An Introduction to Manual Active Directory Querying with Dsquery and Ldapsearch

Hope Walker · Follow
Published in Posts By SpecterOps Team Members · 17 min read · Jun 2, 2021

-- 3

Introduction

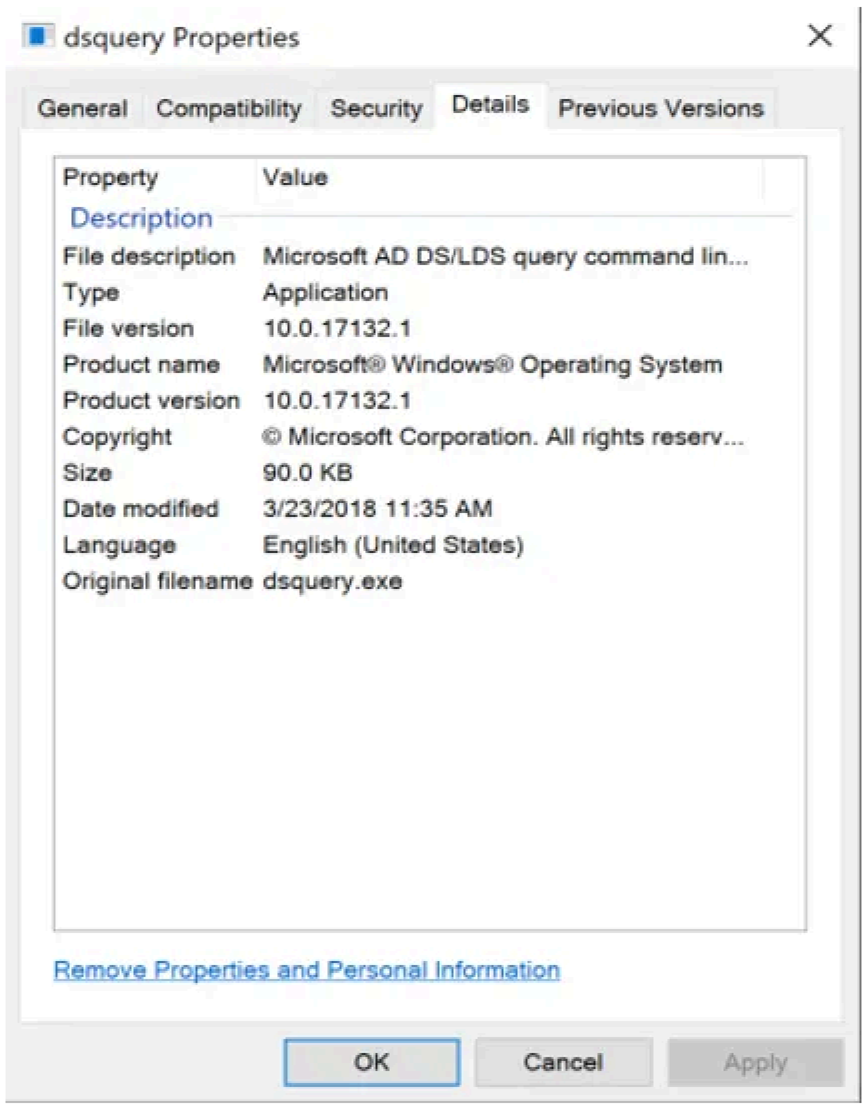
Let's be honest, BloodHound and PowerView are objectively better tools for querying, enumerating, and investigating Active Directory (AD). They are more efficient, intuitive and with BloodHound you can track queries easily. It is also worth noting before we dive in, using the `-v` flag in PowerView will show you the query that is being run and can save a bit of time. However, you may one day find yourself in a situation, as I did in a recent assessment, where those tools are not readily available or viable. In that circumstance, the team could not run either tool from our host and had difficulty proxying in the tools. While we battled to get a solution working to use these tools, we still needed to make progress towards our objectives. Therefore, we took to manually querying with a set of credentials we attained earlier. Manual LDAP searches can be done with ldapsearch on *nix systems, and dsquery on Windows machines.

For this blog, I will not be going through suggestions on how to get credentials or context to start querying, but assume that you already have the prerequisite information. Instead, I am going to focus on how to build queries with these tools and how to get the ball rolling while you figure out another solution.

Tools

Alright, so you are in a situation where BloodHound, PowerView, and other sane options will not work so you need to start querying AD manually. I will go over two tools that you can use to get started. Dsquery and ldapsearch are both tools used for querying AD (relatively) normally and can be used for offensive AD situational awareness.

First up, I will talk about dsquery which is a Windows binary, so it can (potentially) be uploaded to a target without raising too many alarms if it is not already present. The dsquery.dll is on most, if not all, Windows systems in `C:\Windows\System32\dsquery.dll` by default.



Property	Value
Description	
File description	Microsoft AD DS/LDS query command lin...
Type	Application
File version	10.0.17132.1
Product name	Microsoft® Windows® Operating System
Product version	10.0.17132.1
Copyright	© Microsoft Corporation. All rights reserv...
Size	90.0 KB
Date modified	3/23/2018 11:35 AM
Language	English (United States)
Original filename	dsquery.exe

The binary dsquery.exe may already be present on some servers at `C:\Windows\System32\dsquery.exe`. Dsquery is part of the Windows Remote Server Administrations Tools (RSAT) package and can be downloaded from Microsoft here. Also at that location, Microsoft explains "Starting with Windows 10 October 2018 Update, RSAT is included as a set of "Features on Demand" in Windows 10 itself." Depending on your situation following the instructions there may be an option for getting what you need on target. Dsquery is a command line utility and requires the following:

- Active Directory Domain Services (AD DS) server role installed (i.e., there must be a domain to query)
- An elevated command prompt (i.e., NT AUTHORITY\SYSTEM context)

The second tool is ldapsearch, which is native to macOS and *nix systems. While it may be present on your system already, you can install it by installing the `ldap-utils` package. On macOS, if it is not already installed you can install the `openldap` package via brew. To run ldapsearch queries, you will need to have the credentials for a valid AD account that can query AD. The best guide I have found (other than the man page) is at this website. The context for the ldapsearch queries here will be on Ubuntu Windows Subsystem for Linux with a domain service account's plaintext credentials.

Structuring Queries

Dsquery and ldapsearch have similar query structures, so going between the two is easy. Their output format is different but will provide mostly the same information. It is also important to note, there are several ways to get to the same results, but I will be going over my preferred methods. Both have a plethora of command line options depending on what you want to do. This blog will go over some of the most common ones I have used in the past, but I encourage you to consult the documentation for any gaps in your use case.

Dsquery Structure

The basic structure for dsquery is:

```
dsquery <object type> <filters> <options>
```

Object type has the following options:

- Computer
- Contact
- Group
- OU
- Site
- Server
- User
- Quota
- Partition
- * (Wildcard)

You can specify the object type in the first parameter as well as in the filter of a wildcard search or leave it open to all object types. Filters differ when you specify an object type and when you use a wildcard as will the output. Most often, I use the wildcard option for object type because the other object types do not provide as much information; this is a personal preference and when you are first getting started both will work to get the ball rolling. In the example below, you can see a query with user specified for the object type and then a query with a wildcard for the object type as well as the `objectclass` filter for users in a test domain (`PLANETEXPRESS`).

`dsquery user`

In this example, a simple query for users in the domain will show the distinguished name of every user in the domain. The upside is it is very easy and succinct; the downside is, you are missing a lot of information. Let's look at an example with a wildcard and limit the results to one user.

`dsquery * -filter "(objectclass=user)" -attr * -limit 1`

This is a much longer and relatively complicated filter compared to the last one, but you get a lot more information. For example, the user's group membership, description, and name can all been seen here. You will also get slightly different results between the two, but I will go into that more later. One other quick note: capitalization is not a concern in most cases, you can use proper format if you like, but it will likely not affect your results. If you find that it is affecting your results, my suggestion is to get all of the attributes for one object and copy the names out so they are in the correct format. For queries with dsquery, the context of the queries will be in the NT AUTHORITY\SYSTEM command prompt on a domain joined host.

Ldapsearch Structure

Ldapsearch has more flexibility on how queries are structured. For consistency throughout this blog here is the format I will use:

```
ldapsearch <format options> <authentication options> <domain options>
<query filters> <attribute list>
```

This probably is not entirely accurate, but it is how I will frame this discussion and how I recommend viewing it as a “get started” method. I will not be going over all the options, but I encourage everyone to read the man page or review additional options here. For consistency, let’s use the same query we did with dsquery to enumerate all users.

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D
'PLANETEXPRESS\SService' -w 'L1feD3@thSeamlessContinuum' -b
'DC=PLANETEXPRESS,DC=LOCAL' "(objectClass=user)" dn
```

Ldapsearch is going to be more complicated. Often, I will build out my first query with the options I need then go in and tweak the last two parts for the filters and attribute list for what I need. An important thing to note is that the assumption here is that you have plaintext credentials for an account that can access AD. For these queries, I used the plaintext username and password for the SService account, which would simulate a compromised service account. You may not necessarily need username and password, but you will need authentication of some sort. Throughout this blog that is how I will be structuring queries; I will go over other methods in the Options section.

Building Queries

Compound Filters

Compound filters are essential for querying successfully. They can be used to narrow down the scope of what you are looking for drastically and prevent you from wasting time.

A compound filter will look like these:

```
"(&(attribute1=value1)(attribute2=value2))"
```

Here the `&` indicates that it is a compound filter, and any results must have both values for the corresponding attribute. With the `!` symbol you can exclude items in your filters, but be sure to wrap it in a set of parenthesis. You can also exclude some results like this:

```
"(&(attribute1=value1)(!(attribute2=value2)))"
```

This filter will produce a list of results where the objects have `value1` for `attribute1` but do not have `value2` for `attribute2`. These filters will work the same between dsquery and ldapsearch. Depending on what command line utility you are using, you may have difficulty with the `!` symbol.

Wildcard

All my queries using these tools are wildcard searches. It is incredibly useful as you are getting started and looking to get oriented in the environment. My general approach is to start very broad and narrow down the query based on what I find. You can wrap a name or phrase in asterisks or put them at the beginning or end of a phrase.

In dsquery, we will look at an example where we want the name and sAMAccountName of all Windows machines.

```
dsquery * -filter “(&(objectclass=computer)(name=*win*))” -attr name samaccountname -d 192.168.88.195
```

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D ‘PLANETEXPRESS\SService’ -w ‘L1feD3@thSeamlessContinuum’ -b ‘DC=PLANETEXPRESS,DC=LOCAL’ “(&(objectclass=computer)(name=*win*))” name samaccountname
```

In practice, you would want to start this with just returning the attributes you need in order to move forward. Large AD environments would likely return a lot of results and depending on your OPSEC considerations and attention span for reading through AD objects, less attributes may be better. As you narrow down potential targets, you will likely move away from using wildcards to get specific results.

Another benefit of using the wildcard is you can use partial words to obscure what you are looking for. For example, instead of searching for `password` or `administrator` you can search `*sword*` or `*minis*`.

Users

Finding users in AD can be tricky, especially when the domain does not use names for usernames. In many cases, users are issued a unique identifier when they are onboarded that does not translate directly to their name. One important nuance to keep in mind when you are querying for users, is that computers objects are considered users as well. Depending on your query, you may need to exclude computers from your results.

In this example, the query will return all objects that are users, not computers and have w in the name:

```
dsquery * -filter “(&(objectclass=user)(!(objectclass=computer)
(name=*W*)))” -attr name samaccountname -d 192.168.88.195
```

It is common practice for administrators to have different accounts for administrative functions and everyday use. With a query like this you can look for accounts with names that indicate additional permissions such as `-sa` or `-da` appended to the end.

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D
‘PLANETEXPRESS\SService’ -w ‘L1feD3@thSeamlessContinuum’ -b
‘DC=PLANETEXPRESS,DC=LOCAL’ “(&(objectclass=user)(name=*W*))” name
samaccountname
```

As seen in this ldapsearch example, a computer object was returned along with the users. It can be difficult to exclude objects in ldapsearch because the ! operator has specific meaning on *nix systems. I was not able to find a successful way to escape this (`\!` and the many others I tried returned a bad filter).

Groups

Groups can be incredibly difficult to find and track manually. In a well-structured AD environment, there will be groups with granular permissions and users will be placed in their groups depending on work need. It is also common to find nested groups adding another level of complexity. Finding groups with the specific permissions can be difficult if you do not know the naming convention and nomenclature.

For dsquery, using the group object type can be a quick way to find groups by name. The wildcard object type will return more attributes which you will need when looking for members of the groups. Below are a series of queries where first groups with `*admin*` in the name are listed using the group object type. Then a query with a wildcard type is used to enumerate the groups that have *admin* in the group name. Lastly, a specific group, Domains Admins, is selected and the members of that group are listed (only one member in this case).

```
dsquery group -name *admin* -d 192.168.88.195
```

```
dsquery * -filter “(&(objectclass=group)(name=*admin*))” -attr name
samaccountname -d 192.168.88.195
```

```
dsquery * -filter “(&(objectclass=group)(samaccountname=Domain Admins))” -
attr name samaccountname member -d 192.168.88.195
```

In ldapsearch, the syntax is very similar to dsquery. Below is the ldapsearch syntax for finding groups with *admin* in the name.

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D
‘PLANETEXPRESS\SService’ -w ‘L1feD3@thSeamlessContinuum’ -b
‘DC=PLANETEXPRESS,DC=LOCAL’ “(&(objectclass=group)(name=*admin*))” name
samaccountname
```

An operational note for groups, I would start with less attributes and expand when you narrow down the list. It is also a good idea to look at the descriptions for the groups as it often has details on the purpose of the group. I have seen numerous times where the group description will spell out any acronyms or abbreviations in the group name.

Computers

In my experience, when you are looking for a specific computer, or a group of computers, you generally already have some information to back your search. Fortunately, searching for computers is easier than searching for users.

In dsquery, you can use either the computer object type or wildcard. With the computer object type there are quite a few options you can use to filter computers out, but I will not be exploring much of these in this blog. With the wildcard object type, I would suggest adding operating system (`operatingsystem` attribute) to your output or even your filter. This can provide very useful information when choosing systems to target.

The example below shows a search in dsquery for computer objects that have DC in the name. The assumption is that domain controllers are labeled in the environment. This environment only has one domain controller, but in a larger environment there may be many.

```
dsquery computer -name *DC* -d 192.168.88.195
```

```
dsquery * -filter “(&(objectclass=computer)(name=*DC*))” -attr name
samaccountname operatingsystem -d 192.168.88.195
```

Ldapquery will be very similar, and again, I recommend adding the operating system attribute to the filter or output.

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D
‘PLANETEXPRESS\SService’ -w ‘L1feD3@thSeamlessContinuum’ -b
‘DC=PLANETEXPRESS,DC=LOCAL’ “(&(objectclass=computer)(name=*DC*))” name
samaccountname operatingsystem
```

As with most queries, I would suggest getting the full information by listing out all attributes for computers before targeting it. Additionally, note that the sAMAccountName for the computer is the name with a $ appended. When using the command line, this can cause problems depending on what you are doing.

Description

Searching by the description is a great way to find out more information about potential targets in the domain. A lot of times abbreviations and acronyms are spelled out in the description, which is helpful if you have been given limited information. Sometimes, passwords may even be present in the descriptions.

When creating the filter for this query, I usually will not specify an object type unless it is required. I find this opens the results a bit more and can lead me to find situations or setups I would not have expected. It is important to also include the description in the attributes you are outputting for this because while it may have keywords or phrases you are looking for; it may also be in a context that is not helpful.

Here are a couple examples in dsquery of searching for objects with password and admin in the description.

```
dsquery * -filter “(description=*password*)” -attr name description -d
192.168.88.195
```

```
dsquery * -filter “(description=*admin*)” -attr name description -d
192.168.88.195
```

Note that the output is not as clean because the descriptions tend to be a bit longer. Now let’s look at the same queries with ldapsearch.

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D
‘PLANETEXPRESS\SService’ -w ‘L1feD3@thSeamlessContinuum’ -b
‘DC=PLANETEXPRESS,DC=LOCAL’ “(description=*password*)” name description
```

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D
'PLANETEXPRESS\SService' -w 'L1feD3@thSeamlessContinuum' -b
'DC=PLANETEXPRESS,DC=LOCAL' "(description=*admin*)" name description
```

The operational note for this is that if you are given only an acronym and want to figure out the full name, the acronym may not be in the description.

Password Last Set

The `pwdLastSet` attribute may help you when you are looking for accounts to target. Service accounts sometimes have older and more simple passwords because they are needed for specific functions and changing them can be difficult depending on what they are used for. This can also be a quick check during an assessment to see if the organization's password policy is properly implemented.

Dsquery and ldapsearch both allow for comparison other than equals (`=`). In this case, you would look for objects whose `pwdLastSet` attribute was less than a specified amount. This value is stored in epoch time so it is a bit harder to read.

Although this domain is new, we can still take a look at what this query would look like in dsquery. The query with just the `<` does not work so `<=` is necessary.

```
dsquery * -filter “(&(objectclass=user)(pwdlastset<=132655849658851779))” -attr name pwdlastset -d 192.168.88.195
```

In ldapsearch you will also have to use `<=` for your query.

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D ‘PLANETEXPRESS\SService’ -w ‘L1feD3@thSeamlessContinuum’ -b ‘DC=PLANETEXPRESS,DC=LOCAL’ “(&(objectclass=user)(pwdlastset<=132655849658851779))” name pwdlastset
```

Operational notes for this would mostly be about getting the time. There are tons of online converters you can use to get the time and convert it. You can also use PowerShell or Windows command line (`w32tm.exe /ntte <epoch time>` as shown in the dsquery example) to get a human readable version. As mentioned before this can also be a quick item to check to buff up findings for your client.

Member Of

Searching by the `memberof` attribute is best used when you already have some information to work with. It is very helpful in finding users to target based on their group membership. If you already know the name of the groups you can of course list the members of groups in your attributes. However, with combining filters, you can see if there is a user who is a member of multiple groups, such as Domain Admins and Server Operators.

In dsquery, a query for users who are members of a group would look like this:

```
dsquery * -filter “(&(memberof=CN=Staff,DC=PLANETEXPRESS,DC=LOCAl)
(memberof=CN=ShipCrew,DC=PLANETEXPRESS,DC=LOCAL))” -attr name memberof -d
192.168.88.195
```

Note that you will likely need the full `distinguishedName` of the group for your query. In ldapsearch you will build a similar query.

```
ldapsearch -LLL -x -h DC-THESHIP.PLANETEXPRESS.LOCAL -p 389 -D
‘PLANETEXPRESS\SService’ -w ‘L1feD3@thSeamlessContinuum’ -b
‘DC=PLANETEXPRESS,DC=LOCAL’ “(&(memberof=CN=Staff,DC=PLANETEXPRESS,DC=LOCAl)
(memberof=CN=ShipCrew,DC=PLANETEXPRESS,DC=LOCAL))” name memberof
```

Make sure throughout your querying you are saving this information outside of your terminal or application. Queries can take up a lot of space and saving the information off will prevent you from having to redo queries for information.

Options

Dsquery

There are several options you can tack on to the end of your query. These are the ones used in the above queries or that I use most commonly.

`-attr`

This is the attribute filter you can use with the wildcard type. Using `-attr *` will print out all of the attributes for the objects. To get specific attributes you will just use the attribute name in AD (e.g., sAMAccountName, pwdLastSet, etc.) in a comma separated list. Operationally I tend to start with very broad object filters and narrow the attributes. As I get closer to narrowing down my list of target objects I will open up attributes more to get additional information. When you are in a very large AD environment, and depending on what your C2 platform is, this can be a real time saver. Generally, when I start out, I cast a very wide net and then narrow down as I learn more.

`-d`

This is the domain flag and can be very useful with working in a large domain environment. You can specify the server to query by FQDN or IP address and, if the environment allows, use it to query into other domains. Querying different domain controllers can provide interesting points of data. One such point is while most data is replicated across domain controllers, last login is not always replicated. If you are looking for when a specific user or users last logged in, you may get different answers depending on the domain controller they authenticated to.

`-u -p`

These two flags can be used to specify the username and password to use for querying AD.

`-limit`

This is another option that can be useful when working in a large domain. By default, results are limited to the first 100 in dsquery. Setting this to 0 will return all the results from the query. Another way I use this operationally is setting the limit to one (`-limit 1`) to make sure my filters and options are set correctly before I return a larger number of results.

Ldapsearch

Ldapsearch has enough options to make a blog on those alone. For this, I will also focus on the ones used in here or other more useful ones.

`-L(L)(L)`

This option is at the beginning of all my ldapsearch commands. It removes generic ldap information in the output. There are three levels, I generally chose to remove all the information because it makes the output more succinct.

`-x`

This uses simple authentication (i.e., username and password) instead of Simple Authentication and Security Layer (SASL). In these examples, the context we were required to use was username and password, but that may not always be the case.

`-h`

Use this to specify an alternate host to run the query against. As the environment used for examples only had one domain controller it was consistent.

`-p`

Specify the port number on which the ldap server is listening. This is usually 389 or 636 for LDAPS, but it may be best to check first or during troubleshooting.

`-D`

This will be the `distinguishedName` to bind to the LDAP directory. If you are not using the -x option, the server will ignore this value.

`-w -W -y`

All of these are for the password. In the examples `-w` was used because if you are using this through some sort of C2 this is likely the option you will need. This is because you will likely not have a way to enter a password when using C2. Although it is not a great practice to send plaintext passwords through commands, if you do not have a way to send the password when prompted your query will not complete. If you are interactive, you can use the `-W` option instead to prompt for a password. Alternatively, you can use the `-y` option which will use a password file.

`-b`

This is to specify the search base for the query. This will likely be the distinguished name for the domain, but it can be narrowed down further depending on your query.

Conclusion

This is neither the easiest nor most optimal way to query AD. As stated in the beginning, tools such as BloodHound and PowerView are much better at these. If you find yourself in a situation where you need to use dsquery or ldapsearch, this guide should help you to save a bit of time researching how to create these queries. There is much more in terms of queries and options that can be covered for these two tools. After I have recovered from the painful experience of setting up my own domain, writing this blog and trying to maintain my sanity, I may write a follow up with additional information.

Didn't see what you were looking for in here? Tweet me or SpecterOps. A follow up blog will be coming *sometime* and suggestions are welcome.

Active Directory Offensive Security Hacking Linux Windows