Cloudflare Plugin (Allow only CF-Traffic to your Server)

Overview

This plugin ensures that incoming requests must originate from Cloudflare’s network (or other CIDRs that you explicitly allow). It is particularly useful when you only want Cloudflare-proxied traffic to reach your services. By using Cloudflare’s IP ranges, the plugin can block all other sources of traffic and help enhance security.

Plugin Page

Requirements

- A working Traefik setup (for instance, Traefik Reverse Proxy for Docker Swarm).
- A valid **DNS-01 Challenge** configuration with Cloudflare to manage certificates (see Docker Compose with Let's Encrypt DNS Challenge).

> With these prerequisites in place, you can integrate the Cloudflare plugin to filter and rewrite traffic so that only Cloudflare IP ranges can access your services through Traefik.

Features

- □□ Only allow traffic originating from Cloudflare IP v4 and v6
- □□ Custom CIDRs list can be added to allow requests not from Cloudflare
- ♻ Refresh Cloudflare CIDRs from the Cloudflare API
- ⚙ Handle `X-Forwarded-For` original header to allow Cloudflare requests from a trusted reverse proxy behind Traefik
- □□ Rewrite requests `X-Forwarded-For` header with the user IP provided by `CF-Connecting-IP`
- □□ Rewrite requests `X-Forwarded-Proto` header with the scheme provided by `CF-Visitor`
- □□ Rewrite requests `X-Real-IP` header with the user IP provided by `CF-Connecting-IP`
- □□ Rewrite `RemoteAddress` to permit Traefik *ipwhitelist* middleware to work on IP provided by `CF-Connecting-IP`

Configuration

Plugin Options

Key	Type	Default	Description
`trustedCIDRs`	`[]string`	`[]`	Requests coming from a source not matching any of these CIDRs will be terminated with a 403. If empty, it is populated with Cloudflare's CIDRs.
`allowedCIDRs`	`[]string`	`[]`	Requests coming from a source matching any of these CIDRs will not be terminated with a 403 and no overwrite of request header append.
`refreshInterval`	`time.Duration`	`24h`	When `trustedCIDRs` is empty, Cloudflare's CIDRs will be refreshed after this duration. Using a value of 0 seconds disables the refresh.
`overwriteRequestHeader`	`bool`	`true`	When `true`, the request's header is rewritten. When `false`, any header or Traefik `RemoteAddress` is modified, filtering only the request from Cloudflare IP.
`appendXForwardedFor`	`bool`	`false`	Works only when `overwriteRequestHeader` is `true`. When `true`, prepend Cloudflare IP to `X-Forwarded-For` instead of replacing the first value.
`debug`	`bool`	`false`	Output debug messages in Traefik logs.

Traefik Static Configuration

```
experimental:
  plugins:
    cloudflare:
      moduleName = "github.com/agence-gaya/traefik-plugin-cloudflare"
      version = "v1.2.0"
```

Where to Add This?

In your *static configuration* file (for instance, `static.yaml` or `static.toml`), add the above lines under the `experimental` section to enable the plugin. Ensure Traefik is restarted or reloaded to pick up these changes.

Dynamic Configuration

To configure and instantiate the plugin, you need a *dynamic configuration* file as well. This can be in `YAML`, `TOML`, or another format supported by Traefik.

```
http:
  middlewares:
    cloudflare:
      plugin:
        cloudflare:
          trustedCIDRs: []
          overwriteRequestHeader: true


  routers:
    foo-router:
      rule: Path(`/foo`)
      service: foo-service
      entryPoints:
        - web
      middlewares:
        - cloudflare
```

Step-by-Step: Creating a `http.plugins.yaml` File

1. **Create a dedicated file** at `/mnt/glustermount/data/traefik_data/dynamic/http.plugins.yaml` (or a suitable location).

2. **Paste the dynamic configuration** for the plugin into this file. For example:

```
http:
  middlewares:
    cloudflare:
      plugin:
        cloudflare:
          trustedCIDRs: []
          allowedCIDRs: []
          refreshInterval: 24h
          overwriteRequestHeader: true
          appendXForwardedFor: false
          debug: false

  routers:
    foo-router:
      rule: Path(`/foo`)
      service: foo-service
      entryPoints:
        - web
      middlewares:
        - cloudflare

  services:
    foo-service:
      loadBalancer:
        servers:
          - url: "http://127.0.0.1:8080"
```

3. **Reference this dynamic file** in your Traefik *static configuration* (e.g., `--providers.file.filename=/mnt/glustermount/data/traefik_data/dynamic/http.plugins.yaml`).
4. **Restart or reload** Traefik to apply these changes.

Verifying the Plugin Works

To ensure that the plugin is blocking traffic not coming from Cloudflare, you can attempt a **cURL** request from a source IP that is *not* one of Cloudflare's documented IP ranges:

```
curl -kH "Host: test.aeoneros.com" https://<public-ip>
```

You should receive a 403 (Forbidden) response if your IP is not allowed. If you route through Cloudflare, the request should pass normally.

Middleware Plugins in Traefik

Once loaded, these plugins behave like statically compiled middlewares. Their instantiation and behavior are driven by the dynamic configuration. For example, you can add multiple plugins in the **experimental** section of your static config, and then configure them in your dynamic config similarly to built-in middlewares.

Visit the Traefik Plugin Catalog for more community-contributed plugins.

Revision #6
Created 4 February 2025 14:46:41 by aeoneros
Updated 11 February 2025 10:19:27 by aeoneros