You're Off the Hook: Blinding Security Software

Alex Matrosov | Principal Research Scientist
Jeff Tang | Senior Security Researcher

CYLANCE™

Who We Are - Alex

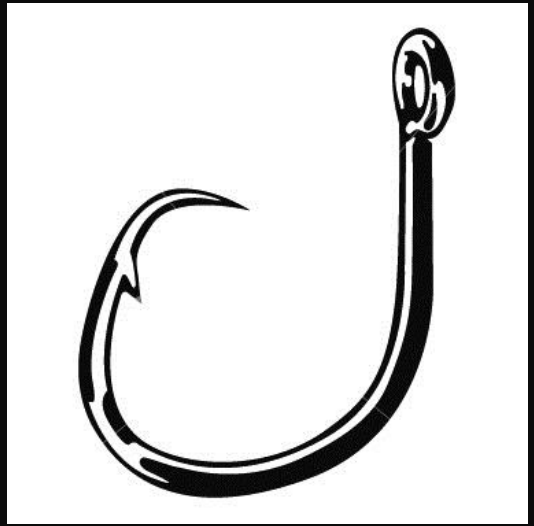
- Principal REsearch Scientist @CylanceInc
- Previously @Intel / @IntelSecurity / @ESET
- Co-author of Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats
- @matrosov @matrosov

Who We Are - Jeff

- Senior Security Researcher @CylanceInc
- Formerly @VAHNA / @NSAGov
- Hates computers – career dedicated to breaking computers
- @mrjefftang @mrjefftang

Overview

- Why user-mode hooks?
- Hooking Basics
- Hooking Vulnerabilities (Captain Hook)
- Control Flow Guard / Return Flow Guard
- Universal Unhooking
- Demo / Results



Why user-mode hooking?

- Kernel Patch Protection (KPP) a.k.a. “PatchGuard”
  - Introduced in 2005 for Windows 2003 SP1 x64 – multiple revisions since then
- Prevents modification to the Windows kernel and kernel data structures
  - IDT, GDT, SSDT, MSR, System PE Images
- Analysis and bypasses documented by various security researchers (skape, skywing)
- Bypasses implemented by malware rootkits (Uroburos)
- Incrementally updated to address deficiencies and block bypasses
  - Creates an arms race between independent software vendors (ISV) and Microsoft
- Forces security vendors to rely on user-mode hooking in order to monitor processes for malicious behavior

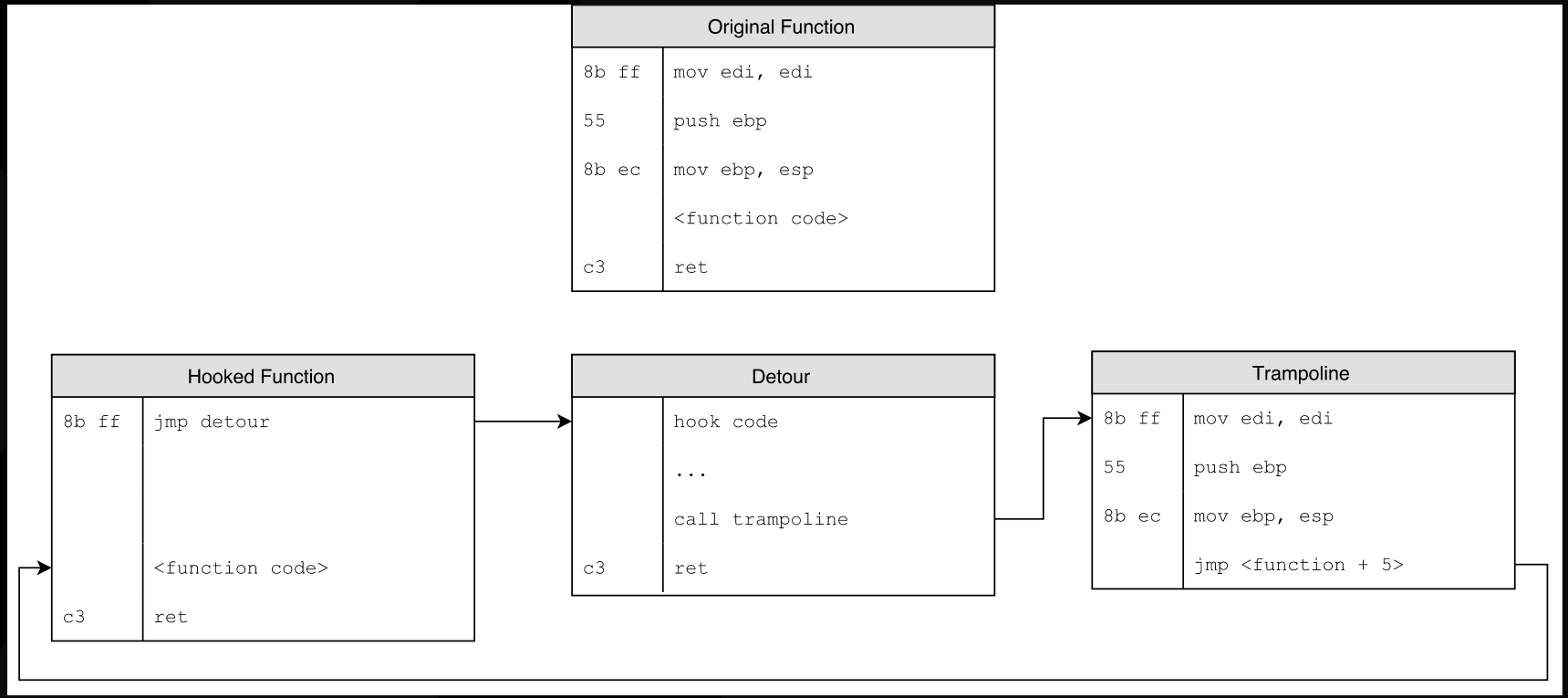
- http://blog.talosintel.com/2014/08/the-windows-81-kernel-patch-protection.html

User-Mode Hooking Basics

- IAT Hooking
  - Replace address of target function in IAT with address to a new function
- Inline Hooking
  - Replaces the prologue of target functions with a `jmp` to a detour
  - Detour function calls a trampoline function
  - Trampoline function contains the function's original prologue and `jmp` into the original function
  - Original function returns into detour
  - Detour returns to caller

- http://jbremer.org/x86-api-hooking-demystified/

(Inline) Hooking Basics



Inline Hooking

```
0:006> u ntdll!ZwMapViewOfSection
ntdll!ZwMapViewOfSection:
00007ff9`96875350 4c8bd1          mov     r10,rcx
00007ff9`96875353 b828000000      mov     eax,28h
00007ff9`96875358 f604250803fe7f01 test    byte ptr [SharedUserData+0x308 (00000000`7ffe0308)],1
00007ff9`96875360 7503            jne     ntdll!ZwMapViewOfSection+0x15 (00007ff9`96875365)
00007ff9`96875362 0f05            syscall
00007ff9`96875364 c3              ret
00007ff9`96875365 cd2e            int     2Eh
00007ff9`96875367 c3              ret
```

```
0:004> u ntdll!ZwMapViewOfSection
ntdll!ZwMapViewOfSection:
00007ffe`78085350 48b880008e6ff67f0000 mov rax,7FF66F8E0080h
00007ffe`7808535a 50              push    rax
00007ffe`7808535b c3              ret
00007ffe`7808535c 03fe            add     edi,esi
00007ffe`7808535e 7f01            jg      ntdll!ZwMapViewOfSection+0x11 (00007ffe`78085361)
00007ffe`78085360 7503            jne     ntdll!ZwMapViewOfSection+0x15 (00007ffe`78085365)
00007ffe`78085362 0f05            syscall
00007ffe`78085364 c3              ret
```

Hooking Issues (Captain Hook)

- Documented by the enSilo Research Team & presented at BlackHat US 2016
  - Captain Hook: Pirating AVs to Bypass Exploit Mitigations
- Discovered 6 classes of major vulnerabilities in user-mode hooking

Hooking Issues (Captain Hook)

Issue		Severity	Affected underlying systems
1	Unsafe injection	Very high	All windows versions
2	Predictable RWX code stubs	Very high	All windows versions
3	Predictable RX code stubs	High	All windows versions
4	Predictable RWX code stubs	High	Windows 7 and below
5	RWX hook code stubs	Medium	All windows versions
6	RWX hooked modules	Medium	All windows versions

https://www.blackhat.com/docs/us-16/materials/us-16-Yavo-Captain-Hook-Pirating-AVs-To-Bypass-Exploit-Mitigations.pdf

Products/Vendors	UnSafe Injection	Predictable RWX(Universal)	Predictable RX(Universal)	Predictable RWX	RWX Hook code stubs	RWX Hooked Modules	Time To Fix (Days)
Symantec				X			90
McAfee				X	X		90
Trend Micro		X	X (Initial Fix)		X		210
Kaspersky			X	X			90
AVG				X			30
BitDefender					X	X	30
WebRoot			X			X	29
AVAST			X		X		30
Emsisoft					X		90
Citrix - Xen Desktop					X	X	90
Microsoft Office*			X				180
WebSense	X			X		X	30
Vera	X			X			?
Invincea		X(64-bit)			X	X	?
Anti-Exploitation*				X			?
BeyondTrust			X	X			Fixed Independently
TOTALS	2	2	6	8	7	5	79.9

https://www.blackhat.com/docs/us-16/materials/us-16-Yavo-Captain-Hook-Pirating-AVs-To-Bypass-Exploit-Mitigations.pdf

Control Flow Guard (CFG)

- Introduced with Windows 10 / Windows 8.1 Update 3
- Requires support from the compiler/linker and operating system
- Compiler generates CFG instrumented binaries (`/guard:cf`)
  - Embeds a CFG Bitmap representing valid indirect call locations
  - Inserts `_guard_check_icall` before indirect calls
- Operating system supports CFG at run time:
  - Supported OS maps guard check to `ntdll!LdrpValidateUserCallTarget`
  - Unsupported OS is just a simple `ret`
- Not perfect, there are bypasses available
- Functions left RWX by a hooking engine bypass CFG as they can be overwritten with shellcode

Return Flow Guard (RFG)

- Upcoming technology likely to be introduced in Windows 10 Creators Update
- Protects return address by comparing it to the saved value on a shadow stack (`fs:[rsp]`)
- Requires support from the compiler/linker and operating system
- Compiler generates an RFG instrumented binary
  - Inserts no-op padding in function prologue and epilogues
- Operating system supports RFG at run time:
  - Overwrites prologue with instructions to save return address to shadow stack
  - Overwrites epilogue with instructions to check return address against shadow stack
- http://xlab.tencent.com/en/2016/11/02/return-flow-guard/

Return Flow Guard (RFG)

```
.text:000000014000176C wWinMain
.text:000000014000176C                 xchg    ax, ax
.text:000000014000176E                 nop     dword ptr [rax+00000000h]
```

```
0:000> u calc!wWinMain
calc!wWinMain:
00007ff7`91ca176c 488b0424        mov     rax,qword ptr [rsp]
00007ff7`91ca1770 6448890424      mov     qword ptr fs:[rsp],rax
```

Return Flow Guard (RFG)

```
.text:00000001400025BC __guard_ss_common_verify_stub
.text:00000001400025BC                 retn
.text:00000001400025BD                 db 0Eh dup(90h)
.text:00000001400025CB                 retn
```

```
0:000> u calc!_guard_ss_common_verify_stub
calc!_guard_ss_common_verify_stub:
00007ff7`91ca25bc 644c8b1c24      mov     r11,qword ptr fs:[rsp]
00007ff7`91ca25c1 4c3b1c24        cmp     r11,qword ptr [rsp]
00007ff7`91ca25c5 0f85f5000000    jne     calc!_guard_ss_verify_failure (00007ff7`91ca26c0)
00007ff7`91ca25cb c3              ret
```

Prior Work

- Unhooking has been around for a long time
- Generally relies on identifying specific hooking methods/signatures
- Focuses in hooks installed in function prologues
  - https://breakdev.org/defeating-antivirus-real-time-protection-from-the-inside/
- In-memory RFG instrumented binaries will be different than on-disk – unhooking will remove the RFG prologue/epilogue patches
- Endpoint detect & respond (EDR) products rely heavily on user-mode hooking for telemetry

Universal Unhooking DLL

- For each module in `PEB_LDR_DATA->InMemoryOrderModuleList`:
  - ✓ Load the module from disk
  - ✓ Allocate a new memory space of `NtHeader->OptionalHeader.SizeOfImage`
  - ✓ Perform PE relocations based off original module base address
  - ✓ Resolve Import Address Table (IAT)
  - ✓ Compare our new copy with the original copy
  - ✓ Copy over our pristine copy if changes are detected
- Based off of Stephen Fewer's Reflective DLL Injection code

Universal Unhooking DLL - Loading

- Load the file into memory
  - ✓ `CreateFile("ntdll.dll", …)`
  - ✓ `CreateFileMapping(...)`
  - ✓ `MapViewOfFile(...)`
  - ✓ `<new base address> = VirtualAlloc(..., NtHeader->Optionalheader.SizeOfImage, ...)`
- Copy PE sections into correct location in memory
  - ✓ `memcpy(<new base address> + SectionHeader->VirtualAddress, SectionHeader->PointerToRawData)`

Universal Unhooking DLL - Relocation

- The normal relocation process adds an image delta to each relocation
  - ✓ `delta = loaded base address - OptionalHeader.ImageBase`
  - Or it adds the high/low word of the delta depending on relocation type
- Our unhooking code performs relocation with the loaded base address of the original loaded DLL
  - ✓ `delta = original base address - OptionalHeader.ImageBase`
- This gives us a binary copy of what the DLL looked like after being loaded but before being hooked

Universal Unhooking DLL – Import Resolution

- Need to resolve the import table without bringing in a hooked function
- Custom `GetProcAddress` to parse the `IMAGE_EXPORT_DIRECTORY` of each module
- Must support export forward descriptors and API Sets
- A forward descriptor is a function that resolves to a virtual address within the `IMAGE_EXPORT_DIRECTORY` address space

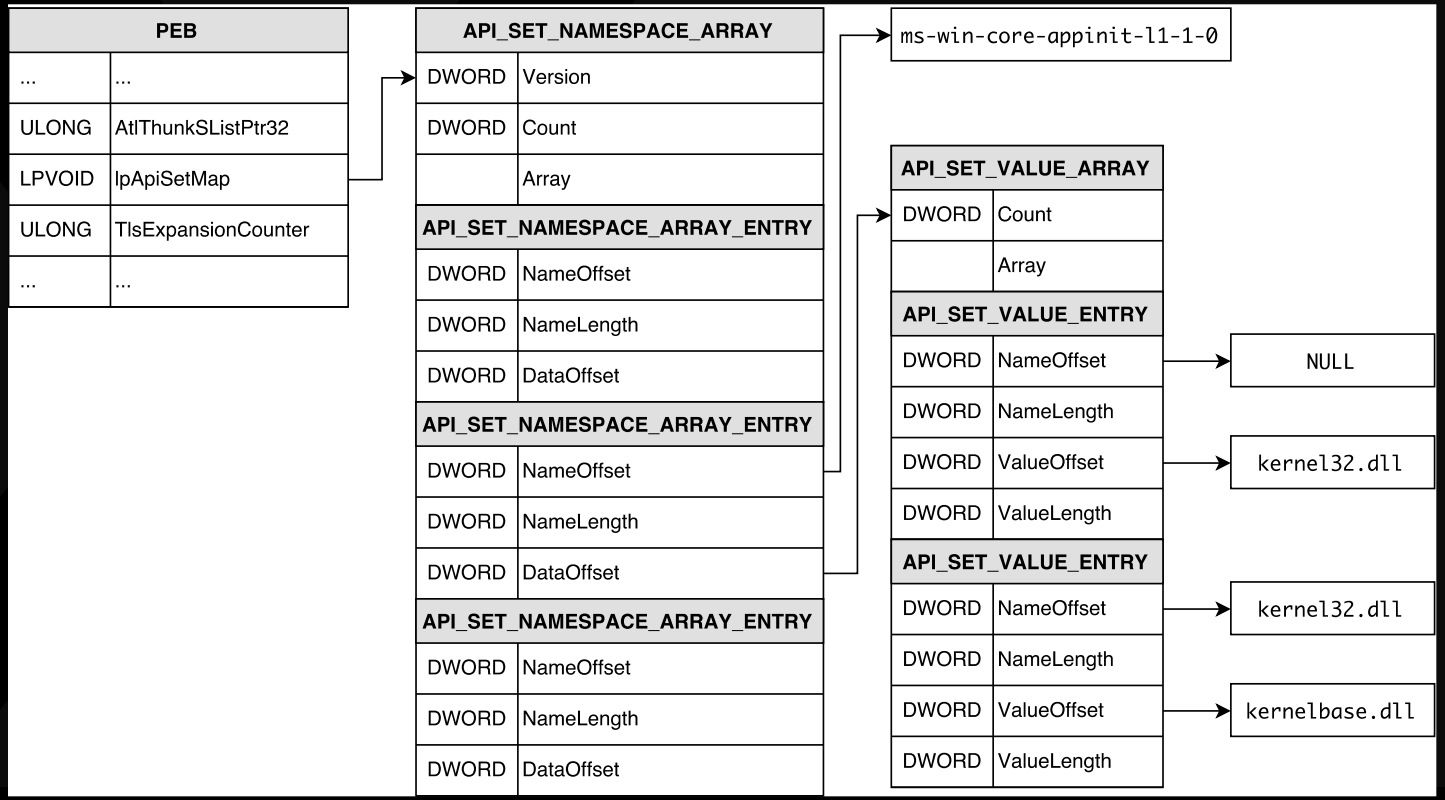
Universal Unhooking DLL – Import Resolution

```
Dump of file c:\windows\system32\kernel32.dll

File Type: DLL

  Section contains the following exports for KERNEL32.dll

    00000000 characteristics
    578990D7 time date stamp Fri Jul 15 21:41:43 2016
        0.00 version
           1 ordinal base
        1607 number of functions
        1607 number of names

    ordinal hint RVA      name

          1    0          AcquireSRWLockExclusive (forwarded to NTDLL.RtlAcquireSRWLockExclusive)
          2    1          AcquireSRWLockShared (forwarded to NTDLL.RtlAcquireSRWLockShared)
          3    2 0001EB70 ActivateActCtx
          4    3 00019EC0 ActivateActCtxWorker
          5    4 00023010 AddAtomA
          6    5 00010CC0 AddAtomW
          7    6 000628F0 AddConsoleAliasA
          8    7 00062A50 AddConsoleAliasW
          9    8          AddDllDirectory (forwarded to api-ms-win-core-libraryloader-l1-1-0.AddDllDirectory)
```

API Set Schema

- Introduced in Windows 7 (v2) as part of project “MinWin”
  - Updated in Windows 8.1 (v4) and Windows 10 (v6)
- Mapping of “Virtual DLLs” to “Logical DLLs” stored in `apisetschema.dll`
- Very little documented information on API Set Schema
  - http://www.geoffchappell.com/studies/windows/win32/apisetschema/index.htm
  - http://blog.quarkslab.com/runtime-dll-name-resolution-apisetschema-part-i.html
  - http://blog.quarkslab.com/runtime-dll-name-resolution-apisetschema-part-ii.html

API Set Schema - Example



API Set Schema - Evolution

API_SET_NAMESPACE_ARRAY_V2	
DWORD	Version
DWORD	Count
	Array
API_SET_NAMESPACE_ARRAY_ENTRY_V2	
DWORD	NameOffset
DWORD	NameLength
DWORD	DataOffset

API_SET_NAMESPACE_ARRAY_V4	
DWORD	Version
DWORD	Size
DWORD	Flags
DWORD	Count
	Array
API_SET_NAMESPACE_ARRAY_ENTRY_V4	
DWORD	Flags
DWORD	NameOffset
DWORD	NameLength
DWORD	AliasOffset
DWORD	AliasLength
DWORD	DataOffset

API_SET_NAMESPACE_ARRAY_V6	
DWORD	Version
DWORD	Size
DWORD	Flags
DWORD	Count
DWORD	DataOffset
DWORD	HashOffset
DWORD	HashMultiplier
DWORD	Array
API_SET_NAMESPACE_ARRAY_ENTRY_V6	
DWORD	Flags
DWORD	NameOffset
DWORD	Size
DWORD	NameLength
DWORD	DataOffset
DWORD	Count

API Set Schema

API_SET_VALUE_ARRAY_V2	
DWORD	Count
	Array

API_SET_VALUE_ENTRY_V2	
DWORD	NameOffset
DWORD	NameLength
DWORD	ValueOffset
DWORD	ValueLength

API_SET_VALUE_ARRAY_V4	
DWORD	Flags
DWORD	Count
	Array

API_SET_VALUE_ENTRY_V4	
DWORD	Flags
DWORD	NameOffset
DWORD	NameLength
DWORD	ValueOffset
DWORD	ValueLength

API_SET_VALUE_ENTRY_V6	
DWORD	Flags
DWORD	NameOffset
DWORD	NameLength
DWORD	ValueOffset
DWORD	ValueLength

API_SET_HASH_ENTRY_V6	
DWORD	Hash
DWORD	Index

Universal Unhooking DLL – Compare & Restore

- Compare each PE section of the original loaded DLL to our version
- Replace any sections which are different
- Return back to initial program to continue execution without hooks

Reflective DLL

- The universal unhooking code is compiled as a reflective DLL
- Meterpreter: `post/windows/manage/reflective_dll_inject`
- Can be injected into everything!

Meterpreter

- Meterpreter server is already delivered as a Reflective DLL
- Security software could detect Meterpreter initialization prior to unhooking
- Solution: Modify meterpreter server to call unhooking code before connection initialization
- Drop in replacement for Metasploit installs
  - Copy modified `metsrv.dll` to `metasploit-framework/data/meterpreter/`

UPX

- Why UPX?
  - It's well known and open-source
- Modify UPX to insert an extra section '`UPX3`' containing arbitrary reflective DLL
- UPX decoder stub unpacks original binary and resolves imports
  - Decoder calls the first export in the reflective DLL (`ReflectiveLoader()`)
  - Unhooking code runs and removes any hooks detected
  - Stub decoder jumps to original entry point (OEP)
- Note: Reflective DLL is not compressed by UPX
- Also blows up the size of the final packed executable

UPX

upx.x86.packed.exe | upx.x86.packed_dll.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
UPX0	00154000	00001000	00000000	00000400	00000000	00000000	0000	0000	E0000080
UPX1	0006D000	00155000	0006CC00	00000400	00000000	00000000	0000	0000	E0000040
.rsrc	00001000	001C2000	00000400	0006D000	00000000	00000000	0000	0000	C0000040

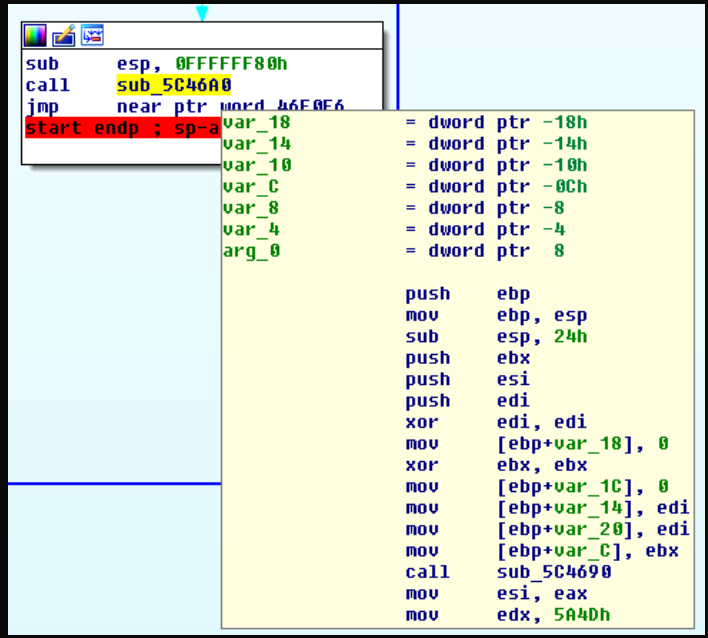
upx.x86.packed.exe | upx.x86.packed_dll.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
UPX0	00154000	00001000	00000000	00000400	00000000	00000000	0000	0000	E0000080
UPX1	0006D000	00155000	0006CC00	00000400	00000000	00000000	0000	0000	E0000040
.rsrc	00001000	001C2000	00000400	0006D000	00000000	00000000	0000	0000	C0000040
UPX3	00012C00	001C3000	00012C00	0006D400	00000000	00000000	0000	0000	60000040

UPX

- Final basic block of UPX stub decoder
- `call sub_5C46A0` – `ReflectiveLoader()`
- `jmp word_46E0E6` – Original Entry Point (OEP)

- Note: On 64-bit platforms, `rsp` must be 16-bit aligned before the call to `ReflectiveLoader()`



UPX

- Problem: Once UPX unpacks binary into memory, it will be different compared to on disk copy – how does the unhooking code avoid reloading packed binary into memory?

UPX

- Unhooking code ignores any PE section with `IMAGE_SCN_MEM_WRITE (0x80000000)`
- UPX marks every section as `PEFL_WRITE == IMAGE_SCN_MEM_WRITE`

```
1272    osection[0].flags = (unsigned) (PEFL_BSS|PEFL_EXEC|PEFL_WRITE|PEFL_READ);
1273    osection[1].flags = (unsigned) (PEFL_DATA|PEFL_EXEC|PEFL_WRITE|PEFL_READ);
1274    osection[2].flags = (unsigned) (PEFL_DATA|PEFL_WRITE|PEFL_READ);
```

UPX

upx.x86.packed.exe | upx.x86.packed_dll.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
UPX0	00154000	00001000	00000000	00000400	00000000	00000000	0000	0000	E0000080
UPX1	0006D000	00155000	0006CC00	00000400	00000000	00000000	0000	0000	E0000040
.rsrc	00001000	001C2000	00000400	0006D000	00000000	00000000	0000	0000	C0000040

upx.x86.packed.exe | upx.x86.packed_dll.exe

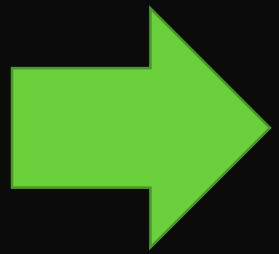
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
UPX0	00154000	00001000	00000000	00000400	00000000	00000000	0000	0000	E0000080
UPX1	0006D000	00155000	0006CC00	00000400	00000000	00000000	0000	0000	E0000040
.rsrc	00001000	001C2000	00000400	0006D000	00000000	00000000	0000	0000	C0000040
UPX3	00012C00	001C3000	00012C00	0006D400	00000000	00000000	0000	0000	60000040

Demo

Results

Software	Result
BitDefender	Success - Bypassed
Dr. Web	Success - Bypassed
ESET	Blocked
Kaspersky	Blocked
Symantec	Success - Unaffected
McAfee	Success - Unaffected
TrendMicro	Success - Unaffected
EMET	Success - Unaffected

Blinding Security Software



Future Work

- Improve UPX modifications to compress/pack the reflective DLL
- Update `ReflectiveLoader()` to perform unhooking checks and repairs
- Support for other packing software?

Closing

- User-mode hooking is fragile and easily bypassed
- Relying on user-mode hooking for protection is ineffective
- Critical need for OS support to provide callbacks and APIs for monitoring system behavior
- Source code will be released soon @ https://github.com/CylanceVulnResearch/

QUESTIONS AND ANSWERS

THANK YOU
ZERONIGHTS 2016

CYLANCE™