**Exploring the Depths of Cmd.exe Obfuscation and Detection Techniques**

Skilled attackers continually seek out new attack vectors while employing evasion techniques to maintain the effectiveness of old vectors in an ever-changing defensive landscape. Numerous threat actors employ obfuscation frameworks for common scripting languages like JavaScript and PowerShell to thwart signature-based detections of common offensive tradecraft written in these languages.

However, as defenders’ visibility into these popular scripting languages increases through better logging practices1 and inline inspection of the execution phases of these languages via Microsoft’s Antimalware Scan Interface2, some stealthy attackers have shifted their tradecraft to languages that do not support this additional visibility. At a minimum, determined attackers are adding dashes of simple obfuscation to previously detected payloads and commands to break rigid detection rules.

FireEye’s Advanced Practices Team is dedicated to developing detection capabilities for advanced TTPs (Tools, Techniques and Procedures) that attackers use in the wild. The author’s role as a Senior Applied Security Researcher on this team entails researching existing and new areas of obfuscation and evasion to ultimately build more robust detection capabilities. Enumerating new problem spaces empowers one to more effectively detect the elusive tricks used by today’s threat actors. This approach also drives forward detection capabilities for obfuscation techniques not yet identified in the wild.

In June 2017, the Advanced Practices Team identified FIN7 (a financially-motivated threat actor also known as Carbanak) testing a novel obfuscation technique native to cmd.exe. Prompted by this discovery, the author began researching obfuscation techniques supported by cmd.exe and hunting for their usage across client and customer environments and in public and private file repositories. These findings represent nine months of dedicated research, detection development and threat hunting across 10+ million endpoints all around the world.

The goal of this research is to enumerate the problem space of cmd.exe-supported obfuscation techniques to stay ahead of the next obfuscation trick that FIN7 or other threat actors might employ. It is with this defensive mindset that the author presents these research findings so other defenders can more effectively detect these obfuscation and evasion techniques.

Obfuscation in the Wild

Numerous threat actors that FireEye tracks have increasingly used obfuscation to attempt to evade rigid detections. In June 2017, the author co-authored a blog post3 with FireEye Incident Response Manager Nicholas Carr outlining three separate command line obfuscation techniques their team identified being used in the wild by three separate threat actors.

The first example originates from a phishing document attributed to FIN8, a financial threat actor with notably aggressive phishing campaigns. This document contains an obfuscated macro that uses process-level environment variables and PowerShell’s standard input command functionality to hide all meaningful command line arguments from winword.exe’s child process of cmd.exe and its grandchild process of powershell.exe.

De-obfuscated macro from FIN8 phishing document (February 2017)



Mandiant incident responders captured a second example in real-time event data while responding to an APT32 (aka OceanLotus) intrusion in April 2017. This Vietnam-based threat actor, whose motivations appear to align with Vietnamese-government interests, frequently uses the Invoke-Obfuscation4 PowerShell obfuscation framework to heavily obfuscate Cobalt Strike Beacon backdoor downloaders, but often downloads this second stage using the regsvr32.exe remote download technique known as “Squiblydoo”. To evade rigid signatures for this technique that rely on command line argument values /i:http:// or /i:https:// being present, APT32 first used cmd.exe’s escape character, the caret (^), and then in this later example used double quotes to break up these arguments.

Obfuscated regsvr32.exe command from APT32 (April 2017)

****

The FireEye Advanced Practices Team identified the final example in a phishing document attributed to FIN7. The document employs novel execution and obfuscation techniques spread across multiple payloads. The document first drops a LNK file to disk and executes it. The LNK file writes an obfuscated JScript file to disk at %HOMEPATH%\ md5.txt and executes it with wscript.exe. The JScript file then retrieves and executes the final payload from the original document via a Word.Application COM object. The JScript file contains a combination of concatenation (“Wor”+”d.Application”) and ASCII encoding to obfuscate the suspicious eval function: (this[String. fromCharCode(101)+’va’+’l’]). However, the LNK file contains the more novel obfuscation technique highlighted in the below screenshot:

Obfuscated cmd.exe command from malicious FIN7 LNK file (June 2017)

****

The attacker sets the wscript.exe command in a process-level environment variable called x before passing it to the final cmd.exe as standard input. The attacker also obfuscates the strings wscript and /e:jscript in the original cmd.exe command using @ characters. The @ characters are later removed from the command contents stored in the environment variable x using cmd.exe’s native variable string replacement functionality. This string replacement functionality follows the form %VariableName:StringToFind=NewString% where StringToFind is the @ character and NewString is blank, so the @ character is simply removed. This string replacement technique allows the LNK file’s obfuscated wscript.exe command to be de-obfuscated in memory before being passed to the final cmd.exe execution via standard input.

A simplified illustration of this sample’s variable string replacement technique is shown below:

Simplified illustration of variable string replacement

****

This technique was effective in bypassing several static detections and prompted the author to begin this research initiative of exploring cmd.exe-supported obfuscation techniques.