CHAPTER 2

Identity and Access Management

In this chapter, we are going to learn about one of the fundamental building blocks of cloud-native security – identity and access management. After reading this chapter, you will be able to design secure access control solutions for your Azure administrative and application access.

Azure Active Directory

Identity is the new perimeter in the public cloud world, and there is no identity and access management in Azure without Azure Active Directory (Azure AD or AAD). Regardless of your identity provider (IdP) solution, authentication to Azure subscriptions and resources is always performed by Azure Active Directory.

Overview of Azure Active Directory

Azure Active Directory (Azure AD or AAD) is an identity solution as a service (IDaaS), managed by Microsoft. As it is the sole supported authentication solution for Microsoft's own SaaS services, such as the Microsoft 365 suite ranging from Teams to Exchange Online, Azure AD is as ubiquitous in the cloud as Active Directory Domain Services (AD DS) is in the on-premises world. With support for single sign-on, Azure AD can extend to third-party SaaS services outside Microsoft, too.

Azure AD is delivered as an evergreen software as a service (SaaS) by Microsoft. As such, the operational obligations of IT departments are replaced by configuration and change management responsibilities. Microsoft is constantly adding features to Azure AD, so simply keeping up with, verifying, and rolling out new features is a challenge.

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_2

Identity Sources

Azure Active Directory supports multiple sources of identities, as described in Figure 2-1. You can provision users and other identities directly into Azure AD, or you can synchronize identities from your existing systems. If your organization is using Active Directory Domain Services, you can synchronize the identities with Azure Active Directory Connect synchronization services. This is the approach most established organizations are taking. As this combines the usage of on-premises and cloud services, it is referred to as **hybrid identity**.

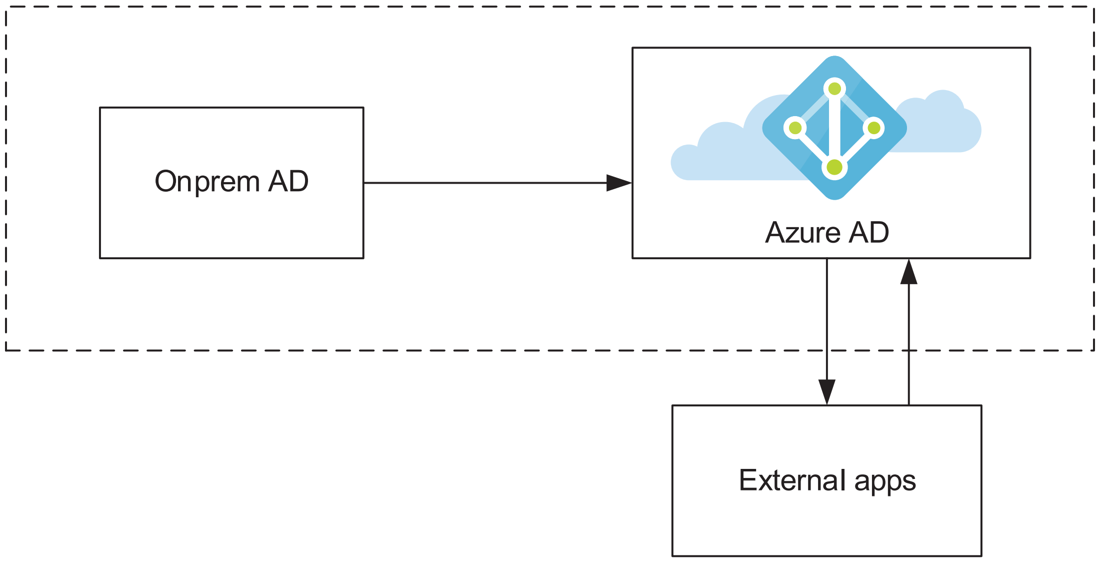


Figure 2-1. *Options for identity sources*

If your organization does not have traditional Active Directory Domain Services in use, you can use cloud-based HR systems to provision Azure AD users.

Furthermore, you can user management APIs of System for Cross-Domain Identity Management (SCIM) to provision users from your Azure AD into other cloud applications. This lets you use Azure AD as your main identity source for other services, such as Adobe, AWS console, Slack, or Salesforce.

Relationship Between Azure Active Directory and Azure

Each Azure subscription is linked to an Azure Active **Directory**, sometimes referred to as an **account** or a **tenant**. However, not all directories are linked to Azure subscriptions. This is the case when, for example, an organization is only using Microsoft SaaS services, but not (yet) any Azure services.

Note that even if there has not been any formal usage of Azure, there could still be several Azure subscriptions linked to the Directory. Any new subscriptions created by users in your Directory are by default linked to the Directory. There is no approval needed from any elevated Azure Active Directory Role assignee.

As all Azure subscriptions are linked to an Azure AD, some Azure services leverage it in a highly integrated manner. This lets you control authentication in an effective and centralized manner across your Azure subscriptions. This link is important to understand across your whole application life cycle. If your Azure resources are moved to another Directory in case of mergers, outsourcing, or other business reasons, you need to remove and re-create the link between your services and Azure AD. As of the writing of this book, this applies to the following services:

- Azure SQL database
- Azure Key Vault
- Azure role-based access control (RBAC) assignments

Intelligent Security Graph

The Intelligent Security Graph is a collection of security-related data points gathered across the Microsoft 365 suite of services. As Azure AD is used for authentication for each of those services, the Graph can combine the data in an intelligent manner. Altogether, Azure AD is used for several hundreds of billions of authentications per month. In addition to data from their first-party services, Microsoft collects data from other sources, such as external feeds and dark markets. Altogether, Microsoft collects over eight million indicators of compromise (IoC) per day.

Before using this data from various sources, several privacy and compliancy boundaries need to be passed. The data is anonymized and synthetized before being fed into the Graph.

Conditional Access

Based on this large and diverse set of data, the Graph can provide us with risk ratings for each authentication event. These risk scores can be leveraged automatically to take a risk-based approach for securing access, such as blocking an authentication attempt by a user that is trying to sign in from geographically distant locations within a short amount of time (so-called impossible travel scenario). This is available as a feature called **conditional access**.

Conditional access lets you configure rules for automatically applying certain controls for any authentication events across our identity perimeter. These rules can be as straightforward as blocking users attempting to log in from devices infected with malware, or more complex, such as combining network risks with past behavior of the user attempting to authenticate.

After the conditional access evaluates the attempted authentication, it will either allow or deny access. It can even be configured to allow access after certain additional controls, such as multifactor authentication challenge, are satisfied.

Securing Your Azure AD

Securing your Azure AD is such a vast topic that it deserves its own book. For the purposes of this book, I am introducing you briefly to the Azure AD security topics that have the most direct impact on your Azure usage.

Directory Splitting

Directory splitting, or separation of users and subscriptions of the same organization into multiple Azure AD directories, is not considered a best practice in the cloud era. Microsoft recommends using a single Azure Active Directory per organization in the Cloud Adoption Framework,[1] as illustrated in Figure 2-2. Most of your isolation requirements can be met with Azure RBAC and separation of subscriptions, instead of Azure AD directories.

[1] https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/decision-guides/identity/

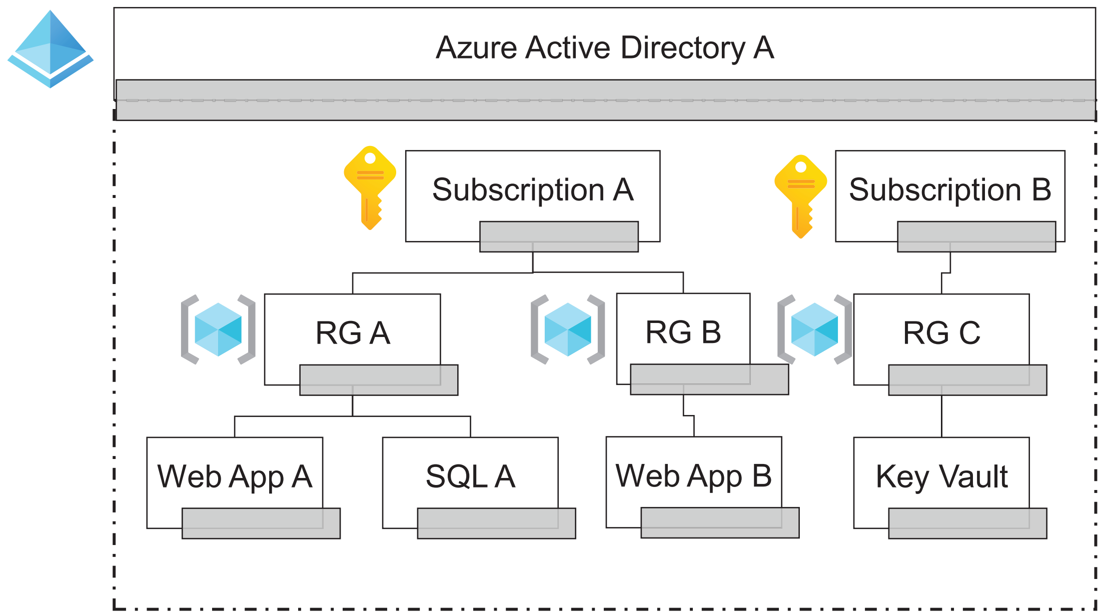


***Figure 2-2.** The relationship between the Azure AD directory and subscriptions*

If your organization has already experimented with Azure usage, you might have multiple Azure AD directories. In that case, you might need to start your Azure security journey by consolidating your subscriptions under a single directory. One of the byproducts of this consolidation exercise is that you will need to re-create all RBAC assignments.[2] I encourage you to use this opportunity to enforce your new access control standards.

As always, there are some exceptions to this. You might operate in a business environment, such as joint ventures, where a separate Azure AD directory is necessary. Or you might be using one of the sovereign clouds, such as Azure US Government or Azure China. These sovereign clouds are already by nature disconnected from the global Azure infrastructure, including Azure AD.

[2] https://docs.microsoft.com/en-us/azure/role-based-access-control/transfer-subscription

Guest Management

You should set controls on Azure AD B2B guest management. Without proper controls, any regular member in your Azure AD can invite guests from virtually anywhere! The guests do not need to belong to the Azure ADs of your trusted partners or even any Azure ADs. Microsoft personal accounts and Google accounts are among the possible identity providers for guests.

Once a guest user has been invited and they have accepted the invitation, the guest can be assigned to any Azure AD administrative roles and Azure RBAC roles. This means that you need to be able to trust the identity of the guest user with limited visibility to their activities!

Note Guest users can be assigned to the same Azure AD administrative roles and Azure RBAC roles as native members of your Azure AD directory.

You can use the **Guests can invite** and **Members can invite** settings in your Azure AD to control who can invite guests. When both are set to No, only users with Azure AD administrative roles, such as Guest Inviter, can invite guests. Additionally, you can choose from the following **collaboration settings** to further control guest inviting:

- Allow invitations to be sent to any domain (most inclusive).
- Deny invitations to the specified domains.
- Allow invitations only to the specified domains (most restrictive).

As you do not have full visibility to the guest user identities, it is worth considering adding a conditional access policy to require multifactor authentication from guest user authentications to Azure Management plane (labeled **Microsoft Azure Management** in Azure AD Conditional Access).[3]

[3] https://docs.microsoft.com/en-us/azure/active-directory/external-identities/b2b-tutorial-require-mfa

Default Access Management

Contrary to Azure RBAC, every Azure AD user (including guest) has a set of default permissions.[4] I recommend you control the following permissions to secure your Azure environment:

- Users can register application.
- Ability to create security groups.
- Restrict access to Azure AD administration portal.

Privileged Access Management

Azure subscriptions trust their linked Azure AD directories. A key feature of this trust is the fact that Azure AD Global Admins can self-elevate themselves as User Access Administrator to any Azure resources that are linked to your Azure AD.[5]

Note Azure AD Global Admins can take over any Azure subscriptions linked to your Azure AD!

There are several controls available for you in securing privileged Azure AD users. First, to mitigate compromised on-premises users pivoting to the cloud,[6] or vice versa, do not synchronize accounts with high Active Directory privileges to Azure AD.

Second, you should define two or more emergency accounts (sometimes called Break-the-Glass accounts) that are as resilient to disruptions in Azure AD as possible. These users should be created directly in the cloud, that is, using your *.onmicrosoft.com domain instead of your domain. They should be exempt from your regular conditional

[4] https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/users-default-permissions#compare-member-and-guest-default-permissions

[5] https://docs.microsoft.com/en-us/azure/role-based-access-control/elevate-access-global-admin

[6] https://techcommunity.microsoft.com/t5/azure-active-directory-identity/protecting-microsoft-365-from-on-premises-attacks/ba-p/1751754

access policies, as well as phone-based multifactor authentication.[7] These accounts should have the highest level of monitoring controls applied, and their usage should be extremely limited.

Third, you should only use Azure AD accounts with your Microsoft Billing portals, to mitigate attacks against personal Microsoft accounts.

Finally, you should separate your user accounts from Global Admin accounts.

Conditional Access Policies

To complement your Azure RBAC access control, you should consider setting conditional access policies to protect the following authentications using MFA:

- Authentication of Azure AD administrators.
- Authentication of any user to Azure Management (through Azure Portal or a command-line interface).
- Authentication using a legacy authentication protocol. This setting is especially effective in mitigating password spray attacks, as it enforces MFA.

In addition to enforcing MFA, you can also control the network locations or devices allowed to perform the authentication.

Authorization: Azure Role-Based Access Control

All access to any Azure resources needs to be explicitly granted using Azure role-based access control (RBAC). Figure 2-3 decomposes the RBAC assignment, the key unit of access control in Azure.

[7] https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-resilient-controls

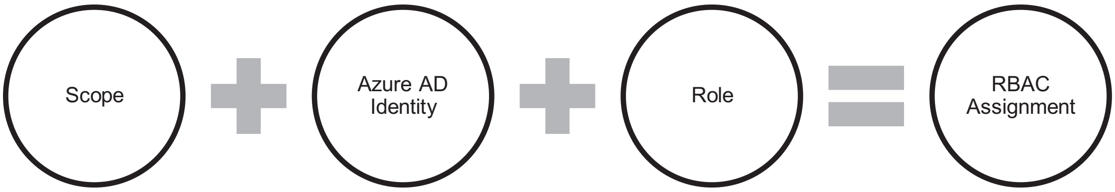


Figure 2-3. *Azure role-based access control*

In addition to role-based access control, Azure has recently introduced support for attribute-based access control (ABAC).[8] This gives you more granularity to access control, by requiring specific attributes from the scope or the Azure AD Identity, in addition to the RBAC role assignment. As of the writing of this book, ABAC is supported in Azure storage account. We will discuss ABAC in Chapter 5 of this book.

Scope

Azure role-based access control assignment starts with the **scope** of the assignment. A scope is the target of the assignment and can be one of the following:

1. Management group
2. Subscription
3. Resource group
4. Resource, such as a Cosmos DB account
5. Sub-resource, such as a subnet of a virtual network resource

Access assigned to a higher level of scope automatically grants access to lower levels. This inheritance applies throughout the life cycle of the RBAC assignment: access granted to a subscription will grant access to every resource group, resource, and sub-resource in that subscription starting from the assignment time. This means that this same RBAC assignment will automatically grant you access to every new resource any user will add to the subscription, anytime in the future.

[8] https://csrc.nist.gov/publications/detail/sp/800-162/final

Identity

Identity refers to the subject of the Azure RBAC assignment. The identity can be one of the following:

- **User** in the Azure AD linked to the subscription. Both member users and guest users are accepted.
- **Group** in the Azure AD linked to the subscription.
- **Application** registered to the Azure AD or created as a managed identity for Azure resources (see section managed identities for Azure resources for more details).

As the RBAC assignment subject is always an entity in your Azure AD, any changes to those entities are also reflected in the effect of RBAC assignment. For example, if a developer leaves your organization and their user is disabled in your identity provider, that information is synchronized to your Azure AD and the user will not be able to successfully authenticate, even if their RBAC assignment was not removed.

Note Regardless of your approach to Azure AD B2B, you should always avoid creating RBAC assignments to guest users that have not yet accepted the invitation to the Directory! The guest users can accept the invitation to join your Azure AD from a variety of identity providers, not all of which may meet your security requirements.

Role

As there are no default RBAC assignments in place for users, you need to always allow any kind of access explicitly to your Azure environment. A **role** of the Azure role-based access control defines a list of **Resource Provider actions** that are allowed or not allowed in the **scope** of the assignment. Identities can be assigned multiple roles simultaneously, in which case the resulting permission is the sum of all permission granted in the assigned roles.

Role Definition

Resource Providers are collections of APIs that provide functionality for all Azure services. This functionality may vary from listing resource properties to virtually any functionality possible with the Azure service you are working with. For example, the `Microsoft.Storage/storageAccounts/listkeys/action` provides the functionality to request the shared account keys of a storage account.

A role definition describes which Resource Provider operations are available for the assignee of the role in the action block of the role definition. The role definition can also include exclusions, in the notActions scope. If a user attempts to perform an action not permitted by their role, they will not succeed and are presented with an error message. The unsuccessful attempt is also logged in the activity log.

Note that the Reader role does not grant you access to read data stored in your Azure resources, such as the content of a storage account. The Reader role only grants access to the control plane. To manage access to content, you need to control expand the role definition to consider dataActions and notDataActions. With these, the role definitions can also control access to the **data plane** of Azure resources. The Storage Blob Data Reader role grants you access to read the data stored in the storage account.

The role definition syntax allows you to use wildcards. Typically, they are used to cover all operations of a Resource Provider, such as `Microsoft.Web/*`. But the wildcard can also be used to replace any other part of the Resource Provider operation path. In fact, the Owner and Reader roles are defined using this syntax.

Built-In Roles

There are several **built-in RBAC roles** available in Azure. Each Azure Product Group is responsible for designing, threat modeling, and updating their built-in roles whenever they introduce new functionality (new operations to their Resource Providers). I call these service-specific built-in roles. In addition, there are some general built-in roles that you should look at more closely, especially at the beginning of your Azure RBAC implementation.

General Built-In Roles

Owner grants you access to manage any operations in any Resource Provider. This includes the Resource Provider `Microsoft.Authorization`. In practice, this role lets you create, read, update, and delete any resource or RBAC assignment. The Owner role is a powerful role, and its usage should be minimized. Owner is the default role for any new subscriptions. The role definition at Listing 2-1 describes the explicit permissions of this role.

Listing 2-1. Permissions of the Owner built-in role

```
"permissions": [
    {
      "actions": [
        "*"
      ],
      "notActions": [],
      "dataActions": [],
      "notDataActions": []
    }
  ]
```

User access admin grants you access to the `Microsoft.Authorization` Resource Provider. Whenever possible, you should replace your justified Owner assignments with assignments to User Access Administrator. The role definition at Listing 2-2 describes the explicit permissions of this role.

Listing 2-2. Permissions of the User Access Administrator built-in role

```
"permissions": [
    {
      "actions": [
        "*/read",
        "Microsoft.Authorization/*",
        "Microsoft.Support/*"
      ],
      "notActions": [],
      "dataActions": [],
      "notDataActions": []
    }
  ]
```

Note Azure does not prevent you from creating new Owner and User Access Administrator role assignments, even while you have been assigned to either of the roles only temporarily using Privileged Identity Management. Any RBAC assignments you assign directly will stay in effect until explicitly removed.

Contributor grants you access to create, read, update, or delete any resource. When it comes to access control, the Contributor differs from the Owner role. The Contributor role has only read access to the `Microsoft.Authorization` Resource Provider. In practice, the Contributor is the most highly privileged role you should assign to developers. Once your RBAC practices mature, you should start replacing your Contributor assignments with either service-specific built-in roles or Contributor roles assigned in lower scopes. The role definition in Listing 2-3 describes the explicit permissions of this role.

Listing 2-3. Permissions of the Contributor built-in role

```
"permissions": [
    {
      "actions": [
        "*"
      ],
      "notActions": [
        "Microsoft.Authorization/*/Delete",
        "Microsoft.Authorization/*/Write",
        "Microsoft.Authorization/elevateAccess/Action",
        "Microsoft.Blueprint/blueprintAssignments/write",
        "Microsoft.Blueprint/blueprintAssignments/delete",
        "Microsoft.Compute/galleries/share/action"
      ],
      "dataActions": [],
      "notDataActions": []
    }
  ]
```

Reader role grants the `/read` action to any Resource Provider. In practice, this role has access to read control plane data, but not data plane. This means that the Reader role does not let you read the content of your resources, such as files stored in a storage account.

Listing 2-4. Permissions of the Reader built-in role

```
"permissions": [
    {
      "actions": [
        "*/read"
      ],
      "notActions": [],
      "dataActions": [],
      "notDataActions": []
    }
  ]
```

Service-Specific Built-In Roles

To grant access to manage data stored in a storage account, you should use one of the built-in service-specific roles for storage.[9] For read access, you should assign the Storage Blob Data Reader role, which contains the permissions defined in Listing 2-5.

Listing 2-5. Permissions for the Storage Blob Data Reader role

```
"permissions": [
    {
      "actions": [
        "Microsoft.Storage/storageAccounts/blobServices/containers/read",
        "Microsoft.Storage/storageAccounts/blobServices/
        generateUserDelegationKey/action"
      ],
```

[9] https://docs.microsoft.com/en-us/rest/api/storageservices/authorize-with-azure-active-directory#manage-access-rights-with-rbac

```
      "notActions": [],
      "dataActions": [
        "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/
        read"
      ],
      "notDataActions": []
    }
  ]
```

Custom Roles

In addition to built-in roles, it is possible to define role assignments yourself. Custom roles are defined using the same syntax as built-in roles. They can also be assigned to the same identities or scopes as any built-in roles. Once you know the specific Resource Provider actions needed for your use cases, it is tempting to start replacing built-in role assignments with custom roles. While this presents the opportunity for granular access control and the application of the principle of least privilege, I recommend you consider custom roles only in exceptional cases.

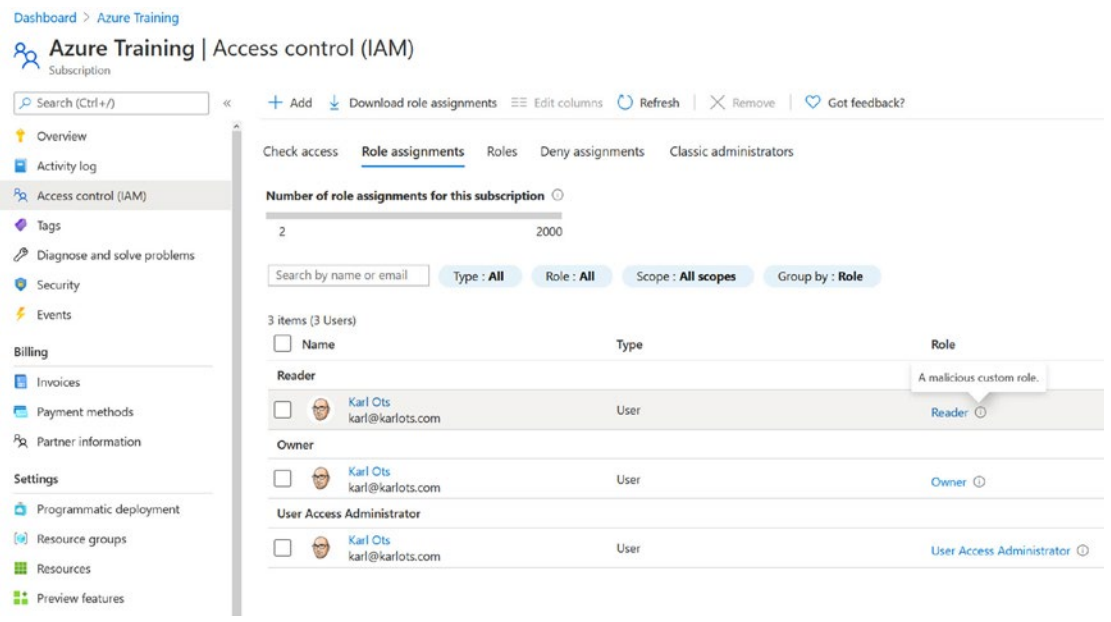
You are responsible for threat modeling and updating your custom roles. Whenever there is new functionality release to the Resource Providers, you need to re-evaluate whether your role definition is still resulting in the expected behavior. Tracking Resource Provider changes remains your responsibility, as there are no out-of-the-box alerts available for them.

Custom roles might present you with new governance challenges. Your organization needs to establish change management processes for any custom roles introduced to your Azure environment to maintain consistency on naming and control the number of roles present in your environments. There are even hard limitations on the number of custom roles you can have deployed simultaneously.[10]

[10] https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits#azure-role-based-access-control-limits

Account Manipulation

Uncontrolled use of custom roles could leave you vulnerable to account manipulation.[11] Custom role names can be tricked to use names already used by built-in roles as demonstrated in Figure 2-4.



Number of role assignments for this subscription

2 2000

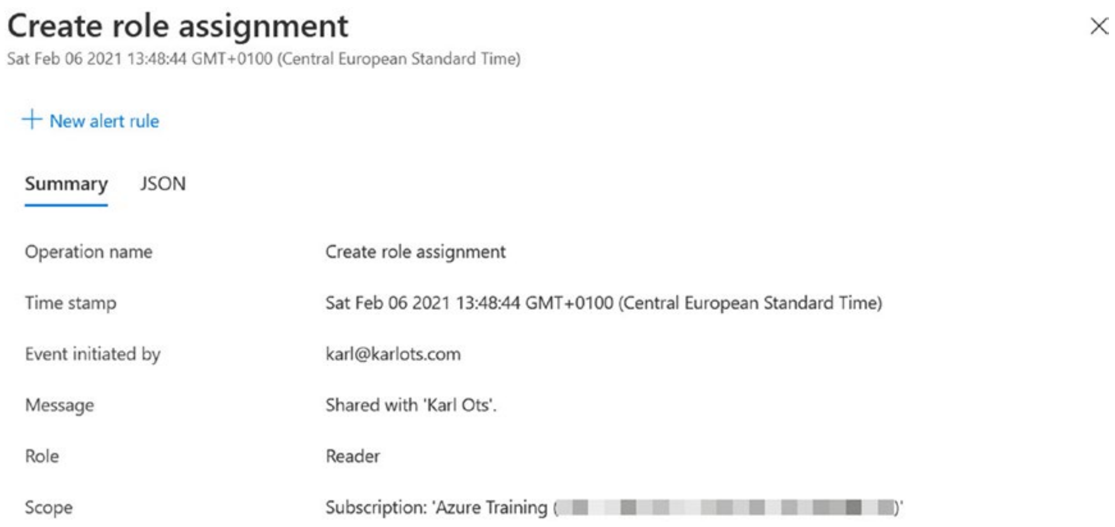
3 items (3 Users)

Name	Type	Role
Reader		
Karl Ots karl@karlots.com	User	Reader (A malicious custom role.)
Owner		
Karl Ots karl@karlots.com	User	Owner
User Access Administrator		
Karl Ots karl@karlots.com	User	User Access Administrator

Figure 2-4. *A malicious custom role assigned with an innocent-looking built-in role name*

Role assignments to custom roles can look the same as assignments to built-in roles. Figure 2-5 shows how the role assignment looks indistinguishable from a role assignment to a built-in role in the Azure activity logs.

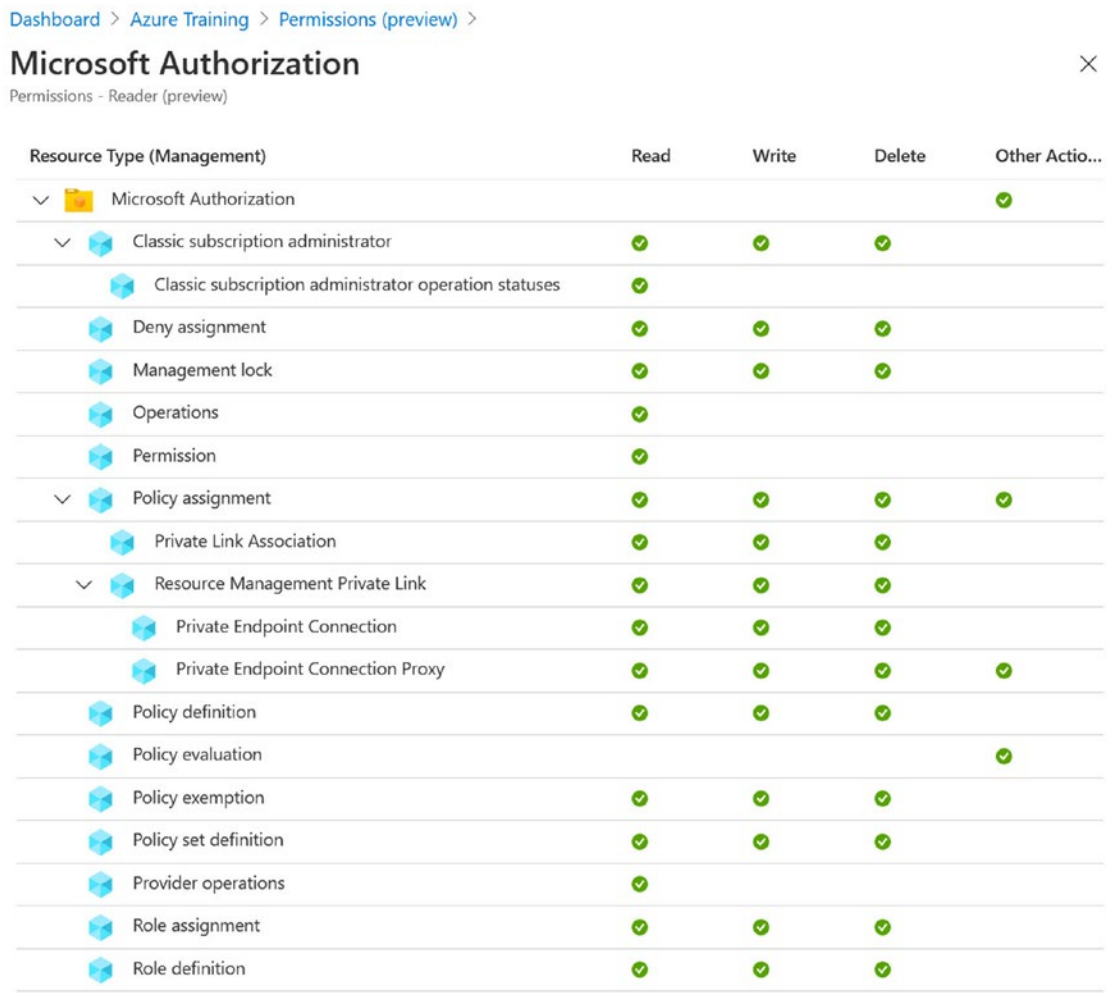
[11] https://attack.mitre.org/techniques/T1098/


Create role assignment
Sat Feb 06 2021 13:48:44 GMT+0100 (Central European Standard Time)

Operation name	Create role assignment
Time stamp	Sat Feb 06 2021 13:48:44 GMT+0100 (Central European Standard Time)
Event initiated by	karl@karlots.com
Message	Shared with 'Karl Ots'.
Role	Reader
Scope	Subscription: 'Azure Training ( )'

Figure 2-5. *Activity log entry of a role assignment to a custom role*

You should train your users and analysts to detect custom roles. Users with the built-in Reader role can enumerate the permissions for any role with the `az role definition list` command. Figure 2-6 shows the similar view in the Azure Portal. In this example, the malicious custom role has effectively the same permission as the built-in Owner role. The built-in Reader role does not allow any Write, Delete, or Other Actions.


Dashboard > Azure Training > Permissions (preview) >

Microsoft Authorization

Permissions - Reader (preview)

Resource Type (Management)	Read	Write	Delete	Other Actio...
Microsoft Authorization				✓
Classic subscription administrator	✓	✓	✓	
Classic subscription administrator operation statuses	✓			
Deny assignment	✓	✓	✓	
Management lock	✓	✓	✓	
Operations	✓			
Permission	✓			
Policy assignment	✓	✓	✓	✓
Private Link Association	✓	✓	✓	
Resource Management Private Link	✓	✓	✓	
Private Endpoint Connection	✓	✓	✓	
Private Endpoint Connection Proxy	✓	✓	✓	✓
Policy definition	✓	✓	✓	
Policy evaluation				✓
Policy exemption	✓	✓	✓	
Policy set definition	✓	✓	✓	
Provider operations	✓			
Role assignment	✓	✓	✓	
Role definition	✓	✓	✓	

Figure 2-6. *Enumerating permissions of a custom role in the Azure Portal*

EXERCISE

Consider the Azure environment described in Figure 2-7.

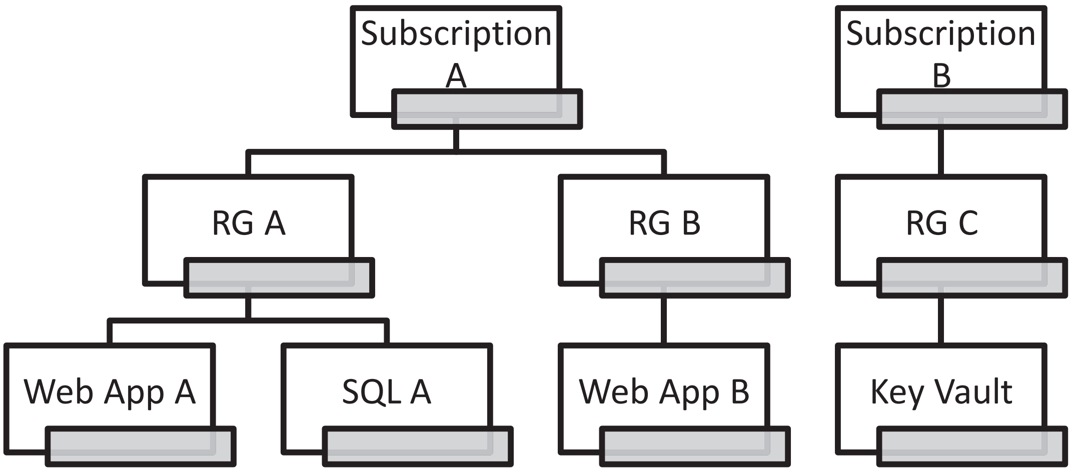


***Figure 2-7.** RBAC inheritance*

You need to grant access to two Azure developers while ensuring the access is granted following the principle of least privilege:

1. Developer A needs to create, read, update, and possibly delete resources in resource group A.
2. Developer B needs to configure the settings of Web App B to read data stored in the Key Vault.
3. Web App B should be able to read data from SQL A.

Your answer should include both Azure role-based access control assignments and data-plane access control.

Advanced: How could you use policies and locks to further limit the access control of this environment?

Azure AD Administrative Roles

In addition to RBAC roles, you also need to consider whether your users in Azure need any Azure AD administrative roles. Azure AD roles do not have granular scopes, but rather provide elevated access across the whole directory. Key roles to consider from Azure perspective are

- **Application Administrator,** which grants the user access to manage enterprise applications, application registrations, and application proxy settings
- **Application Developer**, which grants the user access to create enterprise applications and application registrations. Users assigned to this role are automatically added as owners of the application they create. Applicable regardless of the value of the *Users can register applications* setting
- **Guest Inviter**, which grants the user access to invite Azure AD B2B Guest users. Only applicable when the directory user setting *Members can invite* is set to No

Note Assigning a user to the Application Administrator role gives them the ability to impersonate an application's identity, which can lead to an elevation of privilege.[12]

Assignment Life cycle

Azure RBAC assignment life cycle can vary based on your governance requirements and cloud security approach. RBAC assignments are perpetual, that is, they last throughout the life cycle of the scope they are assigned to. This can add complexity when evaluating appropriate roles and considering scope inheritance.

[12] https://docs.microsoft.com/en-us/azure/active-directory/roles/permissions-reference#application-administrator

Note You cannot edit RBAC assignments. Instead, you would need to delete the assignment you wish to modify and create a new one.

You have three main options when considering RBAC assignment life cycles.

The most common approach is to create **static RBAC assignments to each user** directly. This is the default approach and provides your application teams with the greatest agility. This agility comes with a cost of complexity and lack of central control, however. With this approach, your application development teams can manage the access themselves, bypassing your existing access management processes.

The next option would be to use **static RBAC assignments to AAD groups** with dynamic group membership assignments. In this approach, you pre-create RBAC assignments and accompanying AAD groups. Instead of making changes to the Azure RBAC assignments, access is granted by adding the users as members of the pre-assigned AAD groups. This is the likely approach when you want to continue utilizing your existing identity and access management concepts, processes, and tools. There are downsides to this approach, however. This approach requires a significant amount of upfront planning, in contrast to the perceived agility of cloud adoption. This approach also requires you to create a large number of Azure AD groups, which could result in reaching Azure AD service limits in larger enterprises.

The third option you have is to use **dynamic RBAC assignments using Privileged Identity Management** (PIM). This lets you minimize both the number of RBAC assignments and the need to create AAD groups per scope and per role. Managing access with PIM allows you to assign **just-in-time access** to your users: PIM access is automatically removed by the PIM system after the approved time.

Policies

Policies are an additional option for controlling allowed Resource Provider actions within the same **scope** as RBAC assignments. Once you are authenticated with Azure AD, and authorized to perform certain Resource Provider actions, your request is sent to Azure Resource Manager. Before the Azure Resource Manager sends your request to a respective Resource Provider, the request is filtered through the policy engine, which checks whether any policies are in effect for the scope you want to perform your action. Figure 2-8 illustrates the relationship between Azure RBAC roles and policies.

Note Even if the role you have been assigned grants you the permission to perform the requested action, a policy can still deny you from performing it. Even an Owner role assignment does not allow for changes denied by policies.

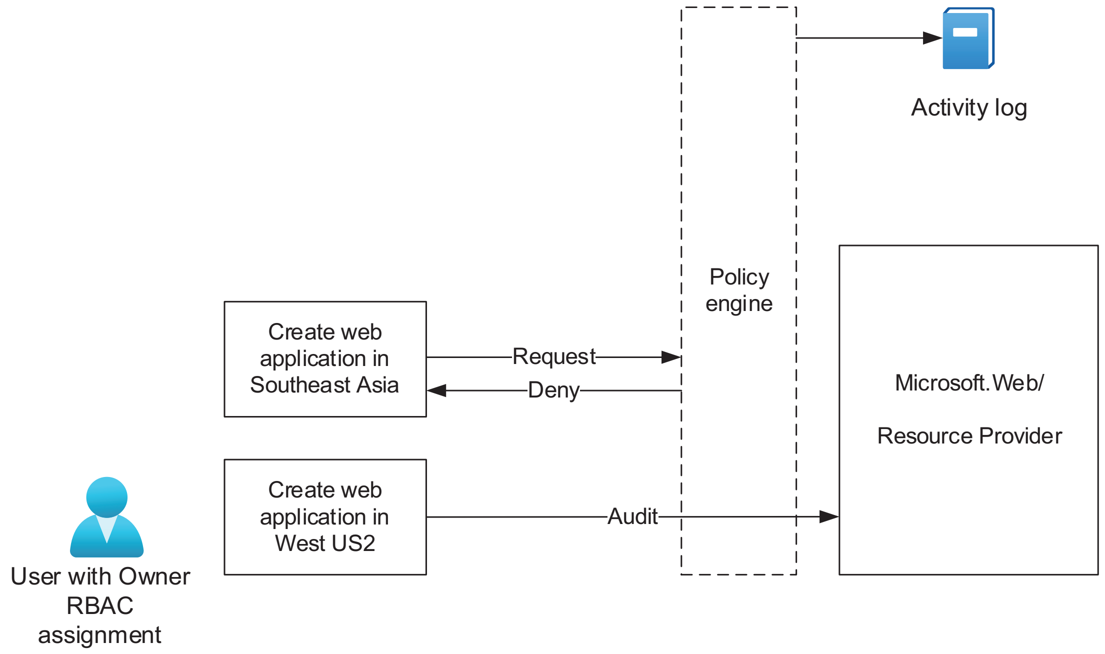

Activity log
Policy engine
Create web application in Southeast Asia
Request
Deny
Microsoft.Web/
Resource Provider
Create web application in West US2
Audit
User with Owner RBAC assignment

Figure 2-8. *Owner attempting to create a web app within a scope where Allowed Locations policy is in effect*

Policy Effects

A policy effect determines what happens when the policy is applied. As of the writing of this book, the following policy effects are supported:

- Append
- Audit
- AuditIfNotExists
- Deny
- DeployIfNotExists

- Disabled
- Modify

In addition to complementing access control, policies are one of the key Azure features used for monitoring and enforcing compliance. Several governance goals, such as geo-compliance, tagging, and service catalogues, can be achieved with policies. As we learn in Chapter 3, policies are also one of the core tools used for detection and monitoring.

Like RBAC assignments, policies are assigned to a **scope** and the same inheritance rules apply. Contrary to RBAC assignments, however, policies support exceptions to the scope. This allows you to assign policies in high scopes of the inheritance hierarchy to enforce controls across your whole Azure environment.

Locks

In addition to RBAC assignments and policies, you can control access to Azure resources using **locks**. Locks are recommended to prevent from accidental misuse of resources. Like RBAC assignments and policies, locks are assigned to a **scope**. Locks should not be considered effective controls against malicious users, as the locks themselves can be removed by Owners or User Access Administrators of the scope they are assigned in. There are two types of locks:

- **Delete** locks allow authorized users to perform Create, Read, and Update operations, but not Delete.
- **Read-only** locks allow authorized users to perform only Read operations. This lock effectively turns all RBAC assignments into Reader for the scope it is set in.

Note The Read-only lock also applies to the ***Microsoft.Authorization*** Resource Provider, effectively preventing you from making changes to RBAC assignments.[13]

---

[13] https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/lock-resources#considerations-before-applying-locks

Managed Identities and Service Principals

A common challenge when building cloud applications is how to securely manage the credentials of your application code for authenticating to other applications, APIs, or even Azure itself. The use cases may include

- Front-end to back-end access
- API to API access
- Resource deployment access

Managed identities for Azure resources automatically create and manage an identity in your Azure AD for your Azure resources without exposing the used credentials to you. Your application can use this identity to authenticate any service that supports Azure AD authentication.

If a malicious actor is able to access the authentication credentials of a service principal, they are able to use it anywhere without MFA. As of the writing of this book, service principal authentication bypasses any conditional access policies. Microsoft has communicated that they are working on enabling policies based on IP address range but have not committed to a timeline.

If you need to use service principals, use certificate credentials for authentication, instead of password credentials (client secrets).

Access Control Throughout the Secure Development Life cycle

What is the appropriate level of access to Azure environment for developers? The answer to this question spans from meaningful developer access in isolated sandbox environments to appropriate access in production.

Developer Sandbox Access

To enable faster time to market and agile evaluation of new cloud technologies, developers need a level of freedom that would have been unthinkable before the era of the cloud.

In staging and production environments, any infrastructure and configuration changes should be done through infrastructure as code, which can be templatized, version controlled, and tested. Before the developers know what to build, however, they need access to an environment with less restrictions where they can explore the available services. For the purposes of this book, I am calling these **sandbox environments**.

A sandbox environment in Azure can be as simple as a dedicated Azure subscription or a resource group. It might not necessarily differ from your other environments in any technical implementation. It is relatively easy and cost-effective to create Azure sandbox environments, but you should beware of lateral movement attacks! Even if there are no shared resources such as databases or network connectivity, your sandbox environments might still be lined to the same Azure AD tenant, allowing for account discovery attacks that are not limited to the sandbox environment.

Note Beware of account discovery, lateral movement, and denial of wallet attacks when managing developer sandbox access in Azure!

Your options for creating developer sandbox environments in Azure are summarized in Table 2-1.

Table 2-1. *Options for Azure sandbox environments*

Option name	Sandbox scope	RBAC roles	Azure AD roles
Business as usual	Subscription	Direct assignment to Contributor	AAD administrative roles according to standard processes Application registrations according to standard processes
Cost-optimized business as usual	Subscription with Dev/Test Offer type	PIM eligibility assignment to Contributor	AAD administrative roles according to standard processes Application registrations according to standard processes

(*continued*)

Table 2-1. (*continued*)

Option name	Sandbox scope	RBAC roles	Azure AD roles
Shared sandbox	Resource group in a subscription with a Dev/Test offer type	Direct assignment to Contributor	AAD administrative roles according to standard processes Application registrations according to standard processes

Continuous Deployment Pipeline Access

Most deployments to your Azure environments are not performed by a developer accessing Azure using their own credentials directly. Instead, the most common model for deploying to production would be to use a continuous deployment pipeline that performs the resource deployment in a controlled way. Centralized deployments to Azure can be done declaratively using various infrastructure-as-code template methods, or imperatively using Azure PowerShell or Azure CLI.

Figure 2-9 illustrates how the developers would typically delegate the deployment access to the continuous deployment pipeline. The benefit of this approach is that any changes to the application resources need to go through the continuous deployment pipeline. This method is often accompanied with a controlled execution of static and dynamic security testing and approval gates.

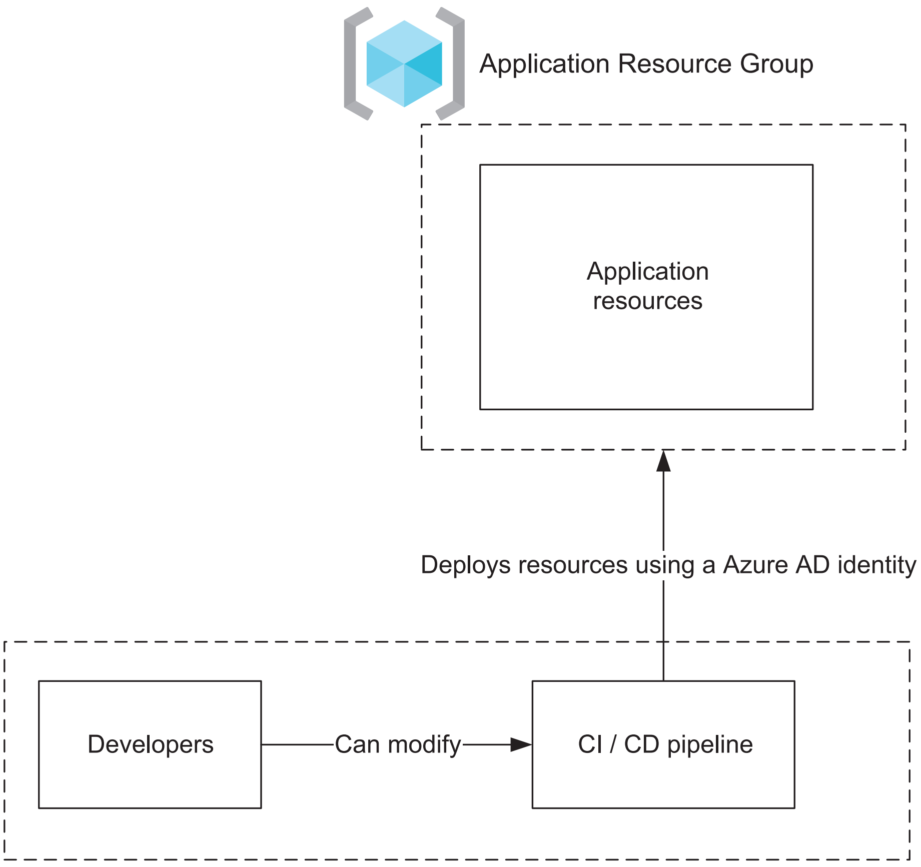

Application Resource Group
Application resources
Deploys resources using a Azure AD identity
Developers
Can modify
CI / CD pipeline

Figure 2-9. *Continuous deployment pipeline access to Azure*

To secure this approach, you need to control the identity used by the continuous deployment pipeline. As this is a system account, you can use either a managed identity or a service principal. If your deployment agent is running in Azure, you should use managed identities. If you are not able to use managed identities, you need to control access to the continuous deployment pipeline in a manner that prevents the developers accessing the service principal authentication credentials as this would let them impersonate the continuous deployment pipeline and deploy changes to the Azure resources without going through the security controls of your pipeline. You will also need to control access to modifying the continuous deployment pipeline.

Summary

In this chapter, you learned about Azure authentication and authorization models and how your approach will influence your cloud security architecture.

We looked more closely at Azure role-based access control and discussed how you can use it across the entire cloud application life cycle.