

MXFlix VAPT Compliance Document

Security Architecture and Compliance Assessment

This document serves as the authoritative source-of-truth for security architecture, vulnerability assessment readiness, and compliance posture for the MXFlix platform. It is intended for technical security reviewers, DevOps engineers, and application security teams responsible for validating and maintaining the platform's defenses.

CLASSIFICATION: CONFIDENTIAL

VERSION: 1.0

STACK: LARAVEL 12 · PHP 8.2+ · MYSQL 8.X · SANCTUM · SPATIE PERMISSION

Table of Contents

This document covers the full spectrum of security controls implemented across the MXFlix application layer. Use the sections below to navigate to specific control domains.

01	Executive Security Summary
	High-level assessment of defense-in-depth posture
02	Security Posture Overview
	Control area status matrix
03	Authentication Architecture
	Six-method auth including OTP, OAuth, WebQR, and TV
04	Authorization and Access Control
	RBAC via Spatie Permission, model policies
05	Input Validation and Sanitization
	FormRequest classes, XSS and SQLi prevention
06	Session, Token and API Key Handling
	Sanctum scoped tokens, SHA-256 hashing
07	CSRF, CORS and Headers Hardening
	Middleware enforcement, cookie security attributes
08	File and Media Upload Security
	MIME validation, signed URL delivery
09	Billing and Payment Security
	Gateway tokenization, webhook signature verification
10	Abuse Prevention and Rate Limiting
	Per-route throttling, OTP brute force protection

01	Audit Logging and Monitoring
	Spatie Activitylog, security event alerting
02	Data Privacy Controls
	GDPR export/erasure, data retention policies
03	OWASP Top 10 Mapping
	Control-to-category mitigation mapping
04	Deployment Hardening
	Production configuration checklist
05	Webhook Security
	Idempotent handlers, signature validation per gateway
06	WebQR and TV Session Security
	One-time tokens, device-bound sessions
07	Parental Controls Security
	Bcrypt PIN, maturity enforcement, profile isolation
08	Compliance Considerations
	PCI DSS, GDPR, COPPA, DMCA alignment
09	VAPT Testing Recommendations
	Scope, tools, and testing frequency guidance

1. Executive Security Summary

MXFlix implements a **defense-in-depth security architecture** across the full application layer. No single control is treated as a silver bullet; rather, security is enforced at every tier — application, data, and transport — such that a failure in one layer does not cascade into a systemic compromise. The platform is built on Laravel 12, a framework with a strong security baseline, augmented by purpose-built middleware, industry-standard encryption libraries, and third-party packages with proven security track records.

The core security capabilities span six functional domains: **multi-method authentication** (Email/Password, OTP, Social OAuth, Sanctum tokens, WebQR, TV OTP), **role-based authorization** enforced at both the route and model levels, **comprehensive input validation** preventing injection and XSS, **encrypted storage and transport** using AES-256-CBC and TLS 1.2+, **structured activity logging** for forensic and compliance purposes, and **payment security** that ensures no raw cardholder data ever touches MXFlix infrastructure.

This document is intended as a living source-of-truth for security reviewers, penetration testers, and engineering teams responsible for maintaining and extending the platform. Each section maps implementation details to specific security controls, making this document directly usable for VAPT scoping, compliance audits, and internal security reviews.



This document reflects Version 1.0 of the MXFlix security architecture. All controls described herein are available in the current codebase unless explicitly noted as requiring configuration.

2. Security Posture Overview

The following matrix summarizes the implementation status of all major security control areas across the MXFlix platform. Controls marked **Available** are implemented and active. Controls marked **Config Required** require environment-specific configuration before they are fully effective in production.

Control Area	Implementation	Status
Authentication	Multi-method (Email, OTP, Social, Sanctum)	✓ Available
Authorization	RBAC via Spatie Laravel Permission	✓ Available
Encryption at Rest	AES-256-CBC (config/app.php cipher)	✓ Available
Password Hashing	Bcrypt cost=12 (config/hashing.php)	✓ Available
CSRF Protection	VerifyCsrfToken middleware on all web routes	✓ Available
CORS	Configurable allowed origins (config/cors.php)	⚙️ Config Required
Rate Limiting	Laravel throttle middleware on API routes	✓ Available
Activity Logging	Spatie Activitylog with old/new values	✓ Available
Mass Assignment Protection	Eloquent fillable/guarded on all models	✓ Available
Soft Deletes	Enabled on all critical entities	✓ Available
Security Misconfiguration	CORS hardening, env protection, debug off	⚙️ Config Required
Vulnerable Components	Composer audit, dependency management	✓ Operational

3. Authentication Architecture

MXFlix supports six distinct authentication methods, each purpose-built for a specific client type or user journey. This multi-method approach ensures that every entry point — web browser, mobile app, smart TV, and third-party integration — has a secure, context-appropriate authentication flow backed by standardized token management via Laravel Sanctum.



Email / Password

Laravel built-in authentication with Bcrypt hashing at cost factor 12. Passwords are never logged, returned in API responses, or stored in plaintext. Per-user salts prevent rainbow table attacks. Minimum 8-character enforcement at the FormRequest layer.



OTP Authentication

Cryptographically random 6-digit codes valid for 10 minutes, stored as hashes in `user_otps`. Rate-limited to 3 generation requests per phone per hour and 5 verification attempts per OTP. SMS delivery via integrated gateway (Razorpay/PhonePe).



Social OAuth

Laravel Socialite integration supporting Google, Apple, Facebook, and GitHub. OAuth 2.0 with PKCE for web flows. Social accounts linked via `social_accounts` table. Email verification is mandatory before social accounts can access subscription features.



WebQR & TV Session

QR codes encode a 64-character hex token stored in `webqr_sessions` with 5-minute expiry and single-use enforcement. TV OTP flow sends code to email, polls a status endpoint, then issues a Sanctum token. All flows record device fingerprint for audit.

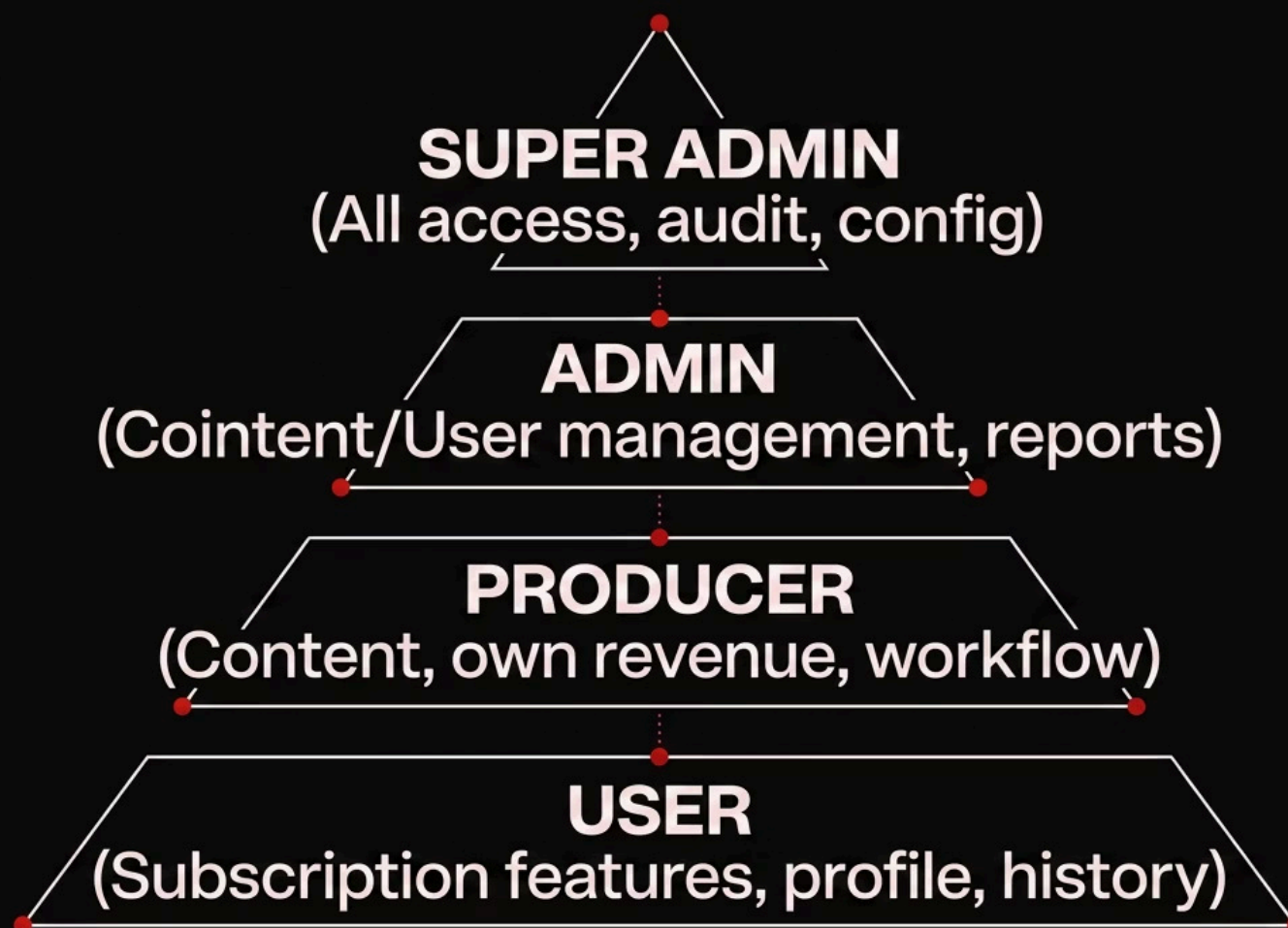


Sanctum personal access tokens are SHA-256 hashed before storage. Token abilities (scopes) are validated on every request, and tokens can be revoked individually or globally via "logout everywhere."

4. Authorization and Access Control

Role-Based Access Control (RBAC) is implemented using Spatie Laravel Permission, assigning roles at the user level. The four defined roles create a clear privilege hierarchy: **Super Admin** has full system access including configuration and audit logs; **Admin** manages content and users; **Producer** can submit content and access revenue data; and **User** accesses subscriber features only. Permissions are grouped into `permission_sets` for efficient bundle assignments.

Model-level authorization is enforced via Laravel policies — `ContentPolicy`, `UserPolicy`, and `SubscriptionPolicy` — which check both role membership and resource ownership at every gate. This prevents horizontal privilege escalation where a user of equal role might access another user's data. Multi-profile authorization ensures that watchlist, continue-watch, and watch history data is strictly scoped to `profile_id`.



Route-level enforcement is handled by three custom middleware classes: `CheckAdmin`, `CheckUser`, and `CheckProducer`. These middleware guards ensure that role checks occur before any controller logic is executed, providing a reliable first line of authorization defense regardless of policy configuration.

5. Input Validation and Sanitization

All user input is validated using Laravel `FormRequest` classes before reaching controllers. Validation rules are defined per-field with explicit constraints: string length limits, numeric ranges, email format verification, URL format checks, and UUID validation for all foreign key inputs. Array inputs such as genre lists and cast members are validated for both type and count to prevent parameter pollution attacks.

File uploads undergo MIME type validation using the File facade and PHP's `finfo` extension — critically, this checks the actual file content, not merely the extension. Image dimension constraints and file size limits are enforced at the validation layer before any storage operation is initiated.

XSS Prevention

Laravel Blade auto-escapes all output by default using `{{ }}` syntax. Raw output via `{!! !!}` requires explicit opt-in and must never be used with user-supplied content. User-generated content (reviews, bios) is stored and rendered exclusively through Blade escaping. JavaScript-based rendering of user content requires whitelist-based sanitization.

SQL Injection Prevention

Eloquent ORM uses parameterized queries exclusively throughout the codebase. Raw queries use `DB::select()` with bound parameters — never string interpolation. Query builder methods (`where`, `join`, `orderBy`) are preferred over raw expressions in all cases. No user input is ever concatenated directly into SQL.

Sanctum Token Input

All Sanctum token inputs are validated through `FormRequest` before processing. Token abilities submitted by clients are validated against a whitelist of permitted scopes to prevent privilege escalation via crafted token requests.

6. Session, Token and API Key Handling

Laravel Sanctum provides the core API token management layer with fine-grained scoping. Tokens are hashed using SHA-256 before storage in the `personal_access_tokens` table, meaning a database breach does not expose usable tokens. Token abilities are validated on every authenticated request, and scopes such as `read:content` and `write:subscription` restrict tokens to their intended routes.

Tokens can be revoked individually (single device logout) or globally (logout from all devices), providing users and administrators with full session control. Session lifetime is configurable: the default is 2 weeks for web sessions and 1 year for mobile clients, reflecting the differing security postures of those environments. Inactive users have all tokens automatically invalidated upon account status change.

Session Driver

Database-backed session storage, configurable to Redis for high-availability deployments. Session lifetime: 120 minutes (configurable).

Token Storage

SHA-256 hashed before persistence. Only the hash is stored; the plaintext token is shown to the user once at creation and never again.

Device Tracking

`devices` table records `session_id` and `last_activity` for each authenticated device, enabling per-device revocation and forensic review.

Token Expiration

Configurable per token type. Expired tokens are rejected at the middleware layer. Automatic cleanup via scheduled jobs prevents accumulation of stale records.

7. CSRF, CORS and Headers Hardening

CSRF protection is enforced via Laravel's `VerifyCsrfToken` middleware on all state-changing web routes. All HTML forms include the CSRF token via the `csrf_field()` helper, and AJAX requests must include the `X-CSRF-TOKEN` header. API routes authenticated via Sanctum Bearer tokens are appropriately exempt from CSRF verification, as the token itself serves as the anti-CSRF mechanism.

CORS headers are configured per-environment in `config/cors.php`, specifying allowed origins, HTTP methods, and permitted headers. Production deployments must restrict `allowed_origins` to verified MXFlix domains only — wildcard origins are explicitly prohibited in production configuration. The `TrustProxies` middleware ensures correct IP attribution and HTTPS enforcement for deployments behind load balancers or CDN edge nodes.

Cookie Security Attributes

Secure: Cookies transmitted over HTTPS only. **HttpOnly:** Cookies inaccessible to JavaScript, mitigating XSS-based session theft. **SameSite=Strict:** Cookies not sent on cross-site requests, providing CSRF defense-in-depth even if the CSRF token is somehow bypassed.

Recommended Security Headers

Production deployments should enforce: `Strict-Transport-Security (HSTS)` with a minimum 1-year max-age, `X-Frame-Options: DENY` to prevent clickjacking, `X-Content-Type-Options: nosniff`, and a restrictive `Content-Security-Policy` header. These are configured at the Nginx/server level, not application level.

TrustProxies Configuration

The `TrustProxies` middleware is configured to trust load balancer IP ranges only. This prevents IP spoofing via forged `X-Forwarded-For` headers, ensuring accurate client IP attribution in rate limiting and audit logs.

8. File and Media Upload Security

File uploads are restricted by MIME type validation using PHP's `finfo` extension to inspect actual file content rather than relying on client-supplied extensions. Permitted types include: `mp4`, `mov`, `avi`, `mkv` (video), `jpg`, `png`, `webp` (images), and `srt`, `vtt`, `json` (subtitles/metadata). Maximum file sizes are enforced at the `FormRequest` layer: 50 GB for video, 10 MB for images, and 1 MB for subtitle files.

All uploaded files are stored in **private S3 or BunnyCDN buckets** that are not directly accessible via URL. Stream URLs are generated on-demand with signed expiry tokens: 15 minutes for video streams and 1 hour for image assets. Video delivery is protected by `VIDEO_SECRET_KEY` and `VIDEO_MAX_AGE` (86,400 seconds) configured in `config/video.php`. This ensures that even if a signed URL is extracted from a client session, it has a bounded window of validity and cannot be redistributed.

Upload Validation Summary

File Type	Allowed Formats	Max Size
Video	mp4, mov, avi, mkv	50 GB
Images	jpg, png, webp	10 MB
Subtitles	srt, vtt	1 MB
Metadata	json	1 MB

Storage Security

- **Private Bucket Policy**
S3 and BunnyCDN buckets configured with `deny-public-access` policies. No direct URL access permitted.
- **Signed URL Generation**
All media access requires a time-limited signed URL generated server-side, bound to the authenticated session.
- **CDN Token Auth**
BunnyCDN enforces token authentication on all edge delivery requests, preventing hotlinking or URL sharing.

9. Billing and Payment Security

MXFlix achieves **PCI DSS compliance through gateway tokenization** — raw card data never touches MXFlix servers at any point in the payment flow. All payment processing is delegated to trusted third-party gateways (Stripe, Razorpay, PayPal, and others), which return payment reference IDs that are stored in the transactions table instead of any cardholder data. API keys for all gateways are stored exclusively in environment variables and never committed to version control.

Every incoming payment webhook is verified using provider-specific signature validation before any business logic is executed. Stripe webhooks use `WebhookSignature::constructEvent` with a timing-safe comparison against the webhook secret. Razorpay validates the `X-Razorpay-Signature` header using HMAC SHA-256. PayPal verifies using the PayPal-IDN and webhook ID mechanism. Refund operations require admin role authorization and generate a complete audit trail in the transactions log.

No Card Storage

Zero raw cardholder data stored. Only gateway-issued payment reference IDs persist in the database.



Webhook Verification

HMAC signature validation per gateway. Timing-safe comparison prevents timing oracle attacks on signature checks.

Full Audit Trail

All transactions, refunds, and subscription changes are logged with user, timestamp, IP, and gateway reference.



Secret Management

Gateway API keys stored in environment variables only. Never in code, database, or logs. Rotation capability is available.

10. Abuse Prevention and Rate Limiting

Laravel's built-in throttle middleware is applied at the route level with differentiated limits by endpoint sensitivity and authentication state. The rate limiting configuration is designed to block automated abuse while maintaining a comfortable experience for legitimate users. All rate limit breach events are logged to enable pattern analysis and adversary identification.

60

Requests/min
Authenticated API users

30

Requests/min
Guest users on auth routes

300

Requests/min
Content browsing endpoints

5

Attempts/min
Login attempts per IP address

5

Max Attempts
OTP verification per issued code

3

OTPs/hour
Per phone number generation limit

WebQR session generation is limited to 10 requests per minute per user to prevent token flooding attacks. Failed login attempts are tracked and contribute toward IP-based blocking. OTP brute force is mitigated by both the per-OTP attempt limit (5 tries before invalidation) and the per-phone generation limit (3 per hour), making automated guessing of 6-digit codes computationally infeasible within the validity window.

11. Audit Logging and Monitoring

Spatie Activitylog captures all significant user and system events, including model create, update, and delete operations with full old/new attribute value snapshots. This enables forensic reconstruction of any data state at any point in time. Content acquisition logs independently track the full approval pipeline state transitions, providing an unbroken chain of custody for content publication decisions.

The `activity_logs` table stores: `user_id`, `event_type`, `event_description`, `ip_address`, `user_agent`, and `created_at` timestamp for every logged event. Session tracking records IP address, user agent, and device fingerprint, enabling correlation of suspicious activity across sessions. Critical security events — including failed login attempts and privilege escalation attempts — trigger immediate alerts via the platform's notification system.

Events Captured

- Authentication attempts (success and failure)
- Content CRUD operations with before/after state
- Subscription status changes
- Payment transactions and refund operations
- Admin privileged actions
- API token creation and revocation
- Webhook delivery outcomes
- Rate limit breach events

Log Record Schema

Field	Description
<code>user_id</code>	Authenticated user who performed the action
<code>event_type</code>	Categorized event classification
<code>event_description</code>	Human-readable action detail
<code>ip_address</code>	Originating IP (post TrustProxies resolution)
<code>user_agent</code>	Client user agent string
<code>created_at</code>	UTC timestamp of event

12. Data Privacy Controls

MXFlix is architected to support GDPR compliance obligations. Users can export all their personal data — including watch history, profile information, and subscription records — via the `DataExportJob`, which generates a structured JSON file delivered to the user's verified email address. Account deletion triggers the `SoftDeletesTrait`, followed by a background job that anonymizes or permanently removes personal data within 30 days, satisfying GDPR Article 17 (Right to Erasure) requirements.

Personal data is scoped strictly to the user entity, with cascading delete capability ensuring that profile-level data, watchlist entries, and viewing history are all removed as part of the account deletion lifecycle. Parental consent tracking is implemented for COPPA compliance for under-13 users. Data retention policies balance compliance obligations with operational needs: watch history is retained for 2 years, while transaction logs are retained for 7 years to meet financial regulatory requirements.



GDPR Data Export

Structured JSON export of all personal data via `DataExportJob`. Delivered to verified email. Covers watch history, profiles, and subscriptions.



Right to Erasure

Account deletion anonymizes personal data within 30 days per GDPR Article 17. Soft deletes preserve referential integrity during the grace period.



Encryption at Rest

AES-256-CBC encryption for sensitive data fields. TLS 1.2+ enforced for all data in transit. Secrets never stored in code or database.



Retention Policies

Watch history: 2-year retention. Transaction logs: 7-year retention (financial regulations). Parental consent records retained per COPPA requirements.

13.OWASP Top 10 Mapping

The following table maps each OWASP Top 10 (2021) vulnerability category to the specific MXFlix control or configuration that mitigates it. Categories marked **Mitigated** have active controls in the current codebase. Categories marked **Config Required** require environment-specific configuration in production deployments to be fully effective.

OWASP ID	Category	MXFlix Mitigation	Status
A01	Broken Access Control	RBAC middleware, model policies, Sanctum token scopes, profile isolation	Mitigated
A02	Cryptographic Failures	AES-256-CBC at rest, Bcrypt passwords, HTTPS enforced, signed URLs only	Mitigated
A03	Injection	Eloquent ORM parameterized queries, Blade auto-escaping, FormRequest validation	Mitigated
A04	Insecure Design	Approval workflows, role separation, threat modeling, rate limiting as design control	Mitigated
A05	Security Misconfiguration	CORS hardening, env protection, APP_DEBUG=false, hardened PHP config	Config Required
A06	Vulnerable Components	Composer audit, dependency management, quarterly review cadence	Operational
A07	Auth Failures	Multi-method auth, rate limiting, token expiry, OTP brute force protection	Mitigated
A08	Software and Data Integrity	CSRF protection, webhook signature verification, Composer lock file	Mitigated
A09	Logging Failures	Activitylog on all models, session tracking, acquisition logs, security event alerts	Mitigated
A10	SSRF	Input validation on URL fields, URL allowlist filtering, no server-side URL fetching of user input	Mitigated

14. Deployment Hardening

Production hardening requires explicit configuration of both application-level and server-level settings. The following checklist represents the minimum required configuration for a secure production deployment. DevOps teams must verify each item before any production release or infrastructure change.

Application-Level Hardening

-  **APP_DEBUG=false** in production
Debug mode must never be enabled in production — it exposes stack traces, environment variables, and internal configuration to any user who triggers an error.
-  **APP_ENV=production**
Ensures Laravel loads production-optimized configuration and disables development-only features such as query logging and detailed error pages.
-  **Private S3 / BunnyCDN Buckets**
All media buckets must enforce private access with signed-URL-only delivery. Public access must be blocked at the bucket policy level, not just ACL level.
-  **Redis Password Configured**
Redis instances used for cache and session storage must require authentication. Default open Redis configurations are a common lateral movement vector.

Server-Level Hardening

-  **UFW Firewall Rules**
Allow only ports 80 (HTTP redirect), 443 (HTTPS), and a non-default SSH port. All other inbound traffic must be denied by default. Database ports must not be exposed to the public internet.
-  **Nginx with TLS 1.3**
Minimum TLS version set to 1.2; TLS 1.3 preferred. Cipher suites restricted to ECDHE-based key exchange. HTTP security headers (HSTS, CSP, X-Frame-Options) configured at the Nginx level.
-  **Unattended Upgrades**
Ubuntu **unattended-upgrades** package enabled for automatic security patch application. PHP and system package CVEs are addressed without manual intervention delay.
-  **Queue Worker Monitoring**
Laravel Horizon or pm2 ensures queue workers restart automatically on failure. Critical security-sensitive jobs (data deletion, webhook processing) must never silently stall.

15-19. Additional Security Domains

The following sections summarize four additional security control areas that apply to specialized MXFlix platform features, followed by the complete VAPT testing recommendations for security assessment teams.

Webhook Security All incoming webhooks from payment gateways are verified using provider-specific HMAC signature validation before processing. Handlers are idempotent — duplicate event IDs are detected via <code>webhook_logs</code> and silently discarded to prevent replay-based double-processing. Failed deliveries are retried up to 5 times with exponential backoff. All webhook events are immutably logged for audit purposes.	WebQR & TV Session Security WebQR tokens are 64-character cryptographically random hex values, SHA-256 hashed in <code>webqr_sessions</code>, and expire after 5 minutes. Tokens are single-use — invalidated immediately upon consumption or expiry, preventing replay attacks. TV OTP flows issue Sanctum tokens post-validation. All cross-device authentication events record device fingerprint for audit trail continuity.
Parental Controls Security Parental PINs are Bcrypt-hashed and stored in <code>users.parental_pin</code>. The <code>is_kids</code> flag per profile enforces content filtering against the profile's configured maturity level at the playback authorization layer. Profile switching requires PIN re-entry after 30 minutes of inactivity. Age ratings on content are compared server-side on every playback request — client-side filtering is never trusted.	Compliance Considerations PCI DSS: No card data stored; all processing via gateway tokenization. GDPR: Data export, right to erasure, consent management, 72-hour breach notification readiness. COPPA: Parental consent for under-13 users, kids profile content enforcement. DMCA: Take-down procedure with repeat infringer policy. WCAG 2.1 AA: Accessibility compliance target for web interface.

VAPT Testing Recommendations

Security testing should be performed **quarterly** and after any significant feature release. The recommended testing scope covers the full attack surface of the MXFlix platform.

Authentication Auth bypass, OTP brute force, token enumeration, session fixation	Authorization Horizontal and vertical privilege escalation, profile boundary bypass, RBAC policy gaps	Injection SQL injection across all input vectors, XSS in review/bio fields, command injection via file upload	Transport CSRF on state-changing ops, webhook replay attacks, rate limit circumvention, SSRF via stream URL fields
Exposure Sensitive data in API responses, debug mode leakage, S3 bucket enumeration, error message disclosure		Dependencies Composer audit against CVE database, PHP version vulnerabilities, outdated Spatie/Sanctum packages	

 Recommended tools: OWASP ZAP for automated scanning, Burp Suite Professional for manual penetration testing, and composer audit integrated into CI/CD pipelines for continuous dependency vulnerability monitoring.

Defense-in-Depth: Closing Statement

MXFlix operates as a defense-in-depth platform where security controls are layered such that a failure in any single control does not compromise the entire system.

The application layer uses **Laravel 12 built-in protections** augmented by custom security middleware for authentication, authorization, rate limiting, and input validation. The data layer enforces constraints at the database level through foreign keys, unique indexes, and check constraints. The transport layer enforces **TLS 1.2+ for all communications**. Secrets are never stored in code or database — environment variables are the single source of truth for all sensitive configuration.

This document represents Version 1.0 of the MXFlix VAPT Compliance documentation. It is intended to be maintained as a living document, updated with each significant security control change, new feature introduction, or completed penetration test cycle. Security reviewers and engineering teams are encouraged to reference this document when scoping assessments, planning feature work, or responding to compliance inquiries.

Application Layer

Laravel 12 framework protections, custom middleware, FormRequest validation, Blade escaping, Sanctum token auth

Data Layer

Eloquent ORM parameterized queries, AES-256-CBC encryption, Bcrypt hashing, soft deletes, foreign key constraints

Transport Layer

TLS 1.2+ enforced via TrustProxies, HSTS headers, signed media URLs, HTTPS-only cookies

Operations Layer

Spatie Activitylog, rate limiting, webhook signature verification, quarterly VAPT, dependency auditing

MXFlix VAPT Compliance Document — Classification: Confidential — Security, Compliance, Engineering — Version 1.0