BinDbg: Easy Windows Debugging for Binary Ninja

Dave Kukfa
https://kukfa.co
@kukfa_

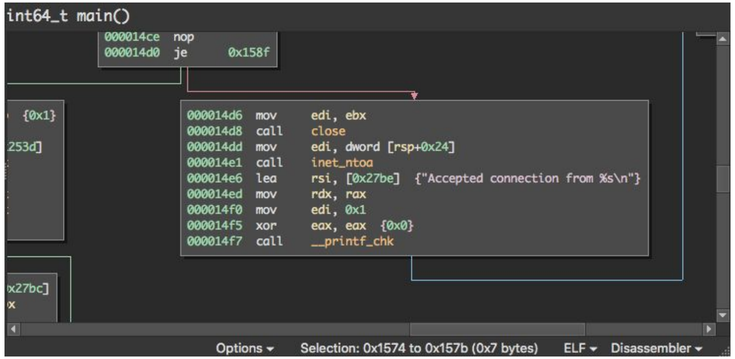
whoami

- Dave Kukfa
- Corporate security engineer by day
- Hobbyist reverse engineer by night
- RIT CSEC Graduate
- SF Bay Area
- Blog: https://kukfa.co
- Twitter: @kukfa_

Disclaimer: This talk and all materials are being released on my own behalf, not on behalf of my employer

Binary analysis tools

- Gold standard: IDA Pro
  - $$$
  - Typically only justifiable by professionals
- Several recent challengers:
  - Radare
  - Hopper
  - **Binary Ninja**
- Binary Ninja is a powerful **static analysis** tool
  - Looking at the program's disassembly without executing it
- I missed the **dynamic** (debugger) integrations that IDA had
  - So I set out to recreate it in Binja!

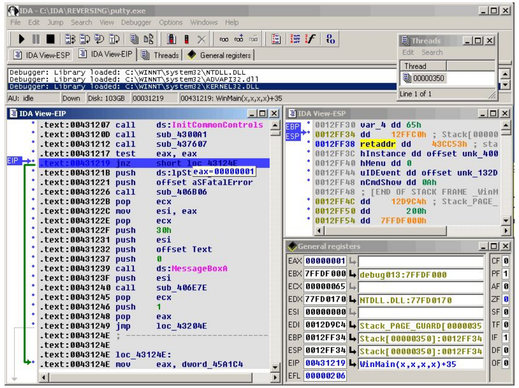


```
int64_t main()

000014ce  nop
000014d0  je      0x158f

000014d6  mov     edi, ebx
000014d8  call    close
000014dd  mov     edi, dword [rsp+0x24]
000014e1  call    inet_ntoa
000014e6  lea     rsi, [0x27be]  {"Accepted connection from %s\n"}
000014ed  mov     rdx, rax
000014f0  mov     edi, 0x1
000014f5  xor     eax, eax  {0x0}
000014f7  call    __printf_chk
```

Source: https://binary.ninja/



IDA - C:\IDA\REVERSING\putty.exe
File Edit Jump Search View Debugger Options Windows Help
IDA View-ESP
IDA View-EIP
Threads
General registers

```
Debugger: Library loaded: C:\WINNT\system32\NTDLL.DLL
Debugger: Library loaded: C:\WINNT\system32\ADVAPI32.dll
Debugger: Library loaded: C:\WINNT\system32\KERNEL32.DLL
AU: idle   Down   Disk: 103GB   00031219   00431219: WinMain(x,x,x,x)+35
```

```
IDA View-EIP
.text:00431207 call    ds:InitCommonControls
.text:0043120D call    sub_4300A1
.text:00431212 call    sub_437607
.text:00431217 test    eax, eax
.text:00431219 jnz     short loc_43124E
.text:0043121B push    ds:lpSt eax=00000001
.text:00431221 push    offset aSFatalError
.text:00431226 call    sub_406B06
.text:0043122B pop     ecx
.text:0043122C mov     esi, eax
.text:0043122E pop     ecx
.text:0043122F push    30h
.text:00431231 push    esi
.text:00431232 push    offset Text
.text:00431237 push    0
.text:00431239 call    ds:MessageBoxA
.text:0043123F push    esi
.text:00431240 call    sub_406E7E
.text:00431245 pop     ecx
.text:00431246 push    1
.text:00431248 pop     eax
.text:00431249 jmp     loc_43204E
.text:0043124E ; ---------------------------------
.text:0043124E
.text:0043124E loc_43124E:
.text:0043124E mov     eax, dword_45A1C4
```

```
Threads
Thread
00000350
Line 1 of 1
```

```
IDA View-ESP
0012FF30 var_4 dd 65h
0012FF34 dd     12FFC0h ; Stack[00000
0012FF38 retaddr dd     43CC53h ; sta
0012FF3C hInstance dd offset unk_400
0012FF40 hMenu dd 0
0012FF44 uIDEvent dd offset unk_132D
0012FF48 nCmdShow dd 0Ah
0012FF48 ; [END OF STACK FRAME _WinM
0012FF4C dd     12D9C4h ; Stack_PAGE_
0012FF50 dd        200h
0012FF54 dd   7FFDF000h
```

```
General registers
EAX 00000001                                  CF 0
EBX 7FFDF000  debug013:7FFDF000               PF 1
ECX 00000065                                  AF 0
EDX 77FD0170  NTDLL.DLL:77FD0170              ZF 0
ESI 00000000                                  SF 0
EDI 0012D9C4  Stack_PAGE_GUARD[0000035        TF 0
EBP 0012FF34  Stack[00000350]:0012FF34        IF 1
ESP 0012FF34  Stack[00000350]:0012FF34        DF 0
EIP 00431219  WinMain(x,x,x,x)+35             OF 0
EFL 00000206
```

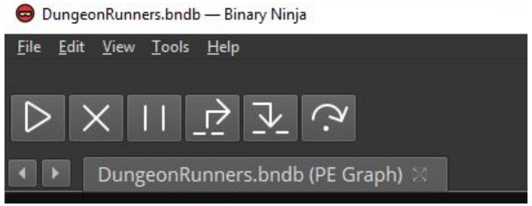
Source: https://www.hex-rays.com/products/ida/

BinDbg

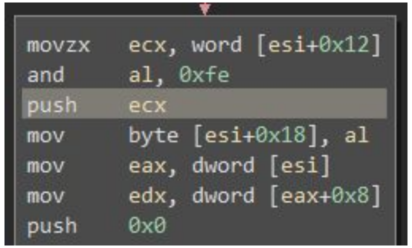
- Binary Ninja plugin that syncs a running debugger (WinDbg) to Binary Ninja
    - Combining static and dynamic analyses
    - Use the debugger's information to supplement Binary Ninja's analysis
    - Control the debugger within Binary Ninja
- This has been done before
    - snare's Binjatron: https://github.com/snare/binjatron
    - Eric Hennenfent's Binja Dynamics: https://github.com/ehennenfent/binja_dynamics
- Windows support on existing solutions is lacking
    - Because I primarily reverse PEs, I wanted to create an easy-to-use Windows solution

Primary features (1/4)

- Launch and control debugging sessions directly from Binary Ninja


DungeonRunners.bndb — Binary Ninja
File Edit View Tools Help
DungeonRunners.bndb (PE Graph)

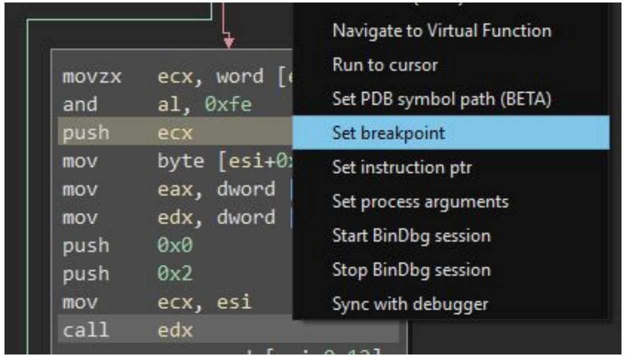
- Syncs Binja disassembly graph with WinDbg instruction pointer



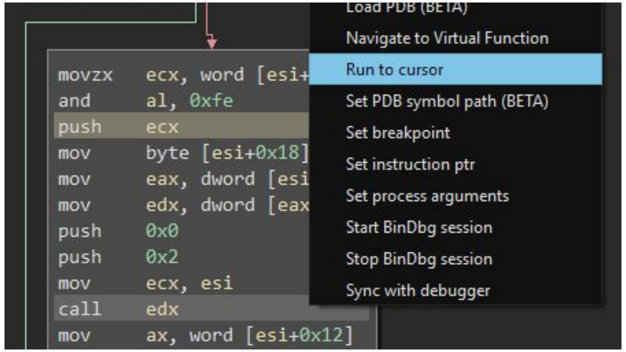
```
movzx   ecx, word [esi+0x12]
and     al, 0xfe
push    ecx
mov     byte [esi+0x18], al
mov     eax, dword [esi]
mov     edx, dword [eax+0x8]
push    0x0
```

Primary features (2/4)

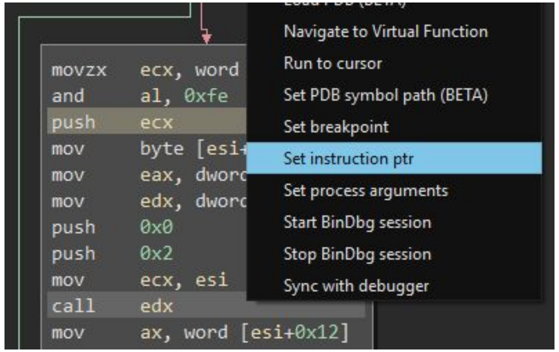
- Set breakpoints and move instruction pointer directly on Binja’s disassembly graph



```
movzx   ecx, word [e
and     al, 0xfe
push    ecx
mov     byte [esi+0x
mov     eax, dword
mov     edx, dword
push    0x0
push    0x2
mov     ecx, esi
call    edx
```



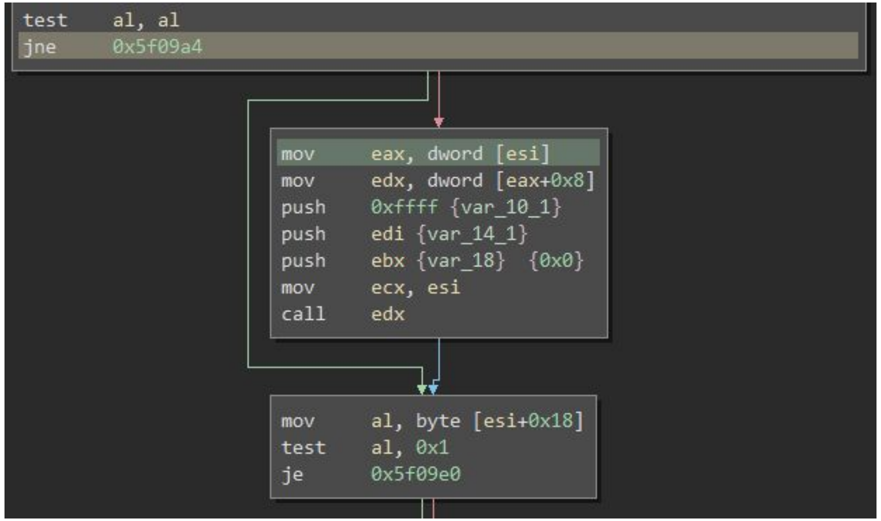
```
movzx   ecx, word [esi+
and     al, 0xfe
push    ecx
mov     byte [esi+0x18]
mov     eax, dword [esi
mov     edx, dword [eax
push    0x0
push    0x2
mov     ecx, esi
call    edx
mov     ax, word [esi+0x12]
```



```
movzx   ecx, word
and     al, 0xfe
push    ecx
mov     byte [esi+
mov     eax, dword
mov     edx, dword
push    0x0
push    0x2
mov     ecx, esi
call    edx
mov     ax, word [esi+0x12]
```

Primary features (3/4)

- Highlight branch decisions on disassembly graph (see where jumps are going)



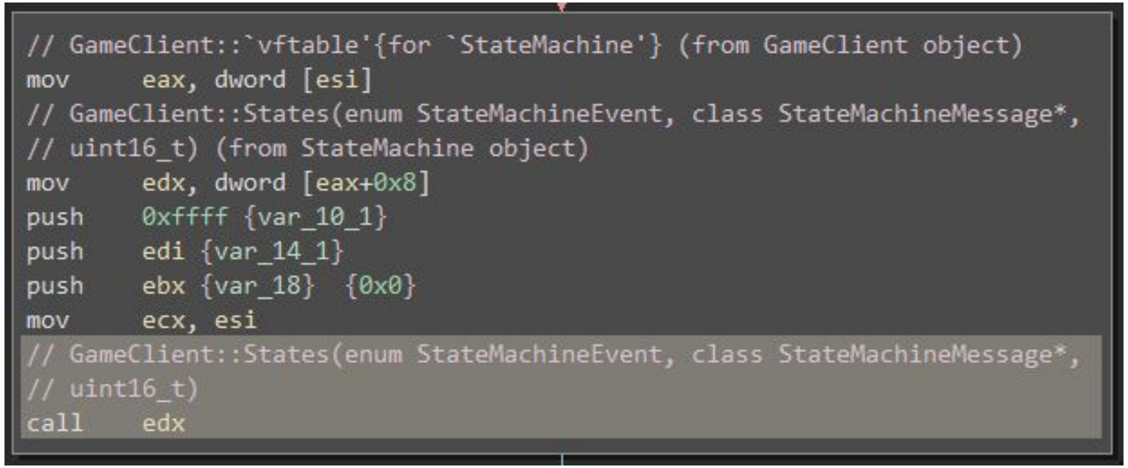
```
test    al, al
jne     0x5f09a4
```

```
mov     eax, dword [esi]
mov     edx, dword [eax+0x8]
push    0xffff {var_10_1}
push    edi {var_14_1}
push    ebx {var_18}  {0x0}
mov     ecx, esi
call    edx
```

```
mov     al, byte [esi+0x18]
test    al, 0x1
je      0x5f09e0
```

Primary features (4/4)

- Resolve vtable calls and vtable references (determine object types)

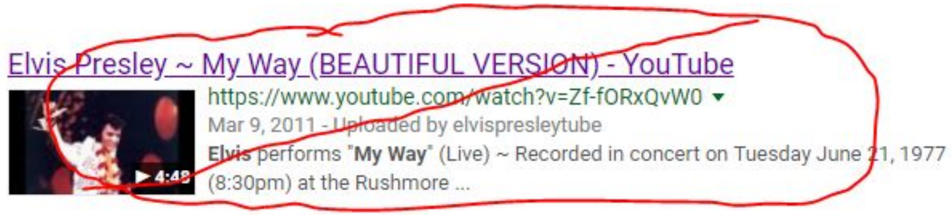


```
// GameClient::`vftable'{for `StateMachine'} (from GameClient object)
mov     eax, dword [esi]
// GameClient::States(enum StateMachineEvent, class StateMachineMessage*,
// uint16_t) (from StateMachine object)
mov     edx, dword [eax+0x8]
push    0xffff {var_10_1}
push    edi {var_14_1}
push    ebx {var_18}  {0x0}
mov     ecx, esi
// GameClient::States(enum StateMachineEvent, class StateMachineMessage*,
// uint16_t)
call    edx
```

Demo

Lessons learned

- Lots of time spent wrestling with pykd
  - Just catch its exceptions and keep going ¯_(ツ)_/¯
- Determining object type using vtables is not 100% reliable
  - In the case of multiple inheritance, can't just observe the first vtable and call it a day
- Windows is weird
  - Named pipes implementation
  - API and COM interfaces
- In hindsight, would have been easier to improve Windows support on existing tools



Elvis Presley ~ My Way (BEAUTIFUL VERSION) - YouTube
https://www.youtube.com/watch?v=Zf-fORxQvW0
Mar 9, 2011 - Uploaded by elvispresleytube
Elvis performs "My Way" (Live) ~ Recorded in concert on Tuesday June 21, 1977 (8:30pm) at the Rushmore ...

Questions?

https://github.com/kukfa/bindbg