BloodHound Tips and Tricks

medium.com/@riccardo.ancarani94/bloodhound-tips-and-tricks-e1853c4b81ad

Riccardo Ancarani

11 августа 2019 г.

Riccardo Ancarani

This is going to be a quick post on some tips that will make your BloodHound analysis much more fluid and painless 😎. BloodHound is a great tool for both attackers and defenders, but too often people (including myself) only use its top 5% functionalities, leaving a lot of goodies behind.

I'm not inventing anything here, and if you already have experience using BloodHound this may be useless. This is meant to be a collection (and a reminder for myself) of things that I whish I knew from the beginning. Most of this stuff comes from basic cypher and observing what senior consultants did while I was working with them!

A good couple of pre-requisites for appreciating this post:

BloodHound: Intro to Cypher

One of the most overlooked features of BloodHound is the ability to enter raw Cypher queries directly into the user…

blog.cptjesus.com

Introducing the Adversary Resilience Methodology — Part Two

Note: This is the second in a two-part blog series. This companion blog post covers the more technical, prescriptive…

posts.specterops.io

I'll try to update this post whenever I find something useful.

Data Collection

Data gathering is done using the C# ingestor called *SharpHound*. I will not talk too much about it since it's widely known. However I'll just give you my go-to options that I usually use during engagements:

```
SharpHound.exe -c All -s SharpHound.exe -c SessionLoop -s
```

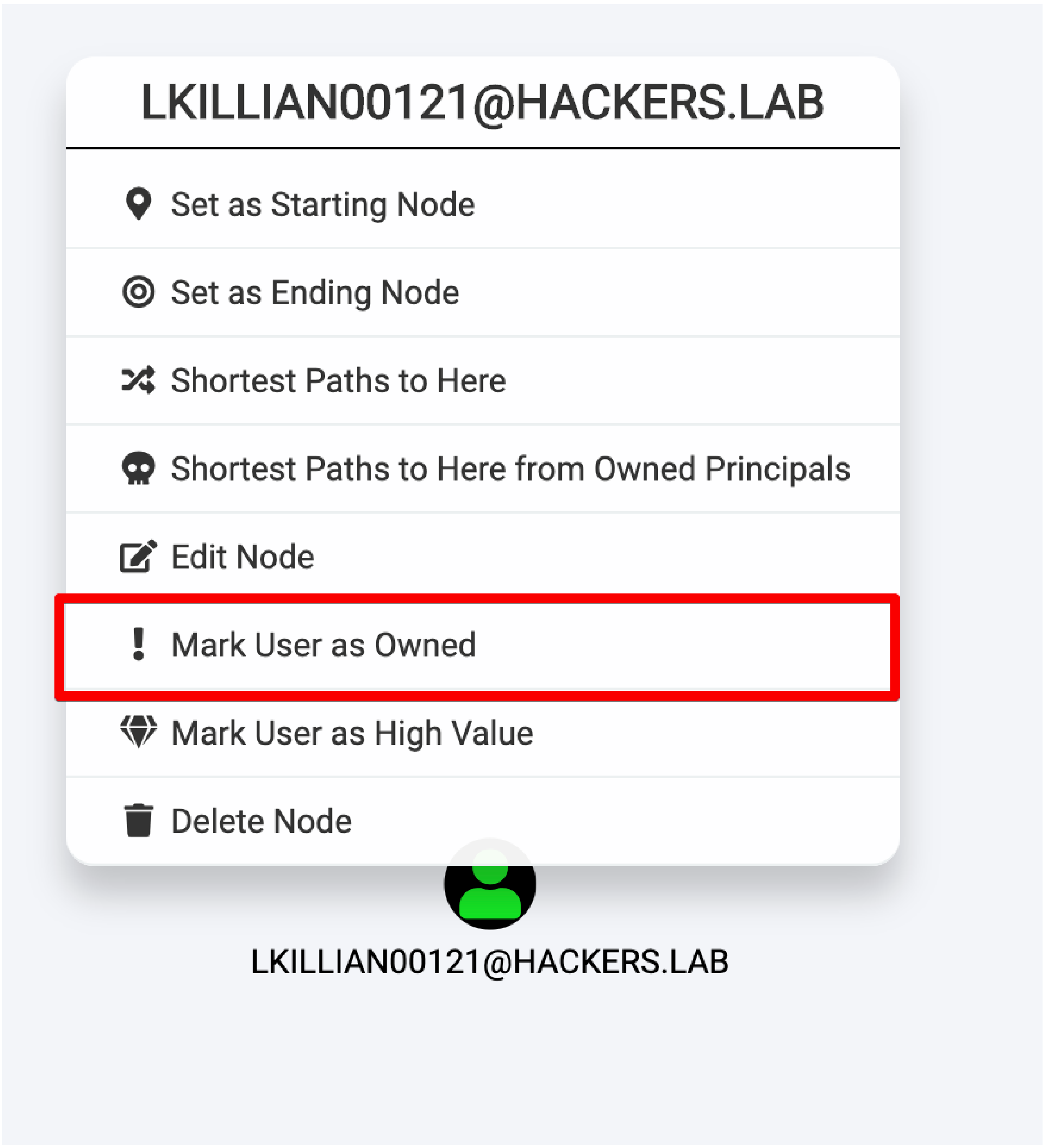
No magic here, in the first command I use all the collection methods available (not really opsec) and expand the search to all the domains within the current forest.

The second command is used for session collection, I usually leave it running for an hour or two every day of the engagement to have some baseline of session activities.

Tips and Tricks

Remember to Mark Things as 'Owned'

This is an easy one, but when you compromise a user or a computer always remember to mark it as 💀 .



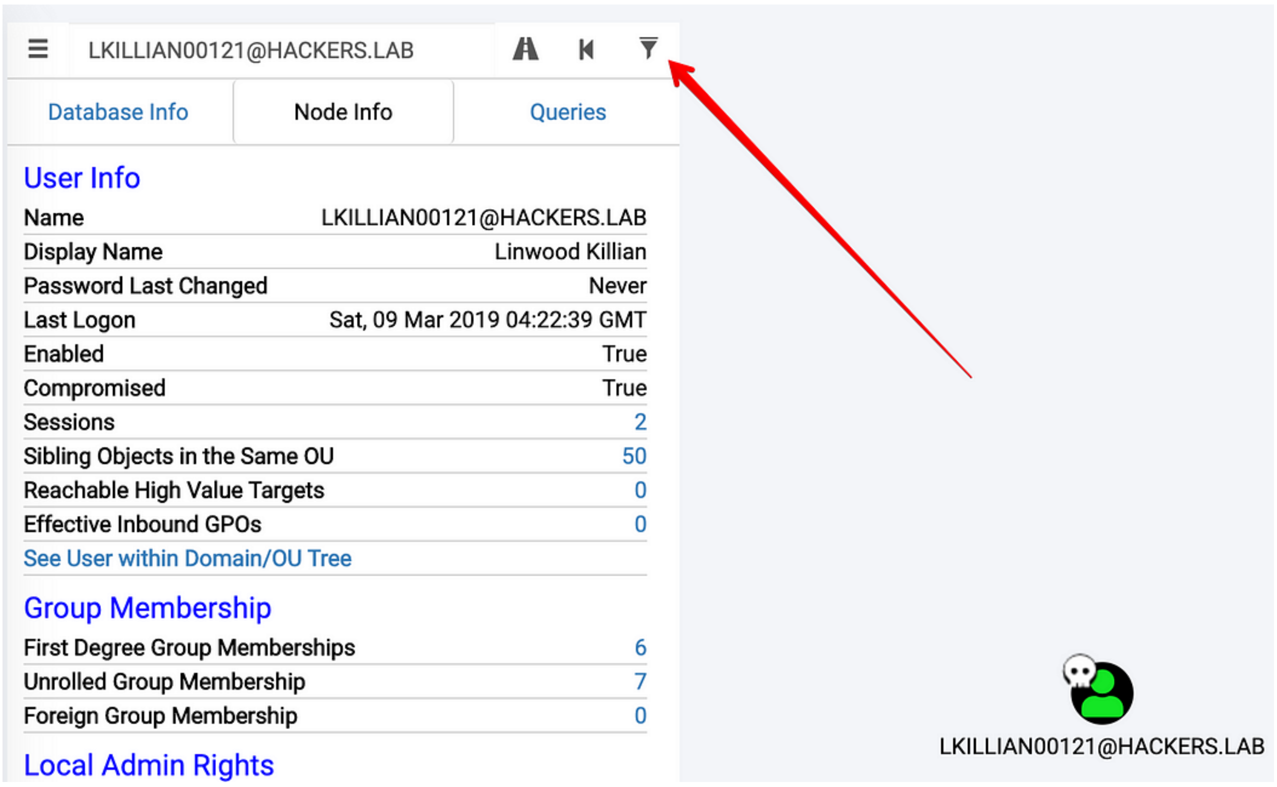
Why? Because a lot of useful built-in queries (and some custom ones that we'll see next) will use the node property `owned` to identify:

- Shortest paths from principals we owned to high value targets
- Show the set of actions that we can execute next fom the principals we owned

Edge Filtering

How often you found a path to DA and you were 🎉 🎉 just before finding that you can only RDP into a box where another high privileged user is logged in (and therefore you need to privesc) and suddenly you become 😞?

Enter edge filtering! There is a small icon in the top of the search box, just right here:



User Info

Name	LKILLIAN00121@HACKERS.LAB
Display Name	Linwood Killian
Password Last Changed	Never
Last Logon	Sat, 09 Mar 2019 04:22:39 GMT
Enabled	True
Compromised	True
Sessions	2
Sibling Objects in the Same OU	50
Reachable High Value Targets	0
Effective Inbound GPOs	0

See User within Domain/OU Tree

Group Membership

First Degree Group Memberships	6
Unrolled Group Membership	7
Foreign Group Membership	0

Local Admin Rights

Click it and remove the damn `CanRDP` relationship or your most hated one. Remember to right-click on the screen and *Reload Query* to apply the changes.

Mark Other Targets as ‘High Value’

Another easy one, by default there is only a small subset of targets that are considered to be high value, however for different reasons it may not be enough:

- Some targets are just skipped, an example of them are the systems with . By default unconstrained delegation systems are not marked as high-value, however if you happen to compromise just one of them within a domain, you’re domain admin 🔫
- The business context is missing, for example the client you’re working for/with may consider a specific SQL server as their 👑 so maybe getting domain admin is not all that matters.

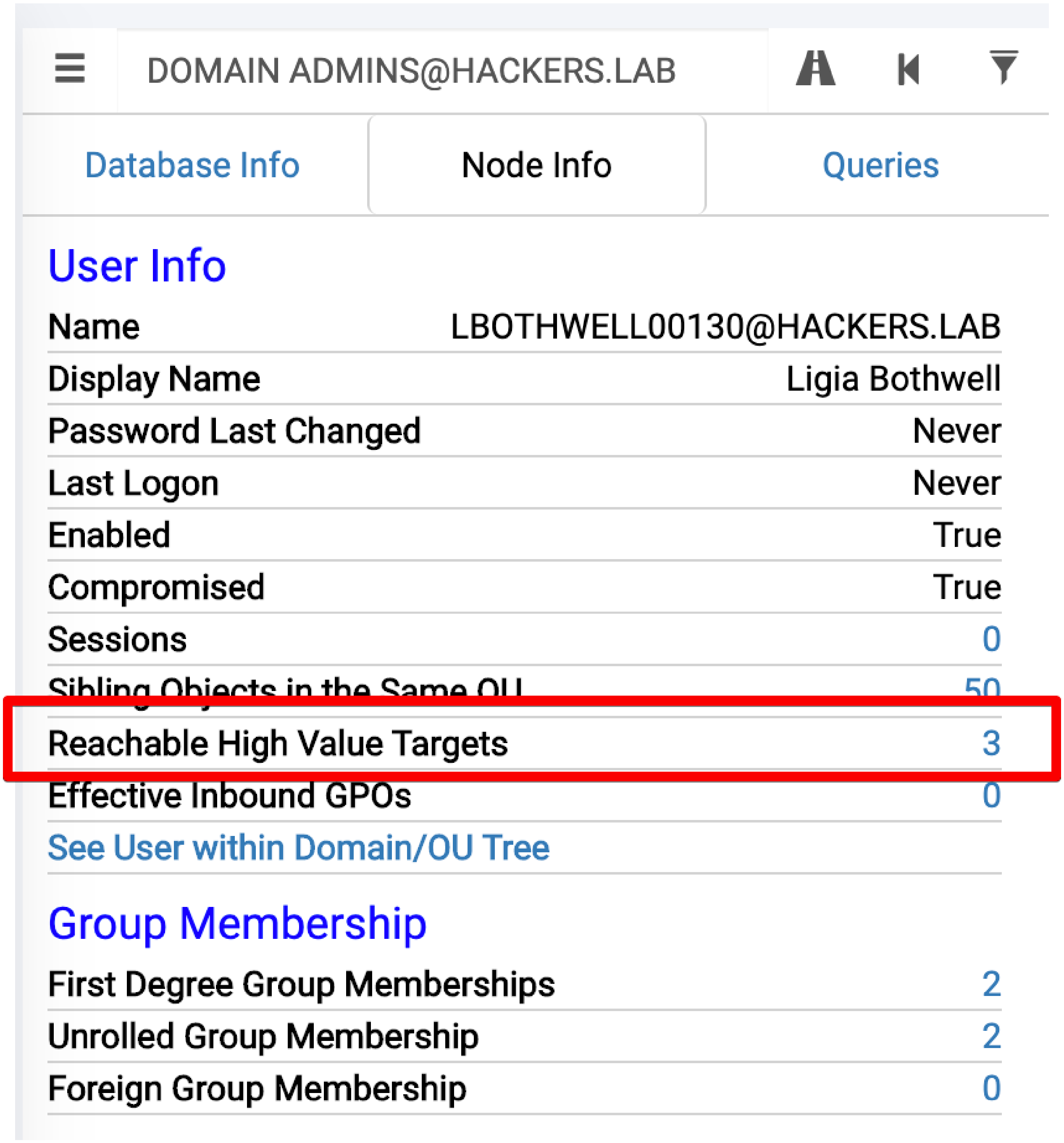
Let’s focus on the unconstrained delegation case, for example you can discover all the unconstrained delegation systems that are not part of the domain controllers group using the following Cypher query (thanks to wald0 and twitter https://twitter.com/_wald0/status/1108660095800479744):

```
MATCH (dc:Computer)-[:MemberOf*1..]->(g:Group) WHERE g.objectsid ENDS WITH "516"
WITH COLLECT(dc) as domainControllers MATCH p = (d:Domain)-[:Contains*1..]->
(c:Computer {unconstraineddelegation:true}) WHERE NOT c in domainControllers
RETURN c
```

We can tweak the query a little bit to mark all the aforementioned systems as high value:

```
MATCH (dc:Computer)-[:MemberOf*1..]->(g:Group) WHERE g.objectsid ENDS WITH "516"
WITH COLLECT(dc) as domainControllers MATCH p = (d:Domain)-[:Contains*1..]->
(c:Computer {unconstraineddelegation:true}) WHERE NOT c in domainControllers SET
c.highvalue = true RETURN c
```

After those mass assignments, always give a look to the reachable high value target pre-compiled field of the node that you owned:



User Info

Name	LBOTHWELL00130@HACKERS.LAB
Display Name	Ligia Bothwell
Password Last Changed	Never
Last Logon	Never
Enabled	True
Compromised	True
Sessions	0
Sibling Objects in the Same OU	50
Reachable High Value Targets	3
Effective Inbound GPOs	0
See User within Domain/OU Tree	

Group Membership

First Degree Group Memberships	2
Unrolled Group Membership	2
Foreign Group Membership	0

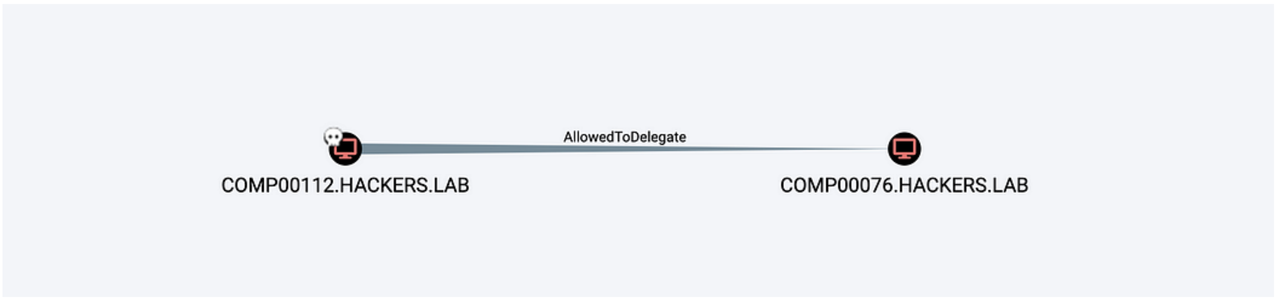
What to do now? a.k.a shortest path to I don’t know

Nice, you kerberoasted a couple of users but there is no direct path from them to DA, what do you do? What else can you still compromise? There is a very nice built-in cyper query to show you that and it’s called *Shortest Path from Owned Principals*.

However, the built-in query is using the `shortestPath` neo4j function without an upper bound limit for relationships, which sometimes it can be good since it will be showing you how far you can go with the users you owned, but some other times you just want to know that's the next step! You can use this query as an example:

```
MATCH p=shortestPath((c {owned: true})-[*1..3]->(s)) WHERE NOT c = s RETURN p
```

An example of result is shown in the figure below:



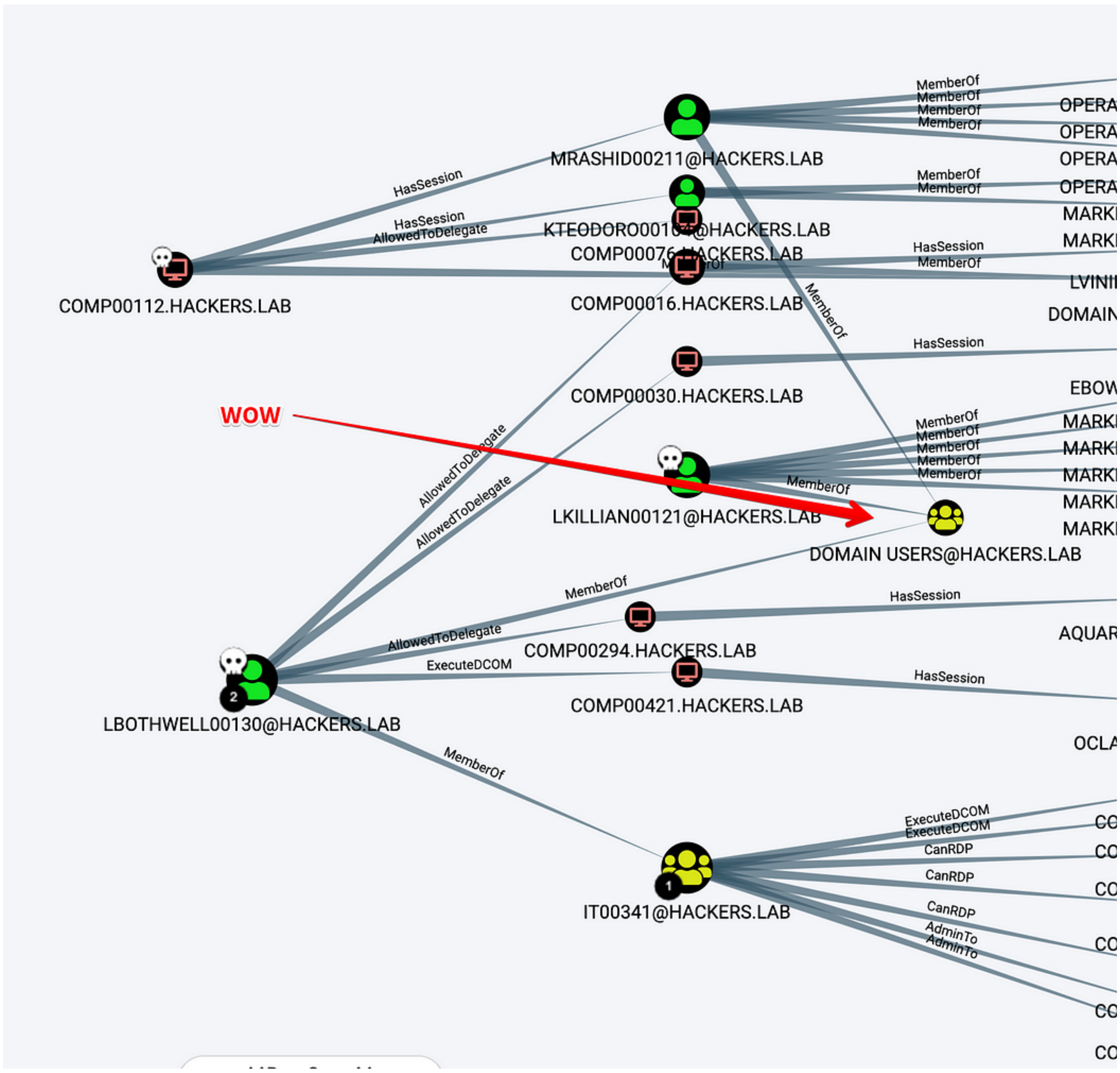
It would be now clear that the next step from the computer we managed to compromise would be to use resource based constrain delegation to access COMP00076 (you can right-click on the edge and for abuse info and resources)

Note that in my cypher query I used `[*1..3]` in the `shortestPath` function, that means that the upper limit for the path search will be 3 hops, you can tweak it as you like but remember that it could take a while 😴.

The original cypher query *Shortest Path from Owned Principals* was something like:

```
MATCH p=shortestPath((c {owned: true})-[*1..]->(s:Computer)) WHERE NOT c = s
RETURN p
```

Note that the target node was set to be a Computer object, while that may be a little bit not enough (what about control over user objects for example?) the query that I showed you before will output some useless information (not too much I promise), such as membership to the Domain Users group 👏 👏 👏 nice finding!



You can tune the query for sure to avoid that level of noise, maybe being explicit in the relationship type or filtering edges. But you know it's Saturday, and Cypher and Saturdays don't go very well together so forgive me if I'm a bit lazy here.

Shortest Path Revisited

One thing that can be very annoying about finding the shortest paths to something within bloodhound is that it's not taking into account the `highvalue` node property. Let's say that you want to find the shortest path to a specific user and the majority of the paths imply that you get DA before 👏 good finding again!

The following cypher query will show you the dodgiest paths to DA or to your favorite target 😉

```
MATCH p=shortestPath((u {highvalue: false})-[*1..]->(g:Group {name: 'DOMAIN
ADMINS@HACKERS.LAB'})) WHERE NOT (u)-[:MemberOf*1..]->(:Group {highvalue: true})
RETURN p
```

Operational Tips

BloodHound is my best friend when it comes to perform AD-related engagements. However, you need to take into account a couple of things when you're using it:

- Local group membership collection requires local admin rights over the target machine. That means that if a group is pushed in the Administrators group instead of using a GPO, you won't be able to see it unless you have local admin rights over that box. If you compromise a user that has local admin rights over a number of boxes, re-run SharpHound again using that user!
- To perform session collectiom, you need connectivity to the target box. If you are in a segmented network and you can see only a couple of machines it could be worth re-running SharpHound again when you obtain more visibility to collect more sessions.

Originally published at on August 11, 2019.