🎯 Checklist d'Implémentation - Corrections Sécurisées

📋 Vue d'ensemble

Cette checklist vous guide pas-à-pas pour appliquer toutes les corrections de sécurité et d'architecture.

Durée estimée : 2-3 heures
Difficulté : Moyenne
Risque : Faible (tests E2E inclus)

✅ Phase 1 : Préparation (15 min)

1.1 Backup & Branch

```bash
# Créer une branche de travail
git checkout -b security/crypto-improvements

# Backup de la DB (si prod)
pg_dump oliver_db > backup_$(date +%Y%m%d).sql
```

1.2 Mise à jour des dépendances

```bash
# Frontend
cd apps/web
pnpm add -D @svgr/webpack eslint-plugin-boundaries
pnpm remove class-validator class-transformer  # ❌ Si présents

# Backend
cd apps/api
pnpm remove class-validator class-transformer  # ✅ OBLIGATOIRE
pnpm add raw-body  # Pour webhook middleware

# Vérifier les versions
pnpm list next react zod
```

1.3 Variables d'environnement

```bash
# apps/api/.env
echo "SECURITY_BCRYPT_ROUNDS=12" >> .env
echo "KYC_WEBHOOK_SECRET=your-webhook-secret-here" >> .env
echo "ENCRYPTION_KEY=your-32-char-encryption-key-here" >> .env

# Générer un encryption key sécurisé
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"
```

✅ Phase 2 : Corrections Frontend (30 min)

2.1 Next.js 15 - Webpack SVGR

- [ ] Mettre à jour `apps/web/package.json`
  - `next: ^15.0.0`
  - `react: ^19.0.0`
  - `react-dom: ^19.0.0`
  - Ajouter `@svgr/webpack: ^8.1.0`
- [ ] Remplacer `apps/web/next.config.js`
  - Supprimer bloc `experimental.turbo`
  - Ajouter règle `webpack` pour SVGR

```bash
# Tester le build
cd apps/web
pnpm build
```

2.2 ESLint Boundaries

- [ ] Copier `.eslintrc.json` depuis l'artifact
- [ ] Tester les règles

```bash
cd apps/web
pnpm lint
# Devrait détecter les imports interdits (ui → admin)
```

2.3 Migration Zod (Frontend)

- [ ] Vérifier que tous les forms utilisent Zod (pas class-validator)
- [ ] Centraliser les schémas dans `lib/schemas/`

✅ Phase 3 : Crypto Utility (45 min)

3.1 Créer CryptoUtil

- [ ] Copier `apps/api/src/common/utils/crypto.util.ts`
- [ ] Vérifier les imports (crypto, bcrypt)

3.2 Middleware Raw Body

- [ ] Créer `apps/api/src/common/middleware/raw-body.middleware.ts`
- [ ] Mettre à jour `apps/api/src/main.ts`

```typescript
// AVANT json parser
app.use('/api/admin/kyc/webhook', express.raw({ type: 'application/json' }));
```

3.3 Tests unitaires CryptoUtil

```bash
# Créer test file
touch apps/api/src/common/utils/crypto.util.spec.ts

# Lancer tests
cd apps/api
pnpm test crypto.util
```

Tests minimaux à écrire :

- [ ] `hashPassword` + `comparePassword`
- [ ] `sha256` avec string et object
- [ ] `createAuditLogHash` avec previousHash
- [ ] `verifyHmacSignature` (valid + invalid)
- [ ] `encrypt` → `decrypt` round-trip

✅ Phase 4 : Migration Base de Données (20 min)

4.1 Ajouter colonne `hashedAt`

```bash
cd packages/database

# Créer migration
npx prisma migrate dev --name add_hashedAt_to_audit_log

# Vérifier le SQL généré
cat prisma/migrations/XXX_add_hashedAt_to_audit_log/migration.sql
```

- [ ] Vérifier que le SQL contient :
  - `ADD COLUMN "hashedAt" TIMESTAMP(3)`
  - `CREATE INDEX "AuditLog_hashedAt_idx"`
  - `UPDATE` pour backfill

4.2 Générer Prisma Client

```bash
npx prisma generate
```

4.3 Vérifier le schéma

```bash
npx prisma studio
# Ouvrir AuditLog → vérifier colonne hashedAt
```

✅ Phase 5 : Services Backend (60 min)

5.1 Audit Log Service

- [ ] Mettre à jour `audit-log.service.ts`
  - Utiliser `CryptoUtil.createAuditLogHash`
  - Stocker `hashedAt` exact
  - Implémenter `verifyIntegrity()` amélioré

```typescript
// Test rapide
const result = await auditLogService.verifyIntegrity();
console.log(result); // { valid: true, totalEntries: X, errors: [] }
```

5.2 KYC Service (Webhooks)

- [ ] Mettre à jour `kyc.service.ts`
  - Utiliser `req.rawBody` (middleware)
  - `CryptoUtil.verifyHmacSignature(rawBody, signature, secret)`

5.3 Settings Service (API Keys)

- [ ] Mettre à jour `settings.service.ts`
  - `CryptoUtil.generateApiKey()` avec retry
  - `CryptoUtil.hashApiKey()` pour stockage

5.4 Users Service (Passwords)

- [ ] Remplacer tous les bcrypt directs

```typescript
// ❌ Avant
const hash = await bcrypt.hash(password, 10);

// ✅ Après
const hash = await CryptoUtil.hashPassword(password);
```

✅ Phase 6 : Tests E2E (45 min)

6.1 Setup Tests

```bash
cd apps/api
mkdir -p test/admin test/helpers
```

- [ ] Copier `test/helpers/test-helper.ts`
- [ ] Copier `test/jest-e2e.json`
- [ ] Copier `test/setup.ts`

6.2 Tests Audit Chain

- [ ] Copier `test/admin/audit-chain.e2e-spec.ts`
- [ ] Lancer : `pnpm run test:e2e:audit`

Tests critiques :

- ✅ Créer 3 logs → `verifyIntegrity() === true`
- ✅ Altérer `previousHash` → `valid === false`
- ✅ Altérer `hash` → erreur détectée
- ✅ 10+ entrées → chaîne valide

6.3 Tests Webhook HMAC

- [ ] Copier `test/admin/webhook-hmac.e2e-spec.ts`
- [ ] Lancer : `pnpm run test:e2e:webhook`

Tests critiques :

- ✅ Signature correcte → 200
- ✅ Signature incorrecte → 400
- ✅ Payload tampered → rejeté
- ✅ Timing-safe (pas de leak)

6.4 Tests AES Encryption

- [ ] Copier `test/admin/aes-encryption.e2e-spec.ts`
- [ ] Lancer : `pnpm run test:e2e:aes`

Tests critiques :

- ✅ Encrypt → Decrypt round-trip
- ✅ Unicode (emoji, cyrillique, arabe)
- ✅ Mauvaise clé → throw
- ✅ Ciphertext tampered → throw

6.5 Lancer tous les tests

```bash
pnpm run test:e2e:crypto
# Devrait afficher ~50+ tests passed
```

✅ Phase 7 : Refactoring Global (30 min)

7.1 Supprimer les imports directs

Chercher et remplacer tous les usages de crypto directs :

```bash
# Trouver les imports crypto directs
grep -r "import \* as crypto" apps/api/src/modules/

# Remplacer par CryptoUtil
```

Checklist :

- [ ] Aucun `crypto.createHash` direct (sauf dans CryptoUtil)
- [ ] Aucun `bcrypt.hash` direct (sauf dans CryptoUtil)
- [ ] Aucun `crypto.timingSafeEqual` direct

7.2 Audit de sécurité

```bash
# Chercher les patterns dangereux
grep -r "Math.random()" apps/api/src/
grep -r "JSON.stringify" apps/api/src/modules/admin/kyc/
grep -r "password.*plain" apps/api/src/
```

✅ Phase 8 : Documentation (15 min)

8.1 README technique

- [ ] Créer `docs/SECURITY.md`

markdown

```
# Security Guidelines

## Crypto Operations
All crypto operations MUST use `CryptoUtil`:
- Passwords: `CryptoUtil.hashPassword()`
- HMAC: `CryptoUtil.verifyHmacSignature()`
- AES: `CryptoUtil.encrypt()` / `decrypt()`
- Audit: `CryptoUtil.createAuditLogHash()`

## Audit Chain Integrity
- hashedAt timestamp is canonical
- Chain verified via verifyIntegrity()
- Never alter previousHash or hash manually
```

8.2 Comments dans le code

- [ ] Ajouter JSDoc sur méthodes publiques CryptoUtil
- [ ] Commenter les choix (IV=12 bytes, timing-safe, etc.)

✅ Phase 9 : Validation & Déploiement (30 min)

9.1 Tests complets

bash

```
# Backend
cd apps/api
pnpm test          # Unit tests
pnpm test:e2e      # E2E tests
pnpm build         # Compilation

# Frontend
cd apps/web
pnpm lint          # ESLint boundaries
pnpm type-check    # TypeScript
pnpm build         # Next.js build
```

9.2 Checklist pré-déploiement

- [ ] Tous les tests passent (unit + e2e)
- [ ] Build sans erreur (frontend + backend)
- [ ] Variables d'environnement configurées
- [ ] Migration DB testée (staging)
- [ ] Backup DB effectué

9.3 Déploiement progressif

```bash
# 1. DB Migration (production)
cd packages/database
npx prisma migrate deploy

# 2. Backend (avec rollback plan)
cd apps/api
pm2 start dist/main.js --name oliver-api

# 3. Frontend
cd apps/web
pm2 start npm --name oliver-web -- start

# 4. Vérifier les logs
pm2 logs oliver-api --lines 100
```

9.4 Tests post-déploiement

- [ ] Audit chain integrity : `GET /admin/audit-log/verify`
- [ ] Webhook test : envoyer un webhook Yoti de test
- [ ] API Key generation : créer une nouvelle clé admin
- [ ] Login : vérifier bcrypt avec SECURITY_BCRYPT_ROUNDS

✅ Phase 10 : Monitoring (15 min)

10.1 Alertes

Configurer des alertes pour :

- [ ] Échec de vérification audit chain
- [ ] Webhook signature invalide (pic de rejets)
- [ ] Échec de décryptage AES

10.2 Métriques

```typescript
// Ajouter métriques Prometheus/StatsD
cryptoOperations.inc({ operation: 'encrypt', status: 'success' });
auditChainVerifications.observe(duration);
webhookSignatureFailures.inc();
```

🚨 Rollback Plan

Si problème en production :

```bash
# 1. Rollback code
git revert HEAD
pm2 restart oliver-api

# 2. Rollback DB (si nécessaire)
psql oliver_db < backup_20250121.sql

# 3. Vérifier
curl https://api.oliver.com/health
```

📊 Résumé des Changements

Composant	Action	Impact
Next.js	Upgrade 14→15 + SVGR	Frontend build
CryptoUtil	Centralisé	Tous les modules
AuditLog	Colonne hashedAt	DB migration
Webhooks	Raw body middleware	KYC endpoints
API Keys	Génération sécurisée	Settings service
Tests	50+ tests E2E	Couverture crypto

✅ Validation Finale

Cocher quand tout est ✅ :

- [ ] **Frontend** : Build Next.js 15 OK, ESLint boundaries actif
- [ ] **Backend** : CryptoUtil centralisé, aucun crypto direct
- [ ] **Database** : Migration hashedAt déployée
- [ ] **Tests** : 50+ tests E2E passent
- [ ] **Services** : Tous utilisent CryptoUtil
- [ ] **Docs** : SECURITY.md créé
- [ ] **Déploiement** : Staging OK, Production OK
- [ ] **Monitoring** : Alertes configurées

🎉 C'est terminé !

Prochaines étapes recommandées :

1. Audit de sécurité externe (si budget)
2. Upgrade scrypt/argon2 pour KDF (remplacer SHA-256)
3. Implémenter key rotation pour API keys
4. Ajouter rate limiting sur webhooks

Questions ? Consulte les artifacts ou demande de l'aide !

📚 Ressources

- Next.js 15 Migration Guide
- Node.js Crypto Best Practices
- OWASP Cryptographic Storage Cheat Sheet
- GCM Mode IV Length

Dernière mise à jour : 21 octobre 2025
Version : 1.0.0
Auteur : Architecture & Security Team