CFM

Cybersecurity & Firewall Manager Documentation

Overview

Project purpose, structure, key components

Overview

CFM Project Overview

Purpose
CFM is a Security - Firewall Management project designed to manage spam detection, blocklists, feed processing, and admin control using a web interface.

Layers
- Models: Handle data and relationships
- Controllers: Process web/API requests
- Commands: Automate backend tasks
- Filament: Admin UI for managing keywords, feeds, and settings
- Routes: Web and API access
- Cron Jobs: Scheduled feed updates and maintenance

Key Features
- Spam keyword detection (loose & strict modes)
- Blocklist/whitelist management (IP and domain-based)
- Feed processing system (with logging and rule generation)
- Central API for querying blocklists
- Admin UI for managing everything
- Clamav Rule generation
- Phishlist database
- RBLDNS database
- Hash Management (MD5 / SHA1 / SHA2) for Clamav signatures
- Mail filters
- Synchronizing files to server through agents
- Updating lists through agents
- Manage global user unblocking

This documentation is organized by feature and component to help understand and extend the system.

Overview

Introduction - Welcome to CFM

□□ CFM Feature Summary

CFM is a powerful system for managing threat intelligence, spam filtering, phishing protection, and reputation data — backed by automation and agent-based sync.

□□ Security & Threat Intelligence

- **Blocklist & Whitelist Management** (IP & domain)
- **Reverse DNS**, **ASN**, **GeoIP**, and **country resolution**
- **Keyword-based spam detection**
- **Phishing URL detection & logging**

□□ Automation & Scheduling

- Scheduled **feed imports**, **auto-deletion**, and **rule generation**
- Commands for IP list generation, rule updates, config sync
- Cron-style job scheduling with overlap protection

□□ Agent Infrastructure (C++-Based)

- Syncs config and rule files
- Reports **blocks**, **unblocks**, and **last seen**
- Triggers service restarts after updates
- Integrates with unblock portal for auto-removal
- Sends **Slack alerts** for offline agents

□□ Antivirus & RBL Integration

- Generates **ClamAV signatures** from phishing URLs and file hashes (MD5/SHA1/SHA256)
- Maintains **SpamAssassin-compatible phishing DB**
- Exports **RBL and URIBL zones** for RBLDNSD

□□ API & External Access

- Token-authenticated API for:
  - Checking block status
  - Reporting blocks/unblocks
  - Fetching rules/feeds
  - Submitting config/trigger reports
- Optional **rate limiting** and **IP filtering**

□□ Admin Panel (Filament)

- Dashboard with real-time widgets and charts
- Interfaces for:
  - Spam keywords
  - Block/allow lists
  - Feed logs
  - Unblock requests
  - Agent activity

□□ Web Interface

- Public-facing **Unblock Request Form**
- Feed endpoints (IP, domain, phishing, etc.)
- Admin redirect and login flow

□□ Bonus Features

- File-based config sync with integrity hashing
- Config-targeting for agent groups
- Slack alerts and activity logs

- Multi-source feed support (manual, API, auto)

□□ Key Features

□ Blocklist & Whitelist Management
Manage IPs and domains across multiple lists, including manual entries, feed imports, and API-reported threats.

□ Spam & Phishing Protection

- Keyword-based spam filtering (supports Greek/Greeklish, loose/strict)
- Maintains a live phishing URL database
- Generates **ClamAV-compatible** virus definitions from phishing URLs and file hashes (MD5/SHA1/SHA256)

□ RBL & URIBL Generator
Creates real-time blocklists and URI lists (RBLDNSD format) for DNS-based blacklisting — used by SpamAssassin, Postfix, etc.

□ GeoIP Intelligence for Blocklist Entries
Automatically resolves:

- Reverse DNS (PTR)
- ASN and ISP
- Country and region This enables rich filtering, analytics, and decision-making.

□ Automated Feed Processing
Processes threat feeds on a schedule with logs and rule generation.

□ Agent Communication & API
Lightweight agents (or servers) can:

- Report blocked IPs back to CFM
- Fetch updates and policy
- Submit files, triggers, logs, etc.

□ Dashboard with Widgets & Metrics
Summarized view of:

- Top IPs by country or source
- Phishing trends
- Recent feed activity
- System health and jobs

Unblock Request Portal
Public-facing form for users to request delisting — reviewed via admin panel.

Full Admin UI via Filament
Modern interface for managing:

- Spam keywords
- Feeds & logs
- Phishing database
- Block/allow lists
- Scheduled jobs
- Settings & tokens

Scheduled Jobs & Artisan Tools

- Generate IP and domain blocklists
- Run cleanup jobs
- Sync filesystem configs
- Rebuild ClamAV signatures
- Trigger per-feed processing

Agent Infrastructure (C++ Powered)
Includes high-performance **C++ agents** deployed on remote servers that:

- Sync configuration and rule files from CFM
- Report blocked and unblocked IPs
- Remove blocks upon updates or unblocks
- ♻ Restart services (e.g., mail, firewall) when needed
- Report "last seen" heartbeat to monitor health
- Trigger Slack alerts if an agent goes offline
- Integrate with the public unblock form to re-allow mistakenly blocked users

Blocklist & Whitelist Management
Manage IPs and domains across multiple lists (manual, API, or feed-driven), enriched with PTR, ASN, country, and GeoIP.

Phishing & Spam Defense

- Greek-aware spam keyword detection (strict/loose)
- Maintains a phishing URL database
- Generates **ClamAV virus signatures** from URLs and hashes (MD5/SHA1/SHA256)
- Exports phishing data for **SpamAssassin compatibility**

RBL & URIBL Generation
Creates and serves real-time DNS blacklists (RBLDNSD format) for both IP and domain-based blocklists.

Scheduled Feed Ingestion & Rule Generation
Automates external feed syncing and keyword/rule building via Laravel Scheduler and Artisan commands.

Admin Dashboard
Modern UI with dashboard widgets, charts, and management panels for:

- Blocked items
- Keyword rules
- Feed logs
- Unblock requests
- Agent status

Unblock Request Portal
Frontend form where blocked users can request removal — triggers backend unblock workflows and agent sync.

API Interface
Secure, token-authenticated API to:

- Check IP/domain status
- Report blocks/unblocks
- Pull feed or rule updates
- Trigger diagnostics or config checks

ClamAV + CSF Integration
Outputs live files for:

- IP blocklists (`csf.deny`)
- ClamAV custom signatures
- RBLDNSD-based DNS lists

Bonus Features

- Slack integration for agent down alerts
- Per-country analytics of blocked IPs
- Top reporters / sources breakdown
- File-based config sync and hashing
- Agent group targeting for rules

Use Cases

- Internal **spam firewall**
- Self-hosted **RBL/URIBL provider**
- **CSF / UFW / iptables** blocklist hub
- Aggregator for multiple **threat feeds**
- **Email security gateway** enhancement
- Coordinated threat response via **reporting agents**

□□ Built With

- Laravel + Filament (UI)
- MySQL (DB)
- Tailwind (optional UI)
- GeoLite2 (GeoIP)
- Artisan + Laravel Scheduler
- RBLDNSD & SpamAssassin compatibility
- API-first design

Models

Model summaries, fields, relationships, usage

Models Overview

This project includes the following Eloquent models, each representing a key part of the system's architecture:

Model	Purpose / Description
`Agent`	Represents remote scanning or reporting agents.
`AgentGroup`	Groups agents for easier configuration and targeting.
`AutoDeleteRule`	Defines automatic deletion logic for old data.
`Blocklist`	Stores IPs flagged as malicious or suspicious.
`Config`	Represents configuration entries, either file-based or database-based.
`ConfigTarget`	Associates configs with agents or systems.
`DataFeed`	Represents external data sources (e.g., threat intel feeds).
`ExecutionTarget`	Tracks where specific actions or rules were executed.
`Hash`	Handles hash-based uniqueness or matching logic.
`MailFeeder`	Ingests and analyzes email headers or body content.
`MailFromFilter`	Applies filtering rules based on mail sender.
`Notifier`	Manages alerts and notifications.
`PhishList`	Stores phishing domains or URLs.
`RbldnsdUri`	Manages RBLDNSD-compatible URI blocklists/whitelists.
`SpamKeyword`	Contains keyword rules for spam detection (Greek/Greeklish/etc.).
`Token`	Access tokens for API or agent auth.
`Trigger`	Defines rule triggers and conditions.
`UnblockRequest`	Handles user-submitted unblock appeals.
`User`	Represents authenticated users in the system.

API Routes

Route list, methods, paths, expected input/output

API routes

API Routes

These routes provide a secure, token-authenticated interface for interacting with blocklists, agent configurations, and unblocking systems.

All routes are protected via the `TokenAuthentication` middleware.

□□ Authentication

All API endpoints require a valid token passed via headers or parameters.

□□ Blocklist Endpoints

Method	Path	Description
POST	`/blocklist/report`	Report an IP to be blocked
POST	`/blocklist/unblock`	Request removal of an IP from blocklist
GET	`/blocklist/check`	Check if an IP is blocked
GET	`/blocklist/fetch`	Fetch all blocklisted IPs

□□ Whitelist Endpoints

Method	Path	Description
POST	`/whitelist/report`	Report a whitelisted IP or domain
POST	`/whitelist/remove`	Remove a record from the whitelist

□□ Agent Endpoints

Method	Path	Description
GET/POST	`/agent/config-check`	Validate agent config from server side
GET/POST	`/agent/list-files`	List tracked files for integrity checks
GET	`/blocklist/pending-unblocks`	Fetch unblock requests for review
POST	`/blocklist/unblock-confirm`	Confirm that an IP was unblocked

These routes form the backbone of external system interaction with the CFM platform, especially useful for:

- **Server agents** checking their config
- **Security automation** scripts reporting IPs
- **Unblock portals** submitting requests for delisting

□□ Web Routes

These routes handle the core frontend and admin-facing interactions, including redirects, unblock forms, and feed outputs.

□□ Root Redirect

Method	Path	Behavior
GET	`/`	Redirects to `/admin` if logged in, otherwise to `/admin/login`

□□ Public Unblock Interface

Method	Path	Description
GET	`/unblock`	Shows unblock request form
POST	`/unblock`	Submits unblock request to backend

Used by users or systems mistakenly blocked to appeal removal.

□□ Feed Files (Token Protected)

Accessible only with valid token via `TokenAuthentication`.

Method	Path	Description
GET	`/whitelist.txt`	IP/domain whitelist
GET	`/blacklist.txt`	IP blacklist (for CSF, etc.)
GET	`/phishlist.txt`	Phishing domain list
GET	`/domainblacklist.txt`	Domain blocklist for RBLDNSD
GET	`/domainwhitelist.txt`	Domain whitelist for RBLDNSD

Artisan Commands and Scheduled Tasks (Crons)

Command list, arguments/options, usage

Artisan Commands

Command Line Commands (Artisan)

CFM includes a number of custom Laravel Artisan commands to help automate and manage blocklists, phishing detection, feed ingestion, and other system behaviors.

To view available commands, use the following in your project root:

```
php artisan
```

Below is an overview of the key command groups available in CFM:

---

agents

Command	Description
`agents:check-notifications`	Checks the last-seen status of each agent and sends notifications (e.g., Slack) if any are down.

---

autodelete

Command	Description
`autodelete:run`	Runs `AutoDeleteJob` to remove old records (e.g., blocklist entries, logs) based on custom rules.

blocklist

Command	Description
`blocklist:fetch-geodata`	Bulk fetches GeoIP data for blocklist entries (country, ASN, etc.).
`blocklist:resolve-geodata`	Resolves GeoIP data with optional processing limits for batching.
`blocklist:resolve-ptr`	Resolves PTR (Reverse DNS) records for IPs in batches. Parallel processing supported.

cache

Command	Description
`cache:clear`	Clears the Laravel application cache.

clamav

Command	Description
`clamav:generate-signatures`	Generates ClamAV-compatible signature files from phishing URLs and malware file hashes (MD5, SHA1, SHA256).

⚙ config

> *This section may include config sync-related commands, depending on future additions.*

data-feeds

Command	Description
`data-feeds:fetch`	Fetches and processes all active data feeds configured in the system.
`import:blocklist`	Imports a list of IPs into the blocklist, whitelist, or greylist.
`import:domains`	Imports domains into the blocklist or whitelist.
`import:mail-filters`	Imports email sender/domain filters from a file.

iplists

Command	Description
`iplists:generate`	Generates `whitelist.txt`, `greylist.txt`, and `blacklist.txt` files from the database for CSF or DNS use.

mailfromfilters

Command	Description
`mailfromfilters:generate`	Generates `mailfromfilters.cf` used by the mail filtering system, based on DB entries.

spamassassin

Command	Description
`spamassassin:generate-rules`	Generates a SpamAssassin custom keyword rule file from the `spam_keywords` table. Supports strict and loose matching, Greek normalization, etc.

These commands allow you to maintain and automate your security infrastructure directly from the console, and can be scheduled or run on-demand as needed.

Scheduled Tasks / Cron and Supervisor settings

Scheduled Commands (Cron)

Scheduled tasks are defined in `routes/console.php` and executed using Laravel's `schedule:run` command. They can be run via a cron shell script or with a process manager like **Supervisor**.

Shell Cron Example:

```
#!/bin/bash
cd "$(dirname "$0")"
php artisan schedule:run >> /dev/null 2>&1
```

Supervisor Setup (Recommended)

Service file: `supervisor.service`

```
[Unit]
Description=Supervisor process control system for UNIX
After=network.target

[Service]
ExecStart=/usr/bin/supervisord -n -c /etc/supervisor/supervisord.conf
ExecStop=/usr/bin/supervisorctl $OPTIONS shutdown
ExecReload=/usr/bin/supervisorctl -c /etc/supervisor/supervisord.conf $OPTIONS reload
Restart=on-failure
RestartSec=50s
```

Program config: `/etc/supervisor/conf.d/cfm.conf`

```
[program:cfm]
command=php /home/cfm/artisan queue:work redis --memory=2048 --tries=3 --timeout=600
user=cfm
autostart=true
autorestart=true
numprocs=2
redirect_stderr=true
stdout_logfile=/var/log/supervisor/cfm.log
```

Scheduled Commands

Command	Schedule	Description
`iplists:generate`	Every 10 minutes	Generate updated IP lists from the DB
`clamav:generate-signatures`	Hourly at :50	Generate ClamAV signature files
`agents:check-notifications`	Every minute	Check agent status and send alerts
`autodelete:run`	Daily at 08:00	Run AutoDelete cleanup job
`config:sync-storage`	Every 5 minutes	Sync configuration files from disk
`blocklist:resolve-ptr`	Every 10 minutes	Resolve PTR records in batches
`blocklist:resolve-geodata`	Every 10 minutes	Resolve ASN/Country data in bulk
`unblocks:cleanup`	Every 5 minutes	Cleanup old unblock requests
`phishlist:generate`	Every 30 minutes	Regenerate phishing domain list
`mailfromfilters:generate`	Hourly	Regenerate mailfromfilters.cf
`execution:dispatch`	Every minute	Dispatch queued execution rules
`DataFeedJob (callback)`	Every minute	Dispatch data feed jobs for active feeds

These scheduled commands are critical for keeping data up-to-date, automating cleanup, syncing feeds and configurations, and dispatching jobs to agents or queues.

DataFeedProcessor

How the core logic handles feeds

Admin Interface

Pages, forms, tables, actions

Configuration

.env setup, configs, scheduler, RBL settings

Integration / Agent Guide

How external systems use the API or CLI

Testing

Example API tests, test scenarios

FAQ / Common Scenarios

Troubleshooting, usage examples

Notification handling

- How the system handles and works around with notifications.

Notifications & Triggers

□□ CFM Notifications System

Overview

The **CFM Notifications System** monitors the availability of network agents and provides customizable alerting when agents go offline or recover. It also integrates with **user-triggered unblock request monitoring**, helping administrators stay informed about potential abuse or false positives in blocking behavior.

The system is composed of four coordinated components:

1. **Agent Uptime Monitoring Controller**
2. **Notifier Configuration Resource**
3. **Trigger Mechanism Resource**
4. **Unblock Request Monitoring**

These components work together to detect agent status changes, define notification methods, and execute appropriate alerting workflows.

---

1. □□ Agent Uptime Monitoring

Component: `AgentNotificationController`

This controller is responsible for continuously checking the status of registered agents. It does so by comparing each agent's last heartbeat timestamp (`last_seen_at`) against the current time. If an agent hasn't communicated within a predefined time window (e.g., 60 seconds), it is considered **offline**.

Key Behaviors

- **Recovery Detection**: If an agent previously marked as offline starts reporting again, the system logs a restoration event.
- **Outage Detection**: Agents not seen in the defined window are marked as down.
- **Notification Throttling**: Notifications are suppressed if a recent alert was already sent (e.g., within the past hour), avoiding redundant notifications.

---

2. □□ Notifier Configuration

Component: `NotifierResource`

This resource allows administrators to configure various **notification channels**, which define **how alerts are sent**.

Supported Notifier Types (examples):

- Email
- Slack
- Webhook
- Custom API callouts

Each notifier contains:

- A unique name.
- A target configuration (e.g., email address or webhook URL).
- A type identifier (used to trigger the appropriate method).
- Optionally, a list of tags or filters to determine relevance to specific agents.

---

3. □□ Trigger Mechanism

Component: `TriggerResource`

This module acts as the **brain of the notification system**, determining **when and which notifications** should be triggered based on agent status changes.

Core Features

- **Linkage** between triggers and notifiers (one-to-many).

- **Conditions**: Triggers can be configured to react only to specific types of status changes (e.g., only on downtime, or both up/down events).
- **Tag Matching**: Allows targeting subsets of agents based on metadata.
- **Last Notified Tracking**: Stores timestamps of previous alerts per trigger-agent pair to control re-alerting frequency.

---

4. □□ Unblock Request Monitoring

Integration Point: `UnblockController` (external module)

This integration captures and processes **unblock requests submitted by end users** who are temporarily blocked by the system (e.g., via CSF or custom firewall logic).

Features

- **Alerting on Unblock Attempts**: Notifies admins when a user requests to be unblocked.
- **Request Outcome Visibility**: Indicates if the user:
  - Was indeed blocked and unblocked.
  - Was never blocked.
  - Has requested unblocking too frequently (possible abuse).
- **Escalation Signals**: Excessive or suspicious unblock activity can flag issues for deeper inspection.

This feature provides security teams with real-time context on potentially malicious or misbehaving clients.

□□ Security Considerations

To protect system integrity and minimize attack surface:

- Internal logic is abstracted and modularized to reduce exposure.
- Logging is selectively enabled and avoids sensitive data leakage.
- Time-based checks and cooldown intervals prevent alert spam or abuse.
- All external interactions (e.g., sending to Slack or webhooks) should be validated and rate-limited.
- All external outbound notifications should be validated and rate-limited.
- Unblock request alerts help detect misuse or false positive blocks.

⚙ Workflow Summary

1. The **Agent Uptime Controller** routinely checks agent heartbeat timestamps.
2. If a status change is detected (up or down), it invokes the **Trigger Mechanism**.
3. The **Trigger Mechanism** determines if and which notifications should be sent, based on tag matching and cooldown logic.
4. Matched **Notifiers** are executed asynchronously or in queue, depending on system setup.
5. If a user **requests unblocking**, an internal event is logged and optionally notifies admins based on thresholds or flags.