



Search 

Active Directory Security

Active Directory & Enterprise Security, Methods to Secure Active Directory, Attack Methods & Effective Defenses, PowerShell, Tech Notes, & Geek Trivia...

Home

About

AD Resources

Attack Defense & Detection

Contact

Mimikatz

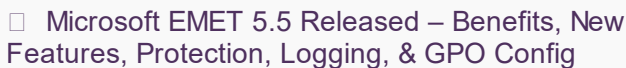
Presentations

Schema Versions

Security Resources

SPNs

Top Posts



Detecting Offensive PowerShell Attack Tools 

FEB
11
2016

PowerShell Version 5 Security Enhancements

By [Sean Metcalf](#) in [Microsoft Security](#), [PowerShell](#), [Technical Reference](#)

PowerShell version 5 is RTM (As of 12/18/2015). Prior to this there was a “production preview” available since August which means it was supported, but not final. With the final release of PowerShell v5 now available, I highly recommend you [download PowerShell v5](#) and start testing to prepare for production deployment.

While the PowerShell v5 download was pulled previously due to an [issue with PSModule Path](#), the [Windows Management Framework \(WMF\) 5.0 RTM is available for download once again.](#)

PowerShell provides extensive management capability for administrators, though this capability can also be used by attackers to exploit and persist in an enterprise. [There are ways to detect offensive PowerShell attack tools](#) and PowerShell v5 extends this capability further.

Microsoft provides the following PowerShell v5 benefits and updates on the [download site](#):

Windows Management Framework (WMF) 5.0 brings functionality that has been updated from WMF 4.0. WMF 5.0 is available for

installation only on Windows Server 2012 R2, Windows Server 2012, Windows 2008 R2, Windows 8.1, and Windows 7 SP1. Some of the new and updated features in this release include:

- Develop with classes in Windows PowerShell
- Just Enough Administration (JEA)
- Extract and parse structured object out of string content
- More control in Windows PowerShell Remote Debugging
- PowerShell Information Stream
- New and updated cmdlets based on community feedback
- Generate Windows PowerShell cmdlets based on an OData endpoint with ODataUtils
- Manage .ZIP archives through new cmdlets
- Interact with symbolic links using improved Item cmdlets
- DSC authoring improvements in Windows PowerShell ISE
- 32-bit support for the configuration keyword in DSC
- Audit Windows PowerShell usage by transcription and logging
- Configure DSC's Local Configuration Manager with the meta-configuration attribute
- Configure piece by piece with partial configurations in DSC
- Manage with cross-computer dependencies in DSC
- More control over configurations in DSC
- Find more detail about configuration status in DSC
- Support for -? during DSC configuration compilation
- Support for DSC RunAsCredential
- Rich information for DSC LCM State
- Side-by-Side installation of DSC Resources and PowerShell Modules
- PSDesiredStateConfiguration Module version updated to 1.1
- Report configuration status from DSC to a central location
- Discover and install software with PackageManagement

- Discover PowerShell Modules, PowerShell Scripts and DSC resources with PowerShellGet
- Network Switch management with Windows PowerShell
- Software Inventory Logging (SIL)

There are several compelling PowerShell v5 security features that make it a must deploy (IMHO). I [presented on several of these at security conferences in 2015](#).

These security features include:

- Script block logging
- System-wide transcripts
- Constrained PowerShell
- Antimalware Integration aka AMSI (Windows 10)

Script block logging

Script block logging provides the ability to log de-obfuscated PowerShell code to the event log. Most attack tools are obfuscated, often using Base64 encoding, before execution to make it more difficult to detect or identify what code actually ran. Script block logging logs the actual code delivered to the PowerShell engine before execution which is possible since the script code needs to be de-obfuscated before execution.

Since many PowerShell attacks obfuscate the attack code, it is difficult to identify what the script code does. Script block logging de-obfuscates the code and logs the code that is executed. Since this code is logged, it can be alerted on when seen by a central logging system.

One key challenge with identifying offensive PowerShell code is that most of the time it is obfuscated (Base64, Base64+XOR, etc). This makes real-time analysis nearly impossible since there is no keyword to trigger alerts on.

Deep Script Block Logging records the content of the script blocks it processes as well as the generated script code at execution time.

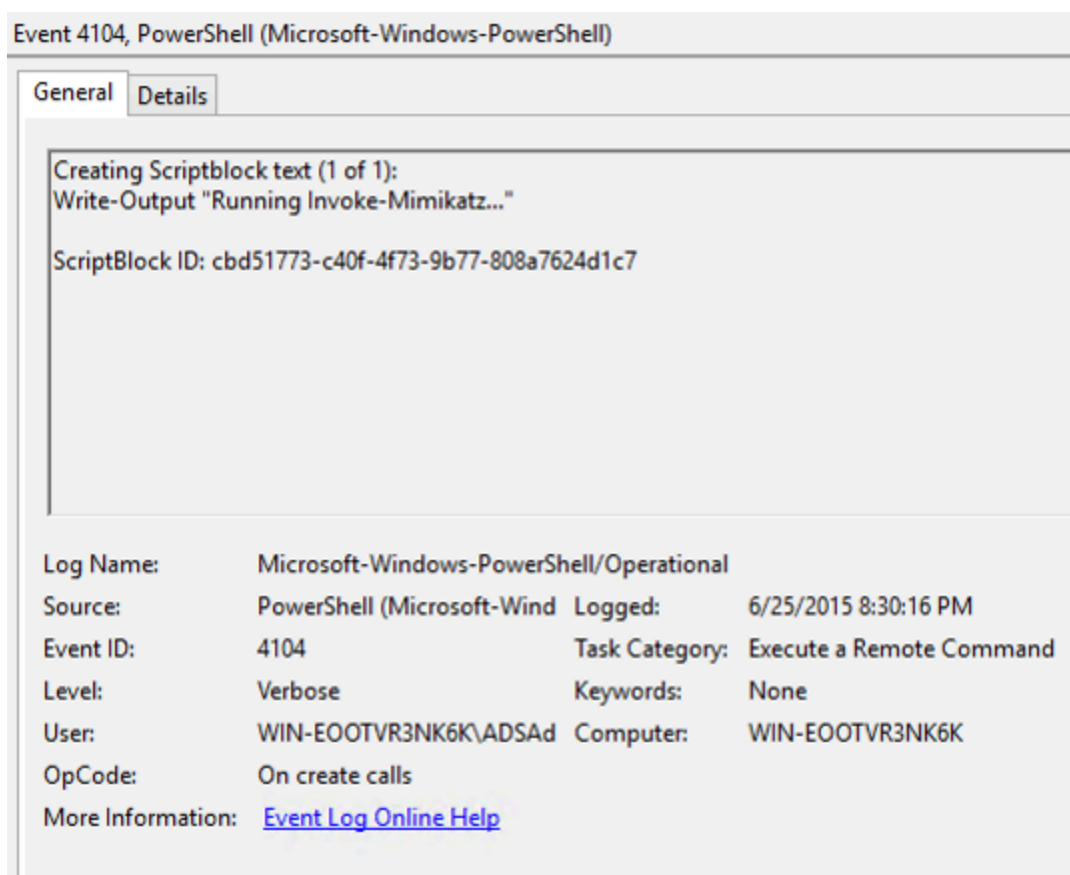
Microsoft-provided example of obfuscated command code

```
## Malware
function SuperDecrypt
{
param($script)
$bytes = [Convert]::FromBase64String($script)
## XOR “encryption”
$xorKey = 0x42
for($counter = 0; $counter -lt $bytes.Length; $counter++)
{
$bytes[$counter] = $bytes[$counter] -bxor $xorKey
}
[System.Text.Encoding]::Unicode.GetString($bytes)
}
$decrypted = SuperDecrypt
“FUlwQitCNklnQm9CCkltQjFCNkJiQmVCEkl1QixCJkJlQg==”
Invoke-Expression $decrypted
```

The original script block passed to PowerShell for processing is logged (what you see above) as well as the actual command that PowerShell executed.

Note that Script Block Logging is enabled by default.

```
PS C:\Users\ADSAdmin> powershell -encodedcommand VwByAgkAdAB1AC0ATwB1AHQAcAB1AHQAIAA1AFIAdQBuaG4AaQ8uAGcAIABJAG4AdgBvAGsAZQAtAE0AaQ8tAGk/
Running Invoke-Mimikatz...
```



System-wide transcripts

System-wide transcripting can be enabled via Group Policy and provides an “over the shoulder” transcript file of every PowerShell command and code block executed on a system by every user on that system. This transcript can be directed to a write-only share on the network for later analysis and SIEM tool ingesting.

PowerShell has the ability to save text written to the console (screen) in a “transcript” file which requires the user (or script) to run “start-transcript \$FileName”. This provides a simple script log file. The drawback to this method is that only one transcript could be active at a time, the PowerShell ISE editor didn’t support transcripts, and that Start-Transcript would have to be added to every user’s PowerShell profile in order for a record of run commands to be saved.

System-wide transcripts provides a simple method to write all PowerShell commands (including those run inside scripts or downloaded from another location) into a computer-specific transcript file that is

stored in a network share. This enables rapid analysis of near-real time PowerShell usage as well as identification of “known-bad” activity.

The system-wide transcript can be enabled via Group Policy and includes the following information in the header:

- Start time
- User Name
- RunAs User
- Machine (Operating System)
- Host Application
- Process ID

Parameters:

- IncludeInvocationHeader – includes start time headers for every command run.
- OutputDirectory – enables writing transcripts to a central location, such as a network share.

Microsoft-provided example PowerShell script to configure ACL on the central transcripts share:

```
md c:\Transcripts
```

```
## Kill all inherited permissions
```

```
$acl = Get-Acl c:\Transcripts
```

```
$acl.SetAccessRuleProtection($true, $false)
```

```
## Grant Administrators full control
```

```
$administrators = [System.Security.Principal.NTAccount]
```

```
“Administrators”
```

```
$permission =
```

```
$administrators, “FullControl”, “ObjectInherit, ContainerInherit”, “Non
```

e", "Allow"

\$accessRule = New-Object

System.Security.AccessControl.FileSystemAccessRule

\$permission

\$acl.AddAccessRule(\$accessRule)

Grant everyone else Write and ReadAttributes. This prevents users from listing

transcripts from other machines on the domain.

\$everyone = [System.Security.Principal.NTAccount] "Everyone"

\$permission =

\$everyone, "Write, ReadAttributes", "ObjectInherit, ContainerInherit", "None", "Allow"

\$accessRule = New-Object

System.Security.AccessControl.FileSystemAccessRule

\$permission

\$acl.AddAccessRule(\$accessRule)

Deny "Creator Owner" everything. This prevents users from

viewing the content of previously written files.

\$creatorOwner = [System.Security.Principal.NTAccount] "Creator Owner"

\$permission =

\$creatorOwner, "FullControl", "ObjectInherit, ContainerInherit", "InheritOnly", "Deny"

\$accessRule = New-Object

```
System.Security.AccessControl.FileSystemAccessRule  
$permission  
$acl.AddAccessRule($accessRule)
```

Set the ACL

```
$acl | Set-Acl c:\Transcripts\
```

Create the SMB Share, granting Everyone the right to read and write files. Specific

actions will actually be enforced by the ACL on the file folder.

```
New-SmbShare -Name Transcripts -Path c:\Transcripts -  
ChangeAccess Everyone
```

Turn on System-wide Transcription via Group Policy:

Windows Components -> Administrative Templates -> Windows PowerShell -> Turn on PowerShell Transcription

This Group Policy setting configures the registry key:

HKLM:\Software\Policies\Microsoft\Windows\PowerShell\Transcription


```
PS C:\> get-content C:\Users\ADSAdmin\Documents\PowerShell_transcript.ADSWK10.6CuHE1fY.20150730171748.txt
*****
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
Running     Audiosrv            Windows Audio
Stopped     AxInstSV            ActiveX Installer (AxInstSV)
Stopped     BDESVC              BitLocker Drive Encryption Service
Running     BFE                 Base Filtering Engine
Running     BITS                Background Intelligent Transfer Ser...
Running     BrokerInfrastru...  Background Tasks Infrastructure Ser...
Stopped     Browser             Computer Browser
Stopped     BthHFSrv            Bluetooth Handsfree Service
Stopped     bthserv             Bluetooth Support Service
Stopped     CDPSvc              Connected Device Platform Service
Running     CertPropSvc         Certificate Propagation
Running     ClipSVC             Client License Service (ClipSVC)
Stopped     COMSysApp           COM+ System Application
Running     CoreUIRegistrar     CoreMessaging
Running     CryptSvc            Cryptographic Services
Stopped     CscService          Offline Files
Running     DcomLaunch          DCOM Server Process Launcher
Stopped     DcpSvc              DcpSvc
Stopped     defragsvc           Optimize drives
Stopped     DeviceAssociati...  Device Association Service
Stopped     DeviceInstall       Device Install Service
```

Constrained PowerShell

PowerShell supports several “[language modes](#)“. One of the more interesting language mode is “Constrained Language mode” which locks down PowerShell to basic functionality.

PowerShell v5 also supports automatic lock-down when AppLocker is deployed in “Allow” mode. Applocker Allow mode is true whitelisting and can prevent any unauthorized binary from being executed. PowerShell v5 detects when Applocker Allow mode is in effect and sets the PowerShell language to Constrained Mode, severely limiting the attack surface on the system. With Applocker in Allow mode and PowerShell running in Constrained Mode, it is not possible for an attacker to change the PowerShell

language mode to full in order to run attack tools. When AppLocker is configured in “Allow Mode”, PowerShell reduces its functionality to “Constrained Mode” for interactive input and user-authored scripts. Constrained PowerShell only allows core PowerShell functionality and prevents execution of the extended language features often used by offensive PowerShell tools (direct .NET scripting, invocation of Win32 APIs via the Add-Type cmdlet, and interaction with COM objects).

Note that scripts allowed by AppLocker policy such as enterprise signed code or in a trusted directory are executed in full PowerShell mode and not the Constrained PowerShell environment.

Antimalware Integration (Windows 10)

The new Windows 10 [Antimalware Scan Interface \(AMSI\)](#) enables all of the scripting engines (PowerShell, VBScript, and JScript) to request analysis of dynamic content, from a script file, typed commands at the command line, and even code downloaded and executed in memory. This enables scanning of PowerShell code before it is executed on the computer and is a potential game-changer when it comes to defending systems from offensive PowerShell code. When code is delivered to the PowerShell “engine” ([System.Management.Automation.dll](#)), it is sent to the [AMSI](#) for anti-malware checks. The anti-malware solution installed on the system needs to support AMSI in order for the code to be scanned. Windows Defender supports AMSI on Windows 10 out of the box. After scanning, if the AMSI returns a status OK, the code is executed. If it returns a negative, then the code is not executed.

This means that PowerShell attack code can be prevented from executing on Windows 10 computers, as long as the anti-virus/anti-malware solution supports the [AMSI](#).

The Antimalware Scan Interface (AMSI) is a generic interface standard that allows applications and services to integrate with any antimalware product present on a machine. It provides enhanced malware protection for users and their data, applications, and workloads.

AMSI is antimalware vendor agnostic, designed to allow for the most common malware scanning and protection techniques provided by today's antimalware products that can be integrated into applications. It supports a calling structure allowing for file and memory or stream scanning, content source URL/IP reputation checks, and other techniques.

AMSI also supports the notion of a session so that antimalware vendors can correlate different scan requests. For instance, the different fragments of a malicious payload can be associated to reach a more informed decision, which would be much harder to reach just by looking at those fragments in isolation.

References:

- [PowerShell {Hearts} the Blue Team](#)
- Microsoft Whitepaper: [Scripting Security and Protection Advances in Windows 10](#)
- [What's New in PowerShell v5](#)
- [Detecting Offensive PowerShell Attack Tools](#)

(Visited 18,793 times, 3 visits today)

□ [AMSI](#), [PowerShell Antimalware Integration](#), [PowerShellSecurity](#), [PowerShellv5](#), [script block logging](#), [System-wide transcript](#), [Windows10](#)



Sean Metcalf

I improve security for enterprises around the world working for
TrimarcSecurity.com

Read the About page (top left) for information about me. :)
https://adsecurity.org/?page_id=8



RECENT POSTS

[BSides Dublin – The Current State of Microsoft Identity Security: Common Security Issues and Misconfigurations – Sean Metcalf](#)

[DEFCON 2017: Transcript – Hacking the Cloud](#)

[Detecting the Elusive: Active Directory Threat Hunting](#)

[Detecting Kerberoasting Activity](#)

[Detecting Password Spraying with Security Event Auditing](#)

TRIMARC ACTIVE DIRECTORY SECURITY SERVICES

Have concerns about your Active Directory environment? Trimarc helps enterprises improve their security posture.

[Find out how...](#) [TrimarcSecurity.com](#)

POPULAR POSTS

[PowerShell Encoding & Decoding \(Base64\)](#)

[Attack Methods for Gaining Domain Admin Rights in...](#)

[Kerberos & KRBTGT: Active Directory's...](#)

[Finding Passwords in SYSVOL & Exploiting Group...](#)

[Securing Domain Controllers to Improve Active...](#)

[Securing Windows Workstations: Developing a Secure Baseline](#)

[Detecting Kerberoasting Activity](#)

[Mimikatz DCSync Usage, Exploitation, and Detection](#)

[Scanning for Active Directory Privileges &...](#)

[Microsoft LAPS Security & Active Directory LAPS...](#)

CATEGORIES

[ActiveDirectorySecurity](#)

[Apple Security](#)

[Cloud Security](#)

[Continuing Education](#)

[Entertainment](#)

[Exploit](#)

[Hacking](#)

[Hardware Security](#)

[Hypervisor Security](#)

[Linux/Unix Security](#)

Malware

Microsoft Security

Mitigation

Network/System Security

PowerShell

RealWorld

Security

Security Conference Presentation/Video

Security Recommendation

Technical Article

Technical Reading

Technical Reference

TheCloud

Vulnerability

TAGS

ActiveDirectory Active Directory Active Directory Security ActiveDirectorySecurity

ADReading AD Security ADSecurity Azure AzureAD DCSync **DomainController** GoldenTicket GroupPolicy

HyperV Invoke-Mimikatz KB3011780 KDC **Kerberos** KerberosHacking KRBTGT LAPS LSASS **MCM**

MicrosoftEMET MicrosoftWindows **mimikatz** MS14068 PassTheHash **PowerShell**

PowerShellCode Pow erShellHacking Pow erShellv5 PowerSploit Presentation Security SilverTicket

SneakyADPersistence SPN TGS TGT Window s7 Windows10 WindowsServer2008R2 WindowsServer2012

WindowsServer2012R2



RECENT POSTS

[BSides Dublin – The Current State of Microsoft Identity Security: Common Security Issues and Misconfigurations – Sean Metcalf](#)

[DEFCON 2017: Transcript – Hacking the Cloud](#)

[Detecting the Elusive: Active Directory Threat Hunting](#)

[Detecting Kerberoasting Activity](#)

[Detecting Password Spraying with Security Event Auditing](#)

RECENT COMMENTS

Derek on [Attacking Read-Only Domain Controllers \(RODCs\) to Own Active Directory](#)

Sean Metcalf on [Securing Microsoft Active Directory Federation Server \(ADFS\)](#)

Brad on [Securing Microsoft Active Directory Federation Server \(ADFS\)](#)

Joonas on [Gathering AD Data with the Active Directory PowerShell Module](#)

Sean Metcalf on [Gathering AD Data with the Active Directory PowerShell Module](#)

ARCHIVES

[June 2024](#)

[May 2024](#)

[May 2020](#)

[January 2020](#)

August 2019

March 2019

February 2019

October 2018

August 2018

May 2018

January 2018

November 2017

August 2017

June 2017

May 2017

February 2017

January 2017

November 2016

October 2016

September 2016

August 2016

July 2016

June 2016

April 2016

March 2016

February 2016

January 2016

December 2015

November 2015

October 2015

September 2015

August 2015

July 2015

June 2015

May 2015

April 2015

March 2015

February 2015

January 2015

December 2014

November 2014

October 2014

September 2014

August 2014

July 2014

June 2014

May 2014

April 2014

March 2014

February 2014

July 2013

November 2012

March 2012

February 2012

CATEGORIES

ActiveDirectorySecurity

Apple Security

Cloud Security

Continuing Education

Entertainment

Exploit

Hacking

Hardware Security

Hypervisor Security

Linux/Unix Security

Malware

Microsoft Security

Mitigation

Network/System Security

PowerShell

RealWorld

Security

Security Conference Presentation/Video

Security Recommendation

[Technical Article](#)

[Technical Reading](#)

[Technical Reference](#)

[TheCloud](#)

[Vulnerability](#)

META

[Log in](#)

[Entries feed](#)

[Comments feed](#)

[WordPress.org](#)

COPYRIGHT


Content Disclaimer: This blog and its contents are provided "AS IS" with no warranties, and they confer no rights. Script samples are provided for informational purposes only and no guarantee is provided as to functionality or suitability. The views shared on this blog reflect those of the authors and do not represent the views of any companies mentioned. Content Ownership: All content posted here is intellectual work and under the current law, the poster

owns the copyright of the article.

Terms of Use Copyright © 2011

- 2020.

Content Disclaimer: This blog and its contents are provided "AS IS" with no warranties, and they confer no rights. Script samples are provided for informational purposes only and no guarantee is provided as to functionality or suitability. The views shared on this blog reflect those of the authors and do not represent the views of any companies mentioned.

Made with  by Graphene Themes.