**COMMUNAUTE FRANCAISE DE BELGIQUE**

**Institut des Carrières Commerciales**

Ville de Bruxelles

Rue de la Fontaine 4 - 1000 BRUXELLES

**LIVRABLE 16**

**STRATEGIE DE SECURITE**

*Epreuve intégrée réalisée en vue de l'obtention du titre de*

**Bachelier en Informatique de gestion**

Orientation développement d'applications

**MEFTAH Soufiane**

Année académique 2024-2025

# Table des matières

Introduction

1. Security by Design - Principes fondamentaux

1.1 Approche Security by Design

1.2 Implémentation dans FarmShop

1.3 Méthodologie de sécurisation

2. Systèmes d'authentification et gestion des identifiants

2.1 Architecture d'authentification Laravel Sanctum

2.2 Politique de mots de passe robustes

2.3 Authentification multi-facteurs (2FA)

2.4 Système de récupération des identifiants

2.5 Gestion des sessions sécurisées

3. Techniques d'autorisation et contrôle d'accès

3.1 Modele RBAC avec Spatie Laravel-Permission

3.2 Gates et Policies Laravel

3.3 Contrôle d'accès aux ressources

3.4 Ségrégation des privileges

4. Protection des données personnelles et conformité RGPD

4.1 Analyse d'impact relative a la protection des données

4.2 Chiffrement des données sensibles

4.3 Anonymisation et pseudonymisation

4.4 Droit a l'oubli et portabilité

5. Analyse OWASP Top 10 et contre-mesures

5.1 A01 Broken Access Control

5.2 A02 Cryptographic Failures

5.3 A03 Injection

5.4 A04 Insecure Design

5.5 A05 Security Misconfiguration

5.6 A06 Vulnerable Components

5.7 A07 Authentication Failures

5.8 A08 Software Integrity Failures

5.9 A09 Logging and Monitoring Failures

5.10 A10 Server-Side Request Forgery

6. Système de détection d'intrusion (IDS)

6.1 Monitoring et alertes temps reel

6.2 Détection d'anomalies comportementales

6.3 Rate limiting et protection DDoS

7. Plan de reprise d'activité (DRP)

7.1 Stratégie de sauvegarde

7.2 Procédures de restauration

7.3 Plan de continuité de service

8. Tests de sécurité et audit

8.1 Tests de pénétration

8.2 Audit de sécurité continu

8.3 Veille technologique sécurité

Conclusion

# Introduction

La sécurité d'une application e-commerce comme FarmShop constitue un enjeu critique, particulièrement dans un contexte où les données personnelles et financières des utilisateurs sont manipulées quotidiennement. Cette stratégie de sécurité adopte une approche holistique basée sur les principes du Security by Design.

FarmShop, en tant que plateforme hybride de vente et location de produits agricoles, présente des défis sécuritaires spécifiques : gestion des paiements et cautions, protection des données clients, sécurisation des transactions financières, et protection contre les fraudes liées aux locations d'équipements couteux.

Cette réflexion critique examine systématiquement chaque aspect de la sécurisation de l'application, depuis l'authentification jusqu'au plan de reprise d'activité, en s'appuyant sur les meilleures pratiques industrielles et les recommandations OWASP.

# 1. Security by Design - Principes fondamentaux

## 1.1 Approche Security by Design

**Principes directeurs implémentes :**

**Proactive vs Reactive :**

La sécurité est intégrée dès la conception plutôt qu'ajoutée après coup. Chaque fonctionnalité FarmShop est analysée sous l'angle sécuritaire avant implémentation.

**Security by Default :**

Les paramètres par défaut privilégient toujours l'option la plus sécurisée. Exemple : HTTPS obligatoire, cookies sécurises, headers de sécurité actives.

**Minimisation des privilèges :**

Chaque utilisateur et composant système ne dispose que des permissions strictement nécessaires a ses fonctions.

**Défense en profondeur :**

Multiples couches de sécurité : WAF, middleware Laravel, validation applicative, contrôles base de données.

## 1.2 Implémentation dans FarmShop

**Architecture sécurisée Laravel 11 :**

Utilisation des middlewares de sécurité natifs Laravel

Configuration HTTPS obligatoire avec HSTS

Headers de sécurité automatiques (CSP, X-Frame-Options)

Validation robuste des entrées utilisateurs

Chiffrement transparent des données sensibles

## 1.3 Méthodologie de sécurisation

**Cycle de développement sécurise :**

1. Analyse des menaces par fonctionnalité

2. Design sécurise avec threat modeling

3. Implémentation avec secure coding practices

4. Tests de sécurité automatises

5. Audit de sécurité préproduction

6. Monitoring continu post-déploiement

# 2. Systèmes d'authentification et gestion des identifiants

## 2.1 Architecture d'authentification Laravel Sanctum

**Solution technique retenue :**

Laravel Sanctum pour authentification API et SPA

JWT tokens pour sessions longue duree

Personal Access Tokens pour API externe

**Configuration sécurisée :**

Expiration automatique des tokens (24h par defaut)

Révocation immédiate possible

Limitation du nombre de tokens actifs par utilisateur

Audit trail complet des connexions

## 2.2 Politique de mots de passe robustes

|  |  |  |
| --- | --- | --- |
| **Critère** | **Exigence** | **Implémentation Laravel** |
| Longueur minimale | 12 caractères | Rule::min(12) dans FormRequest |
| Complexité | Mixte requis | Password::min(12)->mixedCase()->numbers()->symbols() |
| Dictionnaire | Interdit | Password::uncompromised() HaveIBeenPwned API |
| Historique | 5 derniers | Table password\_history avec hachage |
| Expiration | 90 jours max | Carbon diff dans middleware |

## 2.3 Authentification multi-facteurs (2FA)

**Implémentation 2FA obligatoire :**

TOTP (Time-based One-Time Password) via Google Authenticator

SMS backup pour utilisateurs sans smartphone

Codes de récupération uniques (10 codes génères)

**Package utilise :**

pragmarx/google2fa-laravel pour intégration TOTP

Configuration : fenêtre de 30 secondes, tolérance 1 fenêtre

## 2.4 Système de récupération des identifiants

**Processus sécurise de récupération :**

1. Vérification email + question de sécurité

2. Génération token cryptographiquement fort (64 caractères)

3. Expiration token court (15 minutes)

4. Lien unique usage avec vérification IP

5. Notification utilisateur sur toute tentative

6. Log audit complet avec géolocalisation

## 2.5 Gestion des sessions sécurisées

**Configuration session Laravel :**

Stockage Redis pour performance et sécurité

Cookies sécurises : HttpOnly, Secure, SameSite=Strict

Régénération ID session a chaque login

Timeout inactivité : 30 minutes standard, 15 minutes admin

Détection sessions concurrentes avec limitation

# 3. Techniques d'autorisation et contrôle d'accès

## 3.1 Modelé RBAC avec Spatie Laravel-Permission

**Architecture des roles :**

|  |  |  |
| --- | --- | --- |
| **Rôle** | **Permissions principales** | **Restrictions spécifiques** |
| Super Admin | Toutes permissions système | IP whitelist, 2FA obligatoire |
| Admin | Gestion produits, commandes, users | Pas de suppression données critiques |
| Modérateur | Modération blog, support client | Read-only données financières |
| Membre Premium | Achat, location, remises spéciales | Limite montant transactions élevée |
| Membre Standard | Achat uniquement, blog read-only | Limite montante, pas de location |
| Visiteur | Consultation catalogue public | Pas d'accès données sensibles |

## 3.2 Gates et Policies Laravel

**Implémentation fine-grained permissions :**

ProductPolicy : controle CRUD produits selon ownership

OrderPolicy : acces commandes limitées au propriétaire

RentalPolicy : vérification éligibilité location

AdminPolicy : ségrégation administrative par modules

## 3.3 Contrôle d'accès aux ressources

**Principe du moindre privilège :**

API endpoints protèges par middleware auth:sanctum

Rate limiting par rôle utilisateur

Validation ownership sur ressources privées

Logs accès détailles avec contexte utilisateur

## 3.4 Ségrégation des privilèges

**Séparation des environnements :**

Comptes dédies par environnement (dev/staging/prod)

Permissions base données minimales par service

Isolation network entre services critiques

Rotation automatique credentials non-humains

# 4. Protection des données personnelles et conformité RGPD

## 4.1 Analyse d'impact relative à la protection des données

**Données sensibles identifiées :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type de donnée** | **Sensibilité** | **Rétention** | **Protection** |
| Identité civile | Elevée | 5 ans | Chiffrement AES-256 |
| Données financières | Critique | 7 ans | PCI-DSS compliance |
| Adresses IP | Moyenne | 1 an | Hachage SHA-256 |
| Comportement site | Faible | 2 ans | Anonymisation |

## 4.2 Chiffrement des données sensibles

**Stratégie de chiffrement :**

En transit : TLS 1.3 obligatoire, certificats EV

Au repos : Laravel Encrypt facade, cles rotatives

Base de données : chiffrement transparent MySQL

Sauvegardes : chiffrement asymétrique GPG

## 4.3 Anonymisation et pseudonymisation

**Techniques implémentées :**

Hachage irréversible des identifiants pour analytics

Suppression automatique données après expiration

Masquages donnés en environnements non-production

## 4.4 Droit a l'oubli et portabilité

**Fonctionnalités RGPD implémentées :**

Export données personnelles format JSON/PDF

Suppression compte avec cascade contrôle

Rectifications données via interface utilisateur

Consentement granulaire cookies et tracking

# 5. Analyse OWASP Top 10 et contre-mesures

## 5.1 A01 Broken Access Control

**Risques identifies :**

Accès non autorise aux commandes d'autres utilisateurs

Elévation de privilèges via manipulation URL

Contournement contrôles cote client

**Contre-mesures implémentées :**

Validation ownership systématique via Policies Laravel

Middleware d'autorisation sur toutes routes sensibles

Tests automatises contrôles d'accès avec PHPUnit

Logs détailles tentatives d'accès non autorises

## 5.2 A02 Cryptographic Failures

**Vulnérabilités potentielles :**

Transmission données sensibles en clair

Algorithmes cryptographiques faibles ou obsolètes

Gestion inadéquate des clés de chiffrement

**Solutions mises en place :**

HTTPS/TLS 1.3 obligatoire avec HSTS active

Chiffrement AES-256 pour donnees au repos

Hachage bcrypt (cost 12) pour mots de passe

Gestion centralisée clés via Laravel Vault

## 5.3 A03 Injection

**Types d'injection prevenus :**

SQL Injection via Eloquent ORM et requetés préparées

XSS via échappement automatique Blade templates

Command Injection via validation stricte entrées

LDAP/NoSQL Injection via sanitisation paramètres

**Mesures preventives :**

Validation robuste avec Laravel Form Requests

Sanitisation entrees via htmlspecialchars

Whitelist caracteres autorises par champ

Content Security Policy restrictive

## 5.4 A04 Insecure Design

**Principes design sécurise :**

Threat modeling systématique par fonctionnalité

Architecture défense en profondeur

Fail-safe defaults dans toute l'application

Ségrégation logique métier/sécurité

## 5.5 A05 Security Misconfiguration

**Configurations sécurisées :**

Headers sécurité automatiques (Helmet middleware)

Désactivation debug mode en production

Suppression endpoints non utilises

Configuration minimale services (principe moindre surface)

## 5.6 A06 Vulnerable and Outdated Components

**Gestion vulnérabilités composants :**

Audit automatise des dépendances avec composer audit

Veille sécurité via GitHub Security Advisories

Mise à jour régulière packages tiers

Tests régression après chaque mise à jour

## 5.7 A07 Identification and Authentication Failures

**Renforcement authentification :**

2FA obligatoire comptes privilèges

Politique mots de passe robuste appliquée

Rate limiting tentatives connexion

Détection comptes compromis (HaveIBeenPwned)

## 5.8 A08 Software and Data Integrity Failures

**Protection integrite :**

Vérification checksums packages Composer

Signature cryptographique deployments

Pipeline CI/CD avec controles integrite

Immutable infrastructure principe

## 5.9 A09 Security Logging and Monitoring Failures

**Stratégie logging sécurité :**

Logs centralises avec ELK Stack

Alertes temps réel événements critiques

Rétention logs 1 an minimum

Correlation events multi-sources

## 5.10 A10 Server-Side Request Forgery (SSRF)

**Prevention SSRF :**

Whitelist URLs externes autorisées

Validation stricte paramètres URL

Isolation network services internes

Monitoring requetés sortantes anormales

# 6. Système de détection d'intrusion (IDS)

## 6.1 Monitoring et alertes temps réel

**Architecture monitoring :**

Laravel Telescope pour debug et monitoring dev

ELK Stack (Elasticsearch, Logstash, Kibana) production

Prometheus + Grafana métriques système

Sentry pour monitoring erreurs applicatives

**Alertes configurées :**

Tentatives connexion échoues répétées (5+)

Accès ressources non autorisées

Pics trafic anormaux (>200% baseline)

Erreurs applicatives critiques

Tentatives injection détectées

## 6.2 Détection d'anomalies comportementales

**Algorithmes de détection :**

Baseline comportemental par utilisateur

Machine Learning pour patterns anormaux

Géolocalisation connexions suspectes

Analyse temporelle activités

## 6.3 Rate limiting et protection DDoS

|  |  |  |  |
| --- | --- | --- | --- |
| **Endpoint** | **Limite/minute** | **Fenêtre** | **Action dépassement** |
| Login | 5 tentatives | 15 min | Blocage IP temporaire |
| API produits | 100 req/min | 1 min | HTTP 429 + retry-after |
| Contact form | 3 soumissions | 10 min | CAPTCHA obligatoire |
| Paiement | 10 tentatives | 1 heure | Blocage compte + alerte |

# 7. Plan de reprise d'activité (DRP)

## 7.1 Stratégie de sauvegarde

**Politique de sauvegarde 3-2-1 :**

3 copies des données critiques

2 supports de stockage différents

1 copie hors site (cloud)

**Fréquences de sauvegarde :**

Base de données : snapshot quotidien + binlog temps réel

Fichiers utilisateurs : sauvegarde incrémentale 4h

Configuration système : sauvegarde hebdomadaire

Code source : Git avec mirrors multiples

**Solutions techniques :**

Laravel Backup package pour automatisation

AWS S3 pour stockage long terme

MySQL replication master-slave

Chiffrement GPG toutes sauvegardes

## 7.2 Procédures de restauration

**RTO (Recovery Time Objective) cibles :**

Services critiques : 2 heures maximum

Base de données : 1 heure maximum

Fonctionnalités secondaires : 8 heures

**RPO (Recovery Point Objective) :**

Données transactionnelles : 15 minutes maximum

Fichiers utilisateurs : 4 heures maximum

Configurations : 24 heures maximum

## 7.3 Plan de continuité de service

**Architecture haute disponibilité :**

Load balancer avec health checks automatiques

Auto-scaling horizontal selon charge

Failover automatique base de données

CDN global pour assets statiques

**Procédures d'escalade :**

Niveau 1 : Alertes automatiques équipe technique

Niveau 2 : Notification management après 30min

Niveau 3 : Communication clients après 1h

Niveau 4 : Activation plan de crise

# 8. Tests de sécurité et audit

## 8.1 Tests de pénétration

**Méthodologie de pentest :**

OWASP Testing Guide v4.2 comme référence

Tests automatises avec OWASP ZAP

Pentest manuel trimestriel par expert externe

Bug bounty programme pour crowdsourced security

**Scenarios de test prioritaires :**

Authentification et autorisation

Injection SQL et XSS

Logique metier (workflow location/achat)

API REST endpoints securite

Configuration infrastructure

## 8.2 Audit de sécurité continu

**Outils d'audit intègres :**

PHPStan pour analyse statique code PHP

SonarQube pour qualité et sécurité code

Snyk pour vulnérabilités dépendances

Laravel Enlightn pour audit configuration

## 8.3 Veille technologique sécurité

**Sources de veille :**

CVE Database pour vulnérabilités nouvelles

OWASP community updates

Laravel Security Advisories

PHP Security Advisories Database

**Processus de mise a jour :**

Evaluation risque dans les 48h

Patch critique applique sous 7 jours

Test régression systématique

Communication équipe et stakeholders

# Conclusion

Cette stratégie de sécurité pour FarmShop établit un cadre robuste et complet, intégrant les meilleures pratiques industrielles et les recommandations OWASP. L'approche Security by Design garantit que la sécurité n'est pas un ajout postérieur mais une composante fondamentale de l'architecture.

Les mesures implémentées couvrent l'ensemble du spectre sécuritaire : de l'authentification multi-facteurs a la détection d'intrusion, en passant par la protection RGPD et le plan de reprise d'activité. Cette stratégie défensive multicouche réduit considérablement la surface d'attaque et assure une résilience maximale.

La conformité OWASP Top 10 et l'intégration d'outils d'audit automatises permettent un maintien continu du niveau de sécurité. Le plan de reprise d'activité assure la continuité de service même en cas d'incident majeur.

Cette stratégie évolue avec les menaces et sera régulièrement mise a jour selon les nouvelles vulnérabilités identifiées et les évolutions technologiques de l'écosystème Laravel.