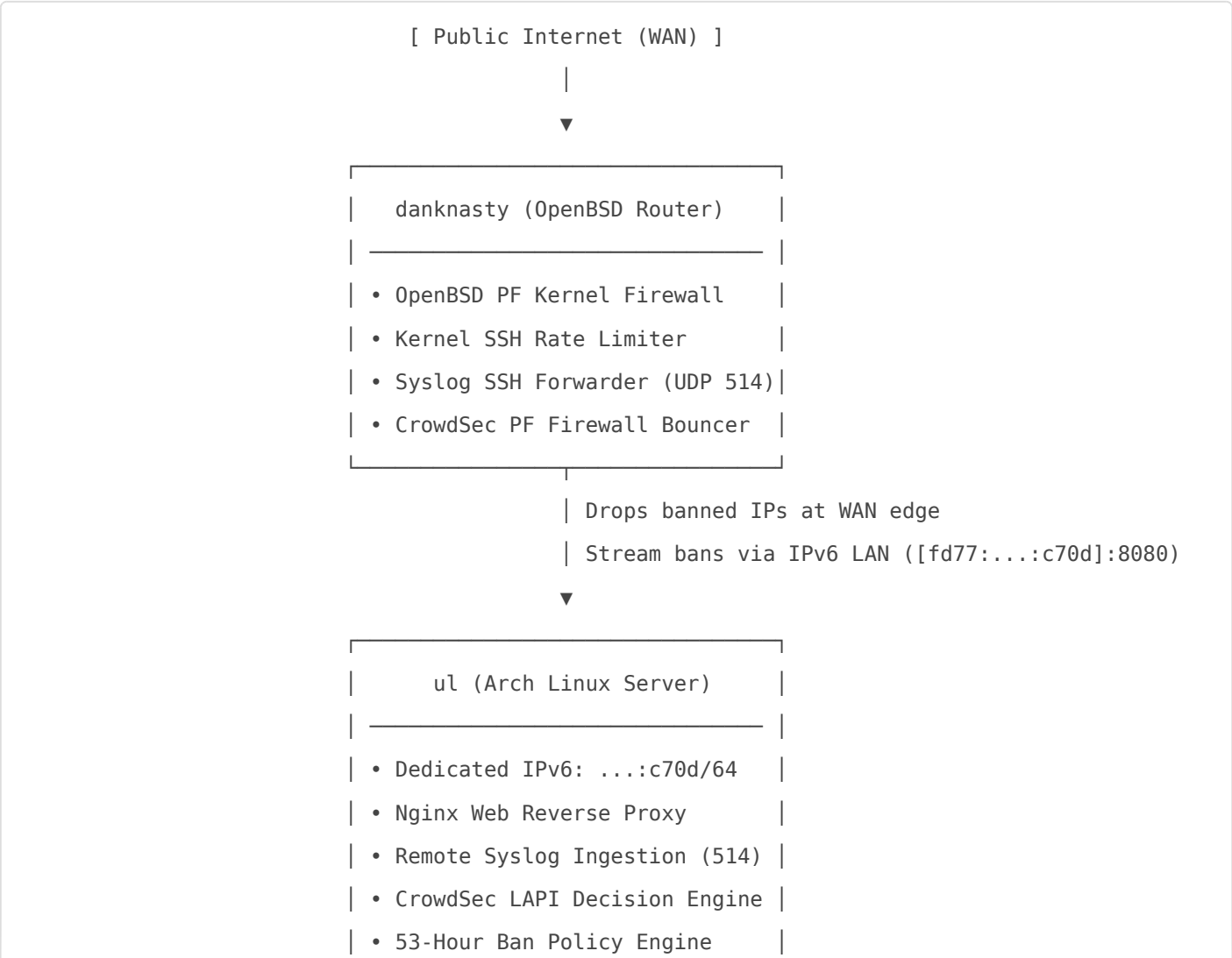


CrowdSec & OpenBSD PF

Perimeter Security Setup

1. Overview & Architectural Concept

This document outlines the distributed intrusion detection and perimeter enforcement setup across the local infrastructure (`drheck.dev`).



- Public Active Enriched JSON
- Append-Only NDJSON Stream

Design Philosophy & Conventions

1. Dedicated Service IPv6 Address (u1):

- Per local network convention (20-wired.network), CrowdSec LAPI listens on a dedicated IPv6 ULA: fd77:7777:5010:0:1337:f1ac:d00d:c70d on port 8080 (LAPI) and UDP port 514 (Syslog Ingestion).

2. Offload All Heavy Detection to Powerful Hardware (u1):

- All heavy log analysis, regex pattern matching, SQLite/LAPI database operations, and scenario evaluations run on the Arch Linux server (u1).
- **Nginx Web Probes & Direct IP Hits:** Parses Nginx access logs to detect scanner bots (e.g., .env harvesting, WordPress probes, webshell attempts, open proxy CONNECT attempts, and direct IP connection attempts).
- **Router SSH Probes, Invalid Users & Root Attempts:** Receives OpenBSD SSH authentication logs (/var/log/authlog) forwarded over UDP IPv6 syslog from danknasty to detect SSH brute-force attempts, instant invalid username bans, and instant root attempt bans.
- **53-Hour Ban Policy:** Issues a global **53-hour ban** for any detected web probe, direct IP scan, or SSH brute-force/invalid user/root attempt.

3. Perimeter Enforcement at Gateway (danknasty):

- The OpenBSD router (danknasty) runs only the lightweight CrowdSec PF Bouncer (crowdsec_firewall_bouncer) and native syslogd.
- Banned IPs are pulled from u1 and injected into OpenBSD PF's kernel <crowdsec-blacklists> table. Banned packets are dropped at the network interface before entering the local network.

4. Zero-Overhead Edge SSH Protection (danknasty):

- OpenBSD PF tracks TCP connection rates directly in kernel state space (max-src-conn-rate 3/30).
- Any bot opening more than 3 SSH connections in 30 seconds is added to the OpenBSD PF <bruteforce> table and dropped with **0% CPU and ~0 MB RAM usage**.

5. LAN Subnet & Dynamic IPv6 Whitelisting:

- Private subnets (10.0.0.0/24, fd77:7777:5010::/64), loopbacks (127.0.0.1/32, ::1/128), and the dynamic public ISP IPv6 prefix (auto-detected via /usr/local/bin/crowdsec-update-ipv6-whitelist) are whitelisted on u1 so internal LAN users and SSH typos can never trigger bans.

6. Rich Context & OSINT Enriched Export (u1):

- Every banned IP is automatically enriched with Reverse DNS (PTR), Country, Region, City, Coordinates (Lat/Lon), Timezone, ISP, Organization, and Autonomous System (ASN) details.
- **Active Array File:** /var/log/crowdsec-bans-public.json (chmod 644).

- **Append-Only NDJSON Stream:** `/var/log/crowdsec-bans.json` (`chmod 644`).

Detection Latency & Pipeline Timeline

Protection Layer	Target Threat	Latency	Mechanism / Action
Layer 1: OpenBSD PF Kernel Limiter	SSH Connection Spikes	Instant (Nanoseconds / ~0s)	In-kernel TCP state tracking (<code>max-src-conn-rate 3/30</code>) instantly moves abusive IPs into <code><bruteforce></code> table.
Layer 2: CrowdSec Distributed Ban	Web Probes, SSH Invalid Users & Root Attempts	~5 seconds (Max 10s)	Syslog/Nginx logs stream to <code>u1</code> . <code>u1</code> triggers 53h ban. <code>danknasty</code> bouncer pulls decision and injects IP into OpenBSD PF <code><crowdsec-blacklists></code> .

CrowdSec Distributed Pipeline Timeline:

1. **Event Generation:** `sshd-session` or Nginx emits log event (`< 1 ms`).
2. **Log Forwarding:** UDP IPv6 syslog or Nginx log tailer streams log to `u1` (`< 1 ms`).
3. **Scenario Trigger & Ban Creation:** CrowdSec parses line, leaks scenario bucket, and writes 53h ban to LAPI database (`~20-50 ms`).
4. **Router Bouncer Polling:** `danknasty` bouncer polls `u1` LAPI over IPv6 LAN (`update_frequency: 10s`, average `~5 seconds`).
5. **Kernel Table Injection:** Bouncer executes `pfctl -t crowdsec-blacklists -T add <IP>` (`~10 ms`).

2. Implementation & Configuration Details

A. Central Decision Engine on `u1`

- **Interface Configuration** (`/etc/systemd/network/20-wired.network`):

```
# crowdsec
Address=fd77:7777:5010::1337:f1ac:d00d:c70d/64
```

- **Listen URI** (`/etc/crowdsec/config.yaml`): Bound to dedicated IPv6 ULA

```
[fd77:7777:5010:0:1337:f1ac:d00d:c70d]:8080:
```

```
api:
  server:
    listen_uri: "[fd77:7777:5010:0:1337:flac:d00d:c70d]:8080"
```

- **Nginx Log Acquisition (/etc/crowdsec/acquis.d/nginx.yaml):**

```
filenames:
  - /var/log/nginx/*.log
  - /var/log/nginx/access.log
labels:
  type: nginx
```

- **Router SSH Syslog Ingestion (/etc/crowdsec/acquis.d/danknasty-ssh.yaml):**

```
source: syslog
listen_addr: "fd77:7777:5010:0:1337:flac:d00d:c70d"
listen_port: 514
labels:
  type: syslog
```

- **LAN & Dynamic IPv6 Whitelist Override (/etc/crowdsec/parsers/s02-override/00-lan-whitelist.yaml):**

```
name: custom/lan-whitelist
description: "Whitelist internal LAN subnets and dynamic public IPv6 prefix"
filter: "evt.Meta.source_ip != nil"
whitelist:
  reason: "Exempt internal LAN subnets and dynamic public IPv6 prefix"
  cidr:
    - "10.0.0.0/24"
    - "fd77:7777:5010::/64"
    - "127.0.0.1/32"
    - "::1/128"
    - "2603:3001:626:9d80::/64" # Auto-updated by crowdsec-update-ipv6-whitelist
```

- **Custom Honeypot Scenario (/etc/crowdsec/scenarios/honeypot-endpoints.yaml):**

```
type: leaky
name: custom/honeypot-endpoints
description: "Instant trigger on bot/scanner endpoints"
filter: |
  evt.Parsed.request in ["/.env", "/cgi-bin/authLogin.cgi", "/IndoSec.php",
"/alfa.php", "/luxex.php", "/BDKR28WP.php"] || evt.Parsed.request matches "(?i)^(wp-
```

```
admin|wp-login|wp-includes|\\.env|cgi-  
bin|alfa\\.php|1\\.php|222\\.php|8\\.php|shell\\.php)"  
capacity: 1  
leakspeed: 1h  
groupby: evt.Meta.source_ip  
blackhole: 1m  
labels:  
  service: nginx  
  type: honeypot_scan  
  remediation: true
```

- **Custom SSH Invalid User Scenario (`/etc/crowdsec/scenarios/ssh-invalid-user.yaml`):**
Instant 53-hour ban for any attempt to log in using an invalid username:

```
type: leaky  
name: custom/ssh-invalid-user  
description: "Instant 53h ban for any SSH login attempt using an invalid user"  
filter: |  
  evt.Meta.log_type == "ssh_failed-auth" && (  
    evt.Parsed.sshd_invalid_user != "" ||  
    evt.Meta.target_user != ""  
  )  
capacity: 1  
leakspeed: 1h  
groupby: evt.Meta.source_ip  
blackhole: 1m  
labels:  
  service: ssh  
  type: bruteforce  
  remediation: true
```

- **Custom SSH Root Attempt Scenario (`/etc/crowdsec/scenarios/ssh-root-login-attempt.yaml`):** Instant 53-hour ban for any attempt targeting the `root` account:

```
type: leaky  
name: custom/ssh-root-login-attempt  
description: "Instant 53h ban for any SSH login attempt targeting user root"  
filter: |  
  evt.Meta.log_type == "ssh_failed-auth" && (  
    evt.Meta.target_user == "root" ||  
    evt.Parsed.sshd_invalid_user == "root" ||
```

```

    evt.Parsed.sshd_user == "root"
  )
capacity: 1
leakspeed: 1h
groupby: evt.Meta.source_ip
blackhole: 1m
labels:
  service: ssh
  type: bruteforce
  remediation: true

```

- **Direct IP Access Scenario (`/etc/crowdsec/scenarios/direct-ip-access.yaml`):** Instant 53-hour ban for any HTTP/HTTPS connection made directly by IP address or invalid host header:

```

type: leaky
name: custom/direct-ip-access
description: "Instant 53h ban for any HTTP/HTTPS connection made directly to IP
address or non-domain host"
filter: |
  evt.Meta.log_type == "http_access-log" && (
    evt.Meta.http_status in ["444", "400", "403"] ||
    evt.Parsed.status in ["444", "400", "403"] ||
    evt.Parsed.target_fqdn matches "^[0-9]{1,3}\.){3}[0-9]{1,3}$" ||
    evt.Parsed.http_host matches "^[0-9]{1,3}\.){3}[0-9]{1,3}$"
  )
capacity: 1
leakspeed: 1h
groupby: evt.Meta.source_ip
blackhole: 1m
labels:
  service: nginx
  type: direct_ip_scan
  remediation: true

```

- **53-Hour Ban Policy (`/etc/crowdsec/profiles.yaml`):**

```

name: default_ip_remediation
filters:
  - Alert.Remediation == true && Alert.GetScope() == "Ip"
decisions:

```

```

- type: ban
  duration: 53h
notifications:
- file_default
on_success: break
---
name: default_range_remediation
filters:
- Alert.Remediation == true && Alert.GetScope() == "Range"
decisions:
- type: ban
  duration: 53h
notifications:
- file_default
on_success: break

```

- **File Notification Plugin (`/etc/crowdsec/notifications/file.yaml`):** Appends real-time events upon trigger to `/var/log/crowdsec-notifications.log`:

```

type: file
name: file_default
log_level: info
group_wait: 0s
group_threshold: 1
format: |
  {{range . -}}
  {{$alert := . -}}
  {{range .Decisions -}}

  {"timestamp": "{{$.CreatedAt}}", "ip": "{{$.Value}}", "action": "{{$.Action}}", "duration": "{{$.Duration}}", "reason": "{{$.Scenario}}", "country": "{{$.Source.Cn}}", "as_name": "{{$.Source.AsName}}"}

  {{end -}}
  {{end -}}
log_path: "/var/log/crowdsec-notifications.log"
rotate:
  enabled: false

```

- **Host Firewall Rules (`/etc/nftables.conf`):**

```
ip6 saddr fd77:7777:5010::/64 ip6 daddr fd77:7777:5010:0:1337:f1ac:d00d:c70d tcp
dport 8080 accept comment "Allow CrowdSec LAPI from local IPv6 LAN"

ip6 saddr fd77:7777:5010::/64 ip6 daddr fd77:7777:5010:0:1337:f1ac:d00d:c70d udp
dport 514 accept comment "Allow CrowdSec Syslog Ingestion from local IPv6 LAN"
```

- **Automated OSINT & Threat Pipeline** (`/usr/local/bin/crowdsec-export-bans`): Runs every 60 seconds (`crowdsec-export-bans.timer`):
 1. Updates dynamic public IPv6 whitelist (`/usr/local/bin/crowdsec-update-ipv6-whitelist`).
 2. Ingests all router SSH auth failures in real time via native UDP IPv6 syslog (`auth.info;authpriv.info` -> port 514), triggering instant sub-second bans (`custom/ssh-invalid-user` via `type: trigger`).
 3. Updates `/var/log/crowdsec-bans-public.json` (active JSON array) and appends newly discovered ban events with full GeoIP/ASN/rDNS metadata to `/var/log/crowdsec-bans.json` (`chmod 644`).

B. Perimeter Enforcement & Syslog Forwarding on danknasty (Router)

- **Syslog Forwarding Configuration** (`/etc/syslog.conf`):

```
auth.info;authpriv.info@@@@@[fd77:7777:5010:0:1337:f1ac:d00d:c70d]:514
```

Note: OpenBSD `syslogd(8)` requires the `:514` port suffix when enclosing IPv6 addresses in square brackets `@[...]`. Without `:514`, `syslogd` silently skips UDP forwarding.

- **Bouncer Configuration** (`/etc/crowdsec-firewall-bouncer/crowdsec-firewall-bouncer.yaml`):

```
mode: pf
retry_initial_connect: true
update_frequency: 10s
api_url: http://[fd77:7777:5010:0:1337:f1ac:d00d:c70d]:8080/
api_key: <hunter2>
deny_action: DROP
blacklists_ipv4: crowdsec-blacklists
blacklists_ipv6: crowdsec6-blacklists
pf:
  anchor_name: ""
```

- **Router Firewall Rules** (`/etc/pf.conf`):

```
# CrowdSec Blacklist Tables
table <crowdsec-blacklists> persist
table <crowdsec6-blacklists> persist
block drop quick from <crowdsec-blacklists>
block drop quick from <crowdsec6-blacklists>

# Table for SSH Brute-force Auto-Blocking
table <bruteforce> persist
block drop quick from <bruteforce>

# SSH rules with stateful connection tracking
pass in quick on $ext proto tcp to self port ssh \
    flags S/SA keep state \
    (max-src-conn 10, max-src-conn-rate 3/30, overload <bruteforce> flush global)
```

3. Interaction, Operations & Data Format

A. Enriched Public JSON Data Schema

The exported JSON files (`/var/log/crowdsec-bans-public.json` and `/var/log/crowdsec-bans.json`) contain rich OSINT, network, and threat context for each banned IP:

```
{
  "decision_id": 4,
  "alert_id": 4,
  "ip": "119.30.119.3",
  "ip_version": "IPv4",
  "origin": "crowdsec",
  "reason": "crowdsecurity/netgear_rce",
  "action": "ban",
  "duration": "52h59m18s",
  "country": "Pakistan",
  "country_code": "PK",
```

```
"region": "Punjab",
"city": "Sialkot",
"latitude": 32.426,
"longitude": 74.3644,
"timezone": "Asia/Karachi",
"isp": "Mobilink GSM, Pakistan Mobile Communication Ltd.",
"org": "Linkdotnet Telecom Limited",
"as_name": "AS58470 IX Peering for Mobilink and Link Direct International.",
"rdns": "static-host119-30-119-3.link.net.pk",
"target_uri": "/setup.cgi?next_file=netgear.cfg&todo=syscmd&cmd=rm+-rf+/tmp/*...",
"user_agent": "-",
"method": "GET",
"status_code": "301",
"created_at": "2026-08-05T02:44:21Z"
}
```

B. Viewing Active System State & OpenBSD PF Kernel Tables

1. View Active Bans on `ul` (CrowdSec LAPI)

```
# View all active decisions (untruncated):
sudo cscli decisions list -a

# Query if a specific IP is banned in CrowdSec:
sudo cscli decisions list --ip <IP_ADDRESS>
```

2. View Active Enriched JSON Array

```
cat /var/log/crowdsec-bans-public.json
```

3. Tail the Append-Only Ban Stream (NDJSON)

```
tail -f /var/log/crowdsec-bans.json
```

4. View Active Bouncers & Connection Status

```
sudo cscli bouncers list
```

5. View Detection Metrics & Statistics

Shows lines read via Nginx logs and remote router syslog:

```
sudo cscli metrics
```

6. View Active OpenBSD PF Kernel Tables on Router (danknasty)

```
# List all blocked IPs in CrowdSec's PF Radix Tree table:
SSH_AUTH_SOCKET=/home/luke/.local/ssh_auth_socket/ssh_auth_socket ssh danknasty "doas pfctl -t crowdsec-blacklists -T show"

# View packet & byte drop statistics for the CrowdSec PF table:
SSH_AUTH_SOCKET=/home/luke/.local/ssh_auth_socket/ssh_auth_socket ssh danknasty "doas pfctl -v -t crowdsec-blacklists -T show"

# Test if a specific IP is currently blocked in the OpenBSD PF kernel table:
SSH_AUTH_SOCKET=/home/luke/.local/ssh_auth_socket/ssh_auth_socket ssh danknasty "doas pfctl -t crowdsec-blacklists -T test <IP_ADDRESS>"

# View OpenBSD SSH kernel brute-force bans:
SSH_AUTH_SOCKET=/home/luke/.local/ssh_auth_socket/ssh_auth_socket ssh danknasty "doas pfctl -t bruteforce -T show"
```

C. Manual Interactivity & Ban Management

1. Manually Ban an IP for 53 Hours

```
sudo cscli decisions add --ip <IP_ADDRESS> --duration 53h --reason "Manual administrative ban"
```

2. Remove a Ban (Unban an IP)

```
sudo cscli decisions delete --ip <IP_ADDRESS>
```

3. Force Immediate Synchronization & OSINT Enrichment

```
sudo /usr/local/bin/crowdsec-export-bans
```

4. Inspect Recent Detection Alerts

```
sudo cscli alerts list
sudo cscli alerts inspect <ALERT_ID>
```

5. Flush Banned IP Tables on the Router (Emergency Reset)

```
SSH_AUTH_SOCK=/home/luke/.local/ssh_auth_sock/ssh_auth_sock ssh danknasty "doas pfctl -t crowdsec-blacklists -T flush"
```

D. Whitelisting Local Trusted Networks

Internal LAN subnets and your dynamic public IPv6 prefix are protected from accidental bans via

```
/etc/crowdsec/parsers/s02-override/00-lan-whitelist.yaml:
```

```
name: custom/lan-whitelist
description: "Whitelist internal LAN subnets from all CrowdSec scenarios"
filter: "evt.Meta.source_ip != nil"
whitelist:
  reason: "Exempt internal LAN subnets"
  cidr:
    - "10.0.0.0/24"
    - "fd77:7777:5010::/64"
    - "127.0.0.1/32"
    - "::1/128"
    - "2603:3001:626:9d80::/64"
```

After modifying whitelists or scenarios, reload CrowdSec:

```
sudo systemctl reload crowdsec
```

E. Adding More Scanner Endpoints

```
/etc/crowdsec/scenarios/honeypot-endpoints.yaml
```

Current Configuration:

```
type: leaky
name: custom/honeypot-endpoints
description: "Instant trigger on bot/scanner endpoints"
filter: |
    evt.Parsed.request in ["/.env", "/cgi-bin/authLogin.cgi", "/IndoSec.php", "/alfa.php",
"/luxex.php", "/BDKR28WP.php"] || evt.Parsed.request matches "(?i)^(wp-admin|wp-login|wp-
includes|\\.env|cgi-bin|alfa\\.php|1\\.php|222\\.php|8\\.php|shell\\.php)"
capacity: 1
leakspeed: 1h
groupby: evt.Meta.source_ip
blackhole: 1m
labels:
    service: nginx
    type: honeypot_scan
    remediation: true
```

How to Add More Scanner Endpoints

1. Add new exact paths to the evt
2. Parsed.request in [...] list (e.g. "/setup.php", "/config.json"), or append regex patterns to evt.Parsed.request matches "(?i)...".
3. Reload Crowd

```
$ sudo cscli scenarios reload
$ sudo systemctl reload crowdsec
```

Revision #7

Created 2026-08-05 01:57:39 UTC by admin

Updated 2026-09-25 13:24:57 UTC by admin