Verslag opdracht 1: Open Policy Agent

Deze week kregen we een opdracht waarbij we gebruik maakten van Open Policy Agent (OPA) om via een Rego-bestand te bepalen of een gebruiker toestemming krijgt om een drankje te bestellen of toe te voegen. Het doel was dat een klant (customer) die jonger is dan 16 jaar geen bier mocht bestellen, maar wel Fristi. Daarnaast mochten klanten geen drankjes toevoegen, dit was voorbehouden aan de barman (bartender).

Voor deze opdracht kregen we al een frontend Single Page Application (SPA) die communiceerde met de API. Onze taak was om Auth0 te configureren en gebruikers aan te maken, zodat we de API konden testen en controleren of onze Rego-policies correct waren.

Een Rego is een high-level programmeertaal die wordt gebruikt in OPA om policies te definiëren. In dit project schreven we Rego-policies en gebruikten we een middleware (https://github.com/build-security/OPA-AspDotNetCore-Middleware) om te bepalen of een gebruiker de juiste rechten heeft om bepaalde acties uit te voeren.

In het begin probeerde ik in “spa.js” de functies aan te passen, zodat de leeftijd van de gebruiker handmatig werd toegevoegd bij het bestellen van een drankje. Dit was een eenvoudige oplossing, maar niet in lijn met de opdracht, omdat het geen gebruik maakte van OPA en de middleware om de gebruikersinformatie uit de JWT-token te halen. Die oplossing zag er als volgt uit:

```
function orderFristi() {
    mgr.getUser().then(function (user) {
        sendRequest(settings.api_bar_uri, user.access_token, { DrinkName: "Fristi", age: user.profile.age }, log);
    });
}

function orderBeer() {
    mgr.getUser().then(function (user) {

        sendRequest(settings.api_bar_uri, user.access_token, { DrinkName: "Beer", age: user.profile.age }, log);
    });
}

function manageDrinks() {
    mgr.getUser().then(function (user) {
        sendRequest(settings.api_manageBar_uri, user.access_token, { DrinkName: "Whiskey", Role: user.profile.role[0] }, log);
    });
}
```

```
1   package barmanagement #namespace
2   default allow := false
3
4
5   allow {
6       input.request.body.DrinkName == "Beer"
7       age := to_number(input.request.body.age)
8       age >= 16
9   }
10
11  allow {
12      input.request.body.DrinkName == "Fristi"
13  }
14
15  allow {
16      input.request.body.Role == "bartender"
17  }
```

Daarna zocht ik naar een oplossing die effectief gebruik maakt van OPA en de middleware om de Rego-policy goed te implementeren. In het begin had ik moeite om de leeftijd van de gebruiker eruit halen zonder deze hardcoded toe te voegen. Het grootste probleem was dat bij de POST-requests van gebruikers, wanneer ze een drankje bestelden, alleen de naam van het drankje werd doorgegeven en geen andere informatie, zelfs niet de JWT-token. Dit gebeurde omdat in de middleware de optie om headers door te geven standaard niet was ingeschakeld, ondanks dat de documentatie van de middleware aangaf dat dit wel zo zou zijn.

4. `IncludeHeaders` : Boolean. Whether or not to pass the request headers to the policy engine. **Default is true**

Om dit probleem op te lossen, moest ik naar “Program.cs” gaan en een regel toevoegen om ervoor te zorgen dat de headers werden meegenomen in het POST-request.

```
// Add OPA integration
builder.Services.AddBuildAuthorization(options =>
{
    options.Enable = true;
    options.BaseAddress = opaBaseAddress;
    options.PolicyPath = "/barmanagement/allow";
    options.AllowOnFailure = false;
    options.IncludeBody = true;
    options.IncludeHeaders = true;
});
```

Na deze wijziging kon ik zien dat de JWT-token van de gebruiker in de POST-request werd meegegeven. Deze token kon vervolgens worden gebruikt om de leeftijd en de rol van de gebruiker te bepalen, zodat we verder konden gaan met het schrijven van de Rego-policy.

```
package barmanagement #namespace
default allow := false

import input.request.headers.Authorization

allow {
    jwt_token := extract_bearer_token(input.request.headers.Authorization)
    [jwt_header, jwt_payload, jwt_signature] := io.jwt.decode(jwt_token)
    user_age := to_number(jwt_payload.age)
    user_role := jwt_payload.role[_]
    user_role == "customer"
    input.request.body.DrinkName == "Beer"
    user_age >= 16
    input.request.method == "POST"
    input.request.path != "/api/managebar"
}

allow {
    jwt_token := extract_bearer_token(input.request.headers.Authorization)
    [jwt_header, jwt_payload, jwt_signature] := io.jwt.decode(jwt_token)
    user_role := jwt_payload.role[_]
    user_role == "customer"
    input.request.body.DrinkName != "Beer"
    input.request.method == "POST"
    input.request.path != "/api/managebar"
}
```

```
allow {
    jwt_token := extract_bearer_token(input.request.headers.Authorization)
    [jwt_header, jwt_payload, jwt_signature] := io.jwt.decode(jwt_token)
    user_role := jwt_payload.role[_]
    user_role == "bartender"
    input.request.method == "POST"
    input.request.path == "/api/managebar"
}

extract_bearer_token(auth_header) = token {
    startswith(auth_header, "Bearer ")
    token := substring(auth_header, 7, -1)
}
```

De Rego bevat drie belangrijke onderdelen: het extraheren en decoderen van de JWT-token, toegangsregels voor klanten en barmannen, en het beperken van toegang tot specifieke paden.

1. JWT-token eruit halen en decoderen
   Er wordt een functie “extract_bearer_token” gebruikt om de JWT-token uit de Authorization-header van het verzoek te halen. Vervolgens wordt de token gedecodeerd met “io.jwt.decode”, waarna de payload wordt gebruikt voor de toegangscontrole.

2. Toegangsregels voor klanten
   De eerste allow-regel in de Rego zorgt ervoor dat gebruikers met de rol “customer” alleen bier mogen bestellen als ze minstens 16 jaar oud zijn. De tweede allow-regel geeft klanten de mogelijkheid om andere drankjes (die geen bier zijn) te bestellen, ongeacht hun leeftijd.

3. Toegangsregels voor barmannen
   Gebruikers met de rol “bartender” krijgen toegang tot het beheren van de bar en mogen nieuwe drankjes toevoegen via de /api/managebar-endpoint.

4. Toegang beperken tot specifieke paden
   De policy controleert of de gebruiker toegang probeert te krijgen tot de /api/managebar-route. Alleen gebruikers met de rol "bartender" krijgen toegang tot deze route, terwijl klanten alleen drankjes mogen bestellen via andere routes.

Samenvattend zorgt deze policy ervoor dat klanten alleen drankjes kunnen bestellen (met een leeftijdscontrole voor bier), en dat alleen barmannen drankjes mogen beheren en toevoegen aan de bar. Dit alles wordt geregeld op basis van de rol en leeftijd van de gebruiker, die worden uitgelezen uit de JWT-token.

Voor de volledige oplossing van deze opdracht kun je terecht op mijn GitHub-repository:

https://github.com/Adri1574/Cyber-Security-Advanced.