Active Directory Enumeration

...with LDAP

Who am I?

- Stephen Bradshaw
- Principal Security Consultant in Offensive Security Team @Seek
- Currently do internally-focused offensive security work including:
  - pentesting,
  - vulnerability hunting,
  - purple and red teaming,
  - building and maintaining tools and systems for offensive security work

Talk Overview

- How Active Directory (AD) and LDAP are interrelated.
- The types of AD information that can be gathered from LDAP and the security relevance.
- Technical details of the process of AD data collection with LDAP for attackers.
- What AD enumeration tools make use of LDAP, how do they work and what are their limitations.
- Approaches for detecting LDAP enumeration (time dependent).

What's LDAP, how's it relate to AD, why should you care?

- LDAP ("Lightweight Directory Access Protocol") provides a protocol/standard for accessing network object information from a directory service
- Each Domain Controller that stores Active Directory (AD) data runs an LDAP server by default
- The relevance for attackers: Default settings allow any authenticated user to quickly gather a large amount of information useful for attacks in one place

Information you can collect

Via LDAP, you can collect the following information from AD:

- Permission relationships between objects (e.g. which objects can act in what way on other objects)
- Domain users and their details
- Domain groups and memberships
- Computers joined to the domain
- Group Policy Objects (links/applicable objects)
- Trust relationships between different domains
- AD Certificate Services information

Facilitated attacks (partial list)

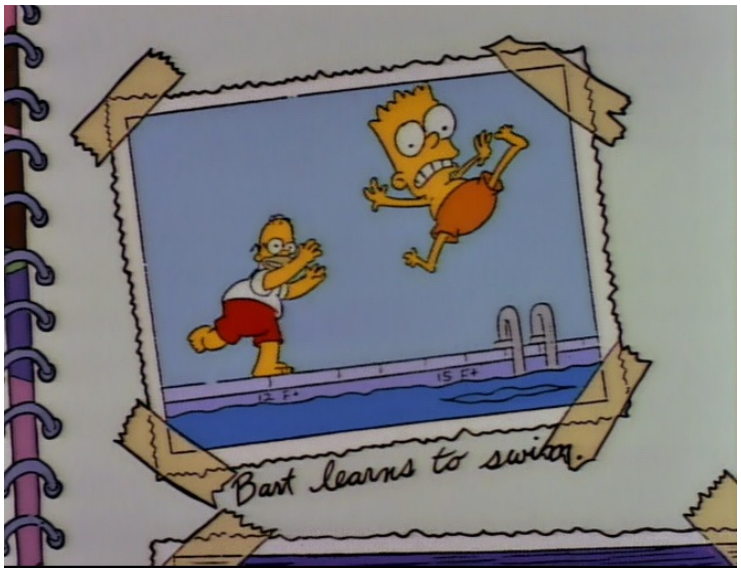
The information you can collect can facilitate /identify potential for attacks such as:

- Kerberoasting and AS-REP roasting
- Gold and silver tickets
- AD ACL abuse
- DCSync
- Domain and Forest Trust abuse
- Constrained and Unconstrained delegation attacks
- AD Certificate Services attacks
- Recovery of alternate password types (e.g. LAPS)

Data collection challenges

- The best way to go about this data collection is to use custom tools designed for the purpose
- However, whilst many of these tools are great there are cases where they fail awkwardly, leaving you without easy options to perform some great attacks
- Given the value of the information that's available, its great to understand how the process works so you can troubleshoot tools and have alternate collection strategies

Diving in to the details of LDAP data collection!



Data collection process overview

The various tools that do LDAP/AD enumeration facilitate the following steps:

1. **Discovering** Domain Controllers.
2. **Connecting** and authenticating to a Domain Controller.
3. **Querying** and parsing the useful data.
4. **Interpreting** and presenting the data.

The following slides will show what's involved at each stage of the process above.

Discovering Domain Controllers

AD Global Catalog Domain Controllers

- Active Directory has a concept of a Global Catalog, that allows a domain controller to provide information on any object in the Forest.
- AD servers with the feature are known as "Global Catalog servers".
- These are the domain controllers we want to preference for connection

DC Discovery method #1 Port Scan

Each AD Domain Controller (DC) exposes a unique set of ports that allow them to be easily identified by port scanning:

- Domain Controllers will normally be exposing ports such as TCP/UDP 53(DNS), TCP 88(Kerberos), TCP 389(LDAP), TCP 445(CIFS), TCP 636(LDAPS), and usually also TCP 3389(RDP), TCP 5985(WinRM) and TCP 5986 (HTTPS WinRM)
- You ***can*** check if a particular AD server is a Global Catalog server by performing an anonymous LDAP bind and checking the info.other.isGlobalCatalogReady property in its LDAP server information (however there are easier approaches).

This method provides a good starting approach for finding individual domain controllers on an unfamiliar network.

DC discovery method #2 DNS

If you know the DNS name of the target AD domain, and can perform resolutions against its DNS servers (directly or via forwarding from another DNS server), we can identify domain controllers in two different ways (given a domain name of example.com)

- Looking up the base domain name (e.g. example.com) will list IP addresses of ALL domain controllers
- Looking up gc._msdcs.example.com will list the IP addresses of **Global Catalog** servers specifically

This method can identify ALL DCs quickly, however requires some domain information and specific access to work.

Finding the DNS domain name?

Compromise of a domain joined computer will allow you to get this information natively, otherwise there are three approaches you can use to get this info remotely from domain joined systems without the requirement for authentication:

- Inferred from the names in certificates presented by TLS services (e.g. LDAPS, RDP, HTTPS, WINRMS),
- Collecting LDAP anonymous server info; or
- Using CIFS enumeration tools like ntlm_challenger (https://github.com/nopfor/ntlm_challenger)

Best results come from combining approaches

Suggested discovery approach

1. **Get the DNS domain name of the domain.** Port scan til you find a domain member system and use one of the afore mentioned domain name discovery techniques if you don't have this.
2. **Find a DNS server that allows you to query the domains DNS database.** Port scan for a DNS server if you don't have this.
3. **Enumerate IPs of Global Catalog servers and Domain Controllers using the DNS technique.**
4. **Port scan these IPs, preferencing the Global Catalogs to find a server you can talk LDAP to (TCP ports 389/636).**
5. **$$$ PROFIT $$$** 🤑

Connecting to LDAP

Connecting to AD LDAP Servers

- Basic communication ***only*** requires connecting to TCP port 389 or 636 (LDAPS)
- Authentication options include:
  - anonymous,
  - NTLM (password or hash),
  - “simple” (username and unencrypted password) and
  - Kerberos
- Kerberos is a more complex protocol and requires use of the servers DNS name (not IP address) and TCP port 88 access if you don’t have an appropriate Service Ticket (ST) already cached
- Basic server information can be retrieved with an anonymous bind, although insecure configurations can reveal more. By default, authentication will be required for getting most of the useful detail

This might seem like belaboring simple details, but becomes relevant if access to a DC is subject to restriction, e.g. via port forward or tunneling through a C2 implant

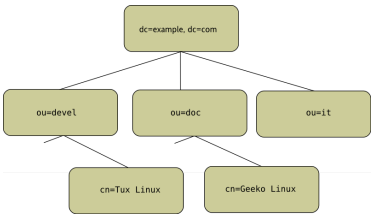
LDAPS connectivity…

- Listens on TCP port 636, requires TLS/SSL negotiation to create a tunnel for communications
- The LDAP SSL ports on DCs sometimes can sometimes have no server certificate configured which means LDAPS wont work even though the port is open
- Some servers can be configured to require LDAPS for certain authentication methods – if you get an access denied with valid credentials on LDAP/389 try with LDAPS/636

Querying LDAP

LDAP querying

- LDAP stores objects in a hierarchical, tree like structure
- It has its own query language, with a unique syntax
- Queries filter objects based on their attribute values



Simple query example

- Queries are comprised of one or more query conditions, which are grouped with brackets
- A simple query to return group objects (where attribute objectClass is “group”) is

(objectClass=group)

More complex example

- More complex queries can be formed by using multiple query conditions, value wildcards (*), appropriate grouping and logical operators like AND (&) and OR (|) and negation(!)
- The logical operator (& | !) goes ***before*** the conditions it applies to and has its own bracketed group
- Example: To return objects that have objectClass of "user" **AND** objectCategory of "msDS-GroupManagedServiceAccount":

(&(objectClass=user)(objectCategory=msDS-GroupManagedServiceAccount))

Syntax breakdown

(&(objectClass=user)(objectCategory=msDS-GroupManagedServiceAccount))

Broken down:

- **AND** (&(condition1)(condition2)(conditionN…))
- (objectClass=user)
- (objectCategory=msDS-GroupManagedServiceAccount)

Query special comparators

Capability name	OID
LDAP_MATCHING_RULE_BIT_AND	1.2.840.113556.1.4.803
LDAP_MATCHING_RULE_BIT_OR	1.2.840.113556.1.4.804
LDAP_MATCHING_RULE_TRANSITIVE_EVAL	1.2.840.113556.1.4.1941
LDAP_MATCHING_RULE_DN_WITH_DATA	1.2.840.113556.1.4.2253

Kerberoastable users

(&(objectClass=user)(servicePrincipalName=*)(!(cn=krbtgt))(!(userAccountControl:1.2.840.113556.1.4.803:=2)))

More examples here: https://podalirius.net/en/articles/useful-ldap-queries-for-windows-active-directory-pentesting/

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/4e638665-f466-4597-93c4-12f2ebfabab5

LDAP_MATCHING_RULE_DN_WITH_DATA - Match on portion of binary value

LDAP_MATCHING_RULE_TRANSITIVE_EVAL - This rule provides recursive search of a link attribute.

Stores an integer that represents a number of properties relating to user account security
To find out of any individual property is present - bitwise and the numeric value against the attribute

Disabled Users = 2

More references with manual LDAP queries

An Introduction to Manual Active Directory Querying with Dsquery and Ldapsearch

- https://posts.specterops.io/an-introduction-to-manual-active-directory-querying-with-dsquery-and-ldapsearch-84943c13d7eb

Manual LDAP Querying: Part 2

- https://posts.specterops.io/manual-ldap-querying-part-2-8a65099e12e3

Interpreting/ presenting the data

This is the data for an LDAP object...

accountExpires	0
adminCount	1
badPasswordTime	29 July 2024, 10:59:31 am AEST (133666883...
badPwdCount	0
codePage	0
countryCode	0
description	Built-in account for administering the comput...
distinguishedName	CN=Administrator,CN=Users,DC=adds,DC=st...
dSCorePropagationData	1 Jan 1601, 10:00:00 am AEST (160101010000...
dSCorePropagationData	28 Aug 2024, 1:36:07 pm AEST (2024082803...
dSCorePropagationData	28 Aug 2024, 2:36:08 pm AEST (202408280...
isCriticalSystemObject	TRUE
lastLogoff	0
lastLogon	28 Aug 2024, 3:13:09 pm AEST (1336929558...
lastLogonTimestamp	18 Aug 2024, 10:21:41 pm AEST (1336845730...
logonCount	320
logonHours	Binary Data (21 Bytes)
memberOf	CN=Administrators,CN=Builtin,DC=adds,DC=s...
memberOf	CN=Domain Admins,CN=Users,DC=adds,DC=...
memberOf	CN=Enterprise Admins,CN=Users,DC=adds,D...
memberOf	CN=Group Policy Creator Owners,CN=Users,D...
memberOf	CN=Schema Admins,CN=Users,DC=adds,DC=...
memberOf	CN=SQLAdmins,OU=IT,DC=adds,DC=stephen...
msDS-SupportedEncryptionTypes	0
name	Administrator
objectGUID	{7dbf67a2-5499-4558-996f-4cf7d5f2db8e}
objectSid	S-1-5-21-1954401763-2568157779-1971089...
primaryGroupID	513
pwdLastSet	10 Apr 2024, 2:28:32 pm AEST (13357196912...
sAMAccountName	Administrator
sAMAccountType	805306368
userAccountControl	66048
uSNChanged	158820
uSNCreated	8196
whenChanged	18 Aug 2024, 10:21:41 pm AEST (2024081812...
whenCreated	26 Feb 2024, 5:34:56 pm AEDT (2024022606...

Interpreting data...

The difficulty of interpreting object attributes ranges from easy ☺ to hard ☹ :

- Most attributes are text or numeric – that's easy ☺
- Other attributes requires lookup tables or bitwise operations (e.g. flags) – this is slightly more difficult 😐
- Then there are binary attributes which need special parsing – hard ☹
- Establishing relationships between objects requires correlation and id to name mappings – also hard ☹

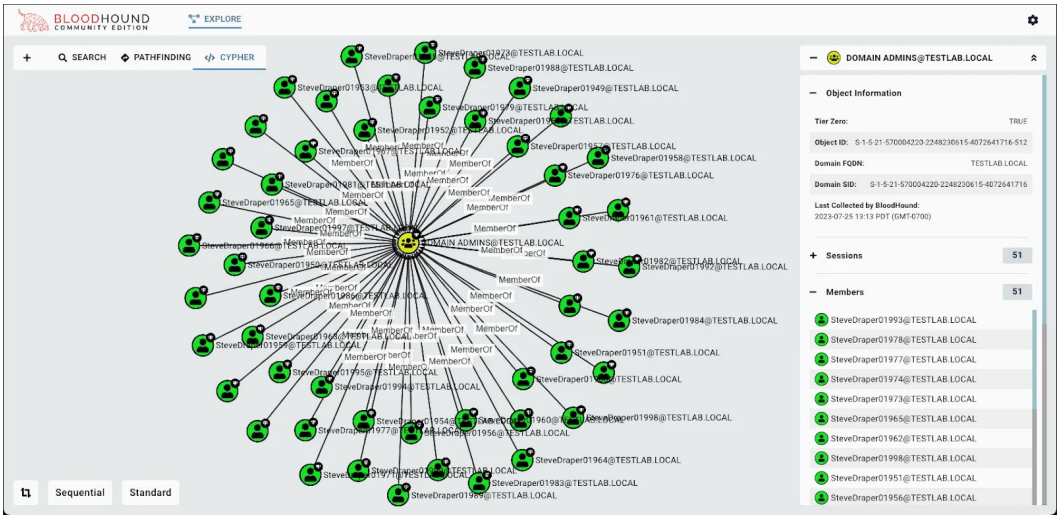
This is where using custom tools starts to become necessary.

Tools!

Bloodhound

- BloodHound is a tool created by SpecterOps to identify hidden relationships and attack paths within Active Directory (and now Azure) environments
- It works by collecting network information, storing it in a database, using graph theory to identify relationships between objects and presenting information using a specially designed UI

Bloodhound data display



Query interface!
Graphical display of objects showing interrelations!

Bloodhound collectors

- Bloodhound uses “collectors” to gather information from the AD environment, which it stores in JSON data files that can be ingested to populate the Bloodhound frontend
- Collectors also do a large amount of the data processing work required to create files in a consistent format that can be easily ingested by the Bloodhound front end

Bloodhound collectors and LDAP

- The majority of the information that Bloodhound collects (for AD data) is available to low privileged users, comes from LDAP, and can be collected rapidly from a single Domain Controller
- Some categories of information require higher privileges, require time consuming connections to a bunch of other servers in the domain and are more likely to fail
  - This includes information such as local computer group membership, currently logged on users, local session information, etc
- Bloodhound has collection “categories” which largely align with the collection technique - we can use this as a guide to identify what we can get quickly and effectively from LDAP

SharpHound collection categories

```
using System;

namespace SharpHoundCommonLib.Enums
{
    [Flags]
    public enum ResolvedCollectionMethod
    {
        None = 0,
        Group = 1,
        LocalAdmin = 1 << 1,
        GPOLocalGroup = 1 << 2,
        Session = 1 << 3,
        LoggedOn = 1 << 4,
        Trusts = 1 << 5,
        ACL = 1 << 6,
        Container = 1 << 7,
        RDP = 1 << 8,
        ObjectProps = 1 << 9,
        SessionLoop = 1 << 10,
        LoggedOnLoop = 1 << 11,
        DCOM = 1 << 12,
        SPNTargets = 1 << 13,
        PSRemote = 1 << 14,
        UserRights = 1 << 15,
        CARegistry = 1 << 16,
        DCRegistry = 1 << 17,
        CertServices = 1 << 18,
        LocalGroups = DCOM | RDP | LocalAdmin | PSRemote,
        ComputerOnly = LocalGroups | Session | UserRights | CARegistry | DCRegistry,
        DCOnly = ACL | Container | Group | ObjectProps | Trusts | GPOLocalGroup | CertServices,
        Default = Group | Session | Trusts | ACL | ObjectProps | LocalGroups | SPNTargets | Container | CertServices,
        All = Default | LoggedOn | GPOLocalGroup | UserRights | CARegistry | DCRegistry
    }
}
```

https://github.com/BloodHoundAD/SharpHoundCommon/blob/1ccdb773d3af19718f410d9795ca9977019b5a85/src/CommonLib/Enums/CollectionMethods.cs

SharpHound default collected data

There are a few types of objects BloodHound collects via LDAP by default that don't have a category, such as:

- Users and user like objects (objectClass of user or objectCategory of msDS-GroupManagedServiceAccount or msDS-ManagedServiceAccount)
- Computers (objectClass of computer)
- Domain and Forest information (objectClass of domain or crossRefContainer)

DCOnly == LDAP?

```
DCOnly = ACL | Container | Group | ObjectProps | Trusts | GPOLocalGroup | CertServices,
```

These components of the DCOnly category make use of LDAP:

- ACL
- Container
- Group
- ObjectProps
- Trusts
- ~~GPOLocalGroup~~
- CertServices

GPOLocalGoup involves parsing of GPO files on SYSVOL... (not LDAP)

Categories that are =~ LDAP object types

These “categories” basically just involve collecting types of LDAP objects:

- **Containers** – container objects including GPO links (container and organizationalUnit objects)
- **Group** – groups and their members (group objects)
- **Trusts** – information on trusts with other domains (trustedDomain objects)
- **CertServices** – AD Certificate Services configuration (certificationAuthority and pKIEnrollmentService and pKICertificateTemplate objects)

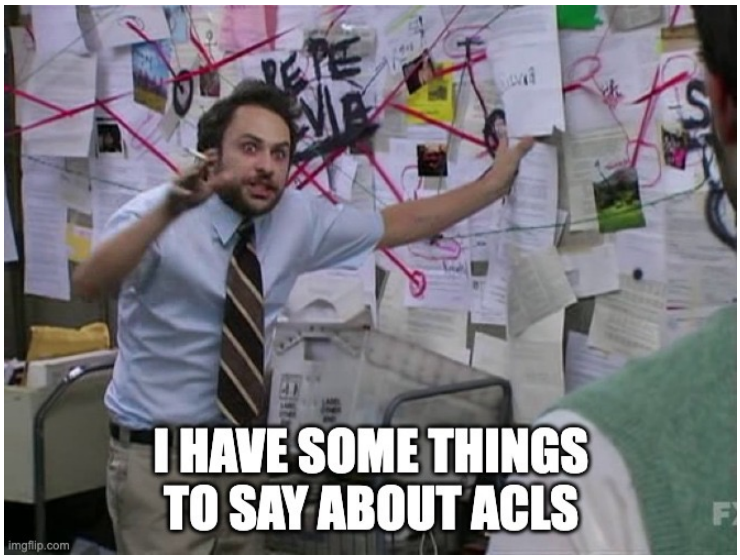
ObjectProps category

The ObjectProps category gathers more attribute data for collected objects.

- Bloodhound collectors limit the attributes returned for LDAP objects
- Doing ObjectProps collection gets additional attributes for objects such as User/Computer LastLogon or PwdLastSet
- Still doesn't mean everything is collected (there is another separate option for collecting all properties)

So far so good 😌

ACLs category 😱



ACLs = object security relationships 😒

- Describes ability to access/control the object – "fills" many of the links/edges in BH
- Required data comes from the nTSecurityDescriptor LDAP attribute on each securable object
- This is a ***binary*** field that contains multiple components. 💣🔥 Requires parsing 🔥💣
- The attribute contains:
  - Owner user and group – referenced by SID (can be resolved to name via a lookup)
  - The DACL (Discretionary Access Control List) – defines permissions to access the object
  - The SACL (System Access Control List) – defines the auditing configuration of the object – this component is not used

SECURITY_DESCRIPTOR

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dtyp/7d4dac05-9cef-4563-a058-f108abecce1d?redirectedfrom=MSDN

Control - control fields affecting things such as inheritance of the various other
Ownersid - security id of owner user
Groupsid - security id of owner group
Dacl - list of entries to control access
Sacl - list of entries to audit access, need the ACCESS_SYSTEM_SECURITY permission to get or set this,

Reading the security descriptor 🥺

- The nTSecurityDescriptor LDAP attribute stores the entire security descriptor structure (Owners, DACLs and SACLs), but not all of it is readable by all users
- The SACL list requires privileged ACCESS_SYSTEM_SECURITY permission to get/set, however other components (owner and DACLs) are readable by all authenticated users by default
- LDAP Control 1.2.840.113556.1.4.801 LDAP_SERVER_SD_FLAGS_OID, supported by AD, allows specifying an access mask to identify which security descriptor components you want to return in a given query
- When specified in an LDAP query with the 0x07 flag value this allows unprivileged users to get owner and DACL information despite not having permissions to read the SACL

DACL and ACEs - more parsing 😩

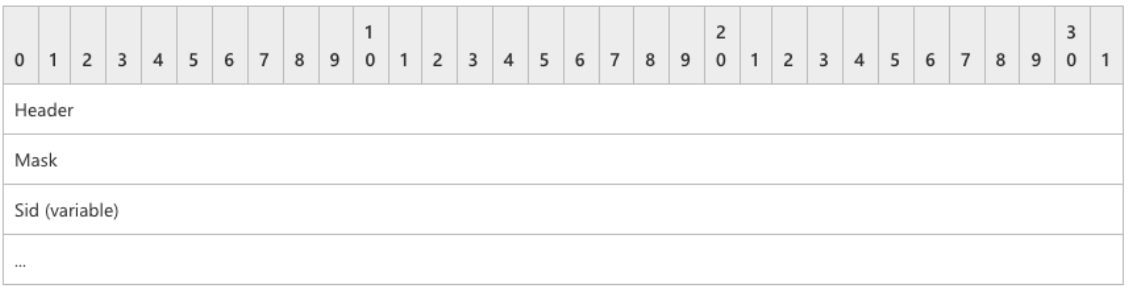
- The DACL has a list of ACEs (Access Control Entry) which comprise the majority of the “ACL” information (user and group Owner information makes up the rest)
- ACEs describe Allow or Deny permissions to control access TO a securable object BY a security principal with a given SID. Bloodhound only looks at Allows!
- Bloodhound collectors present a simplified and focused “abuseable” interpretation of the permissions represented by these ACEs
- There are two main types of the allow ACE that describe permissions:
  - ACCESS_ALLOWED_ACE
  - ACCESS_ALLOWED_OBJECT_ACE
- What do these two types of ACE look like then?

ACE standard header

https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dtyp/628ebb1d-c509-4ea0-a10f-77ef97ca4586

The rest of the ACE is different based on the type....

ACCESS_ALLOWED_ACE



https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dtyp/72e7c7ea-bc02-4c74-a619-818a16bf6adb

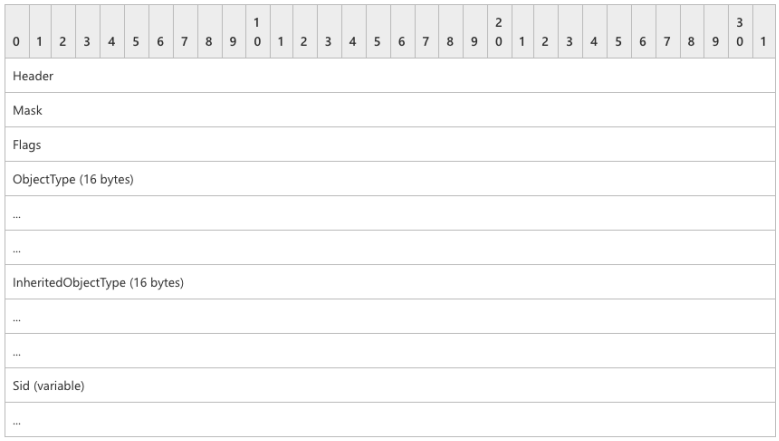
Example ACCESS_ALLOWED_ACE


```
{'Type': 'ACCESS_ALLOWED_ACE',
 'Sid': 'S-1-5-21-1954401763-2568157779-1971089446-519',
 'ResolvedSidName': 'ADDS\\Enterprise Admins',
 'Foreign': False,
 'Flags': ['CONTAINER_INHERIT_ACE', 'INHERITED_ACE'],
 'Mask': 983551,
 'Privs': ['GENERIC_READ',
  'GENERIC_WRITE',
  'GENERIC_EXECUTE',
  'GENERIC_ALL',
  'WRITE_OWNER',
  'WRITE_DACL',
  'READ_CONTROL',
  'DELETE',
  'ADS_RIGHT_DS_CONTROL_ACCESS',
  'ADS_RIGHT_DS_CREATE_CHILD',
  'ADS_RIGHT_DS_DELETE_CHILD',
  'ADS_RIGHT_DS_READ_PROP',
  'ADS_RIGHT_DS_WRITE_PROP',
  'ADS_RIGHT_DS_SELF']},
```

Bloodhound data file representation


```
{
  "PrincipalSID": "S-1-5-21-1954401763-2568157779-1971089446-519",
  "PrincipalType": "Group",
  "RightName": "GenericAll",
  "IsInherited": true
},
```

ACCESS_ALLOWED_OBJECT_ACE



https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dtyp/c79a383c-2b3f-4655-abe7-dcbb7ce0cfbe

ObjectType - A GUID that identifies a property set, property, extended right, or type of child object. The purpose of this GUID depends on the user rights specified in the **Mask** field. This field is valid only if the ACE _OBJECT_TYPE_PRESENT bit is set in the **Flags** field.

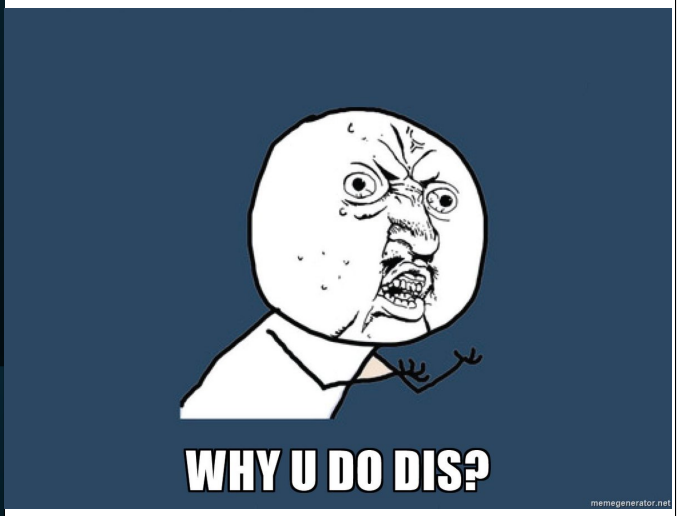
Example
ACCESS_ALLOWED_OBJECT_ACE


```
{'Type': 'ACCESS_ALLOWED_OBJECT_ACE',
 'Sid': 'S-1-5-21-432799531-553593876-76599720-498',
 'ResolvedSidName': 'WINDOMAIN\\Enterprise Read-only Domain Controllers',
 'Foreign': False,
 'Flags': [],
 'Ace_Data_Flags': ['ACE_OBJECT_TYPE_PRESENT'],
 'Mask': 256,
 'Privs': ['ADS_RIGHT_DS_CONTROL_ACCESS'],
 'ControlObjectType': 'DS-Replication-Get-Changes'},
```

Bloodhound data file representation


```
{
  "PrincipalSID": "S-1-5-21-432799531-553593876-76599720-498",
  "PrincipalType": "Group",
  "RightName": "GetChanges",
  "IsInherited": false
},
```

ACLs 😢



ACL parsing takeway

There’s a problem though:

SharpHound does a lot of “work”, but sometimes doesn’t “work”…

SharpHound limitations

- Designed to run from a compromised Windows machine in a target network making it reliant on solid network membership/access and captured session-based authentication – makes it likely to break when your network access is limited or you need to use specific credential information
- Heavily targeted by client-side endpoint security solutions, with various of its network communications detected at the server side by products such as Defender for Identity – strong chance of detection or blocking!

Sharphound Alternatives

- **Bloodhound.py** - https://github.com/dirkjanm/BloodHound.py
  - Python implementation of a BloodHound collector, uses the same collection approaches as SharpHound – not just LDAP
  - Works on non Windows machines, more flexible in terms of authentication
  - Still reliant on DNS, can fail when that is not properly accessible on the executing system
  - Older implementation, no collection of newer capabilities such as Certificate Services info, different versions for new and old BloodHound front ends..

- **ad_ldap_dumper** - https://github.com/stephenbradshaw/ad_ldap_dumper
  - My thing…
  - LDAP only
  - Very configurable but harder to use

ad_ldap_dumper

- Written in Python, runs on multiple platforms
- Authentication options include anonymous, user and password (even blank), NT pass the hash, Kerberos
- Designed to work with the minimum required access, as long as you can talk to TCP port 389/636 and have some form of valid credentials it will likely work
- Tells you if server is a Global Catalog, does the anonymous bind info collection thing
- A number of options to help avoid detection
- (Beta) option to produce BloodHound output files!

ad_ldap_dumper – why?

- Works under very restrictive, minimal access conditions (e.g. port forwarding, implant tunneling)
- Lots of control available to avoid detections
- Allow viewing of data using BloodHound GUI!

Detection

Detection

- There are a few different methods that can be used to identify LDAP enumeration
- Detection will generally rely on logging centralization and alerting infrastructure and needs to be tuned to the environment
- The better your understanding of how the protocol and the attack tools work, the better your detection approaches will be!

Detection:
* Relies on logging/alerting infrastructure
* Test in your environment

Detecting enumeration: Query logging

- Configured via registry settings on each domain controller you want to log
- Sent to "Directory Services" event log with event ID 1644
- Logging is intended for troubleshooting, so its very detailed and generates a high volume of events (can cause performance and storage issues)
- Used to be required for Defender for Identity, however this has recently changed
- Great way to maintain visibility if you can deal with the overheads, but needs good filtering/alerting to get useful information from the volume
- Alert approach could be to check for strings targeting particular object types in the query filter, e.g. "(member=*)" or "(objectClass=group)"

Detecting enumeration: DS access logging

- Configure "Directory Service Access" auditing for Success and Failure on the Domain Controllers
- Set SACLs on directory objects to target the things you are interested in – only events covered by SACLs will be logged!
- Attempts to enumerate the relevant objects or their properties via LDAP will be recorded in the Security event log with event ID 4662
- This creates event entries – then you need to alert on ones of interest as per following examples

DS access logging: Canaries

- Detection strategy for DC access logging
- Create one or more canary objects (users, groups, etc) and set the SACLs on the objects to audit "List contents", "Read all properties", "Read permissions" (and optionally more)
- With a small number of canary objects alerts can be made very specific and false positives should be rare and able to be individually whitelisted

Name your canary objects something sensible that blends in and put them with other objects of the same type

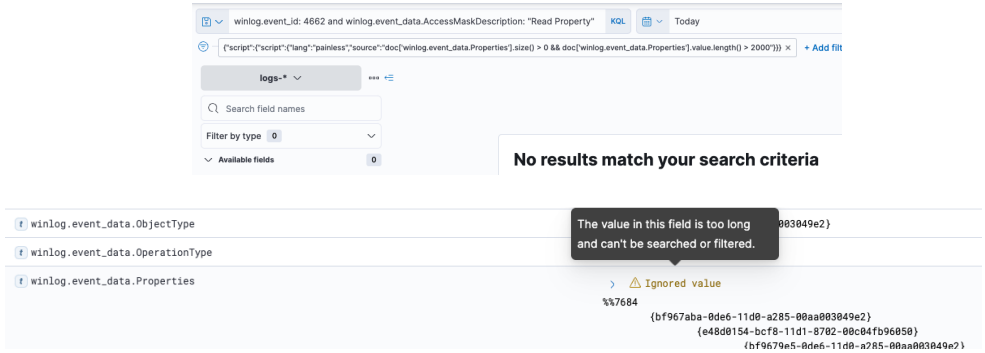
DS access logging: High no# of properties

- Detection strategy for DC access logging
- Set the "Read all properties" SACLs to all objects of a particular type (e.g. all users, all groups)
- Look for large number of properties being returned in 4662 event log entries
- "Suspicious Access to LDAP Attributes" ElasticSearch alert implements this

https://github.com/elastic/detection-rules/blob/374f21fbc46e0bc75fbc606f24bd8381b438d329/rules/windows/discovery_high_number_ad_properties.toml

High no# of properties: example

```
query = '''
any where event.action == "Directory Service Access" and
 event.code == "4662" and not winlog.event_data.SubjectUserSid : "S-1-5-18" and
 winlog.event_data.AccessMaskDescription == "Read Property" and length(winlog.event_data.Properties) >= 2000
'''
```



https://github.com/elastic/detection-rules/blob/374f21fbc46e0bc75fbc606f24bd8381b438d329/rules/windows/discovery_high_number_ad_properties.toml

Default settings result in the properties field not being indexed due to size which will prevent this alert from triggering – test your alerts to make sure they work

DS access logging: Sensitive properties

- Detection strategy for DC access logging
- Set the "Read all properties" SACLs on all objects of the types in which you are interested like users and computers
- Alert on object attributes of interest, they are recorded in the "Properties" field of the event data and logged by their "schemaIDGUID" value as defined in the LDAP schema
- "Access to a Sensitive LDAP Attribute" ElasticSearch alert implements this approach

https://github.com/elastic/detection-rules/blob/374f21fbc46e0bc75fbc606f24bd8381b438d329/rules/windows/credential_access_ldap_attributes.toml

Particular attributes to target using this approach requires you find the GUID associated with the attribute name in the LDAP schema

Sensitive properties: example

```
query = '''
any where event.action == "Directory Service Access" and event.code == "4662" and

  not winlog.event_data.SubjectUserSid : "S-1-5-18" and

  winlog.event_data.Properties : (
   /* unixUserPassword */
  "*612cb747-c0e8-4f92-9221-fdd5f15b550d*",

  /* ms-PKI-AccountCredentials */
  "*b8dfa744-31dc-4ef1-ac7c-84baf7ef9da7*",

  /*  ms-PKI-DPAPIMasterKeys */
  "*b3f93023-9239-4f7c-b99c-6745d87adbc2*",

  /* msPKI-CredentialRoamingTokens */
  "*b7ff5a38-0818-42b0-8110-d3d154c97f24*"
  ) and

  /*
   Excluding noisy AccessMasks
   0x0 undefined and 0x100 Control Access
   https://learn.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4662
   */
  not winlog.event_data.AccessMask in ("0x0", "0x100")
'''
```

Detecting enumeration: Network traffic

- With an IDS with appropriately placed network sensors you could attempt to detect LDAP enumeration via network traffic
- Check contents of network traffic for suspicious filters as per the query logging approach
- Look for anomalous volumes of traffic from a given client to LDAP ports 389 and 636 on a domain controller

LDAPS traffic is encrypted over the wire so wont get caught by thus approach

Detecting enumeration: Client/LOLBas

- Some checking can be done from the client to to determine if inbuilt tools are being abused for enumeration on compromised endpoint systems
- Requires reliable endpoint logging of command execution and centralizing of logs to allow alerting
- For Windows, can also enable ETW logging in the “Microsoft-Windows-LDAP-Client/Debug“ Event Channel, which records query filters
- Again, alerting needs to be quite specific due to volume (lots of commands get run on the average endpoint)

Detecting enumeration: LOLBas, example

```
query = '''
process where host.os.type == "macos" and event.type in ("start", "process_started") and
  (
    process.name : ("ldapsearch", "dsmemberutil") or
    (process.name : "dscl" and
      process.args : ("read", "-read", "list", "-list", "ls", "search", "-search") and
      process.args : ("/Active Directory/*", "/Users*", "/Groups*"))
        ) and
  not process.parent.executable : ("/Applications/NoMAD.app/Contents/MacOS/NoMAD",
    "/Applications/ZoomPresence.app/Contents/MacOS/ZoomPresence",
    "/Applications/Sourcetree.app/Contents/MacOS/Sourcetree",
    "/Library/Application Support/JAMF/Jamf.app/Contents/MacOS/JamfDaemon.app/Contents/MacOS/JamfDaemon",
    "/Applications/Jamf Connect.app/Contents/MacOS/Jamf Connect",
    "/usr/local/jamf/bin/jamf",
    "/Library/Application Support/AirWatch/hubd",
    "/opt/jc/bin/jumpcloud-agent",
    "/Applications/ESET Endpoint Antivirus.app/Contents/MacOS/esets_daemon",
    "/Applications/ESET Endpoint Security.app/Contents/MacOS/esets_daemon",
    "/Library/PrivilegedHelperTools/com.fortinet.forticlient.uninstall_helper"
    )
'''
```


#1
WORLD'S RECOMMENDED BRAND
SKIP THE RINSE
SAVE WATER
finish
POWERBALL
CLASSIC