FireDog — Manuale Utente

Gestione centralizzata del firewall per host Linux

Dognet Technologies S.r.l.

© Dognet Technologies S.r.l. — P.IVA 04867480164

Table of contents

1. Introduzione 4
   1.1 Cos'è FireDog 4
   1.2 A chi si rivolge questo manuale 4
   1.3 Novità della versione 2.0.0 4
   1.4 Cosa fa FireDog 4
   1.5 Cosa NON fa (per evitare aspettative sbagliate) 5
2. 1. Installazione 6
   2.1 Master (server web) 6
   2.2 Target (strumenti firewall) 7
   2.3 dog-agent 7
   2.4 Certificato HTTPS e agent 7
3. 2. Primo accesso 8
4. 3. Ruoli e permessi 9
   4.1 Admin 9
   4.2 Reporter 9
   4.3 Autenticazione 9
5. 4. Target e multi-NIC 10
   5.1 Censire un target 10
   5.2 Interfacce multiple 10
6. 5. Regole firewall 11
   6.1 Creare una regola 11
   6.2 Il campo `interface` 11
   6.3 Rimuovere una regola 11
   6.4 IP bloccati e whitelist 11
7. 6. Configurazione del target 12
   7.1 Versione del pacchetto target 12
   7.2 Impostazioni gestite dal master 12
   7.3 Altre impostazioni di Monitoraggio (sul master) 13
8. 7. Protezione SSH brute-force 14
   8.1 Sorgenti SSH sempre fidate 14
   8.2 Gestire i ban attivi 14
9. 8. dog-agent e pairing 15
10. 9. Server MCP 16
11. 10. Aggiornamenti 17
    11.1 Master 17

© Dognet Technologies S.r.l. — P.IVA 04867480164

11.2 Strumenti target 17
11.3 dog-agent 17
12. 11. Canale di test (pre-release) 18
12.1 Trovare una pre-release 18
12.2 Verifica e installazione 18
13. 12. Domande frequenti 19
13.1 Perché un target appare offline anche se l'agent è attivo? 19
13.2 Perché `ulogd2` va in "failed" o si blocca dopo un restart o un reboot? 19
13.3 Posso installare gli strumenti firewall senza attivare subito la policy DROP? 19
13.4 Come aggiungo una porta sempre aperta su un target già in produzione? 19
13.5 Come faccio a non restare bloccato fuori da SSH durante dei test, senza disattivare la protezione per tutti? 19
13.6 È possibile installare FireDog e CyberSheppard/SentinelCore sullo stesso host? 19
13.7 Perché SSH (o altre connessioni legittime) vengono bloccate dopo pochi tentativi ravvicinati? 19
13.8 Il browser dice "connessione non sicura": è normale? 20
13.9 Dopo l'aggiornamento il salvataggio delle impostazioni fallisce con "Connessione al master non riuscita" 20
13.10 L'agent non si collega più: nel log c'è `invalid peer certificate` 20
13.11 Ho salvato il threat-score in Settings → Monitoraggio ma l'agent usa ancora i vecchi valori 20
13.12 Non ricevo email/Slack per le minacce 20
13.13 Un IP bloccato riesce ancora a collegarsi via SSH 20

© Dognet Technologies S.r.l. — P.IVA 04867480164

1. Introduzione

1.1 Cos'è FireDog

FireDog è una piattaforma di **gestione centralizzata del firewall** per flotte di host Linux. Invece di configurare `iptables` a mano su ogni server, un unico pannello web (il **master**) permette di definire e distribuire regole a tutti gli host gestiti (i **target**), monitorarne il traffico bloccato e le minacce rilevate, e reagire da un solo posto.

Il prodotto è organizzato in tre componenti, ciascuno con un ruolo preciso:

Componente	Dove gira	Cosa fa
Master	una VM/host dedicato	Backend Django + frontend React: interfaccia web, API REST, server MCP; riceve heartbeat, statistiche e minacce dagli agent via WebSocket
Strumenti firewall	ogni target, in `/opt/sentinelsuite/firedog`	`firewall-manager` (CLI iptables), `traffic-analyzer` (analisi PCAP con threat scoring), `firewall-init.sh` (policy DROP + protezioni anti-attacco)
dog-agent	ogni target, `/usr/bin/dog-agent`	Binario Rust statico, condiviso con CyberSheppard e SentinelCore: autentica il target al master (pairing a 2 fasi), poi pusha heartbeat/statistiche/minacce via WebSocket ed esegue i comandi regola ricevuti

1.2 A chi si rivolge questo manuale

- **Chi installa e amministra** FireDog (un amministratore di sistema) — capitoli 1–2.
- **Chi lo usa ogni giorno** per gestire target, regole e configurazione — capitoli 3 in poi.

1.3 Novità della versione 2.0.0

La 2.0.0 consolida tutto quanto introdotto nella serie 1.x in una release stabile. In sintesi, rispetto alle prime versioni:

- **HTTPS di default sul master** — CA locale per ogni installazione che firma il certificato del server, rinnovo automatico, nginx con redirect `:80` → `:443` (`--skip-tls` per restare in solo-HTTP).
- **Trasporto agent cifrato e verificato** — `dog-agent` 1.3+ dialoga col master via `wss://` verificando il certificato contro la CA del master (`ca_cert`); `ws://` resta per compatibilità con master senza HTTPS.
- **Distribuzione flessibile** — immagini VM `.qcow2` (KVM/Proxmox) e `.ova` (VirtualBox/VMware), pacchetto `.deb`, e build settimanali di pre-release.
- **Appliance e identità suite** — login SSH appliance con utente `microcyber` (root bloccato), pagina di login condivisa `FireDog | SentinelSuite`.
- **Analisi del traffico configurabile** — impostazioni pcap (path, scelta `tcpdump` / `tshark`, fino a 5 filtri custom) e **threat-score per-target editabile dalla UI** e spinto all'agent, con default ricalibrati meno aggressivi.
- **Aggiornamenti sicuri** — `upgrade.sh` con backup del DB e rollback automatico se le migration falliscono.
- **Minacce e alert affidabili** — le minacce rilevate dall'agent arrivano sempre al master; alert email/Slack/Discord per minacce gravi e target offline; conservazione dei log configurabile.
- **Pagina Threats ridisegnata** con filtri e Top Attackers, e **dettaglio minaccia** con output tcpdump dal pcap.
- **Più piattaforme** — installazione strumenti target anche su Fedora (`dnf`) e openSUSE/SLES, oltre a Debian/Ubuntu.
- **Backend su Django 5.2 LTS** e dipendenze aggiornate.

1.4 Cosa fa FireDog

- **Gestione centralizzata delle regole** — CRUD standard su `/api/rules/`, con dispatch in tempo reale all'agent via WebSocket; se l'agent non è connesso la regola resta persistita in attesa della prossima riconciliazione.

© Dognet Technologies S.r.l. — P.IVA 04867480164

- **Supporto host multi-NIC** — un target può avere più interfacce di rete; le regole possono essere scoped su un'interfaccia specifica, con contatori rx/tx per NIC.
- **Policy di default DROP** con protezioni integrate contro SYN-flood, port-scan e brute-force SSH (soglia, finestra e tipo di ban configurabili).
- **Cattura e analisi del traffico** — il traffico bloccato viene loggato in PCAP (ulogd2) e analizzato dal `traffic-analyzer`, che assegna un punteggio di minaccia.
- **Dashboard e monitoraggio** — stato dei target, statistiche di traffico, minacce rilevate, tutto in tempo reale via WebSocket.
- **Audit log** — ogni operazione di scrittura (regole, IP bloccati, configurazione) viene registrata con utente, azione e valori.
- **Server MCP** — un'interfaccia programmatica (Model Context Protocol) che permette ad agenti AI autorizzati di consultare — e, con una chiave di scrittura, agire su — regole, minacce, traffico e stato dei target.

1.5 Cosa NON fa (per evitare aspettative sbagliate)

- **Non installa il firewall via push SSH dal master**: il provisioning di un target avviene sempre in autonomia sul target stesso, tramite lo script di bootstrap (capitolo 1) — nessuna credenziale SSH è mai condivisa tra target e master.
- **Non ha un terminale SSH integrato** nella UI: la gestione del target avviene tramite la CLI `firewall-manager` in locale o tramite le regole distribuite dal master.
- **Non è multi-tenant**: un'installazione master serve un'unica organizzazione/flotta.
- **Non sostituisce un IDS/IPS completo**: il *threat scoring* del traffic-analyzer è un'euristica su traffico bloccato, non un'analisi approfondita dei pacchetti.

© Dognet Technologies S.r.l. — P.IVA 04867480164

2. 1. Installazione

L'installazione di FireDog è divisa in tre passi indipendenti, nell'ordine in cui vanno normalmente eseguiti: prima il master, poi gli strumenti firewall su ogni target, infine il pairing di dog-agent.

2.1 Master (server web)

Il master è il pannello web. Può essere installato in tre modi; il primo è quello consigliato.

2.1.1 Requisiti

Risorsa	Dettaglio
Sistema operativo	Debian 12/13 · Ubuntu 22.04+ (installer anche Fedora/openSUSE per i target)
Runtime	Python 3.11–3.13 · Node 20
Servizi	PostgreSQL · Redis · nginx
Memoria	2 GB minimo, 4 GB consigliati

2.1.2 Modalità A — pacchetto `.deb` (consigliata)

Sulla **branch** `prerelease` ogni push produce un pacchetto `.deb` per Debian 13 (test channel; scaricabile da *Actions* → *Prerelease* → *Artifacts* → `firedog-deb`). Per la produzione si usano i `.deb` delle release.

```
sudo apt install ./firedog_<versione>_amd64.deb
```

Usa `apt`, non `dpkg -i`, così risolve le dipendenze. Il primo install configura master e database; su un'installazione esistente esegue backup del DB e migration. L'installer configura da sé **HTTPS di default** (CA locale dell'installazione che firma il certificato del server, nginx con redirect `:80` → `:443`; vedi Certificato HTTPS e agent) e genera `SECRET_KEY` e password DB. Per un master di sviluppo o test usa `--tls-profile testing`: certificati marcati "TESTING solo uso interno", CA 12 mesi, server 6 mesi.

2.1.3 Modalità B — appliance VM (`.ova` / `.qcow2`)

Immagini pronte all'uso: `.ova` (VirtualBox/VMware) e `.qcow2` (KVM/Proxmox). Al primo avvio ogni clone rigenera segreti, password DB, CA locale e certificato TLS (legato all'IP reale; ogni appliance ha la sua CA), e stampa le credenziali iniziali in `/etc/motd`. Sulle appliance il login SSH usa l'utente `microcyber` con sudo (root bloccato); le credenziali iniziali spariscono dopo il primo cambio password.

2.1.4 Modalità C — installer automatico / manuale

Sul repository clonato:

```
sudo ./deploy/install.sh       # HTTPS di default; --skip-tls per solo-HTTP; --help per le opzioni
```

Configura PostgreSQL/Redis, virtualenv Python, build del frontend React, unit systemd per Daphne (ASGI) e Celery, e nginx come reverse proxy in HTTPS. Per l'installazione completamente manuale, passo-passo, vedi `INSTALL.md` nel repository.

© Dognet Technologies S.r.l. — P.IVA 04867480164

2.2 Target (strumenti firewall)

Su ogni host da gestire, esegui lo script di bootstrap: scarica il pacchetto degli strumenti firewall e li installa in autonomia — nessun accesso dal master è necessario.

```
# scarica, ispeziona ed esegui (consigliato)
curl -fsSL https://raw.githubusercontent.com/Dognet-Technologies/firedog/stabile/firedog-package/get-firedog.sh -o get-firedog.sh
less get-firedog.sh
sudo bash get-firedog.sh           # oppure: sudo bash get-firedog.sh --skip-init

# in alternativa, one-liner diretto
curl -fsSL https://raw.githubusercontent.com/Dognet-Technologies/firedog/stabile/firedog-package/get-firedog.sh | sudo bash
```

Distribuzioni supportate:

Famiglia	Gestore pacchetti	Note
Debian / Ubuntu	`apt`	persistenza regole via `iptables-persistent`
openSUSE / SLES	`zypper`	persistenza via `firewall-fm.service`; **firewalld** viene disabilitato all'attivazione (con conferma)

Attenzione: l'attivazione del firewall applica una policy **DROP** su INPUT/OUTPUT. Assicurati di avere accesso console/seriale prima di confermare, oppure usa `--skip-init` e attiva in un secondo momento con `sudo firewall-init.sh && sudo systemctl enable --now firewall-fm`.

Lo script è idempotente: si può rilanciare per aggiornare gli strumenti su un target già installato.

2.3 dog-agent

Il pairing tra target e master avviene tramite dog-agent, l'agent condiviso da tutta la suite Dognet. Vedi il capitolo 8 per i dettagli del pairing.

2.4 Certificato HTTPS e agent

Ogni installazione ha la **propria CA**, creata da `firedog-tls`. La chiave della CA resta solo sul master; la CA firma il certificato del server (il solo IP o nome del master, mai wildcard), che un timer giornaliero rinnova prima della scadenza senza toccare agent e browser.

Percorso	Contenuto
`/etc/sentinelsuite/firedog/tls.conf`	`PROFILE` (`production` / `testing`), `SERVER_NAME`, `MANAGED`
`/etc/sentinelsuite/firedog/tls/`	CA e certificato del server; chiavi in `tls/private/` (solo root)

- **Impronta e scadenze**: *Settings → API Keys Agent → Certificato del master*, oppure `sudo firedog-tls info`.
- **dog-agent 1.3 e successivi** verificano il certificato del master: scarica la CA dalla stessa scheda (o da `https://<master>/api/tls/ca.pem`), confronta l'impronta e impostala in `ca_cert` dell'agent (capitolo 8).
- **Browser**: importa la stessa CA come autorità per i siti web, oppure accetta l'avviso.
- **Il master ha cambiato IP**: `sudo firedog-tls ensure --name <nuovo-ip>` (stessa CA, nuovo certificato; `SERVER_NAME` in `tls.conf` si aggiorna da solo), aggiungi il nuovo IP ad `ALLOWED_HOSTS` in `backend/.env` e riavvia `firedog-backend`. Negli agent cambia solo `url`.
- **Certificato della PKI aziendale**: `MANAGED=no` in `tls.conf` e nginx puntato ai tuoi file; `firedog-tls` non tocca più nulla.
- **Nuova CA** (es. chiave compromessa): `sudo firedog-tls renew --force-ca`; gli agent vanno riconfigurati con la nuova CA.

© Dognet Technologies S.r.l. — P.IVA 04867480164

3. 2. Primo accesso

Apri il browser all'indirizzo del master. Con l'installer automatico, il `.deb` o l'appliance il sito è servito in **HTTPS di default** (`https://<indirizzo-master>`), con un certificato firmato dalla CA locale dell'installazione: finché il browser non conosce quella CA mostra un avviso. Puoi accettarlo, oppure importare la CA (*Settings → API Keys Agent → Certificato del master*, dopo aver confrontato l'impronta) e l'avviso sparisce. Per un certificato della tua PKI aziendale vedi il capitolo 1. Solo con `install.sh --skip-tls` o un'installazione manuale senza TLS il sito resta su `http://`.

La pagina di login è quella **condivisa dalla SentinelSuite** (titolo scheda `FireDog | SentinelSuite`). Accedi con le credenziali admin create durante l'installazione; sulle appliance le credenziali iniziali sono in `/etc/motd` e il primo login SSH usa l'utente `microcyber`.

Dopo il primo accesso troverai:

- **Targets** — elenco degli host gestiti, stato online/offline, versione firedog installata, interfacce di rete;
- **Firewall** — vista unificata delle regole, IP bloccati e whitelist su tutti i target;
- **Threats** — minacce rilevate dal traffic-analyzer, con punteggio e possibilità di risoluzione;
- **Dashboard** — panoramica aggregata di traffico e stato della flotta;
- **Settings** — API key per l'agent, API key MCP, notifiche, gestione utenti.

© Dognet Technologies S.r.l. — P.IVA 04867480164

4. 3. Ruoli e permessi

FireDog usa i gruppi standard di Django per due ruoli.

4.1 Admin

Permessi completi:

- visualizzare tutti i dati (target, regole, statistiche, minacce);
- creare/modificare/eliminare target;
- aggiungere/rimuovere regole firewall e bloccare/sbloccare IP;
- modificare configurazioni di sistema;
- accesso a Django Admin.

4.2 Reporter

Permessi in sola lettura:

- visualizzare target e loro stato, regole firewall, minacce rilevate, statistiche e dashboard, audit log;
- **non può** creare, modificare o eliminare alcuna risorsa.

4.3 Autenticazione

FireDog usa **JWT (JSON Web Token)**: un token di accesso (60 minuti) e uno di refresh (24 ore), ottenuti da `POST /api/token/` e usati come header `Authorization: Bearer <token>` su tutte le chiamate API.

© Dognet Technologies S.r.l. — P.IVA 04867480164

5. 4. Target e multi-NIC

5.1 Censire un target

Un target si crea da **Targets → Add** indicando **IP, hostname e MAC address esatti** della macchina: il master ne calcola l' `identity_hash = SHA512(ip+hostname+mac)` , usato in fase di pairing dell'agent (capitolo 8) per verificare che sia davvero quella macchina a connettersi.

5.2 Interfacce multiple

Se un host espone più interfacce di rete, FireDog le rileva tutte automaticamente tramite l'export periodico dell'agent e le elenca nella vista di dettaglio del target, con:

- nome interfaccia, indirizzo IP, indirizzo MAC;
- indicatore di interfaccia primaria;
- contatori di traffico ricevuto/inviato (rx/tx bytes e packets), aggiornati a ogni heartbeat;
- un **selettore NIC** per filtrare statistiche e regole per interfaccia specifica.

Per default vengono riportate tutte le interfacce rilevate; è possibile restringere l'elenco (es. per escludere interfacce virtuali come `docker0` o `veth*`) tramite `MONITORED_INTERFACES` nel file di configurazione del target — vedi capitolo 6.

© Dognet Technologies S.r.l. — P.IVA 04867480164

6. 5. Regole firewall

Le regole sono un CRUD standard su `/api/rules/` : lettura per tutti gli utenti autenticati, scrittura riservata al ruolo Admin.

6.1 Creare una regola

```
POST /api/rules/
Authorization: Bearer <token>
Content-Type: application/json

{
    "target": 1,
    "chain": "INPUT",
    "port": 80,
    "protocol": "tcp",
    "action": "ACCEPT",
    "source_ip": "192.168.1.0/24",  // opzionale
    "interface": "eth0",             // opzionale, NIC specifica su host multi-NIC
    "comment": "HTTP traffic"        // opzionale
}
```

Alla creazione, il master tenta subito di inviare il comando al target via WebSocket. Se l'agent è connesso, la regola viene applicata immediatamente e marcata `is_synced: true` ; se il target è offline, resta comunque salvata nel database (`is_synced: false`) e verrà applicata alla prossima riconnessione dell'agent.

6.2 Il campo `interface`

Su un target con più NIC, valorizzare `interface` applica la regola come `-i <interfaccia>` (chain INPUT) o `-o <interfaccia>` (chain OUTPUT) invece che su tutte le interfacce. Non è supportato sulla chain FORWARD.

6.3 Rimuovere una regola

```
DELETE /api/rules/{id}/
Authorization: Bearer <token>
```

6.4 IP bloccati e whitelist

Oltre alle regole generiche, FireDog gestisce due liste dedicate — **Blocked IPs** (IP bloccati manualmente o dal threat-scoring) e **Whitelist** (IP sempre permessi, mai bloccati automaticamente) — entrambe distribuite ai target con lo stesso meccanismo di dispatch via WebSocket.

© Dognet Technologies S.r.l. — P.IVA 04867480164

7. 6. Configurazione del target

`/etc/firewall/firedog.conf` è il file di configurazione locale del target, seedato al primo install (come `custom_rules.conf`: non viene mai sovrascritto agli aggiornamenti). Formato `KEY="value"`, leggibile sia da bash (`firewall-init.sh`) sia da `firewall-manager.py`.

```
# Interfacce da monitorare/riportare al master (separate da virgola).
# Vuoto = tutte le interfacce rilevate (default).
MONITORED_INTERFACES="eth0,eth1"

# Porte da tenere sempre aperte in INPUT, prima della policy DROP finale.
# La porta SSH è già protetta a parte, non va elencata qui.
ALWAYS_OPEN_PORTS="80/tcp,443/tcp"

# Sorgenti SSH sempre fidate (IP/CIDR, separate da virgola) — vedi capitolo 7.
TRUSTED_SSH_SOURCES="10.10.0.0/24"

# Protezione SSH brute-force — vedi capitolo 7. Default ricalibrati meno
# aggressivi (prima bastavano pochi tentativi ravvicinati per un blocco).
SSH_PROTECT_MAX_ATTEMPTS="20"
SSH_PROTECT_WINDOW_SECONDS="60"
SSH_PROTECT_BAN_DURATION="0"

# Controllo periodico aggiornamenti del pacchetto target (giorni, "0" =
# disabilitato di default). Solo segnalazione, non installa nulla da solo.
UPDATE_CHECK_INTERVAL_DAYS="0"
```

7.1 Versione del pacchetto target

Il file `/opt/sentinelsuite/firedog/VERSION` (seedato da `install.sh`, aggiornato ad ogni `get-firedog.sh`/update) è la fonte unica della versione installata — **distinta** dalla versione del master (backend+frontend), che si aggiorna con un processo separato (`git pull` + migration) e può quindi trovarsi temporaneamente a una versione diversa da quella di un target non ancora aggiornato.

```
firewall-manager --version               # versione installata su questo target
sudo firewall-manager --check-updates    # controlla subito se c'è una versione più recente
```

Il controllo periodico (`UPDATE_CHECK_INTERVAL_DAYS`) è agganciato al già esistente export ogni 5 minuti verso il master (`firewall-manager --export-json`, via crontab): non serve un cron job dedicato, si auto-limita all'intervallo configurato e confronta il file `VERSION` locale con quello pubblicato sul branch `stabile`. Disabilitato di default (nessuna chiamata di rete non richiesta); è solo segnalazione — non installa nulla in autonomia.

Importante: `ALWAYS_OPEN_PORTS` e `TRUSTED_SSH_SOURCES` vanno valorizzati **prima** di lanciare `firewall-init.sh` la prima volta — senza, qualunque servizio in ascolto su quelle porte diventerebbe irraggiungibile (o SSH resta soggetto al rate-limit) non appena la policy DROP entra in vigore. Dopo una modifica, applica con `sudo firewall-init.sh` (idempotente: va rilanciato per raccogliere le modifiche).

Attenzione: aggiungere la porta 22 a `ALWAYS_OPEN_PORTS` **non** esenta SSH dalla protezione brute-force — quella porta è già intercettata dalla chain `SSH_PROTECT` prima che `ALWAYS_OPEN_PORTS` venga anche solo valutato. Per bypassare interamente `SSH_PROTECT` da sorgenti fidate (rete di management, VPN, jump host) usa `TRUSTED_SSH_SOURCES`, non `ALWAYS_OPEN_PORTS` — vedi capitolo 7.

7.2 Impostazioni gestite dal master

Alcune impostazioni non si toccano in `firedog.conf` ma si configurano dal pannello e vengono applicate al target senza intervento manuale:

- **Threat-score per-target** *(pagina del target → Configurazione)* — la formula di scoring di `dog-agent` (soglia di blocco automatico, soglie di innesco brute-force e SYN-flood, punti per evento) si modifica dalla UI e viene **spinta all'agent** via comando remoto (`agent.conf` riscritto, servizio riavviato in automatico). I valori in *Settings → Monitoraggio → Threat-score agent* sono solo i **default proposti**: non arrivano agli agent finché non li invii dalla pagina del singolo target.
- **Impostazioni pcap** *(Settings → Monitoraggio)* — path di salvataggio dei PCAP, scelta tra `tcpdump` e `tshark`, fino a **5 filtri custom** e i comandi tcpdump per il pannello *Dettagli minaccia*. A differenza del threat-score, al salvataggio vengono **inviate subito a tutti i target collegati** (l'agent le scrive in `firedog.conf`, senza riavvii); un target offline le riceve al salvataggio successivo.

© Dognet Technologies S.r.l. — P.IVA 04867480164

7.3 Altre impostazioni di Monitoraggio (sul master)

- **Soglia invio alert** (1-10) — le minacce *high* e *critical* con score almeno pari a questo valore × 10 (es. 8 = score 80/100) inviano un alert email/Slack/Discord, una volta quando vengono rilevate o quando si aggravano, non a ogni ripetizione. Non è la soglia di blocco automatico dell'agent.
- **Conservazione log (giorni)** — ogni notte vengono cancellati minacce, alert, notifiche inviate, comandi conclusi e statistiche più vecchi di così (7-365 giorni). L'audit log non viene toccato.
- Gli **alert di target offline** partono quando un agent non risponde da più di 2 minuti, una sola volta per disconnessione.

© Dognet Technologies S.r.l. — P.IVA 04867480164

8. 7. Protezione SSH brute-force

8.1 Sorgenti SSH sempre fidate

`TRUSTED_SSH_SOURCES` in `firedog.conf` (lista di IP/CIDR separati da virgola, es. `"10.10.0.0/24,203.0.113.5"`) fa bypassare **interamente** la chain `SSH_PROTECT` alle sorgenti elencate: niente rate limit, niente ban, nessuna dipendenza dalle soglie sotto. Pensato per la propria rete di management/bastion (VPN, jump host) — **non** un modo per esporre SSH al pubblico: le sorgenti elencate restano protette solo dall'autenticazione SSH stessa. Vuoto di default, nessun impatto se non configurato.

Oltre alla soglia di tentativi/finestra temporale (`SSH_PROTECT_MAX_ATTEMPTS` / `SSH_PROTECT_WINDOW_SECONDS`), FireDog può reagire al superamento della soglia con un **ban vero e proprio**, non solo il drop nella finestra:

Valore `SSH_PROTECT_BAN_DURATION`	Comportamento
`0` (default)	Nessun ban: la sorgente resta droppata solo finché continua a generare nuovi tentativi nella finestra
`<N>m` / `<N>h` / `<N>d`	Ban temporaneo (minuti/ore/giorni): la sorgente viene aggiunta a una ipset persistente e scade da sola
`permanent`	Ban permanente, finché non viene rimosso a mano

Il ban richiede il pacchetto `ipset` (installato di default dallo script target) ed è **persistente**: sopravvive a un rilancio di `firewall-init.sh` e, tramite salvataggio periodico da cron, anche a un riavvio della macchina.

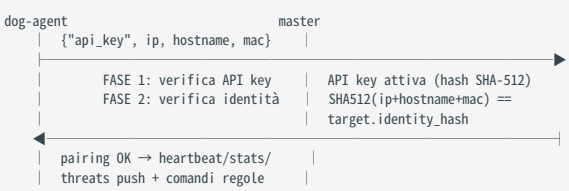
8.2 Gestire i ban attivi

```
firewall-manager --list-bans        # IP bannati e tempo alla scadenza (o "permanente")
firewall-manager --unban 203.0.113.5  # rimuove un ban, incluso uno permanente
```


© Dognet Technologies S.r.l. — P.IVA 04867480164

9. 8. dog-agent e pairing

L'agent si autentica al master con un **pairing a 2 fasi** sul WebSocket `ws(s)://<master>/ws/agent/`:



1. **Sul master** — *Settings* → *API Keys Agent* → genera una nuova API key (globale per la flotta, hashata SHA-512).
2. **Sul master** — censisci il target (capitolo 4) con IP, hostname e MAC esatti.
3. **Sul master** — clicca **Pair** sul target appena creato (o `POST /api/agent/pairing/start/` con `{"target_id": <id>}`). Apre una finestra di pairing di **3 minuti**: l'agent deve connettersi entro quel tempo. Passo facile da saltare seguendo solo la creazione del Target — senza, la fase 2 fallisce con "Identity hash verification failed" anche con IP/hostname/MAC corretti, perché il server non ha ancora una sessione di pairing in attesa a cui associare la richiesta.
4. **Sul target** — installa dog-agent (vedi la pagina dedicata) e configura `/etc/dog-agent/agent.conf`:

```
[agent]
ca_cert     = "/etc/sentinelsuite/dog_agent/ca-bundle.pem"   # CA del master (vedi sotto)

[[targets]]
system_type = "firedog"
url         = "https://<master>"        # → wss:// cifrato (consigliato); http:// → ws:// in chiaro
api_key     = "<chiave generata al passo 1>"
ip          = "<ip del target>"         # identici al Target censito
hostname    = "<hostname>"
mac         = "aa:bb:cc:dd:ee:ff"
```

Certificato del master. Con `url` in `https://` l'agent dialoga col master via **`wss://`** e, da dog-agent 1.3, **verifica il certificato** contro le CA in `ca_cert`: senza `ca_cert` l'agent non parte. Scarica la CA da *Settings* → *API Keys Agent* → *Certificato del master* (o `https://<master>/api/tls/ca.pem`), confronta l'impronta SHA-256 con quella mostrata nella scheda e copiala in `/etc/sentinelsuite/dog_agent/ca-bundle.pem`. Se l'agent parla anche con CyberShepherd, metti entrambe le CA nello stesso file. `url` deve usare lo stesso IP o nome del certificato. Dettagli nel manuale di dog-agent.

5. **Sul target** — `sudo systemctl enable --now dog-agent` **entro la finestra di 3 minuti** del passo precedente, e verifica con `journalctl -u dog-agent -f` (cerca "pairing success").

Se il pairing fallisce: API key errata/disattivata (fase 1); oppure, in fase 2, ip/hostname/mac che non coincidono col Target sul master — o la finestra di pairing non è stata avviata o è scaduta — l'errore è visibile sia nei log dell'agent sia nella UI del master.

Se l'agent non si collega e nel log c'è `invalid peer certificate`: `UnknownIssuer` = in `ca_cert` manca la CA di questo master; `Other(OtherError(CaUsedAsEndEntity))` = il master usa ancora il vecchio certificato self-signed (aggiornalo); `NotValidForName` = `url` usa un indirizzo diverso da quello del certificato.

© Dognet Technologies S.r.l. — P.IVA 04867480164

10. 9. Server MCP

FireDog espone un server **MCP** (Model Context Protocol) su `POST /api/mcp` (JSON-RPC 2.0, autenticazione Bearer con API key dedicata) che permette ad agenti AI autorizzati di consultare — e, se la chiave lo consente, agire su — la piattaforma in modo programmatico: elenco target, regole firewall, minacce, statistiche di traffico e network flow.

Le chiavi si generano da **Settings**, con lo stesso principio di scope in lettura/scrittura usato per le altre integrazioni della suite: una chiave in sola lettura può solo consultare dati, una chiave di scrittura può anche creare/eliminare regole e bloccare IP. Le operazioni di scrittura via MCP sono registrate nell'audit log come qualsiasi altra modifica.

© Dognet Technologies S.r.l. — P.IVA 04867480164

11. 10. Aggiornamenti

11.1 Master

A seconda di come è stato installato:

- **Pacchetto `.deb`** (via principale) — reinstalla la nuova versione; su un'installazione esistente esegue da sé **backup del DB e migration**:

```
sudo apt install ./firedog_<versione>_amd64.deb
```

- **Tarball di release / appliance / installazione da sorgente** — usa `deploy/upgrade.sh`, che fa **backup del database** e **rollback automatico** se le migration falliscono (prima lo si faceva a mano, col rischio di lasciare i servizi spenti):

```
sudo ./deploy/upgrade.sh
```

In entrambi i casi la versione del master è distinta da quella installata sui target, che si aggiornano a parte (vedi sotto).

Aggiornando da una versione con il vecchio certificato self-signed (`/etc/ssl/firedog`), l'aggiornamento crea la CA locale e un certificato nuovo e ripunta nginx (con controllo `nginx -t` e ripristino se fallisce). Dopo:

- il browser chiede di nuovo di accettare il certificato: una pagina già aperta va **ricaricata**, altrimenti i salvataggi falliscono con "Connessione al master non riuscita";
- gli agent dog-agent 1.3+ vanno configurati con la nuova CA (capitolo 8).

Su un'installazione da **clone git** (`update.sh`) il certificato non viene migrato in automatico: installa una volta `firedog-tls` e il suo timer e lancia `sudo firedog-tls ensure` (istruzioni in `INSTALL.md` del repository).

11.2 Strumenti target

Rilancia lo script di bootstrap — è idempotente, aggiorna gli strumenti già installati senza toccare `custom_rules.conf` o `firedog.conf`:

```
curl -fsSL https://raw.githubusercontent.com/Dognet-Technologies/firedog/stabile/firedog-package/get-firedog.sh -o get-firedog.sh
sudo bash get-firedog.sh
```

11.3 dog-agent

Scarica il nuovo pacchetto dalla release più recente e reinstalla (upgrade in place, la configurazione esistente non viene toccata):

```
sudo dpkg -i dog-agent_<versione>-1_amd64.deb   # Debian/Ubuntu
sudo zypper install ./dog-agent-<versione>-1.x86_64.rpm  # openSUSE/SLES
```


© Dognet Technologies S.r.l. — P.IVA 04867480164

12. 11. Canale di test (pre-release)

Oltre alle release ufficiali su `stabile`, è possibile provare funzionalità già pronte ma non ancora passate dai test completi di produzione. Non è un branch separato da seguire: ogni release — ufficiale o di test — nasce da un **tag git** sulla GitHub Releases del repo. Un tag con un suffisso (`-beta`, `-rc1`, `-test`…) marca automaticamente la release come **Pre-release** su GitHub, visibile come tale nella pagina delle release.

Attenzione: una pre-release ha superato la CI automatica (test backend/frontend, audit di sicurezza) ma **non** il ciclo di validazione completo che precede una release su `stabile`. Contiene funzionalità reali e funzionanti, non un prototipo — ma non è pensata per ambienti di produzione. Usala su un target di test, non sul firewall che protegge infrastruttura reale.

12.1 Trovare una pre-release

Nella pagina Releases del repo, le pre-release sono etichettate **Pre-release** (badge giallo) accanto al tag — distinte dalle release ufficiali (**Latest**). Ogni release, ufficiale o di test, porta gli stessi asset:

Asset	Uso
`firedog-<tag>-linux-x86_64.tar.gz`	Pacchetto installabile (backend + frontend + `install.sh`) su un host Debian 13 già pronto
`firedog-<tag>-linux-x86_64.qcow2`	VM appliance pronta per KVM/Proxmox — niente da installare, si avvia e basta
`firedog-<tag>-linux-x86_64.ova`	VM appliance pronta per VirtualBox/VMware
`SHA256SUMS` / `SHA256SUMS.asc`	Checksum di tutti gli asset, firmati GPG

12.2 Verifica e installazione

```
# Verifica l'integrità e l'autenticità prima di eseguire qualunque cosa
gpg --import firedog-release.pub.asc
gpg --verify SHA256SUMS.asc SHA256SUMS
sha256sum -c SHA256SUMS --ignore-missing

# Pacchetto su un host esistente
tar xzf firedog-<tag>-linux-x86_64.tar.gz
cd firedog-<tag>-linux-x86_64
sudo ./deploy/install.sh
# per aggiornare un'istanza già esistente invece di un primo install:
#   sudo ./deploy/upgrade.sh

# In alternativa: importa il file .qcow2/.ova come VM e avvia —
# credenziali iniziali e URL stampati in /etc/motd al primo boot.
```


© Dognet Technologies S.r.l. — P.IVA 04867480164

13. 12. Domande frequenti

13.1 Perché un target appare offline anche se l'agent è attivo?

Controlla che `dog-agent` sia effettivamente in esecuzione e correttamente configurato (`journalctl -u dog-agent -f`). Se il pairing è andato a buon fine ma il master mostra comunque "offline" più a lungo di un paio di minuti dopo un riavvio del backend, verifica di essere su FireDog v1.0.0 o successivo: le versioni precedenti potevano restare bloccate su "offline" dopo un'interruzione temporanea dell'heartbeat.

13.2 Perché `ulogd2` va in "failed" o si blocca dopo un restart o un reboot?

Bug corretto dalla v1.1.0: su Debian/Ubuntu `install.sh` installava `ulogd2` senza il sottopacchetto `ulogd2-pcap`, che contiene il plugin usato dallo stack PCAP di `ulogd.conf` di FireDog. Il demone resta "attivo" solo perché continua a girare con la vecchia configurazione finché non viene riavviato — appena succede (restart manuale o reboot), fallisce con "not even a single working plugin stack" nel log (`/var/log/ulogd/ulogd.log`): su Debian si blocca per circa 90 secondi prima di fallire, su Ubuntu esce subito con un errore. openSUSE/SLES non è mai stato affetto. Su un target già installato con una versione precedente, non serve reinstallare tutto:

```
sudo apt-get install -y ulogd2-pcap
sudo systemctl restart ulogd2
```

13.3 Posso installare gli strumenti firewall senza attivare subito la policy DROP?

Sì: usa `sudo bash get-firedog.sh --skip-init` . Attiverai il firewall in un secondo momento con `sudo firewall-init.sh && sudo systemctl enable --now firewall-fm` , quando avrai verificato `ALWAYS_OPEN_PORTS` in `firedog.conf` .

13.4 Come aggiungo una porta sempre aperta su un target già in produzione?

Modifica `ALWAYS_OPEN_PORTS` in `/etc/firewall/firedog.conf` e rilancia `sudo firewall-init.sh` : lo script ricrea l'intero ruleset, quindi pianifica una finestra di manutenzione — non basta modificare il file, serve il rilancio per applicare la modifica.

13.5 Come faccio a non restare bloccato fuori da SSH durante dei test, senza disattivare la protezione per tutti?

Aggiungi la tua rete di management/VPN a `TRUSTED_SSH_SOURCES` in `firedog.conf` (capitolo 7) e rilancia `sudo firewall-init.sh` . A differenza di `ALWAYS_OPEN_PORTS` — che per SSH non ha alcun effetto, perché quella porta è già intercettata dalla chain `SSH_PROTECT` prima — le sorgenti in `TRUSTED_SSH_SOURCES` bypassano interamente il rate-limit e il ban, mentre tutte le altre sorgenti restano protette come prima: nessuna regressione di sicurezza per il resto del traffico.

13.6 È possibile installare FireDog e CyberSheppard/SentinelCore sullo stesso host?

Sì: `dog-agent` è unico per tutta la suite. Basta aggiungere un blocco `[[targets]]` per ciascun prodotto nello stesso `agent.conf` .

13.7 Perché SSH (o altre connessioni legittime) vengono bloccate dopo pochi tentativi ravvicinati?

Nelle versioni precedenti la v1.0.1 le soglie di `SSH_PROTECT` e `SYN_FLOOD` erano troppo basse (4 tentativi/60s, 10 pacchetti SYN/s) e scattavano anche con un uso normale (tool di amministrazione, riconnessioni rapide), droppando la sorgente con "Connection reset by peer" o timeout. Dalla v1.0.1 i default sono `SSH_PROTECT_MAX_ATTEMPTS=20` e `SYN_FLOOD_LIMIT_PER_SEC=50` / `SYN_FLOOD_BURST=100` (quest'ultimo ora configurabile in `firedog.conf` come già era per SSH_PROTECT). Su un target già installato, aggiorna gli strumenti (sezione Aggiornamenti) o imposta i valori direttamente in `/etc/firewall/firedog.conf` e rilancia `firewall-init.sh` .

© Dognet Technologies S.r.l. — P.IVA 04867480164

13.8 Il browser dice "connessione non sicura": è normale?

Sì, finché il browser non conosce la CA locale dell'installazione. Importala da *Settings → API Keys Agent → Certificato del master* (dopo aver confrontato l'impronta) oppure accetta l'avviso. Vedi capitolo 1, *Certificato HTTPS e agent*.

13.9 Dopo l'aggiornamento il salvataggio delle impostazioni fallisce con "Connessione al master non riuscita"

Il certificato è cambiato mentre la pagina era aperta e il browser rifiuta le richieste senza mostrare l'avviso. Ricarica la pagina e accetta il nuovo certificato (o importa la CA).

13.10 L'agent non si collega più: nel log c'è `invalid peer certificate`

- `UnknownIssuer`: in `ca_cert` dell'agent manca la CA di questo master (o il master ha creato una CA nuova). Riscaricala e verifica l'impronta.
- `Other(OtherError(CaUsedAsEndEntity))`: il master usa ancora il vecchio certificato self-signed; aggiornalo.
- `NotValidForName`: `url` dell'agent usa un indirizzo diverso da quello del certificato.
- `Error: Target '...': endpoint wss:// senza [agent] ca_cert configurato`: da dog-agent 1.3 `ca_cert` è obbligatorio con `wss://`.

13.11 Ho salvato il threat-score in Settings → Monitoraggio ma l'agent usa ancora i vecchi valori

In *Settings* ci sono solo i default proposti: si inviano a un agent dalla pagina del singolo target (*Configurazione*). Le impostazioni pcap invece partono subito verso tutti i target collegati.

13.12 Non ricevo email/Slack per le minacce

Controlla i canali in *Settings → Notifiche* e la **Soglia invio alert** in *Monitoraggio*: partono solo le minacce *high*/*critical* con score almeno pari alla soglia × 10, una volta per minaccia (o quando si aggrava), e non più di una per tipo e target nel periodo di cooldown delle notifiche.

13.13 Un IP bloccato riesce ancora a collegarsi via SSH

Corretto in dog-agent 1.3.1: prima i blocchi erano aggiunti in fondo alla catena INPUT, dopo lo scarto finale di firewall-manager, e non fermavano il traffico che le regole precedenti accettavano (SSH sotto la soglia di `SSH_PROTECT`, porte sempre aperte, connessioni già aperte). Aggiorna l'agent: ora i blocchi vanno in cima alla catena.

© Dognet Technologies S.r.l. — P.IVA 04867480164