Deception in Depth - Hiding AD Users and Groups - Part 1

blog.spookysec.net/DnD-Hiding-Users-and-Groups

22 Oct 2023

Hello darkness, my old friend.

We're back after quite the long hiatus with another entry in the Deception in Depth series, since then I've changed roles from the lead on the deception project at $Employer to the Red Team (I've mentioned this in a few posts before, I think. Too many WIP posts to keep track of)! The change has been much needed for me - it's been a refreshing breath of fresh air. So, I've had a considerable amount of time to break from research on new deception methods, but I think I've come up with a moderately interesting one that may be highly effective.

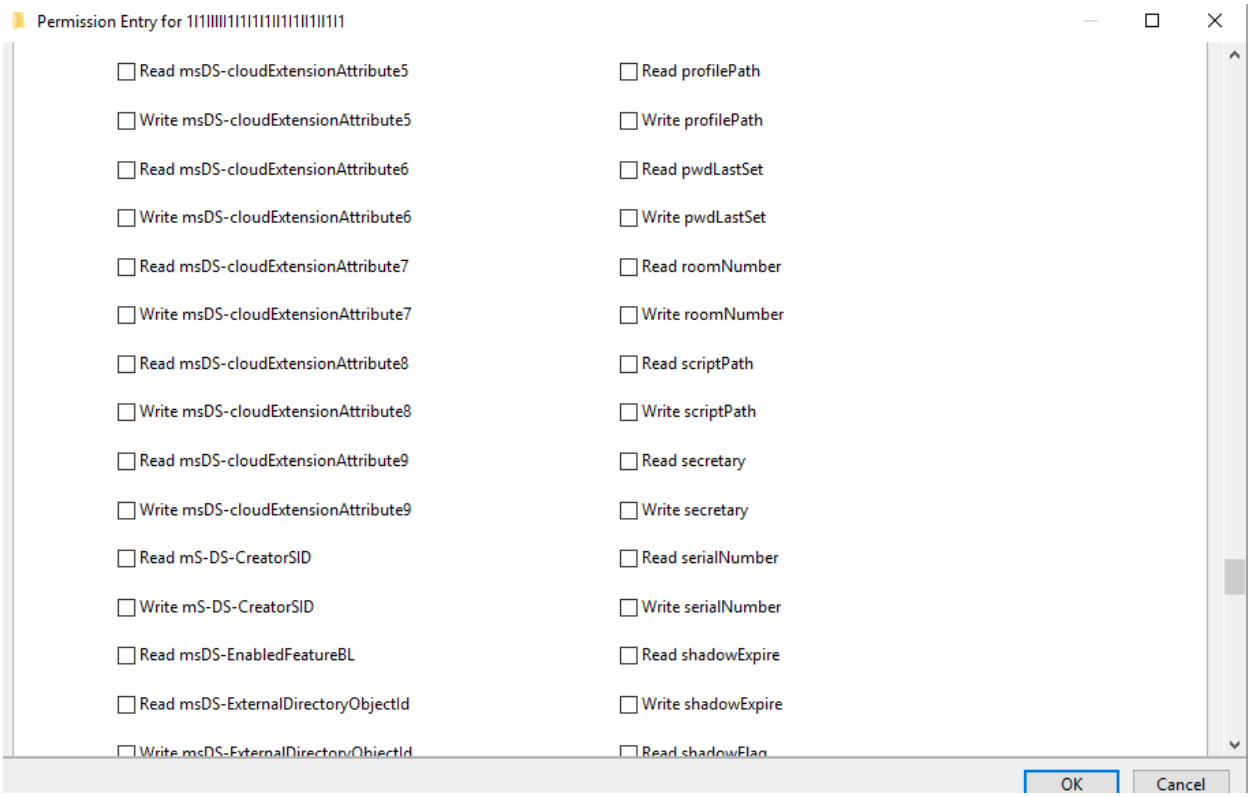
Recently, I was browsing Reddit and a user had posted on the ActiveDirectory subreddit that they got dinged on a recent audit for having "privileged accounts be too easy to discover". As we all know, Security through obscurity doesn't work. I can name my Domain Admin accounts whatever I want and an attacker can still use a tool like BloodHound, or manually enumerate them like so:

Oh no! Our super complex domain admin names have been found!!! What ever will we do! In reality, the question itself is actually a really great one - How can we (as defenders) make Domain Admins (and like groups) more difficult to enumerate and discover? Fortunately, there is a really easy to this challenge that not a lot of people talk about - but first, let's back it up a little bit.

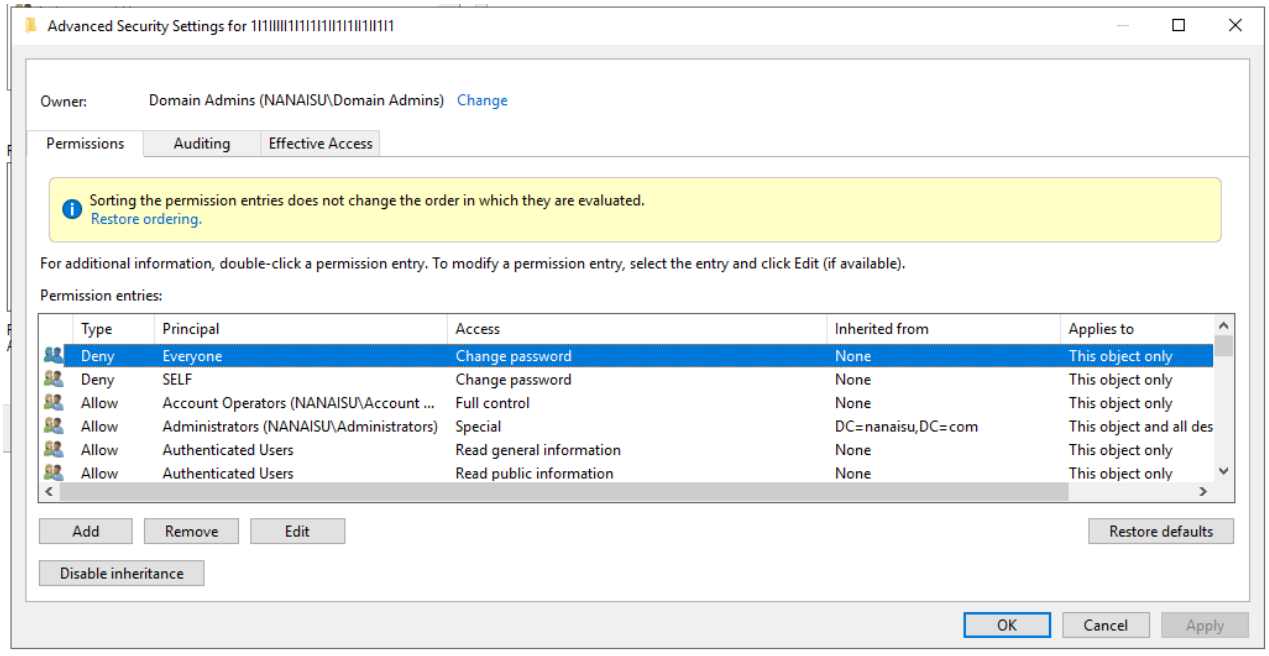
Active Directory Permissions and ACLs

Active Directory (by default) is very read permissive by default and gives you a **ton** of fine grain control over what properties of an account you may be able to enumerate. There's some seriously random stuff like jpegPhoto ipPhoneNumber, msTSConnectClientDrives,

and plenty more:



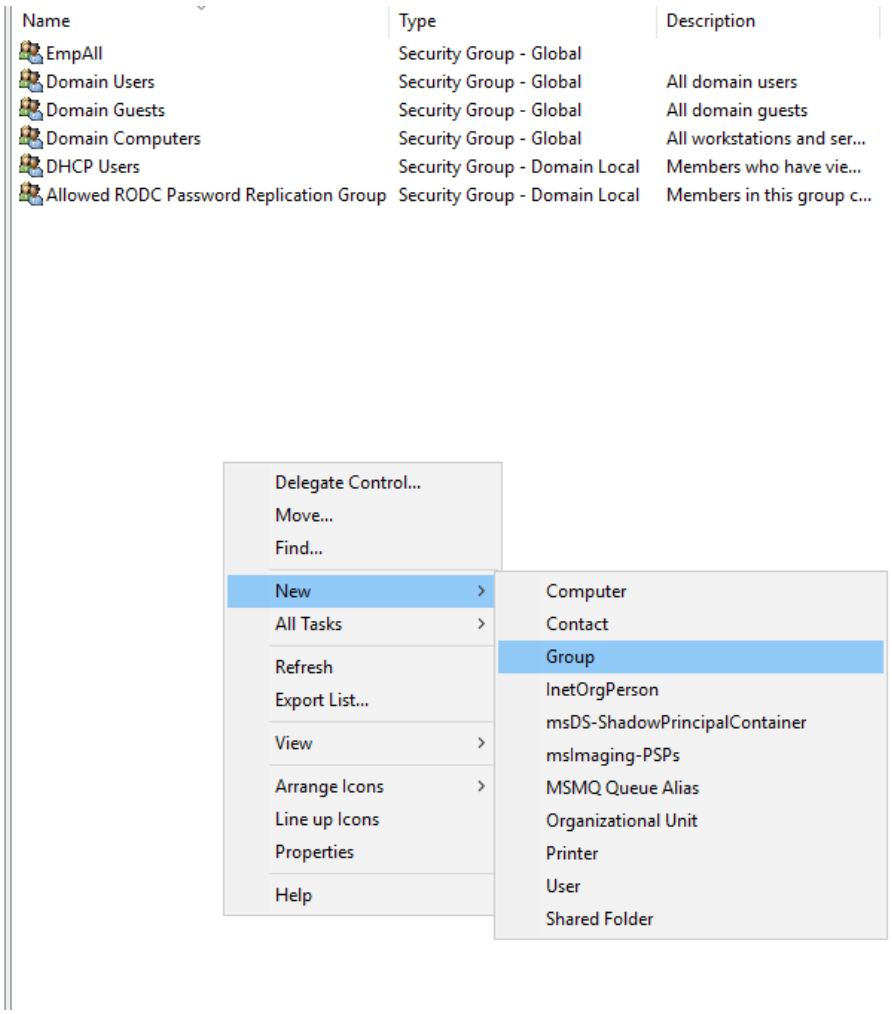
Like I said, AD is very read permissive, and gives you a lot of flexibility in that sense. For each thing you see here (these are called ACEs - Access Control Entries, collectively, they from an ACL - Access Control List) in this list, we can also deny that too! For example, everyone has Deny “Change Password” privileges over one of our DAs:



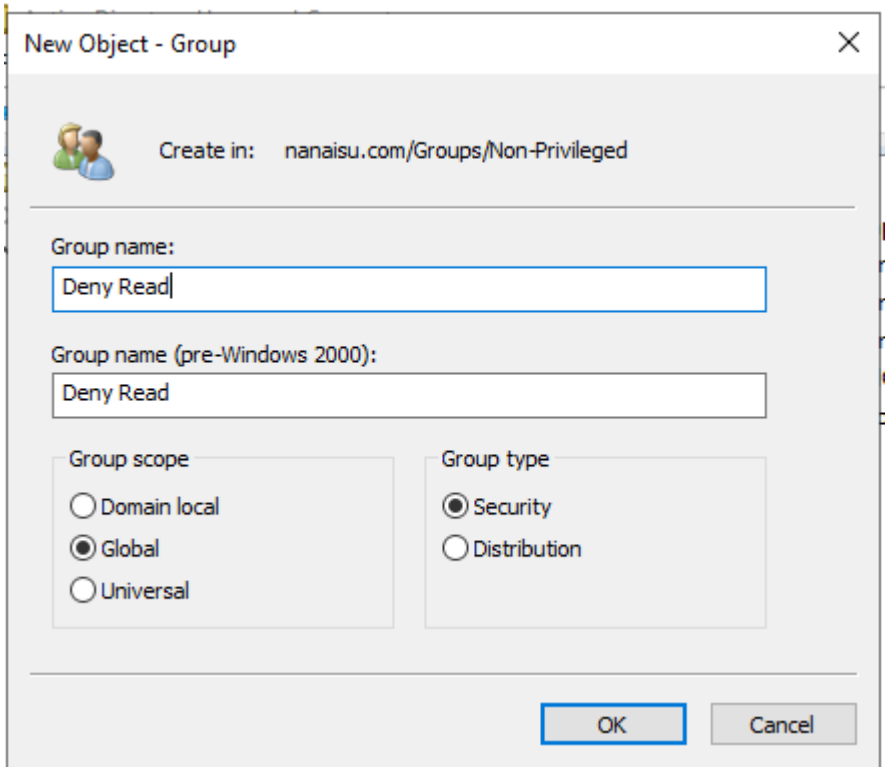
Piecing it Together

So, you may already be starting to pick up on this - You can also deny things (Users, for example) the ability to read properties of a given object. In short, you could theoretically create a new group to deny that is denied the ability to read Members of the Domain

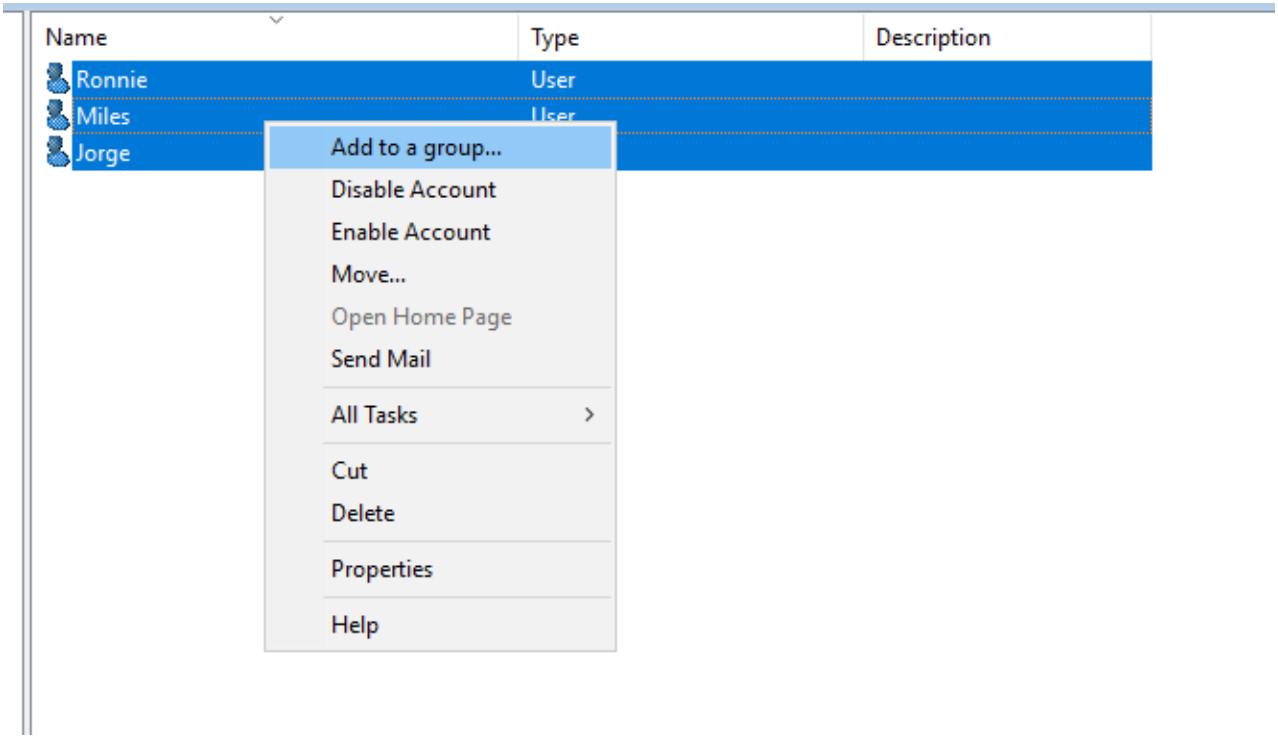
Admins group for example (**After published note: Privileged groups will not survive this change unless you modify the AdminSDHolder object. How to do this is shown later in this post.**). Let's take a look at how this would be accomplished. First, we're going to need to create a new group:



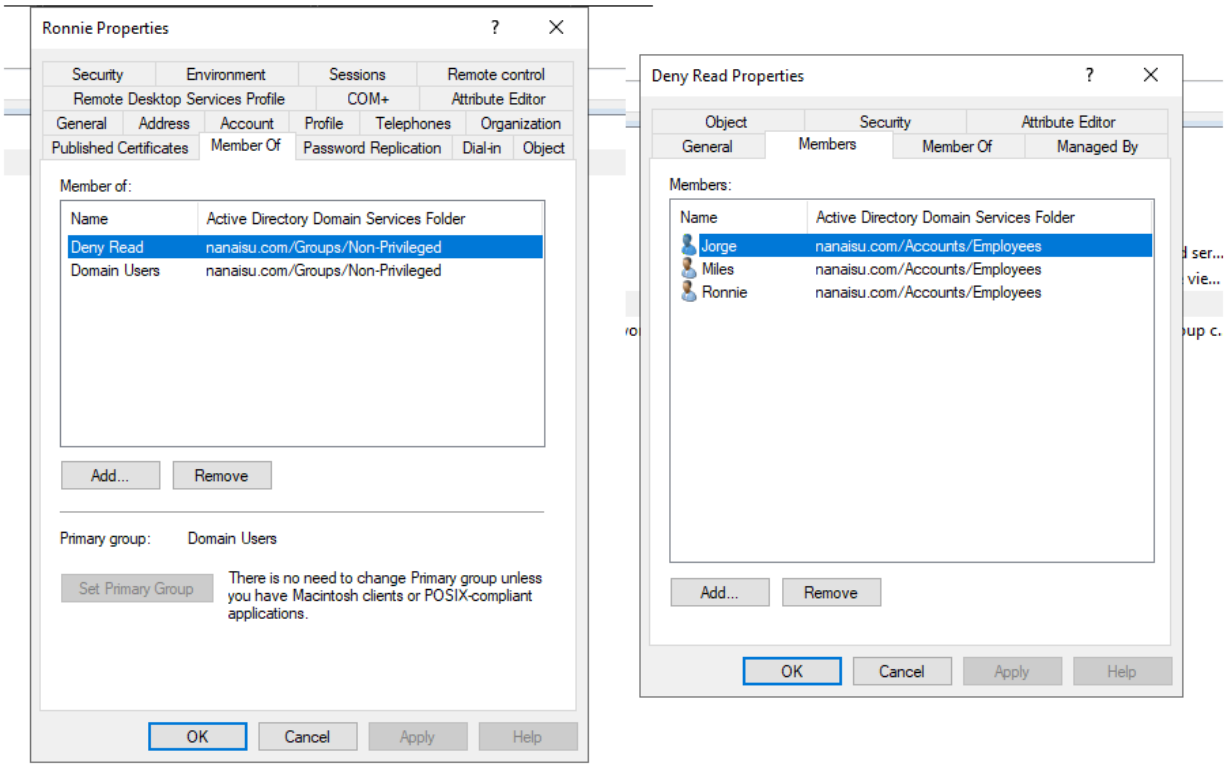
We'll call it "Deny read":



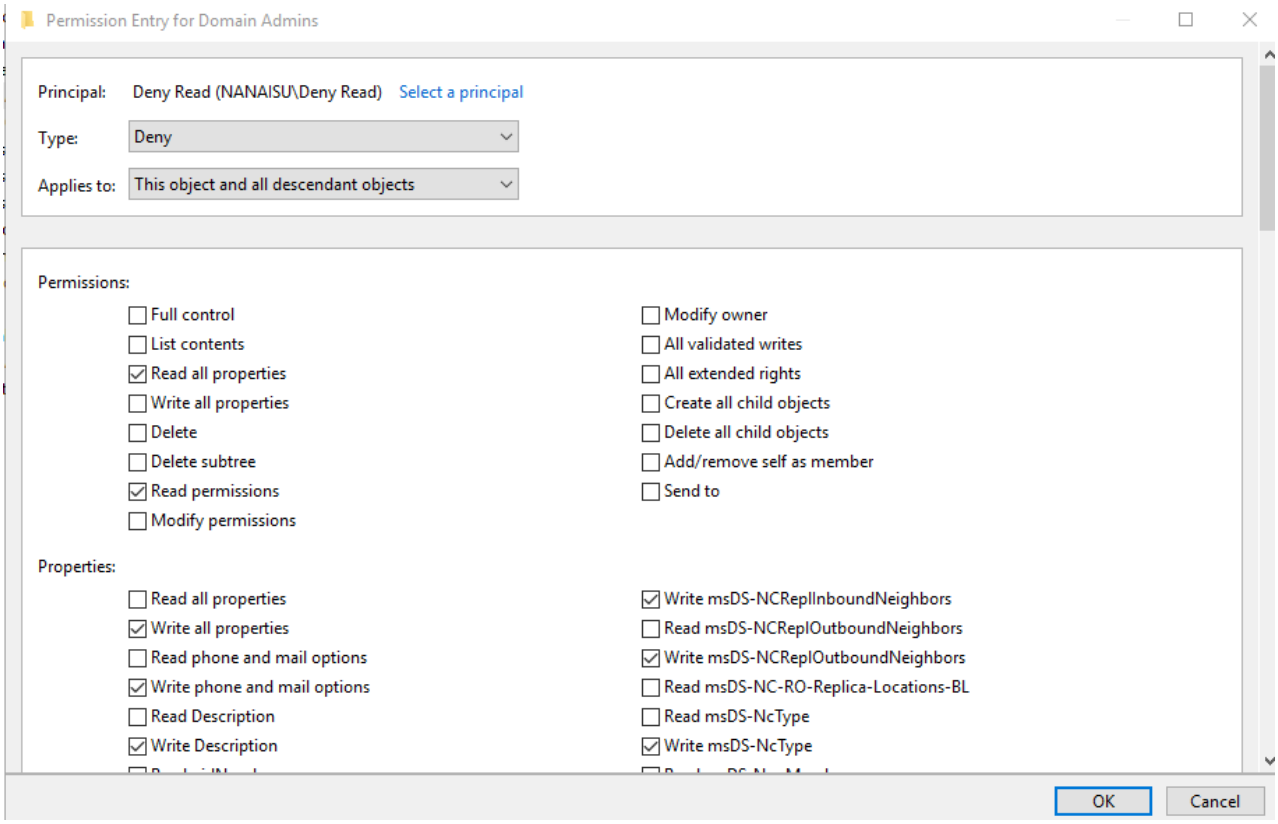
Let's go ahead and assign this group to all of our Employees. This can be done by (for example), navigating to our OU where all our employees are located at, selecting them all, and selecting "Add To Group":



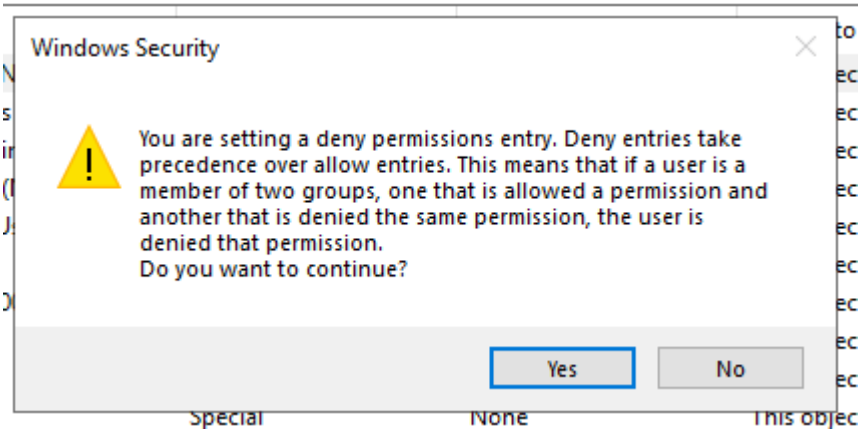
A new prompt will open up, put the group name in (Deny Read) and click "Done". The users should now be added to the "Deny Read" group. You can verify by right clicking on the User, selecting "Properties", then "Member Of", or going to the group, selecting Properties and select "Members"



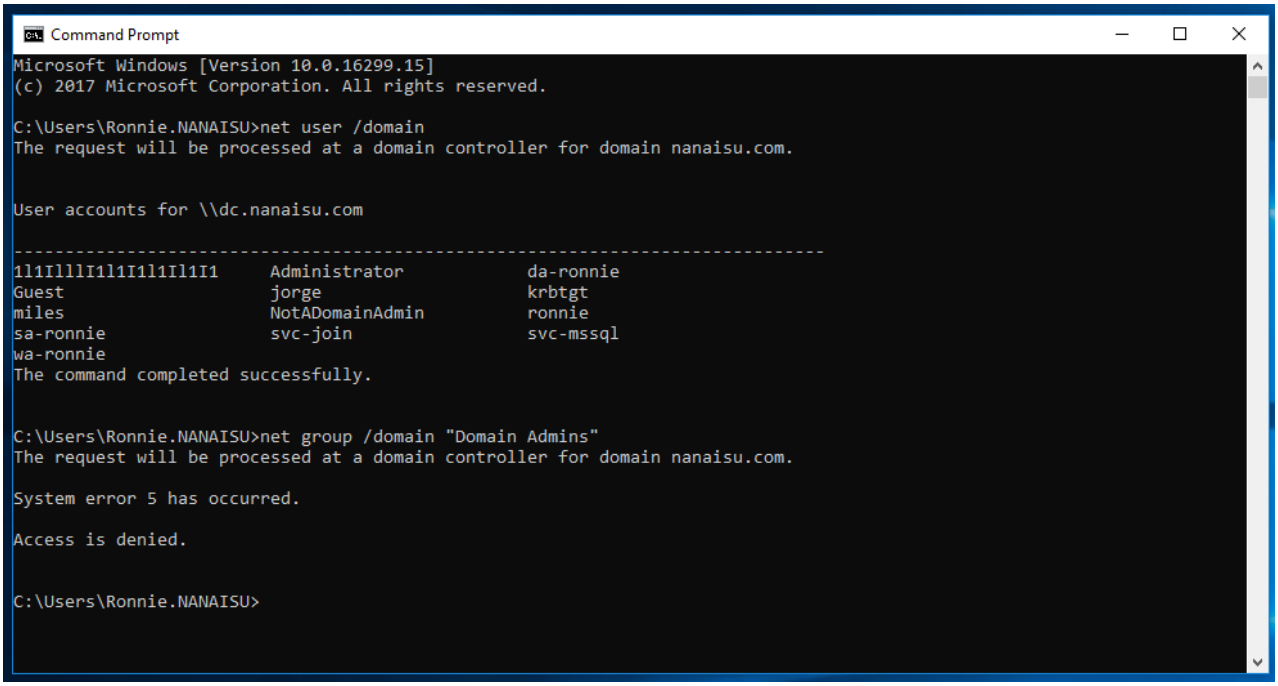
Now, we need to create a new ACE in the Domain Admins group that denies read privileges to members in this group. We can do this by finding our Domain Admins group, right clicking it, selecting “Properties”, then clicking the “Security” tab, Advanced, then “Add”. We’ll select “Type”, and change that to “Deny”. I unchecked everything else just to be safe and only kept “Read All Properties and Read Permissions” checked:



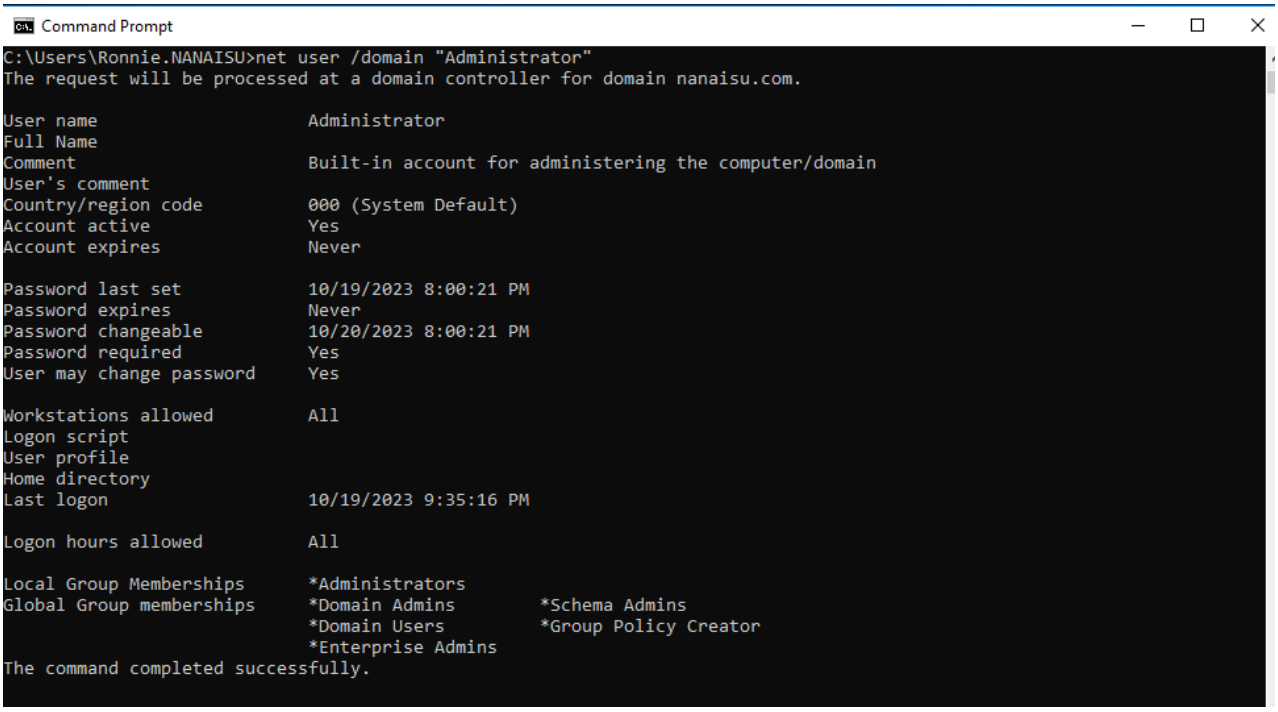
You’ll see a particularly scary prompt that says “hey, you’ve got an allow and a deny ACE that are conflicting. Deny precedes Allow, do you want to continue?”. Yes! Yes we do.



After clicking “Yes”, the ACE should get applied, and the ACL should be updated. We can now pivot over to our user and try to enumerate the Domain Admins group now:



We can see here that I am no longer allowed to enumerate members of the Domain Admins group, awesome! Now, the only issue here is that we can still enumerate users directly for group membership. This is a little bit tedious, but we can definitely take care of this too.



Dealing With Privileged Users

So, this one is a bit different due to Privileged Users being affected by something called “AdminSDHolder”, and the adminCount attributes in Active Directory. If you’re not familiar, Active Directory does some automagic cleanup to prevent misconfiguration and abuse of ACLs…

Disclaimer: I had a whole section written out here, but I found this fking amazing article from Tenable (Yes, the Nessus people) that describes a much easier way for us to pull this off without fighting AD:

1. Every 60 minutes, the SDProp process runs
2. The SDProp process **copies the ACL** from the adminSDHolder object, shown in Figure 1”
3. The **ACL from adminSDHolder is then pasted onto every user and group with an adminCount = 1**.

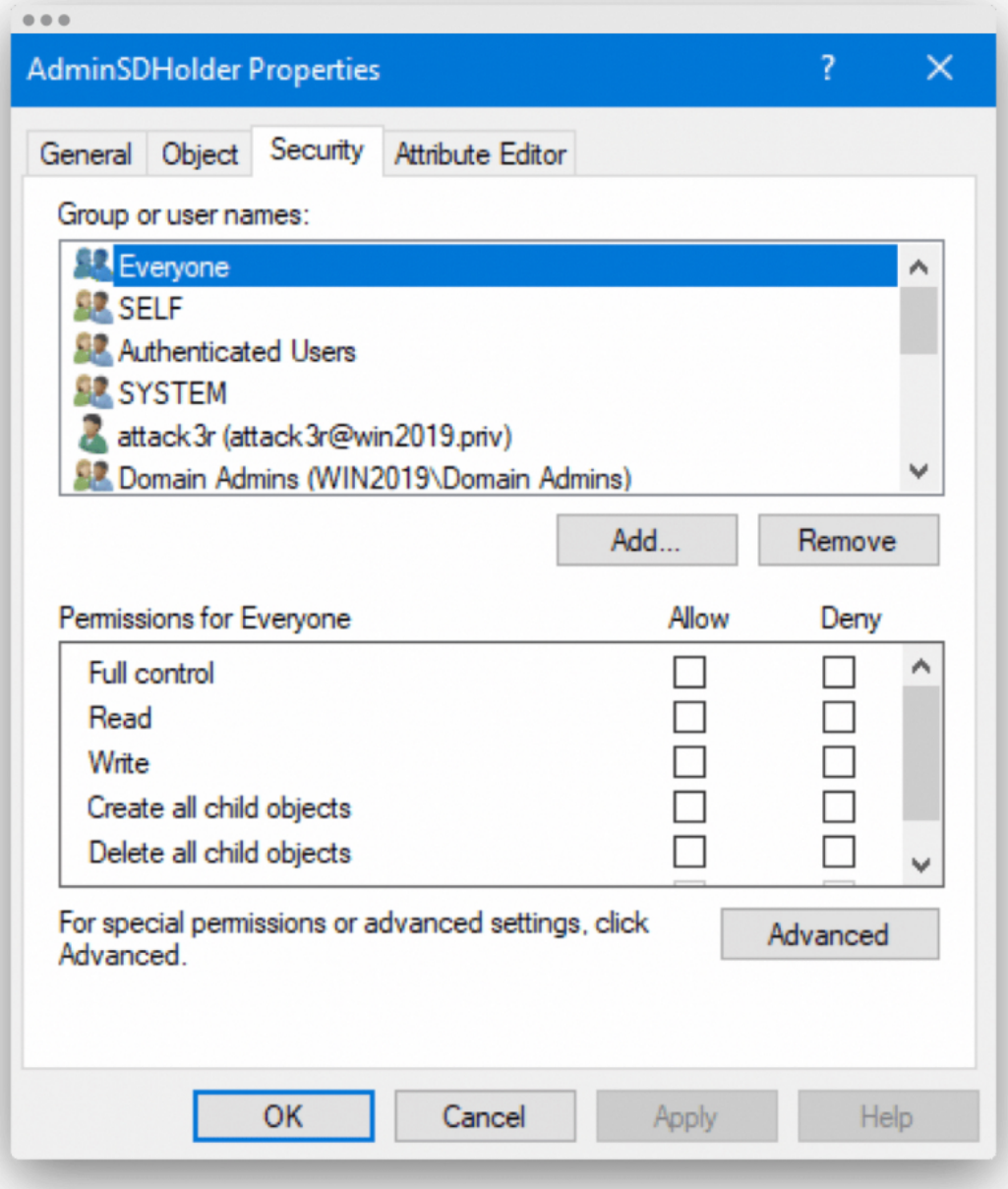
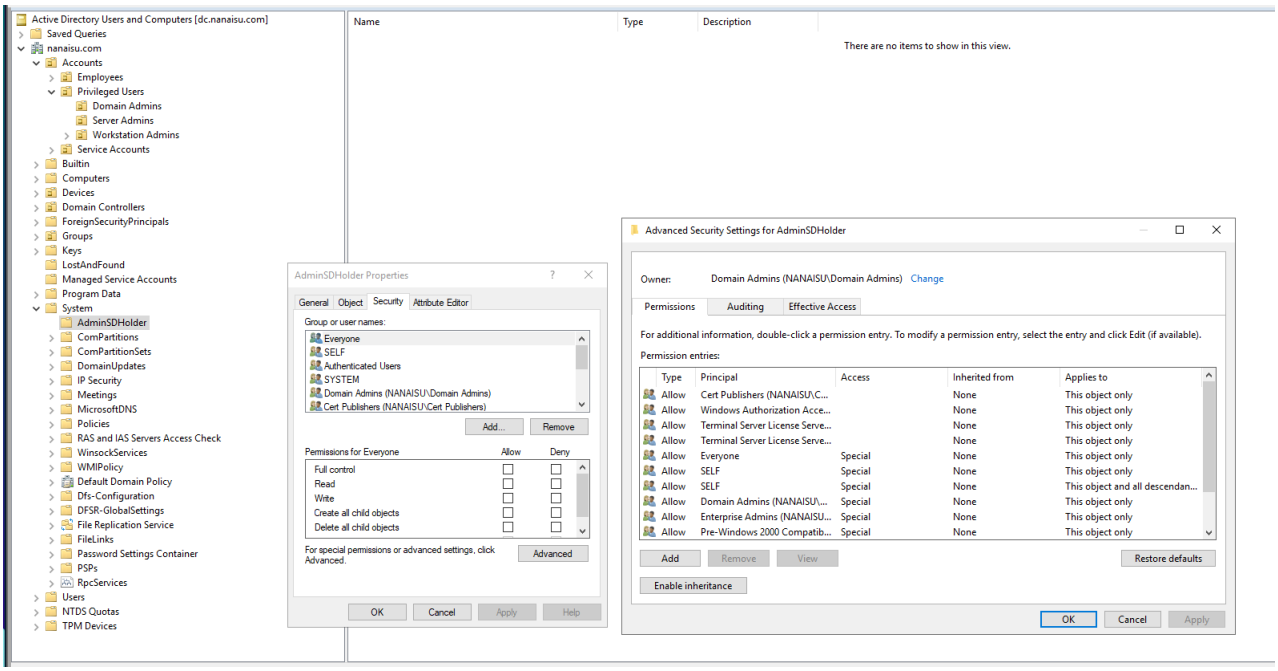


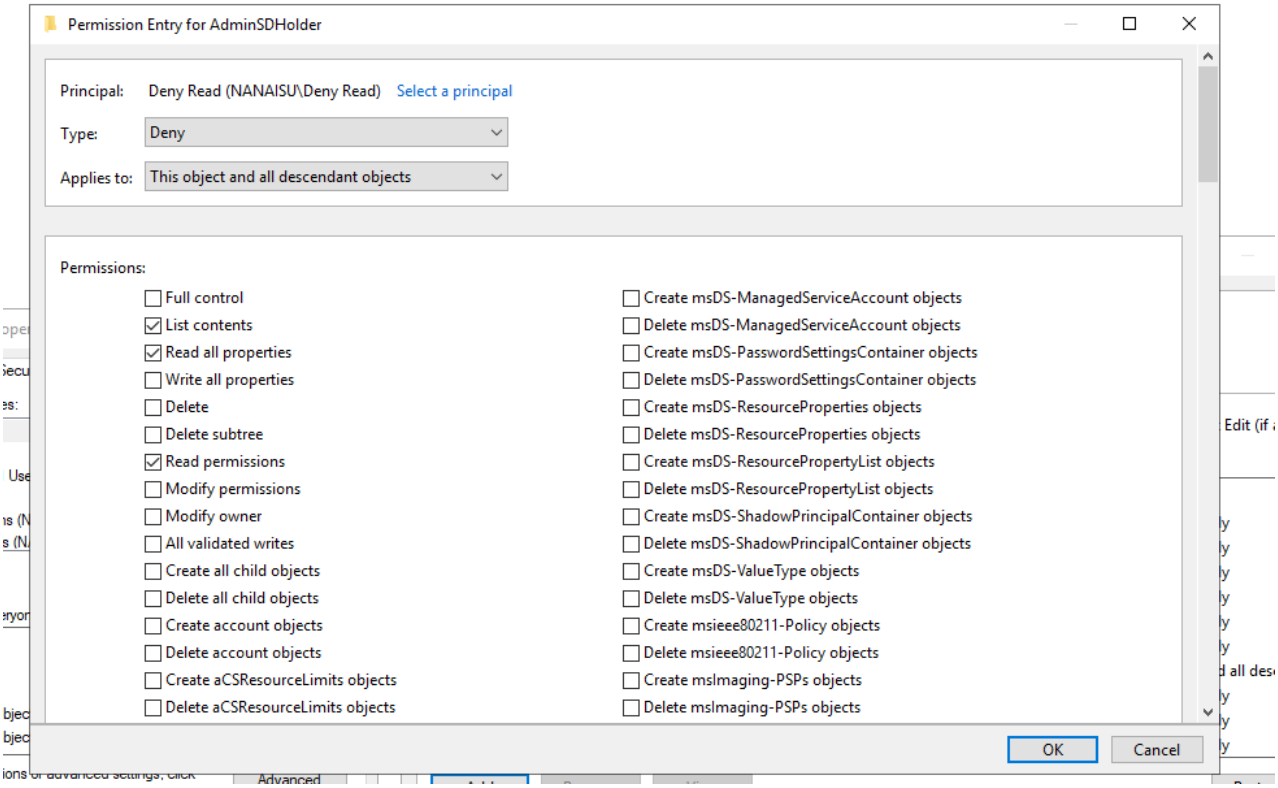
Figure 1 - Credit: Tenable Inc, "Securing Active Directory: How to Prevent the SDProp and adminSDHolder Attack"

Huh, so this means that if you modify the AdminSDHolder objects ACL, it will propagated to **every** user with adminCount set to 1. Today I learned, oh my god that makes life so much easier and better. You guys have no idea - I went down a PowerShell script rabbit hole nightmare nightmare nightmare, it was awful - Here's a screenshot, you don't need it, but its the summation of a couple hours worth of work last night…

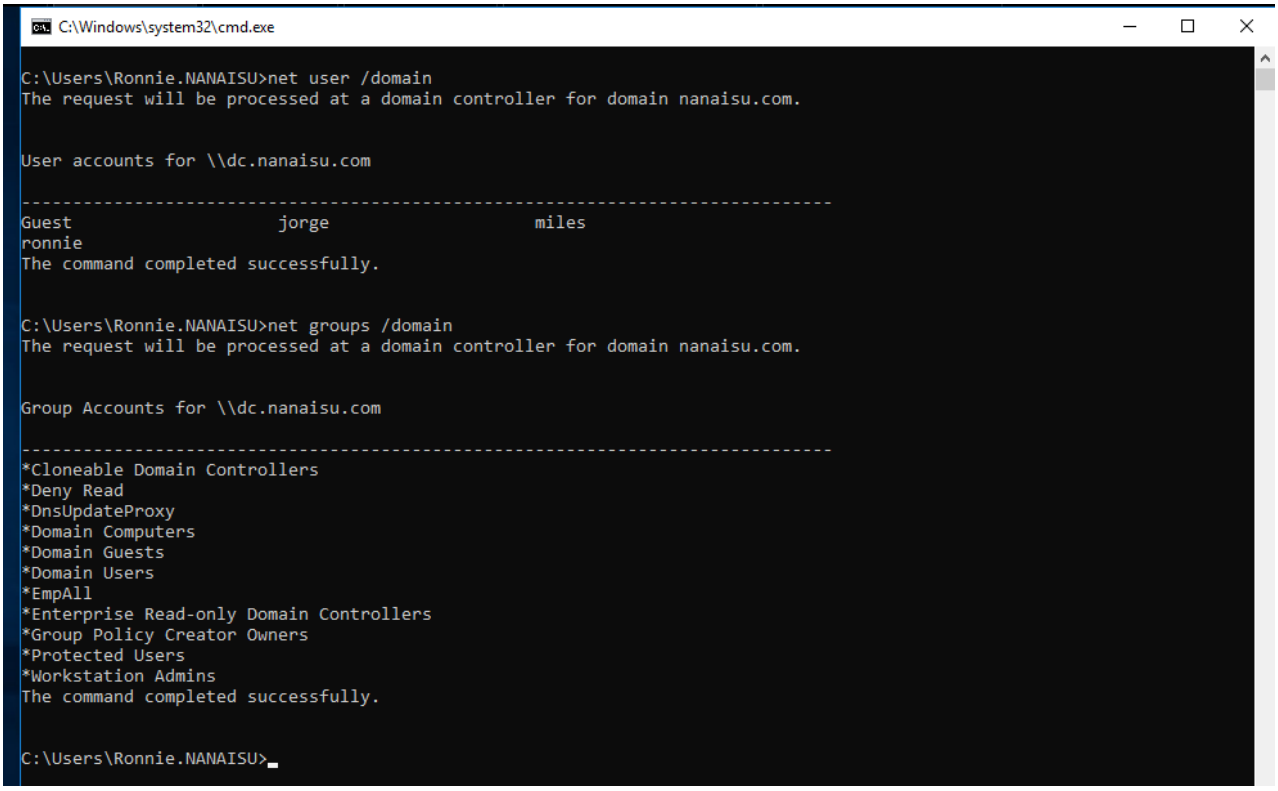
Okay so - enough complaining about me not understanding how AD works; Now that we know this valuable piece of information, we can modify the AdminSDHolder Object. To find that, we need to open Users and Computers, go to View and select “Advanced Features”. You’ll now see a whole bunch of new objects that you didn’t see previously! One of those will be called “System”, expand that list and the first entry should be AdminSDHolder.



We’ll now want to go through the same process as before - Assign the “Deny Read” group the “Deny” privilege over List Contents, Read All Properties and Read Permissions set:

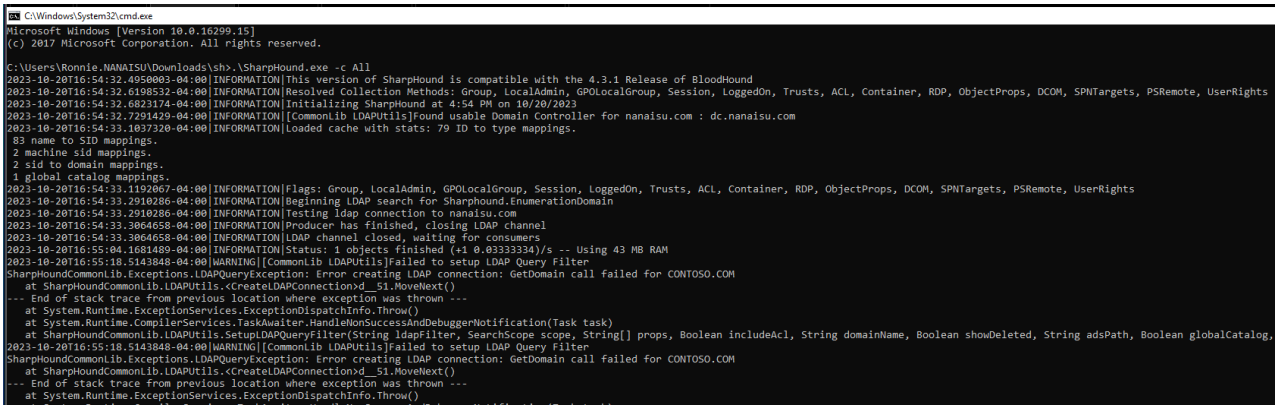


Click Okay, Apply, Okay, Apply and we should be set - We now have to wait an hour (or invoke a script like this: https://github.com/edemilliere/ADSI/blob/master/Invoke-ADSDPropagation.ps1) to force propagation onto objects with adminCount set to 1. For the sake of the lab, I'm going to force it. Now, it's time to check our work… Let's hop over to the workstation and query our users:



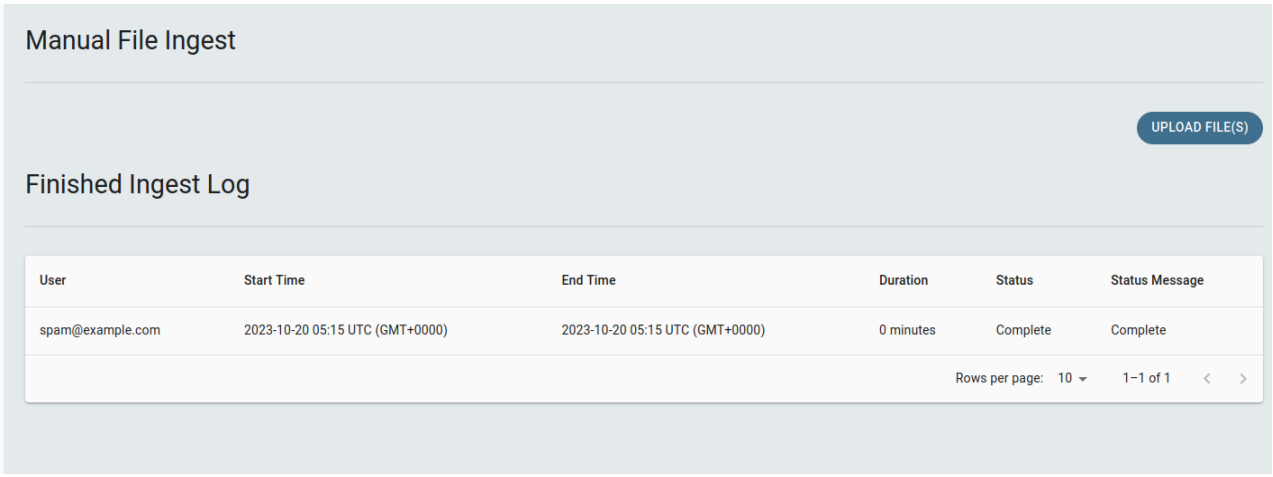
Wow, that even nailed our sensitive user groups too! I couldn't have asked for anything better. So - I originally had a whole section on Analysis in BloodHound written too, but I think that just got scrapped because this worked better than I intended it too… Remember the piece about enumerating groups directly? Yeah, that mostly just solved our problem! No need to do recursive group denials or any of that fun stuff.

So let's re-run SharpHound and see our new results:

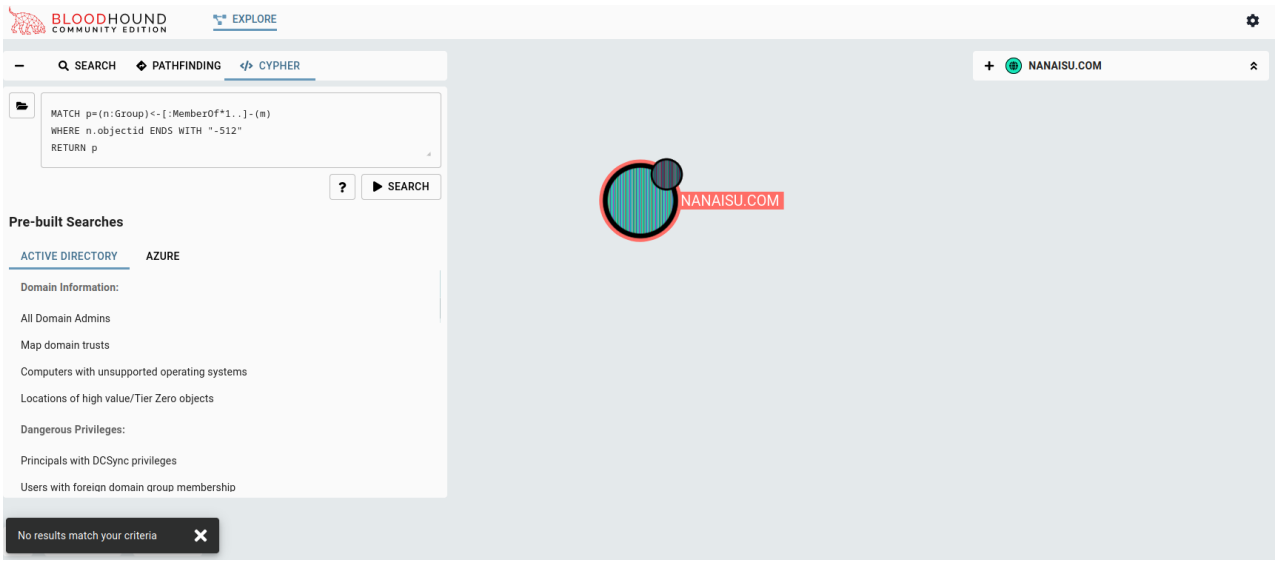


Analysis in BloodHound

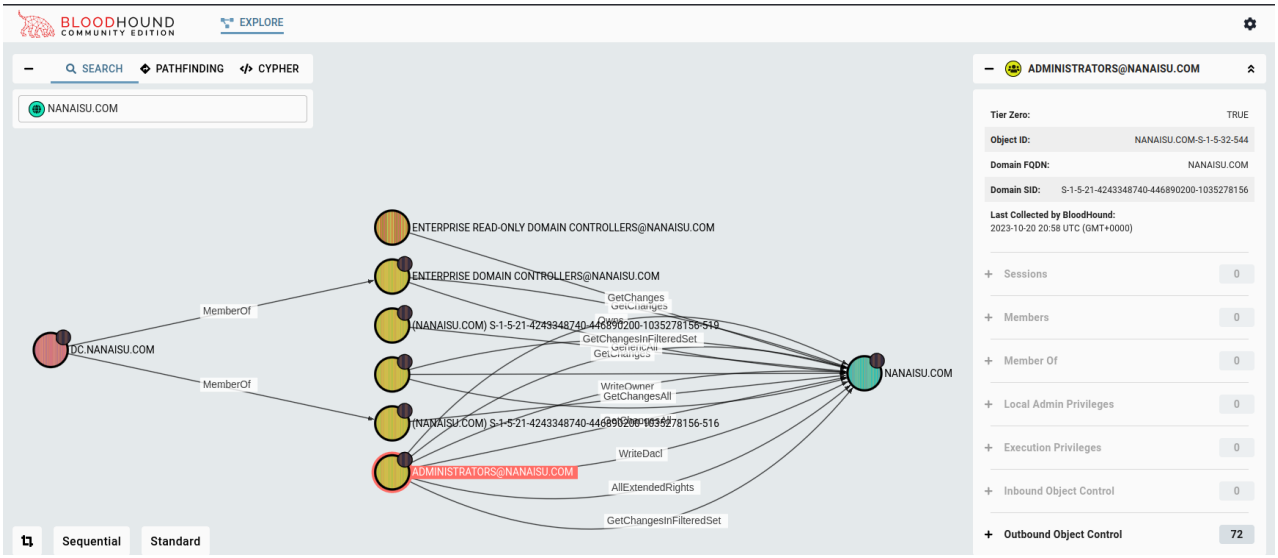
Let's move it over to our Kali VM to see the damage…



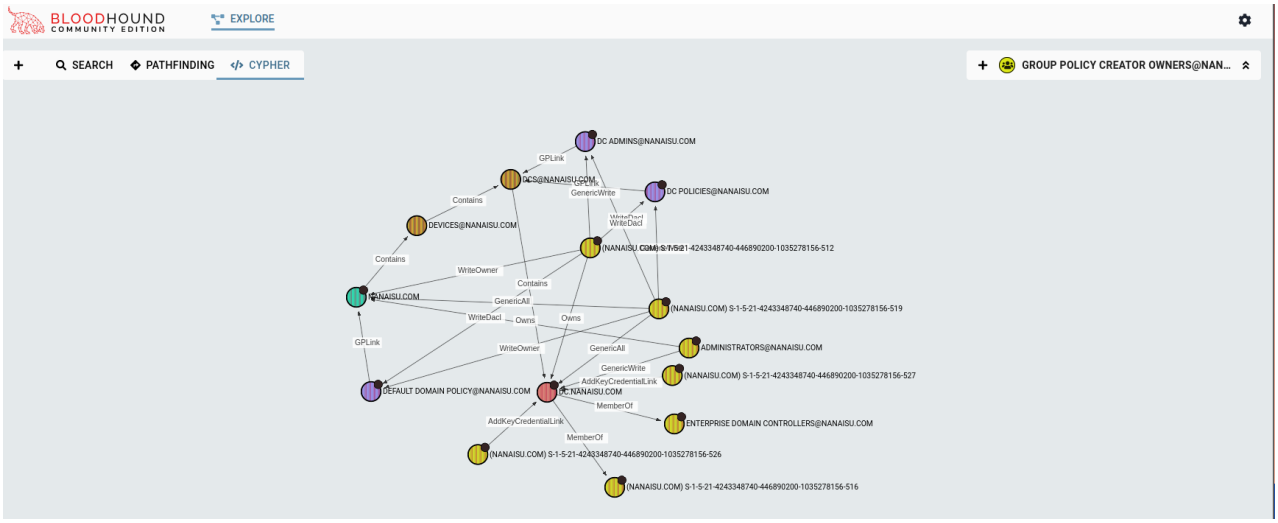
Ingestion process finished with little to no issues, let's check out our graph:



We've dunnit! We broke the pre-built "Find all Domain Admins" query. Okay! So we're well on our way - Jumping back, it is still possible to do some user enumeration in here, we can still gain *some* information via groups the user may be in - Note that we **do not** have permission to view a lot of critical information like the Domain Admin and Enterprise Admins groups and others. A good example is the "Group Policy Creator Owners" group, and another is the Outbound Object Control over the Domain:



So, it would ideally be best to assign this to **all** Tier Zero objects for the best amount of coverage. If you're unfamiliar with "What's Tier Zero and what's not?", I recommend checking out SpecterOps blog post series on What Is/Defining Tier Zero. BloodHound Community can massively help you understand what a Tier Zero asset is, where to look and what groups you might want to disallow enumeration on, etc. etc. Here's one of my favorite "Ruined Graphs":



This query was "Shortest Path to High Value/Tier Zero Targets". So, I think this is enough playing around with BloodHound for now. We've clearly done some damage to make it harder for attackers to enumerate our environment. That's a massive step forward compared to default Active Directory configurations.

To summarize what we've done so far:

- Create a group that is denied user enumeration
- Setup an ad-hoc ACE in the "Domain Admins" ACL to deny user enumeration
- Modify the "AdminSDHolder" Objects ACL to include an ACE to deny user enumeration
    - This was propagated to all users with adminCount set to 1

Caveats

I feel like I have to put one more section in before we I close out the blog post, so let's talk about it - There's obviously some caveats that need to be addressed here, so I'm going to try to nail as many as I (from an attackers perspective that I can think of):

- This is useless as RIDs of privileged users will almost always be the same
    - This is partially true, you are 100% right that RIDs 500, 512, 519 and others will always be sensitive; though we've been taught to **not** use these accounts for our daily administrative tasks. These accounts and groups should be hardened well before doing this.

- This requires a very mature environment and a good understanding
  - This is 100% correct. It requires a mature administrative team that understands seperation of privileges and what privileges are needed by what accounts in their environment. If you've got technologies like Microsoft Defender for Identity, Crowdstrike's Identity platform, or Sentinel One's platform, they can help analyze queries and help you identify where you may need to create exceptions
- This is security through obscurity. Security through obscurity doesn't work
  - Yes and No. This is following the prinicipals of least privilege. If a user does not need to query a privileged account, do not give them privileges to. There are valid cases where this can improve security. I'll cover this in my next blog posts
- Machine Accounts?
  - Yep. They're a really good concern here and are often forgotten about. I don't see any reason why you couldn't/shouldn't deny machine accounts the ability to query privileged users. If you follow Microsofts recommended T0/1/2 model for securing the enterprise, you should be able to deny Workstations from being able to query information about Domain Administrators without any issues. You should already have ACL/ACEs to prevent Domain Admins from logging into workstations to prevent incidental credential exposure.
- Trusted Domains
  - This is a big one that 100% needs to be addressed. Foreign domain members could also be a point where you could enumerate privileged users. I would suggest working with your Identity team to discuss and work on reviewing Domain Trusts, their configurations, permissions given in your primary domain, and see if they're necessary. It may be best to add them to a "Deny Read" style group as well.
- This does not address misconfigurations of ACLs
  - This is correct. You need to work with your PenTest/Red Team/Identity team to identify and remediate ACL abuse and misconfiguration within your environment. BloodHound is an awesome tool to help do so.
- This has the potential to break things
  - Yep! Proceed with caution. Never add a T0 member to this group. You could seriously break things. Test this. Pilot this. Slowly integrate this into production. At the end of the day, you know your environment best. Lots of users don't have a reason to do this, so don't let them do this. You wouldn't grant everyone the ability to read LAPS passwords, would you? Why does Tony from accounting need to know about Domain Admins? He really doesnt. It's an unnecessary risk and we know criminal threat actor groups are all about elevating privileges.

- This seems great - I still don't like the idea of targeting T0 groups. What are my other options?

  You could identify your crown jewels, see whats important for you. Do you have OT Systems? Deny enumeration on those objects. Do you have OT Domains? Same Deal. Do you have Jump Hosts into privileged network segments? Deny read/list access to those. No one needs to know they exist aside from the people who use them. Attackers have to hunt for users in specific groups too - Don't make their lives easy. Hide those systems and hide the users.
- It's not feasible for my team to implement across the domain today

  That's fine! It's a massive undertaking. There's several suggestions that I have that could make sense:
  - Going forward, you could implement a policy to deny domain user enumeration on newly provisioned accounts. This will reduce risk while not risking breaking any new systems.
  - Following a risk based alerting approach in the SOC, you could create a new SOAR playbook that locks down the users account permissions tightly **if** they begin exhibiting activities that may commonly be associated with infection (ex. LOLBAs, executing newly downloaded binaries from Outlook, or anything else you could think of). This is more of a reactive response. We generally strive to be proactive, but this would be better than nothing.

Closing

Okay! I think I hit them all, or at least all I wanted to cover, or at least all that I can think of. It's a big daunting topic. This was a big post, but is going to set the foundation for our next blog post where we dive into the methodology of creating Deceptive Accounts, OUs and Groups. In theory, we've thwarted off initial attackers initial attempts at enumerating and will now be counting on attackers to resort to manual enumeration to attempt to identify paths of lateral movement. This is where we will start to engage deceptive tactics to identify & trap attackers. The way I like to phrase this is "We've backed a wild animal in the corner - if it wants to attack, it has one clear path forward." That being to enumerate harder and make more noise.

That's all for today friends! I'm planning on splitting this up in a few different posts to be able to target different people & organize my thoughts a bit better for different sections. I think this could be a game changer for deception & security ops in general.

~ Ronnie

Comments