOrMapDll(patterns, result);
    GetPattern_KiUserApcDispatcher(patterns, result);
    GetPattern_RtlLookupFunctionTable(patterns, result);
}
```

Let’s find LdrpHandleTlsData

Exported Function?

Function Name	Function Address
NtCreateProcess	0x0009E520
NtCreateThread	0x0009D7D0
NtListenPort	0x0009EE80
LdrLoadDll	0x00016A10

Exported vs. Unexported Functions

Exported

We can import a Dll and call this function directly.

Can be located in memory using GetProcAddress()

Unexported

Exported vs. Unexported Functions

Exported	Unexported
We can import a Dll and call this function directly.	✗
Can be located in memory using GetProcAddress()	✗

Offline Preparations

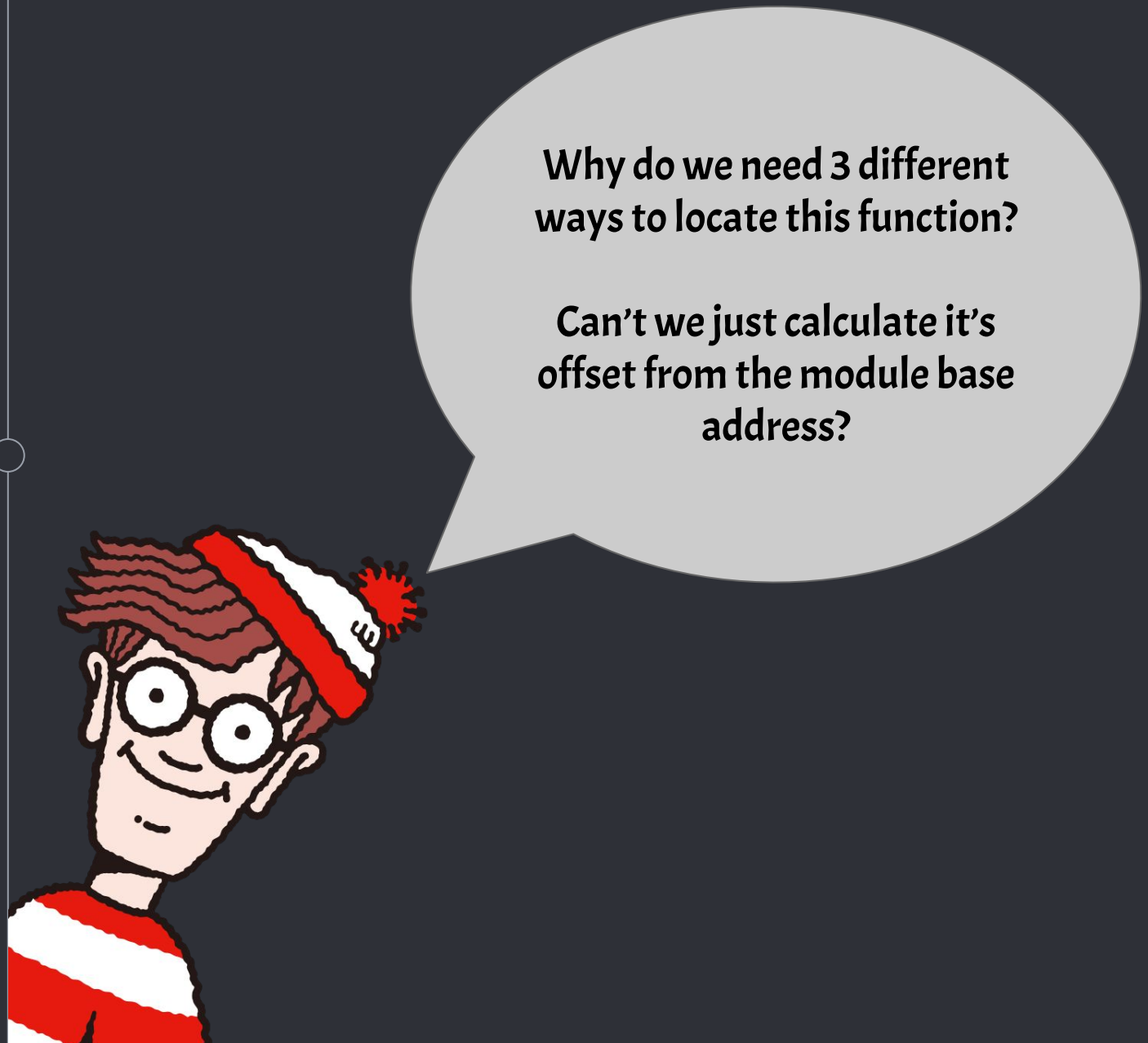
- Reflective loaders need to find these functions during **runtime**.
  - No symbols
  - No function names
  - No guarantee we hit the right function

Offline Preparations

- Reflective loaders need to find these functions during **runtime**.
  - No symbols
  - No function names
  - No guarantee we hit the right function
- Requires a research process in advance

OUR PLAN FOR TODAY

- Explore 3 ways to locate LdrpHandleTlsData
- Automate their search process
- Compare their stability on windows versions
- Choose our favorite way


Why do we need 3 different ways to locate this function?
Can't we just calculate it's offset from the module base address?

1

Unique Byte Sequence

And calculate the offset to start address

LdrpHandleTlsData

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

LdrpHandleTlsData

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

LdrpHandleTlsData

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

LdrpHandleTlsData

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

LdrpHandleTlsData

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

Byte Sequence Address − Offset = **LdrpHandleTlsData()**

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

Byte Sequence Address − Offset = LdrpHandleTlsData()

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

0x180047C5A − Offset = LdrpHandleTlsData()

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

Byte Sequence Address − Offset = LdrpHandleTlsData()

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

Byte Sequence Address − 0x46 = LdrpHandleTlsData()

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

0x180047C5A = 0x46 = LdrpHandleTlsData()

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

0x180047C5A − 0x46 = **LdrpHandleTlsData()** 0x180047C14

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

0x180047C5A − 0x46 = **LdrpHandleTlsData()** 0x180047C14

Automation

Automation - IDAPython

IDA - ntdll_1903.dll.i64 (ntdll_1903.dll) C:\Users\odepaz\Documents\Personal\Talks\BSidesTLV_2022\NtdlIs\64\ntdll_1903.dll.i64

File Edit Jump Search View Debugger Lumina Options Windows Help

Functions

Function name

- RtlStackDbContextCleanup
- RtlStackDbContextSerialize
- RtlStackDbStackAdd
- RtlStackDbStackRemove
- RtlpStackDbEntryCleanup
- RtlpStackDbEntryCreate
- RtlpStackDbEntryIsEqual
- RtlpStackDbRefCountIncrement
- RtlpStackDbSegmentComparitor
- RtlpStackDbSegmentFindOrCreate
- RtlpStackDbSegmentRemoveRef
- RtlpStackDbStackComparitor
- OpenGlobalizationUserSettingsKey_ForMua
- OpenGlobalizationUserSettingsKey_ForSingleUserMode
- RtlAllocateMemoryBlockLookaside
- RtlAllocateMemoryZone
- RtlFreeMemoryBlockLookaside

Line 3914 of 3914

IDA View-A | Hex View-1 | Structures | Enums | Imports | Exports

```
.text:0000000180022180                 var_140         = word ptr -140h
.text:0000000180022180                 var_40          = qword ptr -40h
.text:0000000180022180
.text:0000000180022180                 ; __unwind { // __GSHandlerCheck
.text:0000000180022180 40 55                           push    rbp
.text:0000000180022182 53                              push    rbx
.text:0000000180022183 56                              push    rsi
.text:0000000180022184 57                              push    rdi
.text:0000000180022185 41 56                           push    r14
.text:0000000180022187 41 57                           push    r15
.text:0000000180022189 48 8D 6C 24 88                  lea     rbp, [rsp-78h]
.text:000000018002218E 48 81 EC 78 01 00 00            sub     rsp, 178h
.text:0000000180022195 48 8B 05 44 B3 15 00            mov     rax, cs:__security_cookie
.text:000000018002219C 48 33 C4                        xor     rax, rsp
.text:00000001800219F 48 89 45 60                      mov     [rbp+0A0h+var_40], rax
.text:00000001800221A3 48 8B FA                        mov     rdi, rdx
.text:00000001800221A6 44 89 44 24 40                  mov     [rsp+1A0h+var_160], r8d
.text:00000001800221AB 48 8B D1                        mov     rdx, rcx
.text:00000001800221AE 48 8B D9                        mov     rbx, rcx
.text:00000001800221B1 33 C9                           xor     ecx, ecx
.text:00000001800221B3 41 B8 A8 14 00 00               mov     r8d, 14A8h
.text:00000001800221B9 49 8B F1                        mov     rsi, r9
.text:00000001800221BC E8 E7 33 00 00                  call    LdrpLogDllState
.text:00000001800221C1 48 8D 44 24 60                  lea     rax, [rsp+1A0h+var_140]
```

00021580 0000000180022180: LdrpLoadDll (Synchronized with Hex View-1)

```
BinDiff 7 (@377901646, Jun  7 2021), (c)2004-2011 zynamics GmbH, (c)2011-2021 Google LLC.
BinExport 12 (@377901646, Jun  7 2021), (c)2004-2011 zynamics GmbH, (c)2011-2021 Google LLC.

  bytes   pages size description
-------- ----- ---- --------------------------------------------
  8724480  1065 8192 allocating memory for b-tree...
  6070272   741 8192 allocating memory for virtual array...
   262144    32 8192 allocating memory for name pointers...
--------------------------------------------------------------
 15056896            total memory allocated

Loading processor module C:\Program Files\IDA Pro 7.6\procs\pc64.dll for metapc...Initializing processor module metapc...OK
Loading type libraries...
Autoanalysis subsystem has been initialized.
Database for file 'ntdll_1903.dll' has been loaded.
Hex-Rays Decompiler plugin has been loaded (v7.6.0.210427)
  License:
  The hotkeys are F5: decompile, Ctrl-F5: decompile all.

  Please check the Edit/Plugins menu for more informaton.
-------------------------------------------------------------------------------
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
IDAPython 64-bit v7.4.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
-------------------------------------------------------------------------------
```

Python

AU: idle Down Disk: 738GB

Enter any Python expression

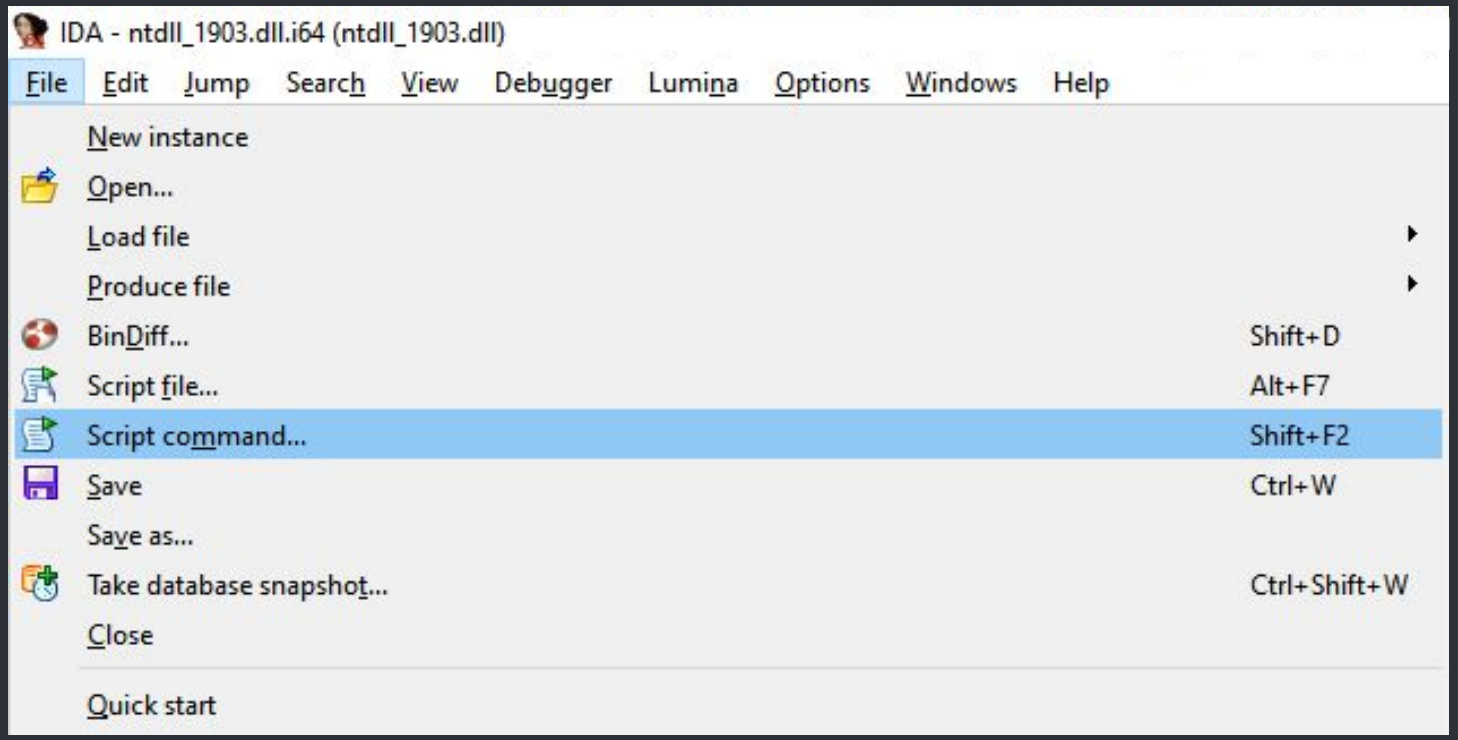
Automation - IDAPython

```
Loading processor module C:\Program Files\IDA Pro 7.6\procs\pc64.dll for metapc...Initializing process
Loading type libraries...
Autoanalysis subsystem has been initialized.
Database for file 'ntdll_1903.dll' has been loaded.
Hex-Rays Decompiler plugin has been loaded (v7.6.0.210427)
  License:
  The hotkeys are F5: decompile, Ctrl-F5: decompile all.

  Please check the Edit/Plugins menu for more informaton.
------------------------------------------------------------------------------------------
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
IDAPython 64-bit v7.4.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
------------------------------------------------------------------------------------------

Python
AU:  idle   Down   Disk: 733GB
```

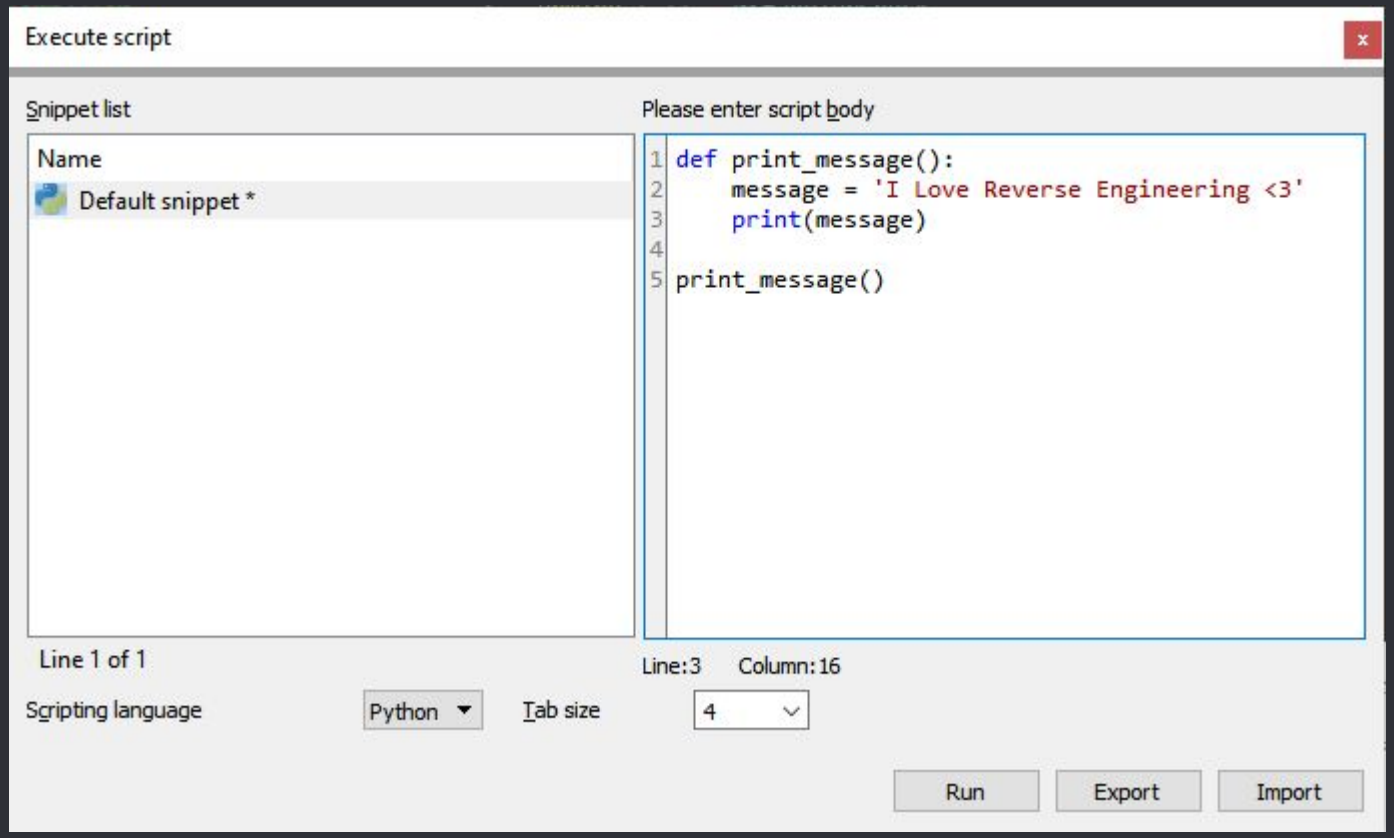
Automation - IDAPython

```
Loading processor module C:\Program Files\IDA Pro 7.6\procs\pc64.dll for metapc...Initializing process
Loading type libraries...
Autoanalysis subsystem has been initialized.
Database for file 'ntdll_1903.dll' has been loaded.
Hex-Rays Decompiler plugin has been loaded (v7.6.0.210427)
  License:
  The hotkeys are F5: decompile, Ctrl-F5: decompile all.

  Please check the Edit/Plugins menu for more informaton.
------------------------------------------------------------------------------------------
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
IDAPython 64-bit v7.4.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
------------------------------------------------------------------------------------------
Python>print('Welcome to BsidesTLV 2022!')
Welcome to BsidesTLV 2022!
Python
AU:  idle   Down   Disk: 733GB
```

Automation - IDAPython



Automation - IDAPython



Automation - IDAPython

Execute script

Snippet list

Name

Default snippet

Please enter script body

```
def print_message():
    message = 'I Love Reverse Engineering <3'
    print(message)

print_message()
```

Line 1 of 1

Line:3 Column:16

Scripting language Python

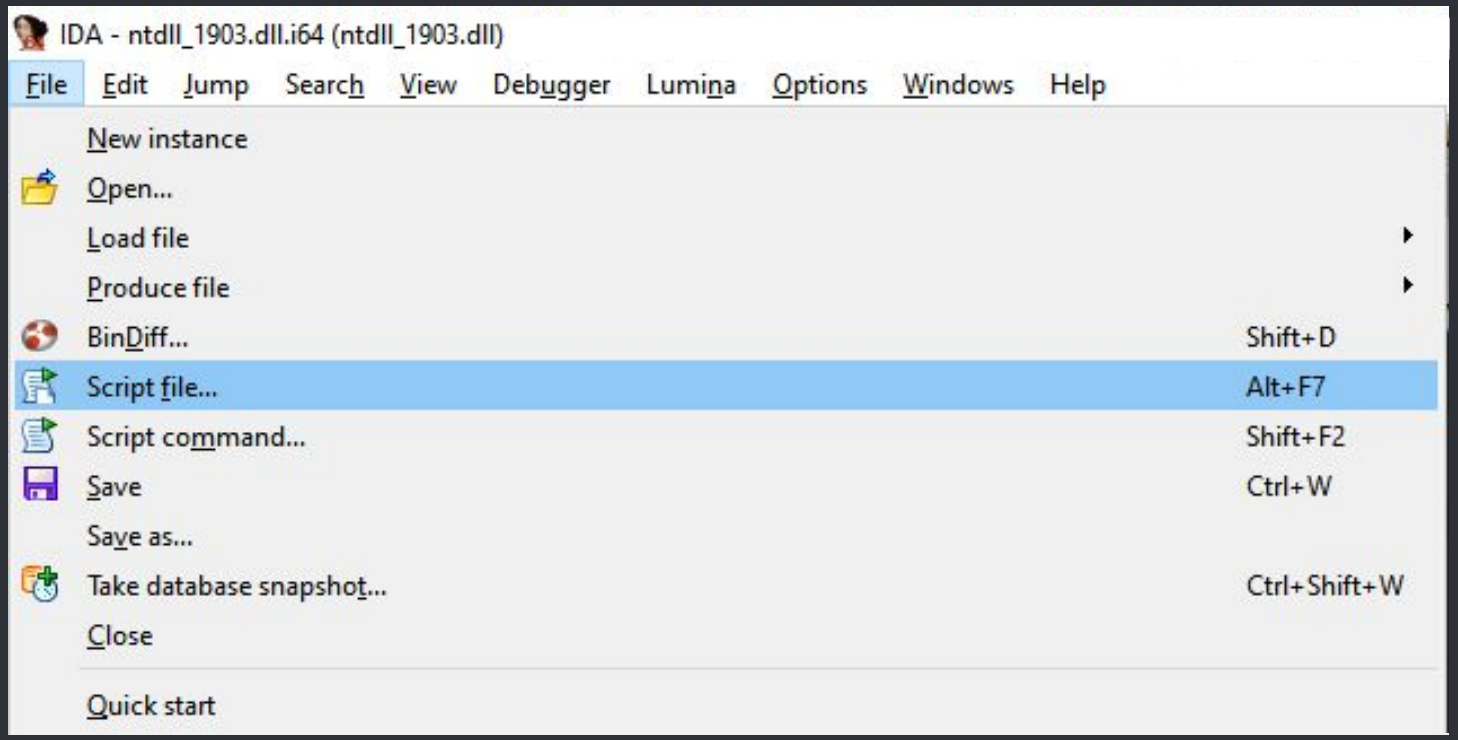
Tab size 4

Run Export Import

```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
IDAPython 64-bit v7.4.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
---------------------------------------------------------------------------------------------
Python>print('Welcome to BSidesTLV 2022!')
Welcome to BSidesTLV 2022!
I Love Reverse Engineering <3
```

Python

Automation - IDAPython



Automation - IDAPython

```
def search_func_by_byte_seq_and_offset(function_name, byte_seq, offset):
    seq_addr = find_byte_sequence(byte_seq)
    return seq_addr - offset == idc.get_name_ea_simple(function_name)

def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = ‘74 33 44 8D 43 09’
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)

main()
```

Automation - IDAPython

```python
def search_func_by_byte_seq_and_offset(function_name, byte_seq, offset):
    seq_addr = find_byte_sequence(byte_seq)
    return seq_addr - offset == idc.get_name_ea_simple(function_name)

def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = ‘74 33 44 8D 43 09’
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)

main()
```

Automation - IDAPython

```
def search_func_by_byte_seq_and_offset(function_name, byte_seq, offset):
    seq_addr = find_byte_sequence(byte_seq)
    return seq_addr - offset == idc.get_name_ea_simple(function_name)

def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = '74 33 44 8D 43 09'
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)

main()
```

Automation - IDAPython

```python
def search_func_by_byte_seq_and_offset(function_name, byte_seq, offset):
    seq_addr = find_byte_sequence(byte_seq)
    return seq_addr - offset == idc.get_name_ea_simple(function_name)

def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = '74 33 44 8D 43 09'
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)

main()
```

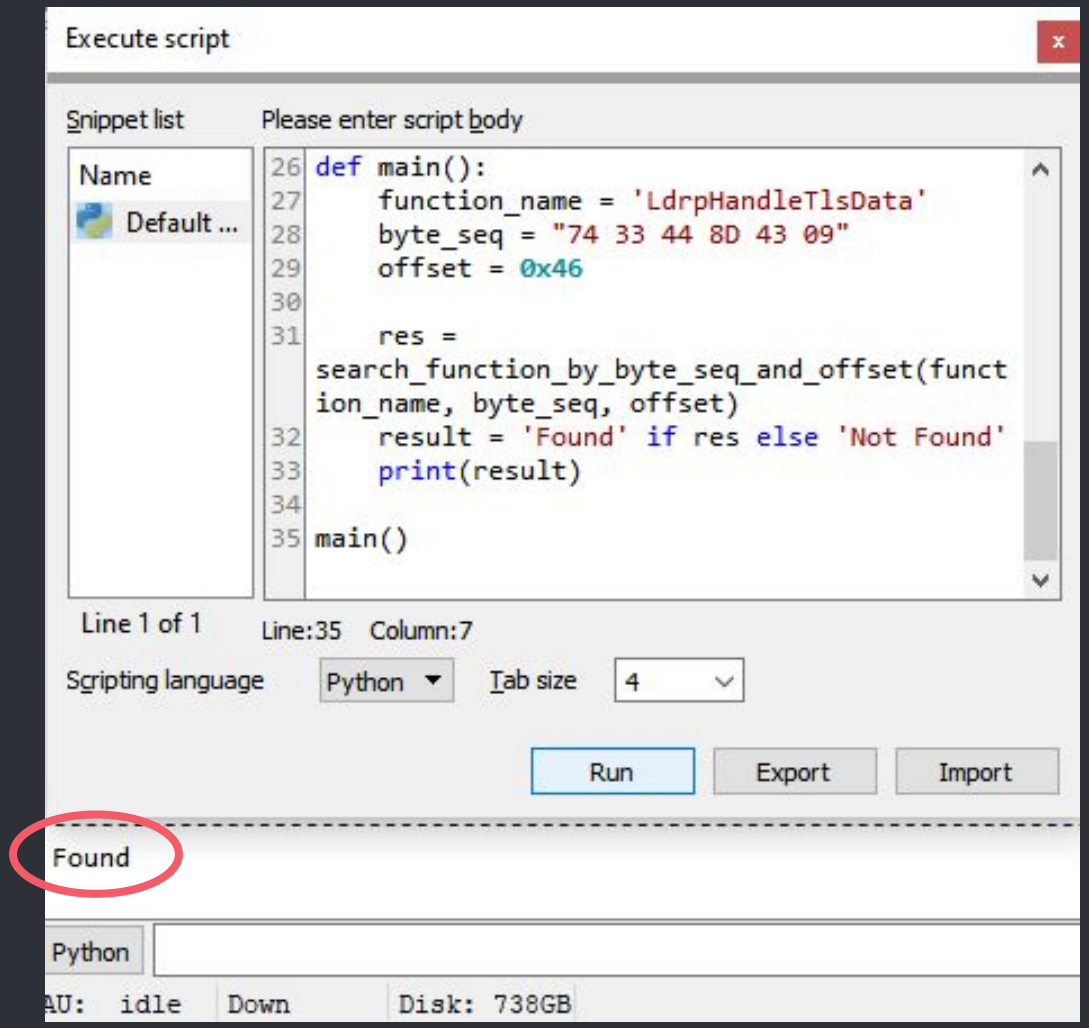
Automation - IDAPython

```
def search_func_by_byte_seq_and_offset(function_name, byte_seq, offset):
    seq_addr = find_byte_sequence(byte_seq)
    return seq_addr - offset == idc.get_name_ea_simple(function_name)

def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = '74 33 44 8D 43 09'
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)

main()
```

Automation - IDAPython

```
def search_func_by_byte_seq_and_offset(function_name, byte_seq, offset):
    seq_addr = find_byte_sequence(byte_seq)
    return seq_addr - offset == idc.get_name_ea_simple(function_name)

def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = ‘74 33 44 8D 43 09’
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)

main()
```

Automation - IDAPython



Run it on multiple Dlls

- How?

Run it on multiple Dlls

- How?

+

Run it on multiple Dlls

- How?
- IDA command line arguments:
  - -A for autonomous mode. IDA will not display dialog boxes.
  - -S for executing scripts,
    in our case running IDAPython scripts

```
> ida64.exe -A -S".\idapython_script.py arg1 arg2"
"C:\Users\odepaz\Documents\ntdll_1903.dll"
```

Run it on multiple Dlls

- How?
- IDA command line arguments:
  - -A for autonomous mode. IDA will not display dialog boxes.
  - -S for executing scripts,
    in our case running IDAPython scripts
- Running multiple IDA cmdlines using **another** Python script

Automation - IDAPython

```python
def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = ‘74 33 44 8D 43 09’
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)


main()
```

Automation - IDAPython

```python
def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = '74 33 44 8D 43 09'
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)


main()
```

```python
win_version = idc.ARGV[1]
is_64 = eval(idc.ARGV[2])
function_name = idc.ARGV[3]
byte_seq = idc.ARGV[4]
offset = int(idc.ARGV[5])
output_path = idc.ARGV[6]
```

Automation - IDAPython

```
def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = '74 33 44 8D 43 09'
    offset = 0x46

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    print(result)

    update_results_in_csv(win_version, function_name, byte_seq, offset,
                          result, output_path)

main()
```

```
win_version = idc.ARGV[1]
is_64 = eval(idc.ARGV[2])
function_name = idc.ARGV[3]
byte_seq = idc.ARGV[4]
offset = int(idc.ARGV[5])
output_path = idc.ARGV[6]
```

Automation - IDAPython

```
def main():
    win_version = idc.ARGV[1]
    is_64 = eval(idc.ARGV[2])
    function_name = idc.ARGV[3]
    byte_seq = idc.ARGV[4]
    offset = int(idc.ARGV[5])
    output_path = idc.ARGV[6]

    idaapi.auto_wait()

    res = search_func_by_byte_seq_and_offset(func_name, byte_seq, offset)
    result = 'Found' if res else 'Not Found'
    update_results_in_csv(win_version, function_name, byte_seq, offset,
                          result, output_path)

    idaapi.set_database_flag(idaapi.DBFL_KILL)
    idc.qexit(0)
main()
```

Automation - Step #2

```
ntdll_files_x64 = [
('ntdll_1507.dll', "Win10_1507", True, "74 33 44 8D 43 09", 0x46),
('ntdll_1511.dll', "Win10_1511", True, "74 33 44 8D 43 09", 0x46),
('ntdll_1607.dll', "Win10_1607", True, "74 33 44 8D 43 09", 0x46),
('ntdll_1803.dll', "Win10_1803", True, "74 33 44 8D 43 09", 0x46),
('ntdll_1809.dll', "Win10_1809", True, "74 33 44 8D 43 09", 0x46),
('ntdll_1903.dll', "Win10_1903", True, "74 33 44 8D 43 09", 0x46),
('ntdll_1909.dll', "Win10_1909", True, "74 33 44 8D 43 09", 0x46),
('ntdll_20h1.dll', "Win10_2004", True, "74 33 44 8D 43 09", 0x46),
('ntdll_20h2.dll', "Win10_20h2", True, "74 33 44 8D 43 09", 0x46)]
```

Automation - Step #2

```
def run_script_for_all_ntdlls(script_path, ntdll_files, func_name,
                              dll_name='ntdll'):
    # Create output files with headers
    header_list = ['Os Version', 'Byte Sequence', 'Offset', 'Result']
    …
    with open(output_path, 'w', newline='') as f:
        dw = csv.DictWriter(f, delimiter=',', fieldnames=header_list)
        dw.writeheader()

    for dll_path, win_version, is_x64, byte_seq, offset in ntdll_files:
        print(win_version)
        ida_filename = 'ida64.exe' if is_x64 else 'ida.exe'
        ida_path = os.path.join(IDA_DIR, ida_filename)

        subprocess.call([ida_path, '-A', '-S"{}" "{}" "{}" "{}" "{}"
 "{}" "{}"'.format(script_path, win_version, is_x64, func_name,
 byte_seq, offset, output_path), dll_path])
```

Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	74 33 44 8D 43 09	70	Not Found
Win10_1511_TH2_x64	74 33 44 8D 43 09	70	Not Found
Win10_1607_RS1_x64	74 33 44 8D 43 09	70	Not Found
Win10_1803_RS4_x64	74 33 44 8D 43 09	70	Not Found
Win10_1809_RS5_x64	74 33 44 8D 43 09	70	Not Found
Win10_1903_19H1_x64	74 33 44 8D 43 09	70	Found
Win10_1909_19H2_x64	74 33 44 8D 43 09	70	Found
Win10_2004_20h1_x64	74 33 44 8D 43 09	70	Found
Win10_20h2_x64	74 33 44 8D 43 09	70	Found

Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	74 33 44 8D 43 09	70	Not Found
Win10_1511_TH2_x64	74 33 44 8D 43 09	70	Not Found
Win10_1607_RS1_x64	74 33 44 8D 43 09	70	Not Found
Win10_1803_RS4_x64	74 33 44 8D 43 09	70	Not Found
Win10_1809_RS5_x64	74 33 44 8D 43 09	70	Not Found
Win10_1903_19H1_x64	74 33 44 8D 43 09	70	Found
Win10_1909_19H2_x64	74 33 44 8D 43 09	70	Found
Win10_2004_20h1_x64	74 33 44 8D 43 09	70	Found
Win10_20h2_x64	74 33 44 8D 43 09	70	Found

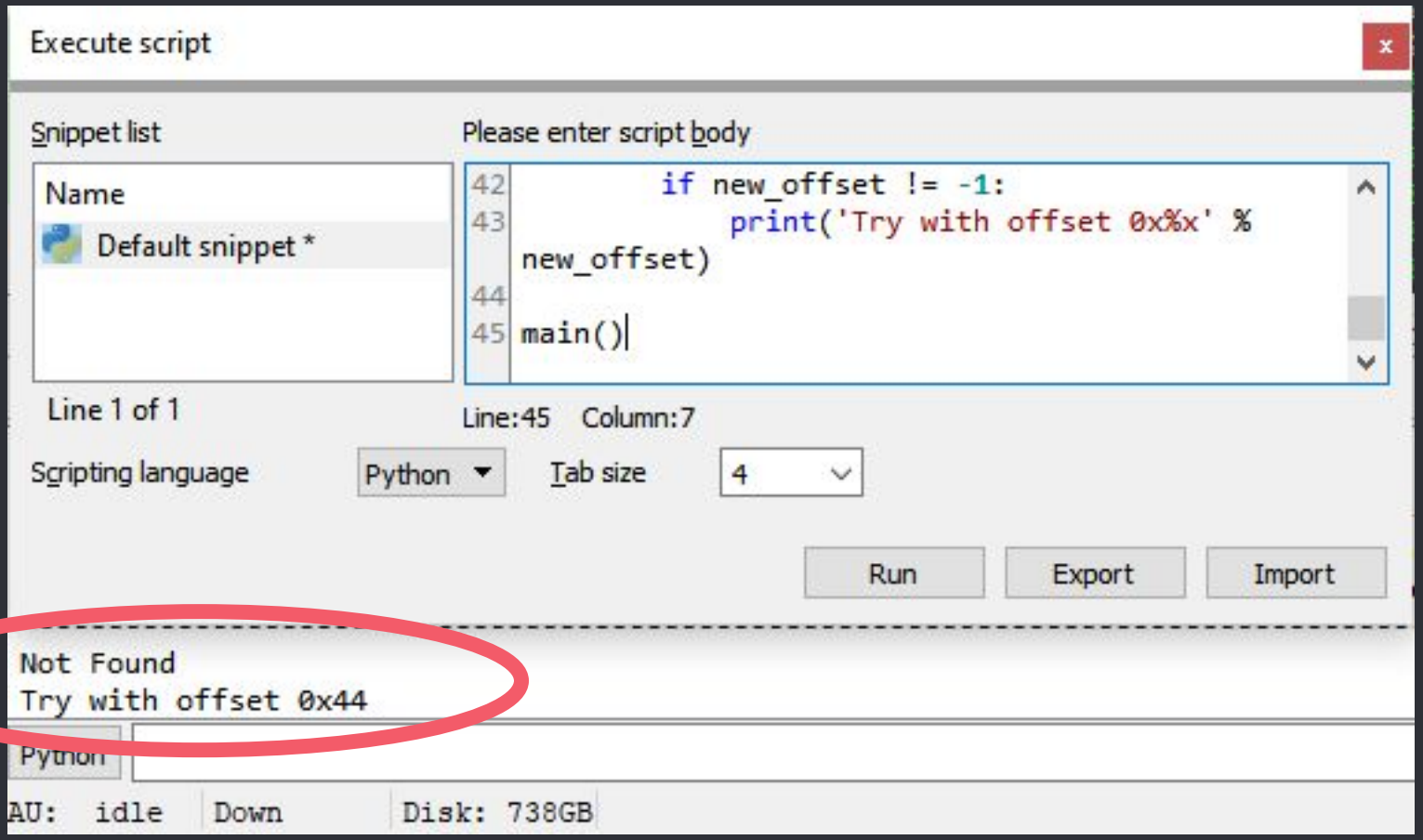
Fixing Search Data

- Does the byte sequence exists?
- Do we need to fix the offset?

Fixing Search Data

```
def get_offset(function_name, byte_seq):
    seq_addr = find_byte_sequence(byte_seq)
    if seq_addr == 0xffffffffffffffff:
        print('Byte sequence was not found')
        return -1
    return seq_addr - idc.get_name_ea_simple(function_name)

def main():
    func_name = 'LdrpHandleTlsData'
    byte_seq = '74 33 44 8D 43 09'
    offset = 0x46

    result = get_func_by_byteseq_and_offset(func_name, byte_seq, offset)
    print('Found' if result else 'Not Found')
    if not result:
        new_offset = get_offset(function_name, byte_seq)
        if new_offset != -1:
            print('Try with offset 0x%x' % new_offset)
```

Fixing Search Data on Win10 1809



Fixing Offsets

```
ntdll_files_x64 = [
('ntdll_1507.dll', "Win10_1507", True, "74 33 44 8D 43 09", 0x44),
('ntdll_1511.dll', "Win10_1511", True, "74 33 44 8D 43 09", 0x44),
('ntdll_1607.dll', "Win10_1607", True, "74 33 44 8D 43 09", 0x44),
('ntdll_1803.dll', "Win10_1803", True, "74 33 44 8D 43 09", 0x44),
('ntdll_1809.dll', "Win10_1809", True, "74 33 44 8D 43 09", 0x44),
('ntdll_1903.dll', "Win10_1903", True, "74 33 44 8D 43 09", 0x46),
('ntdll_1909.dll', "Win10_1909", True, "74 33 44 8D 43 09", 0x46),
('ntdll_20h1.dll', "Win10_2004", True, "74 33 44 8D 43 09", 0x46),
('ntdll_20h2.dll', "Win10_20h2", True, "74 33 44 8D 43 09", 0x46)]
```

First Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	74 33 44 8D 43 09	70	Not Found
Win10_1511_TH2_x64	74 33 44 8D 43 09	70	Not Found
Win10_1607_RS1_x64	74 33 44 8D 43 09	70	Not Found
Win10_1803_RS4_x64	74 33 44 8D 43 09	70	Not Found
Win10_1809_RS5_x64	74 33 44 8D 43 09	70	Not Found
Win10_1903_19H1_x64	74 33 44 8D 43 09	70	Found
Win10_1909_19H2_x64	74 33 44 8D 43 09	70	Found
Win10_2004_20h1_x64	74 33 44 8D 43 09	70	Found
Win10_20h2_x64	74 33 44 8D 43 09	70	Found

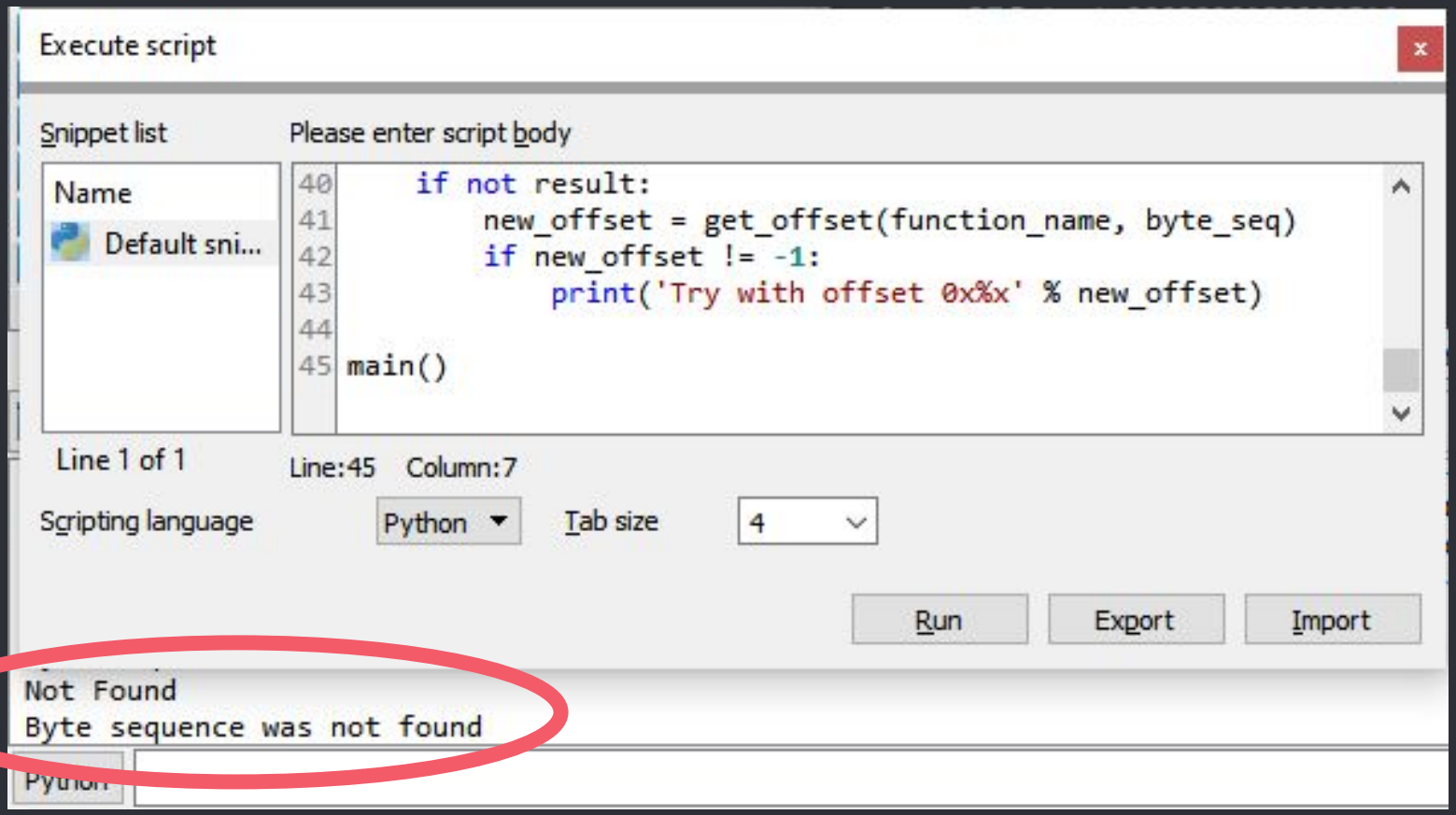
New Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	74 33 44 8D 43 09	68	Not Found
Win10_1511_TH2_x64	74 33 44 8D 43 09	68	Not Found
Win10_1607_RS1_x64	74 33 44 8D 43 09	68	Not Found
Win10_1803_RS4_x64	74 33 44 8D 43 09	68	Found
Win10_1809_RS5_x64	74 33 44 8D 43 09	68	Found
Win10_1903_19H1_x64	74 33 44 8D 43 09	70	Found
Win10_1909_19H2_x64	74 33 44 8D 43 09	70	Found
Win10_2004_20h1_x64	74 33 44 8D 43 09	70	Found
Win10_20h2_x64	74 33 44 8D 43 09	70	Found

New Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	74 33 44 8D 43 09	68	Not Found
Win10_1511_TH2_x64	74 33 44 8D 43 09	68	Not Found
Win10_1607_RS1_x64	74 33 44 8D 43 09	68	Not Found
Win10_1803_RS4_x64	74 33 44 8D 43 09	68	Found
Win10_1809_RS5_x64	74 33 44 8D 43 09	68	Found
Win10_1903_19H1_x64	74 33 44 8D 43 09	70	Found
Win10_1909_19H2_x64	74 33 44 8D 43 09	70	Found
Win10_2004_20h1_x64	74 33 44 8D 43 09	70	Found
Win10_20h2_x64	74 33 44 8D 43 09	70	Found

Fixing Search Data on Win10 1607



Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1511_TH2_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1607_RS1_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1803_RS4_x64	74 33 44 8D 43 09	68	Found
Win10_1809_RS5_x64	74 33 44 8D 43 09	68	Found
Win10_1903_19H1_x64	74 33 44 8D 43 09	70	Found
Win10_1909_19H2_x64	74 33 44 8D 43 09	70	Found
Win10_2004_20h1_x64	74 33 44 8D 43 09	70	Found
Win10_20h2_x64	74 33 44 8D 43 09	70	Found

Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1511_TH2_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1607_RS1_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1803_RS4_x64	74 33 44 8D 43 09	68	Found
Win10_1809_RS5_x64	74 33 44 8D 43 09	68	Found
Win10_1903_19H1_x64	74 33 44 8D 43 09	70	Found
Win10_1909_19H2_x64	74 33 44 8D 43 09	70	Found
Win10_2004_20h1_x64	74 33 44 8D 43 09	70	Found
Win10_20h2_x64	74 33 44 8D 43 09	70	Found

Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1511_TH2_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1607_RS1_x64	44 8D 43 09 4C 8D 4C 24 38	67	Found
Win10_1803_RS4_x64	74 33 44 8D 43 09	68	Found
Win10_1809_RS5_x64	74 33 44 8D 43 09	68	Found
Win10_1903_19H1_x64	74 33 44 8D 43 09	70	Found
Win10_1909_19H2_x64	74 33 44 8D 43 09	70	Found
Win10_2004_20h1_x64	74 33 44 8D 43 09	70	Found
Win10_20h2_x64	74 33 44 8D 43 09	70	Found

Pros

+ The closer we get to the function address, it is less likely that the offset would change

Pros

+ The closer we get to the function address, it is less likely that the offset would change

Cons

− What if there is no unique byte sequence inside the function?

2

Locate Direct Function Call

Locate the unique byte sequence of the function call

LdrpDoPostSnapWork

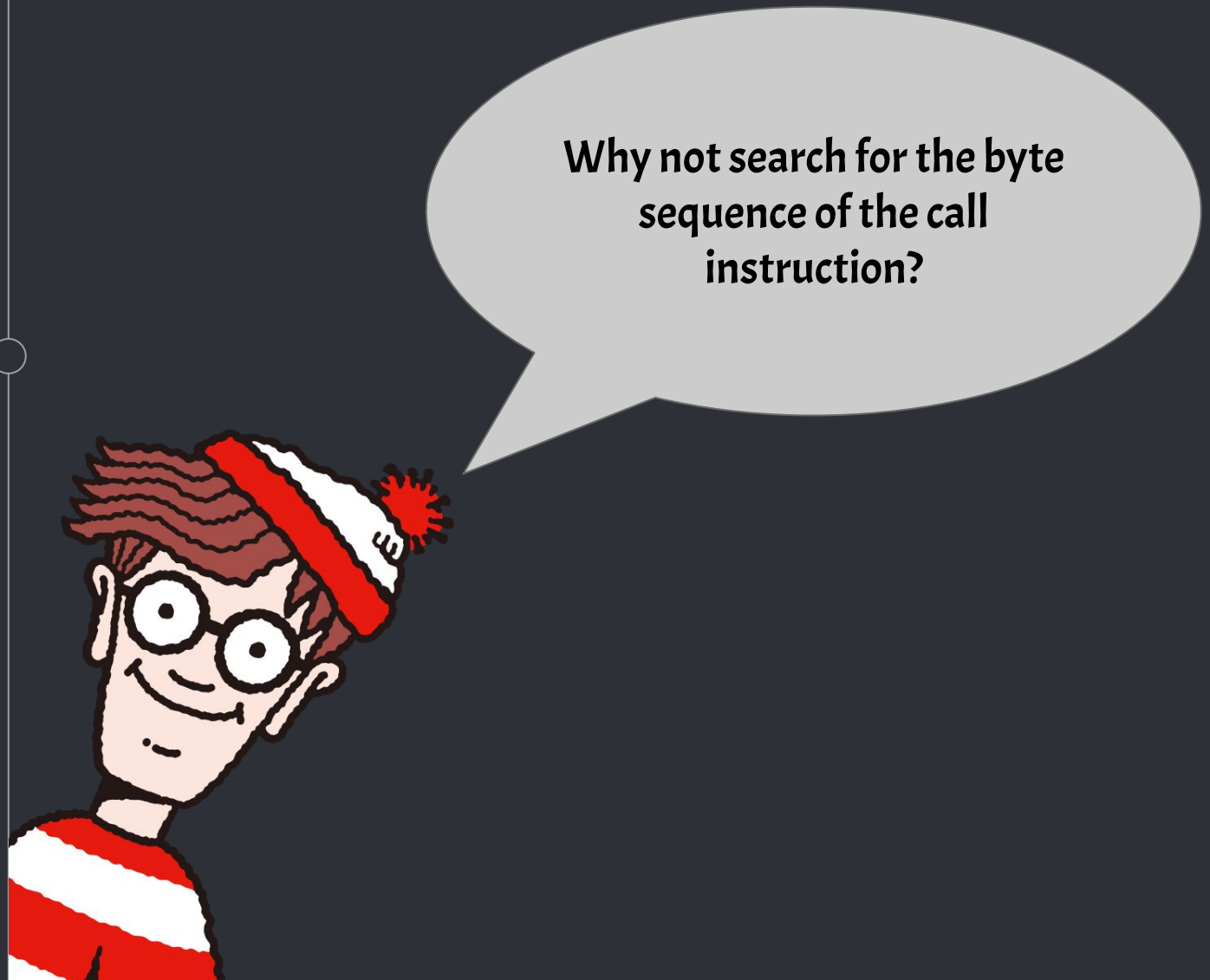
```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```


Why not search for the byte sequence of the call instruction?

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

Byte Sequence Address + Byte Sequence Length + Offset = LdrpHandleTlsData()

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```

Byte Sequence Address + Byte Sequence Length + Offset = LdrpHandleTlsData()

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

0x180047BBE + Byte Sequence Length + Offset = LdrpHandleTlsData()

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```

0x180047BBE + 11 + Offset = LdrpHandleTlsData()

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

0x180047BBE + 11 + 4 15 + Offset = LdrpHandleTlsDa ta()

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

0x180047BBE + 15 + 0x47 = LdrpHandleTlsData()

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

0x180047BBE + 15 + 0x47 = LdrpHandleTlsData() 0x180047C14

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00     cmp     rax, [rdi+90h]
0F 85 BE 09 07 00        jnz     loc_1800B857C
48 8B 4F 38              mov     rcx, [rdi+38h]
66 39 71 6E              cmp     [rcx+6Eh], si
75 0B                    jnz     short loc_180047BD3
E8 47 00 00 00           call    LdrpHandleTlsData
8B D8                    mov     ebx, eax
85 C0                    test    eax, eax
78 0B                    js      short loc_180047BDE
```

0x180047BBE + 15 + 0x47 = LdrpHandleTlsData() 0x180047C14

LdrpHandleTlsData

Offset

```
48 89 4C 24 50              mov     [rsp+118h+var_C8], rcx
48 89 8C 24 C8 00 00 00     mov     [rsp+118h+var_50], rcx
33 DB                       xor     ebx, ebx
39 1D E6 26 12 00           cmp     cs:LdrpActiveThreadCount, ebx
74 33                       jz      short loc_180047C8F
44 8D 43 09                 lea     r8d, [rbx+9]
48 8D 4C 24 68              lea     rcx, [rsp+118h+var_B0]
48 89 4C 24 20              mov     [rsp+118h+var_F8], rcx
4C 8D 4C 24 38              lea     r9, [rsp+118h+var_E0]
```

0x180047C5A − 0x46 = **LdrpHandleTlsData()** 0x180047C14

Automation - IDAPython

```
def get_func_by_byteseq_from_call(function_name, byte_seq, offset):
    seq_addr = find_byte_sequence(byte_seq)
    call_addr = seq_addr + offset - 1
    found_function_addr = get_operand_value(call_addr, 0)

    return found_function_addr == idc.get_name_ea_simple(function_name)

def main():
    win_version = idc.ARGV[1]
    …
    from_call = eval(idc.ARGV[7])
    …
    if from_call:
        result = get_func_by_byteseq_from_call(func_name, byte_seq, offset)
    else:
        result = get_func_by_byteseq_and_offset(func_name, byte_seq, offset)
    …
main()
```

Run it on multiple Dlls

```
ntdll_files_x64_second_method = [
('ntdll_1507.dll', "Win10_1507", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1511.dll', "Win10_1511", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1607.dll', "Win10_1607", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1803.dll', "Win10_1803", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1809.dll', "Win10_1809", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1903.dll', "Win10_1903", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1909.dll', "Win10_1909", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_20h1.dll', "Win10_2004", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_20h2.dll', "Win10_20h2", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11)]
```

Run it on multiple Dlls

```
ntdll_files_x64_second_method = [
('ntdll_1507.dll', "Win10_1507", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1511.dll', "Win10_1511", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1607.dll', "Win10_1607", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1803.dll', "Win10_1803", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1809.dll', "Win10_1809", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1903.dll', "Win10_1903", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_1909.dll', "Win10_1909", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_20h1.dll', "Win10_2004", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11),
('ntdll_20h2.dll', "Win10_20h2", True, "48 8B 4B 30 66 39 79 6E 75 0C E8", 11)]

def run_script_for_all_ntdlls(script_path, ntdll_files, func_name, dll_name):
    # Create output files with headers
    if from_call:
        output_file = '%s_%s_by_byteseq_from_call.csv' % (dll_name, func_name)
    else:
        output_file = '%s_%s_by_byteseq.csv' % (dll_name, func_name)
    …
    with open(output_path, 'w', newline='') as f:
        …
```

Results

Os Version	Byte Sequence	Offset	Result
Win10_1507_TH1_x64	48 8B 4B 30 66 39 79 6E 75 0C E8	11	Found
Win10_1511_TH2_x64	48 8B 4B 30 66 39 79 6E 75 0C E8	11	Found
Win10_1607_RS1_x64	48 8B 4B 30 66 39 79 6E 75 0C E8	11	Found
Win10_1803_RS4_x64	48 8B 4F 38 66 39 71 6E 75 0B E8	11	Found
Win10_1809_RS5_x64	48 8B 4F 38 66 39 71 6E 75 0B E8	11	Found
Win10_1903_19H1_x64	48 8B 4F 38 66 39 71 6E 75 0B E8	11	Found
Win10_1909_19H2_x64	48 8B 4F 38 66 39 71 6E 75 0B E8	11	Found
Win10_2004_20h1_x64	48 8B 4F 38 66 39 71 6E 75 0B E8	11	Found
Win10_20h2_x64	48 8B 4F 38 66 39 71 6E 75 0B E8	11	Found

Pros

+ It doesn’t count on random bytes, it has some logic behind it

Pros

+ It doesn't count on random bytes, it has some logic behind it

Cons

− What if the byte sequence prior to the call is **not unique**?

3

Locate Indirect Function Call

Find the exported function and dive into the inner calls until we find the call to LdrpHandleTlsData

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```

LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```

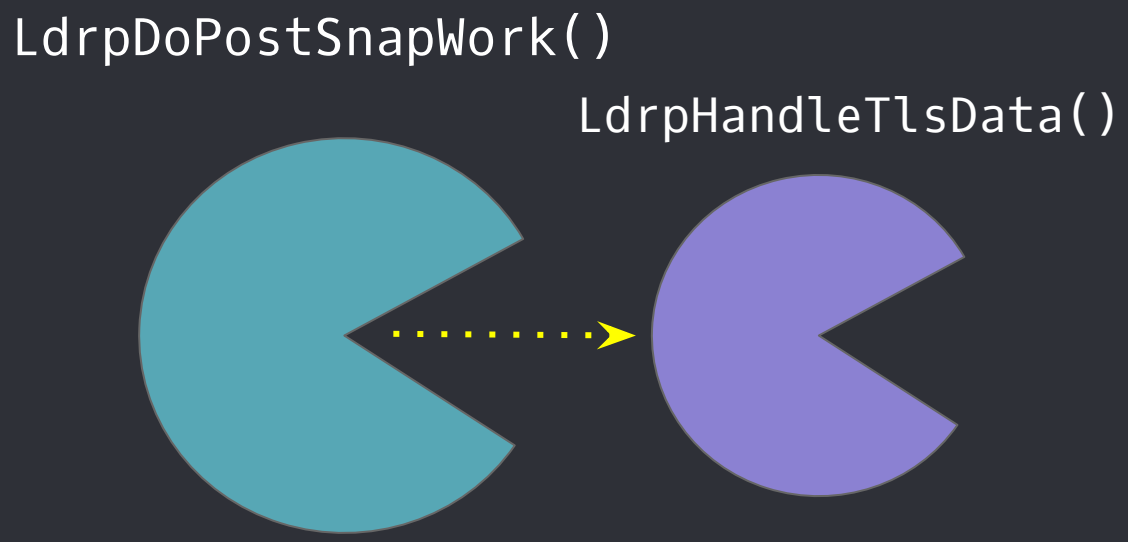
LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```

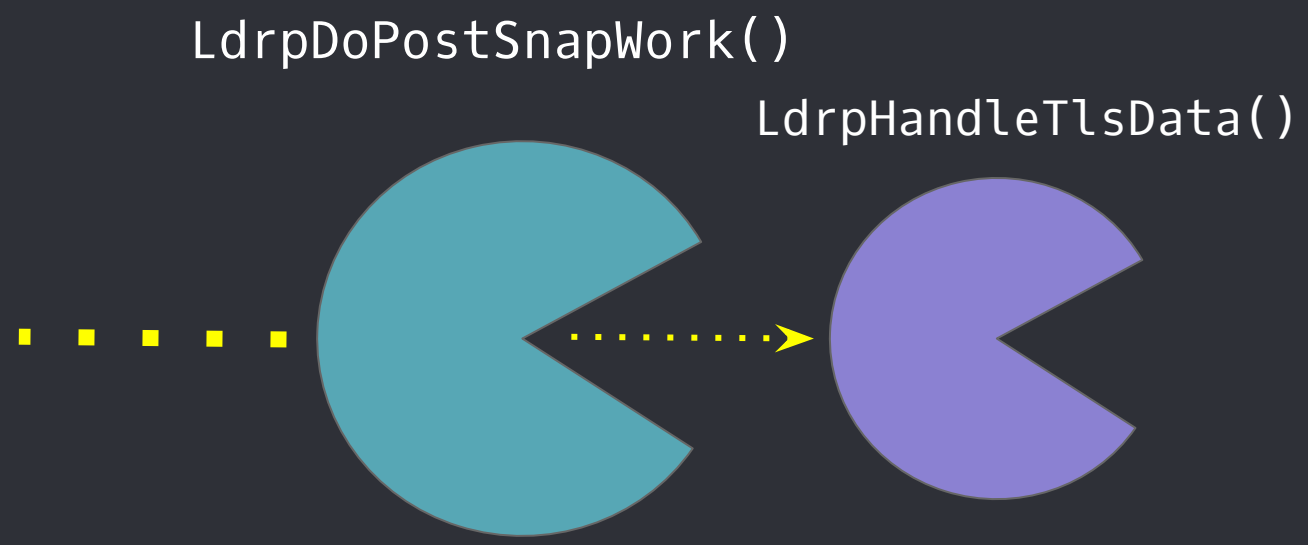
Locate Indirect Function Call

LdrpHandleTlsData()

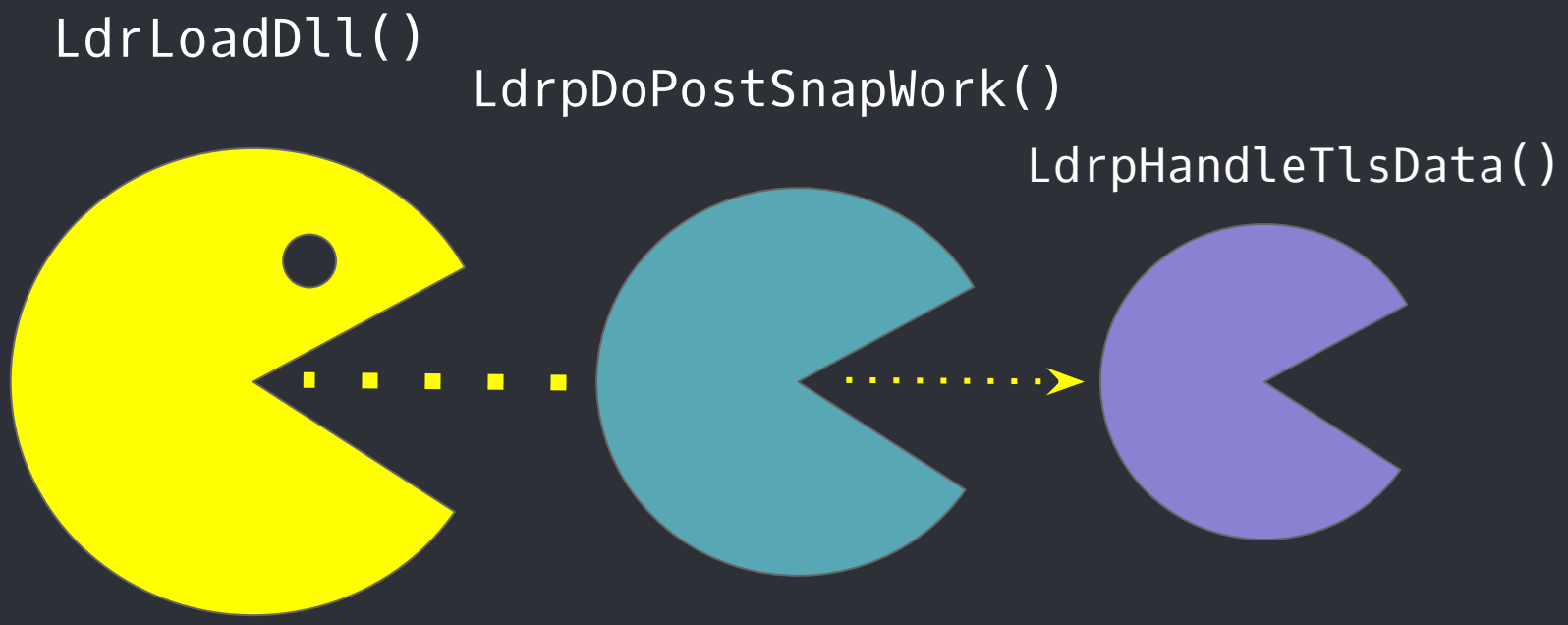
Locate Indirect Function Call



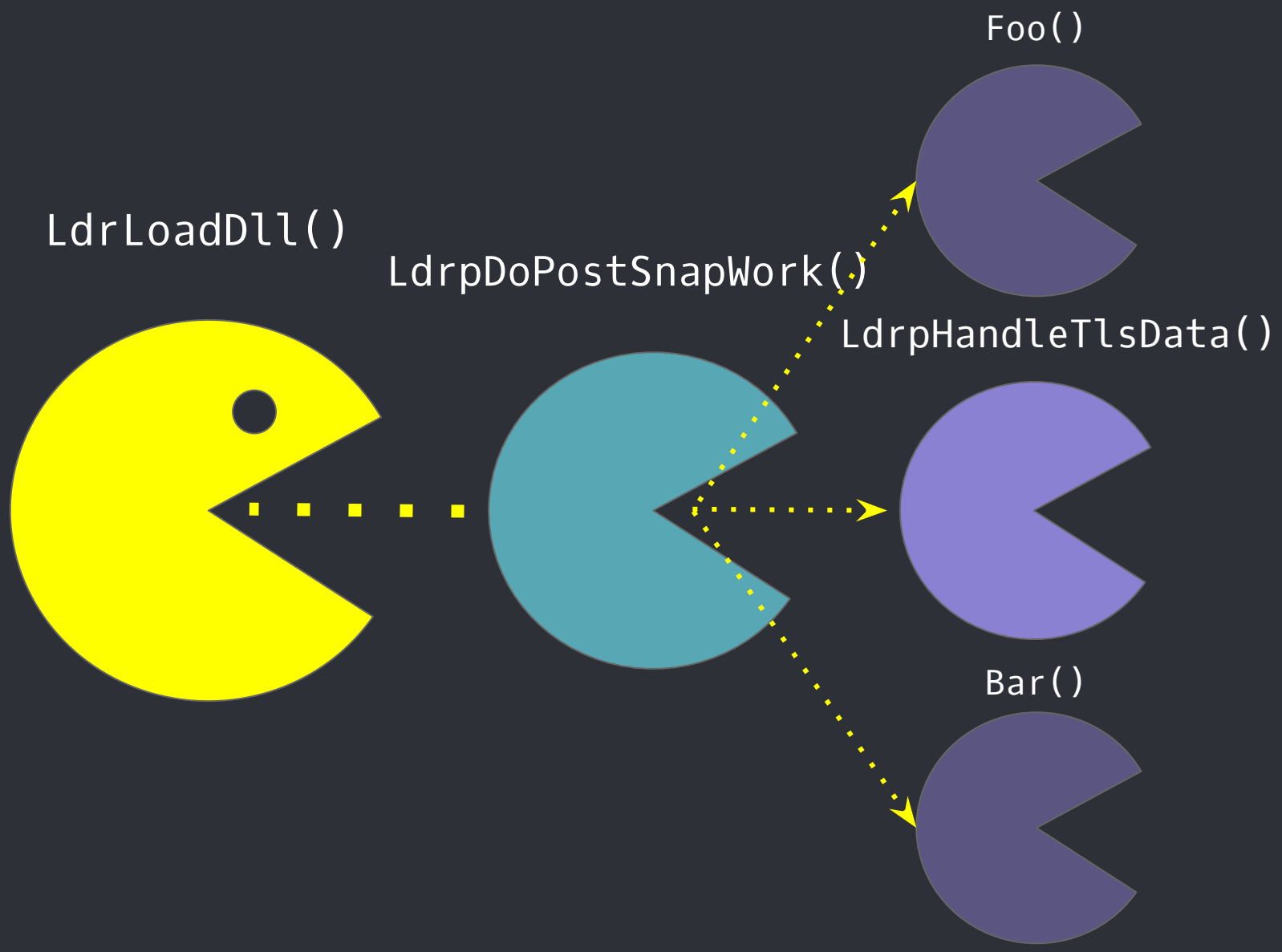
Locate Indirect Function Call



Locate Indirect Function Call



Locate Indirect Function Call



LdrpDoPostSnapWork

```
48 3B 87 90 00 00 00      cmp     rax, [rdi+90h]
0F 85 BE 09 07 00         jnz     loc_1800B857C
48 8B 4F 38               mov     rcx, [rdi+38h]
66 39 71 6E               cmp     [rcx+6Eh], si
75 0B                     jnz     short loc_180047BD3
E8 47 00 00 00            call    LdrpHandleTlsData
8B D8                     mov     ebx, eax
85 C0                     test    eax, eax
78 0B                     js      short loc_180047BDE
```

LdrpDoPostSnapWork

```
mov     [r11-28h], rax
call    ZwProtectVirtualMemory
...
call    LdrpHandleTlsData
mov     ebx, eax
...
call    LdrControlFlowGuardEnforcedWithExportSuppression
test    eax, eax
...
call    LdrpUnsuppressAddressTakenIat
mov     ebx, eax
...
call    LdrpLogDbgPrint
mov     eax, cs:LdrpDebugFlags
...
retn
```

LdrpDoPostSnapWork

```
mov     [r11-28h], rax
1 call  ZwProtectVirtualMemory
...
2 call  LdrpHandleTlsData
mov     ebx, eax
...
3 call  LdrControlFlowGuardEnforcedWithExportSuppression
test    eax, eax
...
4 call  LdrpUnsuppressAddressTakenIat
mov     ebx, eax
...
5 call  LdrpLogDbgPrint
mov     eax, cs:LdrpDebugFlags
...
retn
```

5

Function Calls

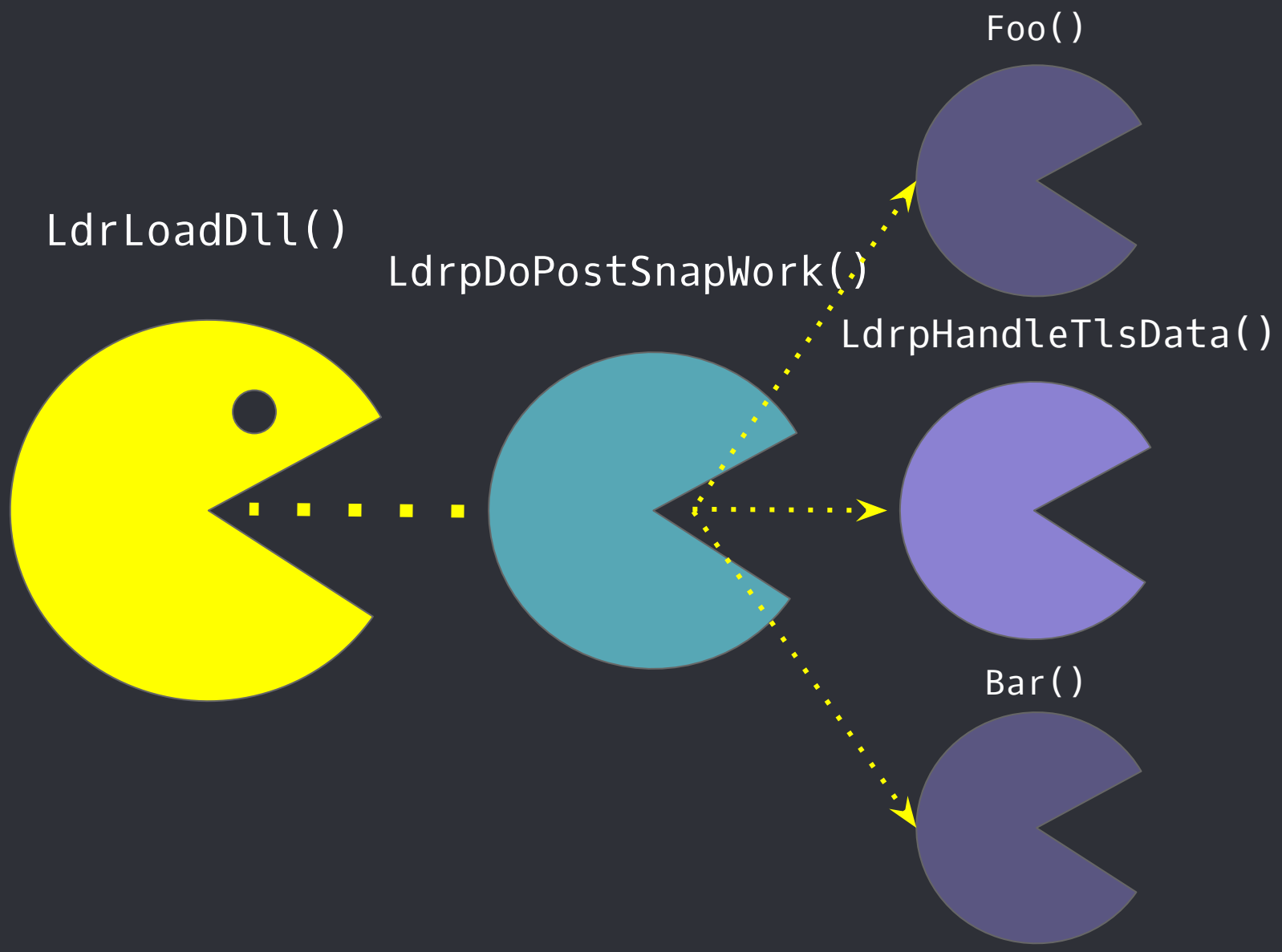
LdrpDoPostSnapWork

```
mov     [r11-28h], rax
1 call  ZwProtectVirtualMemory
...
2 call  LdrpHandleTlsData
mov     ebx, eax
...
3 call  LdrControlFlowGuardEnforcedWithExportSuppression
test    eax, eax
...
4 call  LdrpUnsuppressAddressTakenIat
mov     ebx, eax
...
5 call  LdrpLogDbgPrint
mov     eax, cs:LdrpDebugFlags
...
retn
```

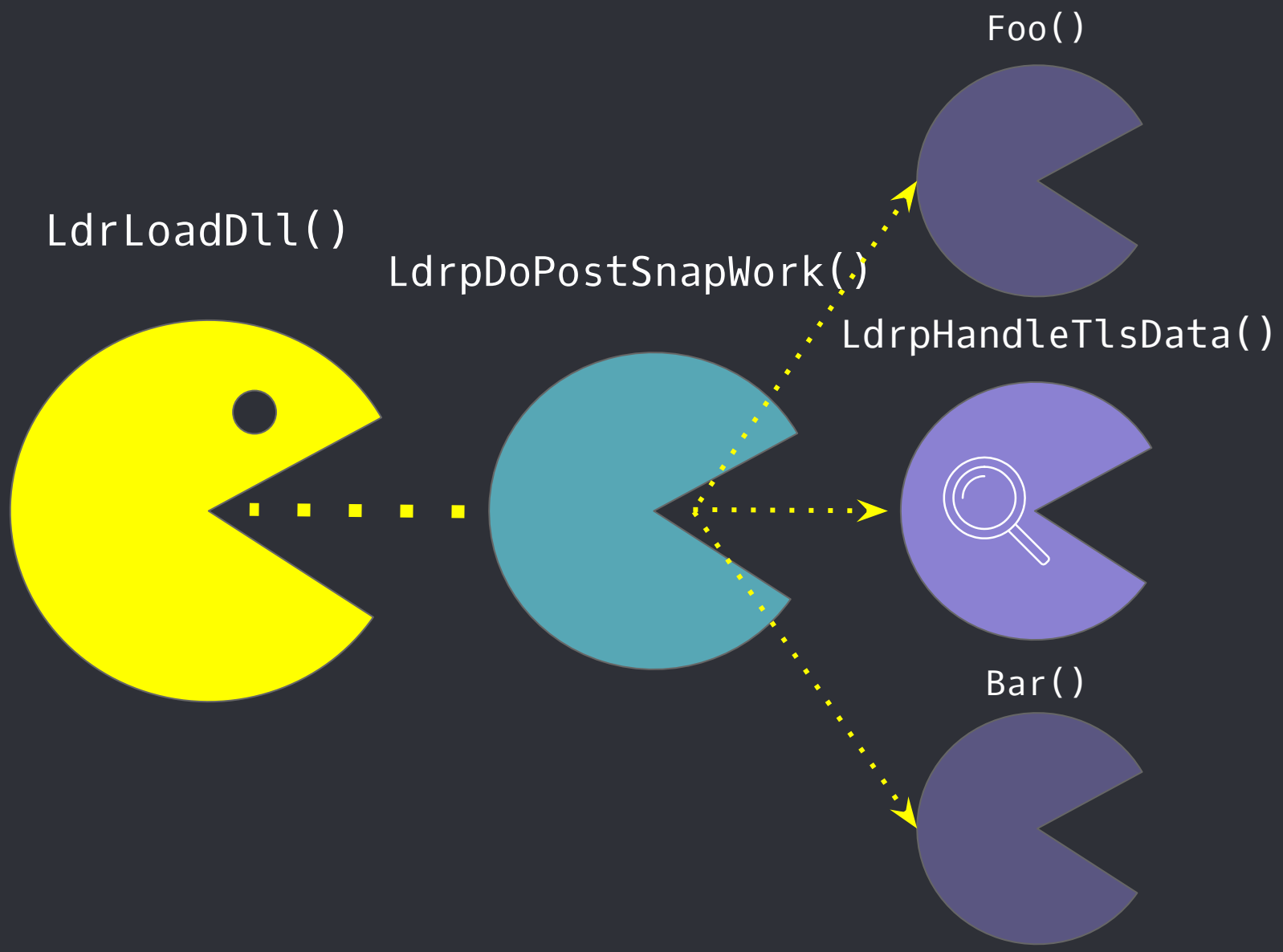
2/5

Function Calls

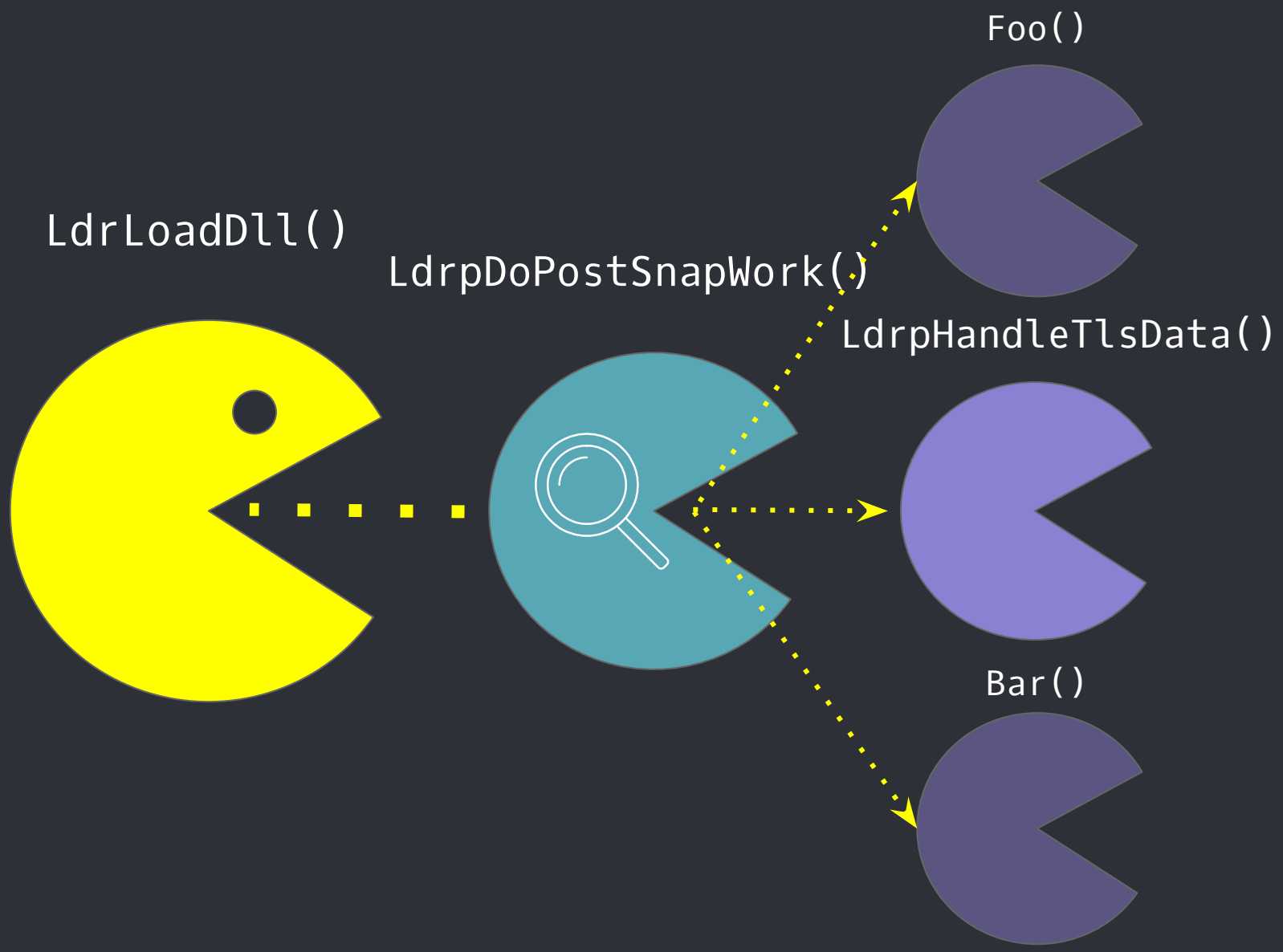
Locate Indirect Function Call



Locate Indirect Function Call



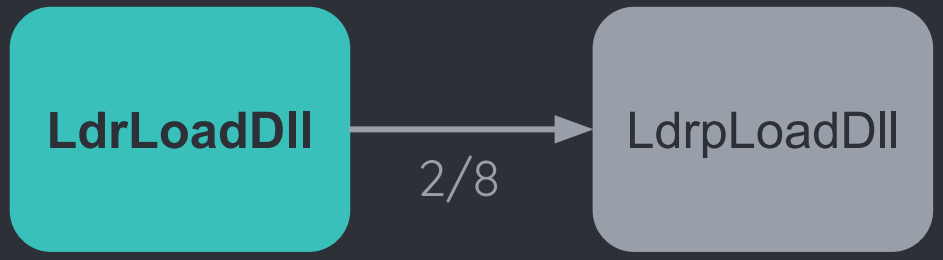
Locate Indirect Function Call



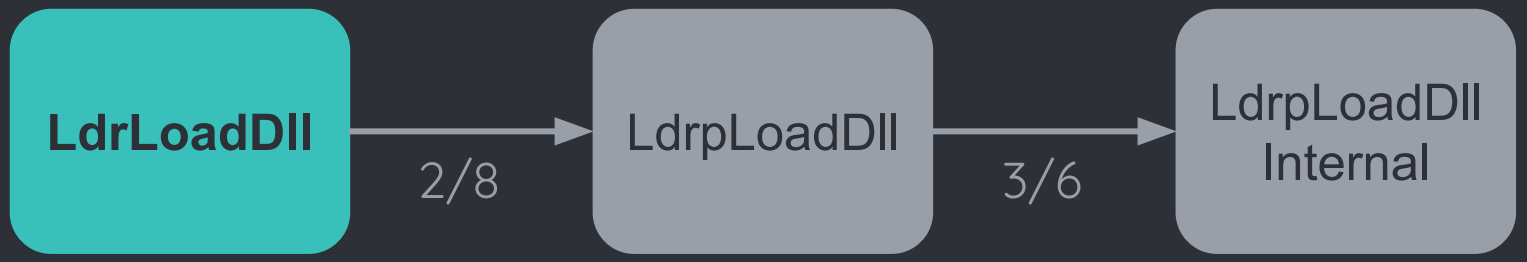
Calls Flow

LdrLoadDll

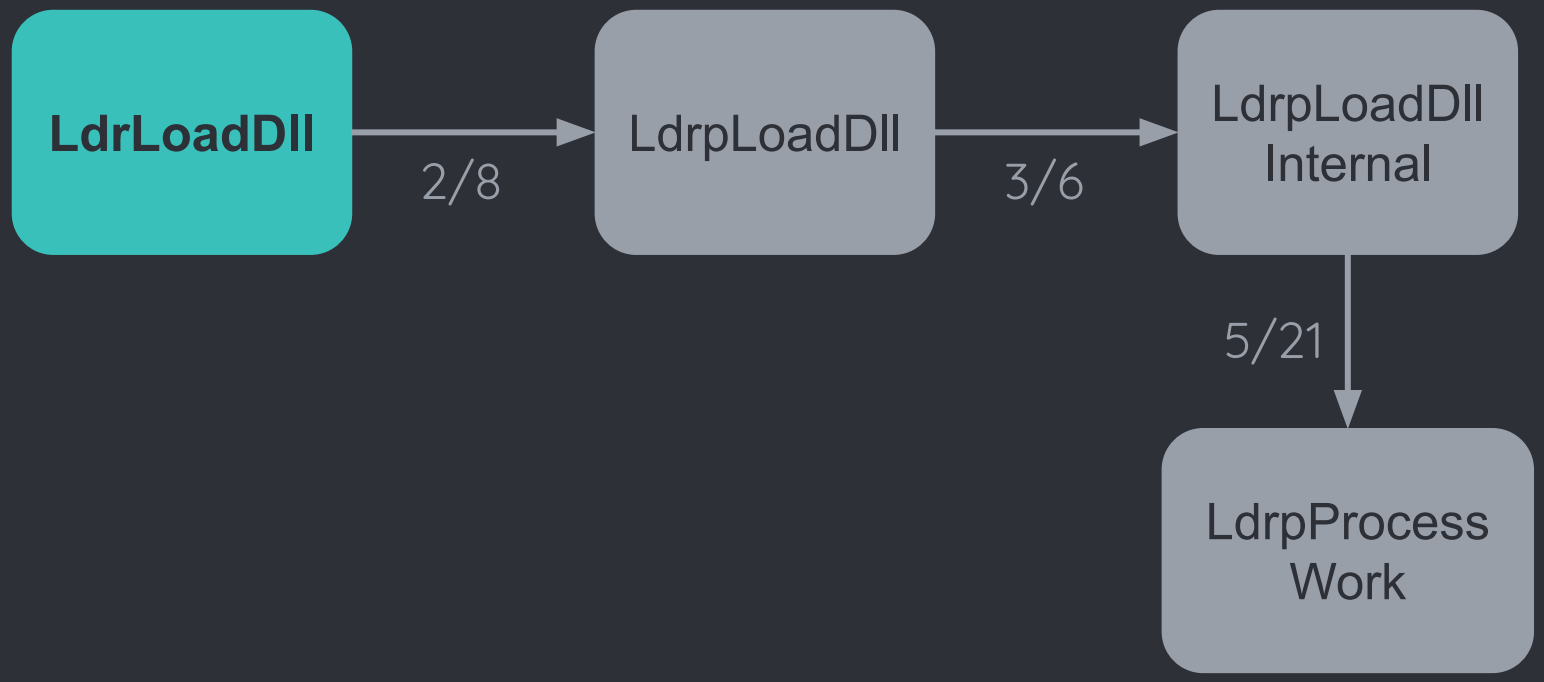
Calls Flow



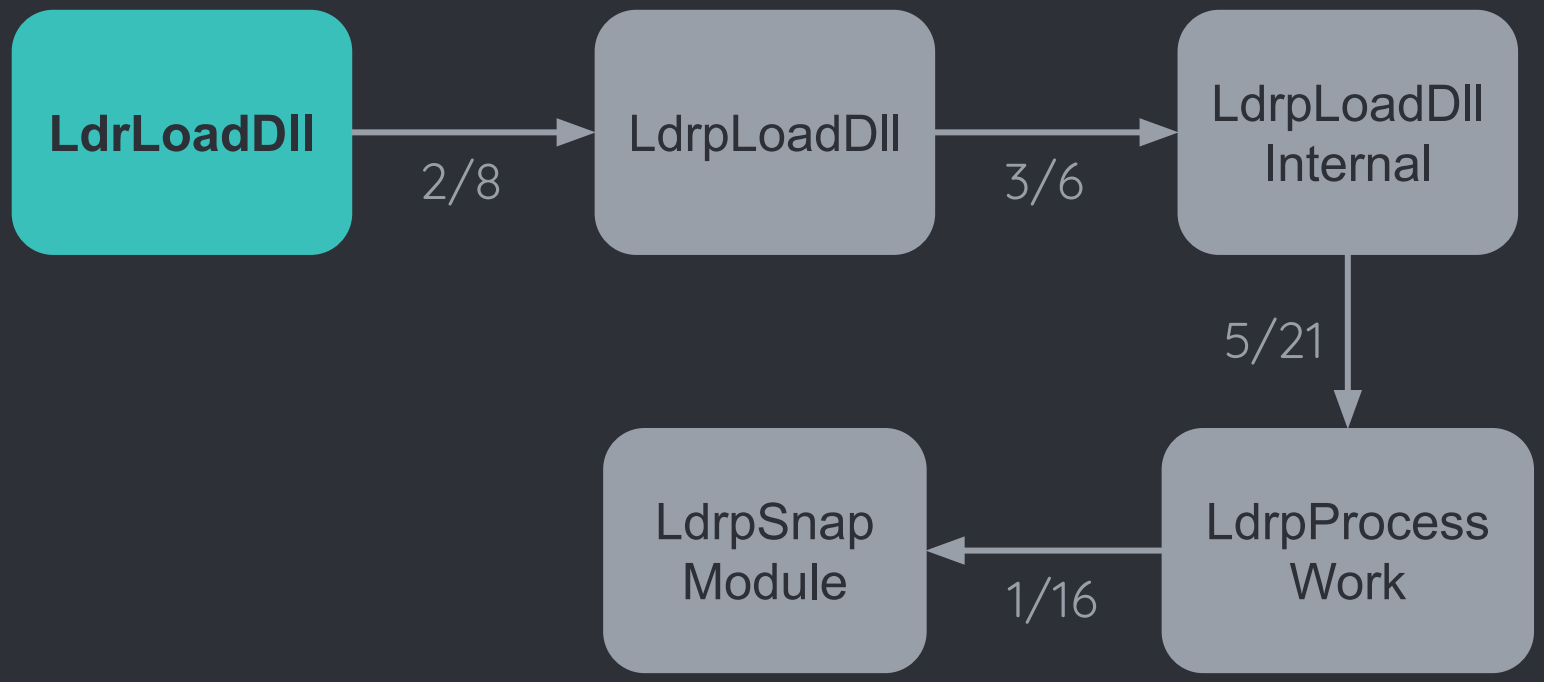
Calls Flow



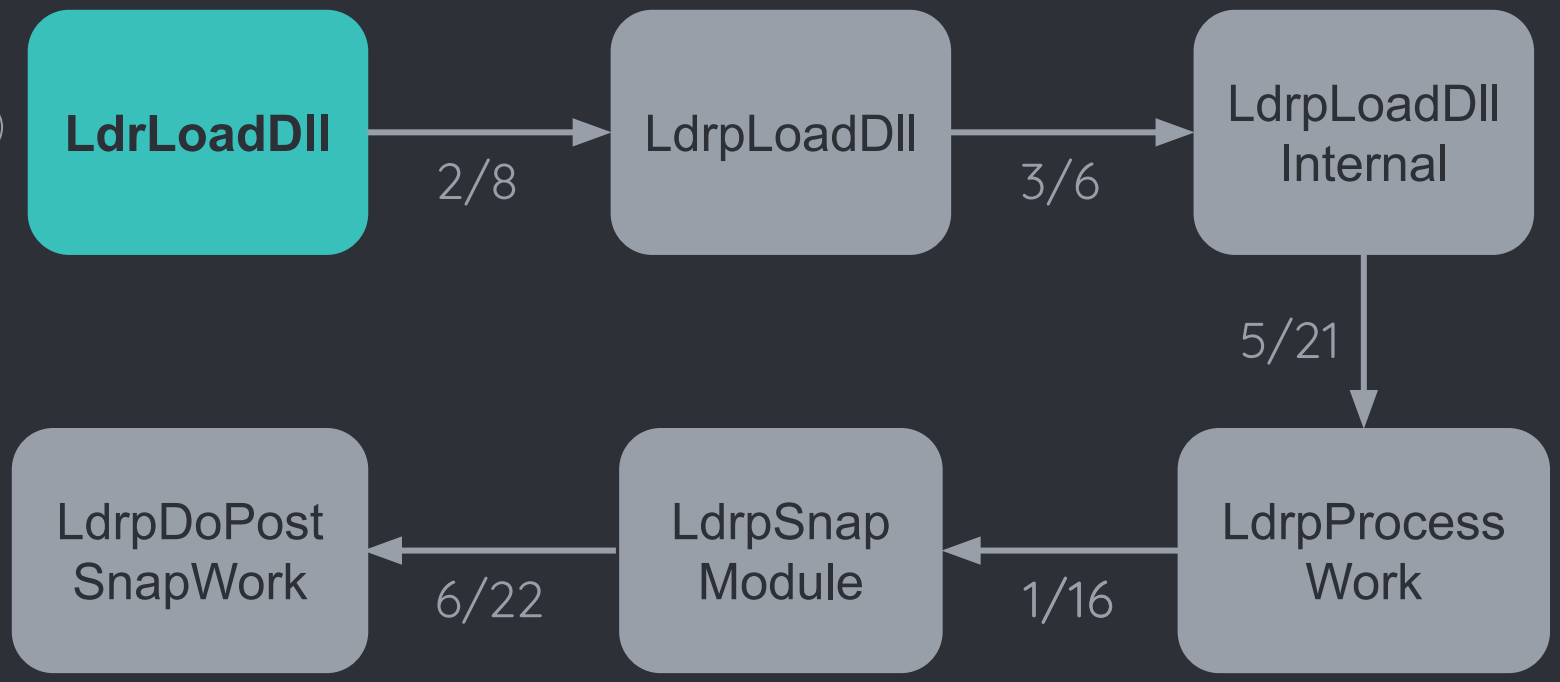
Calls Flow


LdrLoadDll
2/8
LdrpLoadDll
3/6
LdrpLoadDll Internal
5/21
LdrpProcess Work

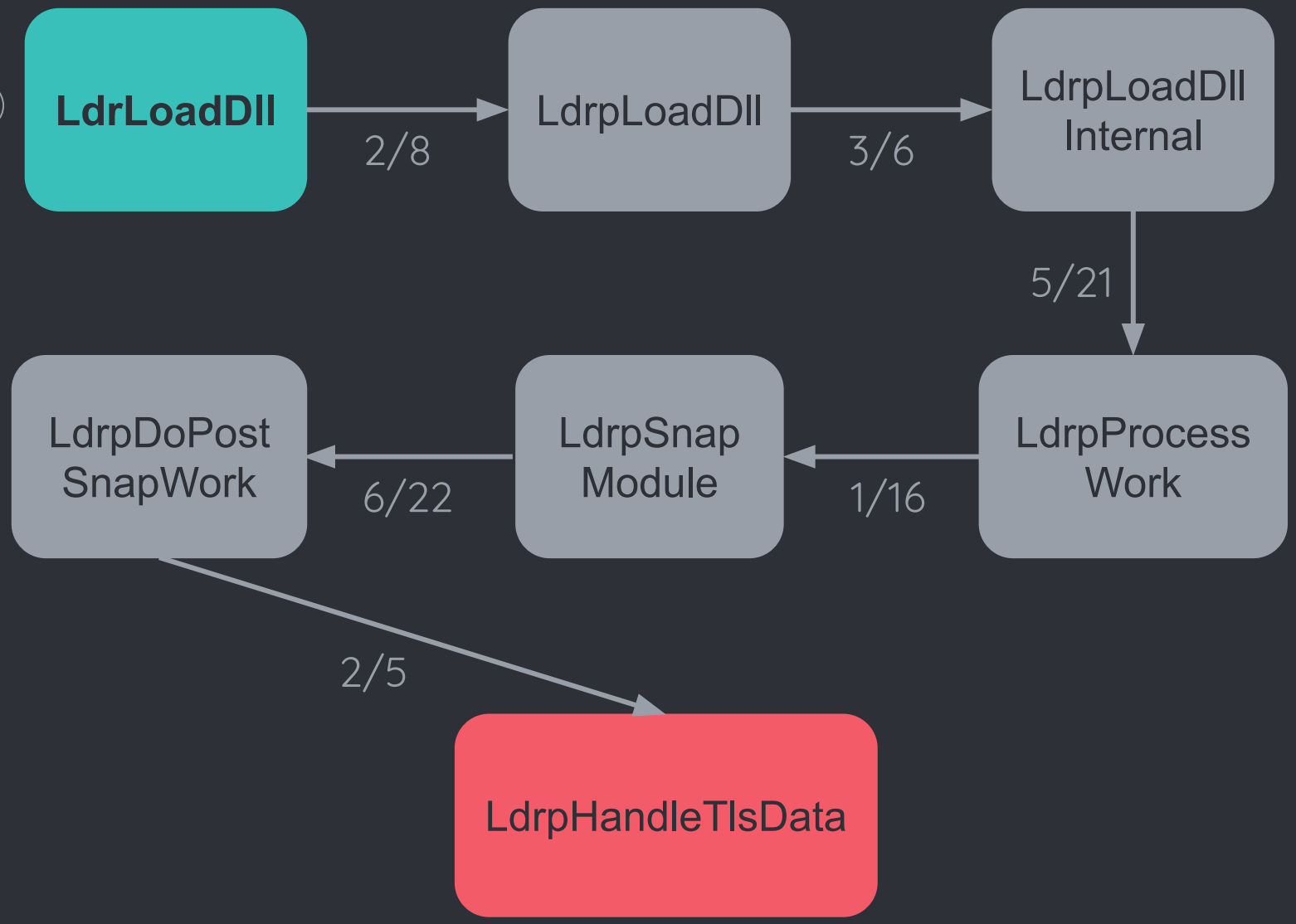
Calls Flow


LdrLoadDll
2/8
LdrpLoadDll
3/6
LdrpLoadDll Internal
5/21
LdrpProcess Work
1/16
LdrpSnap Module

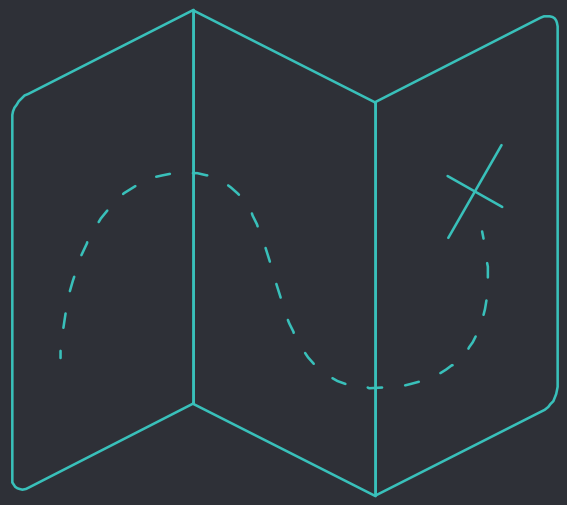
Calls Flow


LdrLoadDll
2/8
LdrpLoadDll
3/6
LdrpLoadDll Internal
5/21
LdrpProcess Work
1/16
LdrpSnap Module
6/22
LdrpDoPost SnapWork

Calls Flow


LdrLoadDll
2/8
LdrpLoadDll
3/6
LdrpLoadDll Internal
5/21
LdrpProcess Work
1/16
LdrpSnap Module
6/22
LdrpDoPost SnapWork
2/5
LdrpHandleTlsData

Once We've Found The Function's Address



We can call it from our code

Automation - IDAPython

```python
def get_calls_count(func_name, dest_func):
    calls = get_call_instructions(func_name)
    for i, call in enumerate(calls, 1):
        func_addr = get_operand_value(call.ea, 0)
        if func_addr == get_name_ea_simple(dest_func):
            return (i, len(calls))

    # return -1 to indicate the function is not found
    return (-1, len(calls))
```

Automation - IDAPython

```
def get_calls_count(func_name, dest_func):
    …

def main():
    win_version = 'Win10_1903_19H1_x64'
    # funcs_to_search struct: [(caller, callee)]
    funcs_to_search = [('LdrLoadDll', 'LdrpLoadDll'),
                       ('LdrpLoadDll', 'LdrpLoadDllInternal'),
                       ('LdrpLoadDllInternal', 'LdrpProcessWork'),
                       ('LdrpProcessWork', 'LdrpSnapModule'),
                       ('LdrpSnapModule', 'LdrpDoPostSnapWork'),
                       ('LdrpDoPostSnapWork', 'LdrpHandleTlsData')]

    for caller_func, callee_func in funcs_to_search:
        callee_num, total_calls = get_calls_count(caller_func, callee_func)
        print('Function %s was found in function %s after %d/%d calls' % \
            (callee_func, caller_func, callee_num, total_calls))

main()
```

Automation Output

```
---------------------------------------------------------------------------------------------
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
IDAPython 64-bit v7.4.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
---------------------------------------------------------------------------------------------
Function LdrpLoadDll was found in function LdrLoadDll after 2/8 calls
Function LdrpLoadDllInternal was found in function LdrpLoadDll after 3/6 calls
Function LdrpProcessWork was found in function LdrpLoadDllInternal after 5/21 calls
Function LdrpSnapModule was found in function LdrpProcessWork after 1/16 calls
Function LdrpDoPostSnapWork was found in function LdrpSnapModule after 6/22 calls
Function LdrpHandleTlsData was found in function LdrpDoPostSnapWork after 2/5 calls
```

Python

AU: idle Down Disk: 738GB

Run it on multiple Dlls

```
def main():
    …
    new_rows = []
    for caller_func, callee_func in funcs_to_search:
        callee_num, total_calls = get_calls_count(caller_func, callee_func)
        row_to_update = {}
        for row in rows:
            if row['Caller Function'] == caller_func and
                                    row['Callee Function'] == callee_func:
                row[win_version] = "' %d/%d '" % (callee_num, total_calls)
                row_to_update = row
        if row_to_update == {}:
                row_to_update = {'Caller Function': caller_func, 'Callee
                Function': callee_func, win_version: "' %d/%d '" %
                (callee_num, total_calls)}
        new_rows.append(row_to_update)
    with open(output_path, 'w', newline='') as f:
        writer = csv.DictWriter(f, fieldnames=header_list)
        writer.writeheader()
        writer.writerows(new_rows)
    …
```

Results

Caller Function	Callee Function	Win10_1507_	Win10_1511_	Win10_1607_	Win10_1803	Win10_1809_	Win10_1909_	Win10_1903_	Win10_2004_	Win10_20h2_
LdrLoadDll	LdrpLoadDll	'2/8'	'4/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'
LdrpLoadDll	LdrpLoadDllInternal	'3/5'	'3/5'	'3/5'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'
LdrpLoadDllInternal	LdrpProcessWork	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'
LdrpProcessWork	LdrpSnapModule	'1/15'	'8/15'	'1/15'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'
LdrpSnapModule	LdrpDoPostSnapWork	'6/19'	'21/24'	'9/29'	'6/20'	'10/32'	'6/22'	'6/22'	'6/22'	'6/22'
LdrpDoPostSnapWork	LdrpHandleTlsData	'2/2'	'2/2'	'2/2'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'

Results

Caller Function	Callee Function	Win10_1507_	Win10_1511_	Win10_1607_	Win10_1803_	Win10_1809_	Win10_1909_	Win10_1903_	Win10_2004_	Win10_20h2_
LdrLoadDll	LdrpLoadDll	'2/8'	'4/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'
LdrpLoadDll	LdrpLoadDllInternal	'3/5'	'3/5'	'3/5'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'
LdrpLoadDllInternal	LdrpProcessWork	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'
LdrpProcessWork	LdrpSnapModule	'1/15'	'8/15'	'1/15'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'
LdrpSnapModule	LdrpDoPostSnapWork	'6/19'	'21/24'	'9/29'	'6/20'	'10/32'	'6/22'	'6/22'	'6/22'	'6/22'
LdrpDoPostSnapWork	LdrpHandleTlsData	'2/2'	'2/2'	'2/2'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'

Results

Caller Function	Callee Function	Win10_1507_	Win10_1511_	Win10_1607_	Win10_1803_	Win10_1809_	Win10_1909_	Win10_1903_	Win10_2004_	Win10_20h2_
LdrLoadDll	LdrpLoadDll	'2/8'	'4/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'
LdrpLoadDll	LdrpLoadDllInternal	'3/5'	'3/5'	'3/5'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'
LdrpLoadDllInternal	LdrpProcessWork	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'
LdrpProcessWork	LdrpSnapModule	'1/15'	'8/15'	'1/15'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'
LdrpSnapModule	LdrpDoPostSnapWork	'6/19'	'21/24'	'9/29'	'6/20'	'10/32'	'6/22'	'6/22'	'6/22'	'6/22'
LdrpDoPostSnapWork	LdrpHandleTlsData	'2/2'	'2/2'	'2/2'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'

Results

Caller Function	Callee Function	Win10_1507_	Win10_1511_	Win10_1607_	Win10_1803_	Win10_1809_	Win10_1903_	Win10_1909_	Win10_2004_	Win10_20h2_
LdrLoadDll	LdrpLoadDll	'2/8'	'4/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'
LdrpLoadDll	LdrpLoadDllInternal	'3/5'	'3/5'	'3/5'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'
LdrpLoadDllInternal	LdrpProcessWork	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'
LdrpProcessWork	LdrpSnapModule	'1/15'	'8/15'	'1/15'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'
LdrpSnapModule	LdrpDoPostSnapWork	'6/19'	'21/24'	'9/29'	'6/20'	'10/32'	'6/22'	'6/22'	'6/22'	'6/22'
LdrpDoPostSnapWork	LdrpHandleTlsData	'2/2'	'2/2'	'2/2'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'

Results

Caller Function	Callee Function	Win10_1507_	Win10_1511_	Win10_1607_	Win10_1803_	Win10_1809_	Win10_1909_	Win10_1903_	Win10_2004_	Win10_20h2_
LdrLoadDll	LdrpLoadDll	'2/8'	'4/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'
LdrpLoadDll	LdrpLoadDllInternal	'3/5'	'3/5'	'3/5'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'
LdrpLoadDllInternal	LdrpProcessWork	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'
LdrpProcessWork	LdrpSnapModule	'1/15'	'8/15'	'1/15'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'
LdrpSnapModule	LdrpDoPostSnapWork	'6/19'	'21/24'	'9/29'	'6/20'	'10/32'	'6/22'	'6/22'	'6/22'	'6/22'
LdrpDoPostSnapWork	LdrpHandleTlsData	'2/2'	'2/2'	'2/2'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'

Results

Caller Function	Callee Function	Win10_1507_	Win10_1511_	Win10_1607_	Win10_1803_	Win10_1809_	Win10_1909_	Win10_1903_	Win10_2004_	Win10_20h2_
LdrLoadDll	LdrpLoadDll	'2/8'	'4/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'
LdrpLoadDll	LdrpLoadDllInternal	'3/5'	'3/5'	'3/5'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'
LdrpLoadDllInternal	LdrpProcessWork	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'
LdrpProcessWork	LdrpSnapModule	'1/15'	'8/15'	'1/15'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'
LdrpSnapModule	LdrpDoPostSnapWork	'6/19'	'21/24'	'9/29'	'6/20'	'10/32'	'6/22'	'6/22'	'6/22'	'6/22'
LdrpDoPostSnapWork	LdrpHandleTlsData	'2/2'	'2/2'	'2/2'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'

Results

Caller Function	Callee Function	Win10_1507_	Win10_1511_	Win10_1607_	Win10_1803_	Win10_1809_	Win10_1909_	Win10_1903_	Win10_2004_	Win10_20h2_
LdrLoadDll	LdrpLoadDll	'2/8'	'4/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'	'2/8'
LdrpLoadDll	LdrpLoadDllInternal	'3/5'	'3/5'	'3/5'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'	'3/6'
LdrpLoadDllInternal	LdrpProcessWork	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'	'5/21'
LdrpProcessWork	LdrpSnapModule	'1/15'	'8/15'	'1/15'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'	'1/16'
LdrpSnapModule	LdrpDoPostSnapWork	'6/19'	'21/24'	'9/29'	'6/20'	'10/32'	'6/22'	'6/22'	'6/22'	'6/22'
LdrpDoPostSnapWork	LdrpHandleTlsData	'2/2'	'2/2'	'2/2'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'	'2/5'

Results

Caller Function	Callee Function	Win10_1507_
LdrLoadDll	LdrpLoadDll	' 2/8 '
LdrpLoadDll	LdrpLoadDllInternal	' 3/5 '
LdrpLoadDllInternal	LdrpProcessWork	' 5/21 '
LdrpProcessWork	LdrpSnapModule	' 1/15 '
LdrpSnapModule	LdrpDoPostSnapWork	' 6/19 '
LdrpDoPostSnapWork	LdrpHandleTlsData	' 2/2 '

Results

Caller Function	Callee Function	Win10_1507_
LdrLoadDll	LdrpLoadDll	' 2/8 '
LdrpLoadDll	LdrpLoadDllInternal	' 3/5 '
LdrpLoadDllInternal	LdrpProcessWork	' 5/21 '
LdrpProcessWork	LdrpSnapModule	' 1/15 '
LdrpSnapModule	LdrpDoPostSnapWork	' 6/19 '
LdrpDoPostSnapWork	LdrpHandleTlsData	' 2/2 '

Results

Caller Function	Callee Function	Win10_1507_
LdrLoadDll	LdrpLoadDll	' 2/8 '
LdrpLoadDll	LdrpLoadDllInternal	' 3/5 '
LdrpLoadDllInternal	LdrpProcessWork	' 5/21 '
LdrpProcessWork	LdrpSnapModule	' 1/15 '
LdrpSnapModule	LdrpDoPostSnapWork	' 6/19 '
LdrpDoPostSnapWork	LdrpHandleTlsData	' 2/2 '

Pros

+ Relies only on logic.

  Nothing has to be unique

Pros

+ Relies only on logic.

  Nothing has to be unique

Cons

− The farther we get away from the exported function, the less stable this method is,
  And it relies on other functions changes

Stability

On Different OS Versions

Stability Between Different OS Versions - LdrpHandleTlsData

Unique Byte Sequence

And calculate the offset to start address.

Direct Function Call

Locate the unique byte sequence of the function call.

Indirect Function Call

Find the exported function and dive into the inner calls until we find the call to LdrpHandleTlsData.

Stability Between Different OS Versions - LdrpHandleTlsData

Unique Byte Sequence

And calculate the offset to start address.

Direct Function Call

Locate the unique byte sequence of the function call.

Indirect Function Call

Find the exported function and dive into the inner calls until we find the call to LdrpHandleTlsData.

Stability Between Different OS Versions - LdrpHandleTlsData

Unique Byte Sequence

And calculate the offset to start address.

Direct Function Call

Locate the unique byte sequence of the function call.

Indirect Function Call

Find the exported function and dive into the inner calls until we find the call to LdrpHandleTlsData.

Note That -

- There is no right or wrong answers here
- Every unexported function might fit a different search method
- Use the automation in order to decide which one is better for your target function.


This is nice!
How can I try this at home?

Resources

- PE format & Windows PE loader - https://guidedhacking.com/threads/pe-file-format-windows-pe-loader-tutorial.14310/
- Reflective Loader - https://www.ired.team/offensive-security/code-injection-process-injection/reflective-dll-injection
- Github -
  - ReflectiveDLLInjection by Stephen Fewer - https://github.com/stephenfewer/ReflectiveDLLInjection
  - Blackbone - https://github.com/DarthTon/Blackbone
- https://github.com/oryandp/LocateUnexportedFunctions

Thanks!

ANY QUESTIONS?

You can find me at

@OryanDP

Git repository:

https://github.com/oryandp/LocateUnexportedFunctions

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Icons by Icons8