Seance: Divination of Tool-Breaking Changes in Forensically Important Binaries

By:
Ryan Maggio (Louisiana State University), Andrew Case (Volatility Foundation), Aisha Ali-Gombe (Towson University), and Golden G. Richard III (Louisiana State University)

From the proceedings of

The Digital Forensic Research Conference

DFRWS USA 2021

July 12-15, 2021

SEANCE: FORENSIC DIVINATION

Ryan Maggio

Overview

- Motivation
  - Memory forensics tools require frequent checks for compatibility with new software versions
  - Data structure reconstruction is a historically tricky problem
  - Existing work is brittle or manual
- Internals
  - Existing tools used to build Seance
  - Two Components, API and controlling code
  - Example workflow for analysis and database construction
- Testing
  - Tested on Objective-C and Windows networking stack binaries
  - Objective-C results
  - Windows Networking Stack results
- Future Work

Motivation

- Verifying compatibility of a forensic tool and a target binary is a time consuming, error prone task
  - Largely based on data structure reconstruction
  - Existing work is brittle or incomplete
- Compatibility-breaking changes have caused real world issues, e.g. issues raised with Volatility
- Our previous projects on malware analysis used emulation, we wanted to see if that technique works here, too
  - In that space, encountered issues where certain behavior would be missed by emulation
  - Instead of just emulating, try symbolic execution for better capturing behavior

Seance

- Built on top of angr
- Explores “true behavior” of code under analysis
- Get more detailed execution data
  - Produce a CFG
  - Record all memory accesses
    - Get concrete results
- Can answer questions about tool compatibility
  - Potentially useful in other domains, too

The glowing bit is code

Internals - API

- Much of the functionality implemented in an API
  - Does not integrate with Volatility (yet)
- Four angr callbacks, mostly helper functions
  - Assumes an instantiated angr project
- Callbacks for memory reads, writes and register reads, writes
  - Check if execution is ongoing
  - Check that target can be concretized
  - Record access length, target, conditions, ordered list of basic blocks
- Handle stashes
  - Associate memory accesses with correct end states
  - Concretize values
  - Generate permit-list for CFG
- Generate CFG
- Print CFG

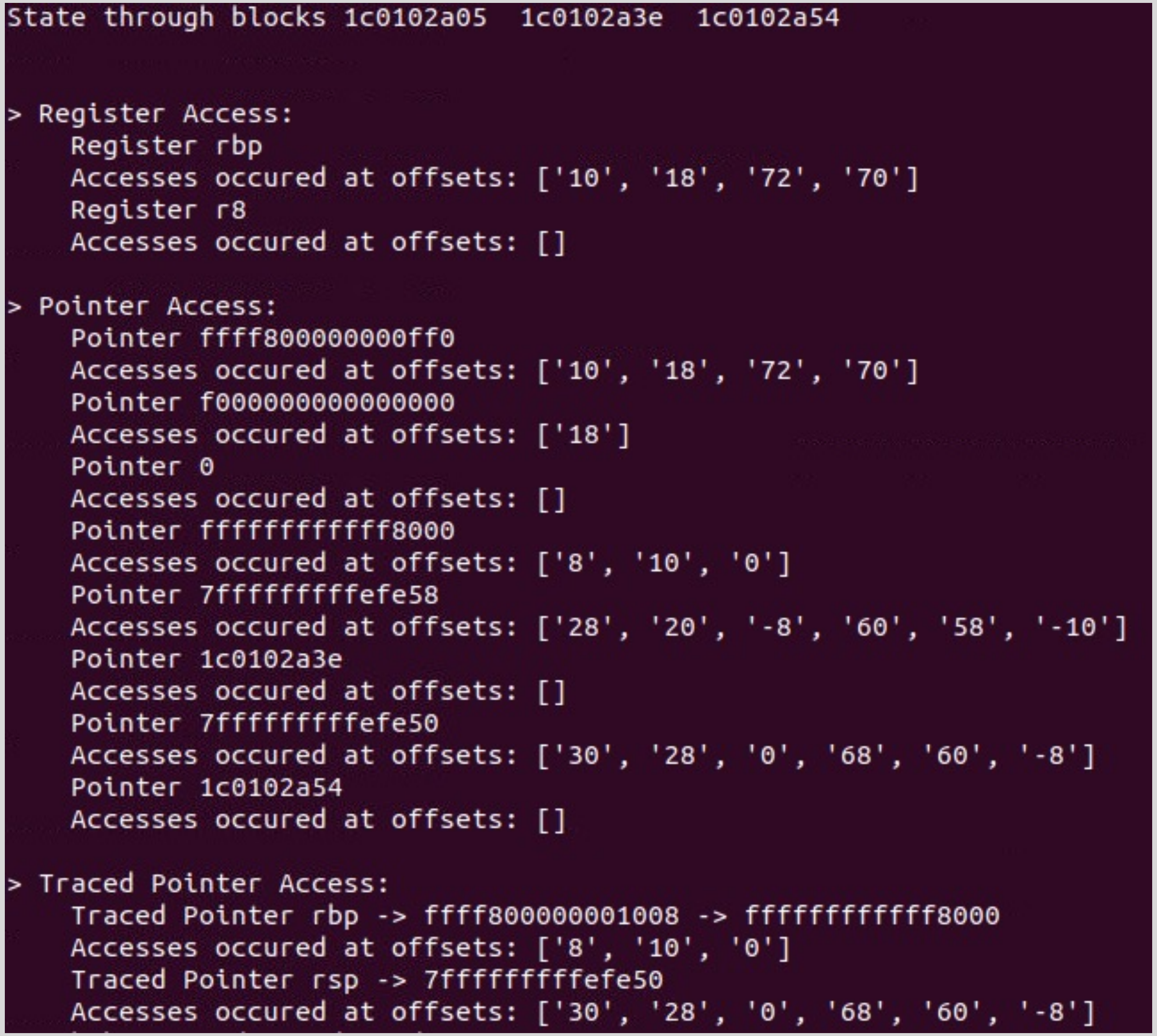
Internals - Controlling Code

- Input
  - Binary
  - Symbol
    - Well, maybe a bit more
  - Database
- Output
  - CFG of target symbol
  - Detailed execution data
  - Offsets referenced
    - From memory addresses
    - From memory addresses accessed via pointer
- Post processing
  - Comparison against database
  - Updated database

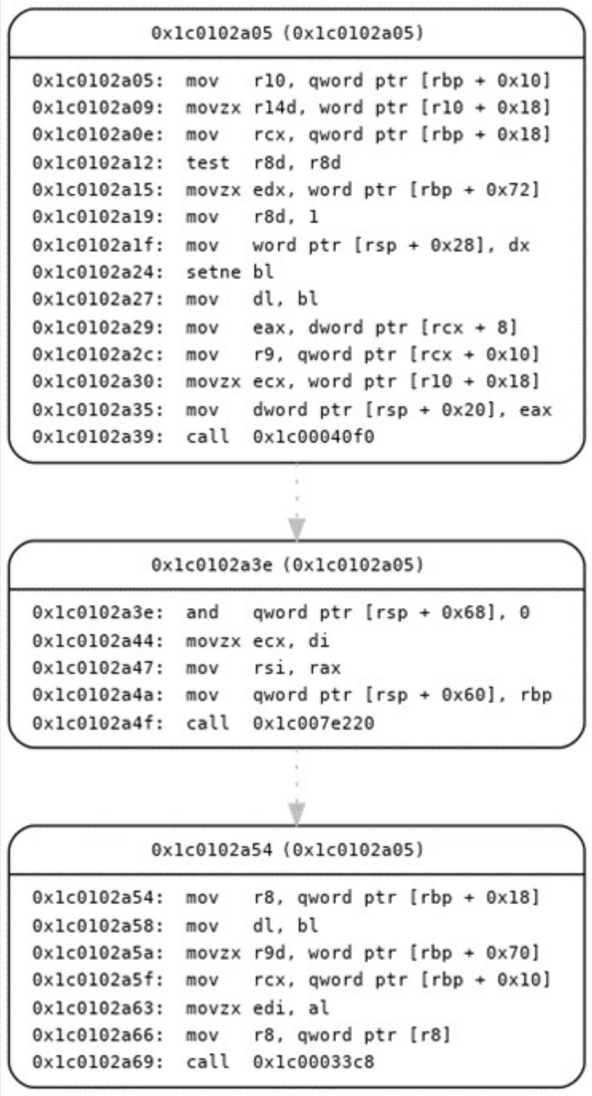
Test Data

- Two sets of test data
- Objective-C
  - Open source
  - Many important algorithms and data structures
    - Abused by malware
  - Research efforts are dated
- Windows networking stack
  - Closed source
    - Debug information not published
  - Network activities are often central to investigations
  - Updated often, causing issues

Example from TcpConnectTimeout



```
State through blocks 1c0102a05  1c0102a3e  1c0102a54


> Register Access:
    Register rbp
    Accesses occured at offsets: ['10', '18', '72', '70']
    Register r8
    Accesses occured at offsets: []

> Pointer Access:
    Pointer ffff800000000ff0
    Accesses occured at offsets: ['10', '18', '72', '70']
    Pointer f000000000000000
    Accesses occured at offsets: ['18']
    Pointer 0
    Accesses occured at offsets: []
    Pointer fffffffffffff8000
    Accesses occured at offsets: ['8', '10', '0']
    Pointer 7ffffffffefe58
    Accesses occured at offsets: ['28', '20', '-8', '60', '58', '-10']
    Pointer 1c0102a3e
    Accesses occured at offsets: []
    Pointer 7ffffffffefe50
    Accesses occured at offsets: ['30', '28', '0', '68', '60', '-8']
    Pointer 1c0102a54
    Accesses occured at offsets: []

> Traced Pointer Access:
    Traced Pointer rbp -> ffff800000001008 -> fffffffffffff8000
    Accesses occured at offsets: ['8', '10', '0']
    Traced Pointer rsp -> 7ffffffffefe50
    Accesses occured at offsets: ['30', '28', '0', '68', '60', '-8']
```



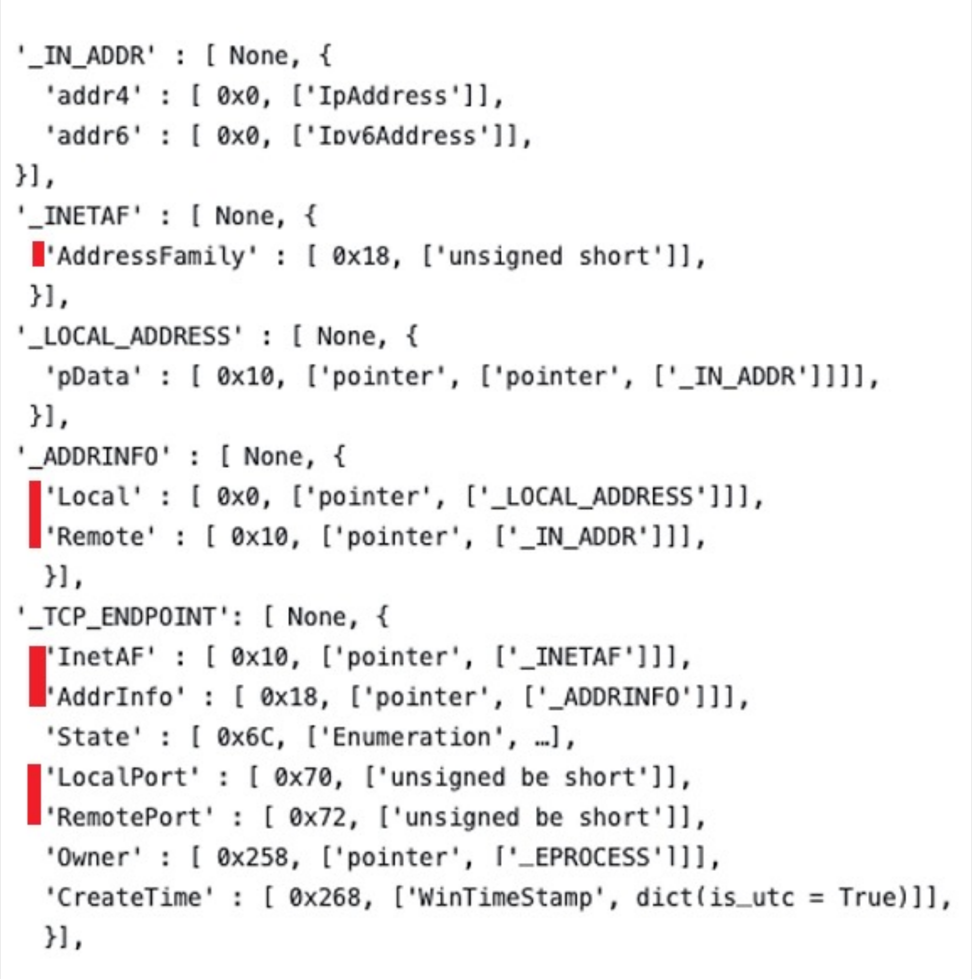
```
0x1c0102a05 (0x1c0102a05)

0x1c0102a05:  mov   r10, qword ptr [rbp + 0x10]
0x1c0102a09:  movzx r14d, word ptr [r10 + 0x18]
0x1c0102a0e:  mov   rcx, qword ptr [rbp + 0x18]
0x1c0102a12:  test  r8d, r8d
0x1c0102a15:  movzx edx, word ptr [rbp + 0x72]
0x1c0102a19:  mov   r8d, 1
0x1c0102a1f:  mov   word ptr [rsp + 0x28], dx
0x1c0102a24:  setne bl
0x1c0102a27:  mov   dl, bl
0x1c0102a29:  mov   eax, dword ptr [rcx + 8]
0x1c0102a2c:  mov   r9, qword ptr [rcx + 0x10]
0x1c0102a30:  movzx ecx, word ptr [r10 + 0x18]
0x1c0102a35:  mov   dword ptr [rsp + 0x20], eax
0x1c0102a39:  call  0x1c00040f0
```

```
0x1c0102a3e (0x1c0102a05)

0x1c0102a3e:  and   qword ptr [rsp + 0x68], 0
0x1c0102a44:  movzx ecx, di
0x1c0102a47:  mov   rsi, rax
0x1c0102a4a:  mov   qword ptr [rsp + 0x60], rbp
0x1c0102a4f:  call  0x1c007e220
```

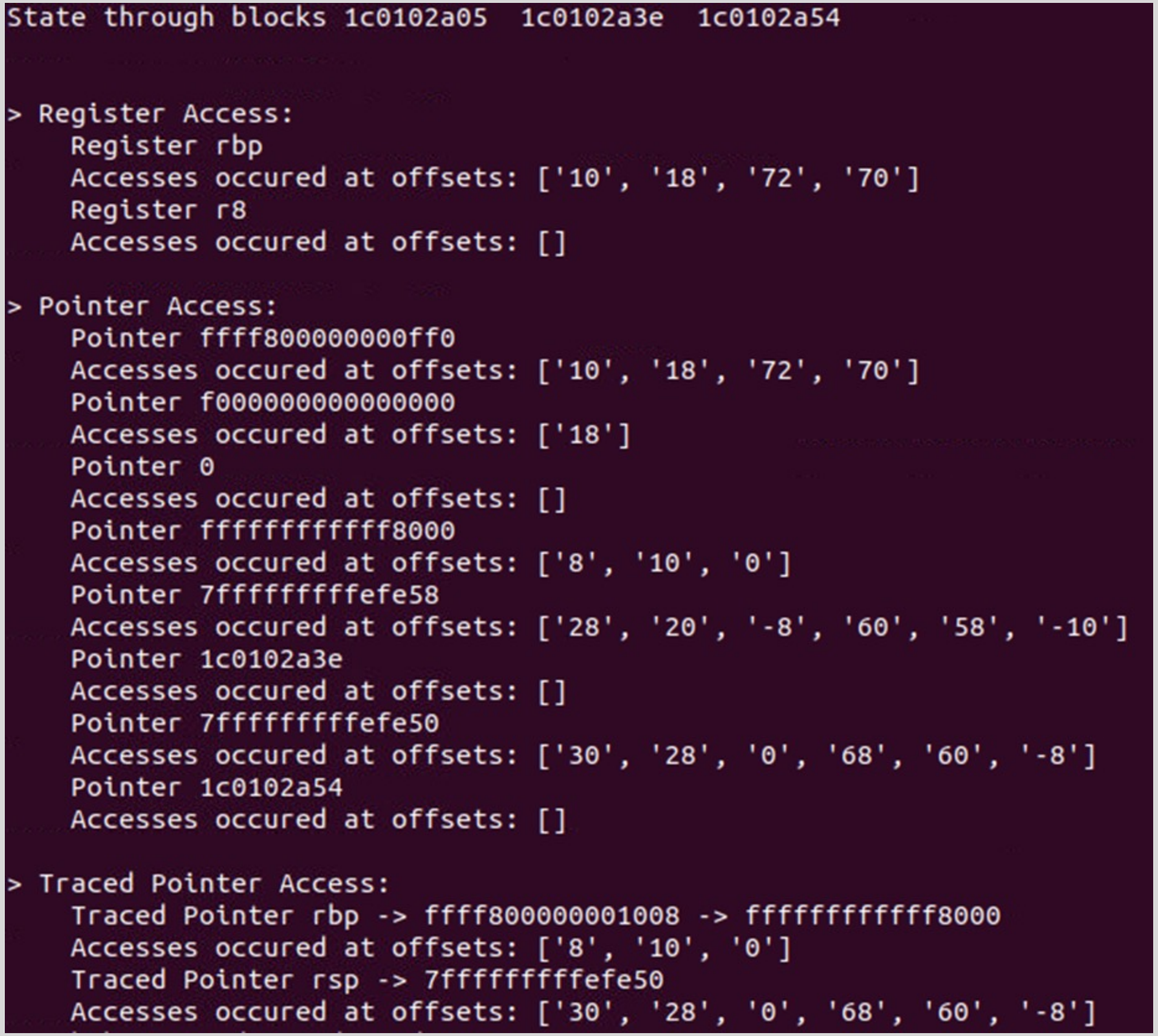
```
0x1c0102a54 (0x1c0102a05)

0x1c0102a54:  mov   r8, qword ptr [rbp + 0x18]
0x1c0102a58:  mov   dl, bl
0x1c0102a5a:  movzx r9d, word ptr [rbp + 0x70]
0x1c0102a5f:  mov   rcx, qword ptr [rbp + 0x10]
0x1c0102a63:  movzx edi, al
0x1c0102a66:  mov   r8, qword ptr [r8]
0x1c0102a69:  call  0x1c00033c8
```

Was that actually helpful?

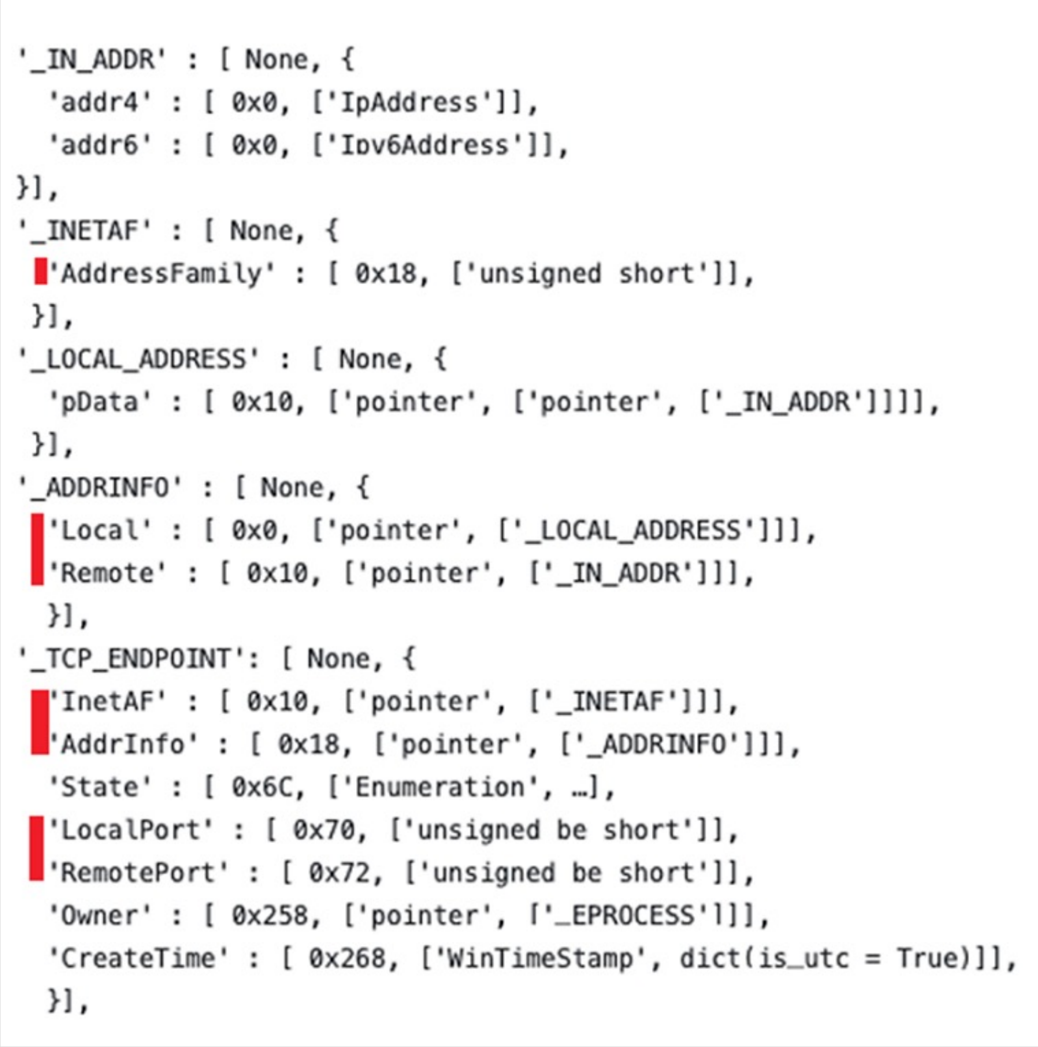
- Targets the TCP_ENDPOINT data structure
  - Complicated, nested structure
  - Despite this, offsets correctly recognized



```
'_IN_ADDR' : [ None, {
  'addr4' : [ 0x0, ['IpAddress']],
  'addr6' : [ 0x0, ['Ipv6Address']],
}],
'_INETAF' : [ None, {
 'AddressFamily' : [ 0x18, ['unsigned short']],
 }],
'_LOCAL_ADDRESS' : [ None, {
  'pData' : [ 0x10, ['pointer', ['pointer', ['_IN_ADDR']]]],
 }],
'_ADDRINFO' : [ None, {
 'Local' : [ 0x0, ['pointer', ['_LOCAL_ADDRESS']]],
 'Remote' : [ 0x10, ['pointer', ['_IN_ADDR']]],
  }],
'_TCP_ENDPOINT': [ None, {
 'InetAF' : [ 0x10, ['pointer', ['_INETAF']]],
 'AddrInfo' : [ 0x18, ['pointer', ['_ADDRINFO']]],
  'State' : [ 0x6C, ['Enumeration', …],
 'LocalPort' : [ 0x70, ['unsigned be short']],
 'RemotePort' : [ 0x72, ['unsigned be short']],
  'Owner' : [ 0x258, ['pointer', ['_EPROCESS']]],
  'CreateTime' : [ 0x268, ['WinTimeStamp', dict(is_utc = True)]],
  }],
```

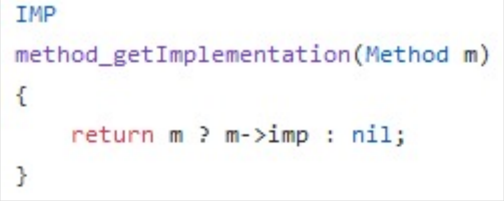


```
State through blocks 1c0102a05  1c0102a3e  1c0102a54

> Register Access:
    Register rbp
    Accesses occured at offsets: ['10', '18', '72', '70']
    Register r8
    Accesses occured at offsets: []

> Pointer Access:
    Pointer ffff800000000ff0
    Accesses occured at offsets: ['10', '18', '72', '70']
    Pointer f000000000000000
    Accesses occured at offsets: ['18']
    Pointer 0
    Accesses occured at offsets: []
    Pointer fffffffffff8000
    Accesses occured at offsets: ['8', '10', '0']
    Pointer 7ffffffffefe58
    Accesses occured at offsets: ['28', '20', '-8', '60', '58', '-10']
    Pointer 1c0102a3e
    Accesses occured at offsets: []
    Pointer 7ffffffffefe50
    Accesses occured at offsets: ['30', '28', '0', '68', '60', '-8']
    Pointer 1c0102a54
    Accesses occured at offsets: []

> Traced Pointer Access:
    Traced Pointer rbp -> ffff800000001008 -> fffffffffff8000
    Accesses occured at offsets: ['8', '10', '0']
    Traced Pointer rsp -> 7ffffffffefe50
    Accesses occured at offsets: ['30', '28', '0', '68', '60', '-8']
```

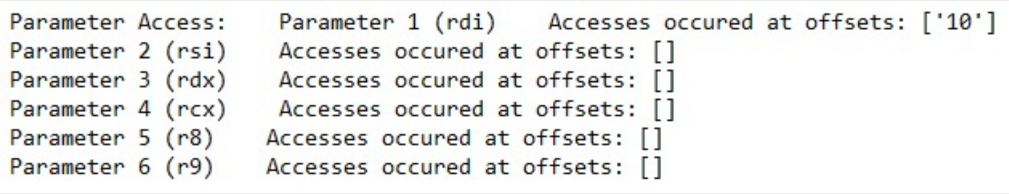


```
'_IN_ADDR' : [ None, {
  'addr4' : [ 0x0, ['IpAddress']],
  'addr6' : [ 0x0, ['Ipv6Address']],
}],
'_INETAF' : [ None, {
 'AddressFamily' : [ 0x18, ['unsigned short']],
 }],
'_LOCAL_ADDRESS' : [ None, {
  'pData' : [ 0x10, ['pointer', ['pointer', ['_IN_ADDR']]]],
 }],
'_ADDRINFO' : [ None, {
 'Local' : [ 0x0, ['pointer', ['_LOCAL_ADDRESS']]],
 'Remote' : [ 0x10, ['pointer', ['_IN_ADDR']]],
  }],
'_TCP_ENDPOINT': [ None, {
 'InetAF' : [ 0x10, ['pointer', ['_INETAF']]],
 'AddrInfo' : [ 0x18, ['pointer', ['_ADDRINFO']]],
 'State' : [ 0x6C, ['Enumeration', …],
 'LocalPort' : [ 0x70, ['unsigned be short']],
 'RemotePort' : [ 0x72, ['unsigned be short']],
 'Owner' : [ 0x258, ['pointer', ['_EPROCESS']]],
 'CreateTime' : [ 0x268, ['WinTimeStamp', dict(is_utc = True)]],
 }],
```

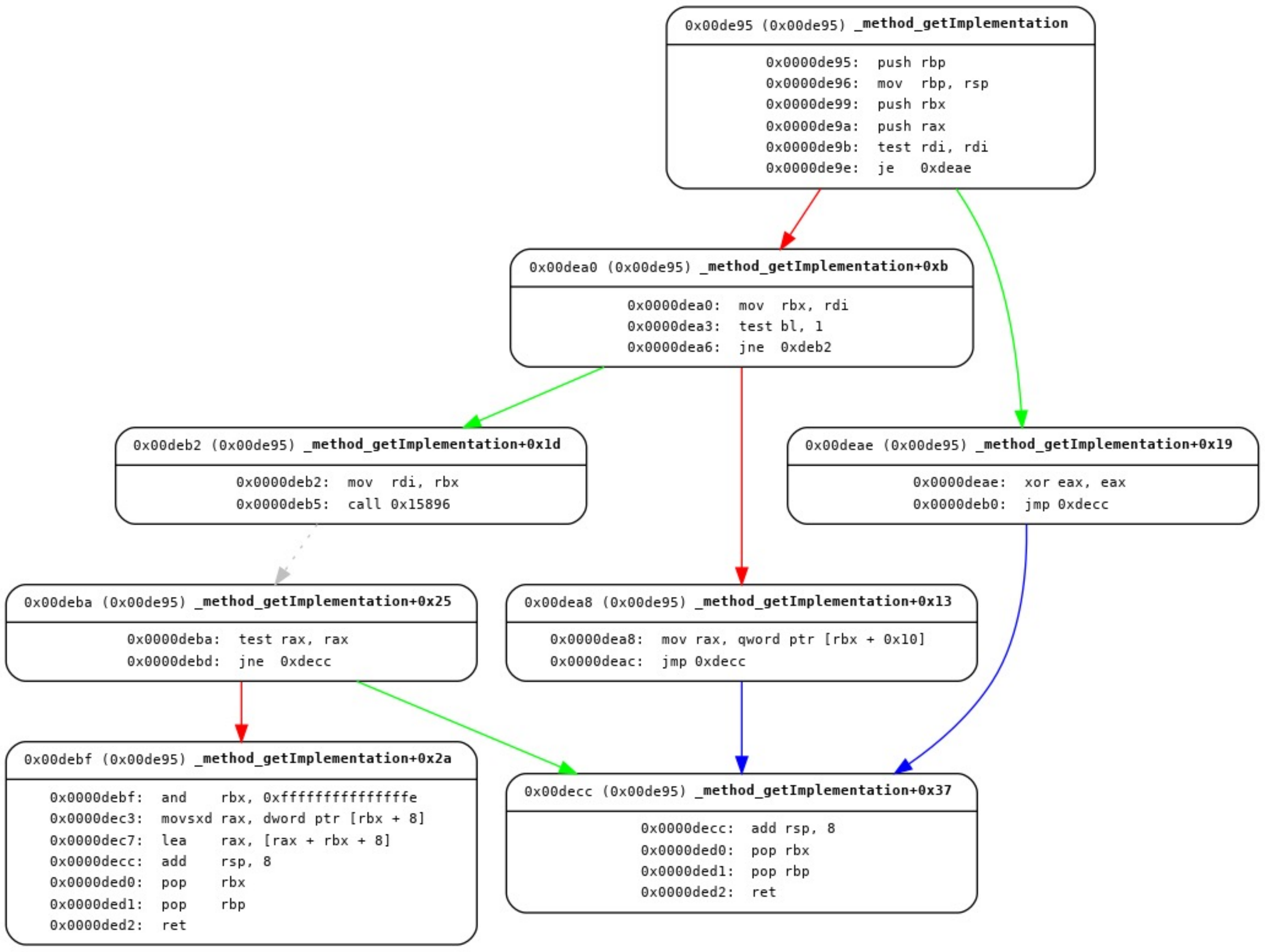
method_getImplementation



```
IMP
method_getImplementation(Method m)
{
    return m ? m->imp : nil;
}
```



```
Parameter Access:    Parameter 1 (rdi)    Accesses occured at offsets: ['10']
Parameter 2 (rsi)    Accesses occured at offsets: []
Parameter 3 (rdx)    Accesses occured at offsets: []
Parameter 4 (rcx)    Accesses occured at offsets: []
Parameter 5 (r8)    Accesses occured at offsets: []
Parameter 6 (r9)    Accesses occured at offsets: []
```



```
0x00de95 (0x00de95) _method_getImplementation
        0x0000de95:  push rbp
        0x0000de96:  mov  rbp, rsp
        0x0000de99:  push rbx
        0x0000de9a:  push rax
        0x0000de9b:  test rdi, rdi
        0x0000de9e:  je   0xdeae
```

```
0x00dea0 (0x00de95) _method_getImplementation+0xb
        0x0000dea0:  mov  rbx, rdi
        0x0000dea3:  test bl, 1
        0x0000dea6:  jne  0xdeb2
```

```
0x00deb2 (0x00de95) _method_getImplementation+0x1d
        0x0000deb2:  mov  rdi, rbx
        0x0000deb5:  call 0x15896
```

```
0x00deae (0x00de95) _method_getImplementation+0x19
        0x0000deae:  xor eax, eax
        0x0000deb0:  jmp 0xdecc
```

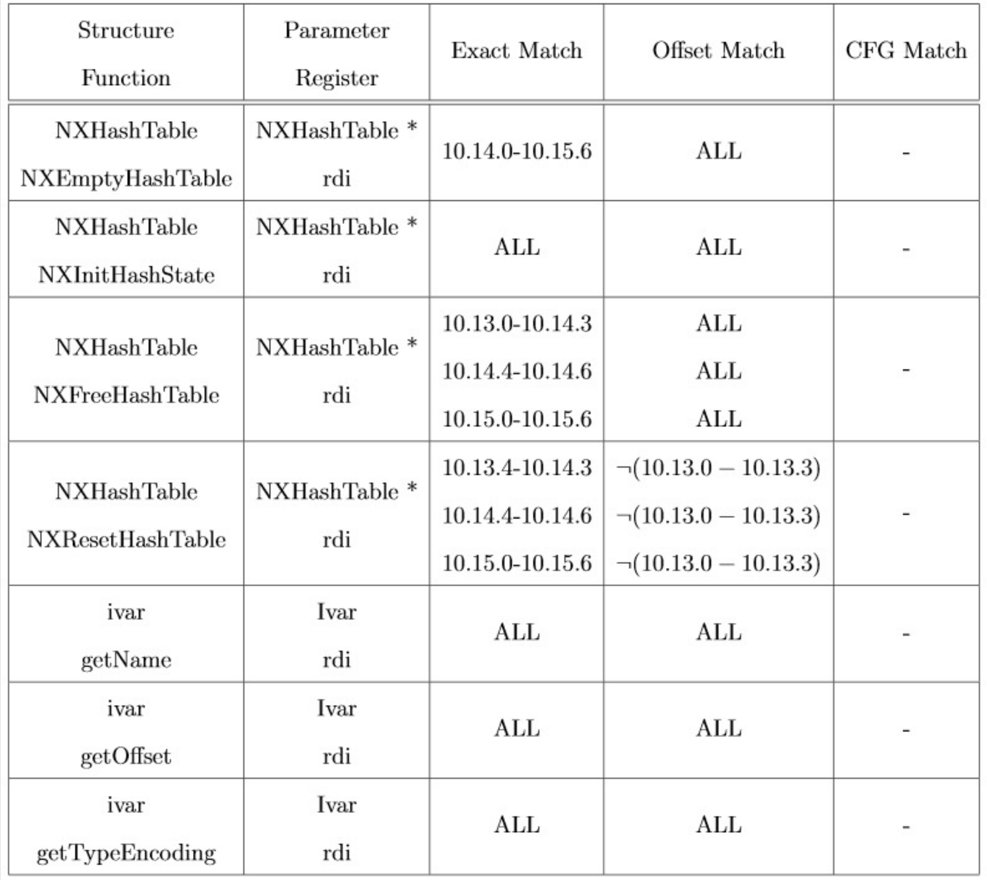
```
0x00deba (0x00de95) _method_getImplementation+0x25
        0x0000deba:  test rax, rax
        0x0000debd:  jne  0xdecc
```

```
0x00dea8 (0x00de95) _method_getImplementation+0x13
    0x0000dea8:  mov rax, qword ptr [rbx + 0x10]
    0x0000deac:  jmp 0xdecc
```

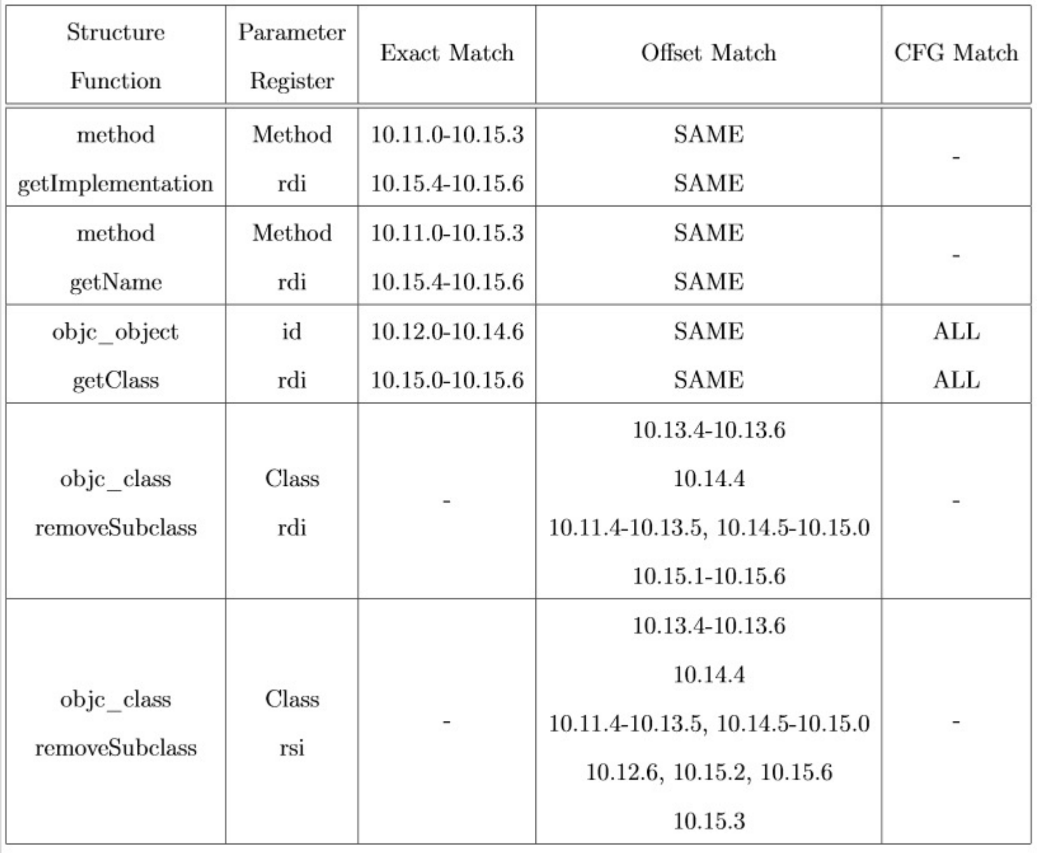
```
0x00debf (0x00de95) _method_getImplementation+0x2a
    0x0000debf:  and    rbx, 0xfffffffffffffffe
    0x0000dec3:  movsxd rax, dword ptr [rbx + 8]
    0x0000dec7:  lea    rax, [rax + rbx + 8]
    0x0000decc:  add    rsp, 8
    0x0000ded0:  pop    rbx
    0x0000ded1:  pop    rbp
    0x0000ded2:  ret
```

```
0x00decc (0x00de95) _method_getImplementation+0x37
        0x0000decc:  add rsp, 8
        0x0000ded0:  pop rbx
        0x0000ded1:  pop rbp
        0x0000ded2:  ret
```

Database Matching Results



Structure Function	Parameter Register	Exact Match	Offset Match	CFG Match
NXHashTable NXEmptyHashTable	NXHashTable * rdi	10.14.0-10.15.6	ALL	-
NXHashTable NXInitHashState	NXHashTable * rdi	ALL	ALL	-
NXHashTable NXFreeHashTable	NXHashTable * rdi	10.13.0-10.14.3 10.14.4-10.14.6 10.15.0-10.15.6	ALL ALL ALL	-
NXHashTable NXResetHashTable	NXHashTable * rdi	10.13.4-10.14.3 10.14.4-10.14.6 10.15.0-10.15.6	¬(10.13.0 − 10.13.3) ¬(10.13.0 − 10.13.3) ¬(10.13.0 − 10.13.3)	-
ivar getName	Ivar rdi	ALL	ALL	-
ivar getOffset	Ivar rdi	ALL	ALL	-
ivar getTypeEncoding	Ivar rdi	ALL	ALL	-



Structure Function	Parameter Register	Exact Match	Offset Match	CFG Match
method getImplementation	Method rdi	10.11.0-10.15.3 10.15.4-10.15.6	SAME SAME	-
method getName	Method rdi	10.11.0-10.15.3 10.15.4-10.15.6	SAME SAME	-
objc_object getClass	id rdi	10.12.0-10.14.6 10.15.0-10.15.6	SAME SAME	ALL ALL
objc_class removeSubclass	Class rdi	-	10.13.4-10.13.6 10.14.4 10.11.4-10.13.5, 10.14.5-10.15.0 10.15.1-10.15.6	-
objc_class removeSubclass	Class rsi	-	10.13.4-10.13.6 10.14.4 10.11.4-10.13.5, 10.14.5-10.15.0 10.12.6, 10.15.2, 10.15.6 10.15.3	-

Future Work

- Rework database construction
  - Include pointers, second degree pointers
- Make publicly available
  - Clean up code
  - Containerize environment, write detailed instructions for setting one up
- Volatility integration

Questions?