Azure Privilege Escalation via Service Principal Abuse

By Andy Robbins

Published: 2021-10-12 · Archived: 2026-04-05 15:53:32 UTC

Intro and Prior Work

On-prem Active Directory is here to stay, and so is Azure Active Directory. While these products have similar names, they in fact work very differently from one another. There are hints of vanilla AD's discretionary access control model in Azure, but permissions in Azure are mostly controlled through granted roles. Regardless of these differences, configuration-based attack primitives are alive and well in Azure, and researchers discover new attack primitives in Azure all the time.

In this blog post, I'll explain how a particular kind of attack path can emerge in Azure based on Azure's RBAC system – an attack path we have seen in the vast majority of Azure tenants we've gotten access to. I'll show you how the attack primitive works, how to protect yourself against it, and offer my own commentary on the opportunity this presents to Microsoft.

This work is not new – in fact, Microsoft's own documentation describes these attack paths. Additionally, two great researchers have discussed this attack primitive at length:

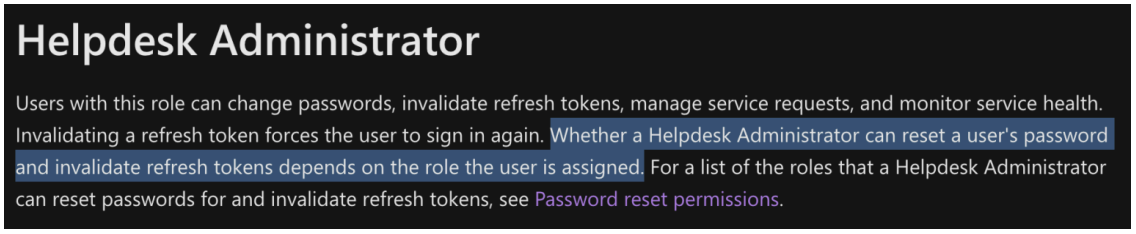
Dirk-jan Mollema wrote about abusing the Application Administrator role to add new secrets for service principals and escalate privileges here: https://dirkjanm.io/azure-ad-privilege-escalation-application-admin/

Karl Fosaaen spoke about Azure privilege escalation techniques in this talk:

https://www.netspi.com/webinars/lunch-learn-webinar-series/adventures-in-azure-privilege-escalation

Azure's Built-In Privesc Prevention System

Before we talk about the Service Principal-based privilege escalation, I want to describe Azure's built-in attack path prevention system. Azure Active Directory has a built-in system to protect against the emergence of attack paths, particularly around password reset privileges. When looking at the documentation for administrator roles that provide password reset privileges, you will often see wording like this:



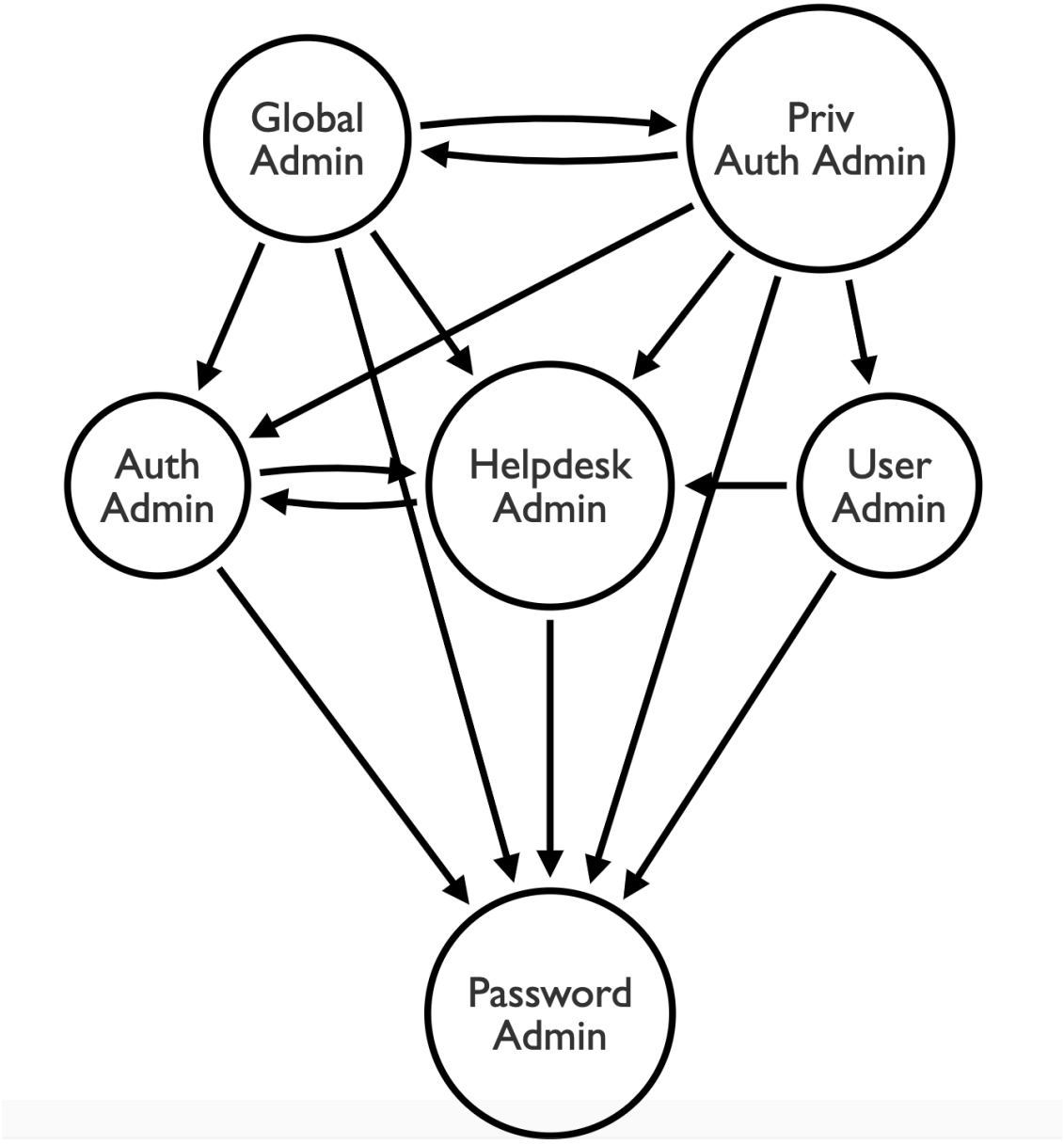
Source: https://docs.microsoft.com/en-us/azure/active-directory/roles/permissions-reference#helpdesk-administrator

In particular, note the highlighted text: “Whether a Helpdesk Administrator can reset a user’s password and invalidate refresh tokens depends on the role the user is assigned.”

This, to me, is confusing. So I set out to understand all the different possibilities for password reset privileges in Azure, and wound up creating this table:

Can a User with Role in Column A reset a password for a user with a Role in Row 2?														
	(No Role)	Global Administrator	Privileged Authentication Administrator	Helpdesk Administrator	Authentication Administrator	User Administrator	Password Administrator	Directory Readers	Guest Inviter	Message Center Reader	Privileged Role Administrator	Reports Reader	Groups Administrator	(Any Other Role)
Global Administrator	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Privileged Authentication Administrator	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Helpdesk Administrator	Yes	No	No	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	No	No
Authentication Administrator	Yes	No	No	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	No	No
User Administrator	Yes	No	No	Yes	No	Yes	No	Yes	Yes	Yes	No	Yes	No	No
Password Administrator	Yes	No	No	No	No	No	Yes	Yes	Yes	No	No	No	No	No

This brings a little more clarity, but look at what happens when we model these permissions using a graph:

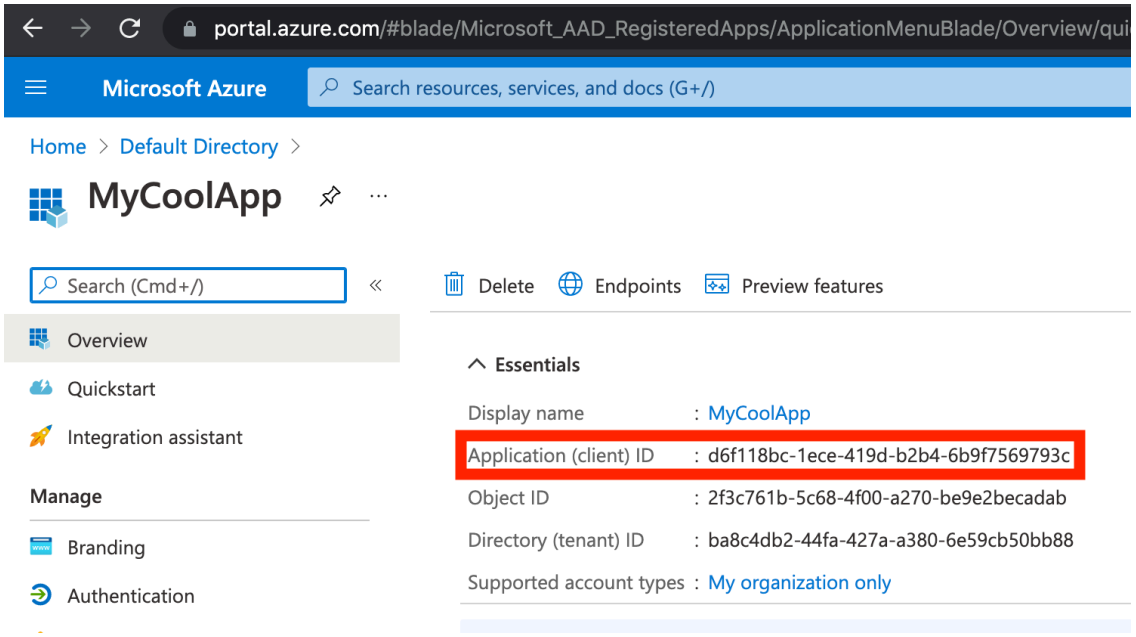


This is where the brilliance of this system finally becomes clear. Looking at the top row, we can see that Global Admins and Privileged Authentication Admins can reset each others', plus all other users' passwords, but the opposite is **not** true. In other words, **only** GA and PAA users can reset a GA password. This is part of the back-end Azure system and something that Azure admins have no control of.

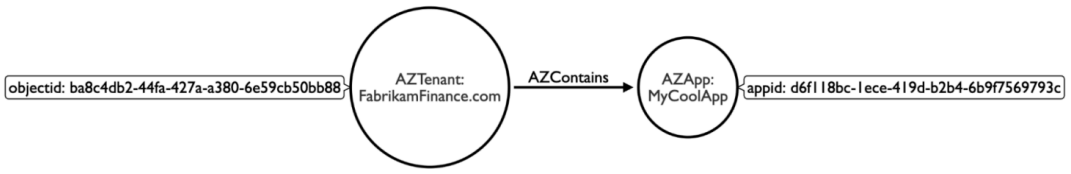
I love this system because it provides a highly effective, non-configurable, frictionless safety rail for admins that prevents the emergence of attack paths that include resetting a Global Admin's password. Azure admins can safely dole out the "Password Admin" role without worrying about the safety of their Global Admins – at least within the context of this particular attack primitive.

Escalating Rights via Service Principal Abuse

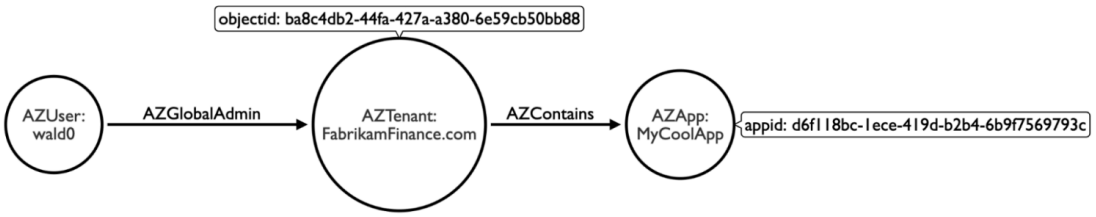
When you create or register an application in your Azure tenant, an application object is created with a unique application (client) ID:



Here, the app "MyCoolApp" has a unique identifier starting with d6f118bc. Below that, you can see the ID of my tenant as well. This app object "resides" within my tenant. Let's start to think of this in terms of a graph and see how the attack path emerges:

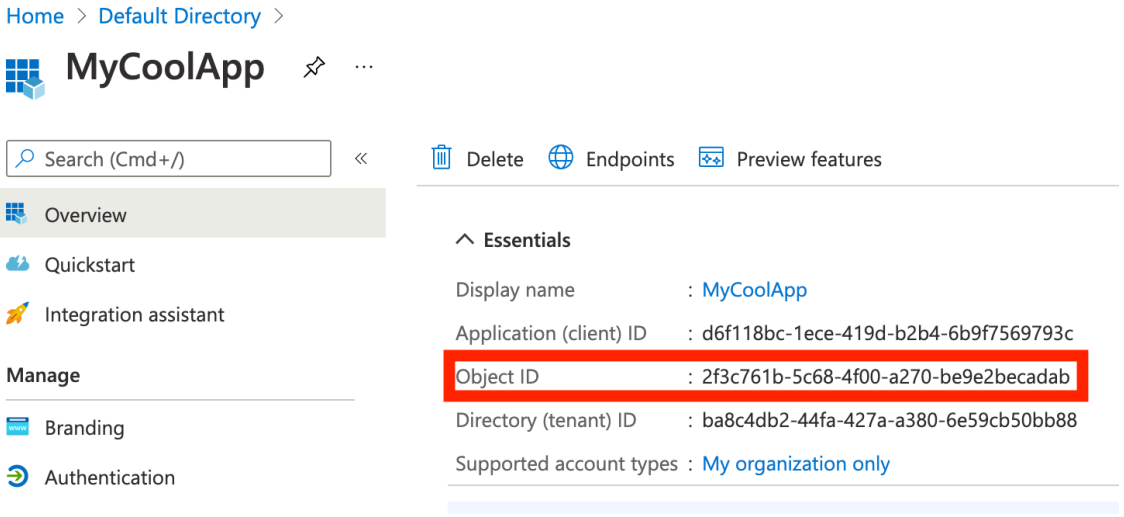


Because the app "resides" in my tenant, anyone with control of my tenant has control of the app. Let's model this with showing how my own Global Admin user has control of the tenant:

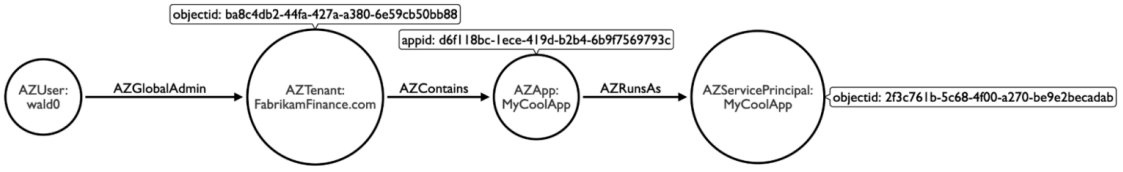


Nothing scary here: my user is a global admin against the tenant, the tenant contains the app, therefore the "path" here is that I have control of the app. Now let's extend this model to include service principals.

Azure apps needing to authenticate to the tenant to perform some action do so using an object called a Service Principal. Service Principals work kind of like users – you authenticate to the tenant with a "username" (object id) and a "password" (a certificate or secret). You can see the object id of the app's service principal in this screenshot:

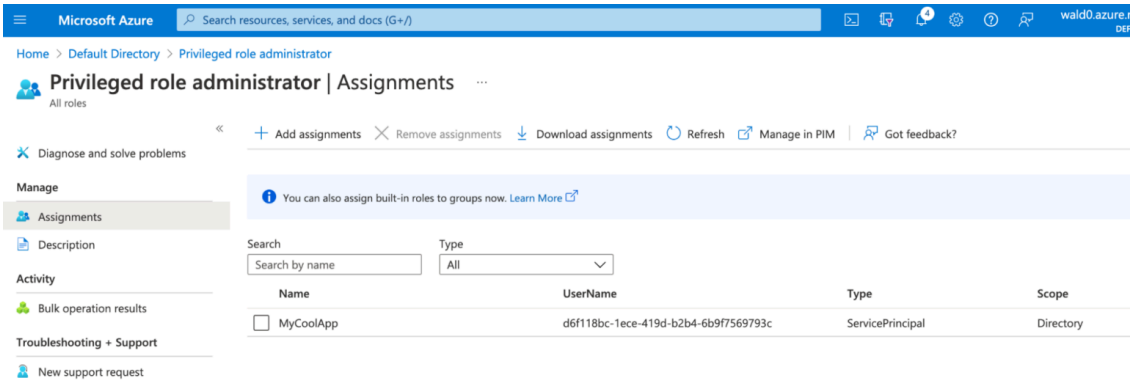


Let's go ahead and add that to our graph model:

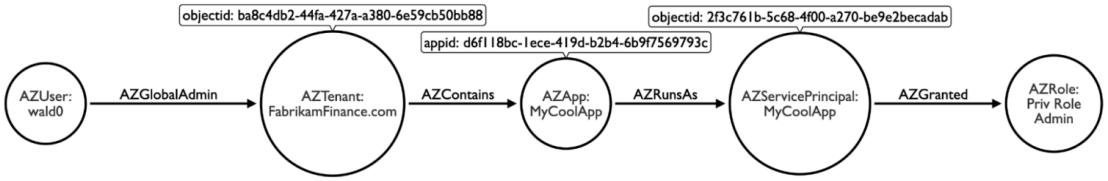


Again, nothing surprising here: my Global Admin user has control of everything here.

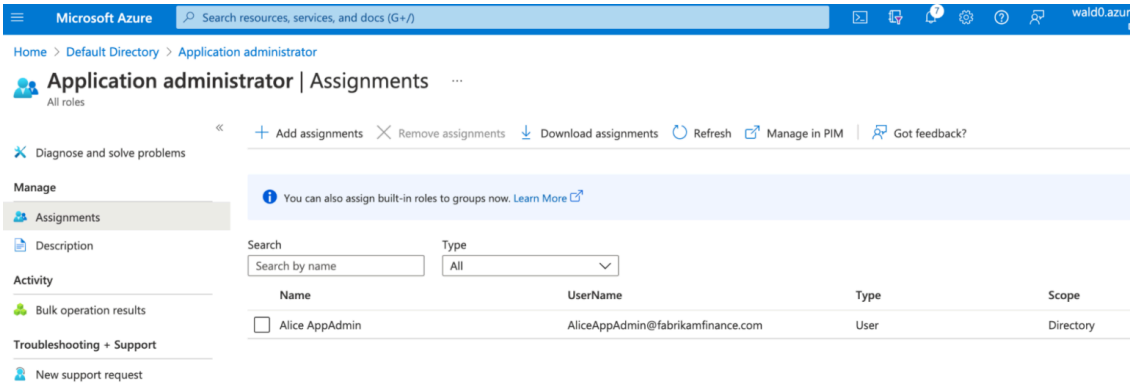
Service Principals can have admin roles in Azure just like users. For example, we have seen several Azure environments where at least one Service Principal has the Privileged Role Admin role. Let's recreate that in my own tenant:



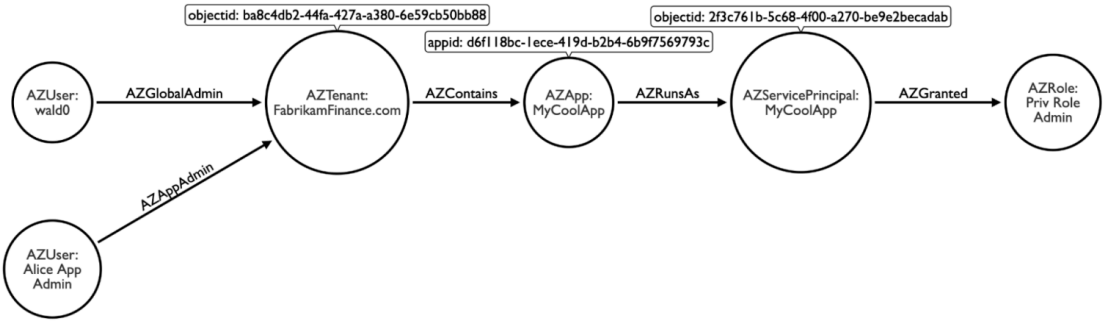
And let's extend the graph model to include this:



Again, nothing scary: an attack path from a Global Admin to the PRA role isn't a problem. Let's bring another user into the mix: Alice App Admin. Like the user's name suggests, we will grant this user the "Application Administrator" role in my tenant:



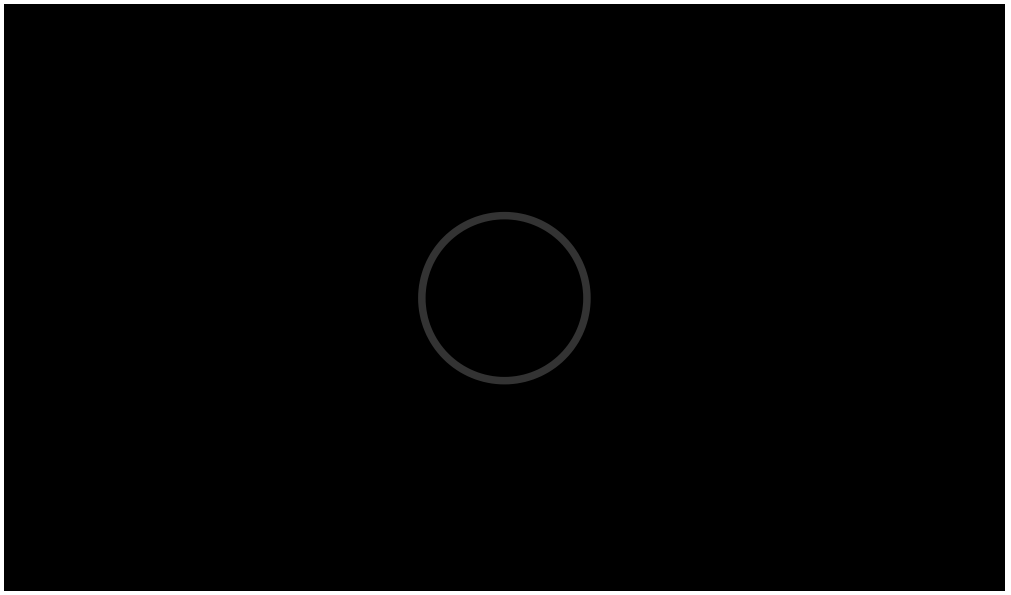
Let's also add this user and their role assignment into our graph model:



And… there it is. Our attack path has emerged, connecting the Alice App Admin user to the PRA role. The attack path works like this:

1. Alice App Admin has the Application Admin role, scoped to the tenant.
2. The tenant contains the MyCoolApp app, granting Alice App Admin control of the app.
3. Alice App Admin can add a new secret for this app's service principal.
4. MyCoolApp authenticates to the tenant as the MyCoolApp service principal.
5. The MyCoolApp service principal has the PRA role.
6. Alice App Admin can authenticate to the tenant as the MyCoolApp service principal and use that service principal's rights as a PRA to promote themself or another user to Global Admin.

Here's a video of the attack path in action, escalating from "Application Administrator" to "Global Administrator":



Prevention

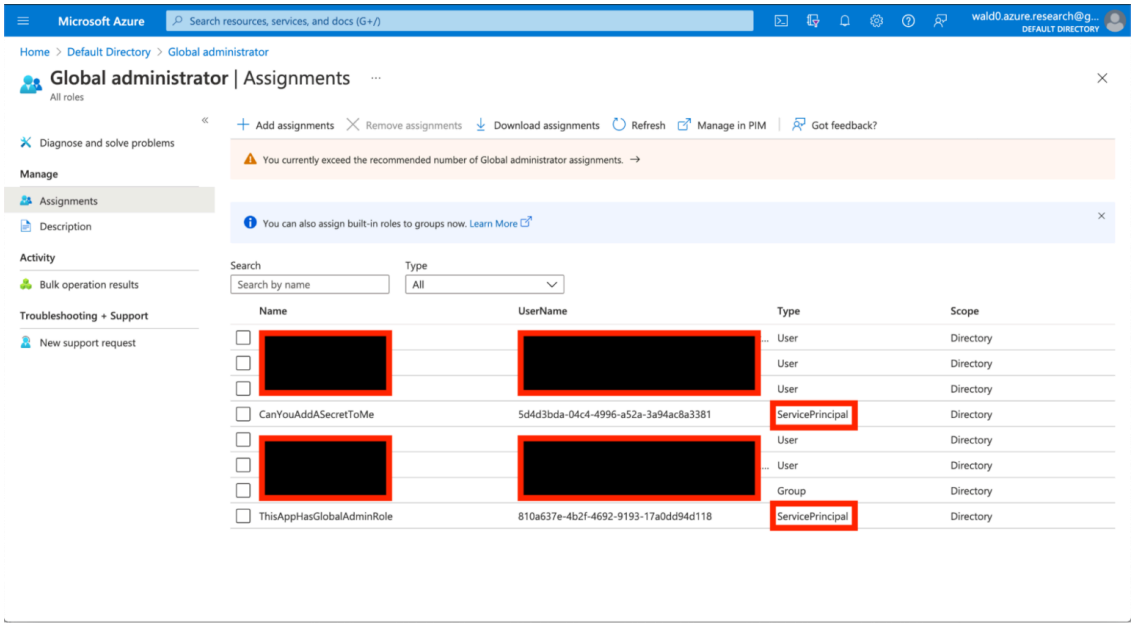
As discussed earlier, Azure has built-in mechanisms to prevent privilege escalation through role-based attack paths. Only those users with Global Admin or the Privileged Authentication Administrator role can reset a Global Admin's password. This simple, brilliant mechanism provides a highly effective safety rail to protect Azure admins from unknowingly introducing attack paths into their tenants. But this mechanism could go further.

I believe the best possible prevention against the privilege escalation technique described in this blog post is for Microsoft themselves to extend this mechanism to cover service principals as well. Just as the system provides built-in protections for Global Admin users, Azure could (perhaps should) also provide built-in protection for Global Admin service principals. I don't know the inner-workings of this system and whether this extension is even technically feasible, but I believe Microsoft can completely eliminate this attack primitive by extending the system to cover service principals, creating an even more secure platform for all of their customers.

In the meantime, Azure administrators should audit roles held by service principals, and determine the exposure of those service principals to the rest of their identities.

Azure admins can prevent this attack path by auditing roles held by service principals and comparing those roles to the other identities with control of apps. The most dangerous role in Azure is “Global Admin”, so let’s start there. Open the Azure portal, navigate to your tenant, then click “Roles and Administrators”:

Scroll down to and click on “Global Administrator”. Here, you will see the list of identities with this role currently activated. In the “type” column, look for “ServicePrincipal” entries:



There are two service principals in my tenant with the Global Admin role: “CanYouAddASecretToMe” and “ThisAppHasGlobalAdminRole”. These are the names of the service principals, and they are also the names of the apps associated with these service principals.

This is your first prevention opportunity: determine whether these apps actually need Global Admin rights in order to function. If these are third party apps, work with your vendor to get this level of privilege reduced to only the level required for the application to function.

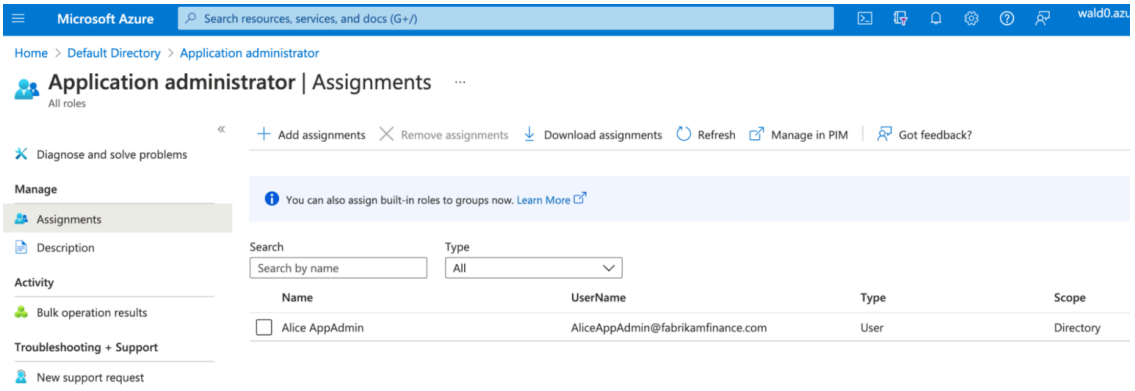
If these roles must persist, then your next step is to look at the identities in your tenant with the tenant-level roles that grant a principal the ability to abuse service principals. Those roles are:

Application Administrator
Cloud Application Administrator
Hybrid Identity Administrators

Additionally, these legacy/hidden roles can be abused to take over service principals:

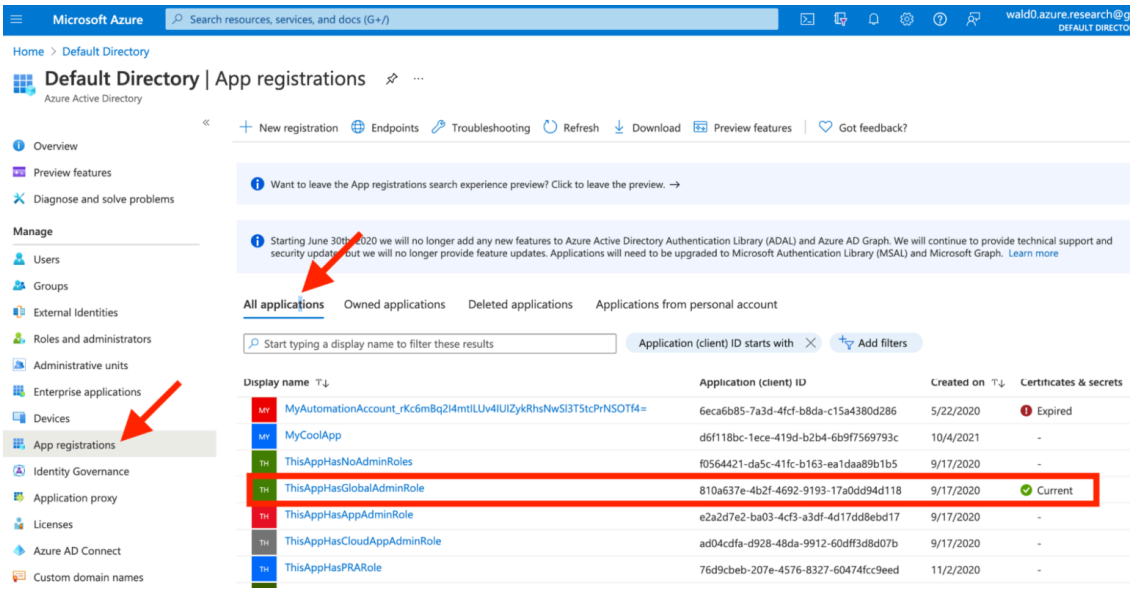
Directory Synchronization Accounts
Partner Tier1 Support
Partner Tier2 Support

Let’s audit the “Application Administrator” role by navigating to our tenant, clicking “Roles and Administrators”, then click on the Application Administrator role:

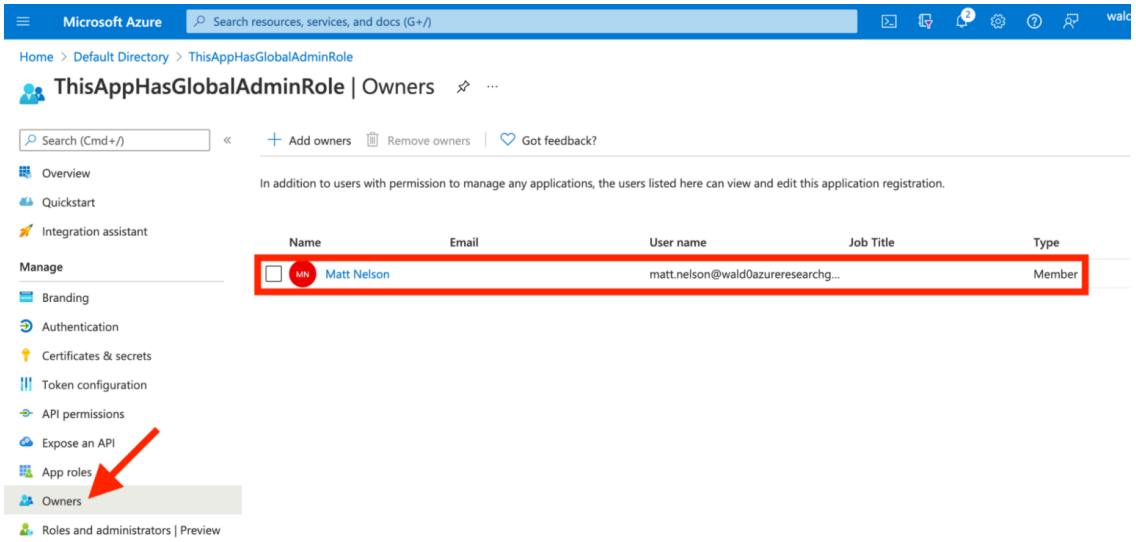


Here we see all users and service principals with this role currently active. This is your second opportunity for prevention: do all these users need the ability to manage credentials for all service accounts in your tenant?

We also need to check the app-level roles and per-app owners. Earlier we saw that the service principal, “ThisAppHasGlobalAdminRole”, was currently a Global Admin. Navigate to your tenant, click on “App registrations”, click on “All Applications”, then find the app associated with that Service Principal:

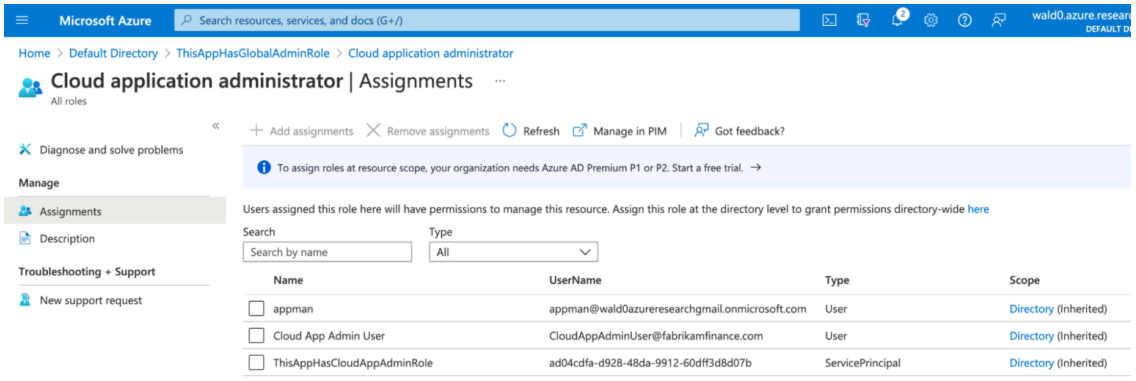


First, click on “Owners”, which will show you all principals that own the app (there can be more than one):



Do you trust Matt Nelson to add a secret to an app with Global Admin powers? I wouldn't.

Next, click on "Roles and administrators", which will show the app-level roles particular to this app. Click on "Cloud Application" administrator, for example, which will show the principals that have this role against the app both explicitly (scoped to the app), and where that role inherits down from the tenant:



These are your third and fourth opportunities for prevention: do you trust all these principals to have control of a service principal with Global Admin rights?

You should repeat this process for Service Principals that hold the following roles:

Application Administrator
Authentication Administrator
Azure AD joined device local administrator
Cloud Application Administrator
Cloud device Administrator
Exchange Administrator
Groups Administrator
Helpdesk Administrator
Hybrid Identity Administrator
Intune Administrator

Password Administrator
Privileged Authentication Administrator
User Administrator

This is also the point where the built-in password reset protections against Global Admins can break, creating attack paths from low-privilege users all the way to Global Admin by abusing control of service principals. In the real world, this is the most common avenue of escalating to Global Admin we have seen without needing to pivot down to on-prem AD first.

Conclusion

Microsoft is continually making improvements to Azure, whether those improvements are related to uptime, features, or security. There is an opportunity here for Microsoft to extend its existing design choices related to password reset rights to also cover Service Principals. I do not know the inner-workings of how that system works, or whether it’s even feasible to extend that system to cover Service Principals. Either way, this is a great opportunity for Microsoft to protect all of its customers and make Azure an even more secure platform.

Post Views: 1,841
