Implement Azure security

└ **3.1 Implement user authentication and authorization**

└ **3.1.1 Authenticate and authorize users by using the Microsoft Identity platform**

1. What is the Microsoft Identity Platform?
2. How do you register an application with Microsoft Entra ID (formerly Azure AD)?
3. What is the difference between single-tenant and multi-tenant apps?
4. What authentication flows are supported in the Microsoft Identity Platform?
5. How do you implement authentication using MSAL?
6. How do you configure permissions (scopes) and consent?
7. How do you acquire and validate access tokens?
8. How do you secure an API using the Microsoft Identity Platform?
9. How do you configure redirect URIs and reply URLs?
10. What is the difference between delegated and application permissions?

---

1. What is the Microsoft Identity Platform?

A Microsoft authentication system that provides OAuth 2.0 and OpenID Connect protocols for authenticating users and securing APIs. It integrates with Microsoft Entra ID (Azure AD).

---

2. How do you register an application with Microsoft Entra ID?

Use Azure Portal → Entra ID → App registrations → New registration.

Set a name, supported account types, and redirect URI. Save the Application (client) ID.

---

3. What is the difference between single-tenant and multi-tenant apps?

- Single-tenant: Only users in one Entra ID tenant can access the app.
- Multi-tenant: Users in any Entra ID tenant can authenticate.

---

4. What authentication flows are supported in the Microsoft Identity Platform?

- Authorization Code (interactive user login)
- Client Credentials (daemon apps)
- Device Code (devices without browser)
- ROPC (username/password; not recommended)
- On-Behalf-Of (service-to-service delegation)

---

5. How do you implement authentication using MSAL?

Use the Microsoft Authentication Library (MSAL) to acquire tokens:

```
var result = await app.AcquireTokenInteractive(scopes).ExecuteAsync();
```

MSAL handles caching, token renewal, and multiple flows.

---

6. How do you configure permissions (scopes) and consent?

Define scopes in the app registration under Expose an API.

- Admins or users must consent to scopes (e.g., user.read).
- API permissions tab controls delegated vs. application scopes.

---

7. How do you acquire and validate access tokens?

Use MSAL to acquire tokens (e.g., AcquireTokenInteractive, AcquireTokenForClient).

Validate tokens in the API using middleware (e.g., ASP.NET JwtBearerOptions) and Microsoft identity metadata endpoint.

8. How do you secure an API using the Microsoft Identity Platform?

- Register the API as an application.
- Define scopes under "Expose an API".
- Protect routes using [Authorize] and validate tokens using middleware (AddAuthentication().AddJwtBearer()).

---

9. How do you configure redirect URIs and reply URLs?

Set in Azure Portal → App registration → Authentication.

- Must match what's used in your app exactly.
- Used during OAuth flows to redirect users back after authentication.

---

10. What is the difference between delegated and application permissions?

- Delegated: Act on behalf of a user. Used with signed-in users.
- Application: Act as the app itself. Used in background services (e.g., daemons).

Implement Azure security

└ 3.1 Implement user authentication and authorization

└ 3.1.2 Authenticate and authorize users and apps by using Microsoft Entra ID

1. What is Microsoft Entra ID and how is it used in authentication and authorization?
2. What are managed identities and when should you use them?
3. How do you assign roles to users and apps in Entra ID?
4. How do you implement role-based access control (RBAC)?
5. How do you use Microsoft Graph to check user roles or group membership?
6. How do you authenticate using client credentials (app-only access)?
7. How do you configure an app to use a managed identity?
8. How do you restrict access to Azure resources using Entra ID?
9. How do you authorize apps to access APIs on behalf of a user?
10. What are best practices for securing app access via Entra ID?

---

1. What is Microsoft Entra ID and how is it used in authentication and authorization?

Microsoft Entra ID (formerly Azure AD) is Microsoft's cloud-based identity service.

- Authentication: Verifies user or app identity.
- Authorization: Controls access via roles, groups, or policies to Azure and custom resources.

---

2. What are managed identities and when should you use them?

System- or user-assigned identities created in Entra ID for Azure resources (e.g., App Service, Functions).
Use them to authenticate without secrets when calling Entra-secured resources like Key Vault or Graph.

---

3. How do you assign roles to users and apps in Entra ID?

- Go to the Azure resource → Access control (IAM) → Add role assignment.
- Assign roles (e.g., Reader, Contributor) to users, groups, or service principals.
  Use az role assignment create to script this.

---

4. How do you implement role-based access control (RBAC)?

Use Entra ID roles (built-in or custom) and assign them to identities.
Access is enforced based on assigned role scopes (e.g., resource group, subscription).

---

5. How do you use Microsoft Graph to check user roles or group membership?

Use Graph API endpoint /me/memberOf or /users/{id}/getMemberGroups.
Requires Group.Read.All or similar delegated/app permission.
Example:

```
GET https://graph.microsoft.com/v1.0/me/memberOf
```

---

6. How do you authenticate using client credentials (app-only access)?

Register the app in Entra ID → Generate a client secret or certificate → Grant API permissions.
Use MSAL or REST to request a token with client_id, client_secret, tenant_id, and scope.
Flow: OAuth 2.0 client credentials grant.

---

7. How do you configure an app to use a managed identity?

- Enable system-assigned identity in the Azure resource (App Service, Function, VM).
- Assign RBAC role to that identity (e.g., Key Vault Reader).
- Access tokens via Azure SDK's DefaultAzureCredential or IMDS endpoint.

8. How do you restrict access to Azure resources using Entra ID?
Use RBAC:

- Assign specific roles (e.g., Reader) to Entra identities.
- Scope can be subscription, resource group, or individual resource.
- Enforced via Entra token claims and role assignments.

---

9. How do you authorize apps to access APIs on behalf of a user?
Use delegated permissions via OAuth 2.0 authorization code flow.
The app receives an access token with the user's identity.
Ensure scopes like User.Read are consented to during sign-in.

---

10. What are best practices for securing app access via Entra ID?

- Use managed identity instead of storing secrets.
- Assign minimum required RBAC roles.
- Use conditional access policies where applicable.
- Validate token issuer, audience, and scopes in APIs.

Implement Azure security

└ 3.1 Implement user authentication and authorization

└ 3.1.3 Create and implement shared access signatures

1. What is a Shared Access Signature (SAS)?
2. What are the types of SAS and when should each be used?
3. How do you create a SAS using Azure Storage SDK or CLI?
4. What permissions can be specified in a SAS token?
5. How do you specify expiration, allowed IPs, and protocols in a SAS?
6. What is the difference between service SAS and account SAS?
7. How do you restrict SAS access by resource type?
8. How do you implement stored access policies?
9. What are security best practices for using SAS?
10. How do you revoke a SAS token?

---

1. What is a Shared Access Signature (SAS)?

A SAS is a signed URI that grants limited access to Azure Storage resources without exposing account keys. It defines permissions, scope, and expiry.

---

2. What are the types of SAS and when should each be used?

- User delegation SAS: Uses Azure AD credentials. Most secure.
- Service SAS: Grants access to specific resource (blob, file, etc.).
- Account SAS: Grants access to any service in the account (blob, queue, file, table). Use for broader access needs.

---

3. How do you create a SAS using Azure Storage SDK or CLI?

- CLI:

```
az storage blob generate-sas --account-name <name> --container-name <c> --name <blob> --permissions r --expiry <time>
```

- SDK: Use BlobSasBuilder in .NET or equivalent in other languages.

---

4. What permissions can be specified in a SAS token?

Depends on resource type. Examples:

- Blob: r (read), w (write), d (delete), l (list), a (add), c (create)
- Queue: r, a, u (update), p (process)

---

5. How do you specify expiration, allowed IPs, and protocols in a SAS?

In the SAS definition:

- --expiry (e.g., 2025-05-01T00:00Z)
- --ip (e.g., 168.1.5.60-168.1.5.70)
- --https-only true to restrict to HTTPS.

---

6. What is the difference between service SAS and account SAS?

- Service SAS: Grants access to a specific resource (e.g., a blob).
- Account SAS: Grants access across services (Blob, File, Queue, Table) in a storage account. Account SAS is broader and riskier if leaked.

7. How do you restrict SAS access by resource type?

Use the --resource-types parameter (for account SAS):

- s (service), c (container), o (object)
  Example:
  --resource-types sco limits access to services, containers, and objects.

---

8. How do you implement stored access policies?

Stored access policies are defined on containers and linked to SAS tokens to centrally manage expiry and permissions.

Create with:

```
az storage container policy create
```

Then reference the --policy-name in SAS generation.

---

9. What are security best practices for using SAS?

- Set short expiry times.
- Use HTTPS only.
- Restrict IP range if possible.
- Prefer user delegation SAS over account SAS.
- Avoid hardcoding SAS; store securely.

---

10. How do you revoke a SAS token?

- For account/service SAS: Rotate the storage account key.
- For stored access policy SAS: Modify or delete the policy; tokens linked to it become invalid.

Implement Azure security

└ 3.1 Implement user authentication and authorization

└ 3.1.4 Implement solutions that interact with Microsoft Graph

1. What is Microsoft Graph and what can it access?
2. How do you register an app to use Microsoft Graph?
3. What permissions are required to access Microsoft Graph?
4. How do you authenticate and call Microsoft Graph using MSAL?
5. How do you read user profile data from Microsoft Graph?
6. How do you list groups or check group membership?
7. How do you call Microsoft Graph from a background service?
8. How do you handle access token scopes and consent?
9. How do you use Graph SDK vs direct REST API?
10. What are best practices for calling Microsoft Graph securely?

---

1. What is Microsoft Graph and what can it access?

Microsoft Graph is a unified API endpoint (graph.microsoft.com) for accessing Microsoft 365 services like Entra ID (users, groups), Outlook, SharePoint, OneDrive, Teams, and more.

---

2. How do you register an app to use Microsoft Graph?

- In Azure Portal → Entra ID → App registrations → New registration
- Add API permissions for Microsoft Graph
- Optionally configure redirect URI and generate client secret or cert

---

3. What permissions are required to access Microsoft Graph?

- Delegated (signed-in user): e.g., User.Read, Mail.Read
- Application (daemon app): e.g., User.Read.All, Group.Read.All
  Some permissions require admin consent.

---

4. How do you authenticate and call Microsoft Graph using MSAL?

Acquire token via MSAL, then use HTTP or SDK.

Example (C#):

```
var result = await app.AcquireTokenForClient(scopes).ExecuteAsync();
var token = result.AccessToken;
```

---

5. How do you read user profile data from Microsoft Graph?

Use GET https://graph.microsoft.com/v1.0/me (delegated)
or GET /users/{id} (application permission).
Include access token in Authorization header:
Authorization: Bearer <token>

---

6. How do you list groups or check group membership?

- List groups: GET /groups
- Check membership: GET /me/memberOf or /users/{id}/memberOf
  Requires permissions like Group.Read.All.

7. How do you call Microsoft Graph from a background service?
Use application permissions with the client credentials flow:

- Acquire token via `AcquireTokenForClient()`
- Call Graph API using token; no user context needed.

---

8. How do you handle access token scopes and consent?
Scopes define the resources and actions an app can request.

- Delegated: Scopes like `User.Read` are granted on sign-in.
- Application: Requires admin consent via Azure Portal or admin consent URL.

---

9. How do you use Graph SDK vs direct REST API?

- SDK (e.g., Microsoft.Graph NuGet): Typed clients, fluent syntax, easier integration.
- REST: More control, immediate support for latest endpoints.
  Both use the same access tokens.

---

10. What are best practices for calling Microsoft Graph securely?

- Use least privilege scopes.
- Store secrets in Key Vault.
- Use managed identities if available.
- Validate token claims in APIs.
- Handle token caching and expiration properly.

3. Implement Azure security

└ **3.2 Implement secure Azure solutions**

└ **3.2.1 Secure app configuration data by using App Configuration or Azure Key Vault**

1. What is Azure App Configuration and when should it be used?
2. What is Azure Key Vault and when should it be used?
3. What types of secrets can be stored in Azure Key Vault?
4. How do you access secrets from Azure Key Vault in code using DefaultAzureCredential?
5. How do you integrate Azure Key Vault with App Service or Functions securely?
6. How do you use Azure App Configuration in .NET apps?
7. How do you enable Key Vault reference integration in Azure App Configuration?
8. How do you use managed identities to authenticate to Key Vault and App Configuration?
9. What are best practices for securing app settings and secrets?
10. How can you audit or monitor access to secrets in Azure Key Vault?

---

1. What is Azure App Configuration and when should it be used?

A centralized service for managing application settings and feature flags. Use it to decouple config from code across environments, especially in microservices or distributed apps.

---

2. What is Azure Key Vault and when should it be used?

A secure store for secrets, keys, and certificates. Use it for managing sensitive data (e.g., DB passwords, API keys) with RBAC and audit logging. Ideal for securing runtime secrets.

---

3. What types of secrets can be stored in Azure Key Vault?

- Secrets (e.g., passwords, connection strings)
- Keys (RSA, EC keys for encryption/signing)
- Certificates (incl. auto-renewing SSL certs)

---

4. How do you access secrets from Azure Key Vault in code using DefaultAzureCredential?

Use Azure SDK:

```
var client = new SecretClient(new Uri(kvUrl), new DefaultAzureCredential());
KeyVaultSecret secret = await client.GetSecretAsync("MySecret");
```

Requires proper RBAC role (e.g., Key Vault Secrets User) and managed identity.

---

5. How do you integrate Azure Key Vault with App Service or Functions securely?

Enable managed identity on the app, assign Key Vault Secrets User role, and reference secrets using:

```
@Microsoft.KeyVault(SecretUri=https://<vault-name>.vault.azure.net/secrets/<secret-name>/)
```

Used in app settings; no code change needed.

---

6. How do you use Azure App Configuration in .NET apps?

Install the package:

Microsoft.Extensions.Configuration.AzureAppConfiguration

Example usage:

```
builder.Configuration.AddAzureAppConfiguration(options =>
  options.Connect("<connection-string>")
      .Select("*"));
```

Use FeatureManagement for feature flags.

7. How do you enable Key Vault reference integration in Azure App Configuration?

In Azure App Configuration, add a key with a value using this format:

```
@Microsoft.KeyVault(SecretUri=https://<vault-name>.vault.azure.net/secrets/<secret-name>/)
```

Requires managed identity access to Key Vault and EnableKeyVault option in code.

---

8. How do you use managed identities to authenticate to Key Vault and App Configuration?

Enable system/user-assigned identity on the app. Assign roles:

- Key Vault: Key Vault Secrets User
- App Configuration: App Configuration Data Reader

In code, use DefaultAzureCredential to authenticate.

---

9. What are best practices for securing app settings and secrets?

- Never store secrets in code or config files
- Use managed identities with least privilege
- Reference secrets from Key Vault via environment/config
- Enable Key Vault logging and soft-delete

---

10. How can you audit or monitor access to secrets in Azure Key Vault?

Enable diagnostic settings to stream logs to Log Analytics.

Track:

- Secret access (AuditEvent)
- Failed attempts

Use Azure Monitor or Sentinel for alerting and analytics.

3. Implement Azure security

└ **3.2 Implement secure Azure solutions**

└ **3.2.2 Develop code that uses keys, secrets, and certificates stored in Azure Key Vault**

1. How do you retrieve a secret from Azure Key Vault using the Azure SDK?
2. How do you use a certificate from Key Vault in an HTTPS client or service?
3. How do you use Key Vault to perform cryptographic operations with stored keys?
4. What roles or permissions are needed to access keys, secrets, or certificates?
5. How do you handle secret rotation using Azure Key Vault?
6. What are the differences between software-protected and HSM-protected keys?
7. How do you access a certificate's private key from Azure Key Vault?
8. How do you manage access to Key Vault from an Azure Function or Web App?
9. What are best practices for using Key Vault in application code?
10. How do you configure Key Vault references in an ARM or Bicep deployment?

---

1. How do you retrieve a secret from Azure Key Vault using the Azure SDK?

```
var client = new SecretClient(new Uri(kvUrl), new DefaultAzureCredential());
KeyVaultSecret secret = await client.GetSecretAsync("MySecret");
string value = secret.Value;
```

Requires Key Vault Secrets User role and managed identity or credential.

---

2. How do you use a certificate from Key Vault in an HTTPS client or service?

Download the certificate as a PFX with private key:

```
var certClient = new CertificateClient(new Uri(kvUrl), new DefaultAzureCredential());
KeyVaultCertificateWithPolicy cert = await certClient.GetCertificateAsync("MyCert");
var x509 = new X509Certificate2(cert.Cer);
```

For private key use, export from a secret or use GetSecretAsync with content type application/x-pkcs12.

---

3. How do you use Key Vault to perform cryptographic operations with stored keys?

Use CryptographyClient:

```
var cryptoClient = new CryptographyClient(new Uri(keyId), new DefaultAzureCredential());
EncryptResult result = await cryptoClient.EncryptAsync(EncryptionAlgorithm.RsaOaep, data);
```

Key must allow crypto operations (e.g., encrypt, sign).

---

4. What roles or permissions are needed to access keys, secrets, or certificates?

- Secrets: Key Vault Secrets User
- Keys: Key Vault Crypto Service Encryption User, Key Vault Key User
- Certificates: Key Vault Certificates Officer
  Use RBAC or Key Vault access policies (legacy).

---

5. How do you handle secret rotation using Azure Key Vault?

- For manual rotation: update secret value and update app references.
- For automatic rotation (certs): configure lifetimeAction in certificate policy.
- Enable soft-delete and purge protection for rollback and audit.

6. What are the differences between software-protected and HSM-protected keys?

- *Software-protected*: Stored and processed in software; suitable for general use.
- *HSM-protected*: Backed by FIPS 140-2 Level 2+ compliant Hardware Security Modules; use for high-security needs like compliance-bound apps.

---

7. How do you access a certificate's private key from Azure Key Vault?

Download as a secret in PFX format:

```
var secret = await secretClient.GetSecretAsync("MyCert");
var certBytes = Convert.FromBase64String(secret.Value);
var cert = new X509Certificate2(certBytes, (string)null, X509KeyStorageFlags.Exportable);
```

Ensure certificate is imported with the private key.

---

8. How do you manage access to Key Vault from an Azure Function or Web App?

- Enable system-assigned identity
- Assign appropriate RBAC role (e.g., Key Vault Secrets User)
- Use DefaultAzureCredential in app code for auth
  No secrets stored in config needed.

---

9. What are best practices for using Key Vault in application code?

- Use DefaultAzureCredential
- Use caching to minimize latency and throttling
- Do not log secret values
- Handle retries and transient failures with SDK policies

---

10. How do you configure Key Vault references in an ARM or Bicep deployment?

Use @Microsoft.KeyVault reference in resource parameters:

```
"mySecret": {
  "reference": {
    "keyVault": {
      "id": "[resourceId('Microsoft.KeyVault/vaults', 'my-kv')]"
    },
    "secretName": "my-secret"
  }
}
```

Used to inject secrets at deploy time into app settings or parameters.

3. Implement Azure security

└ **3.2 Implement secure Azure solutions**

└ **3.2.3 Implement Managed Identities for Azure resources**

1. What are managed identities and what problem do they solve?
2. What is the difference between system-assigned and user-assigned managed identities?
3. How do you enable a managed identity on an Azure resource?
4. How do you assign RBAC roles to a managed identity?
5. How do you authenticate to Azure services using managed identities in code?
6. How do you use managed identity with Azure Key Vault?
7. How do you troubleshoot managed identity access issues?
8. How do managed identities behave during resource deletion or scaling?
9. What services support managed identities?
10. What are best practices when using managed identities in cloud apps?

---

1. What are managed identities and what problem do they solve?

Managed identities provide Azure-hosted identities for applications to access Azure resources securely without storing credentials in code or config.

---

2. What is the difference between system-assigned and user-assigned managed identities?

- *System-assigned*: Tied to the resource lifecycle; deleted with the resource.
- *User-assigned*: Standalone; reusable across multiple resources; managed separately.

---

3. How do you enable a managed identity on an Azure resource?

Via Azure CLI:

```
az webapp identity assign --name <app-name> --resource-group <rg>
```

Or in ARM/Bicep: identity: { type: 'SystemAssigned' }

---

4. How do you assign RBAC roles to a managed identity?

Use Azure CLI:

```
az role assignment create \
  --assignee <clientId-or-objectId> \
  --role <role-name> \
  --scope <resource-scope>
```

---

5. How do you authenticate to Azure services using managed identities in code?

Use DefaultAzureCredential from Azure SDK:

```
var client = new SecretClient(new Uri(kvUrl), new DefaultAzureCredential());
```

Automatically uses the managed identity of the running resource.

---

6. How do you use managed identity with Azure Key Vault?

1. Enable managed identity on the resource
2. Assign Key Vault Secrets User role to the identity at Key Vault scope
3. Use DefaultAzureCredential in code to access secrets

7. How do you troubleshoot managed identity access issues?

- Verify identity is enabled
- Confirm role assignment at correct scope
- Check az role assignment list and Key Vault diagnostics logs
- Ensure DefaultAzureCredential is used correctly in code

---

8. How do managed identities behave during resource deletion or scaling?

- *System-assigned*: Deleted when the resource is deleted
- *User-assigned*: Must be manually managed; survives resource deletion
  Scaling (e.g., in App Service) automatically reuses the same identity

---

9. What services support managed identities?

Supported in:

- App Service, Functions
- VMs, VMSS
- Logic Apps
- Azure Container Apps
- Azure Kubernetes Service (AKS)
- Azure Data Factory, and more

---

10. What are best practices when using managed identities in cloud apps?

- Prefer system-assigned for single-resource use
- Use user-assigned for cross-resource or lifecycle-independent needs
- Always use RBAC for access control
- Avoid storing credentials; rely on identity + DefaultAzureCredential