

WoW64Jit:
from Reversing to Abuse the Heaven's Gate

aaaddress1@chroot.org

/?whoami



- Threat Researcher, TXOne Networks
- **Security Researcher** - chr0.ot
- **Speaker** - BlackHat, DEFCON, HITCON
- aaaddress1@chroot.org
- 30cm.tw
- Hao's Arsenal

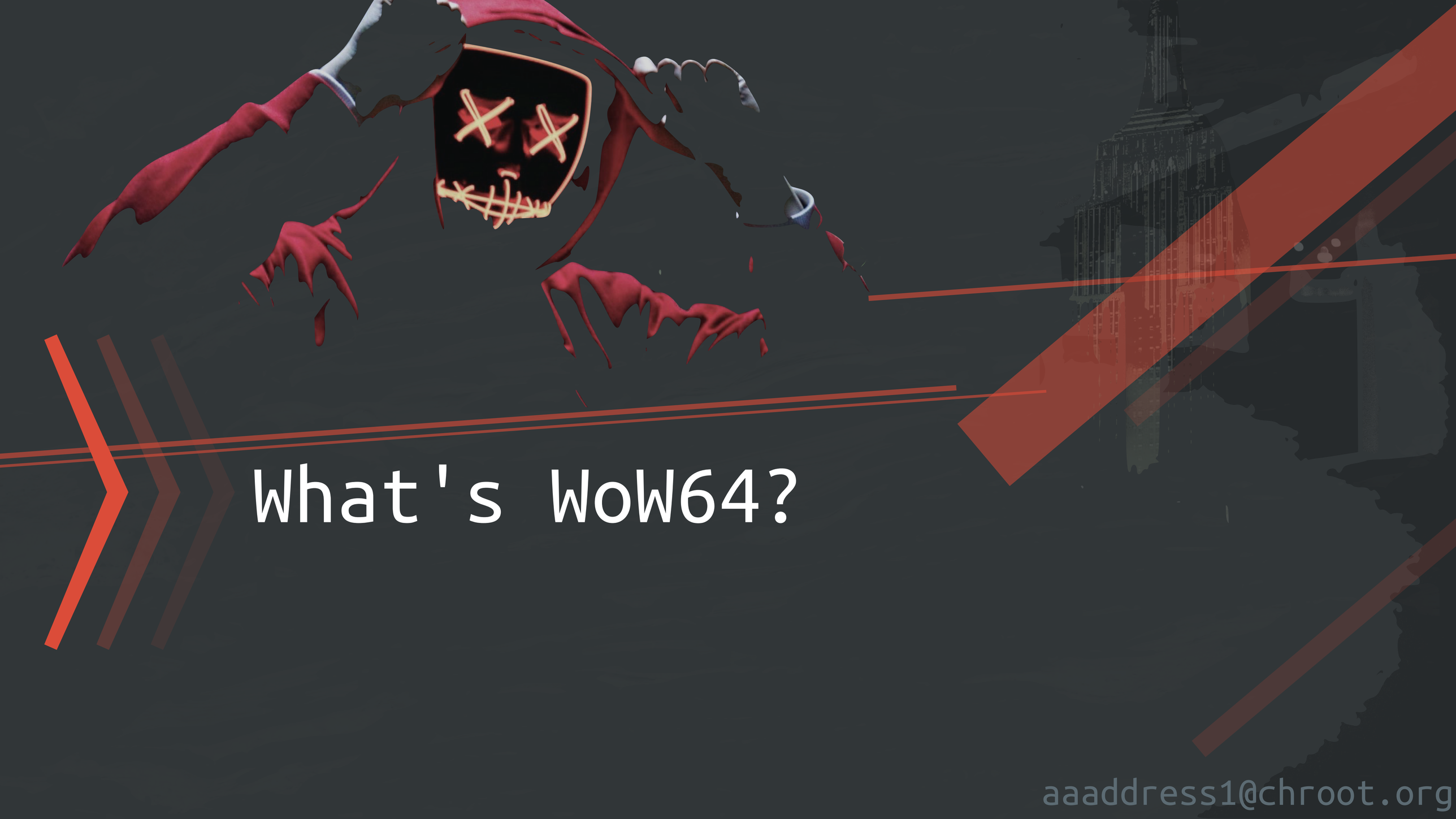


#Windows #Reversing #Pwn #Exploit

/?outline

1. What's WoW64?
2. Reversing the WoW64 Layer
   1. Loader & Process Init
   2. WoW64 Init
   3. Trampoline
   4. Translation
3. Abuse & Weaponize
   1. Abuse WoW64 Hook Trap
   2. Rebuild Translation Engine
   3. Call 32bit NtDLL API directly from WoW64 Layer
   4. from thread context injection to exploit Anti-Virus HIPS

aaaddress1@chroot.org



What's WoW64?

aaaddress1@chroot.org

/? WoW64

en.wikipedia.org/wiki/WoW64

In computing on Microsoft platforms, WoW64 (Windows 32-bit on Windows 64-bit) is a subsystem of the Windows operating system capable of running 32-bit applications on 64-bit Windows.

..., WoW64 aims to take care of many of the differences between 32-bit Windows and 64-bit Windows, particularly involving structural changes to Windows itself.

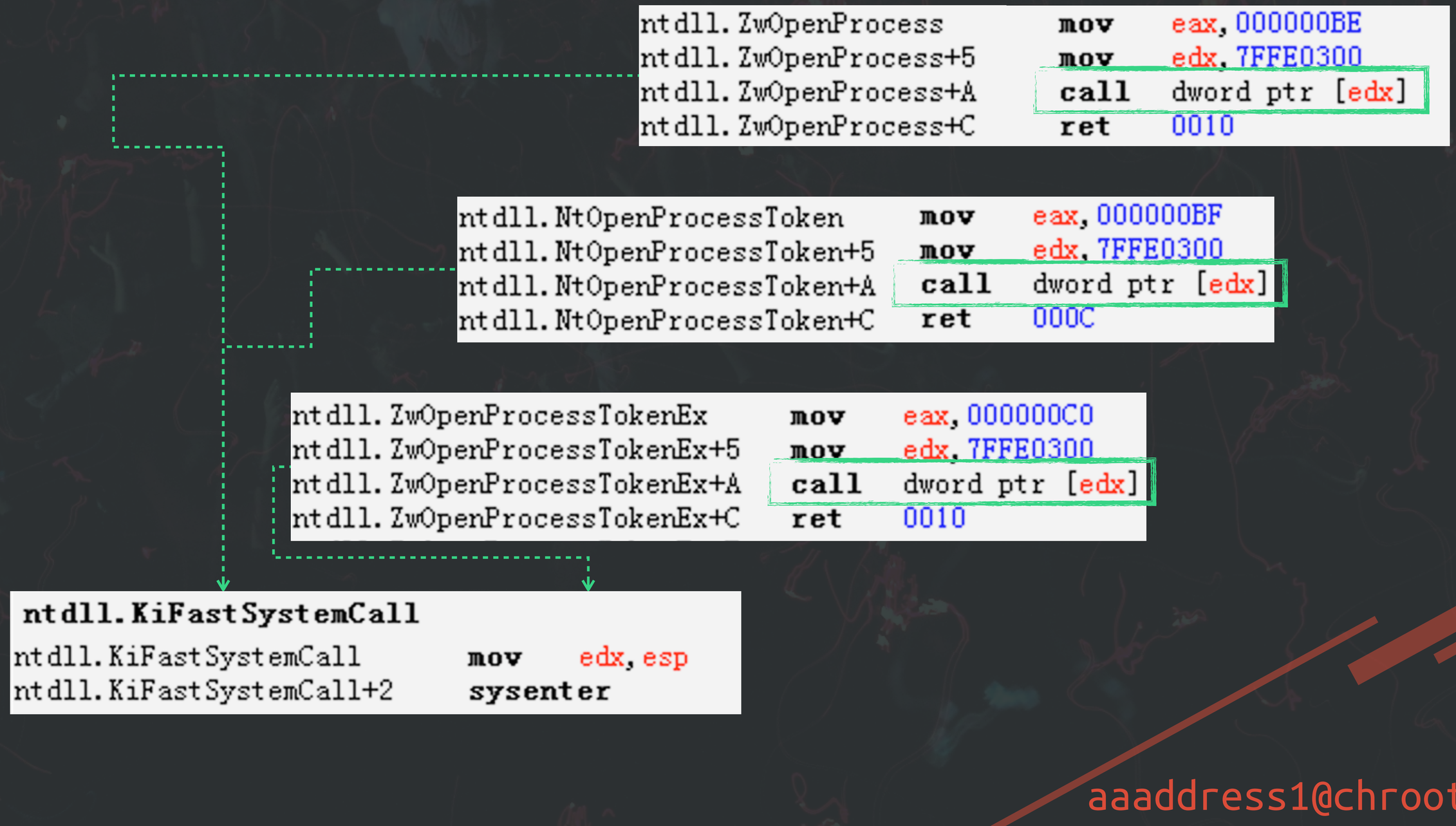
aaaddress1@chroot.org

#NtAPI

```
ntdll.ZwOpenProcess
ntdll.ZwOpenProcess          B8 BE000000    mov   eax,000000BE
ntdll.ZwOpenProcess+5        BA 0003FE7F    mov   edx,7FFE0300
ntdll.ZwOpenProcess+A        FF 12          call  dword ptr [edx]
ntdll.ZwOpenProcess+C        C2 1000        ret   0010
ntdll.ZwOpenProcess+F        90             nop
ntdll.NtOpenProcessToken
ntdll.NtOpenProcessToken     B8 BF000000    mov   eax,000000BF
ntdll.NtOpenProcessToken+5   BA 0003FE7F    mov   edx,7FFE0300
ntdll.NtOpenProcessToken+A   FF 12          call  dword ptr [edx]
ntdll.NtOpenProcessToken+C   C2 0C00        ret   000C
ntdll.NtOpenProcessToken+F   90             nop
ntdll.ZwOpenProcessTokenEx
ntdll.ZwOpenProcessTokenEx   B8 C0000000    mov   eax,000000C0
ntdll.ZwOpenProcessTokenEx+5 BA 0003FE7F    mov   edx,7FFE0300
ntdll.ZwOpenProcessTokenEx+A FF 12          call  dword ptr [edx]
ntdll.ZwOpenProcessTokenEx+C C2 1000        ret   0010
```

aaaddress1@chroot.org

#NtAPI


ntdll.ZwOpenProcess mov eax,000000BE
ntdll.ZwOpenProcess+5 mov edx,7FFE0300
ntdll.ZwOpenProcess+A call dword ptr [edx]
ntdll.ZwOpenProcess+C ret 0010
ntdll.NtOpenProcessToken mov eax,000000BF
ntdll.NtOpenProcessToken+5 mov edx,7FFE0300
ntdll.NtOpenProcessToken+A call dword ptr [edx]
ntdll.NtOpenProcessToken+C ret 000C
ntdll.ZwOpenProcessTokenEx mov eax,000000C0
ntdll.ZwOpenProcessTokenEx+5 mov edx,7FFE0300
ntdll.ZwOpenProcessTokenEx+A call dword ptr [edx]
ntdll.ZwOpenProcessTokenEx+C ret 0010
ntdll.KiFastSystemCall
ntdll.KiFastSystemCall mov edx,esp
ntdll.KiFastSystemCall+2 sysenter

#NtAPI

Process



a.exe

ntdll.dll

kernel32.dll

```
ntdll.ZwOpenProcess
ntdll.ZwOpenProcess      mov    eax,000000BE
ntdll.ZwOpenProcess+5    mov    edx,7FFE0300
ntdll.ZwOpenProcess+A    call   dword ptr [edx]
ntdll.ZwOpenProcess+C    ret    0010
```

```
ntdll.KiFastSystemCall
ntdll.KiFastSystemCall       mov    edx,esp
ntdll.KiFastSystemCall+2     sysenter
ntdll.KiFastSystemCallRet
ntdll.KiFastSystemCallRet    ret
```



Ring0

aaaddress1@chroot.org

#NtAPI

Process



a.exe

ntdll.dll

kernel32.dll

```
ntdll.ZwOpenProcess
ntdll.ZwOpenProcess      mov     eax,000000BE
ntdll.ZwOpenProcess+5    mov     edx,7FFE0300
ntdll.ZwOpenProcess+A    call    dword ptr [edx]
ntdll.ZwOpenProcess+C    ret     0010

ntdll.KiFastSystemCall
ntdll.KiFastSystemCall           mov     edx,esp
ntdll.KiFastSystemCall+2         sysenter

ntdll.KiFastSystemCallRet
ntdll.KiFastSystemCallRet        ret
```



Ring0

aaaddress1@chroot.org

#syscall

[Windows x86] ZwOpenProcess:

→ KiFastCall (32bit)

→ Kernel (32bit)

→ KiFastSystemCallRet (32bit)

→ ZwOpenProcess +C [Return]

aaaddress1@chroot.org

#syscall

[Windows x64] ZwOpenProcess:

→ KiFastCall (64bit)

→ Kernel (64bit)

→ KiFastSystemCallRet (64bit)

→ ZwOpenProcess +C [Return]

aaaddress1@chroot.org

#syscall

[Windows x86 on x64] ZwOpenProcess:

→ KiFastCall (32bit)

→ Kernel (64bit)

→ KiFastSystemCallRet (32bit)

→ ZwOpenProcess +C (32bit)



aaaddress1@chroot.org

#syscall

[WoW64] ZwOpenProcess:

→ KiFastCall (32bit)

→ WoW64 Translation Entry (32+64)

→ Kernel (64bit)

→ WoW64 Translation Exit (32+64)

→ KiFastSystemCallRet (32bit)

→ ZwOpenProcess +C (32bit)

aaaddress1@chroot.org



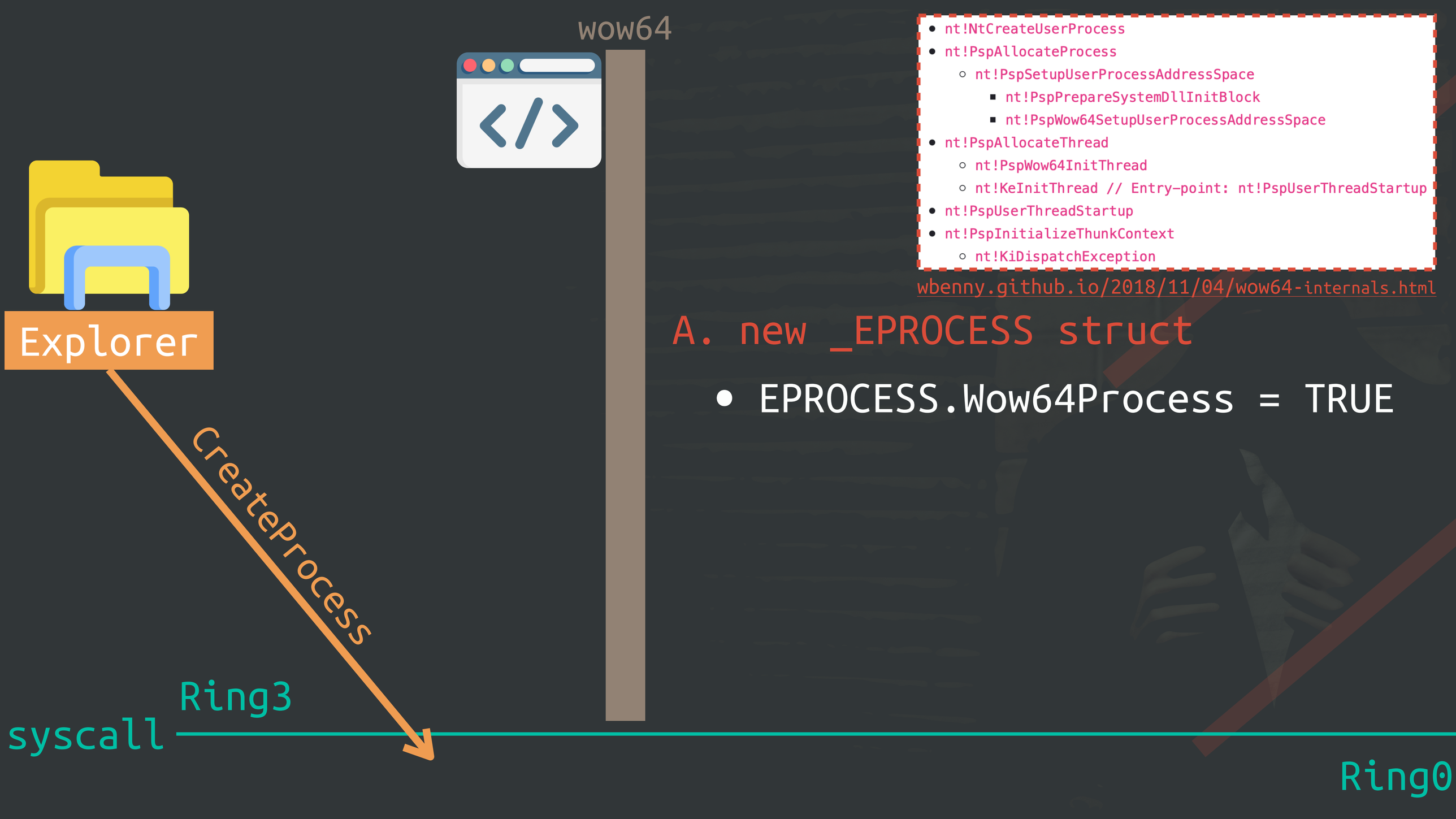
Reversing the WoW64 Layer

aaaddress1@chroot.org



WoW64 Process Init

aaaddress1@chroot.org

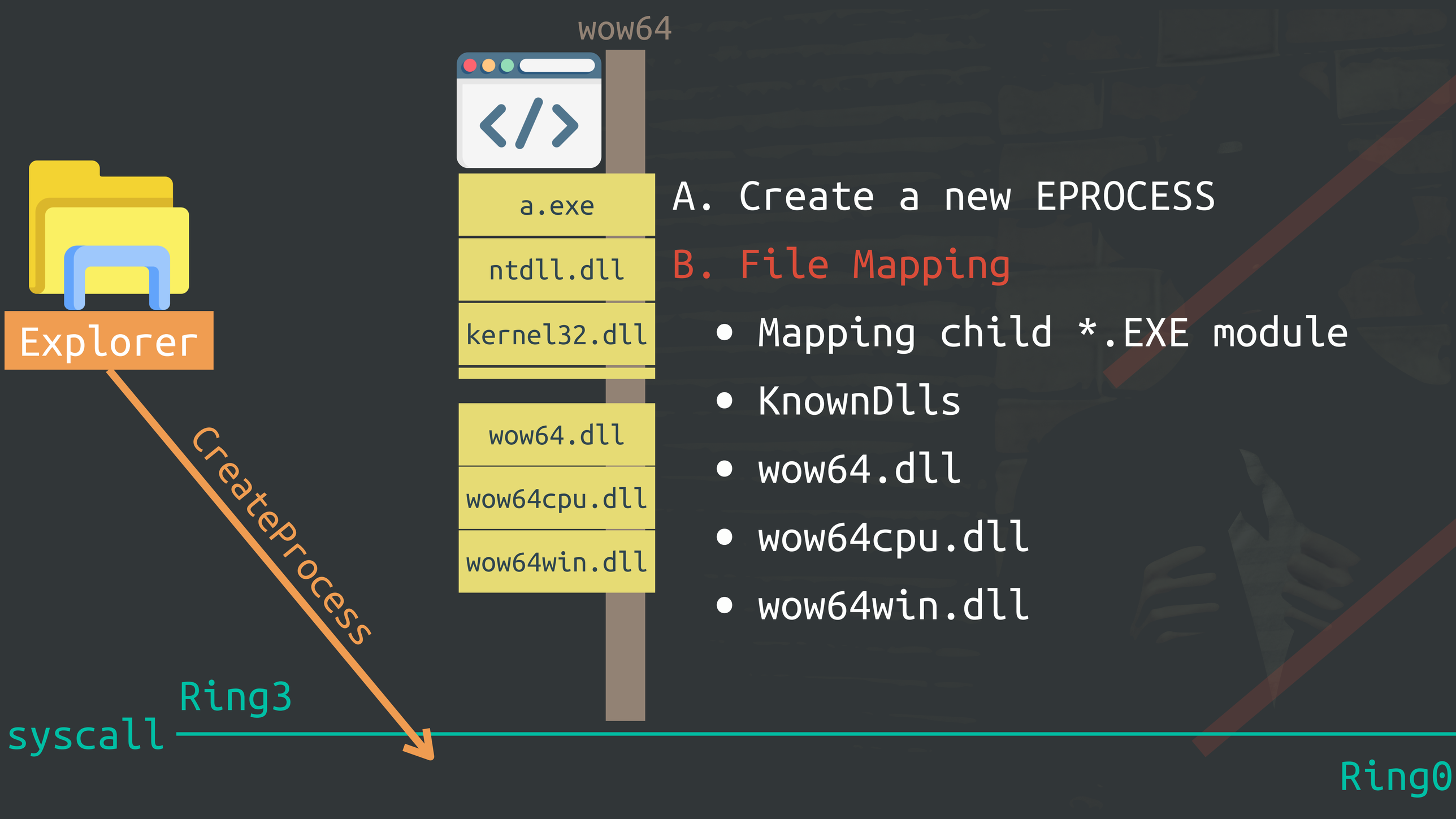


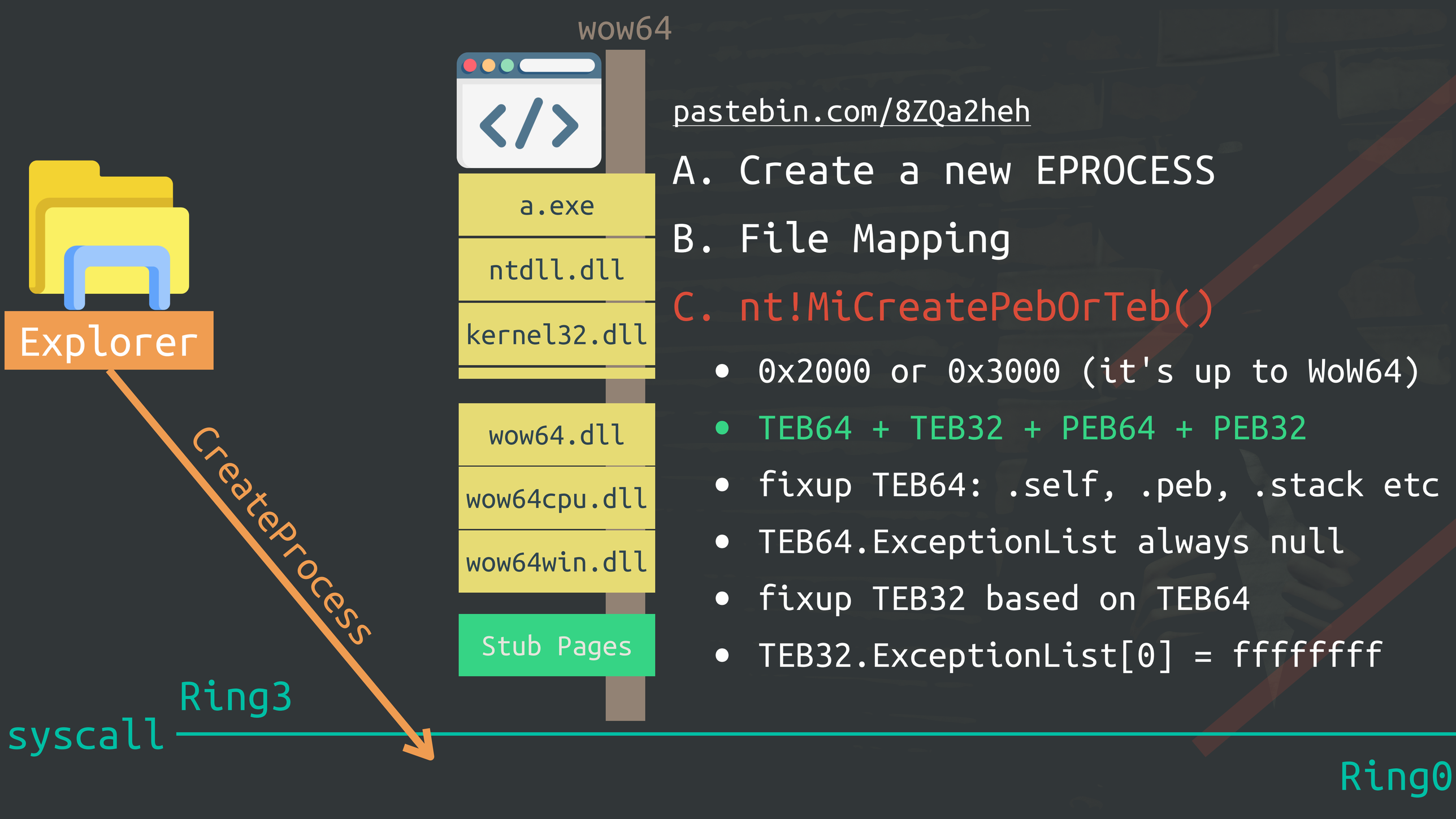
- nt!NtCreateUserProcess
- nt!PspAllocateProcess
  - nt!PspSetupUserProcessAddressSpace
    - nt!PspPrepareSystemDllInitBlock
    - nt!PspWow64SetupUserProcessAddressSpace
- nt!PspAllocateThread
  - nt!PspWow64InitThread
  - nt!KeInitThread // Entry-point: nt!PspUserThreadStartup
- nt!PspUserThreadStartup
- nt!PspInitializeThunkContext
  - nt!KiDispatchException

wbenny.github.io/2018/11/04/wow64-internals.html

A. new _EPROCESS struct

- EPROCESS.Wow64Process = TRUE


wow64
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
A. Create a new EPROCESS
B. File Mapping
• Mapping child *.EXE module
• KnownDlls
• wow64.dll
• wow64cpu.dll
• wow64win.dll
Explorer
CreateProcess
Ring3
syscall
Ring0



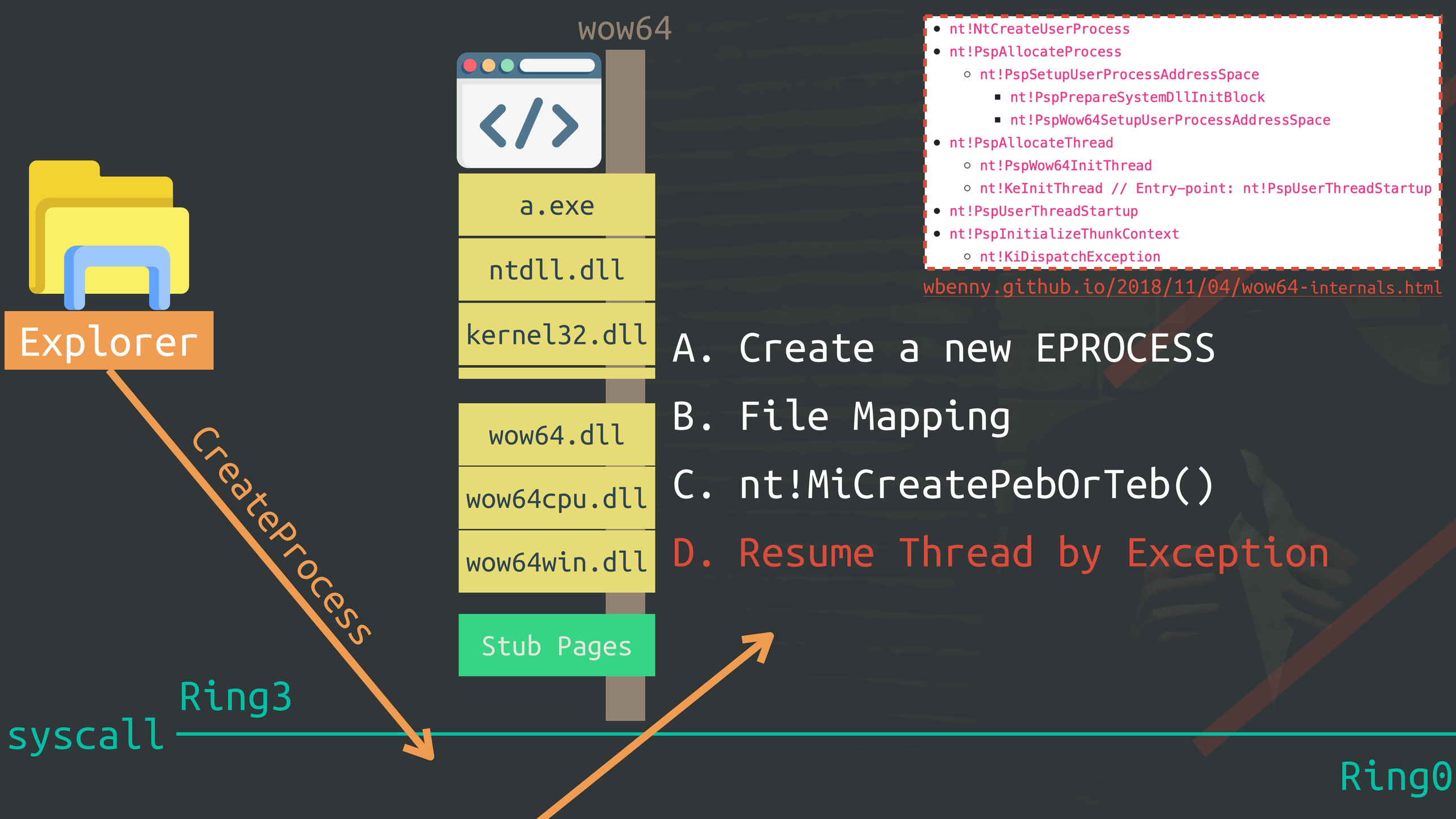
pastebin.com/8ZQa2heh

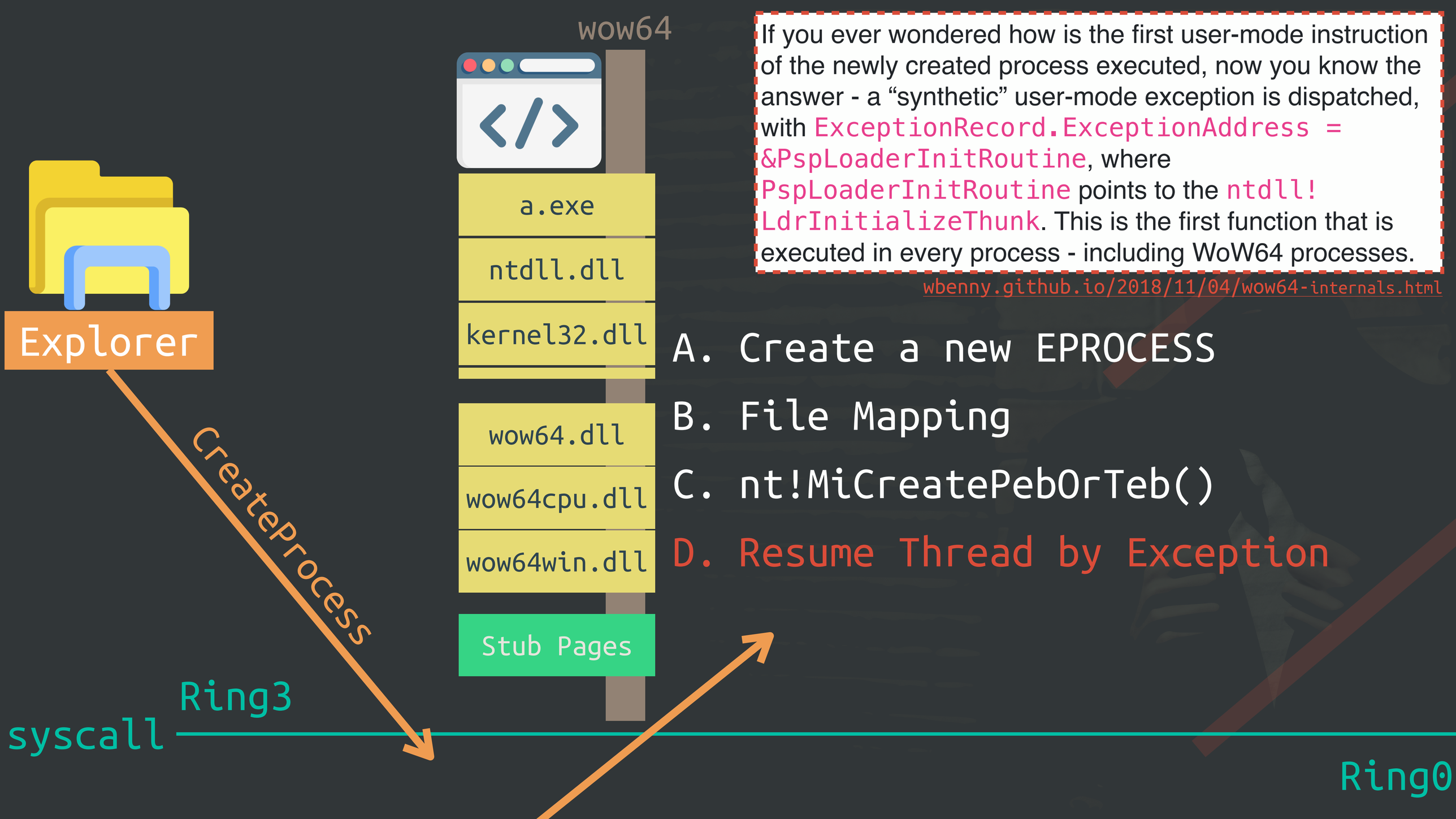
A. Create a new EPROCESS

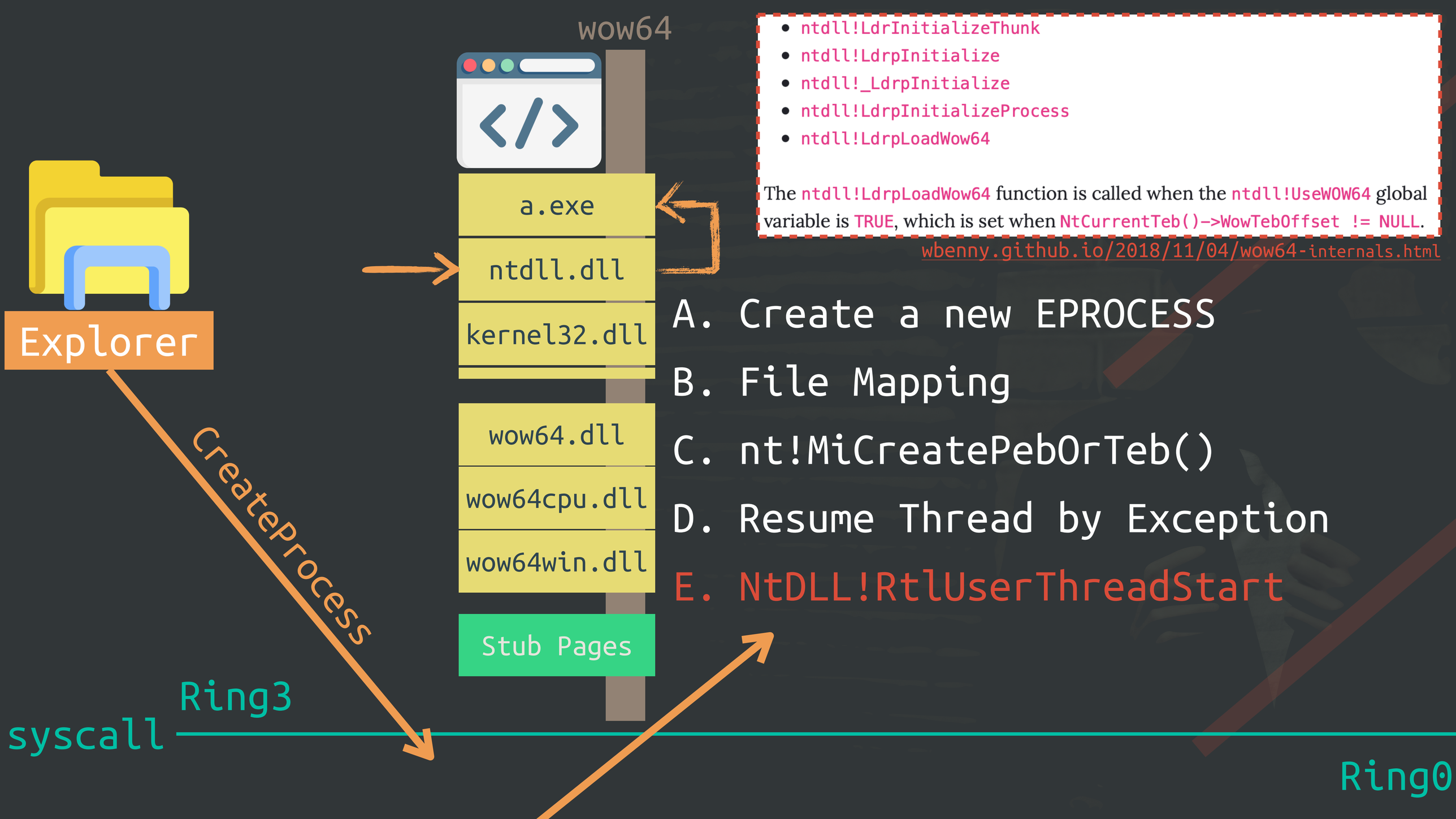
B. File Mapping

C. 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


wow64
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
Stub Pages
Explorer
CreateProcess
Ring3
syscall
Ring0
A. Create a new EPROCESS
B. File Mapping
C. nt!MiCreatePebOrTeb()
D. Resume Thread by Exception
nt!NtCreateUserProcess
nt!PspAllocateProcess
nt!PspSetupUserProcessAddressSpace
nt!PspPrepareSystemDllInitBlock
nt!PspWow64SetupUserProcessAddressSpace
nt!PspAllocateThread
nt!PspWow64InitThread
nt!KeInitThread // Entry-point: nt!PspUserThreadStartup
nt!PspUserThreadStartup
nt!PspInitializeThunkContext
nt!KiDispatchException
wbenny.github.io/2018/11/04/wow64-internals.html


wow64
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
Stub Pages
Explorer
CreateProcess
Ring3
syscall
Ring0
A. Create a new EPROCESS
B. File Mapping
C. nt!MiCreatePebOrTeb()
D. Resume Thread by Exception
If you ever wondered how is the first user-mode instruction of the newly created process executed, now you know the answer - a “synthetic” user-mode exception is dispatched, with ExceptionRecord.ExceptionAddress = &PspLoaderInitRoutine, where PspLoaderInitRoutine points to the ntdll!LdrInitializeThunk. This is the first function that is executed in every process - including WoW64 processes.
wbenny.github.io/2018/11/04/wow64-internals.html

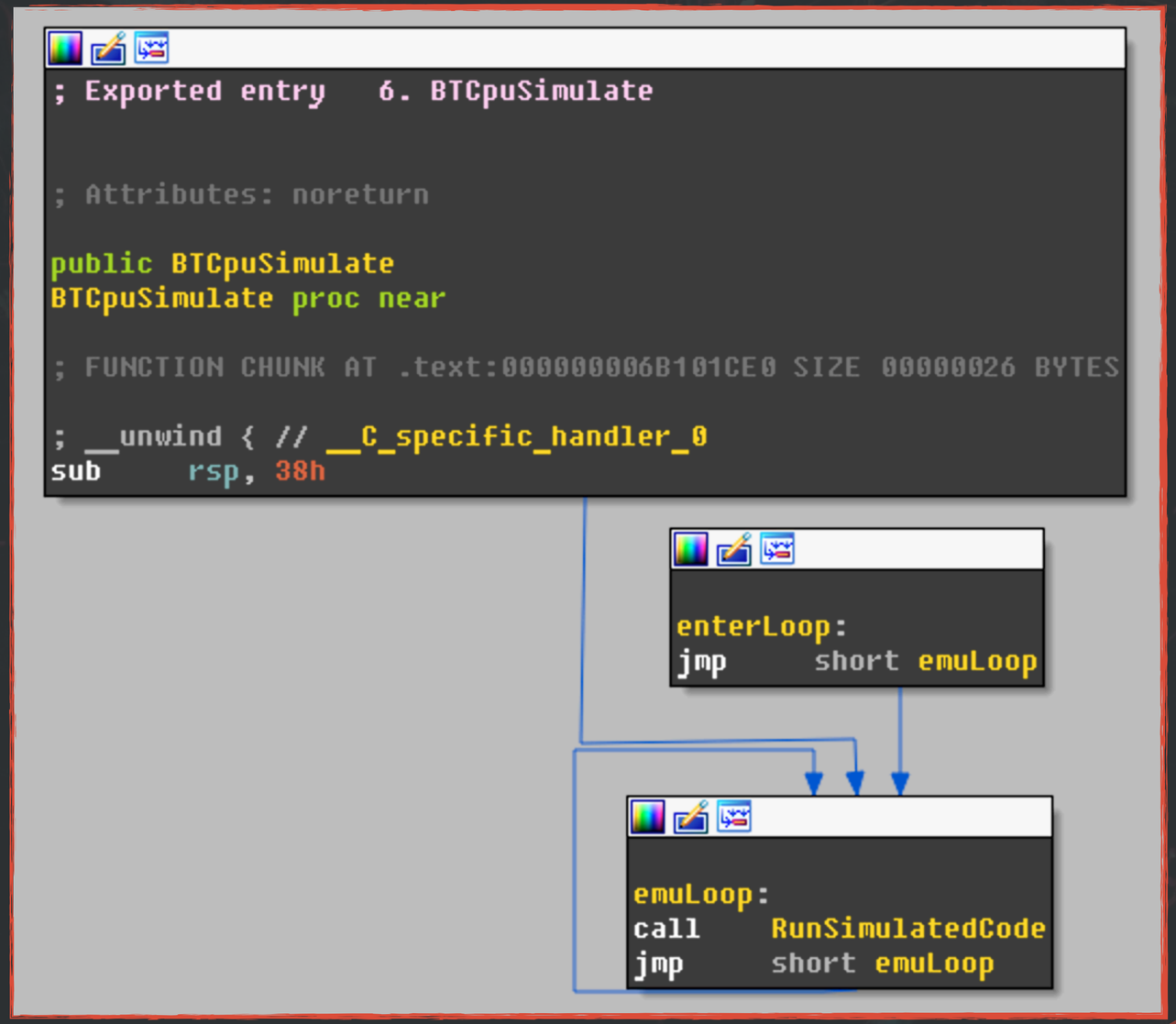

wow64
a.exe
ntdll.dll
kernel32.dll
wow64.dll
wow64cpu.dll
wow64win.dll
Stub Pages
Explorer
CreateProcess
Ring3
syscall
Ring0
ntdll!LdrInitializeThunk
ntdll!LdrpInitialize
ntdll!_LdrpInitialize
ntdll!LdrpInitializeProcess
ntdll!LdrpLoadWow64
The ntdll!LdrpLoadWow64 function is called when the ntdll!UseWOW64 global variable is TRUE, which is set when NtCurrentTeb()->WowTebOffset != NULL.
wbenny.github.io/2018/11/04/wow64-internals.html
A. Create a new EPROCESS
B. File Mapping
C. nt!MiCreatePebOrTeb()
D. Resume Thread by Exception
E. NtDLL!RtlUserThreadStart



WoW64 Layer Init

aaaddress1@chroot.org

#BTCpuSimulate



wow64cpu!BtCpuSimulate

aaaddress1@chroot.org

#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

aaaddress1@chroot.org

#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 XRE
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



aaaddress1@chroot.org



NtAPI Trampoline

aaaddress1@chroot.org

#Trampoline

Native

```
0:013> u ntdll!NtResumeThread
ntdll!NtResumeThread:
00007ff9`1d72c7f0 4c8bd1          mov     r10,rcx
00007ff9`1d72c7f3 b852000000      mov     eax,52h
00007ff9`1d72c7f8 f604250803fe7f01 test    byte ptr [SharedUserData+0x308 (00000000`7ffe0308)],1
00007ff9`1d72c800 7503            jne     ntdll!NtResumeThread+0x15 (00007ff9`1d72c805)
00007ff9`1d72c802 0f05            syscall
00007ff9`1d72c804 c3              ret
00007ff9`1d72c805 cd2e            int     2Eh
00007ff9`1d72c807 c3              ret
```

WoW64

```
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
```


aaaddress1@chroot.org

#Trampoline

```
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

aaaddress1@chroot.org

#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]
```

1. save current 32bit context status, stack, and caller retAddr

2. simulate 32bit behavior to 64bit KiFastCall by wow64SystemServiceEx

3. save NTSTATUS into WoW64 Thread CONTEXT.eax

```
                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
```


aaaddress1@chroot.org

#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]
```

```
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
```


aaaddress1@chroot.org



Translation Engine

aaaddress1@chroot.org

#Translation

wow64!Wow64SystemServiceEx

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

```
ntdll.ZwOpenProcess
ntdll.ZwOpenProcess     mov     eax,000000BE
ntdll.ZwOpenProcess+5   mov     edx,7FFE0300
ntdll.ZwOpenProcess+A   call    dword ptr [edx]
ntdll.ZwOpenProcess+C   ret     0010
```


aaaddress1@chroot.org

#Translation

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];
  logData.ServiceTable  = (*&syscall >> 12) & 3;
  logData.ServiceNumber = syscall.SystemCallNumber & 0xFFF;
  TEB64->LastErrorValue = TEB32->LastErrorValue;
```

```
if ( ptrWow64LogSystemService ) {
  logData.Argumentos = &args;
  logData.postCall = 0;
  ptrWow64LogSystemService(logData);

  ret = ptrNtAPI64_TurboFunc(args);

  logData.postCall = 1;
  logData.NTSTATUS = ret;
  ptrWow64LogSystemService(logData);
}
```


aaaddress1@chroot.org

#Translation

wow64!Wow64SystemServiceEx

```
  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;
}
```

```
if ( ptrWow64LogSystemService ) {
  logData.Argumentos = &args;
  logData.postCall = 0;
  ptrWow64LogSystemService(logData);

  ret = ptrNtAPI64_TurboFunc(args);

  logData.postCall = 1;
  logData.NTSTATUS = ret;
  ptrWow64LogSystemService(logData);
}
```


aaaddress1@chroot.org



Back to Heaven's Gate

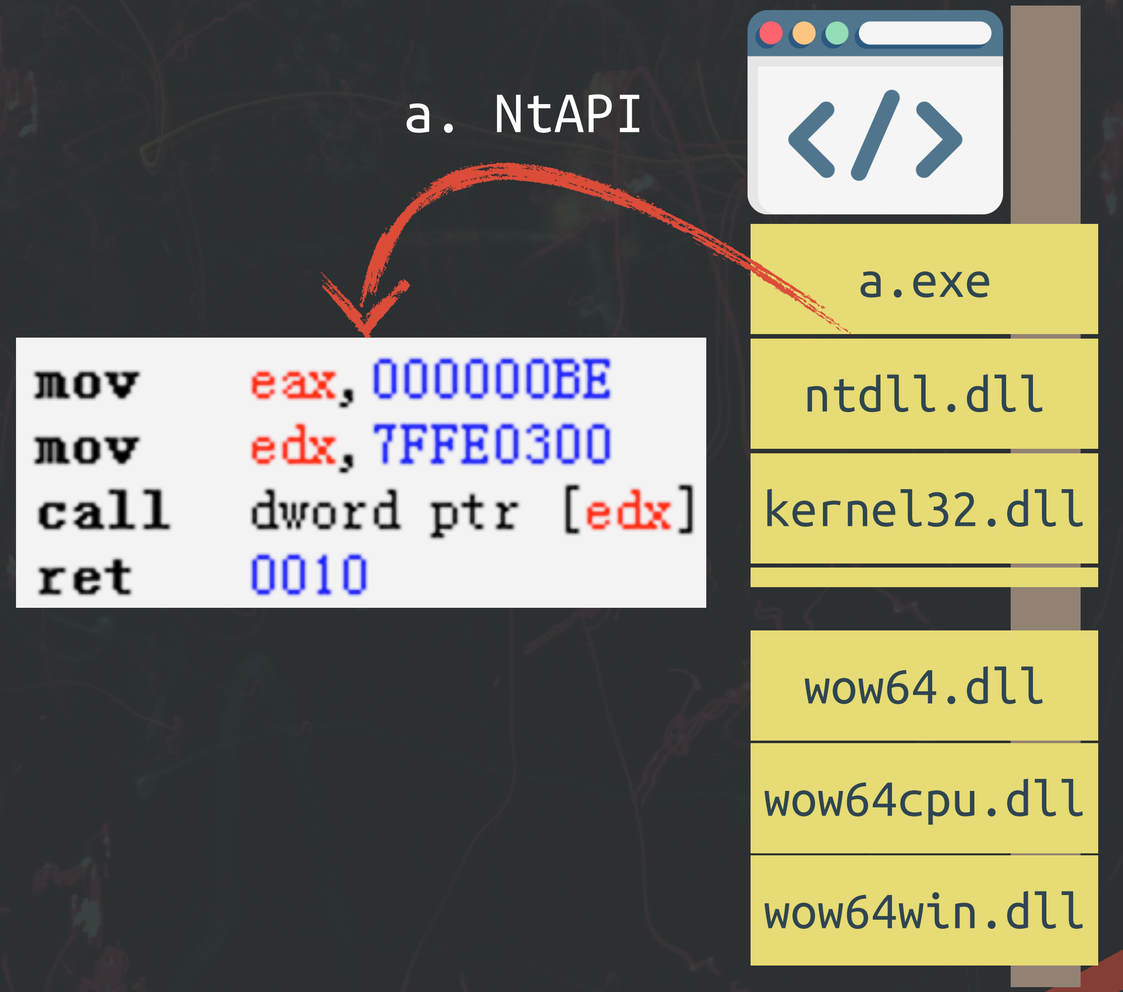
aaaddress1@chroot.org



Overview

aaaddress1@chroot.org

#WoW64



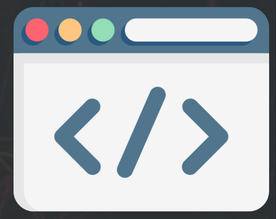
aaaddress1@chroot.org

#WoW64

b. switch x86 → x64 architecture

a. NtAPI

wow64



a.exe

ntdll.dll

kernel32.dll

wow64.dll

wow64cpu.dll

wow64win.dll

wow64cpu!X86SwitchTo64BitMode

```
mov     eax, 000000BE
mov     edx, 7FFE0300
call    dword ptr [edx]
ret     0010
```

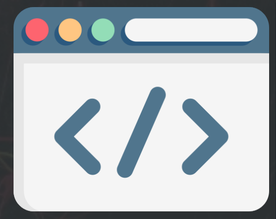
aaaddress1@chroot.org

#WoW64

wow64

b. switch x86 → x64 architecture

a. NtAPI



a.exe

ntdll.dll

kernel32.dll

wow64cpu!X86SwitchTo64BitMode

```
mov     eax, 000000BE
mov     edx, 7FFE0300
call    dword ptr [edx]
ret     0010
```

c. save context status

wow64cpu!CpupReturnFromSimulatedCode

wow64.dll

wow64cpu.dll

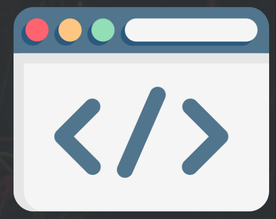
wow64win.dll

aaaddress1@chroot.org

#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
```

d. lookup turbo function

wow64cpu!X86SwitchTo64BitMode

c. save context status

wow64cpu!CpupReturnFromSimulatedCode

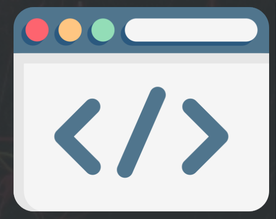
wow64!Wow64SystemServiceEx

aaaddress1@chroot.org

#WoW64

b. switch x86 → x64 architecture

a. NtAPI



a.exe

ntdll.dll

kernel32.dll

wow64.dll

wow64cpu.dll

wow64win.dll

wow64cpu!X86SwitchTo64BitMode

```
mov     eax,000000BE
mov     edx,7FFE0300
call    dword ptr [edx]
ret     0010
```

d. lookup turbo function

c. save context status

wow64cpu!CpupReturnFromSimulatedCode

wow64!Wow64SystemServiceEx

e. translate x86 arguments &
invoke ntdll64!NtAPI

wow64!turbo_func

aaaddress1@chroot.org

#WoW64

wow64

b. switch x86 → x64 architecture

a. NtAPI



a.exe

ntdll.dll

kernel32.dll

wow64cpu!X86SwitchTo64BitMode

```
mov     eax, 000000BE
mov     edx, 7FFE0300
call    dword ptr [edx]
ret     0010
```

c. save context status

d. lookup turbo function

wow64cpu!CpupReturnFromSimulatedCode

wow64.dll

wow64cpu.dll

wow64win.dll

Ring0



wow64!Wow64SystemServiceEx

e. translate x86 arguments &
invoke ntdll64!NtAPI

f. syscall

wow64!turbo_func

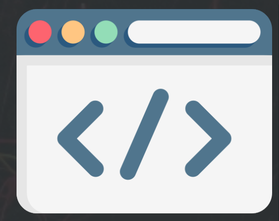
aaaddress1@chroot.org

#WoW64

wow64

b. switch x86 → x64 architecture

a. NtAPI



a.exe

ntdll.dll

kernel32.dll

wow64.dll

wow64cpu.dll

wow64win.dll

wow64cpu!X86SwitchTo64BitMode

```
mov     eax,000000BE
mov     edx,7FFE0300
call    dword ptr [edx]
ret     0010
```

d. lookup turbo function

c. save context status

wow64cpu!CpupReturnFromSimulatedCode

g. back to caller

Ring0



wow64!Wow64SystemServiceEx

e. translate x86 arguments &
invoke ntdll64!NtAPI

f. syscall

wow64!turbo_func

wow64cpu!restoreStatus

aaaddress1@chroot.org



Rewolf: Heaven's Gate

aaaddress1@chroot.org

#WoW64

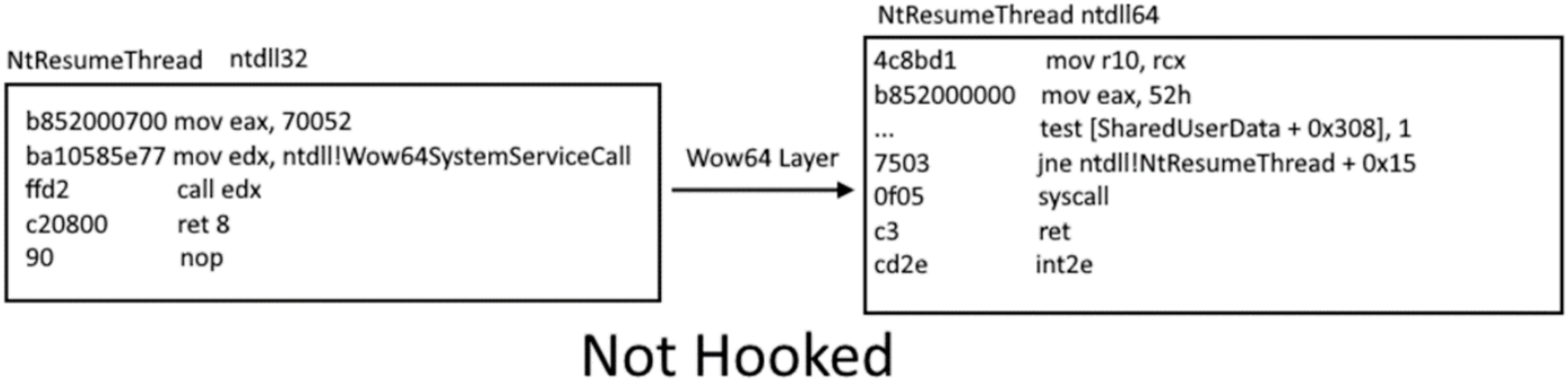
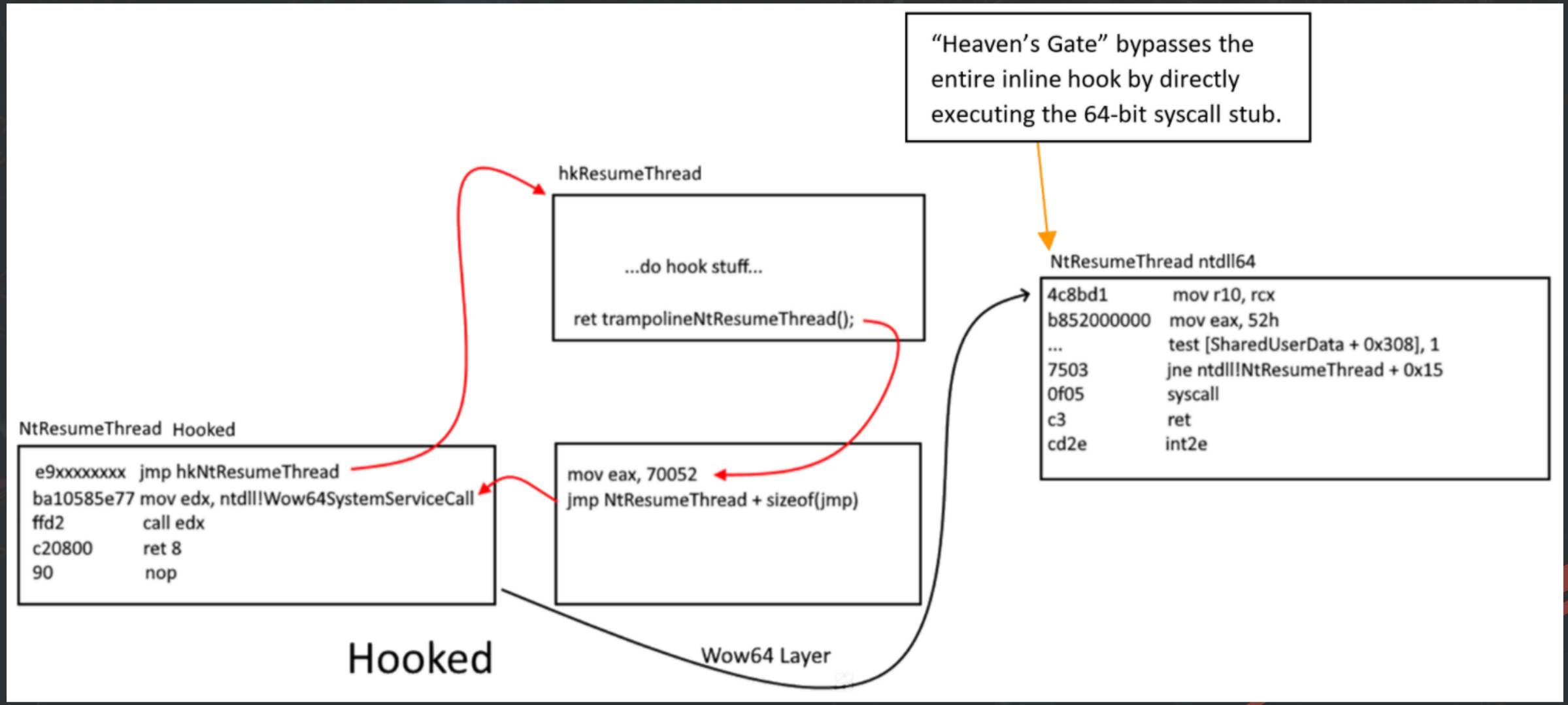
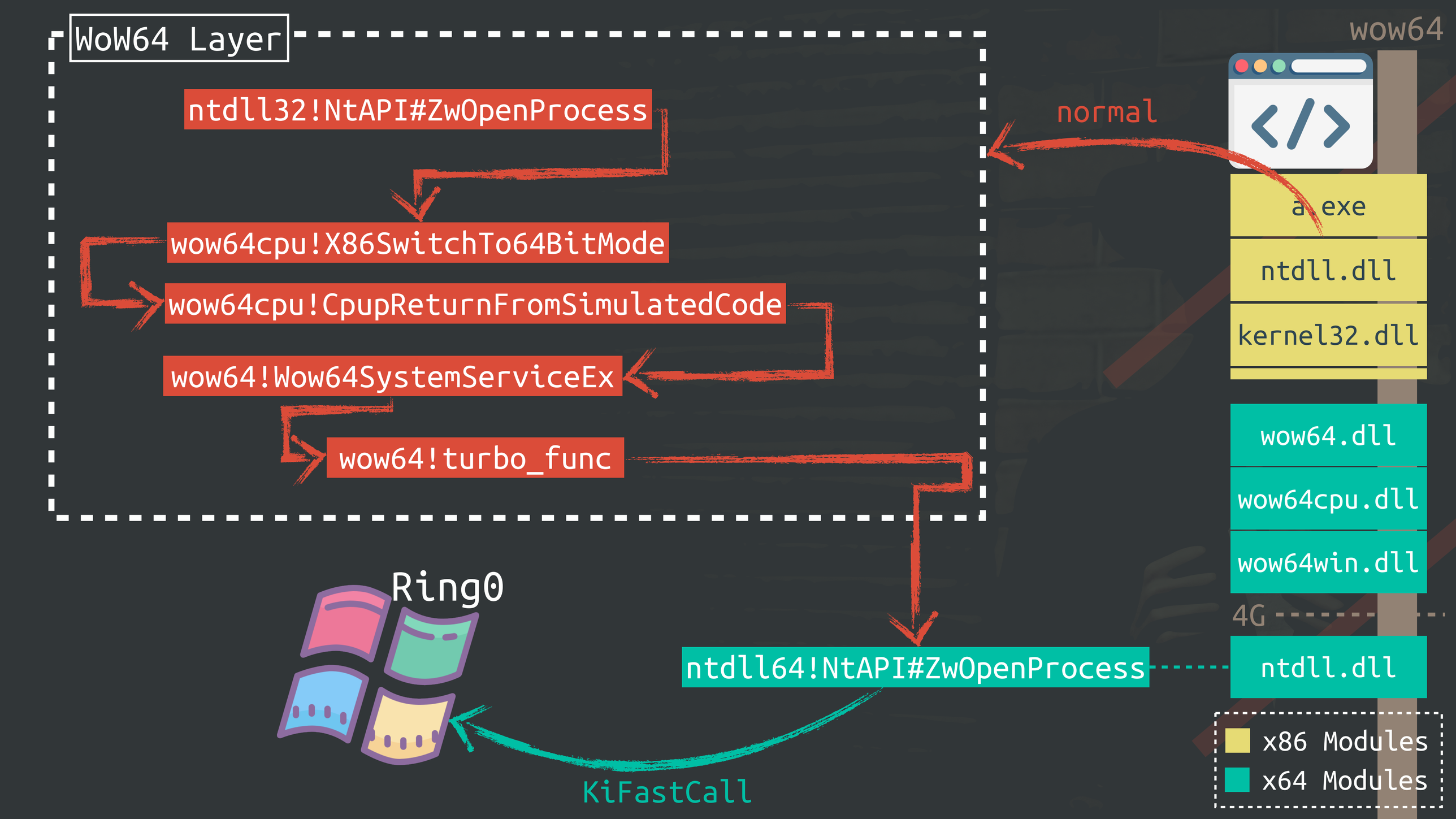


Figure 14: NtResumeThread transitioning through the WOW64 layer

#WoW64



aaaddress1@chroot.org


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
ntdll64!NtAPI#ZwOpenProcess
Ring0
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 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

aaaddress1@chroot.org

#Heaven's Gate

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
- 2020 - wow64Jit - from Reversing to abusing the Heaven's Gate

aaaddress1@chroot.org



Weaponize & Demo

aaaddress1@chroot.org

/? docs



- 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





Thanks !

aaaddress1@chroot.org

Github

Slide

Facebook

@aaaddress1