# The Light Side of the Force

## PowerShell for Incident Responders

Steve Anson

www.AppliedIncidentResponse.com

@ForwardDefense

# PowerShell can be used for Evil

- Empire
- Death Star

# Let’s see how it can be used for good

# PowerShell for Padawans

- PowerShell is object oriented
- Objects have methods and properties
- Pipeline moves objects, not text, down the pipe
- Uses a verb-noun structure for cmdlets
  - Get-Process
  - Set-NetIPAddress

# PowerShell for Padawans

- Windows PowerShell

| PS Version | Included with: |
|---|---|
| 1.0 | Server 2008 |
| 2.0 | Server 2008 R2, Windows 7 |
| 3.0 | Server 2012, Windows 8 |
| 4.0 | Server 2012 R2, Windows 8.1 |
| 5.0 | Windows 10 |
| 5.1 | Server 2016, Windows 10 Anniversary Update |

- PowerShell Core 6
  - Works on Windows, Linux, MacOS
  - Reduced set of cmdlets

# PowerShell for Padawans

- Get-help
- Help
- Get-help –ShowWindow
- Get-Command
- Get-Member
- PowerShell ISE

# PowerShell Remoting for Padawans

- Web Services Management (WSMan) – SOAP based, open standard for managing IT resources of HTTP
- Windows Remote Management (WinRM) – Microsoft's implementation of WSMan for Windows systems
- HTTP on TCP 5985 (default)
- HTTPS on TCP 5986 (to support NTLM)
- All traffic encrypted, even over HTTP
- Connect by computer name, not IP

# PowerShell Remoting for Padawans

- WinRM enabled by default on Server 2012 and up
- To enable on clients or older servers use GPO

  Computer Configuration | Policies | Administrative Templates | Windows Components | Windows Remote Management (WinRM) | WinRM Service
- Enable “Allow Remote Server Management Through WinRM” and set both IP filters to *
- Also use GPO to allow access through Windows Firewall

  Computer Configuration | Policies |Windows Settings | Security Settings | Windows Firewall with Advanced Security
- Set WinRM service to automatically start in the following GPO

  Computer Configuration | Policies |Windows Settings | Security Settings | System Services

# PowerShell Remoting for Padawans

- To enable PowerShell on one machine locally
  - From PowerShell use *Enable-PSRemoting*
  - From cmd.exe use *winrm quickconfig*
- To enable PowerShell on one machine remotely
  - Psexec \\*Computer* -s winrm.cmd quickconfig –q
  - Wmic /node:*Computer* process call create “winrm quickconfig”

# PowerShell Remoting for Padawans

- Sometimes, remote access does not equal “Remoting”
- -ComputerName parameter may use RPC (pre Core 6) in cmdlets like:

Add-Computer
Clear-EventLog
Get-EventLog
Get-HotFix
Get-Process
Get-PSSession
Get-Service
Get-WmiObject
Invoke-WmiMethod
Limit-EventLog
New-EventLog
Register-WmiEvent
Remove-Computer
Remove-EventLog
Remove-WmiObject
Rename-Computer
Restart-Computer
Set-Service
Set-WmiInstance
Show-EventLog
Stop-Computer
Test-Connection
Write-EventLog

# PowerShell Cmdlets for the Alliance

- Get-Process
- Get-Service
- Get-ItemProperty (HKLM:\Software\Microsoft\Windows\CurrentVersion\Run)
- Get-ADComputer

# PowerShell Cmdlets for the Alliance

- Where-Object
- Select-Object
- Sort-Object
- Group-Object
- Measure-Object

# PowerShell Commands for the Alliance

- Format-Table
- Format-List
- Export-CSV

# Protecting Your Credentials

- Interactive logons expose your credentials in RAM
- Mimikatz is waiting
- PowerShell Remoting protects your credentials

# One-to-One Remoting

- Enter-PSSession –ComputerName *Computer*
- Like ssh for Windows

# One-to-Many Remoting

- Invoke-Command –ComputerName *name1, name2, name3* –ScriptBlock {Get-Process | Where-Object name –eq svchost | Get-Process –FileVersionInfo | Group-Object FileName}

# One-to-Many Remoting

- $s = New-PSSession -ComputerName (Get-Content Servers.txt) -Credential Domain\Administrator
- Invoke-Command -Session $s -ScriptBlock {*script1*}
- Invoke-Command -Session $s -ScriptBlock {*script2*}
- Remove-PSSession -Session $s

# Bring Reinforcements

- Need more help, push executables to remote machines and run them (Rekall, Autoruns, etc.)
- Copy-Item
- Start-Process

# PowerShell for Jedi

- Common Information Model (CIM) is an open standard defining a common set of objects and relationships for managed IT resources
- Windows Management Instrumentation (WMI) is Microsoft's implementation of CIM
- Can be accessed via wmic, VBScripts, and PowerShell

# PowerShell for Jedi

- Get-WMIObject and other WMI cmdlets are the older PowerShell way to access WMI. Use RPC/DCOM to connect to other systems with -ComputerName parameter
- Get-CIMInstance and other CIM cmdlets are the new way. Use WinRM for connecting to remote systems.

# PowerShell for Jedi

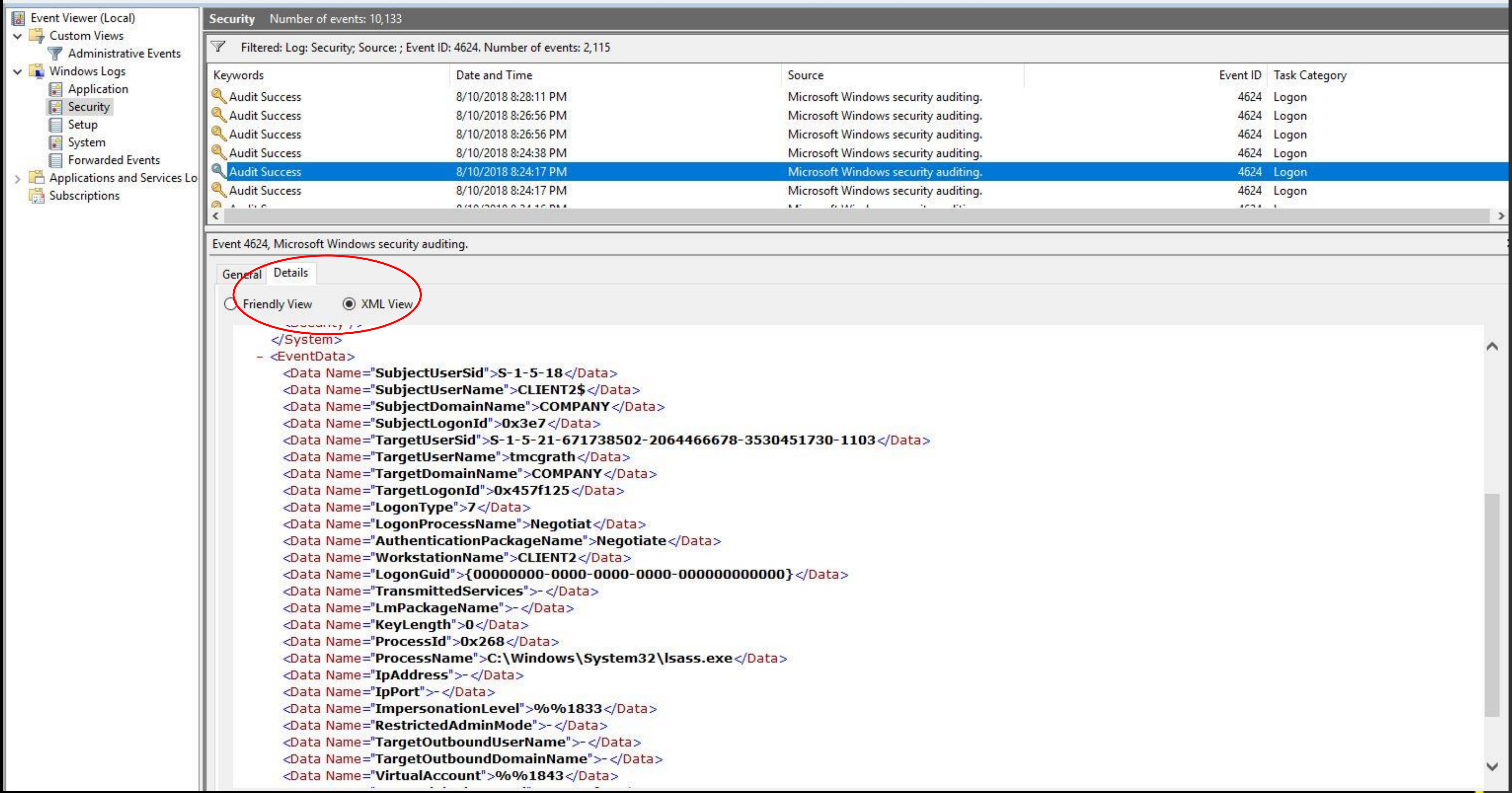
```
Get-CimInstance -ClassName Win32_BIOS
Get-CimInstance -ClassName Win32_Processor
Get-CimInstance -ClassName Win32_ComputerSystem
Get-CimInstance -ClassName Win32_Process
Get-CimInstance -ClassName Win32_QuickFixEngineering
Get-CimInstance -ClassName Win32_LogicalDisk
Get-CimInstance -ClassName Win32_LogonSession
Get-CimInstance -ClassName Win32_Service
```

Use -Property *   to see all properties returned

# PowerShell for Jedi

- Query Windows Event Logs on local or remote systems
- Can also parse archived logs with Get-WinEvent –Path parameter
- Get more granular results using XML filters
- Example: Find logons by a particular user account
- No SIEM needed

# PowerShell for Jedi



Security Number of events: 10,133

Filtered: Log: Security; Source: ; Event ID: 4624. Number of events: 2,115

| Keywords | Date and Time | Source | Event ID | Task Category |
|---|---|---|---|---|
| Audit Success | 8/10/2018 8:28:11 PM | Microsoft Windows security auditing. | 4624 | Logon |
| Audit Success | 8/10/2018 8:26:56 PM | Microsoft Windows security auditing. | 4624 | Logon |
| Audit Success | 8/10/2018 8:26:56 PM | Microsoft Windows security auditing. | 4624 | Logon |
| Audit Success | 8/10/2018 8:24:38 PM | Microsoft Windows security auditing. | 4624 | Logon |
| Audit Success | 8/10/2018 8:24:17 PM | Microsoft Windows security auditing. | 4624 | Logon |
| Audit Success | 8/10/2018 8:24:17 PM | Microsoft Windows security auditing. | 4624 | Logon |

Event 4624, Microsoft Windows security auditing.

○ Friendly View ● XML View

```
  </System>
- <EventData>
    <Data Name="SubjectUserSid">S-1-5-18</Data>
    <Data Name="SubjectUserName">CLIENT2$</Data>
    <Data Name="SubjectDomainName">COMPANY</Data>
    <Data Name="SubjectLogonId">0x3e7</Data>
    <Data Name="TargetUserSid">S-1-5-21-671738502-2064466678-3530451730-1103</Data>
    <Data Name="TargetUserName">tmcgrath</Data>
    <Data Name="TargetDomainName">COMPANY</Data>
    <Data Name="TargetLogonId">0x457f125</Data>
    <Data Name="LogonType">7</Data>
    <Data Name="LogonProcessName">Negotiat</Data>
    <Data Name="AuthenticationPackageName">Negotiate</Data>
    <Data Name="WorkstationName">CLIENT2</Data>
    <Data Name="LogonGuid">{00000000-0000-0000-0000-000000000000}</Data>
    <Data Name="TransmittedServices">-</Data>
    <Data Name="LmPackageName">-</Data>
    <Data Name="KeyLength">0</Data>
    <Data Name="ProcessId">0x268</Data>
    <Data Name="ProcessName">C:\Windows\System32\lsass.exe</Data>
    <Data Name="IpAddress">-</Data>
    <Data Name="IpPort">-</Data>
    <Data Name="ImpersonationLevel">%%1833</Data>
    <Data Name="RestrictedAdminMode">-</Data>
    <Data Name="TargetOutboundUserName">-</Data>
    <Data Name="TargetOutboundDomainName">-</Data>
    <Data Name="VirtualAccount">%%1843</Data>
```

# PowerShell for Jedi

- Create a query.xml file with:

```
<QueryList>
  <Query Id="0">
    <Select Path="Security">
      *[EventData[Data[@Name='TargetUserName'] and Data=‘user']]
      and
      *[System[(EventID=4624)]]</Select>
  </Query>
</QueryList>
```

# PowerShell for Jedi

- Fire at will with:

```
Get-WinEvent -FilterXml ([xml](Get-Content .\query.xml))
```

# PowerShell for Jedi

- Narrow that down to just Network logons (Type 3) with:

```
<QueryList>
  <Query Id="0">
    <Select Path="Security">
        *[EventData[Data[@Name='TargetUserName'] and
Data=‘user']]
        and
        *[EventData[Data[@Name='LogonType'] and Data='3']]
        and
        *[System[(EventID=4624)]]</Select>
  </Query>
</QueryList>
```

# Summon the Fleet

- Kansa by Dave Hull
- Freely distributed on GitHub
- Modules written in PowerShell
  - ASEP
  - Config
  - Disk
  - IOC
  - Log
  - Memory
  - Net
  - Process

# Summon the Fleet

- Works on Windows 7 clients with PowerShell 2
- Can collect data at scale
- Use it to collect baseline data
- Data simple CSV, not much space needed
- Run it at periodic intervals

# Summon the Fleet

- Also does long tail analysis of the collected data
- Stack the data, find outliers

# Strike Back

- If you detect an incident, catalog impacted systems
- Contain systems as needed
- Then use PowerShell scripts to launch coordinated eradication efforts

# Hunt Down the Dark Side

- With PowerShell, you can:
  - Maintain system baselines
  - Detect anomalies
  - Look for specific indicators of compromise
  - Collect information at scale
  - Threat hunt

# For Further Research

- Getting Started with Microsoft PowerShell
  - Jason Helmick and Jeffrey Snover
  - mva.microsoft.com/en-us/training-courses/getting-started-with-powershell-3-0-jump-start-8276
- Kansa
  - github.com/davehull/Kansa
- Get this presentation and other references for free
  - www.AppliedIncidentResponse.com