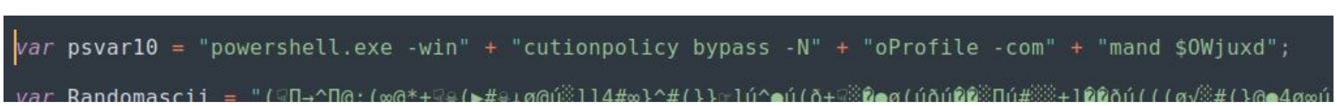
Analysis of malicious jscript file October 27, 2023
SHA1 hash: cfa56fbf470791ebf07578c82795a3fab77c6ea2
MD5 hash: c00dda7545e32fe221c2e7ba777cfed3
Malware obtained from Malware Bazaar
First seen: 2023-10-26 12:45:10 UTC

The Jscript file showed lots of obfuscated code. I could see lots of long variable names, long string values. Looking at the code preview on the right side of Sublime-Text I could see a pattern in it.

I began by making a copy of the Jscript file to work on. The first step was to replace names of variables. If I could figure what they were the name would explain this. Many variables were nonsensical output values. I began by giving them a shorter variable name. This cleared up some things. As I proceeded the variables looked like they were repeating, doing the same thing over and over again. This appeared to be useless code to mask the real instructions and prevent analysis.

There was one fuction that was used in random places. This was a reverse string function. It was simple. I cleaned it up a bit with easier to see variable names and proceeded further.

I did notice in each repeated section there was a variable buried that was defining strings using char code values and some of these strings would call the reverse function. Not all of them. I quickly ran each line through NodeJS. I replaced each variable with its proper string. I began to see things that showed it was calling a Wscript shell.

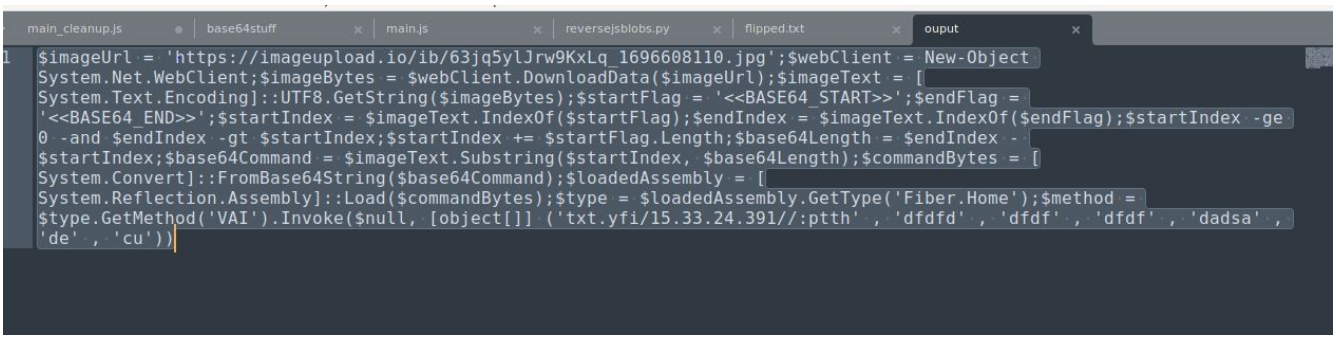


There was a chunk of garbled text and symbols that. It began with two equal signs and it was being put through the reverse function. I reversed it and tried decoding. This was unsuccessful. The odd characters were spread throughout the blob of text.

I decided to see where this was being decoded. There was no functions or other calls to decode it.

Confident the repeated variables were just noise, I removed them. This left the blob of encoded text, the reverse string function, and the variables that were combined at the bottom. I began to put them together and saw the Wscript code was decoding the blob of text and before it did that it was replacing the strange character with ‘A’. I quickly did this separately with the basic find and replace. I then reversed it and decoded it. It was a PowerShell script.

The malware would reverse, replace and decode the text and run powershell.exe bypassing execution policy and running the script code.



I then began to investigate this section of the attack. I could see two URLs within this script. The first was download a jpg file. The last URL was reversed, but would download a txt file.

I spent some time attempting to curl, wget and even use the first two commands in this payload to grab the jpg file. This proved fruitless. All methods failed. I decided to see what URLScan.io said it would not scan it as it is blacklisted.

I threw it into VirusTotal. 4 vendors showed it was malicious. It did have a banner indicating this was related to async.RAT malware.

I did grab the txt file from the second URL. This was base64 encoded. I decoded, but the results showed it as data. Strings did not reveal anything. I noticed in the PowerShell script it talked about Assembly.

Tried running the lines of code with PowerShell, on my Remnux system. Most lines failed with errors. Couldn't get it to do anything or tell me anything more about what it was trying to do. I can only assume it would take the txt file and change it into something that would be the next stage of the infection. The jpg file most likely contained code that would help facilitate this. Unable to confirm if that's the case.

At this time I've ended my analysis. I've been able to get more experience into methods malware authors use. I can see how base64 can be obfuscated further to prevent decoding, and how to identify noise within code.

New skills I need to learn, how Assembly is used in PowerShell and what that does.