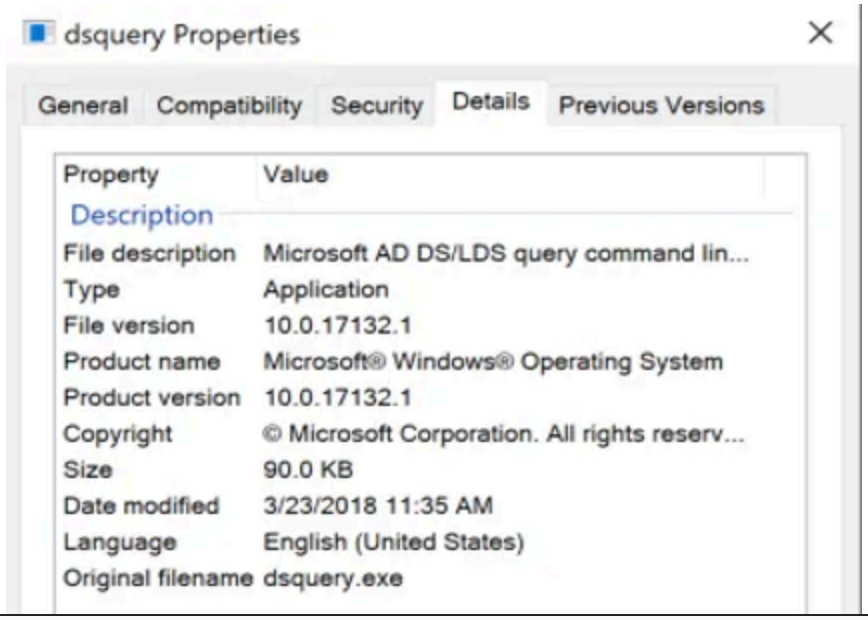
An Introduction to Manual Active Directory Querying with Dsquery and Ldapsearch

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.



- 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.

Object type has the following options:

- Computer
- Contact
- Group
- OU
- Site
- S

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

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

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

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:

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

```
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.

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

```
dsquery * -filter "(&(objectclass=computer)(name=*DC*))" -attr name
```

```
'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

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.

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:

`-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

`-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

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.

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

Written by Hope Walker

58 Followers · Editor for Posts By SpecterOps Team Members

Follow