Alessandro Biggiogera 20047397

Relazione Applicazioni Web

Fra le scelte progettuali fatte, la più complessa è stata l'organizzazione e realizzazione del login tramite Google. Un primo approccio che ho provato è stato quello di gestirlo interamente nel frontend in C#. Per questo approccio avevo installato il pacchetto NuGet di Google ed una volta completato direttamente nel frontend generavo un token JWT firmato con la stessa chiave privata del backend (usata per il login con email e password). Ma così facendo nel frontend esponevo sia la chiave privata per il token JWT, sia il client secret usato per il login con Google. Tramite C# ci sono diversi metodi per tenere nascoste delle informazioni, come gli user-secret, le variabili d'ambiente oppure appsettings.json ma ho trovato più corretto e semplice spostare tutti i dati sensibili nel backend.
Per implementare questo secondo metodo sono partito dalla documentazione Google per l'autenticazione OAuth 2.0 (https://developers.google.com/identity/openid-connect/openid-connect), il flusso si divide in 6 punti:

1. Creare un token antifalsificazione (token CSRF)
2. Inviare una richiesta di autenticazione a Google
3. Verificato lo stato del token anti-falsificazione
4. Scambiare Code con un Token Id
5. Ottenere le informazioni dell'utente dal Token Id
6. Autenticare l'utente

Quindi dalla pagina LoginGoogle(1) generiamo il token CSRF che (2) verrà poi inviato come parametro per la richiesta GET a "https://accounts.google.com/o/oauth2/v2/auth"questa request ci farà reindirizzare alla pagina GoogleCallback che (3) si occuperà di controllare la validità del token CSRF e (4) di ricevere il Code generato da Google, questo Code è un codice di autorizzazione monouso che può essere scambiato per un access token ed id token.
Quindi il Code verrà passato al backend tramite una GET e sarà compito del backend scambiare effettivamente il Code con i Token tramite una POST ad "https://oauth2.googleapis.com/token" ed una volta ottenuto e verificato il token id posso usarlo per generare un token JWT personalizzato simile a quello ottenuto con il login tramite username e password. Una volta terminato il primo login con Google, l'utente verrà inserito nel database ma con una password casuale, dandogli la possibilità dalla home del suo account di cambiarla, in modo da avere un singolo account che supporta sia Google che email e password.
Per quanto riguarda il sistema di autorizzazione si basa sulla validazione di un token JWT, generato nel backend. Un primo approccio prevedeva l'uso di protocollo a chiave simmetrica (HS256), perciò per evitare di condividere la chiave segreta il controllo sul frontend si basava su una semplice decodifica del token, senza alcuna verifica dell'autenticità del token passato. Per evitare questo problema ho deciso di passare ad un algoritmo asimmetrico (RS256), in modo da usare la chiave privata solo nel backend e per verificare l'autenticità e integrità del token nel frontend uso la chiave pubblica.