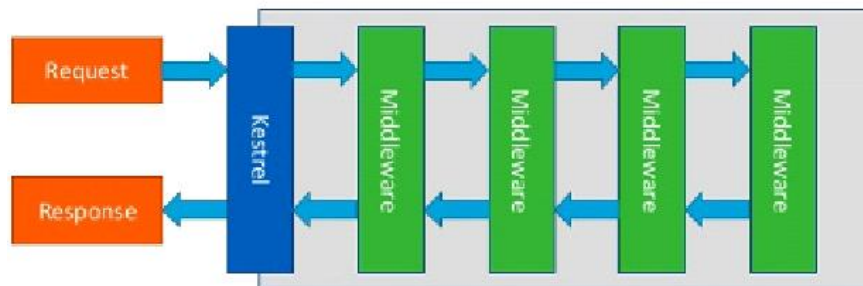
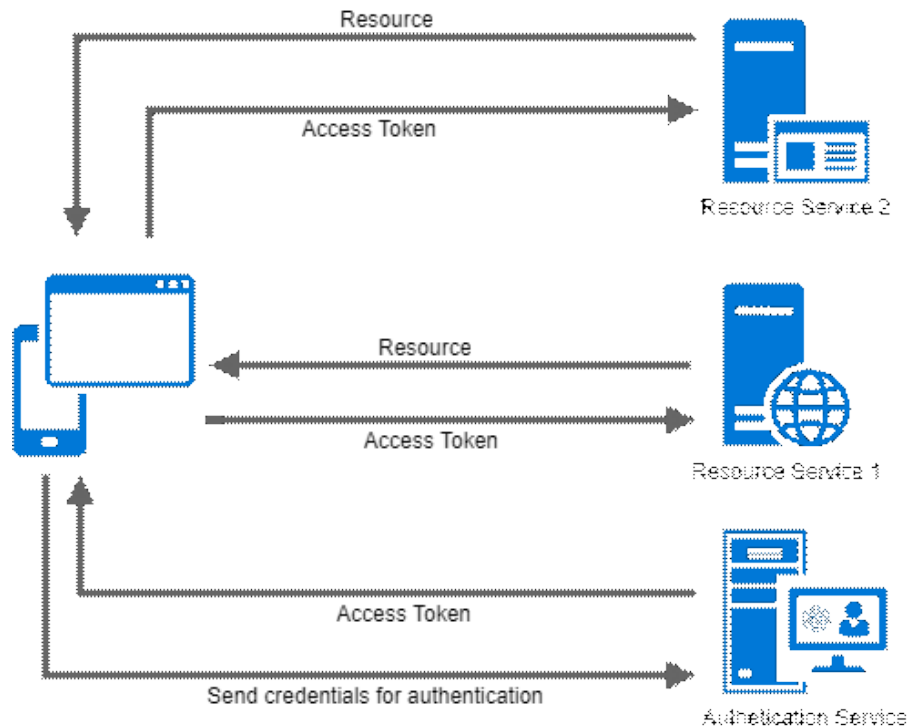


第84期-gRPC中的身份认证与授权

2020年1月19日 10:12

配置身份认证与授权中间件



```
public void Configure(IApplicationBuilder app)
{
    app.UseRouting();

    app.UseAuthentication();
    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        routes.MapGrpcService<GreeterService>();
    });
}
```

务必在 UseRouting 之后和 UseEndpoints之前调用 UseAuthentication 和 UseAuthorization 方法。

获取当前请求服务的用户标识

```
public override Task<BuyTicketsResponse> BuyTickets(
    BuyTicketsRequest request, ServerCallContext context)
{
    var user = context.GetHttpContext().User;

    // ... access data from ClaimsPrincipal ...
}
```

持有者令牌身份验证

每次请求附加头部令牌

```
public bool DoAuthenticatedCall(
    Ticketer.TicketerClient client, string token)
{
    var headers = new Metadata();
    headers.Add("Authorization", $"Bearer {token}");

    var request = new BuyTicketsRequest { Count = 1 };
    var response = await client.BuyTicketsAsync(request, headers);

    return response.Success;
}
```

基于通道的多个客户端公共配置

```
private static GrpcChannel CreateAuthenticatedChannel(string address)
{
    var credentials = CallCredentials.FromInterceptor((context, metadata) =>
    {
        if (!string.IsNullOrEmpty(_token))
        {
            metadata.Add("Authorization", $"Bearer {_token}");
        }
        return Task.CompletedTask;
    });

    // SslCredentials is used here because this channel is using TLS.
    // CallCredentials can't be used with ChannelCredentials.Insecure on non-
    TLS channels.
    var channel = GrpcChannel.ForAddress(address, new GrpcChannelOptions
    {
        Credentials = ChannelCredentials.Create(new SslCredentials(),
```

```
credentials)
    });
    return channel;
}
```

集成 gRPC 与 IdentityServer4

https://github.com/longxianghui/grpc_identityserver

<https://www.cnblogs.com/stulzq/p/11897628.html>

客户端证书身份认证

```
public Ticker.TickerClient CreateClientWithCert(
    string baseAddress,
    X509Certificate2 certificate)
{
    // Add client cert to the handler
    var handler = new HttpClientHandler();
    handler.ClientCertificates.Add(certificate);

    // Create the gRPC channel
    var channel = GrpcChannel.ForAddress(baseAddress, new GrpcChannelOptions
    {
        HttpClient = new HttpClient(handler)
    });

    return new Ticker.TickerClient(channel);
}
```

其它身份认证机制

许多支持 ASP.NET Core 的身份验证机制都适用于 gRPC:

- Azure Active Directory
- 客户端证书
- IdentityServer
- JWT 令牌
- OAuth 2.0
- OpenID Connect
- WS-Federation

使用授权机制

身份认证成功就有权限

```
[Authorize]
public class TicketerService : Ticketer.TicketerBase
{
}
```

基于策略的授权

```
[Authorize("MyAuthorizationPolicy")]
public class TicketerService : Ticketer.TicketerBase
{
}
```

也支持基于角色和资源的授权，完全与 ASP.NET Core 认证授权机制兼容，可复用，参考之间课程学习。