Powershell Power Hell: Hunting For Malicious Powershell With Splunk

Ryan Chapman & Lisa Tawfall

Bechtel Corporation

.conf2016

splunk>

Disclaimer

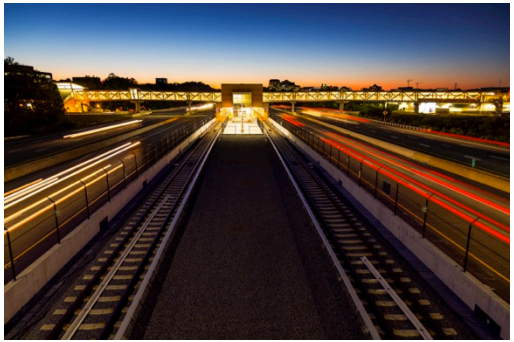
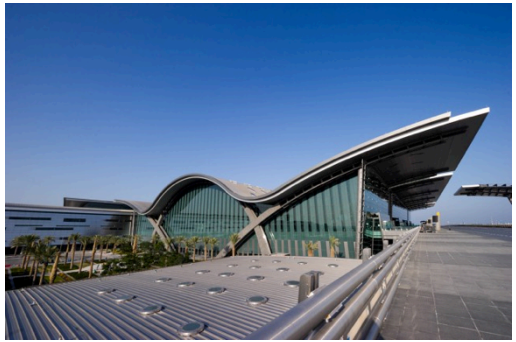
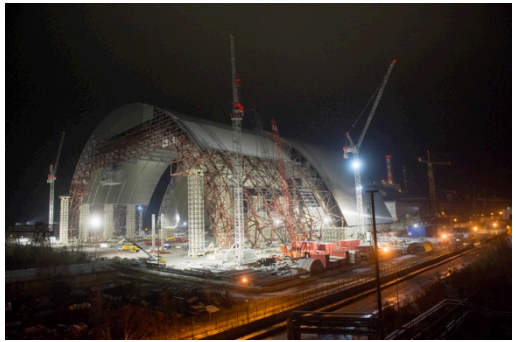
During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not, be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Agenda

- Who Are We?
- Why Focus On Powershell?
- Setting Up Powershell Logging
- Finding Malicious Powershell

Bechtel Corporation

Bechtel Corporation is the largest construction and civil engineering company in the U.S., making the company a target rich environment. Since 2011, Bechtel has set out to build a world-class Security Operations Center, which relies heavily on Splunk.

Ryan Chapman

@rj_chap

- Network Security Monitoring Analyst

- Incident Handler
- CIRT / SOC Liaison
- “Did You Check Splunk?” Guy

- *No, Really* – Did You Check Splunk?

Lisa Tawfall

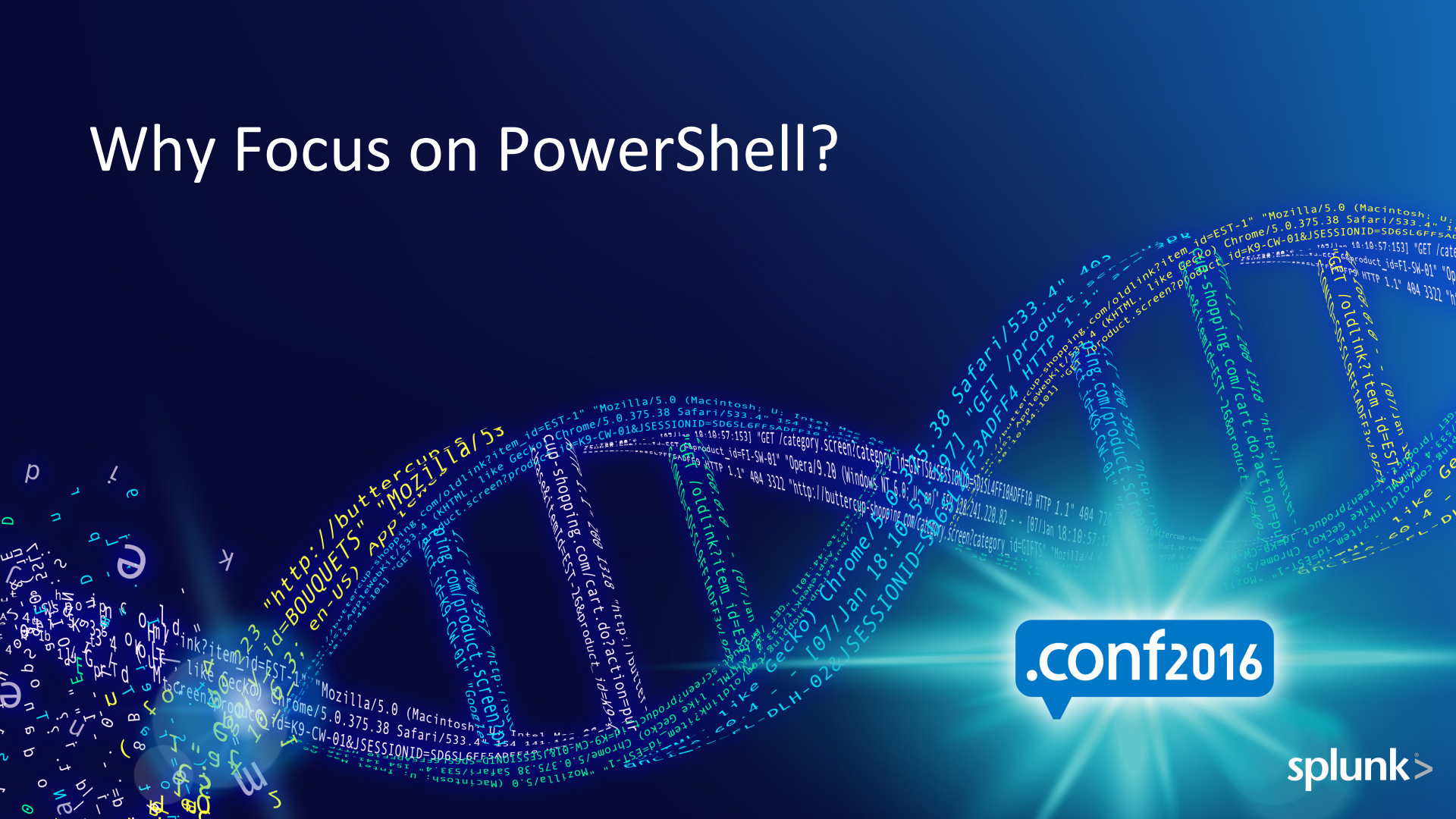
@ltawfall

- Security Unicorn (Yes, really)
- Lead for the team that manages security infrastructure at Bechtel
- Splunk Administrator
- Breaker of Splunk
- Fixer of Splunk

Obligatory Splunk Quote

"We wouldn't be able to do our jobs without Splunk."

Why Focus on PowerShell?



PowerShell Shenanigans

Attackers LOVE PowerShell

- Why Are Attackers Using PowerShell?
  - Powerful, Built-in Tool – (Nearly) Always Available
    - **PowerShell is Already in Your Environment!**
    - A Hacker's Best Toolkit = Tools on the Box!
  - Can Execute in Memory (Diskless)
  - Easy to Avoid Detection
- PowerShell is a Growing Concern

Powershell Shenanigans Cont'd.

Cmdlet Mayhem

- Remote Access Methods (more later)
- Integrates w/Core Windows Components
  - Native Win32 API
  - .NET Framework
  - WMI
- Can Access Registry, Firewall, etc.

Nature Of The Threat

Some Statistics

- Per Carbon Black Analysis of 1,100 Incidents:
  - PowerShell Used: **38%**
    - No Security Alerts Reported: 31%
  - APT-Related Attacks: 13%

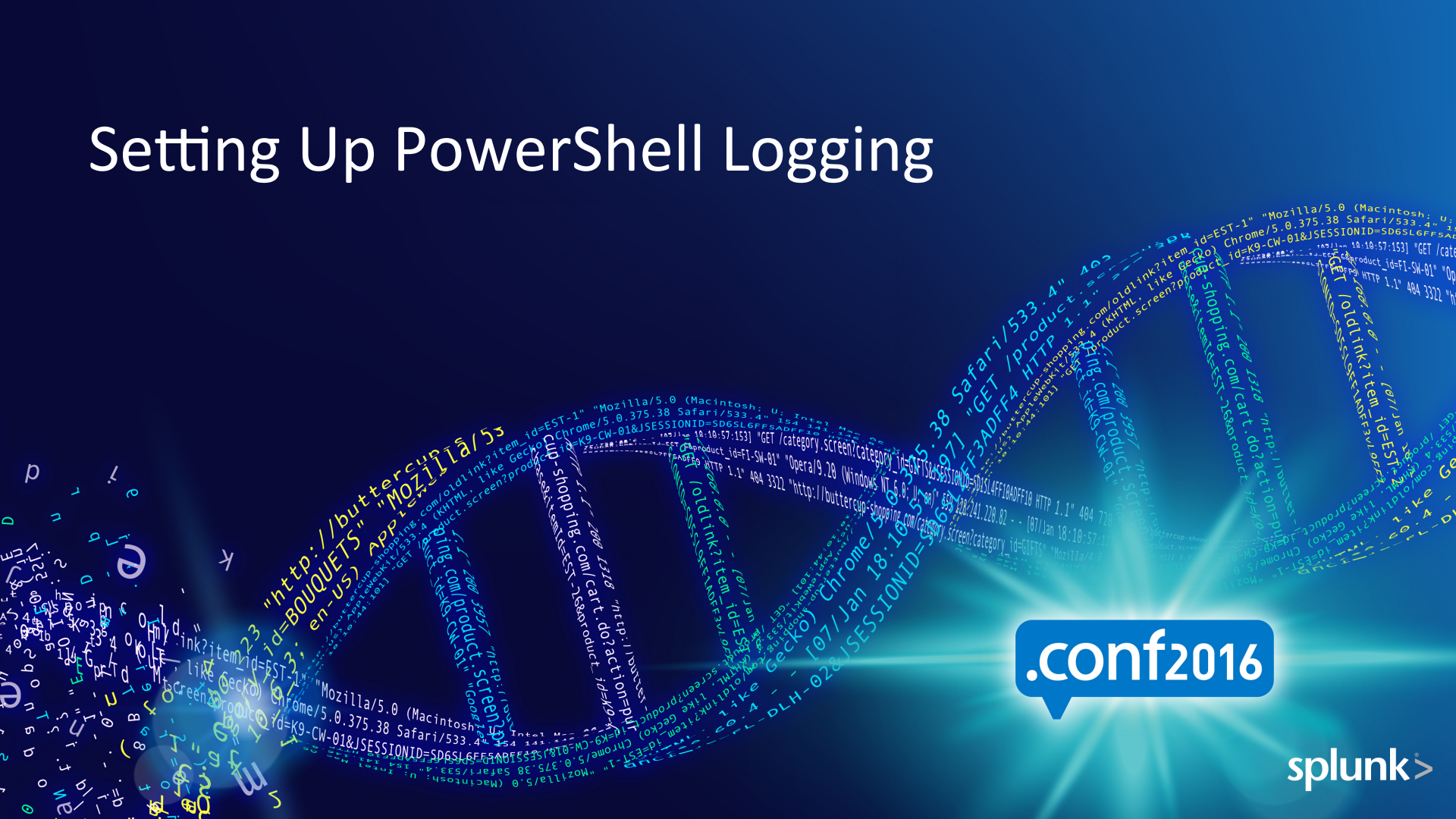
- Frameworks Leveraging PowerShell:
  - PowerSploit, Nishang, MetaSploit, etc.

0x0000FF > 0xFF0000

Not In Our House!

- In Q4 2015, We Were Pentested
  - Very Strong Red Team
- We Caught Them Immediately ☺
  - They Love PowerShell
  - *But We Love PowerShell Too*
- Pre-Existing Saved Search FTW
  - Team Member Read an OSINT Article
    - **Hunting → Saved Search**
- This was a Huge Win For Us

Setting Up PowerShell Logging



Types of PowerShell Logging

Four Ways to Win the Game

- Process Auditing – Event ID **4688**
  - Also enable Command Line Capture
- Module Logging – Event ID **4103** (**Payload**)
  - Records Pipeline Execution
- Script Block Logging – Event ID **4104** (**ScriptBlockText**)
  - Captures Script Blocks w/Deobfuscated Commands
- Transcription (**std_out**)
  - Logs All PowerShell Events to Text Files

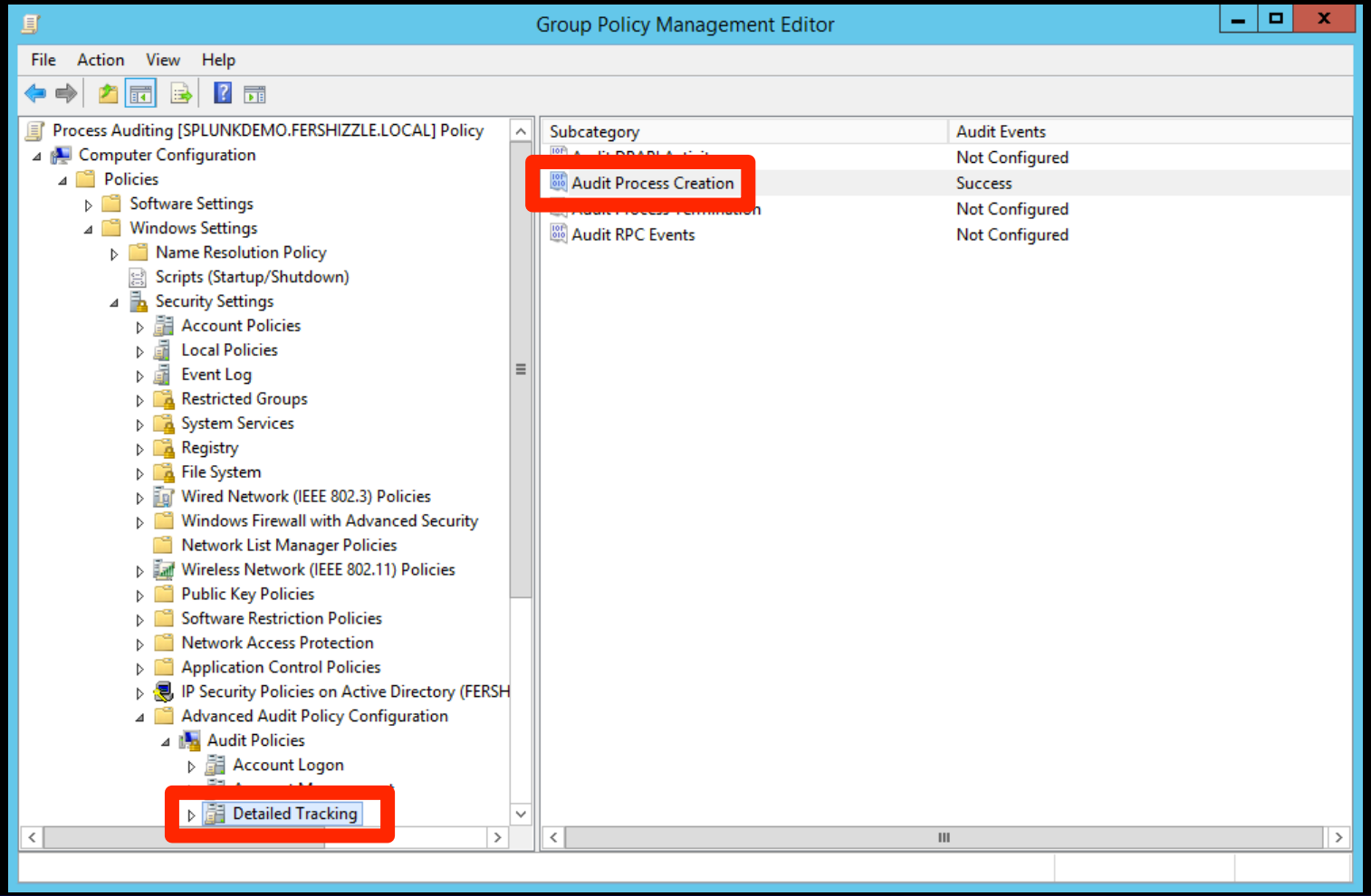
Enabling Process Creation Auditing

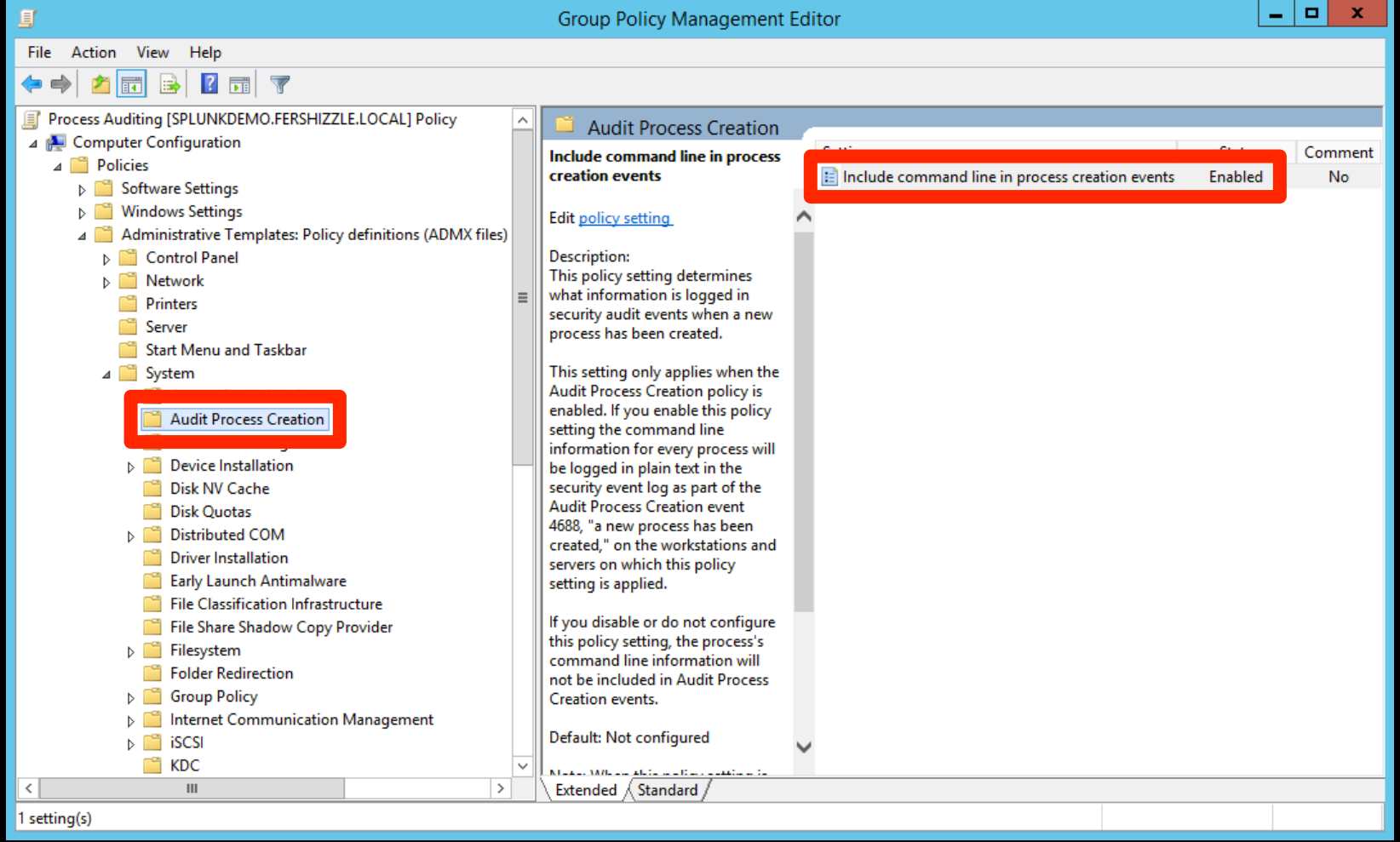
Event ID 4688 + Command Line Logging

- No WLS? *No Problem!* Bad Credit? [Actually That's a Problem]
- Computer Configuration → Policies → …
- **Enable Process Creation Auditing**

  Windows Settings → Security Settings → Advanced Audit Policy Configuration → Audit Policies → Detailed Tracking → Audit Process Creation
- **Include Command Line**

  Administrative Templates → System → Audit Process Creation → Include command line in process creation events


Group Policy Management Editor
File
Action
View
Help
Process Auditing [SPLUNKDEMO.FERSHIZZLE.LOCAL] Policy
Computer Configuration
Policies
Software Settings
Windows Settings
Name Resolution Policy
Scripts (Startup/Shutdown)
Security Settings
Account Policies
Local Policies
Event Log
Restricted Groups
System Services
Registry
File System
Wired Network (IEEE 802.3) Policies
Windows Firewall with Advanced Security
Network List Manager Policies
Wireless Network (IEEE 802.11) Policies
Public Key Policies
Software Restriction Policies
Network Access Protection
Application Control Policies
IP Security Policies on Active Directory (FERSH
Advanced Audit Policy Configuration
Audit Policies
Account Logon
Detailed Tracking
Subcategory
Audit Events
Audit Process Creation
Success
Not Configured
Audit RPC Events
Not Configured

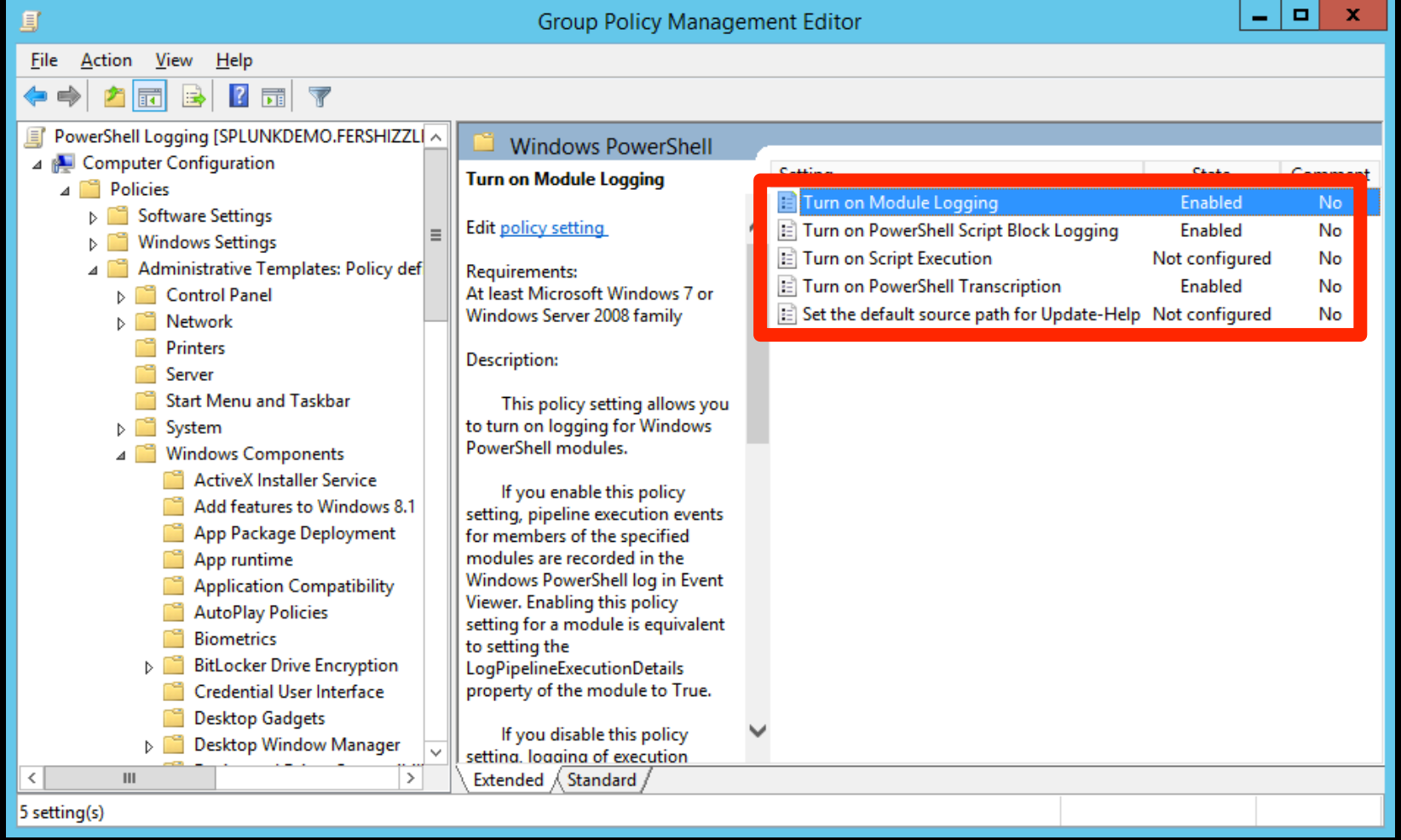


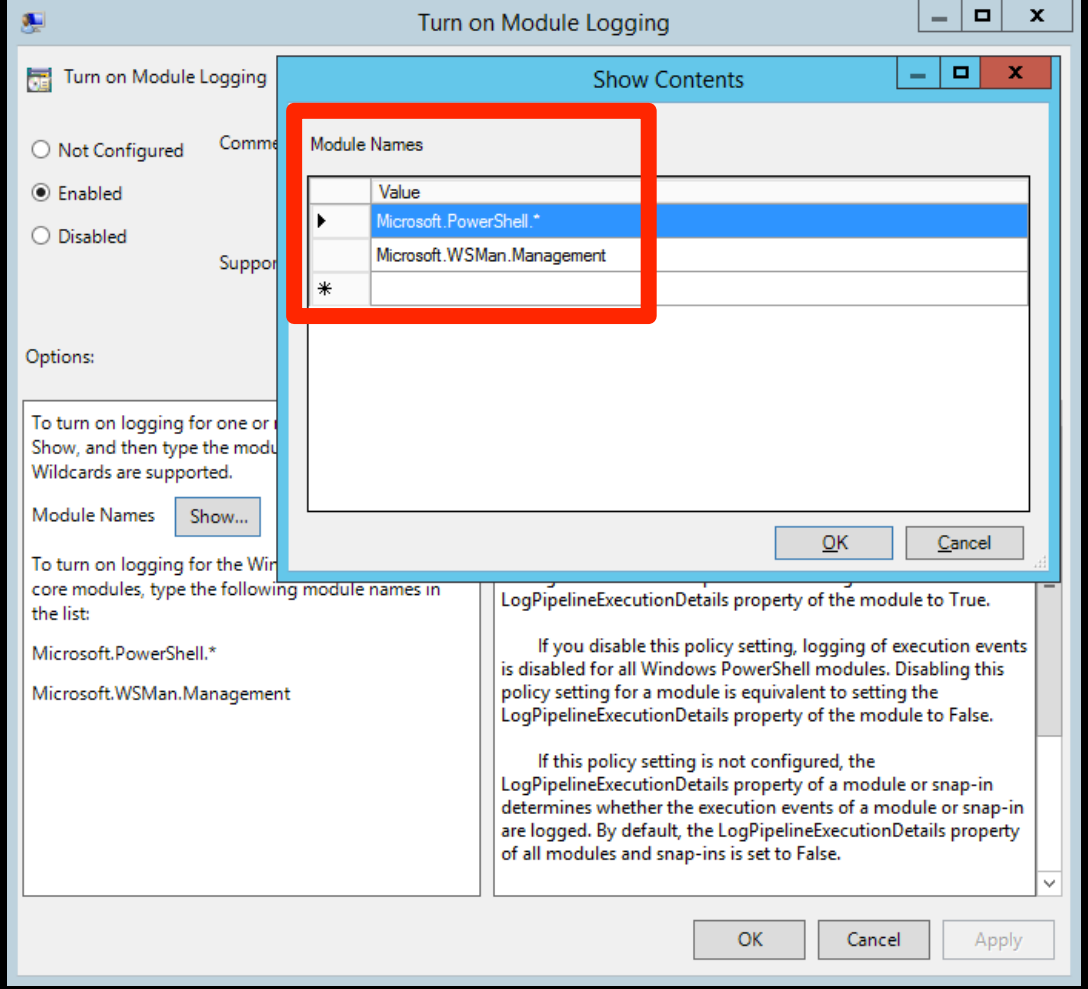
splunk> .conf2016

Enabling PowerShell Logging

Event IDs 4103 + 4104 & Transcription

- Let's Create Another GPO!
- Computer Configuration → Policies → Administrative Templates → Windows Components → Windows PowerShell
- **Enable:** Turn on Module Logging
  - Add Module Names
- **Enable:** Turn on PowerShell Script Block Logging
- **Enable:** Turn on PowerShell Transcription




Turn on Module Logging
Turn on Module Logging
Not Configured
Enabled
Disabled
Options:
To turn on logging for one or
Show, and then type the modu
Wildcards are supported.
Module Names
Show...
To turn on logging for the Wir
core modules, type the following module names in
the list:
Microsoft.PowerShell.*
Microsoft.WSMan.Management
LogPipelineExecutionDetails property of the module to True.
If you disable this policy setting, logging of execution events is disabled for all Windows PowerShell modules. Disabling this policy setting for a module is equivalent to setting the LogPipelineExecutionDetails property of the module to False.
If this policy setting is not configured, the LogPipelineExecutionDetails property of a module or snap-in determines whether the execution events of a module or snap-in are logged. By default, the LogPipelineExecutionDetails property of all modules and snap-ins is set to False.
OK
Cancel
Apply
Show Contents
Module Names
Value
Microsoft.PowerShell.*
Microsoft.WSMan.Management
OK
Cancel

Ingesting Transcription Logs

The Basics

- Custom SplunkUF Deployment App
  - `[monitor://C:\windows\system32\logfiles\powershell\...\*.txt]`
    `followTail=false`
    `sourcetype=ps_transcript`
    `index=powershell`
    `disabled = false`
    `crcSalt=<SOURCE>`
- index=powershell
- Props/Transforms

Transcription Logs

****
Windows PowerShell transcr
Start time: 20160714111156
Username: CURLY\SYSTEM
RunAs User: CURLY\SYSTEM
Machine: XXXXXXXX (Micros
Host Application: C:\Windo
bypass -file C:\Program File
Process ID: 4820
PSVersion: 5.0.10586.122
PSCompatibleVersions: 1.0, 2
BuildVersion: 10.0.10586.12
CLRVersion: 4.0.30319.4200
WSManStackVersion: 3.0
PSRemotingProtocolVersion:
SerializationVersion: 1.1.0.1
****

ell.exe -ExecutionPolicy
pipes\bin\enum_pipes.ps1

Transcription Logs

They're Ugly 2/2

```
**********************
Command start time: 20160714111156
**********************
PS>CommandInvocation(enum_pipes.ps1): "enum_pipes.ps1"
\\.\\pipe\\InitShutdown
\\.\\pipe\\lsass
\\.\\pipe\\ntsvcs
....
\\.\\pipe\\TDLN-5692-41
\\.\\pipe\\PSHost.131129935166577748.4820.DefaultAppDomain.powershell
**********************
Command start time: 20160714111156
**********************
PS>$global:?
True
**********************
Windows PowerShell transcript end
End time: 20160714111156
**********************
```

Script's Std Out

Line Breaker Challenge

```
**********************
Windows PowerShell transcript start
Start time: 20160803161741
Username: CURLY\SYSTEM
RunAs User: CURLY\SYSTEM
Machine: XXXXXX (Microsoft Windows NT 10.0.10586.0)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
-ExecutionPolicy bypass -file C:\Program
Files\SplunkUniversalForwarder\etc\apps\windows_pipes\bin\enum_pipes.ps1
Process ID: 8492
PSVersion: 5.0.10586.494
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0.10586.494
BuildVersion: 10.0.10586.494
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
**********************
Command start time: 20160803161741
**********************
PS>CommandInvocation(enum_pipes.ps1): "enum_pipes.ps1"
\\.\\pipe\\lsass
\\.\\pipe\\ntsvcs
\\.\\pipe\\scerpc
....
\\.\\pipe\\a5b1bc44-53bc-464a-8eba-e799025fa282
\\.\\pipe\\browser
**********************
Command start time: 20160803161741
**********************
PS>$global:?
True
**********************
Windows PowerShell transcript end
End time: 20160803161741
**********************
```

MUST_BREAK_AFTER = End time:.+$*+$
MUST_NOT_BREAK_AFTER = *+$
NO_BINARY_CHECK = true
TIME_FORMAT = %Y%m%d%H%M%S
TIME_PREFIX = Start time:

The Gory Details

Regex Is Your Friend

```
EXTRACT-bun = (?m)(?:Username: )(?<domain>[\w-]+)\\(?<bun>\w+)
EXTRACT-runasbun = (?m)(?:RunAs User: )(?<runas_domain>[\w-]+)\\(?<runas_bun>\w+)
EXTRACT-username = (?m)(?:Username: )(?<username>[\w-\\]+)
EXTRACT-machine = (?m)(?:Machine: )(?<computer>[^\s].+?)(?:\s\()(?<os_ver>.+?)\)\r?$
EXTRACT-app = (?m)(?:Host Application: )(?<app>[^\s].+?)\r?$
EXTRACT-version = (?m)(?:PSVersion: )(?<PSVersion>[\d\.]+)
EXTRACT-compatible = (?m)^(?:PSCompatibleVersions: )(?<PSCompatible>[0-9\.,\s]+)
EXTRACT-build = (?m)(?:BuildVersion: )(?<build>[\d\.]+)
EXTRACT-CLR = (?m)(?:CLRVersion: )(?<CLR>[0-9\.]+)
EXTRACT-WSMan = (?m)(?:WSManStackVersion: )(?<WSMan>[\d\.]+)
EXTRACT-protocol = (?m)(?:PSRemotingProtocolVersion: )(?<protocol>[\d\.]+)
EXTRACT-std_out = (?sm)(?:\*+.+SerializationVersion: [\d+\.]+)(?<std_out>.+)(?:\*+.+Windows PowerShell transcript end.+)
EVAL-end_time = strptime(end_time,"%Y%m%d%H%M%S")
```

Field Parsing

```
**********************
Windows PowerShell transcript start
Start time: 20160803161741
Username: CURL
RunAs User: CURLY\SYSTEM
Machine: XXXXX
Machine: XXXXXX (Microsoft Windows NT 10.0.10586.0)
Host Applicat
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Process ID: 8492
-ExecutionPolicy bypass -file C:\Program
Process ID: 8492
PSVersion: 5.0.10586.494
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0.10586.494
BuildVersion: 10.0.10586.494
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
```

EXTRACT-machine = (?m)(?:Machine:)(?<computer>[^\s].+?)(?:\s\()(?<os_ver>.+?)\)**\r?$**

EXTRACT-app = (?m)(?:Host Application:)(?<app>[^\s].+?)**\r?$**

Regex Magic

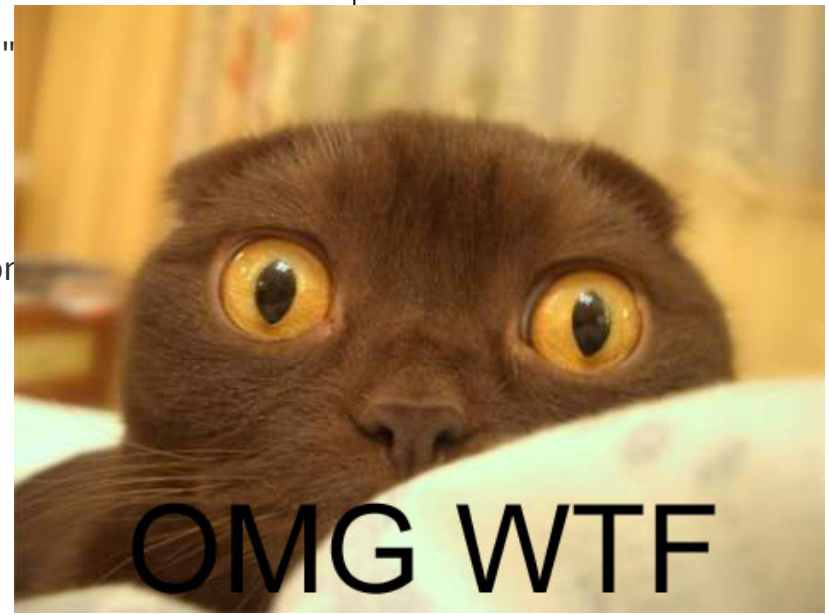
Remember this Giant Std. Out Bit?

EXTRACT-std_out = **(?sm)**(?:*+.+SerializationVersion: [\d+\.]+)(?<std_out>.+)(?:*+.+Windows PowerShell transcript end.+)

```
**********
Command start time: 20160714111156
**********
PS>CommandInvocation(enum_pipes.ps1): "enum_pipes.ps1"
\\.\\pipe\\InitShutdown
\\.\\pipe\\lsass
\\.\\pipe\\ntsvcs
....
\\.\\pipe\\TDLN-5692-41
\\.\\pipe\\PSHost.131129953166377748.4820.DefaultAppDor
**********
Command start time: 20160714111156
**********
PS>$global:?
True
**********
Windows PowerShell transcript end
End time: 20160714111156
**********
```

(?s) for "single line mode" makes the dot match all characters, including line breaks.

(?m) for "multi-line mode" makes the caret and dollar match at the start and end of each line in the subject string.

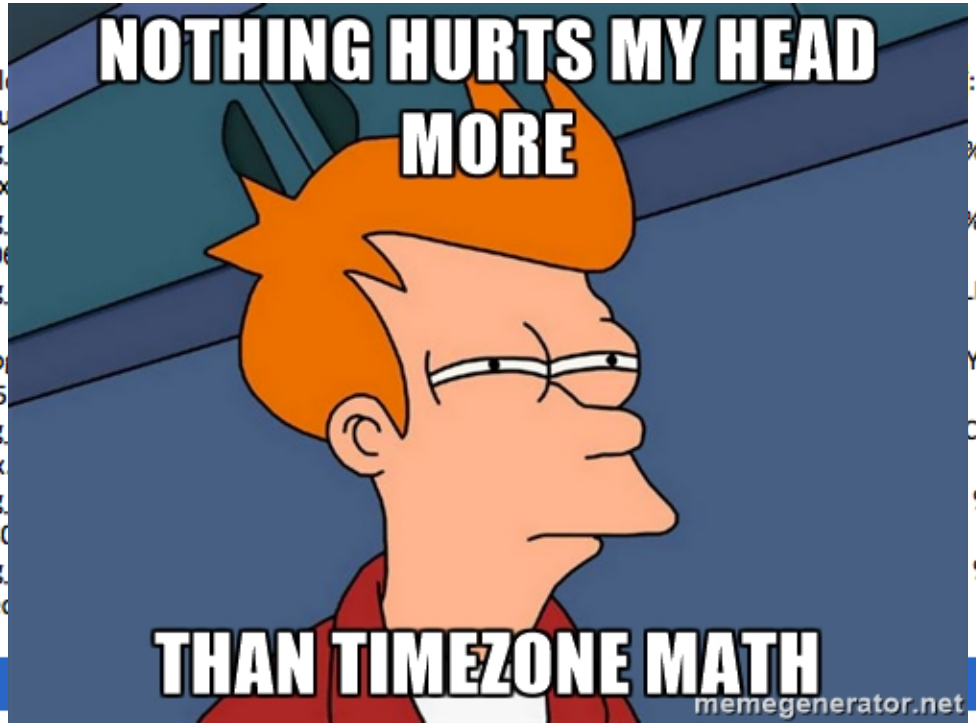


What Timezone is This Log?

Things that Drive Your Splunk Admin Ballistic

Aug 4 15:59:01 xx.xxx.121.250 Kiwi_Sysl : %ADJ-3-RESOLVE_REQ: Adj resolve request: Failed to resolve xx.xxx.121.18(u
Aug 4 15:59:01 xx.xxx.0.160 Kiwi_Syslog %FMANFP-6-IPACCESSLOGP: F0: fman_fp_image: list 130 denied udp xx.x
Aug 4 15:59:59 xx.xxx.0.189 Kiwi_Syslog %SEC-6-IPACCESSLOGP: list 130 denied tcp xx.240.144.65(s2.xxxxx.360.cn)(4080
Aug 4 16:00:03 xx.xxx.7.240 Kiwi_Syslog LPOWER-5-IEEE_DISCONNECT: Interface Fa1/0/22: PD removed
Aug 4 16:00:08 xx.xxx.103.49 Kiwi_Syslo YS-4-CONFIG_RESOLVE_FAILURE: System config parse from (tftp://255.255
Aug 4 16:00:19 xx.xxx.1.191 Kiwi_Syslog CESSLOGP: list 130 denied tcp xx.119.170.85(unresolved)(61659) -> xxx
Aug 4 16:00:29 xx.xxx.0.109 Kiwi_Syslog %SEC-6-IPACCESSLOGP: list 140 denied tcp xxx.240.144.65(s2.xxxxx.360.cn)(4080
Aug 4 16:00:32 xx.xxx.0.109 Kiwi_Syslog %SEC-6-IPACCESSLOGDP: list 140 denied icmp 208.123.150.193(unresolvec

```
[root@xxx-syslog1 08]# date
Thu Aug  4 20:47:01 UTC 2016
```



Finding Malicious PowerShell

Base64 Encoded Commands

This Caught Our Red Team ☺

```
index=wls* EventID=4688 BaseFileName="powershell.exe"
CommandLine="*-en*" NOT ([REDACTED])
| sort 0 _time
| table _time, host, SubjectDomainName,
  SubjectUserName, BaseFileName, CommandLine,
  CompanyName, CreatorProcessName, NewProcessName,
  FileDescription, FileVersion, MD5
```

IMPORTANT !

Enhanced Alert w/B64 Decryption

Requires Decrypt (App 2655)

```
index=wls* EventID=4688 (BaseFileName="powershell*.exe"
OR BaseFileName="cmd.exe")
(CommandLine="*-en*" OR CommandLine="*base64string*")
| rex field=CommandLine "-((?i)enc|encodedcommand|
encode|en)\s\'?(?<base64_command>\w{20,1000}\=?\=?)\'?"
| decrypt field=base64_command atob()
emit('base64_decoded_command')
| search base64_decoded_command=*
```

Saved Search Results

[Most Columns Removed]

_time	Computer	CommandLine
11/9/15 15:35	[REDACTED]	**powershell.exe -nop -w hidden -encodedcommand** JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG…AKQA7AA==
11/9/15 15:49	[REDACTED]	**powershell -nop -exec bypass -EncodedCommand** 'SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0A…AKQA7AA=='

Saved Search Results

Decoded Base64 – MOAR Base64 + Zipped!

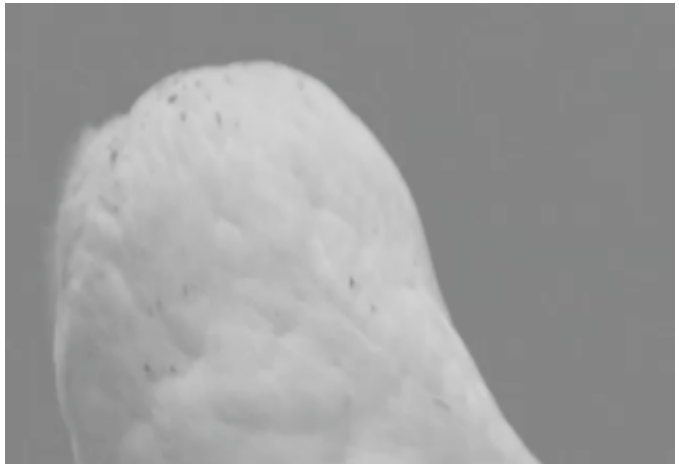
```
$s=New-Object IO.MemoryStream(,
[Convert]::FromBase64String("H4sIAAAAAAAAAL1We2/aSBD/
Gz7FqopkW+UZuJREi…DAAA"));
```

```
IEX (New-Object IO.StreamReader(New-Object
IO.Compression.GzipStream $s,
[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
```

Saved Search Results

Decoding Continued

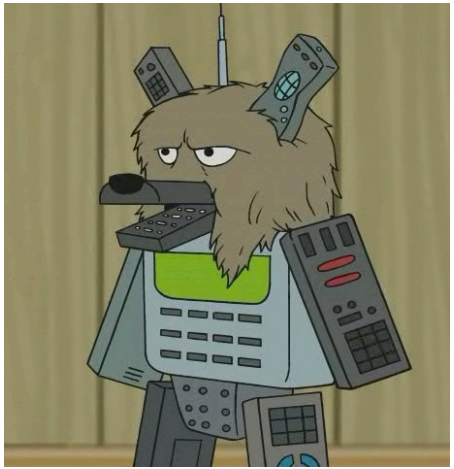
- Secondary PowerShell Script
  - $var_code == **Shellcode**
- Shellcode Creates Named Pipe
  - Inter-Process Communication
- Errrrrr…. No
  - **ALL HANDS ON DECK**



Remote PowerShell

Common Remote Methods

```
Get-Service winrm
Enable-PSRemoting
    New-PSSession
 Enter-PSSession
Invoke-Command -computername
```



General use of: **-computer**

NOTE: -computer can specify 127.0.0.1)

PowerShell: WSMan

Get-WSManInstance

Displays management information for a resource instance specified by a Resource URI.

```
PS C:\> get-help *wsman*

Name                                Category  Module                    Synopsis
----                                --------  ------                    --------
Disable-WSManCredSSP                Cmdlet    Microsoft.WSMan.Manage... Disables Credential Security Support Pr
Enable-WSManCredSSP                 Cmdlet    Microsoft.WSMan.Manage... Enables Credential Security Support Pro
Get-WSManCredSSP                    Cmdlet    Microsoft.WSMan.Manage... Gets the Credential Security Support Pr
Set-WSManQuickConfig                Cmdlet    Microsoft.WSMan.Manage... Configures the local computer for remot
Test-WSMan                          Cmdlet    Microsoft.WSMan.Manage... Tests whether the WinRM service is runn
Invoke-WSManAction                  Cmdlet    Microsoft.WSMan.Manage... Invokes an action on the object that is
Connect-WSMan                       Cmdlet    Microsoft.WSMan.Manage... Connects to the WinRM service on a remo
Disconnect-WSMan                    Cmdlet    Microsoft.WSMan.Manage... Disconnects the client from the WinRM s
Get-WSManInstance                   Cmdlet    Microsoft.WSMan.Manage... Displays management information for a r
Set-WSManInstance                   Cmdlet    Microsoft.WSMan.Manage... Modifies the management information tha
Remove-WSManInstance                Cmdlet    Microsoft.WSMan.Manage... Deletes a management resource instance.
New-WSManInstance                   Cmdlet    Microsoft.WSMan.Manage... Creates a new instance of a management
New-WSManSessionOption              Cmdlet    Microsoft.WSMan.Manage... Creates a WS-Management session option
Disable-PSWSManCombinedTrace        Function  PSDiagnostics             ...
Disable-WSManTrace                  Function  PSDiagnostics             ...
Enable-PSWSManCombinedTrace         Function  PSDiagnostics             ...
Enable-WSManTrace                   Function  PSDiagnostics             ...
```

Remote PowerShell Search

Module Logging Method (4103)

```
index=wls* EventID=4103
ProviderName="Microsoft-Windows-PowerShell"

(Payload="*winrm*"                    OR
 Payload="*psremoting*"               OR
 Payload="*pssession*"                OR
 Payload="*invoke-command*"           OR
 Payload="*wsman*"
 [OR Payload="*-computer*"])
```

Remote PowerShell Search

Script Block Logging Method (4104)

```
index=wls* EventID=4104
ProviderName="Microsoft-Windows-PowerShell"

(ScriptBlockText="*winrm*"                OR
 ScriptBlockText="*psremoting*"           OR
 ScriptBlockText="*pssession*"            OR
 ScriptBlockText="*invoke-command*"       OR
 ScriptBlockText="*wsman*"
  [OR ScriptBlockText="*-computer*"])
```

PowerShell Cornucopia

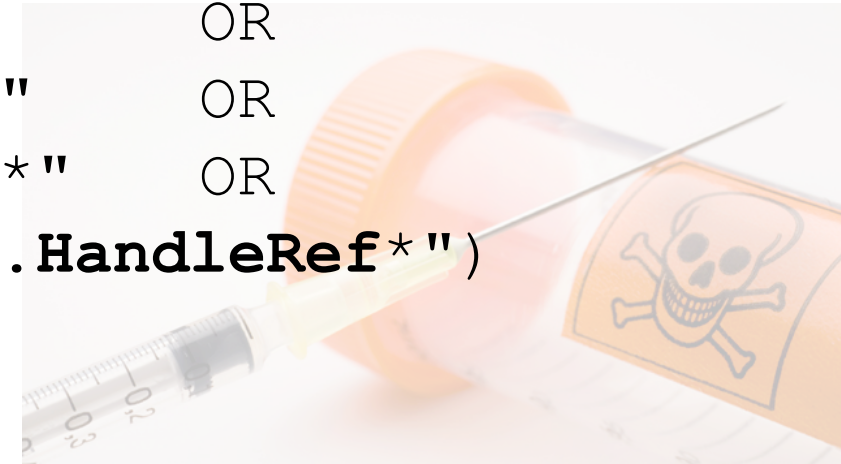
These Are *Interesting*

```
index=wls* EventID=4688
(BaseFileName=powershell*.exe    OR
 BaseFileName=cmd.exe)
(CommandLine="*get-process*"     OR
 CommandLine="*get-service*"     OR
 CommandLine="*get-filehash*"    OR
 CommandLine="*get-hotfix*"      OR
 CommandLine="*cd hk*"           OR
 CommandLine="*get-itemproperty hk*" OR
 CommandLine="*netfirewallrule*")
```

Common Injection Methods

INJECTION SON!

```
([field]="*getassemblies*"          OR
 [field]="*assemblyname*"           OR
 [field]="*system.dll*"             OR
 [field]="*GetProcAddress*"         OR
 [field]="*GetModuleHandle*"        OR
 [field]="*InteropServices.HandleRef*")
```



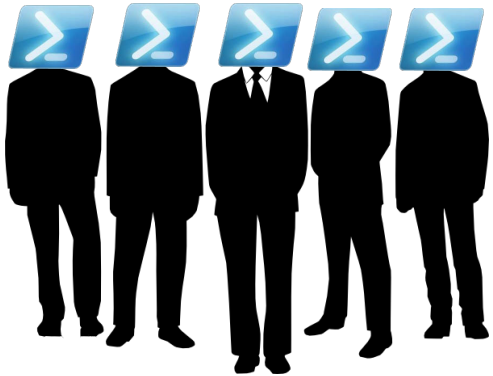
PowerShellMafia's PowerSploit

Dirty Dirty Tricks

- Open Source PowerShell Attack Framework
  - Becoming More and More Common
- We Can Enumerate PowerSploit Modules
  - And Look For Them
    - *And yell/cry/smile if we find any*

Q: Is Anyone Running PowerSploit?

(BETTER NOT BE!)



PowerShellMafia / **PowerSploit**

Watch 318 | Star 1,540 | Fork 625

Code | Issues 8 | Pull requests 0 | Pulse | Graphs

PowerSploit - A PowerShell Post-Exploitation Framework

321 commits | **2** branches | **2** releases | **12** contributors

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

mattifestation Merge pull request #132 from pyllyukko/sch_hourly ... Latest commit `262a260` on May 29

AntivirusBypass	Set all module versions to 3.0	7 months ago
CodeExecution	Set all module versions to 3.0	7 months ago
Exfiltration	Don't search for SYSTEM token by using hard coded English name for SY…	7 months ago
Mayhem	Set all module versions to 3.0	7 months ago
Persistence	Added ScheduledTaskHourly to New-UserPersistenceOption	3 months ago
Privesc	Set all module versions to 3.0	7 months ago
Recon	Set all module versions to 3.0	7 months ago
ScriptModification	Set all module versions to 3.0	7 months ago

CodeExecution

Execute code on a target machine.

`Invoke-DllInjection`

Injects a Dll into the process ID of your choosing.

`Invoke-ReflectivePEInjection`

Reflectively loads a Windows PE file (DLL/EXE) in to the powershell process, or reflectively injects a DLL in to a remote process.

`Invoke-Shellcode`

Injects shellcode into the process ID of your choosing or within PowerShell locally.

`Invoke-WmiCommand`

Executes a PowerShell ScriptBlock on a target computer and returns its formatted output using WMI as a C2 channel.

Example: CodeExecution Modules

Script Transcription Method (index=powershell)

```
 index=powershell std_out="*"

(std_out="*DllInjection*"                  OR
 std_out="*ReflectivePEInjection*"  OR
 std_out="*Shellcode*"                     OR
 std_out="*WmiCommand*")
```

Recap of 5 Takeaways

‘Member These Things

- PowerShell is **Already in Your Environment**
- PowerShell Logging **Must Be Enabled**
- Event Codes of Interest: **4103, 4104, & 4688**
- Create, Test, and Perfect **Baselines** to Avoid FPs
- **Ongoing Research** Leads to New Search Ideas

@rj_chap **@ltawfall**

THANK YOU

QUESTIONS?

.conf2016

Ryan Chapman & Lisa Tawfall

Bechtel Corporation

splunk>

References

Check ‘Em Out!

- Carbon Black. (2016, April). ‘PowerShell’ deep dive: A united threat research report. Retrieved from https://www.carbonblack.com/wp-content/uploads/2016/04/Cb-Powershell-Deep-Dive-A-United-Threat-Research-Report-1.pdf
- DFIR Blog. (2015, September). Dissecting powershell attacks Retrieved from https://dfir-blog.com/2015/09/27/dissecting-powershell-attacks/
- Splunk. (2016). Decrypt. Retrieved from https://splunkbase.splunk.com/app/2655/

Bonus Reference

Highly Recommended

- Malware Archaeology. (2015). Cheat sheets. Retrieved from http://www.malwarearchaeology.com/cheat-sheets/

- Malware Archaeology's Cheat Sheets Include:
  - Windows PowerShell Logging
  - Windows Logging
  - Windows Splunk Logging
  - And more!