Inhaltsverzeichnis

Single Sign On (SSO) mit Authelia & NGINX

Wenn man viele Anwendungen in einem gemeinsamen Docker- Umfeld betreiben will, kommt man kaum umhin, eine zentrale Anmelde- Maske bereit zustellen, damit der Anwender sich nicht bei jeder App separat anmelden und man nicht jede App für sich nneu konfigurieren muß. Da bietet sich die Kombination aus Authelia als SSO- Authenticator und NGINX als Reverse- Proxy an. NGINX nimmt dabei die Anfragen entgegen und läßt den User durch, sobals Authelia die Anmeldung bestätigt hat.

Warum nicht Traefik?

In den meisten HowTos wird Treafik als Proxy verwendet. Läßt man sich aber auf YouTube mal die Unterschiede zwischen NGINX und Treafik erklären, erfährt man, das NGINX die einfachere Wahl ist, wenn man einzelne Container auf einem eigenen Server laufen lassen möchte, wohingegen Traefik seine Vorteile ausspielt, wenn man cloud-native Apps mit Loadbalancing, Kubernetes usw. ochestrieren und skalieren musss. So wird hier aus der Notwendigkeit heraus, schnell eine kleine Lösung am Start zu haben, NGINX benutzt, wenn auch mit dem Risiko, bei einem Komplettumzug in die Cloud diesen Teil nochmal neu mit Traefik machen zu müssen-.

Als Basis dient die Authelia- NGINX-Proxy- Manager- Doku, allerdings wird hier versucht, das dortige Gewurschtel aus Querverweisen mal in eine sinnvolle Reihenfolge zu bringen..

Zuerst man jede Menge Setup:

Setup

Anlegen der grundlegenden Config- Dateien

NGINX

Anlegen des `authelia` Dockerverzeichnisses mit seinen darunterliegenden Config- Verzeichnissen:

```
mkdir -p authelia/data/nginx/snipplets
mkdir -p authelia/data/nginx/site-confs
mkdir -p authelia/data/nginx-proxy-manager/letsencrypt
mkdir -p authelia/data/nginx-proxy-manager/data
```

und diversen config- files.

Bitte Beachten: Auf dem Host liegen diese ganzen Dateien im `data` Folder, werden aber im Container alle auf einen `config` Folder gemappt. Der Grund dafür bleibt mir verborgen...

Bitte Beachten 2: Dokuwiki kann gemein sein: Wenn man die Code- Schnipsel hier aus der Webseite kopiert, fügt Dokuwiki bei Leerzeilen oft ein <Space> in die Zeile ein, was die einlesenden Programme schnell dazu bringt, sich beim Lesen zu verschlucken und man sich bei der Fehlersuche einen Wolf sucht...

data/nginx/site-confs/auth.conf

```
server {
    listen 80;
    server_name auth.*;

    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name auth.*;

    include /config/nginx/snippets/ssl.conf;

    set $upstream http://authelia:9091;

    location / {
        include /config/nginx/snippets/proxy.conf;
        proxy_pass $upstream;
    }

    location /api/verify {
        proxy_pass $upstream;
    }
}
```

Zum Testen legen wir mal eine whoami Applikkation an, die nur die HTTP- Header wieder zurückgibt

data/nginx/site-confs/whoami.conf

```
server {
    listen 80;
    server_name whoami.*;

    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name whoami.*;

    include /config/nginx/snippets/ssl.conf;
    include /config/nginx/snippets/authelia-location.conf;

    set $upstream http://whoami;

    location / {
        include /config/nginx/snippets/proxy.conf;
        include /config/nginx/snippets/authelia-authrequest.conf;
```

```
        proxy_pass $upstream;
    }
}
```

data/nginx/snippets/proxy.conf

```
## Headers
proxy_set_header Host $host;
proxy_set_header X-Original-URL $scheme://$http_host$request_uri;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header X-Forwarded-Host $http_host;
proxy_set_header X-Forwarded-Uri $request_uri;
proxy_set_header X-Forwarded-Ssl on;
proxy_set_header X-Forwarded-For $remote_addr;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header Connection "";

## Basic Proxy Configuration
client_body_buffer_size 128k;
proxy_next_upstream error timeout invalid_header http_500 http_502
http_503; ## Timeout if the real server is dead.
proxy_redirect  http://  $scheme://;
proxy_http_version 1.1;
proxy_cache_bypass $cookie_session;
proxy_no_cache $cookie_session;
proxy_buffers 64 256k;

## Trusted Proxies Configuration
## Please read the following documentation before configuring this:
##
https://www.authelia.com/integration/proxies/nginx/#trusted-proxies
# set_real_ip_from 10.0.0.0/8;
# set_real_ip_from 172.16.0.0/12;
# set_real_ip_from 192.168.0.0/16;
# set_real_ip_from fc00::/7;
real_ip_header X-Forwarded-For;
real_ip_recursive on;

## Advanced Proxy Configuration
send_timeout 5m;
proxy_read_timeout 360;
proxy_send_timeout 360;
proxy_connect_timeout 360;
```

data/nginx/snippets/authelia-location.conf

```
set $upstream_authelia http://authelia:9091/api/verify;

## Virtual endpoint created by nginx to forward auth requests.
```

```
location /authelia {
    ## Essential Proxy Configuration
    internal;
    proxy_pass $upstream_authelia;

    ## Headers
    ## The headers starting with X-* are required.
    proxy_set_header X-Original-URL $scheme://$http_host$request_uri;
    proxy_set_header X-Forwarded-Method $request_method;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Host $http_host;
    proxy_set_header X-Forwarded-Uri $request_uri;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header Content-Length "";
    proxy_set_header Connection "";

    ## Basic Proxy Configuration
    proxy_pass_request_body off;
    proxy_next_upstream error timeout invalid_header http_500 http_502
http_503; # Timeout if the real server is dead
    proxy_redirect http:// $scheme://;
    proxy_http_version 1.1;
    proxy_cache_bypass $cookie_session;
    proxy_no_cache $cookie_session;
    proxy_buffers 4 32k;
    client_body_buffer_size 128k;

    ## Advanced Proxy Configuration
    send_timeout 5m;
    proxy_read_timeout 240;
    proxy_send_timeout 240;
    proxy_connect_timeout 240;
}
```

data/nginx/snippets/authelia-authrequest.conf

```
## Send a subrequest to Authelia to verify if the user is authenticated
and has permission to access the resource.
auth_request /authelia;

## Set the $target_url variable based on the original request.

## Comment this line if you're using nginx without the http_set_misc
module.
set_escape_uri $target_url $scheme://$http_host$request_uri;

## Uncomment this line if you're using NGINX without the http_set_misc
module.
# set $target_url $scheme://$http_host$request_uri;
```

```
## Save the upstream response headers from Authelia to variables.
auth_request_set $user $upstream_http_remote_user;
auth_request_set $groups $upstream_http_remote_groups;
auth_request_set $name $upstream_http_remote_name;
auth_request_set $email $upstream_http_remote_email;

## Inject the response headers from the variables into the request made
to the backend.
proxy_set_header Remote-User $user;
proxy_set_header Remote-Groups $groups;
proxy_set_header Remote-Name $name;
proxy_set_header Remote-Email $email;

## If the subreqest returns 200 pass to the backend, if the subrequest
returns 401 redirect to the portal.
error_page 401 =302 https://auth.example.com/?rd=$target_url;
```

Authelia

```
mkdir -p authelia/data/authelia/config
```

Bitte Beachten: Auf dem Host liegen diese ganzen Dateien im `data` Folder, werden aber im Container alle auf einen `config` Folder gemappt. Der Grund dafür bleibt mir verborgen...

data/authelia/config/users_database.yml

```
---
###############################################################
#                         Users Database                      #
###############################################################

# This file can be used if you do not have an LDAP set up.

# List of users
users:
  authelia:
    disabled: false
    displayname: "Authelia User"
    # Password is authelia
    password:
"$6$rounds=50000$BpLnfgDsc2WD8F2q$Zis.ixdg9s/UOJYrs56b5QEZFiZECu0qZVNsI
YxBaNJ7ucIL.nlxVCT5tqh8KHG8X4tlwCFm5r6NTOZZ5qRFN/"  # yamllint disable-
line rule:line-length
    email: authelia@authelia.com
    groups:
      - admins
      - dev
```

```
...
```

data/authelia/config/configuration.yml

```
---
###############################################################
#                   Authelia configuration                    #
###############################################################

jwt_secret: a_very_important_secret
default_redirection_url: https://public.example.com

server:
  host: 0.0.0.0
  port: 9091

log:
  level: debug
# This secret can also be set using the env variables
AUTHELIA_JWT_SECRET_FILE

totp:
  issuer: authelia.com

# duo_api:
#  hostname: api-123456789.example.com
#  integration_key: ABCDEF
#  # This secret can also be set using the env variables
AUTHELIA_DUO_API_SECRET_KEY_FILE
#  secret_key: 1234567890abcdefghifjkl

authentication_backend:
  file:
    path: /config/users_database.yml

access_control:
  default_policy: deny
  rules:
    # Rules applied to everyone
    - domain: public.example.com
      policy: bypass
    - domain: traefik.example.com
      policy: one_factor
    - domain: secure.example.com
      policy: two_factor

session:
  name: authelia_session
  # This secret can also be set using the env variables
AUTHELIA_SESSION_SECRET_FILE
```

```
  secret: unsecure_session_secret
  expiration: 3600  # 1 hour
  inactivity: 300  # 5 minutes
  domain: example.com  # Should match whatever your root protected
domain is

  redis:
    host: redis
    port: 6379
    # This secret can also be set using the env variables
AUTHELIA_SESSION_REDIS_PASSWORD_FILE
    # password: authelia

regulation:
  max_retries: 3
  find_time: 120
  ban_time: 300

storage:
  encryption_key:
you_must_generate_a_random_string_of_more_than_twenty_chars_and_configu
re_this
  local:
    path: /config/db.sqlite3

notifier:
#  smtp:
#    username: test
#    # This secret can also be set using the env variables
AUTHELIA_NOTIFIER_SMTP_PASSWORD_FILE
#    password: password
#    host: mail.example.com
#    port: 25
#    sender: admin@example.com

  disable_startup_check: false
  filesystem:
    filename: /config/notification.txt
...
...
```

Docker Compose

Nachdm man die ganzen obigen Dateinem im Config- Ordner angelegt hat, gibts im `authelia`Folder die eigentliche `docker-compose.yaml` Datei

docker-compose.yaml

```
---
version: "3.8"

networks:
  net:
    driver: bridge

services:
  nginx:
    container_name: nginx
    image: jc21/nginx-proxy-manager
    restart: unless-stopped
    networks:
      net:
        aliases: []
    ports:
      - '80:80'
      - '81:81'
      - '443:443'
    volumes:
      - ${PWD}/data/nginx-proxy-manager/data:/data
      - ${PWD}/data/nginx-proxy-manager/letsencrypt:/etc/letsencrypt
      - ${PWD}/data/nginx/snippets:/config/nginx/snippets:ro
    environment:
      TZ: 'Europe/Berlin'
  authelia:
    container_name: authelia
    image: authelia/authelia
    restart: unless-stopped
    networks:
      net:
        aliases: []
    expose:
      - 9091
    volumes:
      - ${PWD}/data/authelia/config:/config
    environment:
      TZ: 'Europe/Berlin'
  whoami:
    container_name: whoami
    image: docker.io/traefik/whoami
    restart: unless-stopped
    networks:
      net:
        aliases: []
    expose:
      - 80
    environment:
      TZ: 'Europe/Berlin'
...
```

Test

Wenn die obigen Dateien alle angelegt wurden, kann man es ja mal krachen lassen:

```
docker compose up
```

dann auf den NGINX Proxy Manager Login Screen gehen (<hostname>:81) und anmelden mit den Default- Werten `admin@example.com` und `changeme` und erstmal den Admin- Account einrichten

SSH Zertifikat erzeugen

Warum auch einfach..:

Erzeugen eines Zertifikats mit

```
# non-interactive and 10 years expiration
openssl req -newkey rsa:4096 \
 -x509 \
 -sha256 \
 -days 3650 \
 -nodes \
 -out example.crt \
 -keyout example.key \
 -subj "/C=US/ST=Oregon/L=Portland/O=Company Name/OU=Org/CN=www.example.com"
```

Die beiden Dateien kopieren wir dann mit SCP auf den Rechner, wo auch der Browser läuft und gehen im NGINX Proxy Manager dann auf `SSL Certificates` und dort auf `AD Certificate` **aber auf den Button oben rechts, nicht auf den in der Mitte!**, denn der in der Mitte kann nur Let's Encrypt- Certificates, aber der Button oben rechts kann auch Custom Certificates (WTF!)

Hinzufügen neuer Apps (Proxy Hosts)

Wenn der NGINX endlich läuft, können neue Trusted Hosts hinzugefügt werden, die ja letztlich jedesmal eigene Docker- Container mit einer eigenen App sind.

Diese fügt man als Proxy Hosts hinzu. Als Protokoll nimmt man http, als Host den Namen, unter dem die App als Container läuft, weil dies auch der Host- Name ist, unter dem sie im Docker- internen Netzwerk erreichbar ist.

Unter SSH fügt man immer das jeweils oben erzeugte SSH - Zertifikat hinzu

Unter `Advanced` gibt man bei Public Pages diese Settings an

Achtung: Der Pfad für `include` ist hier anders als in der originalen Authelia-NGINX- Doku. Der Pfad in der Original Doku ist **falsch!** (und wieder war ein Tag dahin…)

```
location / {
```

```
    include /config/nginx/snippets/proxy.conf;
    proxy_pass $forward_scheme://$server:$port;
}
```

für protected Pages gelten diese Advanced Settings:

```
include /config/nginx/snippets/authelia-location.conf;

location / {
    include /config/nginx/snippets/proxy.conf;
   include /config/nginx/snippets/authelia-authrequest.conf;
    proxy_pass $forward_scheme://$server:$port;
}
```