Set Up a CrowdSec Using OPNsense LAPI on Caddy

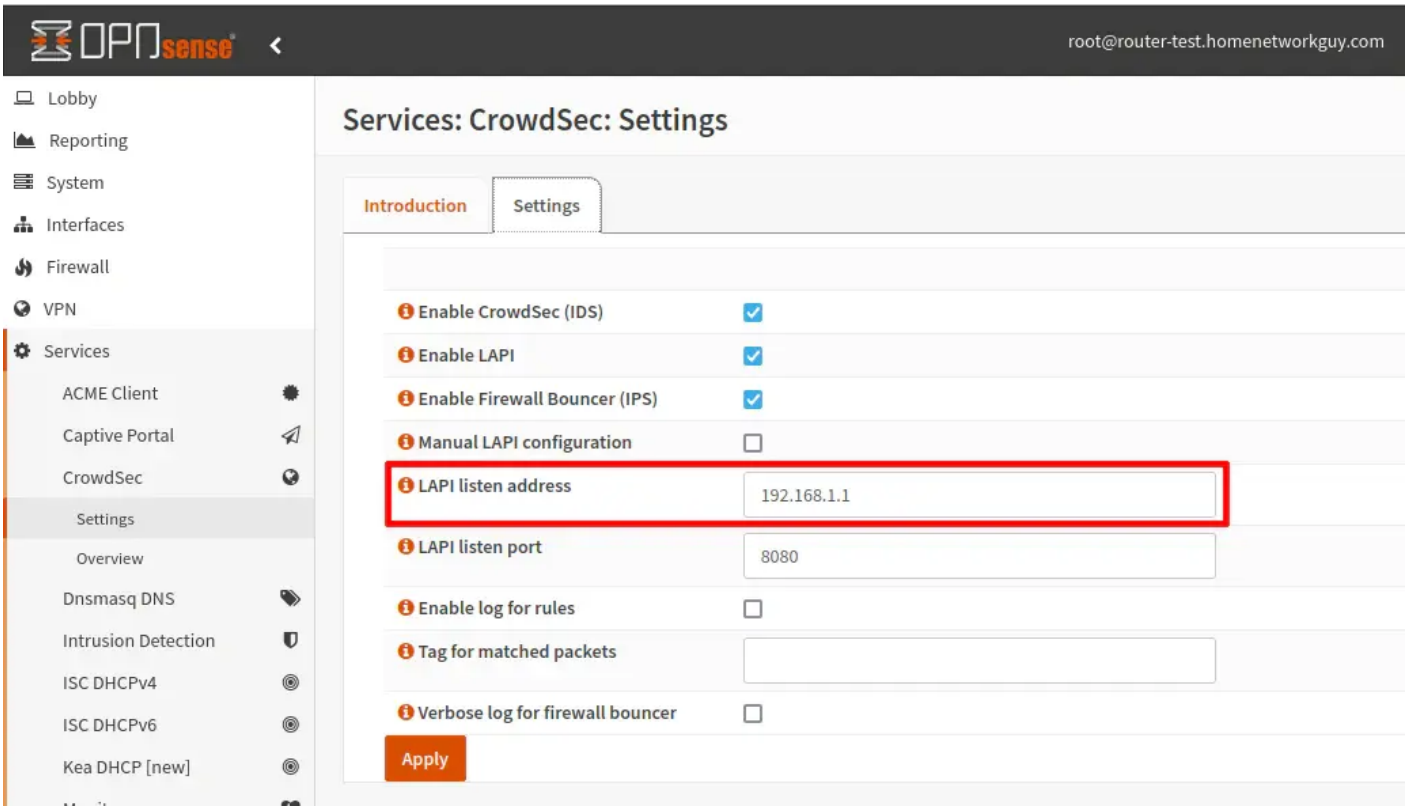
Prepare the OPNsense CrowdSec Configuration

Before setting up the Caddy reverse proxy, some settings for CrowdSec and firewall rules can be configured in OPNsense to prepare for a CrowdSec multi-server environment.

Update the Existing CrowdSec Plugin Configuration

The first thing you can do is change your CrowdSec plugin settings in OPNsense to allow other CrowdSec agents/bouncers to use the LAPI (Local API) on OPNsense. By default, the LAPI on OPNsense only listens on `localhost` (`127.0.0.1`).

In this example, I am setting the IP address to be on the LAN interface of `192.168.1.1`. You may wish to put it on a different interface.



You will need to start/stop the CrowdSec plugin for changes to take effect.

Create API Key for Caddy CrowdSec Bouncer

To prepare for setting up the CrowdSec bouncer for Caddy in a later step, you will need an API key generated for the bouncer.

Log into your OPNsense system via SSH or the console and issue the following command to create an API key. You may use any name you wish in place of `caddyDmz`. I used that name since this will be for a Caddy instance on the DMZ network.

```
sudo cscli bouncers add caddy
```

You should see the API key in the console output. Copy/paste this key until it is needed later.

```
API key for 'caddy':

   xxxxxxxxxxxxxxxxxxxxxx

Please keep this key since you will not be able to retrieve it!
```

Build Caddy with the Desired Modules

In order to use DNS challenges with Let's Encrypt and to use a CrowdSec bouncer, you will need to a build a custom Caddy executable to extend the base functionality. Fortunately, the build process is easy with `xcaddy` since you can build a Caddy executable with a single command.

I will be using the Cloudflare module for Let's Encrypt but you may use a different provider.

Replace `github.com/caddy-dns/cloudflare` with a provider from the list found on GitHub.

```
xcaddy build \
    --with github.com/caddy-dns/cloudflare \
    --with github.com/hslatman/caddy-crowdsec-bouncer/crowdsec
```

Delete the old file and move the file to the `/usr/bin/` folder:

```
sudo rm /usr/bin/caddy
sudo mv caddy /usr/bin/
```

You should be able to run the `caddy` executable to ensure it can be found on the path.

```
caddy version
```

Modify Caddyfile

In the configuration file, you will need to enter the Cloudflare API key used for editing DNS zones for the `acme_dns cloudflare` option.

In addition to the DNS API key, newer versions of Caddy (v2.8.0+) require you to enter an email address for ZeroSSL (Caddy uses both Let's Encrypt and ZeroSSL for issuing certificates).

Then in a `crowdsec` block, you will need to enter the API key what was generated from OPNsense earlier. The URL is for the CrowdSec LAPI on OPNsense which is `192.168.1.1:8080`.

These first two settings should be contained in the global settings block as shown below.

```
{
        email nicolas.lespinasse@vainsta.fr
        acme_dns cloudflare xxxxxxxxxxxxxxxxxxxxxxx
        admin :2019
        metrics
        crowdsec {
```

```
                api_key xxxxxxxxxxxxxxx
                api_url http://192.168.1.1:8080/
        }
}

panel.vainsta.fr {
        log {
                output file /var/log/caddy/panel-access.log {
                        roll_size 100mb
                        roll_keep 20
                        roll_keep_for 720h
                }
                format json
                level INFO
        }
        reverse_proxy https://192.168.1.4:444 {
                transport http {
                        tls_insecure_skip_verify
                }
        }
}
```

There are two additional CrowdSec bouncer options you may include.

If you wish for the bouncer to check the LAPI each time instead of caching the decisions in memory and polling the LAPI every 10 seconds by default, you can disable streaming by adding the `disable_streaming` option to the `crowdsec` block. Streaming decision information is more efficient if you have a lot of requests, but it is possible there will be a slight increase in delay when decisions on the LAPI have changed.

```
...
        crowdsec {
            ...
            disable_streaming
...
```

There is an option to enable “hard fails” if there is an issue connecting to the CrowdSec LAPI. This feature might be nice if you wish to prevent your services from being accessed if something is wrong with the LAPI since they will be unprotected by CrowdSec. Of course, that would negatively affect uptime, but it would make it very apparent something bad has happened.

I noticed it takes about 30 seconds to fail after the CrowdSec service is stopped in OPNsense. I was starting to wonder if this option was working properly, but I simply was not patient enough during my testing.

```
...
        crowdsec {
            ...
            enable_hard_fails
...
```

Press “Ctrl + O”, “Enter”, and “Ctrl + X” to save and close the `Caddyfile`.

Install CrowdSec Agent on Caddy Server

The CrowdSec bouncer is what will block connections to services behind the reverse proxy, but the Caddy server will need a CrowdSec agent installed so it can run the parsers and scenarios on the server. The agent sends the information to the LAPI on OPNsense to make decisions on blocking content. For a single CrowdSec instance, this usually occurs on the same system, but in a multi-server configuration, the CrowdSec agent and bouncer communicates with the LAPI on a different server (in this case, OPNsense).

Install CrowdSec using the following commands. Basically the next step is following the CrowdSec installation guide. It is very simple to install.

```
curl -s https://packagecloud.io/install/repositories/crowdsec/crowdsec/script.deb.sh | sudo bash
sudo apt install crowdsec
```

The CrowdSec agent needs to be registered with OPNsense CrowdSec LAPI.

```
sudo cscli lapi register -u http://192.168.1.1:8080
```

Copy the default CrowdSec `systemd` file from `/lib/systemd/system` to `/etc/systemd/system` so customizations can be made to the service file.

```
sudo cp /lib/systemd/system/crowdsec.service /etc/systemd/system/crowdsec.service
```

Edit the `/etc/systemd/system/crowdsec.service` to add the `-no-api` flag to the end of the `ExecStart` command. This disables the LAPI on the Caddy server since it is not needed because the LAPI on OPNsense will be used instead (see CrowdSec’s multi-server configuration example).

```
[Unit]
Description=Crowdsec agent
After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]
Type=notify
Environment=LC_ALL=C LANG=C
ExecStartPre=/usr/bin/crowdsec -c /etc/crowdsec/config.yaml -t -error
ExecStart=/usr/bin/crowdsec -c /etc/crowdsec/config.yaml -no-api
#ExecStartPost=/bin/sleep 0.1
ExecReload=/bin/kill -HUP $MAINPID
Restart=always
RestartSec=60

[Install]
WantedBy=multi-user.target
```

You will need to reload the `systemd` service since changes to the service was made.

```
sudo systemctl daemon-reload
```

Do not reload the CrowdSec service just yet because with the LAPI on the Caddy server disabled, CrowdSec will error on startup until you have validated the Caddy machine on OPNsense. CrowdSec will be unable to connect to the LAPI on OPNsense until validation occurs. When there is no reachable LAPI, the CrowdSec agent will fail to load.

Validate the Caddy Machine in OPNsense

While logged into OPNsense via SSH or the console, list the machines which have been registered or requesting to be registered:

```
sudo cscli machines list
```

You should see similar output to below. Notice the status of the machine `02a3nfadce4ez4b19zh582e0f68f72a4CX4EzFJ2Th4PNkj1` shows the “No” symbol under “Status”.

```
────────────────────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
 Name                                                IP Address       Last Update
Status   Version                    Auth Type   Last Heartbeat
────────────────────────────────────────────────────────────────────────────────────────────────
```

```
──────────────────────────────────────────────────────────────────────

 localhost                                          192.168.1.1      2024-03-11T14:11:50Z   ✔
        v1.6.0-freebsd-4b8e6cd7   password    21s
 02a3nfadce4ez4b19zh582e0f68f72a4CX4EzFJ2Th4PNkj1   192.168.1.2    2024-03-11T13:55:29Z
                                   password    ⚠ 16m42s
──────────────────────────────────────────────────────────────────────────────────────────────

──────────────────────────────────────────────────────────────────────
```

To validate the machine, you can enter the following command.

```
sudo cscli machines validate 02a3nfadce4ez4b19zh582e0f68f72a4CX4EzFJ2Th4PNkj1
```

Add Collection(s) to CrowdSec Agent on the Caddy Server

To add extra Caddy-specific parsers, you can add the following collection to your CrowdSec installation on your Caddy server. The Caddy collection includes a Caddy log parser and basic HTTP protections.

While the Caddy log parser may only be beneficial if you are using Caddy as a web server instead of a reverse proxy, the included basic HTTP protections should be helpful to protect web apps that are behind the reverse proxy.

```
sudo cscli collections install crowdsecurity/caddy
```

If you are hosting a public service (which great care must be taken to address security), it may not be a bad idea to also include the HTTP DoS collection to help detect denial of service attacks.

Of course, you should test this does not interfere with the normal operation of your app/service. I also do not know if this would be beneficial if you are using a Cloudflare proxy which includes DDoS protection.

```
sudo cscli collections install crowdsecurity/http-dos
```

You may now finally restart the CrowdSec agent on the Caddy server.

```
sudo systemctl reload crowdsec
```

After reloading CrowdSec, you should check if it is running properly.

```
sudo systemctl status crowdsec
```

Created 2025-03-23 21:35:37 UTC by Nicolas
Updated 2025-03-24 00:24:38 UTC by Nicolas