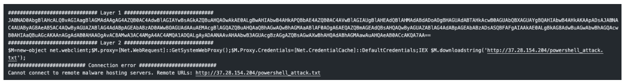
Bulk De-obfuscation and Analysis of PowerShell Malware with PowerDrive

Jeff Malavasi
Dept. of Computing Security
Rochester Institute of Technology
Email: jm3378@rit.edu

I. Tool Analysis

PowerShell is an extremely useful and robust scripting language that is primarily used to administer Windows computers and servers at scale. Recently, it has even expanded to be cross compatible with Unix systems. One of the reasons that the language is so popular is due to it's ability to be extremely forgiving. For example, aliases are automatically created for all MS-DOS commands for backwards compatibility and cmdlets are able to fill in incomplete instructions. Due to this ability to call the same command in multiple ways, PowerShell is also extremely useful to bad actors, as obfuscation becomes part of the language.

When creating a malicious PowerShell script, threat actors often use multiple layers of obfuscation including reordering strings, encoding and compression. Being able to properly deobfuscate these scripts is extremely important as it allows analysts to detect potential files or URLs in the code as well as determine the attackers goals. One tool that can be used to deobfuscate PowerShell scripts is called PowerDrive [1]. It was created in 2019 and is able to deobfuscate multi-layer scripts using both static and dynamic analysis. The module takes an input file as a parameter and starts by using a set of functions to determine the number of layers in the file. For example the script could use sting manipulation before encoding the entire fire. During this stage PowerDrive stores each of the layers in an array. The majority of the rules are accomplished using Regex filters. The next stage, pre-processing works by linting the file and will remove any syntax errors, endless loops, try-catch blocks, null redirection, and sleep functions. Next, the script is deobfuscated using a series of functions including a Base64 decoder. Additionally, PowerDrive relies on the automatic decoding ability of Invoke-Expression. When this cmdlet is called PowerShell automatically deobfuscates the provided string. PowerDrive uses cmdlet override in order to capture this data. Additionally, the module also overrides Invoke-WebRequest and Invoke-Rest in order to protect the analyzing system and intercept web traffic. Next, in the final stage, the script is executed for dynamic analysis. Finally, the script returns a string output with each of the layers written out like the below example:



```
############################## Layer 1 ##############################
JABNAD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAOwAkAE0ALgBwAHIAbwB4AHkAPQBbAE4AZQB0AC4AVwBlAGIAUgBlAHEAdQBlAHMAdABdADoAOgBHAGUAdABTAHkAcwB0AGUAbQBXAGUAYgBQAHIAbwB4AHkAKAApADsAJABNA
C4AUAByAG8AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwByAGUAZABlAG4AdABpAGEAbABzADsASQBFAFgAIAAkAE0ALgBkAG8AdwBuAGwAbwBhAGQAcw
B0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMwA3AC4AMgA4AC4AMQA1ADQALgAyADAANAAvAHAAbwB3AGUAcgBzAGgAZQBsAGwAXwBhAHQAdABhAGMAawAuAHQAeAB0ACcAKQA7AA==
############################## Layer 2 ##############################
$M=new-object net.webclient;$M.proxy=[Net.WebRequest]::GetSystemWebProxy();$M.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $M.downloadstring('http://37.28.154.204/powershell_attack.
txt');
######################### Connection error #########################
Cannot connect to remote malware hosting servers. Remote URLs: http://37.28.154.204/powershell_attack.txt
```

II. Limitations

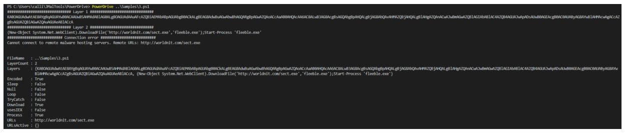
While PowerDrive is extremely effective at analyzing a single malicious file it has a few limitations. Firstly, the module returns string output which is difficult to parse or analyze programmatically. This makes the tool difficult to scale and be useful for a large enterprise. PowerShell has the ability to create custom objects, which can be passed through the pipeline for further analysis. Additionally, the tool does not return any information about the results of the dynamic analysis besides whether or not the remote URL is still active. It fails to tell the analyst which type of obfuscation techniques were used, if anti-debugging was employed, or what actions the malware took during execution. Finally, provided with the tool is an example dataset, however all the files are grouped together and requires manual extraction and conversion to PowerShell.

III. Proposed Improvements

An updated version of PowerDrive was created and uploaded to Github [2]. The code was written and tested on a Windows 10 virtual machine in a contained network.

A. Custom PSObject

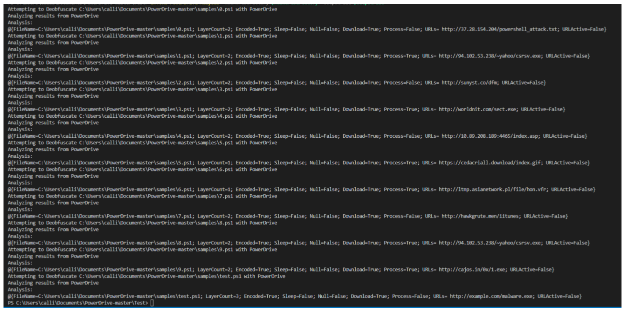
In order to address the output limitation, a custom PSObject was created that stores the results of the analysis both before and after the change (below). This object stores the number of layers, all techniques used by the threat actor, as well as a behavioral analysis of the execution. Additionally, this output can be redirected using the PowerShell pipeline in order to store the results in a CSV that could be ingested by a SIEM for automatic detection and alerting.



```
PS C:\Users\calli\JMalTools\PowerDrive> PowerDrive ..\Samples\3.ps1
############################## Layer 1 ##############################
KABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQARgBpAGwAZQAoACcAaAB0AHQAcAA6AC8ALwB3AG8AcgBsAGQAbgBpAHQALgBjAG8AbQAvAHMAZQBjAHQALgBlAHgAZQAnACwAJwBmAGwAZQBlAGIAbABlAC4AZQB4AGUAJwApADsAUwB0AGEAcgB0AC0AUAByAG8AYwBlAHMAcwAgACcAZ
gBsAGUAZQBiAGwAZQAuAGUAeABlACcA
############################## Layer 2 ##############################
(New-Object System.Net.WebClient).DownloadFile('http://worldnit.com/sect.exe','fleeble.exe');Start-Process 'fleeble.exe'
######################### Connection error #########################
Cannot connect to remote malware hosting servers. Remote URLs: http://worldnit.com/sect.exe


FileName   : ..\Samples\3.ps1
LayerCount : 2
Layers     : {KABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQARgBpAGwAZQAoACcAaAB0AHQAcAA6AC8ALwB3AG8AcgBsAGQAbgBpAHQALgBjAG8AbQAvAHMAZQBjAHQALgBlAHgAZQAnACwAJwBmAGwAZQBlAGIAbABlAC4AZQB4AGUAJwApADsAUwB0AGEAcgB0AC0AUAByAG8AYw
             BlAHMAcwAgACcAZgBsAGUAZQBiAGwAZQAuAGUAeABlACcA, (New-Object System.Net.WebClient).DownloadFile('http://worldnit.com/sect.exe','fleeble.exe');Start-Process 'fleeble.exe'}
Encoded    : True
Sleep      : False
Null       : False
Loop       : False
TryCatch   : False
Download   : True
usesIEX    : False
Process    : True
URLs       : http://worldnit.com/sect.exe
URLsActive : {}
```

B. Bulk Analysis Script

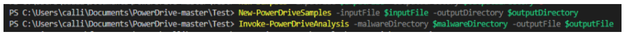
The next improvement that was made was the creation of a bulk analysis script (Invoke-PowerDriveAnalysis). Instead of scanning a single file, the script recursively scans a directory for all PowerShell files and stores the results in a CSV. This script can be run as a scheduled task and can be deployed to many machines in order to use PowerDrive at scale.



```
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\0.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\0.ps1; LayerCount=2; Encoded=True; Sleep=False; Null=False; Download=True; Process=False; URLs= http://37.28.154.204/powershell_attack.txt; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\1.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\1.ps1; LayerCount=2; Encoded=True; Sleep=False; Null=False; Download=True; Process=True; URLs= http://94.102.53.238/~yahoo/csrsv.exe; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\2.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\2.ps1; LayerCount=2; Encoded=True; Sleep=False; Null=False; Download=True; Process=False; URLs= http://sunyst.co/dfm; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\3.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\3.ps1; LayerCount=2; Encoded=True; Sleep=False; Null=False; Download=True; Process=True; URLs= http://worldnit.com/sect.exe; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\4.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\4.ps1; LayerCount=2; Encoded=True; Sleep=False; Null=False; Download=True; Process=False; URLs= http://10.89.208.189:4465/index.asp; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\5.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\5.ps1; LayerCount=1; Encoded=True; Sleep=False; Null=False; Download=True; Process=True; URLs= https://cedacriall.download/index.gif; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\6.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\6.ps1; LayerCount=1; Encoded=True; Sleep=False; Null=False; Download=True; Process=True; URLs= http://ltmp.asianetwork.pl/file/hon.vfr; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\7.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\7.ps1; LayerCount=2; Encoded=True; Sleep=False; Null=False; Download=True; Process=False; URLs= http://hawkgrute.men/iitunes; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\8.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\8.ps1; LayerCount=2; Encoded=True; Sleep=False; Null=False; Download=True; Process=True; URLs= http://94.102.53.238/~yahoo/csrsv.exe; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\9.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\9.ps1; LayerCount=2; Encoded=True; Sleep=False; Null=False; Download=True; Process=True; URLs= http://cajos.in/0x/1.exe; URLActive=False}
Attempting to Deobfuscate C:\Users\calli\Documents\PowerDrive-master\samples\test.ps1 with PowerDrive
Analyzing results from PowerDrive
Analysis:
@{FileName=C:\Users\calli\Documents\PowerDrive-master\samples\test.ps1; LayerCount=3; Encoded=True; Sleep=False; Null=False; Download=True; Process=False; URLs= http://example.com/malware.exe; URLActive=False}
PS C:\Users\calli\Documents\PowerDrive-master\Test> 
```

C. *Sample Conversion Script*

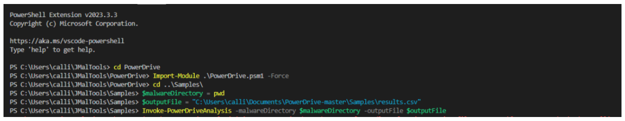
Finally, a second script (New-Samples) was created in order to extract all of the samples from the provided text file and convert them to PowerShell. It works by parsing out the first layer from each script and storing it in a provided directory.



```
PS C:\Users\calli\Documents\PowerDrive-master\Test> New-PowerDriveSamples -inputFile $inputFile -outputDirectory $outputDirectory
PS C:\Users\calli\Documents\PowerDrive-master\Test> Invoke-PowerDriveAnalysis -malwareDirectory $malwareDirectory -outputFile $outputFile
```

IV. Case Study: Automatic Detection using PowerDrive and Splunk

With the added improvements to the tool, it can now be used for automatic detection with a SIEM. In order to do this we need to deploy the module to our target machine or server that we want to scan. Then we can import the module, point it at the directory we want to scan, and provide a location for the results file. Additionally, this task can be setup using Microsoft's Task Scheduler in order to run on a regulary basis.

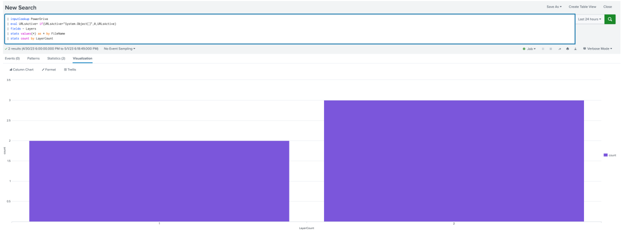


```
PowerShell Extension v2023.3.3
Copyright (c) Microsoft Corporation.

https://aka.ms/vscode-powershell
Type 'help' to get help.

PS C:\Users\calli\JMalTools> cd PowerDrive
PS C:\Users\calli\JMalTools\PowerDrive> Import-Module .\PowerDrive.psm1 -Force
PS C:\Users\calli\JMalTools\PowerDrive> cd ..\Samples\
PS C:\Users\calli\JMalTools\Samples> $malwareDirectory = pwd
PS C:\Users\calli\JMalTools\Samples> $outputFile = "C:\Users\calli\Documents\PowerDrive-master\Samples\results.csv"
PS C:\Users\calli\JMalTools\Samples> Invoke-PowerDriveAnalysis -malwareDirectory $malwareDirectory -outputFile $outputFile
```

Once the results file is generated, it can be forwarded to Splunk using Universal Forwarder or by simply copying the folder to a network location that Splunk is actively monitoring. For this example, I have uploaded the results file to a Splunk server as a lookup named PowerDrive. Once this is complete we can create alerts and dashboards based on behavior, number of layers, or even active URLs. For example, if we want to alert on any scripts that have at least one active URL, we can run the query:

```
| inputlookup PowerDrive
| eval URLsActive= if(URLsActive="System.Object[]",0,URLsActive)
| fields - Layers
| stats values(*) as * by FileName
| search URLsActive!=0
```

Additionally, since the URLs are captured, they can be ran through VirusTotal or another reputation filter for additional analysis. Finally, a graph of the number of layers could be created in order to get additional feedback about what types of threat actors may be involved.



```
| inputlookup PowerDrive
| eval URLsActive= if(URLsActive="System.Object[]",0,URLsActive)
| fields - Layers
| stats values(*) as * by FileName
| stats count by LayerCount
```

✓ 2 results (4/30/23 6:00:00.000 PM to 5/1/23 6:18:49.000 PM)

Future work could include analyzing the results using a machine learning model, similar to previous work in the field [3], [4].

References

[1] D. Ugarte, D. Maiorca, F. Cara, and G. Giacinto, "Powerdrive: Accurate de-obfuscation and analysis of powershell malware," 2019.

[2] J. Malavasi, "Powerdrive," https://github.com/ursaMaj0r/PowerDrive, 2023.

[3] M.-H. Tsai, C.-C. Lin, Z.-G. He, W.-C. Yang, and C.-L. Lei, "Powerdp: De-obfuscating and profiling malicious powershell commands with multi-label classifiers," *IEEE Access*, vol. 11, pp. 256–270, 2023.

[4] S. Choi, "Malicious powershell detection using attention against adversarial attacks," *Electronics*, vol. 9, no. 11, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/11/1817