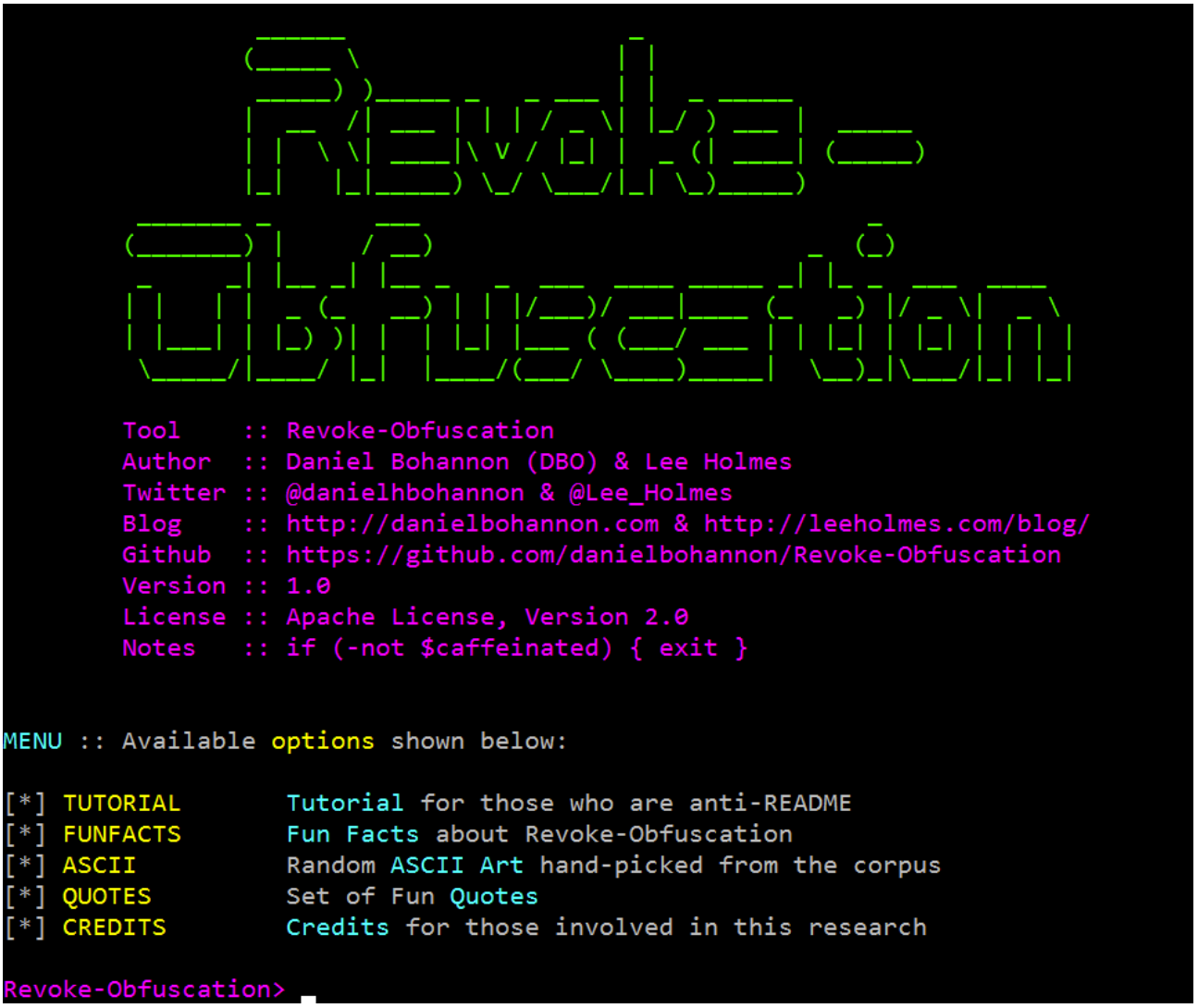
GitHub - danielbohannon/Revoke-Obfuscation: PowerShell Obfuscation Detection Framework

By LeeHolmes

Archived: 2026-04-05 20:28:32 UTC



Introduction

Revoke-Obfuscation is a PowerShell v3.0+ compatible PowerShell obfuscation detection framework.

Authors

Daniel Bohannon (@danielhbohannon) Lee Holmes (@Lee_Homes)

Research

Blog Post: https://www.fireeye.com/blog/threat-research/2017/07/revoke-obfuscation-powershell.html

White Paper: https://www.fireeye.com/content/dam/fireeye-www/blog/pdfs/revoke-obfuscation-report.pdf

Background

In the Fall of 2016 and Spring of 2017, Daniel Bohannon (@danielhbohannon) released Invoke-Obfuscation and Invoke-CradleCrafter, two open-source PowerShell obfuscation frameworks. The goal of this research and these frameworks was to highlight the limitations of a purely signature-based approach to detecting attackers' usage of

PowerShell. The core message to defenders has been to focus on detecting **Indicators of Obfuscation** in addition to known suspicious syntax.

However, the extreme levels of randomization in Invoke-Obfuscation and Invoke-CradleCrafter paired with the token-layer obfuscation options that are not deobfuscated in PowerShell's script block logging have led defenders to look for a new, scalable means of generically detecting both known and unknown obfuscation techniques.

A few weeks after the release of Invoke-Obfuscation, Lee Holmes (@Lee_Homes) authored a blog posted entitled More Detecting Obfuscated PowerShell in which he highlighted statistical analysis techniques to detect anomalous features found in heavily obfuscated scripts, specifically those produced by Invoke-Obfuscation.

Since this exchange, Daniel and Lee became good friends and shared many common interests/obsessions -- namely, a love of fine coffee and the pursuit of creating new ways to thoroughly detect obfuscated PowerShell.

The amount of time both Blue Teamers spent pouring over research and POC code would equate to several thousand cups of Chemex-brewed coffee assuming the proper 4-minute target brew time (assuming at least one other coffee enthusiast picked up on this "pour over" pun).

Revoke-Obfuscation is the final hand-crafted product of these efforts.

Purpose

Revoke-Obfuscation is an open-source PowerShell v3.0+ framework for detecting obfuscated PowerShell commands and scripts at scale. It relies on PowerShell's AST (Abstract Syntax Tree) to rapidly extract thousands of features from any input PowerShell script and compare this feature vector against one of several pre-defined weighted feature vectors computed through an automated learning process conducted against a corpus of 408K+ PowerShell scripts. This full corpus can be downloaded from (https://aka.ms/PowerShellCorpus). You can find the details behind the data science aspects of this work in the 'DataScience' subdirectory of the repository.

Since Revoke-Obfuscation relies on feature extraction and comparison instead of pure IOCs or RegEx matching, it is more robust in its ability to identify unknown obfuscation techniques even when attackers attempt to subdue their obfuscation by padding it with unobfuscated script contents to overthrow basic checks like character frequency analysis.

Revoke-Obfuscation can easily measure most input PowerShell scripts within 100-300 milliseconds. This level of performance allows an organization to measure the obfuscation of (at worst) 12K+ PowerShell scripts per hour without the need to index verbose PowerShell script block logs in a SIEM.

Lastly, Revoke-Obfuscation supports easy whitelisting functionality along with the ability to ingest PowerShell Operational event log records and reassemble script blocks that are recorded across numerous script block EID 4104 records. It can easily become a one-stop shop for ingesting an environment's PowerShell Operational event logs, reassembling and unique'ing all scripts within those logs, and then identifying obfuscated PowerShell scripts that deserve manual inspection.

Installation

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

To install (from Github):

```
Import-Module .\Revoke-Obfuscation.psd1
```

The source code can also be installed directly from the PowerShell Gallery via the following commands:

To install (from PowerShell Gallery):

```
 Install-Module Revoke-Obfuscation
Import-Module Revoke-Obfuscation
```

Usage

`Revoke-Obfuscation` will provide a detailed tutorial as well as a few other fun surprises. But if you are not into the lulz then you can simply run `Get-Help Measure-RvoObfuscation` to see usage syntax or just continue reading.

There are two primary functions used in this framework:

- **Get-RvoScriptBlock** -- reassembles scripts from EID 4104 script block logs
- **Measure-RvoObfuscation** -- measures input script(s) and returns obfuscation score

If you need to reassemble and extract script block logs from PowerShell Operational logs then `Get-RvoScriptBlock` is your function of choice. It automatically returns only unique script blocks and excludes certain default script block values deemed not malicious. This can be overridden with the -Deep switch.

- `Get-RvoScriptBlock -Path 'C:\Windows\System32\Winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx' -Verbose`
- `Get-ChildItem .\Demo\demo.evtx | Get-RvoScriptBlock -Verbose`
- `Get-WinEvent -LogName Microsoft-Windows-PowerShell/Operational | Get-RvoScriptBlock -Verbose`

`Get-RvoScriptBlock` also supports MIR/HX audit results as well as PowerShell Operational logs retrieved via Matt Graeber's (@mattifestation) CimSweep project (https://github.com/PowerShellMafia/CimSweep). For CimSweep there is a minor registry tweak required to trick WMI into querying a non-classic event log. Details can be found in the NOTES section of `Get-RvoScriptBlock`.

- `Get-ChildItem C:\MirOrHxAuditFiles\*_w32eventlogs.xml | Get-RvoScriptBlock -Verbose`
- `Get-CSEventLogEntry -LogName Microsoft-Windows-PowerShell/Operational | Get-RvoScriptBlock`

A full example against test data recorded in demo.evtx can be found below:

```
$obfResults = Get-WinEvent -Path .\Demo\demo.evtx | Get-RvoScriptBlock | Measure-RvoObfuscation -OutputToDisk
```

A full example against local and remotely hosted test scripts can be found below:

- `Measure-RvoObfuscation -Url 'http://bit.ly/DBOdemo1' -Verbose -OutputToDisk`
- `Get-Content .\Demo\DBOdemo*.ps1 | Measure-RvoObfuscation -Verbose -OutputToDisk`
- `Get-ChildItem .\Demo\DBOdemo*.ps1 | Measure-RvoObfuscation -Verbose -OutputToDisk`

The `-OutputToDisk` switch will automatically output all obfuscated scripts to .\Results\Obfuscated. Regardless, all results will be returned as PSCustomObjects containing the script content along with metadata like an obfuscation score, measurement time, whitelisting result, all extracted script features, etc.

Three whitelisting options exist in two locations in Revoke-Obfuscation:

1. On Disk (automatically applied if present):
    1. **.\Whitelist\Scripts_To_Whitelist** -- All scripts placed in this directory will be hashed and any identical scripts will be whitelisted. This whitelisting method is preferred above the next two options.
    2. **.\Whitelist\Strings_To_Whitelist.txt** -- A script containing ANY of the strings in this file will be whitelisted. *Syntax: Rule_Name,string_to_whitelist*
    3. **.\Whitelist\Regex_To_Whitelist.txt** -- A script containing ANY of the regular expressions in this file will be whitelisted. *Syntax: Rule_Name,regex_to_whitelist*
2. Arguments for Measure-RvoObfuscation (applied in addition to above whitelisting options):
    1. **-WhitelistFile** -- `-WhitelistFile .\files\*.ps1,.\more_files\*.ps1,.\one_more_file.ps1`
    2. **-WhitelistContent** -- `-WhitelistContent 'string 1 to whitelist','string 2 to whitelist'`
    3. **-WhitelistRegex** -- `-WhitelistRegex 'regex 1 to whitelist','regex 2 to whitelist'`

If interested in creating your own set of training data and generating a weighted vector for the Measure-Vector function, then ModelTrainer.cs/ModelTrainer.exe can be executed against a labeled data set. The following command will extract feature vectors from all input scripts and aggregate them into a single CSV used in this training phase:

```
Get-ChildItem .\*.ps1 | ForEach-Object { [PSCustomObject](Get-RvoFeatureVector -Path $_.FullName) | Export-Csv
```

Lastly, if looking for a platform for creating indicators (IOCs) that harness the power of PowerShell's AST (Abstract Syntax Tree) -- which we would highly recommend for identifying malicious PowerShell activity that is NOT obfuscated -- then PS Script Analyzer is an excellent framework designed to handle such tasks.

License

Revoke-Obfuscation is released under the Apache 2.0 license.

Release Notes

v1.0 - 2017-07-27 Black Hat USA & 2017-07-30 DEF CON: PUBLIC Release of Revoke-Obfuscation.

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