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 team, red team, incident response (causing and remediating), building and maintaining tools and systems, etc
- Have worked in security for nearly 20 years in both internally and externally focused roles

Overview

- Minimal background on Active Directory and LDAP
- What sort of Active Directory information can be gathered from LDAP and what's the security relevance?
- How do you identify LDAP/Active Directory (AD) Domain Controller(DC) servers?
- What is required to connect to and query AD LDAP servers?
- What is Bloodhound, and how does it use LDAP?
- Why another LDAP tool?
- Detecting LDAP enumeration, and evading detections.

LDAP basics

- Stands for “Lightweight Directory Access Protocol”
- Provides a vendor neutral protocol for accessing network object information stored in a directory, which includes objects such as users, groups, computers, etc
- Objects are stored in a hierarchical structure, and have attributes that store information about the object
- LDAP object and attribute types are stored in an internal schema, which is extensible
- LDAP has its own unique naming format and query language

Protocol for accessing network objects
Heirarchical structure, objects have attributes storing info
Has an internally stored schema that's extensible
Has its own unique naming format and query language

LDAP in AD

- Microsoft Active Directory (AD) uses LDAP to allow clients to query information about directory objects
- Domain Controllers in AD store directory information
- Each AD Domain Controller runs an LDAP server
- Default settings allow any authenticated user to query a significant amount of useful directory information via LDAP
- Attackers with a working domain user account can obtain information for lots of privilege escalation and lateral movement attacks via LDAP…

Identifying AD LDAP servers

- **Port Scan**: Each AD Domain Controller (DC) runs LDAP and a number of other services, so DCs can be identified on the network using scans for the associated ports
  - Start by looking for TCP 389(LDAP) and TCP 636(LDAPS)
  - Confirm its a DC and not some Unix nonsense by also checking for services like TCP/UDP 53(DNS), TCP 445(CIFS), TCP 88(Kerberos), TCP 3389(RDP), TCP 5985(WinRM), TCP 5986 (HTTPS WinRM)

- **DNS**: If you know the DNS name of the target AD domain, and can perform resolutions against its DNS servers, performing an A lookup against the domain name will list all of the DCs
  - If you find one DC with a port scan direct a DNS query for the domain name against the DNS service on that DC to find the rest
  - If you don't know the domain name, get AD DNS domain names from Windows targets by looking for names in certificates of TLS protected services (LDAPS, RDP, HTTPS, WINRMS), collect LDAP anon server info or use CIFS enumeration tools like ntlm_challenger, etc

AD Global Catalog

- 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”.
- If you are attempting to extract AD information via LDAP, try and do it via a Global Catalog server…

Identifying Global Catalog Servers

- **DNS**: Global Catalog servers can be specially identified by performing a DNS lookup for a record with the name of "gc._msdcs.<domain_name>"
  - For a domain with the DNS name of example.com you would lookup gc._msdcs.example.com
- **LDAP Server Info**: Another option for identifying if an LDAP server is a Global Catalog server is to perform an anonymous bind and check the info.other.isGlobalCatalogReady property in its LDAP server information

Connecting to AD LDAP servers

- Basic communication 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) and TCP port 88 access if you don’t have the appropriate Service Ticket (ST) cached
- Basic server information can be retrieved with an anonymous bind, although insecure configurations can reveal more
- In the default configuration for AD, any authenticated user can retrieve most of the useful (to attackers) LDAP data, with only certain sensitive attributes being restricted

LDAPS connectivity…

- Listens on TCP port 636
- The LDAP SSL ports on AD DCs sometimes do not have a certificate configured which means LDAPS wont work
- SSL/TLS protocol and cipher restrictions can apply to LDAPS connections, so connections might fail unless the supported protocol versions and ciphers are used
- 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

LDAP queries

- LDAP uses its own (unusual) dedicated query language that can be used to retrieve lists of objects from the directory
- Queries can be performed at various levels of the LDAP directory “tree” to fetch records below that point, recursively or not
- Queries work by filtering based on the values (or presence) of objects attributes
- You can apply controls to queries that modify the way they are performed (more on this when we discuss ACLs)

LDAP query simple 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 queries

- 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 and the operation needs to be in its own bracketed group
- Example: To return objects that have objectClass of "user" **AND** objectCategory of "msDS-GroupManagedServiceAccount":

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

Breakdown of this weird syntax…

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

Broken down:

- (objectClass=user)
- **AND &**
- (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)))

- AS-REP roastable users

(&(objectClass=user)(userAccountControl:1.2.840.113556.1.4.803:=4194304))

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

Disabled Users = 2
Does not require Kerberos Pre Authentication = 4194304

More recent(ish) resources on LDAP querying

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

LDAP for AD attackers…

LDAP can be used to collect information about the following directory objects (amongst others) that can be useful for attackers looking to move laterally or escalate privileges:

- 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 info

LDAP for AD attackers, ctd…

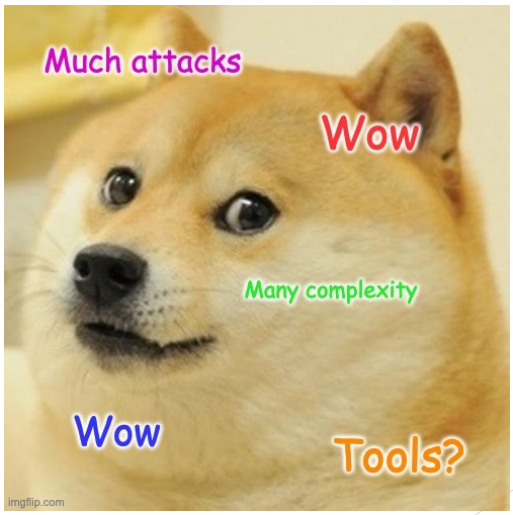
LDAP information extraction can be used to facilitate/identify potential for performing a number of AD attacks such as:

- 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)

Mo attacks, Mo problems?

- There are lots of potential attacks that you can facilitate via LDAP querying, each with specific requirements/things to focus on
- Some are relatively straightforward
- Some require access to properties that are hard to collect and interpret or mapping relationships between multiple different objects…

Lots of attacks, lots of complexity -> tools?



Bloodhound

- BloodHound is a tool created by SpecterOps to identify hidden relationships and attack paths within an Active Directory (and now Azure) environment
- It does this by querying and storing network information in a database and using graph theory to identify relationships between objects
- Uses collectors (like SharpHound or BloodHound.py) to collect information from the AD environment which it stores in categorized JSON files. These are then imported into a Neo4j graph database and viewed in the Bloodhound user interface to identify attack opportunities

Bloodhound collectors

- SharpHound (the reference collector) is usually run from a Windows computer joined to the domain you want to enumerate, and does most of the work for you of identifying and authenticating to the correct servers (including LDAP)
  - This has the downside of hiding the details from users and making the process somewhat fragile when dependent services are not accessible
- Bloodhound collectors make extensive (but not sole) use of LDAP for data collection

LDAP vs non LDAP collection in Bloodhound

- The majority of the information that Bloodhound collects 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 and are collected using time consuming and expensive non-LDAP queries made to individual computers in the domain
  - 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 approach/methodology…

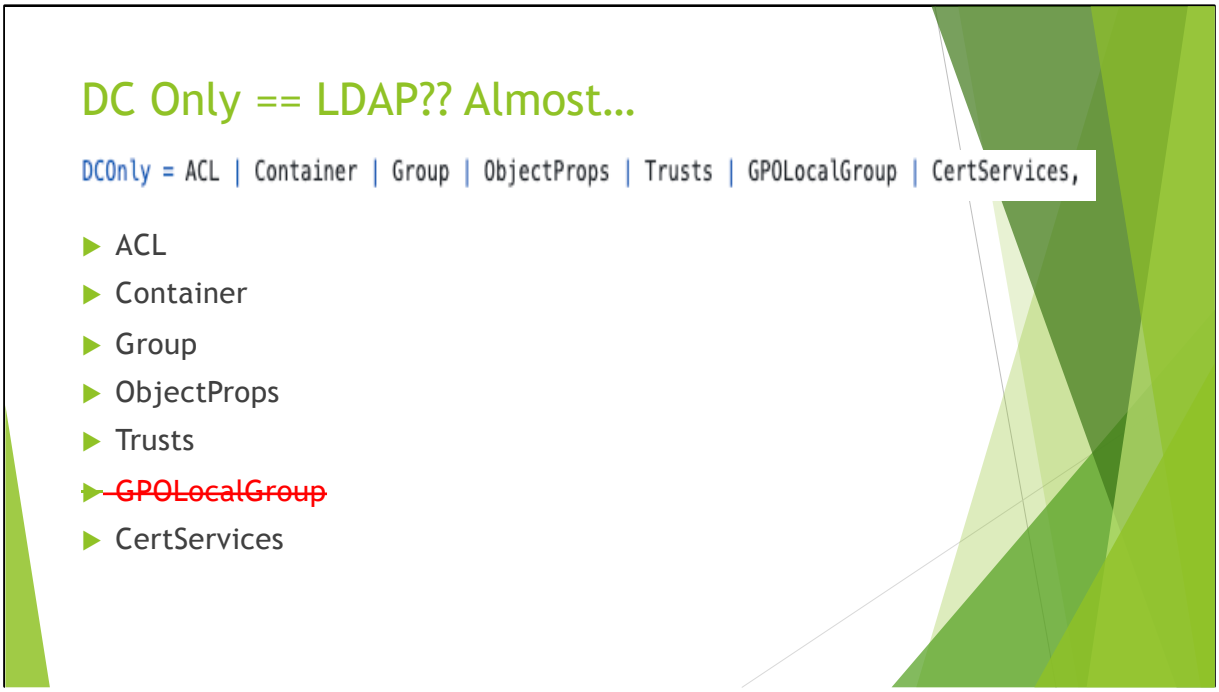
SharpHound collection categories

```
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
    }
}
```

```
"meta": {
    "methods": 33505,
    "type": "domains",
    "count": 1,
    "version": 6
}
```

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



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

Collection with no category

There are a few important things BloodHound collects from LDAP that there is no explicit “category” for:

- Domain users and some user like objects like managed service accounts
- Computers joined to the domain
- Information about the forest and the current domain

Categories == LDAP object types

- **Containers** – container objects and GPO links (container and organizationalUnit objects)
- **Group** – security group memberships (group objects)
- **Trusts** – information on trust relationships between domains (trustedDomain objects)
- **CertServices** – AD Certificate Services information (certificationAuthority and pKIEnrollmentService and pKICertificateTemplate objects)

ObjectProps collection category

- Bloodhound collectors by default limit attributes returned for LDAP objects, especially on default collected object types
- Doing ObjectProps collection gets additional potentially useful properties for objects such as User/Computer LastLogon or PwdLastSet
- Still doesn't mean everything is collected (there is a separate option for collecting all properties)

Bloodhound properties for a domain


```
"Properties": {
  "collected": true,
  "description": null,
  "distinguishedname": "DC=WINDOMAIN,DC=LOCAL",
  "domain": "WINDOMAIN.LOCAL",
  "domainsid": "S-1-5-21-432799531-553593876-76599720",
  "highvalue": false,
  "name": "WINDOMAIN.LOCAL",
  "whencreated": "",
  "isaclprotected": false,
  "functionallevel": "2016"
},
```

LDAP property names collected from same domain…

Not all are useful

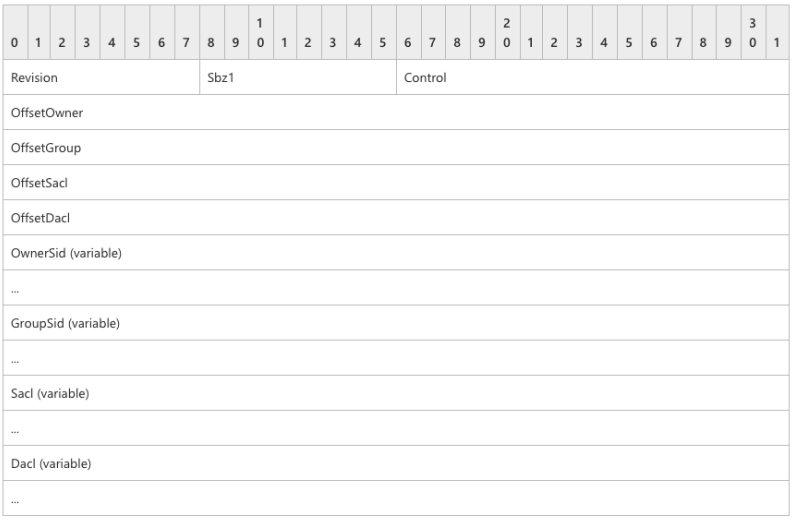
So far, so easy…

ACLs… ☹

- What can act on what – ACL abuse!
- Included in the DACL component of nTSecurityDescriptor LDAP attribute on each securable object
- This is a binary field that contains Owner user and group, plus DACL and SACL lists. 💣🔥 Requires interpretation and parsing to understand 🔥💣
- User and group entities describing access permissions in ACLs are referred to by their SID
- Object attributes in ACLs defining attributes that are individually secured are referred to by their GUID
- Both user/group SIDs and attribute GUIDs need to be resolved to friendly names via lookups as part of ACL parsing

SACL = System Access Control List
DACL = Discretionary Access Control List

SECURITY_DESCRIPTOR



0 1 2 3 4 5 6 7	8 9 10 1 2 3 4 5	6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
Revision	Sbz1	Control
OffsetOwner		
OffsetGroup		
OffsetSacl		
OffsetDacl		
OwnerSid (variable)		
...		
GroupSid (variable)		
...		
Sacl (variable)		
...		
Dacl (variable)		
...		

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

DACLs == list of ACEs == ACLs

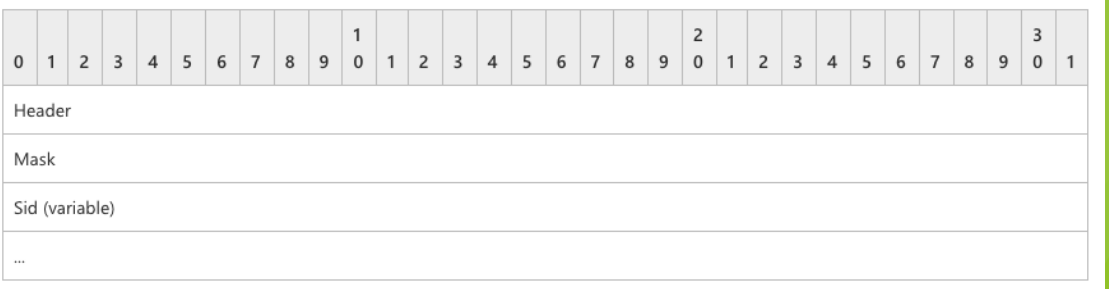
- The DACL has a list of ACEs (Access Control Entry) which represent the “ACLs”
- These describe Allow or Deny permissions to control access TO a securable object BY an object with a given SID
- Bloodhound only looks at the Allow entries, not the Deny ones
- The two main types of allow ACE are
  - ACCESS_ALLOWED_ACE
  - ACCESS_ALLOWED_OBJECT_ACE
- Bloodhound collectors present an “abuseable” subset of the permissions represented by these ACEs, and are not 1 to 1 representations of the actual ACE

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



0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Header																															
Mask																															
Sid (variable)																															
...																															

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

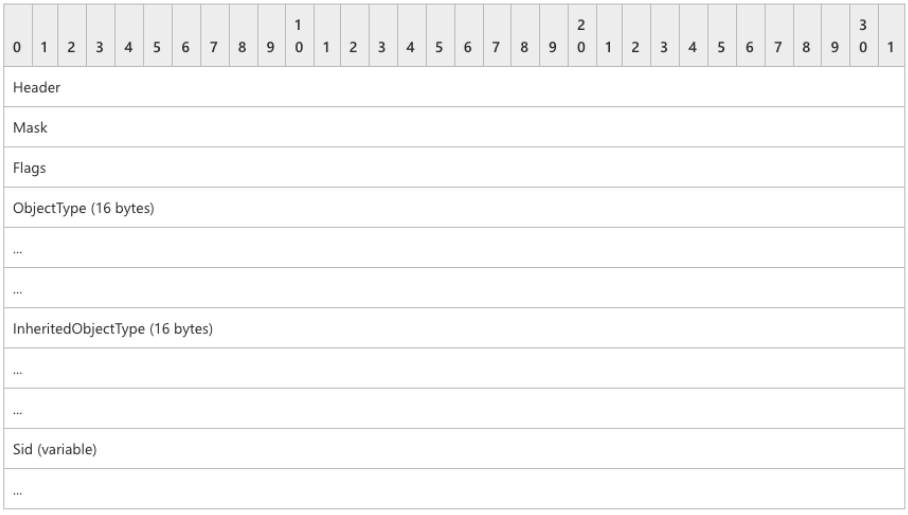
Example ACCESS_ALLOWED_ACE


```
{'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 representation


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

ACCESS_ALLOWED_OBJECT_ACE



0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Header																															
Mask																															
Flags																															
ObjectType (16 bytes)																															
...																															
...																															
InheritedObjectType (16 bytes)																															
...																															
...																															
Sid (variable)																															
...																															

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

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 representation


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

No, really, use a tool.

What do you do when the necessary tool doesn’t work??

I'll write my own…

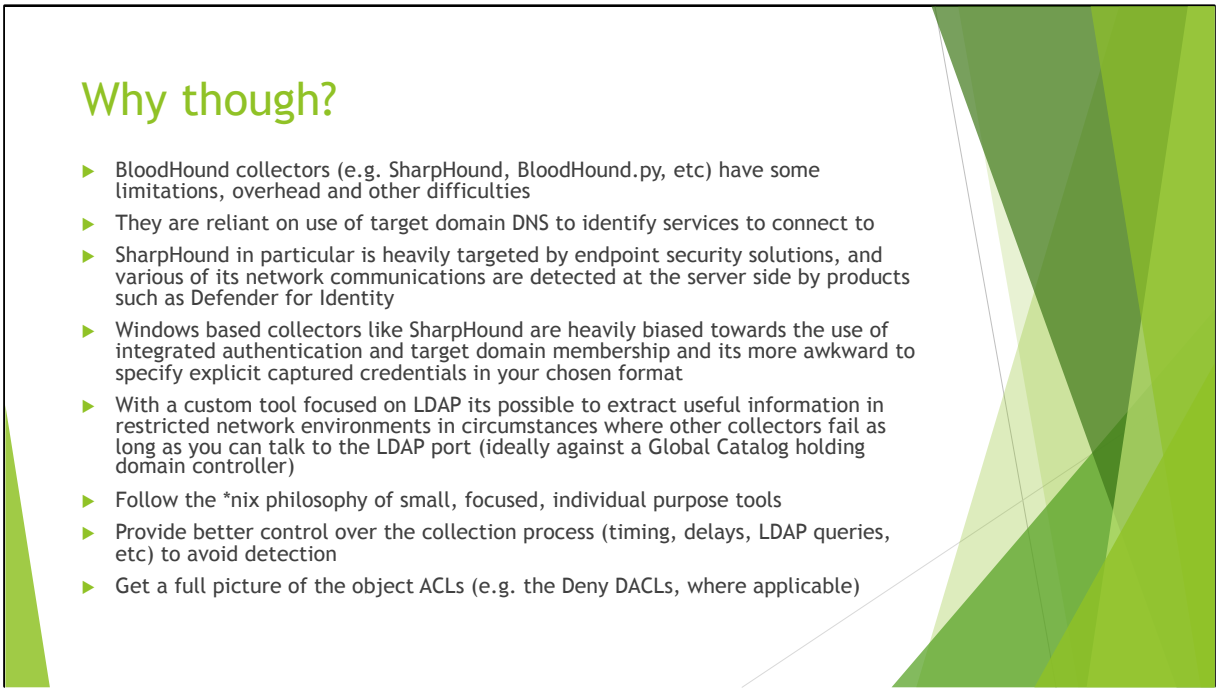
ad_ldap_dumper...

I wrote a tool to do LDAP based information collection, focusing on the same information that Bloodhound can collect, to allow more control over the enumeration process…

https://github.com/stephenbradshaw/ad_ldap_dumper

ad_ldap_dumper

- Written in Python using impacket to parse various Windows structures
- Runs on MacOS, Linux and Windows (probably, I haven't tested ☺)
- 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
- Options to specify custom queries and configurable, randomizable delays to hide from detection
- Dumps output into a big JSON file for easer integration and parsing with other tools
- Now has (Beta) option to produce BloodHound output files!



When other tools wont work
Works with minimal required access
Explicitly supply credentials in chosen format
EDR targets powershell and SharpHound
Allows greater control over process to facilitate evasion

Detection and Evasion

- 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
- Evasion approaches should be informed by an understanding of the target environment, with particular attention paid to the security products that are known to be in use and the estimated maturity of the security monitoring capability
  - The more mature the environment the more cautious and targeted the enumeration should be
  - Specific testing of approaches against known products can be performed if time/resources permit
- The better your understanding of how the protocol and the attack tools work, the better you can come up with novel evasion/detection approaches

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

Evasion:
* Target approach based on understanding of maturity of environment
 - More mature more cautious
* Try and identify products and test against them if possible

Both:
* Understand the protocol to better identify opportunities

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
- 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 full 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)"

Evade by:

- Avoid using obvious or well known filters to try and get lost in the noise

Detecting enumeration: Canaries

- 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 more if you want)
- Configure "Directory Service Access" auditing for Success and Failure on the Domain Controllers
- Attempts to enumerate the objects or their properties via LDAP will be recorded in the Security event log with event ID 4662
- 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

Evade by:

- This is difficult. An environment secured like this would require a minimal, cautious and goal oriented enumeration approach. For example, using custom LDAP queries to only enumerate groups or users with specific properties that interest you

Detecting enumeration: DS access

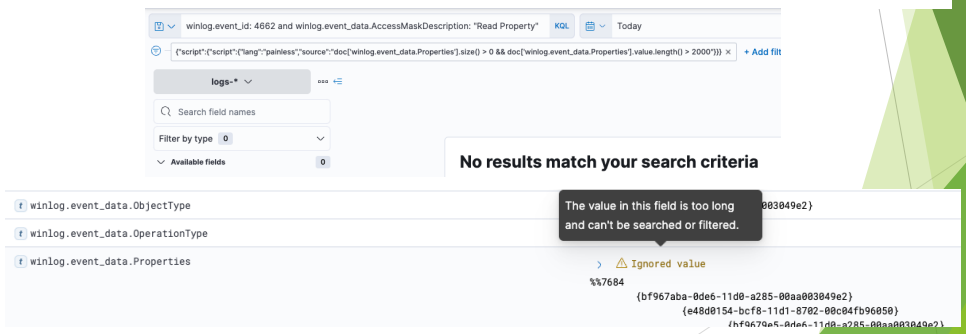
- The previously mentioned “canary” approach can also be performed more generally by applying the “Read all properties”, etc SACLs to all objects of a given type (e.g. all users, all groups)
- The greater the scope of applicable objects the greater the storage and performance implications
- Correct alert filtering is necessary to deal with the noise (make sure you test your alerts…)
- One approach is to search for lots of properties being queried

Evade by:

- As before design enumeration approach to work around filters (or hope for a mistake, see example..)

Detecting enumeration: DS access, 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
'''
```



S-1-5-18 = SECURITY_LOCAL_SYSTEM_RID A special account used by the operating system.
https://github.com/elastic/detection-rules/blob/374f21fbc46e0bc75fbc606f24bd8381b438d329/rules/windows/discovery_high_number_ad_properties.toml

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

Evade by:

- Use LDAPS to hide the content of traffic on the network
- Space the queries over a greater period of time to spread the volume

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)

Evade by:

- Run the command from an unmonitored machine (maybe using SOCKS C2 tunneling)
- Obfuscate commands to avoid command filters – copy and rename executables, avoid or modify target options, modify parent process, etc
- Obfuscate query filters to avoid ETW alerting

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"
    )
'''
```

https://github.com/elastic/detection-rules/blob/374f21fbc46e0bc75fbc606f24bd8381b438d329/rules/macos/discovery_users_domain_built_in_commands.toml#L52

The end!

Questions?