Sign in

danielbohannon / Invoke-DOSfuscation
Public

Notifications | Fork 133 | Star 822

Code | Issues | Pull requests | Actions | Projects | Security | Insights

master | Go to file | Code

- Samples
- Tests
- Invoke-DOSfuscation...
- Invoke-DOSfuscation...
- Invoke-DOSfuscation...
- Invoke-DOSfuscation...
- LICENSE
- README.md

About

Cmd.exe Command Obfuscation Generator & Detection Test Harness

- Readme
- Apache-2.0 license
- Activity
- 822 stars
- 28 watching
- 133 forks

Report repository

Releases

No releases published

Packages

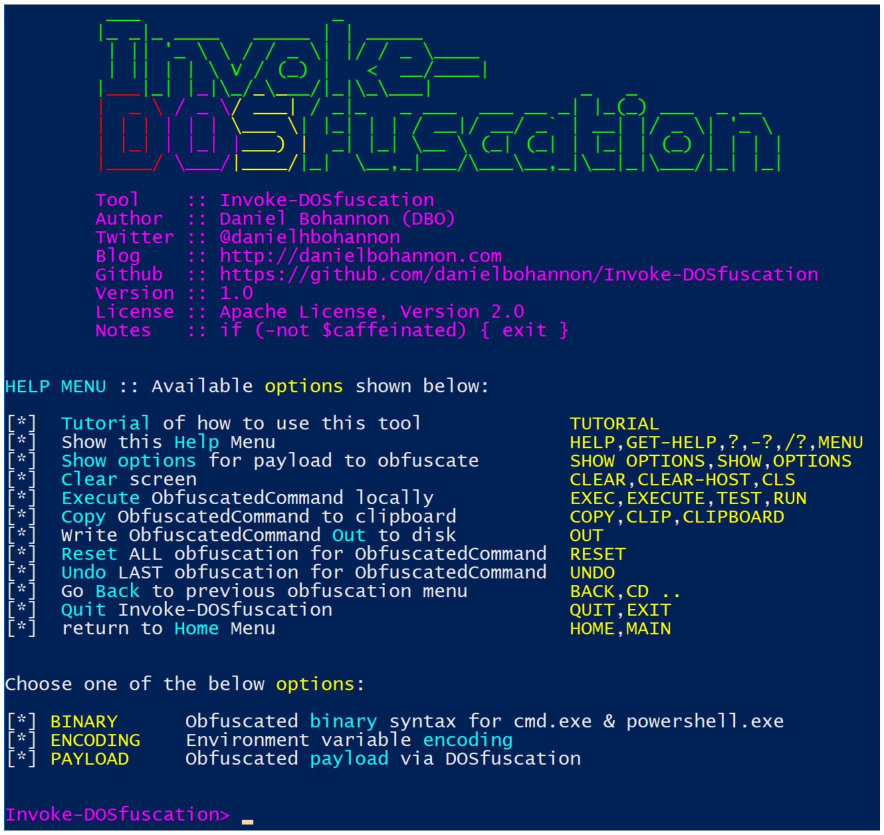
No packages published

Languages

● PowerShell 100.0%

README | Apache-2.0 license

Invoke-DOSfuscation v1.0


```
        Tool    :: Invoke-DOSfuscation
        Author  :: Daniel Bohannon (DBO)
        Twitter :: @danielhbohannon
        Blog    :: http://danielbohannon.com
        Github  :: https://github.com/danielbohannon/Invoke-DOSfuscation
        Version :: 1.0
        License :: Apache License, Version 2.0
        Notes   :: if (-not $caffeinated) { exit }

HELP MENU :: Available options shown below:

[*]  Tutorial of how to use this tool              TUTORIAL
[*]  Show this Help Menu                           HELP,GET-HELP,?,-?,/?,MENU
[*]  Show options for payload to obfuscate         SHOW OPTIONS,SHOW,OPTIONS
[*]  Clear screen                                  CLEAR,CLEAR-HOST,CLS
[*]  Execute ObfuscatedCommand locally             EXEC,EXECUTE,TEST,RUN
[*]  Copy ObfuscatedCommand to clipboard           COPY,CLIP,CLIPBOARD
[*]  Write ObfuscatedCommand Out to disk           OUT
[*]  Reset ALL obfuscation for ObfuscatedCommand   RESET
[*]  Undo LAST obfuscation for ObfuscatedCommand   UNDO
[*]  Go Back to previous obfuscation menu          BACK,CD ..
[*]  Quit Invoke-DOSfuscation                      QUIT,EXIT
[*]  return to Home Menu                           HOME,MAIN


Choose one of the below options:

[*] BINARY      Obfuscated binary syntax for cmd.exe & powershell.exe
[*] ENCODING    Environment variable encoding
[*] PAYLOAD     Obfuscated payload via DOSfuscation


Invoke-DOSfuscation> _
```

Introduction

Invoke-DOSfuscation is a PowerShell v2.0+ compatible cmd.exe command obfuscation framework. (White paper: https://www.fireeye.com/blog/threat-research/2018/03/dosfuscation-exploring-obfuscation-and-detection-techniques.html)

Background

Over the past several years as an Incident Response consultant I have witnessed a myriad of obfuscation and evasion techniques employed by several threat actors. Some of these techniques are incredibly complex while others are tastefully simple, but both categories are employed to evade detection. In my experience, I have found APT32 and FIN7 to pull out the most alluring obfuscation techniques and their creativity is noteworthy.

In June 2017 after a slew of incremental command line obfuscation techniques, FIN7 used an environment variable string substitution capability native to cmd.exe that at the time I did not know even existed. Spurred by this discovery I co-authored a blog post with Nick Carr (@ItsReallyNick) called Obfuscation in the Wild: Targeted Attackers Lead the Way in Evasion Techniques where we highighted numerous groups' obfuscation techniques we identified in the wild.

The following weekend I explored this native cmd.exe substitution functionality and built and released a simple Proof-of-Concept tool called Out-FINcodedCommand. As soon as I released this POC I stepped back and thought, "*Maybe there are more obfuscation capabilities within cmd.exe that I should explore and build detections for before we discover it being used in the wild.*"

That was the inception of my research that eventually led me to develop this Invoke-DOSfuscation framework.

Purpose

Attackers are increasingly using obfuscation techniques to evade detections based heavily on command line argument values. To counter this I spent five months researching and developing obfuscation and encoding techniques native to cmd.exe so that I could create robust detections for these core techniques that I have not yet seen in the wild.

This framework's sole purpose is to enable defenders to randomly generate thousands of uniquely obfuscated sample commands to test and tune their detection capabilities against these techniques. I included my full test harness in this release to automate this detection testing process.

In many ways this framework enables defenders to fuzz cmd.exe's obfuscation techniques, and in building this tool I discovered numerous additional obfuscation opportunities that I did not uncover in my initial research.

I also shared this information with Microsoft in November 2017 and inquired about opportunities to expose additional visibility into the inner workings of cmd.exe's usage of the core techniques that are the building blocks of all of the obfuscation functions built into this framework.

As a defender the end goal of this research and development is to raise awareness and drive change that will help end users better protect themselves against attackers' ever-changing TTPs (Tools, Techniques and Procedures).

Usage

While all of the obfuscation components are built out into standalone formal functions, most users will find the `Invoke-DOSfuscation` function to be the easiest way to explorer and visualize the obfuscation techniques that this framework supports. For fuzzing and deep exploration of the numerous tuning options for each obfuscation category, though, it is recommended that the individual functions be used directly outside of the `Invoke-DOSfuscation` function wrapper.

To enable defenders to easily begin fuzzing and testing detection ideas, this framework also includes an additional module, `Invoke-DOSfuscationTestHarness.psm1` that is automatically imported with the rest of the project. The two key functions in this module for defenders are:

1. `Invoke-DosTestHarness` - Generates (with default argument settings) over 1000 randomly-obfuscated commands from a list of test commands for both payload integrity and detection purposes. Each test harness iteration randomizes all available function arguments and calls the four obfuscation functions directly instead of using the more standardized `-ObfuscationLevel` values (1-3) that the Invoke-DOSfuscation menu-driven function

uses by default. This produces a significantly wider range of obfuscation output against which one can build more thorough detections. Each obfuscated command is then checked against the second function:

2. `Get-DosDetectionMatch` - Checks an input command (string) against all regex detection values input into the `$regexDetectionTerms` array in the function. This is automatically called by `Invoke-DosTestHarness` but can be called in a stand-alone fashion as well.

At the end of each test harness invocation statistics for proper command execution and detection will be displayed so defenders can quickly identify which commands have 0 or only 1-2 detection matches and which might need better coverage.

Finally, to avoid requiring defenders to have to run the test harness to get started, I have provided 1000 sample commands in the Samples directory broken out across each of the four obfuscation functions. The formats include .txt files, and Security and Sysmon .evtx files. I included a couple sample detection rules in the `$regexDetectionTerms` array in `Invoke-DOSfuscationTestHarness.psm1`, but you will want to add and test many more rules as you test the sample obfuscated commands). In addition to .txt files with the raw commands there are .evtx files containing the process execution logs (Sysmon EID 1 events) for each of the 1000 sample commands. These event logs are tremendously helpful for building indicators based on nuanced child process executions that are byproducts of several building block techniques upon which Invoke-DOSfuscation is built.

To get started, display sample obfuscated commands that do *not* match any of your current detection rules by simply running:

```
(Get-Content .\Samples\STATIC_1-of-4_Out-DosConcatenatedCommand.txt) | where-object { -not (Get-DosDetectionMatch -Command $_).Detected }
```

To test both obfuscated commands and child processes from execution event logs recorded in Security EID 4688, simply run:

```
(Get-WinEvent -Path .\Samples\DYNAMIC_SECURITY_EID4688_1-of-4_Out-DosConcatenatedCommand.evtx) | where-object { $_.Message } | foreach-object { ($_.Message.Split("`n") | select-string '^\tProcess Command Line:\t') -replace '^\tProcess Command Line:\t','' } | where-object { -not (Get-DosDetectionMatch -Command $_).Detected }
```

Installation

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

To install:

```
Import-Module .\Invoke-DOSfuscation.psd1
Invoke-DOSfuscation
```

License

Invoke-DOSfuscation is released under the Apache 2.0 license