Docker Container Situational Awareness

Upon gaining a shell on a web server, the initial step involves performing situational awareness to determine if it's a containerized environment. Identifying a container impacts the attack surface significantly, as containers typically reside in isolated network segments with access to internal services not exposed externally.

Identifying Containerization Indicators

Several indicators help confirm if you're operating within a Docker container:

- **Process Count**: Containers, by their isolated nature, often run very few processes compared to a virtual machine. The ps aux command can be used to print running processes and observe this.

```
root@ip-10-10-197-180:~# docker exec -it 966650153f23 /bin/sh
/ # ps aux
PID   USER     COMMAND
    1 root     /usr/sbin/uhttpd -f -p 80 -h /www .
    7 root     /bin/sh
   12 root     ps aux
/ #
```

- **.dockerenv file**: A .dockerenv file is present in the root (/) directory of a container, even if no environment variables are explicitly provided from the host operating system. This file is a key indicator of a containerized environment.

```
/ # cd /&& ls -lah
total 64
drwxr-xr-x    1 root     root        4.0K May 16 00:29 .
drwxr-xr-x    1 root     root        4.0K May 16 00:29 ..
-rw-------    1 root     root          22 May 16 00:31 .ash_history
-rwxr-xr-x    1 root     root           0 Jul 22  2022 .dockerenv
drwxrwxr-x    2 root     root        4.0K May 23  2014 bin
drwxr-xr-x    5 root     root         340 May 16 00:05 dev
drwxr-xr-x    1 root     root        4.0K Jul 22  2022 etc
drwxrwxr-x    4 root     root        4.0K May 23  2014 home
drwxr-xr-x    1 root     root        4.0K Jun  7  2014 lib
lrwxrwxrwx    1 root     root           3 May 23  2014 lib64 -> lib
lrwxrwxrwx    1 root     root          11 May 23  2014 linuxrc -> bin/busybox
drwxrwxr-x    2 root     root        4.0K Feb 27  2014 media
drwxrwxr-x    2 root     root        4.0K Feb 27  2014 mnt
drwxrwxr-x    2 root     root        4.0K Feb 27  2014 opt
dr-xr-xr-x  291 root     root           0 May 16 00:05 proc
drwx------    2 root     root        4.0K Feb 27  2014 root
lrwxrwxrwx    1 root     root           3 Feb 27  2014 run -> tmp
drwxrwxr-x    2 root     root        4.0K May 23  2014 sbin
dr-xr-xr-x   13 root     root           0 May 16 00:05 sys
drwxrwxrwt    1 root     root        4.0K Jun  7  2014 tmp
drwxr-xr-x    1 root     root        4.0K Jul 22  2022 usr
drwxrwxr-x    1 root     root        4.0K Jun  7  2014 var
drwxr-xr-x    8 root     root        4.0K Jul 22  2022 www
/ #
```

- **cgroup file**: The cgroup file, located at /proc/1/cgroup, can contain paths that include the word "docker," indicating the use of containerization software like Docker. Examination of this file can confirm that the web server process operates within a confined set of namespaces, contributing to the container's isolation.

```
File  Edit  View  Search  Terminal  Help
/ # cd /proc/1
/proc/1 # ls
arch_status         fd                  numa_maps           smaps_rollup
attr                fdinfo              oom_adj             stack
autogroup           gid_map             oom_score           stat
auxv                io                  oom_score_adj       statm
cgroup              limits              pagemap             status
clear_refs          loginuid            patch_state         syscall
cmdline             map_files           personality         task
comm                maps                projid_map          timens_offsets
coredump_filter     mem                 root                timers
cpu_resctrl_groups  mountinfo           sched               timerslack_ns
cpuset              mounts              schedstat           uid_map
cwd                 mountstats          sessionid           wchan
environ             net                 setgroups
exe                 ns                  smaps
/proc/1 # cat cgroup
13:devices:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c7f0
12:blkio:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c7f0
11:cpuset:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c7f0
10:perf_event:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c
7f0
9:cpu,cpuacct:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c
7f0
8:freezer:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c7f0
7:misc:/
6:rdma:/
5:net_cls,net_prio:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113d
df38c7f0
4:memory:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c7f0
3:pids:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c7f0
2:hugetlb:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38c7f0
1:name=systemd:/docker/966650153f23fe24940bfce660c655990c891d89ee0fd36e5897113ddf38
c7f0
0::/system.slice/containerd.service
/proc/1 #
```

```
/proc/1 $ cat cgroup
13:misc:/
12:memory:/docker/476639372a777ded4f1309431545e0b5d3a145410f
1c19c229e0d946194a8864
11:perf_event:/docker/476639372a777ded4f1309431545e0b5d3a145
410f1c19c229e0d946194a8864
10:hugetlb:/docker/476639372a777ded4f1309431545e0b5d3a145410
f1c19c229e0d946194a8864
9:blkio:/docker/476639372a777ded4f1309431545e0b5d3a145410f1c
19c229e0d946194a8864
8:pids:/docker/476639372a777ded4f1309431545e0b5d3a145410f1c1
9c229e0d946194a8864
7:devices:/docker/476639372a777ded4f1309431545e0b5d3a145410f
1c19c229e0d946194a8864
6:net_cls,net_prio:/docker/476639372a777ded4f1309431545e0b5d
3a145410f1c19c229e0d946194a8864
5:rdma:/
4:cpuset:/docker/476639372a777ded4f1309431545e0b5d3a145410f1
c19c229e0d946194a8864
3:cpu,cpuacct:/docker/476639372a777ded4f1309431545e0b5d3a145
410f1c19c229e0d946194a8864
2:freezer:/docker/476639372a777ded4f1309431545e0b5d3a145410f
1c19c229e0d946194a8864
1:name=systemd:/docker/476639372a777ded4f1309431545e0b5d3a14
5410f1c19c229e0d946194a8864
0::/system.slice/containerd.service
/proc/1 $
```

Answer the questions below

Read the above section and familiarize yourself with your new environment

Completed

Submit the flag on L-SRV02

Completed

Docker Container Network Enumeration

Once containerization is confirmed, enumerating the internal network becomes critical to uncover services and potential lateral movement paths. A primary focus is identifying the **default gateway** of the container, which often represents the Docker bridge network interface. In this scenario, the default gateway was identified as **192.168.100.1**.

Common tools and scripting languages available within most Linux environments are utilized for port scanning:

- **Bash-based port scanner**: Leverages the /dev/tcp/ pseudo-device to check port responsiveness.
- **Python-based port scanner**: Utilizes the socket library to probe known service ports.
- **Netcat (nc)**: Used in scanning mode to quickly check ranges of ports across the gateway.

Through these methods, two significant open ports were identified on the container's gateway:

- **Port 8080**: A high-numbered port often used for alternative HTTP services or administrative panels.
- **Port 3306**: The default port for MySQL databases, confirming an earlier finding of a MySQL backend.

This enumeration confirmed the presence of critical services that can be further leveraged during exploitation or post-exploitation.

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.100.1   0.0.0.0         UG    0      0        0 eth0
192.168.100.0   0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

```
Port 22 open
Port 80 open
Port 3306 open
Port 8080 open
```

What is the Default Gateway for the Docker Container?

Answer: 192.168.100.1
This is typically obtained using:

What is the high web port open in the container gateway?

Answer: 8080
This is a common alternative HTTP port, often used for web applications.

What is the low database port open in the container gateway?

Answer: 3306
Port 3306 is the default port for MySQL, which aligns with the earlier context regarding a MySQL backend.

Database Credential Discovery

A logical assumption for a confirmed web server is the presence of a backend database, likely MySQL, supporting dynamic content delivery. Databases, while often protected from remote access, can be poorly secured locally, exposing sensitive configuration files.

One such critical file, db_connect.php, is typically found at the root of the web directory (e.g., /var/www/) and contains hardcoded database credentials for PHP-to-MySQL connectivity. In this engagement, the db_connect.php file was successfully accessed, revealing the following cleartext connection parameters:

- **Database host address**: 192.168.100.1
- **Username**: admin
- **Password**: 1123SecureAdminDashboard321!
- **Database Name**: DashboardDB

These credentials allowed local access to the MySQL service. Basic SQL commands were then used to enumerate the database structure and contents:

```
define('DB_SRV', '192.168.100.1');
define('DB_PASSWD', "!123SecureAdminDashboard321!");
define('DB_USER', 'admin');
define('DB_NAME', 'DashboardDB');
```

- SHOW DATABASES;: To list available databases.
- USE DashboardDB;: To select the target database.
- SHOW TABLES;: To enumerate tables.
- SHOW COLUMNS FROM <table>;: To view table schema.
- SELECT * FROM <table>;: To extract data from specific tables.

This phase highlighted the importance of internal configuration security and the risks associated with hardcoded credentials in web-accessible locations. For instance, within the DashboardDB, the users table was found, containing usernames and hashed passwords.

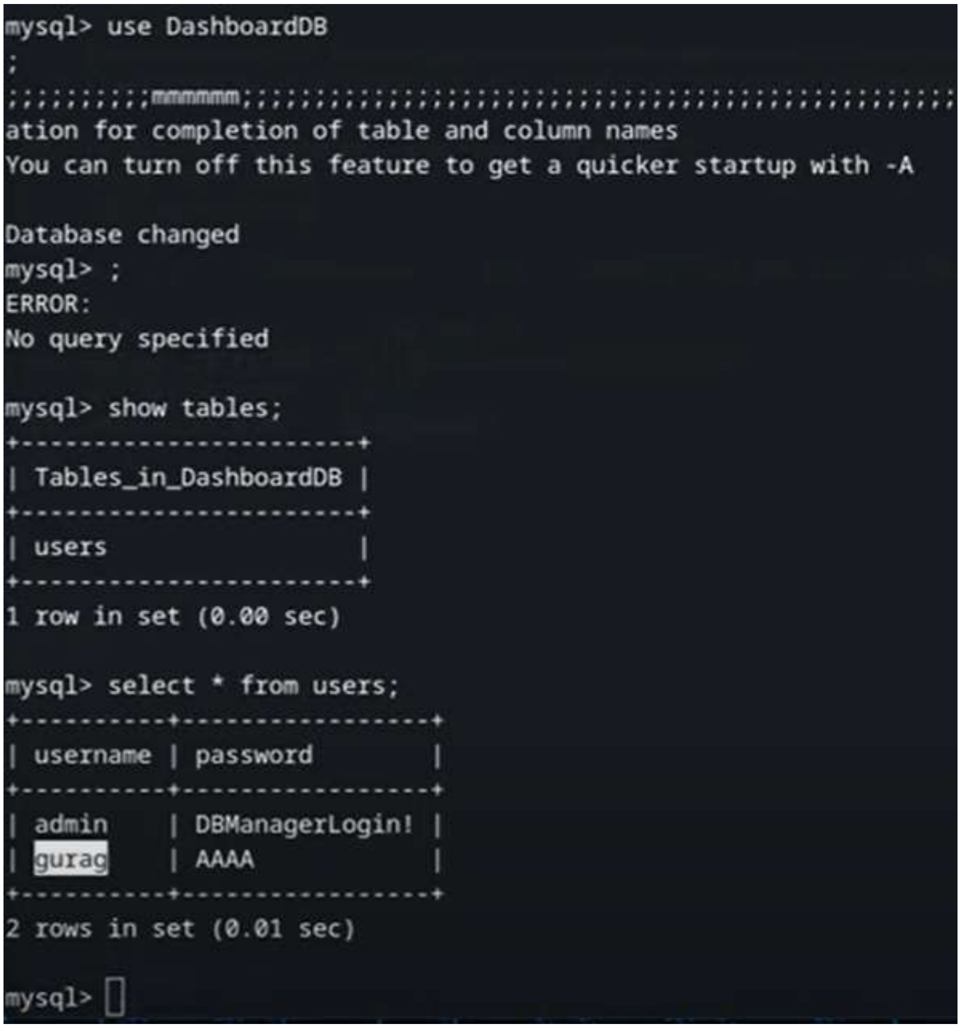
```
www-data@d00976975e91:/var/www/admin$ mysql -h 192.168.100.1 -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 109
Server version: 8.0.22-0ubuntu0.20.04.2 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```



```
mysql> use DashboardDB
;
;;;;;;;;;;mmmmmm;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ation for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> ;
ERROR:
No query specified

mysql> show tables;
+----------------------+
| Tables_in_DashboardDB |
+----------------------+
| users                |
+----------------------+
1 row in set (0.00 sec)

mysql> select * from users;
+----------+----------------+
| username | password       |
+----------+----------------+
| admin    | DBManagerLogin! |
| gurag    | AAAA           |
+----------+----------------+
2 rows in set (0.01 sec)

mysql> 
```

Windows Host Situational Awareness

After gaining a user account on a Windows machine (PC-FILESRV01), situational awareness shifts to understanding the system's security posture and available resources for further exploitation. This includes identifying active security products (AV/EDR), system and user information, privilege escalation opportunities, and potential lateral movement or persistence paths.

Defensive Measures Identification

Identifying defensive measures is a critical initial step. Two primary tools for this are Seatbelt and SharpEDRChecker.

Seatbelt

Seatbelt is a C#-based utility designed for host reconnaissance and security checks. It can perform various security-oriented "safety checks" relevant from both offensive and defensive perspectives. Key commands for AV and security enumeration include:

- AMSIProviders: Lists registered AMSI providers.
- AntiVirus: Displays registered antivirus software via WMI.
- Sysmon: Extracts Sysmon configuration from the registry.
- WindowsDefender: Shows Windows Defender settings, including exclusions.

```
====== AMSIProviders ======

  GUID                          : {2781761E-28E0-4109-99FE-B9D127C57AFE}
  ProviderPath                  : "C:\ProgramData\Microsoft\Windows Defender\Platform\4.1

====== AntiVirus ======

  Engine                        : Windows Defender
  ProductEXE                    : windowsdefender://
  ReportingEXE                  : %ProgramFiles%\Windows Defender\MsMpeng.exe

  Engine                        : Norton Security
  ProductEXE                    : C:\Program Files\Norton Security\Engine\22.14.0.54\WSCS
  ReportingEXE                  : C:\Program Files\Norton Security\Engine\22.14.0.54\WSCS
```

```
====== InterestingProcesses ======

    Category     : defensive
    Name         : MsMpEng.exe
    Product      : Windows Defender AV
    ProcessID    : 4668
    Owner        :
    CommandLine  :
```

SharpEDRChecker

SharpEDRChecker provides a more detailed output than Seatbelt, offering greater insight into advanced anti-virus and detection agents. It checks running processes, process metadata, loaded DLLs, common install directories, installed services, and drivers for the presence of known defensive products. Its key functions include:

- **FileChecker**: Checks the metadata of files, which cannot be changed without invalidating code signing.
- **ProcessChecker**: Inspects all processes and checks for DLLs loaded by processes to identify products like Cylance and AMSI.
- **ServiceChecker**: Inspects installed services.
- **DriverChecker**: Performs checks on all drivers using PInvoke.

- **DirectoryChecker**: Dumps interesting subdirectories in common locations (e.g., Program Files, ProgramData).

```
[!] Process Summary:
        [-] MsMpEng.exe : msmpeng
        [-] smartscreen.exe : defender
        [-] SecurityHealthService.exe : securityhealthservice
        [-] SecHealthUI.exe : defender


[!] Modload Summary:
        [-] C:\Windows\SYSTEM32\amsi.dll : amsi.dll, anti-malware, malware
        [-] C:\ProgramData\Microsoft\Windows Defender\platform\4.18.2102.4-0\MpOav.dll : a


[!] Directory Summary:
        [-] C:\Program Files\Windows Defender : defender
        [-] C:\Program Files\Windows Defender Advanced Threat Protection : defender, threa
        [-] C:\Program Files (x86)\Windows Defender : defender


[!] Service Summary:
        [-] mpssvc : defender
        [-] PolicyAgent : defender
        [-] SecurityHealthService : securityhealthservice
        [-] Sense : defender, threat
        [-] WdNisSvc : antivirus, defender, nissrv
        [-] WinDefend : antimalware, antivirus, defender, malware, msmpeng
        [-] wscsvc : antivirus


[!] Driver Summary:
        [-] WdFilter.sys : antimalware, malware
```

System Enumeration Using Seatbelt

After understanding the system's detection mechanisms and clarifying what actions are possible within our attack surface, the next step is to perform system enumeration. Enumeration helps us gather detailed information about the endpoint, allowing us to map out the attack surface and identify potential paths for privilege escalation.

For this purpose, we will use **Seatbelt**, a powerful enumeration tool designed to perform multiple system checks and gather valuable data about the endpoint.

What is Seatbelt?

Seatbelt is a comprehensive enumeration tool that runs a variety of checks to collect system information quickly and efficiently. It helps uncover configuration details, user privileges, running processes, and much more. In this phase, we will be leveraging *all* the modules Seatbelt provides to get a full picture of the system's state.

How to Run Seatbelt

To run Seatbelt and collect all available data, the command syntax is:

Syntax: .\Seatbelt.exe all

```
=== Basic OS Information ===

  Hostname                      :  WinDev2101Eval
  Domain Name                   :
  Username                      :  WINDEV2101EVAL\User
  ProductName                   :  Windows 10 Enterprise Evaluation
  EditionID                     :  EnterpriseEval
  ReleaseId                     :  2009
  BuildBranch                   :  vb_release
  CurrentMajorVersionNumber     :  10
  CurrentVersion                :  6.3
  Architecture                  :  AMD64
  ProcessorCount                :  8
  IsVirtualMachine              :  True
  BootTime (approx)             :  4/2/2021 10:48:49 PM
  HighIntegrity                 :  False
  IsLocalAdmin                  :  True
    [*] In medium integrity but user is a local administrator- UAC can be bypassed.
```

We can also run Seatbelt from Covenant using the Seatbelt module found below.

Module: Seatbelt

What CLR version is installed on PC-FILESRV01?

PowerShell module

```
====== DotNet ======

  Installed CLR Versions
      4.0.30319
```

What PowerShell version is installed on PC-FILESRV01?

PowerShell module

5.1.17763.1

```
Installed PowerShell Versions
    2.0
      [!] Version 2.0.50727 of the CLR is not installed - PowerShell v2.0 won'
    5.1.17763.1
```

What Windows build is PC-FILESRV01 running on?

OSInfo module

17763.1577

```
Hostname                    :  PC-FILESRV01
Domain Name                 :  holo.live
Username                    :  HOLOLIVE\watamet
ProductName                 :  Windows Server 2019 Datacenter
EditionID                   :  ServerDatacenter
ReleaseId                   :  1809
Build                       :  17763.1577
```

Situational Awareness ALL THE POWER!

In environments where EDR systems might block third-party tools like Seatbelt and PowerView, native PowerShell commands serve as a fallback for situational awareness and system/environment enumeration. PowerShell is built into Windows and offers a wide range of native commands and modules for deep system visibility, especially within Active Directory environments.

Important PowerShell commands and modules for situational awareness include

1) **Syntax: Get-NetLocalGroup**

   Lists local groups

```
PS C:\Users\User\Downloads> Get-NetLocalGroup

ComputerName   GroupName                           Comment
------------   ---------                           -------
WINDEV2101EVAL Access Control Assistance Operators Members of this group
WINDEV2101EVAL Administrators                      Administrators have c
WINDEV2101EVAL Backup Operators                    Backup Operators can
WINDEV2101EVAL Cryptographic Operators             Members are authorize
WINDEV2101EVAL Device Owners                       Members of this group
```

2) **Syntax: Get-NetLocalGroupMember -Group <group>** Lists users in a local group

```
PS C:\Users\User\Downloads> Get-NetLocalGroupMember -Group Administrator


ComputerName : WINDEV2101EVAL
GroupName    : Administrators
MemberName   : WINDEV2101EVAL\Administrator
SID          : S-1-5-21-921566831-3611186360-1917773840-500
IsGroup      : False
IsDomain     : False

ComputerName : WINDEV2101EVAL
GroupName    : Administrators
MemberName   : WINDEV2101EVAL\User
SID          : S-1-5-21-921566831-3611186360-1917773840-1001
IsGroup      : False
IsDomain     : False
```

Syntax: Get-NetLoggedon Lists users logged into the system

```
PS C:\Users\User\Downloads> Get-NetLoggedon


UserName     : User
LogonDomain  : WINDEV2101EVAL
AuthDomains  :
LogonServer  : WINDEV2101EVAL
ComputerName : localhost

UserName     : User
LogonDomain  : WINDEV2101EVAL
AuthDomains  :
LogonServer  : WINDEV2101EVAL
ComputerName : localhost
```

Step 1: Scheduled Tasks Enumeration

In environments where EDR systems might block third-party tools like Seatbelt and PowerView, native PowerShell commands serve as a fallback for situational awareness and system/environment enumeration. PowerShell is built into Windows and offers a wide range of native commands and modules for deep system visibility, especially within Active Directory environments.

Important PowerShell commands and modules for situational awareness include:

1) **Get-ScheduledTask**: Lists all scheduled tasks on the local machine. These can be abused for persistence or privilege escalation. Filters can be applied to focus on obscure tasks

```
PS C:\Users\User> Get-ScheduledTask -TaskPath "\Micr

TaskPath                                          TaskN
--------                                          -----
\Microsoft\VisualStudio\                          VSIX
\Microsoft\VisualStudio\Updates\                  Updat


PS C:\Users\User>
```

2) **Get-ScheduledTaskInfo:** Lists specific information on specified tasks, allowing an attacker to identify how a task could be exploited.

```
PS C:\Users\User> Get-ScheduledTaskInfo -TaskName "M



LastRunTime        : 4/5/2021 5:30:30 PM
LastTaskResult     : 0
NextRunTime        : 4/6/2021 3:08:08 AM
NumberOfMissedRuns : 0
TaskName           : Microsoft\VisualStudio\VSIX Aut
TaskPath           :
PSComputerName     :



PS C:\Users\User>
```

For more information about Get-ScheduledTaskInfo, check out the Microsoft docs, Step 2: Privilege Awareness

whoami /priv

This command is actually a standard Windows command, not PowerShell-specific, but it's useful when run in a PowerShell terminal.

It lists all privileges currently assigned to your user token — for example, SeDebugPrivilege, SeImpersonatePrivilege, etc. These can help determine whether you already have rights that could be used in an attack.

```
PS C:\Users\User> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description
============================= ======================
SeShutdownPrivilege           Shut down the system
SeChangeNotifyPrivilege       Bypass traverse checki
SeUndockPrivilege             Remove computer from d
SeIncreaseWorkingSetPrivilege Increase a process wor
SeTimeZonePrivilege           Change the time zone
```

The fourth PowerShell command we will be looking at is Get-ADGroup; this module, part of the active directory module package, will allow us to enumerate a user's groups or all groups within the domain. To get the most out of this command, we will already need to enumerate the users present on the machine. Since this command is part of the ActiveDirectory module, you will need first to import the module. Find the syntax for the command below.

Syntax: Import-Module ActiveDirectory; Get-ADGroup

After running the command, you will be prompted with a CLI to apply filters to the command; we recommend filtering by the samAccountName. Find example usage for this filter below.

Syntax: samAccountName -like "*"

```
PS C:\Users\Administrator> Import-Module ActiveDirectory; Get-ADGroup

cmdlet Get-ADGroup at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
Filter: samAccountName -like "*"


DistinguishedName : CN=Administrators,CN=Builtin,DC=holo,DC=live
GroupCategory     : Security
GroupScope        : DomainLocal
Name              : Administrators
ObjectClass       : group
ObjectGUID        : df986e49-0a48-4d60-8022-92c0b171a8a4
SamAccountName    : Administrators
SID               : S-1-5-32-544

DistinguishedName : CN=Users,CN=Builtin,DC=holo,DC=live
GroupCategory     : Security
GroupScope        : DomainLocal
Name              : Users
ObjectClass       : group
ObjectGUID        : 07846b42-331f-4f72-ab68-3f229e94bd5c
SamAccountName    : Users
SID               : S-1-5-32-545
```

To get the most out of this command, you will need to play with the filters and parameters used to get the most efficient output to enumerate the critical information.

For more information about Get-ADGroup, check out the Microsoft

For more information about Get-ADGroupMember,

The final PowerShell command we will be looking at is Get-ADPrincipalGroupMembership, similar to Get-ADGroupMember, this command will retrieve the groups a user, computer group, or service account is a member of. In order to get the most out of this command we will need to already have some targeted users enumerated using other commands like Get-ADUser. Since this command is part of the ActiveDirectory module you will need to first import the module. Find the syntax for the command below.

The final PowerShell command we will be looking at is Get-ADPrincipalGroupMembership; similar to Get-ADGroupMember; this command will retrieve the groups a user, computer group, or service

account is a member. To get the most out of this command, we will need to have enumerated target users using other commands like Get-ADUser. Since this command is part of the *ActiveDirectory* module, you will need first to import the module. Find the syntax for the command below.

Syntax: Import-Module ActiveDirectory; Get-ADPrincipalGroupMembership

After running the command, you will be prompted with a CLI to specify the user(s) you want to enumerate.

```
PS C:\Users\Administrator> Import-Module ActiveDirectory; Get-ADPrincipa

cmdlet Get-ADPrincipalGroupMembership at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
Identity: SRV-ADMIN


distinguishedName : CN=Domain Users,OU=Groups,DC=holo,DC=live
GroupCategory     : Security
GroupScope        : Global
name              : Domain Users
objectClass       : group
objectGUID        : b84fadf8-91ec-424a-9d7f-3ada9795ec9c
SamAccountName    : Domain Users
SID               : S-1-5-21-471847105-3603022926-1728018720-513

distinguishedName : CN=Administrators,CN=Builtin,DC=holo,DC=live
GroupCategory     : Security
GroupScope        : DomainLocal
name              : Administrators
objectClass       : group
objectGUID        : df986e49-0a48-4d60-8022-92c0b171a8a4
SamAccountName    : Administrators
SID               : S-1-5-32-544
```

When using PowerShell for offensive operations, you will need to play around with the commands and modules to see what works for you and develop your methodology similar to working with other tools.

Answer the questions below

Read the above and enumerate PC-FILESRV01 using PowerShell.

Completed