PowerShell Intro & Basics

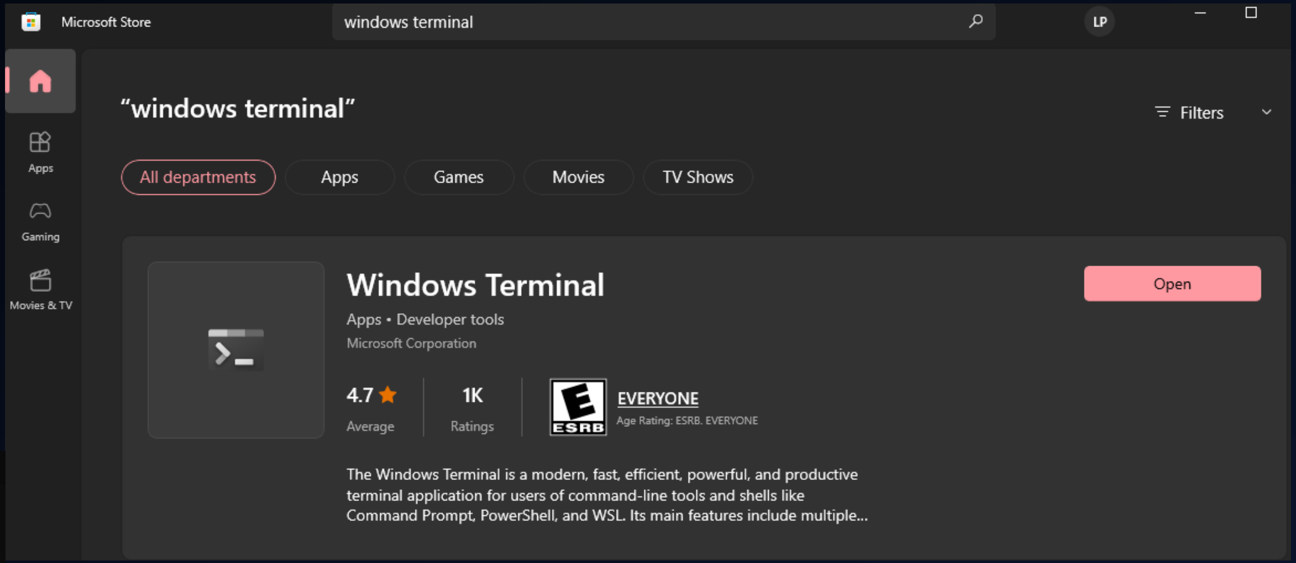
LIAM POWELL

About Me

- CFSO Secretary and CCDC Co-Captain
- Windows Server Fanatic
  - Custom internal domain with DNS, IIS, and AD integration
- Avid PowerShell Scripter
  - https://github.com/lpowell
  - Notable Projects: Vynae, Network Enum, EnumToWeb

Installing Windows Terminal

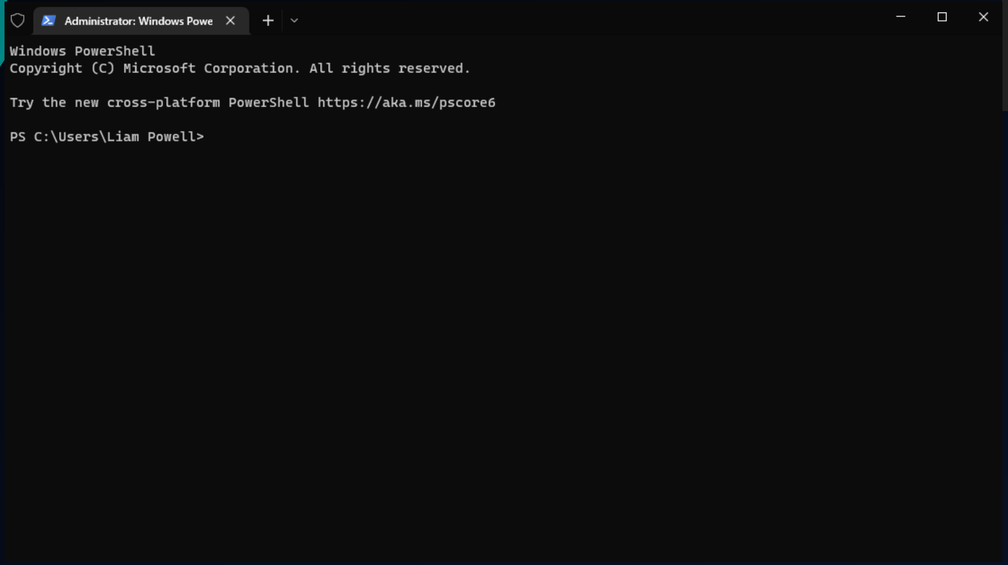
Windows Terminal is a Powerful tool that can replace your CMD and PowerShell Shells



"windows terminal"

Windows Terminal
Apps • Developer tools
Microsoft Corporation
4.7 Average
1K Ratings

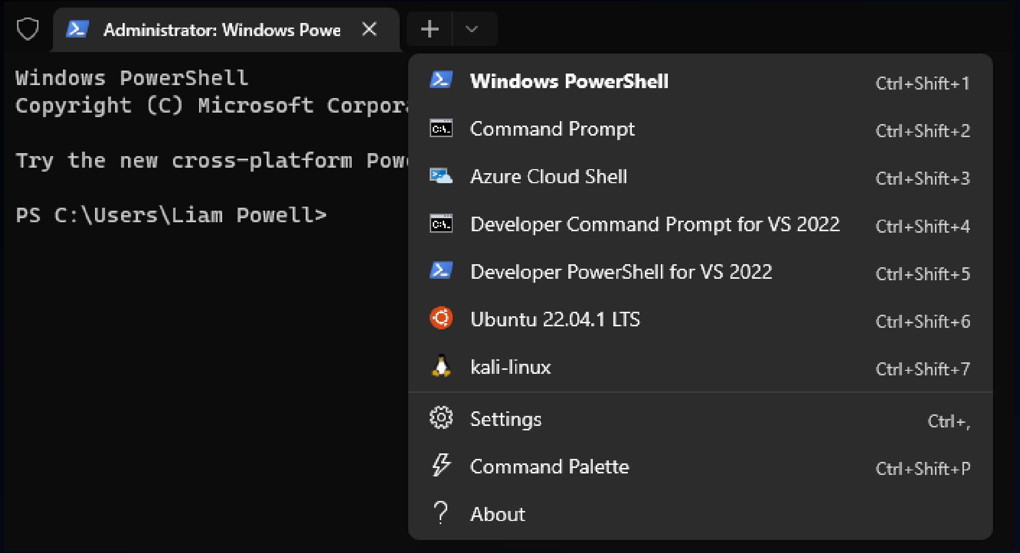
The Windows Terminal is a modern, fast, efficient, powerful, and productive terminal application for users of command-line tools and shells like Command Prompt, PowerShell, and WSL. Its main features include multiple...



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Liam Powell>
```

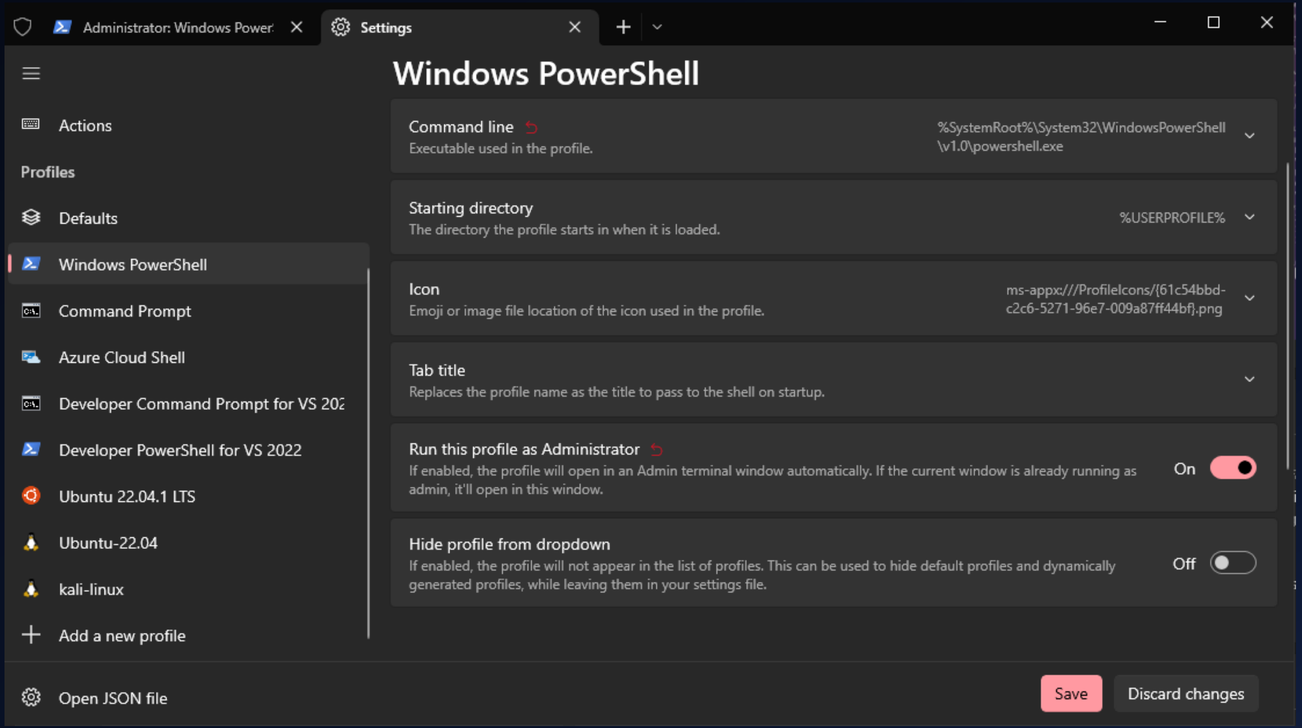
You can find it in the Microsoft Store by searching for "Windows Terminal"

Configuring Windows Terminal


Administrator: Windows Powe
Windows PowerShell Ctrl+Shift+1
Command Prompt Ctrl+Shift+2
Azure Cloud Shell Ctrl+Shift+3
Developer Command Prompt for VS 2022 Ctrl+Shift+4
Developer PowerShell for VS 2022 Ctrl+Shift+5
Ubuntu 22.04.1 LTS Ctrl+Shift+6
kali-linux Ctrl+Shift+7
Settings Ctrl+,
Command Palette Ctrl+Shift+P
About

```
Windows PowerShell
Copyright (C) Microsoft Corpor

Try the new cross-platform Pow

PS C:\Users\Liam Powell>
```

Open settings in the drop-down menu



Windows PowerShell

Command line
Executable used in the profile.
%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe

Starting directory
The directory the profile starts in when it is loaded.
%USERPROFILE%

Icon
Emoji or image file location of the icon used in the profile.
ms-appx:///ProfileIcons/{61c54bbd-c2c6-5271-96e7-009a87ff44bf}.png

Tab title
Replaces the profile name as the title to pass to the shell on startup.

Run this profile as Administrator
If enabled, the profile will open in an Admin terminal window automatically. If the current window is already running as admin, it'll open in this window.
On

Hide profile from dropdown
If enabled, the profile will not appear in the list of profiles. This can be used to hide default profiles and dynamically generated profiles, while leaving them in your settings file.
Off

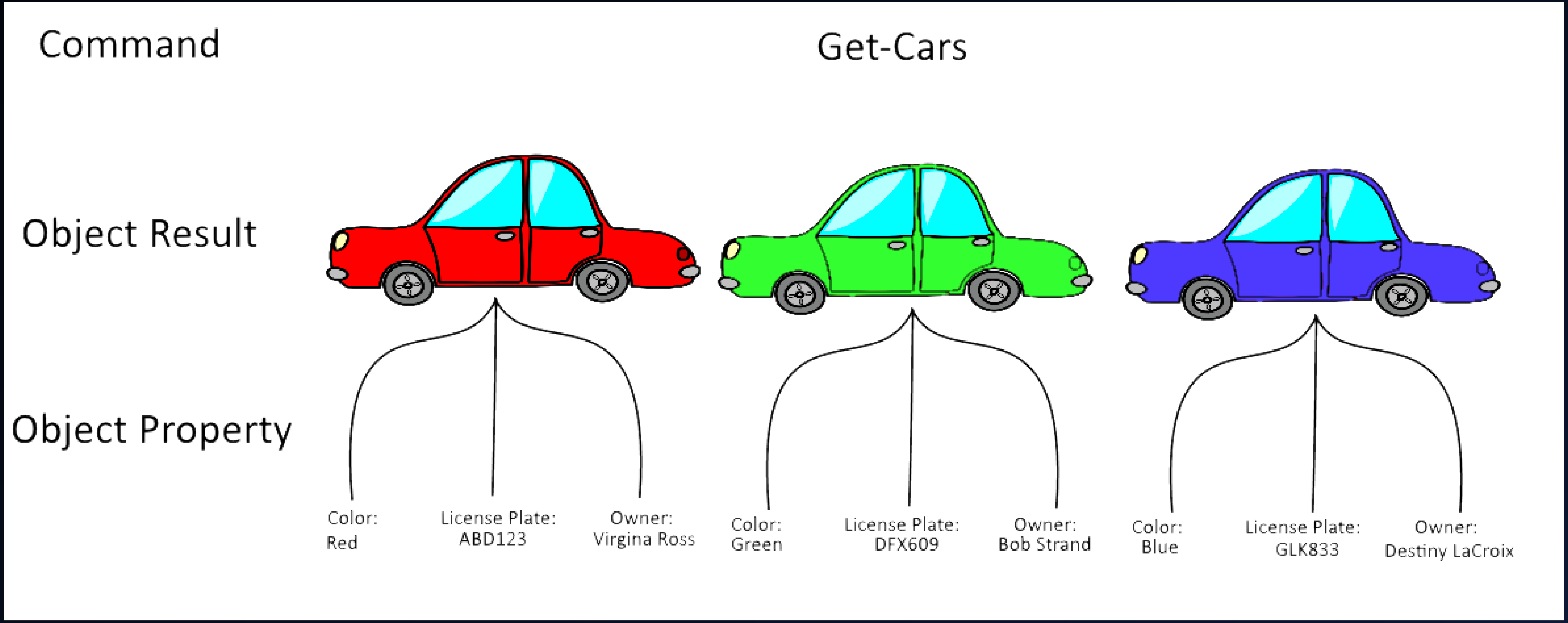
For our demos, we want to enable the “Run this profile as Administrator” setting for the PowerShell profile

What is PowerShell?

- PowerShell is an automation tool that consists of:
- A CLI Shell
  - Command-Line Interface Shell
- A Scripting Language
  - Built on the .Net Common Language Runtime (CLR)
- A management framework
  - PowerShell can deploy to AWS, VMWare, Azure, Windows, Exchange, SQL, and more

What are Objects?

- PowerShell Commands return .Net Objects



Example PowerShell command return

- Get-Car

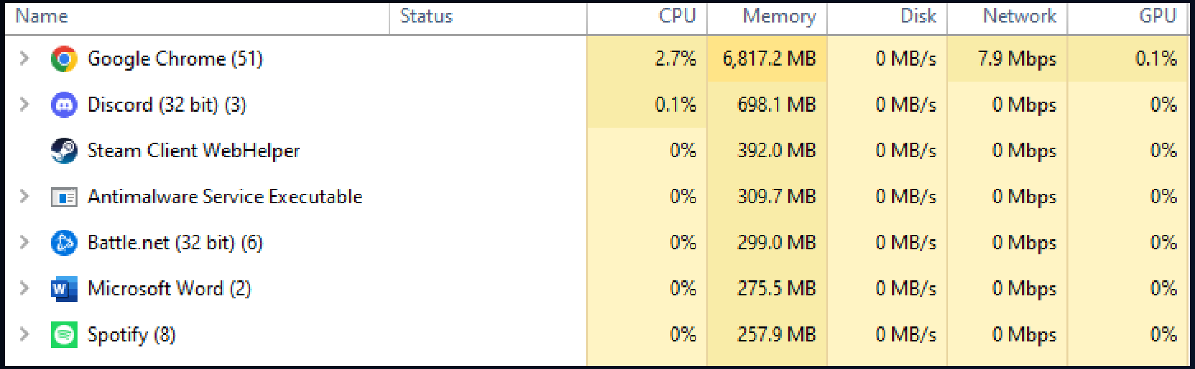
ID	Color	License Plate	Owner
001	Red	ABD123	Virginia Ross
002	Green	DFX609	Bob Strand
003	Blue	GLK833	Destiny LaCroix

- Get-Car | ? Color –eq Red

ID	Color	License Plate	Owner
001	Red	ABD123	Virginia Ross

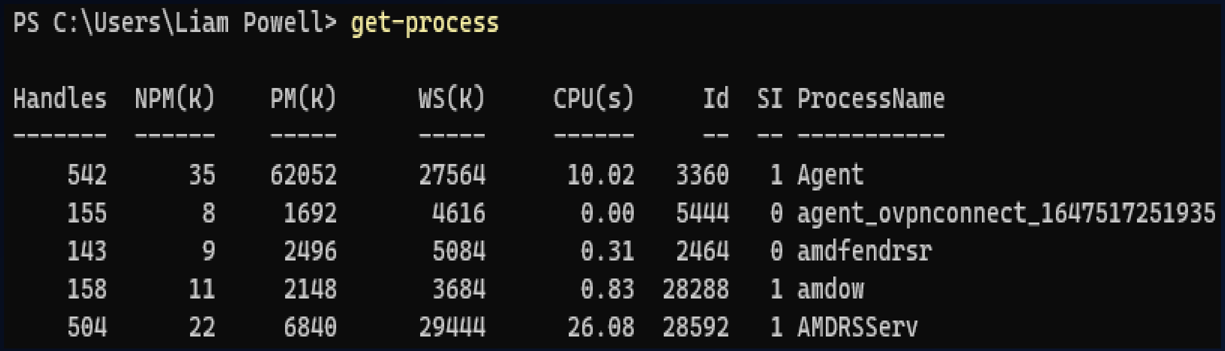
Why is PowerShell important?

- Getting Processes through Task Manager vs PowerShell
- Task Manager



Name	Status	CPU	Memory	Disk	Network	GPU
Google Chrome (51)		2.7%	6,817.2 MB	0 MB/s	7.9 Mbps	0.1%
Discord (32 bit) (3)		0.1%	698.1 MB	0 MB/s	0 Mbps	0%
Steam Client WebHelper		0%	392.0 MB	0 MB/s	0 Mbps	0%
Antimalware Service Executable		0%	309.7 MB	0 MB/s	0 Mbps	0%
Battle.net (32 bit) (6)		0%	299.0 MB	0 MB/s	0 Mbps	0%
Microsoft Word (2)		0%	275.5 MB	0 MB/s	0 Mbps	0%
Spotify (8)		0%	257.9 MB	0 MB/s	0 Mbps	0%

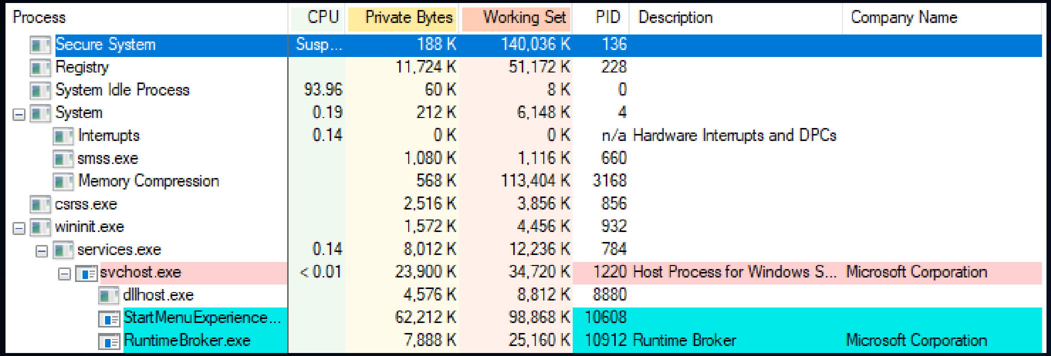
- Get-Process



```
PS C:\Users\Liam Powell> get-process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    542      35    62052      27564      10.02   3360   1 Agent
    155       8     1692       4616       0.00   5444   0 agent_ovpnconnect_1647517251935
    143       9     2496       5084       0.31   2464   0 amdfendrsr
    158      11     2148       3684       0.83  28288   1 amdow
    504      22     6840      29444      26.08  28592   1 AMDRSServ
```

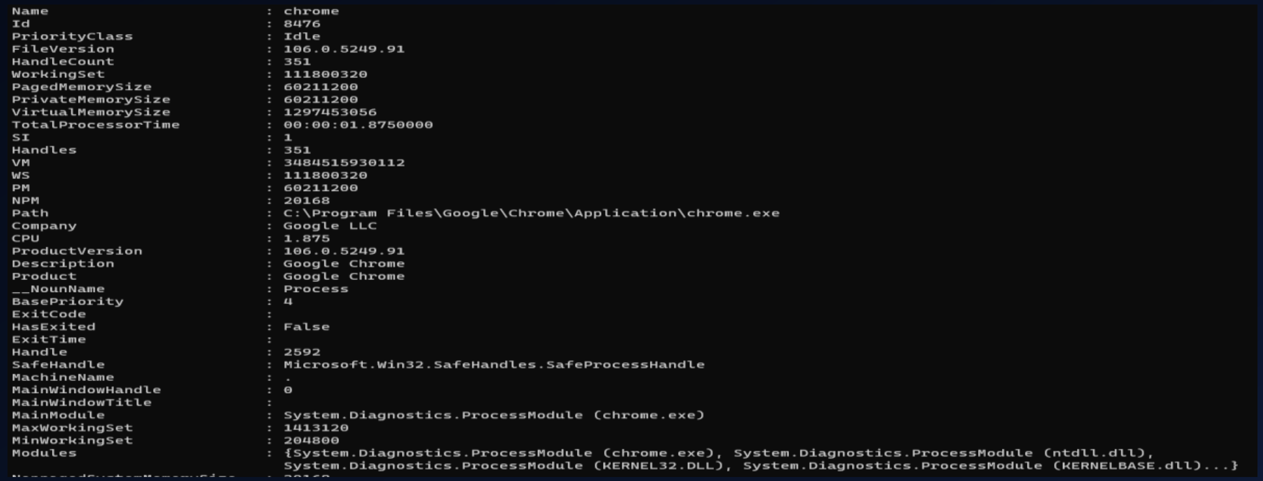
Why is PowerShell Important?

- What happens if I want to get more information?
  - Process Explorer



Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Secure System	Susp...	188 K	140,036 K	136		
Registry		11,724 K	51,172 K	228		
System Idle Process	93.96	60 K	8 K	0		
System	0.19	212 K	6,148 K	4		
Interrupts	0.14	0 K	0 K	n/a	Hardware Interrupts and DPCs	
smss.exe		1,080 K	1,116 K	660		
Memory Compression		568 K	113,404 K	3168		
csrss.exe		2,516 K	3,856 K	856		
wininit.exe		1,572 K	4,456 K	932		
services.exe	0.14	8,012 K	12,236 K	784		
svchost.exe	< 0.01	23,900 K	34,720 K	1220	Host Process for Windows S...	Microsoft Corporation
dllhost.exe		4,576 K	8,812 K	8880		
StartMenuExperience...		62,212 K	98,868 K	10608		
RuntimeBroker.exe		7,888 K	25,160 K	10912	Runtime Broker	Microsoft Corporation

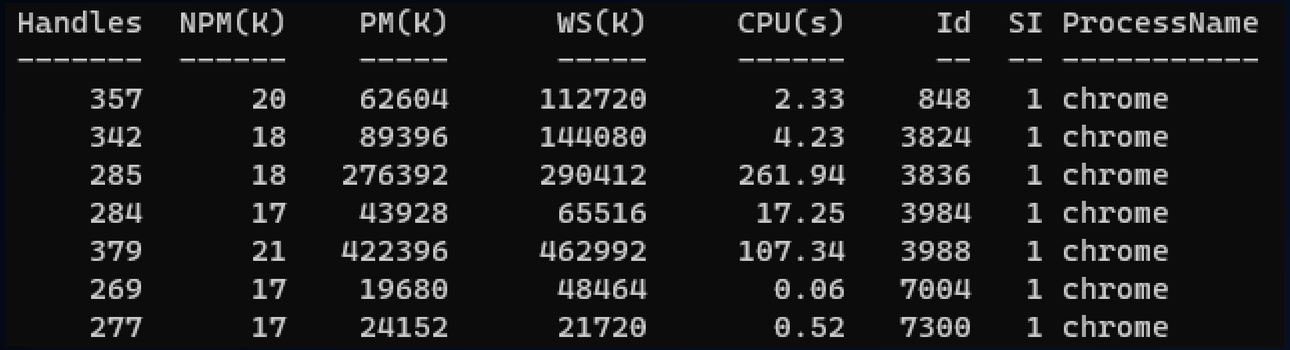
  - Get-Process | ft *



```
Name                       : chrome
Id                         : 8476
PriorityClass              : Idle
FileVersion                : 106.0.5249.91
HandleCount                : 351
WorkingSet                 : 111800320
PagedMemorySize            : 60211200
PrivateMemorySize          : 60211200
VirtualMemorySize          : 1297453056
TotalProcessorTime         : 00:00:01.8750000
SI                         : 1
Handles                    : 351
VM                         : 3484515930112
WS                         : 111800320
PM                         : 60211200
NPM                        : 20168
Path                       : C:\Program Files\Google\Chrome\Application\chrome.exe
Company                    : Google LLC
CPU                        : 1.875
ProductVersion             : 106.0.5249.91
Description                : Google Chrome
Product                    : Google Chrome
__NounName                 : Process
BasePriority               : 4
ExitCode                   :
HasExited                  : False
ExitTime                   :
Handle                     : 2592
SafeHandle                 : Microsoft.Win32.SafeHandles.SafeProcessHandle
MachineName                : .
MainWindowHandle           : 0
MainWindowTitle            :
MainModule                 : System.Diagnostics.ProcessModule (chrome.exe)
MaxWorkingSet              : 1413120
MinWorkingSet              : 204800
Modules                    : {System.Diagnostics.ProcessModule (chrome.exe), System.Diagnostics.ProcessModule (ntdll.dll),
                             System.Diagnostics.ProcessModule (KERNEL32.DLL), System.Diagnostics.ProcessModule (KERNELBASE.dll)...}
```

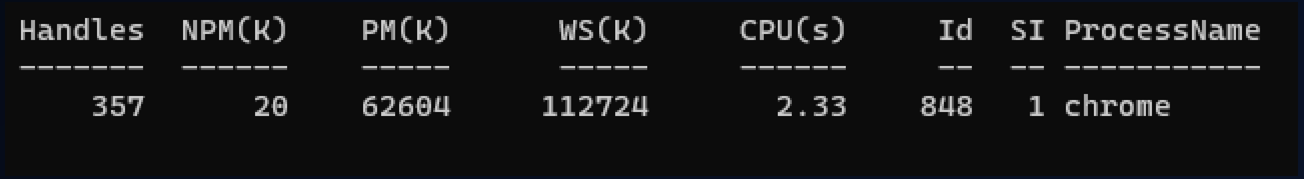
Why is PowerShell Important?

- What if I want a specific process and not all the others?
- Get-Process | ? Name –match chrome



```
Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    357      20    62604     112720       2.33    848   1 chrome
    342      18    89396     144080       4.23   3824   1 chrome
    285      18   276392     290412     261.94   3836   1 chrome
    284      17    43928      65516      17.25   3984   1 chrome
    379      21   422396     462992     107.34   3988   1 chrome
    269      17    19680      48464       0.06   7004   1 chrome
    277      17    24152      21720       0.52   7300   1 chrome
```

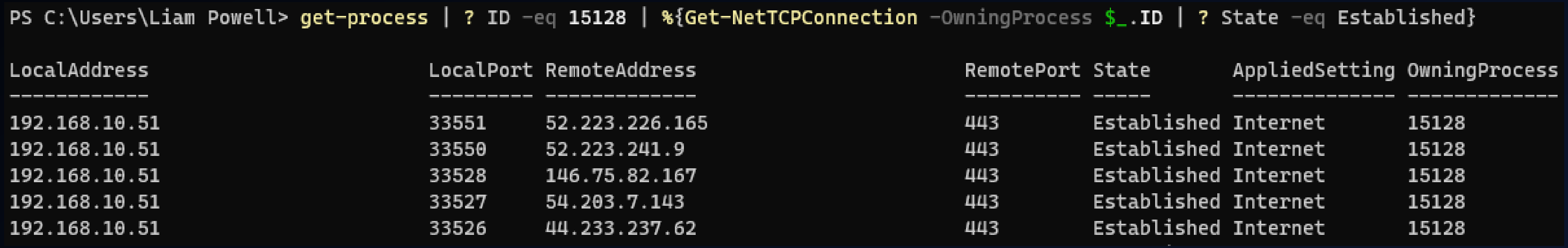
- Get-Process | ? ID –eq 848



```
Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    357      20    62604     112724       2.33    848   1 chrome
```

Why is PowerShell Important?

- How can I use that process information in PowerShell?
- Use the Process ID to get its Network Connections



```
PS C:\Users\Liam Powell> get-process | ? ID -eq 15128 | %{Get-NetTCPConnection -OwningProcess $_.ID | ? State -eq Established}

LocalAddress                        LocalPort RemoteAddress                        RemotePort State       AppliedSetting OwningProcess
------------                        --------- -------------                        ---------- -----       -------------- -------------
192.168.10.51                       33551     52.223.226.165                       443        Established Internet       15128
192.168.10.51                       33550     52.223.241.9                         443        Established Internet       15128
192.168.10.51                       33528     146.75.82.167                        443        Established Internet       15128
192.168.10.51                       33527     54.203.7.143                         443        Established Internet       15128
192.168.10.51                       33526     44.233.237.62                        443        Established Internet       15128
```

Why is PowerShell Important?

- GUI tasks can be easily done with more detail in PowerShell
- Tasks can be automated into scripts
- Management tools like PowerShell Remoting make PowerShell solutions scalable across an organization
- Its cool to use the CLI!

PowerShell Basics - Aliases

- Aliases
  - Previous examples showed the use of %, ?, and | symbols in the CLI
  - | is a pipe operator, and it passes the output of the command to the next command
      - Get-Process | Select Name
    - This passes the processes through a filter that will only output the name of the processes
  - ? is an alias for Where-Object. Where-Object lets you filter a commands output.
      - Get-Process | ? Name –match chrome
    - This puts out all process that match the name chrome
  - % is an alias for ForEach-Object
      - This lets you operate on each object passed to it.
        - Get-Process | %{Get-NetTCPConnection –OwningProcess $_.ID}
      - This will get all processes and for each one, it will get the active network connections

PowerShell Basics – Automatic Variables

- Automatic Variables are variables that are built-in to PowerShell
- $_ is the variable for the current object
    - Get-Process | %{echo $_.Name}
  - This will echo the names of each command
  - $_.Property will let us access the property of the object in action
- $HOME contains the user's home folder
- $env:X contains the specified environment variable (dir env: to list them)
  - $env:USERPROFILE, $env:TEMP, $env:ProgramFiles, etc…
- $? Contains the status of the previously executed command (true/false)

PowerShell Basics – ForEach-Object

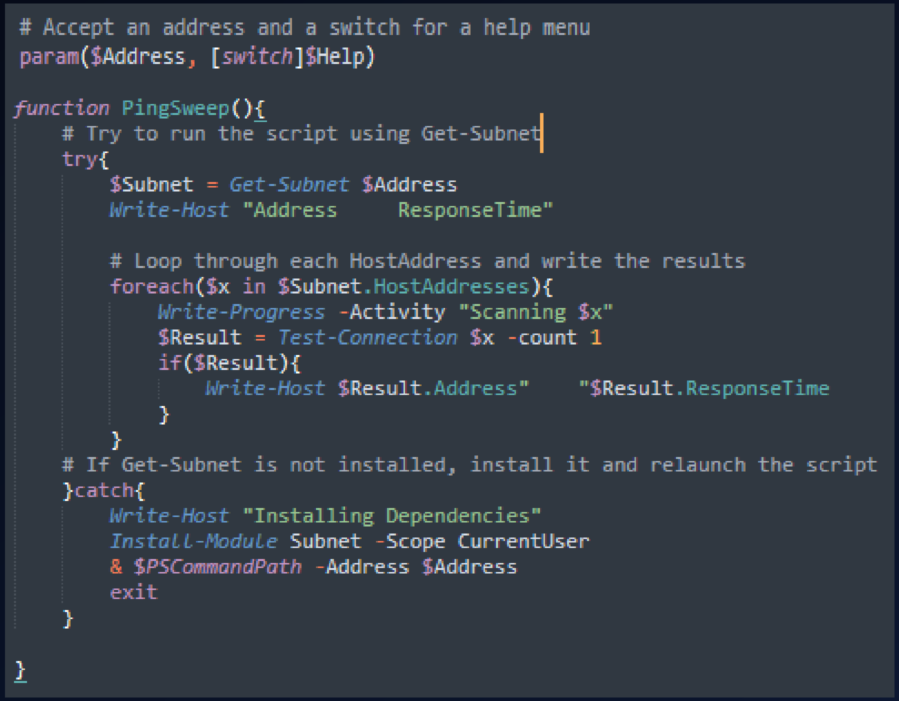
- ForEach-Object can be used to make your CLI commands more powerful
  - Get-Process | %{echo $_.Name; $t = Get-NetTCPConnection –OwningProcess $_.ID -ErrorAction SilentlyContinue; if($t){echo $t}else{Echo “No Connections”}; write-host;}
- The script block in the foreach loop contains multiple lines separated by semi-colons (;)
- Using semi-colons and the foreach loop, you can write small one-line commands to do complex actions

Example Scripts – PingSweep

- 1..A | % {test-connection 1.1.1.$_ -count 1 –erroraction 0}
  - For every X in 1-A, test-connection 1.1.1.X
  - Limited to changing 1 octet, need to know the subnet before hand, etc…
- As a script, we can add more logic to the function
  - Detect subnet and find all host addresses
  - Add a progress display to track scans
  - Automatically install dependencies

Example Scripts – PingSweep

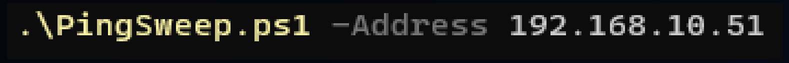
- Use param($param1, $param2) to create parameters
- Try/Catch blocks to test code
- Foreach loops in script form
- Write-Host vs Echo
    - Write-Host only writes to the console



```powershell
# Accept an address and a switch for a help menu
param($Address, [switch]$Help)

function PingSweep(){
    # Try to run the script using Get-Subnet
    try{
        $Subnet = Get-Subnet $Address
        Write-Host "Address     ResponseTime"

        # Loop through each HostAddress and write the results
        foreach($x in $Subnet.HostAddresses){
            Write-Progress -Activity "Scanning $x"
            $Result = Test-Connection $x -count 1
            if($Result){
                Write-Host $Result.Address"    "$Result.ResponseTime
            }
        }
    # If Get-Subnet is not installed, install it and relaunch the script
    }catch{
        Write-Host "Installing Dependencies"
        Install-Module Subnet -Scope CurrentUser
        & $PSCommandPath -Address $Address
        exit
    }

}
```

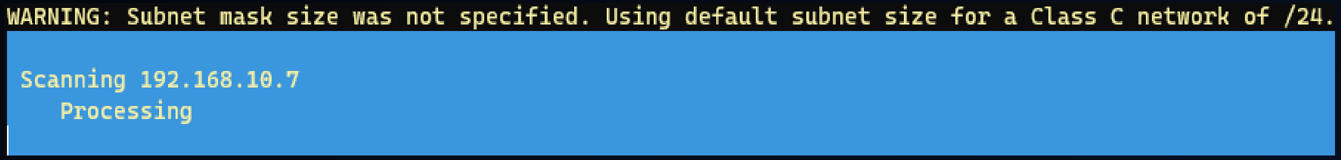
Example Scripts – PingSweep

- Our script accepting the $Address parameter



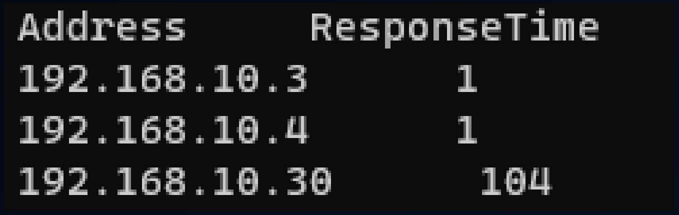
```
.\PingSweep.ps1 -Address 192.168.10.51
```

- The Write-Progress marker



```
WARNING: Subnet mask size was not specified. Using default subnet size for a Class C network of /24.

Scanning 192.168.10.7
   Processing
```

- Output of the script

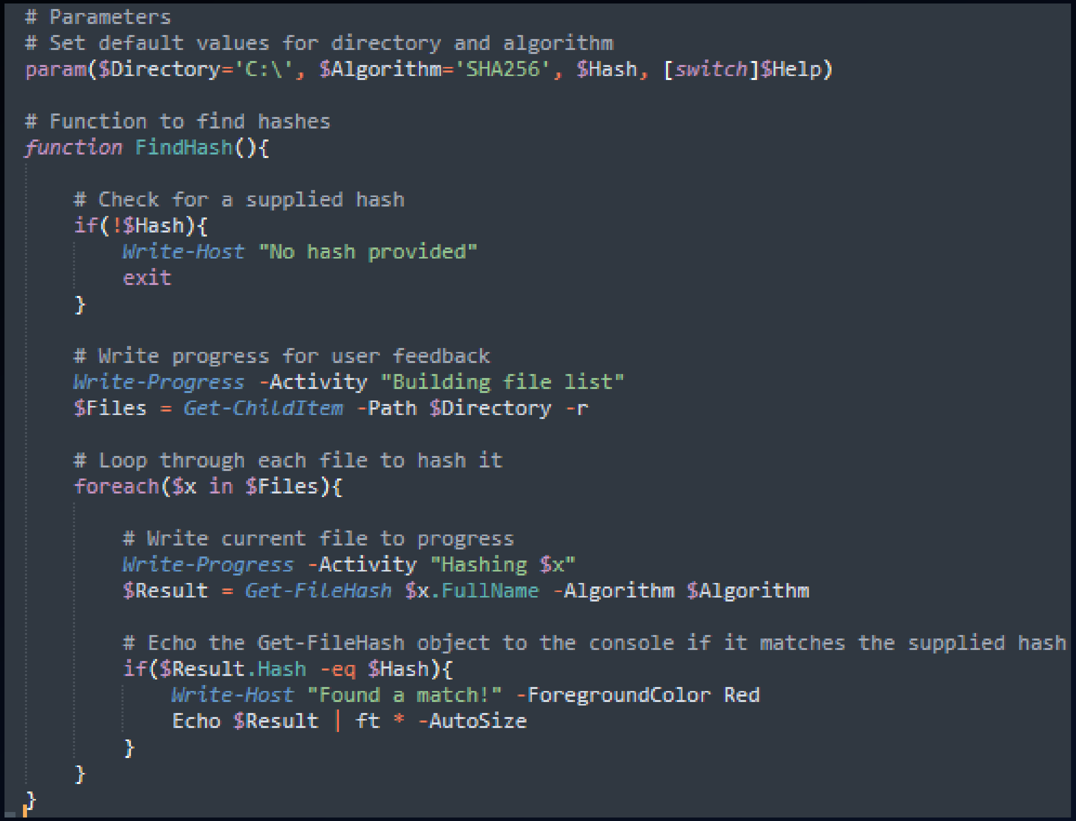


```
Address         ResponseTime
192.168.10.3         1
192.168.10.4         1
192.168.10.30         104
```

Example Scripts – FindHash

- Get-ChildItem -r| %{$h=Get-FileHash $_.FullName –ErrorAction 0;if($h.Hash -eq “xxxx"){echo $h;}}
  - Get-ChildItem –r is a recursive search from the location it’s launched
    - Like a recursive ls
  - Get-FileHash will hash a file you pass to it using SHA256 by default
  - Our conditional logic will test each hashed file with the supplied hash
- As a script, we can add algorithm and root directory parameters

Example Scripts – FindHash

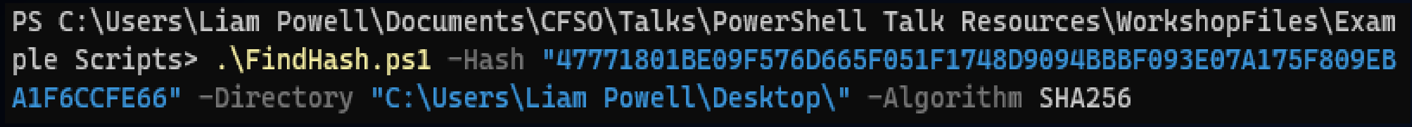


```
# Parameters
# Set default values for directory and algorithm
param($Directory='C:\', $Algorithm='SHA256', $Hash, [switch]$Help)

# Function to find hashes
function FindHash(){

    # Check for a supplied hash
    if(!$Hash){
        Write-Host "No hash provided"
        exit
    }

    # Write progress for user feedback
    Write-Progress -Activity "Building file list"
    $Files = Get-ChildItem -Path $Directory -r

    # Loop through each file to hash it
    foreach($x in $Files){

        # Write current file to progress
        Write-Progress -Activity "Hashing $x"
        $Result = Get-FileHash $x.FullName -Algorithm $Algorithm

        # Echo the Get-FileHash object to the console if it matches the supplied hash
        if($Result.Hash -eq $Hash){
            Write-Host "Found a match!" -ForegroundColor Red
            Echo $Result | ft * -AutoSize
        }
    }
}
```

- Parameters Directory, Algorithm, Hash, Help
- Set default values for parameters
- Check for supplied hash value
  - Report to the user if they do not provide one
- Write the current file to the progress bar
- Echo the File-Hash object to the console on a match

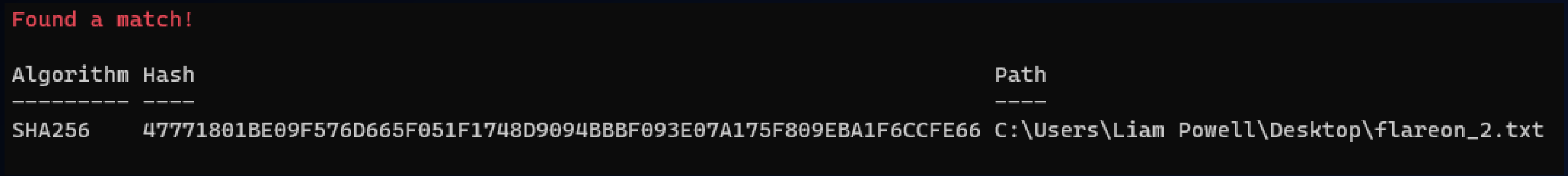
Example Scripts – FindHash

- CLI invoke with parameters



```
PS C:\Users\Liam Powell\Documents\CFSO\Talks\PowerShell Talk Resources\WorkshopFiles\Exam
ple Scripts> .\FindHash.ps1 -Hash "47771801BE09F576D665F051F1748D9094BBBF093E07A175F809EB
A1F6CCFE66" -Directory "C:\Users\Liam Powell\Desktop\" -Algorithm SHA256
```

- Output of the script



```
Found a match!

Algorithm Hash                                                             Path
--------- ----                                                             ----
SHA256    47771801BE09F576D665F051F1748D9094BBBF093E07A175F809EBA1F6CCFE66 C:\Users\Liam Powell\Desktop\flareon_2.txt
```

Demo Time!

- Process Explorer
  - Time: 10 Minutes
  - /Resources/Process-Explorer/Start.ps1
- Network Explorer
  - Time: 10 Minutes
  - /Resources/Network-Explorer/Start.ps1
- Log Parser
  - Time: 20 Minutes
  - /Resources/Log-Parse/Start.ps1
- Schedule Task Creation
  - Time: 20 Minutes
  - /Resources/Task-Creation/Start.ps1