Blue Team Level 1 Certification (Standard)

Introduction to BTL1

- ~~Welcome to Blue Team Level 1!~~ — 4 Topics
- ~~Lab and Forum Access~~

SECURITY FUNDAMENTALS DOMAIN

- ~~Introduction to Security Fundamentals~~ — 1 Topic
- ~~Soft Skills~~ — 7 Topics
- ~~Security Controls~~ — 5 Topics | 1 Quiz
- ~~Networking 101~~ — 6 Topics | 1 Quiz
- ~~Management Principles~~ — 4 Topics | 1 Quiz

PHISHING ANALYSIS DOMAIN

- ~~PA1) Introduction to Emails and Phishing~~ — 7 Topics | 1 Quiz
- ~~PA2) Types of Phishing Emails~~ — 10 Topics | 2 Quizzes
- ~~PA3) Tactics and Techniques Used~~ — 12 Topics | 2 Quizzes
- ~~PA4) Investigating a Phishing Email~~ — 8 Topics | 2 Quizzes
- ~~PA5) Analysing URLs, Attachments, and Artifacts~~ — 8 Topics | 1 Quiz
- PA6) Taking Defensive Actions — 12 Topics | 1 Quiz
- PA7) Report Writing — 7 Topics | 1 Quiz
- PA8) Phishing Response Challenge — 3 Topics | 1 Quiz

THREAT INTELLIGENCE DOMAIN

- TI1) Introduction to Threat Intelligence — 7 Topics
- TI2) Threat Actors & APTs — 6 Topics | 2 Quizzes
- TI3) Operational Threat Intelligence — 7 Topics | 1 Quiz
- TI4) Tactical Threat Intelligence — 7 Topics | 1 Quiz
- TI5) Strategic Threat Intelligence — 5 Topics | 1 Quiz
- TI6) Malware and Global Campaigns — 6 Topics | 1 Quiz

DIGITAL FORENSICS DOMAIN

- DF1) Introduction to Digital Forensics — 5 Topics
- DF2) Forensics Fundamentals — 10 Topics | 5 Quizzes

CMD and PowerShell For Incident Response

This lesson will introduce you to some useful CMD and PowerShell commands to assist with incident response. Some of the actions we can take include:

- **List networking information** (to gather information such as IP address, MAC, and more)
- **Viewing open and listening ports** (to detect backdoors and beaconing)
- **Viewing running processes and their related executable files** (to detect malware or backdoors)
- **List all users and admins on the local system** (to identify unusual accounts)
- **List programs that launch at system boot** (to detect malicious files)
- **List services and detailed information** (to identify malicious services)
- **And more!**

Why do we use CMD?

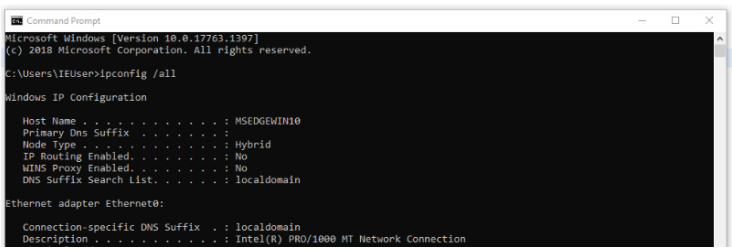
Although using it requires the memorization of many different commands, it can allow us to complete tasks faster than interacting with the Windows graphical user interface, and also allows us to automate tasks using batch scripts. This can help in incident response scenarios, as we can query the system for information about almost anything, from users to running processes!

CMD For Incident Response

Below we're going to cover a number of commands that may be useful for security investigations and incident response. We will also include examples so you can see what the input could look like, and how to interpret the information that is printed to the terminal. These commands should be ran as an administrator to function correctly.

```
ipconfig /all
```

Description: This command will get network configuration information from the local system, including assigned IP address and the device's MAC address.
Example: In this example we can see that we have the host name "**MSEDGEWIN10**", a MAC address of "**00-0C-29-AA-02-FA**", and IPv4 address of "**192.168.125.129**" . We can also see that the DNS server being used for name resolution is currently "**102.168.125.2**".



DF3) Digital Evidence Collection
8 Topics | 1 Quiz

DF4) Windows Investigations
3 Topics | 3 Quizzes

DF5) Linux Investigations
4 Topics | 2 Quizzes

DF6) Volatility
3 Topics | 1 Quiz

DF7) Autopsy
4 Topics | 1 Quiz

SECURITY INFORMATION AND EVENT MANAGEMENT DOMAIN

SI1) Introduction to SIEM
7 Topics | 1 Quiz

SI2) Logging
6 Topics | 2 Quizzes

SI3) Aggregation
2 Topics | 1 Quiz

SI4) Correlation
6 Topics | 1 Quiz

SI5) Using Splunk
5 Topics | 2 Quizzes

INCIDENT RESPONSE DOMAIN

IR1) Introduction to Incident Response
8 Topics | 1 Quiz

IR2) Preparation Phase
10 Topics | 2 Quizzes

IR3) Detection and Analysis Phase
7 Topics | 4 Quizzes

Section Introduction, Detection & Analysis

Common Events & Incidents

Using Baselines & Behavior Profiles

Introduction to Wireshark (GUI)

Introduction to Wireshark (Analysis)

Lab) Network Traffic Analysis

YARA Rules For Detection

Legacy Activity) Threat Hunting With YARA

CMD and PowerShell For Incident Response

Lab) CMD and PowerShell

Activity) End of Section Review, Detection & Analysis

IR4) Containment, Eradication, and Recovery Phase
5 Topics | 1 Quiz

IR5) Lessons Learned and Reporting
7 Topics

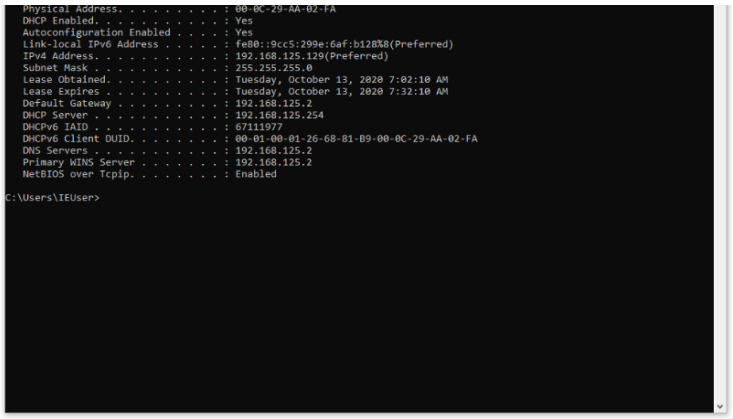
IR6) MITRE ATT&CK
13 Topics | 2 Quizzes

BTL1 EXAM

Exam Preparation

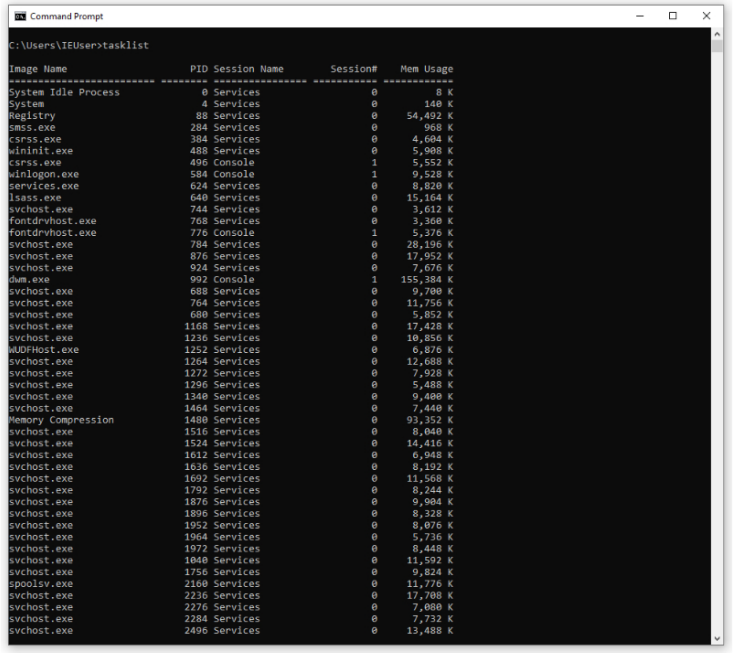
Using RDP and SSH

How to Start Your Exam



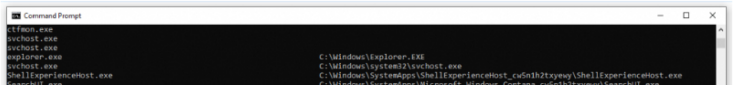

```
tasklist
```

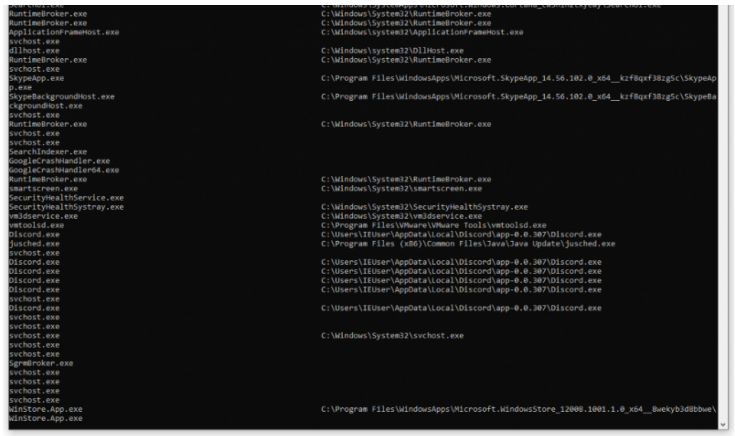
Description: This command will check running processes and programs and print a list to the terminal.
Example: In the below screenshot we are presented with a list of running processes, their process identifiers (PIDs), and the memory usage in the final column. This can be helpful to identify processes that are running in the background and how much system resources they are consuming. This can be a good way to identify malware such as crypto miners which will work silently, but consume a lot of system memory to work.




```
wmic process get description, executablepath
```

Description: This command will display running processes and the associated binary file that was executed to create the process.
Example: As the description states we are able to view running processes and the executable file that initiated them. Look just below halfway down the list and you'll find Discord.exe on the left-hand side (process name). To the right on the same row we can see that Discord was launched from C:\Users\IEUser\AppData\Local\Discord\app-0.0.307\Discord.exe – we now have the full file path! We can use this command to identify unusual process names and identify where the executable file is so we can analyze it. Process that are running out of unusual locations such as /tmp/ and /Downloads/ are definitely worth investigating further!

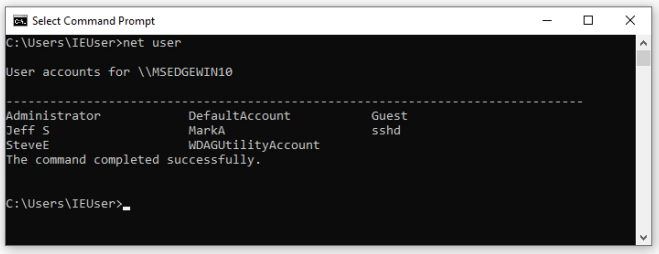




```
net user
```

Description: This command will print a list of all system users to the terminal.
Example: Using this cmd command we have printed all local system users, regardless of usergroup, to the terminal, in this case we can see:

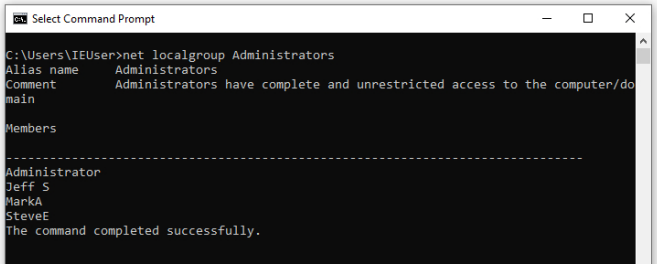
- Administrator
- DefaultAccount
- Guest
- Jeff S
- MarkA
- sshd
- SteveE
- WDAGUtilityAccount




```
net localgroup administrators
```

Description: This command will list all users that are in the administrators user group.
Example: In the below screenshot we can see all administrator accounts on this system. In this case it is the following:

- Administrator
- Jeff S
- MarkA
- SteveE



```
C:\Users\IEUser>
```

We can replace "administrators" with any local group that we want to enumerate. To see a list of all groups, use the command **`net localgroup`**. If you want to search for users in a group that includes spaces, you'll need to run your commands like this: **`net localgroup "Remote Desktop Users"`**.

```
sc query | more
```

Description: This command will list all services and detailed information about each one.

```
netstat -ab
```

Description: This command will list open ports on a system, which could show the presence of a backdoor.

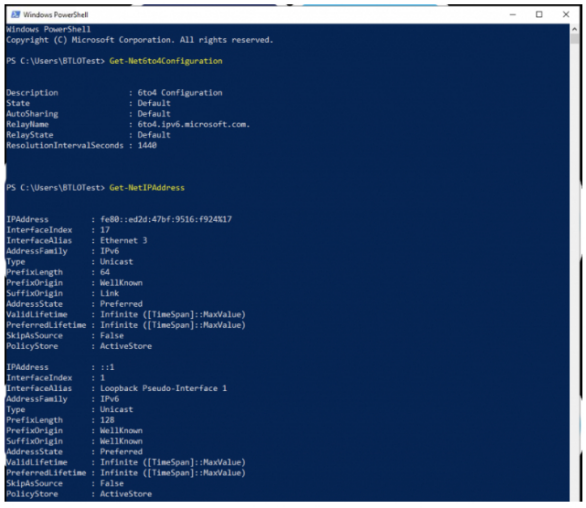
Why do we use PowerShell?

PowerShell is amazing, and chances are you'll use it a lot while working in the security industry. We can automate complex tasks, use it for offensive security purposes, or during security investigations to get more information about a user or system. For incident response we can use it similarly to CMD, but we can often retrieve much more information.

PowerShell For Incident Response

`Get-NetIPConfiguration` and **`Get-NetIPAddress`**

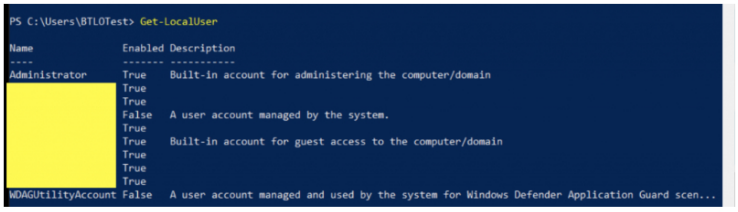
Description: Similar to ifconfig in CMD, we can use the two above commands to get network-related information from the system.
Example:



```
Get-LocalUser
```

Description: Using the above command we can list all local users on the system.

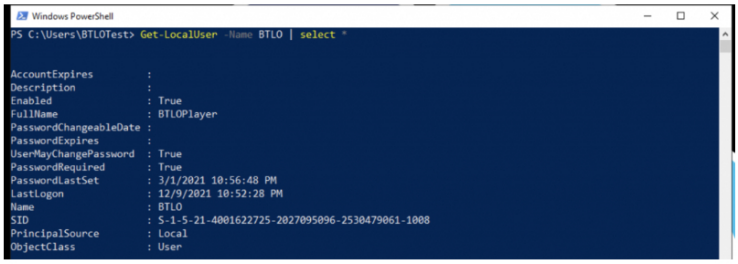
Example:




```
Get-LocalUser -Name BTLO | select *
```

Description: We can provide a specific user to the command to only get information about them. Piping (|) the results to a "select" with a wildcard (*) will give us all of the properties for the command, providing us with valuable information about the account. This can be extremely useful for us as incident responders, especially when we find local accounts that do not expire or have passwords that don't expire.

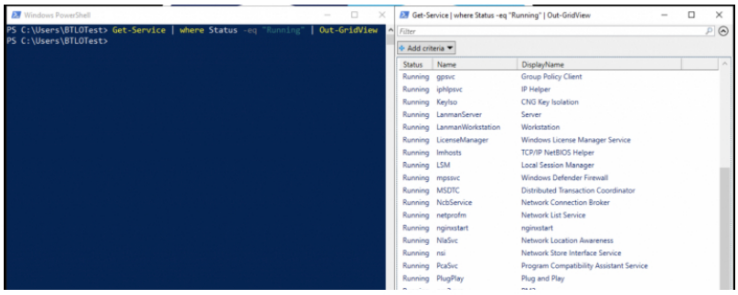
Example:




```
Get-Service | Where Status -eq "Running" | Out-GridView
```

Description: The above command let's us quickly identify running services on the system. By piping (|) the command to Out-GridView, we are telling PowerShell to show us the results in a nice windows, which is much easier to work with than outputting the results to the PowerShell window.

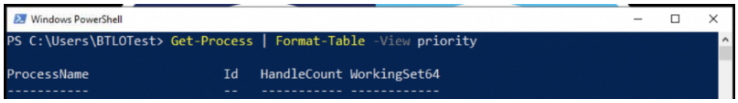
Example:




```
Get-Process | Format-Table View priority
```

Description: Another great command is the ability to group running processes by their priority value. Using the above command we can see the process name, the process ID (PID), and other information, where different priority ratings are grouped into tables.

Example:



```
amazon-ssm-agent          2440           148     13619200
ApplicationFrameHost      5688           264     25161728
cmd                       3336            79      4214784
cmd                       5896            75      4218880
conhost                   2968           152     12800000
conhost                   3508           120     10940416
conhost                   3744           142     12541952


   PriorityClass: Normal

ProcessName                Id   HandleCount WorkingSet64
-----------                --   ----------- ------------
conhost                  4052           259     20770816
conhost                  5100           151     12779520
conhost                  6672           251     20918272
csrss                     396           541      5677056
csrss                     472           324      5713920
csrss                    6544           315     10792960
csrss                    7532           296     10838016
ctfmon                   2008           365     15462400
ctfmon                   5492           387     15527936
ctfmon                   7712           369     15912960
```

```
Get-Process -Id 'idhere' | Select *
```

Description: We can collect specific information from a service by including the name in the command (-Name 'namehere') or the Id, as shown above and below. Piping to Select * provides us with all the properties.

Example:

```
PS C:\Users\BTLOTest> Get-Process -Name 'cmd'

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
     79       5     2664       4116              3336   0 cmd
     75       5     2368       4120              5896   2 cmd


PS C:\Users\BTLOTest> Get-Process -Id '5896' | Select *


Name                       : cmd
Id                         : 5896
PriorityClass              :
FileVersion                :
HandleCount                : 75
WorkingSet                 : 4218880
PagedMemorySize            : 2424832
PrivateMemorySize          : 2424832
VirtualMemorySize          : 58249216
TotalProcessorTime         :
SI                         : 2
Handles                    : 75
VM                         : 2203376472064
WS                         : 4218880
PM                         : 2424832
NPM                        : 5064
Path                       :
Company                    :
CPU                        :
ProductVersion             :
Description                :
Product                    :
```

```
Get-ScheduledTask
```

Description: Similar to Services, Scheduled Tasks are often abused and utilized a common persistence technique. With the above command we can list tasks that are set to run after certain conditions are met.

Example:

```
Windows PowerShell
PS C:\Users\BTLOTest\Desktop> Get-ScheduledTask

TaskPath                                       TaskName                          State
--------                                       --------                          -----
[redacted]
\Microsoft\Windows\AppID\                      VerifiedPublisherCertStoreCheck   Disabled
\Microsoft\Windows\Application Experience\     Microsoft Compatibility Appraiser Ready
\Microsoft\Windows\Application Experience\     ProgramDataUpdater                Ready
\Microsoft\Windows\Application Experience\     StartupAppTask                    Ready
\Microsoft\Windows\ApplicationData\            appuriverifierdaily               Ready
\Microsoft\Windows\ApplicationData\            appuriverifierinstall             Ready
\Microsoft\Windows\ApplicationData\            CleanupTemporaryState             Ready
\Microsoft\Windows\ApplicationData\            DsSvcCleanup                      Ready
\Microsoft\Windows\AppxDeploymentClient\       Pre-staged app cleanup            Disabled
\Microsoft\Windows\Autochk\                    Proxy                             Ready
\Microsoft\Windows\BitLocker\                  BitLocker Encrypt All Drives      Ready
\Microsoft\Windows\BitLocker\                  BitLocker MDM policy Refresh      Ready
\Microsoft\Windows\Bluetooth\                  UninstallDeviceTask               Disabled
```

```
Get-ScheduledTask -TaskName 'PutANameHere' | Select *
```

Description: We can dig deeper by specifying the task we're interested in, and retrieving all properties for it.

Quizzes

Lab) CMD and PowerShell

Previous Topic

Back to Lesson