config.yml Mangement and parameters

Overview

The PatchMon agent is configured through a YAML configuration file located at `/etc/patchmon/config.yml`. This file controls how the agent communicates with the PatchMon server, where logs are stored, which integrations are active, and other runtime behaviour. A separate credentials file (`/etc/patchmon/credentials.yml`) stores the host's API authentication details.

Both files are owned by root and set to `600` permissions (read/write by owner only) to protect sensitive information.

File Locations

File	Default Path	Purpose
Configuration	`/etc/patchmon/config.yml`	Agent settings, server URL, integrations
Credentials	`/etc/patchmon/credentials.yml`	API ID and API Key for host authentication
Log File	`/etc/patchmon/logs/patchmon-agent.log`	Agent log output
Cron File	`/etc/cron.d/patchmon-agent`	Scheduled reporting (fallback for non-systemd systems)

Full Configuration Reference

Below is a complete `config.yml` with all available parameters, their defaults, and descriptions:

```
# PatchMon Agent Configuration
# Location: /etc/patchmon/config.yml


# ── Server Connection ──────────────────────────────────────────
# The URL of the PatchMon server this agent reports to.
```

```
# Required. Must start with http:// or https://
patchmon_server: "https://patchmon.example.com"

# API version to use when communicating with the server.
# Default: "v1" — do not change unless instructed.
api_version: "v1"

# ─── File Paths ──────────────────────────────────────────────────
# Path to the credentials file containing api_id and api_key.
# Default: "/etc/patchmon/credentials.yml"
credentials_file: "/etc/patchmon/credentials.yml"

# Path to the agent log file. Logs are rotated automatically
# (max 10 MB per file, 5 backups, 14-day retention, compressed).
# Default: "/etc/patchmon/logs/patchmon-agent.log"
log_file: "/etc/patchmon/logs/patchmon-agent.log"

# ─── Logging ─────────────────────────────────────────────────────
# Log verbosity level.
# Options: "debug", "info", "warn", "error"
# Default: "info"
log_level: "info"

# ─── SSL / TLS ───────────────────────────────────────────────────
# Skip SSL certificate verification when connecting to the server.
# Set to true only if using self-signed certificates.
# Default: false
skip_ssl_verify: false

# ─── Reporting Schedule ──────────────────────────────────────────
# How often (in minutes) the agent sends a full report to the server.
# This value is synced from the server on startup. If the server has
# a different value, the agent updates config.yml automatically.
# Default: 60
update_interval: 60

# Report offset (in seconds). Automatically calculated from the host's
# api_id to stagger reporting across hosts and avoid thundering-herd.
# You should not need to set this manually — the agent calculates and
```

```
# persists it automatically.
# Default: 0 (auto-calculated on first run)
report_offset: 0

# ─── Integrations ──────────────────────────────────────────────
# Integration toggles control optional agent features.
# Most integrations can be toggled from the PatchMon UI and the server
# will push the change to the agent via WebSocket. The agent then
# updates config.yml and restarts the relevant service.
#
# EXCEPTION: ssh-proxy-enabled CANNOT be pushed from the server.
# It must be manually set in this file (see below).
integrations:
  # Docker integration — monitors containers, images, volumes, networks.
  # Can be toggled from the PatchMon UI (Settings → Integrations).
  # Default: false
  docker: false

  # Compliance integration — OpenSCAP and Docker Bench security scanning.
  # Three modes:
  #   false       — Disabled. No scans run.
  #   "on-demand" — Scans only run when triggered from the PatchMon UI.
  #   true        — Enabled with automatic scheduled scans every report cycle.
  # Can be toggled from the PatchMon UI.
  # Default: "on-demand"
  compliance: "on-demand"

  # SSH Proxy — allows browser-based SSH sessions through the agent.
  #     SECURITY: This setting can ONLY be enabled by manually editing
  #     this file. It cannot be pushed from the server to the agent.
  #     This is intentional — enabling remote shell access should require
  #     deliberate action by someone with root access on the host.
  # Default: false
  ssh-proxy-enabled: false
```

Parameters In Detail

`patchmon_server`

Type	String (URL)
Required	Yes
Default	None — must be provided
Example	`https://patchmon.example.com`

The full URL of the PatchMon server. Must include the protocol (`http://` or `https://`). Do not include a trailing slash or path.

`api_version`

Type	String
Required	No
Default	`v1`

The API version string appended to API calls. Leave as `v1` unless directed otherwise by PatchMon documentation or release notes.

`credentials_file`

Type	String (file path)
Required	No
Default	`/etc/patchmon/credentials.yml`

Path to the YAML file containing the host's `api_id` and `api_key`. The credentials file has this structure:

```
api_id: "patchmon_abc123def456"
api_key: "your_api_key_here"
```

`log_file`

Type	String (file path)
Required	No
Default	`/etc/patchmon/logs/patchmon-agent.log`

Path to the agent's log file. The directory is created automatically if it does not exist. Logs are rotated using the following policy:

- **Max file size**: 10 MB
- **Max backups**: 5 rotated files
- **Max age**: 14 days
- **Compression**: Enabled (gzip)

`log_level`

Type	String
Required	No
Default	`info`
Options	`debug`, `info`, `warn`, `error`

Controls the verbosity of agent logging. Use `debug` for troubleshooting — it includes API request/response bodies and detailed execution flow. Can also be overridden at runtime with the `--log-level` CLI flag.

`skip_ssl_verify`

Type	Boolean
Required	No
Default	`false`

When `true`, the agent skips TLS certificate verification when connecting to the PatchMon server. Use this only for internal/testing environments with self-signed certificates. **Not recommended for production.**

`update_interval`

Type	Integer (minutes)
Required	No
Default	`60`

How frequently the agent sends a full system report (installed packages, updates, etc.) to the server. This value is **synced from the server** — if you change the global or per-host reporting interval in the PatchMon UI, the agent will update this value in `config.yml` automatically on its next

startup or when it receives a settings update via WebSocket.

If the value is `0` or negative, the agent falls back to the default of 60 minutes.

`report_offset`

Type	Integer (seconds)
Required	No
Default	`0` (auto-calculated)

A stagger offset calculated from the host's `api_id` and the current `update_interval`. This ensures that agents across your fleet do not all report at the exact same moment (avoiding a thundering-herd problem on the server).

You should not set this manually. The agent calculates it on first run and saves it. If the `update_interval` changes, the offset is recalculated automatically.

`integrations`

A map of integration names to their enabled/disabled state. See the Integrations section below for details on each.

Integrations

Docker (`docker`)

Type	Boolean
Default	`false`
Server-pushable	Yes

When enabled, the agent monitors Docker containers, images, volumes, and networks on the host. It sends real-time container status events and periodic inventory snapshots to the PatchMon server.

Requirements: Docker must be installed and the Docker socket must be accessible.

Toggle from UI: Go to a host's detail page → Integrations tab → Toggle Docker on/off. The server pushes the change to the agent via WebSocket, the agent updates `config.yml`, and the service

restarts automatically.

Compliance (`compliance`)

Type	Boolean or String
Default	`"on-demand"`
Server-pushable	Yes
Valid values	`false`, `"on-demand"`, `true`

Controls OpenSCAP and Docker Bench security compliance scanning.

Value	Behaviour
`false`	Compliance scanning is fully disabled. No scans run.
`"on-demand"`	Scans only run when manually triggered from the PatchMon UI. Tools are installed but no automatic scheduled scans occur.
`true`	Fully enabled. Scans run automatically on every report cycle in addition to being available on-demand.

When first enabled, the agent automatically installs the required compliance tools (OpenSCAP, SSG content packages, Docker Bench image if Docker is also enabled).

SSH Proxy (`ssh-proxy-enabled`)

Type	Boolean
Default	`false`
Server-pushable	**No — manual edit required**

Enables browser-based SSH terminal sessions that are proxied through the PatchMon agent. When a user opens the SSH terminal in the PatchMon UI, the server sends the SSH connection request to the agent via WebSocket, and the agent establishes a local SSH connection on behalf of the user.

Why SSH Proxy Requires Manual Configuration

This is a deliberate security design decision. Enabling SSH proxy effectively allows remote shell access to the host through the PatchMon agent. Unlike Docker or compliance integrations, this has direct security implications:

- It opens an SSH connection path through the agent
- It could be exploited if a PatchMon server or user account were compromised

- The host administrator should make an informed, deliberate choice to enable it

For these reasons, `ssh-proxy-enabled` **cannot be toggled from the PatchMon UI or pushed from the server**. If the server attempts to initiate an SSH proxy session while this is disabled, the agent rejects the request and returns an error message explaining how to enable it.

How to Enable SSH Proxy

1. SSH into the host where the PatchMon agent is installed
2. Open the config file:

```
sudo nano /etc/patchmon/config.yml
```

3. Find the `integrations` section and change `ssh-proxy-enabled` to `true`:

```
integrations:
  docker: false
  compliance: "on-demand"
  ssh-proxy-enabled: true    # ← Change from false to true
```

4. Save the file and restart the agent:

```
# Systemd
sudo systemctl restart patchmon-agent.service

# OpenRC (Alpine)
sudo rc-service patchmon-agent restart
```

5. The SSH terminal feature is now available for this host in the PatchMon UI

How to Disable SSH Proxy

Set `ssh-proxy-enabled` back to `false` in `config.yml` and restart the agent service. Existing SSH sessions will be terminated.

How `config.yml` Is Generated

Initial Generation (Installation)

The `config.yml` file is created during agent installation by the `patchmon_install.sh` script. The installer generates a fresh config with:

- `patchmon_server` set to the server URL used during installation
- `skip_ssl_verify` set based on whether `-k` curl flags were used
- All integrations defaulted to `false` (Docker, SSH proxy) or `"disabled"` (compliance)
- Standard file paths for credentials and logs

```
# What the installer generates:
cat > /etc/patchmon/config.yml << EOF
# PatchMon Agent Configuration
# Generated on $(date)
patchmon_server: "https://patchmon.example.com"
api_version: "v1"
credentials_file: "/etc/patchmon/credentials.yml"
log_file: "/etc/patchmon/logs/patchmon-agent.log"
log_level: "info"
skip_ssl_verify: false
integrations:
  docker: false
  compliance: "disabled"
  ssh-proxy-enabled: false
EOF

chmod 600 /etc/patchmon/config.yml
```

Reinstallation Behaviour

If the agent is reinstalled on a host that already has a working configuration:

1. The installer **checks if the existing configuration is valid** by running `patchmon-agent ping`
2. If the ping succeeds, the installer **exits without overwriting** — the existing configuration is preserved
3. If the ping fails (or the binary is missing), the installer:
   - Creates a timestamped backup: `config.yml.backup.YYYYMMDD_HHMMSS`
   - Keeps only the last 3 backups (older ones are deleted)
   - Writes a fresh `config.yml`

This means **a reinstall on a healthy agent is safe** and will not destroy your configuration.

How `config.yml` Is Regenerated / Updated at Runtime

The agent updates `config.yml` automatically in several scenarios. These are **in-place updates** — the agent reads the file, modifies the relevant field, and writes it back. Your other settings (including `ssh-proxy-enabled`) are preserved.

Server-Driven Updates

Trigger	What Changes	How
Agent startup	`update_interval`, `report_offset`	Agent fetches the current interval from the server. If it differs from config, the agent updates config.yml.
Agent startup	`integrations.docker`, `integrations.compliance`	Agent fetches integration status from the server. If it differs from config, the agent updates config.yml.
WebSocket: `settings_update`	`update_interval`, `report_offset`	Server pushes a new interval. Agent saves it and recalculates the report offset.
WebSocket: `integration_toggle`	`integrations.*` (except SSH proxy)	Server pushes a toggle for Docker or compliance. Agent saves the change and restarts the relevant service.

Agent-Calculated Updates

Trigger	What Changes	How
First run	`report_offset`	Calculated from `api_id` hash and `update_interval` to stagger reports.
Interval change	`report_offset`	Recalculated whenever `update_interval` changes.
CLI: `config set-api`	`patchmon_server`, credentials	Running `patchmon-agent config set-api` overwrites the server URL and saves new credentials.

What Is Never Changed Automatically

Parameter	Why
`ssh-proxy-enabled`	Security — requires manual host-level action
`log_level`	Only changed by manual edit or `--log-level` CLI flag
`log_file`	Only changed by manual edit
`credentials_file`	Only changed by manual edit or `config set-api`
`skip_ssl_verify`	Only changed by manual edit

Important: How SaveConfig Works

When the agent calls `SaveConfig()` internally, it writes **all parameters** back to the file. This means:

- Your `ssh-proxy-enabled: true` setting is **preserved** across server-driven updates
- New integrations added in agent updates are **automatically added** to the file with their defaults (you'll see them appear after an agent update)
- The file format may be slightly reorganised by the YAML serialiser (key ordering may change), but all values are preserved

CLI Configuration Commands

The agent provides CLI commands for configuration management:

View Current Configuration

```
sudo patchmon-agent config show
```

Output:

```
Configuration:
  Server: https://patchmon.example.com
  Agent Version: 1.4.0
  Config File: /etc/patchmon/config.yml
  Credentials File: /etc/patchmon/credentials.yml
  Log File: /etc/patchmon/logs/patchmon-agent.log
  Log Level: info

Credentials:
  API ID: patchmon_abc123def456
  API Key: Set □
```

Set API Credentials

```
sudo patchmon-agent config set-api <API_ID> <API_KEY> <SERVER_URL>
```

Example:

```
sudo patchmon-agent config set-api patchmon_1a2b3c4d abcdef123456 https://patchmon.example.com
```

This command:

1. Validates the server URL format
2. Saves the server URL to `config.yml`
3. Saves the credentials to `credentials.yml`
4. Tests connectivity with a ping to the server
5. Reports success or failure

Custom Config File Path

All commands support a `--config` flag to use an alternative config file:

```
sudo patchmon-agent --config /path/to/custom/config.yml serve
```

Credentials File (`credentials.yml`)

The credentials file is separate from the config file for security isolation. It contains:

```
api_id: "patchmon_abc123def456"
api_key: "your_api_key_here"
```

- **Permissions**: `600` (root read/write only)
- **Written using atomic rename**: The agent writes to a temp file first, then atomically renames it. This prevents partial writes or race conditions.
- **Never contains the hashed key**: The plain-text API key is stored here; the server stores only the bcrypt hash.

Troubleshooting

Config File Missing

If `/etc/patchmon/config.yml` does not exist, the agent uses built-in defaults. This means it will not know which server to connect to. Reinstall the agent or create the file manually.

Config File Permissions

```
# Check permissions (should be 600, owned by root)
ls -la /etc/patchmon/config.yml

# Fix if needed
sudo chmod 600 /etc/patchmon/config.yml
sudo chown root:root /etc/patchmon/config.yml
```

SSH Proxy Not Working

If the SSH terminal in the PatchMon UI shows an error like:

> SSH proxy is not enabled. To enable SSH proxy, edit the file /etc/patchmon/config.yml...

This means `ssh-proxy-enabled` is set to `false` (the default). Follow the How to Enable SSH Proxy instructions above.

Config Gets Overwritten

If you notice settings being changed unexpectedly, check:

1. **Server sync**: The `update_interval` and integration toggles (Docker, compliance) are synced from the server on startup and via WebSocket. Changes made in the PatchMon UI will override local values for these fields.
2. **Agent updates**: After an agent update, new integration keys may appear in the file with default values.
3. **Reinstallation**: A reinstall only overwrites config if the existing ping test fails.

Your `ssh-proxy-enabled`, `log_level`, `skip_ssl_verify`, and file path settings are **never overwritten** by server sync.

Viewing Debug Logs

```
# Temporarily enable debug logging
sudo patchmon-agent --log-level debug serve

# Or set permanently in config.yml
sudo nano /etc/patchmon/config.yml
```

```
# Change: log_level: "debug"
# Then restart the service
sudo systemctl restart patchmon-agent.service
```

Example Configurations

Minimal Configuration

```
patchmon_server: "https://patchmon.example.com"
```

All other values use defaults. The agent will function with just the server URL (and valid credentials in `credentials.yml`).

Full Configuration with SSH Proxy Enabled

```
patchmon_server: "https://patchmon.internal.company.com"
api_version: "v1"
credentials_file: "/etc/patchmon/credentials.yml"
log_file: "/etc/patchmon/logs/patchmon-agent.log"
log_level: "info"
skip_ssl_verify: false
update_interval: 30
report_offset: 847
integrations:
  docker: true
  compliance: "on-demand"
  ssh-proxy-enabled: true
```

Self-Signed SSL with Debug Logging

```
patchmon_server: "https://patchmon.lab.local"
api_version: "v1"
credentials_file: "/etc/patchmon/credentials.yml"
log_file: "/etc/patchmon/logs/patchmon-agent.log"
log_level: "debug"
skip_ssl_verify: true
update_interval: 60
integrations:
```

```
  docker: false
  compliance: false
  ssh-proxy-enabled: false
```

Revision #3
Created 2026-02-12 03:04:31 UTC by Iby
Updated 2026-02-12 03:06:21 UTC by Iby