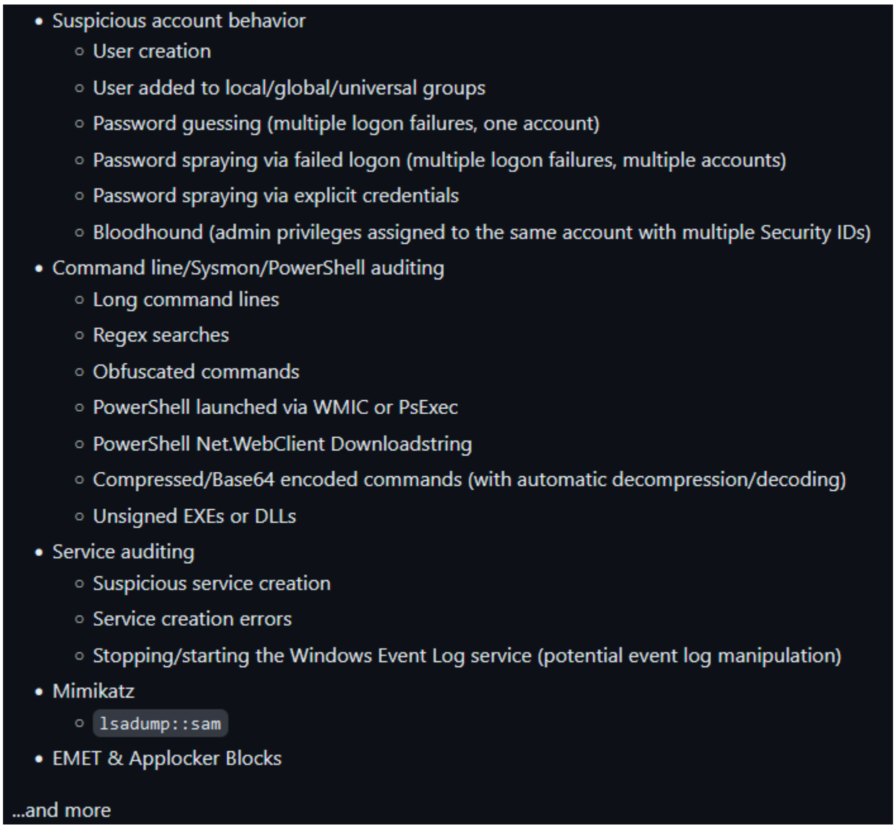
DeepBlueCLI Report

DeepBlueCLI: Enhancing Threat Hunting with Windows Event Logs

Up until recently, I had only a basic understanding of Windows Event Logs and Sysmon logs, primarily knowing that they recorded important events that occur on a system like logins, process creation, and more. Thanks to the TryHackMe SOC Level 1 path, I was able to significantly deepen my understanding of these logs and proficiency using PowerShell cmdlets like 'Get-WinEvent' and the Event Viewer (GUI log reader). Through further research, I came across an awesome PowerShell script by Eric Conrad called DeepBlueCLI, designed to enhance threat hunting via Windows Event Logs.

In essence, DeepBlueCLI is a PowerShell script that analyses Windows Event Logs for threat hunting. It extracts and correlates significant events, identifying attacks such as password spraying and more. It is important to note however that it is not an intrusion detection system (IDS) as it is unable to (by default) generate alerts. The screen from the DeepBlueCLI GitHub page below details some, but not all the detectable events:

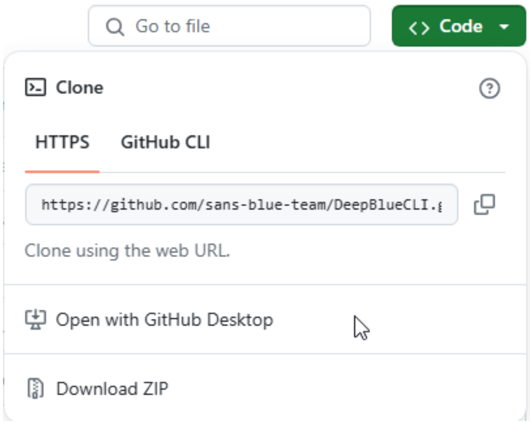


- Suspicious account behavior
  - User creation
  - User added to local/global/universal groups
  - Password guessing (multiple logon failures, one account)
  - Password spraying via failed logon (multiple logon failures, multiple accounts)
  - Password spraying via explicit credentials
  - Bloodhound (admin privileges assigned to the same account with multiple Security IDs)
- Command line/Sysmon/PowerShell auditing
  - Long command lines
  - Regex searches
  - Obfuscated commands
  - PowerShell launched via WMIC or PsExec
  - PowerShell Net.WebClient Downloadstring
  - Compressed/Base64 encoded commands (with automatic decompression/decoding)
  - Unsigned EXEs or DLLs
- Service auditing
  - Suspicious service creation
  - Service creation errors
  - Stopping/starting the Windows Event Log service (potential event log manipulation)
- Mimikatz
  - `lsadump::sam`
- EMET & Applocker Blocks

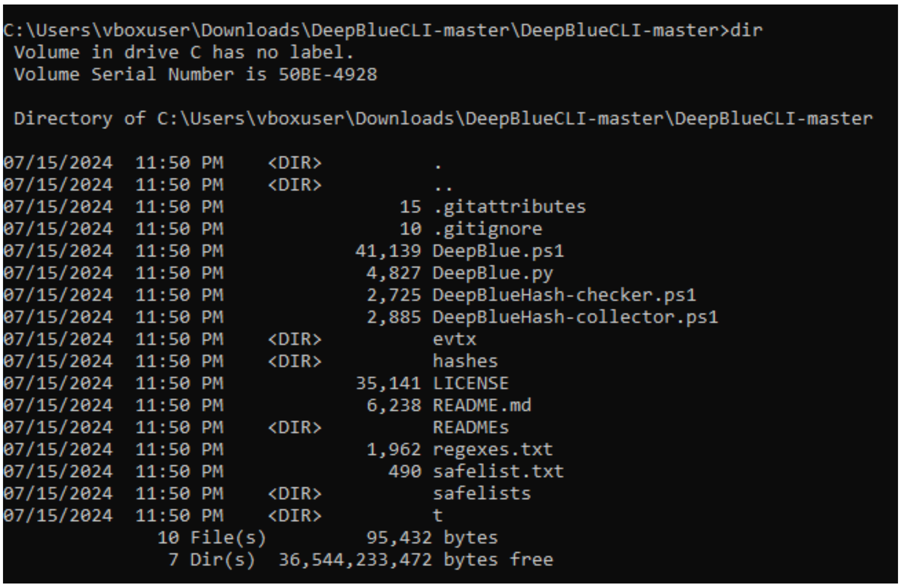
...and more

Using DeepBlueCLI

I will be using DeepBlueCLI on a Windows 10 VM, but you can also use it on your local host since Windows Event Logs themselves are not malicious, nor is the tool. To install the script, navigate to the GitHub repo and download the entire repository as a ZIP file. This download also includes sample event log files and the actual tool script.

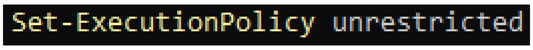


After download the ZIP file, extract it and navigate to the installation directory using PowerShell or CMD (then enter PowerShell) as Administrator:



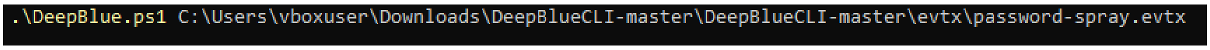
```
C:\Users\vboxuser\Downloads\DeepBlueCLI-master\DeepBlueCLI-master>dir
 Volume in drive C has no label.
 Volume Serial Number is 50BE-4928

 Directory of C:\Users\vboxuser\Downloads\DeepBlueCLI-master\DeepBlueCLI-master

07/15/2024  11:50 PM    <DIR>          .
07/15/2024  11:50 PM    <DIR>          ..
07/15/2024  11:50 PM                15 .gitattributes
07/15/2024  11:50 PM                10 .gitignore
07/15/2024  11:50 PM            41,139 DeepBlue.ps1
07/15/2024  11:50 PM             4,827 DeepBlue.py
07/15/2024  11:50 PM             2,725 DeepBlueHash-checker.ps1
07/15/2024  11:50 PM             2,885 DeepBlueHash-collector.ps1
07/15/2024  11:50 PM    <DIR>          evtx
07/15/2024  11:50 PM    <DIR>          hashes
07/15/2024  11:50 PM            35,141 LICENSE
07/15/2024  11:50 PM             6,238 README.md
07/15/2024  11:50 PM    <DIR>          READMEs
07/15/2024  11:50 PM             1,962 regexes.txt
07/15/2024  11:50 PM               490 safelist.txt
07/15/2024  11:50 PM    <DIR>          safelists
07/15/2024  11:50 PM    <DIR>          t
              10 File(s)         95,432 bytes
               7 Dir(s)  36,544,233,472 bytes free
```

Luckily for us that's all we need to do to start using the tool. Let's now go through an example of using the DeepBlueCLI tool on a sample evtx file. The syntax to analyse a log file in its entirety is .\DeepBlue.ps1 <filename>. Please note, before we execute the script you need to change the execution policy to unrestricted like as follows:



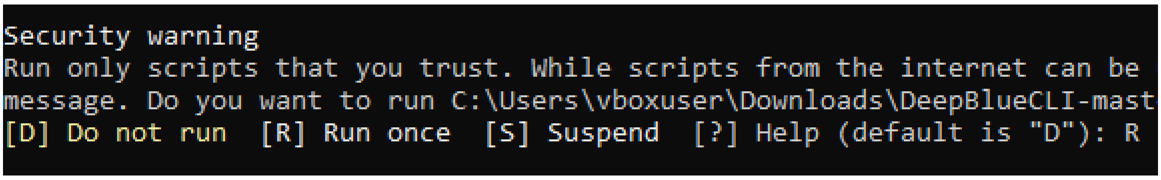
```
Set-ExecutionPolicy unrestricted
```

We can now use the DeepBlueCLI tool:



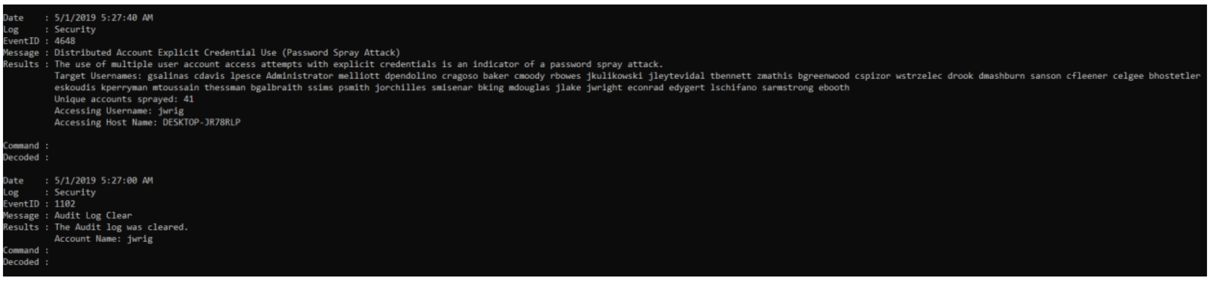
```
.\DeepBlue.ps1 C:\Users\vboxuser\Downloads\DeepBlueCLI-master\DeepBlueCLI-master\evtx\password-spray.evtx
```

Once you have pressed enter, make sure to type ‘R’:



```
Security warning
Run only scripts that you trust. While scripts from the internet can be
message. Do you want to run C:\Users\vboxuser\Downloads\DeepBlueCLI-mast
[D] Do not run  [R] Run once  [S] Suspend  [?] Help (default is "D"): R
```

After giving it some time to analyse the file, you will be presented with hopefully helpful output like seen below:

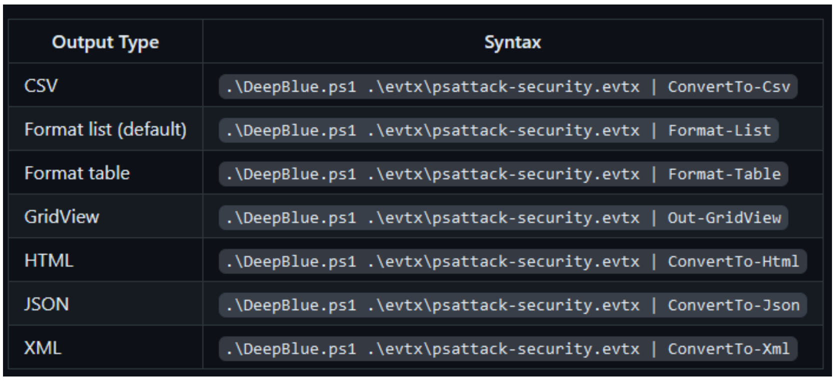


```
Date    : 5/1/2019 5:27:40 AM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a password spray attack.
          Target Usernames: gsalinas cdavis lpesce Administrator melliott dpendolino cragoso baker cmoody rbowes jkulikowski jleytevidal tbennett zmathis bgreenwood cspizor wstrzelec drook dmashburn sanson cfleener celgee bhostetler
          eskoudis kperryman mtoussain thessman bgalbraith ssims psmith jorchilles smisenar bking mdouglas jlake jwright econrad edygert lschifano sarmstrong ebooth
          Unique accounts sprayed: 41
          Accessing Username: jwrig
          Accessing Host Name: DESKTOP-JR78RLP

Command :
Decoded :

Date    : 5/1/2019 5:27:00 AM
Log     : Security
EventID : 1102
Message : Audit Log Clear
Results : The Audit log was cleared.
          Account Name: jwrig
Command :
Decoded :
```

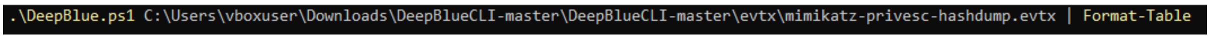
From the output, we can see that the username ‘jwrig’ performed a password spray attack against 41 different user accounts. For context, a password spray attack is when a threat actor tries the same password across multiple usernames. The script also reveals that the same user cleared the log file. This example highlights how helpful DeepBlueCLI is for threat hunting and log analysis, saving time compared to manually exploring log files and correlating events in the Event Viewer or similar tools.

Detecting Mimikatz

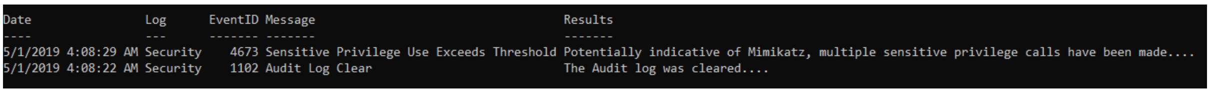
Let’s use this tool against logs produced after Mimikatz was used to extract credentials. This time, I will output the results in a table format (the full list of output types can be seen in the image below):



Output Type	Syntax
CSV	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | ConvertTo-Csv`
Format list (default)	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | Format-List`
Format table	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | Format-Table`
GridView	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | Out-GridView`
HTML	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | ConvertTo-Html`
JSON	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | ConvertTo-Json`
XML	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | ConvertTo-Xml`



```
.\DeepBlue.ps1 C:\Users\vboxuser\Downloads\DeepBlueCLI-master\DeepBlueCLI-master\evtx\mimikatz-privesc-hashdump.evtx | Format-Table
```

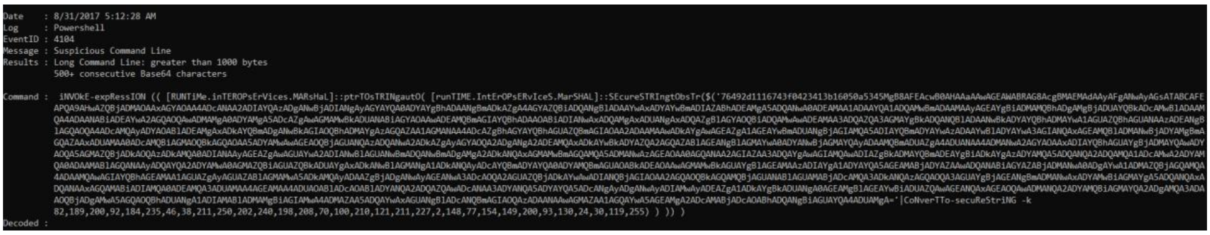


```
Date                 Log      EventID Message                           Results
----                 ---      ------- -------                           -------
5/1/2019 4:08:29 AM Security    4673 Sensitive Privilege Use Exceeds Threshold Potentially indicative of Mimikatz, multiple sensitive privilege calls have been made....
5/1/2019 4:08:22 AM Security    1102 Audit Log Clear                    The Audit log was cleared....
```

As seen in the output above, DeepBlueCLI has detected the use of Minikatz which is a popular password dumping tool used by threat actors.

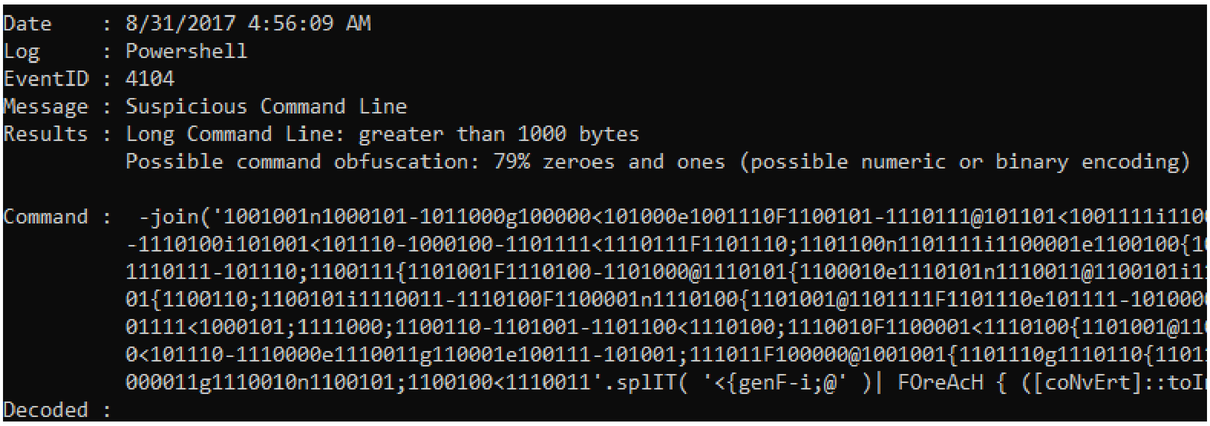
Decoding Obfuscated/Encoded Commands

Threat actors often use encoding tactics to obfuscate their attacks, bypassing basic signature based detection. DeepBlueCLI can detect such anomalies, like unusually long sequences of Base64 characters or binary encoding patterns, which are uncommon in legitimate commands. For example, let’s analyse the Powershell-Invoke-Obfuscation-many.evtx file:



```
Date     : 8/31/2017 5:12:28 AM
Log      : Powershell
EventID  : 4104
Message  : Suspicious Command Line
Results  : Long Command Line: greater than 1000 bytes
           500+ consecutive Base64 characters

Command  :  iNVOkE-expRessION (( [RUNTiMe.inTEROPsErVices.MARsHaL]::ptrTOsTRINgautO( [runTIME.IntErOPsERvIceS.MarSHAL]::SEcureSTRIngtObsTr($('76492d1116743f0423413b16050a5345MgB8AFEAcwB0AHAAaAAwAGEAWABRAG8AcgBMAEMAdAAyAFgANwAyAGsATABCAFE
           APQA9AHwAZQBjADMAOAAxAGYAOAA4ADcANAA2ADIAYQAzADgANwBjADIANgAyAGYAYQA0ADYAYgBhADAANgBmADkAZgA4AGYAZQBiADQANgB1ADAAYwAxADYAYwBmADIAZABhADEAMgA5ADQANwA0ADEAMAA1ADAAYQA1ADQAMwBmADAAMAAyAGEAYgBiADMAMQBhADgAMgBjADUAYQBkADcAMwB1ADAAM
           QA4ADAANABiADEAYwA2AGQAOQAwADMAMgA0ADYAMgA5ADcAZgAwAGMAMwBkADUANABiAGYAOAAwADEAMQBmAGIAYQBhADAAOABiADIANwAxADQAMgAxADUANgAxADQAZgB1AGYAOQBiADQAMwAwADEAMAA3ADQAZQA3AGMAYgBkADQANQB1ADAANwBkADYAYQBhADMAYwA1AGUAZQBhAGUANAAzADEANgB
           1AGQAOQA4ADcAMQAyADYAOAB1ADEAMgAxADkAYQBmADgANwBkAGIAOQBhADMAYgAzAGQAZAA1AGMANAA4ADcAZgBhAGYAYQBhAGUAZQBmAGIAOAA2ADAAMAAwADkAYgAwAGEAZgA1AGEAYwBmADUANgBjAGIAMQA5ADIAYQBmADYAYwAzADAAYwB1ADYAYwA3AGIANQAxAGEAMQB1ADMANwBjADYAMgBmA
           GQAZAAxADUAMAA0ADcAMQB1AGMAOQBkAGQAOAA5ADYAMwAwAGEAOQBjAGUANQAzADQANwA2ADkAZgAyAGYAOQA2ADgANgA2ADEAMQAxADkAYwBkADYAZQA2AGQAZAB1AGEANgB1AGMAYwA0ADYANwBjAGMAYQAyADAAMQBmADUAZgA4ADUANAA4ADMANwA2AGYAOAAxADIAYQBhAGUAYgBjADMAYQAwADY
           AOQA5AGMAZQBjADkAOQAzADkAMQA0ADIANAAyAGEAZgAwAGUAYwA2ADIANwB1AGUANwBmADQANwBmADgAMgA2ADkANQAxAGMAMwBmAGQAMQA5ADMANwAzAGEAOAA0AGQANAA2AGIAZAA3ADQAYgAwAGIAMQAwADIAZgBkADMAYQBmADEAYgBiADkAYgAzADYAMQA5ADQANQA2ADQAMQA1ADcAMwA2ADYAM
           QA0ADAAMAB1AGQANAAyADQAYQA2ADYAMwA0AGMAZQBiAGUAZQBkADUAYgAxADkANwB1AGMANgA1ADkANQAyADcAYQBmADYAYQA0ADYAMQBmAGUAOABkADEAOAAwAGMAMwBkAGUAYgB1AGEAMAAzADIAYgA1ADYAYQA5AGEAMABjADYAZAAwADQANABiAGYAZABjADMANwA0ADgAYwA1ADMAZQBjAGQAMQA
           4ADAAMQAwAGIAYQBhAGEAMAA1AGUAZgAyAGUAZAB1AGMAMwA5ADkAMQAyADAAZgBjADgANwAyAGEANwA3ADcAOQA2AGUAZQBjADkAYwAwADIANQBjAGIAOAA2AGQAOQBkAGQAMQBjAGUANAB1AGUAMABjADcAMQA3ADkANQAzAGQAOQA3AGUAYgBjAGEANgBmADMANwAxADYAMwBiAGMAYgA5ADQANQAxA
           DQANAAxAGQAMABiADIAMQA0ADEAMQA3ADUAMAA4AGEAMAA4ADUAOAB1ADcAOAB1ADYANQA2ADQAZQAwADcANAA3ADYANQA5ADYAYQA5ADcANgAyADgANwAyADIAMwAyADEAZgA1ADkAYgBkADUANgA0AGEAMgB1AGEAYwBiADUAZQAwAGEANQAxAGEAOQAwADMANQA2ADYAMQBiAGMAYQA2ADgAMQA3ADA
           AOQBjADgAMwA5AGQAOQBhADUANgA1ADIAMAB1ADMAMgBiAGIAMwA4ADMAZAA5ADQAYwAxAGUANgB1ADcANQBmAGIAOQAzADAANAAwAGMAZAA1AGQAYwA5AGEAMgA2ADcAMABjADcAOABhADQANgBiAGUAYQA4ADUAMgA='|CoNverTTo-secuReStriNG -k
           82,189,200,92,184,235,46,38,211,250,202,240,198,208,70,100,210,121,211,227,2,148,77,154,149,200,93,130,24,30,119,255) ) )) )
Decoded :
```

As seen in the above image, the script was able to identify that there is 500+ consecutive Base64 characters, and it has alerted it as suspicious. You could then simply echo the base64 encoded text and pipe it to base64 decode to see the decoded command.



```
Date     : 8/31/2017 4:56:09 AM
Log      : Powershell
EventID  : 4104
Message  : Suspicious Command Line
Results  : Long Command Line: greater than 1000 bytes
           Possible command obfuscation: 79% zeroes and ones (possible numeric or binary encoding)

Command  :  -join('1001001n1000101-1011000g100000<101000e1001110F1100101-1110111@101101<1001111i1110
           -1110100i101001<101110-1000100-1101111<1110111F1101110;1101100n1101111i1100001e1100100{1
           1110111-101110;1100111{1101001F1110100-1101000@1110101{1100010e1110101n1110011@1100101i1
           01{1100110;1100101i1110011-1110100F1100001n1110100{1101001@1101111F1101110e101111-101000
           01111<1000101;1111000;1100110-1101001-1101100<1110100;1110010F1100001<1110100{1101001@11
           0<101110-1110000e1110011g110001e100111-101001;111011F100000@1001001{1101110g1110110{1101
           000011g1110010n1100101;1100100<1110011'.splIT( '<{genF-i;@' )| FOreAcH { ([coNvErt]::toI
Decoded :
```

As hinted earlier, you can see that it also detects binary encoding based off of the large amount of 1s and 0s, something that is not typically seen in non-malicious commands.

DeepBlueHash

DeepBlueHash is a separate tool from DeepBlueCLI that parses Sysmon event logs to extract SHA256 from process creation, driver load, and image load events. I will not be going over this script today, but please refer to the following link if you wish to explore it for yourself (note, you will require a free VirusTotal API key).

Conclusion

This post covers only a snippet of DeepBlueCLI's potential. It can also detect frameworks like Metasploit and much more. For more resources on this great tool check out the following:

- https://github.com/strandjs/IntroLabs/blob/master/IntroClassFiles/Tools/IntroClass/deepbluecli/DeepBlueCLI.md
- https://github.com/sans-blue-team/DeepBlueCLI
- https://youtu.be/6sMluvfLsI8?si=aFnkEGmQIcoZ_e_P

I hope you enjoyed this post. DeepBlueCLI is an incredibly fun tool to use and I highly recommend it for those who struggle with analysing event logs using the Event Viewer or PowerShell cmdlets.