BOFHound: Session Integration

posts.specterops.io/bofhound-session-integration-7b88b6f18423

Matt Creel January 30, 2024

Background

If you've found yourself on a red team assessment without SharpHound (maybe due to OPSEC or stealth requirements), you'd probably agree that mapping Active Directory is significantly more difficult. Tying down nested group memberships and trying to map ACL-based attack paths can become exceedingly complex outside of BloodHound's user interface and its Cypher queries. In early 2022, Adam Brown and I released BOFHound (now being maintained in a fork) as one approach to address these difficulties. Despite the name, BOFHound is written in Python and is designed to parse raw LDAP search results out of command and control (C2) logfiles (originally from Cobalt Strike logs and the `ldapsearch` beacon object file [BOF]) into BloodHound compatible JSON. This allows for manual, operator-guided LDAP enumeration that can avoid triggering detection mechanisms (e.g., expensive LDAP query thresholds) that may alert on SharpHound, while still allowing for usage of BloodHound's beloved relationship graphing.

Prior blog posts here and here contain additional background and usage examples.

> *: BOFHound is not officially affiliated with BloodHound Enterprise (BHE) or BloodHound Community Edition (BHCE), nor does the BloodHound product team support it. This is simply tooling I co-created and maintain for fairly niche LDAP enumeration scenarios.*

What's New?

One of the biggest pitfalls of BOFHound's prior usage strategies was the total absence of user session and local group membership data. If operational requirements prevent you from running SharpHound, BOFHound previously served as a way to approximate its data collection/processing with manual LDAP queries, but did not allow for the parsing of non-LDAP data.

Of course, session and local group data have always been obtainable from a C2 implant without the use of SharpHound; for example, BOFs such as `netsession`, `netloggedon` and `netLocalGroupListMembers` (all part of the amazing TrustedSec situational awareness [SA] BOF collection) leave this information a few keystrokes away. However, the output data can't be visualized in BloodHound's interface or used in its attack path mapping, leaving information gathered up to the operator for organization and tracking.

Today, I've submitted a pull request to the SA BOF collection, which consists of four "new" BOFs (which are really slightly modified versions of existing SA BOFs) with BOFHound compatibility. The modded BOFs don't include any material upgrades to their original purpose, but are more suitable for log parsing and include more verbose information for

BOFHound where possible (e.g., returning member SIDs and object types alongside local group member user names). Additionally, a new version of BOFHound (0.3.0) has been released that includes logic for parsing and processing session data and local group membership data these BOFs gather.

The remainder of this post delves into BOF details and a contrived example attack path visualized using the BOFs, BOFHound, and BHCE.

Operator Guidance and Examples

I've configured a test domain (*REDANIA.LOCAL*) with three domain machines and one Cobalt Strike team server (Figure 1).

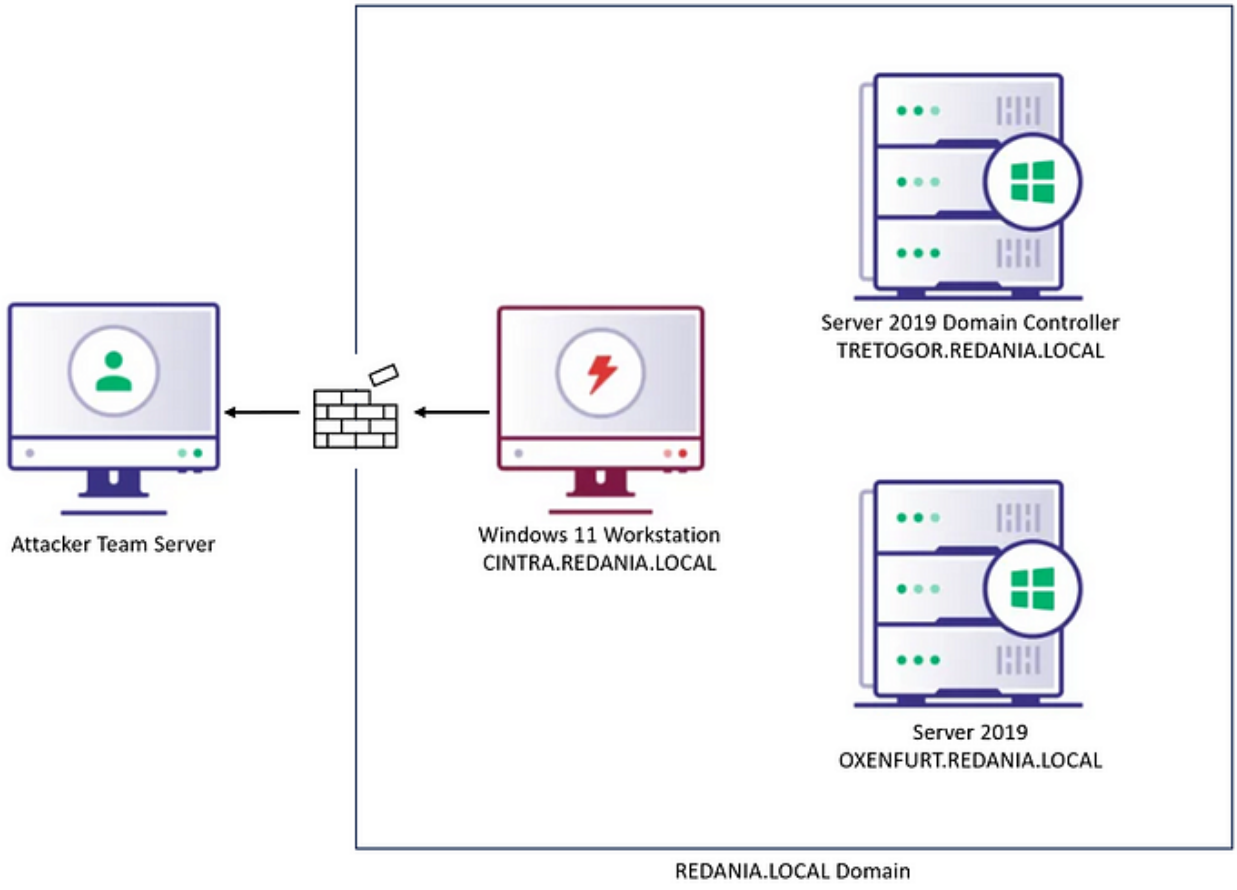


Figure 1 — Lab Domain

The *CINTRA* workstation is running our Cobalt Strike Beacon, which is where we'll simulate reconnaissance. For the sake of time, let's assume I already collected all the LDAP objects we're interested in with the `ldapsearch` BOF, using `(objectClass=*)` for the filter and the default search base of `DC=redania,DC=local` (remember to include `*,ntsecuritydescriptor` as the attributes for the search to return if you want to process ACL attack paths). Obviously, if we're running this query, we may just want to run SharpHound with the `DCOnly` collection method, but that's outside the scope of this post.

SharpHound offers two collection flags (`Session` and `LoggedOn`) for sessions, which perform remote collection using three methods:

SharpHound also offers collection methods for individual local groups (`LocalAdmin`, `RDP`, `DCOM` and `PSRemote`) and the `LocalGroup` method, which targets all of the four local groups.

Each of the four compatible BOFs serve as a type of “collector” for one of the functions discussed above.

Privileged Session Enumeration

The `netloggedon` BOF makes a `NetWkstaUserEnum` Windows API call, targeting the computer the operator specified. This API requires admin rights on the target machine and, according to Microsoft, “lists entries for service and batch logons, as well as for interactive logons.”

`netloggedon2` is the least modified of the situational awareness BOFs; the only change is to include the operator-supplied servername in the output. If the BOF is used to query logged on users on localhost, the fully qualified computer DNS name from `GetComputerNameExW` is used.

Targeting the lab’s Server 2019 host (*OXENFURT*) with the BOF (where our simulated initial access user has admin rights) reveals several users logged in, either interactively or via a service (Figure 2).

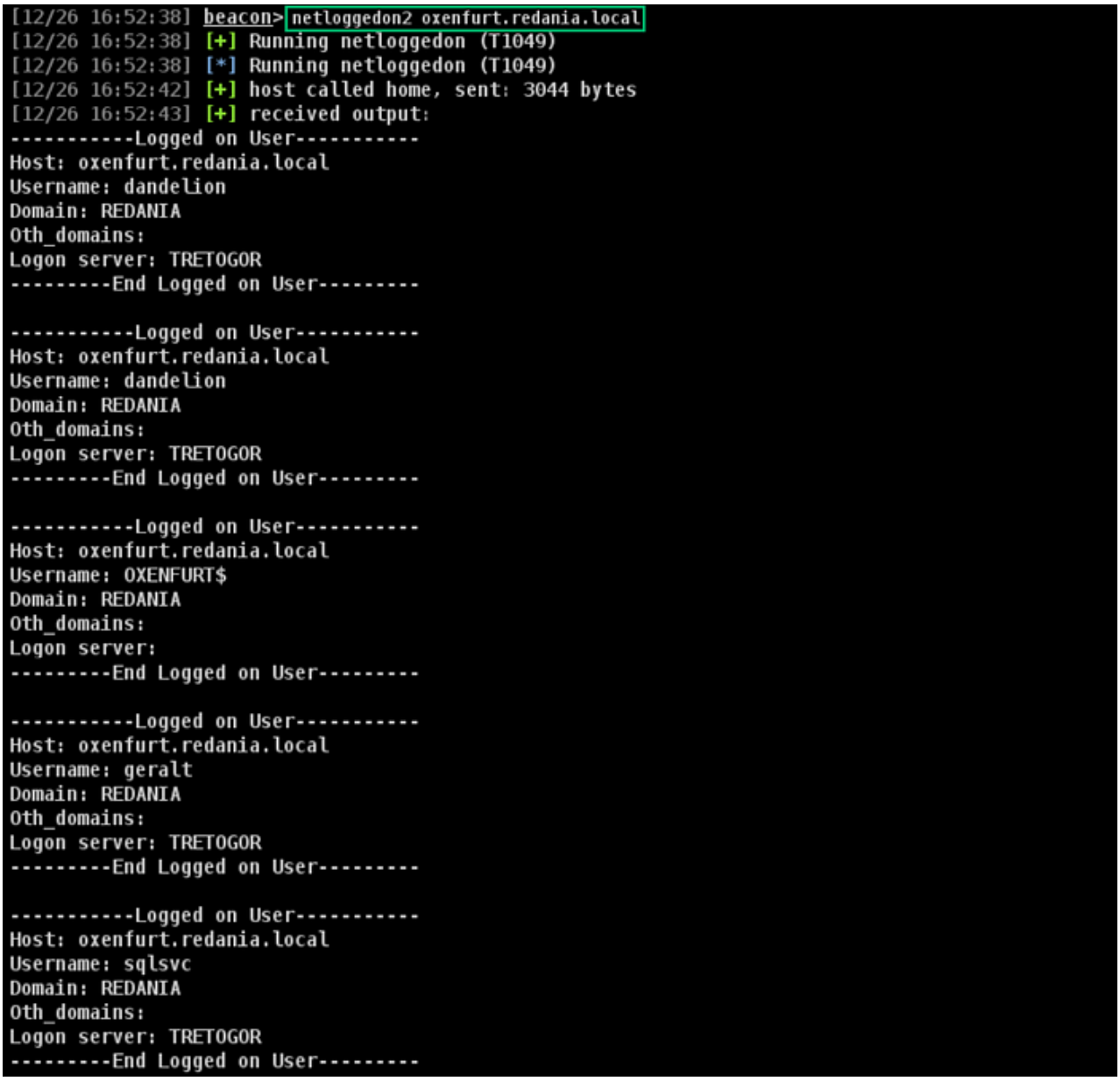


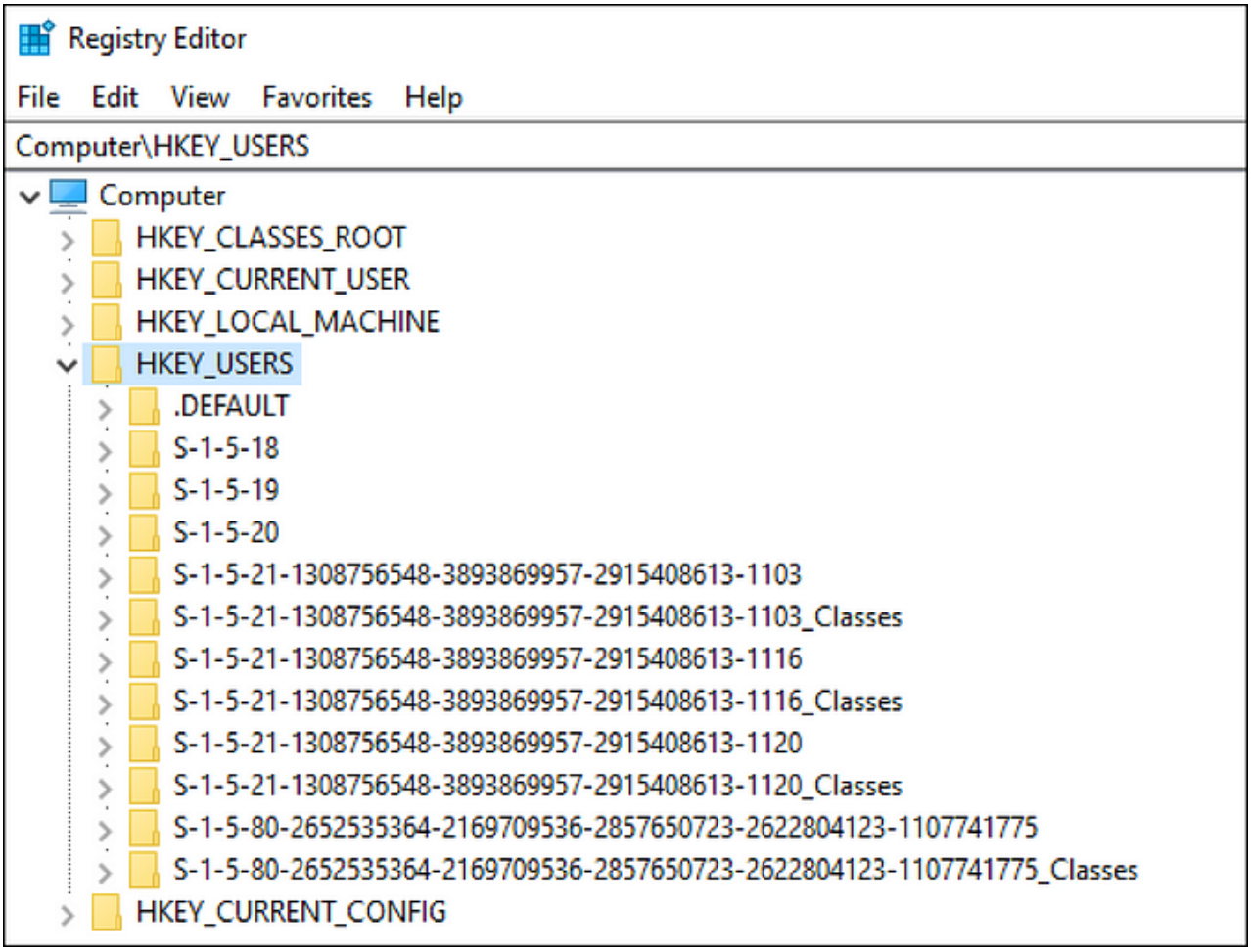
Figure 2 — Sessions Gathered with NetWkstaUserEnum

An important note here is that fully qualified DNS names are important when supplying the `servername` argument. BOFHound tries to match the session data to an LDAP computer object based on a matching `dNSHostName` attribute. If that fails, the DNS suffix (e.g., *REDANIA.LOCAL*) will be converted to a distinguished name (e.g., *DC=REDANIA,DC=LOCAL*) and the hostname (e.g., *OXENFURT*) will be used to match a computer object's `sAMAccountName` attribute (e.g., `OXENFURT$`), within that domain. Both of these rely on supplying fully-qualified domain names (FQDNs) as arguments. Targeting hosts by unqualified name or IP address is not supported (the BOFs will still work, but BOFHound won't be able to match the session to a computer object). These same principles will apply to the `regsession` and `netLocalGroupListMember2` BOFs.

Registry Session Enumeration

This is the second method SharpHound uses when the `LoggedOn` collection method is specified and also requires admin rights on the remote target. The `HKEY_USERS` registry hive on a system will contain SIDs as subkeys for currently logged on users (both local

and domain). Taking a look at the registry on the enumeration target for my lab (*OXENFURT*), there are subkeys indicating the presence of three domain user logons (Figure 3).



Subkeys Indicating Sessions

These correspond to the *GERALT*, *SQLSVC*, and *DANDELION* domain users seen in the output from the `netloggedon` BOF. Running the `regsession` BOF (based on the SA `reg_query` BOF), these three SIDs are obtained from the remote `HKEY_USERS` hive (Figure 4).

```
[12/26 16:58:22] beacon> regsession oxenfurt.redania.local
[12/26 16:58:22] [+] Running regsession (T1049)
[12/26 16:58:22] [*] Running regsession (T1049)
[12/26 16:58:25] [+] host called home, sent: 3521 bytes
[12/26 16:58:26] [+] received output:
[*] Querying registry on oxenfurt.redania.local...
----------Registry Session---------
UserSid: S-1-5-21-1308756548-3893869957-2915408613-1103
Host: oxenfurt.redania.local
---------End Registry Session-------

----------Registry Session---------
UserSid: S-1-5-21-1308756548-3893869957-2915408613-1116
Host: oxenfurt.redania.local
---------End Registry Session-------

----------Registry Session---------
UserSid: S-1-5-21-1308756548-3893869957-2915408613-1120
Host: oxenfurt.redania.local
---------End Registry Session-------

[*] Found 3 sessions in the registry
```

Figure 4 — Sessions Gathered With Registry Queries

While maybe less visually appealing, BOFHound benefits from this session enumeration method returning user SIDs directly, meaning there is no need to obtain a matching SID via user object lookups on the `sAMAccountName` attribute. Local account SIDs will also show up here, but BOFHound will ignore them.

Session Enumeration

The final session enumeration method leverages the `NetSessionEnum` API and differs from the other methods for two reasons. First, it *may* not require local admin rights over the target to execute. Admin rights are required from Server 2016 onward and Windows 10 version 1607 onward (by default). Second, it doesn't show users logged into the targeted system, but instead reveals the client IP a user has established a session on your targeted system from. One example of this is share browsing; a user browsing a remote file server is not "logged on" to the file server, but has established a session from their client (where the logon actually is) (Figure 5).

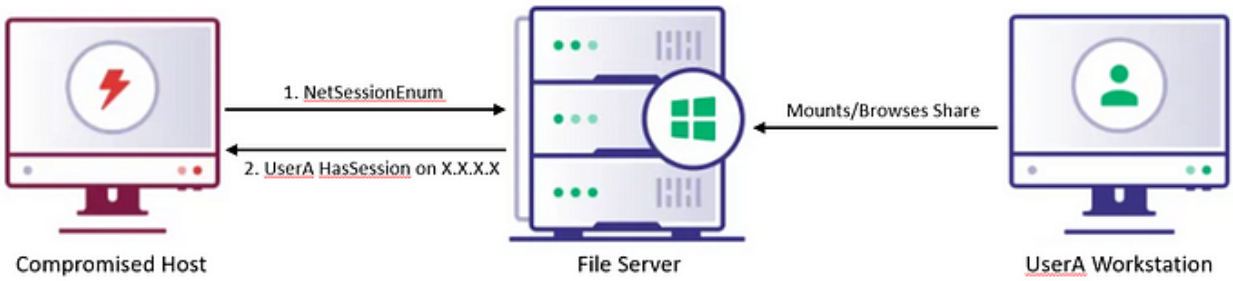


Figure 5 — Example NetSessionEnum Workflow

Somewhat problematic for BOFHound, the API call returns the client the user established the session from as an IP which is unable to be matched to a LDAP search result. Modifications to the `netsession` BOF include options for resolving returned client IPs to DNS names or computer names.

Option 1: Reverse DNS Lookups

This is the default resolution method that will be used if one is not explicitly specified. A reverse DNS lookup is attempted using the client IP address, in hopes of returning a FQDN (again, to match to a computer object's `dNSHostName` attribute from LDAP).

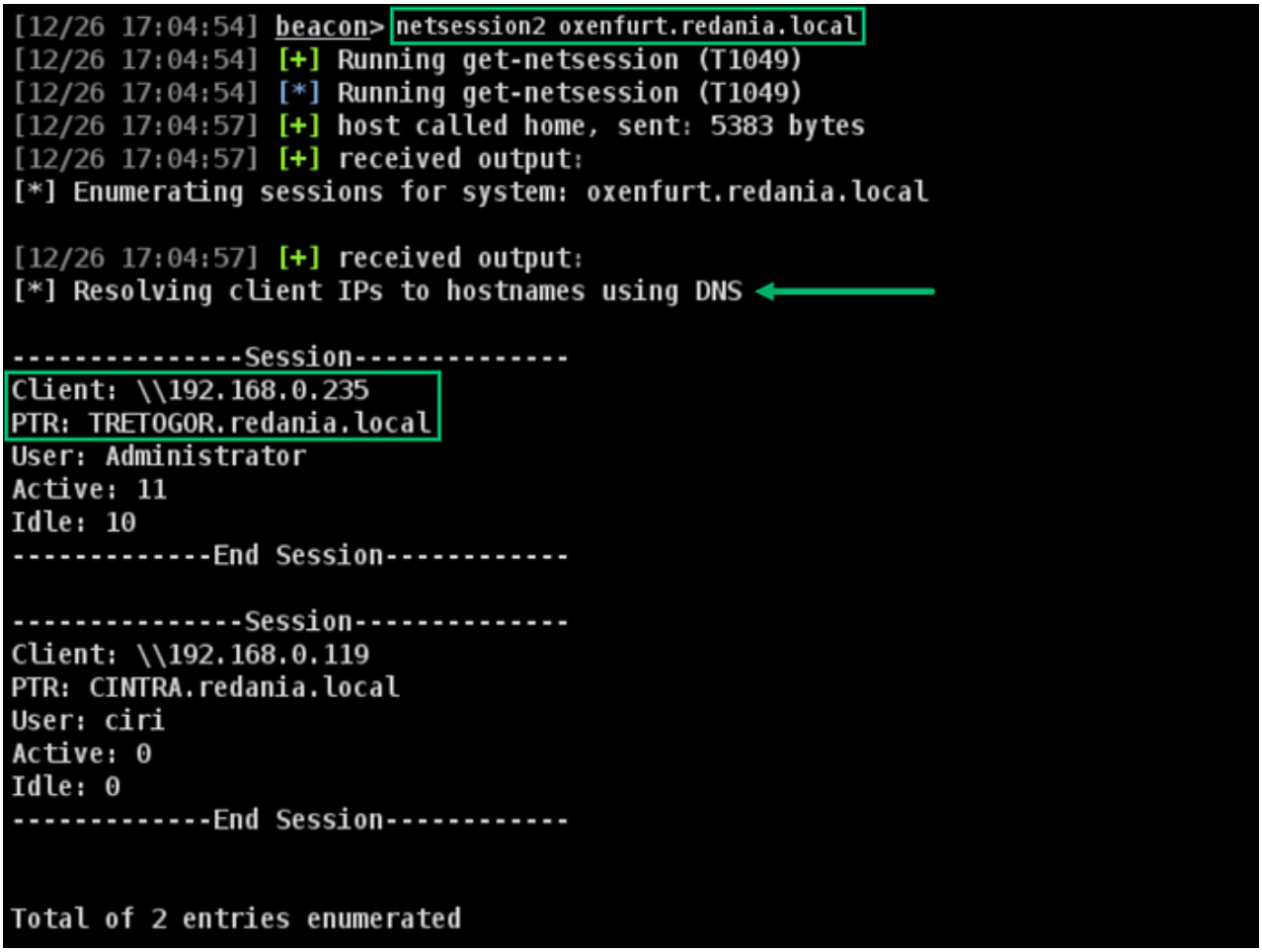


Figure 6 — Sessions Gathered with `NetSessionEnum`and DNS Resolution

During lab setup, I used the *ADMINISTRATOR* user to map a share on *OXENFURT* from *TRETOGOR* to simulate a session for the results. The second result, for the user *CIRI,* is just a result of running the BOF from the *CINTRA* host.

A specific DNS server can also be used with the syntax `netsession2 <target> 1 <DNS Server>`.

Reverse DNS lookup is the default option because it's quieter than making connections to each remote client; however, it depends on reverse lookup zones being configured. In the DNS Manager snap-in, you can see the PTR records that allow the lookup to be successful (Figure 7).

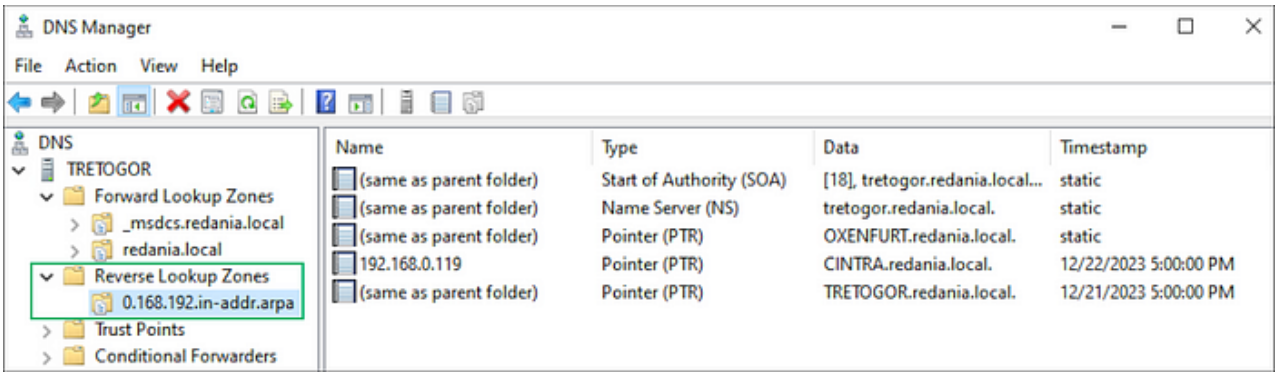


Figure 7 — Reverse Lookup Zone Configuration

Option 2: NetWkstaGetInfo

In the event that reverse lookup zones are not configured, the second resolution method available involves calling the `NetWkstaGetInfo` API. This is not the default method because it involves creating a SMB connection to each client associated with a session to retrieve the client's NetBIOS computer name and NetBIOS domain name. Depending on the host you enumerate sessions on, this may cause an excess number of SMB connections to remote hosts (another potential SharpHound indicator).

This resolution method can be specified with a `2` for the resolution method argument (Figure 8).

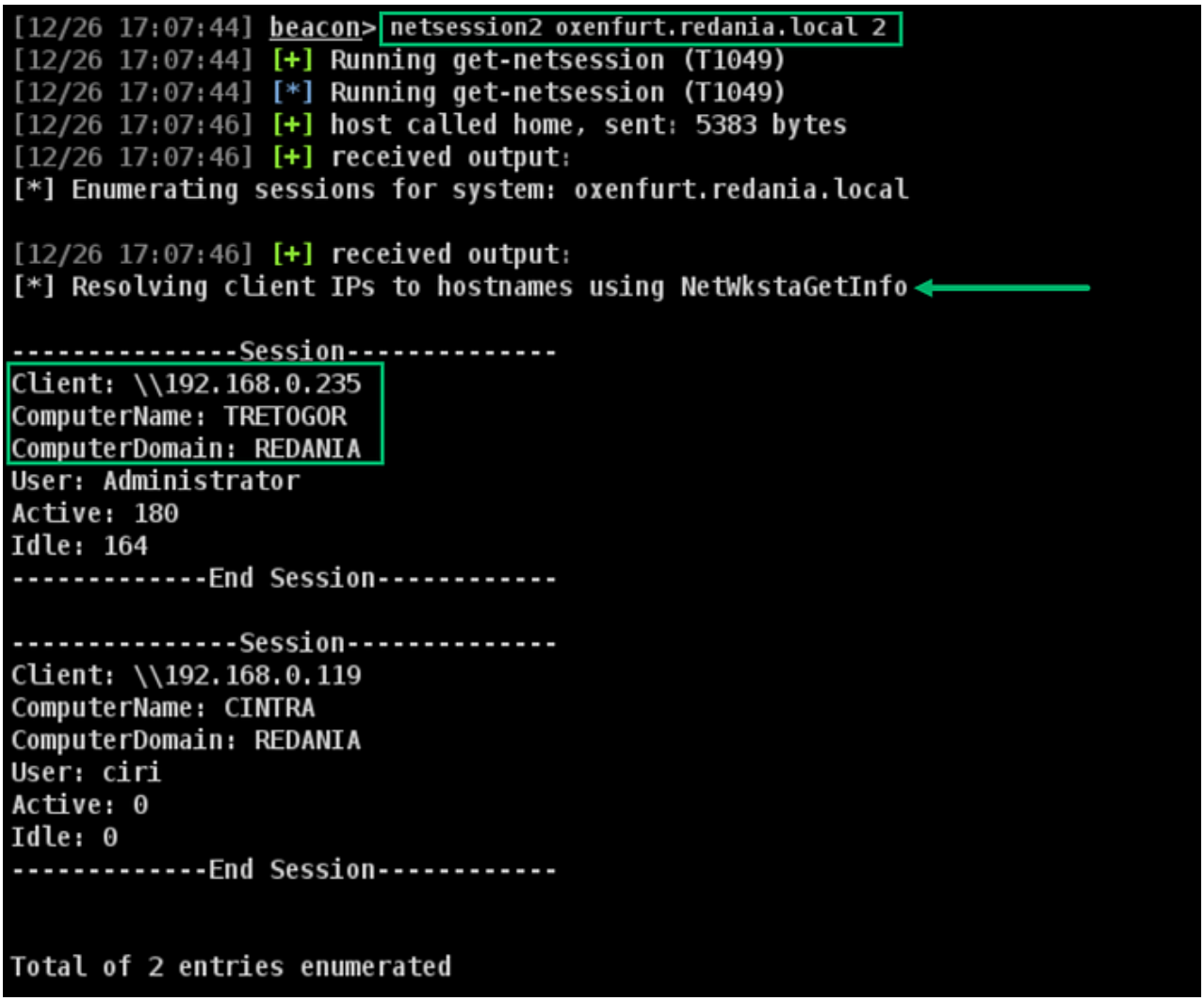


Figure 8 — Sessions Gathered With NetSessionEnumand NetWkstaGetInfo Resolution

The data returned from `NetWkstaGetInfo` presents one last hurdle for parsing. Since the API returns NetBIOS names, we can no longer match a computer object on `dNSHostName`. Instead, BOFHound has to:

1. Map the NetBIOS domain name to a known LDAP domain object, containing the domain SID
2. Find a computer object in that domain where the `sAMAccountName` attribute matches the NetBIOS computer name

The NetBIOS domain name translation relies on referral (crossRef) LDAP objects, which contain a `nETBIOSName` attribute. These can be identified with the query below (Figure 9).

```
ldapsearch (netbiosname=*) * 0 ""
"CN=Partitions,CN=Configuration,DC=domain,DC=local"
```

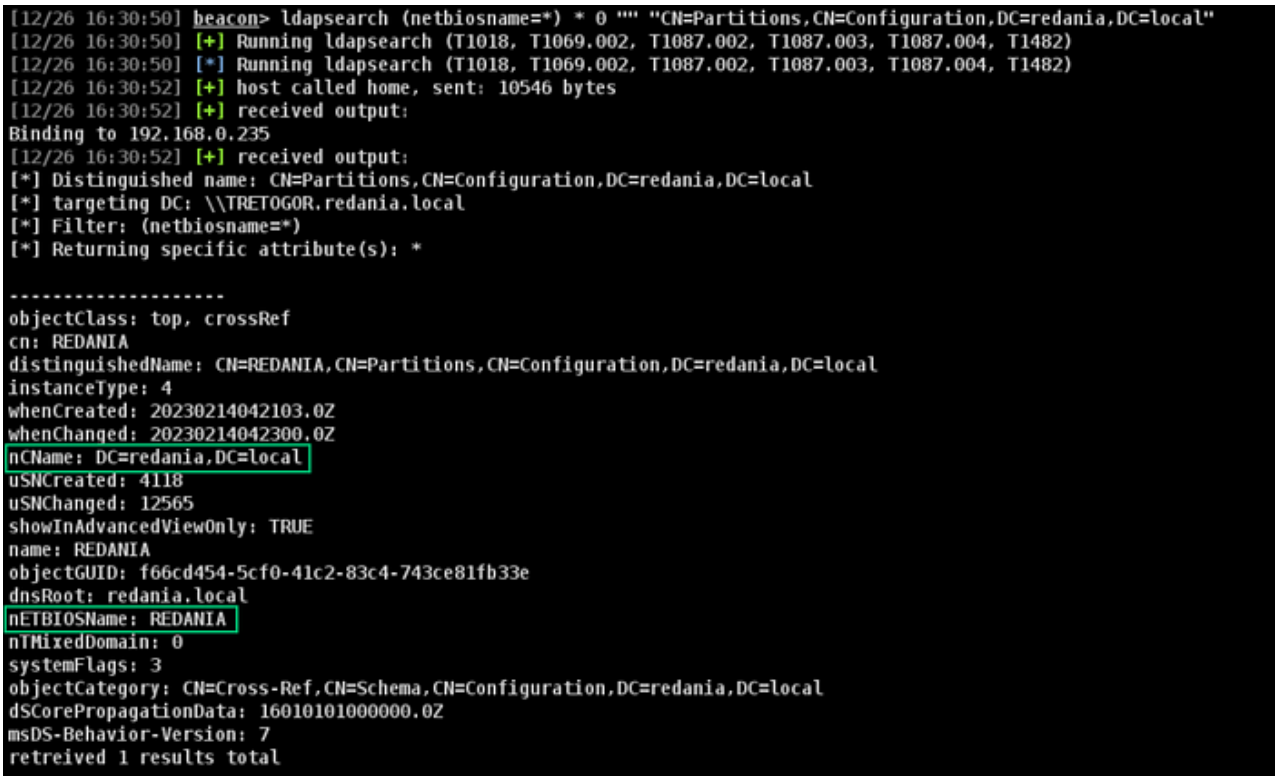


Figure 9 — Referral Object Containing the nETBIOSName Attribute

TL;DR if you opt to use this second method for session enum, execute the above LDAP query so that BOFHound can properly tie the session to a computer.

Local Group Membership Enumeration

Local group memberships allow BloodHound to populate the `AdminTo`, `CanRDP`, `CanPSRemote`, and `ExecuteDCOM` edges. SharpHound queries four local groups for this: *ADMINISTRATORS*, *REMOTE DESKTOP USERS*, *REMOTE MANAGEMENT USERS*, and *DISTRIBUTED COM USERS*.

The existing `netLocalGroupListMembers` BOF already allows us to query these groups. With a modification to one of the returned structures, we can also obtain group members' SIDs and SID types — much better for tying back to LDAP objects than just member names (Figure 10).

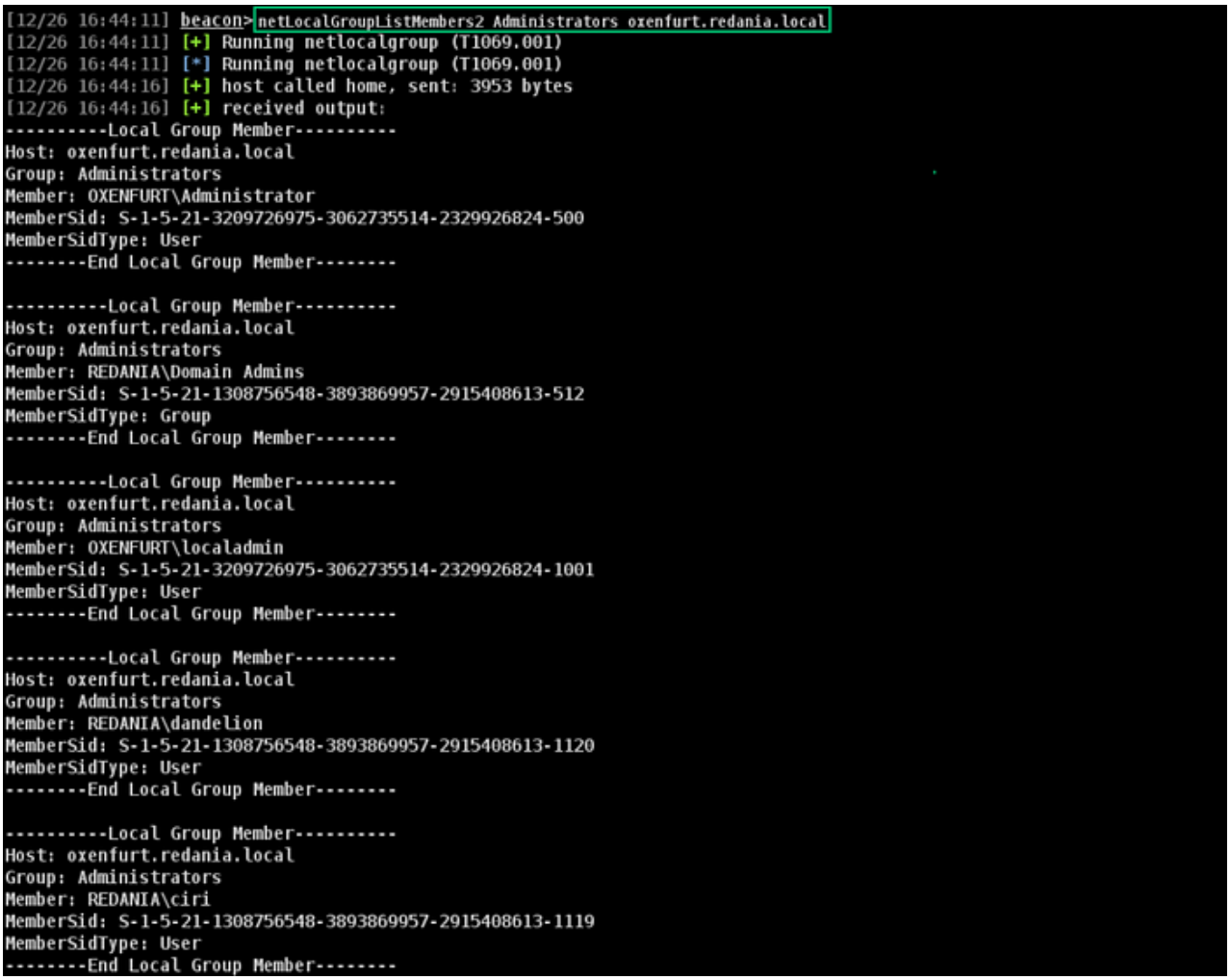


Figure 10 — Local Group Membership Collection

The modified BOF can also be run without a group name, using `""` (e.g., `netLocalGroupListMembers2 "" host.domain.local`), to query all four of the groups we care about.

Visualization

With all the enumeration we're interested in out of the way, we can parse the Beacon logs and load the results into BHCE! The new version of BOFHound doesn't introduce any new command-line interface flags, so we can simply run it as usual (Figure 11).

```
[2023-12-26 5:21:28] root /opt/bofhound
# poetry run bofhound -i /home/defaultuser/Toolkit/cobaltstrike/logs
[17:21:47] INFO     Located 1 beacon log files
[17:21:47] INFO     Parsed 259 LDAP objects from 1 log files
[17:21:47] INFO     Parsed 33 local group/session objects from 1 log files
[17:21:47] INFO     Sorting parsed objects by type...
[17:21:47] INFO     Parsed 10 Users
[17:21:47] INFO     Parsed 52 Groups
[17:21:47] INFO     Parsed 3 Computers
[17:21:47] INFO     Parsed 1 Domains
[17:21:47] INFO     Parsed 0 Trust Accounts
[17:21:47] INFO     Parsed 1 OUs
[17:21:47] INFO     Parsed 3 GPOs
[17:21:47] INFO     Parsed 0 Schemas
[17:21:47] INFO     Parsed 1 Referrals
[17:21:47] INFO     Parsed 188 Unknown Objects
[17:21:47] INFO     Parsed 4 Sessions
[17:21:47] INFO     Parsed 3 Privileged Sessions
[17:21:47] INFO     Parsed 3 Registry Sessions
[17:21:47] INFO     Parsed 5 Local Group Memberships
[17:21:47] INFO     Parsed 0 ACL relationships
[17:21:47] INFO     Created default users
[17:21:47] INFO     Created default groups
[17:21:47] INFO     Resolved group memberships
[17:21:47] INFO     Resolved delegation relationships
[17:21:47] INFO     Resolved OU memberships
[17:21:47] INFO     Linked GPOs to OUs
[17:21:47] INFO     Resolved local group memberships
[17:21:47] INFO     Resolved sessions
[17:21:47] INFO     JSON files written to current directory
```

Figure 11 — Log File Processing With

After we load the JSON files into BHCE, we can run a pathfinding query starting from our compromised user or host (`CIRI@REDANIA.LOCAL` or `CINTRA@REDANIA.LOCAL`) and ending at the *DOMAIN ADMINS* group (Figure 12).

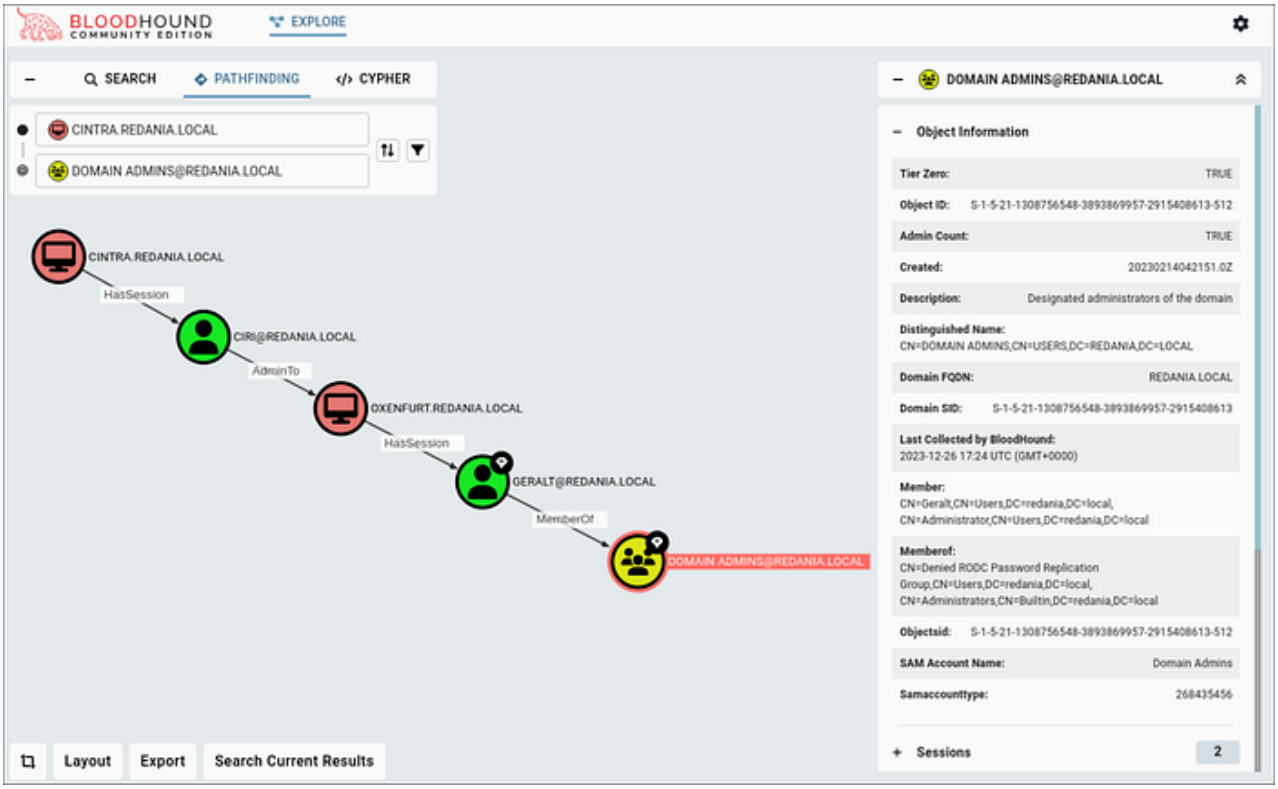


Figure 12 — Attack Path Leveraging HasSession and AdminTo Edges

We can see a path utilizing some of the session and local group data queried via the BOFs!

Augmenting SharpHound's DCOnly Collection Method

Session and local group collection is often the noisiest part of SharpHound execution. It's not entirely uncommon to be in a scenario where LDAP object collection via SharpHound is on the table, but session/local group collection is off limits. In one of these situations, it's possible to combine approaches using SharpHound's `DCOnly` collection method and then manually enumerating sessions with the new BOFs to add to SharpHound's data pull (similar concept to using SharpHound's `-ComputerFile` flag).

This is fairly straightforward to do — after blowing away my graph database container, we can start with a `DCOnly` collection and upload it to our BHCE instance (Figure 13).

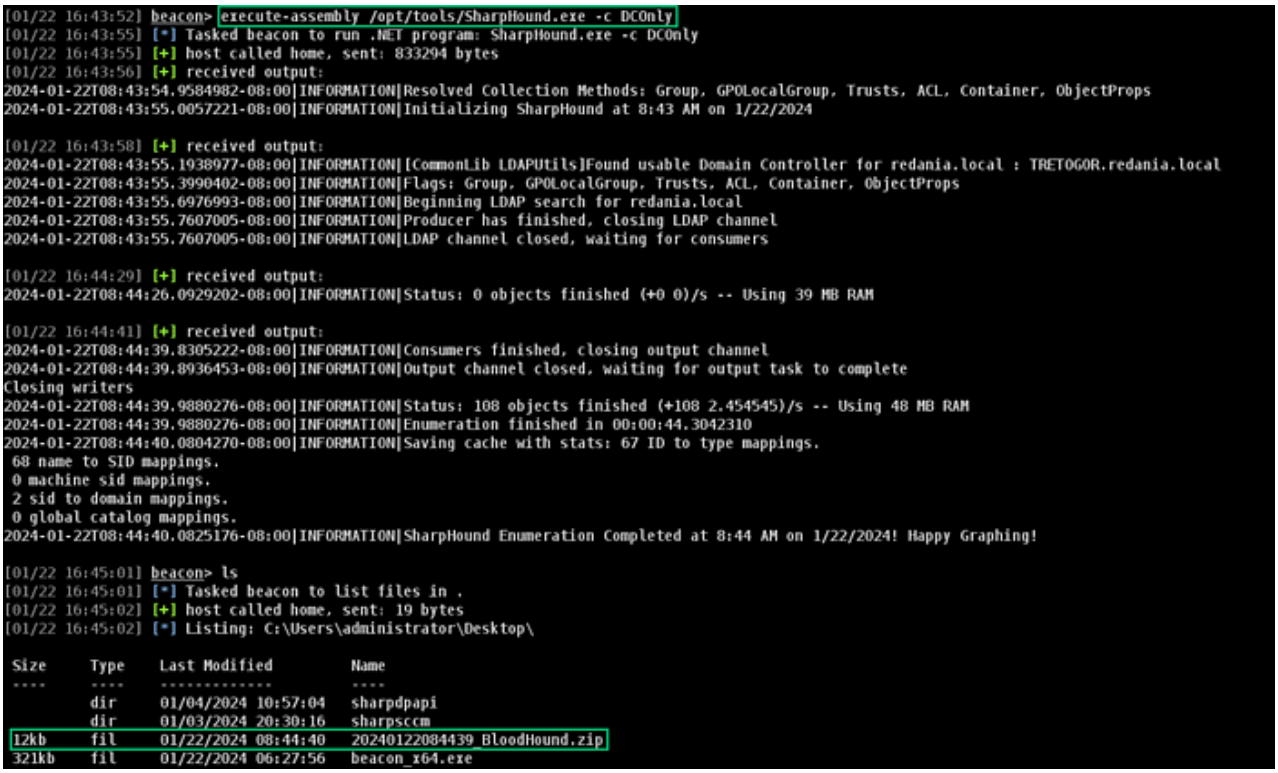


Figure 13 — SharpHound DCOnly Collection

Once uploaded, we can verify that no sessions or local admins are tied to our test machine, *OXENFURT* (Figure 14).

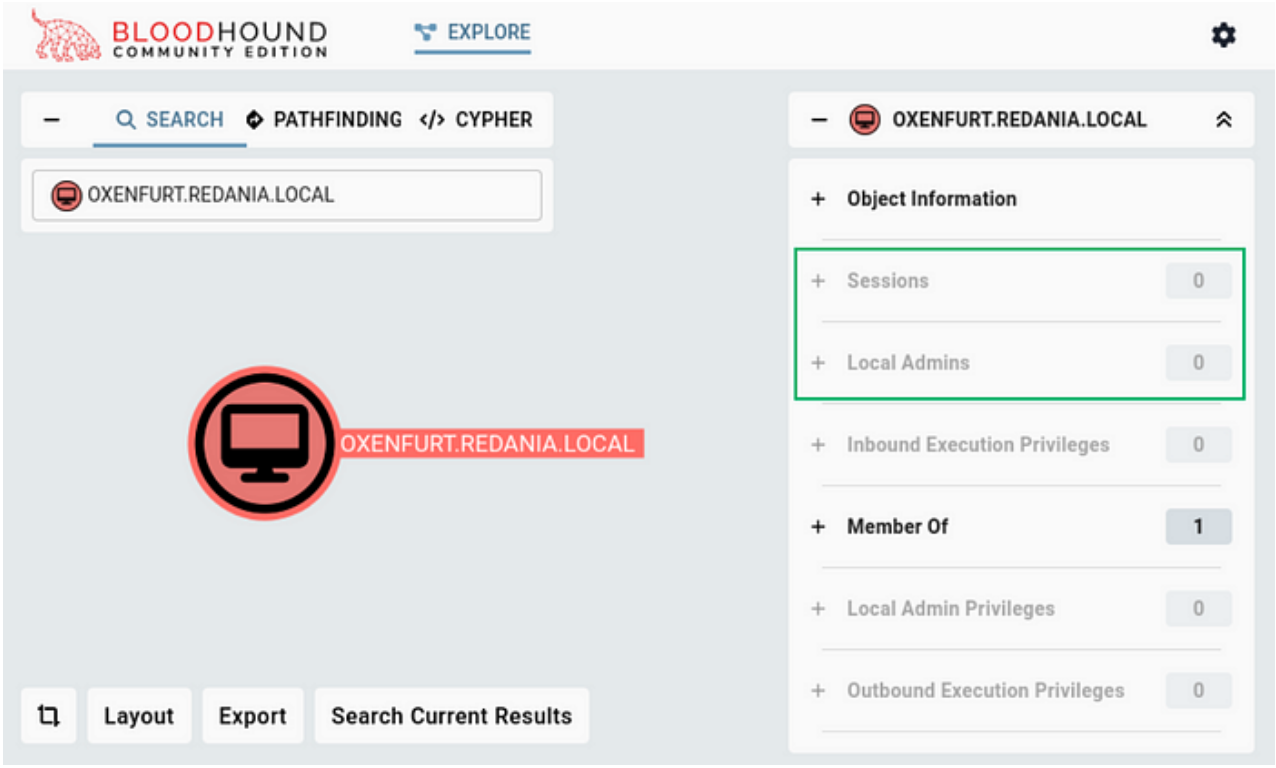


Figure 14 — Empty Session and Local Admin Counts

We can add some local data to this object by querying sessions with regsession and the local admins group with netLocalGroupListMembers2.

```
beacon> regsession oxenfurt.redania.local…beacon> netLocalGroupListMembers2
Administrators oxenfurt.redania.local…
```

Remember that BloodHound session objects are comprised of a user SID and a computer SID. Since we're using the regsession, BOF we already have the SIDs of users with sessions. If using `netsession2` or `netloggedon2`, we'd have to issue an LDAP query for the user object with the corresponding `sAMAccountName` attribute. We will have to do this on the computer side to obtain the computer SID, and also to have a computer object to attach the session and local group data to. The last minimum requirement for this to work is the related domain object, so we'll query that as well.

```
beacon> ldapsearch (samaccountname=oxenfurt$)…beacon> ldapsearch
(objectclass=domain)…
```

The logs generated by those actions contain 1 computer object, 1 domain object, 3 registry sessions and 3 local group memberships for BOFHound to parse (Figure 15).

```
[2024-01-22 5:27:49] root /opt/bofhound
# poetry run bofhound -i /home/defaultuser/Toolkit/cobaltstrike/logs
[17:27:53] INFO     Located 1 beacon log files
[17:27:53] INFO     Parsed 2 LDAP objects from 1 log files
[17:27:53] INFO     Parsed 8 local group/session objects from 1 log files
[17:27:53] INFO     Sorting parsed objects by type...
[17:27:53] INFO     Parsed 0 Users
[17:27:53] INFO     Parsed 0 Groups
[17:27:53] INFO     Parsed 1 Computers
[17:27:53] INFO     Parsed 1 Domains
[17:27:53] INFO     Parsed 0 Trust Accounts
[17:27:53] INFO     Parsed 0 OUs
[17:27:53] INFO     Parsed 0 GPOs
[17:27:53] INFO     Parsed 0 Schemas
[17:27:53] INFO     Parsed 0 Referrals
[17:27:53] INFO     Parsed 0 Unknown Objects
[17:27:53] INFO     Parsed 0 Sessions
[17:27:53] INFO     Parsed 0 Privileged Sessions
[17:27:53] INFO     Parsed 3 Registry Sessions
[17:27:53] INFO     Parsed 3 Local Group Memberships
[17:27:53] INFO     Parsed 0 ACL relationships
```

Figure 15 — Supplemental Session Data Parsing

BOFHound will generate several JSON files as a result, but the only one we need to upload is the computers JSON file. After uploading it to BHCE, a second look at the *OXENFURT* computer shows the sessions and local admins we manually enumerated (Figure 16).

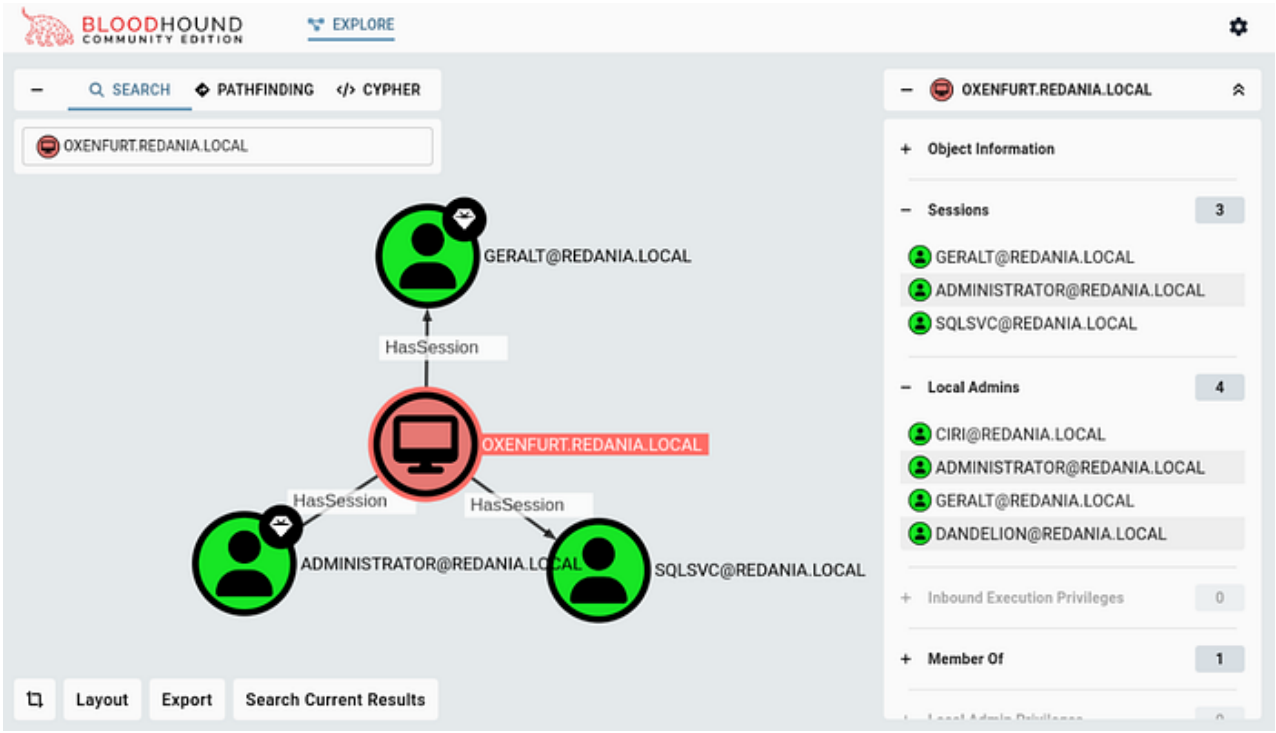


Figure 16 — Manual Addition of Session & Local Group Data

In small enough doses, this approach could provide valuable additions to the `DCOnly` data for pathfinding, while helping avoid the indicators of a full SharpHound session pull.

Conclusion

BOFHound previously served as a solid component of manual approaches to approximate SharpHound's data collection; however, one of the major missing pieces was the ability to manually enumerate session data and local group data and process those alongside LDAP search results to approximate a more complete SharpHound collection. In this post, we examined several new(ish) BOFHound-compatible BOFs and usage examples, that allow an operator to take a manual and targeted approach to attack path mapping that relies on BloodHound's `HasSession` and `AdminTo` edges.