Configuring Fail2ban with Traefik

https://blog.lrvt.de/configuring-fail2ban-with-traefik/

Traefik gained popularity as reverse proxy amongst selfhosters, homelabbers and enterprise organizations. It's unique feature of auto-detecting containerized services and the ability to automatically expose them is apparently in hot demand. Especially for power users of Kubernetes, Docker Swarm and AWS ECS.

Combined with the **Infrastructure as Code** approach, Traefik does not need a database and can be configured by configuration files or labels only. This syncs perfectly with containerized services and DevOPS (e.g. CI/CD).

Since the access logs of Traefik are structured in a normalized format, Fail2ban can easily be used to monitor those logs for malicious activities such as forceful browsing or brute-forcing. In this blog post, I'll guide you through implementing this.

Prerequisites

For this blog post, we will focus on implementing Fail2ban in conjunction with Traefik as reverse proxy behind Cloudflare. I will not go into detail on how to run and configure Traefik, set up port forwardings on your router or define the necessary DNS entries on Cloudflare to get you started. I assume that these things are already running and working for you.

This blog post will only focus on installing and configuring a dockerized Failban container to work with your Traefik reverse proxy logs. If you are not using Cloudflare, don't worry. You can follow this blog post and neglect the relevant things for Cloudflare just fine. Your main focus will then be on banning threat actors using iptables only.

The only important Traefik configurations would be:

1. Define Cloudflare's public IPv4 and IPv6 ranges as trusted IPs to obtain the real IP address of your site visitors. This is only necessary, if your Traefik instance runs behind Cloudflare.
2. Enable Traefik to store access logs and also allow logging of the HTTP user agent string of your site visitors, which is disabled by default. I recommend using the non-default JSON log format, since it contains a lot more details and can be more easily parsed (e.g. by

Grafana Loki). Finally, I recommend disabling asynchronous log pushing (buffering) so that fail2ban obtains logs instantly.

You can find an example setup of Traefik with all relevant configuration files at the following blog post or here on GitHub.

Please adjust your `traefik.yml` static configuration file to enable access logs in the non-default JSON format. Here is an excerpt of an example configuration:

```
accessLog:
  filePath: "/logs/traefik.log"
  format: json
  #filters:
  #  statusCodes:
  #    - "200"
  #    - "400-599"
    #retryAttempts: true
    #minDuration: "10ms"
  # collect logs as in-memory buffer before writing into log file
  bufferingSize: 0
  fields:
    headers:
      defaultMode: drop # drop all headers per default
      names:
          User-Agent: keep # log user agent strings
```

Excerpt of the static traefik.yml configuration file to enable access logs with user agents

Note that Traefik logs can increase in size really fast. As Traefik itself does not provide log rotation, it's your task to rotate them. I've added an example script to rotate Traefik logs. You can find it here.

Detection Capabilities

Our goal will be to identify and block threat actors that actively conduct forceful browsing and brute-forcing attacks on our web services. Forceful browsing is the act of enumerating a web service for sensitive files, interesting directories and potentially vulnerable endpoints in an automated manner. Moreover, some threat actors may also try brute-forcing login areas or Basic Authentication prompts.

Since all these potential hacking attempts will generate log entries with HTTP errors like 401, 403, 404 etc. in a very short time period, we can easily identify such attacks and ban the threat actor's IP address via Fail2ban locally and globally on Cloudflare. Fail2ban will monitor all proxy host logs of our Traefik reverse proxy. Therefore, once a misbehaving threat actor is detected and banned by Fail2ban, the attacker won't be able to access any of our other web services subsequently. This applies to all proxy hosts of Traefik. Basically a universal ban as soon as one hacking attempt is detected on one of our HTTP web services.

Running Fail2ban with Docker Compose

Fail2ban can be run and installed on bare metal, but there also exists a containerized version. For this blog post, we will focus on using a Docker container provided by crazymax.

You can use the following docker-compose file to spawn a dockerized Fail2ban container. Please adjust your volume mappings and environment variables if needed.

⚡
Ensure that you properly bind mount the logs of your Traefik reverse proxy into the Fail2ban docker container at `/var/log/traefik`. Otherwise, Fail2ban is not able to inspect your logs!

The access logs can be enabled in the traefik.yml static configuration file. In this blog post, the Traefik access logs are available at `/logs/traefik.log` of the corresponding Traefik Docker volume bind mount.

```
version: "3"

services:
  fail2ban:
    container_name: fail2ban
    hostname: fail2ban
    cap_add:
```

```
    - NET_ADMIN
    - NET_RAW
  environment:
    - TZ=Europe/Berlin
    - F2B_DB_PURGE_AGE=14d
  image: crazymax/fail2ban:latest
  network_mode: host
  restart: unless-stopped
  volumes:
    - <PATH-ON-YOUR-SERVER-FOR-PERSISTED-F2B-DATA>:/data
    - <PATH-ON-YOUR-SERVER-FOR-PERSISTED-TRAEFIK-DATA>/logs:/var/log/traefik
```

docker-compose.yml

Afterwards, start the Docker container with the following command:

```
sudo docker compose up -d
```

starting the docker container

Configuring Fail2ban

Upon starting our Fail2ban Docker container, we will notice four new folders at the persisted data storage we defined in the above `docker-compose.yml` file:

- action.d
- db
- filter.d
- jail.d

These folders are necessary and used to define our actual Fail2ban configuration in order to detect malicious behavior in log files as well as ban the IP address of misbehaving threat actors. Note that the `db` folder can be ignored for now since it only holds the sqlite3 database for Fail2ban.

Put the following configuration file called `jail.local` inside the `jail.d` directory. This is our main configuration file for Fail2ban. It defines various settings for our jails such as which log files to monitor and when as well as how long a threat actor should be banned. Adjust to your needs and

liking. Consult the Fail2ban documentation for more details regarding configuration options and syntax.

```
[DEFAULT]
# "bantime.increment" allows to use database for searching of previously banned ip's to increase a
# default ban time using special formula, default it is banTime * 1, 2, 4, 8, 16, 32...
bantime.increment = true

# "bantime.rndtime" is the max number of seconds using for mixing with random time
# to prevent "clever" botnets calculate exact time IP can be unbanned again:
bantime.rndtime = 2048

# following example can be used for small initial ban time (bantime=60) - it grows more aggressive at begin,
# for bantime=60 the multipliers are minutes and equal: 5 min, 30 min, 1 hour, 5 hour, 12 hour, 1 day, 2 day
bantime.multipliers = 1 5 30 60 300 720 1440 2880

[traefik-general-forceful-browsing]
# bots that trigger too many error codes like 404, 403 etc.
enabled = true
# ignore cloudflare cdn and private ip addresses
ignoreip = 103.21.244.0/22 103.22.200.0/22 103.31.4.0/22 104.16.0.0/13 104.24.0.0/14 108.162.192.0/18 131.0.72.0/22 141.101.64.0/18 162.158.0.0/15 172.64.0.0/13 173.245.48.0/20 188.114.96.0/20 190.93.240.0/20 197.234.240.0/22 198.41.128.0/17 2400:cb00::/32 2606:4700::/32 2803:f800::/32 2405:b500::/32 2405:8100::/32 2a06:98c0::/29 2c0f:f248::/32 127.0.0.0/8 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16
filter = traefik-general-forceful-browsing
logpath = /var/log/traefik/traefik.log
# this action bans every IP via DOCKER-USER chain. So the IP won't be able to access docker containers!
# Note: This only works for containers that don't use the dockernet MACVLAN interface
chain = DOCKER-USER
action = action-ban-docker-forceful-browsing
□□ action-ban-cloudflare
maxretry = 15
findtime = 60
bantime = 600
```

/jail.d/jail.local

⚡

Note that we actively whitelist Cloudflare IPv4 and IPv6 addresses as well as internal IP addresses with the `ignoreip` parameter. This ensures that we are not banning ourselves or our CDN provider by accident.

Remove the Cloudflare IP addresses if you do not plan on using Cloudflare as proxy. If Cloudflare is not in use, only define `ignoreip = 127.0.0.0/8 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16`. Also disable the action `action-ban-cloudflare` if you do not plan on using Cloudflare.

Note that we are using an incremental ban method. Threat actors are therefore banned for 10 minutes when first seen misbehaving. For any additionally detected hacking attempts from the same IP address afterwards, the ban time will increase exponentially.

Put the following configuration file called `traefik-general-forceful-browsing.conf` inside the `filter.d` directory. This configuration file is used to define which Traefik log entries are relevant for Fail2ban to monitor and act on. Please choose the correct failregex and ignoreregex based on your used Traefik log format (JSON vs. CLF). I recommend using JSON formatting, as it contains more data to filter on.

```
[INCLUDES]

[Definition]

# fail regex based on traefik JSON access logs with enabled user agent logging
failregex = ^{"ClientAddr":"<F-CLIENTADDR>.*</F-CLIENTADDR>","ClientHost":"<HOST>","ClientPort":"<F-CLIENTPORT>.*</F-CLIENTPORT>","ClientUsername":"<F-CLIENTUSERNAME>.*</F-CLIENTUSERNAME>","DownstreamContentSize":<F-DOWNSTREAMCONTENTSIZE>.*</F-DOWNSTREAMCONTENTSIZE>,"DownstreamStatus":<F-DOWNSTREAMSTATUS>.*</F-DOWNSTREAMSTATUS>,"Duration":<F-DURATION>.*</F-DURATION>,"OriginContentSize":<F-ORIGINCONTENTSIZE>.*</F-ORIGINCONTENTSIZE>,"OriginDuration":<F-ORIGINDURATION>.*</F-ORIGINDURATION>,"OriginStatus":(405|404|403|402|401),"Overhead":<F-OVERHEAD>.*</F-OVERHEAD>,"RequestAddr":"<F-REQUESTADDR>.*</F-REQUESTADDR>","RequestContentSize":<F-REQUESTCONTENTSIZE>.*</F-REQUESTCONTENTSIZE>,"RequestCount":<F-REQUESTCOUNT>.*</F-REQUESTCOUNT>,"RequestHost":"<F-CONTAINER>.*</F-CONTAINER>","RequestMethod":"<F-REQUESTMETHOD>.*</F-REQUESTMETHOD>","RequestPath":"<F-REQUESTPATH>.*</F-REQUESTPATH>","RequestPort":"<F-REQUESTPORT>.*</F-REQUESTPORT>","RequestProtocol":"<F-REQUESTPROTOCOL>.*</F-
```

REQUESTPROTOCOL>","RequestScheme":"<F-REQUESTSCHEME>.*</F-REQUESTSCHEME>","RetryAttempts":<F-RETRYATTEMPTS>.*</F-RETRYATTEMPTS>,.*"StartLocal":"<F-STARTLOCAL>.*</F-STARTLOCAL>","StartUTC":"<F-STARTUTC>.*</F-STARTUTC>","TLSCipher":"<F-TLSCIPHER>.*</F-TLSCIPHER>","TLSVersion":"<F-TLSVERSION>.*</F-TLSVERSION>","entryPointName":"<F-ENTRYPOINTNAME>.*</F-ENTRYPOINTNAME>","level":"<F-LEVEL>.*</F-LEVEL>","msg":"<F-MSG>.*</F-MSG>",("request_User-Agent":"<F-USERAGENT>.*</F-USERAGENT>",){0,1}?"time":"<F-TIME>.*</F-TIME>"}$

# custom date pattern for traefik JSON access logs
# based on https://github.com/fail2ban/fail2ban/issues/2558#issuecomment-546738270
datepattern = "StartLocal"\s*:\s*"%%Y-%%m-%%d[T]%%H:%%M:%%S\.%%f\d*(%%z)?",

# ignore common errors like missing media files or JS/CSS/TXT/ICO stuff
ignoreregex = ^{"ClientAddr":"<F-CLIENTADDR>.*</F-CLIENTADDR>","ClientHost":"<HOST>","ClientPort":"<F-CLIENTPORT>.*</F-CLIENTPORT>","ClientUsername":"<F-CLIENTUSERNAME>.*</F-CLIENTUSERNAME>","DownstreamContentSize":<F-DOWNSTREAMCONTENTSIZE>.*</F-DOWNSTREAMCONTENTSIZE>,"DownstreamStatus":<F-DOWNSTREAMSTATUS>.*</F-DOWNSTREAMSTATUS>,"Duration":<F-DURATION>.*</F-DURATION>,"OriginContentSize":<F-ORIGINCONTENTSIZE>.*</F-ORIGINCONTENTSIZE>,"OriginDuration":<F-ORIGINDURATION>.*</F-ORIGINDURATION>,"OriginStatus":(405|404|403|402|401),"Overhead":<F-OVERHEAD>.*</F-OVERHEAD>,"RequestAddr":"<F-REQUESTADDR>.*</F-REQUESTADDR>","RequestContentSize":<F-REQUESTCONTENTSIZE>.*</F-REQUESTCONTENTSIZE>,"RequestCount":<F-REQUESTCOUNT>.*</F-REQUESTCOUNT>,"RequestHost":"<F-REQUESTHOST>.*</F-REQUESTHOST>","RequestMethod":"<F-REQUESTMETHOD>.*</F-REQUESTMETHOD>","RequestPath":"<F-REQUESTPATH>.*(\.png|\.txt|\.jpg|\.ico|\.js|\.css|\.ttf|\.woff|\.woff2)(/)*?</F-REQUESTPATH>","RequestPort":"<F-REQUESTPORT>.*</F-REQUESTPORT>","RequestProtocol":"<F-REQUESTPROTOCOL>.*</F-REQUESTPROTOCOL>","RequestScheme":"<F-REQUESTSCHEME>.*</F-REQUESTSCHEME>","RetryAttempts":<F-RETRYATTEMPTS>.*</F-RETRYATTEMPTS>,.*"StartLocal":"<F-STARTLOCAL>.*</F-STARTLOCAL>","StartUTC":"<F-STARTUTC>.*</F-STARTUTC>","TLSCipher":"<F-TLSCIPHER>.*</F-TLSCIPHER>","TLSVersion":"<F-TLSVERSION>.*</F-TLSVERSION>","entryPointName":"<F-ENTRYPOINTNAME>.*</F-ENTRYPOINTNAME>","level":"<F-LEVEL>.*</F-LEVEL>","msg":"<F-MSG>.*</F-MSG>",("request_User-Agent":"<F-USERAGENT>.*</F-USERAGENT>",){0,1}?"time":"<F-TIME>.*</F-TIME>"}$

/filter.d/traefik-general-forceful-browsing.conf

The `failregex` parameter specifies a typical log entry format of failed HTTP requests. If such entries occur multiple times, it is usually an indicator of an ongoing forceful browsing or brute-forcing attack.

The `ignoreregex` parameter is used to ignore specific log entries, e.g. static or media files that cannot be found (404) on the server.

By using fail2ban's failregex notiation `<F-*>...</F-*>` we can even introduce custom variables. These variables may later be used when sending Telegram notifications or email alerts.
Finally, put the following two configuration files inside the `action.d` directory.

The first configuration file is used to ban threat actors on Cloudflare itself using the Cloudflare API. This is necessary if you are using Cloudflare as CDN provider with the orange cloud symbol enabled. If not, just neglect this action.d script at all and remove the `action-ban-cloudflare` action reference in the `jail.local` configuration file above.

If you will use Cloudflare as proxy, please adjust the below configuration file and define your Cloudflare credentials at the variables `cftoken` and `cfuser`. Use your account's email address and your `Global API Key` available at https://dash.cloudflare.com/profile/api-tokens.

```
#
# Author: Mike Rushton
#
# IMPORTANT
#
# Please set jail.local's permission to 640 because it contains your CF API key.
#
# This action depends on curl (and optionally jq).
# Referenced from http://www.normyee.net/blog/2012/02/02/adding-cloudflare-support-to-fail2ban by NORM YEE
#
# To get your CloudFlare API Key: https://www.cloudflare.com/a/account/my-account
#
# CloudFlare API error codes: https://www.cloudflare.com/docs/host-api.html#s4.2

[Definition]
```

```
# Option:  actionstart
# Notes.:  command executed on demand at the first ban (or at the start of Fail2Ban if
actionstart_on_demand is set to false).
# Values:  CMD
#
#actionstart = bash /data/action.d/telegram_notif.sh -a start

# Option:  actionstop
# Notes.:  command executed at the stop of jail (or at the end of Fail2Ban)
# Values:  CMD
#
#actionstop = bash /data/action.d/telegram_notif.sh -a stop

# Option:  actioncheck
# Notes.:  command executed once before each actionban command
# Values:  CMD
#
actioncheck =

# Option:  actionban
# Notes.:  command executed when banning an IP. Take care that the
#          command is executed with Fail2Ban user rights.
# Tags:    <ip>  IP address
#          <failures>  number of failures
#          <time>  unix timestamp of the ban time
# Values:  CMD
#
# API v1
#actionban = curl -s -o /dev/null https://www.cloudflare.com/api_json.html -d 'a=ban' -d
'tkn=<cftoken>' -d 'email=<cfuser>' -d 'key=<ip>'
# API v4
actionban = curl -s -o /dev/null -X POST <_cf_api_prms> \
          -d
'{"mode":"block","configuration":{"target":"<cftarget>","value":"<ip>"},"notes":"Fail2Ban
<name>"}' \
          <_cf_api_url>
          #bash /data/action.d/telegram_notif.sh -b <ip> -r "above reasons on Cloudflare - <name>"
```

```
# Option:  actionunban
# Notes.:  command executed when unbanning an IP. Take care that the
#          command is executed with Fail2Ban user rights.
# Tags:    <ip>  IP address
#          <failures>  number of failures
#          <time>  unix timestamp of the ban time
# Values:  CMD
#
# API v4
actionunban = id=$(curl -s -X GET <_cf_api_prms> \

"<_cf_api_url>?mode=block&configuration_target=<cftarget>&configuration_value=<ip>&page=
1&per_page=1&notes=Fail2Ban%%20<name>" \
                | { jq -r '.result[0].id' 2>/dev/null || tr -d '\n' | sed -nE
's/^.*"result"\s*:\s*\[\s*\{\s*"id"\s*:\s*"([^"]+)".*$/\1/p'; })
            if [ -z "$id" ]; then echo "<name>: id for <ip> cannot be found"; exit 0; fi;
            curl -s -o /dev/null -X DELETE <_cf_api_prms> "<_cf_api_url>/$id"
            #bash /data/action.d/telegram_notif.sh -u <ip> -r "above reasons on Cloudflare -
<name>"

_cf_api_url = https://api.cloudflare.com/client/v4/user/firewall/access_rules/rules
_cf_api_prms = -H 'X-Auth-Email: <cfuser>' -H 'X-Auth-Key: <cftoken>' -H 'Content-Type:
application/json'

[Init]

cftoken = <CF-API-TOKEN>
cfuser = <CF-USER-EMAIL>

cftarget = ip

[Init?family=inet6]
cftarget = ip6
```

/action.d/action-ban-cloudflare.conf

Additionally, use the following configuration file to ban the threat actor's IP address via `iptables` on your server directly. This is optional when using Cloudflare as proxy with the orange cloud symbol enabled, but mandatory if you are not using Cloudflare at all.

Since we are utilizing the itables string matching extension in `action-ban-docker-forceful-browsing.conf`, you have to ensure that iptables version >= 1.3.5 is installed on your system as well as that your kernel supports string matching. May read here.

Note: Newer OS versions seem to use nftables only. Since nftables do not support string matching, your IP bans via fail2ban are not effective. Install and configure legacy iptables if you want to use this feature. Otherwise, you have to use Cloudflare and its API for banning.

You can list kernel modules with the command `ls /lib/modules/`uname -r`/kernel/net/netfilter/` and search for the required `xt_string.ko` kernel module. If you do not get a hit, your kernel seems not yet to support iptables' string matching extension.

Note that we have to use the `DOCKER-USER` chain, since we are using Docker containers only. In detail, we will utilize the iptable's netfilter extension to ban the real IP address of a threat actor in the `X-Forwarded-For` header. Iptables can only see the source IP address of packets, namely the IP of your proxy, but not the real visitor's IP address commonly defined in HTTP headers like `X-Real-IP` or `X-Forwarded-For` etc. So use both, the `DOCKER-USER` and `INPUT` chain to be safe.

> Blacklist IPs with iptables behind a Reverse Proxy
>
> I have a Ubuntu Server 16 and I use iptables as firewall. HTTP and HTTPS traffic is behind a Reverse Proxy that I cannot control, but I have X-Forwarded-For field activated. Is it possible to filt...
>
> Image not found or type unknown
>
> Server Faultjlandercy
>
> Image not found or type unknown

```
[Definition]

actionban = iptables -I DOCKER-USER -m string --algo bm --string 'X-Forwarded-For: <ip>' -j DROP
            iptables -A INPUT -s <ip> -j DROP

actionunban = iptables -D DOCKER-USER -m string --algo bm --string 'X-Forwarded-For: <ip>' -j DROP
              iptables -D INPUT -s <ip> -j DROP
```

/action.d/action-ban-docker-forceful-browsing.conf

If all configuration files are set up, please restart the Fail2ban Docker container to reflect all changes. For example via the following bash command:

```
sudo docker compose up --force-recreate
```

Restarting the Fail2ban container

Configuring Telegram Notifications

Note that you can specify multiple `actionban` and `actionunban` actions. Currently, we only ban a misbehaving IP address of a threat actor via iptables and optionally via the Cloudflare API. However, we can also implement additional Telegram notifications when actual IP bans or unbans occur.

For this, append an additional line of code at the `actionban` and `actionunban` definitions, which calls a shell script that sends Telegram notifications. As an example, a proper configuration should then look like this:

```
[Definition]

actionstart = bash /data/action.d/telegram_notif.sh -a start
actionstop = bash /data/action.d/telegram_notif.sh -a stop

actionban = iptables -I DOCKER-USER -m string --algo bm --string 'X-Forwarded-For: <ip>' -j DROP
            iptables -A INPUT -s <ip> -j DROP
□□□bash /data/action.d/telegram_notif.sh -b <ip> -r "forceful browsing on <F-CONTAINER> (<name>)"

actionunban = iptables -D DOCKER-USER -m string --algo bm --string 'X-Forwarded-For: <ip>' -j
DROP
              iptables -D INPUT -s <ip> -j DROP
□□□  bash /data/action.d/telegram_notif.sh -u <ip>
```

Implementing Telegram Notifications in action.d scripts

The corresponding bash script to send notifications via your Telegram bot can be obtained via the following file download. Place the script within the `action.d` directory and define your Telegram API

token and chat ID inside the script.

Telegram Notification Script

Shell script to send notifications to a Telegram Bot

telegram_notif.sh

2 KB

Testing our Setup

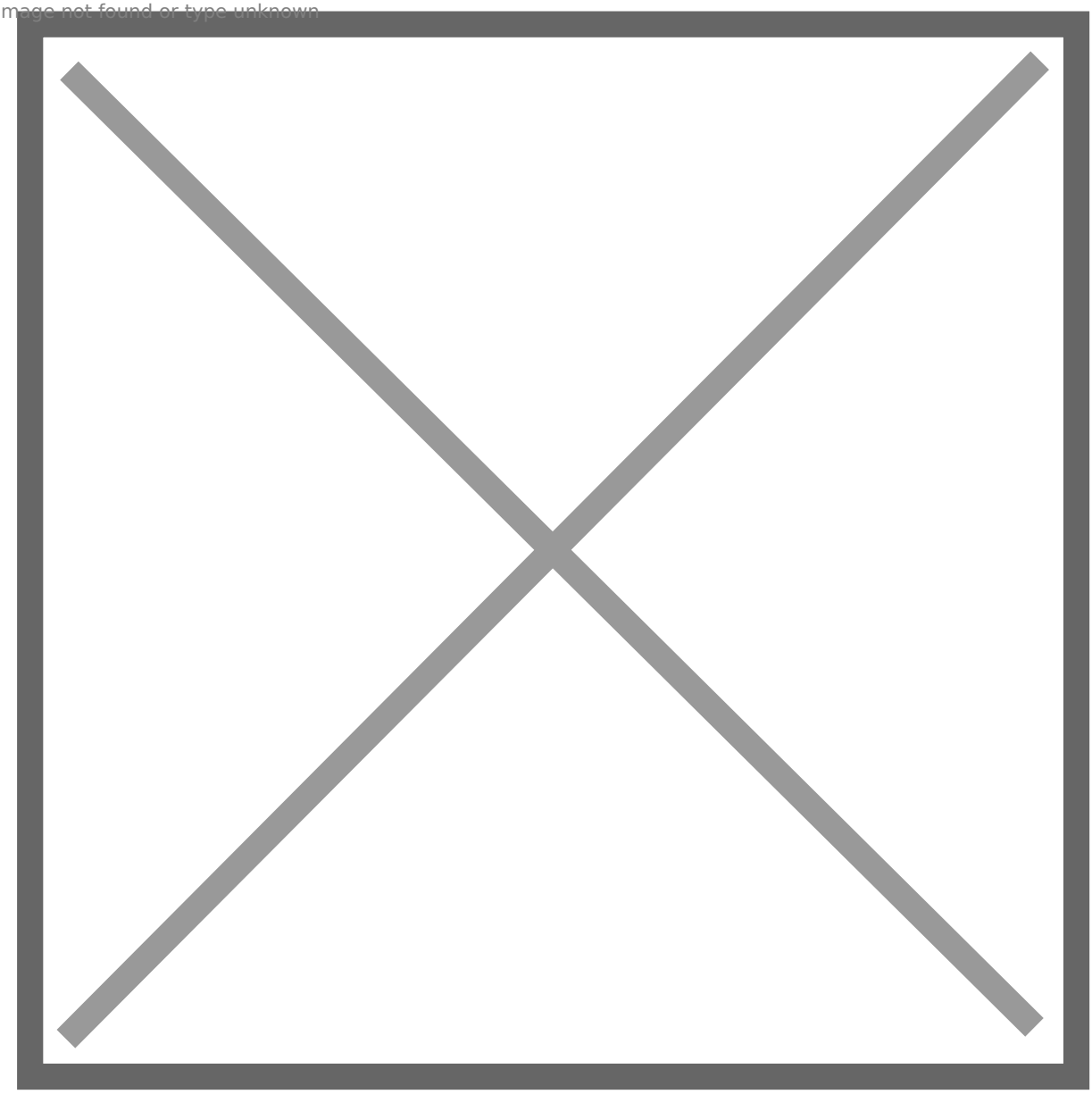
Finally, we should test our Fail2ban setup for proper configuration and that it really works. Head over to your mobile phone, disconnect from your local Wi-Fi network and start using a mobile LTE connection. This ensures that your soon to be made requests are not originating from a whitelisted IP address within the `ignoreip` parameter of the `jail.local` Fail2ban configuration file. Furthermore, we want to ensure that you do not lock yourself out with your real WAN IP address.

Then proceed by accessing a publicly exposed service via your mobile web browser. Start manipulating the browser URL and request a non-existent directory or file to trigger a `404 Not Found` HTTP error. Proceed with these requests (at least 15 times within 1 minute since this is the current threshold defined in Fail2ban) until you notice a ban of your IP address.

You should not be able to access any web service proxied by Traefik anymore. Your browser should also display an error page by Cloudflare (if Cloudflare as proxy is in use) notifying you that your IP address was blocked. This applies to all proxy hosts of your Traefik instance. You won't be able to access any HTTP web service for 10 minutes, until you will be automatically unbanned by Fail2ban again.

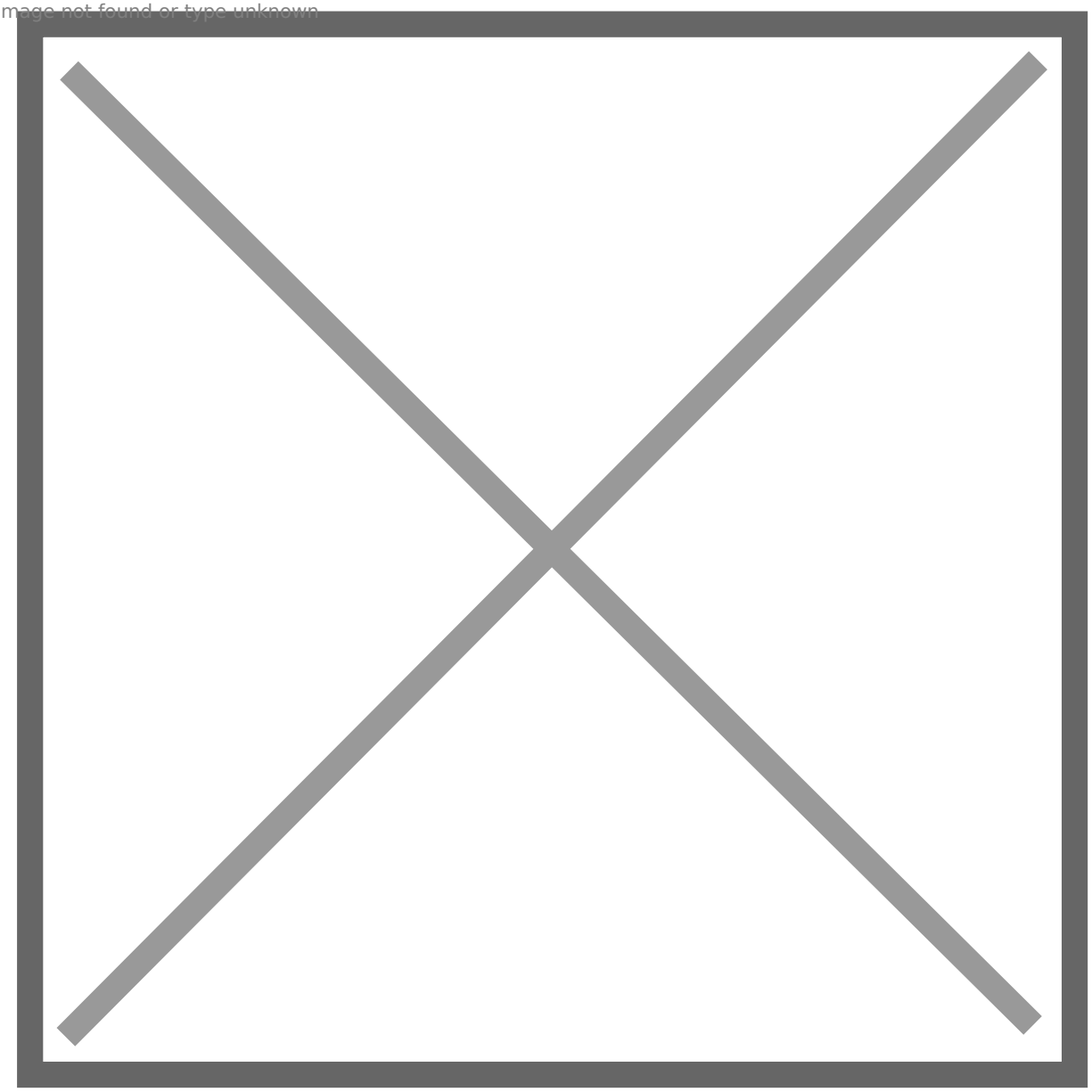
You may also consult the Fail2ban Docker logs for more details. You should see log entries notifying you about an IP address (your LTE IP) behaving maliciously. You may also login at Cloudflare if used and inspect the WAF or firewall rules to confirm an actual IP ban on the Cloudflare network for your managed domain. If configured, you would have also obtained a Telegram notification by now.

Image not found or type unknown



example fail2ban container logs

Image not found or type unknown



example Telegram notification about IP bans

To unban yourself, either wait 10 minutes or use the following bash command inside the Fail2ban docker container:

```
fail2ban-client set traefik-general-forceful-browsing unbanip <YOUR-IP>
```

Unbanning an IP address in Fail2ban Docker container

To test your Fail2ban filters, you can execute the following commands inside the Fail2ban Docker container:

```
# display matched log entries against /filter.d/traefik-general-forceful-browsing.conf
fail2ban-regex /var/log/traefik/traefik.log /etc/fail2ban/filter.d/traefik-general-forceful-browsing.conf
--print-all-matched

# display missed log entries against /filter.d/traefik-general-forceful-browsing.conf
fail2ban-regex /var/log/traefik/traefik.log /etc/fail2ban/filter.d/traefik-general-forceful-browsing.conf
--print-all-missed

# display ignored log entries against /filter.d/traefik-general-forceful-browsing.conf
fail2ban-regex /var/log/traefik/traefik.log /etc/fail2ban/filter.d/traefik-general-forceful-browsing.conf
--print-all-ignored
```

Test Fail2ban filter against Traefik logs

Revision #1
Created 3 August 2024 21:16:23 by ColtM
Updated 7 August 2024 23:22:43 by ColtM