NTDS.DIT Forensics

Csaba Barta
csaba.barta@gmail.com
http://www.csababarta.com

December 2011

For the current version of this document visit www.csababarta.com

Introduction

The author participated in a project where it was required to be able 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!

Table of content

What is NTDS.DIT?

The `NTDS.DIT` file is used to store almost 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 order to perform a forensic analysis on `NTDS.DIT` the following information sources are needed from the domain controller:

1. NTDS.DIT
2. Registry hives (at least the SYSTEM hive)

In case of an online domain controller it is not trivial how one can obtain the `NTDS.DIT` file and the important registry hives, because they are kept locked by the operating system. This means that no userland process can access the files even for reading. Basically there are two options in this case:

1. Use a 3rd party forensic software (which supports acquiring locked files)
2. 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:

1. datatable - used to store the objects accessible in Active Directory
2. link_table - used to provide references to objects (introduced with Server 2003)
3. sd_table - used to store the security descriptors

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 Exchange 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 table contains examples for fieldnames and the purpose of the value stored in the fields:

Attribute name	Data type	Stored data
ATTm590045	Large text	SAMAccountName
ATTm13	Large text	Description
ATTr589970	Large binary data	SID
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	Logon count

Encryption in NTDS.DIT

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.

Password Encryption Key

The main encryption method is based on the Password Encryption Key (PEK). The PEK 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.

Decrypting the PEK

The PEK is encrypted with the BOOTKEY which is different on all the computers across the Active Directory domain. Therefore in order to get the BOOTKEY one will need the SYSTEM hive of the registry obtained from the same domain controller from which the `NDTS.DIT` file was extracted. The method of collecting the BOOTKEY is well documented (http://moyix.blogspot.com/2008/02/syskey-and-sam.html).

The encrypted PEK is stored in the ATTk590689 field of the `datatable` in `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 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).

decrypted data 36 bytes	decrypted PEK 16 bytes

Below is sample algorithm written in python that can be used to decrypt the PEK.

```
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 MD5 hashing is part of the decryption with 1000 rounds. This is for making bruteforce attack against the key more time consuming.

Password hash decrypting
The solution introduced by Microsoft in order to store password hashes is complex and composed of 3 layers of encryption (including the encryption of the PEK itself) of which 2 layers use RC4 and the third layer uses DES.

The following columns are important to dump password hashes:

Attribute name	Data type	Stored data
ATTm590045	Large text	SAMAccountName
ATTr589970	Large binary data	SID
ATTj589832	32 bit Integer	UserAccountControl field
ATTq589876	Windows File Time	Date and time of last login
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)

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

1. Obtain the value of the ATTk589879 (encrypted LM hash) and ATTk589914 (encrypted NT hash) attributes for each user object
2. decrypt the PEK (Password Encryption Key) with bootkey (RC4 - layer 1)
3. hash decryption first round (with PEK and RC4 - layer 2)
4. hash decryption second round (DES - layer 3)

After decrypting the PEK the next step is to remove the second RC4 layer. During this the PEK 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

Below is a sample algorithm written in python that can be used to remove the RC4 encryption layer.

```
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 sample algorithm to do the decryption:

```
(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. The sid_to_key function is borrowed from the excellent framework called “creddump”.

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.

Time information stored in NTDS.DIT

In case of any database there are certain timestamps that are usually stored. Also the `NTDS.DIT` is not an exception to this. On the filesystem level the database creation time and other engine specific timestamps are stored. On the database level the engine stores the record creation timestamps. On the Active Directory level the directory engine stores timestamps related to activities like logins and password change.

Timestamp formats

There are basically 3 types of timeformat used in the `NTDS.DIT` database. These formats are used to represent the time and date of different activities. The formats are the following:

- Database time
- “Truncated” Windows Filetime
- Windows Filetime

Database time

This is a time structure that is 8 bytes long and looks like the following.

Offset	Description
0	Value of seconds (0-59)
1	Value of minutes (0-59)
2	Value of hours (0-23)
3	Value of days (0-31)
4	Value of months (0-12)
5	Value of years (0 is 1900)
6	Padding byte
7	Padding byte

The reader can find more information about this structure in the following document

http://sourceforge.net/projects/libesedb/files/Documentation/Extensible%20Storage%Engine%20%28ESE%29%20Database%20File%20%28EDB%29%20format/

Windows Filetime

The database fields containing Windows Filetime values should 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)
```

“Truncated” Windows Filetime
The name was given by the author (it is possible that the format has been documented before, but the author was not able to find any relevant documentation regarding it). The name indicates the assumption that the format is a version of the original Windows Filetime. The truncation means that the value is stored as the number of seconds not 100 nanoseconds.

Related framework module
The framework module called dstimeline can be used to build a timeline based on the date-time information stored in the database.

Deleted objects

The process of deleting objects from Active Directory is not straightforward. First of all deleted objects are not removed from the database. Some attributes will never be removed from the database.

The process of object deletion

The steps of the process can be found below:

1. The object gets marked as deleted (the `IsDeleted` attribute will be set to `TRUE`, this is called tombstoned) and moved to a special container called `Deleted Objects` (if the object has the flag `0x02000000` in it's `systemFlags` attribute, it won't be moved to the container. At this time the object is renamed by the system. The new name will include:

   - the original object name (the original will be truncated to 75 characters)
   - the `0x0A` character
   - the string "DEL:"
   - the original GUID

   This is because the `Deleted Objects` container is flat (every object is on the same level), and the engine must ensure unique object names. If the object is deleted on a domain controller running Windows Server 2003 the `lastKnownParent` attribute is set to the record id of the container from which the object was moved.
2. For a configurable period of time certain attributes are kept by the engine (this is called tombstone time). During this time the object with the attributes kept can be recovered manually or by a method called "carving" (by using this method more attributes might be recovered).
3. After the tombstone period is over and the next garbage collection is performed the the objects are deleted permanently.

Configuring tombstone time

By default the tombstone period is 60 days (for new domains and forest installed using Windows Server 2003 SP1 it is 180 days). The value is stored in the tombstoneLifetime attribute of the

```
cn=Directory Service,cn=Windows NT,cn=Services,cn=Configuration
```

object. The reader can use ADSIEdit, ldp, a Visual Basic script or any other tool that is capable of modifying attributes in Active Directory.

Configuring garbage collection time

By default the garbage collection period is 12 hours. The value is stored in the garbageCollPeriod attribute of the

```
cn=Directory Service,cn=Windows NT,cn=Services,cn=Configuration
```

object.

Configuring attributes to be kept

Attributes with the flag `0x00000008` in their `searchFlags` attribute won't be removed in case of deletion.

The user can use any tool that is capable to modify the Schema in order to specify the attributes to be kept by the system.
The following attributes are kept by the system by default (http://msdn.microsoft.com/en-us/library/windows/desktop/aa772216(v=vs.85).aspx):

- attributeID
- attributeSyntax
- distinguishedName
- dNReferenceUpdate
- flatName
- governsID
- groupType
- instanceType
- lDAPDisplayName
- legacyExchangeDN
- mS-DS-CreatorSID
- mSMQOwnerID
- name
- nCName
- objectClass
- objectGUID
- objectSid
- oMSyntax
- proxiedObjectName
- replPropertyMetaData
- sAMAccountName
- securityIdentifier
- subClassOf
- systemFlags
- trustAttributes
- trustDirection
- trustPartner
- trustType
- userAccountControl
- uSNChanged
- uSNCreated
- whenCreated

The following attributes are always removed (http://msdn.microsoft.com/en-us/library/windows/desktop/aa772216(v=vs.85).aspx):

- objectCategory
- samAccountType

Related framework module
The `dsdeletedobjects` module of the `NTDSXtract` framework can be used list and extract deleted objects from the database.

Group object related information

Group objects are also stored in the table called datatable. These objects represents the security groups that are used to control access and privileges within the domain. The members of the different groups usually also have different privileges.

Enumerating members

Enumerating members of a group or the list of groups of which the user is a member is not a straightforward task. If the reader queries the directory service (e.g. with the tool called LDP) for the memberof attribute of a user or the members attribute of a group the list won't be complete. For example the members attribute of the group called "Domain users" won't contain all the members. This is because of an old special restriction that affects the way how the system stores and handles group membership information.

In the past in case of Windows 2000 (when Active Directory was introduced) there was a restriction regarding the number of members a group could have. Any group could only have 5000 members. This restriction was introduced because at that time that was the maximum size of a field that the system was able to replicate.

In order to overcome this problem (domains containing more than 5000 users or computers are common nowadays) Microsoft introduced the attribute called primary group. Any user object can have only one primary group. This way a group can include more than 5000 users because this property belongs to and stored along with the user object (so it won't affect the size of the members attribute).

When Windows Server 2003 was introduced Microsoft also introduced a new solution for this problem. A new table was designed called "link_table". In case of new domains this is the table that stores the membership information. Each membership means one record in this table. The primary group was retained for compatibility reasons. It means that the reader should still enumerate the user objects with the primary group attribute set to the RID of the group the members of which he/she want's to query.

The reader can gain more information regarding this restriction at the following URL:

http://support.microsoft.com/kb/275523

Deleted group memberships

The fact that with Windows Server 2003 a new table was introduced also affects the amount of information that can be extracted regarding group memberships. Each record in the link_table has a field containing a timestamp that is filled with the time when the user is removed from that group. It means that it is possible to extract information regarding deleted group memberships.

This technique is limited because of the fact that if the membership is given back to the user the value of the filed will be reset to the default value. The table also does not contain the memberships given by setting the primary group attribute.

User object related information

During a computer forensic investigation it is 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`.

Password hashes

The record containing the user object also stores the password hashes (LM and NT depending on the settings) together with the password history of both hash type. The hashes are stored in an encrypted format.

Certificates

The certificates that are published in Active Directory are usually stored in the ATTk36 field without encryption in DER format.

Supplemental credentials

The information required for authentication in certain cases are stored as supplemental credentials.
The data is stored in a special structure in an encrypted form. More in formation regarding the structure can be found at the following URL

http://msdn.microsoft.com/en-us/library/cc245499(v=PROT.10).aspx

If the UserAccountControl field Encrypted Cleartext Password Allowed is set to TRUE then this field also contains the cleartext password.

Date and time of the last logon

This information is stored in the field called `ATTq589876`. In case of the last logon field there is an important fact 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. In fact Microsoft introduced a solution in order to solve this problem by replicating the last logon time (LastLogonTimestamp - `ATTq591520`) throughout the Active Directory domain. This option was introduced in Windows Server 2003.

Bad password count and bad password time

This value is stored in the field called `ATTj589836`. The high value of the bad password count field could indicate a bruteforce attack against the user account. The bad password time (stored in `ATTq589873`) indicates the last time the user entered bad password.

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 disabled

Value	Description
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).

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

Field name	Content
ATTm590045	SAMAccountName
ATTm590480	User principal name
ATTm13	Description
ATTk589970	SID
ATTq589920	Date and time of last password change
ATTj589832	UserAccountControl field
ATTq589983	Date and time of account expiry
ATTq589876	Date anf time of last logon
ATTj589993	Login count
ATTj589993	Bad password count

Field name	Content
ATTq589873	Bad password time
ATTk589949	Supplemental credentials
ATTk36	ADUserObjects

The fields of the list above containing date-time values can be interpreted as UTC Windows File Time, because the Active Directory usually stores date and time information in this format (see Windows Filetime above).

NTDSXtract (the framework developed by the author)

NTDSXtract is a framework that was developed in order to provide the forensic and IT security community with a tool that is able to extract important information from the NTDS.DIT database. The framework was developed in python which provides an almost platform independent solution.

Modules of the framework

The framework is composed of modules which can be used to extract information from the database regarding different objects and subjects.

Currently the framework includes the following modules:

- dsfileinformation
- dsdeletedobjects
- dstimeline
- dsusers
- dsgroups
- dscomputers

The description and details of the functionality and capabilities of each module can be found in the followings.

dsfileinformation

This module can be used to extract information from the fileheader of the NTDS.DIT file itself. The file header is actually the first database page or page number 0. It contains the following information:

- Database creation time
- Detailed version of the OS on which the database was created
- Last attach time
- Last detach time
- Last recovery time
- Database state (clean or dirty)

The module will also dump the full header for debug purposes.

Default output of the module:

```
$ python dsfileinformation.py
DSFileInformation
Extracts information related to the NTDS.DIT database file

usage: dsfileinformation.py <ntds.dit>
```

dsdeletedobjects

This module can be used to extract deleted objects from the database table “datatable”. It supports extracting the records to an output file that can be further examined if needed.

Default output of the module:

```
$ python dsdeletedobjects.py
DSDeletedObjects
Extracts information related to deleted objects
```

```
usage: dsdeletedobjects.py <datatable> [option]
  options:
    --output <output file name>
        The record containing the object and the preserved
        attributes will be written to this file
    --useIsDeleted
        Extract deleted objects based on the IsDeleted flag

Fields of the main output
    Rec. ID|Cr. time|Mod. time|Obj. name|Orig. container name
```

dstimeline

This module can be used to create a timeline that will represent the activities recorded in the database. The activities currently supported are:

- object creation
- object deletion
- object modification
- Password change
- Login
- Login time synchronization

Default output of the module:

```
$ python dstimeline.py
DSTimeline
Constructs timeline

usage: dstimeline.py <datatable> [option]
  options:
    --b
       mactime body format

Fields of the default output
    Timestamp|Action|Record ID|Obj. name|Obj. type
```

dsusers

This module can be used to extract information regarding user objects. Currently it supports extracting the following type of information:

- SID
- GUID
- SAMaccountName
- Password hashes
- Password history
- Group membership (deleted membership as well)
- Certificates
- Supplemental credentials

In order to extract password hashes and other credential related information the user will also need the SYSTEM registry hive from the same domain controller from which the NTDS.DIT file was obtained.

Default output of the module:

```
$ python dsusers.py
DSUsers
Extracts information related to user objects

usage: dsusers.py <datatable> <linktable> [option]
  options:
    --rid <user rid>
          List user identified by RID
    --name <user name>
          List user identified by Name
    --passwordhashes <system hive>
          Extract password hashes
    --passwordhistory <system hive>
          Extract password history
    --certificates
          Extract certificates
    --supplcreds <system hive>
          Extract kerberos keys
    --membership
          List groups of which the user is a member
```

dsgroups
This module can be used to extract Security group related information. Currently it supports extracting the following type of information:
- SID
- GUID
- Group name
- Members
- Membership deletion time

Default output of the module:

```
$ python dsgroups.py
DSGroups
Extracts information related to group objects

usage: dsgroups.py <datatable> <linktable> [option]
  options:
    --rid <group rid>
    --name <group name>
```

dscomputers
This module can be used to extract computer object related information. Currently it supports extracting the following type of information:
- SID
- GUID
- Computer name
- DNS name
- BitLocker recovery password and history

- BitLocker volume boot record (containing VMK that can be decrypted with the recovery password)

Default output of the module:

```
$ python dscomputers.py
DSComputers
Extracts information related to computer objects

usage: dscomputers.py <datatable> [option]
  options:
    --name <computer name>
    --passwordhashes <system hive>
    --passwordhistory <system hive>
    --bitlocker
```

3rd party tools used in order implement certain functionality
The author should point out that certain parts of the framework are based on the work of other forensic and IT security experts. The reader can find details about and references to the tools on which the framework is based.

For the extraction of record 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/

In order to decrypt the password hashes (after removing the PEK - RC4 encryption) the author decided to use parts of 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/

The author also added support to the framework for dumping LSA secrets on newer operating systems like Windows Vista and Windows 7 (lsadumpw2k8.py). The reader may find this module in the “framework” subdirectory.

It should be mentioned that the framework is still in proof of concept state. The reader can download it at the following URL:

http://www.csababarta.com/downloads/ntdsxtract/ntdsxtract_v1_0.zip

Appendix 1 - Active Directory Field Reference

Important fields related to objects in general

Field name	Field description
DNT_col	Record ID
PDNT_col	Parent Record ID
time_col	Record Creation Time
Ancestors_col	Ancestors
ATTm3	Object Name
ATTm589825	Object Name 2
ATTb590606	Object Type ID (the record ID of the type object)
ATTk589826	Object GUID
ATTl131074	When Created
ATTl131075	When Changed
ATTq131091	USN Created
ATTq131192	USN Changed
OBJ_col	ObjectCol
ATTi131120	IsDeleted

Important fields in connection with user objects

Field name	Field description
ATTr589970	Account SID
ATTm590045	SAMAccountName
ATTj590126	SAMAccountType
ATTm590480	User Principal Name
ATTj589832	UserAccountControl
ATTq589876	Date and time of last logon
ATTq591520	LastLogonTimeStamp
ATTq589983	Date and time of account expiry

Field name	Field description
ATTq589920	Date and time of last password change
ATTq589873	Date and time of last bad password
ATTj589993	Logon count
ATTj589836	Bad password count
ATTj589922	Primary group ID (the RID of the primary group)
ATTk589914	Encrypted NT hash
ATTk589879	Encrypted LM hash
ATTk589918	Encrypted NT hash history
ATTk589984	Encrypted LM hash history
ATTk591734	UnixPassword
ATTk36	ADUserObjects
ATTk589949	Supplemental credentials

Important fields in connection with group objects

Fields from data table

Field name	Field description
ATTr589970	SID

Fields from link_table

Field name	Field description
link_DNT	Target record ID
backlink_DNT	Source record ID
link_deltime	Date and time of link deletion

Important fields in connection with computer objects

Field name	Field description
ATTr589970	SID
ATTm590045	SAMAccountName
ATTj590126	SAMAccountType

Field name	Field description
ATTj589832	UserAccountControl
ATTq589876	LastLogon
ATTq591520	LastLogonTimeStamp
ATTq589983	AccountExpires
ATTq589920	PasswordLastSet
ATTq589873	BadPwdTime
ATTj589993	LogonCount
ATTj589836	BadPwdCount
ATTj589922	PrimaryGroupId
ATTk589914	NTHash
ATTk589879	LMHash
ATTk589918	NTHashHistory
ATTk589984	LMHashHistory
ATTm590443	DNSHostName
ATTm590187	OSName
ATTm590188	OSVersion
ATTm591788	BitLocker recovery password
ATTk591823	FVE Key Package (Full Volume Encryption Key)
ATTk591822	Volume GUID
ATTk591789	Recovery GUID