Decrypting the Directory:

A Journey into a static analysis of the Active Directory NTDS to identify misconfigurations and vulnerabilities

ABOUT US

Bastien Cacace
10+ years as Senior Auditor / Pentester
Product Owner of IAMBuster
@skisedr

Florian Duthu
7+ years as a
Pentester / Security auditor

@ xmco
We deliver cybersecurity expertise

Agenda

01 WHY NTDS ANALYSIS ?

02 INTERESTING FINDINGS

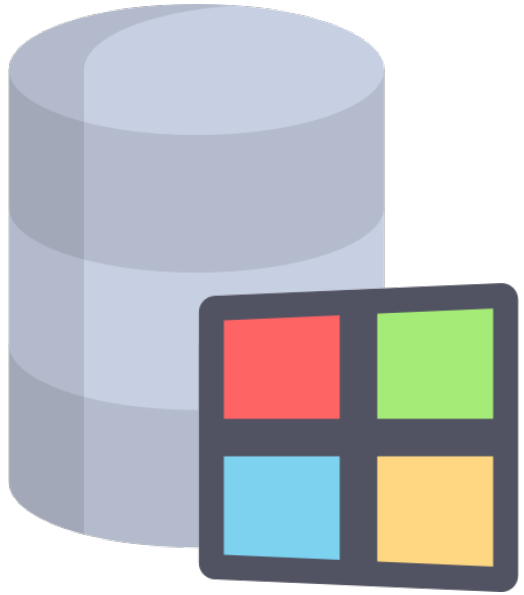
03 FORMAT & STRUCTURE

04 PARSING AND DATA

05 ACL & CONTROL PATH

WHAT IS NTDS ?

- NTDS: **NT D**irectory **S**ervice
- Database of the Active Directory (NTDS.dit) stored on each domain controller in `%SystemRoot%\ntds\NTDS.DIT`
- ESE format (Extensible Storage Engine) also called Jet Blue, first shipping with Windows NT 3.51 (1995) and Exchange 4.0 after.
- Why has MS created its own format instead of SQL? Technical issues and performance
- Today ESE is running in Active Directory, O365 Mailbox, and in Windows Systems



Active Directory Database

WHY NTDS ANALYSIS ?

1

xmco

TROOPERS

THE STARTING POINT

2017

PENTEST IN AD ENVIRONMENTS

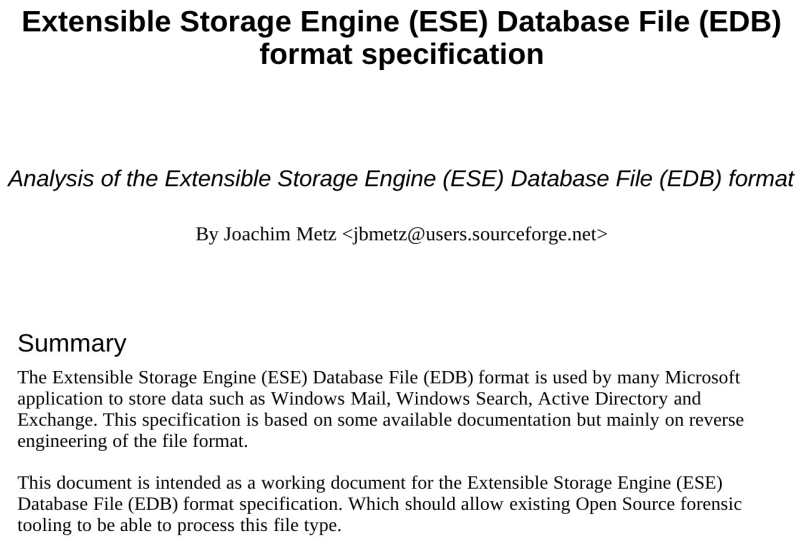
TEST THE PASSWORDS STRENGTH

CUSTOMER ASKING FOR AD PASSWORD AUDIT

BUT CAN WE GO FURTHER ?

PREVIOUS WORK ON THE NTDS

- Work from Joachim Metz, back to 2011 (esent.dll reverse engineering)



Extensible Storage Engine (ESE) Database File (EDB) format specification

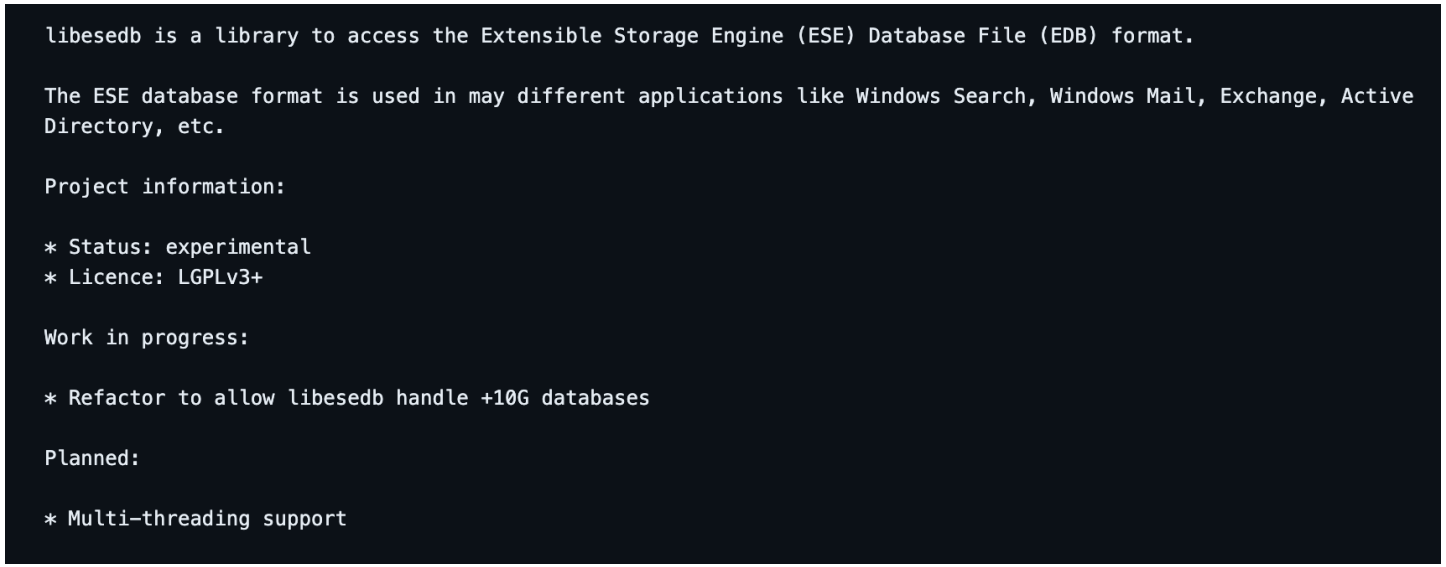
Analysis of the Extensible Storage Engine (ESE) Database File (EDB) format

By Joachim Metz <jbmetz@users.sourceforge.net>

Summary

The Extensible Storage Engine (ESE) Database File (EDB) format is used by many Microsoft application to store data such as Windows Mail, Windows Search, Active Directory and Exchange. This specification is based on some available documentation but mainly on reverse engineering of the file format.

This document is intended as a working document for the Extensible Storage Engine (ESE) Database File (EDB) format specification. Which should allow existing Open Source forensic tooling to be able to process this file type.



```
libesedb is a library to access the Extensible Storage Engine (ESE) Database File (EDB) format.

The ESE database format is used in may different applications like Windows Search, Windows Mail, Exchange, Active
Directory, etc.

Project information:

* Status: experimental
* Licence: LGPLv3+

Work in progress:

* Refactor to allow libesedb handle +10G databases

Planned:

* Multi-threading support
```

- Used by Impacket, Ntdsxtract, Airbus BTA and other tools
- Later, Microsoft has also published ManagedEsent provides managed access to esent.dll (used by DsInternals)

ESEDB specs : https://github.com/libyal/libesedb/blob/main/documentation/Extensible%20Storage%20Engine%20(ESE)%20Database%20File%20(EDB)%20format.asciidoc
ESEDB library : https://github.com/libyal/libesedb

MORE ON NTDS

- NTDS.dit contains all the data related to the Active Directory (objects)
  - Users and computers
  - Groups
  - Password hashes
  - Access Control Entry (ACE)
  - GPO references
  - Etc.

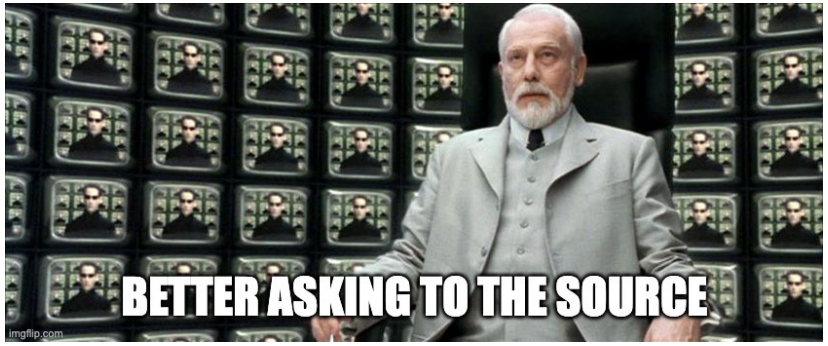
- Can start around 30MB up to more than 15GB in huge Active Directory contexts

- Can be dumped with the NDTSUtil tool (Domain Admins right required)

- Contains almost all the same data on each domain controller (some attributes are not replicated)

WHY NTDS ANALYSIS?

- Extracting information allows to conduct an **offline audit** or **an offline "forensic"**
- No need to be connected to the Information System* and send many requests over the network
- The only way to analyze the passwords strength (DC Sync apart 😇)
- Useful for :
  - blue-teamers who do not want to generate extra-noise on the network;
  - sensitive environments where dynamic testing is prohibited;
  - pentesters once they have compromised a domain to be more exhaustive in their reports regarding AD findings;



* except to dump ntds.dit

INTERESTING FINDINGS

2

xmco

TROOPERS

INTERESTING FINDINGS

1.

USERS

- User inactivity
- Members of high privileged built-in groups
- Users with SPN
- Users with no Pre-auth Kerberos
- Dangerous Kerberos delegations

2.

PASSWORDS

- Weak passwords
- Password storage formats
- Password re-use
- Password change date
- Domain default password policy

SYSTEMS

- Obsolete systems version
- Inactive machines
- Kerberos password renewal
- Trusts relationship
- LAPS deployment
- Machine addition quota to domain

3.

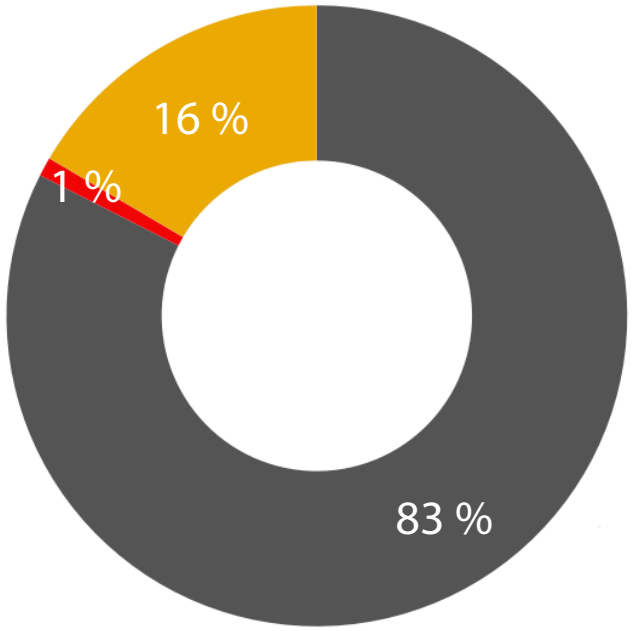
ACL

- ACL Owner
- ACE on object (users, computer, domain, group, GPO, OU, container)

4.

SOME INTERESTING KPI IN PRODUCTION (80 NTDS ANALYZED / 428 600 ACCOUNTS) IN 2023

Passwords strength

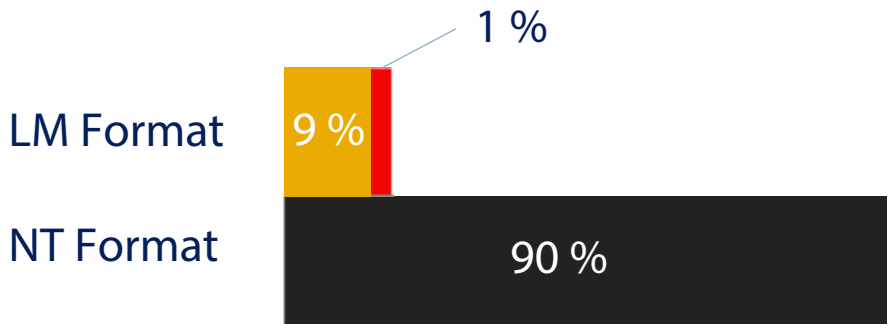


Non Cracked Passwords

Admin / High-Privileged Cracked Passwords

Cracked Passwords

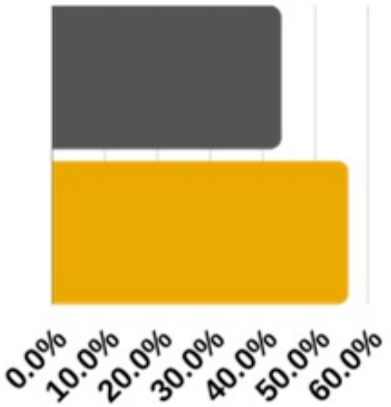
Passwords storage format



Passwords cracked

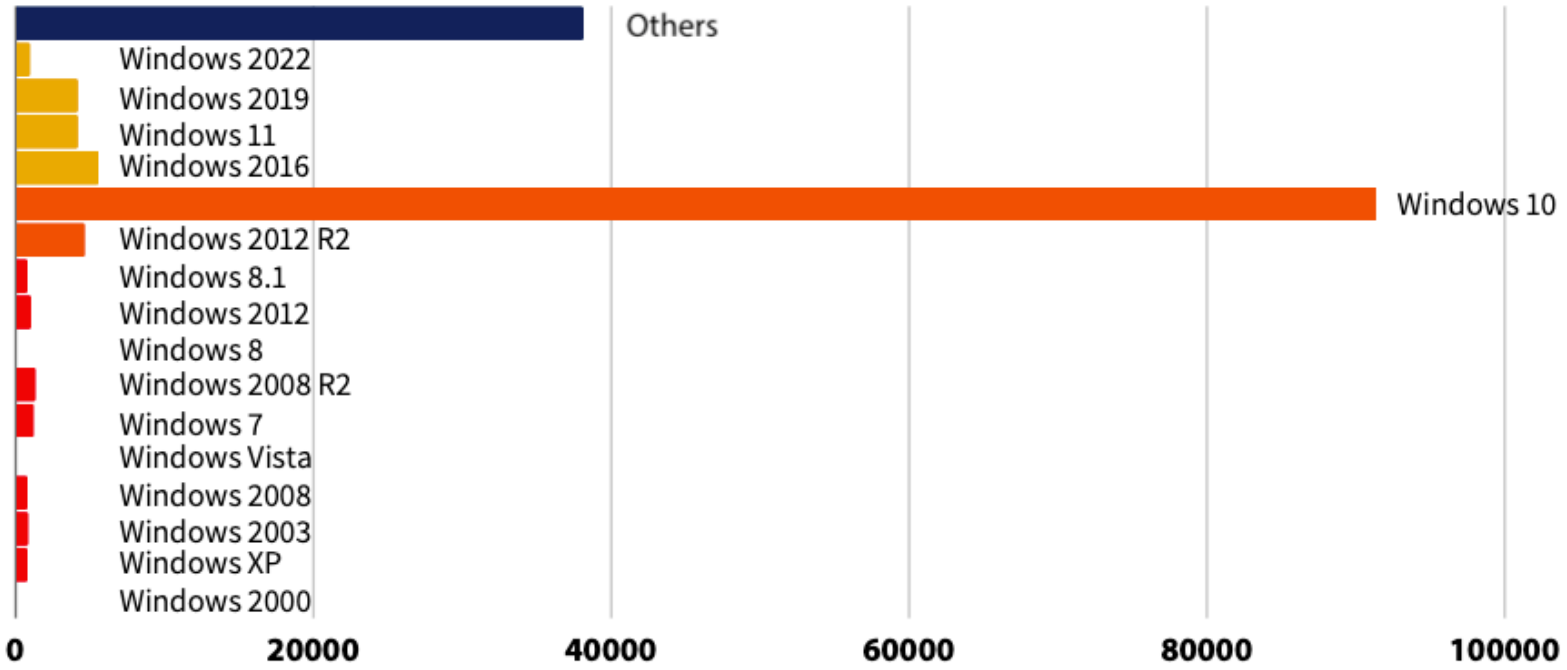
Changed < 6 months ago

Changed > 6 months ago



SOME INTERESTING KPI IN PRODUCTION (80 NTDS ANALYZED / 428 600 ACCOUNTS)

Systems repartition



WHAT YOU WILL NOT FIND IN THE NTDS

- Event logs
- GPO details
- Windows patching level
- Windows systems configuration

FORMAT & STRUCTURE

3

xmco

TROOPERS

NTDS EXTRACTION

```
PS C:\> ntdsutil.exe 'ac i ntds' 'ifm' 'create full c:\XMCO' q q
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\XMCO
Creating snapshot...
Snapshot set {4fc48a60-f407-404e-99c7-89b723eabb35} generated successfully.
Snapshot {124c6f39-0d9e-4b27-82fd-38e4be2ea893} mounted as C:\$SNAP_202403111454_VOLUMEC$\
Snapshot {124c6f39-0d9e-4b27-82fd-38e4be2ea893} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_202403111454_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: c:\XMCO\Active Directory\ntds.dit

                Defragmentation  Status (omplete)

          0    10   20   30   40   50   60   70   80   90  100
          |----|----|----|----|----|----|----|----|----|----|
          ..................................................

Copying registry files...
Copying c:\XMCO\registry\SYSTEM
Copying c:\XMCO\registry\SECURITY
Snapshot {124c6f39-0d9e-4b27-82fd-38e4be2ea893} unmounted.
IFM media created successfully in c:\XMCO
ifm: q
C:\Windows\system32\ntdsutil.exe: q
PS C:\>
```

- NTDSUtil.exe dumps 3 files:
  - NTDS.dit
  - SYSTEM
  - SECURITY

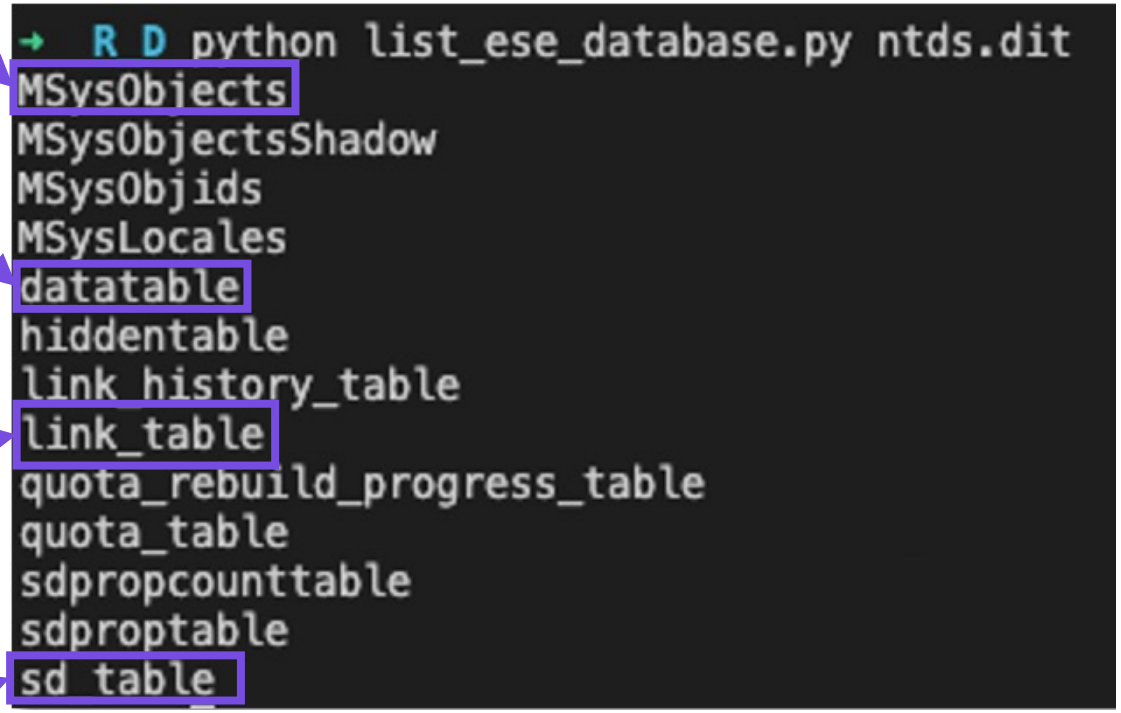
NTDS TABLES

Contains the metadata of the Datatable

Contains information about the objects in the directory (users, machines, groups, etc.)

Contains the links between objects in the directory (e.g. the MemberOf attribute of a user contains links to the groups to which they belong).

Contains the security descriptors of objects contained in the Active Directory, the Access Control Entries (ACE).



DATATABLE

- Contains more than 3500 columns. It's a "hollow" table

- Columns (or attributes) are filled out or empty depending on the object type (entry)
  - Domain
  - DNS zone
  - User account
  - Machine account
  - Container
  - OU
  - Etc.

DATATABLE

- Entry could represent a user, a computer, a domain, a container, etc.
- The name of the column (or object attributes) is not understandable, not documented, and the number of column is not fixed

Datatable					
	ATTm131532	ATTm590045	ATTq589898	ATTj589993	ATTj589922
Entry	lastLogon				
Entry	sAMAccountName				
Entry	lockoutTime				
Entry		Aragorn			
Entry		Legolas			
Entry		Gimli			
Entry	...	...	-864000000000000		
Entry	...	...		24	
Entry	...	...			513

SD TABLE

SD Table			
sd_id	sd_hash	sd_refcount	sd_value
1	bytes	1	bytes
...	bytes	14	bytes
127	bytes	10	bytes
...	...	...	...
296	bytes	2	bytes
297	bytes	32	bytes
...	...	...	...

SECURITY_DESCRIPTOR structure (winnt.h)

Article • 02/22/2024 Feedback

In this article

- Syntax
- Members
- Remarks
- Requirements
- See also

The **SECURITY_DESCRIPTOR** structure contains the security information associated with an object. Applications use this structure to set and query an object's security status.

Because the internal format of a security descriptor can vary, we recommend that applications not modify the **SECURITY_DESCRIPTOR** structure directly. For creating and manipulating a security descriptor, use the functions listed in See Also.

Syntax

C++ Copy

```
typedef struct _SECURITY_DESCRIPTOR {
  BYTE                        Revision;
  BYTE                        Sbz1;
  SECURITY_DESCRIPTOR_CONTROL Control;
  PSID                        Owner;
  PSID                        Group;
  PACL                        Sacl;
  PACL                        Dacl;
} SECURITY_DESCRIPTOR, *PISECURITY_DESCRIPTOR;
```

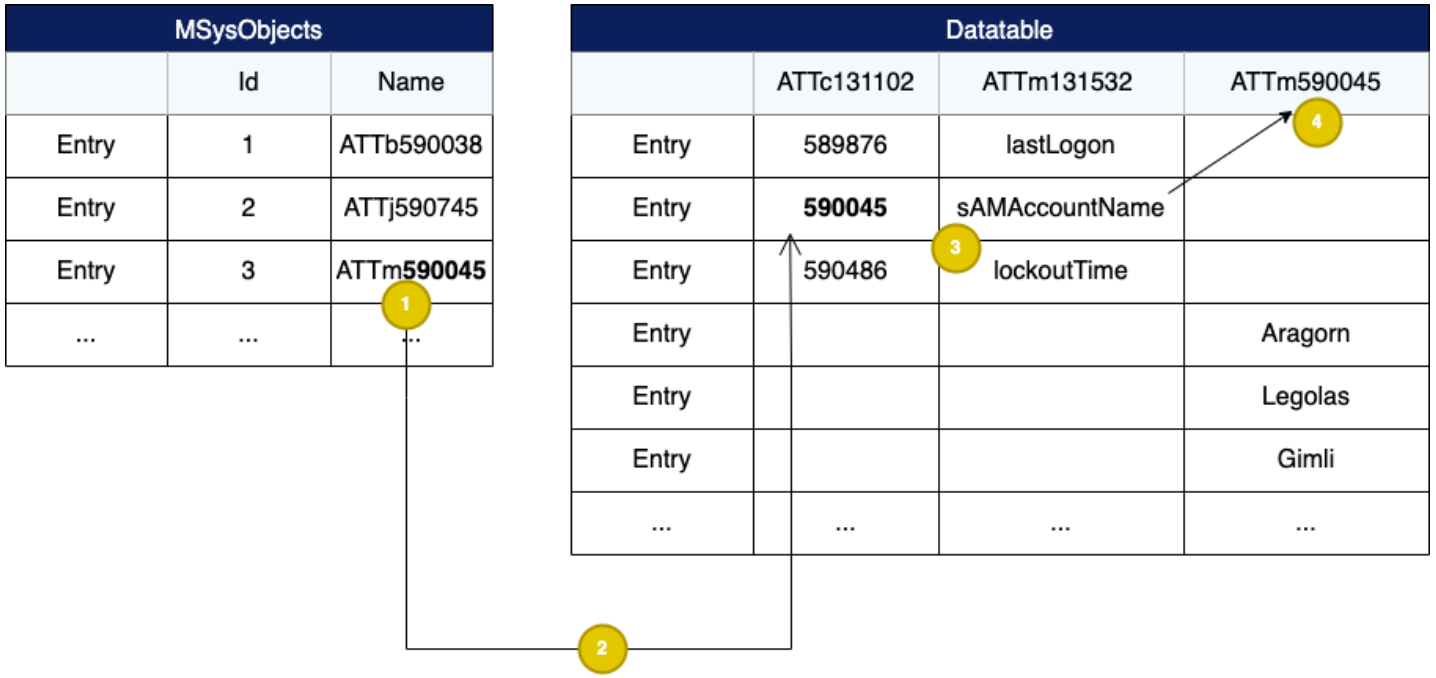
PARSING & DATA

4

xmco

TROOPERS

HOW TO GET OBJECT ATTRIBUTE NAMES ?



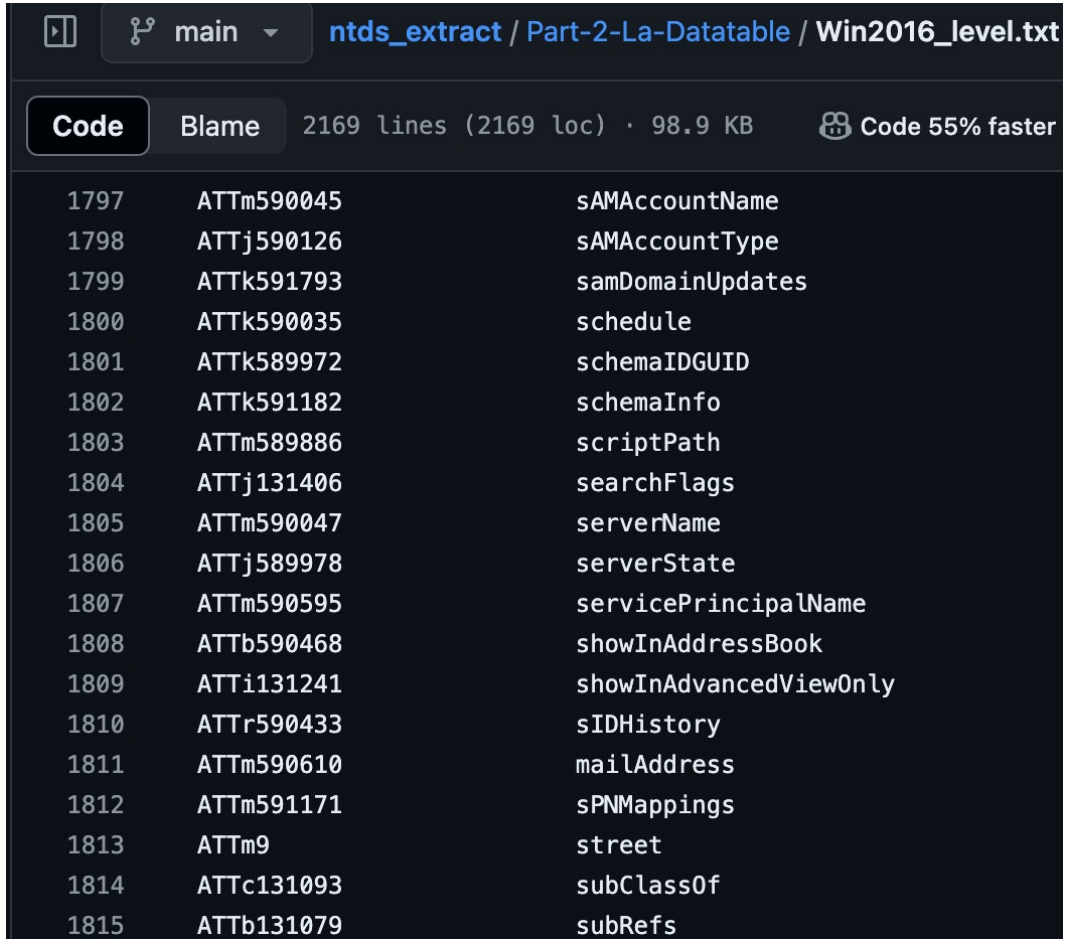
1. Search in the MSysObjects table for the columns present in the NTDS datatable

2. Search in the datatable for the **attribute ID**

3. After identifying the object (the row) that contains the ID and look at the ATTm131532 column for the corresponding **LDAP name**

4. ATTm590045 == sAMAccoutName

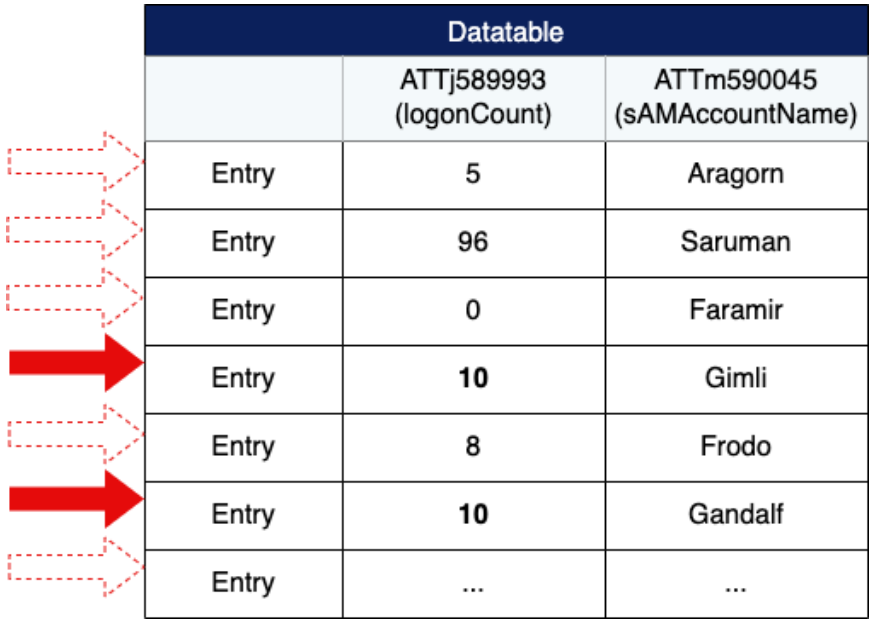
CORRESPONDING ATTRIBUTE NAMES



Attributes reference (Win2016) : https://github.com/xmco/ntds_extract/blob/main/Part-2-La-Datatable/Win2016_level.txt

GET DATA

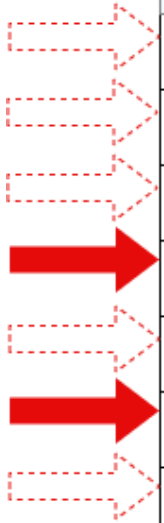
- External libraries exist in different languages (C, Python, Go)
- None of them has a « SQL » like language allowing to perform efficient requests
  - Eg. Get all the accounts where the logonCount == 10
- Require to iterate over all the record to find our matching pattern
- Reading process could be long (from 10 seconds to several hours)
- For further purpose, data must be stored in another database (SQL or NoSQL)
- If working exclusively on Windows environment, ManagedEsent library or DSInternal framework could be used



Datatable		
	ATTj589993 (logonCount)	ATTm590045 (sAMAccountName)
Entry	5	Aragorn
Entry	96	Saruman
Entry	0	Faramir
Entry	**10**	Gimli
Entry	8	Frodo
Entry	**10**	Gandalf
Entry	...	...

GET DATA

```python
def read_record(file_path):
    with open(file_path, "rb") as fh:
        db = EseDB(fh)
        datatable = db.table("datatable")
        logon_count_attr = 'ATTj589993'
        sam_account_name_attr = 'ATTm590045'
        for record in datatable.records():
            if (logon_count_value := record.get(logon_count_attr)) is not None:
                if logon_count_value == 10:
                    sam_account_name_value = record.get(sam_account_name_attr)
                    print(f"{sam_account_name_value} logged in {logon_count_value} times")
                    # Gimli logged in 10 times
```



Datatable		
	ATTj589993 (logonCount)	ATTm590045 (sAMAccountName)
Entry	5	Aragorn
Entry	96	Saruman
Entry	0	Faramir
Entry	**10**	Gimli
Entry	8	Frodo
Entry	**10**	Gandalf
Entry	...	...

Use of dissect.esedb[1] module

[1] https://github.com/fox-it/dissect.esedb

DATA TYPE

- Here a sample of column type (JET_CODTYP) available :

Identifier	Description	Example
JET_coltypLong	Integer 32-bit signed	ATT**j**589836 (badPwdCount)
JET_coltypLongText	Large text	ATT**m**13 (description)
JET_coltypLongBinary	Large binary data	ATT**k**589914 (NT hash)
JET_coltypCurrency	Currency	ATT**q**589884 (lockoutDuration)

The letter in the column name helps identifying the type.

SOME INTERESTING COLUMN NAMES (OR ATTRIBUTES)

Column Name	LDAP Attribute	Description
ATTm590045	sAMAccountName	User name, machine name, domain name (trust), etc.
ATTm13	description	Description of the object
ATTj589836	badPwdCount	Number of incorrect password attempts
ATTm590187	operatingSystem	Name of the operating system (e.g., Windows 10 Professional) when it's a machine account
ATTm590188	operatingSystemVersion	Version of the operating system (e.g., 10.0 (19042)) when it's a machine account
ATTr589970	objectSID	The object's identifier (SID)

ENCRYPTED ATTRIBUTES

Sensitive attributes such as LM hash and NT hash are encrypted

Column Name	LDAP Attribute	Description
ATTk589879	dBCSPwd	Account LM Hash
ATTk589914	unicodePwd	Account NT Hash
ATTk590689	pekList	PEK (Password Encryption Key)
ATTk589949	supplementalCredentials	Potential other passwords

ENCRYPTED ATTRIBUTES

- The SYSTEM hive is required to get the BOOTKEY and decrypt[1] them
- Each domain controller of the domain has its own BOOTKEY

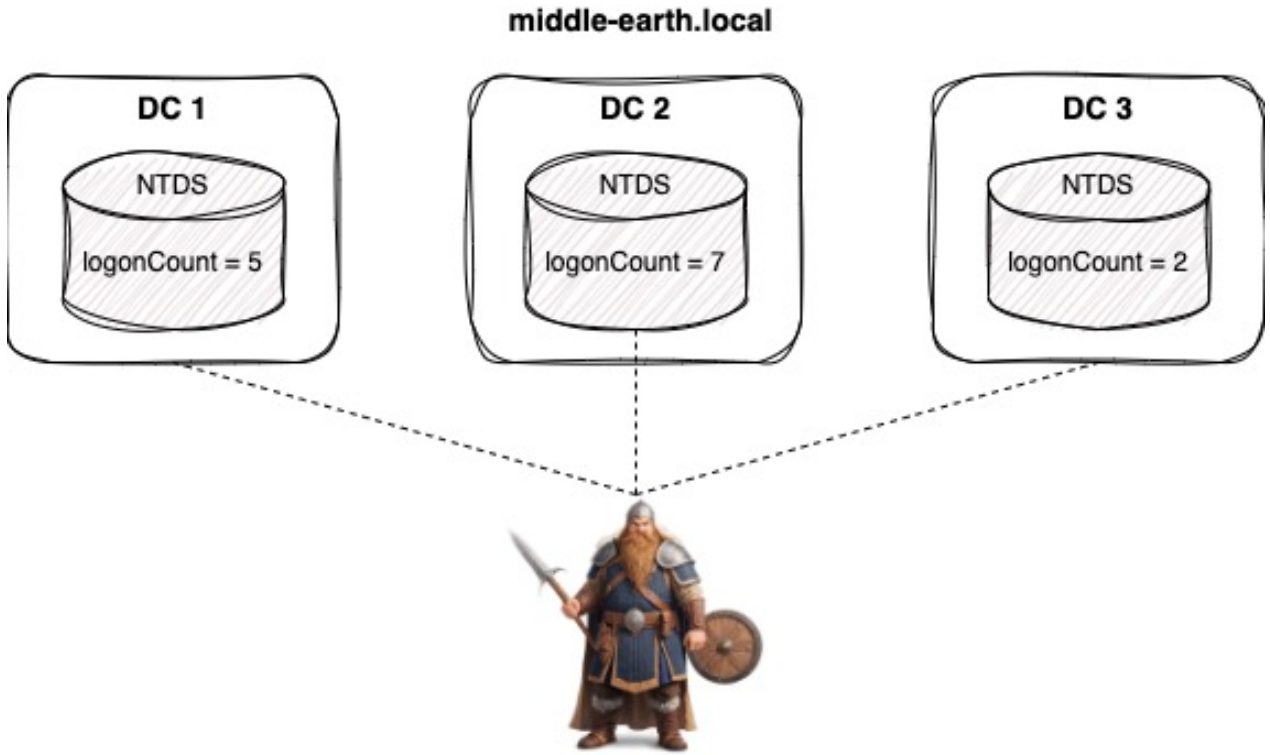
Exception

- LAPS legacy password are stored in cleartext. BOOTKEY is not required!
- Microsoft fixed it in the new version of LAPS released in April 2023 (encrypted with MS-GKDI[2]).

[1] Full decryption process : https://www.xmco.fr/en/active-directory-en/ntds-3-password-hashes/
[2] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-ada2/b6ea7b78-64da-48d3-87cb-2cff378e4597

NON-REPLICATED ATTRIBUTES

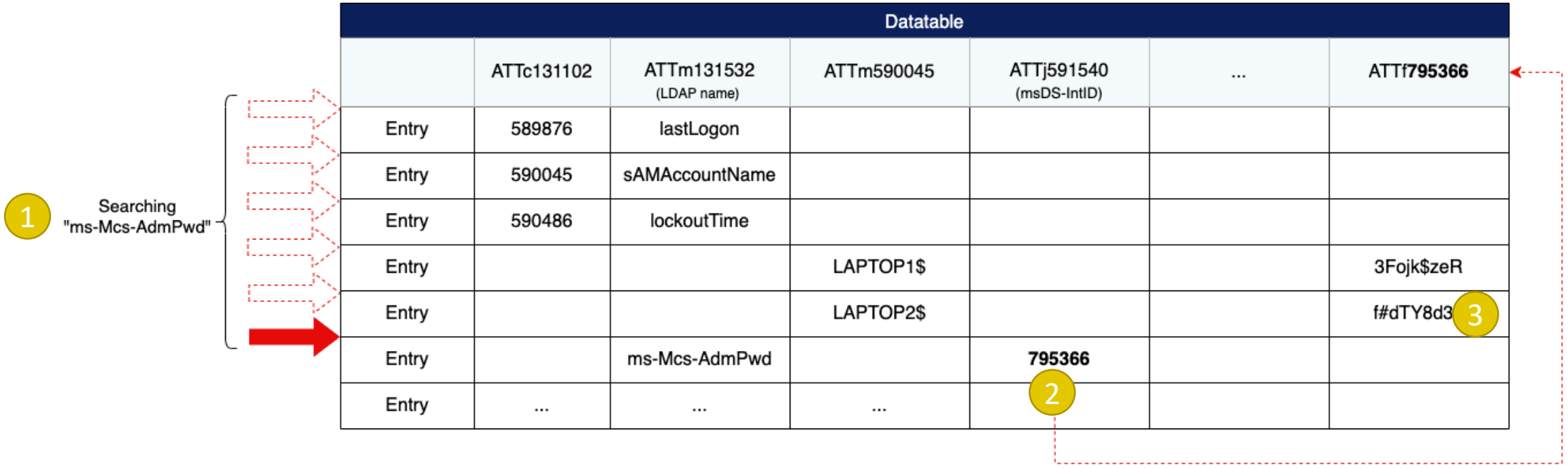
- Some attributes are not replicated between domain controllers : *logonCount, lastLogon, badPwdCount, etc.*
- Each domain controller maintains the attribute value



DYNAMIC ATTRIBUTES

Some column or attributes do not have a fixed value since they are not present at the AD installation. It is applicable for Exchange attributes, LAPS attributes and any software that modifies the AD schema.

Example: LAPS legacy password attributes *ms-Mcs-AdmPwd*



1. Searching for LDAP name ms-Mcs-AdmPwd
2. Get the msDS-IntId attribute to identify the column where the LAPS password is stored
3. Get LAPS password

SD TABLE & ACL

5

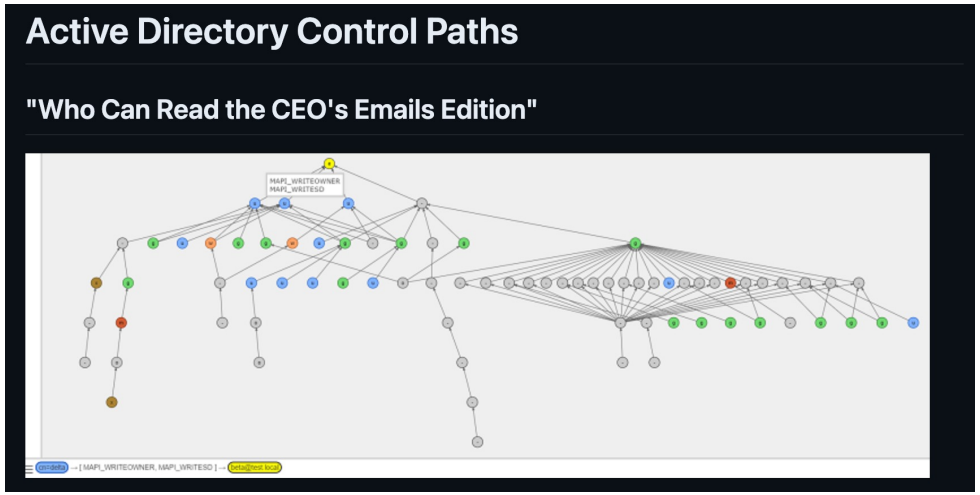
xmco

TROOPERS

EXISTING TOOLS TO ANALYZE ACL/ACE BASED ON THE NTDS

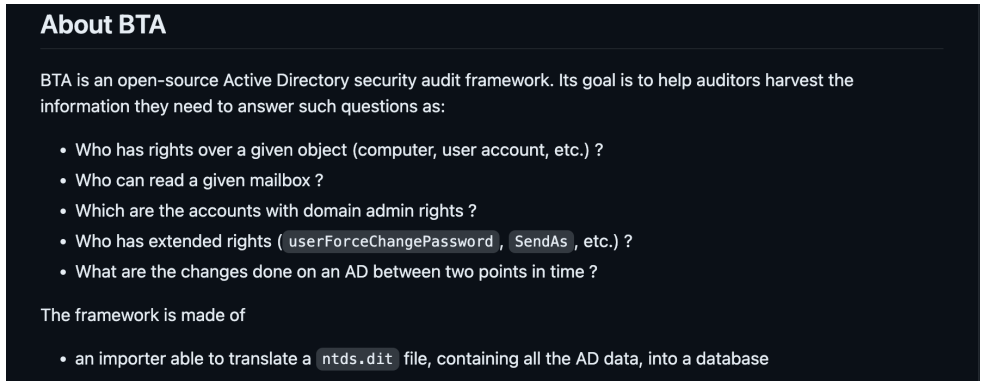
AD Control Paths

- Coded in C
- Not maintained anymore ? (last commit on Jan 7, 2020)



BTA

- Python-based
- Not maintained anymore (last commit on Mar 25, 2016)
- But it is a **GREAT** base to understand the parsing of the SD structures, many thanks to airbus!



Github: https://github.com/ANSSI-FR/AD-permissions/ then https://github.com/ANSSI-FR/AD-control-paths
Whitepaper: https://www.sstic.org/media/SSTIC2012/SSTIC-actes/audit_ace_active_directory/SSTIC2012-Article-audit_ace_active_directory-de-drouas_capillon_2.pdf and https://www.sstic.org/media/SSTIC2014/SSTIC-actes/chemins_de_controle_active_directory/SSTIC2014-Article-chemins_de_controle_active_directory-gras_bouillot.pdf

Github: https://github.com/airbus-seclab/bta

MANDATORY REFERENCES ON THE ACL/ACE TOPIC

- *"An ACE Up the Sleeve: Designing Active Directory DACL Backdoors"* by Andy Robbins (@_wald0) and Will Schroeder (@harmj0y) from SpecterOps
- Whitepaper: https://www.blackhat.com/docs/us-17/wednesday/us-17-Robbins-An-ACE-Up-The-Sleeve-Designing-Active-Directory-DACL-Backdoors-wp.pdf
- Huge work that helped a lot in getting into the topic, thanks!

The plan

- Goal : make a python script that stores the parsed data into CSV files ;
- Reuse the code-base of BTA if possible ;
- Use Dissect (https://github.com/fox-it/dissect.esedb) to parse the NTDS ;
- Use LOAD (https://github.com/0xBallpoint/LOAD) for testing purposes ;
- (future) use BloodHound terminology (in terms of ACE edges) but in a list format, and/or with graphs to be able to integrate it to pentest/audit reports.

FINDING THE ACL FOR AN OBJECT : THE LINK BETWEEN THE DATATABLE AND THE SD TABLE

Datatable

ATTc131102 (Attribute-ID)	ATTm131532 (LDAP-Display-Name of attributes)	ATTp131353 (ntSecurityDescriptor attribute)	ATTm590045 (sAMAccountName attribute)
589876	lastLogon	...	...
590045	sAMAccountName	158	gandalf
590486	lockoutTime	...	
		169	gimili
		...	...
		296	aragorn
...	...	...	...

SD Table

sd_id	sd_hash	sd_refcount	sd_value
1	bytes	1	bytes
...	bytes	14	bytes
127	bytes	10	bytes
...	...	...	...
296	bytes	2	bytes
297	bytes	32	bytes
...	...	...	...

PARSING THE SD TABLE WITH DISSECT TO EXTRACT RAW VALUES

```python
from dissect.esedb import EseDB

def parse_sd_values(file_path):
    output = []
    with open(file_path, "rb") as fh:
        db = EseDB(fh)
        for table in db.tables():
            print(table)
        sd_table = db.table("sd_table")
        for sd_record in sd_table.records():
            try:
                sd_id = sd_record.get("sd_id")
                sd_value = sd_record.get("sd_value")
                print("sd_id: %s, sd_value: %s" % (sd_id,sd_value))
            except Exception as err:
                print("parse_sd_values error: ",err)
        return
```

Retrieving the sd_id and sd_value raw value in the SD Table with dissect

```
sd_id: 5, sd_value: b'\x01\x00\x14\x80\xe8\x02\x00\x00\x04\x03\x00\x00\x14\x00\x00\x00\x94\x00\x00\x00\x04\x00\x80\x00\x04\
x00\x00\x00\x02@\x14\x00 \x00\x0c\x00\x01\x01\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x02@\x18\x00\x00\x01\x00\x00\x01\x02\
x00\x00\x00\x00\x00\x05 \x00\x00\x00 \x02\x00\x00\x02@$\x00\x00\x01\x00\x00\x01\x05\x00\x00\x00\x00\x00\x05\x15\x00\x00\x00
2\xef\xe3\xd3\xc6\x94\x8b\x90\x94\xc9M\x1f\x01\x02\x00\x00\x07@(\x00\x00\x01\x00\x00\x01\x00\x00\x00VQ\xecE~\xdb\xbbG\xb5?\
xdb\xeb-\x03\xc4\x0f\x01\x01\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x04\x00T\x02\x0f\x00\x00\x00\x05\x00(\x00\x00\x01\x00\
x00\x01\x00\x00\x00\xaa\xf61\x11\x07\x9c\xd1\x11\xf7\x9f\x00\xc0O\xc2\xdc\xd2\x01\x01\x00\x00\x00\x00\x00\x05\t\x00\x00\x00
\x05\x00(\x00\x00\x01\x00\x00\x01\x00\x00\x00\xab\xf61\x11\x07\x9c\xd1\x11\xf7\x9f\x00\xc0O\xc2\xdc\xd2\x01\x01\x00\x00\x00
\x00\x00\x05\t\x00\x00\x00\x05\x00(\x00\x00\x01\x00\x00\x01\x00\x00\x00\xac\xf61\x11\x07\x9c\xd1\x11\xf7\x9f\x00\xc0O\xc2\x
dc\xd2\x01\x01\x00\x00\x00\x00\x00\x05\t\x00\x00\x00\x05\x00,\x00\x00\x01\x00\x00\x01\x00\x00\x00\xaa\xf61\x11\x07\x9c\xd1\
x11\xf7\x9f\x0                                                                                    \x00\x00\x01
\x00\x00\x00\x                                                                                    \x00 \x02\x0
0\x00\x05\x00,                                                                                    1\x02\x00\x0
0\x00\x00\x00\                                                                                    \x00\x00\x00
\x00\x02$\x00\                                                                                    xc9M\x1f\x07
\x02\x00\x00\x00\x00\x14\x00\xff\x01\x0f\x00\x01\x01\x00\x00\x00\x00\x00\x05\x12\x00\x00\x00\x00\n$\x00\xbd\x01\x0f\x00\x01
\x05\x00\x00\x00\x00\x00\x05\x15\x00\x00\x002\xef\xe3\xd3\xc6\x94\x8b\x90\x94\xc9M\x1f\x00\x02\x00\x00\x05\x00(\x00\x00\x01
```

Example of raw value of the sd_value attribute retrieved with dissect in the SD Table

SD_VALUE STRUCTURE

SECURITY_DESCRIPTOR structure (winnt.h)

Article • 02/22/2024 Feedback

In this article

The **SECURITY_DESCRIPTOR** structure contains the security information associated with an object. Applications use this structure to set and query an object's security status.

Because the internal format of a security descriptor can vary, we recommend that applications not modify the **SECURITY_DESCRIPTOR** structure directly. For creating and manipulating a security descriptor, use the functions listed in See Also.

Syntax

C++ Copy

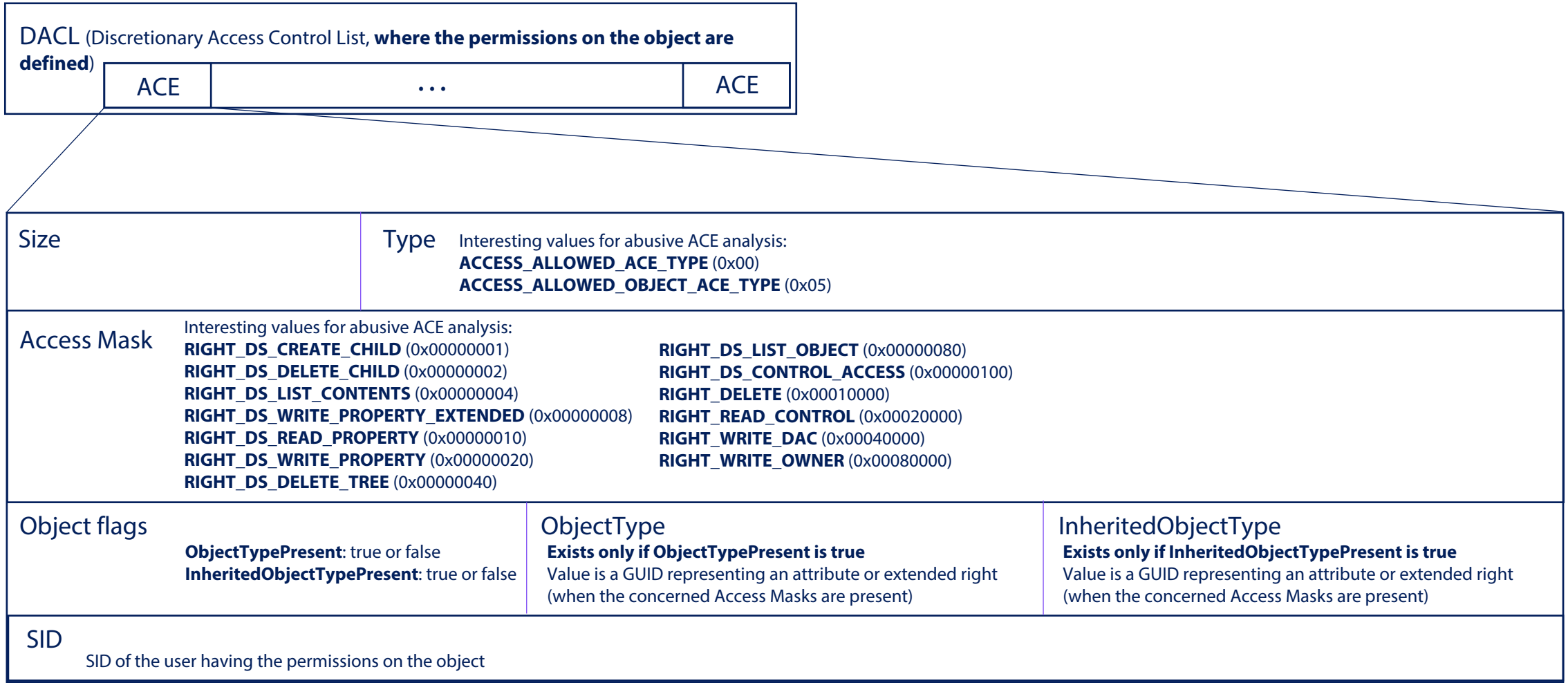
```
typedef struct _SECURITY_DESCRIPTOR {
  BYTE                        Revision;
  BYTE                        Sbz1;
  SECURITY_DESCRIPTOR_CONTROL Control;
  PSID                        Owner;
  PSID                        Group;
  PACL                        Sacl;
  PACL                        Dacl;
} SECURITY_DESCRIPTOR, *PISECURITY_DESCRIPTOR;
```

Owner: S-1-5-21-<domain>-RID

DACL (Discretionary Access Control List, **where the permissions on the object are defined**)

ACE	...	ACE

ACE STRUCTURE



ACE Types : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dtyp/628ebb1d-c509-4ea0-a10f-77ef97ca4586
Details regarding the access masks : https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/990fb975-ab31-4bc1-8b75-5da132cd4584

ObjectType

- Present when specific ACE are set:
  - RIGHT_DS_CREATE_CHILD
  - RIGHT_DS_READ_PROPERTY
  - RIGHT_DS_WRITE_PROPERTY
  - RIGHT_DS_CONTROL_ACCESS (for extended rights)

- Examples:
  - AccessMask = RIGHT_DS_WRITE_PROPERTY + ObjectType = 5b47d60f-6090-40b2-9f37-2a4de88f3063 (msDS-KeyCredentialLink attribute)
    -> AddKeyCredentialLink permissions / Shadow Credentials attack (Whisker/pywhisker)

  - AccessMask = RIGHT_DS_CONTROL_ACCESS + ObjectType = 00299570-246d-11d0-a768-00aa006e0529 (“User-Force-Change-Password” extended right)
    -> Reset the password of the target without their consent

  - 00000000–0000–0000–0000–000000000000 or no GUID means rights on all properties/extended rights

Documentation regarding extended rights: https://learn.microsoft.com/en-us/windows/win32/adschema/extended-rights
Details regarding ObjectType not containing a GUID: https://learn.microsoft.com/en-us/windows/win32/api/iads/ne-iads-ads_rights_enum

PARSING THE SD_VALUE IN PRACTICE

```python
import struct

def sd_to_json(sd):
    jsd = {}
    try:
        rev,_sbz,rctrl,owner,group,saclofs,daclofs = struct.unpack_from("<BBHIIII", sd)
        ctrl = ControlFlags(rctrl)
        jsd["Revision"] = rev
        jsd["Control"] = ctrl.to_json()
        if ctrl.SelfRelative:
            jsd["Owner"] = decode_sid(sd[owner:])
            jsd["Group"] = decode_sid(sd[group:])
            if ctrl.SACLPresent:
                jsd["SACL"] = acl_to_json(sd[saclofs:])
            if ctrl.DACLPresent:
                jsd["DACL"] = acl_to_json(sd[daclofs:])
    except struct.error as e:
        print("sd_to_json error: ",e)
    return jsd
```

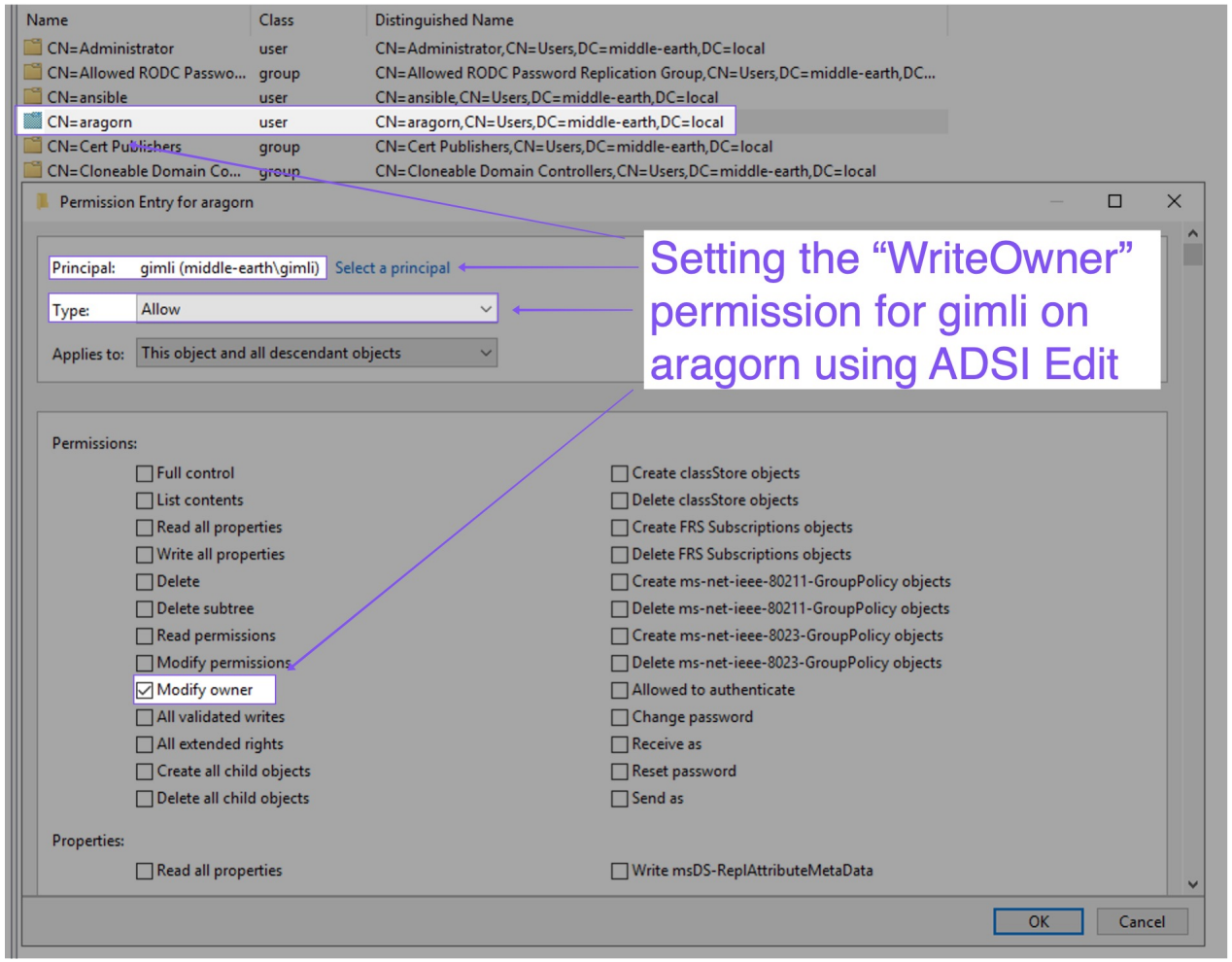
Code from BTA to parse the raw sd_value and convert it to json

This actually involves additional magic that we may not have time to talk about today

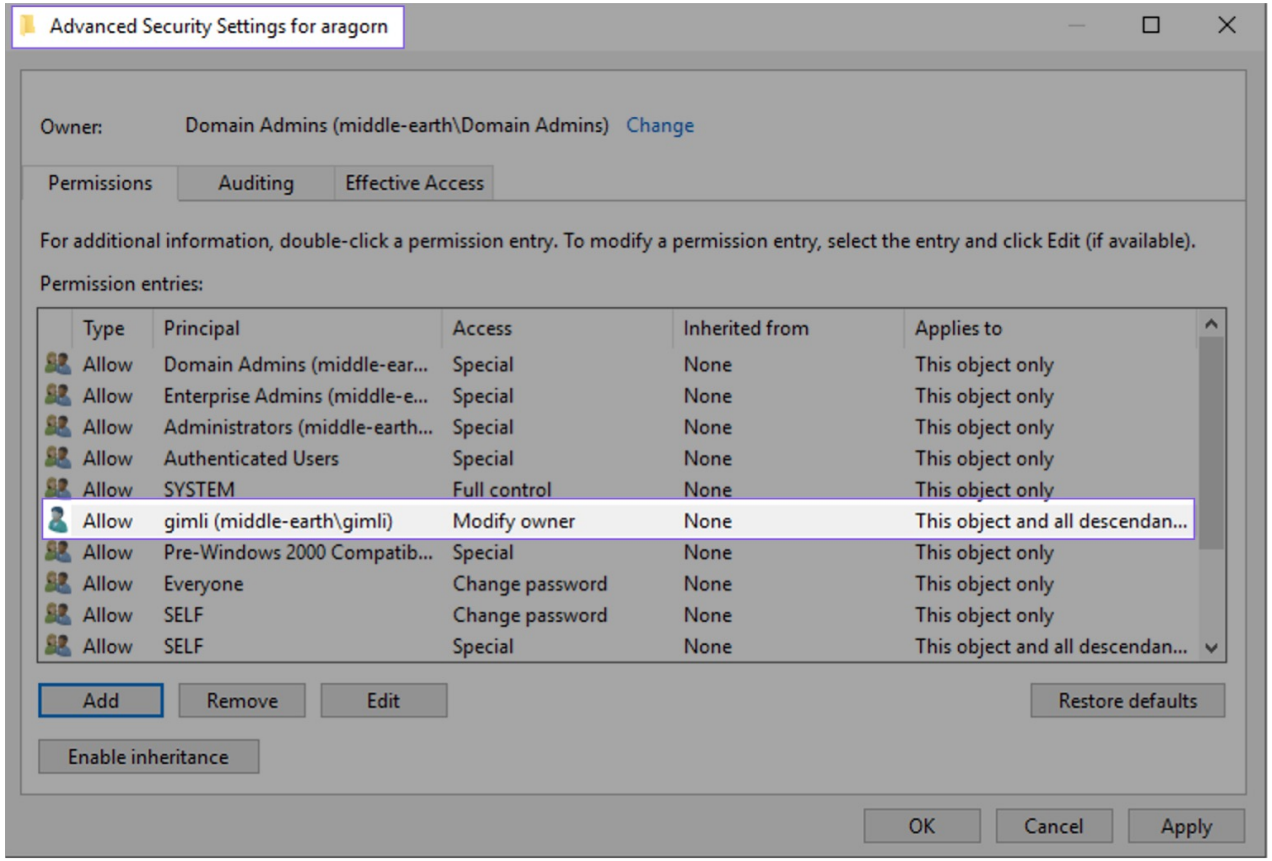
```
[12/03/24 5:29:54] → work_acl_from_ntds git:(master) ✗ python sd_to_json.py
raw sd_value: b'\x01\x00\x14\x8c\x18\x02\x00\x004\x02\x00\x00\x14\x00\x00\x00\xb4\x00\x00\x00\x04\x00\xa0\x00\x03\x00\x00\x0
0\x07Z(\x00 \x00\x04\x00\x02\x00\x00\x00\xc2;\x0e\xf3\xf0\x9f\xd1\x11\xb6\x03\x00\x00\xf8\x03g\xc1\x01\x01\x00\x00\x00\x00\x
00\x01\x00\x00\x00\x00\x07Z8\x00 \x00\x00\x00\x03\x00\x00\x00\xbe;\x0e\xf3\xf0\x9f\xd1\x11\xb6\x03\x00\x00\xf8\x03g\xc1\xa5z
\x96\xbf\xe6\r\xd0\x11\xa2\x85\x00\xaa\x000I\xe2\x01\x01\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x07Z8\x00 \x00\x00\x00\x03\
x00\x00\x00\xbf;\x0e\xf3\xf0\x9f\xd1\x11\xb6\x03\x00\x00\xf8\x03g\xc1\xa5z\x96\xbf\xe6\r\xd0\x11\xa2\x85\x00\xaa\x000I\xe2\x
01\x01\x00\x00\x00\x00\x01\x00\x00\x00\x00\x04\x00d\x01\r\x00\x00\x00\x00\x00\x14\x00\x94\x00\x02\x00\x01\x01\x00\x00\x0
0\x00\x00\x00\x05\x0b\x00\x00\x00\x00\x00$\x00\xbf\x01\x0e\x00\x01\x05\x00\x00\x00\x00\x00\x05\x15\x00\x00\x002\xef\xe3\xd3\xc6\
x94\x8b\x90\x94\xc9M\x1f\x00\x02\x00\x00\x00\x00\x14\x00\xff\x01\x0f\x00\x01\x01\x00\x00\x00\x00\x00\x05\x12\x00\x00\x00\x00
\x00\x14\x00\x94\x00\x02\x00\x01\x01\x00\x00\x00\x00\x00\x05\t\x00\x00\x00\x05\x12(\x00\x00\x01\x00\x00\x01\x00\x00\x00\x8f\
xfd\xac\xed\xb3\xff\xd1\x11\xb4\x1d\x00\xa0\xc9h\xf99\x01\x01\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x05\x12(\x00\x00\x01\x
00\x00\x01\x00\x00\x00\x8f\xfd\xac\xed\xb3\xff\xd1\x11\xb4\x1d\x00\xa0\xc9h\xf99\x01\x01\x00\x00\x00\x00\x00\x05\x0b\x00\x00
\x00\x00\x12$\x00\xff\x00\x0f\x00\x01\x05\x00\x00\x00\x00\x00\x05\x15\x00\x00\x002\xef\xe3\xd3\xc6\x94\x8b\x90\x94\xc9M\x1f\
x00\x02\x00\x00\x00\x12$\x00\xff\x00\x0f\x00\x01\x05\x00\x00\x00\x00\x00\x05\x15\x00\x00\x002\xef\xe3\xd3\xc6\x94\x8b\x90\x9
4\xc9M\x1f\x07\x02\x00\x00\x00\x12\x14\x00\x14\x00\x02\x00\x01\x01\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x12\x14\x00\x
94\x00\x02\x00\x01\x01\x00\x00\x00\x00\x00\x05\t\x00\x00\x00\x00\x12\x14\x00\x94\x00\x02\x00\x01\x01\x00\x00\x00\x00\x00\x05
\x0b\x00\x00\x00\x00\x12\x14\x00\xff\x00\x0f\x00\x01\x01\x00\x00\x00\x00\x00\x05\x12\x00\x00\x00\x00\x1a\x14\x00\xff\x00\x0f
\x00\x01\x01\x00\x00\x00\x00\x00\x03\x00\x00\x00\x00\x01\x05\x00\x00\x00\x00\x00\x05\x15\x00\x00\x002\xef\xe3\xd3\xc6\x94\x8
b\x90\x94\xc9M\x1f\x00\x02\x00\x00\x01\x05\x00\x00\x00\x00\x00\x05\x15\x00\x00\x002\xef\xe3\xd3\xc6\x94\x8b\x90\x94\xc9M\x1f
\x00\x02\x00\x00'
{'Revision': 1, 'Control': {'value': 35860, 'flags': {'OwnerDefaulted': False, 'GroupDefaulted': False, 'DACLPresent': True,
 'DACLDefaulted': False, 'SACLPresent': True, 'SACLDefaulted': False, 'DACLAutoInheritReq': False, 'SACLAutoInheritReq': Fal
se, 'DACLAutoInherited': True, 'SACLAutoInherited': True, 'DACLProtected': False, 'SACLProtected': False, 'SelfRelative': Tr
ue}}, 'Owner': 'S-1-5-21-3554                                                    425066694-525191572-512', 'SAC
L': {'Revision': 4, 'Size': 1                                                    : {'value': 90, 'flags': {'Obj
ectInheritAce': False, 'Conta                                                    lyAce': True, 'InheritedAce':
True, 'SuccessfulAccessAceFlag': True, 'FailedAccessAceFlag': False}}, 'Size': 40, 'AccessMask': {'value': 262176, 'flags':
{'GenericRead': False, 'GenericWrite': False, 'GenericExecute': False, 'GenericAll': False, 'AcessSystemAcl': False, 'Delete
': False, 'ReadControl': False, 'WriteDAC': True, 'WriteOwner': False, 'Synchronize': False, 'AccessSystemSecurity': False,
'MaximumAllowed': False, 'StandardsRightsRequired': False, 'StandardRightsAll': False, 'SpecificRightsAll': False, 'ADSRight
DSCreateChild': False, 'ADSRightDSDeleteChild': False, 'ADSRightACTRLDSList': False, 'ADSRightDSSelf': False, 'ADSRightDSRea
dProp': False, 'ADSRightDSWriteProp': True, 'ADSRightDSDeleteTree': False, 'ADSRightDSListObject': False, 'ADSRightDSControl
Access': False}}, 'ObjectFlags': {'value': 2, 'flags': {'ObjectTypePresent': False, 'InheritedObjectTypePresent': True}}, 'I
nheritedObjectType': 'f30e3bc2-9ff0-11d1-b603-0000f80367c1', 'SID': 'S-1-1-0'}, {'Type': 'SystemAuditObject', 'Flags': {'val
```

raw sd_value parsed to JSON

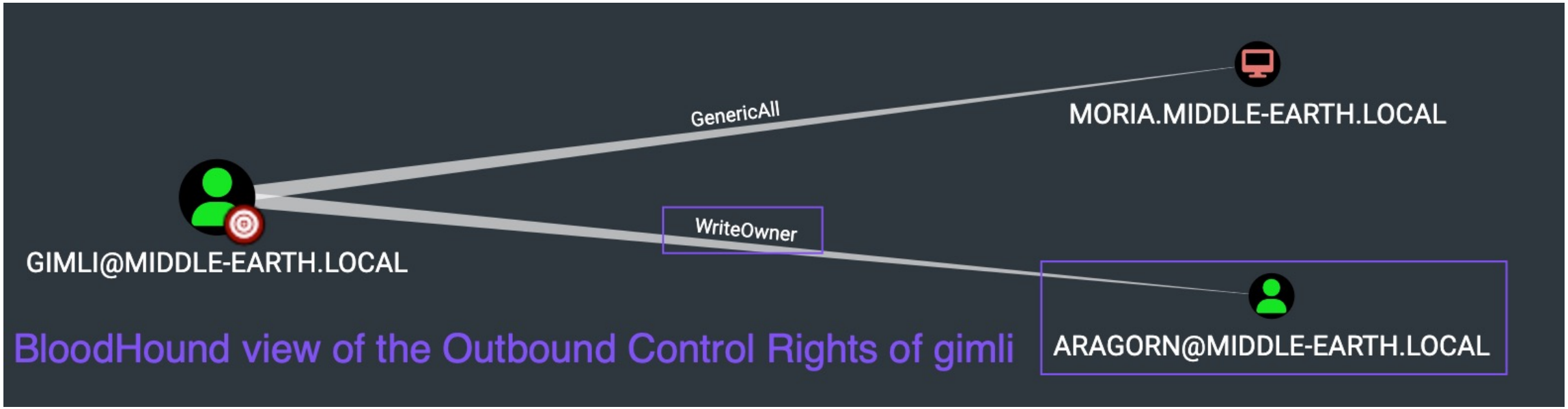
PRACTICAL EXAMPLE: CONFIGURING AN ACE WITH ADSI EDIT



Setting the "WriteOwner" permission for gimli on aragorn using ADSI Edit



PRACTICAL EXAMPLE: LOOKING AT FAMILIAR TOOLS



PRACTICAL EXAMPLE: FINDING THE ACL FOR ARAGORN IN THE PARSED DATATABLE AND SD TABLE

```
[9/06/24 2:03:55] ➜  out git:(master) ✗ head -n 1 report_users.csv
domain,login,uid,SID,hashlm,hashnt,kerberosKeys,clearTextPwds,servicePrincipalName,p
asscount,description,created,change,lastlogon,lastLogonTimestamp,expired,lastpasswor
erIdentity,comment,remark,os,version,operatingSystemServicePack,msMcsAdmPwd,msMcsAdm
xpirationTime,sIDHistory,uniq,uniq_dom,adminCount,ntSecurityDescriptor
[9/06/24 2:04:02] ➜  out git:(master) ✗ rg -i aragorn report_users.csv
15:middle-earth.local,aragorn,4050,S-1-5-21-3679311831-575282908-4264792444-1126,aac
aa341,"aes256-cts-hmac-sha1-96:87db0936c6db3f51568458c26c3cacfdc3601524e02864462dc3b
3517b00215f8d , des-cbc-md5:7a3eb946628cb9a1",,,S-1-5-21-3679311831-575282908-42647
64792444-1118 , S-1-5-21-3679311831-575282908-4264792444-513 , S-1-5-32-545",4,"midc
ship , middle-earth.local|Domain Users , middle-earth.local|Users",0,0,True King of
00,2024-03-11 11:01,"NORMAL_ACCOUNT , DONT_EXPIRE_PASSWORD",,,,,,,,,,,,,1,1,1,296
[9/06/24 2:04:53] ➜  out git:(master) ✗
```

```
[9/06/24 2:10:51] ➜  scripts git:(master) ✗ cat parse_sdtable.json| jq '.[] | select(.sd_id == 296) | .DACL'
{
  "Revision": 4,
  "Size": 516,
  "Num ACEs": 14,
  "ACEs": [
    {
      "Raw Type": 5,
      "Raw Flags": 0,
      "Size": 56,
      "Raw Access Required": 48,
      "Type": "Access Allowed Object",
      "Access Required": {
        "Generic Read": false,
        "Generic Write": false,
        "Generic Execute": false,
        "Generic All": false,
        "Maximum Allowed": false,
        "Access SACL": false,
        "Synchronise": false,
        "Write Owner": false,
        "Write DAC": false,
        "Read Control": false,
        "Delete": false,
        "Ads Control Access": false,
        "Ads List Object": false,
        "Ads Delete Tree": false,
        "Ads Write Prop": true,
        "Ads Read Prop": true,
        "Ads Self Write": false,
        "Ads List": false,
        "Ads Delete Child": false,
        "Ads Create Child": false
      },
      "Raw Object Flags": 1,
      "Object Flags": {
        "Object Type Present": false,
        "Inherited Object Type Present": true
      },
      "Inherited GUID": "{bf967a7f-0de6-11d0-a285-00aa003049e2}",
      "SID": "S-1-5-21-3679311831-575282908-4264792444-517"
    },
    {
```

Looking at the SD Table for the ACL configured on aragorn (sd_id = ntSecurityDescriptor)

PRACTICAL EXAMPLE: FINDING THE ACL CONFIGURED WITH ADSI EDIT IN THE SD TABLE


```
{
  "Raw Type": 0,
  "Raw Flags": 2,
  "Size": 36,
  "Raw Access Required": 524288,
  "Type": "Access Allowed",
  "Access Required": {
    "Generic Read": false,
    "Generic Write": false,
    "Generic Execute": false,
    "Generic All": false,
    "Maximum Allowed": false,
    "Access SACL": false,
    "Synchronise": false,
    "Write Owner": true,
    "Write DAC": false,
    "Read Control": false,
    "Delete": false,
    "Ads Control Access": false,
    "Ads List Object": false,
    "Ads Delete Tree": false,
    "Ads Write Prop": false,
    "Ads Read Prop": false,
    "Ads Self Write": false,
    "Ads List": false,
    "Ads Delete Child": false,
    "Ads Create Child": false
  },
  "SID": "S-1-5-21-3679311831-575282908-4264792444-1124"
},
```


ACE entry in the SD Table showing that the user with SID S-1-5-21-3679311831-575282908-4264792444-1124 (gimli) has “Write Owner” permissions on aragorn (sd_id = ntSecurityDescriptor = 296)

```
[9/06/24 2:24:35] → out git:(master) ✗ head -n 1 report_users.csv
domain,login,uid,SID,hashlm,hashnt,kerberosKeys,clearTextPwds,servicePrincipalName,primar
ogoncount,badpasscount,description,created,change,lastlogon,lastLogonTimestamp,expired,la
To,allowedToActOnBehalfOfOtherIdentity,comment,remark,os,version,operatingSystemServicePa
me,msLAPSPasswordEncrypted,msLAPSPasswordExpirationTime,sIDHistory,uniq,uniq_dom,adminCou
[9/06/24 2:24:40] → out git:(master) ✗ rg -i gimli report_users.csv
13:middle-earth.local,gimli,4048,S-1-5-21-3679311831-575282908-4264792444-1124,aad3b435b5
27d691e2ba8575467,"aes256-cts-hmac-sha1-96:6dd758a74b6703161e05db1457a614dccba668d8b1bf1d
ha1-96:9ea114febb432f79621f38e10135c116 , des-cbc-md5:6b01a2199d8c012f",,,S-1
1-3679311831-575282908-4264792444-1118 , S-1-5-21-3679311831-575282908-426479
lowship , middle-earth.local|Domain Users , middle-earth.local|Users",0,0,-,2
0,2024-03-11 11:01,"NORMAL_ACCOUNT , DONT_EXPIRE_PASSWORD",,,,,,,,,,,,1,1,,1
[9/06/24 2:24:44] → out git:(master) ✗
```

PUTTING IT ALL TOGETHER : AUTOMATED ANALYSIS OF THE POTENTIALLY DANGEROUS ACL

```
[7/06/24 2:29:50] ➜  scripts git:(master) ✗ cat out/report_suspicious_acl.csv
domain,sd_id,trustee,trustee_sid,permission,objectGuid,inheritedobjectGuid,target,owner,owner_sid
middle-earth.local,246,gimli,S-1-5-21-3679311831-575282908-4264792444-1124,Write Owner,,,MORIA$,Domain Admins,S-1-5-21-3679311831-575282908-4264792444-512
middle-earth.local,246,gimli,S-1-5-21-3679311831-575282908-4264792444-1124,Write DAC,,,MORIA$,Domain Admins,S-1-5-21-3679311831-575282908-4264792444-512
middle-earth.local,246,gimli,S-1-5-21-3679311831-575282908-4264792444-1124,AllExtendedRights,,,MORIA$,Domain Admins,S-1-5-21-3679311831-575282908-4264792444-512
middle-earth.local,291,boromir,S-1-5-21-3679311831-575282908-4264792444-1123,Ads Control Access,,Reset Password,faramir,Domain Admins,S-1-5-21-3679311831-575282908-4264792444-512
middle-earth.local,294,boromir,S-1-5-21-3679311831-575282908-4264792444-1123,Write DAC,,,Men,ansible,S-1-5-21-3679311831-575282908-4264792444-1009
middle-earth.local,295,gimli,S-1-5-21-3679311831-575282908-4264792444-1124,Write Owner,,,aragorn,Domain Admins,S-1-5-21-3679311831-575282908-4264792444-512
```

Final words

6

FINAL WORDS

- Release of an open-source script : https://github.com/xmco/parse_ntds
- All in one ! Dump in CSV files :
  - Domain information (Password policy)
  - Users (including LM:NT hashes)
  - Groups
  - OUs and Containers
  - Suspicious ACL

```
[24/06/24 9:33:16] → scripts git:(master) python parse_ntds.py -h
usage: parse_ntds.py [-h] -f NTDS_FILE -s SYSTEM_FILE [-d DOMAIN] [-o OUTPUT_DIR] [-v] [--dump-all] [--dump-users] [--dump-groups] [--dump-trusts] [--dump-domains] [--dump-ou] [--dump-acl]

Python script to parse a NTDS and dumps its content to CSV files

options:
  -h, --help            show this help message and exit
  -f NTDS_FILE, --file NTDS_FILE
                        Absolute path to the ntds.dit file
  -s SYSTEM_FILE, --system SYSTEM_FILE
                        Absolute path to the SYSTEM file
  -d DOMAIN, --domain DOMAIN
                        Domain name
  -o OUTPUT_DIR, --output OUTPUT_DIR
                        Output directory. Default is the current directory
  -v, --verbose         Increase output verbosity to DEBUG level
  --dump-all            Dump all data except ACL (default)
  --dump-users          Dump user data
  --dump-groups         Dump group data
  --dump-trusts         Dump trust data
  --dump-domains        Dump domain data
  --dump-ou             Dump OU/container data
  --dump-acl            Dump ACL data
```

FINAL WORDS

```
[24/06/24 9:37:08] ➜  scripts git:(master) python parse_ntds.py -f ../../../../ntds/ntds_LOAD_WO.dit -s ../../../../ntds/SYSTEM_LOAD_WO -o troopers_output --dump-
all --dump-acl -d middle-earth.local
[+] Parsing datatable...
[+] Parsing datatable done.
Execution time of parse_datable: 9.1710 seconds
[+] Parsing linktable...
[+] Parsing linktable done.
[+] Dumping domains to CSV file...
Execution time of dump_csv_domain: 0.0022 seconds
[+] Creating sqlite Correlations table...
[+] Sqlite Correlations table created.
Execution time of dump_sqlite_correlations: 0.1565 seconds
[+] Dumping data to CSV files...
[+] Dumping users to CSV file...
[+] Users CSV file created.
Execution time of dump_csv_user: 0.0455 seconds
[+] Dumping groups to CSV file...
[+] Groups CSV file created.
Execution time of dump_csv_group: 0.0021 seconds
[+] Dumping trusts to CSV file...
[+] Trusts CSV file created.
Execution time of dump_csv_trust: 0.0006 seconds
[+] Dumping domains to CSV file...
Execution time of dump_csv_domain: 0.0017 seconds
[+] Dumping OU and containers to CSV file...
[+] OU and containers CSV file created.
Execution time of dump_csv_ou_container: 0.0499 seconds
[+] Dumping suspicious ACEs to CSV file...
Execution time of list_ace_raw: 0.0000 seconds
[+] Parsing sdtable...
[+] Parsing sdtable done.
Execution time of parse_sdtable: 0.1646 seconds
[+] Suspicious ACEs CSV file created.
Execution time of dump_csv_suspicious_acl: 1.2243 seconds
[+] All CSV files created.
```

```
[24/06/24 9:37:36] ➜  scripts git:(master) l troopers_output
total 592
drwxr-xr-x   9 fduthu  staff   306B Jun 24 09:37 .
drwxr-xr-x  13 fduthu  staff   442B Jun 24 09:35 ..
-rw-r--r--   1 fduthu  staff   765B Jun 24 09:37 report_domains.csv
-rw-r--r--   1 fduthu  staff    21K Jun 24 09:37 report_groups.csv
-rw-r--r--   1 fduthu  staff    65K Jun 24 09:37 report_ou_containers.csv
-rw-r--r--   1 fduthu  staff   1.0K Jun 24 09:37 report_suspicious_acl.csv
-rw-r--r--   1 fduthu  staff   4.4K Jun 24 09:37 report_trusts.csv
-rw-r--r--   1 fduthu  staff    24K Jun 24 09:37 report_users.csv
-rw-r--r--   1 fduthu  staff   160K Jun 24 09:37 sqlite.db
```

FINAL WORDS

- Still some work to do on the script :
  - A few bugs to be corrected
    - Related to dissect (XPRESS10 compression not supported at the moment)
    - OUs and Containers
    - Decrypting new LAPS attributes
    - Adding mapping GenericX permissions
- Ideas for the future :
  - Dockerize everything
  - Use a proper db to store the objects
  - **Interactions with BloodHound** (e.g. compatible output to draw graphs, adding interesting data to the neo4j db such as password reuse, cracked passwords)
  - Support for ACL on GPO
  - Collecting more interesting attributes (e.g. Bitlocker keys, more dynamic attributes that could contain sensitive information such as password hashes)

We deliver cybersecurity expertise