SECURITY AUDIT REPORT

Sistema Gestionale Weiss Cafè

Data Audit: 11 Gennaio 2026 **Versione:** 1.0 **Classificazione:** CONFIDENZIALE **Auditor:** Claude Opus 4.5 - AI Security Analysis

EXECUTIVE SUMMARY

Questo report presenta un'analisi approfondita della sicurezza del Sistema Gestionale Weiss Cafè, un'applicazione Next.js 16 con PostgreSQL (Supabase) per la gestione contabile e amministrativa.

Valutazione Complessiva

Categoria	Score	Rischio
Autenticazione	6.5/10	MEDIO-ALTO
Autorizzazione (RBAC)	7.0/10	MEDIO
Protezione API	5.5/10	ALTO
Gestione Dati Sensibili	4.5/10	CRITICO
Backup & Recovery	3.0/10	CRITICO
Compliance GDPR	4.0/10	CRITICO
SCORE COMPLESSIVO	**5.1/10**	**ALTO**

Pagina 1 di 18

Riepilogo Vulnerabilità Identificate

Livello	Quantità	Azioni Richieste
CRITICO	11	Immediate (24-48h)
ALTO	10	Breve termine (1-2 settimane)
MEDIO	12	Medio termine (1-2 mesi)
BASSO	6	Lungo termine (3+ mesi)

SEZIONE 1: VULNERABILITÀ CRITICHE

1.1 Assenza Protezione Brute Force Login

File: `src/lib/auth.ts` **Rischio:** CRITICO **CVSS Score:** 8.1

Descrizione: Il sistema di autenticazione non implementa alcuna protezione contro attacchi brute force. Un attaccante può tentare infinite combinazioni di credenziali senza limitazioni.

Impatto:

- Compromissione account utente
- Accesso non autorizzato a dati finanziari
- Violazione dati personali (GDPR)

Raccomandazione:

```
// Implementare rate limiting con Redis
import { Ratelimit } from "@upstash/ratelimit";
import { Redis } from "@upstash/redis";

const ratelimit = new Ratelimit({
  redis: Redis.fromEnv(),
  limiter: Ratelimit.slidingWindow(5, "15 m"), // 5 tentativi / 15 min
  analytics: true,
});
```

1.2 Password Default Esposta in API Response

File: `src/app/api/users/route.ts` (linea 260-267) **Rischio:** CRITICO **CVSS Score:** 9.1

Pagina 2 di 18

Descrizione: Quando viene creato un nuovo utente, la password di default (`1234567890`) viene restituita in chiaro nella risposta JSON dell'API.

Codice Vulnerabile:

```
return NextResponse.json({
  credentials: {
    username,
    password: DEFAULT_PASSWORD, // ESPOSTO!
  },
})
```

Impatto:

- Password visibile nei log del browser
- Intercettabile se connessione non HTTPS
- Violazione principio "zero-knowledge"

Raccomandazione:

- Non restituire MAI password in plain text
- Implementare sistema di token temporanei via email
- Generare password casuali per ogni utente

1.3 Assenza Protezione CSRF

File: Tutti gli endpoint POST/PATCH/DELETE **Rischio:** CRITICO **CVSS Score:** 8.8

Descrizione: Gli endpoint API non verificano token CSRF. Un attaccante può creare una pagina malevola che esegue azioni per conto dell'utente autenticato.

Impatto:

- Creazione/eliminazione utenti non autorizzata
- Modifica dati finanziari
- Trasferimenti fondi fraudolenti

Raccomandazione:

Pagina 3 di 18

```
// Implementare middleware CSRF
import { csrf } from '@/lib/csrf';

export async function POST(req: Request) {
  const csrfValid = await csrf.verify(req);
  if (!csrfValid) {
    return new Response('CSRF token invalid', { status: 403 });
  }
  // ... resto del codice
}
```

1.4 Rate Limiting Non Funzionante in Produzione

File: `src/lib/rate-limit.ts` **Rischio:** CRITICO **CVSS Score:** 7.5

Descrizione: Il rate limiting è implementato in-memory, ma su Vercel (serverless) ogni istanza ha memoria separata. Il rate limiting è quindi inefficace.

Impatto:

- Attacchi brute force possibili
- DoS tramite flooding di richieste
- Abuse di endpoint costosi (import XML)

Raccomandazione: Migrare a soluzione Redis-based (Upstash Ratelimit).

1.5 Secrets Esposti nel Repository

File: `.env` **Rischio:** CRITICO **CVSS Score:** 9.8

Descrizione: Il file `.env` contiene credenziali di produzione in chiaro:

- Password database Supabase
- Secret NextAuth
- API Key Google Maps

Impatto:

- Accesso completo al database di produzione
- Compromissione totale del sistema
- Furto dati utenti

Azioni Immediate:

Pagina 4 di 18

1. **CAMBIARE IMMEDIATAMENTE** la password del database Supabase
2. Rigenerare NEXTAUTH_SECRET e AUTH_SECRET
3. Verificare che `.env` sia in `.gitignore`
4. Controllare git history per rimuovere secrets precedenti

1.6 Mancanza Security Headers

File: `next.config.ts` **Rischio:** CRITICO **CVSS Score:** 7.1

Descrizione: L'applicazione non implementa header di sicurezza HTTP essenziali.

Header Mancanti:

- `X-Content-Type-Options: nosniff`
- `X-Frame-Options: DENY`
- `Content-Security-Policy`
- `Strict-Transport-Security`
- `X-XSS-Protection`

Raccomandazione:

```
// next.config.ts
const securityHeaders = [
  { key: 'X-Content-Type-Options', value: 'nosniff' },
  { key: 'X-Frame-Options', value: 'DENY' },
  { key: 'X-XSS-Protection', value: '1; mode=block' },
  { key: 'Strict-Transport-Security', value: 'max-age=31536000; includeSubDomains' },
  { key: 'Content-Security-Policy', value: "default-src 'self'" },
];
```

SEZIONE 2: VULNERABILITÀ ALTO RISCHIO

2.1 IDOR (Insecure Direct Object References)

File: Multiple API routes **Rischio:** ALTO

Diversi endpoint permettono accesso a risorse basandosi su ID senza verificare adeguatamente i permessi dell'utente.

Pagina 5 di 18

2.2 XXE Risk nel Parser XML Fatture

File: `src/app/api/invoices/route.ts` **Rischio:** ALTO

Il contenuto XML delle fatture elettroniche viene processato senza validazione DTD, esponendo a potenziali attacchi XXE.

2.3 Dati GPS Memorizzati in Chiaro

File: `src/app/api/attendance/punch/route.ts` **Rischio:** ALTO

Coordinate GPS e indirizzi IP dei dipendenti sono memorizzati senza crittografia, violando GDPR Art. 32.

2.4 Sessione JWT Troppo Lunga

File: `src/lib/auth.ts` **Rischio:** ALTO

La sessione JWT dura 30 giorni senza possibilità di invalidazione server-side.

2.5 Stack Trace Esposti negli Errori

File: Tutti gli endpoint API **Rischio:** ALTO

Gli errori vengono loggati con stack trace completi, esponendo informazioni sulla struttura interna del sistema.

Pagina 6 di 18

SEZIONE 3: CONFRONTO OWASP TOP 10:2025

Mappatura Vulnerabilità vs OWASP

OWASP 2025	Categoria	Stato nel Progetto
A01:2025	Broken Access Control	⚠️ PARZIALE - IDOR presenti
A02:2025	Security Misconfiguration	❌ CRITICO - Headers mancanti
A03:2025	Software Supply Chain	✅ OK - Dipendenze aggiornate
A04:2025	Cryptographic Failures	⚠️ PARZIALE - Encryption mancante
A05:2025	Injection	✅ OK - Prisma ORM protegge
A06:2025	Insecure Design	⚠️ MEDIO - Audit trail assente
A07:2025	Authentication Failures	❌ CRITICO - No rate limit
A08:2025	Data Integrity Failures	⚠️ MEDIO - No checksum
A09:2025	Security Logging Failures	❌ CRITICO - Audit assente
A10:2025	Exceptional Conditions	⚠️ MEDIO - Error handling migliorabile

SEZIONE 4: SISTEMA DI AUDIT LOG

4.1 Situazione Attuale

Stato: ❌ NON IMPLEMENTATO

Il sistema attualmente traccia solo:

- `lastLoginAt` per utenti
- `createdAt/updatedAt` per record

Mancano:

- Log di tutte le azioni utente
- Tracciamento accessi a dati sensibili
- Log delle modifiche ai dati
- Log dei tentativi di login falliti

Pagina 7 di 18

- Log delle operazioni amministrative

4.2 Requisiti GDPR per Audit Log

Secondo GDPR Art. 5(f) e Art. 32, è necessario implementare:

1. **Chi** ha eseguito l'azione (userId, IP, userAgent)
2. **Cosa** è stato fatto (action type, entità, campo)
3. **Quando** è successo (timestamp preciso)
4. **Perché** (se disponibile, es. approvazione)
5. **Risultato** (success/failure)

Pagina 8 di 18

4.3 Proposta Implementazione Audit Log

Schema Database

```
model AuditLog {
  id            String   @id @default(cuid())
  timestamp     DateTime @default(now())

  // Chi
  userId        String?
  user          User?    @relation(fields: [userId], references: [id])
  ipAddress     String?
  userAgent     String?
  sessionId     String?

  // Cosa
  action        AuditAction
  entityType    String       // "User", "DailyClosure", "Invoice"...
  entityId      String?
  field         String?      // Campo specifico modificato

  // Valori
  oldValue      Json?        // Valore precedente (criptato per dati sensibili)
  newValue      Json?        // Nuovo valore (criptato per dati sensibili)

  // Contesto
  result        AuditResult  @default(SUCCESS)
  errorMessage  String?
  metadata      Json?        // Info aggiuntive

  @@index([userId])
  @@index([entityType, entityId])
  @@index([timestamp])
  @@index([action])
}

enum AuditAction {
  // Autenticazione
  LOGIN_SUCCESS
  LOGIN_FAILED
  LOGOUT
  PASSWORD_CHANGED
  PASSWORD_RESET

  // CRUD
  CREATE
  READ
  UPDATE
  DELETE

  // Operazioni specifiche
  CLOSURE_SUBMITTED
  CLOSURE_VALIDATED
  INVOICE_IMPORTED
  PAYMENT_RECORDED
```


Pagina 9 di 18

```
  // Amministrazione
  USER_ROLE_CHANGED
  USER_DEACTIVATED
  PERMISSION_GRANTED
  PERMISSION_REVOKED
}

enum AuditResult {
  SUCCESS
  FAILURE
  DENIED
  ERROR
}
```

Componente React per Visualizzazione (Solo Admin)

Proposta per tab "Log Attività" nel pannello admin:

```
// /src/app/(dashboard)/admin/audit-log/page.tsx
export default function AuditLogPage() {
  return (
    <div className="space-y-6">
      <h1>Log Attività Sistema</h1>

      {/* Filtri */}
      <AuditLogFilters />

      {/* Tabella Log */}
      <AuditLogTable />

      {/* Export */}
      <ExportAuditLogButton />
    </div>
  );
}
```

4.4 Best Practices Audit Log

Basato su ricerca GDPR Log Management:

1. **Retention Period:** 12 mesi standard, configurabile per compliance
2. **Immutabilità:** Log append-only, nessuna modifica/cancellazione
3. **Crittografia:** Dati sensibili nei log devono essere mascherati
4. **Accesso:** Solo admin può visualizzare, con log di chi accede ai log
5. **Alerting:** Notifiche automatiche per azioni sospette

Pagina 10 di 18

SEZIONE 5: TIME MACHINE - RECOVERY DATI

5.1 Situazione Attuale

Stato: ❌ CRITICO - NESSUN MECCANISMO

Problemi Identificati:

- Hard delete per molte entità (fatture, prodotti, fornitori)
- Cascade delete aggressivi nello schema
- Nessun soft delete per dati contabili
- Nessun "cestino" per recupero
- Backup solo via Supabase (7 giorni)

5.2 Opzioni di Implementazione

OPZIONE A: Soft Delete Universale (Consigliato)

Costo Implementazione: Medio **Complessità:** Media **Recovery Time:** Immediato

```
// Aggiungere a tutte le entità critiche
model DailyClosure {
  // ... campi esistenti
  deletedAt     DateTime?
  deletedById   String?
  deletedBy     User?     @relation("DeletedClosures", fields: [deletedById],
references: [id])
}
```

Pro:

- Recovery immediato
- Nessuna perdita dati
- Audit trail integrato

Contro:

- Query più complesse (filtrare deletedAt)
- Database più grande

OPZIONE B: PostgreSQL PITR (Point-In-Time Recovery)

Costo Implementazione: Basso **Complessità:** Alta (gestione) **Recovery Time:** Ore

Basato su PostgreSQL PITR Documentation:

Pagina 11 di 18

```
-- Configurazione PostgreSQL
wal_level = replica
archive_mode = on
archive_command = 'cp %p /backup/wal/%f'
```

Pro:

- Recovery granulare al secondo
- Nessuna modifica schema
- Standard industriale

Contro:

- Recovery di tutto il database, non singoli record
- Richiede infrastruttura backup
- Tempi di recovery lunghi

OPZIONE C: Audit Trail con Versioning (Avanzato)

Costo Implementazione: Alto **Complessità:** Alta **Recovery Time:** Immediato

```
model DailyClosureVersion {
  id             String   @id @default(cuid())
  closureId      String
  closure        DailyClosure @relation(fields: [closureId], references: [id])
  version        Int
  data           Json     // Snapshot completo
  changedFields  String[] // Campi modificati
  changedAt      DateTime @default(now())
  changedById    String
  changedBy      User     @relation(fields: [changedById], references: [id])
  changeType     ChangeType

  @@index([closureId, version])
}

enum ChangeType {
  CREATE
  UPDATE
  DELETE
  RESTORE
}
```

Pro:

- Versioning completo
- Recovery selettivo
- Audit trail integrato

Pagina 12 di 18

- Confronto versioni

Contro:

- Complessità implementazione
- Storage significativo
- Performance write

5.3 Proposta Implementazione Ibrida

Raccomandazione: Combinare OPZIONE A + B + elementi di C

Fase 1: Soft Delete (Immediato)

1. Aggiungere `deletedAt`, `deletedById` alle entità critiche
2. Modificare API per soft delete
3. Creare endpoint `/api/trash` per visualizzare elementi eliminati
4. Implementare restore automatico

Fase 2: Supabase PITR (Breve termine)

1. Attivare Point-in-Time Recovery su Supabase (piano Pro)
2. Configurare retention 30 giorni
3. Documentare procedura di recovery

Fase 3: Versioning per Entità Critiche (Medio termine)

1. Implementare versioning per:
    - `DailyClosure` (chiusure cassa)
    - `JournalEntry` (prima nota)
    - `User` (profili utente)
2. UI per visualizzare cronologia modifiche
3. Funzione "Ripristina versione precedente"

Pagina 13 di 18

5.4 UI Proposta: Cestino e Time Machine

```
// /src/app/(dashboard)/admin/trash/page.tsx
export default function TrashPage() {
  return (
    <div className="space-y-6">
      <h1>Cestino</h1>
      <p>Elementi eliminati negli ultimi 30 giorni</p>

      <Tabs>
        <Tab label="Utenti" count={3} />
        <Tab label="Chiusure" count={12} />
        <Tab label="Fatture" count={45} />
        <Tab label="Prima Nota" count={8} />
      </Tabs>

      <TrashTable
        onRestore={(id) => restoreItem(id)}
        onPermanentDelete={(id) => permanentDelete(id)}
      />
    </div>
  );
}

// /src/app/(dashboard)/admin/time-machine/page.tsx
export default function TimeMachinePage() {
  return (
    <div className="space-y-6">
      <h1>Time Machine</h1>
      <p>Recupera dati da un punto nel tempo</p>

      <DateTimePicker
        label="Seleziona data e ora"
        onChange={setTargetTime}
      />

      <EntitySelector
        label="Tipo di dato"
        options={['Chiusure Cassa', 'Prima Nota', 'Utenti']}
      />

      <Button onClick={previewRestore}>
        Anteprima Ripristino
      </Button>
    </div>
  );
}
```


Pagina 14 di 18

SEZIONE 6: PIANO DI REMEDIATION

Fase 1: CRITICO (Entro 48 ore)

#	Azione	Responsabile	Effort
1	Cambiare password DB Supabase	DevOps	30 min
2	Rigenerare NEXTAUTH_SECRET	DevOps	15 min
3	Verificare .gitignore	Dev	15 min
4	Implementare rate limiting Redis	Dev	4 ore
5	Rimuovere password da API response	Dev	1 ora

Fase 2: ALTO (Entro 2 settimane)

#	Azione	Responsabile	Effort
6	Aggiungere Security Headers	Dev	2 ore
7	Implementare CSRF protection	Dev	4 ore
8	Fixare vulnerabilità IDOR	Dev	8 ore
9	Hardening XML parser	Dev	4 ore
10	Ridurre session duration	Dev	1 ora

Fase 3: MEDIO (Entro 2 mesi)

#	Azione	Responsabile	Effort
11	Implementare Audit Log system	Dev	3 giorni
12	Soft Delete per entità critiche	Dev	2 giorni
13	UI Admin per Audit Log	Dev	2 giorni
14	Crittografia dati sensibili	Dev	3 giorni
15	Attivare Supabase PITR	DevOps	2 ore

Pagina 15 di 18

Fase 4: OTTIMIZZAZIONE (3+ mesi)

#	Azione	Responsabile	Effort
16	Implementare 2FA per admin	Dev	2 giorni
17	Versioning entità critiche	Dev	5 giorni
18	UI Time Machine	Dev	3 giorni
19	Password expiration policy	Dev	1 giorno
20	Penetration testing esterno	Security	-

SEZIONE 7: COMPLIANCE E NORMATIVE

7.1 GDPR Compliance Checklist

Articolo	Requisito	Stato	Azione
Art. 5(f)	Integrità e confidenzialità	❌	Encryption + Audit
Art. 13	Informativa raccolta dati	❌	Privacy policy
Art. 17	Diritto all'oblio	⚠️	Soft delete
Art. 20	Portabilità dati	❌	Export function
Art. 32	Misure sicurezza	⚠️	Questo audit
Art. 33	Notifica breach	❌	Procedura

7.2 Normativa Italiana

Normativa	Requisito	Stato
D.Lgs. 196/2003	Misure minime sicurezza	⚠️
Art. 2135-bis CC	Tracciabilità operazioni	❌
Conservazione SDI	10 anni fatture	⚠️

Pagina 16 di 18

SEZIONE 8: CONCLUSIONI

Punti di Forza Identificati

1. ✅ Uso di Prisma ORM (previene SQL injection)
2. ✅ NextAuth per autenticazione
3. ✅ Zod per validazione input
4. ✅ RBAC implementato
5. ✅ Password hashing con bcrypt

Criticità Principali

1. ❌ **Secrets esposti** - Rischio compromissione totale
2. ❌ **No rate limiting** - Vulnerabile a brute force
3. ❌ **No audit trail** - Impossibile tracciare azioni
4. ❌ **No backup strategy** - Rischio perdita dati
5. ❌ **GDPR parziale** - Rischio sanzioni

Raccomandazione Finale

Il sistema **NON è pronto per produzione** nel suo stato attuale. È necessario completare almeno le Fasi 1 e 2 del piano di remediation prima di esporre il sistema a utenti reali.

Priorità Assoluta:

1. Cambiare credenziali database IMMEDIATAMENTE
2. Implementare rate limiting
3. Rimuovere secrets dal codice

APPENDICE A: RIFERIMENTI

Standard e Framework

- OWASP Top 10:2025
- GDPR Compliance Framework
- PostgreSQL PITR

Pagina 17 di 18

Best Practices Consultate

- Audit Log Best Practices
- GDPR Logging and Monitoring
- Security Log Retention

APPENDICE B: GLOSSARIO

Termine	Definizione
CSRF	Cross-Site Request Forgery
IDOR	Insecure Direct Object Reference
PITR	Point-In-Time Recovery
WAL	Write-Ahead Log
XXE	XML External Entity
RBAC	Role-Based Access Control

Fine Documento

Report generato automaticamente da Claude Opus 4.5 Data: 11 Gennaio 2026

Pagina 18 di 18