PatchMon Environment Variables Reference

This document provides a comprehensive reference for all environment variables available in PatchMon. These variables can be configured in your `backend/.env` file (bare metal installations) or in the `.env` file alongside `docker-compose.yml` (Docker deployments using `env_file`).

Database Configuration

PostgreSQL database connection settings.

Variable	Description	Default	Required	Example
`DATABASE_URL`	PostgreSQL connection string	-	Yes	`postgresql://user:pass@localhost:5432/patchmon_db`
`PM_DB_CONN_MAX_ATTEMPTS`	Maximum database connection attempts during startup	`30`	No	`30`
`PM_DB_CONN_WAIT_INTERVAL`	Wait interval between connection attempts (seconds)	`2`	No	`2`

Usage Notes

- The `DATABASE_URL` must be a valid PostgreSQL connection string
- Connection retry logic helps handle database startup delays in containerized environments
- Format: `postgresql://[user]:[password]@[host]:[port]/[database]`
- In Docker deployments, `DATABASE_URL` is constructed automatically in the compose file - you do not set it in `.env`

Database Connection Pool (Prisma)

Connection pooling configuration for optimal database performance and resource management.

Variable	Description	Default	Required	Example
`DB_CONNECTION_LIMIT`	Maximum number of database connections per instance	`30`	No	`30`
`DB_POOL_TIMEOUT`	Seconds to wait for an available connection before timeout	`20`	No	`20`
`DB_CONNECT_TIMEOUT`	Seconds to wait for initial database connection	`10`	No	`10`
`DB_IDLE_TIMEOUT`	Seconds before closing idle connections	`300`	No	`300`
`DB_MAX_LIFETIME`	Maximum lifetime of a connection in seconds	`1800`	No	`1800`

Sizing Guidelines

Small Deployment (1-10 hosts):

```
DB_CONNECTION_LIMIT=15
DB_POOL_TIMEOUT=20
```

Medium Deployment (10-50 hosts):

```
DB_CONNECTION_LIMIT=30  # Default
DB_POOL_TIMEOUT=20
```

Large Deployment (50+ hosts):

```
DB_CONNECTION_LIMIT=50
DB_POOL_TIMEOUT=30
```

Connection Pool Calculation

Use this formula to estimate your needs:

```
DB_CONNECTION_LIMIT = (expected_hosts * 2) + (concurrent_users * 2) + 5
```

Example: 20 hosts + 3 concurrent users:

```
(20 * 2) + (3 * 2) + 5 = 51 connections
```

Important Notes

- Each backend instance maintains its own connection pool
- Running multiple backend instances requires considering total connections to PostgreSQL
- PostgreSQL default `max_connections` is 100 (ensure your pool size doesn't exceed this)
- Connections are reused efficiently - you don't need one connection per host
- Increase pool size if experiencing timeout errors during high load

Detecting Connection Pool Issues

When connection pool limits are hit, you'll see clear error messages in your backend console:

Typical Pool Timeout Error:

```
Host creation error: Error: Timed out fetching a new connection from the connection pool.
DATABASE CONNECTION POOL EXHAUSTED!
Current limit: DB_CONNECTION_LIMIT=30
Pool timeout: DB_POOL_TIMEOUT=20s
Suggestion: Increase DB_CONNECTION_LIMIT in your .env file
```

If you see these errors frequently, increase `DB_CONNECTION_LIMIT` by 10-20 and monitor your system.

Monitoring Connection Pool Usage

You can monitor your PostgreSQL connections to determine optimal pool size:

Check Current Connections:

```
# Connect to PostgreSQL
psql -U patchmon_user -d patchmon_db

# Run this query
SELECT count(*) as current_connections,
       (SELECT setting::int FROM pg_settings WHERE name='max_connections') as max_connections
FROM pg_stat_activity
WHERE datname = 'patchmon_db';
```

Recommended Actions:

- If `current_connections` frequently approaches `DB_CONNECTION_LIMIT`, increase the pool size
- Monitor during peak usage (when multiple users are active, agents checking in)
- Leave 20-30% headroom for burst traffic

Database Transaction Timeouts

Control how long database transactions can run before being terminated.

Variable	Description	Default	Required	Example
`DB_TRANSACTION_MAX_WAIT`	Maximum time (ms) to wait for a transaction to start	`10000`	No	`10000`
`DB_TRANSACTION_TIMEOUT`	Maximum time (ms) for a standard transaction to complete	`30000`	No	`30000`
`DB_TRANSACTION_LONG_TIMEOUT`	Maximum time (ms) for long-running transactions (e.g. bulk operations)	`60000`	No	`60000`

Usage Notes

- These prevent runaway queries from holding database locks indefinitely
- Increase `DB_TRANSACTION_LONG_TIMEOUT` if bulk import or migration operations are timing out
- All values are in milliseconds

Authentication & Security

JWT token configuration and security settings.

Variable	Description	Default	Required	Example
`JWT_SECRET`	Secret key for signing JWT tokens	-	Yes	`your-secure-random-secret-key`
`JWT_EXPIRES_IN`	Access token expiration time	`1h`	No	`1h`, `30m`, `2h`
`JWT_REFRESH_EXPIRES_IN`	Refresh token expiration time	`7d`	No	`7d`, `3d`, `14d`

Generating Secure Secrets

```
# Linux/macOS
openssl rand -hex 64
```

Time Format

Supports the following formats:

- `s`: seconds
- `m`: minutes
- `h`: hours
- `d`: days

Examples: `30s`, `15m`, `2h`, `7d`

Recommended Settings

Development:

```
JWT_EXPIRES_IN=1h
JWT_REFRESH_EXPIRES_IN=7d
```

Production:

```
JWT_EXPIRES_IN=30m
JWT_REFRESH_EXPIRES_IN=3d
```

High Security:

```
JWT_EXPIRES_IN=15m
JWT_REFRESH_EXPIRES_IN=1d
```

Password Policy

Enforce password complexity requirements for local user accounts.

Variable	Description	Default	Required	Example

`PASSWORD_MIN_LENGTH`	Minimum password length	`8`	No	`8`, `12`, `16`
`PASSWORD_REQUIRE_UPPERCASE`	Require at least one uppercase letter	`true`	No	`true`, `false`
`PASSWORD_REQUIRE_LOWERCASE`	Require at least one lowercase letter	`true`	No	`true`, `false`
`PASSWORD_REQUIRE_NUMBER`	Require at least one number	`true`	No	`true`, `false`
`PASSWORD_REQUIRE_SPECIAL`	Require at least one special character	`true`	No	`true`, `false`
`PASSWORD_RATE_LIMIT_WINDOW_MS`	Rate limit window for password changes (ms)	`900000`	No	`900000` (15 min)
`PASSWORD_RATE_LIMIT_MAX`	Maximum password change attempts per window	`5`	No	`5`

Recommended Settings

Standard (default):

```
PASSWORD_MIN_LENGTH=8
PASSWORD_REQUIRE_UPPERCASE=true
PASSWORD_REQUIRE_LOWERCASE=true
PASSWORD_REQUIRE_NUMBER=true
PASSWORD_REQUIRE_SPECIAL=true
```

High Security:

```
PASSWORD_MIN_LENGTH=12
PASSWORD_REQUIRE_UPPERCASE=true
PASSWORD_REQUIRE_LOWERCASE=true
PASSWORD_REQUIRE_NUMBER=true
PASSWORD_REQUIRE_SPECIAL=true
PASSWORD_RATE_LIMIT_MAX=3
```

Usage Notes

- These rules apply to local accounts only - OIDC users authenticate against their identity provider
- Password changes and new account creation both enforce these rules

- `PASSWORD_RATE_LIMIT_*` prevents brute-force password change attempts

Account Lockout

Protect against brute-force login attacks by temporarily locking accounts after repeated failures.

Variable	Description	Default	Required	Example
`MAX_LOGIN_ATTEMPTS`	Failed login attempts before account lockout	`5`	No	`5`, `3`, `10`
`LOCKOUT_DURATION_MINUTES`	Minutes the account stays locked after exceeding attempts	`15`	No	`15`, `30`, `60`

Usage Notes

- Lockout is per-account, not per-IP
- The failed attempts counter has a 15-minute rolling window - if no further failed attempts occur within that window, the counter resets on its own
- A successful login clears the failed attempts counter (before lockout is triggered)
- Once locked out, the account stays locked for the full `LOCKOUT_DURATION_MINUTES` - there is no way to bypass this except waiting
- Setting `MAX_LOGIN_ATTEMPTS` too low may lock out legitimate users who mistype passwords

Recommended Settings

Standard:

```
MAX_LOGIN_ATTEMPTS=5
LOCKOUT_DURATION_MINUTES=15
```

High Security:

```
MAX_LOGIN_ATTEMPTS=3
LOCKOUT_DURATION_MINUTES=30
```

Session Management

Control user session behavior and security.

Variable	Description	Default	Required	Example
`SESSION_INACTIVITY_TIMEOUT_MINUTES`	Minutes of inactivity before automatic logout	`30`	No	`30`

Usage Notes

- Sessions are tracked in the database with activity timestamps
- Each authenticated request updates the session activity
- Expired sessions are automatically invalidated
- Users must log in again after timeout period
- Lower values provide better security but may impact user experience

Recommended Settings

- **High Security Environment**: `15` minutes
- **Standard Security**: `30` minutes (default)
- **User-Friendly**: `60` minutes

Two-Factor Authentication (TFA)

Settings for two-factor authentication when users have it enabled.

Variable	Description	Default	Required	Example
`MAX_TFA_ATTEMPTS`	Failed TFA code attempts before lockout	`5`	No	`5`, `3`
`TFA_LOCKOUT_DURATION_MINUTES`	Minutes locked out after exceeding TFA attempts	`30`	No	`30`, `60`
`TFA_REMEMBER_ME_EXPIRES_IN`	"Remember this device" token expiration	`30d`	No	`30d`, `7d`, `90d`
`TFA_MAX_REMEMBER_SESSIONS`	Maximum remembered devices per user	`5`	No	`5`
`TFA_SUSPICIOUS_ACTIVITY_THRESHOLD`	Failed attempts before flagging suspicious activity	`3`	No	`3`

Usage Notes

- These variables only apply when users have TFA enabled on their account
- "Remember this device" allows users to skip TFA on trusted devices
- `MAX_TFA_ATTEMPTS` and `TFA_LOCKOUT_DURATION_MINUTES` prevent brute-force attacks on TOTP codes
- Suspicious activity detection can trigger additional security measures
- Remembered sessions can be revoked by users or admins

Recommended Settings

Standard:

```
MAX_TFA_ATTEMPTS=5
TFA_LOCKOUT_DURATION_MINUTES=30
TFA_REMEMBER_ME_EXPIRES_IN=30d
TFA_MAX_REMEMBER_SESSIONS=5
TFA_SUSPICIOUS_ACTIVITY_THRESHOLD=3
```

High Security:

```
MAX_TFA_ATTEMPTS=3
TFA_LOCKOUT_DURATION_MINUTES=60
TFA_REMEMBER_ME_EXPIRES_IN=7d
TFA_MAX_REMEMBER_SESSIONS=3
TFA_SUSPICIOUS_ACTIVITY_THRESHOLD=2
```

OIDC / SSO

OpenID Connect configuration for Single Sign-On. Set `OIDC_ENABLED=true` and fill in your identity provider details to enable SSO.

Variable	Description	Default	Required	Example
`OIDC_ENABLED`	Enable OIDC authentication	`false`	No	`true`, `false`
`OIDC_ISSUER_URL`	Identity provider issuer URL	-	If OIDC enabled	`https://auth.example.com`
`OIDC_CLIENT_ID`	OAuth client ID	-	If OIDC enabled	`patchmon`

Variable	Description	Default	Required	Example
`OIDC_CLIENT_SECRET`	OAuth client secret	-	If OIDC enabled	`your-client-secret`
`OIDC_REDIRECT_URI`	Callback URL after authentication	-	If OIDC enabled	`https://patchmon.example.com/api/v1/auth/oidc/callback`
`OIDC_SCOPES`	OAuth scopes to request	`openid email profile groups`	No	`openid email profile groups`
`OIDC_AUTO_CREATE_USERS`	Automatically create PatchMon accounts for new OIDC users	`true`	No	`true`, `false`
`OIDC_DEFAULT_ROLE`	Default role for auto-created OIDC users	`user`	No	`user`, `admin`, `viewer`
`OIDC_DISABLE_LOCAL_AUTH`	Disable local username/password login when OIDC is enabled	`false`	No	`true`, `false`
`OIDC_BUTTON_TEXT`	Login button text shown on the login page	`Login with SSO`	No	`Login with SSO`, `Sign in with Authentik`

Group-to-Role Mapping

Map OIDC groups from your identity provider to PatchMon roles. This keeps role assignments in sync with your IdP.

Variable	Description	Default	Required	Example
`OIDC_ADMIN_GROUP`	OIDC group name that maps to admin role	-	No	`PatchMon Admins`
`OIDC_USER_GROUP`	OIDC group name that maps to user role	-	No	`PatchMon Users`
`OIDC_SYNC_ROLES`	Sync roles from OIDC groups on each login	`true`	No	`true`, `false`

Example: Authentik

```
OIDC_ENABLED=true
OIDC_ISSUER_URL=https://authentik.example.com/application/o/patchmon/
OIDC_CLIENT_ID=patchmon
OIDC_CLIENT_SECRET=your-client-secret-here
OIDC_REDIRECT_URI=https://patchmon.example.com/api/v1/auth/oidc/callback
```

```
OIDC_SCOPES=openid email profile groups
OIDC_AUTO_CREATE_USERS=true
OIDC_DEFAULT_ROLE=user
OIDC_BUTTON_TEXT=Login with Authentik
OIDC_ADMIN_GROUP=PatchMon Admins
OIDC_USER_GROUP=PatchMon Users
OIDC_SYNC_ROLES=true
```

Example: Keycloak

```
OIDC_ENABLED=true
OIDC_ISSUER_URL=https://keycloak.example.com/realms/your-realm
OIDC_CLIENT_ID=patchmon
OIDC_CLIENT_SECRET=your-client-secret-here
OIDC_REDIRECT_URI=https://patchmon.example.com/api/v1/auth/oidc/callback
OIDC_SCOPES=openid email profile groups
OIDC_AUTO_CREATE_USERS=true
OIDC_DEFAULT_ROLE=user
OIDC_BUTTON_TEXT=Login with Keycloak
```

Usage Notes

- `OIDC_REDIRECT_URI` must be registered as an allowed redirect URI in your identity provider
- When `OIDC_DISABLE_LOCAL_AUTH=true`, users can only log in via OIDC - useful for enforcing SSO across the organisation
- When `OIDC_SYNC_ROLES=true`, the user's role is updated on every login based on their OIDC group membership
- If a user is in both `OIDC_ADMIN_GROUP` and `OIDC_USER_GROUP`, the admin role takes precedence
- The `groups` scope must be supported by your identity provider and included in `OIDC_SCOPES` for group mapping to work

---

Server & Network Configuration

Server protocol, host, and CORS settings.

Variable	Description	Default	Required	Example

`PORT`	Backend API server port	`3001`	No	`3001`
`NODE_ENV`	Node.js environment mode	`production`	No	`production`, `development`
`SERVER_PROTOCOL`	Server protocol	`http`	No	`http`, `https`
`SERVER_HOST`	Server hostname/domain	`localhost`	No	`patchmon.example.com`
`SERVER_PORT`	Server port	`3000`	No	`3000`, `443`
`CORS_ORIGIN`	Allowed CORS origin URL	`http://localhost:3000`	No	`https://patchmon.example.com`
`CORS_ORIGINS`	Multiple allowed CORS origins (comma-separated)	-	No	`https://a.example.com,https://b.example.com`
`ENABLE_HSTS`	Enable HTTP Strict Transport Security	`true`	No	`true`, `false`
`TRUST_PROXY`	Trust proxy headers (when behind reverse proxy)	`true`	No	`true`, `false`

Usage Notes

- `SERVER_PROTOCOL`, `SERVER_HOST`, and `SERVER_PORT` are used to generate agent installation scripts
- `CORS_ORIGIN` must match the URL you use to access PatchMon in your browser
- `CORS_ORIGINS` (plural, comma-separated) overrides `CORS_ORIGIN` when set - only needed if PatchMon is accessed from multiple domains
- Set `TRUST_PROXY` to `true` when behind nginx, Apache, or other reverse proxies
- `ENABLE_HSTS` should be `true` in production with HTTPS

Example Configurations

Local Development:

```
SERVER_PROTOCOL=http
SERVER_HOST=localhost
SERVER_PORT=3000
CORS_ORIGIN=http://localhost:3000
ENABLE_HSTS=false
TRUST_PROXY=false
```

Production with HTTPS:

```
SERVER_PROTOCOL=https
SERVER_HOST=patchmon.example.com
SERVER_PORT=443
CORS_ORIGIN=https://patchmon.example.com
ENABLE_HSTS=true
TRUST_PROXY=true
```

Multiple Domains:

```
SERVER_PROTOCOL=https
SERVER_HOST=patchmon.example.com
SERVER_PORT=443
CORS_ORIGINS=https://patchmon.example.com,https://patchmon-alt.example.com
ENABLE_HSTS=true
TRUST_PROXY=true
```

Rate Limiting

Protect your API from abuse with configurable rate limits.

Variable	Description	Default	Required	Example
`RATE_LIMIT_WINDOW_MS`	General rate limit window (milliseconds)	`900000`	No	`900000` (15 min)
`RATE_LIMIT_MAX`	Maximum requests per window (general)	`5000`	No	`5000`
`AUTH_RATE_LIMIT_WINDOW_MS`	Authentication endpoints rate limit window (ms)	`600000`	No	`600000` (10 min)
`AUTH_RATE_LIMIT_MAX`	Maximum auth requests per window	`500`	No	`500`
`AGENT_RATE_LIMIT_WINDOW_MS`	Agent API rate limit window (ms)	`60000`	No	`60000` (1 min)
`AGENT_RATE_LIMIT_MAX`	Maximum agent requests per window	`1000`	No	`1000`

Understanding Rate Limits

Rate limits are applied per IP address and endpoint category:

- **General API**: Dashboard, hosts, packages, user management
- **Authentication**: Login, logout, token refresh
- **Agent API**: Agent check-ins, updates, package reports

Calculating Windows

The window is a sliding time frame. Examples:

- `900000` ms = 15 minutes
- `600000` ms = 10 minutes
- `60000` ms = 1 minute

Recommended Settings

Default (Balanced):

```
RATE_LIMIT_WINDOW_MS=900000      # 15 minutes
RATE_LIMIT_MAX=5000              # ~5.5 requests/second
AUTH_RATE_LIMIT_WINDOW_MS=600000 # 10 minutes
AUTH_RATE_LIMIT_MAX=500          # ~0.8 requests/second
AGENT_RATE_LIMIT_WINDOW_MS=60000 # 1 minute
AGENT_RATE_LIMIT_MAX=1000        # ~16 requests/second
```

Strict (High Security):

```
RATE_LIMIT_WINDOW_MS=900000      # 15 minutes
RATE_LIMIT_MAX=2000              # ~2.2 requests/second
AUTH_RATE_LIMIT_WINDOW_MS=600000 # 10 minutes
AUTH_RATE_LIMIT_MAX=100          # ~0.16 requests/second
AGENT_RATE_LIMIT_WINDOW_MS=60000 # 1 minute
AGENT_RATE_LIMIT_MAX=500         # ~8 requests/second
```

Relaxed (Development/Testing):

```
RATE_LIMIT_WINDOW_MS=900000      # 15 minutes
RATE_LIMIT_MAX=10000             # ~11 requests/second
AUTH_RATE_LIMIT_WINDOW_MS=600000 # 10 minutes
AUTH_RATE_LIMIT_MAX=1000         # ~1.6 requests/second
AGENT_RATE_LIMIT_WINDOW_MS=60000 # 1 minute
AGENT_RATE_LIMIT_MAX=2000        # ~33 requests/second
```

Redis Configuration

Redis is used for BullMQ job queues and caching.

Variable	Description	Default	Required	Example
`REDIS_HOST`	Redis server hostname	`localhost`	No	`localhost`, `redis`, `10.0.0.5`
`REDIS_PORT`	Redis server port	`6379`	No	`6379`
`REDIS_USER`	Redis username (Redis 6+)	-	No	`default`
`REDIS_PASSWORD`	Redis authentication password	-	Recommended	`your-redis-password`
`REDIS_DB`	Redis database number	`0`	No	`0`, `1`, `2`

Usage Notes

- Redis authentication is highly recommended for security
- Redis 6.0+ supports ACL with usernames; earlier versions use password-only auth
- If no password is set, Redis will be accessible without authentication (not recommended)
- Database number allows multiple applications to use the same Redis instance

Docker Deployment

In Docker, set `REDIS_PASSWORD` in your `.env` file. The compose file automatically passes it to both the Redis container (via its startup command) and the backend service (via `env_file`).

Bare Metal Deployment

The setup script configures Redis ACL with a dedicated user and password per instance. The credentials are written to `backend/.env` automatically.

Generating Secure Passwords

```
openssl rand -hex 32
```

Logging

Control application logging behavior and verbosity.

Variable	Description	Default	Required	Example
`LOG_LEVEL`	Logging level	`info`	No	`debug`, `info`, `warn`, `error`
`ENABLE_LOGGING`	Enable/disable application logging	`true`	No	`true`, `false`
`PM_LOG_TO_CONSOLE`	Output logs to the console	`false`	No	`true`, `false`
`PM_LOG_REQUESTS_IN_DEV`	Log HTTP requests in development mode	`false`	No	`true`, `false`
`PRISMA_LOG_QUERIES`	Log all Prisma database queries	`false`	No	`true`, `false`

Log Levels

Ordered from most to least verbose:

1. **`debug`**: All logs including database queries, internal operations
2. **`info`**: General information, startup messages, normal operations
3. **`warn`**: Warning messages, deprecated features, non-critical issues
4. **`error`**: Error messages only, critical issues

Recommended Settings

Development:

```
LOG_LEVEL=debug
ENABLE_LOGGING=true
PM_LOG_TO_CONSOLE=true
PM_LOG_REQUESTS_IN_DEV=true
PRISMA_LOG_QUERIES=true
```

Production:

```
LOG_LEVEL=info
ENABLE_LOGGING=true
PM_LOG_TO_CONSOLE=false
PRISMA_LOG_QUERIES=false
```

Production (Quiet):

```
LOG_LEVEL=warn
ENABLE_LOGGING=true
PRISMA_LOG_QUERIES=false
```

Timezone Configuration

Control timezone handling for timestamps and logs across the application.

Variable	Description	Default	Required	Example
`TZ`	Timezone for timestamps and logs	`UTC`	No	`UTC`, `America/New_York`, `Europe/London`

Usage Notes

- The `TZ` environment variable controls timezone handling across all components:
  - **Backend (Node.js)**: Timestamps in API responses, database records, logs
  - **Agent (Go)**: Agent logs, integration data timestamps
- If `TZ` is not set, the application defaults to UTC
- Database timestamps are always stored in UTC for consistency
- Display timestamps can be converted to the configured timezone

Common Timezone Values

```
# UTC (recommended for servers)
TZ=UTC

# UK
TZ=Europe/London

# US
TZ=America/New_York       # Eastern
TZ=America/Chicago         # Central
TZ=America/Los_Angeles    # Pacific

# Europe
TZ=Europe/Paris
TZ=Europe/Berlin
```

```
# Asia
TZ=Asia/Tokyo
TZ=Asia/Shanghai
```

Body Size Limits

Control the maximum size of request bodies accepted by the API.

Variable	Description	Default	Required	Example
`JSON_BODY_LIMIT`	Maximum JSON request body size	`5mb`	No	`5mb`, `10mb`, `1mb`
`AGENT_UPDATE_BODY_LIMIT`	Maximum body size for agent update payloads	`2mb`	No	`2mb`, `5mb`

Usage Notes

- `JSON_BODY_LIMIT` applies to all standard API endpoints (dashboard actions, user management, etc.)
- `AGENT_UPDATE_BODY_LIMIT` applies specifically to agent check-in and package report payloads
- Increase these if agents are managing a very large number of packages and the payload exceeds the limit
- Keep these as low as practical to limit memory usage and reduce the impact of oversized requests

---

Encryption

Controls encryption of sensitive data stored in the database (e.g. AI provider API keys, bootstrap tokens).

Variable	Description	Default	Required	Example
`AI_ENCRYPTION_KEY`	Encryption key for sensitive data at rest (64 hex characters)	-	No	Output of `openssl rand -hex 32`

Variable	Description	Default	Required	Example
`SESSION_SECRET`	Fallback key used if `AI_ENCRYPTION_KEY` is not set	-	No	Output of `openssl rand -hex 32`

How It Works

The backend uses this priority chain to determine the encryption key:

1. **`AI_ENCRYPTION_KEY`** - used directly if set (64 hex chars = 32 bytes, or any string which gets SHA-256 hashed)
2. **`SESSION_SECRET`** - if `AI_ENCRYPTION_KEY` is not set, SHA-256 hashed to derive the key
3. **`DATABASE_URL`** - if neither above is set, derives a key from the database URL (logs a security warning)
4. **Ephemeral** - last resort, generates a random key (data encrypted with this key will be unreadable after a restart)

What Gets Encrypted

- **AI API keys** - API keys for AI providers (e.g. OpenAI) are AES-256-GCM encrypted before being stored in the database
- **Bootstrap tokens** - Agent auto-enrollment API keys are encrypted before temporary storage in Redis

Usage Notes

- For most deployments, you do not need to set either variable - the key is derived from `DATABASE_URL` which is stable
- Set `AI_ENCRYPTION_KEY` if you need encryption stability across database URL changes or multi-replica deployments
- These do not affect user password storage (passwords are bcrypt hashed, not encrypted)

User Management

Default settings for new users.

Variable	Description	Default	Required	Example
`DEFAULT_USER_ROLE`	Default role assigned to new users	`user`	No	`user`, `admin`, `viewer`

Available Roles

- `admin`: Full system access, can manage users and settings
- `user`: Standard access, can manage hosts and packages
- `viewer`: Read-only access, cannot make changes

Usage Notes

- Only applies to newly created users
- Existing users are not affected by changes to this variable
- First user created through setup is always an admin
- Can be changed per-user through the user management interface

---

Frontend Configuration

Frontend-specific environment variables (used during build and runtime).

Variable	Description	Default	Required	Example
`VITE_API_URL`	Backend API base URL	`/api/v1`	No	`http://localhost:3001/api/v1`
`VITE_APP_NAME`	Application name displayed in UI	`PatchMon`	No	`PatchMon`
`VITE_APP_VERSION`	Application version displayed in UI	(from package.json)	No	`1.4.0`
`BACKEND_HOST`	Backend hostname (Docker only)	`backend`	No	`backend`, `localhost`
`BACKEND_PORT`	Backend port (Docker only)	`3001`	No	`3001`
`VITE_ENABLE_LOGGING`	Enable frontend debug logging	`false`	No	`true`, `false`

Usage Notes

- Frontend variables are prefixed with `VITE_` for the Vite build system
- `VITE_*` variables are embedded at build time - they cannot be changed at runtime
- `VITE_API_URL` can be relative (`/api/v1`) or absolute
- `BACKEND_HOST` and `BACKEND_PORT` are used by the Docker frontend container's Nginx proxy config

---

Complete Example Configuration

Bare Metal (Production)

```
# Database
DATABASE_URL="postgresql://patchmon_user:secure_db_password@localhost:5432/patchmon_db"
PM_DB_CONN_MAX_ATTEMPTS=30
PM_DB_CONN_WAIT_INTERVAL=2

# Database Connection Pool
DB_CONNECTION_LIMIT=30
DB_POOL_TIMEOUT=20
DB_CONNECT_TIMEOUT=10
DB_IDLE_TIMEOUT=300
DB_MAX_LIFETIME=1800

# JWT
JWT_SECRET="generated-secure-secret-from-openssl"
JWT_EXPIRES_IN=30m
JWT_REFRESH_EXPIRES_IN=3d

# Server
PORT=3001
NODE_ENV=production
SERVER_PROTOCOL=https
SERVER_HOST=patchmon.example.com
SERVER_PORT=443
CORS_ORIGIN=https://patchmon.example.com
ENABLE_HSTS=true
TRUST_PROXY=true

# Session
SESSION_INACTIVITY_TIMEOUT_MINUTES=30

# User
DEFAULT_USER_ROLE=user

# Rate Limiting
```

```
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX=5000
AUTH_RATE_LIMIT_WINDOW_MS=600000
AUTH_RATE_LIMIT_MAX=500
AGENT_RATE_LIMIT_WINDOW_MS=60000
AGENT_RATE_LIMIT_MAX=1000

# Redis
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_PASSWORD=secure_redis_password
REDIS_DB=0

# Logging
LOG_LEVEL=info
ENABLE_LOGGING=true

# Timezone
TZ=UTC

# TFA
TFA_REMEMBER_ME_EXPIRES_IN=30d
TFA_MAX_REMEMBER_SESSIONS=5
TFA_SUSPICIOUS_ACTIVITY_THRESHOLD=3
```

Docker (Production)

For Docker deployments, see `docker/env.example` for a complete template. Copy it to `.env`, fill in the required values, and run `docker compose up -d`. The compose file reads all variables via `env_file: .env`.

Troubleshooting

Common Issues

"timeout of 10000ms exceeded" when adding hosts:

- Increase `DB_CONNECTION_LIMIT` (try 30 or higher)

- Increase `DB_POOL_TIMEOUT` (try 20 or 30)
- Check backend logs for "DATABASE CONNECTION POOL EXHAUSTED" messages

Database connection failures on startup:

- Increase `PM_DB_CONN_MAX_ATTEMPTS`
- Increase `PM_DB_CONN_WAIT_INTERVAL`
- Verify `DATABASE_URL` is correct

"Invalid or expired session" errors:

- Check `JWT_SECRET` hasn't changed between restarts
- Verify `SESSION_INACTIVITY_TIMEOUT_MINUTES` isn't too low
- Ensure `JWT_EXPIRES_IN` is reasonable

Rate limit errors (429 Too Many Requests):

- Increase `RATE_LIMIT_MAX` values
- Increase window duration (`*_WINDOW_MS` variables)

CORS errors:

- Verify `CORS_ORIGIN` matches your frontend URL exactly (protocol + domain + port)
- For multiple domains, use `CORS_ORIGINS` (plural, comma-separated)

OIDC login fails:

- Verify `OIDC_REDIRECT_URI` is registered in your identity provider
- Check `OIDC_ISSUER_URL` is reachable from the PatchMon server
- Ensure `OIDC_CLIENT_SECRET` matches the value in your IdP

Encrypted data unreadable after restart:

- Set `AI_ENCRYPTION_KEY` to a stable value so the key persists across restarts
- Re-enter AI provider API keys if the encryption key has changed

---

Security Best Practices

1. **Always set strong secrets:**
   - Use `openssl rand -hex 64` for `JWT_SECRET`
   - Use `openssl rand -hex 32` for database and Redis passwords
2. **Enable HTTPS in production:**
   - Set `SERVER_PROTOCOL=https`
   - Enable `ENABLE_HSTS=true`

    - Use proper SSL certificates
3. **Configure appropriate rate limits:**
    - Adjust based on expected traffic
    - Lower limits for public-facing deployments
4. **Use session timeouts:**
    - Don't set `SESSION_INACTIVITY_TIMEOUT_MINUTES` too high
    - Balance security with user experience
5. **Secure Redis:**
    - Always set `REDIS_PASSWORD`
    - Use Redis ACLs in Redis 6+ for additional security
    - Don't expose Redis port publicly
6. **Enable account lockout:**
    - Keep `MAX_LOGIN_ATTEMPTS` and `LOCKOUT_DURATION_MINUTES` at defaults or stricter
7. **Enforce password policy:**
    - Keep all `PASSWORD_REQUIRE_*` options enabled
    - Consider increasing `PASSWORD_MIN_LENGTH` to 12+

Version Information

- **Last Updated:** February 2026
- **Applicable to PatchMon:** v1.4.0+

Revision #7
Created 2025-10-24 21:19:50 UTC by Iby
Updated 2026-02-12 17:53:01 UTC by Iby