Authorization

In this document I will look at 3 different ways on how you can do authorization in C# and which would be the best for my solution.
I used the information following Bascos (2024) , Yash (2024), Tutorials (2023) and Rick-Anderson (n.d.).

Role based

Role-Based Authorization is a widely used method where users are assigned roles, and each role has predefined permissions. This approach is simple and effective for many applications.

How role based works

You first define some roles. Roles represent a collection of permissions. Common examples include Admin, User, and Manager.
Then you add the roles to an identity like what it can do thus privileges.
With that done you add the roles to the database and assign them to the users. With that done you can finish with adding the attribute above the method like:

```
[Authorize(Roles = "Admin")]
public IActionResult function()
{
    return ok();
}
```

Why choose role based

Role-based is scalable, easy to manage and implement and is generally fast and efficient. While there is no fine-control and thus limiting for complex systems. Another minus point is that due to new permissions needed there can be a role explosion.

Permission based

Permission-Based Authorization offers a more granular approach by assigning specific permissions directly to users or roles. This method provides fine-tuned control over what users can do within the application.

How permission based works

you first define permissions then you create your policy provider.
Then you create the PermissionRequirement which you specify a requirement for permissions.
When that’s done you need to have a PermissionHandler to check if users have these

permissions.
Then you register them and assign users to these roles.

```
[Authorize(Policy = "Permission.ViewUsers")]
public IActionResult ViewUsers()
{
    return ok();
}
```

Why choose permission based

You have precise control about what users can do. It will be easier to manage complex permissions without creating numerous roles. It also Reduces the risk of over-permission users. The few problems with it is that it's more complex and can introduce performance overhead due to the permission checks.

Claim based

Claims-Based Authorization uses claims to make authorization decisions. A claim is a statement about a user (e.g., name, role, age), and it's part of the user's identity. It involves checking the user's claims to make decisions, which can be more granular than role checks.

How claim based works

You first add a policy like:

```
builder.Services.AddAuthorization(options =>
{
   options.AddPolicy("EmployeeOnly", policy =>
policy.RequireClaim("EmployeeNumber"));
});
```

with that done you can add the attribute to the method like:

```
[Authorize(Policy = "EmployeeOnly")]
public IActionResult VacationBalance()
{
    return ok();
}
```

Of course between them you need to add the roles to the user to have them be able to execute the methods with your claim.

Why choose claim based

It is suitable for complex applications with dynamic or highly granular access control requirements. Suitable for scenarios where user permissions are not strictly tied to their role.

Easier to manage in large systems with complex and varying permissions. But depending on implementation, it can be more resource-intensive, as it might involve more checks and balances. It can also just be more complex to implement and understand compared to others.

Conclusion

Based on what I gathered I will choose the Role based because it needs to scale, I don’t have a lot of users needed and the fine control what he others bring is just unnecessary. I will keep the roles to a minium with at least having User, Employee and Developer with each one having more control.

Resources

Bascos, J. (2024, November 23). Securing Your .NET API: Role-Based vs. Permission-Based Authorization. Medium.

Yash. (2024, September 20). Role based vs Permission based Authorization - Yash - Medium. Medium.

Tutorials, D. N. (2023, December 6). Role Based vs Claims Based Authorization in ASP.NET Core - Dot Net Tutorials. Dot Net Tutorials.

Rick-Anderson. (n.d.). Claims-based authorization in ASP.NET Core. Microsoft Learn.