RISK ASSESSMENT REPORT

Storia delle revisioni

Data	Ver.	Descrizione	Autore
28/01/25	1.0	Setup del template, ToC, glossario	Finocchi
29/01/25	1.1	Introduzione	Basili
31/01/25	1.2	Panoramica applicazione e prima funzionalità	Buniy, Finocchi
14/02/25	1.3	Studio minacce e loro soluzioni	Basili, Buniy, Finocchi
16/02/25	1.4	Studio Strategie di mitigazione del rischio	Cappellini
20/02/25	1.5	Studio risposta a recovery e incidenti	Basili

1 Introduzione

1.1 Scopo del report

Questo documento costituisce un **Security Assessment Report (SAR)** per il sistema **MS3**. L'obiettivo principale del report è fornire una valutazione approfondita dei rischi di sicurezza associati all'applicazione, identificando vulnerabilità potenziali, analizzando le minacce esistenti e proponendo adeguate misure di risposta.
Il report intende supportare il processo decisionale degli stakeholder, garantendo un livello adeguato di sicurezza per il sistema e proteggendo asset critici e dati sensibili.
Questo documento è destinato a:

- **Team di sicurezza IT**: per valutare e implementare le misure di protezione necessarie.
- **Sviluppatori e architetti di sistema**: per comprendere e integrare i requisiti di sicurezza nella progettazione del software.
- **Responsabili della conformità**: per verificare l'allineamento del sistema agli standard e alle normative di sicurezza.
- **Stakeholder aziendali**: per essere informati sui potenziali rischi e sulle strategie di mitigazione adottate.

1.2 Ambito della valutazione

L'analisi di sicurezza copre l'intero ciclo di vita del sistema **MS3**, dalla progettazione all'implementazione, includendo:

- **Architettura del sistema** e modello di sicurezza applicato
- **Flusso dei dati** e identificazione di potenziali punti di vulnerabilità
- **Minacce informatiche** che possono compromettere la sicurezza dell'applicazione
- **Vettori di attacco** e strategie di prevenzione
- **Conformità agli standard di sicurezza e regolamenti applicabili**

L'obiettivo è valutare il livello di rischio corrente e identificare azioni correttive che possano migliorare la postura di sicurezza complessiva del sistema.

1.3 Metodologia

L'analisi del rischio è stata condotta seguendo un processo strutturato e riproducibile, basato su best practice di cybersecurity. Il processo si articola nei seguenti passi:

1. **Raccolta delle informazioni**
    - Analisi della documentazione dell'architettura del sistema
    - Identificazione degli asset critici e dei dati sensibili
2. **Identificazione delle vulnerabilità**
    - **Scanning delle vulnerabilità** con strumenti automatizzati per identificare punti deboli sconosciuti
    - **Revisione del codice sorgente** per analizzare possibili vulnerabilità (es. static code analysis con Codacy o dynamic code analysis con OWASP Zap)
3. **Analisi e categorizzazione del rischio**
    - Assegnazione di un punteggio di rischio basato su criteri oggettivi come probabilità di sfruttamento e criticità degli asset coinvolti
    - Confronto con database di vulnerabilità pubblici (es. OWASP Top 10)
4. **Definizione delle misure di mitigazione**
    - Proposta di contromisure in linea con standard di sicurezza (ISO 27001, NIST, CIS Controls)
    - Pianificazione degli interventi e valutazione dei rischi residui

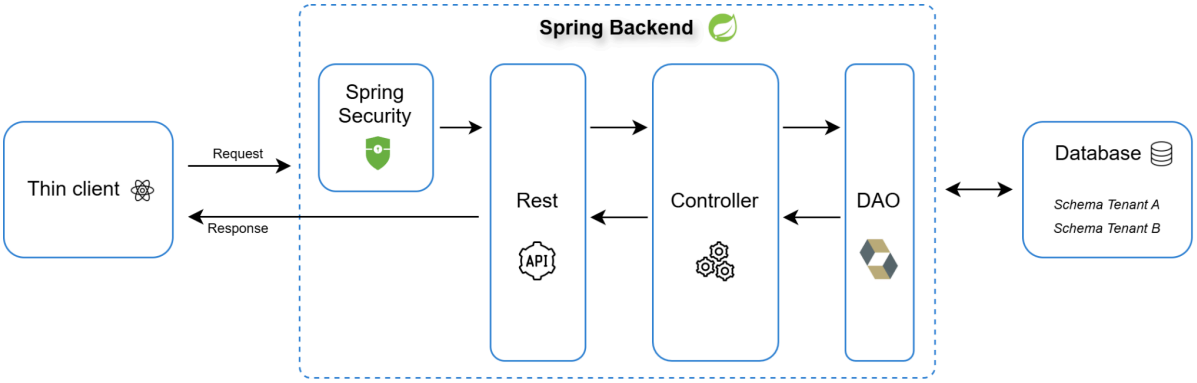
Il processo è stato eseguito garantendo **trasparenza, riproducibilità e tracciabilità**, documentando ogni fase dell'analisi e i risultati ottenuti. L'approccio adottato consente, quindi, un'analisi dettagliata e coerente dei potenziali rischi, facilitando la definizione di un piano d'azione efficace per il miglioramento continuo della sicurezza del sistema **MS3**.

2 Panoramica dell'applicazione

2.1 Descrizione dell'applicazione

L'applicazione vuole gestire i turni ospedalieri dei medici che ne fanno parte, cercando di trovare un'assegnazione che sia equa tra i dottori, ma che al tempo stesso rispetti determinati vincoli temporali e personali. Per fare ciò, diversi attori sono chiamati in causa: il configuratore, il pianificatore e il dottore.

2.2 Architettura del sistema



2.3 Funzionalità e caratteristiche chiave

Vediamo più nel dettaglio ciascun ruolo:

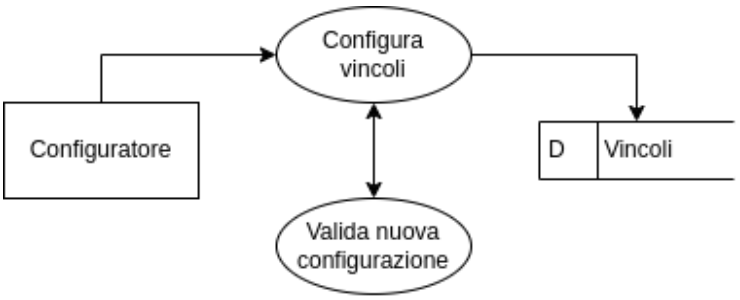
- Il **configuratore** decide i parametri della schedulazione, come il numero minimo di ore che si devono aspettare prima di assegnare un altro turno ad un dottore che ne ha appena svolto uno notturno,
- un **pianificatore** crea le assegnazioni effettive in un arco temporale,
- i **dottori** visualizzano il calendario dei turni, o richiedono degli scambi di turni con altri colleghi

Descriviamo ora le funzionalità a partire dalle quali saranno tratte le prossime considerazioni sul rischio che seguiranno

1. **Configurazione vincoli**: la pianificazione dei turni, oltre a rispettare dei vincoli personali come quelli specifici per le donne in cinta o per le persone con più di 62 anni, deve rispettare anche i vincoli temporali scelti da un configuratore.
2. **Creazione scheduling**: il pianificatore deciderà il periodo in cui i dottori dovranno essere assegnati ai rispettivi turni, in base alle loro specializzazione e in modo più fair possibile (es. facendo in modo che le festività non siano affidate sempre agli stessi dottori).

2.4 Diagramma di flusso dei dati

Configurazione vincoli



3 Valutazione del rischio

3.1 Identificazione degli asset e dei dati sensibili

Per la valutazione degli asset, si è scelto di raggrupparli in base alla struttura dell'applicazione, considerando le componenti principali del sistema: **Database**, **Backend**, **Frontend** e **Infrastruttura di rete**. Ogni componente contiene asset critici che, se compromessi, possono influire sulla sicurezza, integrità e disponibilità dell'applicazione.

Database

Il database rappresenta il cuore dell'applicazione, in quanto custodisce dati sensibili e critici per il funzionamento del sistema. In particolare come assets si sono identificati:

- *Database fisico*
  L'intero database è un asset fondamentale, poiché memorizza le informazioni principali dell'applicazione. La sua integrità e disponibilità sono cruciali.
- *Dati degli utenti*
  Si intendono tutti quei dati che comprendono informazioni personali (PII - Personally Identifiable Information) come nomi, indirizzi email, password (opportunamente criptate), turni di lavoro e altri dati sensibili. La protezione di questi è essenziale per garantire la conformità alle normative sulla privacy (es. GDPR).
- *Log delle transazioni*
  Sono i file che registrano le operazioni effettuate dagli utenti e dal sistema. Questi log possono contenere informazioni sensibili e devono essere protetti per evitare manipolazioni o accessi non autorizzati. Inoltre risultano essere di fondamentale importanza in caso di necessità di ripristino o analisi degli eventi avvenuti sul DB.

Backend (Spring + Hibernate)

Il backend gestisce la logica di business, l'accesso ai dati e l'esposizione delle API. Gli assets qui individuati sono:

- *Codice sorgente Java*
  Contiene la logica di business e le regole di gestione dei dati. La compromissione del codice può portare a vulnerabilità sfruttabili (es. injection, o escalation di privilegi).
- *Configurazioni di Hibernate e delle connessioni al database*
  In particolare i file di configurazione che includono credenziali di accesso al database e stringhe di connessione. Se non adeguatamente protetti, possono esporre il sistema a rischi di accesso non autorizzato.
- *File di configurazione (application.properties o application.yml)*
  Contengono variabili d'ambiente, chiavi API, configurazioni di sicurezza e impostazioni dell'applicazione. È fondamentale proteggerli da accessi non autorizzati e versionarli correttamente.
- *API REST esposte e loro endpoint*
  Gli endpoint REST sono la porta d'ingresso per le comunicazioni tra il frontend e il backend. Devono essere protetti contro attacchi comuni (es. SQL injection, XSS, CSRF). È importante garantire l'autenticazione e l'autorizzazione corrette per ogni chiamata.Sempre per quel che riguarda le API REST, fa parte degli asset dell'applicazione anche il documento di *"Specifica delle Autorizzazioni API in Ambiente Multi-Tenant per MS3*", nel quale si vanno a definire nello specifico le API del sistema ed i ruoli che vi hanno accesso.
- *Librerie utilizzate*
  Le librerie rappresentano un punto critico per la sicurezza e la stabilità del sistema. L'uso di dipendenze obsolete o vulnerabili può esporre l'applicazione a rischi di sicurezza noti, come attacchi di deserializzazione, vulnerabilità zero-day o incompatibilità tra versioni. È essenziale mantenere aggiornate le librerie e monitorare eventuali vulnerabilità.

Frontend (React + JavaScript)

Il frontend è l'interfaccia visibile agli utenti finali e gestisce la comunicazione con il backend. Gli assets in questa parte dell'applicazione sono:

- *Codice sorgente React*
  Il codice frontend è generalmente pubblico, potrebbe contenere informazioni sensibili come chiavi API o configurazioni non sicure. È importante gestire correttamente la minificazione e l'offuscamento del codice per ridurre i rischi.
- *Configurazioni delle librerie e dei moduli JavaScript*
  L'utilizzo di librerie di terze parti può introdurre vulnerabilità se non correttamente aggiornate. La gestione delle dipendenze (es. tramite package.json) è un aspetto critico per evitare l'introduzione di pacchetti compromessi
- *Token di autenticazione (es. JWT)*
  I token JWT utilizzati per l'autenticazione e l'autorizzazione sono asset sensibili. Se memorizzati nel localStorage o sessionStorage, possono essere vulnerabili ad attacchi XSS. È importante adottare strategie sicure per la gestione di questi token.

Comunicazioni e Infrastrutture di Rete

Anche le modalità di comunicazione e le infrastrutture di rete costituiscono asset critici da considerare. Essendo l'applicazione ancora in una fase di prototipizzazione, i seguenti asset non sono ancora presenti, per cui rientrano tra gli asset futuri da aggiungere. In particolare sono stati identificati:

- *Comunicazione sicura (HTTPS/TLS)*
  Tutte le comunicazioni tra frontend e backend devono essere cifrate tramite HTTPS per proteggere i dati in transito da attacchi di tipo *man-in-the-middle* (MITM). Il certificato SSL/TLS utilizzato rappresenta un asset da proteggere e monitorare.
- *Server e infrastruttura (es. VPS)*
  L'infrastruttura fisica o virtuale su cui è ospitata l'applicazione rappresenta un asset critico. L'accesso al server deve essere strettamente controllato tramite firewall, VPN e autenticazione a più fattori.
- *Backup dei dati*
  I backup regolari del database e del codice sorgente sono asset importanti per garantire la disponibilità e il ripristino in caso di incidente. È fondamentale proteggerli da accessi non autorizzati e per effettuare eventuali piani di ripristino.

3.2 Identificazione delle minacce

In questa sezione si andranno a trattare le principali minacce che potrebbero andare a compromettere la sicurezza e l'integrità dell'applicazione. L'obiettivo è quindi quello di fornire una panoramica completa dei rischi potenziali che potrebbero avere un impatto negativo sull'organizzazione.
La metodologia utilizzata per l'identificazione delle minacce è STRIDE, che permette di classificare le minacce in sei categorie: *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service* ed *Elevation of Privilege*.

Spoofing

Lo spoofing è una particolare tipologia di attacco che consente ad un attore malevolo di nascondere la propria identità per risultare "affidabile" alla vittima designata e ottenere accesso a informazioni riservate e dati sensibili. Alcuni dei rischi a cui si va incontro sono:

- **Violazione della riservatezza:** Accessi non autorizzati possono esporre informazioni personali e mediche.
- **Furto di identità:** Informazioni identificative possono essere utilizzate per impersonare individui.
- **Non conformità al GDPR:** La mancata adozione di misure di sicurezza adeguate può comportare sanzioni legali.

Nel sistema MS3 riscontriamo minacce di spoofing ad esempio nel transito del token JWT. Questo viaggia in chiaro nell'header dei pacchetti HTTP nel campo Authentication, rendendo semplice per attacchi di tipo MITM la trafugazione del dato.
Un altro esempio che riguarda comunque il token JWT interessa il dove questo venga conservato lato client. Al momento della scrittura del documento esso viene mantenuto all'interno delle localSession del browser. Questa pratica risulta essere scorretta poiché soggetta ad attacchi di tipo XSS.
Sempre legato alla comunicazione dal browser dell'utente al server, se non vi fosse un'adeguata protezione, l'utente potrebbe venire tratto in inganno e involontariamente fornire le proprie credenziali ad un sito falso che potrebbe utilizzarle per accedere alla piattaforma.

Tampering

Il tampering è un tipo di attacco attivo in cui un attore malevolo modifica in modo non autorizzato i dati o il codice dell'applicazione, compromettendone l'integrità e il corretto funzionamento.
Alcuni dei rischi a cui si va incontro sono:

- **Manomissione dei dati:** Alterazioni non autorizzate portano a informazioni errate.
- **Interruzione del servizio:** Modifiche ai dati critici possono causare disservizi.
- **Non conformità al GDPR:** L'integrità dei dati è essenziale per garantire la protezione delle informazioni personali e per rispettare i requisiti normativi.

Nel sistema MS3, una minaccia di questo tipo può derivare da una mancanza di protezioni sui database o dalla vulnerabilità delle API che permettono la manipolazione dei dati senza adeguati controlli di autorizzazione. Ad esempio, un attaccante potrebbe alterare i turni di lavoro direttamente nel database sfruttando una SQL Injection o modificare le richieste API tramite un attacco di intercettazione. Inoltre, se il codice dell'applicazione non è protetto adeguatamente, un utente malintenzionato potrebbe modificarlo per aggirare i controlli di sicurezza ed eseguire azioni privilegiate senza autorizzazione.
Un caso specifico possibile di SQL Injection è stato riscontrato nella classe *SchemaSwitchingConnectionProviderPostgreSQL*, che gestisce la configurazione del multi-tenancy attraverso la modifica del *search_path* in PostgreSQL. Attualmente, il valore del **tenantIdentifier** viene concatenato direttamente nella query SQL:

```
connection.createStatement().execute( sql: "SET search_path TO " + tenantIdentifier);
```

Se un attaccante riuscisse a manipolare il valore del tenant, potrebbe eseguire comandi SQL arbitrari, compromettendo l'integrità del database e accedendo a dati di altri tenant.

Repudiation

La repudiation (ripudio) è un attacco in cui un attore malevolo nega di aver effettuato un'azione, come ad esempio l'invio di una richiesta o una modifica ai dati, rendendo difficile o impossibile attribuire la responsabilità di tale azione. Questo tipo di attacco può compromettere la tracciabilità delle attività e indebolire la fiducia nelle operazioni svolte all'interno di un sistema.
I rischi associati alla repudiation includono:

- **Mancanza di tracciabilità:** Senza un adeguato sistema di logging e autenticazione, risulta difficile verificare chi ha compiuto determinate azioni, aumentando il rischio di frodi o azioni dannose non rintracciabili.
- **Compromissione dell'affidabilità:** La capacità di negare un'azione compromette la fiducia nel sistema, soprattutto in contesti dove è essenziale poter verificare la responsabilità, come nella configurazione di parametri o gestione di dati sensibili.
- **Non conformità al GDPR:** La gestione dei log e delle informazioni relative alle operazioni è cruciale per rispettare le normative sulla protezione dei dati personali.

La mancanza di prove adeguate potrebbe violare i requisiti di tracciabilità e responsabilità previsti dal GDPR.

Nel sistema MS3, un rischio di repudiation potrebbe manifestarsi qualora non fosse implementato un sistema robusto di logging e audit. Se un attaccante o un utente malintenzionato riuscisse a manipolare i registri delle attività o a disattivare il sistema di log, sarebbe più difficile risalire a chi ha compiuto determinati atti dannosi, come l'accesso non autorizzato a dati sensibili o la modifica di informazioni critiche.

Information disclosure

Il problema di information disclosure si verifica quando vengono esposte informazioni sensibili ad utenti che non hanno l'autorizzazione ad accedervi. Le informazioni in questione spaziano dai dettagli sul sistema alle informazioni personali degli utenti, o ancora configurazioni utilizzate alle credenziali di accesso.

Tra i rischi legati a tale minaccia troviamo:

- **Aumento della superficie d'attacco**: Dettagli tecnici aiutano un potenziale attaccante ad identificare e sfruttare vulnerabilità note tramite attacchi mirati
- **Violazione della privacy**: La divulgazione dei dati sensibili è un problema serio e causa gravi danni all'azienda
- **Non conformità alle normative**: Tali eventualità possono violare gli articoli 32 e 33 del GDPR o le stringenti direttive del NIS2, portando a sanzioni pecuniarie pesanti.

Importanza deve essere dunque data a tutti i punti di accesso alle informazioni sensibili, come i messaggi di errore dettagliati o i dati in transito e a riposo.

Un aspetto di rilievo riguardante i dati sensibili è inoltre il disco fisso. Difatti, in caso di furto dello stesso, un attaccante sarebbe in grado di accedere ai dati sensibili degli utenti, e ciò porterebbe al mancato rispetto della normativa sulla protezione dei dati del GDPR.

Per quel che riguarda i dati sensibili memorizzati nel database vi sono:

- Nello schema ms3_public l'entità ms3_system_user contiene tutti i dati degli utenti. Per un migliore isolamento dei dati, i dati raggiungibili pubblicamente dovrebbero essere cifrati e isolati agli schemi dei tenant, limitandosi ai dati necessari (email e password, e non anche tutti gli altri)
- Negli schemi di ciascun tenant i dati dei dottori (tabella "doctor") sono in chiaro sul database. Come indicato dal GDPR, non potendo garantire sicurezza di anonimizzazione in caso di furto, questi dati dovrebbero essere crittografati (birthday, email, lastname, name, taxcode)

Denial of Service

Il **Denial of Service (DoS)** è un tipo di attacco in cui un attore malevolo mira a rendere un sistema o una rete inutilizzabile, sovraccaricando le risorse disponibili o bloccando l'accesso legittimo agli utenti. Questo tipo di attacco può ottenere impatti devastanti su un sistema, interrompendo operazioni cruciali e compromettendo l'affidabilità del servizio.

I rischi associati al DoS includono:

- **Interruzione dei servizi:** Gli attacchi DoS possono causare il malfunzionamento o l'arresto completo di un'applicazione, compromettendo l'accesso ai servizi da parte degli utenti legittimi.
- **Perdita di produttività:** Un attacco DoS può causare il blocco temporaneo o permanente di un sistema, interrompendo le attività quotidiane e impedendo agli utenti di completare operazioni essenziali. Questo porta a una riduzione dell'efficienza e rallenta i processi aziendali, con conseguenti danni economici e perdita di tempo prezioso.
- **Impatto sulla sicurezza:** Se il sistema è inaccessibile a causa di un attacco DoS, potrebbe esserci un ritardo nell'applicazione delle misure di sicurezza necessarie, lasciando spazio ad altre minacce più gravi.

Questo tipo di attacco potrebbe essere realizzato in diversi modi:

- **Flooding del server** con un elevato numero di richieste, esaurendo le risorse disponibili.
- **Exploiting di vulnerabilità specifiche**, come bug nell'applicazione o nel database, per causare un crash del sistema.
- **Attacchi distribuiti (DDoS)**, in cui più dispositivi compromessi vengono utilizzati per generare traffico artificiale e sovraccaricare il servizio.

In MS3, un attacco DoS potrebbe verificarsi se il sistema non fosse protetto adeguatamente contro il traffico anomalo o gli attacchi distribuiti (DDoS). La mancanza di misure di protezione come i firewall o i sistemi di monitoraggio del traffico potrebbe esporre MS3 a questo tipo di minaccia. Ad esempio, un attacco che mira a congestionare la rete potrebbe impedire al personale ospedaliero di accedere al sistema per consultare o aggiornare i turni di lavoro, causando ritardi e disorganizzazione.

Elevation of Privilege

L'Elevation of Privilege è una vulnerabilità di sicurezza in cui un utente o un processo ottiene un livello di accesso più elevato rispetto a quello permesso. Questo generalmente può avvenire sfruttando bug nel software, configurazioni errate o falle nei meccanismi di autenticazione e autorizzazione.
I rischi all'interno dell'applicazione di MS3 in caso di problemi di Elevation of Privilege sono molteplici:

- **Alterazione dei turni:** nel caso in cui ad esempio un medico riuscisse ad alterare i propri privilegi, potrebbe ad esempio alterare la disposizione dei turni in suo favore.
- **Modifica incontrollata delle configurazioni dell'applicazione:** se un utente riuscisse ad elevare il suo ruolo per arrivare ad essere un configuratore potrebbe essere in grado di alterare le impostazioni dei turni, le regole di assegnazione generali, le ore di lavoro previste per un turno. Questo potrebbe portare ad esempio alla violazione dei contratti firmati dai medici sul numero di ore di lavoro previste.
- **Modifica dei turni di altri dottori:** un altro potenziale rischio, nel caso di elevazione dei privilegi da un dottore all'altro, è quello di inviare richieste di cambio turno senza l'effettiva autorizzazione del richiedente.

Per queste ragioni la gestione corretta dei ruoli nell'applicazione MS3 è un punto critico. Non essendo stata l'applicazione in origine orientata né ad aspetti di sicurezza, né tanto meno al rispetto delle normative vigenti (quali il GDPR), si è ritenuto di fondamentale importanza l'introduzione di Spring Security come framework di sicurezza.

Analisi delle dipendenze

Come osservato nell'analisi degli asset dell'applicazione, le dipendenze rappresentano un punto critico per la sicurezza. Infatti, potrebbero contenere del codice che presenta delle vulnerabilità non note a tempo d'inserimento della dipendenza, e che vengono scoperte solamente successivamente. Per questa ragione sarebbe buona strategia quella di controllarne periodicamente lo stato di salute e sempre procedere al loro aggiornamento dove possibile.
Per l'analisi delle dipendenze lato backend si è sfruttato il tool di analisi messo a disposizione da Owasp chiamato Dependency-Check. Questo tool fornisce un resoconto in un file HTML di tutte quante le vulnerabilità riscontrate. Per l'esecuzione di questa analisi si è utilizzato il seguente comando maven:

```
mvn org.owasp:dependency-check-maven:check
```

Come ci si aspettava, il report fornito dal tool ha evidenziato un grande problema di vulnerabilità a livello di dipendenze. In particolare si evidenziano nel report gravi vulnerabilità a livello di:

- spring-boot-starter-web-2.1.3.RELEASE.jar

    - cpe:2.3:a:vmware:spring_boot:2.1.3:release:*:*:*:*:*:*
- jackson-databind-2.9.8.jar
    - cpe:2.3:a:fasterxml:jackson-databind:2.9.8:*:*:*:*:*:*:*
- spring-core-5.1.5.RELEASE.jar
    - cpe:2.3:a:vmware:spring_framework:5.1.5:release:*:*:*:*:*:*
- postgresql-42.2.5.jar

Per quel che riguarda l'analisi delle dipendenze lato frontend ci si è avvalsi del tool di analisi Snyk.
Snyk è una piattaforma di sicurezza specializzata nell'identificazione e gestione delle vulnerabilità nelle dipendenze open-source, nel codice, nei container e nelle infrastrutture cloud. Integrandosi facilmente con CI/CD, repository di codice e ambienti cloud, Snyk permette agli sviluppatori di rilevare e correggere le falle di sicurezza in modo proattivo.
Per l'utilizzo da riga di comando sono state eseguite le seguenti operazioni nella cartella /frontend:

```
npm install -g snyk
snyk auth
snyk test --json > snyk-report.json
```

Per una lettura facilitata si è utilizzato il comando:
[Powershell]

```
Get-Content snyk-report.json | ConvertFrom-Json | `
      Select-Object -ExpandProperty vulnerabilities | `
      Select-Object packageName, severity, version, title | `
      ConvertTo-Json | Out-File vulnerabilita-pulite.json
```

[bash]

```
jq '[.vulnerabilities[] | {packageName, severity, version, title}]'
snyk-report.json > snyk-report-easy-read.json
```

TODO: allegare il file che verrà messo su MS3-docs dei report di vulnerabilità del tool Snyk e del tool Dependency Check di OWASP

3.4 Categorizzazione del rischio e prioritizzazione

Priorità	Rischio	Categoria	Motivazione
1	SQL Injection	CRITICAL	Essendo riportato nel documento un esempio di codice soggetto a SQL injection in una sezione critica come la query per reperire il tenant, si è scelto di dare massima priorità a questo rischio.
2	Elevation of Privileges	CRITICAL	Per mancanza di competenze tecniche, non si sa valutare il rischio rispetto a SQL injection nel contesto di MS3, perciò basandosi su OWASP si decide comunque di dargli una priorità alta, ma inferiore ad un attacco di SQL injection, il quale è molto più probabile.
3	Comunicazione non sicura	CRITICAL	La mancanza di uno strato di comunicazione sicuro è un problema grave a cui bisogna porre rimedio, altrimenti si diventa deboli a molti attacchi anche semplici, ma con effetti devastanti (es. MITM).

Priorità	Rischio	Categoria	Motivazione
4	Anonimizzazione dati	HIGH	Per conformità ai regolamenti, questo aspetto è essenziale per il deploy dell'applicazione, e se non è rispettato, si incappa in sanzioni pecuniarie con conseguenti danni alla reputazione e possibilità di sospensione delle attività.
5	Mancanza di logging e audit	MEDIUM	I logger e gli audit sono uno degli strumenti principali sia per capire le operazioni che il sistema sta svolgendo, sia per capire lo stato del sistema ma anche per riuscire ad identificare eventuali attacchi alla radice, o, in caso di disastro avvenuti, risalire alle operazioni perpetrate dall'attaccante.
6	Conservazione del JWT lato client	LOW	La conservazione in chiaro del token JWT espone il sistema ad attacchi di tipo XSS. L'utente potrebbe essere facilmente ingannato tramite email mirate che riportano a siti malevoli i quali si occuperebbero di reperire il token ed inviarlo al server attaccante.
7	Attacchi DoS	LOW	La probabilità di subire attacchi DoS viene reputata inferiore rispetto alle altre possibilità di attacchi, per cui gli viene assegnata una priorità bassa.

4 Controlli di sicurezza

4.1 Misure di sicurezza attuali

Aspetti di sicurezza dell'applicazione sono già stati implementati tramite meccanismi di autorizzazione basati su **Spring Security**. Questo framework fornisce una robusta gestione degli accessi, consentendo di proteggere le risorse sensibili e limitare l'esposizione ai rischi di accesso non autorizzato.
L'autenticazione degli utenti è gestita mediante l'uso di **JWT (JSON Web Token)** per garantire sessioni sicure e ridurre il rischio di attacchi. Inoltre, sono stati implementati controlli granulari sui permessi degli utenti, differenziando i livelli di accesso in base ai ruoli definiti: **dottori**, **pianificatori** e **configuratori**.

Spring security

Per contrastare le vulnerabilità riconducibili all' **Elevation of Privilege**, nell'applicazione è stato introdotto **Spring Security**, il framework di riferimento per la sicurezza nelle applicazioni basate su **Spring**. Questo framework offre strumenti avanzati per il controllo dell'**autenticazione** e dell'**autorizzazione**, garantendo una gestione robusta degli accessi alle risorse.
All'interno dell'applicazione è stato implementato un sistema di autorizzazione basato su **JWT (JSON Web Token)**. I token vengono generati al momento dell'autenticazione e utilizzati in ogni richiesta per verificare l'identità dell'utente. Ogni JWT include informazioni fondamentali, tra cui:

- **Ruolo** dell'utente
- **Permessi** associati al ruolo
- **Tenant** di appartenenza

L'uso combinato di Spring Security e JWT permette di definire policy di accesso granulari, basate sui ruoli degli utenti. L'applicazione prevede tre ruoli principali: **DOCTOR**, **CONFIGURATOR** e **PLANNER.**

Ognuno di questi ruoli ha dei permessi specifici su determinate risorse, garantendo che ogni utente possa accedere solo alle funzionalità per cui è autorizzato.
Per quanto riguarda l'integrazione di Spring Security, questa è stata realizzata attraverso una configurazione personalizzata, definendo filtri di sicurezza ed una strategia di gestione dell'autenticazione.
Di seguito è stato approfondito il ruolo delle classi legate al corretto funzionamento di Spring Security.

Il `security/SecurityConfigurer` è colui che ha il compito di configurare e gestire le politiche di sicurezza dell'applicazione, definendo le regole per l'autenticazione, l'autorizzazione e la gestione delle sessioni. Nel dettaglio si occupa di:

- Disabilitare **CSRF** (`Cross-Site Request Forgery`), in quanto l'applicazione utilizza un sistema di autenticazione basato su **token JWT**, rendendo la protezione CSRF superflua. CSRF è una misura di sicurezza principalmente necessaria per applicazioni che gestiscono sessioni basate su cookie.
- Definire le regole di autorizzazione, difatti sono stati configurati endpoint pubblici, come `/login/`, per permettere agli utenti di autenticarsi e ottenere un token JWT. Tutti gli altri endpoint richiedono un'autenticazione valida per essere accessibili.
- Impostare la gestione della sessione nella seguente riga del codice:
  `.sessionCreationPolicy(SessionCreationPolicy.STATELESS)`
  Ciò garantisce che l'applicazione non crei né conservi sessioni utente lato server, rendendo le API completamente **stateless**.
- Aggiungere un filtro personalizzato JwtRequestFilter per l'intercettazione delle richieste in ingresso, l'estrazione del JWT dall'Authorization Header e la sua validazione, arrivando quindi ad autorizzare l'accesso all'applicazione da parte dell'utente.

Il `filters/JwtRequestFilters` è un filtro personalizzato che si occupa di intercettare tutte le richieste in ingresso per verificare la presenza di un token JWT nell'header `Authorization`. Il flusso di lavoro del filtro è il seguente:

- Estrazione del Token JWT dalla richiesta
- Verifica dell'autenticazione tramite `SecurityContextHolder`
- Validazione del Token tramite la classe di utility `JwtUtils`

Questo filtro garantisce quindi che tutte quante le richieste protette siano autenticate tramite un token JWT valido.

4.2 Miglioramenti di sicurezza proposti

Al fine di rafforzare la sicurezza del sistema MS3, si propongono una serie di **miglioramenti** volti a mitigare le vulnerabilità individuate e ad allineare le misure di protezione agli standard di sicurezza. L'obiettivo principale di queste azioni è garantire la riservatezza, l'integrità e la disponibilità dei dati, riducendo il rischio di accessi non autorizzati e di perdita di informazioni sensibili.

Anonimizzazione dei dati

I dati per le tabelle sensibili dovrebbero essere cifrati o hashati, isolati solo nello schema del tenant, e a seconda dei bisogni, si può decidere se optare per approcci di K-Anonymity, L-Diversity e T-Closeness.

Cifratura del Sistema Operativo

Uno degli aspetti critici individuati nel punto 4.2 riguarda il rischio legato al furto del disco rigido contenente dati sensibili. Per affrontare questa problematica si potrebbe pensare a

soluzioni di cifratura del sistema operativo, dove esistono già soluzioni consolidate[1], ma siccome il deploy avviene tramite un'infrastruttura basata su container docker che usano immagini Linux, ed in particolare con un volume contenente il database, la soluzione che si può adottare in questo caso è la **cifratura dei volumi docker**.
Per cifrare volumi docker, una delle soluzioni più sicure e flessibili è sempre l'utilizzo di LUKS. Un aspetto cruciale in questa fase è la sicurezza della chiave di cifratura, aspetto per il quale si può utilizzare un TPM.
La soluzione studiata si può vedere parzialmente nelle modifiche implementate nel branch di MS3 *issue618,* ma non è stata portata a termine*:*

1. Inizialmente si era pensato di rendere automatizzata la cifratura dei soli volumi tramite degli script che andavano eseguiti precedentemente alla composizione.
2. Per difficoltà implementative questa soluzione è stata scartata, si è invece cercato di cifrare i volumi automaticamente tramite il plugin di docker rexray con LUKS per effettuare la cifratura sul disco locale

La soluzione finale probabilmente sarà però cifrare il sistema operativo, non il volume: il motivo è che questa soluzione è economica, standardizzata e semplice da mettere in campo, quindi se la cifratura comprende parti del disco che non sono necessarie, perché i dati di cui garantire la segretezza sono solo quelli nei volumi, e si ha un impatto negativo sulle performance, c'è una importante motivazione che può spingere a voler adottare quest'altra soluzione, ovvero la semplicità, oltre a garantire una sicurezza extra anche nella segretezza delle configurazioni, variabili d'ambiente, o logging persistenti.

Implementazione connessione sicura

Il token deve essere firmato per autenticare il possessore, e la connessione deve essere cifrata per evitare attacchi MITM.
La firma del token è già effettuata di default.
Per quanto riguarda l'implementazione della connessione sicura, essa va cambiata da http ad https, quindi sarà necessario creare un self signed certificate o comunque ottenerlo da una certification authority, dopo di chè si dovrà abilitare https con spring boot, possibile usando il file application.properties e il `SecurityConfigurer` all'interno del quale dobbiamo aggiungere la configurazione per accettare solo richieste HTTPS.

Sicurezza segreti

La soluzione per la connessione sicura può funzionare in fase di sviluppo, ma quando si dovrà fare il deploy, se **i segreti** come la password del keystore e i certificati SSL vengono committati su GitHub, il progetto diventa vulnerabile. Per **proteggere le credenziali** e altre informazioni sensibili, spring boot offre la possibilità di leggere le variabili d'ambiente nei .properties anzichè hardcordarle negli stessi, per cui si possono impostare quelle variabili nel sistema operativo o nella sezione environment del container, se il deploy avviene con essi, appunto come variabili d'ambiente e riferirle poi nell'application properties. Esistono altre soluzioni, come i file .env, ma in ogni caso bisogna fare attenzione a non salvare alcuna chiave sulla repository online, altrimenti la sicurezza della stessa è compromessa.

Autenticazione a due fattori

Il meccanismo della 2 Factor Authentication (2FA) permette di rimanere al sicuro anche dopo phishing, attacchi brute force o qualsiasi altro vettore di attacco che porta al cracking della password da parte di un hacker, il quale comunque non è in grado di completare l'impersonificazione per via della mancanza del secondo token, sicuro grazie alla sua proprietà time-based.

---

[1] Ad esempio LUKS per ambienti Linux, o ancora BitLocker per sistemi Windows.

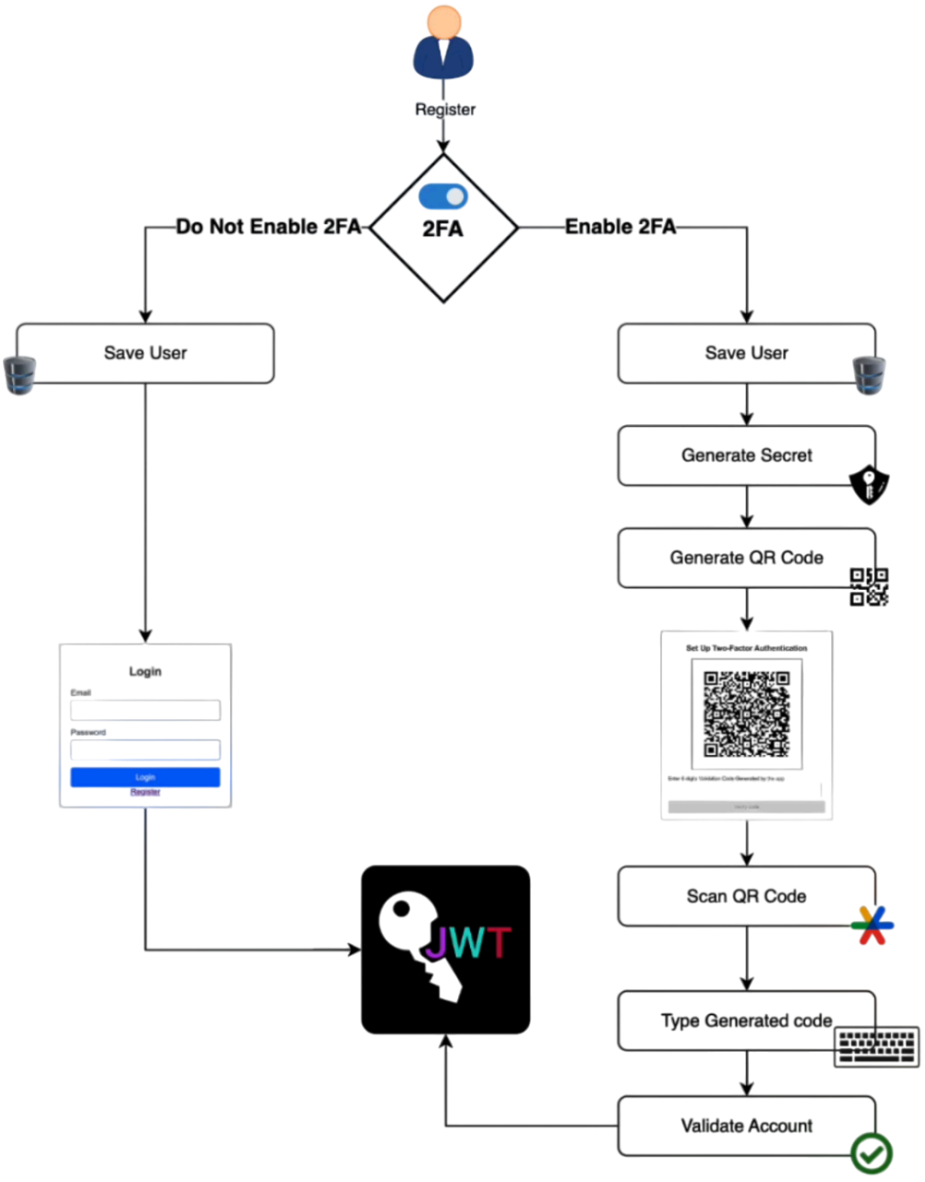
In questo modo saranno necessari 2 diversi tipi di fattori di autenticazione per verificare l'identità di un utente prima di concedergli l'accesso al suo account. Per questi fattori si può usare:

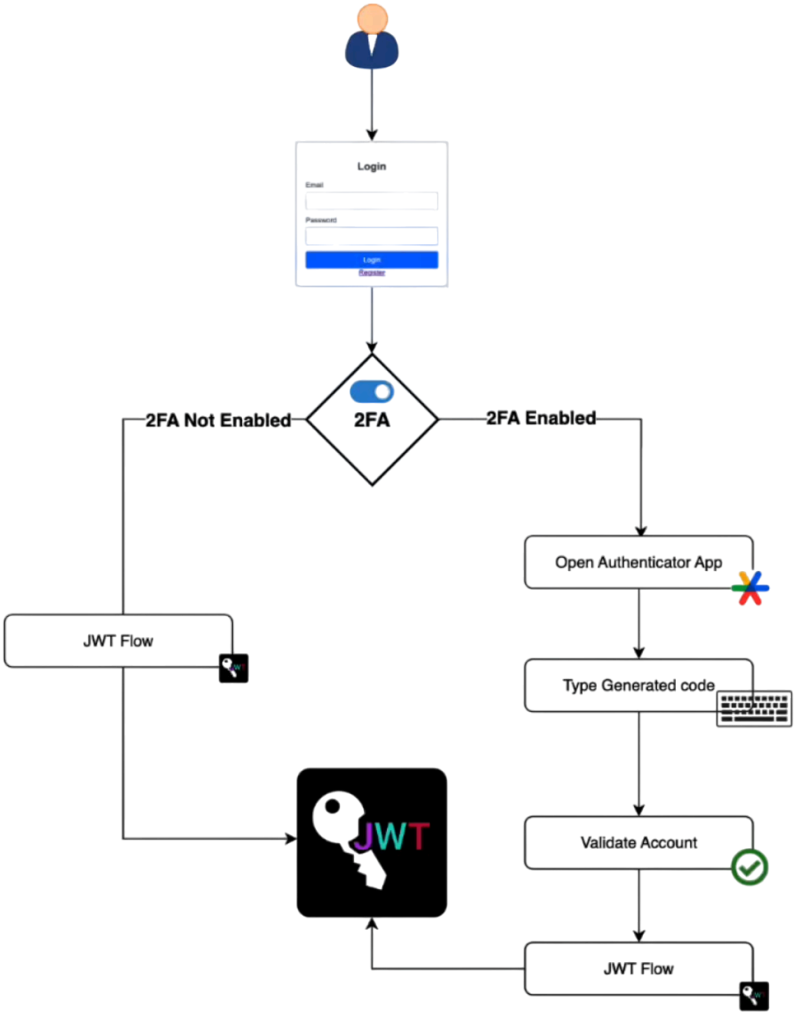
- Qualcosa che si conosce, come una password o un pin
- Qualcosa che si possiede, come uno smartphone o una smart card
- Qualcosa che ci rappresenta, ovvero informazioni biometriche uniche

Per entrare nell'applicazione si devono fornire quindi una password e un token di una delle due categorie rimanenti.

Per quanto riguarda la **registrazione**, nel caso in cui non sia attiva la 2FA, si salva nel database l'utente che potrà entrare nel sistema grazie ad un semplice login, ma se è attiva, dopo aver salvato l'utente si dovrà generare un segreto OTP specifico per quell'utente usato nella creazione di un QR code, che l'utente dovrà scannerizzare con un Authenticator, il quale gli fornirà un codice da inserire infine nell'applicazione per validare l'account.

Per il **login** invece, come prima senza il 2FA si seguirà il flusso normale tipico, mentre con 2FA servirà aprire l'Authenticator e inserire il codice che l'applicazione fornirà, l'Authenticator penserà poi ad inviare la richiesta al backend, il quale valuterà se il codice è valido o meno, generando il JWT e ritornando al flusso di JWT.





Per generare il segreto e il QR Code lo spettro delle librerie utilizzabili è ampio. Vedere ad esempio java-totp. Per quanto riguarda l'authenticator, da precedenti discussioni con il cliente si è deciso già di usare un **Google Authenticator**.

Validazione dell'input

Durante l'analisi della sicurezza del sistema MS3, è stata individuata una vulnerabilità di SQL Injection nella gestione del multi-tenancy (vedi 3.2, Tampering). Sebbene l'applicazione nel suo complesso sia protetta da SQL Injection grazie all'uso di **JPA Repository** e **Hibernate**, che gestiscono le interazioni con il database attraverso **query parametrizzate**, questa vulnerabilità si verifica in una specifica classe dove viene utilizzata una query SQL manuale senza alcuna validazione dell'input.

Per mitigare questo rischio, si propone di introdurre una rigorosa **validazione dell'input**, assicurandosi che il valore del parametro *tenantIdentifier* contenga solo caratteri alfanumerici e underscore (_). Questo controllo può essere implementato attraverso un'espressione regolare:

```
if (!tenantIdentifier.matches("^[a-zA-Z0-9_]+$")) {
    throw new SQLException("Invalid tenant identifier: " + tenantIdentifier);
}
connection.createStatement().execute("SET search_path TO " + tenantIdentifier);
```

Questa soluzione è semplice da implementare, ha un impatto minimo sulle prestazioni e impedisce l'inserimento di caratteri malevoli che potrebbero essere sfruttati per attacchi SQL Injection.

Protezione da attacchi DoS

Per mitigare il rischio di attacchi **Denial of Service (DoS)** e **Distributed Denial of Service (DDoS)** nel sistema MS3, si propone un approccio multi-livello che combina l'uso di un **Web Application Firewall (WAF)**, tecniche di **rate limiting**, e strumenti di **monitoraggio del traffico**.

1. Web Application Firewall (WAF)

L'implementazione di un **WAF**, come AWS WAF, Cloudflare WAF o mod_security per Nginx/Apache, può fornire una protezione avanzata contro attacchi DoS/DDoS basati su richieste HTTP. Il WAF opera filtrando e bloccando traffico sospetto prima che raggiunga l'applicazione, proteggendo il sistema da:

- **Flooding HTTP**: Un numero elevato di richieste provenienti da bot o script automatizzati può essere bloccato analizzando pattern sospetti e utilizzando challenge CAPTCHA.
- **Attacchi basati su exploit**: Tentativi di sfruttare vulnerabilità dell'applicazione vengono identificati e bloccati automaticamente.
- **Blacklist e protezione da IP malevoli**: Il WAF utilizza database di IP noti per attività malevole per bloccare automaticamente traffico dannoso.

2. Rate Limiting e Throttling

L'adozione di tecniche di rate limiting impedisce che un singolo indirizzo IP o utente possa inviare troppe richieste in un breve intervallo di tempo, sovraccaricando il sistema. Questo può essere implementato a livello di API Gateway o direttamente nell'applicazione tramite strumenti come:

- Spring Boot Rate Limiting con Bucket4j o Resilience4j
- Configurazione di throttling su API Gateway (es. AWS API Gateway, Kong, Nginx Rate Limit)

Un esempio di implementazione in Spring Boot utilizzando Bucket4j per limitare a 100 richieste al minuto per IP:

```
@Bean
public FilterRegistrationBean<Filter> rateLimitingFilter() {
    return new FilterRegistrationBean<>(new RateLimitingFilter(100, Duration.ofMinutes(1)));
}
```

3. Monitoraggio del traffico e Logging

Un sistema efficace di difesa dagli attacchi DoS/DDoS richiede il monitoraggio del traffico per individuare comportamenti anomali in tempo reale. Strumenti come **Prometheus + Grafana**, **ELK Stack (Elasticsearch, Logstash, Kibana)** o **AWS CloudWatch** permettono di:

- **Identificare pattern di traffico sospetti**, come picchi improvvisi di richieste.
- **Generare allarmi automatici** in caso di attività anomale.
- **Analizzare log e richieste** per migliorare le strategie di difesa.

Conclusione

Combinando queste soluzioni, il sistema MS3 può essere protetto da attacchi DoS/DDoS, garantendo una maggiore disponibilità e stabilità del servizio. L'uso di un **WAF**, un **sistema**

di rate limiting, e il **monitoraggio del traffico** consentirà di mitigare le minacce e assicurare un accesso continuo al personale ospedaliero senza interruzioni critiche.

4.3 Allineamento con gli standard industriali

Le misure di sicurezza implementate e quelle proposte sono state valutate rispetto agli standard di settore per garantire un adeguato livello di protezione dei dati e della continuità operativa.

Standard di riferimento

L'analisi della sicurezza dell'applicazione è stata condotta considerando i seguenti framework e normative:

- **ISO/IEC 27001** – Standard internazionale per la gestione della sicurezza delle informazioni.
- **NIST Cybersecurity Framework** – Linee guida per la gestione del rischio informatico.
- **OWASP Top 10** – Principali vulnerabilità per le applicazioni web.
- **GDPR** – Regolamento generale sulla protezione dei dati.

Conformità attuale e gap analysis

Di seguito è riportata un'analisi della conformità alle best practice e agli standard sopra citati:

Standard	Conformità Attuale	Aree di miglioramento
ISO/IEC 27001	**PC -** (Parzialmente Conforme – Basso)	Mancano audit log strutturati per il monitoraggio degli accessi e delle modifiche ai dati sensibili. Assente una gestione sicura dei segreti (password e certificati) e un piano di recovery dettagliato.
NIST Cybersecurity Framework	**PC -** (Parzialmente Conforme – Basso)	Assenti misure di cifratura dei dati a riposo e in transito, non implementata una gestione strutturata della sicurezza operativa (es. gestione segreti, monitoraggio attacchi).
OWASP Top 10	**PC -** (Parzialmente Conforme – Basso)	Individuata vulnerabilità di SQL Injection in query manuali. Necessario migliorare la validazione dell'input e implementare misure per prevenire attacchi DoS.
GDPR	**PC -** (Parzialmente Conforme – Basso)	Mancano strategie di anonimizzazione e cifratura dei dati per proteggere le informazioni personali in caso di compromissione. Assente una gestione sicura della conservazione delle credenziali.

Azioni per il miglioramento della conformità

Per migliorare l'aderenza agli standard di sicurezza e ridurre le vulnerabilità identificate, si raccomandano le seguenti azioni:

1. **Implementazione di strategie di cifratura e anonimizzazione dei dati**

    - Applicare tecniche di K-Anonymity, L-Diversity e T-Closeness per proteggere i dati sensibili.
    - Abilitare la cifratura dei volumi Docker con LUKS per proteggere i dati a riposo.
    - Assicurare la cifratura delle connessioni con HTTPS, utilizzando certificati SSL validi.
2. **Miglioramento della gestione dei segreti e delle credenziali**
    - Evitare la memorizzazione di password e certificati nei repository di codice.
    - Utilizzare variabili d'ambiente o vault di gestione segreti per proteggere informazioni sensibili.
    - Applicare una gestione centralizzata dei segreti con strumenti come HashiCorp Vault o AWS Secrets Manager.
3. **Implementazione di misure per la protezione contro attacchi informatici**
    - Validazione dell'input per prevenire SQL Injection, specialmente nelle query manuali.
    - Adozione di un Web Application Firewall (WAF) per filtrare richieste malevole e mitigare attacchi DoS/DDoS.
    - Configurazione di rate limiting e throttling per prevenire abusi delle API.
4. **Miglioramento della sicurezza dell'autenticazione**
    - Implementare Autenticazione a Due Fattori (2FA) utilizzando Google Authenticator.
    - Rafforzare le politiche di gestione delle password, richiedendo password complesse e aggiornamenti periodici.
5. **Monitoraggio e logging della sicurezza**
    - Introdurre strumenti di logging centralizzato (ad es. ELK Stack, Prometheus + Grafana) per individuare attività sospette.
    - Abilitare il monitoraggio degli accessi e delle modifiche ai dati sensibili, con audit log dettagliati.
6. **Allineamento con gli standard ISO 27001 e NIST**
    - Definire una policy di gestione della sicurezza delle informazioni basata sulle best practice di ISO 27001.
    - Introdurre un piano di risposta agli incidenti e una strategia di disaster recovery.

L'implementazione di queste azioni contribuirà a ridurre il rischio di attacchi informatici, garantire la conformità alle normative di settore e migliorare la fiducia nell'affidabilità del sistema.

Benefici del allineamento agli standard

L'adozione di queste misure porterà i seguenti benefici:

- Maggiore sicurezza e riduzione del rischio di violazioni.
- Conformità con le normative, riducendo il rischio di sanzioni.
- Migliore fiducia da parte degli utenti e dei clienti.
- Aumento della resilienza operativa contro le minacce informatiche.

5 Strategie di mitigazione del rischio

5.1 Piano di rimedio proposto

In questa sezione proponiamo una pianificazione delle attività più comprensiva possibile ai fini di fornire tutte le informazioni necessarie in fase esecutiva. Le scadenze sono state ideate come multipli di 2 settimane, ovvero la durata generica di uno sprint.

Fase 1: Mitigazione delle vulnerabilità critiche

Questa fase si concentra sulla risoluzione delle problematiche con il più alto impatto sulla sicurezza del sistema.

Validazione estesa dell'input nelle API

- **Descrizione del rischio:** Il sistema è vulnerabile a SQL Injection a causa della mancanza di controlli rigorosi sull'input nelle API.
- **Azione di rimedio:** Integrare query parametrizzate in tutte le API e utilizzare espressioni regolari per la validazione degli input. Effettuare code review per verificare l'applicazione completa delle modifiche.
- **Responsabile:** Team backend
- **Scadenza:** 1 mese
- **Risorse:** Revisione del codice, strumenti di testing di sicurezza (OWASP Dependency-Check).
- **Verifica del successo:** Nessuna vulnerabilità SQL rilevata nei test di sicurezza.

Cifratura dei dati sensibili nel database

- **Descrizione del rischio:** Dati personali come nome, cognome e codice fiscale sono memorizzati in chiaro, aumentando il rischio di esposizione in caso di compromissione del database.
- **Azione di rimedio:** Implementare la cifratura per i campi sensibili. Integrare una gestione sicura delle chiavi di cifratura per proteggere l'accesso ai dati crittografati.
- **Responsabile:** DBA (Database Admin)
- **Scadenza:** 2 settimane
- **Risorse:** Algoritmi di cifratura, infrastruttura di gestione delle chiavi.
- **Verifica del successo:** Dati sensibili crittografati e accessibili solo tramite chiavi autorizzate.

Fase 2: Rafforzamento della sicurezza operativa

Questa fase mira a migliorare la tracciabilità delle operazioni e la protezione degli account privilegiati.

Implementazione dell'audit trail e del logging centralizzato

- **Descrizione del rischio:** La mancanza di un audit trail rende difficile rilevare attività sospette e ricostruire eventi critici.
- **Azione di rimedio:** Integrare un sistema di audit trail e logging utilizzando strumenti come ELK Stack (Elasticsearch, Logstash, Kibana). Monitorare tutte le operazioni privilegiate e generare alert automatici per attività anomale.

- **Responsabile:** Team sicurezza IT
- **Scadenza:** 2 mesi
- **Risorse:** ELK Stack, personale di monitoraggio.
- **Verifica del successo:** Generazione di report di audit mensili con tracciabilità completa delle attività critiche.

Implementazione della Two-Factor Authentication (2FA)

- **Descrizione del rischio:** Gli utenti privilegiati sono esposti a tentativi di spoofing e accessi non autorizzati.
- **Azione di rimedio:** Integrare un sistema di 2FA per gli utenti con privilegi elevati, utilizzando Google Authenticator.
- **Responsabile:** Team sicurezza IT
- **Scadenza:** 2 settimane
- **Risorse:** Integrazione di Google Authenticator, aggiornamento delle policy di autenticazione.
- **Verifica del successo:** Accesso tramite 2FA abilitato per il 100% degli utenti privilegiati.

Fase 3: Miglioramento della disponibilità e della resilienza del sistema

Questa fase si concentra sulla protezione contro gli attacchi DoS e sull'introduzione di ulteriori misure di sicurezza infrastrutturale.

Protezione contro attacchi DoS (rate limiting)

- **Descrizione del rischio:** Il sistema è vulnerabile a tentativi di sovraccarico che potrebbero compromettere la disponibilità del servizio.
- **Azione di rimedio:** Integrare rate limiting e configurare un sistema di monitoraggio del traffico per identificare picchi anomali.
- **Responsabile:** Team infrastruttura
- **Scadenza:** 3 mesi
- **Risorse:** Strumenti di rate limiting (es. Bucket4j), monitoraggio del traffico.
- **Verifica del successo:** Riduzione degli eventi di overload e assenza di downtime significativo, possibilmente a fronte di un test estensivo.

Cifratura del volume Docker

- **Descrizione del rischio:** Attualmente, il sistema non dispone di cifratura per i volumi Docker. In caso di compromissione fisica del server, i dati archiviati potrebbero essere accessibili senza alcuna protezione crittografica. Sebbene il server si trovi in un ambiente protetto, questa vulnerabilità infrastrutturale rappresenta un rischio potenziale significativo.
- **Azione di rimedio:** Implementare la cifratura dei volumi utilizzando **LUKS (Linux Unified Key Setup)** per proteggere i dati sensibili archiviati nei container Docker. Pianificare la cifratura selettiva dei volumi più critici per ridurre l'impatto sulle performance.
- **Responsabile:** Team infrastruttura
- **Scadenza:** 2 settimane

- **Risorse necessarie:** Configurazione di LUKS, gestione delle chiavi di cifratura, testing delle performance.
- **Verifica del successo:** I dati archiviati risultano cifrati e accessibili solo tramite chiavi autorizzate, senza impatto significativo sulle performance.

5.2 Accettazione del rischio e rischio residuo

Durante l'applicazione del piano di rimedio ma anche nel suo parziale completamento, alcune misure di mitigazione non saranno implementate esaustivamente in ogni loro ambito, e altre potrebbero essere tralasciate. In questi casi, il rischio residuo viene temporaneamente accettato, con la consapevolezza che sarà monitorato e valutato nuovamente in seguito attraverso verifiche programmate. Le attività di revisione consentiranno di valutare l'efficacia delle misure implementate e, se necessario, pianificare ulteriori interventi correttivi.

Protezione contro attacchi DoS

Attualmente, il sistema MS3 non dispone di un meccanismo di **rate limiting** né di un Web Application Firewall (WAF). La configurazione del rate limiting è stata pianificata nei prossimi mesi per ridurre il rischio di sovraccarico. Nel frattempo, il rischio residuo viene accettato per via dell'improbabilità dell'impatto, ma sono importanti monitoraggi regolari del traffico di rete, con report mensili per individuare attività sospette.
Una revisione della situazione sarà effettuata entro **6 mesi**, per valutare i progressi e stabilire la necessità di adottare ulteriori misure come l'integrazione di un WAF.

Two-Factor Authentication (2FA)

La **2FA** per gli utenti con privilegi elevati è pianificata e sarà implementata nei prossimi **3 mesi**. Tuttavia, per gli utenti standard, questa protezione non è ancora prevista.
Il rischio residuo per gli utenti standard è accettato temporaneamente, poiché le loro operazioni hanno un impatto limitato sulla sicurezza generale. La revisione delle policy di accesso e l'estensione progressiva della 2FA agli utenti standard saranno valutate ogni **12 mesi**, sulla base delle risorse disponibili e dell'evoluzione delle minacce.

Cifratura del volume Docker

La cifratura dei volumi Docker non è ancora implementata, ma è già stata pianificata come parte di un progetto infrastrutturale a lungo termine. Fino a quando non sarà completata, il rischio residuo è accettato per via dell'improbabilità dell'impatto.
Saranno effettuate **verifiche annuali delle policy di sicurezza fisica** per garantire la protezione del sistema, e la situazione sarà rivalutata nel contesto degli aggiornamenti infrastrutturali futuri.

---

6 Risposta e recovery ad incidenti

Il piano di risposta agli incidenti è un componente essenziale per la gestione della sicurezza in un ambiente dinamico. Esso definisce le modalità con cui il sistema risponde a minacce o attacchi e si ripristina da incidenti, al fine di ridurre i danni e ripristinare la normalità operativa. Poiché molte delle misure discusse nei capitoli precedenti sono ancora in fase di pianificazione, anche la risposta agli incidenti è basata su procedure che dovranno essere implementate nel futuro prossimo.

6.1 Monitoraggio e rilevamento degli incidenti

Il monitoraggio e il rilevamento sono fondamentali per identificare tempestivamente le minacce. Sebbene alcune delle soluzioni proposte, come l'integrazione di un sistema di audit trail centralizzato e strumenti avanzati di monitoraggio, debbano ancora essere implementate, è prevista la loro attivazione a breve. L'introduzione di questi strumenti permette una rilevazione più rapida e una risposta più efficace agli incidenti.

Logging e audit trail

Attualmente, non è presente un sistema completo di logging centralizzato. Tuttavia, è prevista l'integrazione di una soluzione basata su **ELK Stack**, che permetterà di:

- Registrare tutte le attività critiche, come accessi privilegiati e modifiche ai dati sensibili.
- Monitorare l'integrità dei dati attraverso un sistema di audit trail.
- Generare allarmi automatici per eventi sospetti.

La pianificazione dell'implementazione è fissata per i prossimi mesi.

Protezione degli accessi

L'implementazione della **Two-Factor Authentication (2FA)** è pianificata per gli utenti con privilegi elevati. Attualmente, l'accesso al sistema è solo monitorato, ma la protezione attraverso 2FA ridurrà significativamente il rischio di accessi non autorizzati per gli utenti privilegiati.

6.2 Piano di risposta

Il piano di risposta si compone di diverse fasi che permetteranno di rispondere rapidamente a qualsiasi tipo di incidente, riducendo i rischi e i danni associati. Questo piano è ancora in fase di pianificazione, ma una volta implementato sarà strutturato in 4 fasi principali: identificazione, contenimento, mitigazione e ripristino.

Fase 1: Identificazione e contenimento

In questa fase, l’obiettivo è identificare rapidamente l’incidente e contenerlo per prevenire danni ulteriori. Sebbene gli strumenti per il monitoraggio non siano ancora attivi, una volta implementati, questi permetteranno di:

- Rilevare attività sospette attraverso alert automatici.
- Isolare rapidamente il sistema compromesso (ad esempio, disabilitando un server compromesso o una macchina infetta).
- Avviare un’indagine approfondita per comprendere la portata dell'incidente.

Azione pianificata: Integrare strumenti di monitoraggio delle operazioni privilegiate e creare protocolli di isolamento in caso di incidente.

Fase 2: Analisi e mitigazione

Dopo aver identificato l’incidente, la fase successiva consiste nell’analizzare la causa e mitigare i danni. Le misure per questa fase includono:

- Analisi dei log per identificare come e dove si è verificato l'attacco.
- Attuazione di patch e correzioni tempestive su qualsiasi vulnerabilità sfruttata.
- Implementazione di soluzioni immediate per limitare i danni (ad esempio, disabilitare temporaneamente un servizio compromesso).

Azione pianificata: Quando attivato, il sistema di logging centralizzato fornirà il supporto necessario per eseguire una corretta analisi degli incidenti.

Fase 3: Ripristino delle operazioni

Una volta mitigato l'incidente, sarà necessario ripristinare il sistema alle condizioni operative normali. Sebbene la soluzione per il ripristino da backup non sia ancora formalizzata, sono previste le seguenti azioni di recovery:

- Ripristino da backup crittografati per i dati sensibili.
- Validazione della integrità del sistema per garantire che non ci siano tracce residue di compromissioni.
- Monitoraggio intensivo per alcune settimane dopo il ripristino per assicurarsi che il sistema non venga nuovamente compromesso.

Azione pianificata: Test di recupero da backup su base regolare per garantire l'efficacia delle soluzioni di recovery.

Fase 4: Revisione e miglioramento

Questa fase si concentra sulla revisione dell'incidente e sul miglioramento continuo del piano di risposta. Sarà effettuata una valutazione completa dell'incidente per capire le cause, i punti di vulnerabilità e le misure correttive necessarie per prevenire future minacce. Le principali attività includono:

- Redazione di un rapporto post-mortem per documentare l'incidente e la risposta adottata.
- Aggiornamento delle procedure di risposta agli incidenti, se necessario.
- Pianificazione di simulazioni di incidenti per migliorare la reattività e l'efficacia del team.

Azione pianificata: Creazione di un piano di formazione continua per il personale riguardo alle best practices di sicurezza e risposta agli incidenti.

Appendice A - Acronimi e glossario

Termine	Definizione
DDoS	Distributed Denial of Service
DoS	Denial of Service
GDPR	General Data Protection Regulation
JWT	JSON Web Token
SAR	Security Assessment Report
WAF	Web Application Firewall