Zagrożenia bezpieczeństwa systemów

Laboratorium 1

Adam Biurkowski (20567)

Prowadzący mgr Bartłomiej Kicior

Pytania dodatkowe

Czym są tokeny JWT?

JWT (ang. *JSON Web Token*) to standardowy sposób reprezentacji roszczeń (claims) między dwiema stronami w formacie JSON, który jest stosowany do przesyłania danych między klientem a serwerem. Token JWT zawiera dane użytkownika, które zostały potwierdzone i mogą być używane przez aplikację do autentykacji i autoryzacji.

Rola JWT w aplikacji:

- **Autentykacja**: Tokeny JWT są wykorzystywane do potwierdzenia tożsamości użytkownika. Po zalogowaniu użytkownik otrzymuje token, który jest następnie wysyłany przy każdym zapytaniu, co pozwala serwerowi zidentyfikować użytkownika bez potrzeby ponownego logowania.
- **Autoryzacja**: Na podstawie roli użytkownika zapisanej w tokenie, aplikacja może określić, do jakich zasobów lub funkcji użytkownik ma dostęp.

Do czego można je wykorzystać?

- Autentykacja i autoryzacja w aplikacjach webowych i mobilnych.
- Przekazywanie informacji o użytkowniku między mikroserwisami.
- Jednorazowe logowanie (SSO – Single Sign-On) w różnych aplikacjach.

Przykładowy token JWT – Skład i zawartość

Token JWT składa się z trzech części:

1.Nagłówek (Header): Zawiera informacje o typie tokena (zwykle "JWT") oraz algorytmie używanym do jego podpisania (np. HMAC SHA256, RSA).

json{

"alg": "HS256",

"typ": "JWT"

}

2.Treść (Payload): Zawiera roszczenia (claims), które są informacjami, które chcemy przekazać w tokenie. Istnieją trzy typy roszczeń:

Zarejestrowane roszczenia (np. sub – identyfikator użytkownika, exp – data wygaśnięcia).

Publiczne roszczenia (np. rolę użytkownika).

Prywatne roszczenia (np. dodatkowe informacje o użytkowniku, specyficzne dla aplikacji).

json{

"sub": "1234567890",

"name": "John Doe",

"iat": 1516239022

}

3.Podpis (Signature): Jest używany do weryfikacji integralności tokena i zapewnia, że nie został on zmodyfikowany. Podpis jest tworzony poprzez zakodowanie nagłówka i treści, a następnie podpisanie ich za pomocą klucza i algorytmu (np. HMAC SHA256).

Paczki NuGet do generowania tokenów JWT w .NET

W .NET do generowania tokenów JWT, najczęściej używa się następujących paczek NuGet:

System.IdentityModel.Tokens.Jwt:

Jest to oficjalna paczka NuGet od Microsoft, zawierająca wszystkie niezbędne klasy do generowania i weryfikowania tokenów JWT.

Charakteryzuje się pełną integracją z .NET Core i obsługą zaawansowanych funkcji, takich jak weryfikacja podpisu, walidacja tokena, i integracja z ASP.NET Core Authentication.

Microsoft.AspNetCore.Authentication.JwtBearer:

Jest to paczka, która umożliwia łatwą integrację JWT w aplikacjach ASP.NET Core. Dzięki niej możliwe jest łatwe ustawienie middleware do obsługi tokenów JWT w API.

Używana głównie do autentykacji i autoryzacji na poziomie aplikacji.

Jose.JWT:

Jest to lekka paczka do generowania i walidacji JWT. Może być używana jako alternatywa dla oficjalnej paczki System.IdentityModel.Tokens.Jwt.

Wspiera różne algorytmy kodowania i zapewnia prostszy interfejs.

Jakie informacje można zawrzeć w tokenie JWT?

W tokenie JWT można zawrzeć różne informacje (tzw. claims), w zależności od wymagań aplikacji. Przykłady to:

- **Identyfikator użytkownika (user ID)** – sub (Subject).
- **Adres e-mail** – email.
- **Role i uprawnienia** – np. role (dla autoryzacji).
- **Czas wygaśnięcia tokena** – exp.
- **Unikalny identyfikator tokena** – jti (JWT ID), aby zapobiec ponownemu użyciu tokenu.
- **Czas wydania tokena** – iat (Issued At).
- **Dodatkowe dane specyficzne dla aplikacji** – np. ustawienia aplikacji, konfiguracja sesji.

Algorytmy bezpieczeństwa do kodowania danych w JWT

W klasie Microsoft.IdentityModel.Tokens.SecurityAlgorithms dostępnych jest kilka algorytmów do podpisywania danych w JWT. Poniżej przedstawiam trzy popularne algorytmy i ich różnice:

1. **HMACSHA256 (HS256)**:
    - **Opis**: Algorytm HMAC (Hashed Message Authentication Code) oparty na funkcji SHA-256.
    - **Zalety**: Łatwy do implementacji, szybki, wystarczająco bezpieczny do większości przypadków.
    - **Użycie**: Wymaga tylko jednej tajnej zewnętrznej wartości (klucz).
2. **RS256** (RSA Signature with SHA-256):
    - **Opis**: Algorytm podpisu cyfrowego RSA wykorzystujący funkcję SHA-256.
    - **Zalety**: RSA umożliwia korzystanie z pary kluczy publiczny-prywatny, co oznacza, że klucz prywatny jest używany do podpisywania tokenów, a klucz publiczny do ich weryfikacji.
    - **Użycie**: Bezpieczniejszy niż HMAC, ponieważ klucz publiczny nie jest tajny i może być udostępniony do weryfikacji tokenów.
3. **ES256** (ECDSA with SHA-256):
    - **Opis**: Algorytm oparty na krzywych eliptycznych (ECDSA) z funkcją SHA-256.
    - **Zalety**: Bardzo szybki, wymaga mniejszych kluczy niż RSA przy tym samym poziomie bezpieczeństwa.
    - **Użycie**: Często wykorzystywany w aplikacjach mobilnych, gdzie przestrzeń i wydajność są kluczowe.

Podsumowanie

Tokeny JWT są kluczowym narzędziem w dzisiejszych aplikacjach, zapewniając bezpieczną wymianę informacji między stronami. Ich popularność wynika z łatwości użycia, wydajności oraz możliwości szyfrowania danych za pomocą różnych algorytmów. Wybór odpowiedniego algorytmu zależy od wymagań aplikacji i poziomu bezpieczeństwa, jaki chcemy zapewnić.