Active Directory Offline Hash Dump and Forensic Analysis

Csaba Barta
csaba.barta@gmail.com

July 2011

Disclaimer

The views, opinions and thoughts in this document are the views, opinions and thoughts of the author of the document and do not represent the views, opinions or thoughts of any past or current employer of the author or any other third person. The document is provided 'as is' without warranty of any kind. Use at your own responsibility. The software tools are provided for educational purposes only.

Table of contents

Introduction

The author participated in a project where it was required to extract the password hashes from an offline `NTDS.DIT` file. After searching the Internet for an available tool, the author found that there was no open source tool. Because of that the author decided to research the internals of password encryption and storage of Active Directory and create a tool for the forensic community.

A debt of gratitude to the author's colleague **Laszlo Toth** (http://www.soonerorlater.hu) who helped a lot in researching the encryption algorithms used during password storage. Thank you Laszlo!

What is NTDS.DIT?

The NTDS.DIT file is used to store nearly all the information that is accessible in the Active Directory (user objects, groups, membership information etc.). The file is usually located in the `%WINDIR%\NTDS\` folder after the administrator runs `dcpromo` (which transforms the windows server into a domain controller). In the same folder there are other files that are used to provide some kind of recovery for the database in case of emergency situations like power outage. These files store uncommitted or unsaved transactions that can be rolled back during recovery in order to restore the database to a consistent state.

Obtaining NTDS.DIT and the registry

In case of a live domain controller it is not trivial how one can obtain the NTDS.DIT file and the important registry hives, because they are constantly locked for writing by the user SYSTEM. This means that no userland process can access the files even for reading. Basically there are two options in this case:

- Use a 3rd party forensic software (which supports acquiring locked files)
- Utilise Volume Shadow Copy Services (http://blogs.msdn.com/b/adioltean/archive/2005/01/05/346793.aspx)

Using a 3rd party forensic software is essential for forensically sound acquisition. In case of testing the second option might be sufficient.

Structure of NTDS.DIT

In fact the `NTDS.DIT` file is a database with usually 3 or more tables. The name and purpose of the important tables are the following:

- **`datatable`** used to store the objects accessible in Active Directory
- **`link_table`** used to provide references to objects (like the field `memberof`)
- **`sd_table`** used to store the security descriptors (introduced with Server 2k3)

The database engine which can be used to access the data stored in the tables is called Extensible Storage Engine (ESE for short or JET Blue) and it is one of the proprietary engines of Microsoft. The exact same engine can be used to access data stored in Exchange Server mailboxes. The only difference between Excahnge databases and `NTDS.DIT` is the pagesize. In case of `NTDS.DIT` the pagesize is 8192 bytes, while in case of Exchange it is 4096 bytes.

The columns of the tables (attributes of objects) are described in the schema. Every object stored in the database has it's own record with all the attributes even if that attribute does not relate to the object at all (in this case the value of the attribute is `null`). For example a simple table might look like this:

Object name	Attribute 1	Attribute 2	Attribute 3
Object 1	1	2	null
Object 2	null	2	3

In this case "Object 1" has the "Attribute 1 and 2" and does not have "Attribute 3" while "Object 2" has "Attribute 2 and 3" and no "Attribute 1".

The names of the columns are not too descriptive. It is usually not possible to deduce the purpose of the value stored in the column from the column name.

The following columns are important to dump password hashes and some information about user accounts that might be useful in case of a forensic investigation:

ATTm3	Large text	SAMAccountName
ATTm13	Large text	Description
ATTr589970	Large binary data	SID
ATTq589920	Windows File Time	Date and time of last password change
ATTj589832	32 bit Integer	UserAccountControl field
ATTq589983	Windows File Time	Date and time of account expiry
ATTq589876	Windows File Time	Date and time of last login
ATTj589993	32 bit Integer	Bad password count
ATTk589879	Large binary data	Encrypted LM hash
ATTk589914	Large binary data	Encrypted NT hash
ATTk589918	Large binary data	Encrypted NT hash history
ATTk589984	Large binary data	Encrypted LM hash history
ATTk590689	Large binary data	Encrypted PEK (Password Encryption Key)

Password hash encryption used in Active Directory

Note, that in the previous list there are numerous fields that are described as encrypted. The purpose of this encryption is to provide protection against offline data extraction.

The solution introduced by Microsoft in order to provide this protection is complex and composed of 3 layers of encryption of which 2 layers use RC4 and the third layer uses DES.

In order to decrypt a hash stored in NTDS.DIT the following steps are necessary:

1. decrypt the PEK (Password Encryption Key) with bootkey (RC4 - layer 1)
2. hash decryption first round (with PEK and RC4 - layer 2)
3. hash decryption second round (DES - layer 3)

Password Encryption Key

The PEK or Password Encryption Key is used to encrypt data stored in NTDS.DIT. This key

is the same across the whole domain, which means that it is the same on all the domain controllers. The PEK itself is also stored in the NTDS.DIT in an encrypted form. In order to decrypt it one will need the registry (the SYSTEM hive) from the same domain controller where NDTS.DIT file was obtained. This is because the PEK is encrypted with the BOOTKEY which is different on all domain controllers (and in fact on all computers in the domain).

In order to decrypt the PEK one will have to obtain the ATTk590689 field from the NTDS.DIT. As it was mentioned all the objects stored in the database will have this field. In order to determine which one is needed one has to check whether the value is null or not.

The length of the value is 76 bytes (it is stored as binary data). The structure of the value is the following:

header 8 bytes	key material for RC4 16 bytes	encrypted PEK 52 bytes

After decryption the value of the decrypted PEK can also be divided into 2 parts. One will have to skip the first 36 bytes (so the length of the actual PEK key is 16 bytes).

Here is the python algorithm that can be used to decrypt the PEK key after one has obtained the bootkey (bootkey can be collected from the SYSTEM registry hive and the method is well documented - http://moyix.blogspot.com/2008/02/syskey-and-sam.html):

```
md5=MD5.new()
md5.update(bootkey)
for i in range(1000):
      md5.update(enc_pek[0:16])
rc4_key=md5.digest();
rc4 = ARC4.new(rc4_key)
pek=rc4.encrypt(enc_pek[16:])
return pek[36:]
```

As one can see there is an MD5 hashing part of the decryption with 1000 rounds. This is for making the bruteforce attack against the key more time consuming.

Password Hash Decryption

Now that the PEK is decrypted the next task is decrypt the hashes stored in the ATTk589879 (encrypted LM hash) and ATTk589914 (encrypted NT hash) attributes of user objects.

The first step is to remove the RC4 encryption layer. During this the PEK key and the first 16 bytes of the encrypted hash is used as key material for the RC4 cypher. Below is the structure of the 40 bytes long encrypted hash value stored in the NTDS.DIT database.

header 8 bytes	key material for RC4 16 bytes	encrypted hash 16 bytes

The algorithm to remove the RC4 encryption layer is the following:

```
md5 = MD5.new()
md5.update(pek)
md5.update(enc_hash[0:16])
rc4_key = md5.digest();
rc4 = ARC4.new(rc4_key)
denc_hash = rc4.encrypt(enc_hash[16:])
```

The final step is to remove the DES encryption layer which is in fact very similar to the so called “standard” SYSKEY encryption used in case of password hashes stored in the registry (details of the algorithm can be found here - http://moyix.blogspot.com/2008/02/syskey-and-sam.html).

Below is the last part of the algorithm:

```
(des_k1,des_k2) = sid_to_key(rid)
d1 = DES.new(des_k1, DES.MODE_ECB)
d2 = DES.new(des_k2, DES.MODE_ECB)
hash = d1.decrypt(denc_hash[:8]) + d2.decrypt(denc_hash[8:])
```

Notice, that it is essential to have the SID of the user in order to determine the RID and to compute the keys used for DES.

Decrypting the password hash history

During a computer forensic investigation the password history might play a very important role. In case when the investigator needs to decrypt an encrypted file for which the password is unknown it might be very helpful to see how the person used to choose passwords (what are the “rules” he/she follows).

In order to decrypt the password history the investigator needs to extract the `ATTk589918` (encrypted NT hash history) and `ATTk589984` (encrypted LM hash history) from `NTDS.DIT`. The decryption process is very similar to the one detailed above. The only difference is that the whole history needs to be decrypted with the `PEK` and afterwards the hashes should be decrypted one by one using RC4 and DES because the history is created by concatenating the encrypted hashes resulting in a huge binary value which is encrypted with the `PEK`.

Forensic analysis of user objects stored in NTDS.DIT

During a computer forensic investigation it might be important to extract as much information as possible of a user account. This part of the document describes what user account related information can be extracted from `NTDS.DIT`.

Important fields

One should analyse the following fields in order to gain information about the account:

- **ATTm3** SAMAccountName
- **ATTm13** Description
- **ATTk589970** SID
- **ATTq589920** Date and time of last password change
- **ATTj589832** UserAccountControl field
- **ATTq589983** Date and time of account expiry
- **ATTq589876** Date and time of last login
- **ATTj589993** Bad password count

The fields containing date or time values can be interpreted as UTC Windows File Time, because the Active Directory usually stores date and time information in this format. Windows Filetime is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC) (http://msdn.microsoft.com/en-us/library/ms724284(v=vs.85).aspx).

The following python code snippet transforms the value into human readable form:

```
import datetime
_FILETIME_null_date = datetime.datetime(1601, 1, 1, 0, 0, 0)
timestr = _FILETIME_null_date + \
          datetime.timedelta(microseconds=int(value) / 10)
```

Date and time of the last logon

In case of the last logon field there is an important thing that an investigator should always bear in mind. The time that is stored in `NTDS.DIT` is the last logon that happened on the domain controller from which the file was obtained. It might happen that on another DC the last logon time is different. In order to find out the proper time of the last login one should check the stored value on all the DCs.

Bad password count

The high value of the bad password count field could indicate a bruteforce attack against the user account.

UserAccountControl field

From the UserAccountControl field a wealth of information could be obtained. This value is used to store multiple flags regarding the user account. The details of the important flags can be found in the following table.

Value	Description
0x00000001	Logon script is executed
0x00000002	The user account is disbaled
0x00000010	The user account is locked out
0x00000020	No password is required for the account
0x00000040	The user cannot change password
0x00000200	Account type is normal account
0x00000800	Account type is interdomain trust account
0x00001000	Account type is workstation trust account
0x00002000	Account type is server trust account
0x00010000	Password of the user account will never expire
0x00020000	The account type is MSN logon
0x00040000	Smart card is required for logon
0x00800000	The password of the user account is expired

This field might indicate if the account is disabled or locked and other information that might be important in case of a computer forensic investigation. More information can be found at the following URL:

http://msdn.microsoft.com/en-us/library/ms680832(v=vs.85).aspx

Tools developed by the author

The hashdump process is composed of two main parts

1. Extracting the required data from NTDS.DIT (esedbdumphash)
2. Decrypting the hashes and interpreting other information regarding the user account (dsdump, dsdumphistory, dsuserinfo)

For the extraction of the important records stored in the NTDS.DIT file the author used the `libesedb` library (developed by Joachim Metz) that can be downloaded from the following URL:

http://sourceforge.net/projects/libesedb/

A proof of concept tool called `esedbdumphash` was created based on the source of the tool called `esedbexport` (that is included by default in libesedb) in order to include only the important objects in the export and to minimise the file size of the output.

In order to decrypt the hashes the author decided to extend the excellent framework called `creddump` developed by Brendan Dolan-Gavitt. The original version of the framework can be downloaded from the following URL:

http://code.google.com/p/creddump/

Three new modules were added to the framework in order to achieve the goals:

- `dsdump.py` Password hash dumper (in cooperation with Laszlo)
- `dsdumphistory.py` Password hash history dumper
- `dsuserinfo.py` User account information dumper

A new main library file `dshashdump.py` was created in order to contain the functions needed to decrypt the hashes stored in the `NTDS.DIT` file.

The author also added support for dumping LSA secrets on newer operating systems like Windows Vista and Windows 7 (`lsadumpw2k8.py`).

It should be mentioned the tools are in proof of concept state.

Future work

The NTDS.DIT file contains other important information that can be useful in case of a computer forensic investigation. The author is currently working on the extraction of this information.