Powershell Obfuscation Detection Using Science

Some organizations and SIEM vendors rely entirely on command---line logging (4688) event rather than PowerShell script block logs (4104) to detect malicious PowerShell. If the process name is “PowerShell” and the command---line arguments match some specific patterns, they flag that input as malicious.

There are two main ways that attackers can avoid this form of 4688---based command line logging: obscuring the name of the PowerShell executable, and breaking the link between PowerShell and the code it invoked as viewed from the command line logs. While these launch techniques do not evade PowerShell script block logging, they are challenging behaviours to detect through command---line logging alone.

String concatenation is a common way to break up identifiable strings. If a signature is written for the term, “http”, it can also be written as “h” + “ttp”. The most common form of concatenation is the ‘+’ operator, but PowerShell’s –join operator can also be used. In addition to PowerShell techniques, the String.Join() and String.Concat() methods from .NET can accomplish the same goals.

While the situation may appear dire when it comes to detecting malicious PowerShell due to the vast range of obfuscation opportunities, in fact the opposite is true. The crucial insight is that obfuscated code looks nothing like regular code.

So rather than (or in addition to) detecting known signatures, we must enrich our detection capabilities by post---processing Script Block and command---line logs to look for signs of obfuscation. If we find obfuscated PowerShell, then we can kick off a secondary investigation to determine its purpose and intent.

One of the first pieces of insight we can take action on is based on character frequency. If we analyze the entropy and letter frequency distribution of the variable names, we can see that we can get some pretty strong signals from this approach.

We can take this character frequency approach even further. Rather than analyze the frequency of the top four letters, we can analyze the frequency of each letter in a script. When we compare that to the character frequency of some obfuscated samples, there is clearly a significant difference:

Non-Obfuscated

Name	Percent
E	9.912
T	7.414
A	5.512
R	5.43
S	5.303
I	5.041
N	5.025
O	4.944
L	3.509
M	3.3

Obfuscated

Name	Percent
$	21.808
{	21.659
}	21.659
+	13.313
"	7.452
=	2.832
[	2.086
(	1.689
;	1.54
)	1.341