Project 2 – Security and Privacy by Design

Introduzione

Il presente progetto è stato sviluppato da *Veljko Markovic* ed *Elisa Resch* durante il corso di **Security and Privacy by Design**, sotto commissione del docente *Angelo Consoli* (in specifico, il progetto numero 2).

L'obiettivo è realizzare un sistema IT che rispetti i criteri di **security by design** attraverso l'implementazione di un front-end, un back-end e un sistema di autenticazione e tracciabilità degli eventi.

Il sistema si basa su un database predefinito, **Chinook Database**, adattato per soddisfare le specifiche richieste del progetto. Le principali funzionalità includono:

- Autenticazione e autorizzazione basate su **JWT**.
- Gestione delle password sicura, con memorizzazione crittografata e regole di complessità.
- Session management con scadenze e refresh automatico per i token.
- Tracciabilità delle attività utente mediante logging lato back-end.
- Prevenzione di attacchi come SQL injection e crawling massivo di dati.

Tecnologie utilizzate

- **Front-end**:
  - Framework: **React** con **Vite.js**
  - Librerie: Chakra UI, React Router, Axios
- **Back-end**:
  - Linguaggio: **Node.js**
  - Framework: **Express**
  - Autenticazione: **JWT**
  - Logging: Salvataggio e tracciabilità delle attività utente.
  - Email: Integrazione con **Resend** per l'invio di email.

- **Database**:
    - Tipo: **PostgreSQL** (compatibile con altre versioni di Chinook Database come SQLite e MySQL).
    - Modifiche: Aggiunti attributi e tabelle per supportare l'autenticazione e l'autorizzazione.
- **Containerizzazione**:
    - **Docker**: Configurazione di servizi tramite docker-compose.
- **Sicurezza**:
    - Crittografia delle password (hashing con algoritmi sicuri).
    - Validazione dei dati e gestione delle eccezioni per prevenire exploit.

Funzionalità principali

Utenti

1. **Login**: Ogni utente accede con un default password iniziale modificabile.
2. **Autorizzazione**:
    - Manager: Accesso completo ai clienti.
    - Utenti normali: Accesso limitato ai propri clienti.
3. **Gestione delle password**:
    - Complessità minima (maiolica, numeri, caratteri speciali).
    - Memorizzazione crittografata e storicizzazione delle ultime 5 password.

Sistema

1. **Session Management**:
    - Scadenza dei token ogni 5 minuti.
    - Refresh automatico per i manager.
    - Logout automatico per inattività (2 minuti).

2. **Validazione dei dati**:
    - Prevenzione di SQL injection e data crawling.
3. **Tracciabilità**:
    - Logging dettagliato delle attività lato server.

Note importanti

La prevenzione di SQL injection o simili viene **già** effettuata dalla libreria *mongoose*. Anche se non stiamo utilizzando un database relazionale tradizionale come MySQL o OracleDB, ci sono ancora dei rischi per quanto riguarda MongoDB. Ci sono infatti delle variabili speciali in Mongo che, se utilizzate per scopi maligni, potrebbero creare dei danni (ad esempio **$gt**). Gli input sono tutti stati validati e sanificati, inoltre gli input vengono validati da **Zod** che mi permette in modo semplice di stabilire se un determinato input è valido (numeri, lettere ecc.) prima di avanzare la richiesta e interagire con il database.

Un'altra nota importante è quella che riguarda le richieste eccessive al server in backend. Infatti, è stato imposto un limite di **100 richieste** per **15 minuti** di tempo. Se vengono superate queste richieste, l'utente dovrà aspettare 15 minuti di tempo prima di poter riprovare di nuovo. In questo modo si evita un *"bombardamento"* di richieste al server, evitando il rallentamento (potenzialmente intenzionale).

Successivamente, siccome per scopi didattici l'applicazione non è stata pubblicata da nessuna parte, ergo non è in *produzione* ma in *sviluppo*, è **importante** notare che i file di log (generati dal backend, che loggano tutti gli eventi che avvengono nel backend) sono salvati in un volume Docker, siccome l'applicazione è ***dockerizzata***.

Infine, c'è da notare che sul *repository* di GitHub sono stati pubblicati anche i file *.env* che **NON** andrebbero mai pubblicati online. Tuttavia, per scopi didattici e per facilitare il lavoro di valutazione e prova al docente sono stati inclusi volontariamente. Una buona norma sarebbe quella di passare aggiungere un file *env.sample* e farlo modificare da qualsiasi utente utilizzi l'applicazione.

N. b: gli utenti creati manualmente nel database (employee) hanno una password di default **Jo5hu4!** Pertanto se si cerca di effettuare login con un employee già presente nel database mongo fornito, si dovrà utilizzare quella password.

Prerequisiti per l'avvio

I requisiti per poter utilizzare l'applicazione sono pochi ma vanno **assolutamente** soddisfatti per poter funzionare.

Requisiti software:

- Docker e Docker Compose
- Editor di testo o IDE (opzionale per modifiche)
- Git per clonare il repository remoto

Requisiti vari:

- Creare (se non esistente) un account su ***Resend*** e creare una propria API key
- Modificare il file *.env* con la propria API key di Resend

Installazione del software

Per installare il software sul proprio computer locale, occorre innanzitutto clonare il repository seguente sulla propria macchina:

Repository remoto (clone https): https://github.com/VeljDev/project2-spbd.git

Repository remoto (link al repo): https://github.com/VeljDev/project2-spbd

Comando per la clonazione:

```
git clone <repository>
```

Successivamente, aprire con un editor di testo il file (partendo dalla root del repository) il seguente file:
backend/.env

Successivamente, modificare la seguente riga con la propria API key di Resend:

```
RESEND_API_KEY=your_resend_api_key
```

Infine, partendo dalla root del repository, digitare il seguente comando da terminale per creare i container in Docker:

```
docker compose up
```

Una volta che docker avrà finito di inizializzare i container, l'applicazione potrà essere utilizzata.

Informazioni sulle porte e indirizzi

Una volta che i container Docker stanno funzionando, si può accedere all'applicazione con le seguenti porte e indirizzi:

- **Frontend:** localhost:5173 (su un qualsiasi browser, es.: ***Mozilla Firefox***)
- **Backend:** localhost:4004 (da un qualsiasi client in grado di effettuare richieste http, es.: ***Postman***)

Al database si può accedere utilizzando ad esempio ***MongoDBCompass*** connettendosi al proprio indirizzo IP della macchina sulla porta 27017 (siccome il DB è containerizzato). Es.: **192.168.1.10:27017**

Backend API endpoints

- **POST /auth/register:** permette di registrare un nuovo employee
- **POST /auth/login:** permette di effettuare il login con un employee
- **GET /auth/refresh:** permette di refreshare il token di accesso (se il token di refresh è valido)
- **GET /auth/logout:** permette di effettuare il logout di un employee
- **GET /email/verify/:code:** permette di verificare una email di un employee se il codice è valido
- **POST /password/forgot:** permette di inviare una email di reimpostazione della password per un employee
- **POST /password/reset:** permette di reimpostare la password di un employee
- **GET /user:** permette di ritornare le informazioni dell'employee loggato
- **GET /sessions:** permette di ottenere tutte le sessioni dell'employee loggato
- **DELETE /sessions/:id:** permette di eliminare la sessione specificata
- **GET /customers:** permette di ottenere tutti i clienti dell'employee loggato

Demo del software

Una demo del software viene fornita alla consegna in formato video, dove illustriamo come eseguire tutti i procedimenti spiegati in precedenza e mostriamo le varie funzionalità del prodotto. Naturalmente, tutto quanto mostrato si potrà riprodurre da sé.

Potenziali rischi

(Gravità ALTA) Applicazione (development) in HTTP e non HTTPS.

Anche se in sviluppo, l'applicazione non utilizza HTTPS per le comunicazioni, lasciando dati sensibili (come credenziali e token JWT) vulnerabili ad attacchi man in the middle, che potrebbero intercettare il traffico di rete.

Rimedio: configurare un certificato SSL/TLS per garantire comunicazioni cifrate.

(Gravità MEDIA) Bannare determinati Indirizzi IP

Anche se ci sono delle restrizioni per quanto riguarda le richieste http ogni 15 minuti, l'assenza di un sistema per bloccare (mettere in blacklist) degli indirizzi IP può lasciare vulnerabile l'applicazione a attacchi continuativi come un brute force.

Rimedio: implementare nel backend una lista di IP bannati (blacklist) direttamente in un middleware, oppure (meglio ma più complesso in quanto richiede un'infrastruttura dedicata) configurare un firewall esterno a livello di rete o server.

(Gravità ALTA) Mancanza di Backup o ripristino del Database

In un'applicazione critica con dipendenti e clienti è essenziale avere un sistema di backup/ripristino e un piano di disaster recovery. In assenza di questi sistemi, il database è vulnerabile a perdite dati che potrebbero essere causate da guasti hardware, attacchi o errori umani.

Rimedio: backup regolari e piani di ripristino utilizzando degli script che invochino comandi di mongodump e mongorestore. È importante inoltre considerare per applicazioni critiche di separare lo storage dal server dove risiede l'applicativo in un'altra posizione, ad esempio in una SAN. In questo modo i dati sono centralizzati e ridondanti, quindi in caso che il server vada down oppure si rompa un disco è sempre possibile effettuare un backup. Inoltre è importante specificare per applicazioni critiche un RPO e RTO per garantire la minor perdita di dati e il minor tempo per recuperarli.

Referenze

Il progetto è stato sviluppato su una base tutorial presente su Youtube:

https://www.youtube.com/watch?v=NR2MJk9C1Js&t=472s

L'obiettivo iniziale è stato quello di capire come funzionassero le tecnologie utilizzate, siccome nessuno dei membri coinvolti nel progetto ha mai lavorato con simili strumenti. Pertanto, abbiamo seguito il tutorial che spiega come funzionano tali tecnologie creando da zero un progetto ***MERN*** che consiste in una basilare applicazione con sistema di login e sessioni.

Nonostante il progetto sia stato adattato e ampliato, differendo quindi da quello proposto nel video, abbiamo ritenuto importante citare le fonti di spunto e quindi il creatore ***Nikita Dev*** su youtube.