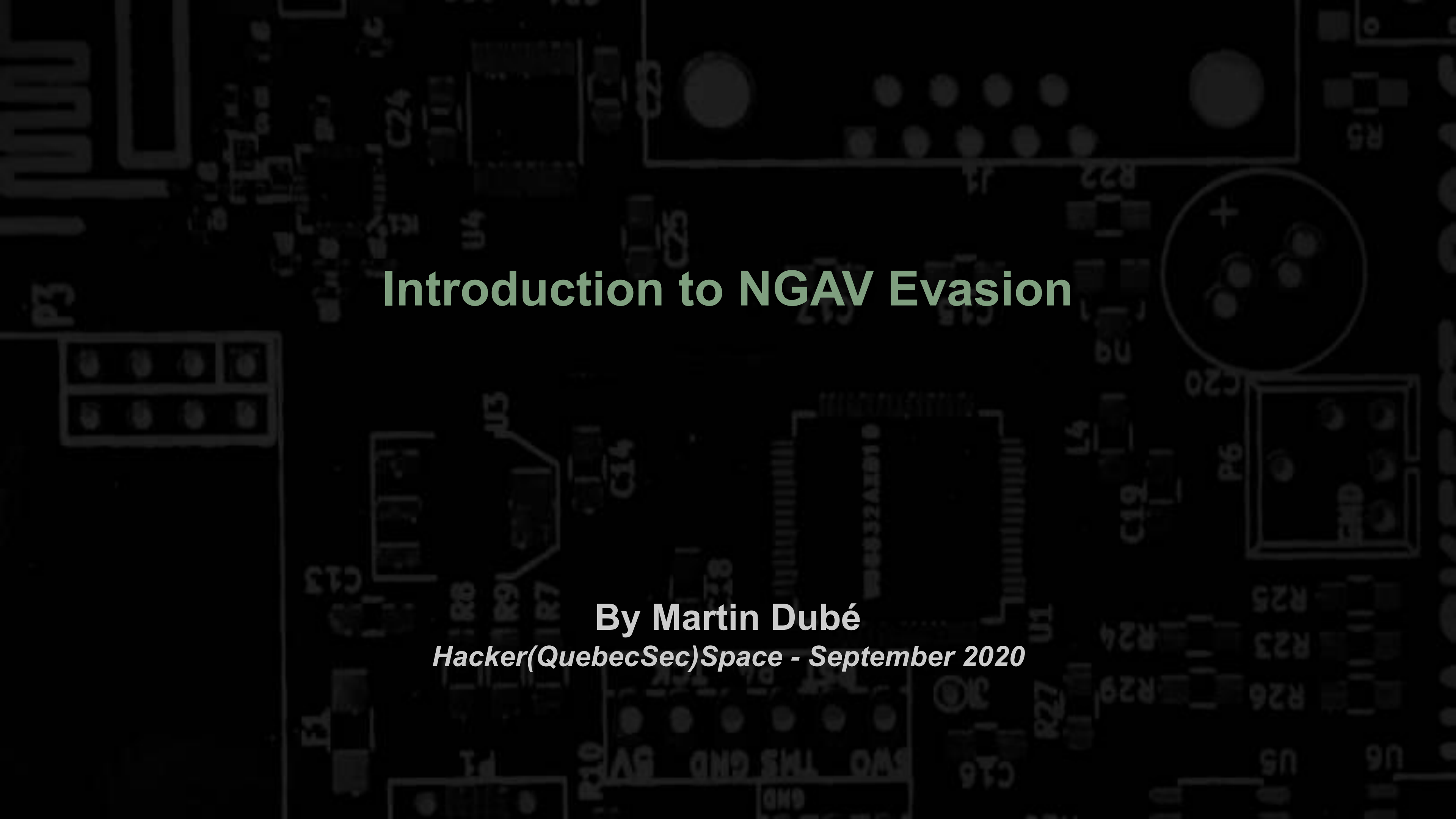


Introduction to NGAV Evasion

By Martin Dubé

Hacker(QuebecSec)Space - September 2020

Talk Objectives

- Introduction to malware development
- Share knowledge of Windows and Endpoint Security Solutions
- Have a good time

$ ./start.sh

- Intro
- Lab Setup
- Demos
  - Let’s build a loader
  - Let’s do recon
  - Let’s dump LSASS
  - [Bonus] Direct System Calls
- Conclusions
- Socialize on discord: https://discord.gg/39fRfa6

Spoiler Alert

- None of this is exclusive.
  - A lot of code snippet comes from https://www.ired.team/offensive-security/
  - Most techniques were developed by people smarter than me
- Use this knowledge with care.
  - I mean… don't attack other people's computer with these techniques…
- Opinions expressed are solely my own and do not express the views or opinions of my employer.

whoami

- [2018-2020] Red Teamer @ Financial Institution
  - Focus less on mitigations, more on detection (AV / EDR Evasion)
  - Focus on a single environment (unlike Consulting)
- [2013-2018] Pentester / Team Lead @ GoSecure
  - Jack of all trades, master of none
  - Say yes to any weird mandate
- [2010-2017] CTF Lead / Board Member / Enthusiast @ Hackfest
  - Particular interest on War Games and CTFs
- Secure by default thinking promoter
  - Hateful, sometimes hostile, about Windows
  - OpenBSD lover
- Woodworker on spare time

Evading AV (Hackerspace - Septembre 2019)

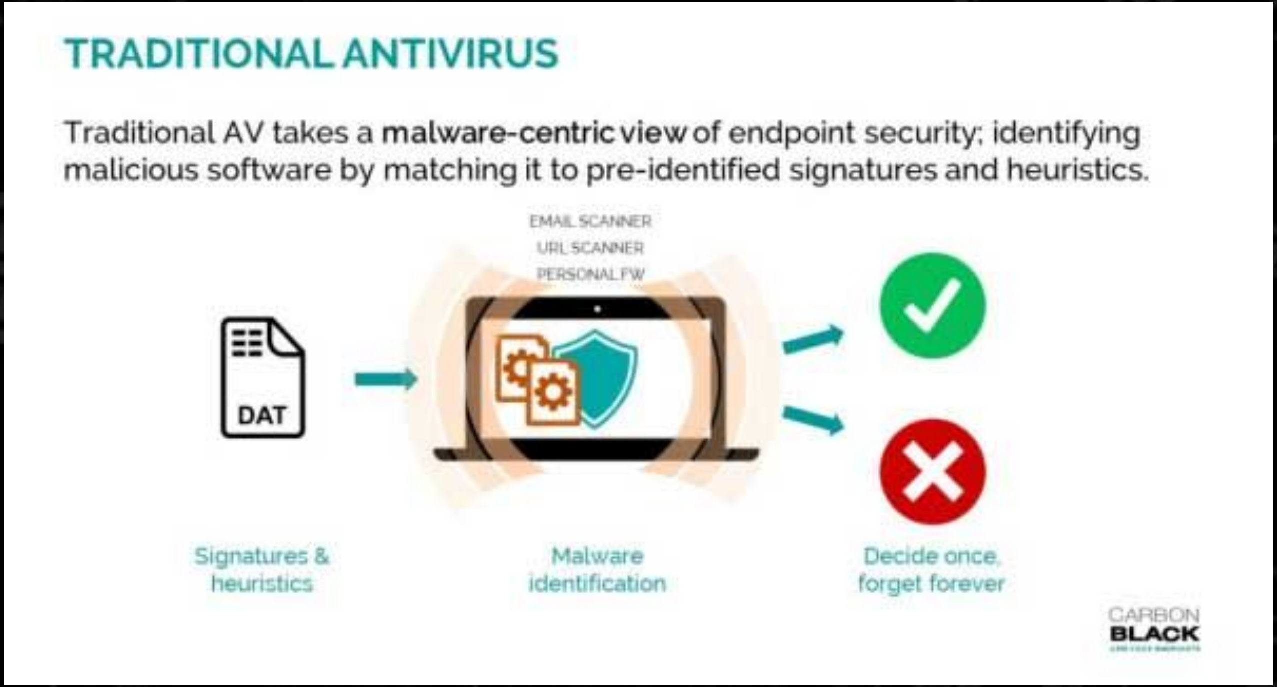
- We focused on traditional detection
  - Not behavioral
  - Not (that much) heuristics
  - Mostly signature based
- We figured out that AV are good at
  - Static+Runtime Analysis
  - They can open base64 encoded blobs 😱
  - They can analyze a script that call a script that call a script and so on.
- But they suck at
  - Actually Defending against Threats

Evading NGAV (Today's talk)

- We will focus on attacks (behavioral detection)
  - Process Injections
  - Execute-Assembly, Custom capabilities
  - Creds Dumping Attacks (Access to LSASS)

- We will not talk about
  - How to write or generate shellcode
  - How to deploy C2 infrastructure
  - How to evade a Blue Team or Threat Hunting (just a bit)

AV vs NGAV vs EDR

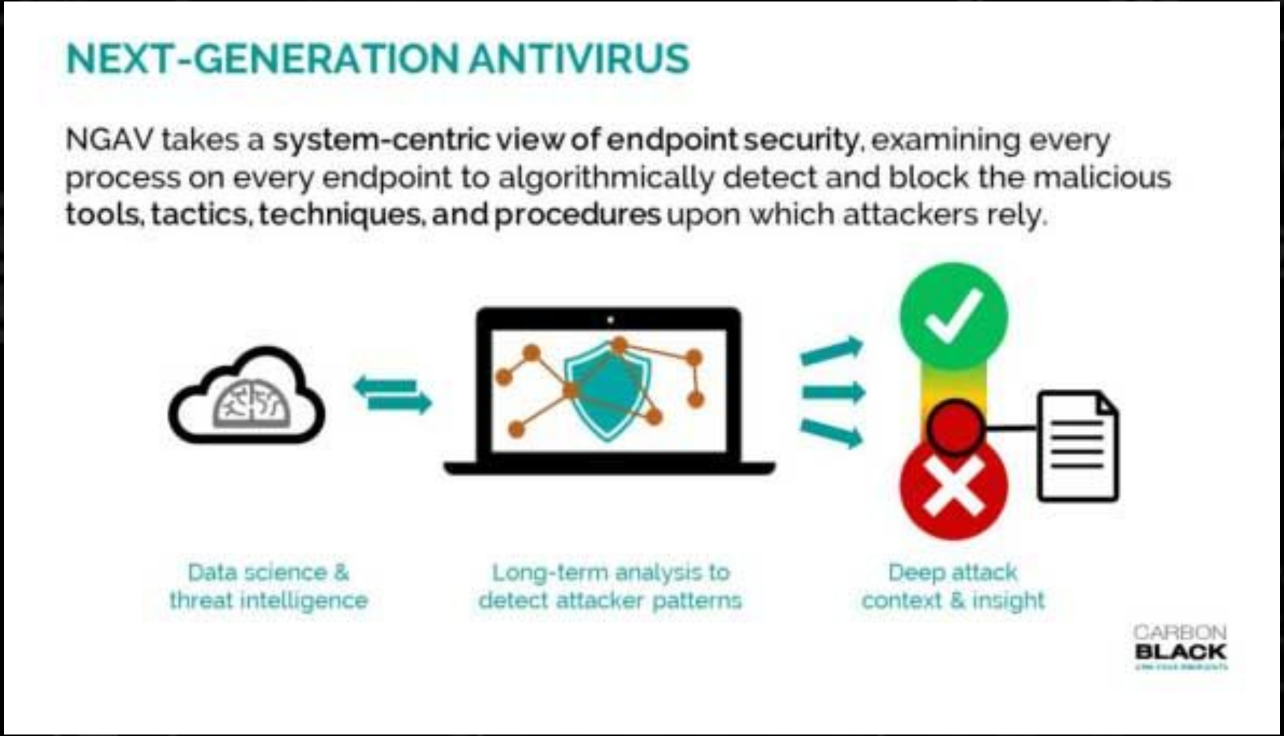


TRADITIONAL ANTIVIRUS

Traditional AV takes a **malware-centric view** of endpoint security; identifying malicious software by matching it to pre-identified signatures and heuristics.

Source: https://www.carbonblack.com/blog/next-generation-antivirus-ngav/

AV vs NGAV vs EDR



NEXT-GENERATION ANTIVIRUS

NGAV takes a **system-centric view of endpoint security**, examining every process on every endpoint to algorithmically detect and block the malicious **tools, tactics, techniques, and procedures** upon which attackers rely.

Data science & threat intelligence
Long-term analysis to detect attacker patterns
Deep attack context & insight
CARBON BLACK

Source: https://www.carbonblack.com/blog/next-generation-antivirus-ngav/

AV vs NGAV vs EDR

- ↱ Oriented on detection rather than mitigation
- ↱ Consolidate endpoints events from all endpoints
- ↱ Provide a full picture of potential threats
- ↱ Raise alerts so a Blue Team can respond
- ↱ Containment mechanism
- ↱ Machine Learning 😱
- ↱ Block Chain 😱
- ↱ AI 😱

Tactics, Techniques and Procedures (TTPs)

- Credential Access (tactics)
  - OS Credential Dumping -> LSASS Memory (technique)
    - Task Manager
    - Procdump
    - Mimikatz
    - Dumpert
    - Invoke-Mimikatz
    - C# Safetykatz
    - Rundll32 comsvcs.dll, MiniDUmp
    - C/C++ comsvcs.dll -> MiniDump
    - C/C++ MiniDumpWriteDump w/ PssCaptureSnapshot

    - Creativity is your limit!

Today’s coverage

- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Command and Control
- Exfiltration

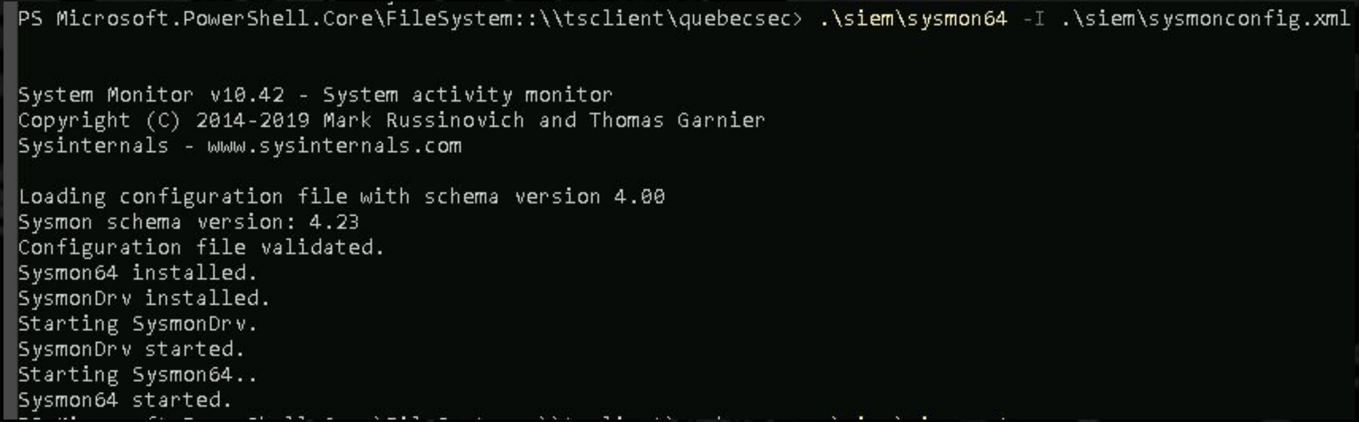
Lab

- A Windows 10 developer machine
  - With Visual Studio
  - A VBox shared folder (to share exe on all VMs)

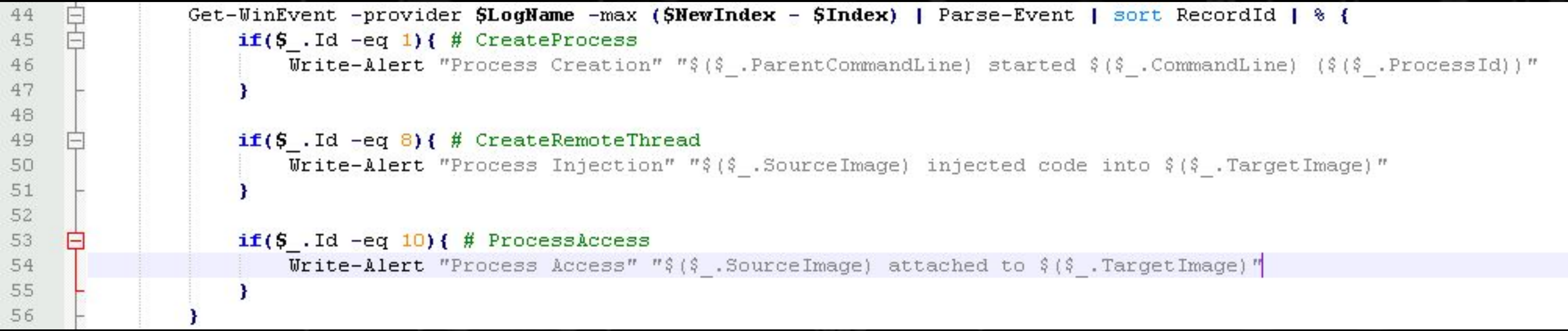
- VMs with NGAV/Basic EDR
  - Windows 10 - Cylance
  - Windows 10 - Defender+Sysmon

- A C2 infrastructure
  - A server (Cobalt Strike Team Server)
  - A client (Beacon shellcode)

Lab - Sysmon+Ps1 (0.05$ SIEM)

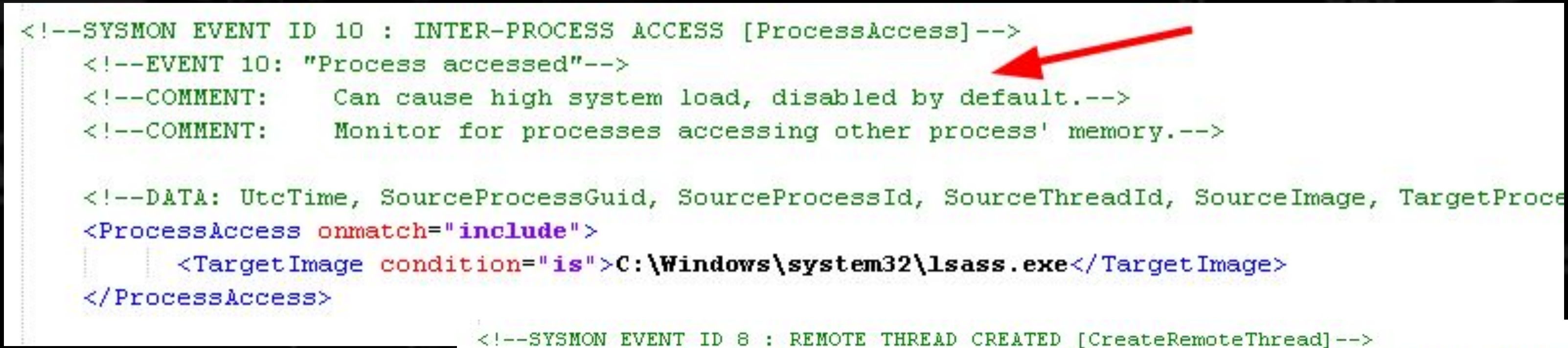


```
PS Microsoft.PowerShell.Core\FileSystem::\\tsclient\quebecsec> .\siem\sysmon64 -I .\siem\sysmonconfig.xml


System Monitor v10.42 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.00
Sysmon schema version: 4.23
Configuration file validated.
Sysmon64 installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon64..
Sysmon64 started.
```

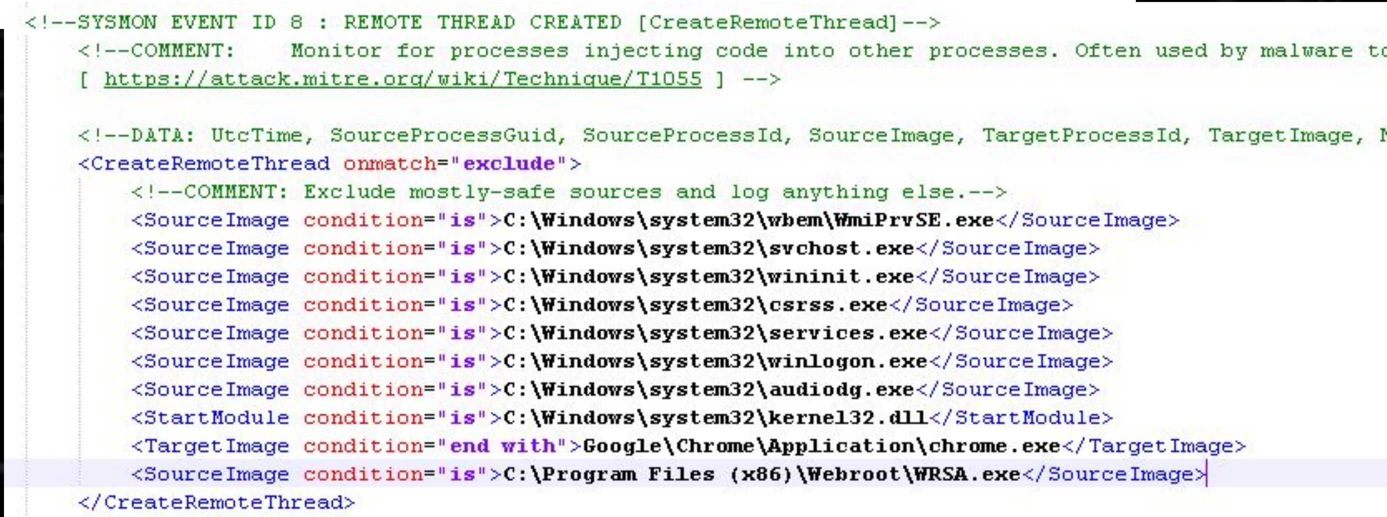


```
        Get-WinEvent -provider $LogName -max ($NewIndex - $Index) | Parse-Event | sort RecordId | % {
            if($_.Id -eq 1){ # CreateProcess
                Write-Alert "Process Creation" "$($_.ParentCommandLine) started $($_.CommandLine) ($($_.ProcessId))"
            }

            if($_.Id -eq 8){ # CreateRemoteThread
                Write-Alert "Process Injection" "$($_.SourceImage) injected code into $($_.TargetImage)"
            }

            if($_.Id -eq 10){ # ProcessAccess
                Write-Alert "Process Access" "$($_.SourceImage) attached to $($_.TargetImage)"
            }
        }
```

Lab - Sysmon+Ps1 (0.05$ SIEM)



```
<!--SYSMON EVENT ID 10 : INTER-PROCESS ACCESS [ProcessAccess]-->
    <!--EVENT 10: "Process accessed"-->
    <!--COMMENT:    Can cause high system load, disabled by default.-->
    <!--COMMENT:    Monitor for processes accessing other process' memory.-->

    <!--DATA: UtcTime, SourceProcessGuid, SourceProcessId, SourceThreadId, SourceImage, TargetProce
    <ProcessAccess onmatch="include">
          <TargetImage condition="is">C:\Windows\system32\lsass.exe</TargetImage>
    </ProcessAccess>
```



```
<!--SYSMON EVENT ID 8 : REMOTE THREAD CREATED [CreateRemoteThread]-->
    <!--COMMENT:    Monitor for processes injecting code into other processes. Often used by malware to
    [ https://attack.mitre.org/wiki/Technique/T1055 ] -->

    <!--DATA: UtcTime, SourceProcessGuid, SourceProcessId, SourceImage, TargetProcessId, TargetImage, N
    <CreateRemoteThread onmatch="exclude">
        <!--COMMENT: Exclude mostly-safe sources and log anything else.-->
        <SourceImage condition="is">C:\Windows\system32\wbem\WmiPrvSE.exe</SourceImage>
        <SourceImage condition="is">C:\Windows\system32\svchost.exe</SourceImage>
        <SourceImage condition="is">C:\Windows\system32\wininit.exe</SourceImage>
        <SourceImage condition="is">C:\Windows\system32\csrss.exe</SourceImage>
        <SourceImage condition="is">C:\Windows\system32\services.exe</SourceImage>
        <SourceImage condition="is">C:\Windows\system32\winlogon.exe</SourceImage>
        <SourceImage condition="is">C:\Windows\system32\audiodg.exe</SourceImage>
        <StartModule condition="is">C:\Windows\system32\kernel32.dll</StartModule>
        <TargetImage condition="end with">Google\Chrome\Application\chrome.exe</TargetImage>
        <SourceImage condition="is">C:\Program Files (x86)\Webroot\WRSA.exe</SourceImage>
    </CreateRemoteThread>
```

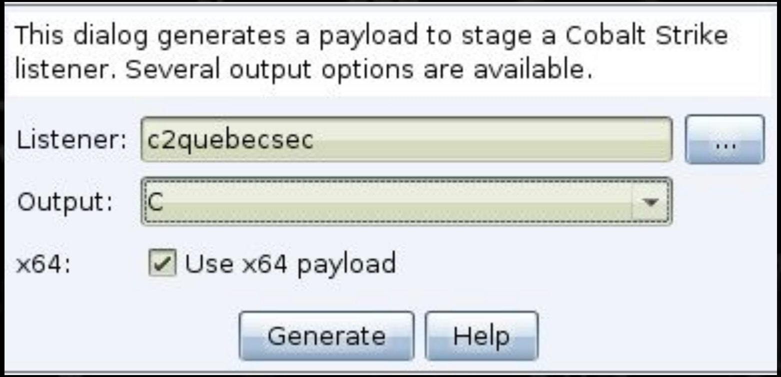
Let’s build a loader

- Also known as stage 0
- Goal: Put a malicious piece of software in memory and run it
- How?
  - Choose a format: Exe, DLL, HTA, Office Macro, Powershell, AutoIT, VBScript
- Challenges?
  - Shellcode will be detected
  - Many Injection Techniques are known
  - The NGAV may sandbox the binary and find the malicious content
  - Some Windows API are hooked in user-mode but some are in kernel-mode
  - Every addition of code can become an IOC (Indicator of Compromission)

Shellcode

- Grab a shellcode from your favourite C2
  - Cobalt Strike
  - Metasploit
  - Powershell Empire
  - Covenant
  - Silent Trinity
  - Shad0w
  - Mythic
  - Ninja

- For this presentation, we will use Cobalt Strike.


This dialog generates a payload to stage a Cobalt Strike listener. Several output options are available.

Listener: c2quebecsec

Output: C

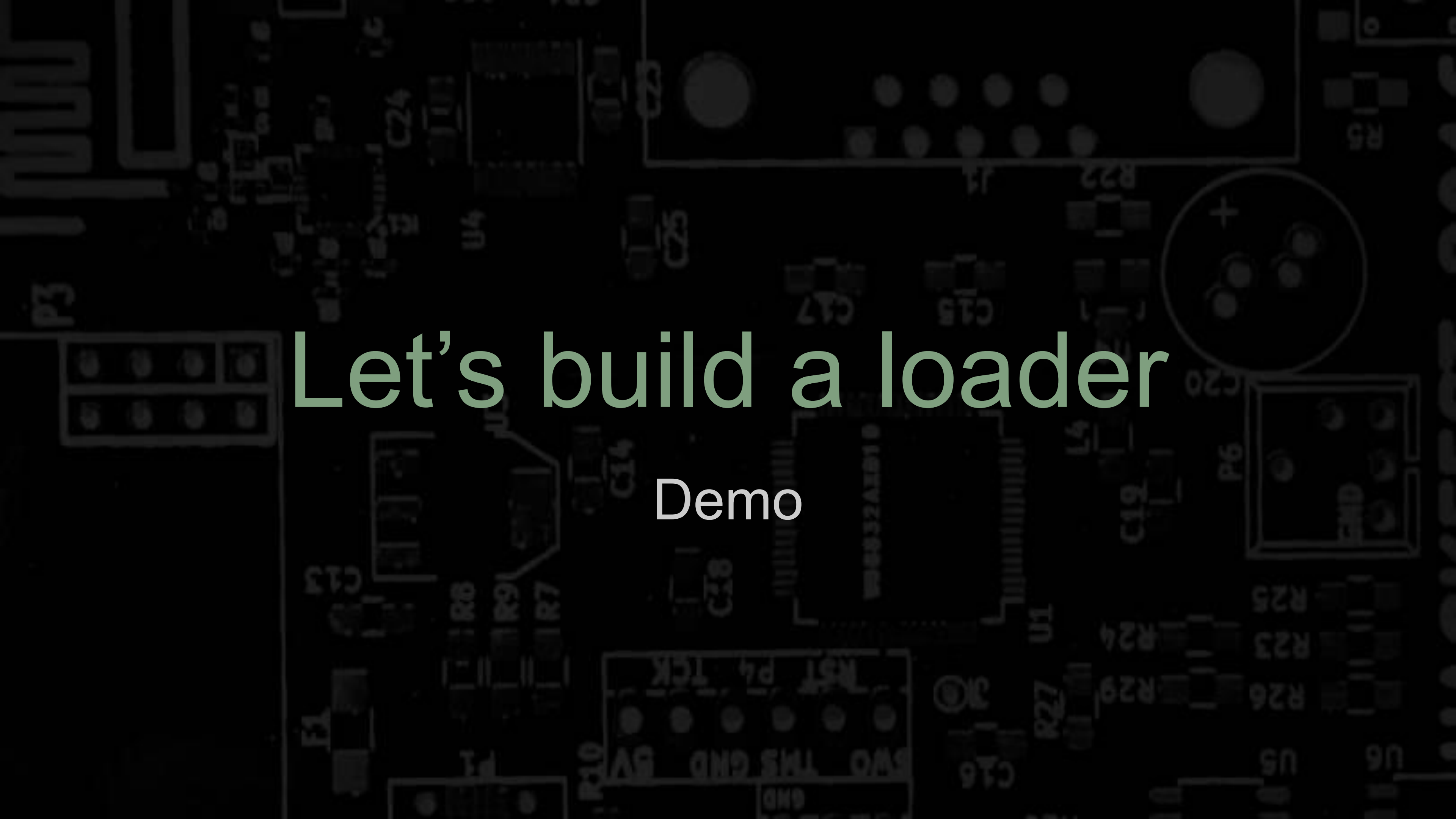
x64: [✓] Use x64 payload

Process Injection

- What is injection?
  - Allocate Memory (VirtualAllocEx, CreateFileMapping)
  - Write Memory (WriteProcessMemory, memcpy)
  - [optional] Change Memory Permissions (VirtualProtectEx, MapViewOfFile2)
  - Execute the payload (CreateThread, CreateRemoteThread, QueueUserAPC, RtlCreateUserThread, NtCreateThreadEx)

- Self-Inject
  - Will start a thread in the current process

- Remote-Inject
  - Will start a thread in another process



Let’s build a loader

Demo

Post-Exploitation

- All steps taken after initial access to achieve goal.

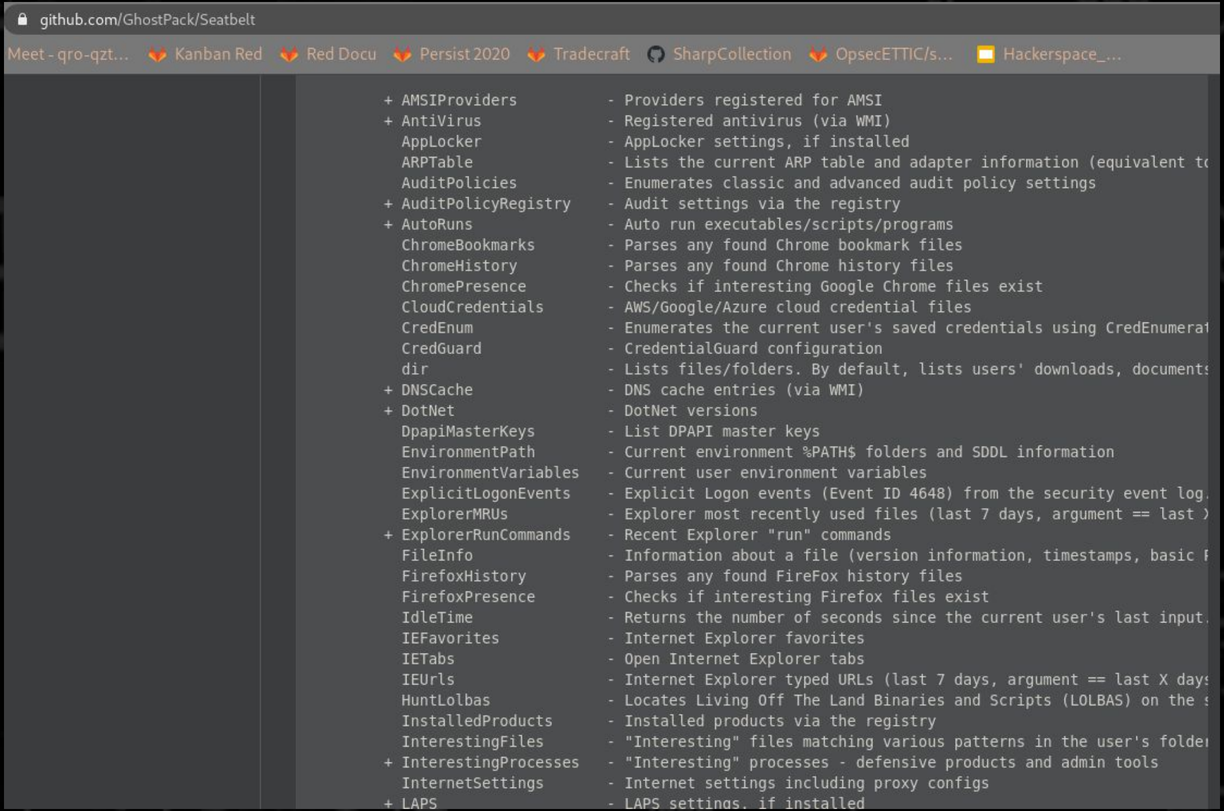
- How?
  - Powershell scripts (PowerView, PowerUp, PowerSploit, etc.)
  - LOLBAS (living off the land binaries and scripts)
  - Built-in beacon commands (ls, ps, rm, mv, upload, download, etc.)
  - Execute-assembly (Run C# .NET binary in memory)
  - Inline-execute (Run a C/C++ capability directly from the beacon)
  - Remote/Local DLL Injection (Run C/C++ DLL in memory)
  - Remote/Local Shellcode Injection (Run ASM in memory)

  - It's all about tradeoff.

Discovery

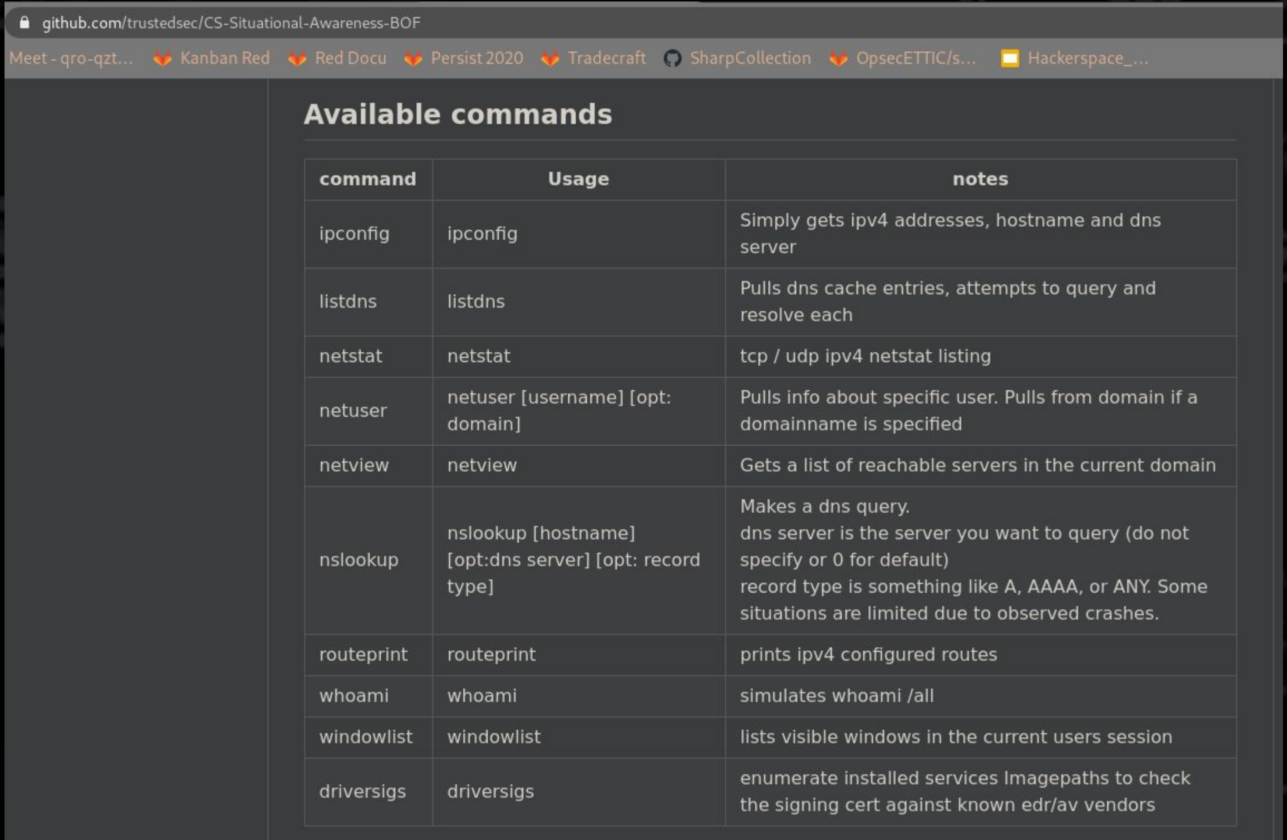
- Goal: Figure out where we are and what is accessible around
- Challenges?
  - Powershell is usually logged
  - Recon tools are usually detected (and even eradicated) by NGAV
  - Against an EDR, we assume that all commands are logged (process creation)
  - Our malware might be killed if a capability is detected (Solution: fork’n run)

Discovery



```
+ AMSIProviders           - Providers registered for AMSI
+ AntiVirus               - Registered antivirus (via WMI)
  AppLocker               - AppLocker settings, if installed
  ARPTable                - Lists the current ARP table and adapter information (equivalent tc
  AuditPolicies           - Enumerates classic and advanced audit policy settings
+ AuditPolicyRegistry     - Audit settings via the registry
+ AutoRuns                - Auto run executables/scripts/programs
  ChromeBookmarks         - Parses any found Chrome bookmark files
  ChromeHistory           - Parses any found Chrome history files
  ChromePresence          - Checks if interesting Google Chrome files exist
  CloudCredentials        - AWS/Google/Azure cloud credential files
  CredEnum                - Enumerates the current user's saved credentials using CredEnumerat
  CredGuard               - CredentialGuard configuration
  dir                     - Lists files/folders. By default, lists users' downloads, documents
+ DNSCache                - DNS cache entries (via WMI)
+ DotNet                  - DotNet versions
  DpapiMasterKeys         - List DPAPI master keys
  EnvironmentPath         - Current environment %PATH$ folders and SDDL information
  EnvironmentVariables    - Current user environment variables
  ExplicitLogonEvents     - Explicit Logon events (Event ID 4648) from the security event log.
  ExplorerMRUs            - Explorer most recently used files (last 7 days, argument == last X
+ ExplorerRunCommands     - Recent Explorer "run" commands
  FileInfo                - Information about a file (version information, timestamps, basic P
  FirefoxHistory          - Parses any found FireFox history files
  FirefoxPresence         - Checks if interesting Firefox files exist
  IdleTime                - Returns the number of seconds since the current user's last input.
  IEFavorites             - Internet Explorer favorites
  IETabs                  - Open Internet Explorer tabs
  IEUrls                  - Internet Explorer typed URLs (last 7 days, argument == last X days
  HuntLolbas              - Locates Living Off The Land Binaries and Scripts (LOLBAS) on the s
  InstalledProducts       - Installed products via the registry
  InterestingFiles        - "Interesting" files matching various patterns in the user's folder
+ InterestingProcesses    - "Interesting" processes - defensive products and admin tools
  InternetSettings        - Internet settings including proxy configs
+ LAPS                    - LAPS settings, if installed
```

Discovery



github.com/trustedsec/CS-Situational-Awareness-BOF

Meet - qro-qzt... Kanban Red Red Docu Persist 2020 Tradecraft SharpCollection OpsecETTIC/s... Hackerspace_...

Available commands

command	Usage	notes
ipconfig	ipconfig	Simply gets ipv4 addresses, hostname and dns server
listdns	listdns	Pulls dns cache entries, attempts to query and resolve each
netstat	netstat	tcp / udp ipv4 netstat listing
netuser	netuser [username] [opt: domain]	Pulls info about specific user. Pulls from domain if a domainname is specified
netview	netview	Gets a list of reachable servers in the current domain
nslookup	nslookup [hostname] [opt:dns server] [opt: record type]	Makes a dns query. dns server is the server you want to query (do not specify or 0 for default) record type is something like A, AAAA, or ANY. Some situations are limited due to observed crashes.
routeprint	routeprint	prints ipv4 configured routes
whoami	whoami	simulates whoami /all
windowlist	windowlist	lists visible windows in the current users session
driversigs	driversigs	enumerate installed services Imagepaths to check the signing cert against known edr/av vendors

Credentials Access

- Credentials are essential to most red team operations
- How?
  - Task Manager
  - Procdump
  - Mimikatz
  - Dumpert
  - Invoke-Mimikatz
  - C# Safetykatz
  - Rundll32 comsvcs.dll, MiniDUmp
  - C/C++ comsvcs.dll -> MiniDump
  - C/C++ MiniDumpWriteDump w/ PssCaptureSnapshot

- Challenges?
  - lsass.exe is highly monitored

Let’s play

Demo

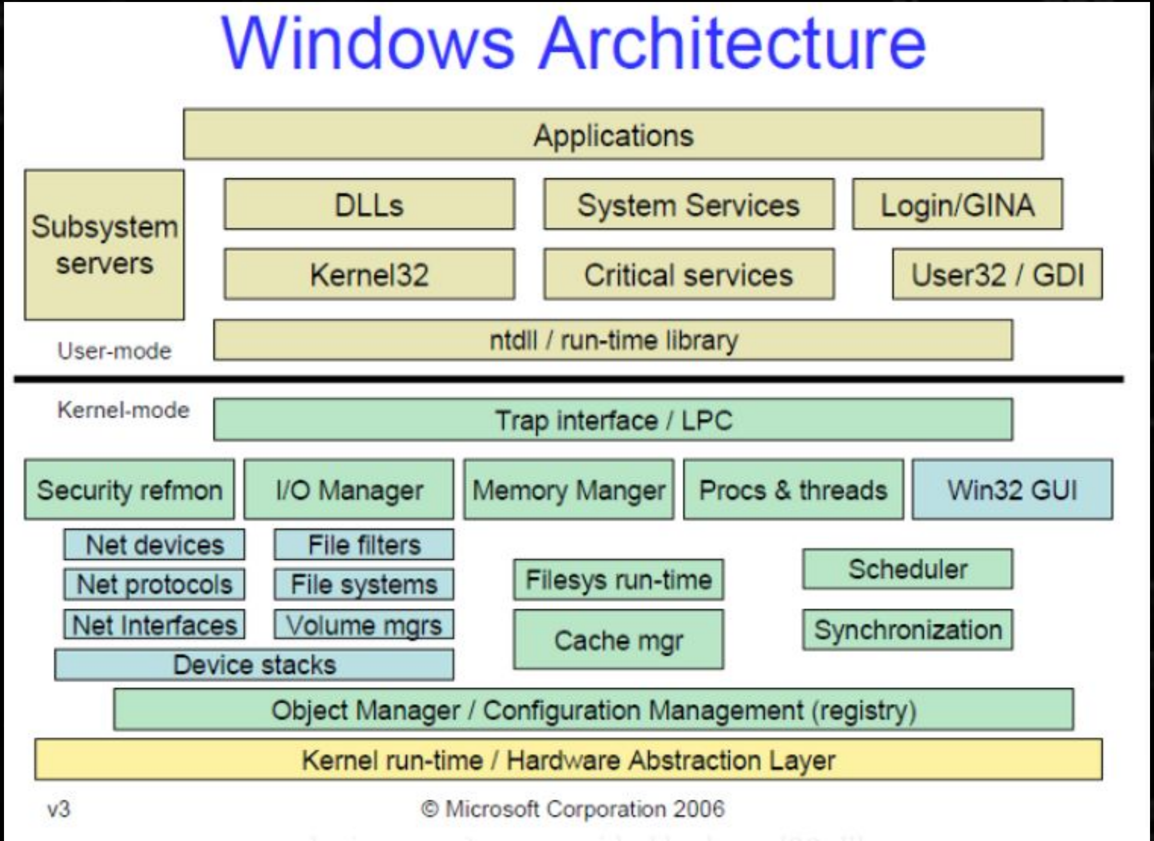
Direct System Calls

- This technique will bypass ALL user-mode hooks
  - I learned the hard way that it is not efficient against sysmon…
  - However, a lot of EDR in the wild still have userland hooks

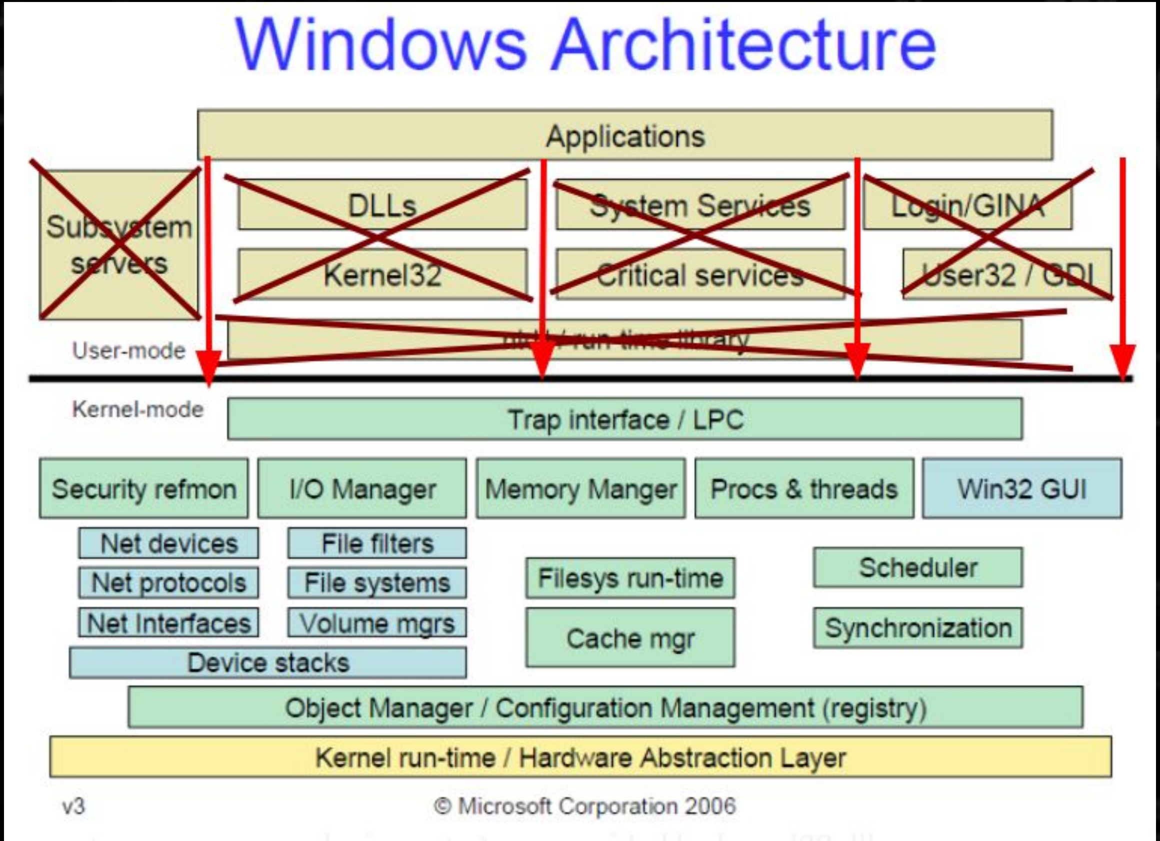
- Caveats
  - Syscalls numbers change across windows versions so it's hard to maintain
  - The program will perform syscalls which is not common
  - Do not bypass kernel-mode hooks

- Epic Source: https://outflank.nl/blog/2019/06/19/red-team-tactics-combining-direct-system-calls-and-srdi-to-bypass-av-edr/

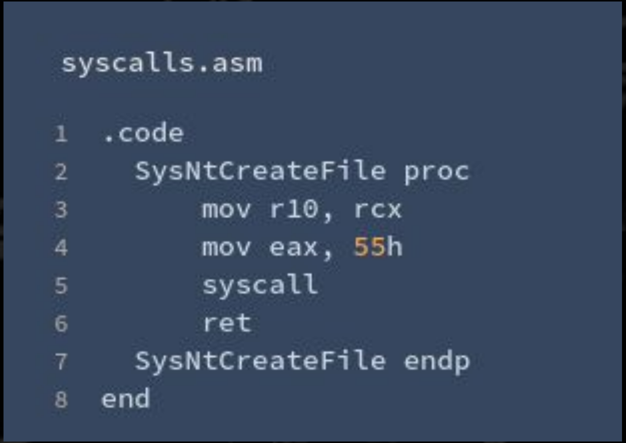
Direct System Calls



Direct System Calls



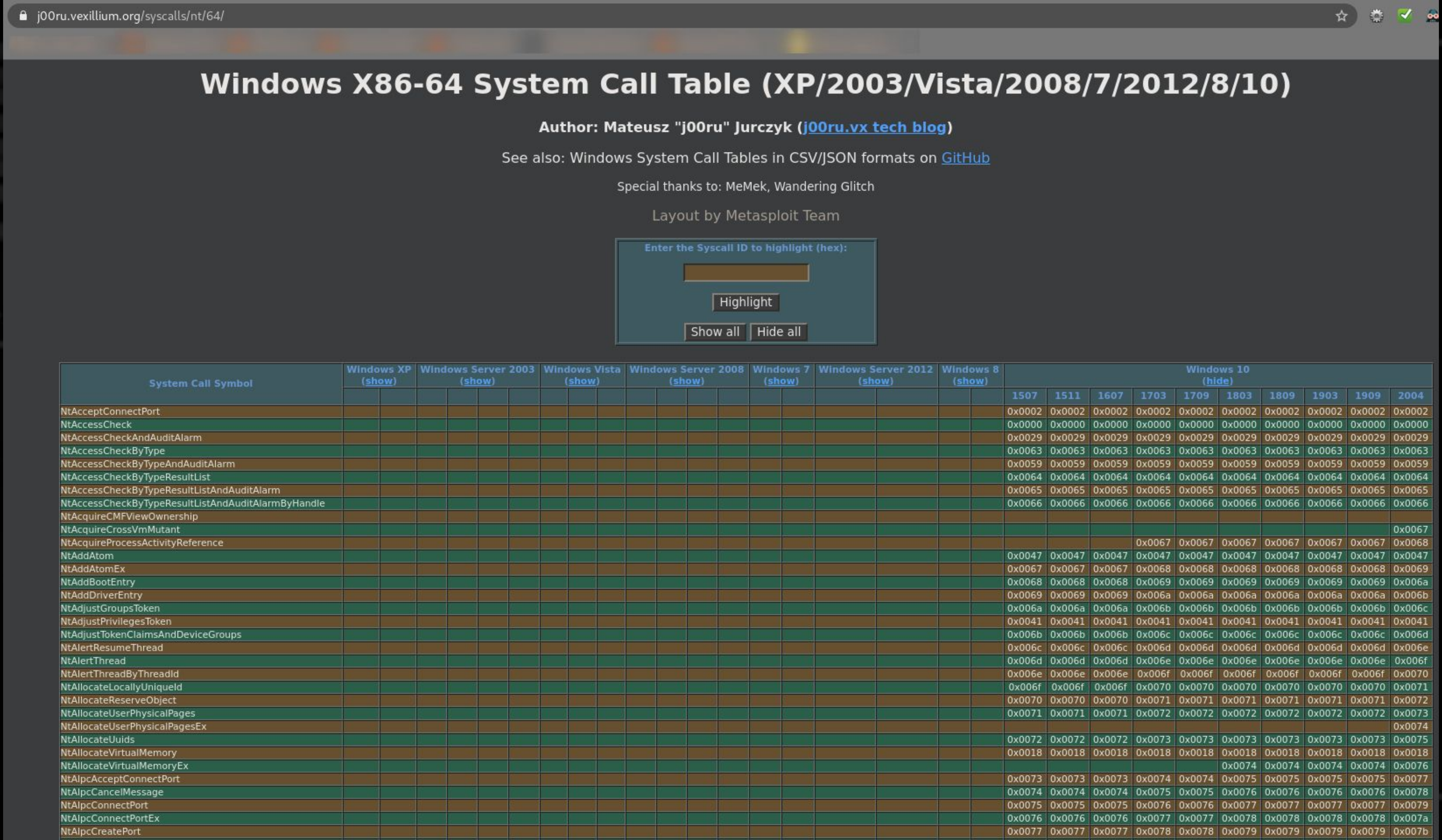
Direct System Calls



syscalls.asm

```
.code
  SysNtCreateFile proc
      mov r10, rcx
      mov eax, 55h
      syscall
      ret
  SysNtCreateFile endp
end
```

Source: https://www.ired.team/offensive-security/defense-evasion/using-syscalls-directly-from-visual-studio-to-bypass-avs-edrs



j00ru.vexillium.org/syscalls/nt/64/

Windows X86-64 System Call Table (XP/2003/Vista/2008/7/2012/8/10)

Author: Mateusz "j00ru" Jurczyk (j00ru.vx tech blog)

See also: Windows System Call Tables in CSV/JSON formats on GitHub

Special thanks to: MeMek, Wandering Glitch

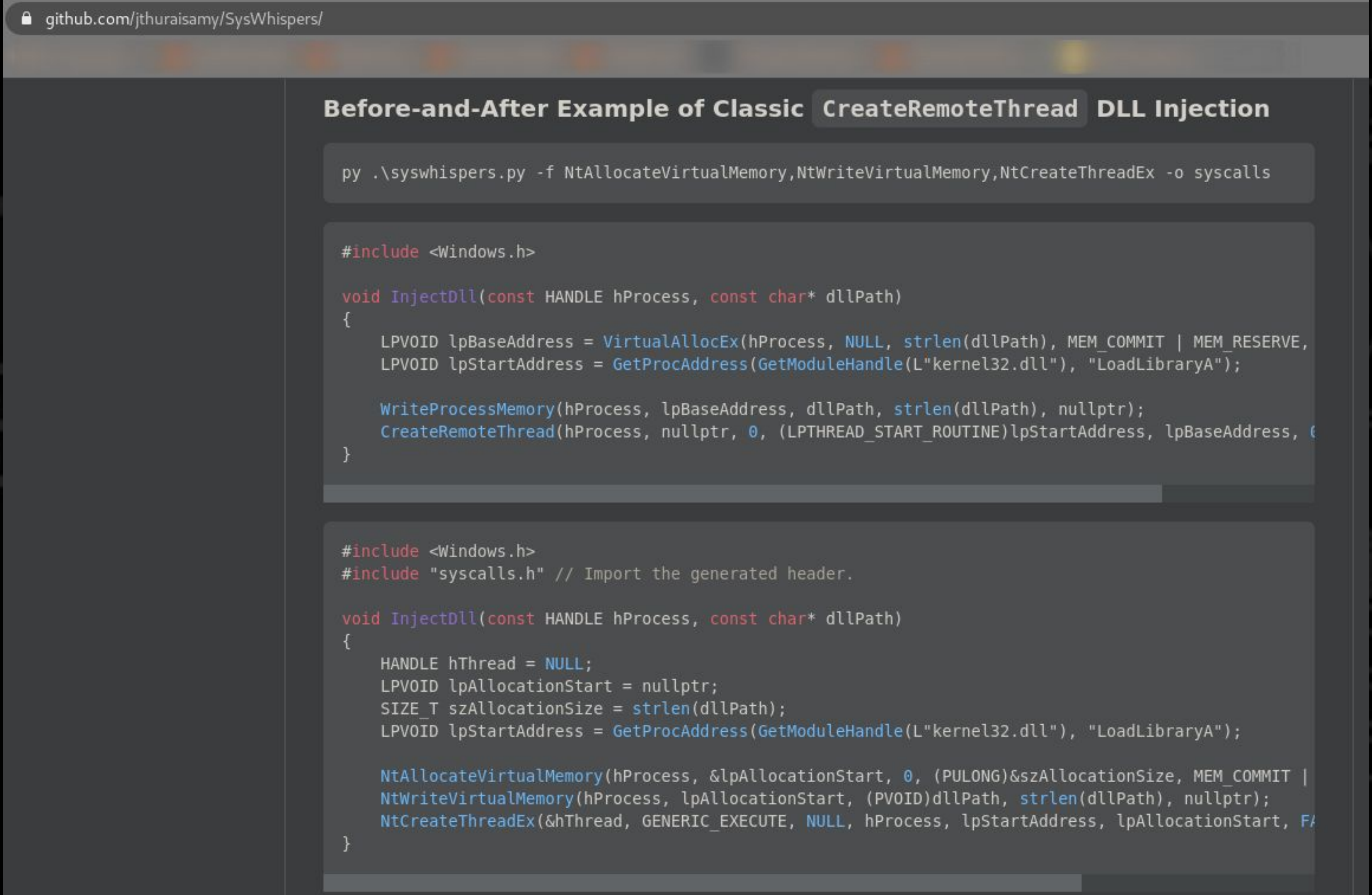
Layout by Metasploit Team

Enter the Syscall ID to highlight (hex):

Highlight

Show all Hide all

System Call Symbol	Windows XP (show)		Windows Server 2003 (show)				Windows Vista (show)			Windows Server 2008 (show)				Windows 7 (show)		Windows Server 2012 (show)		Windows 8 (show)		Windows 10 (hide)									
																				1507	1511	1607	1703	1709	1803	1809	1903	1909	2004
NtAcceptConnectPort																				0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002
NtAccessCheck																				0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
NtAccessCheckAndAuditAlarm																				0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029
NtAccessCheckByType																				0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063
NtAccessCheckByTypeAndAuditAlarm																				0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059
NtAccessCheckByTypeResultList																				0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064
NtAccessCheckByTypeResultListAndAuditAlarm																				0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065
NtAccessCheckByTypeResultListAndAuditAlarmByHandle																				0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066
NtAcquireCMFViewOwnership																													
NtAcquireCrossVmMutant																													0x0067
NtAcquireProcessActivityReference																							0x0067	0x0067	0x0067	0x0067	0x0067	0x0067	0x0068
NtAddAtom																				0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047
NtAddAtomEx																				0x0067	0x0067	0x0067	0x0068	0x0068	0x0068	0x0068	0x0068	0x0068	0x0069
NtAddBootEntry																				0x0068	0x0068	0x0068	0x0069	0x0069	0x0069	0x0069	0x0069	0x0069	0x006a
NtAddDriverEntry																				0x0069	0x0069	0x0069	0x006a	0x006a	0x006a	0x006a	0x006a	0x006a	0x006b
NtAdjustGroupsToken																				0x006a	0x006a	0x006a	0x006b	0x006b	0x006b	0x006b	0x006b	0x006b	0x006c
NtAdjustPrivilegesToken																				0x0041	0x0041	0x0041	0x0041	0x0041	0x0041	0x0041	0x0041	0x0041	0x0041
NtAdjustTokenClaimsAndDeviceGroups																				0x006b	0x006b	0x006b	0x006c	0x006c	0x006c	0x006c	0x006c	0x006c	0x006d
NtAlertResumeThread																				0x006c	0x006c	0x006c	0x006d	0x006d	0x006d	0x006d	0x006d	0x006d	0x006e
NtAlertThread																				0x006d	0x006d	0x006d	0x006e	0x006e	0x006e	0x006e	0x006e	0x006e	0x006f
NtAlertThreadByThreadId																				0x006e	0x006e	0x006e	0x006f	0x006f	0x006f	0x006f	0x006f	0x006f	0x0070
NtAllocateLocallyUniqueId																				0x006f	0x006f	0x006f	0x0070	0x0070	0x0070	0x0070	0x0070	0x0070	0x0071
NtAllocateReserveObject																				0x0070	0x0070	0x0070	0x0071	0x0071	0x0071	0x0071	0x0071	0x0071	0x0072
NtAllocateUserPhysicalPages																				0x0071	0x0071	0x0071	0x0072	0x0072	0x0072	0x0072	0x0072	0x0072	0x0073
NtAllocateUserPhysicalPagesEx																													0x0074
NtAllocateUuids																				0x0072	0x0072	0x0072	0x0073	0x0073	0x0073	0x0073	0x0073	0x0073	0x0075
NtAllocateVirtualMemory																				0x0018	0x0018	0x0018	0x0018	0x0018	0x0018	0x0018	0x0018	0x0018	0x0018
NtAllocateVirtualMemoryEx																									0x0074	0x0074	0x0074	0x0074	0x0076
NtAlpcAcceptConnectPort																				0x0073	0x0073	0x0073	0x0074	0x0074	0x0075	0x0075	0x0075	0x0075	0x0077
NtAlpcCancelMessage																				0x0074	0x0074	0x0074	0x0075	0x0075	0x0076	0x0076	0x0076	0x0076	0x0078
NtAlpcConnectPort																				0x0075	0x0075	0x0075	0x0076	0x0076	0x0077	0x0077	0x0077	0x0077	0x0079
NtAlpcConnectPortEx																				0x0076	0x0076	0x0076	0x0077	0x0077	0x0078	0x0078	0x0078	0x0078	0x007a
NtAlpcCreatePort																				0x0077	0x0077	0x0077	0x0078	0x0078	0x0079	0x0079	0x0079	0x0079	0x007b


github.com/jthuraisamy/SysWhispers/

Before-and-After Example of Classic `CreateRemoteThread` DLL Injection

```
py .\syswhispers.py -f NtAllocateVirtualMemory,NtWriteVirtualMemory,NtCreateThreadEx -o syscalls
```

```
#include <Windows.h>

void InjectDll(const HANDLE hProcess, const char* dllPath)
{
    LPVOID lpBaseAddress = VirtualAllocEx(hProcess, NULL, strlen(dllPath), MEM_COMMIT | MEM_RESERVE,
    LPVOID lpStartAddress = GetProcAddress(GetModuleHandle(L"kernel32.dll"), "LoadLibraryA");

    WriteProcessMemory(hProcess, lpBaseAddress, dllPath, strlen(dllPath), nullptr);
    CreateRemoteThread(hProcess, nullptr, 0, (LPTHREAD_START_ROUTINE)lpStartAddress, lpBaseAddress,
}
```

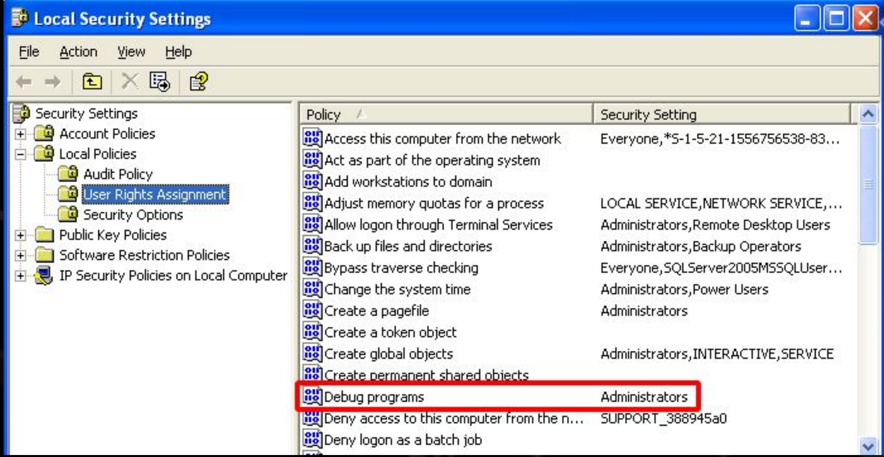
```
#include <Windows.h>
#include "syscalls.h" // Import the generated header.

void InjectDll(const HANDLE hProcess, const char* dllPath)
{
    HANDLE hThread = NULL;
    LPVOID lpAllocationStart = nullptr;
    SIZE_T szAllocationSize = strlen(dllPath);
    LPVOID lpStartAddress = GetProcAddress(GetModuleHandle(L"kernel32.dll"), "LoadLibraryA");

    NtAllocateVirtualMemory(hProcess, &lpAllocationStart, 0, (PULONG)&szAllocationSize, MEM_COMMIT |
    NtWriteVirtualMemory(hProcess, lpAllocationStart, (PVOID)dllPath, strlen(dllPath), nullptr);
    NtCreateThreadEx(&hThread, GENERIC_EXECUTE, NULL, hProcess, lpStartAddress, lpAllocationStart, F
}
```

Direct System Calls

Demo

Conclusions

- Windows is not secure by default
- Harden your endpoints
  - Don’t provide admin rights to users
  - Disable Debug Privileges
- Harden Windows Defender
  - Attack Surface Reduction (ASR)
  - ATP (EDR)



- Security Settings
  - Account Policies
  - Local Policies
    - Audit Policy
    - User Rights Assignment
    - Security Options
  - Public Key Policies
  - Software Restriction Policies
  - IP Security Policies on Local Computer

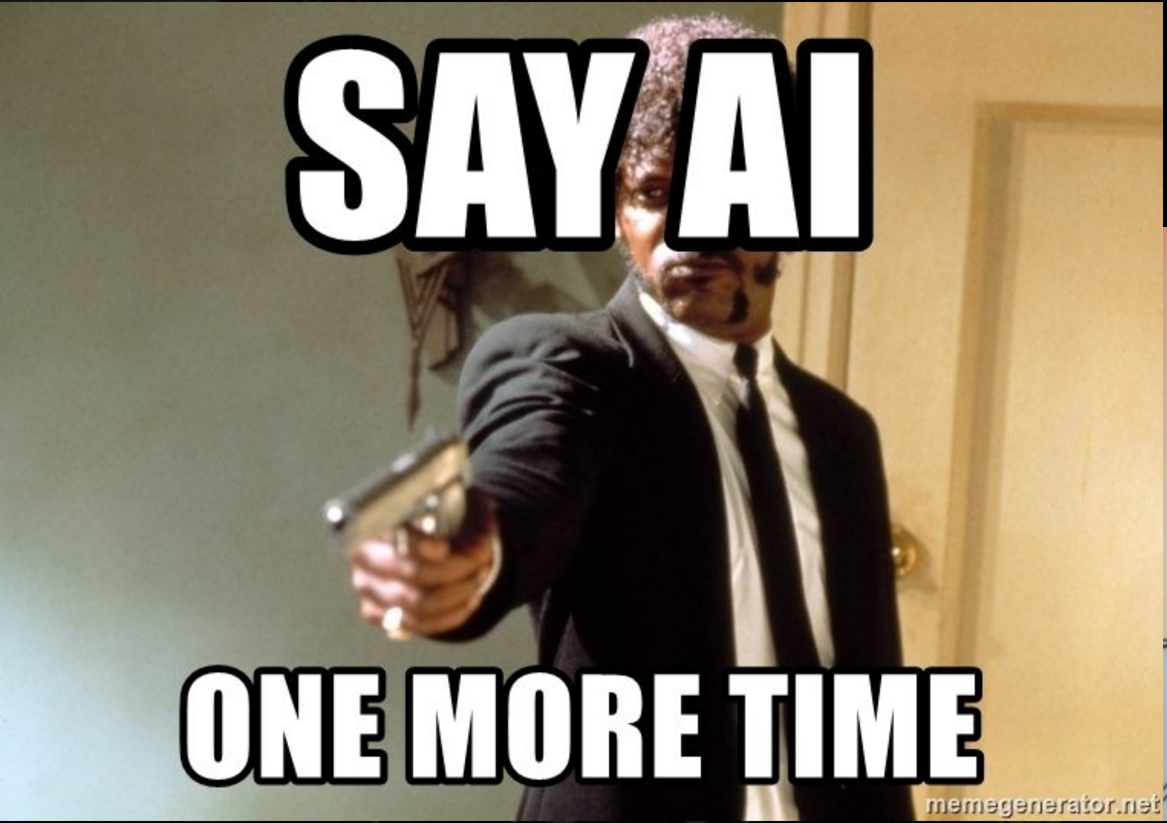
Policy	Security Setting
Access this computer from the network	Everyone,*S-1-5-21-1556756538-83...
Act as part of the operating system	
Add workstations to domain	
Adjust memory quotas for a process	LOCAL SERVICE,NETWORK SERVICE,...
Allow logon through Terminal Services	Administrators,Remote Desktop Users
Back up files and directories	Administrators,Backup Operators
Bypass traverse checking	Everyone,SQLServer2005MSSQLUser...
Change the system time	Administrators,Power Users
Create a pagefile	Administrators
Create a token object	
Create global objects	Administrators,INTERACTIVE,SERVICE
Create permanent shared objects	
Debug programs	Administrators
Deny access to this computer from the n...	SUPPORT_388945a0
Deny logon as a batch job	

Conclusions

- Do more Purple Team!
  - Build your security before testing it.
  - Test your security products before buying it
    - Ask the offense guys to lunch attacks
    - Ask the defense guys to detect and respond
  - Customize your monitoring
    - Buy products that are customizable to your needs
    - Default configuration of sysmon does not detect much by default
    - The default use cases of a SIEM usually suck

Conclusions

What about AI?



Any program with conditional logic
The Media
Is this Artificial Intelligence?

Thank You

Merci!

See you on discord: https://discord.gg/39fRfa6