GitHub - danielbohannon/Invoke-Obfuscation: PowerShell Obfuscator

By cobbr

Archived: 2026-04-06 01:15:57 UTC

Introduction

Invoke-Obfuscation is a PowerShell v2.0+ compatible PowerShell command and script obfuscator.

Background

In the Fall of 2015 I decided to begin researching the flexibility of PowerShell's language and began cataloguing the various ways to accomplish a handful of common techniques that most attackers use on a regular basis.

Initially focusing on encoded command and remote download cradle syntaxes, I discovered that various escape characters that did not hinder the execution of the command persisted in the command line arguments, both in the running process as well as what is logged in Security EID 4688 and Sysmon EID 1 event logs. This led me to systematically explore ways of obfuscating each kind of "token" found in any PowerShell command or script.

I then explored more obscure ways to perform string-level obfuscation, various encoding/encrypting techniques (like ASCII/hex/octal/binary and even SecureString), and finally PowerShell launch techniques to abstract the command line arguments from powershell.exe and to push it back to the parent and even grandparent process.

Purpose

Attackers and commodity malware have started using extremely basic obfuscation techniques to hide the majority of the command from the command line arguments of powershell.exe. I developed this tool to aid the Blue Team in simulating obfuscated commands based on what I currently know to be syntactically possible in PowerShell 2.0-5.0 so that they can test their detection capabilities of these techniques.

The tool's sole purpose is to break any assumptions that we as defenders may have concerning how PowerShell commands can appear on the command line. My hope is that it will encourage the Blue Team to shift to looking for Indicators of Obfuscation on the command line in addition to updating PowerShell logging to include Module, ScriptBlock and Transcription logging as these sources simplify most aspects of the obfuscation techniques generated by this tool.

Usage

While all of the layers of obfuscation have been built out into separate scripts, most users will find the `Invoke-Obfuscation` function to be the easiest way to explorer and visualize the obfuscation techniques that this framework currently supports.

Installation

The source code for Invoke-Obfuscation is hosted at Github, and you may download, fork and review it from this repository (https://github.com/danielbohannon/Invoke-Obfuscation). Please report issues or feature requests through Github's bug tracker associated with this project.

To install:

```
 Import-Module ./Invoke-Obfuscation.psd1
Invoke-Obfuscation
```

License

Invoke-Obfuscation is released under the Apache 2.0 license.

Release Notes

v1.0 - 2016-09-25 DerbyCon 6.0 (Louisville, Kentucky USA): PUBLIC Release of Invoke-Obfuscation.

v1.1 - 2016-10-09 SANS DFIR Summit (Prague, Czech Republic): Added -f format operator re-ordering functionality to all applicable TOKEN obfuscation functions. Also added additional syntax options for setting variable values.

v1.2 - 2016-10-20 CODE BLUE (Tokyo, Japan): Added Type TOKEN obfuscation (direct type casting with string obfuscation options for type name).

v1.3 - 2016-10-22 Hacktivity (Budapest, Hungary): Added two new LAUNCHERs: CLIP+ and CLIP++. Also added additional (and simpler) array char conversion syntax for all ENCODING functions that does not require For-EachObject/%.

v1.4 - 2016-10-28 BruCON (Ghent, Belgium): Added new BXOR ENCODING function. Also enhanced randomized case for all components of all ENCODING functions as well as for PowerShell execution flags for all LAUNCHERs. Finally, added -EP shorthand option for -ExecutionPolicy to all LAUNCHERs as well as the optional integer representation of the -WindowStyle PowerShell execution flag: Normal (0), Hidden (1), Minimized (2), Maximized (3).

v1.5 - 2016-11-04 Blue Hat (Redmond, Washington USA): Added WMIC LAUNCHER with some randomization of WMIC command line arguments.

v1.6 - 2017-01-24 Blue Hat IL (Tel Aviv, Israel):

- Added CLI functionality: E.g., Invoke-Obfuscation -ScriptBlock {Write-Host 'CLI FTW!'} -Command 'Token\All\1, Encoding\1,Launcher\Stdin++\234,Clip' -Quiet -NoExit
- Added UNDO functionality to remove one layer of obfuscation at a time.
- Removed Whitespace obfuscation from Token\All\1 to speed up large script obfuscation.
- Added Process Argument Tree output for all launchers to aid defenders.
- Added base menu auto-detect functionality to avoid needing to use BACK or HOME: E.g., if you ran TOKEN then ALL then 1, then just type LAUNCHER and you will get to the LAUNCHER menu without needing to type HOME or BACK to get back to the home menu.
- Added multi-command syntax utilized by CLI and interactive mode: E.g., Token\All\1,String\3,Encoding\5,Launcher\Ps\234,Clip
- Added regex capability to all menu and obfuscation commands: E.g., Token**,String[13],Encoding(1|6),Launcher.*[+]{2}\234,Clip
- Added OUT FILEPATH single command functionality.
- Added decoding if powershell -enc syntax is entered as a SCRIPTBLOCK value.
- Added alias ForEach to ForEach-Object/% randomized syntax options in all ENCODING functions.
- Added -Key -Ke -K KEY substring syntax options to Out-SecureStringCommand.ps1.
- Added more thorough case randomization to all \Home\String obfuscation functions.
- Added -ST/-STA (Single-Threaded Apartment) flags to CLIP+ and CLIP++ launcher functions since they are required if running on PowerShell 2.0.

- Added Get-Item/GI/Item syntax everywhere where Get-ChildItem is used to get variable values.
- Added Set-Item variable instantiation syntax to TYPE obfuscation function.
- Added additional Invoke-Expression/IEX syntax using PowerShell automatic variables and environment variable value concatenations in Out-ObfuscatedStringCommand.ps1's Out-EncapsulatedInvokeExpression function and copied to all launchers, STRING and ENCODING functions to add numerous command-line syntaxes for IEX.
- Added two new JOIN syntaxes for String\Reverse and all ENCODING obfuscation options:

1. Added [String]::Join('',$string) JOIN syntax
2. Added OFS-variable JOIN syntax (Output Field Separator automatic variable)

- Added two more SecureString syntaxes to Encoding\5:

1. PtrToStringAnsi / SecureStringToGlobalAllocAnsi
2. PtrToStringBSTR / SecureStringToBSTR

- Added six GetMember alternate syntaxes for several SecureString members:

1. PtrToStringAuto, ([Runtime.InteropServices.Marshal].GetMembers()[3].Name).Invoke
2. PtrToStringAuto, ([Runtime.InteropServices.Marshal].GetMembers()[5].Name).Invoke
3. PtrToStringUni , ([Runtime.InteropServices.Marshal].GetMembers()[2].Name).Invoke
4. PtrToStringUni , ([Runtime.InteropServices.Marshal].GetMembers()[4].Name).Invoke
5. PtrToStringAnsi, ([Runtime.InteropServices.Marshal].GetMembers()[0].Name).Invoke
6. PtrToStringAnsi, ([Runtime.InteropServices.Marshal].GetMembers()[1].Name).Invoke

- Updated Out-ObfuscatedTokenCommand.ps1 so that VARIABLE obfuscation won't encapsulate variables in ${} if they are already encapsulated (so ${${var}} won't happen as this causes errors).
- Replaced Invoke-Obfuscation.psm1 with Invoke-Obfuscation.psd1 (thanks @Carlos_Perez).
- Fixed several TOKEN-level obfuscation bugs reported by @cobbr_io and @IISResetMe.

v1.7 - 2017-03-03 nullcon (Goa, India):

- Added 3 new LAUNCHERs: RUNDLL, RUNDLL++ and MSHTA++
- Added additional ExecutionContext wildcard variable strings

v1.8 - 2017-07-27 Black Hat (Las Vegas, Nevada USA):

- Added 2 new ENCODING options: Special Characters and Whitespace

v1.8.1 - 2017-12-19:

- Added COMPRESS function for easier conversion of multi-line scripts to a one-liner command while drastically reducing the command length for cmd.exe command line length limitation purposes.

v1.8.2 - 2018-01-04:

- Added AST obfuscation functions, which obfuscates by manipulating the structure of the AbstractSyntaxTree without using many special characters.

Source: https://github.com/danielbohannon/Invoke-Obfuscation