Azure AD Managed Identity Graph API Permissions Sample

This repository demonstrates how to create a `user-assigned managed identity` in Azure and grant it `application permissions` to the `Microsoft Graph API`. This is a common and secure pattern for Azure-hosted applications (like Web Apps, Functions, or VMs) that need to access Azure Active Directory resources `without managing secrets or credentials`.

Overview

This repository provides two methods for achieving this goal: an `imperative approach` using **shell scripts** and a `declarative approach` using **Terraform**.

Both methods accomplish the same outcome:

1. A user-assigned managed identity is created within a new resource group.
2. The managed identity is granted `User.Read.All` and `Group.Read.All` application permissions for the `Microsoft Graph API`.

This allows an application associated with this managed identity to, for example, read all user and group profiles in the Azure AD tenant.

Prerequisites

Before you begin, ensure you have the following:

- An active Azure subscription.
- One of the following toolsets, depending on your chosen method:
  - **For Scripts**: `Azure CLI` and `jq` (for the Bash version).
  - **For Terraform**: `Terraform` and `Azure CLI`.
- Sufficient permissions in your Azure AD tenant to:
  - Create resource groups and managed identities.
  - Grant application permissions (e.g., `Application Administrator` or `Cloud Application Administrator` role).

Implementation Methods

Method 1: Scripts (Imperative Approach)

The `sh/` and `ps/` directories contain scripts to `imperatively` create the resources and assign permissions.

- **Creation Scripts**: `sh/createUserAssignedManagedIdentiy.sh` and `ps/createUserAssignedManagedIdentiy.ps1` create the resource group and the managed identity.
- **Grant Permission Scripts**: `ps/Grant-GraphPermissionsToManagedIdentity.ps1` grants the Graph API permissions to the identity.
- **Remove Permission Scripts**: `sh/removePermissionsFromManagedIdentities.sh` and `ps/removePermissionsFromManagedIdentities.ps1` remove the Graph API permissions from the

identity.

How the Permission Script Works

1. **Fetches IDs**: Retrieves the unique object IDs for the `managed identity's service principal` and the `Microsoft Graph service principal`.
2. **Finds Role IDs**: Queries the Graph service principal to find the specific IDs for the `User.Read.All` and `Group.Read.All` application permissions (`app roles`).
3. **Assigns Roles**: Makes API calls to Azure AD to create the `appRoleAssignment`, linking the managed identity to the desired Graph permissions.

🔐 Why it's Needed

The core reason these scripts are necessary is that Azure uses **two distinct permission models**:

1. **Azure RBAC (Role-Based Access Control)**: This is the primary permission system for Azure resources, often called the "control plane." It governs actions like creating a VM, deleting a Storage Account, or reading a Key Vault secret. The "Access control (IAM)" blade in the Azure Portal is the GUI for Azure RBAC.

2. **Microsoft Entra ID Application Permissions**: This system governs access to `data within an API`, most notably Microsoft Graph. These permissions (called `app roles`) control whether an application can read all user profiles, send an email as a user, or read SharePoint site contents.

The GUI Limitation: The Portal Only Shows Azure RBAC

When you go to a Managed Identity in the Azure Portal and click on "Azure role assignments," you are interacting *only* with the `Azure RBAC system`. The portal provides a user-friendly way to grant the identity roles like "Storage Blob Data Reader" or "Key Vault Secrets User."

However, the portal **does not have a built-in interface** to browse and assign API permissions (`app roles`) from other applications like Microsoft Graph to a managed identity. There is no "Grant Graph Permissions" button on the managed identity page.

The Solution: `Direct API Interaction`

To grant a managed identity permissions to call the Microsoft Graph API, you must perform a series of steps directly against the underlying Entra ID APIs:

1. Find the **`Service Principal`** for your managed identity. This is its identity within Entra ID.
2. Find the **`Service Principal`** for the target API (Microsoft Graph).
3. Look up the specific **`App Role ID`** on the Graph service principal that corresponds to the permission you want (e.g., the unique ID for `User.Read.All`).
4. Create an **`appRoleAssignment`** object that links the managed identity's service principal to the Graph app role.

This process is complex and not exposed through the Azure Portal GUI. The scripts and Terraform configuration in this repository automate these exact steps, providing a reliable and repeatable way to configure the necessary API permissions.

Method 2: Terraform (Declarative Approach)

The `tf/` directory contains a declarative `Infrastructure as Code (IaC)` solution to deploy and configure all resources in a single step.

How the Terraform Configuration Works

- **Providers**: It uses the `azurerm` provider to create the resource group and managed identity, and the `azuread` provider to interact with Microsoft Entra ID.
- **Resources**: It defines the `azurerm_resource_group` and `azurerm_user_assigned_identity` resources.
- **Data Source**: It uses the `azuread_service_principal` data source to look up the Microsoft Graph service principal by its well-known application ID (`00000003-0000-0000-c000-000000000000`).
- **Role Assignment**: It uses the `azuread_app_role_assignment` resource to grant the permissions. This resource declaratively links the managed identity's principal ID to the desired app role IDs (`User.Read.All` and `Group.Read.All`) on the Graph service principal.
- *Note: The configuration uses a `for_each` loop, making it easy to manage a list of permissions by simply modifying the `graph_permissions` variable.*

How to Use

1. **Log in to Azure**: Open your terminal and log in to your Azure account.

```
az login -t <YOUR_TENANT_ID>
```

Using Scripts

1. **Create the Managed Identity**: Run the creation script for your preferred shell.
    - **Bash:**

```
./sh/createUserAssignedManagedIdentiy.sh
```

    - **PowerShell:**

```
./ps/createUserAssignedManagedIdentiy.ps1
```

2. **Assign Permissions**: Run the corresponding script to assign permissions.
    - **PowerShell:**

```
# This grants User.Read.All and Group.Read.All
./ps/Grant-GraphPermissionsToManagedIdentity.ps1 -ManagedIdentityName
"my-demo-identity"
```

3. **(Optional) Remove Permissions**: To revoke the permissions, run the removal script.
    - **Bash:**

```
# Note: You may need to edit the script to set the
MANAGED_IDENTITY_NAME variable
./sh/removePermissionsFromManagedIdentities.sh
```

    - **PowerShell:**

```
./ps/removePermissionsFromManagedIdentities.ps1 -ManagedIdentityName
"my-demo-identity"
```

Using Terraform

1. **Navigate to Directory**:

```
cd tf
```

2. **Initialize Terraform**: Download the required providers.

```
terraform init
```

3. **Plan and Apply**: Review the planned changes and approve them to create the resources.

```
terraform apply
```

Clean Up Resources

Script Cleanup

To remove the resources created by the scripts, you can delete the resource group.

- **Using Bash / Azure CLI:**

```
az group delete --name my-managed-identity-rg --yes --no-wait
```

- **Using PowerShell:**

```
# This requires the Az PowerShell module (Install-Module Az)
Remove-AzResourceGroup -Name "my-managed-identity-rg" -Force
```

Terraform Cleanup

To destroy the resources managed by Terraform, run the following command from the `tf` directory:

```
terraform destroy
```