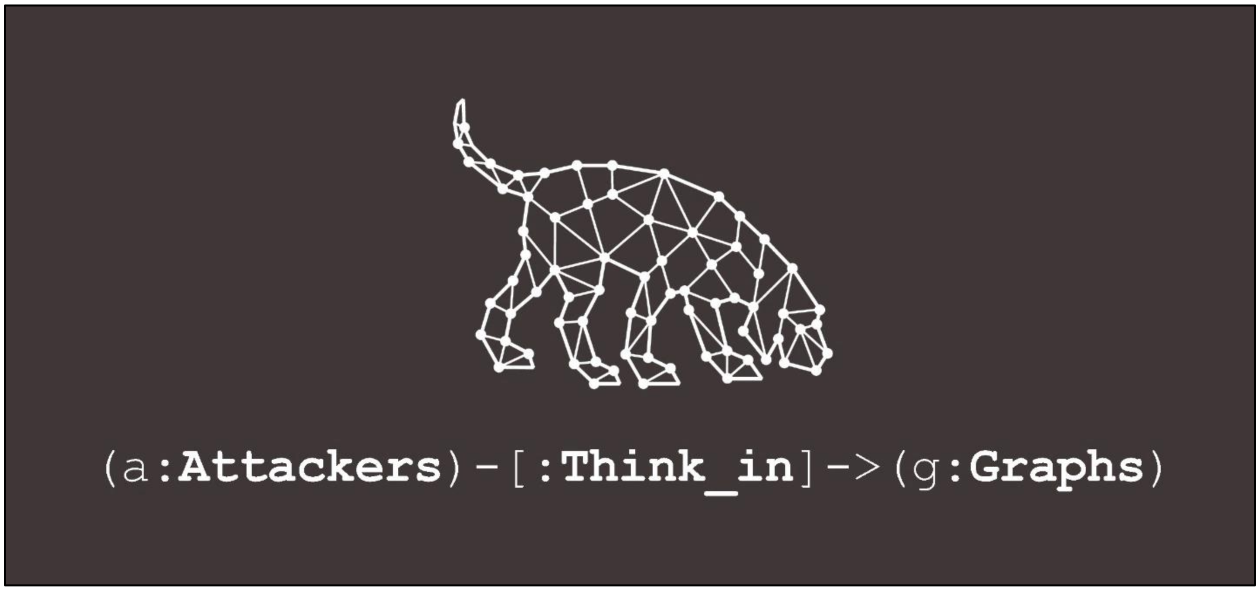
THE DOG WHISPERER'S HANDBOOK

A Hacker's Guide to the BloodHound Galaxy – *@SadProcessor*

Table of Contents

A - BloodHound Concept & Tool Evolution



A.1 – Attackers think in Graphs…

Nobody knows how it really started, nor how much liquid was needed that late evening, but in the tiny galaxy of Active Directory security, most would agree that the tool we are going to talk about has been a real game changer when it comes to attacking (and thus defending) Active directory. No need to introduce the creators I guess, but quick credit where credit is due.

If you are into windows security and are not following @harmj0y, @_Wald0 & @CptJesus on twitter:

1- Stop this for now
2- Grab an internet connected device of your choice
3- Click on **Follow**

... And while you're at it, invite yourself to the BloodHound Slack, it's a really cool place for cyber-cybering your cyberz. @_Wald0 & @CptJesus will welcome you in person like they did for all 2000+ members of the so called BloodHoundGang.

[cherry on the cake, they even have a @PrimaryTyler with its matching #PrimaryTylerHate channel. Honestly, one couldn't ask for more... Really. Check it out.]

And if you ever meet any of these fine gentlemen at a security con, tap them on the shoulder and pay them a couple of beers. These guys are **as thirsty as humble** and have a lot of cyberingz for share...

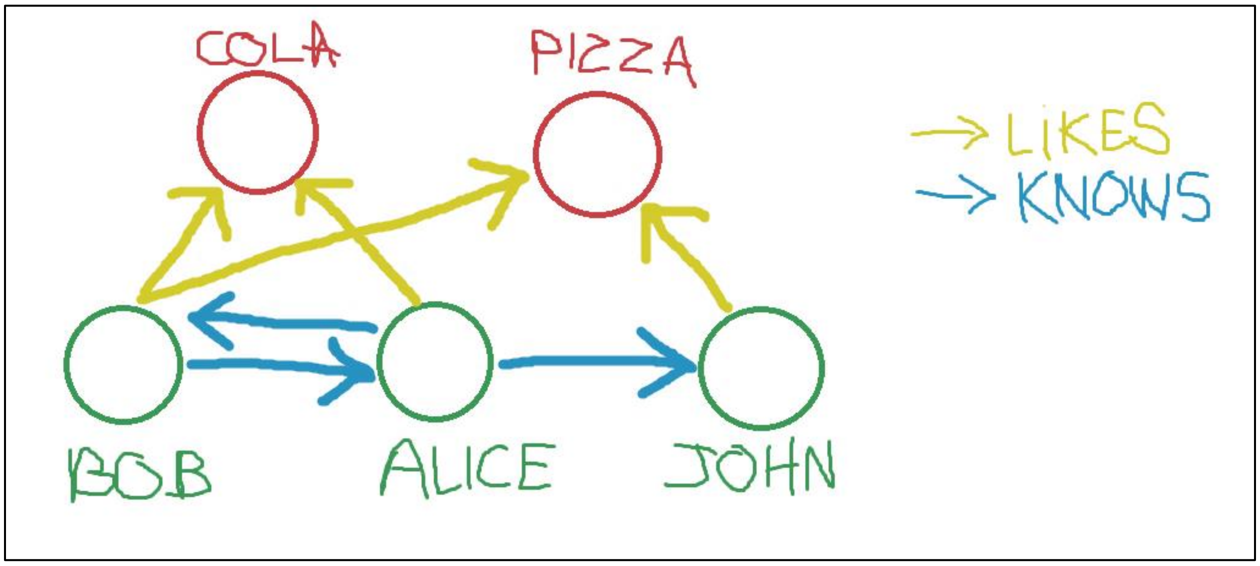
Now that this is done, and to be fully accurate, it has to be said that there was a French tool called "**Chemins de Controle de l'Active Directory**" that has inspired BloodHound. It was made by Jean-Baptiste Galet & Geraud de Drouas from the ANSSI (kind of French version of the NSA).
Check it out. "C'est très très Cyber mon ami...". More info can be found here.

Now to get a better idea of the bigger concept around BloodHound and Graph Theory in general, a **MANDATORY READ** is a foundational post by John Lambert [aka @JohnLaTwC. #FF]. The dude is Head of Microsoft Threat Intel, looks like your Math teacher from the late 80s, goes tripping high up in the mountains AND far out in Cyber Space... The kind of guy who knows stuff... More Cyber-Cool than this, you just die.

The full title of the post goes like **"Attackers think in Graphs, Defenders think in lists. As long as this is true, Attackers win"**. This famous cyber-quote sums up the whole blog-post & concept in an awesome way, but there is much more to it. Like written in bold a few lines above and here->, this is a **MANDATORY READ**. Enjoy it. See you then.

A.2 – Graph Theory in a Nutshell

If I had to explain Graph Theory to a 2yr old, he most probably wouldn't understand much, but I would just open MS Paint and draw something like this:



From this simple graph, we can retrieve a lot of info. I hope you get it… because this is as complicated as it gets.

So let's imagine a hypothetical Orwellian future, where humanity would be in search of an "improved individual advertising experience", and where people would submit there "likes" to the system on a daily basis, and share it with the individuals they "know". Let's just call this system "FaceLook" or something…
This fictional FaceLook would just be a huge graph database. With **Nodes**, **Edges** and **Paths**.
Nodes would be John, Alice, pizza, cola…
Edges would be "Knows" or "Likes".
Paths would be a series of nodes connected by Edges.
And we could ask the system a lot of great stuff. For example, people who like cola, people who know people who like pizza, people who like cola who know people who know people who like pizza…
And so on. It's as easy as that… And at scale, very powerful.

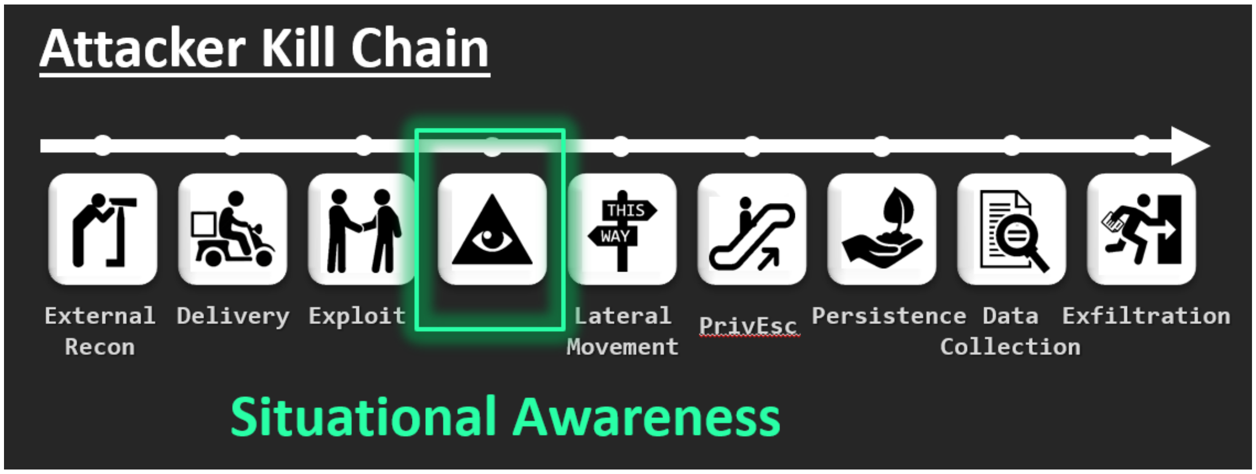
Some say with this tool, one could gain full control over the pizza delivery market by means of targeted advertising… Some even say one could influence foreign elections and threaten modern democracies…

Troll farms, fake news, collusion… Enough of this crazy dystopian non-sense…

Let's get back to reality…

A.3 – Awe! Cute Little Puppy...

Ok, so now that we did the basic introduction, let's dive into BloodHound itself.



Initially designed with offense in mind, Bloodhound is an **Active Directory Object Relationship Graphing Tool**. It serves for the **Situational Awareness** (Internal Recon) phase in the Attacker Kill Chain model. BloodHound does not have any offensive capacity itself, but it is a fantastic tool for **mapping the targeted environment** and **visualizing possible attack paths** to get the job done.

If you have played with it already, you probably have felt a bit overwhelmed by the number of features and quantity of available information it holds. The cute little puppy quickly became an awesome monster...

Many say **@CptJesus is a machine**... Fake News? Maybe. Maybe not... I personally think he is... But like The Dude would say, "that's only like my opinion man..." so I'll let you have your own idea on the matter.

Anyways... I believe the easiest way to get to the full extent of it, is to follow the evolution of the tool in a chronological order. @harmj0y, @_Wald0 and @CptJesus have put in a great amount of effort into documenting all of it, so the bare minimum we can do is enjoy it as it should...

Automated Derivative Admin Search	by @_Wald0
Introducing BloodHound	by @_Wald0
Intro to Cypher	by @CptJesus
The ACL Attack Path Update (v1.3)	by @_Wald0
Evolution of the BloodHound Ingestor	by @CptJesus
The Object Properties Update (v1.4)	by @CptJesus
SharpHound: Technical Details	by @CptJesus
SharpHound: Target Selection and API Usage	by @CptJesus
The Container Update (v1.5)	by @CptJesus
A Red Teamer's Guide to GPOs & OUs	by @_Wald0
BloodHound 2.0 (v2.0)	by @CptJesus

Many of the items in this list will be referenced thru out this guide, and you are not expected to read it now, but you can fulfill all your future needs with a simple click on these links. Feel free to do it at any time. This guide is just a gathering of stuff I collected here and there, the real juicy stuff you will have to go fetch... [<--Good Bloodhound Joke :)]

A.4 – Nodes, Edges & Paths

Neo4j, and more generally Graph Theory, uses a specific terminology. This vocabulary will be used thru out the whole guide. Let's take a look at how this applies to Bloodhound and Active Directory objects…

A.4.1 – Nodes

In the BloodHound world, a **Node** is an AD object.

At first these were **Users**, **Computers**, **Groups** and **Domains**.

Since the "Container" release, **OUs** and **GPOs** have been added (with matching Edges) to extend the tool's capabilities and bring new attack paths to the game.

A.4.2 – Edges

An **Edge** is simply the relationship between two AD objects. Initially, possible Edges were **MemberOf**, **HasSession**, **AdminTo** and **TrustedBy**. More Edges have been added thru out the evolution of the tool. The current version of BloodHound counts 18 possible Edges.

The following table details all of them:

Edge Type	Source Object	Target Object
DEFAULT		
MemberOf	User/Group/Computer	Group
HasSession	Computer	User
AdminTo	User/Group	Computer
TrustedBy	Domain	Domain
ACL		
AllExtendedRights	Group/User	Any
AddMember	Group/User	Group
ForceChangePassword	Group/User	User
GenericAll	Group/User	Any

GenericWrite	Group/User	Any
Owns	Group/User	Any
WriteDACL	Group/User	Any
WriteOwner	Group/User	Any
ReadLAPSPassword	Group/User	Computer
CONTAINER		
Contains	Domain/OU	OU/User/Computer
GpLink	GPO	Domain/OU
SPECIAL		
CanRDP	Group/User	Computer
ExecuteDCOM	Group/User	Computer
AllowedToDelegate	Group/User	Computer

A.4.3 – Paths

Finally, a **Path** is a series of Nodes connected by Edges. In BloodHound, it becomes an Attack Path. Each Edge can be abused to reach the next node. An Attacker can navigate the AD tree just like a subway map to get to his target. A defender should be aware of those existing attack paths in order to properly monitor and defend them.

Note: It is important to note that Edges are directional, and that therefore the attack path is directed. Edges can only be abused in a specific direction.

A.5 – Walking the Dog...

In the following section, we will detail each Edge and its matching abuse. All this info can be found in the UI with a right-click on the selected Edge.

[Disclaimer: It should be noted that this is extracted directly from the BloodHound documentation and that the author does not specifically endorse the use of any of those tools, or the possible consequences of running those commands in your environment.]

A.5.1 – Defaults

MemberOf
Info: Groups in active directory grant their members any privileges the group itself has. If a group has rights to another principal, users/computers in the group, as well as other groups inside the group inherit those permissions.
Abuse Info: No abuse is necessary. This edge simply indicates that a principal belongs to a security group.
Opsec Considerations: No opsec considerations apply to this edge
References: https://adsecurity.org/?tag=ad-delegation https://www.itprotoday.com/management-mobility/view-or-remove-active-directory-delegated-permissions

HasSession
Info: When a user authenticates to a computer, they often leave credentials exposed on the system, which can be retrieved through LSASS injection, token manipulation/theft, or injecting into a user's process.

Any user that is an administrator to the system has the capability to retrieve the credential material from memory if it still exists.

Note: A session does not guarantee credential material is present, only possible.

Abuse Info:

Password Theft

When a user has a session on the computer, you may be able to obtain credentials for the user via credential dumping or token impersonation. You must be able to move laterally to the computer, have administrative access on the computer, and the user must have a non-network logon session on the computer.

Once you have established a Cobalt Strike Beacon, Empire agent, or other implant on the target, you can use mimikatz to dump credentials of the user that has a session on the computer. While running in a high integrity process with SeDebugPrivilege, execute one or more of mimikatz's credential gathering techniques (e.g.: sekurlsa::wdigest, sekurlsa::logonpasswords, etc.), then parse or investigate the output to find clear-text credentials for other users logged onto the system.

You may also gather credentials when a user types them or copies them to their clipboard! Several keylogging capabilities exist, several agents and toolsets have them built-in. For instance, you may use meterpreter's "keyscan_start" command to start keylogging a user, then "keyscan_dump" to return the captured keystrokes. Or, you may use PowerSploit's Invoke-ClipboardMonitor to periodically gather the contents of the user's clipboard.

Token Impersonation

You may run into a situation where a user is logged onto the system, but you can't gather that user's credential. This may be caused by a host-based security product, lsass protection, etc. In those circumstances, you may abuse Windows' token model in several ways. First, you may inject your agent into that user's process, which will give you a process token as that user, which you can then use to authenticate to other systems on the network. Or, you may steal a process token from a remote process and start a thread in your agent's process with that user's token. For more information about token abuses, see the References tab.

User sessions can be short lived and only represent the sessions that were present at the time of collection. A user may have ended their session by the time you move to the computer to target them. However, users tend to use the same machines, such as the workstations or servers they are assigned to use for their job duties, so it can be valuable to check multiple times if a user session has started.

Opsec Considerations:

An EDR product may detect your attempt to inject into lsass and alert a SOC analyst. There are many more opsec considerations to keep in mind when stealing credentials or tokens. For more information, see the References tab.

References:

Gathering Credentials

http://blog.gentilkiwi.com/mimikatz

https://github.com/gentilkiwi/mimikatz

https://adsecurity.org/?page_id=1821

https://attack.mitre.org/wiki/Credential_Access

Token Impersonation

https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-security-implications-of-windows-access-tokens-2008-04-14.pdf

https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-TokenManipulation.ps1

https://attack.mitre.org/wiki/Technique/T1134

AdminTo

Info:

By default, administrators have several ways to perform remote code execution on Windows systems, including via RDP, WMI, WinRM, the Service Control Manager, and remote DCOM execution.

Further, administrators have several options for impersonating other users logged onto the system, including plaintext password extraction, token impersonation, and injecting into processes running as another user.

Finally, administrators can often disable host-based security controls that would otherwise prevent the aforementioned techniques.

Abuse Info:

Lateral movement

There are several ways to pivot to a Windows system. If using Cobalt Strike's beacon, check the help info for the commands "psexec", "psexec_psh", "wmi", and "winrm". With Empire, consider the modules for Invoke-PsExec, Invoke-DCOM, and Invoke-SMBExec. With Metasploit, consider the modules "exploit/windows/smb/psexec", "exploit/windows/winrm/winrm_script_exec", and "exploit/windows/local/ps_wmi_exec". Additionally, there are several manual methods for remotely executing code on the machine, including via RDP, with the service control binary and interaction with the remote machine's service control manager, and remotely instantiating DCOM objects. For more information about these lateral movement techniques, see the References tab.

Gathering credentials

The most well-known tool for gathering credentials from a Windows system is mimikatz. mimikatz is built into several agents and toolsets, including Cobalt Strike's beacon, Empire, and Meterpreter. While running in a high integrity process with SeDebugPrivilege, execute one or more of mimikatz's credential gathering techniques (e.g.: sekurlsa::wdigest, sekurlsa::logonpasswords, etc.), then parse or investigate the output to find clear-text credentials for other users logged onto the system.

You may also gather credentials when a user types them or copies them to their clipboard! Several keylogging capabilities exist, several agents and toolsets have them built-in. For instance, you may use meterpreter's "keyscan_start" command to start keylogging a user, then "keyscan_dump" to return the captured keystrokes. Or, you may use PowerSploit's Invoke-ClipboardMonitor to periodically gather the contents of the user's clipboard.

Token Impersonation

You may run into a situation where a user is logged onto the system, but you can't gather that user's credential. This may be caused by a host-based security product, lsass protection, etc. In those circumstances, you may abuse Windows' token model in several ways. First, you may inject your agent into that user's process, which will give you a process token as that user, which you can then use to authenticate to other systems on the network. Or, you may steal a process token from a remote process and start a thread in your agent's process with that user's token. For more information about token abuses, see the References tab.

Disabling host-based security controls

Several host-based controls may affect your ability to execute certain techniques, such as credential theft, process injection, command line execution, and writing files to disk. Administrators can often disable these host-based controls in various ways, such as stopping or otherwise disabling a service, unloading a driver, or making registry key changes. For more information, see the References tab.

Opsec Considerations:

There are several forensic artifacts generated by the techniques described above. For instance, lateral movement via PsExec will generate 4697 events on the target system. If the target organization is collecting and analyzing those events, they may very easily detect lateral movement via PsExec.

Additionally, an EDR product may detect your attempt to inject into lsass and alert a SOC analyst. There are many more opsec considerations to keep in mind when abusing administrator privileges. For more information, see the References tab.

References:

Lateral movement

https://attack.mitre.org/wiki/Lateral_Movement

Gathering Credentials

http://blog.gentilkiwi.com/mimikatz

https://github.com/gentilkiwi/mimikatz

https://adsecurity.org/?page_id=1821 https://attack.mitre.org/wiki/Credential_Access **Token Impersonation** https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-security-implications-of-windows-access-tokens-2008-04-14.pdf https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Invoke-TokenManipulation.ps1 https://attack.mitre.org/wiki/Technique/T1134 **Disabling host-based security controls** https://blog.netspi.com/10-evil-user-tricks-for-bypassing-anti-virus/ https://www.blackhillsinfosec.com/bypass-anti-virus-run-mimikatz/ **Opsec Considerations** https://blog.cobaltstrike.com/2017/06/23/opsec-considerations-for-beacon-commands/

A.5.2 – ACLs

More info here

AllExtendedRights
Info: Extended rights are special rights granted on objects which allow reading of privileged attributes, as well as performing special actions.
Abuse Info: The AllExtendedRights privilege grants both the DS-Replication-Get-Changes and DS-Replication-Get-Changes-All privileges, which combined allow a principal to replicate objects from the domain. This can be abused using the lsadump::dcsync command in mimikatz.
Opsec Considerations:

When using the PowerView functions, keep in mind that PowerShell v5 introduced several security mechanisms that make it much easier for defenders to see what's going on with PowerShell in their network, such as script block logging and AMSI. You can bypass those security mechanisms by downgrading to PowerShell v2, which all PowerView functions support.

References:

https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1

https://www.youtube.com/watch?v=z8thoG7gPd0

AddMember

Info:

The user has the ability to add arbitrary principals, including itself, to the targeted group. Because of security group delegation, the members of a security group have the same privileges as that group.

By adding itself to the group, an attacker will gain the same privileges that the target Group already has.

Abuse Info:

There are at least two ways to execute this attack. The first and most obvious is by using the built-in net.exe binary in Windows (e.g.: net group "Domain Admins" dfm.a /add /domain). See the opsec considerations tab for why this may be a bad idea. The second, and highly recommended method, is by using the Add-DomainGroupMember function in PowerView. This function is superior to using the net.exe binary in several ways. For instance, you can supply alternate credentials, instead of needing to run a process as or logon as the user with the AddMember privilege. Additionally, you have much safer execution options than you do with spawning net.exe (see the opsec tab).

To abuse this privilege with PowerView's Add-DomainGroupMember, first import PowerView into your agent session or into a PowerShell instance at the console. You may need to authenticate to the Domain Controller if you are not running a process as that user. To do this in conjunction with Add-DomainGroupMember, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force

$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then, use Add-DomainGroupMember, optionally specifying $Cred if you are not already running a process as that user:

```
Add-DomainGroupMember -Identity 'Domain Admins' -Members 'harmj0y' -Credential
$Cred
```

Finally, verify that the user was successfully added to the group with PowerView's Get-DomainGroupMember:

```
Get-DomainGroupMember -Identity 'Domain Admins'
```

Opsec Considerations:

Executing this abuse with the net binary will require command line execution. If your target organization has command line logging enabled, this is a detection opportunity for their analysts.

Regardless of what execution procedure you use, this action will generate a 4728 event on the domain controller that handled the request. This event may be centrally collected and analyzed by security analysts, especially for groups that are obviously very high privilege groups (i.e.: Domain Admins). Also be mindful that Powershell 5 introduced several key security features such as script block logging and AMSI that provide security analysts another detection opportunity.

You may be able to completely evade those features by downgrading to PowerShell v2.

References:

https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1

https://www.youtube.com/watch?v=z8thoG7gPd0

https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4728

ForceChangePassword
Info:

The attacker has the capability to change the target user's password without knowing that user's current password.

Abuse Info:

There are at least two ways to execute this attack. The first and most obvious is by using the built-in net.exe binary in Windows (e.g.: net user dfm.a Password123! /domain). See the opsec considerations tab for why this may be a bad idea. The second, and highly recommended method, is by using the Set-DomainUserPassword function in PowerView. This function is superior to using the net.exe binary in several ways. For instance, you can supply alternate credentials, instead of needing to run a process as or logon as the user with the ForceChangePassword privilege. Additionally, you have much safer execution options than you do with spawning net.exe (see the opsec tab).

To abuse this privilege with PowerView's Set-DomainUserPassword, first import PowerView into your agent session or into a PowerShell instance at the console. You may need to authenticate to the Domain Controller as a member of DC_3.DOMAIN.LOCAL if you are not running a process as a member. To do this in conjunction with Set-DomainUserPassword, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

```
$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then create a secure string object for the password you want to set on the target user:

$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force

Finally, use Set-DomainUserPassword, optionally specifying $Cred if you are not already running a process as target:

```
Set-DomainUserPassword -Identity andy -AccountPassword $UserPassword -Credential
$Cred
```

Now that you know the target user's plain text password, you can either start a new agent as that user, or use that user's credentials in conjunction with PowerView's ACL abuse functions, or perhaps even RDP to a system the target user has access to. For more ideas and information, see the references tab.

Opsec Considerations:

Executing this abuse with the net binary will necessarily require command line execution. If your target organization has command line logging enabled, this is a detection opportunity for their analysts.

Regardless of what execution procedure you use, this action will generate a 4724 event on the domain controller that handled the request. This event may be centrally collected and analyzed by security analysts, especially for users that are obviously very high privilege groups (i.e.: Domain Admin users). Also be mindful that PowerShell v5 introduced several key security features such as script block logging and AMSI that provide security analysts another detection opportunity. You may be able to completely evade those features by downgrading to PowerShell v2.

Finally, by changing a service account password, you may cause that service to stop functioning properly. This can be bad not only from an opsec perspective, but also a client management perspective. Be careful!

References:

https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1

https://www.youtube.com/watch?v=z8thoG7gPd0

https://www.sixdub.net/?p=579

https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4724

GenericAll

Info:

The user has GenericAll privileges to the target object. This is also known as full control. This privilege allows the trustee to manipulate the target object however they wish.

Abuse Info:

Full control of a user allows you to modify properties of the user to perform a targeted kerberoast attack, and also grants the ability to reset the password of the user without knowing their current one.

Targeted Kerberoast

A targeted kerberoast attack can be performed using PowerView's Set-DomainObject along with Get-DomainSPNTicket.

You may need to authenticate to the Domain Controller as source User if you are not running a process as that user. To do this in conjunction with Set-DomainObject, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

```
$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then, use Set-DomainObject, optionally specifying $Cred if you are not already running a process as the source User:

```
Set-DomainObject -Credential $Cred -Identity harmj0y -SET
@{serviceprincipalname='nonexistent/BLAHBLAH'}
```

After running this, you can use Get-DomainSPNTicket as follows:

```
Get-DomainSPNTicket -Credential $Cred harmj0y | fl
```

The recovered hash can be cracked offline using the tool of your choice. Cleanup of the ServicePrincipalName can be done with the Set-DomainObject command:

```
Set-DomainObject -Credential $Cred -Identity harmj0y -Clear serviceprincipalname
```

Force Change Password

There are at least two ways to execute this attack. The first and most obvious is by using the built-in net.exe binary in Windows (e.g.: net user dfm.a Password123! /domain). See the opsec considerations tab for why this may be a bad idea. The second, and highly recommended method, is by using the Set-DomainUserPassword function in PowerView. This function is superior to using the net.exe binary in several ways. For instance, you can supply alternate credentials, instead of needing to run a process as or logon as the user with the ForceChangePassword privilege. Additionally, you have much safer execution options than you do with spawning net.exe (see the opsec tab).

To abuse this privilege with PowerView's Set-DomainUserPassword, first import PowerView into your agent session or into a PowerShell instance at the console. You may need to authenticate to the

Domain Controller as the source User if you are not running a process as that user. To do this in conjunction with Set-DomainUserPassword, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

```
$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then create a secure string object for the password you want to set on the target user:

```
$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

Finally, use Set-DomainUserPassword, optionally specifying $Cred if you are not already running a process as the source User:

```
Set-DomainUserPassword -Identity andy -AccountPassword $UserPassword -Credential
$Cred
```

Now that you know the target user's plain text password, you can either start a new agent as that user, or use that user's credentials in conjunction with PowerView's ACL abuse functions, or perhaps even RDP to a system the target user has access to. For more ideas and information, see the references tab.

Opsec Considerations:

This depends on the target object and how to take advantage of this privilege. Opsec considerations for each abuse primitive are documented on the specific abuse edges and on the BloodHound wiki.

References:

https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1

https://www.youtube.com/watch?v=z8thoG7gPd0

https://adsecurity.org/?p=1729

http://www.harmj0y.net/blog/activedirectory/targeted-kerberoasting/

https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e

GenericWrite

Info:

Generic Write access grants you the ability to write to any non-protected attribute on the target object, including "members" for a group, and "serviceprincipalnames" for a user.

Abuse Info:

A targeted kerberoast attack can be performed using PowerView's Set-DomainObject along with Get-DomainSPNTicket.

You may need to authenticate to the Domain Controller as a member of the source Group if you are not running a process as a member. To do this in conjunction with Set-DomainObject, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

```
$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then, use Set-DomainObject, optionally specifying $Cred if you are not already running a process as group Member:

```
Set-DomainObject -Credential $Cred -Identity harmj0y -SET
@{serviceprincipalname='nonexistent/BLAHBLAH'}
```

After running this, you can use Get-DomainSPNTicket as follows:

```
Get-DomainSPNTicket -Credential $Cred harmj0y | fl
```

The recovered hash can be cracked offline using the tool of your choice. Cleanup of the ServicePrincipalName can be done with the Set-DomainObject command:

```
Set-DomainObject -Credential $Cred -Identity harmj0y -Clear serviceprincipalname
```

Opsec Considerations:

This depends on the target object and how to take advantage of this privilege. Opsec considerations for each abuse primitive are documented on the specific abuse edges and on the BloodHound wiki.

References:

https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1 https://www.youtube.com/watch?v=z8thoG7gPd0

http://www.harmj0y.net/blog/activedirectory/targeted-kerberoasting/

Owns

Info:

Object owners retain the ability to modify object security descriptors, regardless of permissions on the object's DACL

Abuse Info:

To abuse ownership of a user object, you may grant yourself the GenericAll privilege. This can be accomplished using the Add-DomainObjectAcl function in PowerView.

You may need to authenticate to the Domain Controller as a member of the source Group if you are not running a process as a member. To do this in conjunction with Add-DomainObjectAcl, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force

$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then, use Add-DomainObjectAcl, optionally specifying $Cred if you are not already running a process as a member of the source Group:

```
Add-DomainObjectAcl -Credential $Cred -TargetIdentity harmj0y -Rights All
```

Targeted Kerberoast

A targeted kerberoast attack can be performed using PowerView's Set-DomainObject along with Get-DomainSPNTicket.

You may need to authenticate to the Domain Controller as a member of the source Group if you are not running a process as a member. To do this in conjunction with Set-DomainObject, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

```
$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then, use Set-DomainObject, optionally specifying $Cred if you are not already running a process as a member of the source Group:

```
Set-DomainObject -Credential $Cred -Identity harmj0y -SET
@{serviceprincipalname='nonexistent/BLAHBLAH'}
```

After running this, you can use Get-DomainSPNTicket as follows:

```
Get-DomainSPNTicket -Credential $Cred harmj0y | fl
```

The recovered hash can be cracked offline using the tool of your choice. Cleanup of the ServicePrincipalName can be done with the Set-DomainObject command:

```
Set-DomainObject -Credential $Cred -Identity harmj0y -Clear serviceprincipalname
```

Force Change Password

There are at least two ways to execute this attack. The first and most obvious is by using the built-in net.exe binary in Windows (e.g.: net user dfm.a Password123! /domain). See the opsec considerations tab for why this may be a bad idea. The second, and highly recommended method, is by using the Set-DomainUserPassword function in PowerView. This function is superior to using the net.exe binary in several ways. For instance, you can supply alternate credentials, instead of needing to run a process as or logon as the user with the ForceChangePassword privilege. Additionally, you have much safer execution options than you do with spawning net.exe (see the opsec tab).

To abuse this privilege with PowerView's Set-DomainUserPassword, first import PowerView into your agent session or into a PowerShell instance at the console. You may need to authenticate to the Domain Controller as a member of the source Group if you are not running a process as a member. To do this in conjunction with Set-DomainUserPassword, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force

$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then create a secure string object for the password you want to set on the target user:

```
$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

Finally, use Set-DomainUserPassword, optionally specifying $Cred if you are not already running a process as a member of the target Group:

```
Set-DomainUserPassword -Identity andy -AccountPassword $UserPassword -Credential
$Cred
```

Now that you know the target user's plain text password, you can either start a new agent as that user, or use that user's credentials in conjunction with PowerView's ACL abuse functions, or perhaps even RDP to a system the target user has access to. For more ideas and information, see the references tab.

Cleanup of the added ACL can be performed with Remove-DomainObjectAcl:

```
Remove-DomainObjectAcl -Credential $Cred -TargetIdentity harmj0y -Rights All
```

Opsec Considerations:

When using the PowerView functions, keep in mind that PowerShell v5 introduced several security mechanisms that make it much easier for defenders to see what's going on with PowerShell in their network, such as script block logging and AMSI. You can bypass those security mechanisms by downgrading to PowerShell v2, which all PowerView functions support.

Modifying permissions on an object will generate 4670 and 4662 events on the domain controller that handled the request.

Additional opsec considerations depend on the target object and how to take advantage of this privilege. Opsec considerations for each abuse primitive are documented on the specific abuse edges and on the BloodHound wiki.

References:

https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1

https://www.youtube.com/watch?v=z8thoG7gPd0

http://www.selfadsi.org/deep-inside/ad-security-descriptors.htm

WriteOwner
<u>Info:</u> Object owners retain the ability to modify object security descriptors, regardless of permissions on the object's DACL.

<u>Abuse Info:</u>
 	
To change the ownership of the object, you may use the Set-DomainObjectOwner function in PowerView.
 	
You may need to authenticate to the Domain Controller as a member of the source Group if you are not running a process as a member. To do this in conjunction with Set-DomainObjectOwner, first create a PSCredential object (these examples comes from the PowerView help documentation):
 	
`$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force`
 	
`$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a', $SecPassword)`
 	
Then, use Set-DomainObjectOwner, optionally specifying $Cred if you are not already running a process as member of the source Group:
 	
`Set-DomainObjectOwner -Credential $Cred -TargetIdentity dfm -OwnerIdentity harmj0y`
 	
To abuse ownership of a user object, you may grant yourself the GenericAll privilege. This can be accomplished using the Add-DomainObjectAcl function in PowerView.
 	
You may need to authenticate to the Domain Controller as a member of the source Group if you are not running a process as a member. To do this in conjunction with Add-DomainObjectAcl, first create a PSCredential object (these examples comes from the PowerView help documentation):
 	
`$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force`
 	
`$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a', $SecPassword)`
 	
Then, use Add-DomainObjectAcl, optionally specifying $Cred if you are not already running a process as member of the source Group:
 	
`Add-DomainObjectAcl -Credential $Cred -TargetIdentity harmj0y -Rights All`

Targeted Kerberoast

A targeted kerberoast attack can be performed using PowerView's Set-DomainObject along with Get-DomainSPNTicket.

You may need to authenticate to the Domain Controller as a member of the source Group if you are not running a process as a member. To do this in conjunction with Set-DomainObject, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

```
$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then, use Set-DomainObject, optionally specifying $Cred if you are not already running a process as member of the source Group:

```
Set-DomainObject -Credential $Cred -Identity harmj0y -SET
@{serviceprincipalname='nonexistent/BLAHBLAH'}
```

After running this, you can use Get-DomainSPNTicket as follows:

```
Get-DomainSPNTicket -Credential $Cred harmj0y | fl
```

The recovered hash can be cracked offline using the tool of your choice. Cleanup of the ServicePrincipalName can be done with the Set-DomainObject command:

```
Set-DomainObject -Credential $Cred -Identity harmj0y -Clear serviceprincipalname
```

Force Change Password

There are at least two ways to execute this attack. The first and most obvious is by using the built-in net.exe binary in Windows (e.g.: net user dfm.a Password123! /domain). See the opsec considerations tab for why this may be a bad idea. The second, and highly recommended method, is by using the Set-DomainUserPassword function in PowerView. This function is superior to using the net.exe binary in several ways. For instance, you can supply alternate credentials, instead of needing to run a process as or logon as the user with the ForceChangePassword privilege. Additionally, you have much safer execution options than you do with spawning net.exe (see the opsec tab).

To abuse this privilege with PowerView's Set-DomainUserPassword, first import PowerView into your agent session or into a PowerShell instance at the console. You may need to authenticate to the Domain Controller as a member of the source Group if you are not running a process as a member. To do this in conjunction with Set-DomainUserPassword, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force

$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then create a secure string object for the password you want to set on the target user:

```
$UserPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
```

Finally, use Set-DomainUserPassword, optionally specifying $Cred if you are not already running a process as member of the source Group:

```
Set-DomainUserPassword -Identity andy -AccountPassword $UserPassword -Credential
$Cred
```

Now that you know the target user's plain text password, you can either start a new agent as that user, or use that user's credentials in conjunction with PowerView's ACL abuse functions, or perhaps even RDP to a system the target user has access to. For more ideas and information, see the references tab.

Cleanup of the added ACL can be performed with Remove-DomainObjectAcl:

```
Remove-DomainObjectAcl -Credential $Cred -TargetIdentity harmj0y -Rights All
```

Cleanup for the owner can be done by using Set-DomainObjectOwner once again

Opsec Considerations:

This depends on the target object and how to take advantage of this privilege. Opsec considerations for each abuse primitive are documented on the specific abuse edges and on the BloodHound wiki.

References:

https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1

http://www.selfadsi.org/deep-inside/ad-security-descriptors.htm

ReadLAPSPassword

Info:

The attacker has the ability to read the password set by Local Administrator Password Solution (LAPS) on the target computer. The local administrator password for a computer managed by LAPS is stored in the confidential LDAP attribute, "ms-mcs-AdmPwd".

Abuse Info:

To abuse this privilege with PowerView's Get-DomainObject, first import PowerView into your agent session or into a PowerShell instance at the console. You may need to authenticate to the Domain Controller as the source User if you are not running a process as that user. To do this in conjunction with Get-DomainObject, first create a PSCredential object (these examples comes from the PowerView help documentation):

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force

$Cred = New-Object System.Management.Automation.PSCredential('TESTLABdfm.a',
$SecPassword)
```

Then, use Get-DomainObject, optionally specifying $Cred if you are not already running a process as the source User:

```
Get-DomainObject windows1 -Credential $Cred -Properties "ms-mcs-AdmPwd",name
```

Opsec Considerations:

Reading properties from LDAP is an extremely low risk operation.

References:

https://www.specterops.io/assets/resources/an_ace_up_the_sleeve.pdf

https://adsecurity.org/?p=3164

A.5.3 – Containers

Contains
Info: GPOs linked to a container apply to all objects that are contained by the container.
Abuse Info: There is no abuse info related to this edge.
Opsec Considerations: There are no opsec considerations related to this edge.
References: https://wald0.com/?p=179 https://blog.cptjesus.com/posts/bloodhound15

GpLink
Info: A linked GPO applies its settings to objects in the linked container.
Abuse Info: There is no abuse info related to this edge.
Opsec Considerations: There are no opsec considerations related to this edge.
References: https://wald0.com/?p=179 https://blog.cptjesus.com/posts/bloodhound15

A.5.4 – Specials

CanRDP
<u>Info:</u> Remote Desktop access allows you to enter an interactive session with the target computer. If authenticating as a low privilege user, a privilege escalation may allow you to gain high privileges on the system. **Note**: This edge does not guarantee privileged execution.
<u>Abuse Info:</u> Abuse of this privilege will depend heavily on the type of access you have. **PlainText Credentials with Interactive Access** With plaintext credentials, the easiest way to exploit this privilege is using the built in Windows Remote Desktop Client (mstsc.exe). Open mstsc.exe and input the target Computer. When prompted for credentials, input the credentials for the source User to initiate the remote desktop connection. **Password Hash with Interactive Access** With a password hash, exploitation of this privilege will require local administrator privileges on a system, and the remote server must allow Restricted Admin Mode. First, inject the NTLM credential for the user you're abusing into memory using mimikatz: `sekurlsa::pth /user:dfm /domain:testlab.local /ntlm:<ntlm hash> /run:"mstsc.exe /restrictedadmin"` This will open a new RDP window. Input the computer name to initiate the remote desktop connection. If the target server does not support Restricted Admin Mode, the session will fail. **Plaintext Credentials without Interactive Access**

This method will require some method of proxying traffic into the network, such as the socks command in cobaltstrike, or direct internet connection to the target network, as well as the xfreerdp (suggested because of support of Network Level Authentication (NLA)) tool, which can be installed from the freerdp-x11 package. If using socks, ensure that proxychains is configured properly. Initiate the remote desktop connection with the following command:

```
(proxychains) xfreerdp /u:dfm /d:testlab.local /v:<computer ip>
```

xfreerdp will prompt you for a password, and then initiate the remote desktop connection.

Password Hash without Interactive Access

This method will require some method of proxying traffic into the network, such as the socks command in cobaltstrike, or direct internet connection to the target network, as well as the xfreerdp (suggested because of support of Network Level Authentication (NLA)) tool, which can be installed from the freerdp-x11 package. Additionally, the target computer must allow Restricted Admin Mode. If using socks, ensure that proxychains is configured properly. Initiate the remote desktop connection with the following command:

```
(proxychains) xfreerdp /pth:<ntlm hash> /u:dfm /d:testlab.local /v:<computer ip>
```

This will initiate the remote desktop connection, and will fail if Restricted Admin Mode is not enabled.

Opsec Considerations:

If the target computer is a workstation and a user is currently logged on, one of two things will happen. If the user you are abusing is the same user as the one logged on, you will effectively take over their session and kick the logged on user off, resulting in a message to the user. If the users are different, you will be prompted to kick the currently logged on user off the system and log on. If the target computer is a server, you will be able to initiate the connection without issue provided the user you are abusing is not currently logged in.

Remote desktop will create Logon and Logoff events with the access type RemoteInteractive.

References:

https://michael-eder.net/post/2018/native_rdp_pass_the_hash/

https://www.kali.org/penetration-testing/passing-hash-remote-desktop/

ExecuteDCOM

Info:

This can allow code execution under certain conditions by instantiating a COM object on a remote machine and invoking its methods.

Abuse Info:

The PowerShell script Invoke-DCOM implements lateral movement using a variety of different COM objects (ProgIds: MMC20.Application, ShellWindows, ShellBrowserWindow, ShellBrowserWindow, and ExcelDDE). LethalHTA implements lateral movement using the HTA COM object (ProgId: htafile).

One can manually instantiate and manipulate COM objects on a remote machine using the following PowerShell code. If specifying a COM object by its CLSID:

```
# Remote computer

$ComputerName = “TargetComputerName”

# GUID of the COM object

$clsid = “{fbae34e8-bf95-4da8-bf98-6c6e580aa348}”

$Type = [Type]::GetTypeFromCLSID($clsid, $ComputerName)

$ComObject = [Activator]::CreateInstance($Type)
```

If specifying a COM object by its ProgID:

```
# Remote computer

$ComputerName = “TargetComputerName”

$ProgId = “” # GUID of the COM object

$Type = [Type]::GetTypeFromProgID($ProgId, $ComputerName)

$ComObject = [Activator]::CreateInstance($Type)
```

Opsec Considerations:

The artifacts generated when using DCOM vary depending on the specific COM object used.

DCOM is built on top of the TCP/IP RPC protocol (TCP ports 135 + high ephemeral ports) and may leverage several different RPC interface UUIDs(outlined here). In order to use DCOM, one must be authenticated. Consequently, logon events and authentication-specific logs(Kerberos, NTLM, etc.) will be generated when using DCOM.

Processes may be spawned as the user authenticating to the remote system, as a user already logged into the system, or may take advantage of an already spawned process.

Many DCOM servers spawn under the process "svchost.exe -k DcomLaunch" and typically have a command line containing the string " -Embedding" or are executing inside of the DLL hosting process "DllHost.exe /Processid:{}" (where AppId is the AppId the COM object is registered to use). Certain COM services are implemented as service executables; consequently, service-related event logs may be generated.

References:

https://enigma0x3.net/2017/01/05/lateral-movement-using-the-mmc20-application-com-object/

https://enigma0x3.net/2017/01/23/lateral-movement-via-dcom-round-2/

https://enigma0x3.net/2017/09/11/lateral-movement-using-excel-application-and-dcom/

https://enigma0x3.net/2017/11/16/lateral-movement-using-outlooks-createobject-method-and-dotnettojscript/

https://www.cybereason.com/blog/leveraging-excel-dde-for-lateral-movement-via-dcom

https://www.cybereason.com/blog/dcom-lateral-movement-techniques

https://bohops.com/2018/04/28/abusing-dcom-for-yet-another-lateral-movement-technique/

https://attack.mitre.org/wiki/Technique/T1175

Invoke-DCOM

https://github.com/rvrsh3ll/Misc-Powershell-Scripts/blob/master/Invoke-DCOM.ps1

LethalHTA

https://codewhitesec.blogspot.com/2018/07/lethalhta.html

https://github.com/codewhitesec/LethalHTA/

AllowedToDelegate
Info: The constrained delegation primitive allows a principal to authenticate as any user to specific services (found in the msds-AllowedToDelegateTo LDAP property in the source node tab) on the target computer. That is, a node with this privilege can impersonate any domain principal (including Domain Admins) to the specific service on the target host. An issue exists in the constrained delegation where the service name (sname) of the resulting ticket is not a part of the protected ticket information, meaning that an attacker can modify the target service name to any service of their choice. For example, if msds-AllowedToDelegateTo is "HTTP/host.domain.com", tickets can be modified for LDAP/HOST/etc. service names, resulting in complete server compromise, regardless of the specific service listed.
Abuse Info: Abusing this privilege will require either using Benjamin Delpy's Kekeo project on a compromised host, or proxying in traffic generated from the Impacket library. See the references tab for more detailed information on exploiting this privilege.
Opsec Considerations: As mentioned in the abuse info, in order to currently abuse this primitive either the Kekeo binary will need to be dropped to disk on the target or traffic from Impacket will need to be proxied in. See the References for more information.
References: https://labs.mwrinfosecurity.com/blog/trust-years-to-earn-seconds-to-break/ http://www.harmj0y.net/blog/activedirectory/s4u2pwnage/ https://twitter.com/gentilkiwi/status/806643377278173185 https://www.coresecurity.com/blog/kerberos-delegation-spns-and-more

B – BloodHound Install & User Interface

B.1 – Install Guide

BloodHound is a self-contained Electron webApp running on top of a Neo4j database.

All the necessary components can be downloaded from the web.

Java 64/32bit: Download

Neo4j Community Edition: Download

BloodHound Binaries: Download

BloodHound Master: Download

Install procedure depends on your OS flavor.

Most up-to-date install procedure can be found here.

B.1.1 – Windows

1. Neo4j requires Java, so make sure you're running the latest version of Java.
2. Go to neo4j.com/download and click on "Download Server"
3. Download the current version of neo4j Server for Windows, selecting either 32 or 64 bit.
4. Extract the contents of the zip folder you downloaded in step 4.
5. Open cmd.exe running as an administrator, and navigate to the folder you extracted the zip into in step 4.
6. CD into the bin directory, and install neo4j as a service by running neo4j.bat install-service
7. Go back to cmd.exe and start neo4j by typing net start neo4j
8. Verify neo4j is running by navigating to http://localhost:7474/ in a browser. The neo4j web console should show up here.
9. Run BloodHound.exe from the release found here or build BloodHound from source.
10. Authenticate to the provided sample graph database at bolt://localhost:7687. The username is "neo4j", and the password is "BloodHound"

Tip: There is also a full neo4j powershell module in the bin\Neo4j-Management folder.

B.1.2 – Linux

1. Download and install neo4j community edition.
2. Optional: configure the REST API to accept remote connections if you plan to run neo4j and the PowerShell ingestor on different hosts.
3. Clone the BloodHound GitHub repo.
4. git clone https://github.com/adaptivethreat/Bloodhound
5. Start the neo4j server

6. Run BloodHound from the release found here or build BloodHound from source.
7. ./BloodHound
8. Authenticate to the provided sample graph database at bolt://localhost:7687. The username is "neo4j", and the password is "BloodHound".

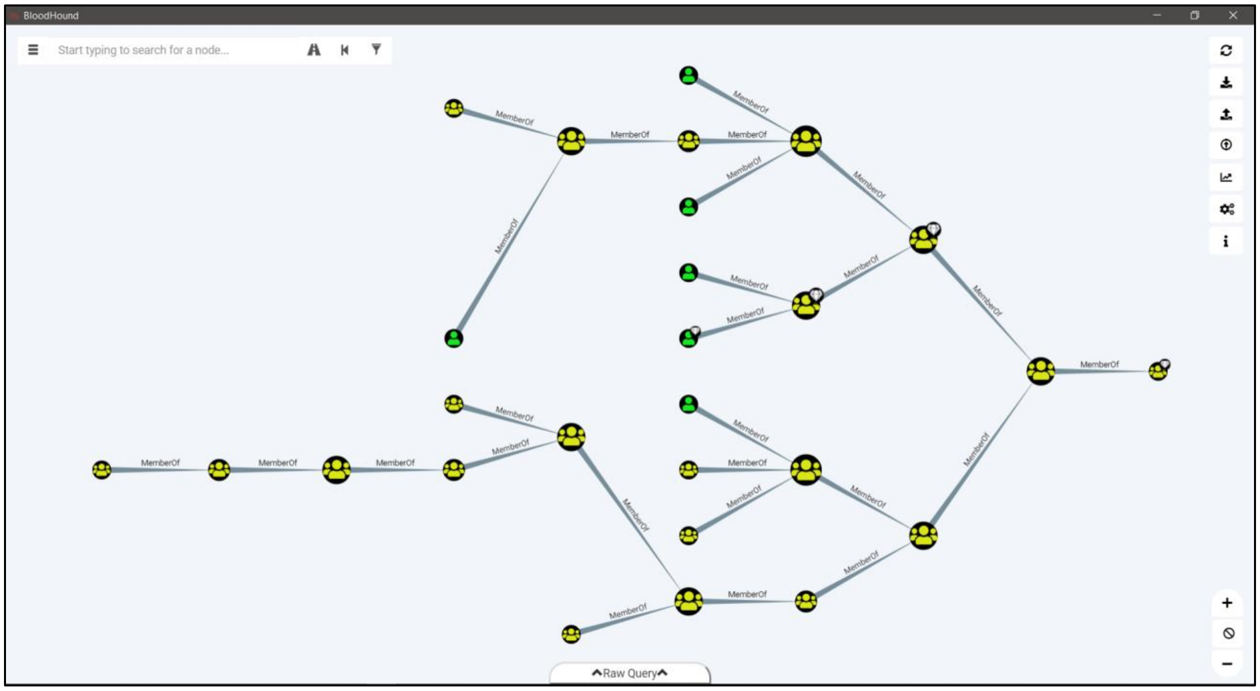
Note: On Kali distro, Bloodhound, including neo4j, can be installed with a simple apt-get.

B.1.3 – Mac/OSX

1. Download and install neo4j community edition.
2. Optional: configure the REST API to accept remote connections if you plan to run neo4j and the PowerShell ingestor on different hosts.
3. Clone the BloodHound GitHub repo.
4. git clone https://github.com/adaptivethreat/Bloodhound
5. Start the neo4j server, pointing neo4j to the provided sample graph database.
6. Run the BloodHound App from the release found here or build BloodHound from source.
7. Authenticate to the provided sample graph database at bolt://localhost:7687. The username is "neo4j", and the password is "BloodHound"

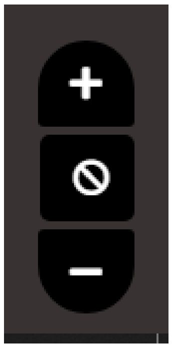
B.2 – UI Tour & Features

Ok so let's take a look at this puppy… BloodHound fits in a single web page. Once logged in, you won't have to move anymore… It's all here.



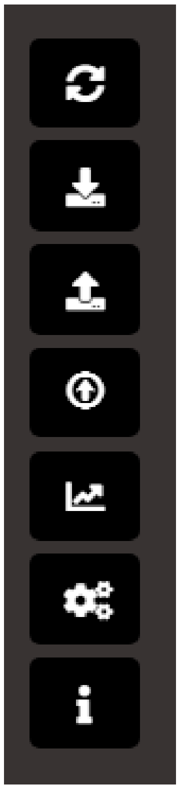
First thing first, bloodhound is a Hacker tool, so the good @CptJesus had the kindness to bless us with a **Dark Theme**. Simply click on the Setting Icon in the top left corner (2nd from the bottom) and select Dark Theme. Toggle it a couple of times and select the one you like…
… Dark? I kind of knew it…

Now let's take a closer look:



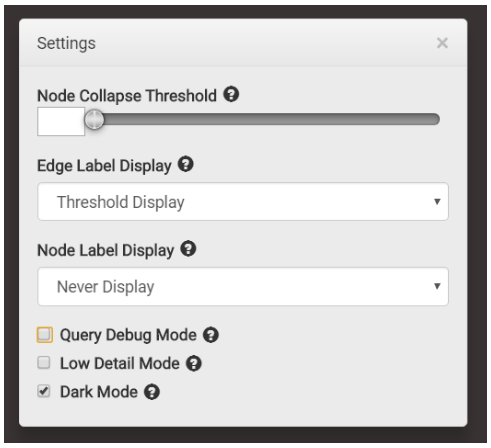
These three icons in the bottom right corner are for the Zoom.

You can **Zoom In/Out** and **Reset to default** view.



In the top right corner, you will find most of the **Settings & Import/Export** functionalities.

The bottom one is for **Software Info**. Click on it and you will see what version of bloodhound you are running.
Above it is the **Settings** icon. See below for more details.
The little graph icon is used to change **Graph Layout** between Directed and Hierarchical. Hierarchical produces a nice and easy to read output. Directed can create hard to read output. But a bit like clouds, sometimes you see stuff… I heard @_Wald0 is collecting them to later expose the finest pieces of the community…
Next are the Import/Export and Upload buttons.
The Circled arrow at the bottom is for **Data Upload**. This is a way to ingest collected zip data. The two other ones are for **Export/Import**. Graph data can be exported as PNG to illustrate engagement reports. Data can also be exported as JSON and then imported back from JSON at a later time.
Finally, the top button is used to **Refresh Current Graph**.


Settings

Node Collapse Threshold

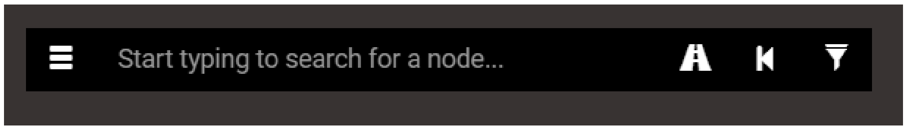
Edge Label Display: Threshold Display

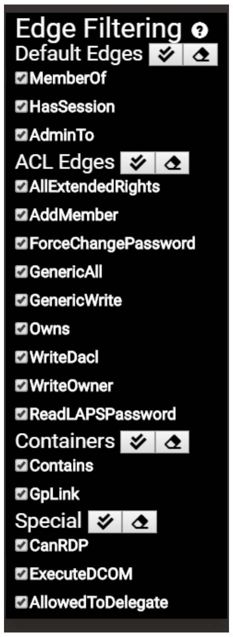
Node Label Display: Never Display

☐ Query Debug Mode
☐ Low Detail Mode
☑ Dark Mode

This is the **Settings Dialogue Box**. Here you can specify how you want nodes and labels to be displayed.
Easier to try it out than to explain.
Check it out.

Tip: The Debug Mode will display the query used in the raw query input box, each time you click somewhere in the UI.
Really cool for learning Cypher.




Edge Filtering

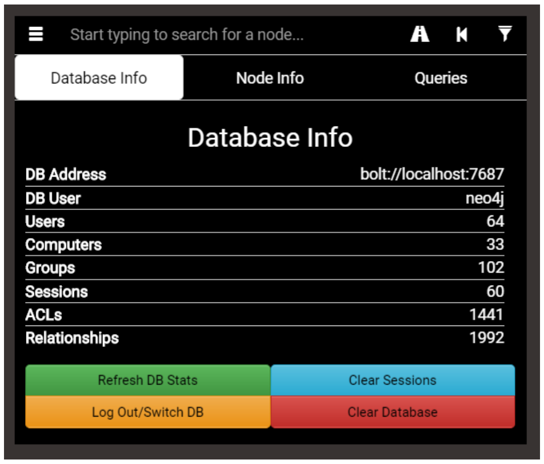
Default Edges
☑ MemberOf
☑ HasSession
☑ AdminTo

ACL Edges
☑ AllExtendedRights
☑ AddMember
☑ ForceChangePassword
☑ GenericAll
☑ GenericWrite
☑ Owns
☑ WriteDacl
☑ WriteOwner
☑ ReadLAPSPassword

Containers
☑ Contains
☑ GpLink

Special
☑ CanRDP
☑ ExecuteDCOM
☑ AllowedToDelegate

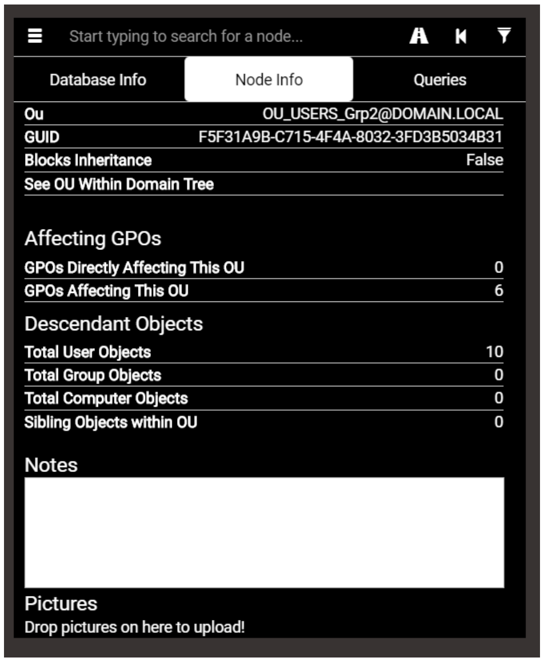
This is where a lot of the action happens. As indicated you can **search for nodes**. By clicking on the little **highway icon**, the box will expand and allow you to insert an End Node and request Paths. Clicking on the **Filter** icon will let you **enable/disable Edges** for all your shortestPath queries.
You could for example decide to disable ACL Edges and not show any path with these. [/!\ do not forget to turn them back on when needed...]
Finally, clicking on the icon in the left corner will as smoothly expand into some essential tabs. See below for details



Database Info

DB Address	bolt://localhost:7687
DB User	neo4j
Users	64
Computers	33
Groups	102
Sessions	60
ACLs	1441
Relationships	1992

This first tab is the **Database Info Tab**.
Quite self-explanatory.
The place to **remove all sessions** if needed (for example when looping session collection)
@CptJesus has put several warnings to prevent you from doing so by accident, but this is also where you can **clear the database**.



Start typing to search for a node...

Database Info	Node Info	Queries

Ou	OU_USERS_Grp2@DOMAIN.LOCAL
GUID	F5F31A9B-C715-4F4A-8032-3FD3B5034B31
Blocks Inheritance	False
See OU Within Domain Tree	

Affecting GPOs

GPOs Directly Affecting This OU	0
GPOs Affecting This OU	6

Descendant Objects

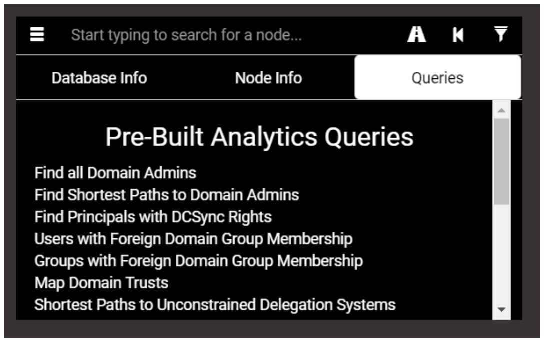
Total User Objects	10
Total Group Objects	0
Total Computer Objects	0
Sibling Objects within OU	0

Notes

Pictures

Drop pictures on here to upload!

The second tab is the **Node Info Tab**.
Here you will find all the info you need per type of node.
You can add notes to a node as you progress in your engagements if needed. You can even store screenshots if needed…
Clicking on any node in the graph will automatically open this tab for that node.
Clicking on a count will display the matching Graph.



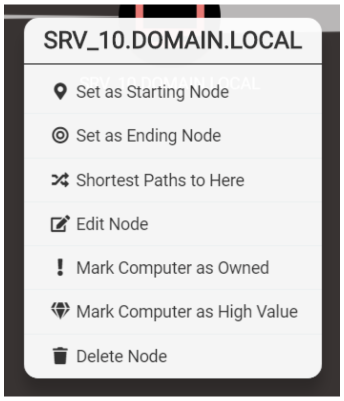
Start typing to search for a node...

Database Info | Node Info | Queries

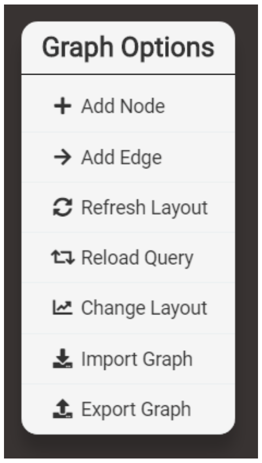
Pre-Built Analytics Queries

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

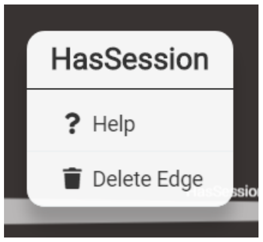
The last tab is the **Query Tab.**
Here you will find all the build-in Cypher queries, and you can also add your favorite custom Queries to the interface. For more info see E.1



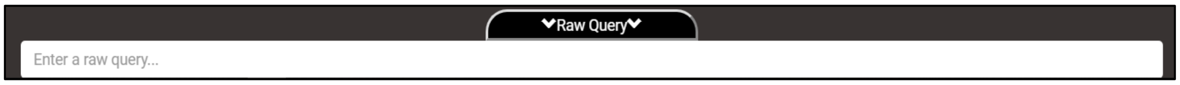
Right-clicking on a Node will open the **Node Context Menu** will all the good stuff. Also quite self-explanatory.
Try them all.



Right-clicking on an empty space in the graph will bring up the **Graph Option Context Menu**.
Here you can **add Nodes and Edges**.
You will also find some options from side bar repeated.



Right-Clicking on an Edge, will open the **Edge Context Menu** and allow you to get **info** on that type of edge, or **remove** it if you need to.



Next, at the bottom of the screen is the **Raw Query** input box.
This is where the magic happens. Here you can **input Cypher**. This is where you **ask questions** to the DB . More on this in the rest of this guide...

Keyboard Shortcuts

On top of all these features, here is a list of **keyboard shortcuts** @CptJesus has hidden in there...

Key	Action
[SPACE]	Node Seach
[CTRL]	Node Names ON/OFF
[CTRL+SHIFT+I]	Dev Console / Debug
[CTRL+R]	Restart Bloodhound

C – Data Collection & Ingestion

Bloodhound data is collected by various **LDAP queries** and **Win32 API calls**. Originally written in pure PowerShell by @harmj0y [and based on PowerView commands], the whole data collector has been rewritten in **C#** by @CptJesus and offers much improvement with regards to speed and stability.

As a true security professional, you are of course interested in how exactly this data is collected before running it in your environment. Np, @CptJesus has got your back. Check it out. (**MANDATORY READ**)

C.1 – Data Collection

The latest version of BloodHound come with two flavors of the Collector: **Sharphound.exe**, a C# executable, and **SharpHound.ps1** (Invoke-Bloodhound), its matching PowerShell script containing a Base64 encoded version of SharpHound to be executed against the targeted AD.
Both have various switches to specify what info is to be collected.
Both tools come with **Help Pages**… RTFM. Invoke-BloodHound code can be found here.

Note: Running the collector should have little to **no impact on a healthy AD environment**. Only a few LDAP and API calls to the DC are enough to collect this data. If you manage to bring your AD down with BloodHound data collection, security is not the only concern you should have…

For more info on all the various switches, check this post by @CptJesus [**MANDATORY READ**]

Finally, If you are interested in a Python collector, check out this work by @_dirkjan.

Note: If the machine you are running the collector from is **not domain joined**, you can use the good old **runas /netonly** to do the trick.

Note: Performance might become an issue when querying very large datasets. When dealing with such AD environments, it could be recommended to collect each domain in a separate database to improve performance.

C.2 – Data Ingestion

Running the collector will output a zip file containing several JSON files with all the collected data. Ingesting data can be done by simply dragging this zip into the graph area of the UI, or by clicking on the upload icon as seen in previous section.

C.3 – Sample Database

BloodHound used to come with a Sample Database for practice. However, this sample database has not been updated since the initial release of the tool and cannot be used with the latest versions.
For practice, and if you don't have an AD at hand to collect data from, you can use this dummy dataset created for training purposes.

To set it up, simply unzip it, drop the folder in
[somewhere] \neo4j\neo4j-community-X.X.X\data\databases\
and adjust the name of the DB in the neo4j config file located at
[somewhere] \neo4j\neo4j-community-X.X.X\conf\neo4j.conf

Should look something like this:

```
# The name of the database to mount
dbms.active_database=DataBaseNameGoesHere
```

Note: This **data is fake and does not reflect any real AD infrastructure**.
It was (pseudo-randomly) generated for practice only.
Some of the Edges between Node would most probably not make much sense in a real AD environment, and hopefully your own AD doesn't look as bad as this one...

If you want to generate some fake data, @CptJesus also has a tool for that right here.

D - Basic Cypher & Common Queries

Ok, so by now you should be comfortable with the basic bloodhound concepts and ready to dig into the core of it: Cypher.

Cypher is the **Neo4j database query language**. It works a bit like Lego and is quite graphic. The creators of neo4j describe it as **a mix of SQL Queries and ASCII art**… In this section, we will introduce some basic cypher query building blocks, and see how they apply to Bloodhound.

For a quick reference on Cypher language click here. [Tip: bookmark for future use…]

A full guide to neo4j Cypher can also be found here if you want to dig deeper.

D.1 – Neo4j Cypher 101

Cypher is a very "visual" language. It was designed with ASCII art in mind. Who doesn't love ASCII art…

A simplified cypher query could look like this:

```
(This)-[IsConnectedTo]->(That)
```

This and **That** are Nodes. **IsConnectedTo** is the Edge between them, but this is not what I want to highlight here.

The important parts are the **brackets and arrow** (the ASCII art). This is your basic Path query construct.

It's a bit confusing at first, but you will get used to it very quickly.

Now let's look at some real query syntax…

In cypher, your two basic instructions will be **MATCH** and **RETURN**.

MATCH will instruct neo4j what to look for.

RETURN indicates what results you want to see.

```
$ MATCH (X) RETURN X
```

returns all Nodes in the database

```
$ MATCH (X:User) RETURN X
```

returns all Users in the database

```
$ MATCH (X:Group) RETURN X.name
```

returns only the name property of all Groups

Note: The type of node is called a “**Label**” in the official neo4j vocabulary.

Now we can make things a bit more interesting. Let’s return all users member of a specific group:

```
MATCH (U:User)
MATCH (G:Group {name: "ADMINISTRATORS@DOMAIN.LOCAL"})
MATCH (U)-[MemberOf]->(G)
RETURN U
```

Here we first ask for all users and store it in a variable U, then for a group called “ADMINISTRATORS@DOMAIN.LOCAL” and we store it in a variable G.

From that list of users U, we filter who is member of the specified group G, and finally return these User nodes.

This query can also be written with the following **equivalent syntaxes**:

```
MATCH (U:User),
(G:Group {name: "ADMINISTRATORS@DOMAIN.LOCAL"}),
(U)-[MemberOf]->(G)
RETURN U
```

```
MATCH (G:Group {name: "ADMINISTRATORS@DOMAIN.LOCAL"}),
(U:User)-[MemberOf]->(G)
RETURN U
```

```
MATCH (U:User)-[MemberOf]->(G:Group {name: "ADMINISTRATORS@DOMAIN.LOCAL"})
RETURN U
```

Note: Cypher language is case-sensitive, Proper casing of Nodes properties, Labels and other syntax elements is the first thing to check when debugging hanging queries…
(Do not worry about the warning icons for now, more on this later…)

Things will get more complicated as we dig deeper, but for now, if you understood the above syntaxes, you are good to go. All good? Cool. So let's dig deeper…

Note: Cypher queries in this guide cannot be copy-pasted. This was done on purpose. The idea is that you type them so as to get the hang of it. Sorry. ;)

D.2 – Common BloodHound Queries

D.2.1 – Querying Nodes

```
$ MATCH (x:Computer {name: 'ThisComputerName'}) RETURN x
```

Returns Computer nodes with name 'ThisComputerName'

```
$ MATCH (x:Computer {domain: 'ThisDomain'}) RETURN x
```

Returns Computer nodes where the domain property is equal to 'ThisDomain'

```
$ MATCH (x:Computer) WHERE x.domain = 'ThisDomain' RETURN x
```

Same as previous using the WHERE clause

The **WHERE** clause is used to filter Nodes per property. It is used in combination with Comparison Operators. In case of "Is Equal To" comparison, the shorter construct ("Map") is preferred.

Node by Property - Property Exists

```
⚠ $ MATCH (n:User) WHERE exists(n.ThisProperty) RETURN n
```

Returns all Nodes that have a property 'ThisProperty' (value or not)

Node by Property - Does Not Exists

```
⚠ $ MATCH (n:User) WHERE NOT exists(n.ThisProperty) RETURN n
```

Returns all Users that don't have a property called 'ThisProperty'

Node by Property - Property Value

```
⚠ $ MATCH (n:User) WHERE n.ThisProperty='ThisValue' RETURN n
```

Returns all Users that have a property 'ThisProperty' with value 'ThisValue'

```
⚠ $ MATCH (X:Group) WHERE X.name CONTAINS 'KeyWord' RETURN X
```

Returns All Groups with 'KeyWord' in name property (case sensitive)

```
  $ MATCH (X:Group) WHERE X.name=~'(?i).*kEywORd.*' RETURN X
```

Same as previous example but using RegEx [(?i) = case insensitive]

Comparison Operators

List of operators that can be used with the WHERE clause:

OPERATOR	SYNTAX
Is Equal To	=
Is Not Equal To	<>
Is Less Than	<
Is Greater Than	>
Is Less or Equal	<=
Is Greater or Equal	>=
Is Null	IS NULL
Is Not Null	IS NOT NULL
Prefix Search*	STARTS WITH
Suffix Search*	ENDS WITH
Inclusion Search*	CONTAINS
RegEx*	=~

* String specific

D.2.2 – Querying Edges

Group Membership – Direct

```
MATCH
(U:User),
(G:Group {name: 'DOMAIN ADMINS@DOMAIN.LOCAL'}),
m=(U)-[r:MemberOf]->(G)
RETURN m
```

Group Membership – Max Degree 3

```
MATCH
(U:User),
(G:Group {name: 'DOMAIN ADMINS@DOMAIN.LOCAL'}),
m=(U)-[r:MemberOf*1..3]->(G)
RETURN m
```

Group Membership – Any Degree

```
MATCH
(U:User),
(G:Group {name: 'DOMAIN ADMINS@DOMAIN.LOCAL'}),
m=(U)-[r:MemberOf*1..]->(G)
RETURN m
```

Note: Here we return paths to visualize nested groups in BloodHound. If you want to return just the User Nodes you can replace m by U in the RETURN clause of the queries.

D.2.3 – Querying Paths

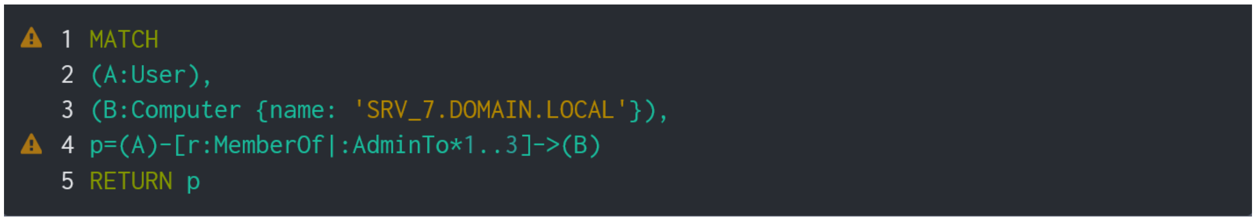
Shortest Path from A to B - any Edge type / One or more hops

```
MATCH
(A:User {name: 'ZACHERY_SPATZ@DOMAIN.LOCAL'}),
(B:Group {name: 'DOMAIN ADMINS@DOMAIN.LOCAL'}),
x=shortestPath((A)-[*1..]->(B))
RETURN x
```

Shortest Path from A to B - specific Edge types / One or more hops

```
MATCH
(A:User {name: 'MICHEAL_MAURER@DOMAIN.LOCAL'}),
(B:Group {name: 'DOMAIN ADMINS@DOMAIN.LOCAL'}),
x=shortestPath((A)-[:HasSession|:AdminTo|:MemberOf*1..]->(B))
RETURN x
```

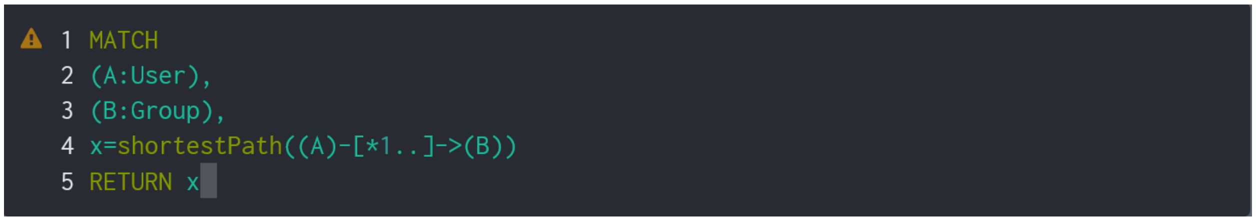
Shortest Path Any to One – Specific Edge type / Max hop count



```
MATCH
(A:User),
(B:Computer {name: 'SRV_7.DOMAIN.LOCAL'}),
p=(A)-[r:MemberOf|:AdminTo*1..3]->(B)
RETURN p
```

All user, max 3 degrees away by group membership, admin to specified target computer

Shortest Path Any to Any

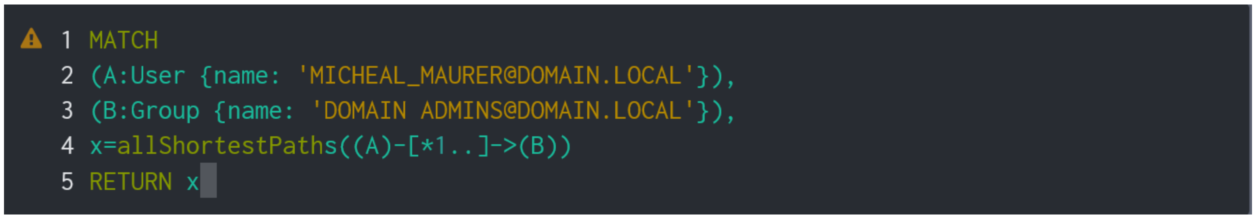


```
MATCH
(A:User),
(B:Group),
x=shortestPath((A)-[*1..]->(B))
RETURN x
```

Shortest paths from any user to any group

/!\ Any-to-Any are heavy queries and might hang with large datasets

All Shortest Paths



```
MATCH
(A:User {name: 'MICHEAL_MAURER@DOMAIN.LOCAL'}),
(B:Group {name: 'DOMAIN ADMINS@DOMAIN.LOCAL'}),
x=allShortestPaths((A)-[*1..]->(B))
RETURN x
```

The **allShortestPaths()** function works the same way as shortestPath() but returns all possible shortest paths [= more ways to get to target with same amount of hops]

/!\ Might need to restrict Edge type/max hops for heavy queries

All Paths

It is possible to request all available paths, even the longer ones, if you remove the **shortestPath()** or **allShortestPaths()** from your queries. This is however risky, and your query might hang. Make sure you specify at least one node name when using it. Do not try this on an Any-to-Any query.

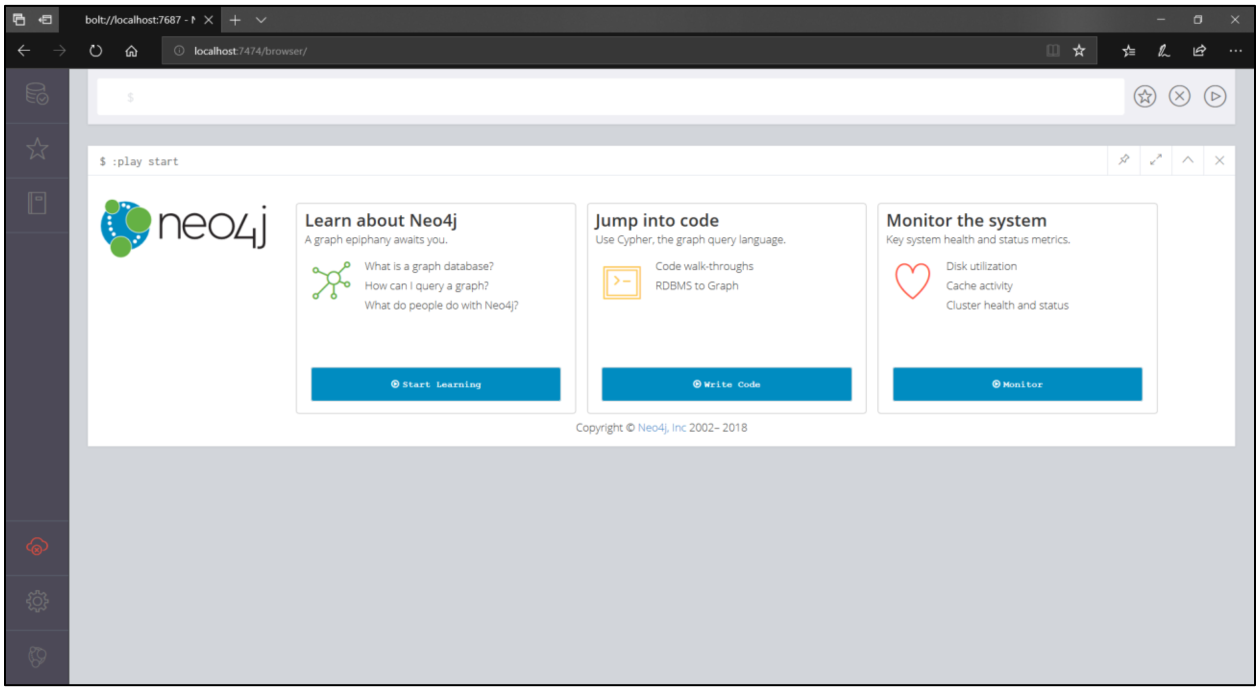
There is way more to the Cypher language, but with the above syntaxes, you should be able to graph most of what you need from the bloodhound database.

D.3 - Neo4j Browser & Basic Metrics

D.3.1 –Neo4j Browser

BloodHound is awesome for visualizing complex Active Directory object relationships in a graphical way. However, the graph output is not always what you are looking for.
Let's say you want to know how many users are member of a specific Group and have a specific property with cypher… Or you want the list of names… The UI is not really designed for this type of queries.

For these queries, you can use the **Neo4j browser** located at http://localhost:7474/Browser
this will take you to the following page:



Since you are a hacker, the first thing you probably ask is:

"Is there a **Dark Theme**…?!"

Rejoice. There is.

Simply click on the Settings icon in the bottom left corner… Et voila!

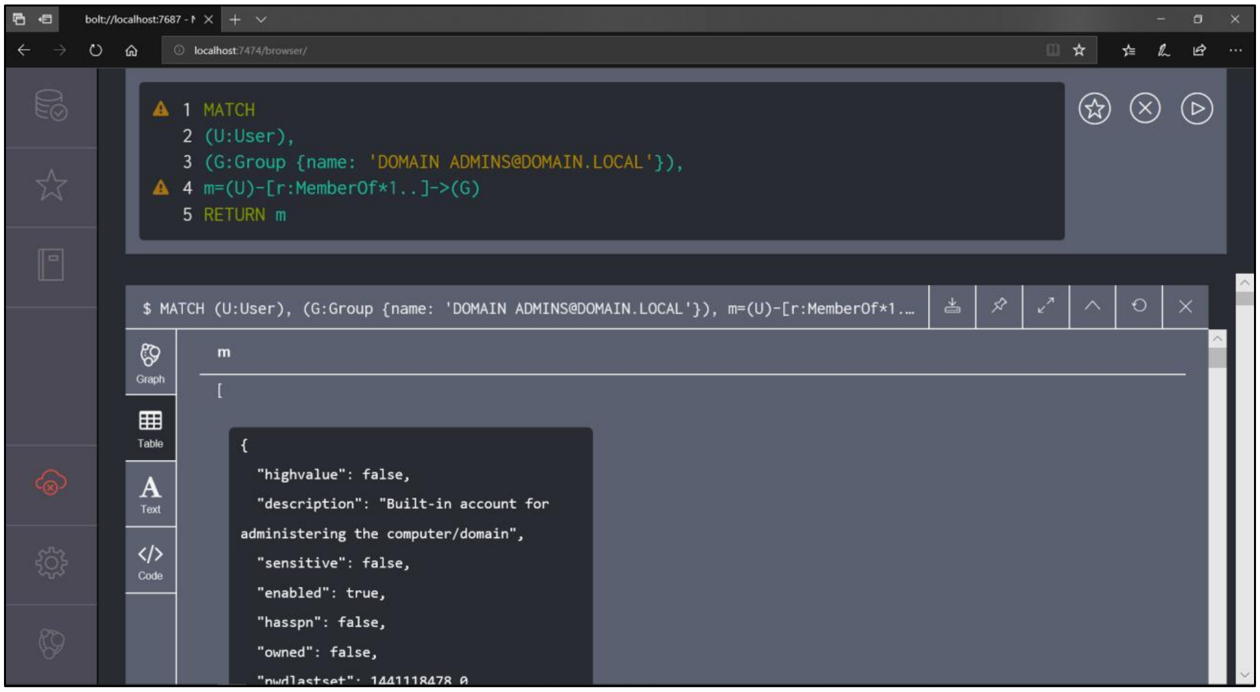
Second thing you can do is click away that Neo4j banner (tiny cross, top right corner…).

Now we are talking…

The neo4j browser is an awesome place to practice writing your queries. You now have multiline, syntax highlighting, error messages, and you can even set the font size (ctrl +/-).

The interface is quite simple and intuitive, I'll let you have a look around and click everywhere…

Just make yourself at home…



```
MATCH
(U:User),
(G:Group {name: 'DOMAIN ADMINS@DOMAIN.LOCAL'}),
m=(U)-[r:MemberOf*1..]->(G)
RETURN m
```

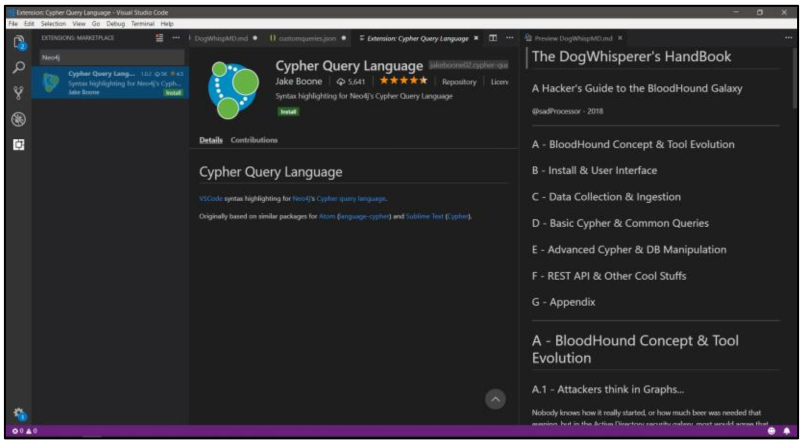
```
$ MATCH (U:User), (G:Group {name: 'DOMAIN ADMINS@DOMAIN.LOCAL'}), m=(U)-[r:MemberOf*1...
```

```
m
[
  {
    "highvalue": false,
    "description": "Built-in account for
administering the computer/domain",
    "sensitive": false,
    "enabled": true,
    "hasspn": false,
    "owned": false,
    "pwdlastset": 1441118478.0
```

And now you know where I made all the gorgeous cypher screenshots found in this guide…

(and you can finally hover over the warning icon to see what they mean…)

<u>TIP:</u> When in query input box, hit the Up arrow for previous queries (or Ctrl+Up if multiline). Hit escape to toggle editor mode. If you really like that query you just wrote, save for future usage by hitting on the star next to the cross and the play buttons. Hit the star on the left sidebar to view all your saves queries.


Extension: Cypher Query Language - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXTENSIONS: MARKETPLACE
Neo4j
DogWhispMD.md
customqueries.json
Extension: Cypher Query Language
Preview DogWhispMD.md
Install
Details
Contributions

Cypher Query Language
Jake Boone | 5,641 | ★★★★★ | Repository
Syntax highlighting for Neo4j's Cypher Query Language

Cypher Query Language

VSCode syntax highlighting for Neo4j's Cypher query language.

Originally based on similar packages for Atom (language-cypher) and Sublime Text (Cypher).

The DogWhisperer's HandBook

A Hacker's Guide to the BloodHound Galaxy

@sadProcessor - 2018

A - BloodHound Concept & Tool Evolution
B - Install & User Interface
C - Data Collection & Ingestion
D - Basic Cypher & Common Queries
E - Advanced Cypher & DB Manipulation
F - REST API & Other Cool Stuffs
G - Appendix

A - BloodHound Concept & Tool Evolution

A.1 - Attackers think in Graphs…

Nobody knows how it really started, or how much beer was needed that

VSCode , has a **Neo4j extension** for Cypher syntax highlighting.

ATOM and Sublime as well.

Choose your favorite weapon…

D.3.2 – Basic Metric Queries

So now that you have a new cool place to practice your Cypher, let's have a look at some basic metrics. For this purpose, we will now introduce new cypher building blocks: the **count()** function, and the **ORDER BY** and **LIMIT** clauses

Top 10 Computers with Most Sessions

```
MATCH p=((S:Computer)-[r:HasSession]->(T:User))
WITH S.name as s,
COUNT(DISTINCT(T)) as t
RETURN {Name: s, Count: t} as MyResult
ORDER BY t DESC
LIMIT 10
```

Top 10 Users with Most Sessions

```
MATCH p=((S:Computer)-[r:HasSession]->(T:User))
WITH T.name as n,
COUNT(S) as c
RETURN {Name: n, Count: c} as MyResult
ORDER BY c DESC
LIMIT 10
```

Top 5 Users with Most Admin Rights

```
MATCH p=((S:User)-[r:MemberOf|:AdminTo*1..]->(T:Computer))
WITH S.name as s,
COUNT(DISTINCT(T)) as t
RETURN {Name: s, Count: t} as MyResult
ORDER BY t DESC
LIMIT 5
```

Notice the cool "Map" construct on the RETURN clause. This is a cool tip to build nice objects on the fly and return only the data you want. You are going to love this trick if you work with the REST API. More on this later…

Using constructs similar to the above queries, you can count about anything you like in BloodHound. Management loves data… SHOW DEM DATA.
(for an advanced metric example with percentages and more, see section E.3)

E – Advanced Cypher & DB Manipulation

Some say that if you look at cypher long enough, it will infuse in you… True, but I think the best way is to practice, make mistakes and understand them. This section will be mostly queries with little to no explanation. The idea here is to try them out in the editor, play with them, modify them a bit, see what happens. Tweak them, break them, fix them… Have fun.

E.1 – Moar Cypher

Cypher is a bit like Lego. And if you are still reading all this, by now you should have all basic blocks in your box to build about any model. But there is more: Lego Technics… The big boy stuff.

Let's take a look at some more queries. Some new neo4j clauses are used here, allowing for new constructs. See if you can spot them… Let's start gently with some basics again:

Retrieving all "Owned" Nodes

```
$ MATCH (O:User {owned: True}) RETURN O
```

Retrieving all "HighValue" Groups

```
$ MATCH (H:Group {highvalue: True}) RETURN H
```

Path from “Owned” to “HighValue”

```
MATCH (O:User {owned: True})
MATCH (H:Group {highvalue: True})
MATCH p=shortestPath((O)-[*1..]->(H))
RETURN p
```

Nothing really new yet, but this last one is a useful one, so I thought I’d put it in here.

Now let’s dive into some more Cypher by looking at the source code of the build-in queries.

Again, observe and try to figure out what is happening. Remember to check the Neo4j Reference Card if you need.

E.1.1 – Build-in Queries

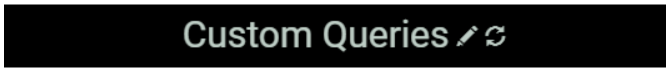
```
{
    "queries": [
        {
            "name": "Find all Domain Admins",
            "queryList": [
                {
                    "final": true,
                    "query":
                        "MATCH (n:Group) WHERE n.objectsid =~ {name} WITH n MATCH p=(n)<-[r:MemberOf*1..]-
(m) RETURN p",
                    "props": {
                        "name": "(?i)S-1-5-.*-512"
                    },
                    "allowCollapse": false
                }
            ]
        },
        {
            "name": "Find Shortest Paths to Domain Admins",
            "queryList": [
                {
                    "final": false,
                    "title": "Select a Domain Admin group...",
                    "query":
                        "MATCH (n:Group) WHERE n.objectsid =~ {name} RETURN n.name ORDER BY n.name DESC",
                    "props": {
                        "name": "(?i)S-1-5-.*-512"
                    }
                },
                {
                    "final": true,
                    "query":
                        "MATCH (n:User),(m:Group {name:{result}}),p=shortestPath((n)-[r:{}*1..]->(m)) RETURN
p",
                    "allowCollapse": true,
                    "endNode": "{}"
                }
            ]
        },
        {
            "name": "Find Principals with DCSync Rights",
            "queryList": [
                {
                    "final": false,
                    "title": "Select a Domain...",
                    "query":
```

```
                    "MATCH (n:Domain) RETURN n.name ORDER BY n.name DESC"
                },
                {
                    "final": true,
                    "query":
                        "MATCH p=(n1)-[:MemberOf|GetChanges*1..]->(u:Domain {name: {result}}) WITH p,n1 MATCH
p2=(n1)-[:MemberOf|GetChangesAll*1..]->(u:Domain    {name:    {result}})    WITH    p,p2    MATCH    p3=(n2)-
[:MemberOf|GenericAll|AllExtendedRights*1..]->(u:Domain {name: {result}}) RETURN p,p2,p3",
                    "allowCollapse": true,
                    "endNode": "{}"
                }
            ]
        },
        {
            "name": "Users with Foreign Domain Group Membership",
            "queryList": [
                {
                    "final": false,
                    "title": "Select source domain...",
                    "query":
                        "MATCH (n:Domain) RETURN n.name ORDER BY n.name DESC"
                },
                {
                    "final": true,
                    "query":
                        "MATCH (n:User) WITH n MATCH (m:Group) WITH n,m MATCH p=(n)-[r:MemberOf]->(m) WHERE
n.domain={result} AND NOT m.domain=n.domain RETURN p",
                    "startNode": "{}",
                    "allowCollapse": false
                }
            ]
        },
        {
            "name": "Groups with Foreign Domain Group Membership",
            "queryList": [
                {
                    "final": false,
                    "title": "Select source domain...",
                    "query":
                        "MATCH (n:Domain) RETURN n.name ORDER BY n.name DESC"
                },
                {
                    "final": true,
                    "query":
                        "MATCH (n:Group) WITH n MATCH (m:Group) WITH n,m MATCH p=(n)-[r:MemberOf]->(m) WHERE
n.domain={result} AND NOT m.domain=n.domain AND NOT n.name=m.name RETURN p",
                    "startNode": "{}",
                    "allowCollapse": false
                }
            ]
        },
        {
            "name": "Map Domain Trusts",
            "queryList": [
                {
                    "final": true,
                    "query": "MATCH p=(n:Domain)-[r]-(m:Domain) RETURN p",
                    "allowCollapse": true
                }
            ]
        },
        {
            "name": "Shortest Paths to Unconstrained Delegation Systems",
            "queryList": [
                {
                    "final": true,
                    "query":  "MATCH  (m:Computer)  WHERE  m.unconstraineddelegation=true  MATCH  (n)  MATCH
p=shortestPath((n)-[r:{}*1..]->(m)) WHERE NOT m=n RETURN p"
                }
            ]
        },
        {
            "name": "Shortest Paths from Kerberoastable Users",
            "queryList": [
                {
                    "final": false,
                    "title": "Select a domain...",
                    "query":
                        "MATCH (n:Domain) RETURN n.name ORDER BY n.name DESC"
                },
                {
                    "final": false,
                    "title": "Select a user",
                    "query":
                        "MATCH (n:User) WHERE n.domain={result} AND n.hasspn=true RETURN n.name, n.pwdlastset
ORDER BY n.pwdlastset ASC"
                },
                {
                    "final": true,
                    "query":
                        "MATCH n=shortestPath((a:User {name:{result}})-[r:{}*1..]->(b:Computer)) RETURN n",
                    "startNode": "{}",
                    "allowCollapse": true
                }
            ]
        },
        {
            "name": "Shortest Paths to Domain Admins from Kerberoastable Users",
            "queryList": [
                {
                    "final": false,
                    "title": "Select a Domain Admin group...",
                    "query":
                        "MATCH (n:Group) WHERE n.objectsid =~ {name} RETURN n.name ORDER BY n.name DESC",
                    "props": {
                        "name": "(?i)S-1-5-.*-512"
                    }
                },
                {
                    "final": true,
                    "query":
                        "MATCH (n:User),(m:Group {name:{result}}),p=shortestPath((n)-[r:{}*1..]->(m)) WHERE
n.hasspn=true RETURN p",
                    "allowCollapse": true,
                    "endNode": "{}"
                }
            ]
        },
        {
            "name": "Shortest Path from Owned Principals",
            "queryList": [
                {
                    "final": false,
                    "title": "Select a domain...",
                    "query":
                        "MATCH (n:Domain) RETURN n.name ORDER BY n.name DESC"
                },
                {
                    "final": false,
                    "title": "Select a user",
                    "query":
                        "MATCH (n) WHERE n.domain={result} AND n.owned=true RETURN n.name, n.PwdLastSet ORDER
BY n.PwdLastSet ASC"
                },
                {
                    "final": true,
                    "query":
                        "MATCH n=shortestPath((a {name:{result}})-[r:{}*1..]->(b:Computer)) RETURN n",
                    "startNode": "{}",
                    "allowCollapse": true
                }
            ]
        },{
            "name": "Shortest Paths to Domain Admins from Owned Principals",
            "queryList": [
                {
                    "final": false,
                    "title": "Select a Domain Admin group...",
                    "query":
                        "MATCH (n:Group) WHERE n.objectsid =~ {name} RETURN n.name ORDER BY n.name DESC",
                    "props": {
                        "name": "(?i)S-1-5-.*-512"
                    }
                },
                {
                    "final": true,
                    "query":
                        "MATCH   (n),(m:Group   {name:{result}}),p=shortestPath((n)-[r:{}*1..]->(m))   WHERE
n.owned=true RETURN p",
                    "allowCollapse": true,
                    "endNode": "{}"
                }
            ]
        },
        {
            "name": "Shortest Paths to High Value Targets",
            "queryList": [
                {
                    "final": false,
                    "title": "Select a Domain",
                    "query": "MATCH (n:Domain) RETURN n.name ORDER BY n.name DESC"
                },
                {
                    "final": true,
                    "query":
                        "MATCH   (n),(m),p=shortestPath((n)-[r:{}*1..]->(m))   WHERE   m.domain={result}   AND
m.highvalue=true AND NOT m = n RETURN p",
                    "allowCollapse": true,
                    "endNode": "{}"
                }
            ]
        }
    ]
}
```

Ok, so I know the format was a pain, but this is the format you will have to use if you want to add your own custom queries under the query tab of the UI.

E.3.2 – Custom Queries

To do so, click on the Query Tab, scroll down, and click on the pen icon.


Custom Queries

This will open a **customquery.json** document located in

C:\Users\<username>\AppData\Roaming\bloodhound\customqueries

Add your own awesome Cypher in there.

Don’t forget to save the document, and click on the refresh icon next to the pen.

Voila, your own custom queries are now available any time you like.

[If you do not see the new query name when you refresh, check your json syntax]

This is an example syntax for the “Owned” to “HighValue”.

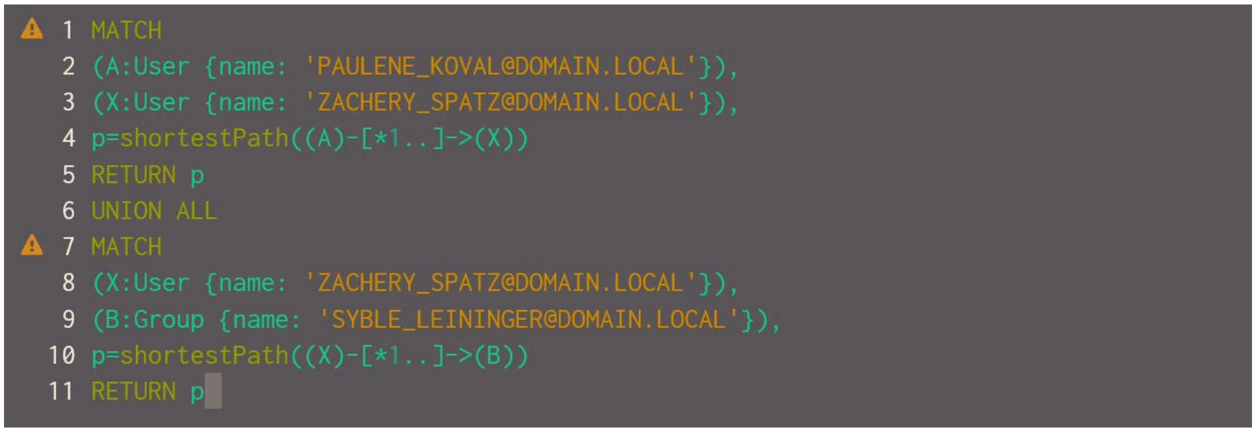
```
        {
            "queries": [
                {
                    "name": "Shortest Path Owned Users to HighValue Groups",
                    "queryList": [
                        {
                            "final": true,
                            "query":
                                "MATCH  p=shortestPath((O:User  {owned:  True})-[*1..]->(H:Group  {highvalue:  True}))
        RETURN p",
                            "props": {},
                            "allowCollapse": true
                        }
                    ]
                }
            ]
        }
```

If you do come up with the ultimate Cypher query, please share it on the BlooHound Slack. There is of course a #Cypher_Query Channel for that. Great place for some Cypher help if you need...

Union / Union All

Multiple queries can be combined into a single output/graph using **UNION** or **UNION ALL**.

In this Example a **Path from A to B via X**



```
MATCH
(A:User {name: 'PAULENE_KOVAL@DOMAIN.LOCAL'}),
(X:User {name: 'ZACHERY_SPATZ@DOMAIN.LOCAL'}),
p=shortestPath((A)-[*1..]->(X))
RETURN p
UNION ALL
MATCH
(X:User {name: 'ZACHERY_SPATZ@DOMAIN.LOCAL'}),
(B:Group {name: 'SYBLE_LEININGER@DOMAIN.LOCAL'}),
p=shortestPath((X)-[*1..]->(B))
RETURN p
```

Cheapest Path

Another cool one is the “Cheapest Path” query. The idea behind is quite simple. The query a bit less, but I’m sure you will get it:

Since a “MemberOf” Edge can be traversed by an attacker at no cost (no attack needed) we want to give it a weight of 0. And because Groups can be nested, a longer path with nesting might actually “cost” less than a shorter one with less “MemberOf” Edges.

To calculate the cheapest Path, we simply **count the number of edges without the MemberOf ones**.

```
1 MATCH (S:User {name: 'AZALEE_CASALE@DOMAIN.LOCAL'}),
2 (T:Computer {name: 'SRV_10.DOMAIN.LOCAL'}),
3 p=((S)-[r*1..4]->(T))
4 WITH p,
5 LENGTH(FILTER(x IN EXTRACT(r in RELATIONSHIPS(p)|TYPE(r)) WHERE x <>'MemberOf')) as Cost
6 RETURN p
7 ORDER BY Cost
8 LIMIT 1
```

The complicated bit happens on line 5. Use the Cypher Ref Card for quick explanation on these **functions()**.

E.2 – Database Manipulation

Until now we have been querying data from the database. But what if I told you can also manipulate that database with Cypher…. Awesome!

For this will use the **SET**, **REMOVE** & **DETACH DELETE** instructions.

You might be asking:”why would I want to manipulate the data?”.

That’s a good question. Here are a few scenarios where you might need it:

- Marking nodes as Owned/HighValue at scale…
- Deleting all sessions in one go for a specific computer or more…
- Removing “out-of-scope” nodes from the graph…
- Adding non-windows systems to the graph… (linux jumphosts etc…)
- Adding your own custom properties to nodes during an engagement…
- Simulating changes to the AD infrastructure…
- Testing environment hardening hypothesis…
- Adding your attack infrastructure to the BloodHound graph… Why not?

Anything you can think of really, BloodHound is just a database, you can do what you want with it.

Be creative, share your ideas on slack… Hack the planet!

E.2.1 – Creating/Deleting Nodes

```
$ MERGE (n:User {name: 'bob'})
```

Creates a User named "bob" <u>if it doesn't already exist</u>

```
$ MATCH (n:User {name: 'bob'}) DETACH DELETE n
```

Delete that User (and connected Edges of course...)

E.2.2 – Adding/Updating/Removing Node property

```
$ MATCH (n) WHERE n.name='bob' SET n.age=23
```

```
$ MATCH (n) WHERE n.name='bob' SET n.age=27, n.hair='black', n.sport='Chess-Boxing'
```

Both create missing properties and overwrite existing property values

```
$ MATCH (n) WHERE n.name='Bob' REMOVE n.sport
```

```
$ MATCH (U:User) WHERE EXISTS(U.age) REMOVE U.age
```

```
$ MATCH (U:User) WHERE EXISTS(U.hair) REMOVE U.age, U.hair RETURN U
```

Remove property from node (Single Node / multiple Nodes / multiple props)

E.2.3 – Creating/Removing Edges

```
MATCH (A:User {name: 'alice'})
MATCH (B:User {name: 'bob'})
CREATE (A)-[r:Loves]->(B)
```

```
MATCH (A:User {name: 'alice'})
MATCH (B:User {name: 'bob'})
CREATE (A)<-[r:Loves]-(B)
```

Create Edges between Nodes.

/!\ Reminder: Edges are directional.

```
$ MATCH (n:User {name: 'alice'})-[r:Loves]->(m:User {name: 'bob'}) DELETE r
```

Remove Edge between Nodes

/!\ not specifying any Edge type will remove all Edges between specified Nodes

/!\ Love doesn’t always last forever…

E.2.4 – Creating Nodes with Properties & Edges

```
$ MERGE (A:User {name:bob})-[r:IsBrother]->(B:User {name:'Paul'})
```

```
$ MERGE (A:User {name:'Jack', age:14, hair:'black'})-[r:IsBrother]->(B:User
  {name:'Jimmy'})
```

/!\ Use these syntaxes only if Nodes don't already exist. Otherwise use MERGE or MERGE/SET for each block separately as shown below

Recommended syntax:

```
MERGE (A:User {name:'bob'})
MERGE (B:User {name: 'Paul'})
MERGE (A)-[r:IsBrother]->(B)
```

```
MERGE (X:User {name:'Jack'})
SET X.age=14, X.hair='black'
MERGE (Y:User {name:'Jimmy'})
SET Y.age=21, Y.hair='black'
MERGE (X)-[r:IsBrother]->(Y)
```

E.2.5 – Nuke the DB

```
$ MATCH (x) DETACH DELETE x
```

/!\ Simple and efficient. Try it once...

E.3 – AD Metrics

Each environment is unique, and finding the right metric to measure an AD's security posture is not an easy task. The idea with the following query is to be able to extract data from BloodHound with a single query and output numbers that can mean something to management and non-IT related folks. Something like:

"23% of our ***employees*** *can gain* ***access to Group X****...*

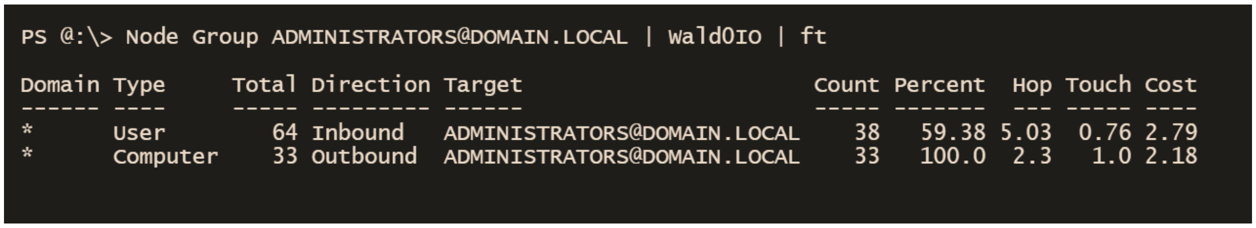
From there, they can get ***access to 77% of the Computers*** *on the network."*

Wald0 shared his Cypher knowledge to help me build it, and he is the one who came up with the idea, so I called it the **Wald0Index**. The query looks complicated, but it's actually not that bad. And because I'm not that bad either, this one can be copy-pasted...

```
// Percent User to target Group + Distance/Cost/ComputerTouched
MATCH (tx:User),
p = shortestPath((x:User)-[r*1..]->(g:Group {name:'ADMINISTRATORS@DOMAIN.LOCAL'}))
WITH g.name as G,
COUNT(DISTINCT(tx)) as TX,
COUNT(DISTINCT(x)) as X, ROUND(100*AVG(LENGTH(RELATIONSHIPS(p))))/100 as H,
ROUND(100*AVG(LENGTH(FILTER(z IN EXTRACT(r IN RELATIONSHIPS(p)|TYPE(r)) WHERE
z<>'MemberOf'))))/100 AS C,
ROUND(100*AVG(LENGTH(FILTER(y IN EXTRACT(n IN NODES(p)|LABELS(n)[0]) WHERE
y='Computer'))))/100 AS T
 WITH G,TX,X,H,C,T,
 ROUND(100*(100.0*X/TX))/100 as P
 RETURN {
     TotalCount: TX,
     PathCount:   X,
     Percent:     P,
     HopAvg:      H,
     CostAvg:     C,
     TouchAvg:    T
     } AS Wald0IndexIO
```

```
// Percent Computer from Target Group + Distance/Cost/ComputerTouched
MATCH
(tx:Computer),
p = shortestPath((g:Group {name:'ADMINISTRATORS@DOMAIN.LOCAL'})-[r*1..]->(x:Computer))
WITH g.name as G,
COUNT(DISTINCT(tx)) as TX,
COUNT(DISTINCT(x)) as X,
ROUND(100*AVG(LENGTH(RELATIONSHIPS(p))))/100 as H,
ROUND(100*AVG(LENGTH(FILTER(z IN EXTRACT(r IN RELATIONSHIPS(p)|TYPE(r)) WHERE
z<>'MemberOf'))))/100 AS C,
ROUND(100*AVG(LENGTH(FILTER(y IN EXTRACT(n IN NODES(p)|LABELS(n)[0]) WHERE
y='Computer'))))/100 AS T
WITH G,TX,X,H,C,T,
ROUND(100*(100.0*X/TX))/100 as P
RETURN {
    TotalCount: TX,
    PathCount:   X,
    Percent:     P,
    HopAvg:      H,
    CostAvg:     C,
    TouchAvg:    T
    } AS Wald0IndexIO
```

And the output could look like this… [taken from CypherDog]



```
PS @:\> Node Group ADMINISTRATORS@DOMAIN.LOCAL | Wald0IO | ft

Domain Type     Total Direction Target                      Count Percent  Hop Touch Cost
------ ----     ----- --------- ------                      ----- -------  --- ----- ----
*      User        64 Inbound   ADMINISTRATORS@DOMAIN.LOCAL    38   59.38 5.03  0.76 2.79
*      Computer    33 Outbound  ADMINISTRATORS@DOMAIN.LOCAL    33   100.0  2.3   1.0 2.18
```

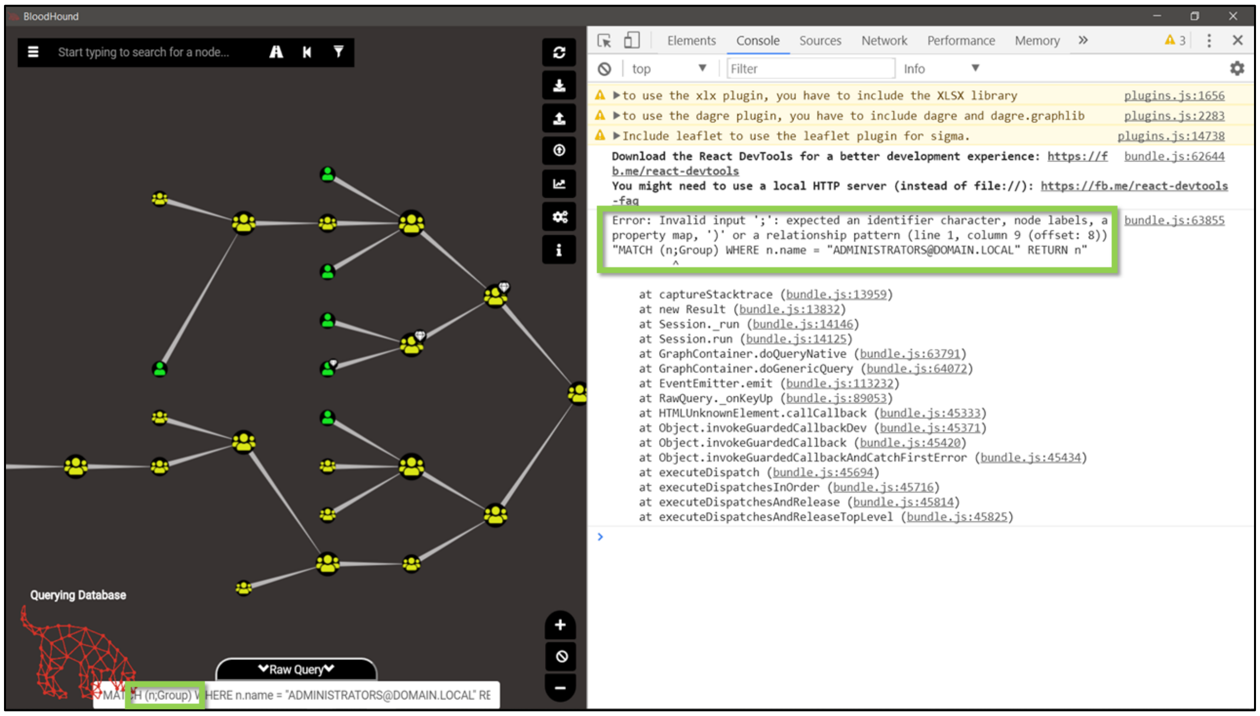
Now this is as complicated as it's going to get for this guide. Take your time. Let it infuse…

Use the Cypher Ref Card…

E.4 – Debugging Queries

Sometimes, you just fat-finger your Cypher… it happens. In that case, the cool BloodHound logo animation will keep walking forever. This is the moment you have to check your Cypher syntax.

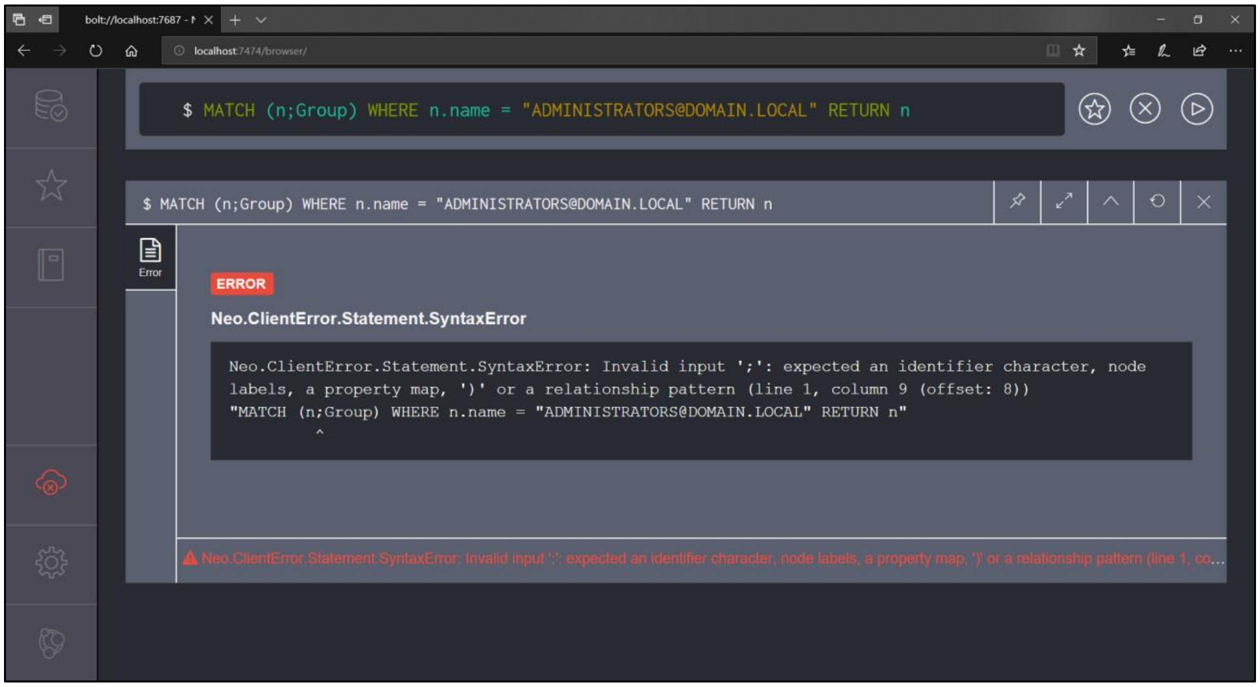
For a quick fix, you can tweak your query directly into the Raw Query input box. If you can't catch your typo there, you can hit **[CTRL+SHIFT+I]**. This will open the web dev tools, and you can find your error messages in the Console tab.


BloodHound
Start typing to search for a node...
Querying Database
Raw Query
MATCH (n;Group) WHERE n.name = "ADMINISTRATORS@DOMAIN.LOCAL" RE
Elements
Console
Sources
Network
Performance
Memory
top
Filter
Info

```
▶to use the xlx plugin, you have to include the XLSX library               plugins.js:1656
▶to use the dagre plugin, you have to include dagre and dagre.graphlib     plugins.js:2283
▶Include leaflet to use the leaflet plugin for sigma.                      plugins.js:14738
Download the React DevTools for a better development experience: https://f bundle.js:62644
b.me/react-devtools
You might need to use a local HTTP server (instead of file://): https://fb.me/react-devtools
-faq
Error: Invalid input ';': expected an identifier character, node labels, a bundle.js:63855
property map, ')' or a relationship pattern (line 1, column 9 (offset: 8))
"MATCH (n;Group) WHERE n.name = "ADMINISTRATORS@DOMAIN.LOCAL" RETURN n"
         ^

    at captureStacktrace (bundle.js:13959)
    at new Result (bundle.js:13832)
    at Session._run (bundle.js:14146)
    at Session.run (bundle.js:14125)
    at GraphContainer.doQueryNative (bundle.js:63791)
    at GraphContainer.doGenericQuery (bundle.js:64072)
    at EventEmitter.emit (bundle.js:113232)
    at RawQuery._onKeyUp (bundle.js:89053)
    at HTMLUnknownElement.callCallback (bundle.js:45333)
    at Object.invokeGuardedCallbackDev (bundle.js:45371)
    at Object.invokeGuardedCallback (bundle.js:45420)
    at Object.invokeGuardedCallbackAndCatchFirstError (bundle.js:45434)
    at executeDispatch (bundle.js:45694)
    at executeDispatchesInOrder (bundle.js:45716)
    at executeDispatchesAndRelease (bundle.js:45814)
    at executeDispatchesAndReleaseTopLevel (bundle.js:45825)
```

A good option when working on complex queries, is to debug them in the neo4j browser. There, you will get tips while writing them (warning icons), a bigger font size with syntax highlighting, and you will get error messages when running them. It's way easier than in Bloodhound itself.



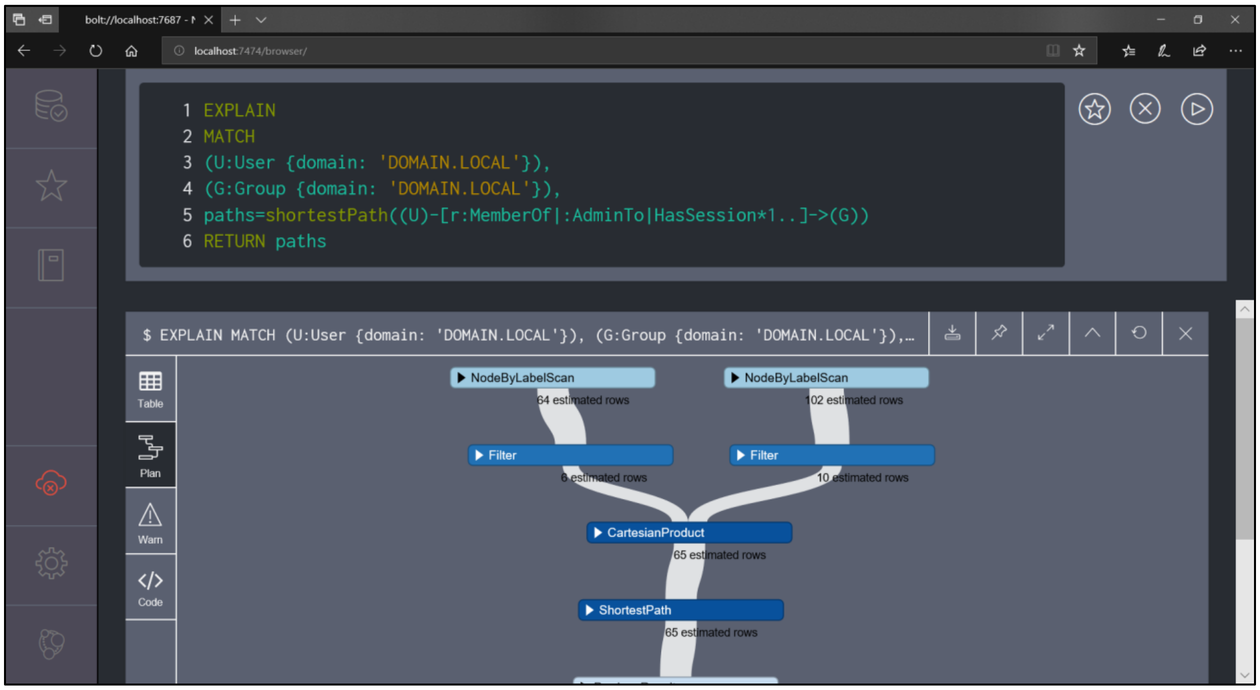
```
$ MATCH (n;Group) WHERE n.name = "ADMINISTRATORS@DOMAIN.LOCAL" RETURN n
```

ERROR

Neo.ClientError.Statement.SyntaxError

```
Neo.ClientError.Statement.SyntaxError: Invalid input ';': expected an identifier character, node
labels, a property map, ')' or a relationship pattern (line 1, column 9 (offset: 8))
"MATCH (n;Group) WHERE n.name = "ADMINISTRATORS@DOMAIN.LOCAL" RETURN n"
         ^
```

Another cool thing in the neo4j browser, is to add **EXPLAIN** or **PROFILE** in front of your query. You will get loads of info on what is happening under the neo4j hood.



```
EXPLAIN
MATCH
(U:User {domain: 'DOMAIN.LOCAL'}),
(G:Group {domain: 'DOMAIN.LOCAL'}),
paths=shortestPath((U)-[r:MemberOf|:AdminTo|HasSession*1..]->(G))
RETURN paths
```

Check it out, it's really quite cool… (See Neo4j Query Tuning for more info)

A Google Search for help|examples on general Cypher will often get you the info you need to fix your BloodHound queries, but if that's not enough, you can always ask on the BloodHound Slack.

@CptJesus never sleeps…

"Sure, you can ask me…

But did you Google first?"

@CptJesus

F - REST API & Other Cool Stuffs

F.1 – REST API

Like if all this wasn't awesome enough, there is a hidden treasure waiting for you: The **Neo4j REST API**. This will allow you to query the database from the command line of your choice and take BloodHound to another level. Only sky is the limit…

F.1.1 – API setup

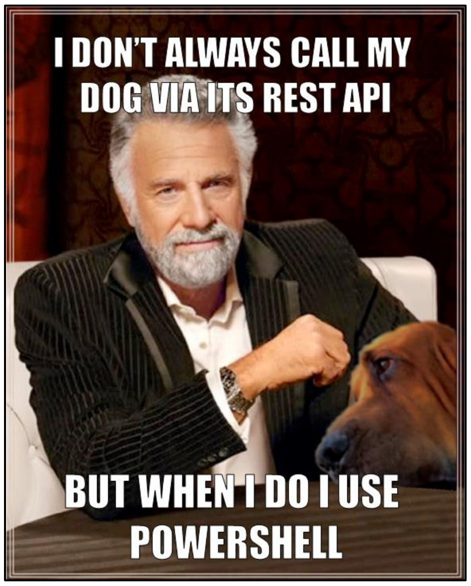
The REST API is a build-in neo4j feature, to enable it, all you have to do is uncomment this line in the neo4j config located at **[Somewhere]\neo4j\neo4j-community-X.X.X\conf\neo4j.conf** like in example shown below.

```
# Whether requests to Neo4j are authenticated.
# To disable authentication, uncomment this line
dbms.security.auth_enabled=false
```

More info can be found here. And if this is not enough, you can also have a look here.

You should then be able to query the database via API call to **localhost** on port **7474**.

This can be done in the language of your choice. See below for an example in PowerShell.

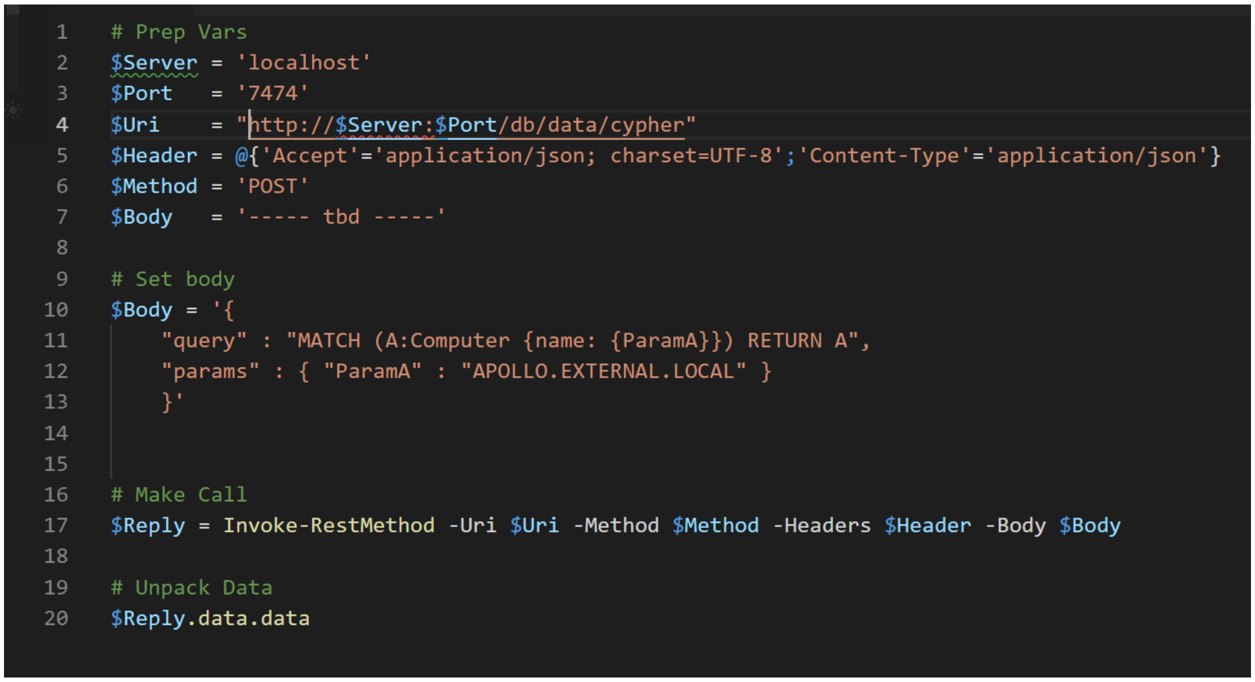

I DON'T ALWAYS CALL MY DOG VIA ITS REST API
BUT WHEN I DO I USE POWERSHELL

If you prefer Bash, have a look here.

If your prefer Python, @_dirkjan is your man. Catch him on Slack…

F.1.2 – API Call - PowerShell Example

Here is what a Call would look like in PowerShell:



```
# Prep Vars
$Server = 'localhost'
$Port   = '7474'
$Uri    = "http://$Server:$Port/db/data/cypher"
$Header = @{'Accept'='application/json; charset=UTF-8';'Content-Type'='application/json'}
$Method = 'POST'
$Body   = '----- tbd -----'

# Set body
$Body = '{
    "query" : "MATCH (A:Computer {name: {ParamA}}) RETURN A",
    "params" : { "ParamA" : "APOLLO.EXTERNAL.LOCAL" }
    }'


# Make Call
$Reply = Invoke-RestMethod -Uri $Uri -Method $Method -Headers $Header -Body $Body

# Unpack Data
$Reply.data.data
```

All you need to change for the next call is the body. You could also build any kind of wrapper around that.

Note: You can use **params**, as shown in the example, to avoid issues with quotes.

You could also have only the $Body with your single-quoted name value instead of the {ParamA} in the Cypher query. Using params makes it easier to read when dealing with complex queries.

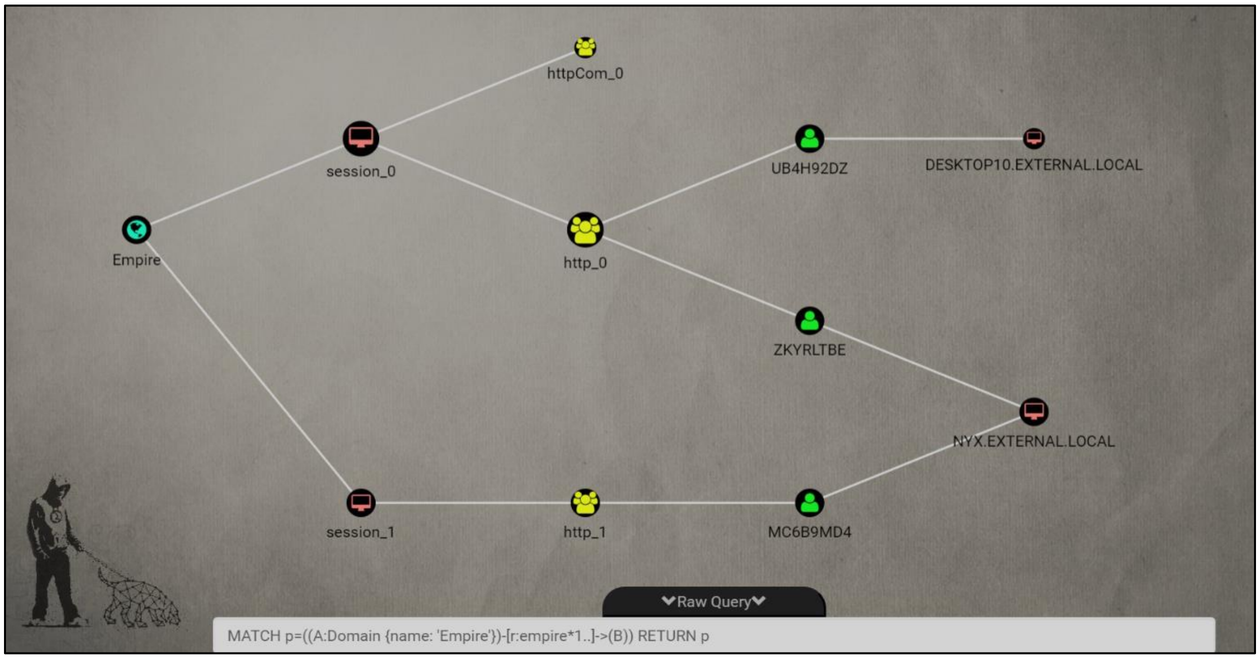
F.2 – Tweaks & Hacks

By now, you are probably having awesome ideas on how you could hack something together with BloodHound and *[InsertToolOfYourChoice]*. Good. BloodHound is just a database… Feed it.

And now you know also how to write a wrapper in the language of your choice… Nice. So we'll be beta testing your next tool on the BloodHound slack in the near future… Sweet. Looking forward to it.

Below an idea of what it would look like to add your Empire infrastructure in the Bloodhound database…

From left to right: **You,** working on 2 **Empire Servers** with **Listeners** listening to **Agents** running on **Target Computers** (regular Bloodhound Nodes). You could now request a path from you to any Node, and return a path containing your empire infrastructure. And from there, automate all the things…

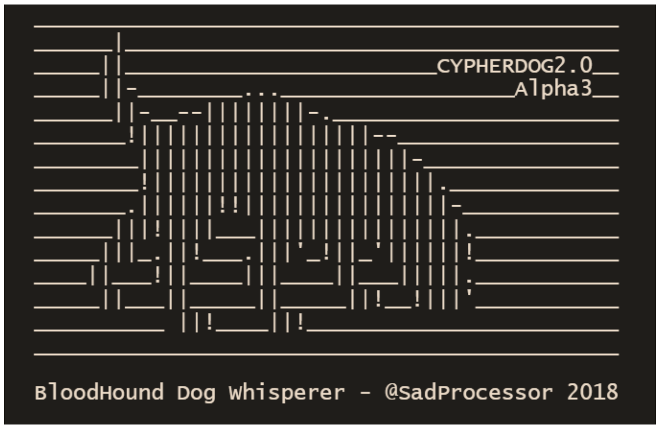


```
MATCH p=((A:Domain {name: 'Empire'})-[r:empire*1..]->(B)) RETURN p
```

Some really cool projects have been built around BloodHound and its REST API. Here are a few you might want to check out.

F.2.1 – CypherDog

A PowerShell BloodHound Wrapper by @SadProcessor (shameless plug…)

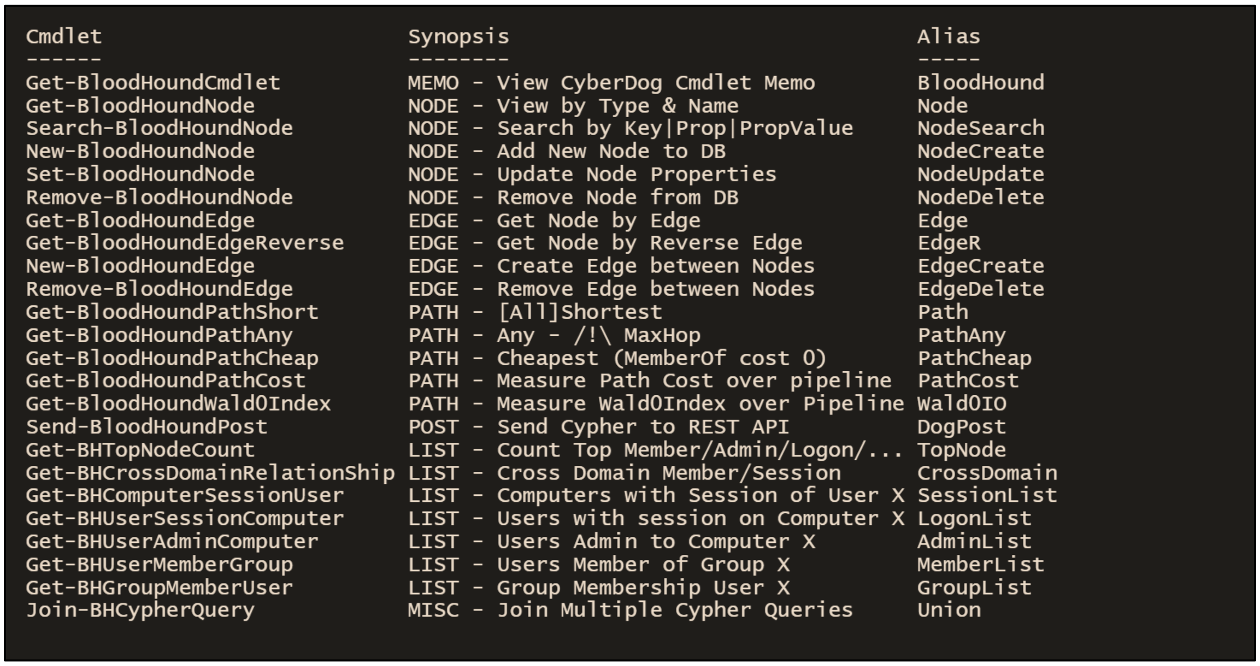


```
CYPHERDOG2.0
Alpha3

BloodHound Dog whisperer - @SadProcessor 2018
```

CypherDog is how I ended up digging into BloodHound & Cypher. I just wanted to practice some PowerShell, and it got a bit out of hand… So I made this **PowerShell** module with **23 Cmdlets** to script/**automate** about **anything in BloodHound**.

Still a work in progress. Not enough time… But for now, you can find the code here if you want to give it a go. Ping me on Slack if you have any questions…Below, a list of the Cmdlets in the Module:



```
Cmdlet                         Synopsis                                 Alias
------                         --------                                 -----
Get-BloodHoundCmdlet           MEMO - View CyberDog Cmdlet Memo         BloodHound
Get-BloodHoundNode             NODE - View by Type & Name               Node
Search-BloodHoundNode          NODE - Search by Key|Prop|PropValue      NodeSearch
New-BloodHoundNode             NODE - Add New Node to DB                NodeCreate
Set-BloodHoundNode             NODE - Update Node Properties            NodeUpdate
Remove-BloodHoundNode          NODE - Remove Node from DB               NodeDelete
Get-BloodHoundEdge             EDGE - Get Node by Edge                  Edge
Get-BloodHoundEdgeReverse      EDGE - Get Node by Reverse Edge          EdgeR
New-BloodHoundEdge             EDGE - Create Edge between Nodes         EdgeCreate
Remove-BloodHoundEdge          EDGE - Remove Edge between Nodes         EdgeDelete
Get-BloodHoundPathShort        PATH - [All]Shortest                     Path
Get-BloodHoundPathAny          PATH - Any - /!\ MaxHop                  PathAny
Get-BloodHoundPathCheap        PATH - Cheapest (MemberOf cost 0)        PathCheap
Get-BloodHoundPathCost         PATH - Measure Path Cost over pipeline   PathCost
Get-BloodHoundWald0Index       PATH - Measure Wald0Index over Pipeline  Wald0IO
Send-BloodHoundPost            POST - Send Cypher to REST API           DogPost
Get-BHTopNodeCount             LIST - Count Top Member/Admin/Logon/...  TopNode
Get-BHCrossDomainRelationShip  LIST - Cross Domain Member/Session       CrossDomain
Get-BHComputerSessionUser      LIST - Computers with Session of User X  SessionList
Get-BHUserSessionComputer      LIST - Users with session on Computer X  LogonList
Get-BHUserAdminComputer        LIST - Users Admin to Computer X         AdminList
Get-BHUserMemberGroup          LIST - Users Member of Group X           MemberList
Get-BHGroupMemberUser          LIST - Group Membership User X           GroupList
Join-BHCypherQuery             MISC - Join Multiple Cypher Queries      Union
```

If you are interested, check out this video from my DerbyCon presentation of this tool (& more).

F.2.2 – DeathStar

Another cool Project is **DeathStar** by @Byt3bl33d3r. With this tool, you can automate the shortest path to DA attack. Much controversy about automating the "Getting Admin" part of the job.
Good or Bad? Won't go into it today, but I do believe it can be used at least once, as an eye opener…
This being said, I also do know (hope) you deliver more than this in your pentest reports…

More Info here & here.

F.2.3 – AngryPuppy

If you like Cobalt Strike, you should check out **AngryPuppy** by @001SPARTaN & @VySecurity.

More Info here.

F.2.4 – GoFetch

Another cool project is **GoFetch** by @TaltheMaor & @TalBeerySec.

"The Industrialization of Lateral Movement…" … Brace yourselves…

More Info here & here.

F.2.5 – More Cool Hacks

This one most probably won't be maintained anymore, as most of the features @porterHau5 had implemented in his BloodHound fork have now been added to the latest version.
Still an great resource if you want to start tinkering with the Bloodhound source code.

More Info here & here.

G – Outro

Ok, so that's all I have... I hope you enjoyed it, and that you have learned stuff that will help you "make the world a safer place"... Red, Blue, Purple, or whatever color you are in the rainbow.

I would like to quickly thank **@harmj0y**, **@_wlad0** & **@CptJesus** for BloodHound itself, for their availability & support while I was digging into it and working on CypherDog; and more generally speaking, for moving the industry forward and for being a great source of inspiration for many of us working with Active Directory security.

I would also like to thank Enno [aka @Enno_Insinuator [aka my boss]] for trusting me on this, and for giving me the opportunity to deliver this BloodHound training and share this gathered knowledge with you and with the community.

I really hope you enjoyed it. Hit me on twitter if you have any questions or feedback. And if you see me in a security con or something, just come and say hi. I would love to share a beer and some AD horror stories with you...

Aaarrroooooooooooo!

HACK THE PLANET.

@SadProcessor

H – Appendix

Here is a **list of external BloodHound links** found in this document (and more…).

H.1 – BloodHound Crew

Twitter:	@harmj0y / @_Wald0 / @CptJesus
Slack:	https://bloodhoundhq.slack.com/
Slack Invite:	https://bloodhoundgang.herokuapp.com

H.2 – Bloodhound Code

GitHub:	https://github.com/BloodHoundAD/BloodHound
Wiki:	https://github.com/BloodHoundAD/BloodHound/wiki

H.3 – BloodHound Posts

Automated Derivative Admin Search	by @_Wald0
Introducing BloodHound	by @_Wald0
Intro to Cypher	by @CptJesus
The ACL Attack Path Update (v1.3)	by @_Wald0
Evolution of the BloodHound Ingestor	by @CptJesus
The Object Properties Update (v1.4)	by @CptJesus
SharpHound: Technical Details	by @CptJesus
SharpHound: Target Selection and API Usage	by @CptJesus
The Container Update (v1.5)	by @CptJesus
A Red Teamer's Guide to GPOs & OUs	by @_Wald0
BloodHound 2.0 (v2.0)	by @CptJesus

H.4 – BloodHound Videos

Six Degrees of Domain Admin	BSides LV 2016
Here Be Dragons...	DerbyCon 2017
An ACE Up the Sleeves	DEFCON 2017
Bloodhound: He Attac, But He Also Protec...	SecDSM 2018
How to Download and Install Neo4j	SpecterOps 2018
How does Session Collection Work	SpecterOps 2018

H.5 – Neo4j Cypher

Cypher Reference Card:	http://neo4j.com/docs/cypher-refcard/current/
Cypher Syntax Online Doc:	https://neo4j.com/docs/developer-manual/current/cypher/syntax/
Common Cypher Confusions:	https://neo4j.com/blog/common-confusions-cypher/

*

***