BloodHound - hackndo

en.hackndo.com/bloodhound

Pixis

July 30, 2019

BloodHound

30 Jul 2019 · 13 min

Author : **Pixis**

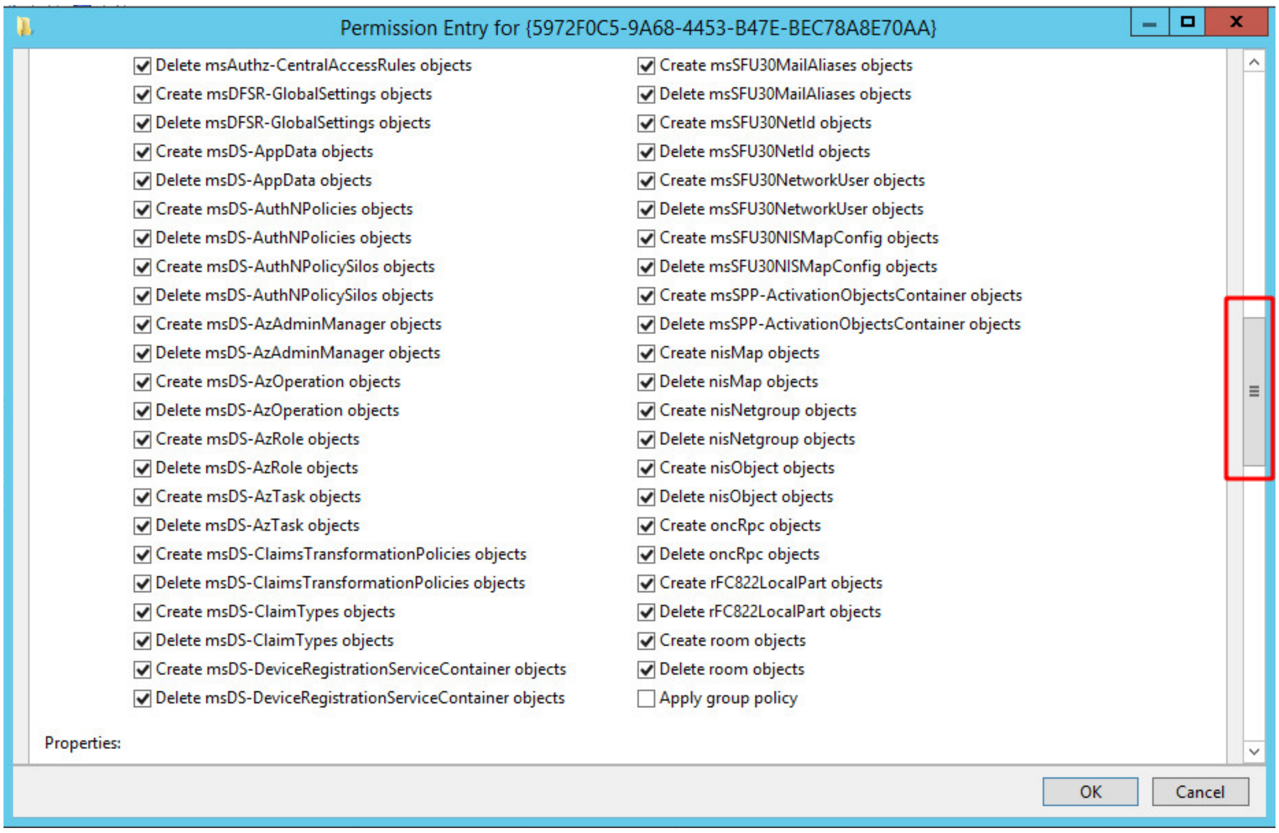
BloodHound is a tool for visualizing an Active Directory environment as a graph. This representation then offers all the power of graph theory to unravel new attack paths that otherwise would have been difficult or impossible to detect.

Active Directory

In an Active Directory environment, access rights management is complex, very complex. It is not uncommon for a user to be part of a group, which is part of 10 other groups, and that one of these groups has the right to modify a GPO which applies to an organizational unit that contains users with administrative rights on a set of hosts. That could go forever.

The issue is that as the administrator of a complex information system these delegations of access rights are extremely complex to detect. Administrators usually look at some access lists that a user or a group has on a particular object. Sometimes there are dozens of different rights that can be applied to an object.

Here is an example of a list of ACE (Access Control Entry) that can be found on a “GPO” object.


Permission Entry for {5972F0C5-9A68-4453-B47E-BEC78A8E70AA}

☑ Delete msAuthz-CentralAccessRules objects	☑ Create msSFU30MailAliases objects
☑ Create msDFSR-GlobalSettings objects	☑ Delete msSFU30MailAliases objects
☑ Delete msDFSR-GlobalSettings objects	☑ Create msSFU30NetId objects
☑ Create msDS-AppData objects	☑ Delete msSFU30NetId objects
☑ Delete msDS-AppData objects	☑ Create msSFU30NetworkUser objects
☑ Create msDS-AuthNPolicies objects	☑ Delete msSFU30NetworkUser objects
☑ Delete msDS-AuthNPolicies objects	☑ Create msSFU30NISMapConfig objects
☑ Create msDS-AuthNPolicySilos objects	☑ Delete msSFU30NISMapConfig objects
☑ Delete msDS-AuthNPolicySilos objects	☑ Create msSPP-ActivationObjectsContainer objects
☑ Create msDS-AzAdminManager objects	☑ Delete msSPP-ActivationObjectsContainer objects
☑ Delete msDS-AzAdminManager objects	☑ Create nisMap objects
☑ Create msDS-AzOperation objects	☑ Delete nisMap objects
☑ Delete msDS-AzOperation objects	☑ Create nisNetgroup objects
☑ Create msDS-AzRole objects	☑ Delete nisNetgroup objects
☑ Delete msDS-AzRole objects	☑ Create nisObject objects
☑ Create msDS-AzTask objects	☑ Delete nisObject objects
☑ Delete msDS-AzTask objects	☑ Create oncRpc objects
☑ Create msDS-ClaimsTransformationPolicies objects	☑ Delete oncRpc objects
☑ Delete msDS-ClaimsTransformationPolicies objects	☑ Create rFC822LocalPart objects
☑ Create msDS-ClaimTypes objects	☑ Delete rFC822LocalPart objects
☑ Delete msDS-ClaimTypes objects	☑ Create room objects
☑ Create msDS-DeviceRegistrationServiceContainer objects	☑ Delete room objects
☑ Delete msDS-DeviceRegistrationServiceContainer objects	☐ Apply group policy

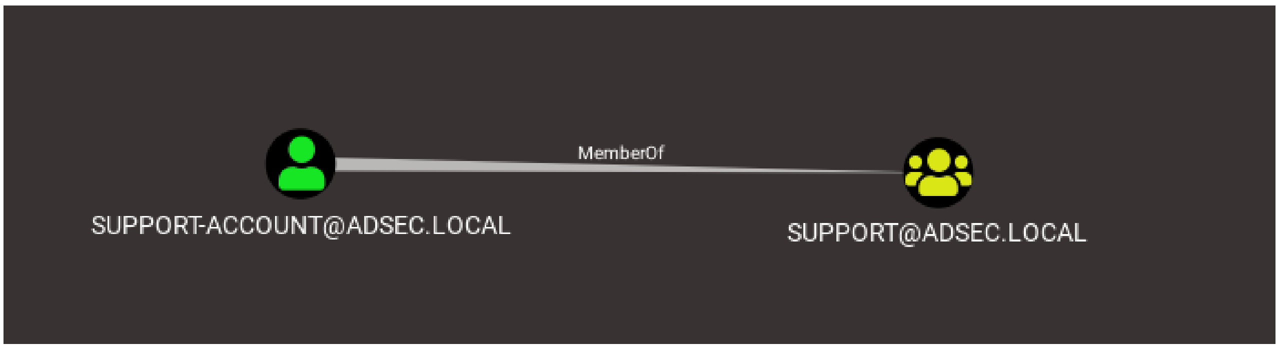
Properties:

The scroll bar on the right shows that there is a very large number of possible entries. Think about the fact that there are entries like these on all objects in the Active Directory, that some of these rights can be applied via delegation, that there are groups included in other groups, inheriting their parent groups rights and so on.

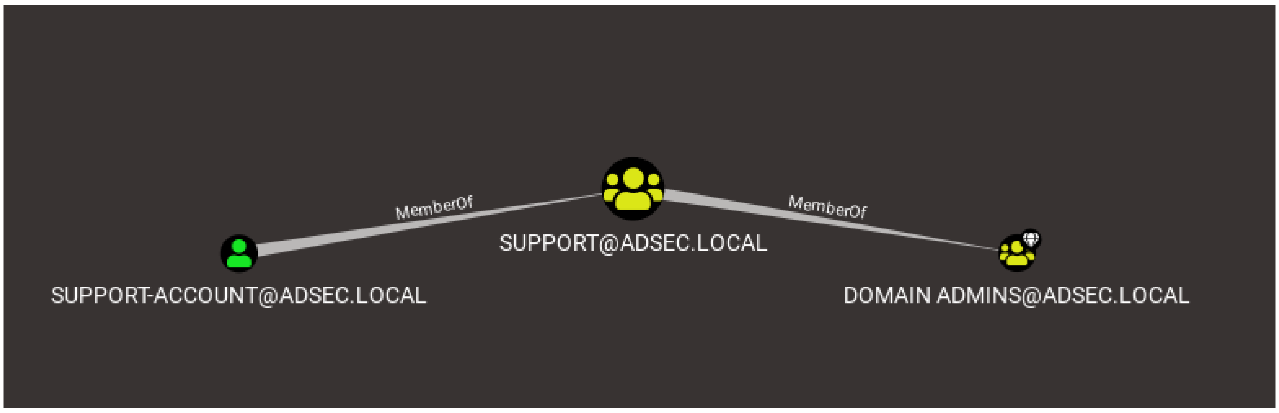
This is the issue BloodHound is trying to solve. When working with lists is a nightmare, working with graphs is way more effective.

BloodHound

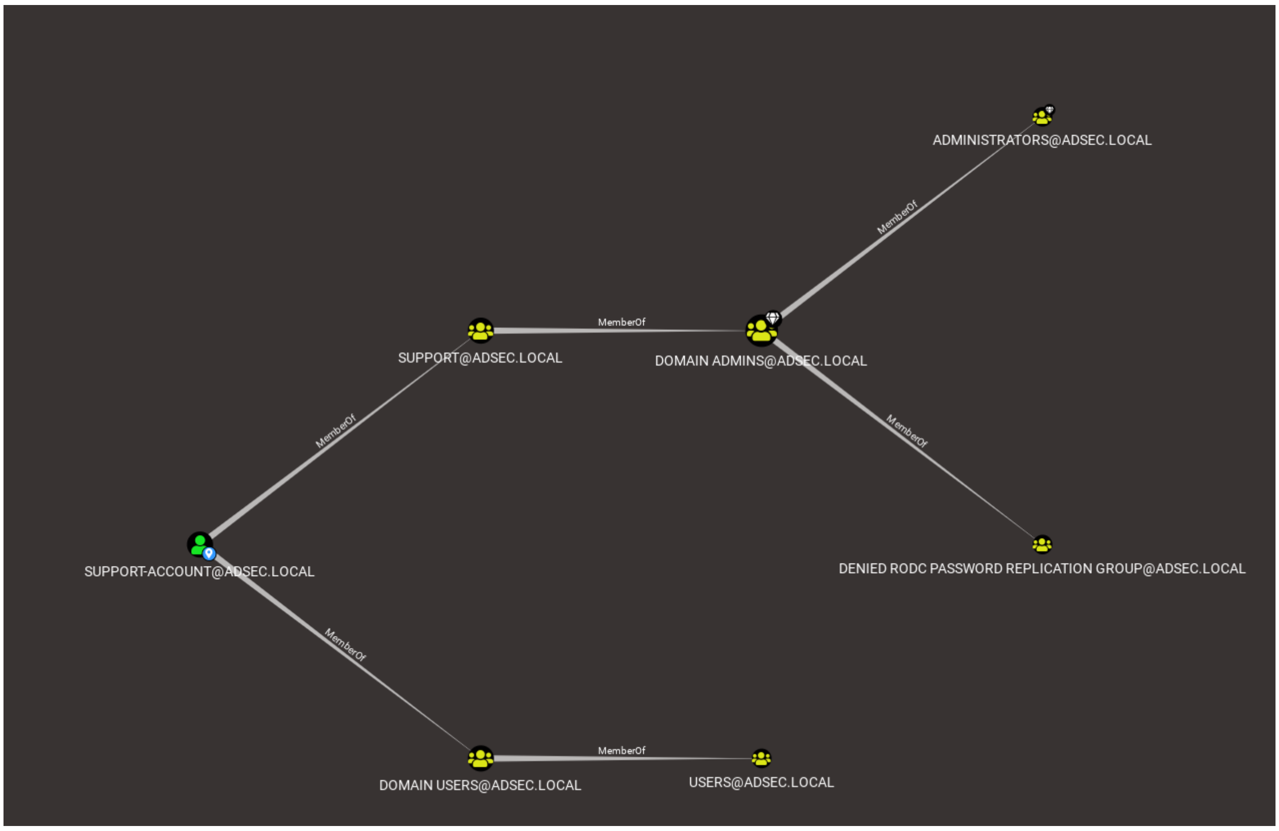
BloodHound is a tool developed by @wald0, @Harmj0y and @CptJesus. The idea of this tool is to analyze an Active Directory environment by enumerating its various objects, and by linking them with some relationships. For example, if the user `support-account` is a member of a group called `support`, the user will be linked with the relationship `MemberOf` to this particular group.



It’s quite easy to visualize this kind of relationship. Something is a **MemberOf** something else, but it doesn’t stop there. It also lets you visualize that the `support` group is a member of the `Domain Admins` group, which means that the `support-account` user is also member of the `Domain Admins` group.



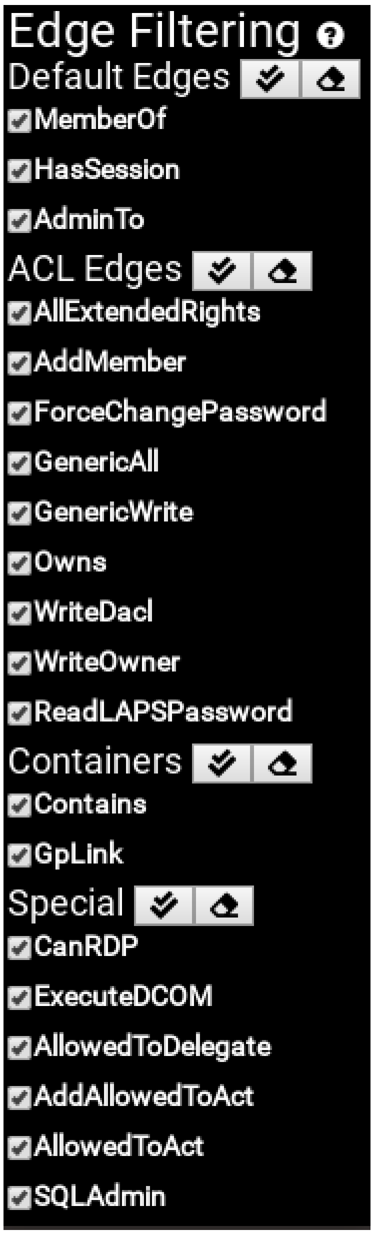
That’s not it! If we ask to display all the groups that `support-account` is a member of, we realize that there is a lot more!


ADMINISTRATORS@ADSEC.LOCAL
MemberOf
SUPPORT@ADSEC.LOCAL
MemberOf
DOMAIN ADMINS@ADSEC.LOCAL
MemberOf
MemberOf
SUPPORT-ACCOUNT@ADSEC.LOCAL
DENIED RODC PASSWORD REPLICATION GROUP@ADSEC.LOCAL
MemberOf
DOMAIN USERS@ADSEC.LOCAL
MemberOf
USERS@ADSEC.LOCAL

And that’s only the `MemberOf` relationship. There are plenty of others. As of today, here is the list of all available relationships.

Ok, so now we have a set of objects linked together with relationships. Now it’s possible to find some attack paths using graph theory.

But wait. What is graph theory?


Edge Filtering

Default Edges
- ✓ MemberOf
- ✓ HasSession
- ✓ AdminTo

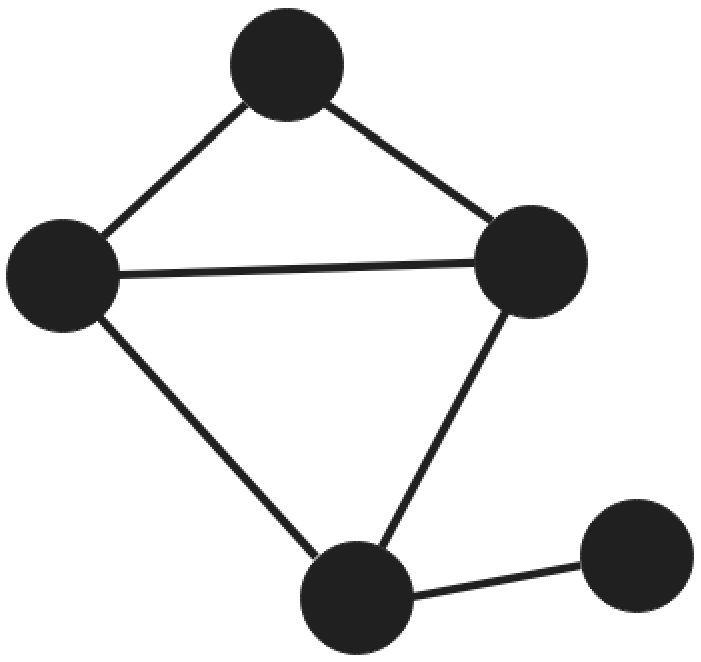
ACL Edges
- ✓ AllExtendedRights
- ✓ AddMember
- ✓ ForceChangePassword
- ✓ GenericAll
- ✓ GenericWrite
- ✓ Owns
- ✓ WriteDacl
- ✓ WriteOwner
- ✓ ReadLAPSPassword

Containers
- ✓ Contains
- ✓ GpLink

Special
- ✓ CanRDP
- ✓ ExecuteDCOM
- ✓ AllowedToDelegate
- ✓ AddAllowedToAct
- ✓ AllowedToAct
- ✓ SQLAdmin

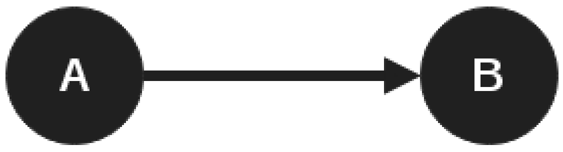
Graph theory

Graph theory is based on a “graph” representation of the data. It is made up of nodes (here the objects in Active Directory) and edges (here the relations between the objects).

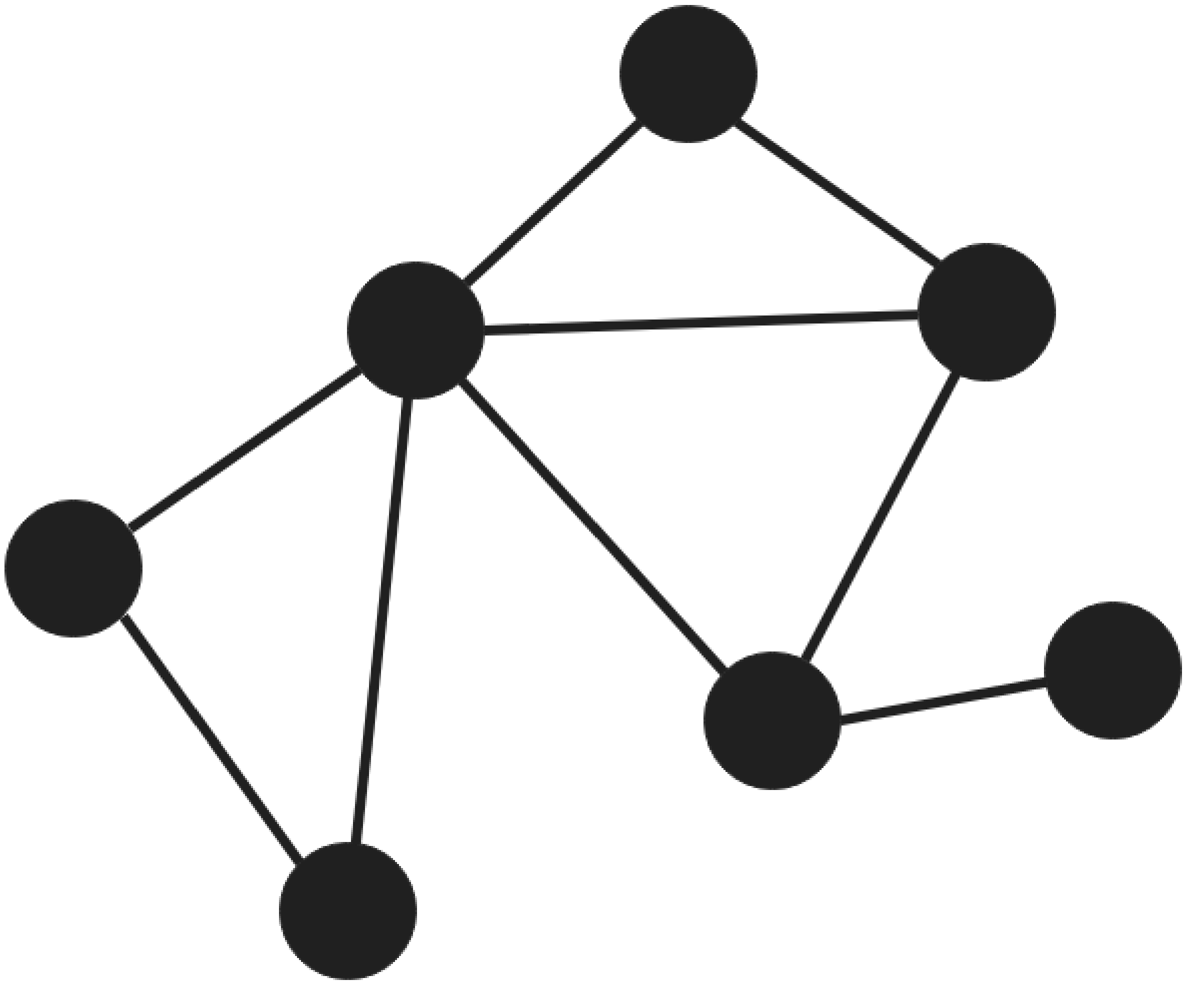


In graph theory, edges can be directed, which is the case with BloodHound. This means that to go from node A to node B, they must be connected by an edge going **from** A **to** B. The other way does not work.

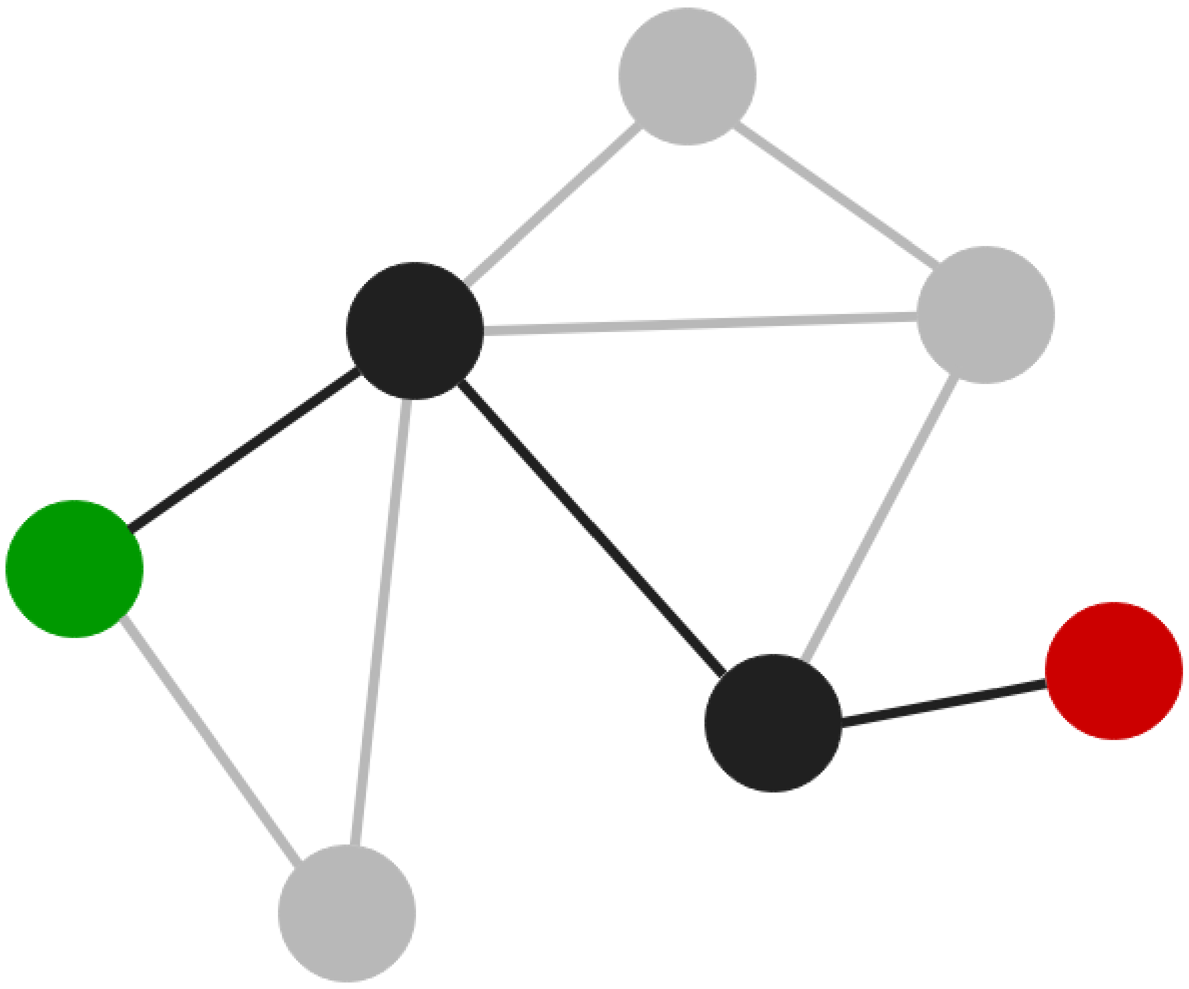
Once we have all these nodes connected by edges, we can look for particular paths to start from a node and arrive at a destination node, passing through a whole set of other nodes. There are often several possible paths, and graph theory makes it possible to find the shortest paths that connect two particular objects.



The following graph consists of 7 nodes and 9 edges.

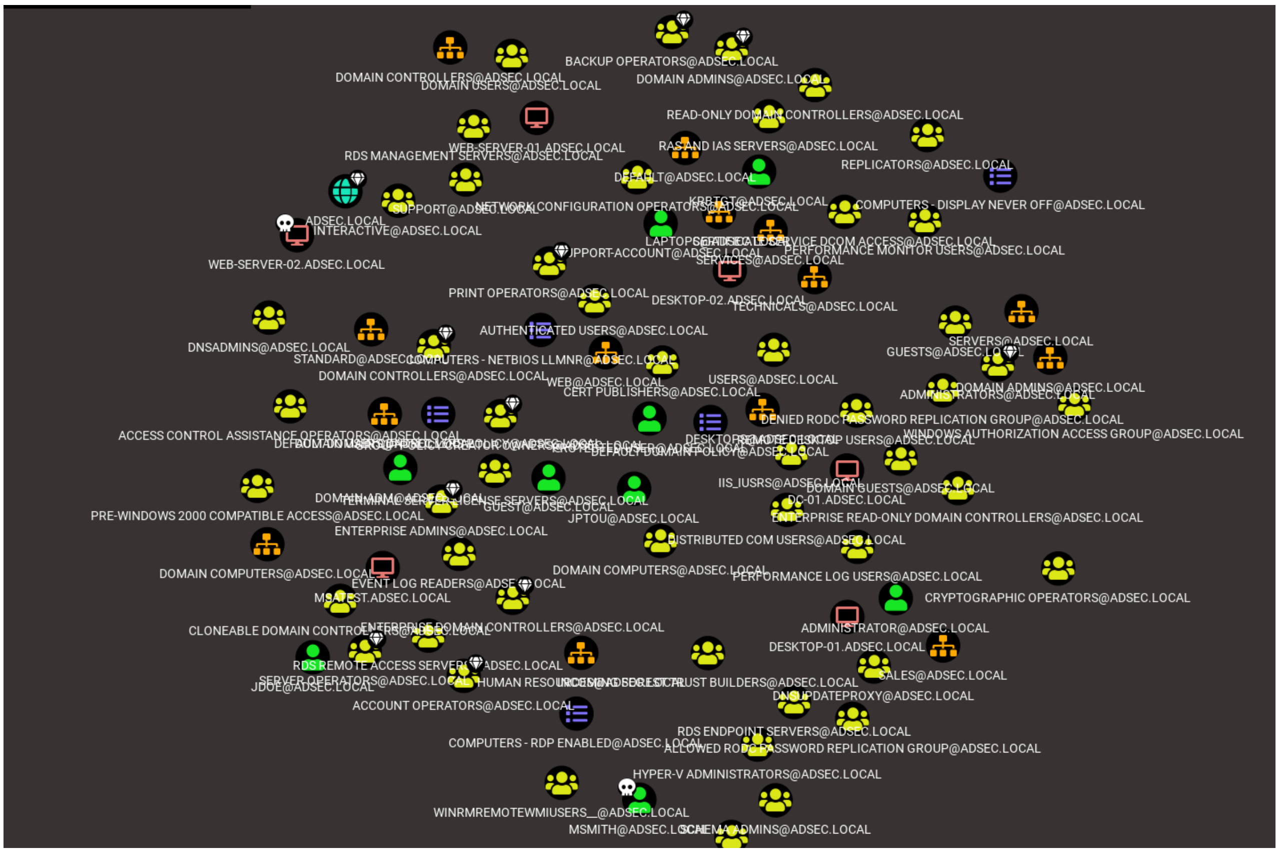


If we managed to take control of the leftmost node, and we wish to reach the rightmost node because it is the `Domain Admins` node, graph theory allows us to find the shortest path between these two.

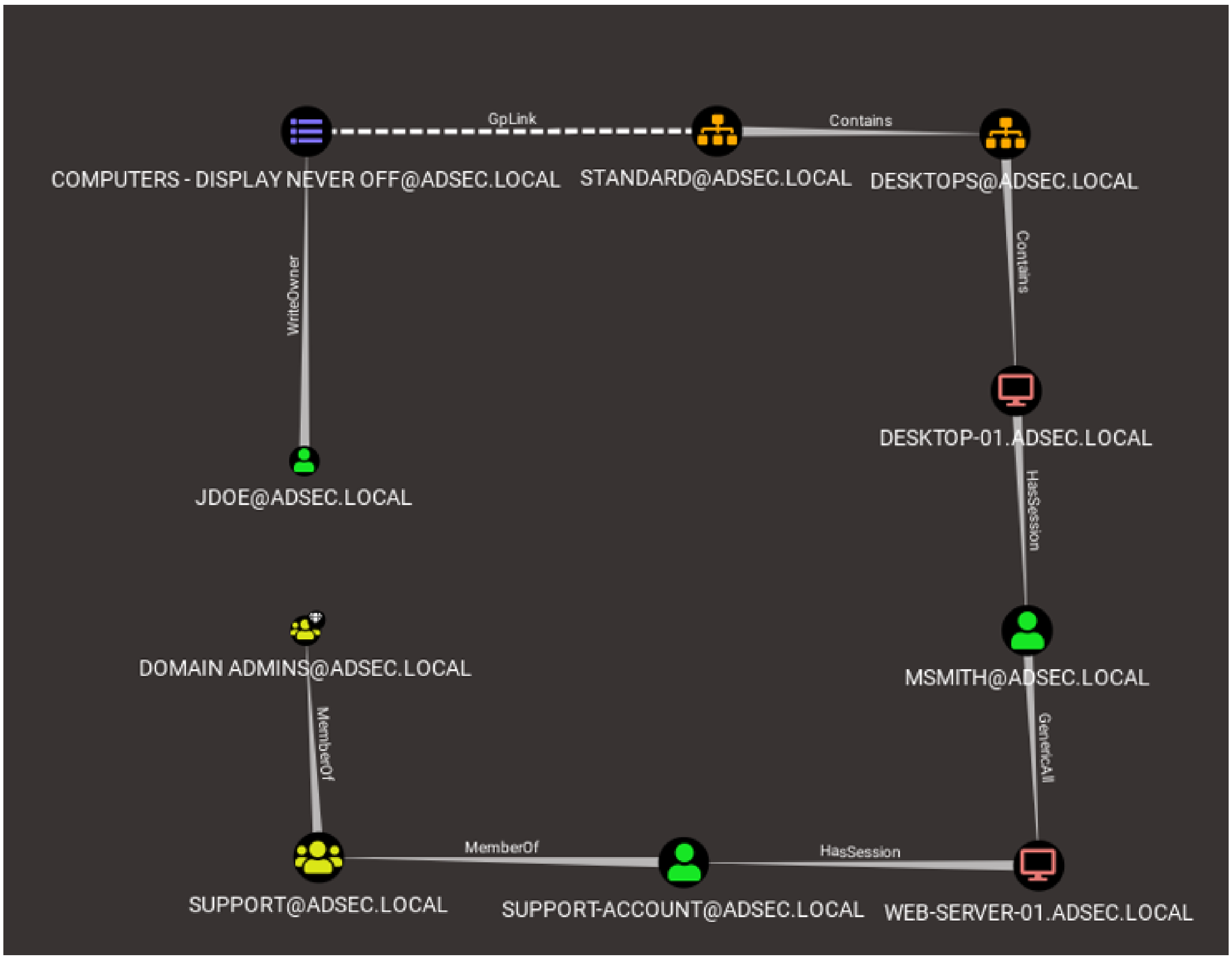


An attacker with this information will know how to move through the network to reach his objective in a minimum number of steps.

Take for example a real Active Directory environment. It is made up of numerous nodes.


BACKUP OPERATORS@ADSEC.LOCAL
DOMAIN CONTROLLERS@ADSEC.LOCAL
DOMAIN USERS@ADSEC.LOCAL
DOMAIN ADMINS@ADSEC.LOCAL
READ-ONLY DOMAIN CONTROLLERS@ADSEC.LOCAL
WEB-SERVER-01.ADSEC.LOCAL
RDS MANAGEMENT SERVERS@ADSEC.LOCAL
RAS AND IAS SERVERS@ADSEC.LOCAL
REPLICATORS@ADSEC.LOCAL
DEFAULT@ADSEC.LOCAL
SUPPORT@ADSEC.LOCAL
NETWORK CONFIGURATION OPERATORS@ADSEC.LOCAL
KRBTGT@ADSEC.LOCAL
COMPUTERS - DISPLAY NEVER OFF@ADSEC.LOCAL
ADSEC.LOCAL
INTERACTIVE@ADSEC.LOCAL
SUPPORT-ACCOUNT@ADSEC.LOCAL
SERVICES@ADSEC.LOCAL
PERFORMANCE MONITOR USERS@ADSEC.LOCAL
WEB-SERVER-02.ADSEC.LOCAL
PRINT OPERATORS@ADSEC.LOCAL
DESKTOP-02.ADSEC.LOCAL
TECHNICALS@ADSEC.LOCAL
AUTHENTICATED USERS@ADSEC.LOCAL
DNSADMINS@ADSEC.LOCAL
STANDARD@ADSEC.LOCAL
COMPUTERS - NETBIOS LLMNR@ADSEC.LOCAL
DOMAIN CONTROLLERS@ADSEC.LOCAL
WEB@ADSEC.LOCAL
USERS@ADSEC.LOCAL
CERT PUBLISHERS@ADSEC.LOCAL
SERVERS@ADSEC.LOCAL
GUESTS@ADSEC.LOCAL
ADMINISTRATORS@ADSEC.LOCAL
DOMAIN ADMINS@ADSEC.LOCAL
DENIED RODC PASSWORD REPLICATION GROUP@ADSEC.LOCAL
WINDOWS AUTHORIZATION ACCESS GROUP@ADSEC.LOCAL
ACCESS CONTROL ASSISTANCE OPERATORS@ADSEC.LOCAL
DEFAULT DOMAIN POLICY@ADSEC.LOCAL
IIS_IUSRS@ADSEC.LOCAL
DOMAIN GUESTS@ADSEC.LOCAL
DC-01.ADSEC.LOCAL
GUEST@ADSEC.LOCAL
JPTOU@ADSEC.LOCAL
PRE-WINDOWS 2000 COMPATIBLE ACCESS@ADSEC.LOCAL
ENTERPRISE ADMINS@ADSEC.LOCAL
ENTERPRISE READ-ONLY DOMAIN CONTROLLERS@ADSEC.LOCAL
DISTRIBUTED COM USERS@ADSEC.LOCAL
DOMAIN COMPUTERS@ADSEC.LOCAL
EVENT LOG READERS@ADSEC.LOCAL
PERFORMANCE LOG USERS@ADSEC.LOCAL
MSATEST.ADSEC.LOCAL
CRYPTOGRAPHIC OPERATORS@ADSEC.LOCAL
CLONEABLE DOMAIN CONTROLLERS@ADSEC.LOCAL
ENTERPRISE DOMAIN CONTROLLERS@ADSEC.LOCAL
ADMINISTRATOR@ADSEC.LOCAL
DESKTOP-01.ADSEC.LOCAL
RDS REMOTE ACCESS SERVERS@ADSEC.LOCAL
SALES@ADSEC.LOCAL
SERVER OPERATORS@ADSEC.LOCAL
JDOE@ADSEC.LOCAL
INCOMING FOREST TRUST BUILDERS@ADSEC.LOCAL
DNSUPDATEPROXY@ADSEC.LOCAL
ACCOUNT OPERATORS@ADSEC.LOCAL
RDS ENDPOINT SERVERS@ADSEC.LOCAL
COMPUTERS - RDP ENABLED@ADSEC.LOCAL
ALLOWED RODC PASSWORD REPLICATION GROUP@ADSEC.LOCAL
HYPER-V ADMINISTRATORS@ADSEC.LOCAL
WINRMREMOTEWMIUSERS__@ADSEC.LOCAL
MSMITH@ADSEC.LOCAL
SCHEMA ADMINS@ADSEC.LOCAL

If we compromise `jdoe` user, this is the shortest path to reach the `Domain Admins` group.



As we can see, the “shortest” path is quite long, which suggests the environment is quite complex. Despite this complexity, using BloodHound allows one to simply and clearly extract an attack path from a user who didn’t seem to have any special access rights.

In order to be able to use graph theory, BloodHound uses “Neo4j” graph oriented database to store Active Directory data.

How does it work?

BloodHound collects data using different techniques:

First, in order to collect directory data, it is necessary to request it from a domain controller. If we have a domain account, even if it has no particular rights, we can browse by default all the users, hosts, GPOs, ACLs, and other information in the directory.

Furthermore, until recently, it was possible to ask the various hosts for the list of their local groups, which made it possible to know who was local administrator of which host by correlating this information with the membership of the users in the different groups.

Then we are able to enumerate the remote sessions on the hosts using the command line `net session \\computer`. This is an output example:

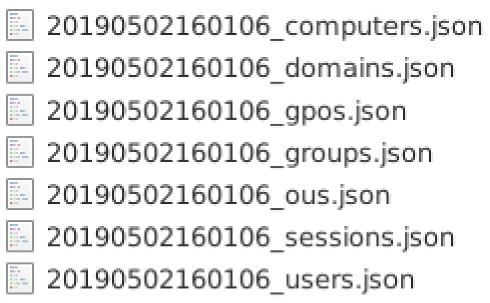
```
net session \\computer

Computer      User name                Client type  Opens   Idle time
---------------------------------------------------------------------
\\SOURCE      SUPPORT-ACCOUNT          Windows 7    1       00:00:13
```

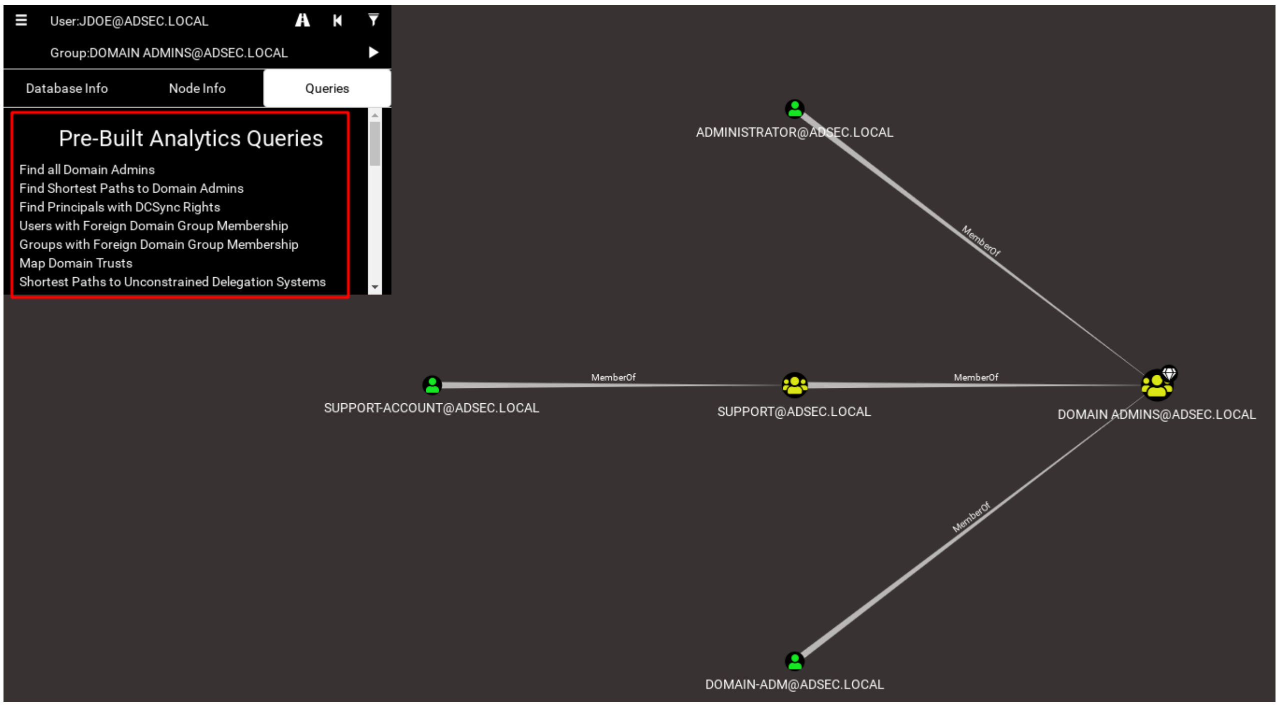
This command lets you know that the user `SUPPORT-ACCOUNT` requested a resource on `computer` from the host `SOURCE`. This information confirms that `SUPPORT-ACCOUNT` currently has an active session on `SOURCE`. This is important information since it indicates that the credentials of `support-account` should be in **lsass** memory on `SOURCE`.

These different pieces of information (not exhaustive) are collected with the SharpHound ingestor, and are saved in `json` format in different files.

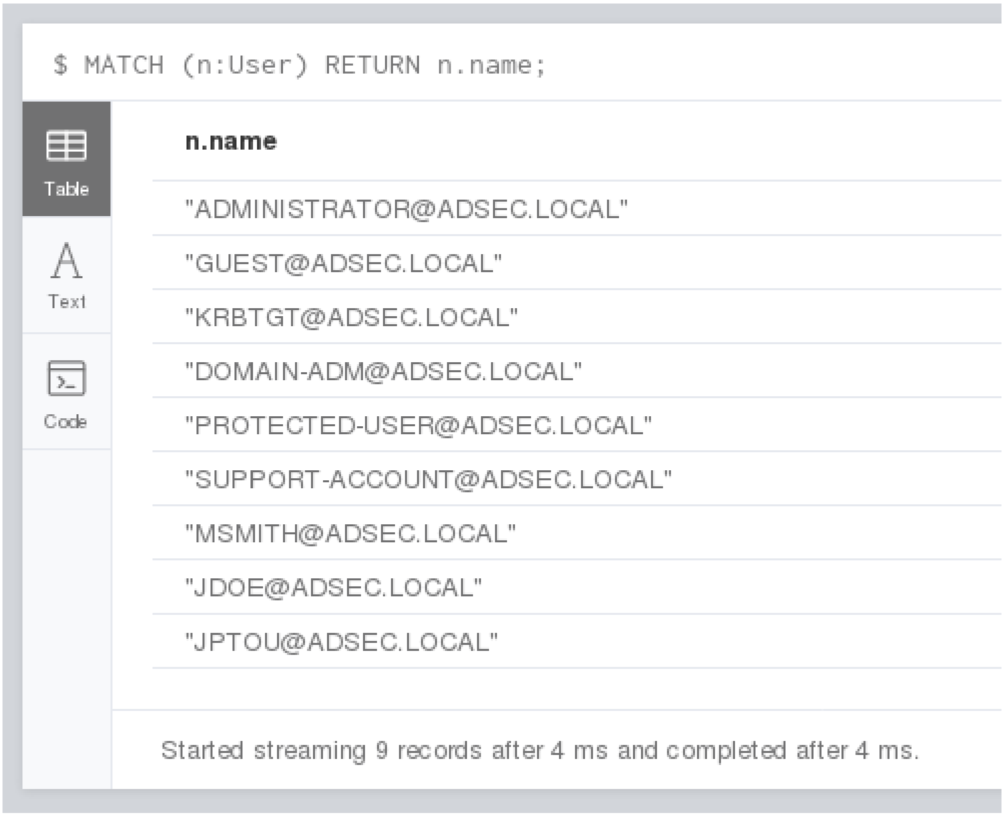
These files are then imported into BloodHound, which stores them in the Neo4j database. Once the import is complete, the BloodHound graphical interface allows you to quickly visualize this data via built-in or custom cypher queries.



20190502160106_computers.json
20190502160106_domains.json
20190502160106_gpos.json
20190502160106_groups.json
20190502160106_ous.json
20190502160106_sessions.json
20190502160106_users.json


User:JDOE@ADSEC.LOCAL
Group:DOMAIN ADMINS@ADSEC.LOCAL
Database Info
Node Info
Queries
Pre-Built Analytics Queries
Find all Domain Admins
Find Shortest Paths to Domain Admins
Find Principals with DCSync Rights
Users with Foreign Domain Group Membership
Groups with Foreign Domain Group Membership
Map Domain Trusts
Shortest Paths to Unconstrained Delegation Systems
ADMINISTRATOR@ADSEC.LOCAL
MemberOf
SUPPORT-ACCOUNT@ADSEC.LOCAL
MemberOf
SUPPORT@ADSEC.LOCAL
MemberOf
DOMAIN ADMINS@ADSEC.LOCAL
MemberOf
DOMAIN-ADM@ADSEC.LOCAL

It is also possible to send requests directly to the Neo4j interface, which can be useful for extracting data in table format.



```
$ MATCH (n:User) RETURN n.name;
```

n.name
"ADMINISTRATOR@ADSEC.LOCAL"
"GUEST@ADSEC.LOCAL"
"KRBTGT@ADSEC.LOCAL"
"DOMAIN-ADM@ADSEC.LOCAL"
"PROTECTED-USER@ADSEC.LOCAL"
"SUPPORT-ACCOUNT@ADSEC.LOCAL"
"MSMITH@ADSEC.LOCAL"
"JDOE@ADSEC.LOCAL"
"JPTOU@ADSEC.LOCAL"

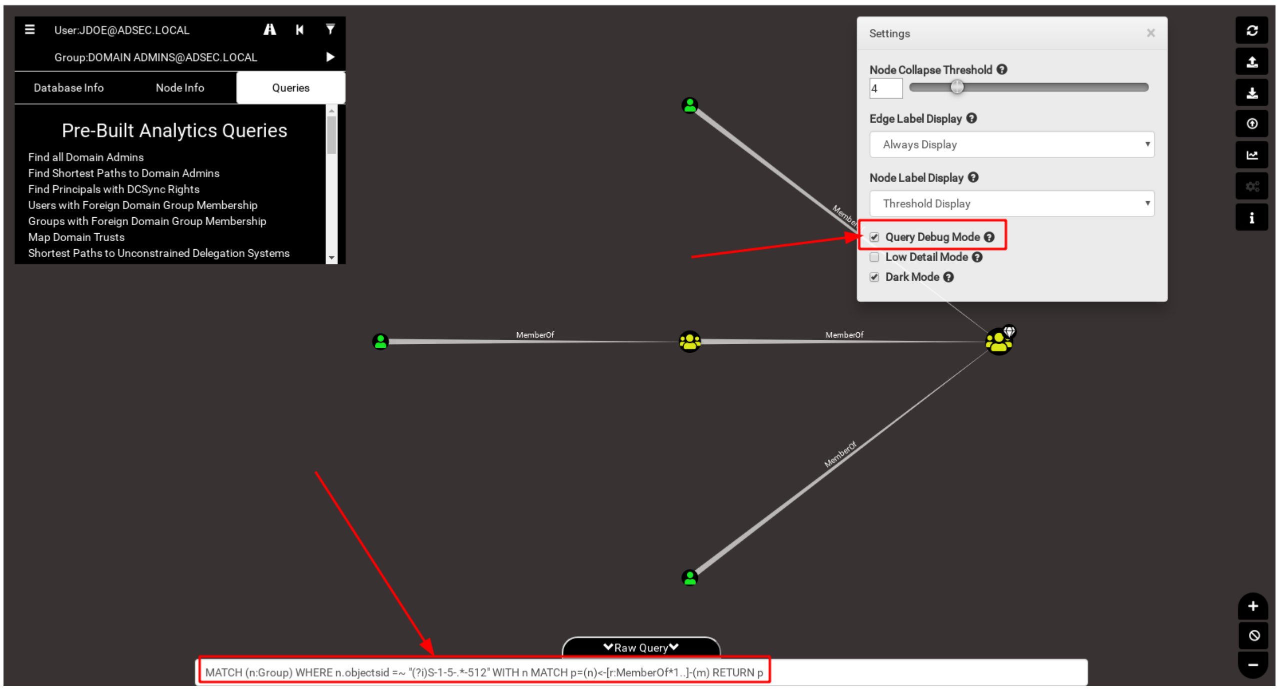
Started streaming 9 records after 4 ms and completed after 4 ms.

Requests

Beyond the built-in requests proposed by BloodHound, it is possible to write your own requests. I'm not going to write a tutorial about this here: CptJesus wrote BloodHound: Intro to Cypher which describes in detail how this works.

You can also go read the paper from SadProcessor called The Dog Whisperer's Handbook which talks about BloodHound and introduces the cypher query language.

You should also know that it is possible to activate the "debug" mode in BloodHound which displays the raw cypher queries executed when you click on a button in the graphical interface.


User:JDOE@ADSEC.LOCAL
Group:DOMAIN ADMINS@ADSEC.LOCAL
Database Info
Node Info
Queries
Pre-Built Analytics Queries
Find all Domain Admins
Find Shortest Paths to Domain Admins
Find Principals with DCSync Rights
Users with Foreign Domain Group Membership
Groups with Foreign Domain Group Membership
Map Domain Trusts
Shortest Paths to Unconstrained Delegation Systems
Settings
Node Collapse Threshold
4
Edge Label Display
Always Display
Node Label Display
Threshold Display
Query Debug Mode
Low Detail Mode
Dark Mode
MemberOf
MemberOf
MemberOf
Raw Query

```
MATCH (n:Group) WHERE n.objectsid =~ "(?i)S-1-5-.*-512" WITH n MATCH p=(n)<-[r:MemberOf*1..]-(m) RETURN p
```

This query can then be used in Neo4J if needed.

For lazy people that didn't read the posts I linked above, here's a quick overview of a cypher query.

```
MATCH p=(n:User {name:"SUPPORT-ACCOUNT@ADSEC.LOCAL"})-[r:MemberOf*1..]->(g:Group)
RETURN p
```

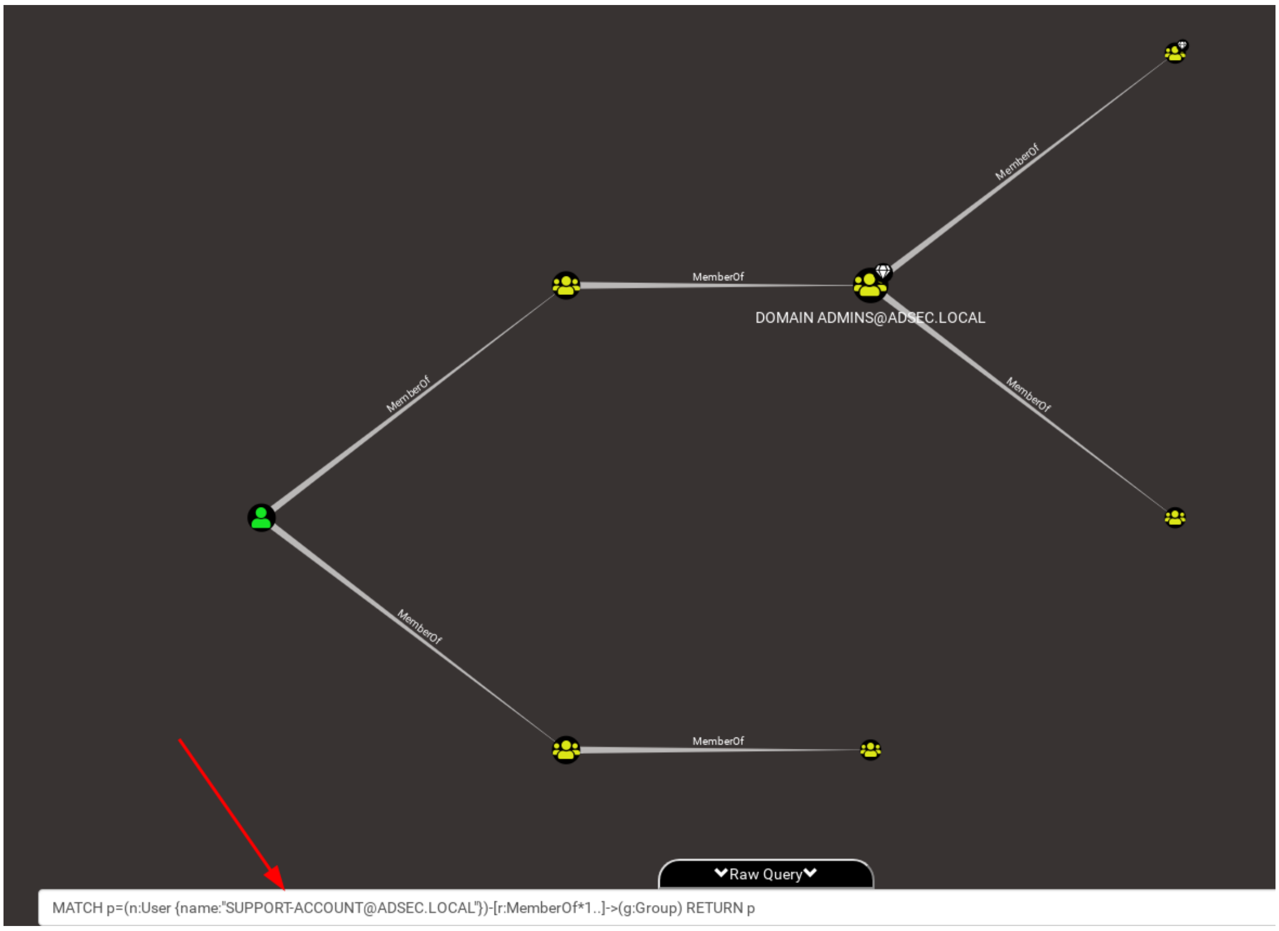
This query lists all the groups the user `support-account` is a member of. It can be split in two.

The first part `MATCH` indicates what we are looking for. What we find between parentheses indicates the nodes, and between square brackets the relationship.

We are therefore looking for a node `n` of type `User` having a relationship `r` of type `MemberOf` to a node `g` of type `Group`. The content between curly braces allows you to apply a filter, here a filter on the name `SUPPORT-ACCOUNT@ADSEC.LOCAL`.

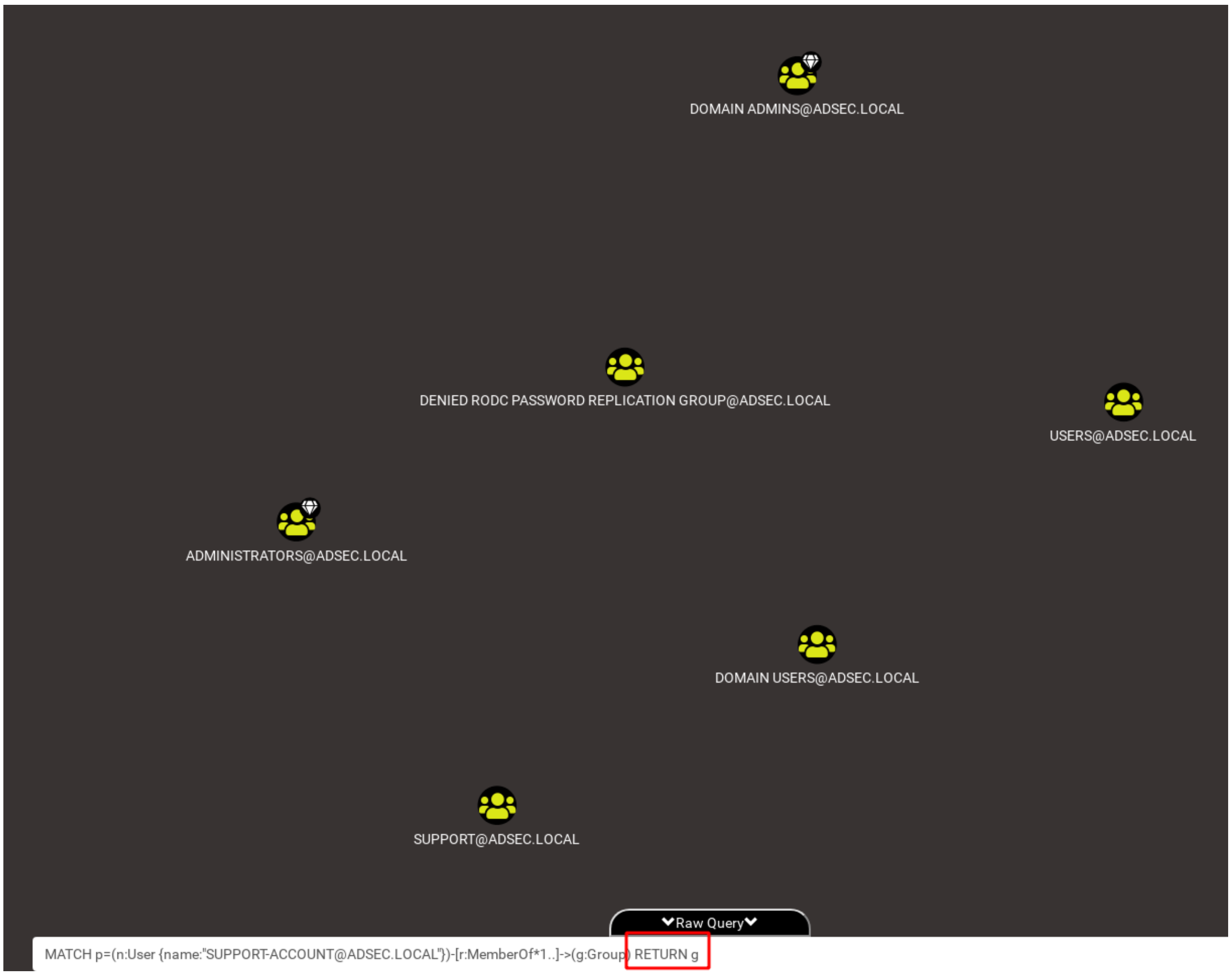
In the relationship part between square brackets, we indicate that this relation must be present one or more times (`* 1..`).

The second part `RETURN` indicates what we want to return. Here, we want to return the whole pattern assigned to the `p` variable.



```
MATCH p=(n:User {name:"SUPPORT-ACCOUNT@ADSEC.LOCAL"})-[r:MemberOf*1..]->(g:Group) RETURN p
```

We could have decided to only list the different groups without displaying the relationships or the users. We just need to return the `g` nodes.


DOMAIN ADMINS@ADSEC.LOCAL
DENIED RODC PASSWORD REPLICATION GROUP@ADSEC.LOCAL
USERS@ADSEC.LOCAL
ADMINISTRATORS@ADSEC.LOCAL
DOMAIN USERS@ADSEC.LOCAL
SUPPORT@ADSEC.LOCAL
Raw Query

```
MATCH p=(n:User {name:"SUPPORT-ACCOUNT@ADSEC.LOCAL"})-[r:MemberOf*1..]->(g:Group) RETURN g
```

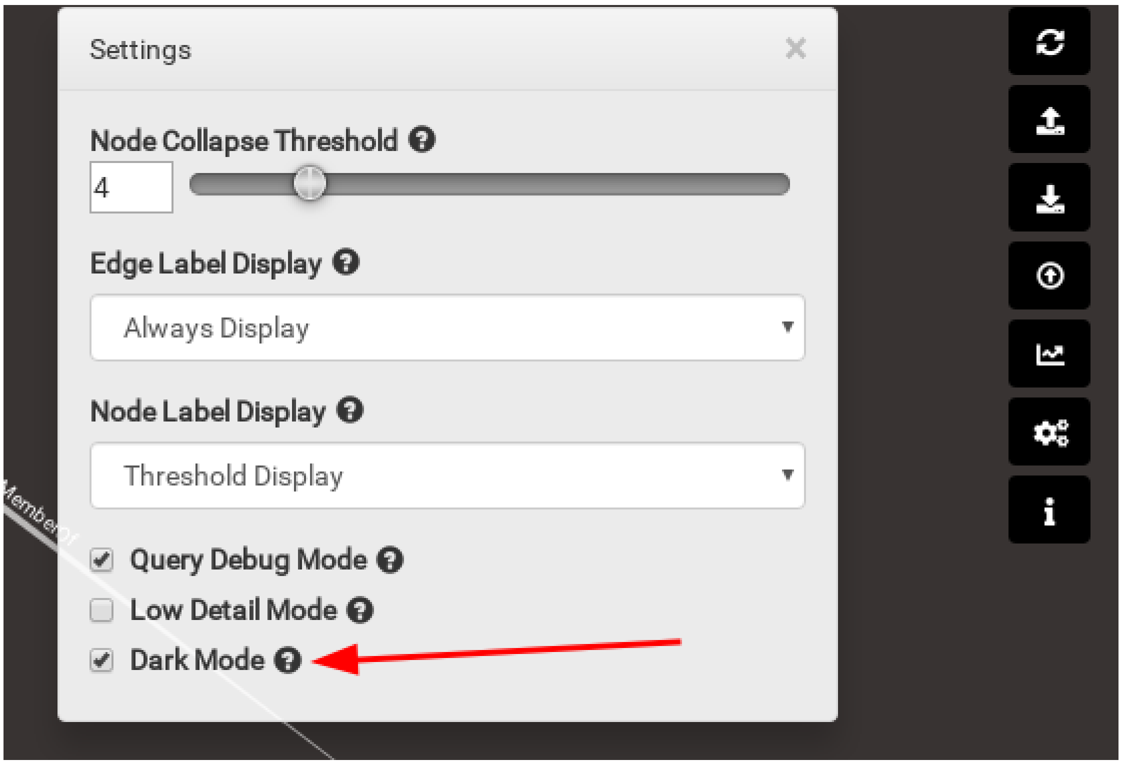
For more complex queries, the posts I linked before will help you go deeper.

Tips & tricks

Here is a list of some little tips or information that may be important or useful.

Dark mode

The first, and by far the most important, is the dark mode of the graphical interface!


Settings

Node Collapse Threshold: 4

Edge Label Display: Always Display

Node Label Display: Threshold Display

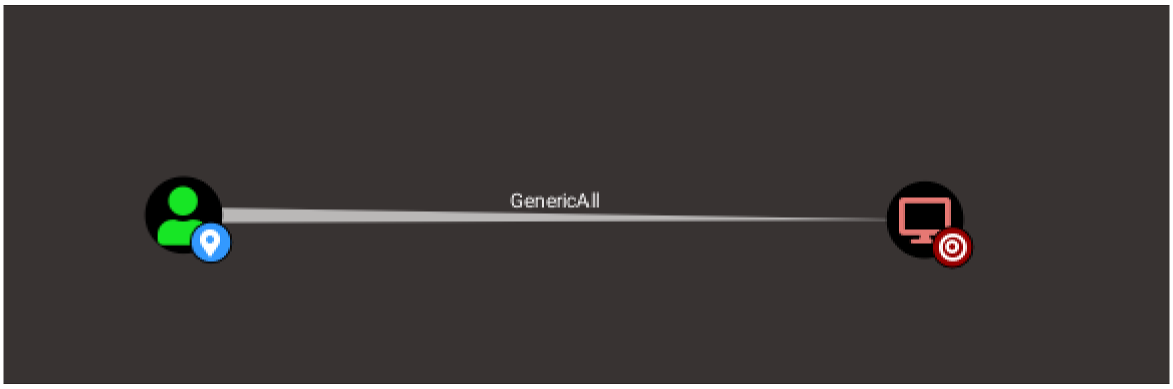
[✓] Query Debug Mode
[ ] Low Detail Mode
[✓] Dark Mode

This mode allows you to have a dark and classy interface, which is super nice. That’s it, it’s beautiful, but that’s it.

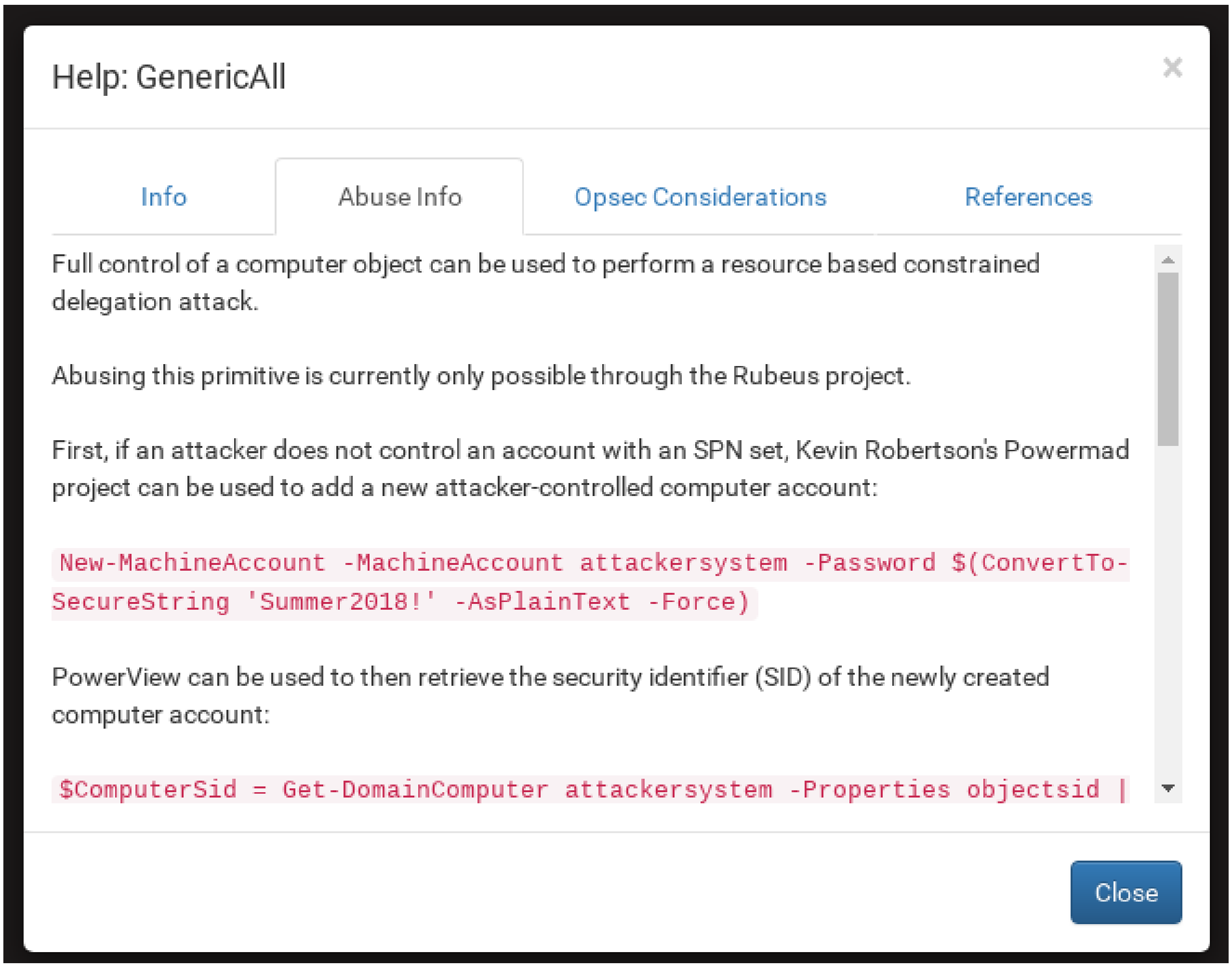
Right click on edges

When there is a relationship between two nodes, you can “right click” the relationship and a pop-up will display a lot of extremely useful information.

You do not know how to use the `GenericAll` link?



No problem, right clicking the edge will display the tooltip.



Full control of a computer object can be used to perform a resource based constrained delegation attack.

Abusing this primitive is currently only possible through the Rubeus project.

First, if an attacker does not control an account with an SPN set, Kevin Robertson's Powermad project can be used to add a new attacker-controlled computer account:

```
New-MachineAccount -MachineAccount attackersystem -Password $(ConvertTo-
SecureString 'Summer2018!' -AsPlainText -Force)
```

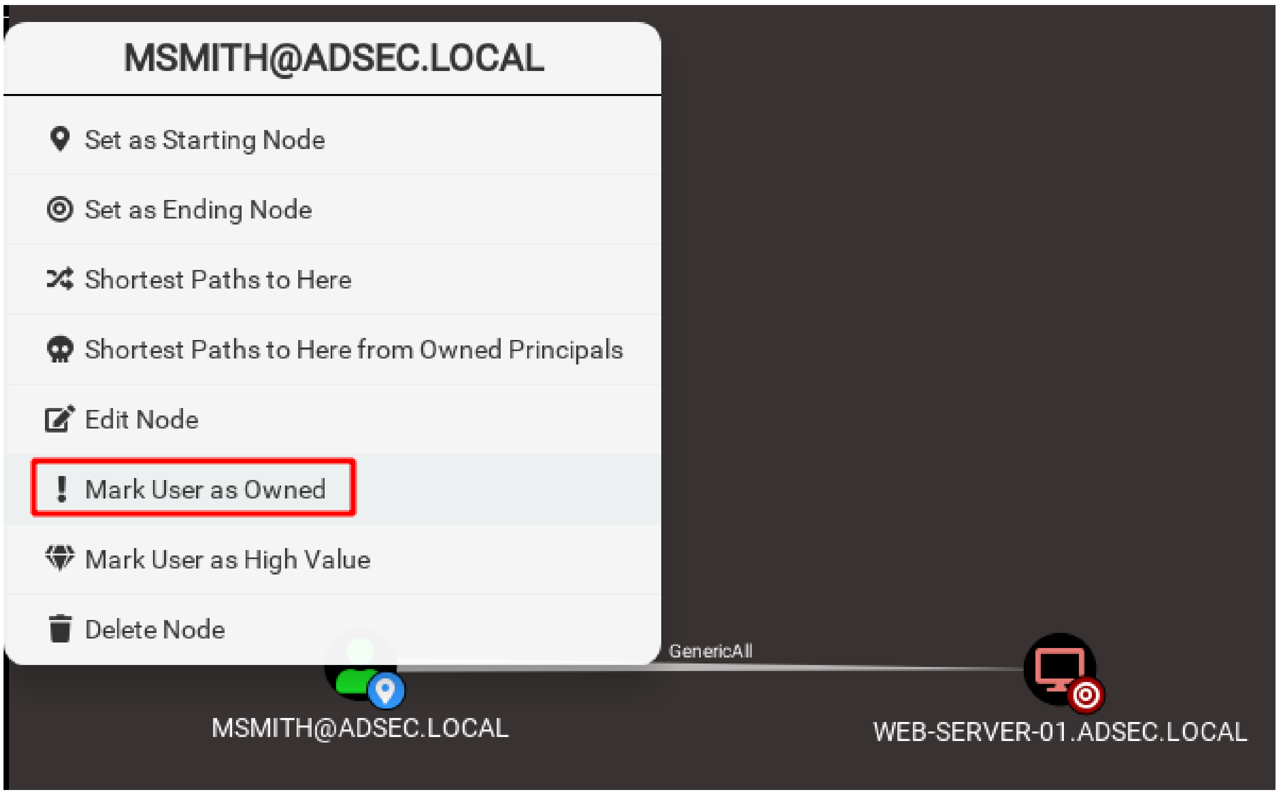
PowerView can be used to then retrieve the security identifier (SID) of the newly created computer account:

```
$ComputerSid = Get-DomainComputer attackersystem -Properties objectsid |
```

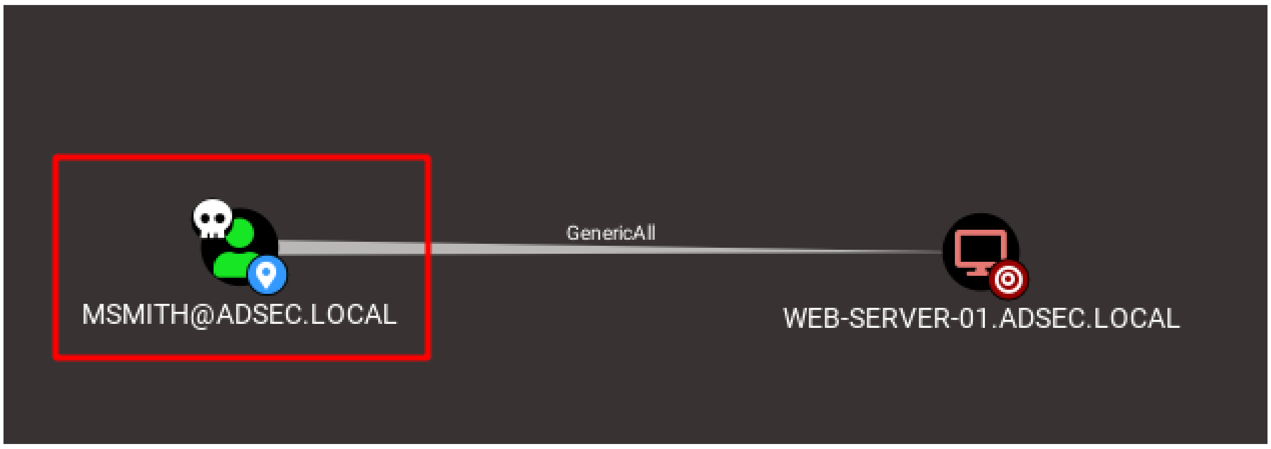
It contains all the necessary information to take control of the destination host.

Owned

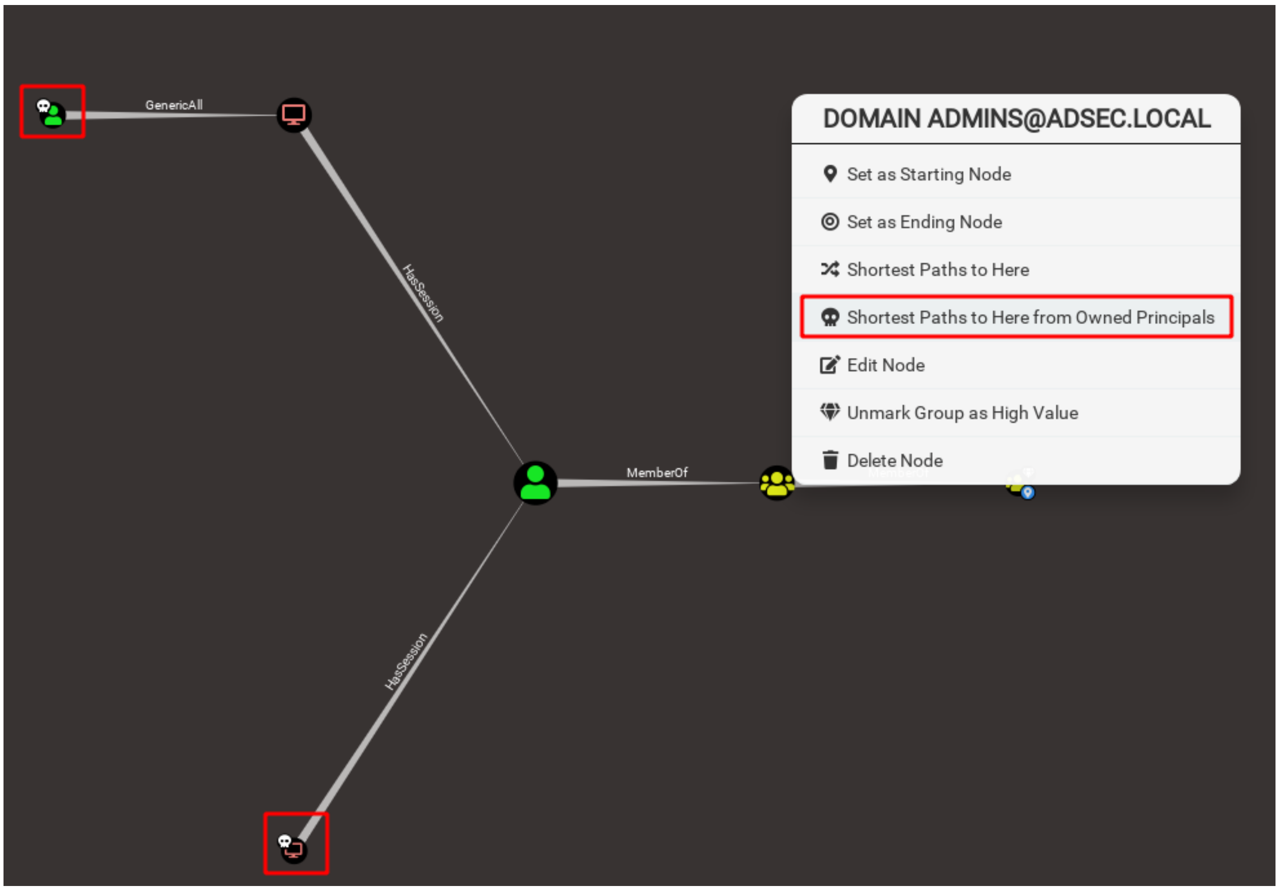
During your penetration test, you will compromise hosts, users, groups. You can mark the objects as “owned” after a right click on them to keep track of your progress.



A skull icon will mark these objects afterwards.

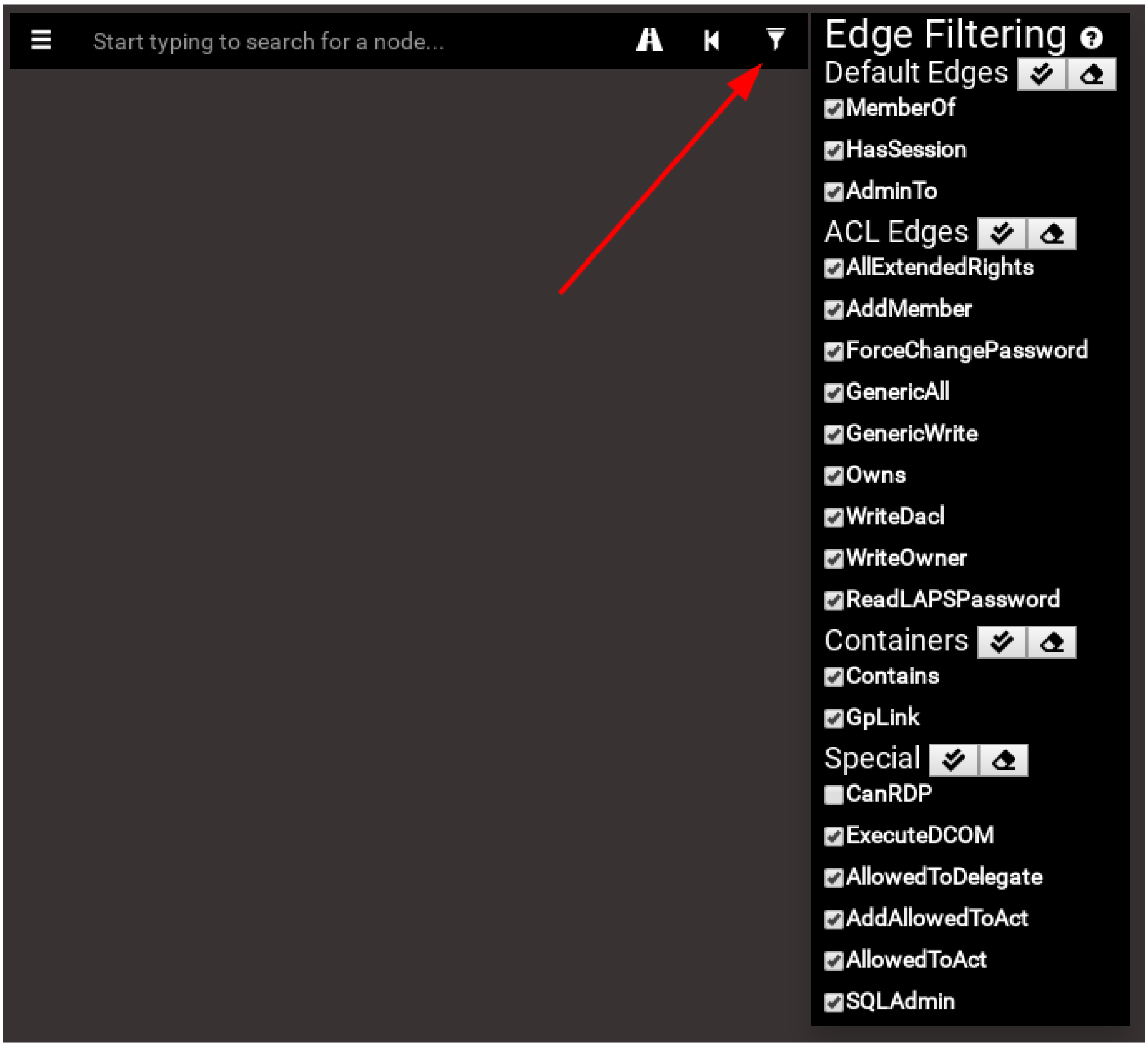


If you are trying to compromise a particular node, you can now request the shortest attack path from the nodes you have already compromised. Neat.



Filters

If you do not want to display certain paths because there are relationships that you cannot exploit, or because you do not have the time, or any other reason, you can decide to uncheck the relationships you do not want to use so that they no longer appear in your queries. To do this, simply click on the filter button to the right of the search bar, and check or uncheck the edges that you want.



Edge Filtering

Default Edges
- [✓] MemberOf
- [✓] HasSession
- [✓] AdminTo

ACL Edges
- [✓] AllExtendedRights
- [✓] AddMember
- [✓] ForceChangePassword
- [✓] GenericAll
- [✓] GenericWrite
- [✓] Owns
- [✓] WriteDacl
- [✓] WriteOwner
- [✓] ReadLAPSPassword

Containers
- [✓] Contains
- [✓] GpLink

Special
- [ ] CanRDP
- [✓] ExecuteDCOM
- [✓] AllowedToDelegate
- [✓] AddAllowedToAct
- [✓] AllowedToAct
- [✓] SQLAdmin

Shortcuts

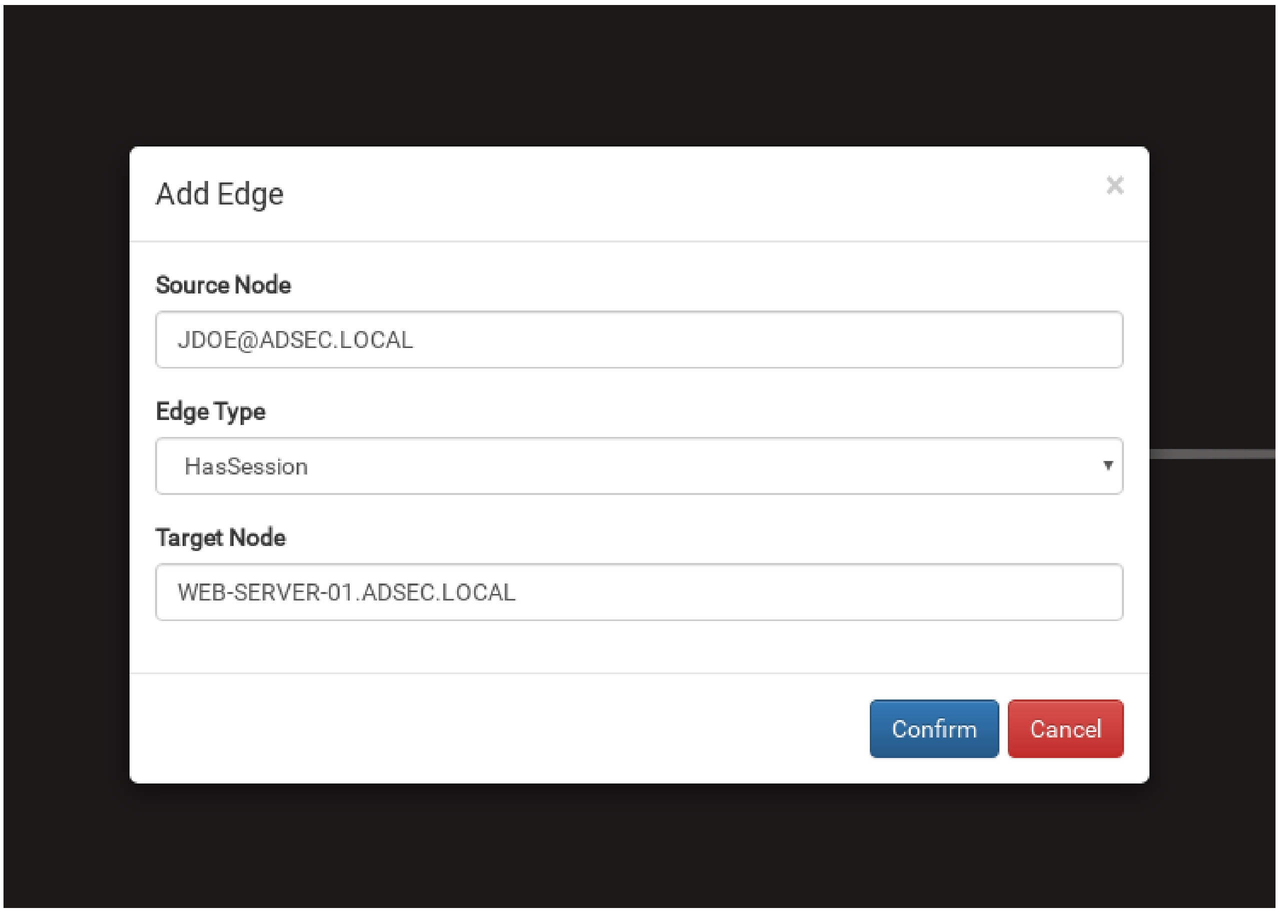
There are a few shortcuts that can be useful when using BloodHound. If you know others, don't hesitate to share them.

- **CTRL**: Allows you to display or hide node and relationship labels.
- **CTRL + SHIFT + I**: Displays the developer tools. If you have a bug in the interface, the javascript console will often allow you to understand the root cause of the issue.
- **CTRL + R**: Reloads the display
- **SPACE**: Pops up a table that lists all currently displayed nodes. That's very convenient if you want to export some query's results into Excel or something.

Editing

During data collection, your computer was part of the collected hosts and you do not want to see it? You have logged on to a machine, but the "HasSession" link of the Domain Administrator is no longer effective since this machine has restarted? On the contrary, another user is connected on this machine?

Once the data has been imported into BloodHound, it is not frozen. You can modify it as you wish, either by right-clicking on an object or relationship to delete it, or by right-clicking in the background to add a node or relationship.


Add Edge

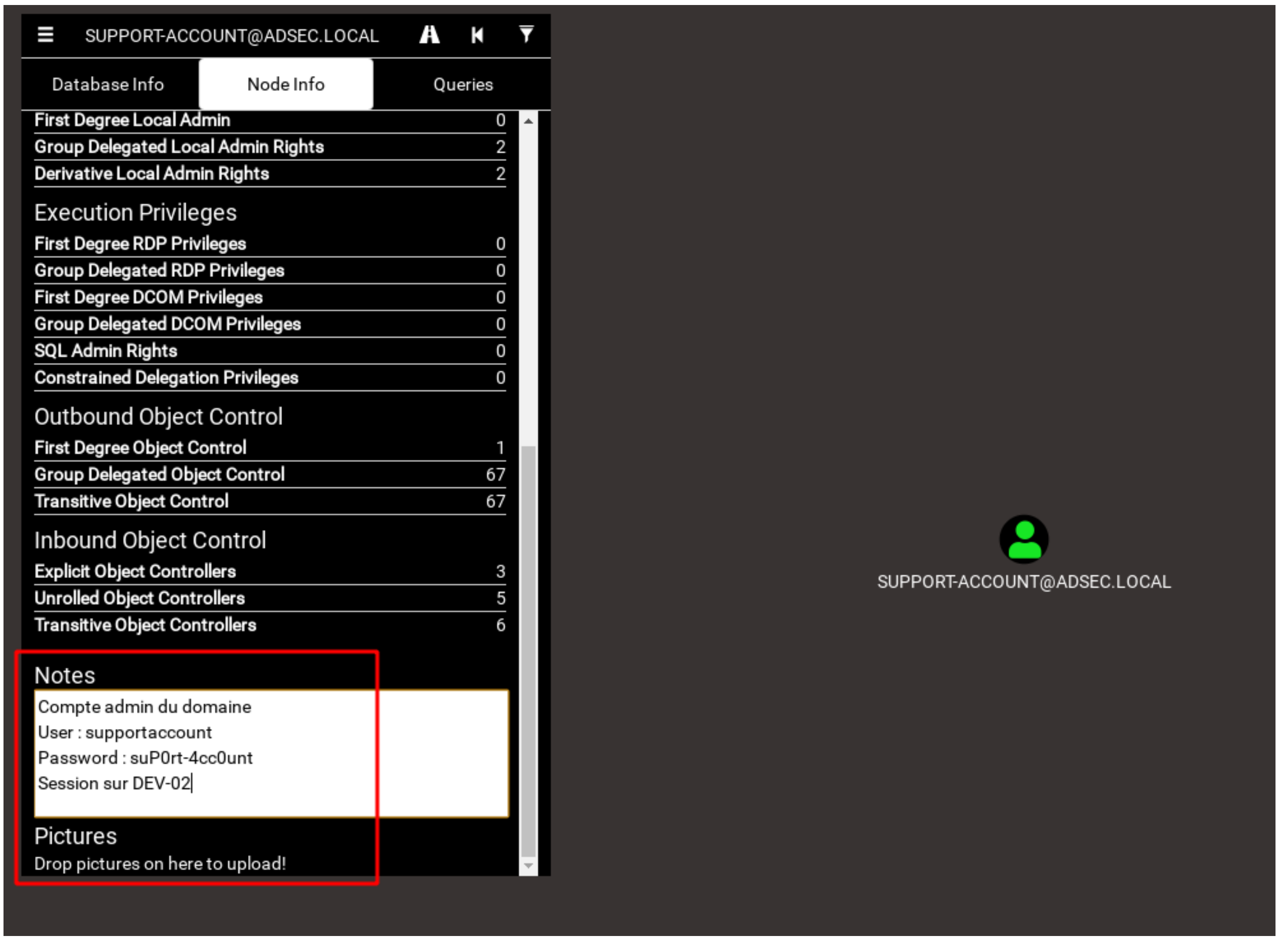
Source Node: JDOE@ADSEC.LOCAL

Edge Type: HasSession

Target Node: WEB-SERVER-01.ADSEC.LOCAL

Notes & Screenshots

You can also save information on objects in Active Directory by left clicking on them and going to the “Notes” section in the BloodHound interface.


SUPPORT-ACCOUNT@ADSEC.LOCAL

Database Info | Node Info | Queries

First Degree Local Admin	0
Group Delegated Local Admin Rights	2
Derivative Local Admin Rights	2

Execution Privileges

First Degree RDP Privileges	0
Group Delegated RDP Privileges	0
First Degree DCOM Privileges	0
Group Delegated DCOM Privileges	0
SQL Admin Rights	0
Constrained Delegation Privileges	0

Outbound Object Control

First Degree Object Control	1
Group Delegated Object Control	67
Transitive Object Control	67

Inbound Object Control

Explicit Object Controllers	3
Unrolled Object Controllers	5
Transitive Object Controllers	6

Notes

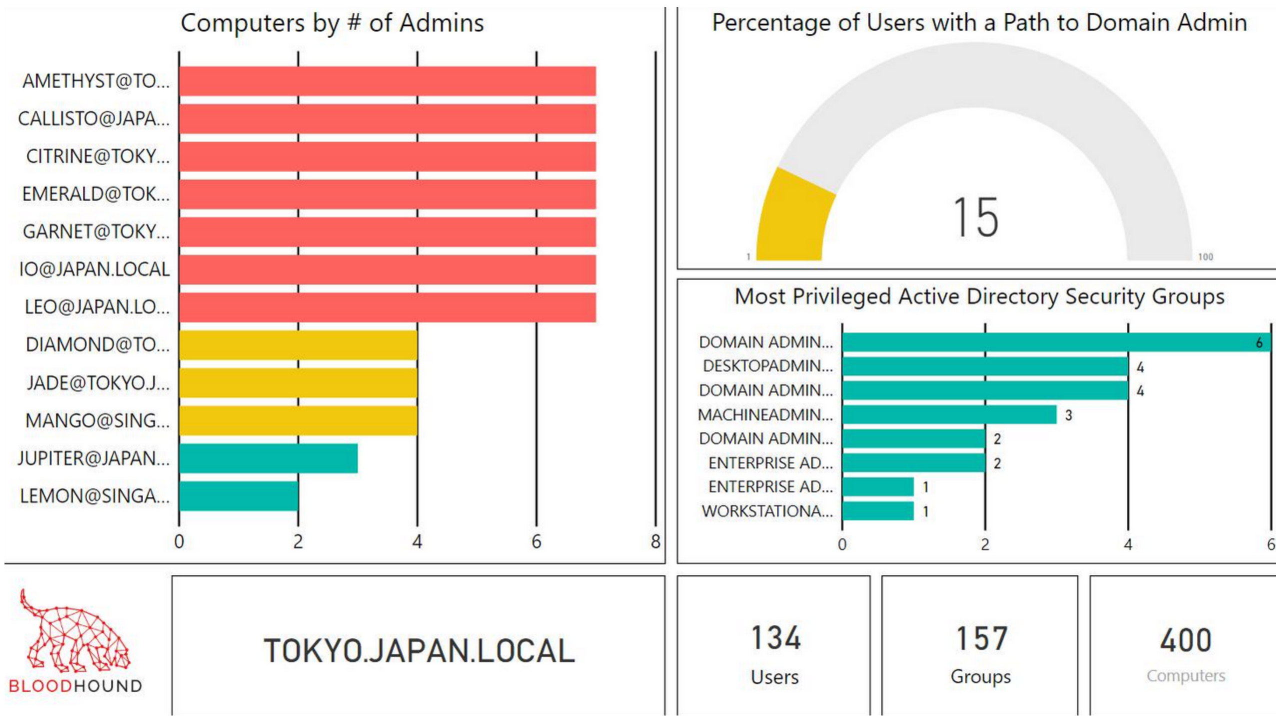
Compte admin du domaine
User : supportaccount
Password : suP0rt-4cc0unt
Session sur DEV-02

Pictures
Drop pictures on here to upload!

BloodHound Analytics

Finally, the same team that developed BloodHound also released some tools to create statistics out of a BloodHound extraction. They are available on their Github, including the `bloodhoundanalytics.pbix` template that uses Microsoft's free software `PowerBI`.

Here is an example taken from Wald0 tweet, which shows a PowerBI dashboard using the template provided on the Github.



The result is superb and very explicit for management.

Going further

A talk was made at BSides by the three authors in 2016 to introduce BloodHound.

The authors of the tool are also present on the dedicated Slack server. There is a large community present on this Slack ready to welcome you and answer your questions. There is even a #french channel!

BloodHound is not the only tool of its kind. We have a Franco-French tool called AD Control Paths developed by ANSSI. A talk was made during the SSTIC on this subject and the associated PDF is very interesting. French readers, you should read it.

Conclusion

As long as the defense teams think in list and the attackers think in graph, attackers will be one step ahead. This is why **BloodHound** can also be used as a defense tool. Collect data on a regular basis and prepare some queries to check your Active Directory health/security and monitor its evolution.

To conclude, BloodHound offers a new approach to visualize data in Active Directory environments. Graph visualization makes it possible to understand the complex relationships and intricacies of objects and relationships in the information system in order to identify attack paths and/or abnormal behavior.

Author : Pixis
Blog author, follow me on twitter or discord

Similar posts