# IPS

# IPS

*Pro Edition feature.* Maps to **System > IPS** (`view_intrusion_prevention.cfm`, `inc/intrusion_prevention_generate_config.cfm`, `inc/intrusion_prevention_get_status.cfm`, `inc/intrusion_prevention_manual_ban.cfm`, `inc/intrusion_prevention_manual_unban.cfm`).

IPS (Intrusion Prevention System) is Hermes's brute-force defense layer. It binds two operational pieces together: the **`hermes_fail2ban`** container that scans authentication logs and inserts iptables drop rules, and a Hermes database/UI layer that lets an admin tune jail thresholds, manage a never-ban whitelist, manually ban or unban IPs, and see live ban counts. The page also doubles as a troubleshooting reference (the Info card lists every `docker exec` command an admin would need to chase a ban from the shell).

# Pipeline placement — where IPS sits in the stack

```
Attacker on the public internet
        |
        ▼
   Host network stack  (hermes_fail2ban runs network_mode: host)
        |
        ├─►  iptables DOCKER-USER chain
        |       └─►  f2b-dovecot, f2b-authelia chains  ◄── ban rules inserted here
        |
        ▼
   nginx / Docker bridge
        |
        ▼
   hermes_nginx ──► hermes_commandbox / hermes_authelia / hermes_dovecot
                              |
```


Two facts are worth pinning down before anything else:

| Fact | Consequence |
|---|---|
| `hermes_fail2ban` runs in **host network mode** | iptables rules apply to the Docker host directly, not to a bridge namespace. The DOCKER-USER chain is the entry point because Docker honors it before its own auto-inserted rules. |
| Docker DNS is **unavailable** inside the container | The notify script reads container IPs from `/opt/hermes/tmp/container_ips.env`, regenerated on every page load by `inc/generate_container_ips.cfm`. If that file is stale or missing, ban events still iptables-block correctly but fail to log to the database. |

# The container always runs — Pro gating is behavioral

`hermes_fail2ban` starts on every install regardless of edition. The Pro license check happens in CFML at page load, not at the container level. What changes on Community is:

- The configuration UI is replaced by the standard "Pro feature required" panel.

- Jail toggles in `intrusion_prevention_jails.enabled` and the master `intrusion_prevention_settings.enabled` switch default to disabled on a fresh install.
- The jail.local on disk reflects whatever the seed gave you; nothing rewrites it without an admin clicking through the page.

> **Operational consequence.** Stopping `hermes_fail2ban` to "turn off IPS on Community" is the wrong move. The container is needed for the schema, the include scripts, and the manual-unban API path. Leave it running; disable IPS through the UI when the page becomes accessible, or leave the seeded jails disabled.

# The two seeded jails

| Jail name | Display name | Log scanned | Filter | Action | Default thresholds |
|---|---|---|---|---|---|
| `dovecot` | Mail Server (Dovecot) | `/remotelogs/dovecot/dovecot-info.log` | `dovecot` (upstream Fail2ban filter) | `hermes-iptables-dovecot` | maxretry 5 / findtime 86400 (1 day) / bantime 1800 (30 min) |
| `authelia` | SSO Portal (Authelia) | `/remotelogs/authelia/authelia.log` | `authelia-auth` (Hermes-shipped) | `hermes-iptables-authelia` | maxretry 5 / findtime 86400 / bantime 1800 |

Both rows are seeded into `intrusion_prevention_jails` on install (see hermes_install.sql lines 845-846). Adding a third jail is a schema-row plus filter/action insertion exercise — there is no UI for it. The two-jail set covers the two real attack surfaces in Hermes: SMTP/IMAP login brute force and the web-console SSO login. Postfix's own brute-force protection (smtpd anvil rate limits) is the first line of defense for SMTP submission; this jail catches what gets past anvil.

The dovecot jail covers the `dovecot-info.log` line for failed authentication, not the Postfix auth log. SMTP-AUTH attempts terminate against Dovecot SASL — Postfix proxies SASL through Dovecot — so the dovecot filter sees both IMAP/POP and SMTP-AUTH failures from the same surface.

# Database schema

Three tables in the `hermes` database carry IPS state. A fourth (`fail2ban_ips`) is shared with the manual ban/unban flow and the API notify script.

| Table | Role | Notes |
|---|---|---|

| | | |
|---|---|---|
| `intrusion_prevention_settings` | Two key/value rows: `enabled` (master switch), `config_synced` (pending-changes flag) | INSERT IGNORE on install, so an admin's local tuning survives upgrades |
| `intrusion_prevention_jails` | One row per jail with display metadata + maxretry/findtime/bantime/enabled/config_synced | Includes the filter and action names that get baked into `jail.local` |
| `intrusion_prevention_whitelist` | One row per IP/CIDR to ignore — three protected entries (`127.0.0.1/8`, `::1`, `172.16.0.0/12`) cannot be deleted | Whitelist rows render into the `ignoreip` directive of `[DEFAULT]` in `jail.local` |
| `fail2ban_ips` | Live ban ledger — one row per (IP, jail) pair currently or recently banned | Written by `hermes-api-notify.sh` (automatic bans) or the CFML manual-ban handler (admin bans) |

The `config_synced` flag works the same way as on other pages: every write handler flips it to `0` and renders a yellow "Pending Changes" badge; **Apply Settings** runs the regen-and-reload sequence and flips it back to `1`. There is no incremental sync — every Apply rewrites the whole `jail.local` from scratch.

# Apply Settings — the regen sequence

`inc/intrusion_prevention_generate_config.cfm` runs five hard-sequenced steps:

1. **Read** `intrusion_prevention_whitelist` (excluding the three protected IPs to avoid double-listing them in `ignoreip`).
2. **Read** `intrusion_prevention_jails` ordered by `jail_name`.
3. **Render** `jail.local` content into a `<cfsavecontent>` block: `[DEFAULT]` with `ignoreip = 127.0.0.1/8 ::1 172.16.0.0/12 <user-whitelist>`, then a `[<jail_name>]` stanza per row.
4. **Write** the rendered config to `/opt/hermes/tmp/jail.local.tmp` (a shared host path mounted into both containers), then `docker exec hermes_fail2ban cp` it into `/config/fail2ban/jail.local` inside the fail2ban container. The two-step copy is required because the `hermes_commandbox` container can't write directly to fail2ban's `/config` mount.
5. **Reload** with `docker exec hermes_fail2ban fail2ban-client reload`, then flip both `intrusion_prevention_settings.config_synced` and every row's `intrusion_prevention_jails.config_synced` to `1`.

If any step fails, `ipSyncSuccess` stays `false`, the sync flags are **not** flipped, and the page surfaces the error banner from `cfcatch.message`. The next attempt retries from scratch — there is no half-applied state to clean up.

# What happens when IPS is disabled

The master `enabled = 0` toggle does two things synchronously, before the redirect:

1. Walks every enabled jail, runs `fail2ban-client status <jail>` to get the live banned IP list, then `fail2ban-client set <jail> unbanip <ip>` for each one. iptables drop rules are removed immediately.
2. Truncates `fail2ban_ips` so the DB ledger matches the now-empty iptables state.

After that, Apply Settings rewrites `jail.local` with `enabled = false` on every jail and reloads fail2ban — meaning **no new bans will be created**, and any in-flight attacker is immediately ungated. This is the right behavior for an emergency "I locked myself out" scenario, but the price is loss of the entire current ban list. Re-enabling does not restore prior bans.

# The IP Whitelist

Whitelist entries are static CIDR ranges that fail2ban's `ignoreip` directive treats as never-banable. The page accepts:

| Format | Example | Validation |
|---|---|---|
| IPv4 single | `192.168.1.100` | `inc/validate_ip_address.cfm` regex |
| IPv4 CIDR | `10.0.0.0/8` | IPv4 regex + numeric prefix 0–32 |
| IPv6 single | `::1` | `inc/validate_ip_address_ipv6.cfm` regex |
| IPv6 CIDR | `fe80::/10` | IPv6 regex + numeric prefix 0–128 |

The three protected entries (localhost v4, localhost v6, the Docker `172.16.0.0/12` block) are seeded on install and the delete handler refuses to remove them. The `172.16.0.0/12` entry exists because internal container-to-container traffic shows up in dovecot/authelia logs as coming from the Docker bridge — without it, an Authelia auth_request loop or a Dovecot LMTP redelivery could end up self-banning the gateway. The lock icon on those rows in the table reflects this.

# Manual Ban and Manual Unban

The Banned IPs card surfaces every row in `fail2ban_ips`, joined to `intrusion_prevention_jails` so the display picks up the friendly name and the bantime for the countdown column. Two admin actions sit on top of it:

# Manual Ban

`inc/intrusion_prevention_manual_ban.cfm` accepts an IP and a jail (or "ALL" to span every enabled jail). For each target jail:

1. Pre-check `fail2ban_ips` for an existing (IP, jail) row — skip if already banned in that jail.
2. Run `docker exec hermes_fail2ban fail2ban-client set <jail> banip <ip>`. Return value 1 (or "already banned" in the output) is treated as success.
3. Sleep 500 ms so the fail2ban action's `hermes-api-notify.sh` invocation has time to insert the row first.
4. `UPDATE fail2ban_ips SET ban_type='MANUAL', ban_source='ADMIN', note='Manually banned via Intrusion Prevention GUI' WHERE ip=... AND jail=...` — overwriting the AUTOMATIC row the notify script just inserted.

The 500 ms sleep is load-bearing: without it, the notify-script INSERT can race the manual UPDATE and the admin attribution is lost.

# Manual Unban

`inc/intrusion_prevention_manual_unban.cfm` accepts pipe-delimited `<ip>|<jail>` pairs from the checkbox row selection, runs `fail2ban-client set <jail> unbanip <ip>` for each pair, and deletes the matching row from `fail2ban_ips`. Errors from individual unbans don't abort the batch — the script counts successes and reports failures separately.

Manual bans are flagged as **Permanent** in the time-remaining column because they have no `bantime` from a jail — the absence of an automatic expiry is the whole point of a manual ban. The admin must explicitly unban them.

# The countdown timer

The Banned IPs DataTable renders a per-row countdown badge using the `banned_at + bantime` arithmetic done CFML-side, then a `data-unban-timestamp` attribute drives a 1-Hz JavaScript tick that recolors the badge as it counts down (yellow > red > expired). The countdown is purely cosmetic — the actual unban happens inside fail2ban's process based on the same arithmetic. If a row shows "Expired" but is still present, it just hasn't been reaped from `fail2ban_ips` yet; reload the page after a few seconds and it'll be gone.

# Operational truths about iptables backends

Modern Ubuntu hosts ship two iptables binaries: `iptables-legacy` (xtables / kernel `xt_*` modules) and `iptables-nft` (nftables backend with iptables-compatible CLI). The fail2ban container ships both. The page surfaces both command variants in the Info card precisely because the right one depends on which backend the host (and Docker) negotiated at install time:

```
docker exec hermes_fail2ban update-alternatives --display iptables
```

Picking the wrong one isn't catastrophic — it just shows empty chains, which can be confusing during a "why isn't my ban working?" investigation. The `hermes-iptables-*` action templates inside fail2ban use the alternatives-resolved `iptables` binary, so the daemon itself always picks the correct backend.

# License gating

The page is wrapped in the standard Pro check:

```
<cfif NOT isDefined("session.edition") OR session.edition NEQ "Pro">
    <cfset proFeatureName = "Intrusion Prevention">
    <cfinclude template="./inc/license_pro_required.cfm">
    <cfabort>
</cfif>
```

Community installs see the gating panel and cannot reach the UI. The `hermes_fail2ban` container continues to run, its seeded jails default to disabled, and `jail.local` on disk reflects whatever was last applied. There is no behind-the-scenes auto-disable on license-state change — switching from Pro to Community does not flip jails off.

# Files and containers touched

| Path | Owner | Role |
|---|---|---|
| `config/hermes/var/www/html/admin/2/view_intrusion_prevention.cfm` | `hermes_commandbox` | Main page (cards, modals, action handlers) |

| Path | Owner | Role |
|---|---|---|
| `config/hermes/var/www/html/admin/2/inc/intrusion_prevention_generate_config.cfm` | `hermes_commandbox` | Render `jail.local` + reload fail2ban |
| `config/hermes/var/www/html/admin/2/inc/intrusion_prevention_get_status.cfm` | `hermes_commandbox` | Live `fail2ban-client status` parsing for jail/ban counters |
| `config/hermes/var/www/html/admin/2/inc/intrusion_prevention_manual_ban.cfm` | `hermes_commandbox` | Multi-jail manual ban with API-notify race handling |
| `config/hermes/var/www/html/admin/2/inc/intrusion_prevention_manual_unban.cfm` | `hermes_commandbox` | Batch unban handler |
| `config/hermes/var/www/html/admin/2/inc/generate_container_ips.cfm` | `hermes_commandbox` | Writes `/opt/hermes/tmp/container_ips.env` for the notify script |
| `config/hermes/var/www/html/admin/2/inc/fail2ban_ban_unban.cfm` | `hermes_commandbox` | API endpoint hit by `hermes-api-notify.sh` (token-authed) |
| `config/fail2ban/config/fail2ban/jail.local` | `hermes_fail2ban` (mounted) | Live jail config — rewritten on every Apply |
| `config/fail2ban/scripts/hermes-api-notify.sh` | `hermes_fail2ban` | Posts ban/unban events back to Hermes API |
| `config/fail2ban/scripts/detect-iptables-backend.sh` | `hermes_fail2ban` | One-shot at container start to pick legacy vs nft |
| `/opt/hermes/tmp/jail.local.tmp` | both | Ephemeral rendered config; `docker exec cp`-ed into the fail2ban mount |
| `/opt/hermes/tmp/container_ips.env` | both | DB and Commandbox IPs for the API notify script (host networking has no DNS) |

# Related

- Admin Console Firewall — the complementary static-allowlist layer; IPS is reactive, Console Firewall is preventative
- Authentication Settings — Authelia's own Login Regulation (per-account brake) — the inner brake that complements this page's per-source-IP brake
- LDAP RemoteAuth — RemoteAuth-mode users also count against the authelia jail
- Console Settings — changing the console host triggers a full nginx regen but does not touch fail2ban

---

Revision #14
Created 2026-05-31 12:51:58 UTC by Dino Edwards
Updated 2026-06-13 12:30:02 UTC by Dino Edwards