txOne networks

2021

Rebuild The Heaven's Gate: from 32-bit Hell back to 64-bit Wonderland

Sheng-Hao Ma

Sheng-Hao Ma

Threat Researcher at TXOne Networks

- **Core member** of CHROOT Security Group
- **Over 10-year experience** in reverse engineering, Windows vulnerability, and Intel 8086.
- **Spoke** at S&P, BlackHat, DEFCON, HITB, HITCON, VXCON, CYBERSEC, and etc.
- **Instructor** of Ministry of National Defense, Ministry of Education, HITCON, and etc.
- **Publication** "Windows APT Warfare: 惡意程式前線作戰指南"

Outline

A. 32-bit Hell & Userland HIPS Design
B. Understanding WOW64 Design by Reversing Engineering
   - WOW64 Process Initiation
   - Path to The Heaven
   - Bishop: The Paradise Translator
C. The 32 bit Hell v.s. 64 bit Heaven
D. wowGrail: Rebuild the Heaven's Gate
E. wowInjector: One Gadget to Take Over The Hell

txOne networks

What "The Hell"

Host-based Intrusion Prevention System (HIPS)

What "The Hell"

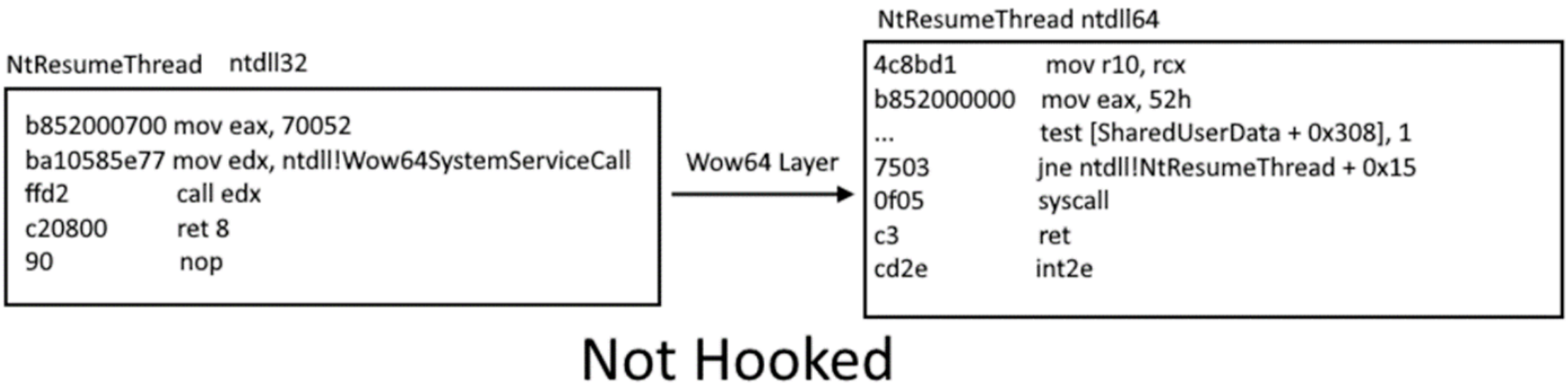
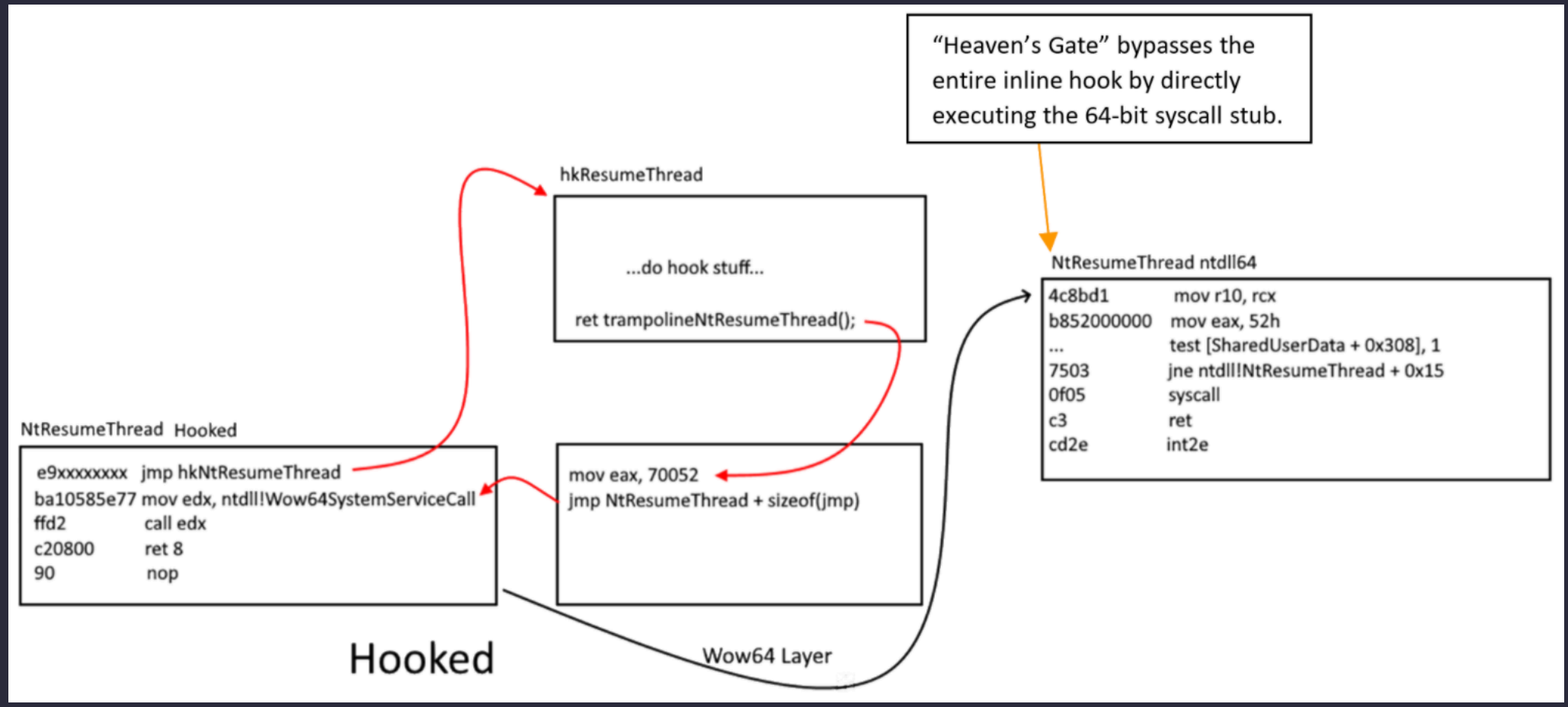


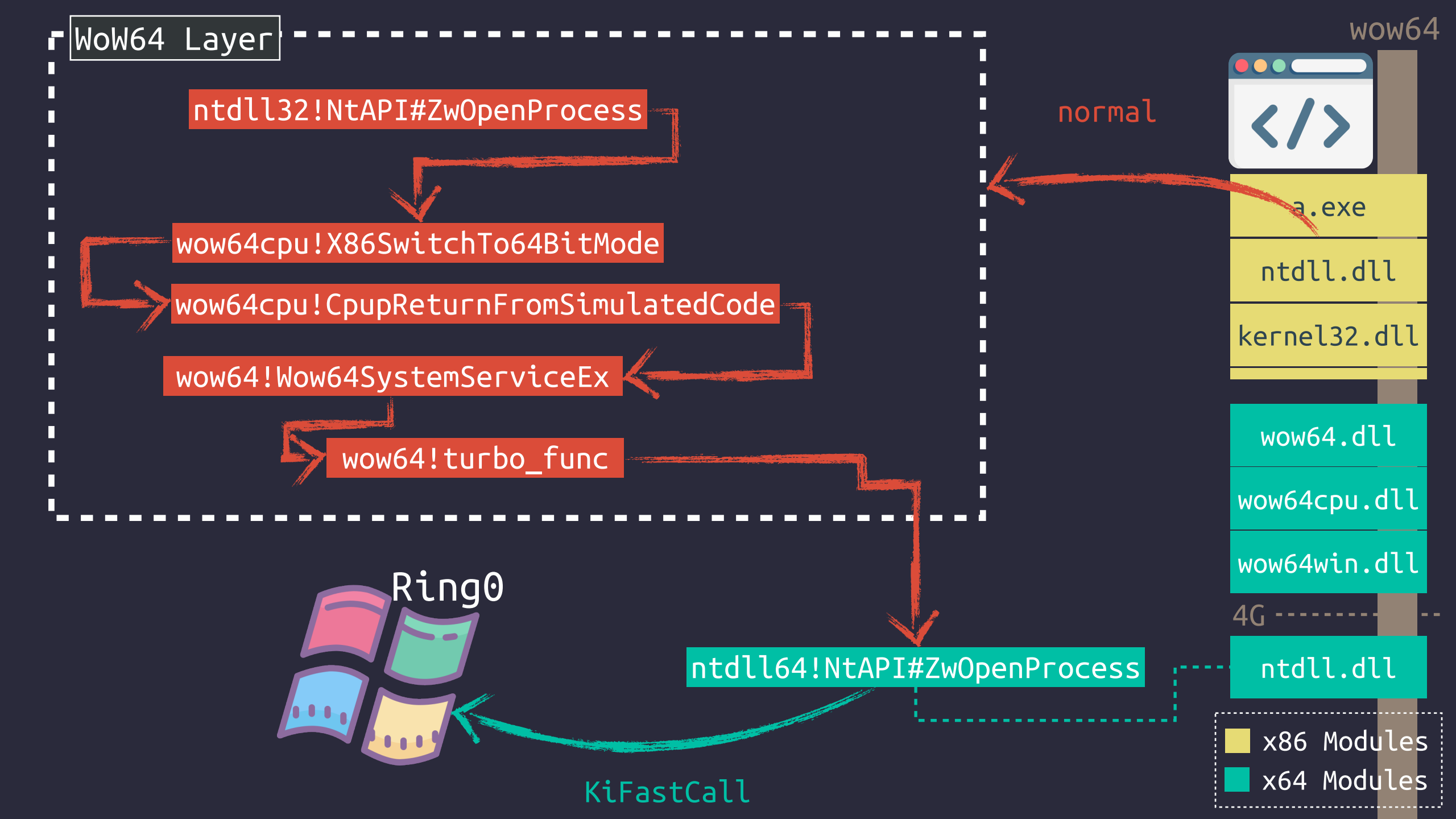
Figure 14: NtResumeThread transitioning through the WOW64 layer

[FireEye] WOW64!Hooks: WOW64 Subsystem Internals and Hooking Techniques

What "The Hell"



[FireEye] WOW64!Hooks: WOW64 Subsystem Internals and Hooking Techniques


WoW64 Layer
ntdll32!NtAPI#ZwOpenProcess
wow64cpu!X86SwitchTo64BitMode
wow64cpu!CpupReturnFromSimulatedCode
wow64!Wow64SystemServiceEx
wow64!turbo_func
normal
wow64
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
4G
ntdll.dll
Ring0
ntdll64!NtAPI#ZwOpenProcess
KiFastCall
x86 Modules
x64 Modules

#Heaven's Gate

A. switch to 64bit CPU mode by setting cs flag
B. get PEB64 by (GS:0x30)->PEB
C. enumerate loaded 64bit modules via PEB->Ldr
D. locate imageBase of NtDll64
E. get expored API ntdll!LdrGetProcedureAddress
F. BOOM! we got the key of Heaven's Gate!

Heaven's Gate

KiFastCall

ntdll64!NtAPI#ZwOpenProcess

wow64

a.exe

ntdll.dll

kernel32.dll

wow64.dll

wow64cpu.dll

wow64win.dll

4G

ntdll.dll

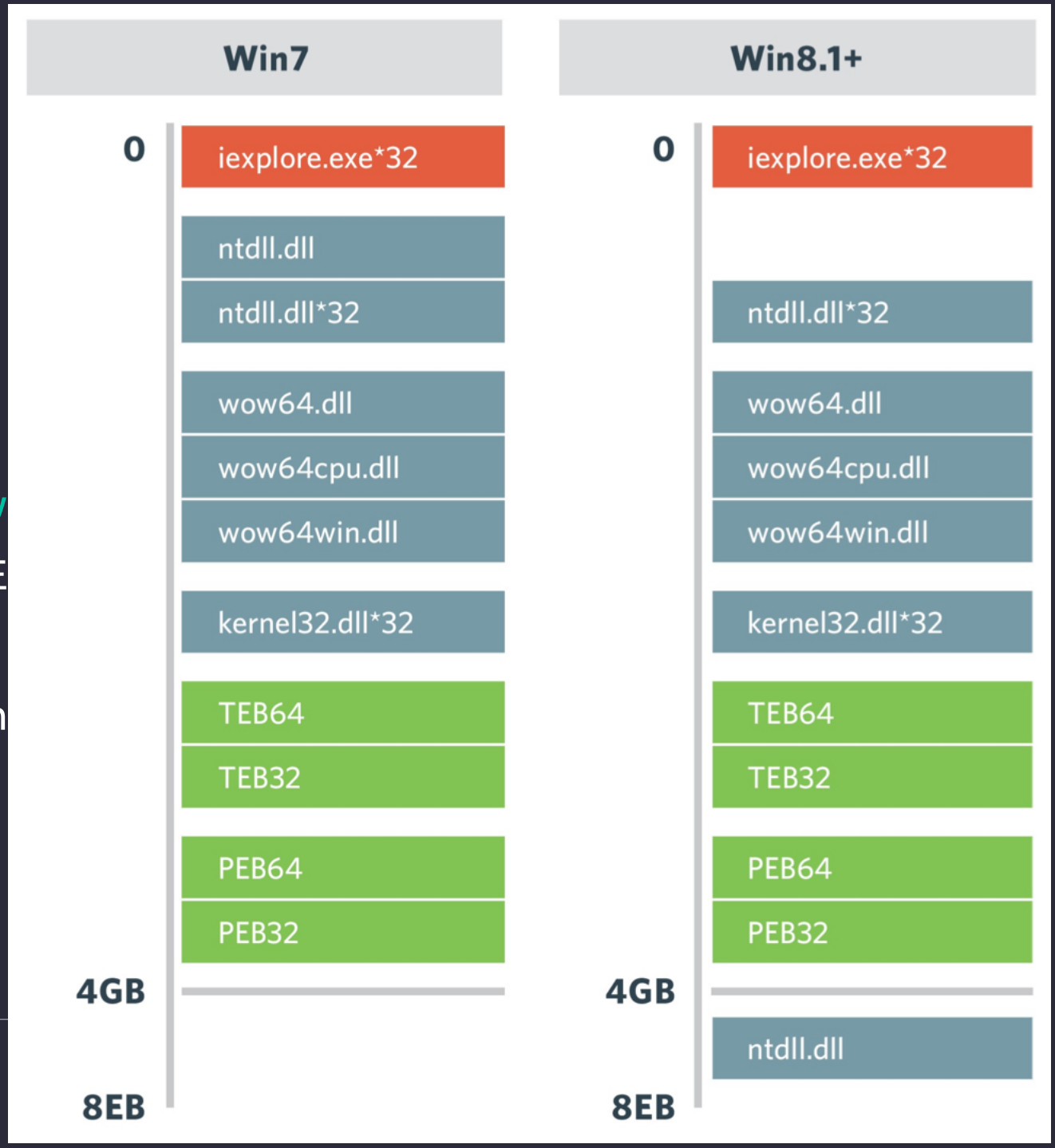
x86 Modules

x64 Modules

Heaven's Gate

Reference

- 2011 - Mixing x86 with x64 code by
- 2012 - Knockin’ on Heaven’s Gate by
- 2012 - KERNEL: Creation of Thread E
- 2018 - WoW64 internals by wbenny
- 2020 - WOW64 Subsystem Internals an



Heaven's Gate

hard to use & not stable enough ¯_(ツ)_/¯

Reference

- 2011 - Mixing x86 with x64 code by ReWolf
- 2012 - Knockin’ on Heaven’s Gate by george_nicolaou
- 2012 - KERNEL: Creation of Thread Environment Block (TEB) by waleedassar
- 2018 - WoW64 internals by wbenny
- 2020 - WOW64 Subsystem Internals and Hooking Techniques by FireEye
- 2021 - wowGrail: Abusing the Translator to Simulate 32-bit Interrupts

txOne networks

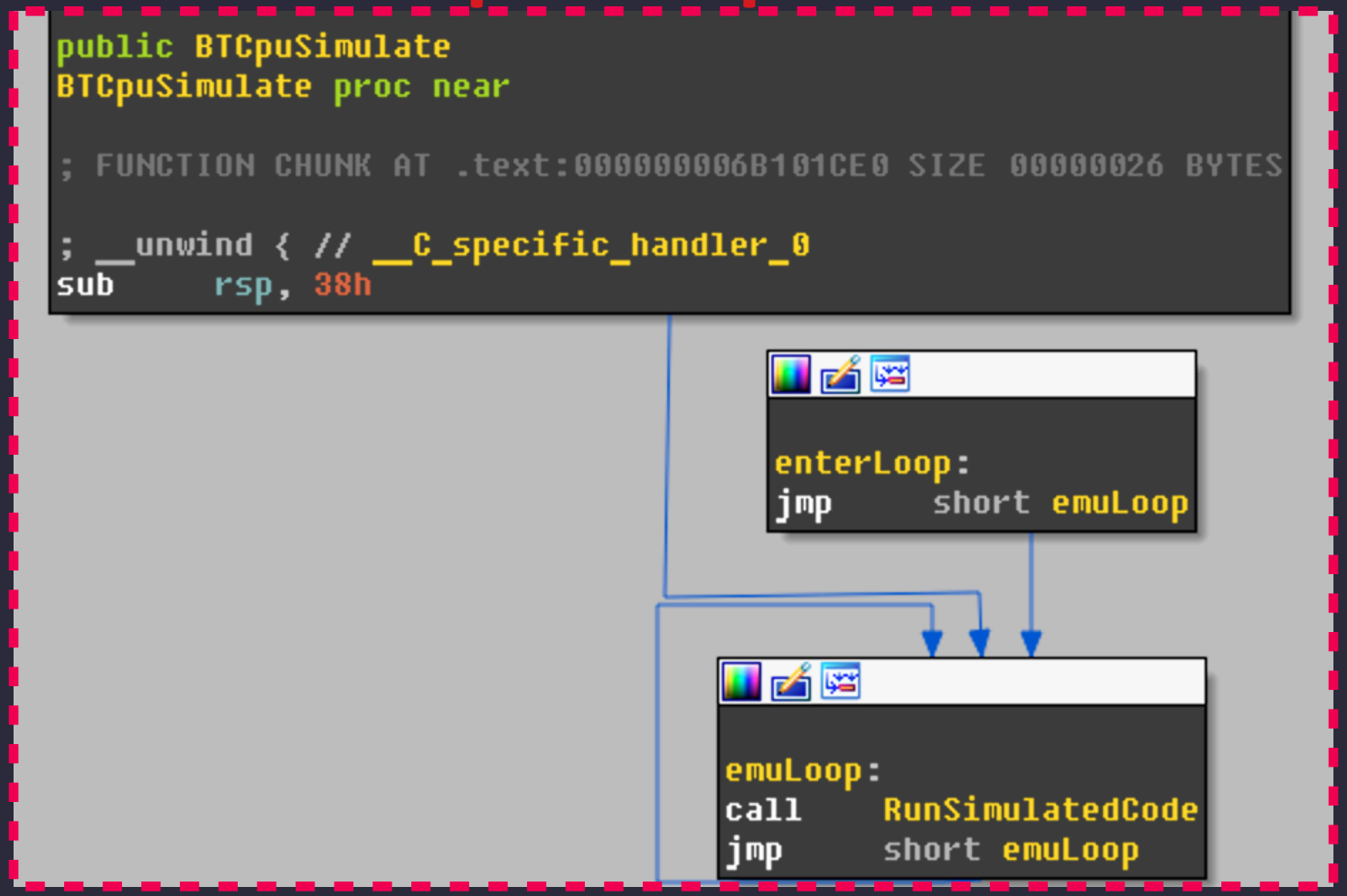
The WOW64 Layer

Understanding WOW64 Design by Reversing Engineering

WOW64 Process Initiation

32BIT PROGRAM MANAGED IN 64BIT PROCESS

wow64cpu!BtCpuSimulate



txOne networks

wow64cpu!RunSimulatedCode

```
push    r15
push    r14
push    r13
push    r12
push    rbx
push    rsi
push    rdi
push    rbp
sub     rsp, 68h
mov     r12, gs:30h       ; r12 = TEB
lea     r15, TurboThunkDispatch
mov     r13, [r12+(_TEB64.TlsSlots+8)] ; WOW64_CPURESERVED
add     r13, 80h ; '€'
```

r12 point to TEB64 struct

r15 point to TurboThunk Table

r13 point to WoW64 Thread Context

wow64cpu!RunSimulatedCode

```
TurboThunkDispatch dq offset TurboDispatchJumpAddressEnd ; DATA XREF: ...
                                        ; Index = 0
off_6B103608    dq offset Thunk0Arg     ; DATA XREF: ...
off_6B103610    dq offset Thunk0ArgReloadState ; DATA XREF: ...
off_6B103618    dq offset Thunk1ArgSp   ; DATA XREF: ...
off_6B103620    dq offset Thunk1ArgNSp  ; DATA XREF: ...
off_6B103628    dq offset Thunk2ArgNSpNSp ; DATA XREF: ...
off_6B103630    dq offset Thunk2ArgNSpNSpReloadState ; DATA XREF: ...
off_6B103638    dq offset Thunk2ArgSpNSp ; DATA XREF: ...
off_6B103640    dq offset Thunk2ArgSpSp ; DATA XREF: ...
off_6B103648    dq offset Thunk2ArgNSpSp ; DATA XREF: ...
off_6B103650    dq offset Thunk3ArgNSpNSpNSp ; DATA XREF: ...
off_6B103658    dq offset Thunk3ArgSpSpSp ; DATA XREF: ...
off_6B103660    dq offset Thunk3ArgSpNSpNSp ; DATA XREF: ...
off_6B103668    dq offset Thunk3ArgSpNSpNSpReloadState ; DATA XREF: ...
off_6B103670    dq offset Thunk3ArgSpSpNSp ; DATA XREF: ...
off_6B103678    dq offset Thunk3ArgNSpSpNSp ; DATA XREF: ...
off_6B103680    dq offset Thunk3ArgSpNSpSp ; DATA XREF: ...
off_6B103688    dq offset Thunk4ArgNSpNSpNSpNSp ; DATA XREF: ...
off_6B103690    dq offset Thunk4ArgSpSpNSpNSp ; DATA XREF: ...
off_6B103698    dq offset Thunk4ArgSpSpNSpNSpReloadState ; DATA XREF: ...
off_6B1036A0    dq offset Thunk4ArgSpNSpNSpNSp ; DATA XREF: ...
off_6B1036A8    dq offset Thunk4ArgSpNSpNSpNSpReloadState ; DATA XREF: ...
off_6B1036B0    dq offset Thunk4ArgNSpSpNSpNSp ; DATA XREF: ...
off_6B1036B8    dq offset Thunk4ArgSpSpSpNSp ; DATA XREF: ...
off_6B1036C0    dq offset QuerySystemTime ; DATA XREF: ...
off_6B1036C8    dq offset GetCurrentProcessorNumber ; DATA XREF: ...
off_6B1036D0    dq offset ReadWriteFile ; DATA XREF: ...
off_6B1036D8    dq offset DeviceIoctlFile ; DATA XREF: ...
off_6B1036E0    dq offset RemoveIoCompletion ; DATA XREF: ...
off_6B1036E8    dq offset WaitForMultipleObjects ; DATA XREF: ...
off_6B1036F0    dq offset WaitForMultipleObjects32 ; DATA XREF: ...
off_6B1036F8    dq offset CpupReturnFromSimulatedCode ; DATA XREF: ...
                dq offset ThunkNone     ; Index: 32
```

r12 point to TEB64 struct

r15 point to TurboThunk Table

r13 point to WoW64 Thread Context

txOne networks

NtAPI Trampoline

32 BIT INTERRUPT BACK TO 64 BIT

```
0:000> u ntdll!NtResumeThread
ntdll!NtResumeThread:
775c6970 b852000700      mov     eax,70052h
775c6975 ba10585e77      mov     edx,offset ntdll!Wow64SystemServiceCall (775e5810)
775c697a ffd2            call    edx
775c697c c20800          ret     8
775c697f 90              nop
```

```
ntdll_77550000!Wow64SystemServiceCall:
775e5810 ff2528c26777    jmp     dword ptr [ntdll_77550000!Wow64Transition (7767c228)]
```

Address	Bytes	Opcode	
wow64cpu.dll+6000	EA 09600277 3300	jmp	0033:wow64cpu.dll+6009
wow64cpu.dll+6007	00 00	add	[rax],al
wow64cpu.dll+6009	41 FF A7 F8000000	jmp	qword ptr [r15+000000F8]

wow64cpu!CpupReturnFromSimulatedCode

#Simulate

wow64cpu!CpupReturnFromSimulatedCode

```
CpupReturnFromSimulatedCode:            ; CODE XREF: W
                                        ; DATA XREF: B
                xchg    rsp, r14
                mov     r8d, [r14]
                add     r14, 4
                mov     [r13+3Ch], r8d
                mov     [r13+48h], r14d
                lea     r11, [r14+4]
                mov     [r13+20h], edi
                mov     [r13+24h], esi
                mov     [r13+28h], ebx
                mov     [r13+38h], ebp
                pushfq
                pop     r8
                mov     [r13+44h], r8d
; Exported entry   9. TurboDispatchJumpAddressStart

                public TurboDispatchJumpAddressStart
TurboDispatchJumpAddressStart:          ; DATA XREF: .
                mov     ecx, eax
                shr     ecx, 10h
                jmp     qword ptr [r15+rcx*8]
```

```
 public TurboDispatchJumpAddressEnd
mpAddressEnd:                           ; CODE XREF: R
                                        ; RunSimulated
                                        ; DATA XREF: .
mov     ecx, eax
mov     rdx, r11
call    cs:__imp_Wow64SystemServiceEx
mov     [r13+34h], eax
jmp     restoreStatus
```

#Simulate

wow64cpu!CpupReturnFromSimulatedCode

```
CpupReturnFromSimulatedCode:            ; CODE XREF: W
                                        ; DATA XREF:
                xchg    rsp, r14
                mov     r8d, [r14]
                add     r14, 4
                mov     [r13+3Ch], r8d
                mov     [r13+48h], r14d
                lea     r11, [r14+4]
                mov     [r13+20h], edi
                mov     [r13+24h], esi
                mov     [r13+28h], ebx
                mov     [r13+38h], ebp
                pushfq
                pop     r8
                mov     [r13+44h], r8d
; Exported entry   9. TurboDispatchJumpAddressStart

                public TurboDispatchJumpAddressStart
TurboDispatchJumpAddressStart:          ; DATA XREF:
                mov     ecx, eax
                shr     ecx, 10h
                jmp     qword ptr [r15+rcx*8]
```

```
restoreStatus:                          ; CODE XREF: RunSi
                btr     dword ptr [r13-80h], 0
                jb      short loc_7702168E
; 6:      __asm { jmp     fword ptr [r14] }
                mov     edi, [r13+20h]
                mov     esi, [r13+24h]
                mov     ebx, [r13+28h]
                mov     ebp, [r13+38h]
                mov     eax, [r13+34h]
                mov     r14, rsp
                mov     dword ptr [rsp+0A8h+var_A8+4], 23h
                mov     r8d, 2Bh ; '+'
                mov     ss, r8d
                mov     r9d, [r13+3Ch]
                mov     dword ptr [rsp+0A8h+var_A8], r9d
                mov     esp, [r13+48h]
                jmp     fword ptr [r14]
```

```
public TurboDispatchJumpAddressEnd
mpAddressEnd:                           ; CODE XREF: R
                                        ; RunSimulated
                                        ; DATA XREF: .
mov     ecx, eax
mov     rdx, r11
call    cs:__imp_Wow64SystemServiceEx
mov     [r13+34h], eax
jmp     restoreStatus
```

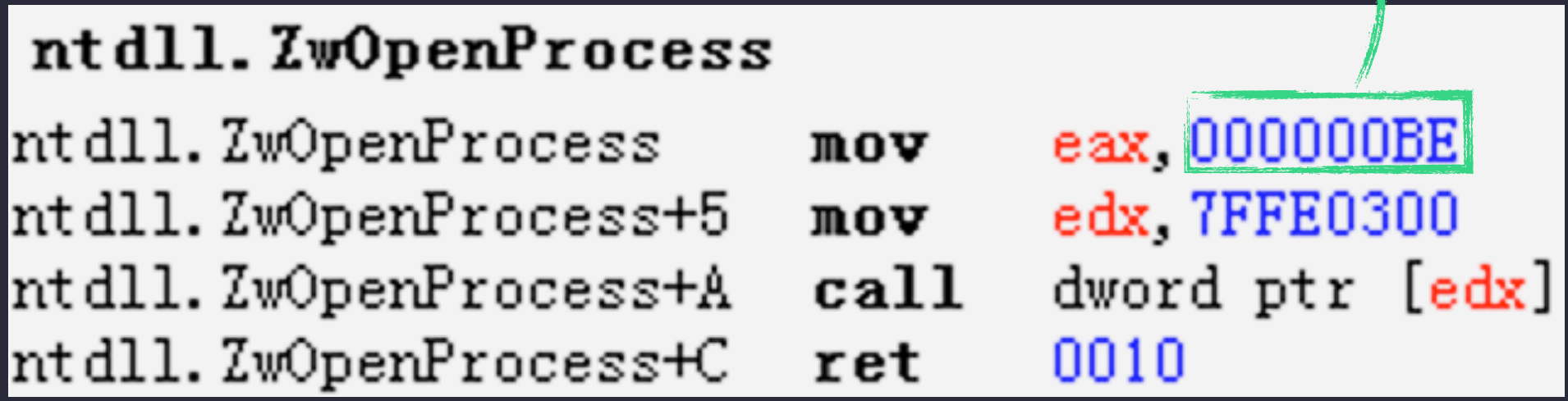
Heaven's Translator

CONVERT X86 CALLING CONVENTION INTO X64 MODE

#Translator

wow64!Wow64SystemServiceEx

```
typedef struct _WOW64_SYSTEM_SERVICE {
    USHORT SystemCallNumber  : 12;
    USHORT ServiceTableIndex :  4;
} WOW64_SYSTEM_SERVICE, *PWOW64_SYSTEM_SERVICE;
```



txOne networks

#Translator

wow64!Wow64SystemServiceEx

```
typedef struct _WOW64_SYSTEM_SERVICE {
    USHORT SystemCallNumber  : 12;
    USHORT ServiceTableIndex :  4;
} WOW64_SYSTEM_SERVICE, *PWOW64_SYSTEM_SERVICE;
```

```
NTSTATUS Wow64SystemServiceEx(_WOW64_SYSTEM_SERVICE syscall, uint32_t *args) {
  TEB64 = __readgsqword(0x30u);
  srvTableIndx = (*&syscall >> 12) & 3;
  srvNumber    = syscall.SystemCallNumber & 0xFFF;
  if ( invalidSyscallNum(srvNumber)  == true ) {
    ret = 0xC000001C;
    goto bye;
  }

  ...
  ptrNtAPI64_TurboFunc = turboAddrTable[ServiceTableIndex].Base[ServiceNumber];
```

#Translator

wow64!Wow64SystemServiceEx

```
NTSTATUS Wow64SystemServiceEx(_WOW64_SYSTEM_SERVI
  TEB64 = __readgsqword(0x30u);
  srvTableIndx = (*&syscall >> 12) & 3;
  srvNumber    = syscall.SystemCallNumber & 0xFFF
  if ( invalidSyscallNum(srvNumber)  == true ) {
    ret = 0xC000001C;
    goto bye;
  }

  ...
  ptrNtAPI64_TurboFunc = turboAddrTable[ServiceTa
```

```
sdwhnt32JumpTable dq offset whNtAccessCheck
                  dq offset whNtWorkerFactoryWorkerReady
                  dq offset whNtAcceptConnectPort
                  dq offset whNtMapUserPhysicalPagesScatter
                  dq offset whNtWaitForSingleObject
                  dq offset whNtCallbackReturn
                  dq offset whNtReadFile
                  dq offset whNtDeviceIoControlFile
                  dq offset whNtWriteFile
                  dq offset whNtRemoveIoCompletion
                  dq offset whNtReleaseSemaphore
                  dq offset whNtReplyWaitReceivePort
                  dq offset whNtReplyPort
                  dq offset whNtSetInformationThread
                  dq offset whNtSetEvent
                  dq offset whNtClose
                  dq offset whNtQueryObject
                  dq offset whNtQueryInformationFile
                  dq offset whNtOpenKey
                  dq offset whNtEnumerateValueKey
                  dq offset whNtFindAtom
                  dq offset whNtQueryDefaultLocale
                  dq offset whNtQueryKey
                  dq offset whNtQueryValueKey
                  dq offset whNtAllocateVirtualMemory
                  dq offset whNtQueryInformationProcess
                  dq offset whNtWaitForMultipleObjects
                  dq offset whNtWriteFileGather
                  dq offset whNtSetInformationProcess
                  dq offset whNtCreateKey
```

#Translator

wow64!Wow64SystemServiceEx

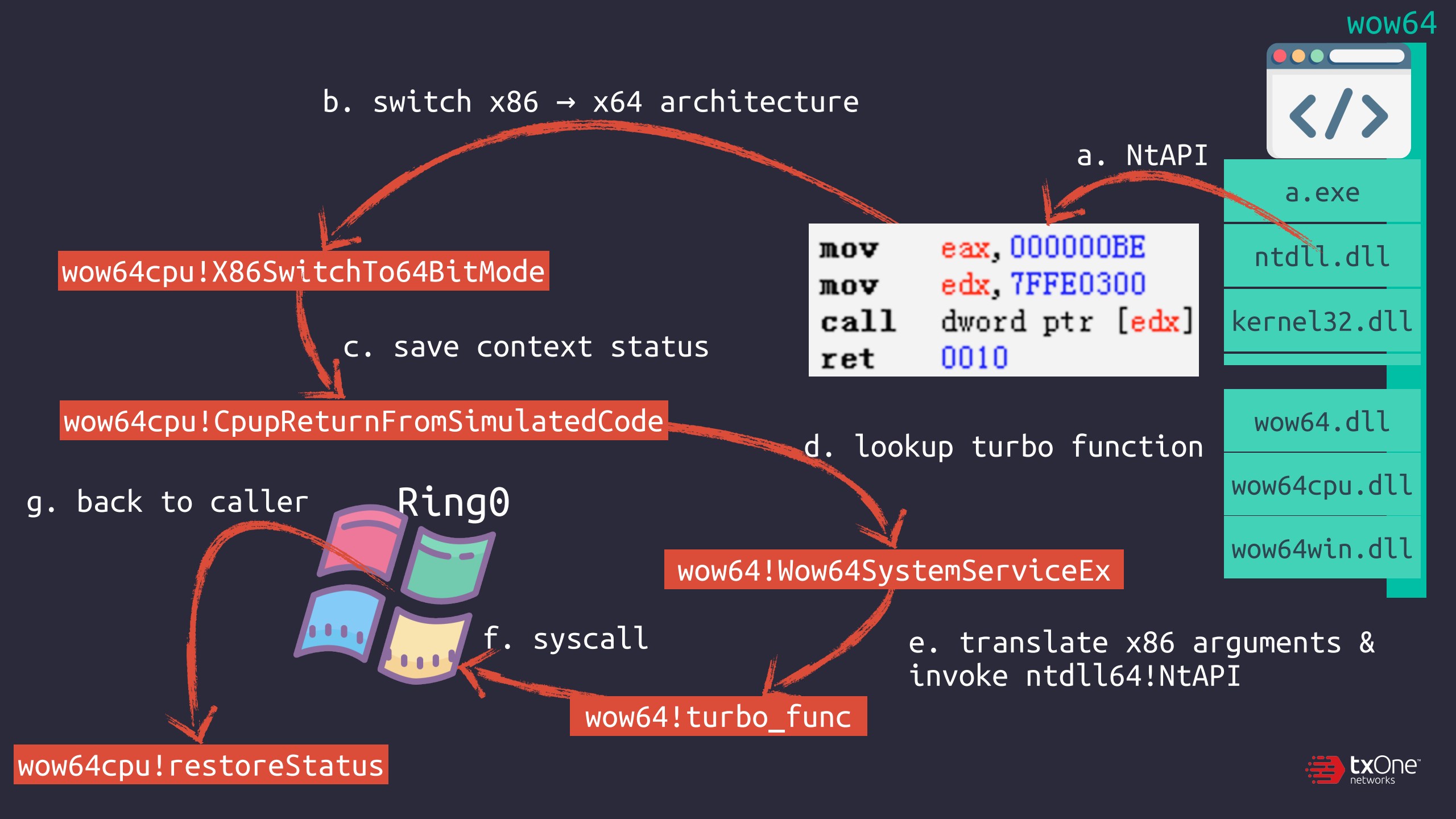
```
typedef struct _WOW64_SYSTEM_SERVICE {
    USHORT SystemCallNumber  : 12;
    USHORT ServiceTableIndex :  4;
} WOW64_SYSTEM_SERVICE, *PWOW64_SYSTEM_SERVICE;
```

```
NTSTATUS Wow64SystemServiceEx(_WOW64_SYSTEM_SERVICE syscall, uint32_t *args) {
  TEB64 = __readgsqword(0x30u);
  srvTableIndx = (*&syscall >> 12) & 3;
  srvNumber    = syscall.SystemCallNumber
  if ( invalidSyscallNum(srvNumber)  == tr
    ret = 0xC000001C;
    goto bye;
  }

  ...
  ptrNtAPI64_TurboFunc = turboAddrTable[Se
```

```
  else if ( ptrNtAPI64_TurboFunc == whNtCallbackReturn )
    ret = whNtCallbackReturn(args);
  else if ( ptrNtAPI64_TurboFunc == whNtQueryVirtualMemory )
    ret = whNtQueryVirtualMemory(args);
  else if ( ptrNtAPI64_TurboFunc == whNtOpenKeyEx )
    ret = whNtOpenKeyEx(args);
  else if ( ptrNtAPI64_TurboFunc == whNtQueryValueKey )
    ret = whNtQueryValueKey(args);
  else if ( ptrNtAPI64_TurboFunc == whNtProtectVirtualMemory )
    ret = whNtProtectVirtualMemory(args);
  else
    ret = ptrNtAPI64_TurboFunc(args);

  ...
  return ret;
}
```


wow64
b. switch x86 → x64 architecture
a. NtAPI
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
mov eax, 000000BE
mov edx, 7FFE0300
call dword ptr [edx]
ret 0010
wow64cpu!X86SwitchTo64BitMode
c. save context status
wow64cpu!CpupReturnFromSimulatedCode
d. lookup turbo function
Ring0
g. back to caller
wow64!Wow64SystemServiceEx
f. syscall
e. translate x86 arguments &
invoke ntdll64!NtAPI
wow64!turbo_func
wow64cpu!restoreStatus
txOne networks

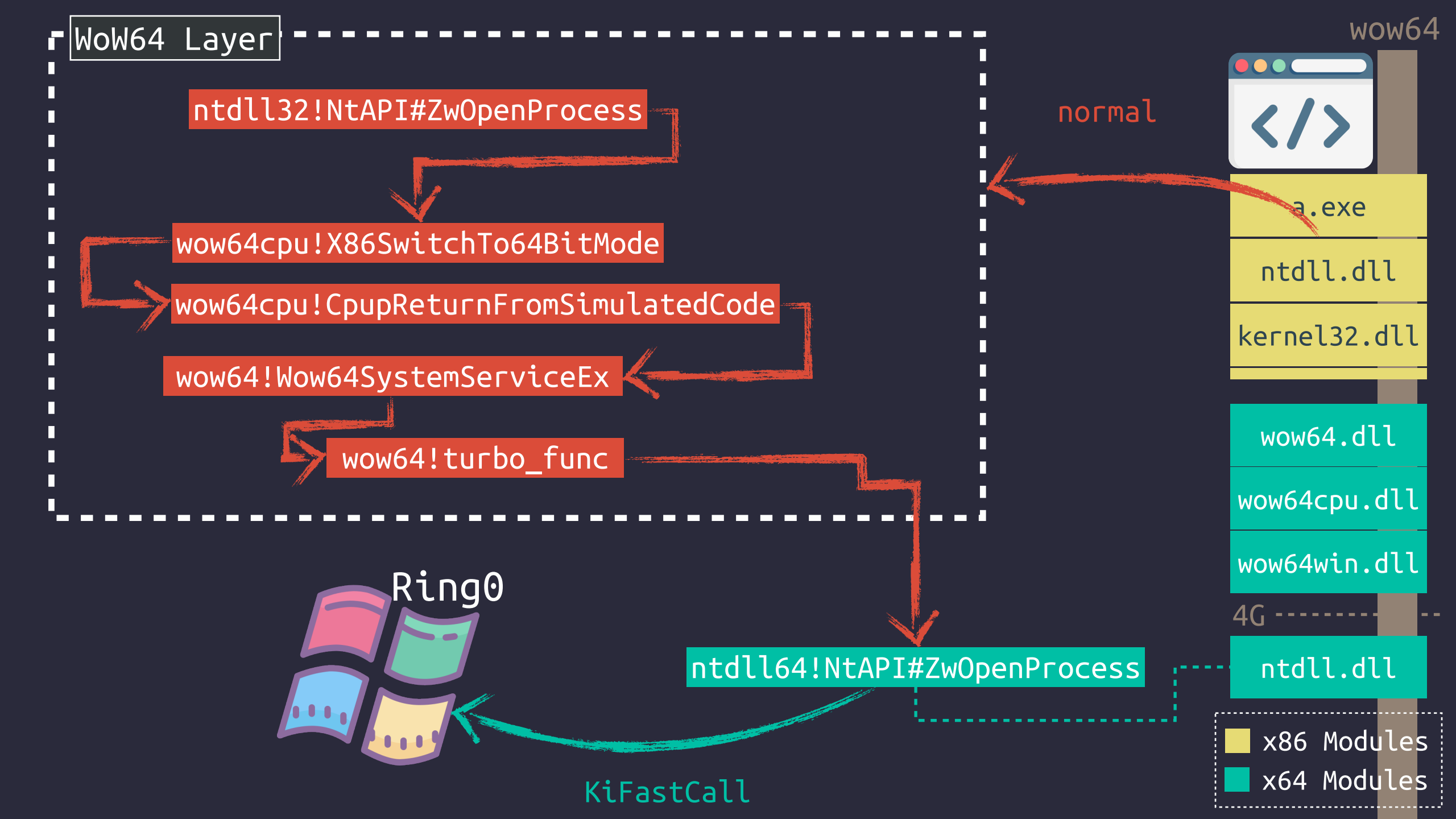
Recap

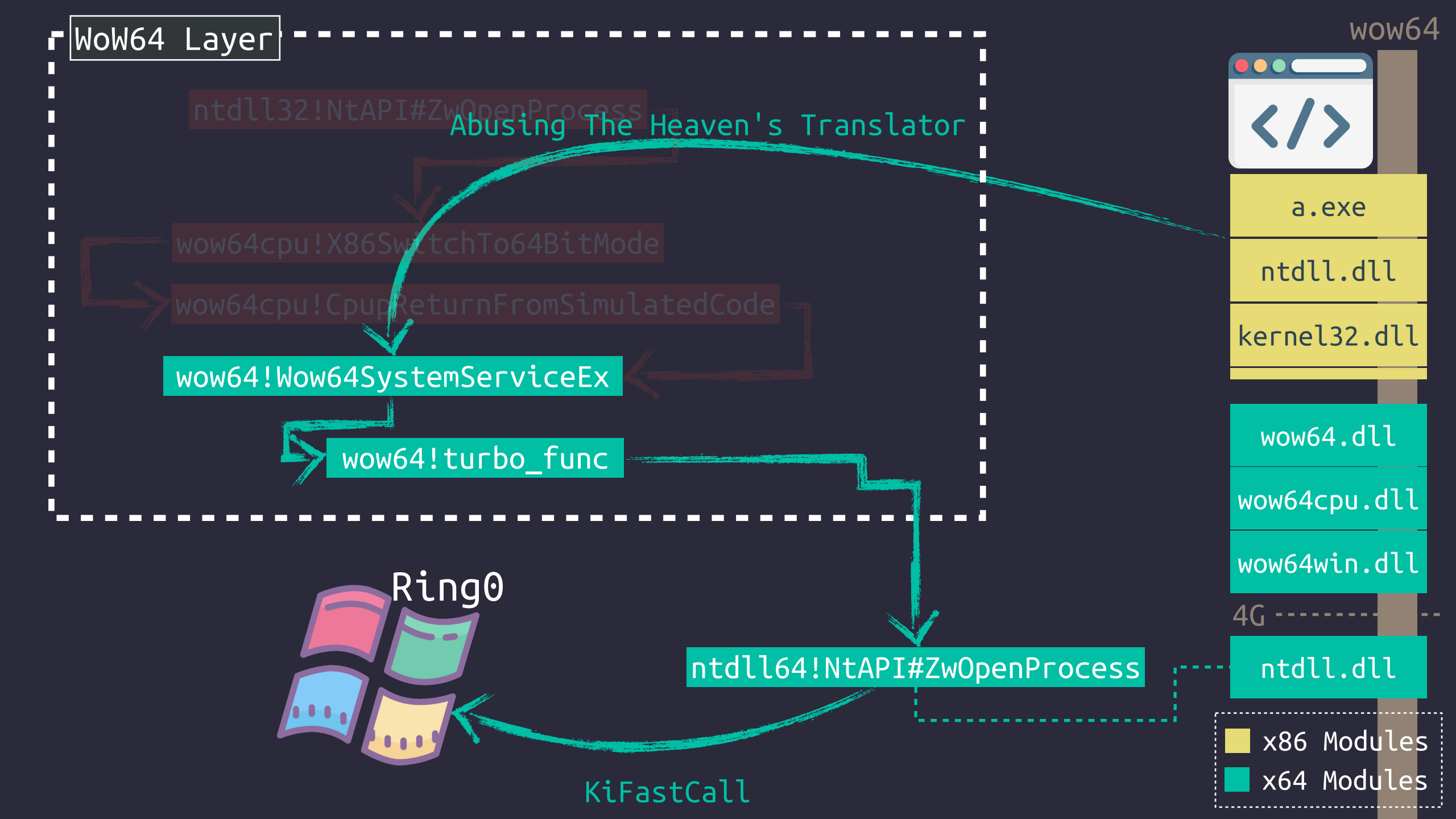
- Switching the CS segment to 23h or 33h makes it possible for the Intel chip to change the chosen instruction set with 32 bit or 64 bit.
- Register r13 point to the 32-bit thread context used as snapshot status. It will be back up when the thread jumps from 32-bit to 64-bit, and reset from 64 bit back to 32-bit.
- wow64!Wow64SystemServiceEx used as translator for us to simulate any 32-bit system interrupt to the 64-bit kernel.

txOne networks

Rebuild A Path To Heaven's Gate

From 32-bit Hell Back to 64-bit Wonderland


WoW64 Layer
ntdll32!NtAPI#ZwOpenProcess
wow64cpu!X86SwitchTo64BitMode
wow64cpu!CpupReturnFromSimulatedCode
wow64!Wow64SystemServiceEx
wow64!turbo_func
normal
wow64
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
4G
ntdll.dll
Ring0
ntdll64!NtAPI#ZwOpenProcess
KiFastCall
x86 Modules
x64 Modules


WoW64 Layer
ntdll32!NtAPI#ZwOpenProcess
Abusing The Heaven's Translator
wow64cpu!X86SwitchTo64BitMode
wow64cpu!CpupReturnFromSimulatedCode
wow64!Wow64SystemServiceEx
wow64!turbo_func
Ring0
ntdll64!NtAPI#ZwOpenProcess
KiFastCall
wow64
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
4G
ntdll.dll
x86 Modules
x64 Modules

#wowGrail

A. switch to 64bit CPU mode by setting cs flag
B. get PEB64 by (GS:0x30)->PEB
C. enumerate loaded 64bit modules via PEB->Ldr
D. locate imageBase of WoW64.dll
E. get expored API wow64!Wow64SystemServiceEx
F. pass 32-bit va_start & executing it to simulate our 32-bit as 64-bit interrupt ;)

wow64

wowGrail

wow64!Wow64SystemServiceEx

ntdll64!NtAPI#ZwOpenProcess

KiFastCall

a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll

4G

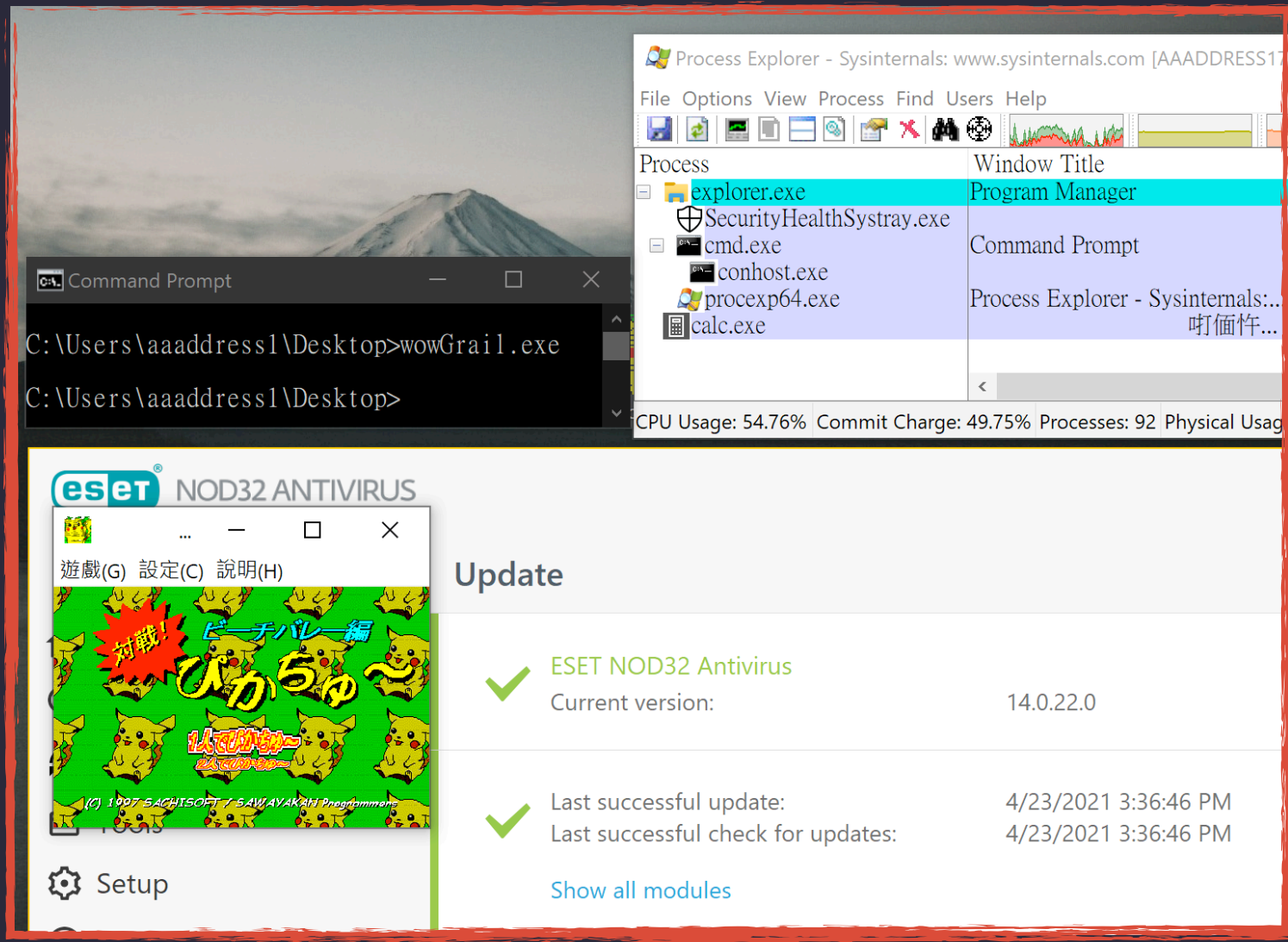
ntdll.dll

x86 Modules
x64 Modules

txOne networks

DEMO: wowGrail

New Path Back to The Heaven

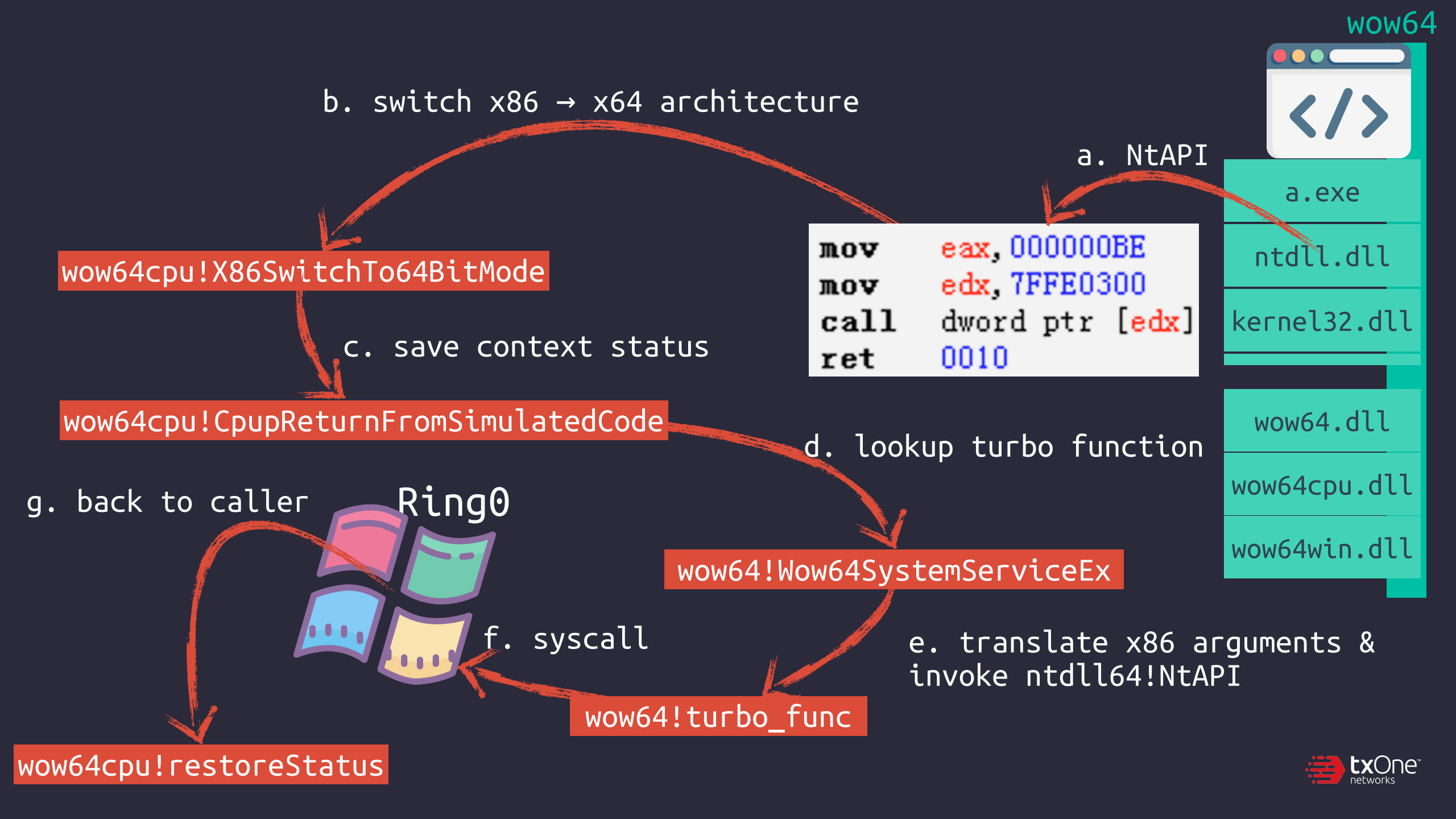


Process Hollowing & Bypass HIPS of NOD32

txOne networks

WOW64 Thread Snapshot

One Gadget To Take Over The 32-bit Hell


wow64
b. switch x86 → x64 architecture
a. NtAPI
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
mov eax, 000000BE
mov edx, 7FFE0300
call dword ptr [edx]
ret 0010
wow64cpu!X86SwitchTo64BitMode
c. save context status
wow64cpu!CpupReturnFromSimulatedCode
d. lookup turbo function
Ring0
g. back to caller
wow64!Wow64SystemServiceEx
f. syscall
e. translate x86 arguments &
invoke ntdll64!NtAPI
wow64!turbo_func
wow64cpu!restoreStatus
txOne networks

wow64

```
CpupReturnFromSimulatedCode:

                xchg    rsp, r14
                mov     r8d, [r14]
                add     r14, 4
                mov     [r13+3Ch], r8d
                mov     [r13+48h], r14d
                lea     r11, [r14+4]
                mov     [r13+20h], edi
                mov     [r13+24h], esi
                mov     [r13+28h], ebx
                mov     [r13+38h], ebp
                pushfq
                pop     r8
                mov     [r13+44h], r8d
```

a.exe

ntdll.dll

kernel32.dll

wow64.dll

wow64cpu.dll

wow64win.dll

b. switch x86 → x64 arch

wow64cpu!X86SwitchTo64BitMode

c. save context status

wow64cpu!CpupReturnFromSimulatedCode

d. lookup turbo function

wow64!Wow64SystemServiceEx

e. translate x86 arguments & invoke ntdll64!NtAPI

wow64!turbo_func

f. syscall

Ring0

g. back to caller

wow64cpu!restoreStatus

txOne networks

wow64

```
CpupReturnFromSimulatedCode:

                xchg    rsp, r14
                mov     r8d, [r14]
                add     r14, 4
                mov     [r13+3Ch], r8d
                mov     [r13+48h], r14d
                lea     r11, [r14+4]
                mov     [r13+20h], edi
                mov     [r13+24h], esi
                mov     [r13+28h], ebx
                mov     [r13+38h], ebp
                pushfq
                pop     r8
                mov     [r13+44h], r8d
```

a.exe

ntdll.dll

kernel32.dll

wow64.dll

wow64cpu.dll

wow64win.dll

b. switch x86 → x64 arch

wow64cpu!X86SwitchTo64BitMode

c. save context status

wow64cpu!CpupReturnFromSimulatedCode

d. lookup turbo function

g. back to caller

wow64cpu!restoreStatus

```
restoreStatus:                                  ; CODE XREF: RunSi
                btr     dword ptr [r13-80h], 0
                jb      short loc_7702168E
; 6:     __asm { jmp     fword ptr [r14] }
                mov     edi, [r13+20h]
                mov     esi, [r13+24h]
                mov     ebx, [r13+28h]
                mov     ebp, [r13+38h]
                mov     eax, [r13+34h]
                mov     r14, rsp
                mov     dword ptr [rsp+0A8h+var_A8+4], 23h
                mov     r8d, 2Bh ; '+'
                mov     ss, r8d
                mov     r9d, [r13+3Ch]
                mov     dword ptr [rsp+0A8h+var_A8], r9d
                mov     esp, [r13+48h]
                jmp     fword ptr [r14]
```

slate x86 arguments &
ntdll64!NtAPI

txOne networks

Recap

- Register r13 point to the 32-bit thread context used as snapshot status. It will be back up when the thread jumps from 32-bit to 64-bit, and reset from 64 bit back to 32-bit.
- when $RIP jump into wow64cpu!X86SwitchTo64BitMode, current thread context status will be saved into the 32-bit thread context dereferenced from r13.
- $RIP jump from 64-bit back to 32-bit, and the thread context will be restored from the 32-bit thread context.

Recap

- Register r13 point to the 32-bit thread context used as snapshot status. It will be back up when the thread jumps from 32-bit to 64-bit, and reset from 64 bit back to 32-bit.
- when $RIP jump into wow64cpu!X86SwitchTo64BitMode, current thread context status will be saved into the 32-bit thread context dereferenced from r13.
- $RIP jump from 64-bit back to 32-bit, and the thread context will be restored from the 32-bit thread context.

→ It can be used as one gadget to exploit the next 32-bit $RIP ;)

0x3000

nt!MiCreatePebOrTeb()

- 0x2000 or 0x3000 (it's up to WoW64)
- TEB64 + TEB32 + PEB64 + PEB32
- fixup TEB64: .self, .peb, .stack etc
- TEB64.ExceptionList always null
- fixup TEB32 based on TEB64
- TEB32.ExceptionList[0] = ffffffff

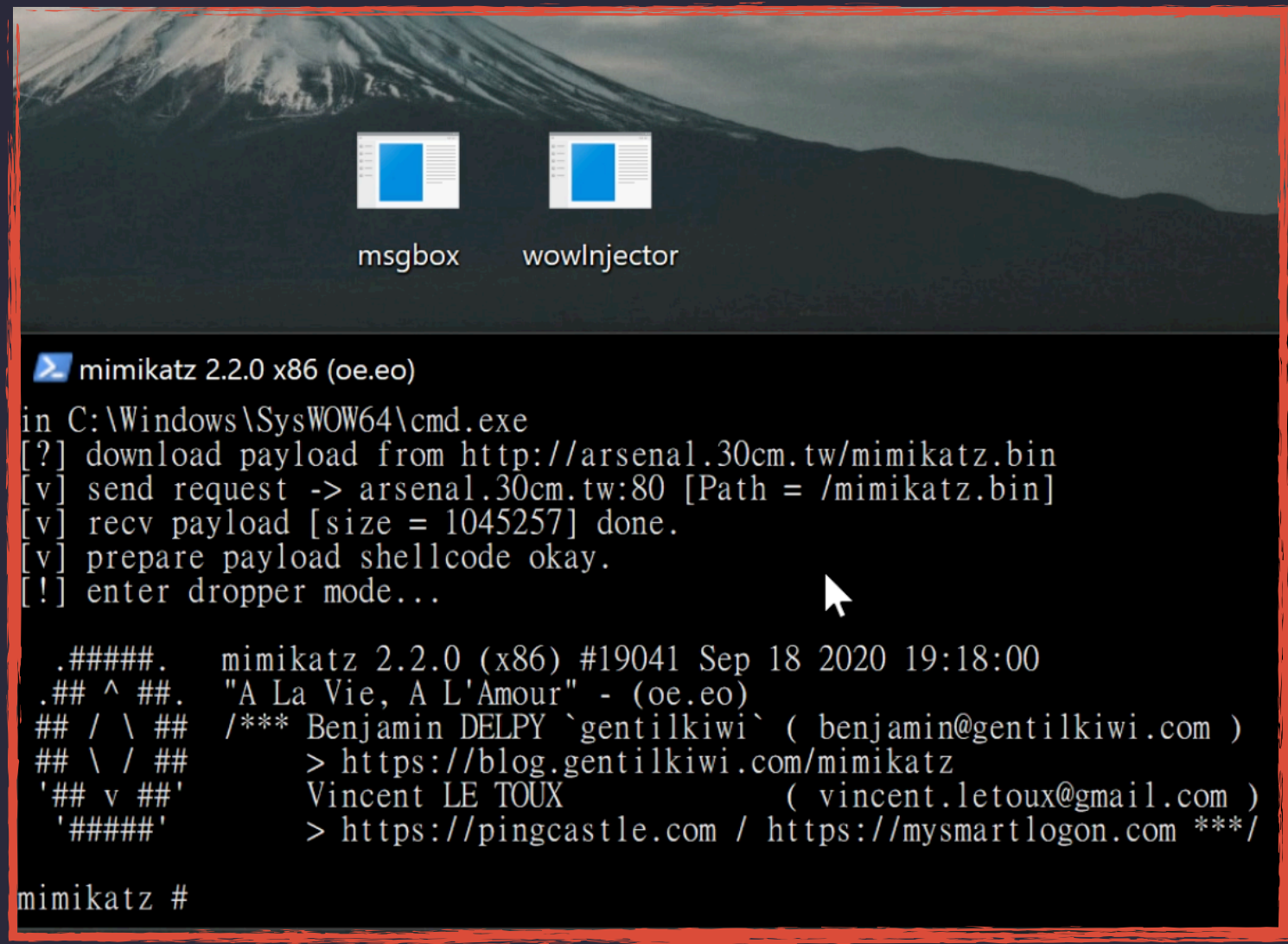
→ Leak any one of the 4 blocks,
and we can get the other 3 blocks.

txOne networks

DEMO: wowInjector

One Gadget Injection

to Take Over The 32-bit Hell



Process Inject & Bypass HIPS of AVAST

HITCON 2021

WORK FROM HOME, HACK INTO HOME

AGENDA & PANEL

全球聯防	人才培育	Cyber Physical System
金融聯防	CISO Round Table	5G Security
資安新創	Exploit	Talent Education
隱私與人權	Malware	Blue Team
企業經驗分享	IoT Hacking	AI Hacking

CALL FOR PAPER
2021.03.23(Tue) ~ 06.01(Tue)

https://hitcon.org

WORK FROM HOME, HACK INTO HOME

Hacking the Data Traversing in Reality-virtuality Hybrid World Recovery and Collaboration - Vaccine for CyberSecurity

格萊天漾大飯店 14-15 樓
2021.8.27(Fri) ~ 28(Sat)

早鳥售票：5/4 開始！

虛擬活動

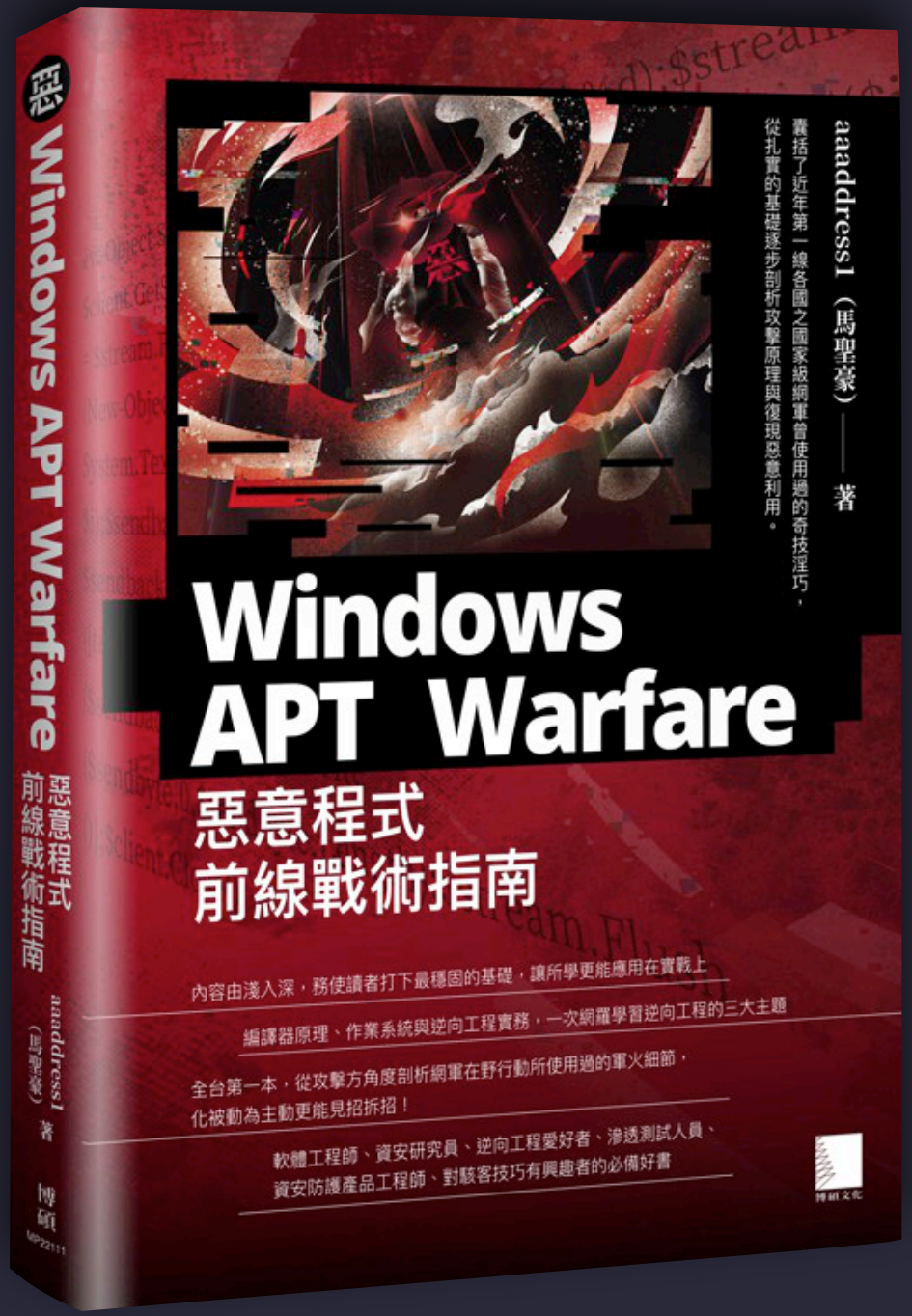
#HITCON Online
#駭客貓歷險記
#Session Live
#煉蠱

實體活動

- 駭客貓歷險記
- 實體駭客 Village
- Bounty House

CALL FOR SPONSORSHIPS

<sponsorship@hitcon.org>



Windows APT Warfare

惡意程式前線戰術指南

發售日 2021/05/05