ENSILO

BYPASSING USER-MODE HOOKS

ANALYZING MALWARE EVASION TREND

FIRST Tel Aviv 2019

ABOUT US

- Omri Misgav
  - Security Research Team Leader @ enSilo
  - Reverse Engineering, OS internals

- Udi Yavo
  - CTO & Co-Founder @ enSilo
  - Former CTO, Rafael Cyber Security Division
  - Past speaker in Blackhat and RSA

- Our technical blog: BreakingMalware.com

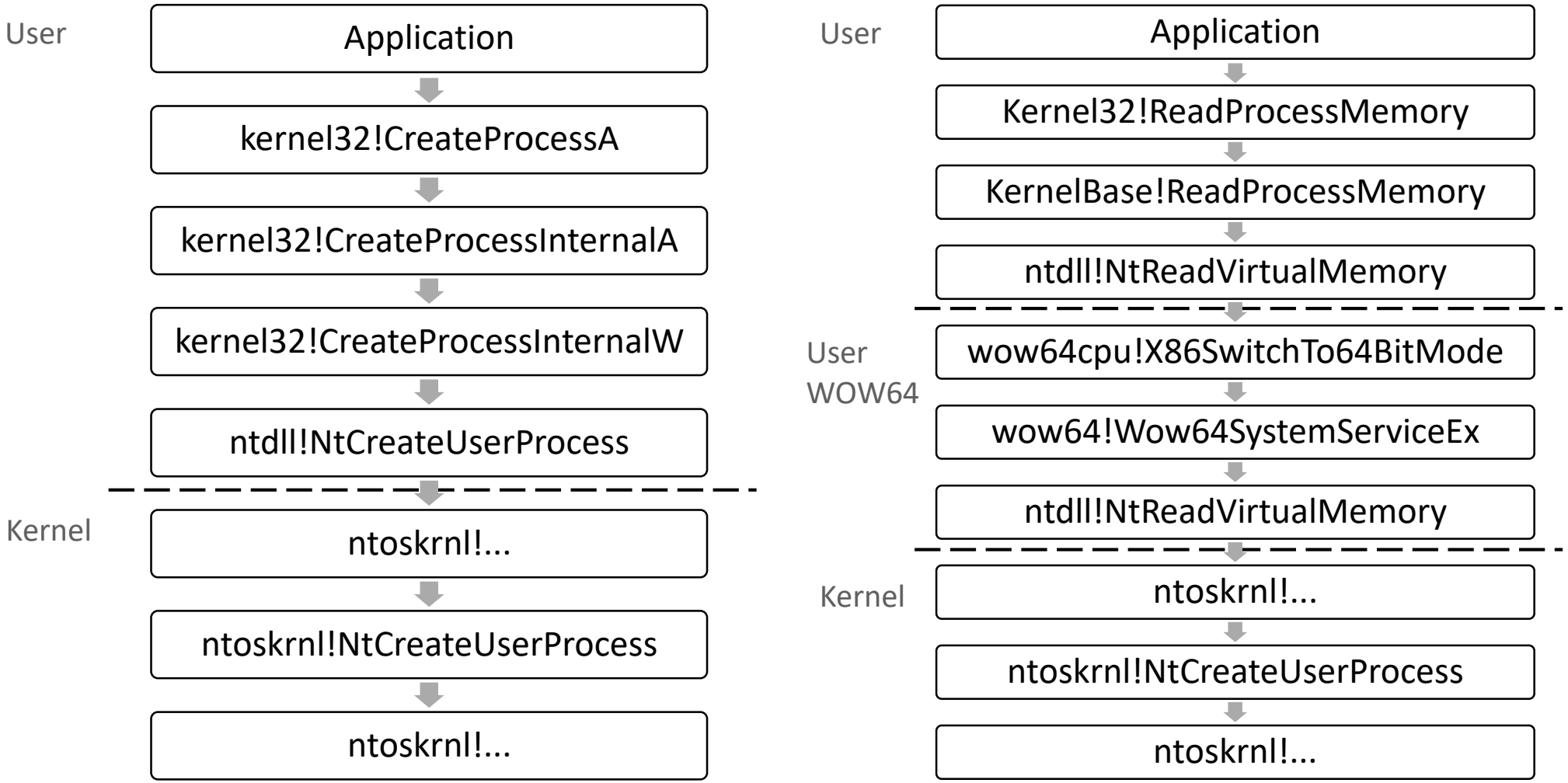
AGENDA

- Intro and background
- Bypass techniques analysis
  - Secondary DLL mapping
  - Direct system call invocation
  - Code splicing
- Comparison and takeaways

INTRO

- Hooking is used to intercept function calls in order to alter or augment their behavior
- User-mode hooks are used in many security products and tools
  - AVs\NGAVs
  - EDRs
  - Sandboxes
  - DLPs
  - And more…
- Why?
  - Stable, simple (nevertheless, not without faults)
  - Lack of Patch Protection
  - Full context
- Bypasses exist for a very long time
- Last ~1.5 years there's an increasing number of reports (malware and pentesters)

HOOKING BACKGROUND



ENSILO

HOOKING BACKGROUND

Inline Hooks

- Control flow instructions

```
function_A:
0x401000:  55              push    ebp
0x401001:  89 e5           mov     ebp, esp
0x401003:  83 ec 40        sub     esp, 0x40
0x401006:  50              push    eax
0x401007:  8b 44 24 0c     mov     eax, [esp + 0xc]
0x40100a:  …
```

```
function_A:
0x401000:  e9 00 20 40 00  jmp     hook_A
0x401005:  89              nop
0x401006:  50              push    eax
0x401007:  8b 44 24 0c     mov     eax, [esp + 0xc]
0x40100a:  …
```

```
hook_A:
0x402000:  55              push    ebp
0x402001:  89 e5           mov     ebp, esp
0x402003:  83 ec 40        sub     esp, 0x40
0x402006:  e9 06 10 40 00  jmp     function_A + 6
```

- Generating exceptions

```
function_B:
0x403000:  55              push    ebp
0x403001:  89 e5           mov     ebp, esp
0x403003:  83 ec 50        sub     esp, 0x50
0x403006:  …
```

```
function_B:
0x403000:  cc              int     3
0x403001:  89 e5           mov     ebp, esp
0x403003:  83 ec 50        sub     esp, 0x50
0x403006:  …
```


ENSILO

ENSILO

BYPASS TECHNIQUES ANALYSIS

Secondary DLL mapping

BYPASS TECHNIQUES
Manually Load DLL From Disk

- ReadFile() + Reflective Loading

- FormBook, reported by FireEye
- Infostealer
- Referred to as "Lagos Island method“
- Loads ntdll.dll
  - Code injection and Process Hollowing
  - File system and registry access

BYPASS TECHNIQUES

Manually Load DLL From Disk

```
ntdll!NtClose:
775b54c8 b832000000      mov     eax,32h
775b54cd ba0003fe7f      mov     edx,offset SharedUserData!SystemCallStub
775b54d2 ff12            call    dword ptr [edx]
775b54d4 c20400          ret     4
0:000> u poi(SharedUserData!SystemCallStub) L3
ntdll!KiFastSystemCall:
775b70b0 8bd4            mov     edx,esp
775b70b2 0f34            sysenter
ntdll!KiFastSystemCallRet:
775b70b4 c3              ret

0:000>
```

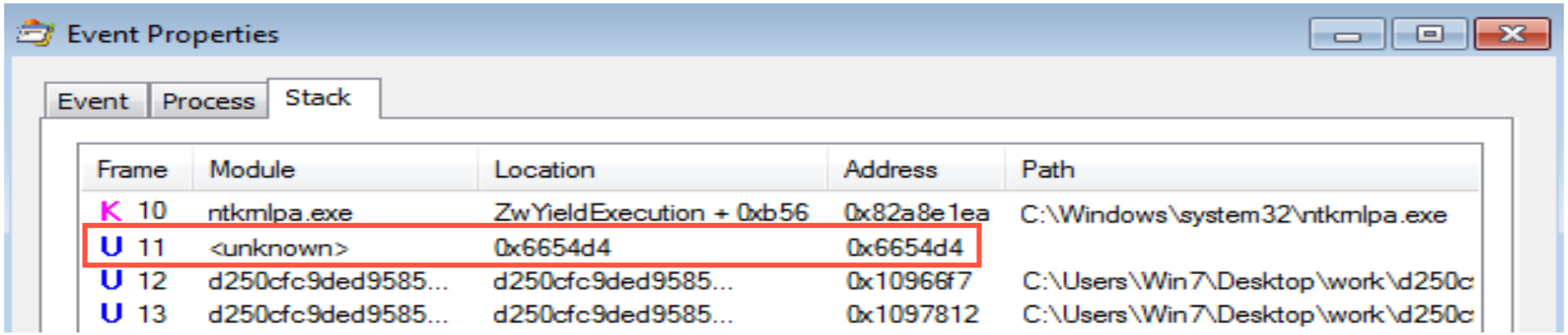
```
0:000> u 006654c8 L4
006654c8 b832000000      mov     eax,32h
006654cd ba4c858900      mov     edx,89854Ch
006654d2 ff12            call    dword ptr [edx]
006654d4 c20400          ret     4
0:000> u poi(0x89854C) L3
006670b0 8bd4            mov     edx,esp
006670b2 0f34            sysenter
006670b4 c3              ret

0:000>
```


ENSILO

BYPASS TECHNIQUES

Manually Load DLL From Disk




```
0:000> !address 0x620000
Usage:                  <unknown>
Base Address:           00620000
End Address:            0089b000
Region Size:            0027b000
State:                  00001000    MEM_COMMIT
Protect:                00000040    PAGE_EXECUTE_READWRITE
Type:                   00020000    MEM_PRIVATE
Allocation Base:        00620000
Allocation Protect:     00000040    PAGE_EXECUTE_READWRITE
0:000> !address 0x77570000
Usage:                  Image
Base Address:           77570000
End Address:            77571000
Region Size:            00001000
State:                  00001000    MEM_COMMIT
Protect:                00000002    PAGE_READONLY
Type:                   01000000    MEM_IMAGE
Allocation Base:        77570000
Allocation Protect:     00000080    PAGE_EXECUTE_WRITECOPY
Image Path:             ntdll.dll
Module Name:            ntdll
Loaded Image Name:      C:\Windows\SYSTEM32\ntdll.dll
Mapped Image Name:
0:000>
```


ENSILO

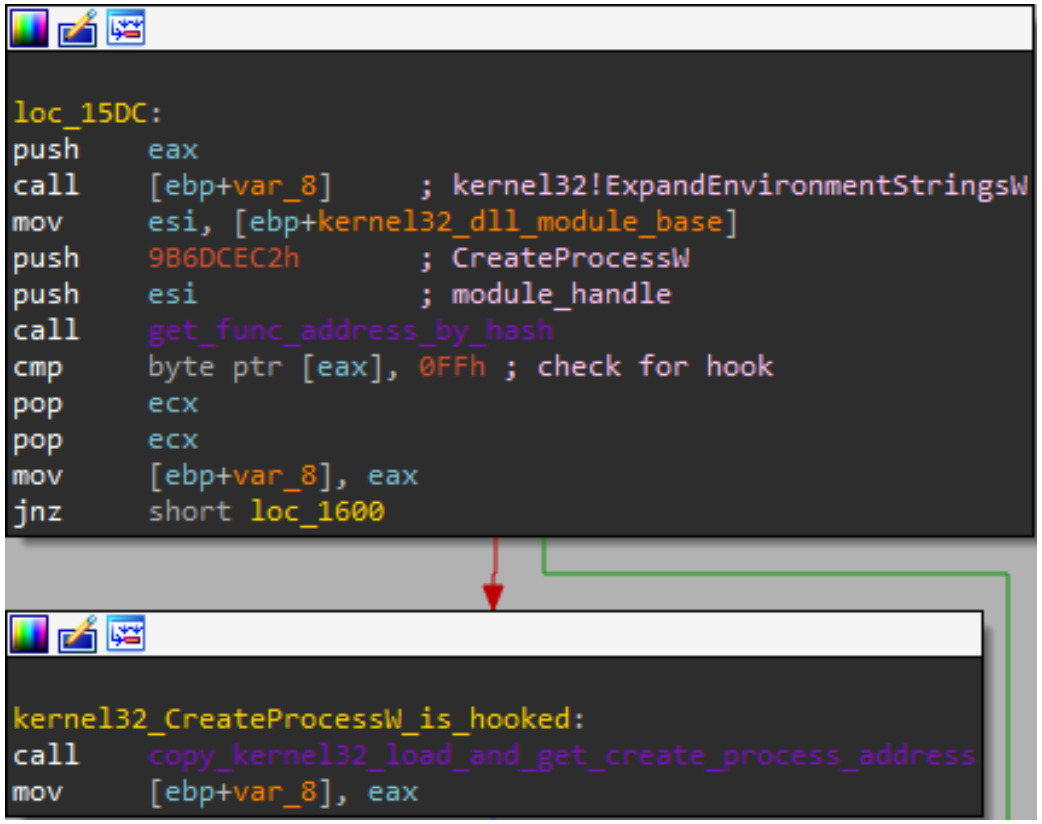
BYPASS TECHNIQUES
Clone DLL

- CopyFile() + LoadLibrary()

- Hancitor, reported by MalwareBytes
- Downloader
- Copies kernel32.dll
  - Call CreateProcess as part of Process Hollowing

BYPASS TECHNIQUES

Clone DLL



```
loc_15DC:
push    eax
call    [ebp+var_8]     ; kernel32!ExpandEnvironmentStringsW
mov     esi, [ebp+kernel32_dll_module_base]
push    9B6DCEC2h       ; CreateProcessW
push    esi             ; module_handle
call    get_func_address_by_hash
cmp     byte ptr [eax], 0FFh ; check for hook
pop     ecx
pop     ecx
mov     [ebp+var_8], eax
jnz     short loc_1600
```

```
kernel32_CreateProcessW_is_hooked:
call    copy_kernel32_load_and_get_create_process_address
mov     [ebp+var_8], eax
```

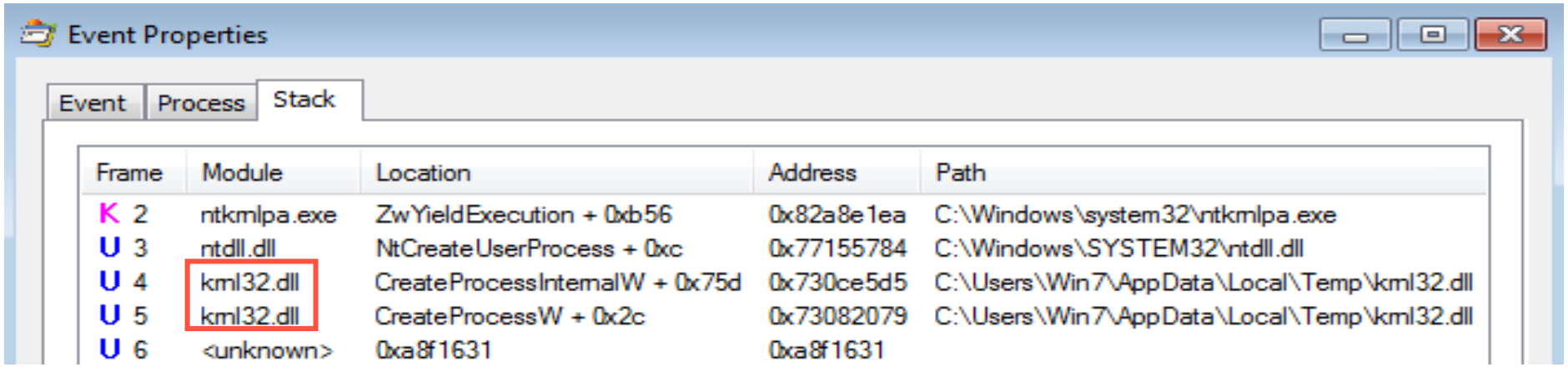
```
Breakpoint 0 hit
kernel32!CopyFileExW:
7669ac6c 6a18            push    18h
0:006> du poi(esp+4)
05c7ec40  "C:\Windows\system32\kernel32.dll"
05c7ec80  ""
0:006> du poi(esp+8)
05c7ee40  "C:\Users\Win7\AppData\Local\Temp"
05c7ee80  "\krnl32.dll"
0:006> g
Breakpoint 1 hit
kernel32!LoadLibraryW:
766b3c01 8bff            mov     edi,edi
0:006> du poi(esp+4)
05c7ee40  "C:\Users\Win7\AppData\Local\Temp"
05c7ee80  "\krnl32.dll"

0:006>
```


ENSILO

BYPASS TECHNIQUES

Clone DLL




```
0:006> !lmi krnl32
Loaded Module Info: [krnl32]
         Module: krnl32
   Base Address: 73080000
     Image Name: C:\Users\Win7\AppData\Local\Temp\krnl32.dll
   Machine Type: 332 (I386)
     Time Stamp: 4ce7b8ef Sat Nov 20 14:02:55 2010
           Size: d4000
       CheckSum: d70eb
Characteristics: 2102  perf
0:006> !lmi kernel32
Loaded Module Info: [kernel32]
         Module: kernel32
   Base Address: 76660000
     Image Name: C:\Windows\system32\kernel32.dll
   Machine Type: 332 (I386)
     Time Stamp: 4ce7b8ef Sat Nov 20 14:02:55 2010
           Size: d4000
       CheckSum: d70eb
Characteristics: 2102  perf

0:006>
```


ENSILO

BYPASS TECHNIQUES

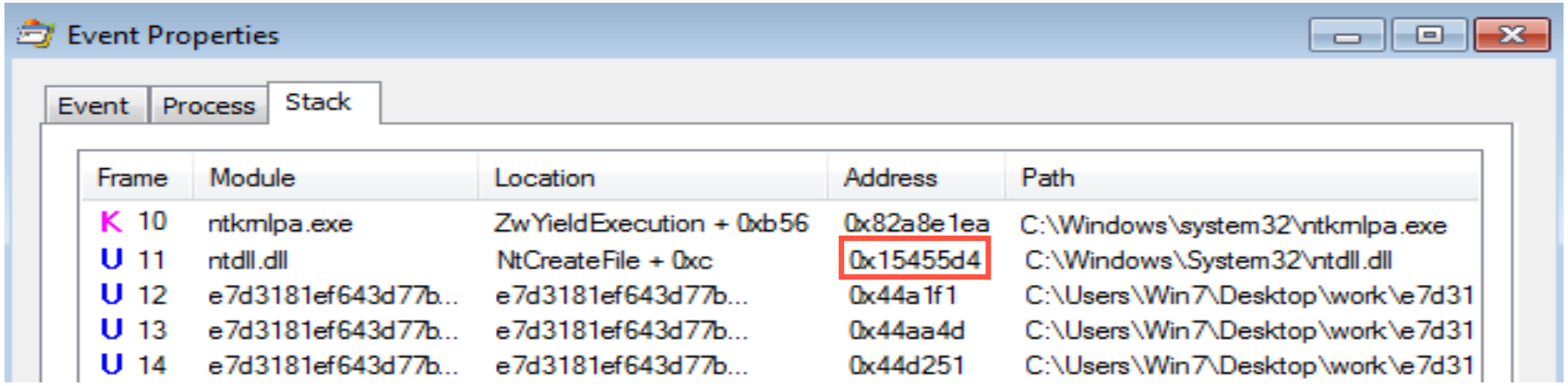
Section Remapping

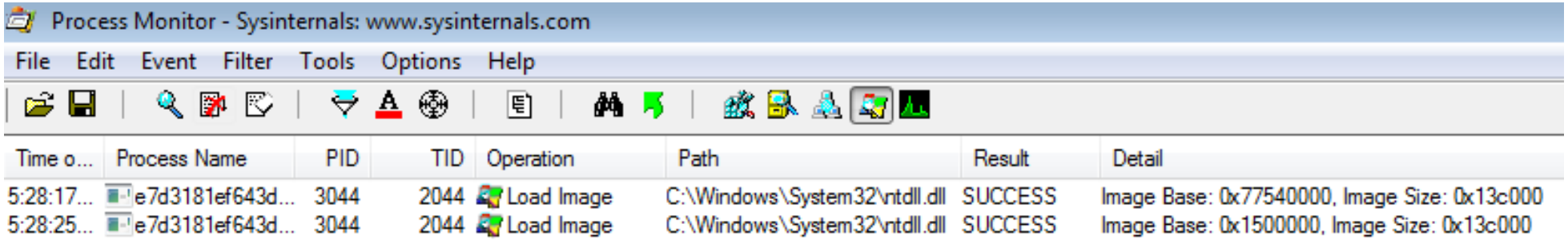
- [Nt]CreateFile() + NtCreateSection(…, SEC_IMAGE, …) + ZwMapViewOfSection()

- Osiris, reported by MalwareBytes
- Banking trojan
- Remaps ntdll.dll
  - Process Doppelgänging\Hollowing hybrid (“Transacted Hollowing”)

BYPASS TECHNIQUES

Section Remapping






```
0:000> !dlls -c ntdll

0x00231d28: C:\Windows\SYSTEM32\ntdll.dll
      Base    0x77540000  EntryPoint  0x00000000  Size        0x0013c000
      Flags   0x80004004  LoadCount   0x0000ffff  TlsIndex    0x00000000
              LDRP_IMAGE_DLL
              LDRP_ENTRY_PROCESSED
0:000> !dlls -c 0x01500000

0:000>
```


ENSILO

ENSILO

BYPASS TECHNIQUES ANALYSIS

Direct system call invocation

BYPASS TECHNIQUES

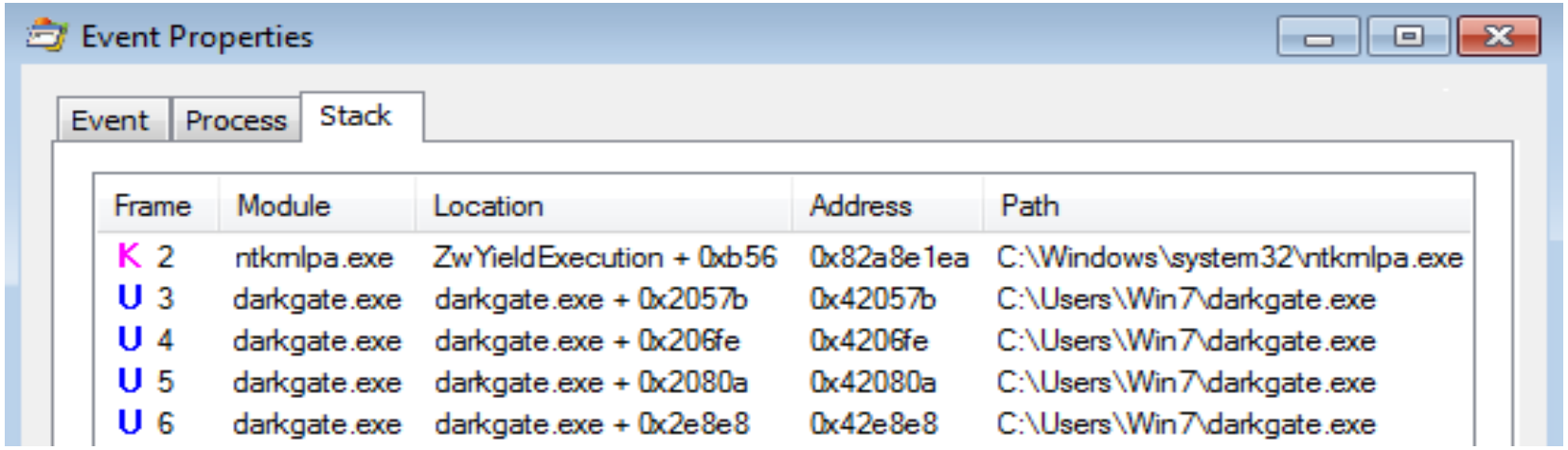
NTDLL Parsing

- Calling system calls directly

- DarkGate, reported by enSilo
- Crypto miner and stealer
- Used for Process Hollowing and writing to the registry

BYPASS TECHNIQUES

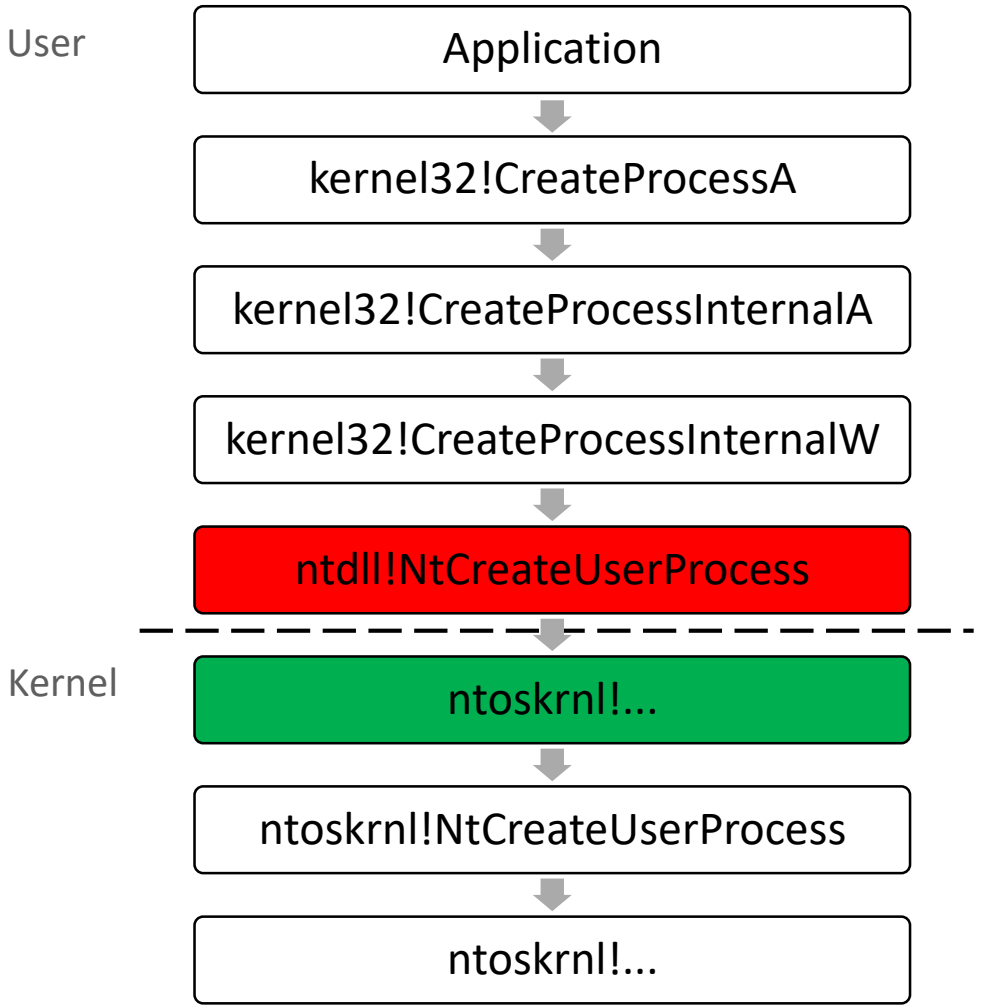
NTDLL Parsing

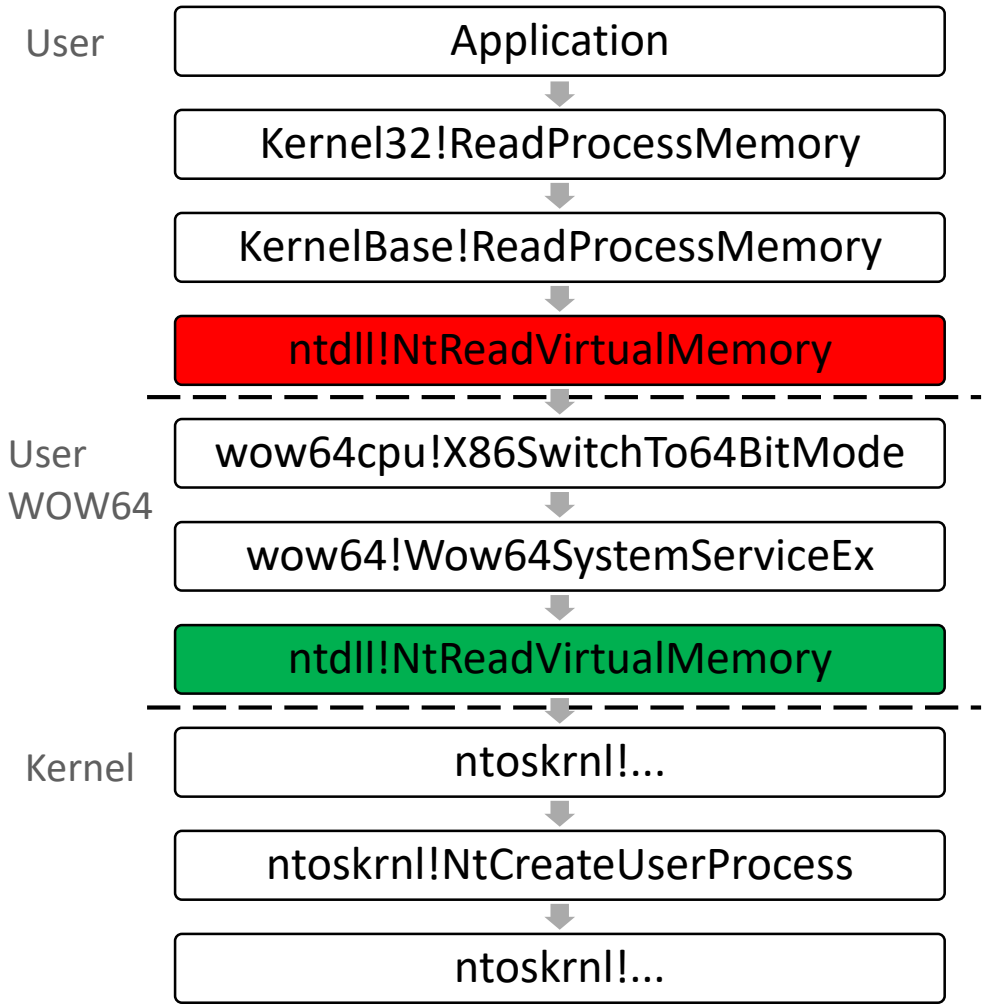


ENSILO

BYPASS TECHNIQUES

NTDLL Parsing





ENSILO

BYPASS TECHNIQUES

Heaven's Gate

- Make system calls from within WOW64 emulation layer
  - 32-bit application on 64-bit Windows

- GlobeImposter, reported by enSilo
- Ransomware
- Used for Process Hollowing

BYPASS TECHNIQUES

Heaven's Gate

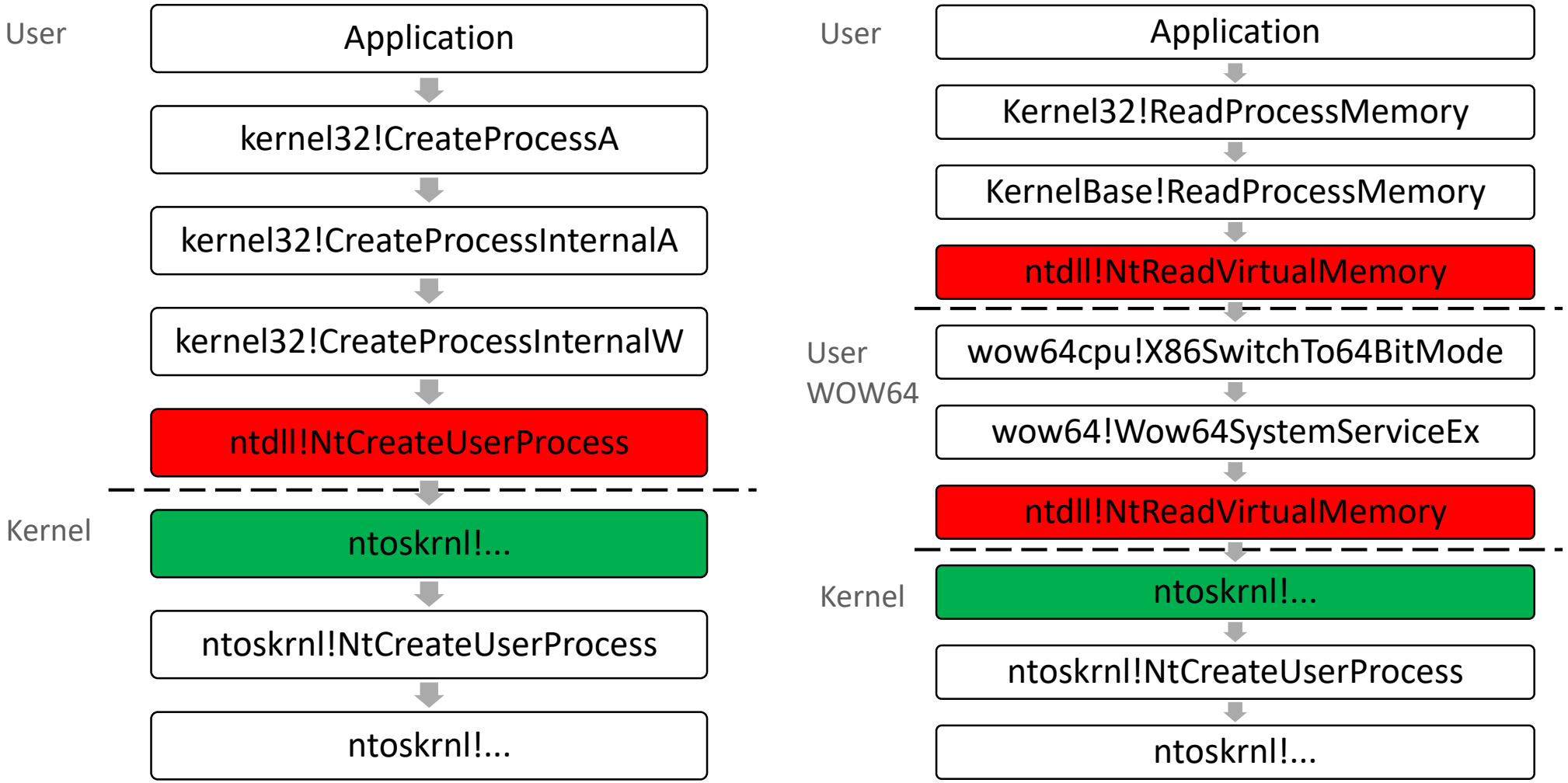
```
0: kd> k
 # RetAddr           Call Site
00 fffff800`02afb9d3 nt!NtCreateSection
01 00000000`01eafb03 nt!KiSystemServiceCopyEnd+0x13
02 75bc192e`75bc13e0 0x1eafb03
03 75bc3fd5`75bc4064 0x75bc192e`75bc13e0
04 75bc183e`75bc193e 0x75bc3fd5`75bc4064
05 75bc481b`75bc48c7 0x75bc183e`75bc193e
06 7751deb6`7751de7c 0x75bc481b`75bc48c7
07 00000000`00000040 0x7751deb6`7751de7c
0: kd> u 0x1eafb01 L7
00000000`01eafb01 0f05            syscall
00000000`01eafb03 8945f0          mov     dword ptr [rbp-10h],eax
00000000`01eafb06 0365fc          add     esp,dword ptr [rbp-4]
00000000`01eafb09 e800000000      call    00000000`01eafb0e
00000000`01eafb0e c744240423000000 mov     dword ptr [rsp+4],23h
00000000`01eafb16 8304240d        add     dword ptr [rsp],0Dh
00000000`01eafb1a cb              retf
0: kd>
```

```
1: kd> k
 # RetAddr           Call Site
00 fffff800`02afb9d3 nt!NtReadFile
01 00000000`74fe2e09 nt!KiSystemServiceCopyEnd+0x13
02 00000000`74fe29f5 wow64cpu!CpupSyscallStub+0x9
03 00000000`7505d286 wow64cpu!ReadWriteFileFault+0x31
04 00000000`7505c69e wow64!RunCpuSimulation+0xa
05 00000000`77a543c3 wow64!Wow64LdrpInitialize+0x42a
06 00000000`77ab9780 ntdll!LdrpInitializeProcess+0x17e3
07 00000000`77a6371e ntdll! ?? ::FNODOBFM::`string'+0x22790
08 00000000`00000000 ntdll!LdrInitializeThunk+0xe
1: kd>
```


ENSILO

BYPASS TECHNIQUES

Heaven's Gate



ENSILO

ENSILO

BYPASS TECHNIQUES ANALYSIS

Code splicing

BYPASS TECHNIQUES

Code Splicing (a.k.a. Byte Stealing)

- Rebuild function stubs elsewhere
- Commonly used by packers

- CodeFork's Gamarue, reported by Radware
- Downloader for bots, spamming, miners…
- Copies the first instruction of library functions it uses

BYPASS TECHNIQUES

Code Splicing (a.k.a. Byte Stealing)

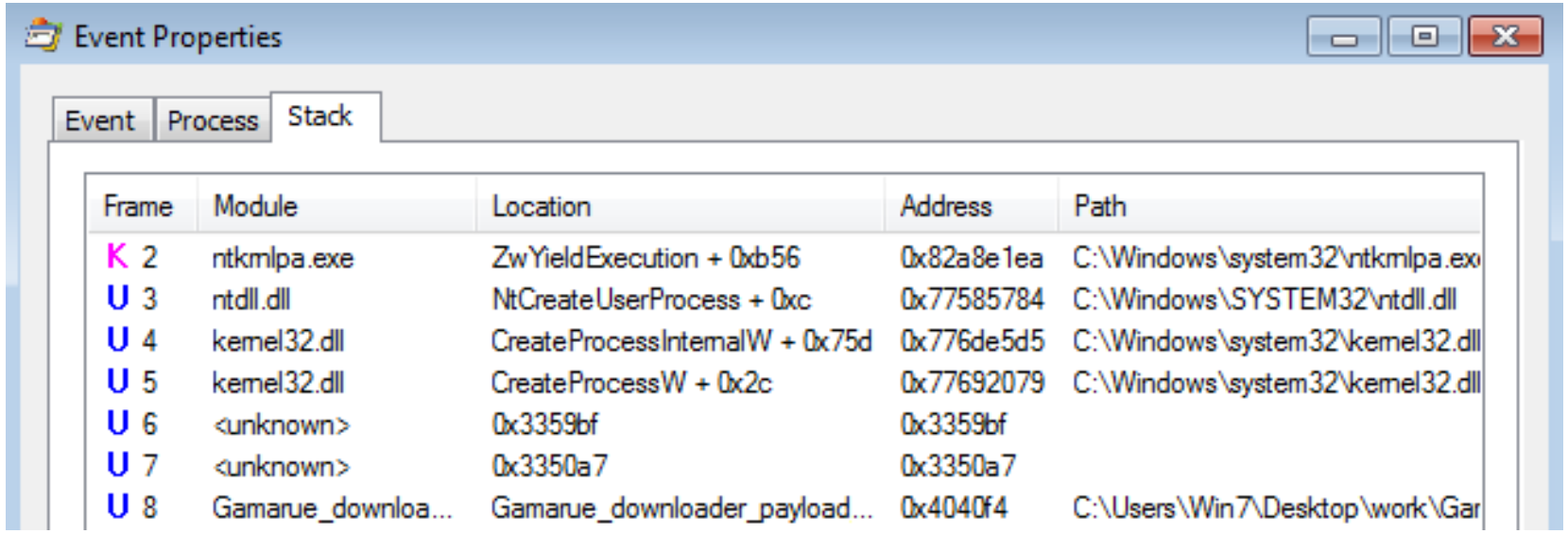
```
0:000> u eip L5
00334f13 8b3d013f3300    mov     edi,dword ptr ds:[333F01h]
00334f19 33f6            xor     esi,esi
00334f1b 6807800000      push    8007h
00334f20 8975fc          mov     dword ptr [ebp-4],esi
00334f23 ffd7            call    edi
0:000> u poi(0x333f01) L2
0033aeb0 8bff            mov     edi,edi
0033aeb2 e99c9b3a77      jmp     kernel32!SetErrorModeStub+0x2 (776e4a53)
0:000> dd 0x333f01
00333f01  0033aeb0 0033aec0 0033aed0 0033aee0
00333f11  0033aef0 0033af00 0033af10 0033af20
00333f21  0033af30 0033af40 0033af50 0033af60
00333f31  0033af70 0033af80 0033af90 0033afa0
00333f41  0033afb0 0033afc0 0033afd0 0033afe0
00333f51  0033aff0 0033b000 00000000 0033b150
0:000> u 0x33aec0 L2
0033aec0 8bff            mov     edi,edi
0033aec2 e988713577      jmp     kernel32!CreateProcessW+0x2 (7769204f)
0:000> u 0x33aff0 L2
0033aff0 6a40            push    40h
0033aff2 e93c473977      jmp     kernel32!CreateToolhelp32Snapshot+0x2 (776cf733)
0:000>
```

```
0:000> u kernel32!CreateProcessW L2
kernel32!CreateProcessW:
7769204d 8bff            mov     edi,edi
7769204f 55              push    ebp
0:000> u kernel32!CreateToolhelp32Snapshot L2
kernel32!CreateToolhelp32Snapshot:
776cf731 6a40            push    40h
776cf733 6818f86c77      push    offset kernel32
0:000>
```


ENSILO

BYPASS TECHNIQUES

Code Splicing (a.k.a. Byte Stealing)



ENSILO

BYPASS TECHNIQUES

Comparison

Technique	Runtime Indicators	Forensic Artifacts	Drawbacks
Manually Load DLL From Disk	Callstacks missing relevant DLLs	Floating PE copy in memory	Significantly different from the norm
Clone DLL	Callstacks with unexpected DLLs	Identical PEs in memory	Changes to file system
			Lower level\dependencies can be hooked
Section Remapping	Multiple mappings of same PE	Multiple mappings of same PE	Can't be used for complex code
NTDLL Parsing	Callstacks missing ntdll.dll		Limited functionality
Heaven's Gate	Callstacks missing WOW64 system DLLs		Limited functionality
Code Splicing			Internal\lower level\dependencies can be hooked

BYPASS TECHNIQUES
Summary

- Used by all sorts of malware families
- Sophisticated actors, though not necessarily APTs
- Usually to mask the initial steps and establishing foothold

- None of the techniques are actually new
- Some techniques are not as commonly used in the wild
  - Unhook Flashbang\ReflectiveDLLRefresher: detectible and reversible
  - Bring Your Own Indexes (BYOI): version dependent

ANALYSIS AND DETECTION TACTICS

- Events regarding system DLLs can be used as indicators
  - Copy, multiple read\load operations
- Check the callstacks

- Place hooks\breakpoints at non-trivial locations
- Randomize as much as you can

- Hook many different layers (“mine” the path)
- Correlate user-mode and kernel-mode data

- Use information provided by the OS (ETW)

CLOSING REMARKS

- These are only a handful of examples

- Trivial to implement, simple to use (most have source code available)
- Hardly any recent innovations, yet still very effective

- MITRE ATT&CK doesn't reference hook bypassing as defense evasion

- Using user-mode hooks for security is not enough

ENSILO

QUESTIONS?

www.breakingmalware.com

omri@ensilo.com in/omri-misgav

udi@ensilo.com in/udiyavo @UdiYavo

ENSILO

THANK YOU

www.breakingmalware.com

omri@ensilo.com in/omri-misgav

udi@ensilo.com in/udiyavo @UdiYavo