Securing PowerShell with Free Techniques

Mr. Ashley McGlone
Twitter: @GoateePFE
Technical Account Manager, Tanium


PS:\>


?!

Matt Graeber
@mattifestation

Following

The security transparency is so good in PowerShell that I'm investing in other languages/frameworks w/ less/no security insight.

7:37 PM - 8 Sep 2017

Matt Graeber
@mattifestation

Following

Burning PS w/ fire in your org doesn't bother me. PS is never the reason ppl get owned and when it is used for evil, I'll have rich insight.

7:46 PM - 8 Sep 2017

A Comparison of Shell and Scripting Language Security – Lee Holmes

Engine	Event Logging	Transcription	Dynamic Evaluation Logging	Encrypted Logging	Application Whitelisting	Antimalware Integration	Local Sandboxing	Remote Sandboxing	Untrusted Input Tracking
Bash	No**	No*	No	No	Yes	No	No*	Yes	No
CMD / BAT	No	No	No	No	Yes	No	No	No	No
Jscript	No	No	No	No	Yes	Yes	No	No	No
LUA	No	No	No	No	No	No	No*	Yes	Yes
Perl	No	No	No	No	No	No	No*	Yes	Yes
PHP	No	No	No	No	No	No	No*	Yes	Yes
PowerShell	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No**
Python	No	No	No	No	No	No	No	No	No**
Ruby	No	No	No	No	No	No	No**	No**	Yes
sh	No**	No*	No	No	No	No	No*	Yes	No
T-SQL	Yes	Yes	Yes	No	No	No	No**	No**	No
VBScript	No	No	No	No	Yes	Yes	No	No	No
zsh	No**	No*	No	No	No	No	No*	Yes	No

* Feature exists, but cannot enforce by policy

** Experiments exist

https://blogs.msdn.microsoft.com/powershell/2017/04/10/a-comparison-of-shell-and-scripting-language-security/

```
PS> Start-Demo
```

PowerShell.exe –ExecutionPolicy Bypass –File c:\temp\invoke-evil.ps1

https://blogs.msdn.microsoft.com/powershell/2008/09/30/powershells-security-guiding-principles/

PowerShell Forensics Feature Matrix

WMF = Windows Management Framework (PowerShell upgrade)

<table>
<tr><th></th><th>Win/Mac/Linux</th><th>Windows 10</th><th>Windows 8.1</th><th>Windows 8</th><th>Windows 7</th></tr>
<tr><th></th><th>PS Core</th><th>WS 2016</th><th>WS 2012 R2</th><th>WS 2012</th><th>WS 2008 R2</th></tr>
<tr><td>Module Logging</td><td rowspan="3">Built-In</td><td rowspan="5">Built-In</td><td>Built-In</td><td>Built-In</td><td rowspan="3">WMF 5.1</td></tr>
<tr><td>Script Block Logging</td><td rowspan="2">WMF 5.1</td><td rowspan="2">WMF 5.1</td></tr>
<tr><td>Transcription</td></tr>
<tr><td>PSReadline Command History</td><td>Available</td><td>Available</td><td>Available</td></tr>
<tr><td>Anti-Malware Scan Interface</td><td>n/a</td><td>n/a</td><td>n/a</td><td>n/a</td></tr>
</table>

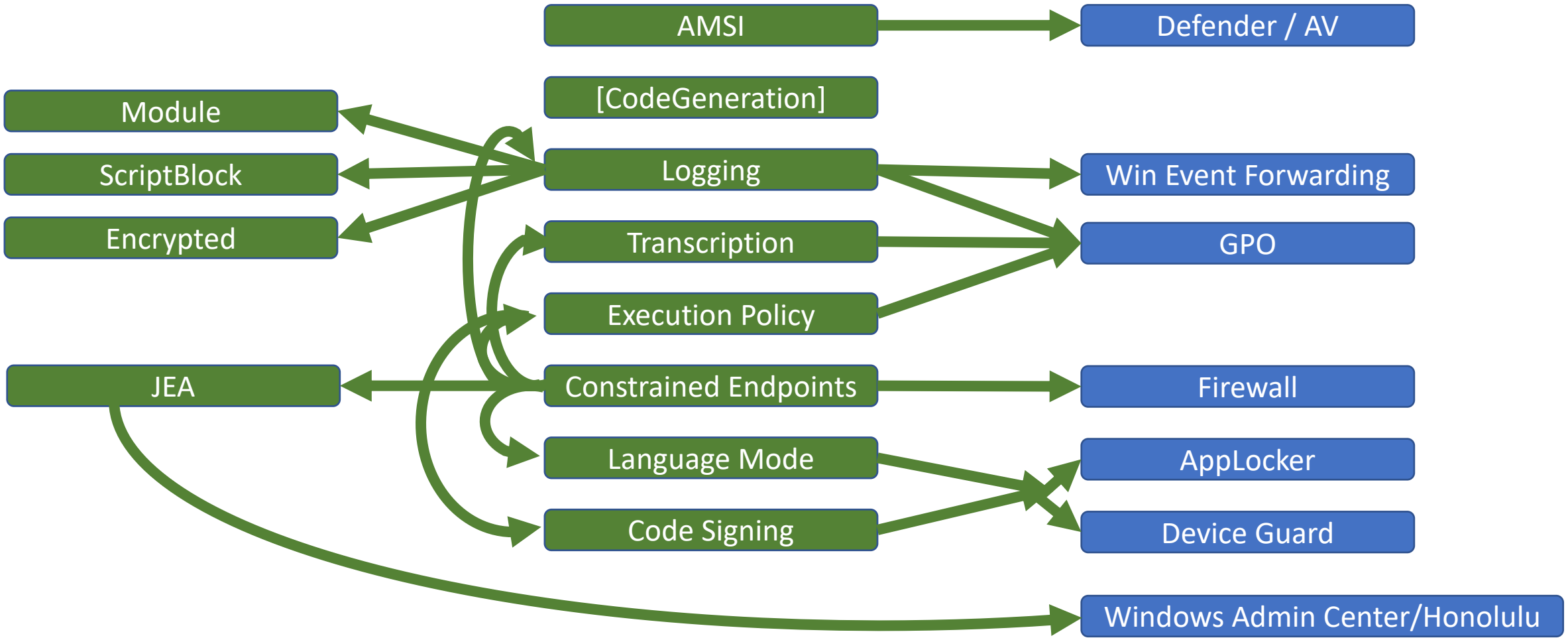
PowerShell Forensics Events & Log Files

	Event Log	Event IDs
Module Logging	Windows PowerShell	400, 800, 403
Module Logging	Microsoft-Windows-PowerShell/Operational	4103
Script Block Logging	Microsoft-Windows-PowerShell/Operational	4105, 4104, 4106
Anti-Malware Scan Interface	Microsoft-Windows-Windows Defender/Operational	1116, 1117
Transcription ◆	HKLM:\Software\Policies\Microsoft\Windows\PowerShell\Transcription\OutputDirectory	
PSReadline Command History ◆◆	C:\Users*\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline*.txt	

◆ *Path configured by policy*

◆◆ *Default path*

Security Feature Windows Integration Points



PowerShell Hygiene

Level 0: Upgrade all machines to Windows PowerShell 5.1 (WMF 5.1)

Level 1: Implement and monitor: logging & transcription

Level 2: Enable and secure remoting (default endpoint, firewall)

Level 3: Block evil code: AppLocker / DeviceGuard / AMSI

Key Takeaways

PowerShell is a POST-exploitation tool, not a vulnerability.

PowerShell is the MOST security-featured scripting language.

Properly configured, PowerShell leaves glowing fingerprints.

Don’t be afraid of PowerShell remoting. Lock it down with JEA.

Upgrade to PowerShell 5.1 and remove the PS 2.0 Windows Feature.

THIS IS ALL FREE!

Resources

Follow on Twitter:

@Lee_Holmes

@mattifestation

@danielhbohannon

@jepayneMSFT

Find a link to this deck and a PowerShell Security hands on lab in my Twitter feed: @GoateePFE

http://aka.ms/pssec