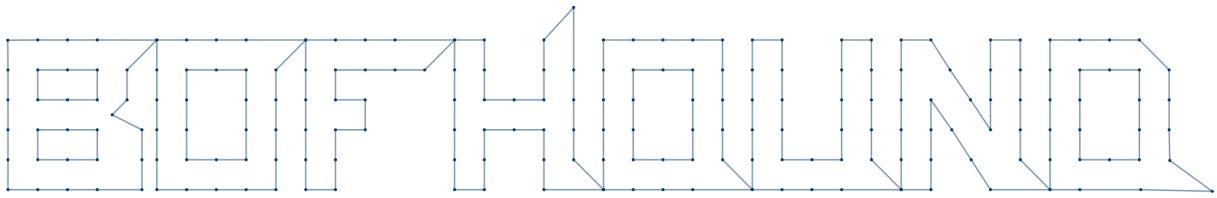
Granularize Your Active Directory Reconnaissance Game

fortalicesolutions.com/posts/bofhound-granularize-your-active-directory-reconnaissance-game



Fortalice BOFHound Release - Granularize Your Active Directory Reconnaissance Game

May 10, 2022

Adam Brown

In the early days of performing offensive testing on environments making use of Active Directory (AD), misconfigurations were a bit more difficult to spot. Teams of testers would scour the directory in search of users with nested group memberships, permissions providing excessive over other AD objects, and attempt to determine what targets may have control of computers based on naming conventions and descriptive object attributes. A few individuals decided they were tired of looking at the output from `net.exe` and `dsquery.exe` while re-running the same queries over and over again and playing the puzzle game. So, they created a tool that the entire Information Security space would embrace: BloodHound.

BloodHound has helped offensive and defensive teams since then conduct efficient and thorough auditing of Active Directory environments. For a while, reviewing Active Directory environments without BloodHound became almost unimaginable, and certainly unattractive. As the tool has evolved and grown, it has become a staple of the offensive tester's toolkit while simultaneously becoming an increasingly desired detection point for defensive teams. Several detection strategies have surfaced over the past 7 years. This post will cover a few helpful detection strategies. Some you may know of, and others, maybe not. Then we'll wrap by introducing two new tools which aim to give red teams a chance at avoiding detection when necessary.

Detection Strategies

Host-Based Indicators

As with any detections, there are several areas in which indicators could occur. Host-based detections are becoming one of the more common areas for offensive operators to get tagged. As such, one strategy for detecting BloodHound usage in your environment (beyond AV/EDR static and heuristic analysis) relies on having insight into the clients and/or servers involved. As MENASEC wrote, Event ID 5145 may be used along with the following indicators from the same source address (excluding domain controllers) to determine that BloodHound is actively running in the environment.

Client-Side

- Multiple outbound SMB and LDAP(S) connections in a short time frame
- Multiple connections to the `lsass` and `srvsvc` named pipes

Server-Side

- Multiple connections to `srvsvc`, `lsarpc`, and `samr`
- Connections to named pipe `srvsvc` with target names containing `Groups.xml` and/or `GpTmpl.inf`

When the above indicators happen within short time intervals from each other, this can be an indication that a BloodHound ingestor is running. However, these detections are focused on the `Default`, and `All` collection methods which involve connections to every computer in the environment.

Honeypot Targets

Honeypot targets are an excellent way to detect an adversary in your environment, including an adversary that is using BloodHound. The use of a honeypot involves enticing an attacker through planned misconfigurations or vulnerable targets, or planting objects in the way of common enumeration patterns in a way that tips off defenders that someone is looking at things that one should be looking at. In the case of BloodHound, the latter is a great way of sniffing out an intruder.

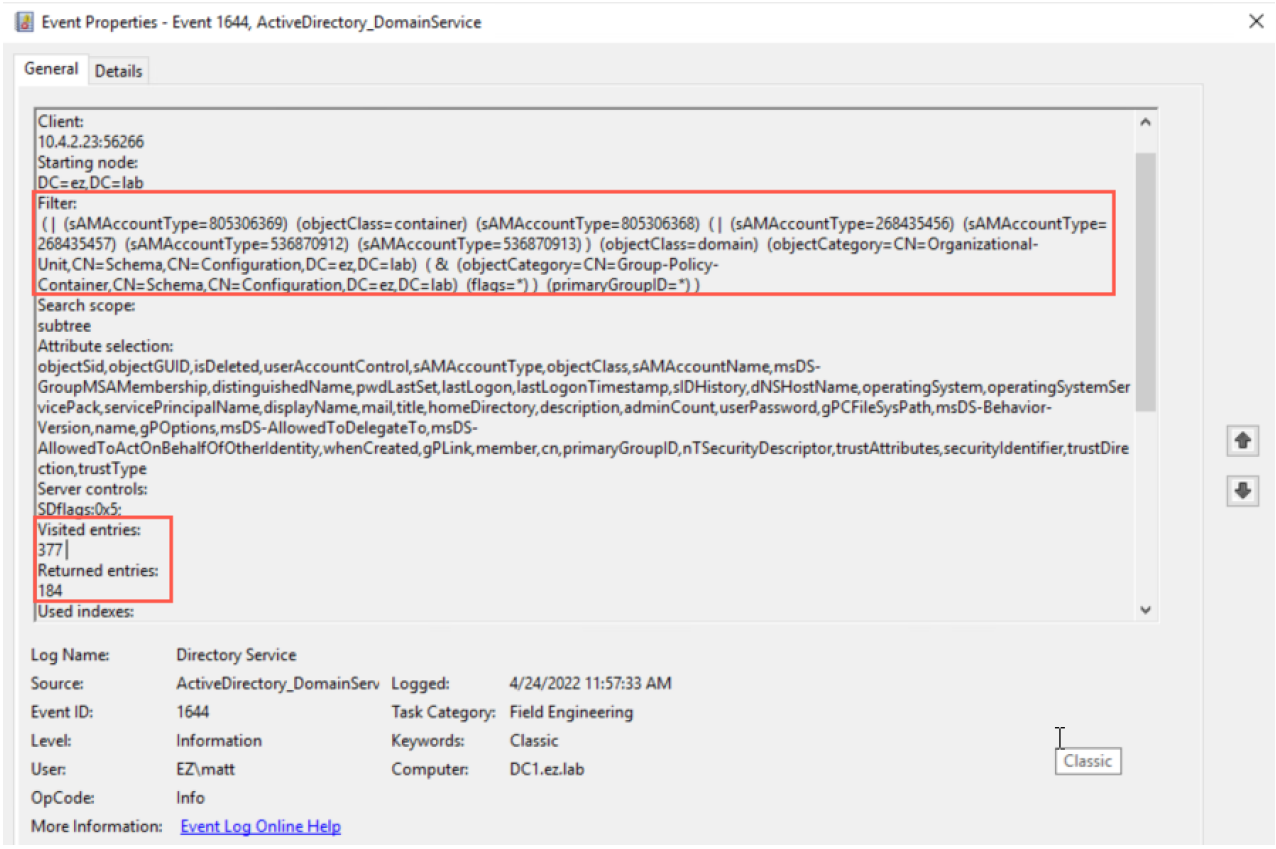
By placing user accounts in the domain that appear real, but have no actual utility, you can enable logs (specifically Event ID 4662) to be generated upon the enumeration of the account. Since BloodHound offers no other option but to get all users or no users, you'll know when BloodHound is run by monitoring for enumeration events on the account. The more honey accounts and groups created, the more accurate the detections can become. This approach will inevitably require some tuning, but is an option for detecting offensive activity early, and sheds light on activity stemming from the `DCOnly` collection method as well.

Expensive LDAP Queries

This last approach is by far one of the more interesting approaches and requires tuning to be effective. Expensive LDAP queries are a feature included by Microsoft that can force the generation of Event ID 1644 any time an LDAP query in the environment accesses too many objects or takes too long to run. If the time and/or efficiency thresholds are tuned, this can provide alerts when BloodHound attempts to use LDAP to query all of the objects in the domain. Between this and honeypot targets, the network becomes a minefield waiting to blow up under your feet as a red teamer.

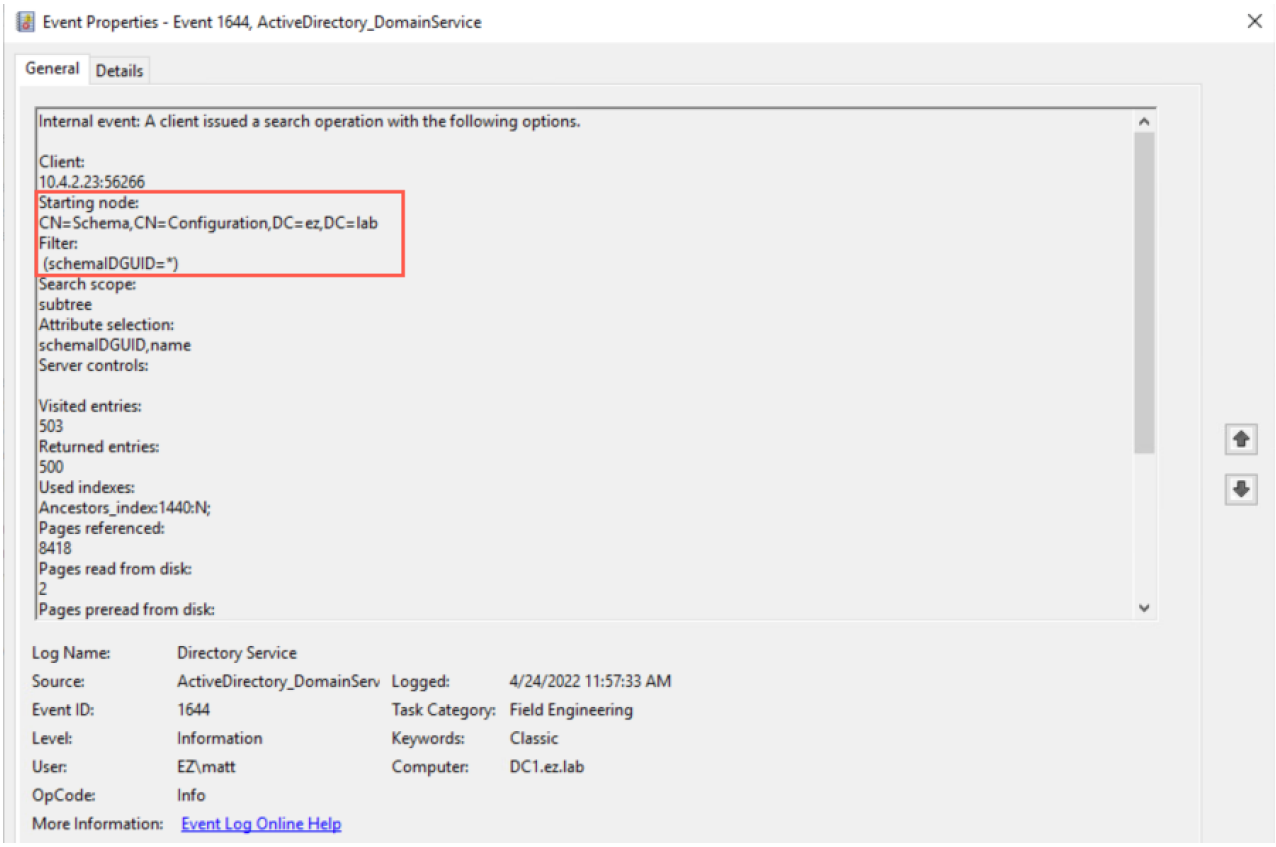
But How Expensive Are They?

As previously hinted, one approach to cutting down the noise generated by BloodHound is to use the `DCOnly` collection method. This will only connect to and query a domain controller, eliminating the excess SMB traffic generated to computers when using the `Default` or `All` collection methods. Let's briefly check out some of the LDAP artifacts on the DC after running a `DCOnly` collection. If we take a look at the *Directory Search* events on the DC, we can begin to get a picture of the LDAP queries that are run under the hood by SharpHound. Immediately, a log showing a query with several wide-ranging `OR` operators stands out.



Copying the filter and issuing this query manually confirms that every domain user, computer, group, and container is returned as a result. Alarmingly, the count of returned entries on this query is 184, and the visited entries field is 377, in a domain with less than 10 combined users and computers.

Returning to the logs, another query that stands out is one with a filter for all schemaIDGUIDs.



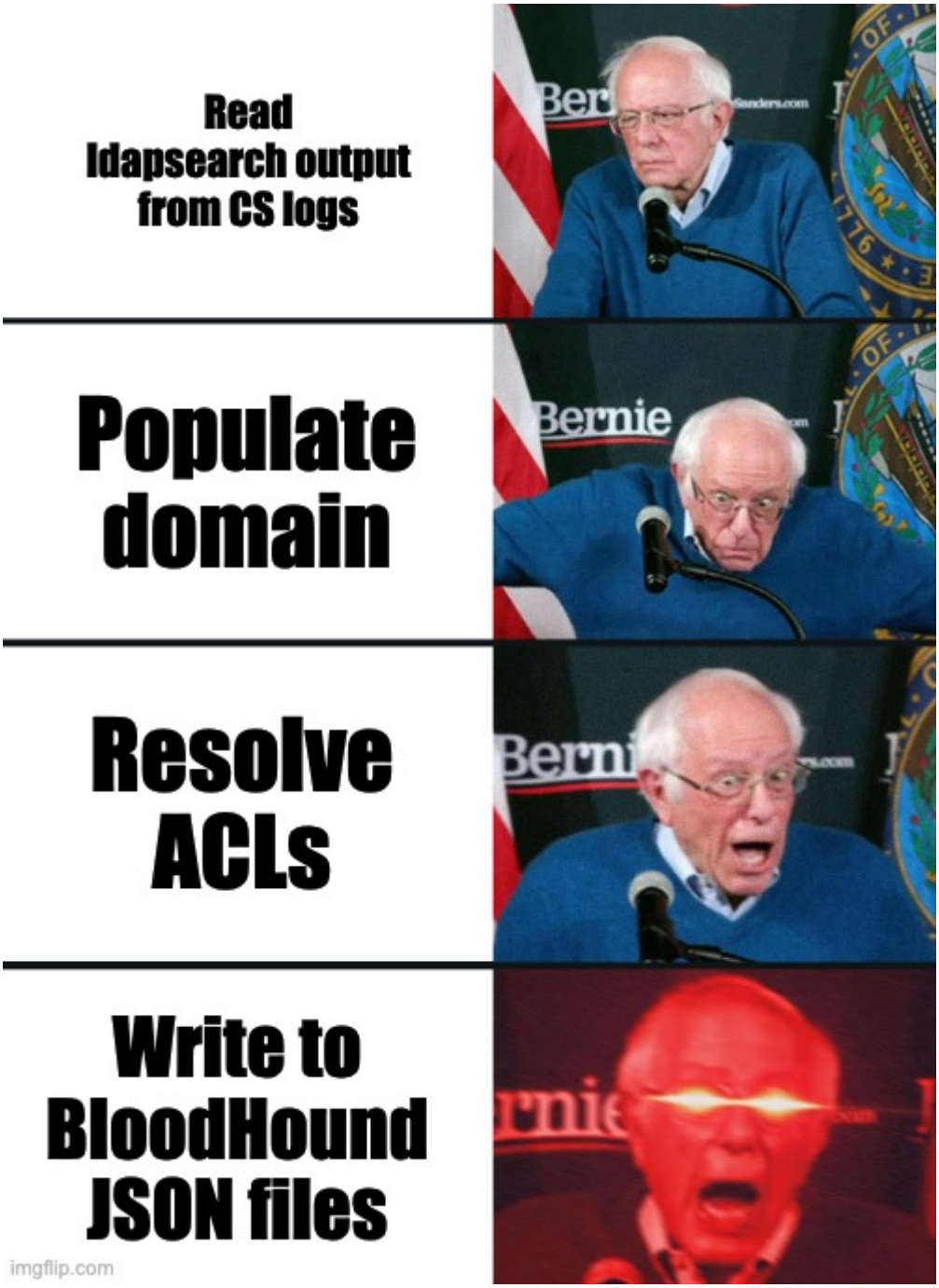
This is used by SharpHound to map out attributes of the AD schema by GUID and name. For the most part, the ones we care about will be static. However, some are dynamic and vary by domain (such as the LAPS related `ms-Mcs-AdmPwd`). While we likely only care about several of these for offensive purposes, this query returns over 1,700 results in the test domain.

These sorts of all-encompassing query filters can send up red flares and are the reason BloodHound, in a DCOnly capacity, is usually off the table when stealth is a concern. So, how are red teams dealing with this?

Avoiding the Indicators

Honestly, a lot of teams have been reverting to granular ways of targeting the data they pull to remain under the radar through the use of old tools similar to `dsquery.exe`. Some incredible tools have been released in the community over the past year or so to assist with this process as well. Utilities such as the `ldapsearch` BOF in TrustedSec's CS-Situational-Awareness-BOF collection, as well as the utilities included in Outflank's C2-Tool-Collection are incredible resources that I think many teams take advantage of. In using some of these tools, I found myself loving the approach, but missing the convenience of quickly digesting the nested misconfigurations often found in BloodHound. I also missed being able to easily rip through the privileges one object had

over others. There had to be a better way; an approach that would merge the old ways with the new. Then, while enumerating current privileges during an operation, I had an idea.



Introducing BOFHound

BOFHound is an offline LDAP result parser and ingestor for BloodHound. BOFHound will parse sets of log files to generate BloodHound compatible JSON files for those times when your team needs more control over the queries being run and the data coming back. With utilities such as the aforementioned `ldapsearch` BOF, BloodHound data can now be pulled low and slow, potentially avoiding the honeypots and expensive LDAP

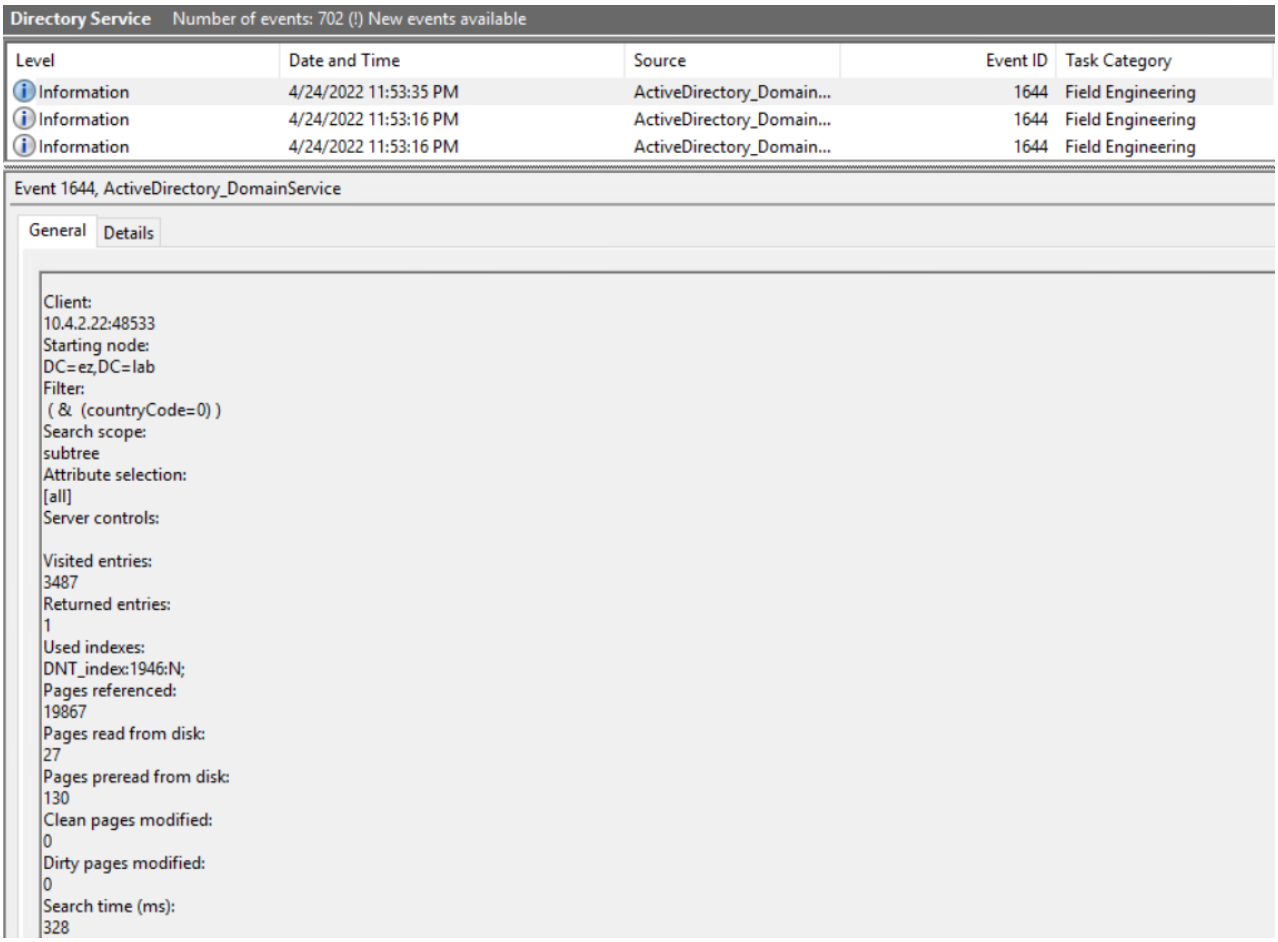
query thresholds, and ingested in a way that allows offensive professionals to use the many features of BloodHound again. While this tool sits on the shoulders of many giants, a huge shoutout has been earned by dirkjanm for the incredible work which we leveraged from BloodHound.py.

BOFHound obviously needs a companion tool for making the LDAP queries, and while I've already mentioned the `ldapsearch` BOF, it originally lacked the ability to pull the `nTSecurityDescriptor` field which is crucial for analyzing inbound and outbound control of objects. With that being said, I would also like to introduce you to BOFHound's companion tool, pyldapsearch! This tool has come in handy on many assessments through the use of SOCKS proxies to pull specific data back from a target environment and store it on disk for ingestion by BOFHound.

While the creation of pyldapsearch has proven fruitful, since TrustedSec's `ldapsearch` BOF played a huge role in the inception of this idea we have also added the ability to pull the `nTSecurityDescriptor` field, as well as the `schemaIDGUID` field, in a pull request to the CS-Situational-Awareness-BOF repository so that teams can also utilize this BOF to generate data compatible with BOFHound.

An OPSEC Note on LDAP Indexing

From the event logs for expensive LDAP queries above, the `Visited entries` and the `Search time (ms)` fields are the ones you'll want to keep low, but it's not as simple as it seems. Take the following screenshot for example…



We can see that a query for `countryCode=0` has visited 3,487 entries even though we have instructed the API to limit the query to one entry. So, while we thought we were being quiet, we still got tagged with an expensive LDAP search. Why is this, and how do we fix it?

LDAP is indexed by default. So, when querying for attributes, you will want to be sure to use an indexed attribute to keep search results as tight as possible. In our experience, `Visited entries` will register as `N + 1`, where `N` is the number of entries to be returned. Keep this in mind while using BOFHound, and if you're unsure of which attributes are indexed, you can use this indexed attributes reference from Microsoft.

Wrapping Up

While this was a fun project and we're so glad to be able to share it with the community, it is not a silver bullet. Great care and consideration are still required when using this methodology, and the previously mentioned detection methods can still be effective even with targeted LDAP queries and AD enumeration in general.

For additional information on Fortalice Solutions service offerings, contact the team via email at watchmen@fortalicesolutions.com.

References

https://github.com/BloodHoundAD/SharpHound.git

https://github.com/fox-it/BloodHound.py.git

https://github.com/trustedsec/CS-Situational-Awareness-BOF.git

https://docs.microsoft.com/en-us/windows/win32/adschema/attributes-indexed

https://blog.menasec.net/2019/02/threat-hunting-7-detecting.html

https://github.com/outflanknl/C2-Tool-Collection.git

Disclaimer

Offensive Cybersecurity Operations