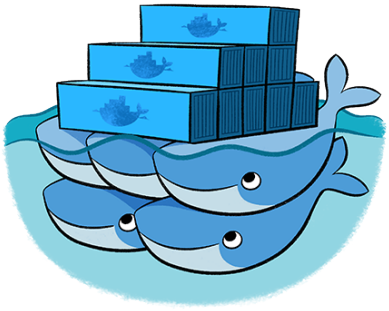
Authelia

Setupguide for Authelia Multi Factor Authentication Middleware

Overview

What is Authelia?



Introduction

Authelia is a two-factor authentication (2FA) and single sign-on (SSO) server focused on enhancing the security of applications and users. Acting as an extension of reverse proxies, it provides various authentication-related features, such as:

- Multiple two-factor authentication methods.
- Identity verification for registering second-factor devices.
- Self-service password reset functionality.
- Account banning after excessive failed login attempts (rate limiting).

Features Summary

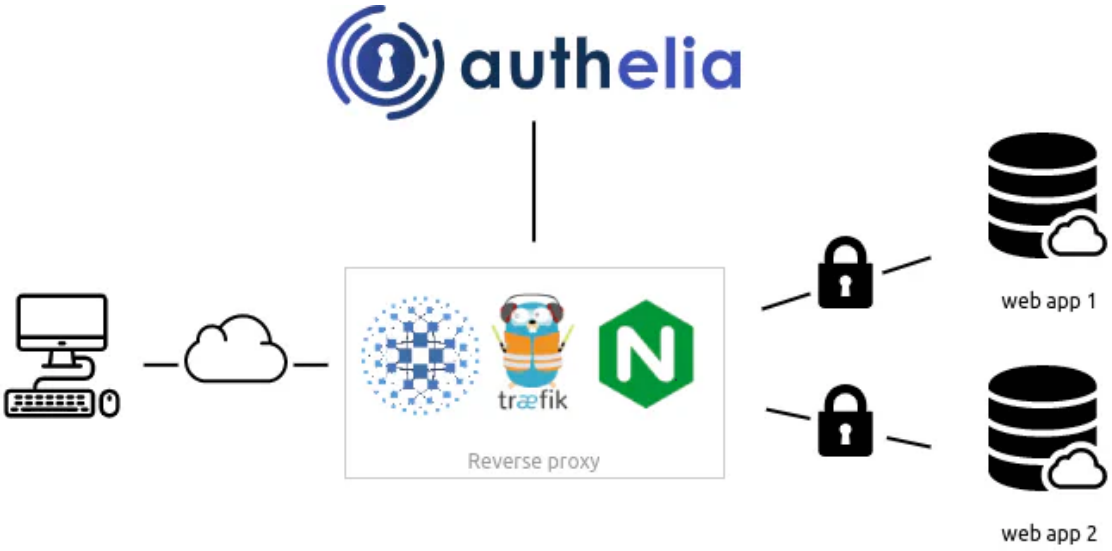
This is a list of the key features of Authelia:

- □□ Several second factor methods:
  - □□ Security Keys that support FIDO2 WebAuthn with devices like a YubiKey.
  - □ Time-based One-Time password with compatible authenticator applications.
  - □□ Mobile Push Notifications with Duo.
- ✉ Password reset with identity verification using email confirmation.
- □□ Access restriction after too many invalid authentication attempts.
- □□ Fine-grained access control using rules which match criteria like subdomain, user, user group membership, request URI, request method, and network.
- ⚙ Choice between one-factor and two-factor policies per-rule.
- □□ Support of basic authentication for endpoints protected by the one-factor policy.
- □□ Highly available using a remote database and Redis as a highly available KV store.
- □□ Compatible with Traefik out of the box using the ForwardAuth middleware.

Architecture

Authelia integrates seamlessly with reverse proxies like Traefik (see the full list of supported proxies). It enhances these proxies by adding authentication and authorization features alongside a login portal.

Authelia is connected to the reverse proxy but does not directly interact with the backend applications. Payloads sent by clients to the protected applications never reach Authelia; only authentication-related information, such as the `Authorization` header, is processed. This allows Authelia to protect any HTTP-based APIs, including REST and GraphQL APIs, without interfering with application data.



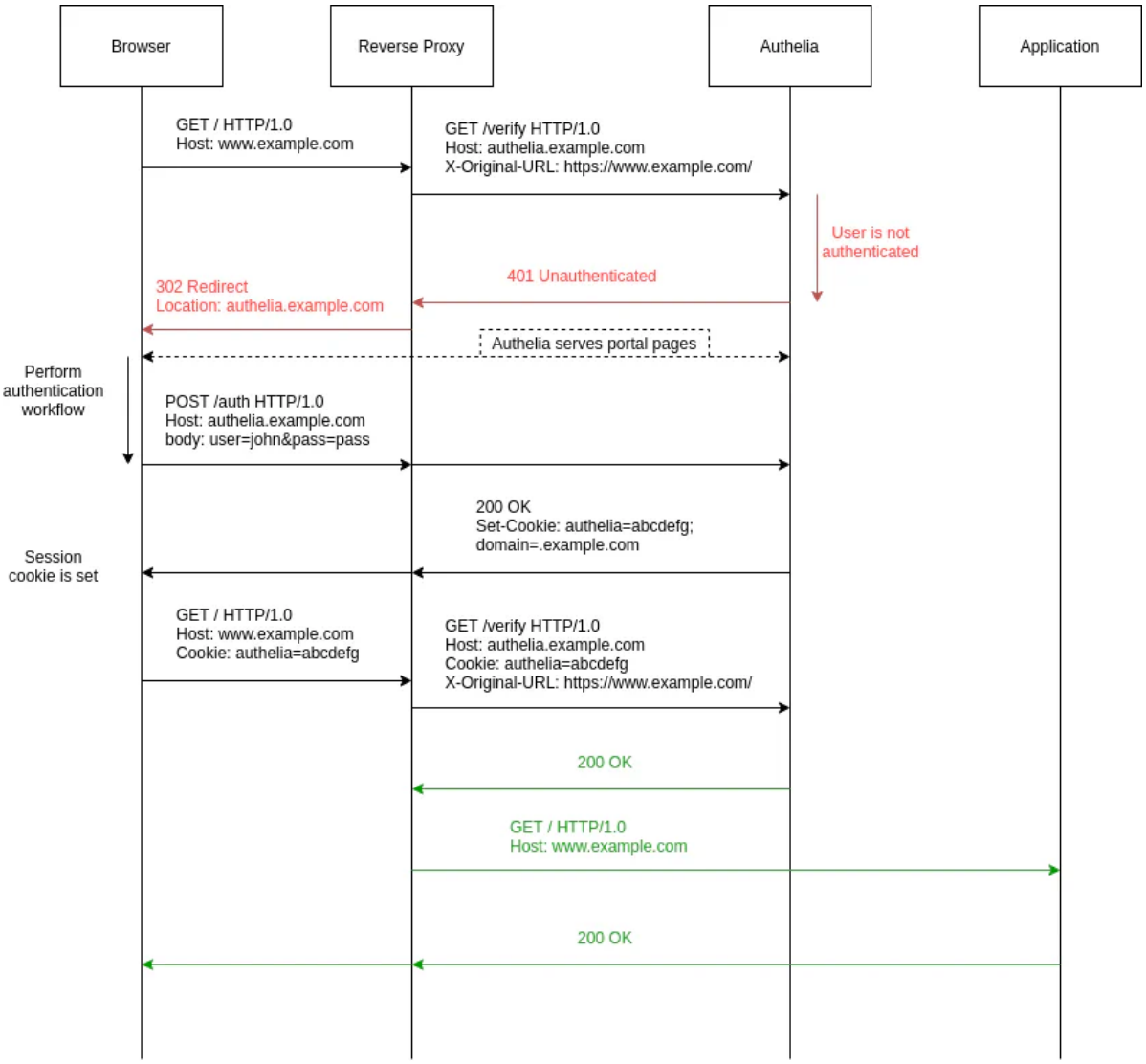
Workflow

Reverse proxies configured with Authelia send all incoming requests to Authelia for authentication verification. Authelia instructs the reverse proxy to either:

- ✔ Allow the request to pass through if the user is authenticated, or
- □ Block the request if the user is unauthenticated or unauthorized.

Step-by-Step Workflow

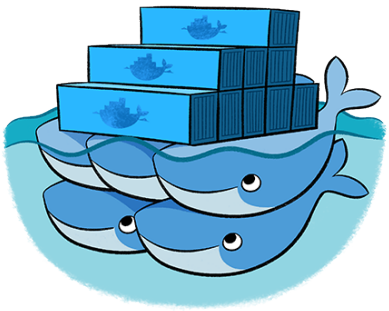
1. □ **Unauthenticated User Request:**
   When an unauthenticated user sends their first request to the reverse proxy, Authelia determines the user is not authenticated (no session cookie provided). The user is redirected to the Authelia login portal.
2. □□ **Authentication Portal:**
   The user completes the authentication workflow via Authelia's portal. Upon successful authentication, a session cookie is issued, valid for all subdomains of the protected domain.
3. ✔ **Authenticated User Request:**
   The user revisits the initial website, now including the session cookie in their requests. Authelia verifies the session and instructs the reverse proxy to allow the request to pass through.


Browser
Reverse Proxy
Authelia
Application
GET / HTTP/1.0
Host: www.example.com
GET /verify HTTP/1.0
Host: authelia.example.com
X-Original-URL: https://www.example.com/
User is not authenticated
401 Unauthenticated
302 Redirect
Location: authelia.example.com
Authelia serves portal pages
Perform authentication workflow
POST /auth HTTP/1.0
Host: authelia.example.com
body: user=john&pass=pass
200 OK
Set-Cookie: authelia=abcdefg;
domain=.example.com
Session cookie is set
GET / HTTP/1.0
Host: www.example.com
Cookie: authelia=abcdefg
GET /verify HTTP/1.0
Host: authelia.example.com
Cookie: authelia=abcdefg
X-Original-URL: https://www.example.com/
200 OK
GET / HTTP/1.0
Host: www.example.com
200 OK

Getting Started

Getting Started

Step by Step Beginners Setup Guide for Authelia



This article provides detailed instructions on integrating Authelia as a middleware with Traefik. Using Docker labels for configuration, this setup allows Traefik to query Authelia for authorization

on every web request. Authelia validates session cookies and access permissions for secure resource control. The information is partially sourced from Brynn Crowley, referencing his setup guide.

Prerequisites

- Docker Swarm
- GlusterFS & Keepalived
- Traefik Reverse Proxy

Important Notes

> Configuration uses Docker labels directly in `docker-compose.yaml` to configure the Traefik Middleware.
> Examples use a *whoami* application for demonstration.
>
> Advanced configurations (e.g., SMTP) are available in Authelia documentation.

Step-by-Step Guide

Step 1: Create Folders in the GlusterFS

```
mkdir -p /mnt/glustermount/data/authelia_data/
mkdir -p /mnt/glustermount/data/authelia_data/logs
mkdir -p /mnt/glustermount/data/authelia_data/config
mkdir -p /mnt/glustermount/data/authelia_data/secrets
```

Step 2: Create External Network for Traefik Proxy (if not already done)

Create the `management_net` network:

```
docker network create -d overlay management_net
```

Step 3: Configure User Database

Create a basic user database:

```
nano mkdir -p /mnt/glustermount/data/authelia_data/config/users.yml
```

Paste this Content:

```
users:
  authelia: ## Username
    displayname: 'Authelia User'
    ## WARNING: This is a default password for testing only!
    ## IMPORTANT: Change this password before deploying to production!
    ## Generate a new hash using the instructions at:
    ## https://www.authelia.com/reference/guides/passwords/#passwords
    ## Password is 'authelia'
    password:
'$6$rounds=50000$BpLnfgDsc2WD8F2q$Zis.ixdg9s/UOJYrs56b5QEZFiZECu0qZVNsIYxBaNJ7ucIL.nlxVCT5tqh8KH
G8X4tlwCFm5r6NTOZZ5qRFN/'
    email: 'authelia@authelia.com'
    groups:
      - 'admin'
      - 'dev'
```

> The current password listed is `authelia`. It is important you Generate a new password hash.

Step 3.1: Generate Password Hash (Optional)

```
docker run --rm -it authelia/authelia:latest authelia crypto hash generate argon2
```

Step 4: Create Secrets

First provide needed Rights:

```
chown 8000:8000 /mnt/glustermount/data/authelia_data/secrets/ && chmod 0700
/mnt/glustermount/data/authelia_data/secrets/
```

Then Create Secret Files:

```
docker run --rm -u 8000:8000 -v /mnt/glustermount/data/authelia_data/secrets/:/secrets
docker.io/authelia/authelia sh -c "cd /secrets && authelia crypto rand --length 64 session_secret.txt
storage_encryption_key.txt jwt_secret.txt"
```

Step 5: Create Basic Authelia Configuration

```
server:
  address: 'tcp4://:9091'

log:
  level: debug
  file_path: '/var/log/authelia/authelia.log'
  keep_stdout: true

identity_validation:
  elevated_session:
    require_second_factor: true
  reset_password:
    jwt_lifespan: '5 minutes'
    jwt_secret: {{ secret "/secrets/jwt_secret.txt" | mindent 0 "|" | msquote }}

totp:
  disable: false
  issuer: 'aeoneros.com'
  period: 30
  skew: 1

password_policy:
  zxcvbn:
    enabled: true
    min_score: 4

authentication_backend:
```

```
  file:
    path: '/config/users.yml'
    password:
      algorithm: 'argon2'
      argon2:
        variant: 'argon2id'
        iterations: 3
        memory: 65535
        parallelism: 4
        key_length: 32
        salt_length: 16

access_control:
  default_policy: 'deny'
  rules:
    - domain: 'traefik.aeoneros.com'
      policy: 'one_factor'
    - domain: 'whoami-secure.aeoneros.com'
      policy: 'two_factor'

session:
  name: 'authelia_session'
  secret: {{ secret "/secrets/session_secret.txt" | mindent 0 "|" | msquote }}
  cookies:
    - domain: 'aeoneros.com'
      authelia_url: 'https://auth.aeoneros.com'

regulation:
  max_retries: 4
  find_time: 120
  ban_time: 300

storage:
  encryption_key: {{ secret "/secrets/storage_encryption_key.txt" | mindent 0 "|" | msquote }}
  local:
    path: '/config/db.sqlite3'

notifier:
  disable_startup_check: false
  filesystem:
```

```
    filename: '/config/notification.txt'
```

Step 6: Create Docker Compose

> It is important to know that Traefik needs to wait for Authelia to startup. Thats what the depends Function is for.
> Otherwise Traefik will not notice the Authelia Middleware and maybe provide an Error.

```
version: '3.3'

services:
  traefik:
    user: 0:0 #Container being started with Root rights
    image: 'traefik:latest'
    security_opt:
      - 'no-new-privileges=true'
    restart: 'unless-stopped'
    depends_on:
      - authelia
    ports:
      # The Web UI (enabled by --api.insecure=true in traefik.toml)
      - '8080:8080'
      # The Available Ports (forward your router's incoming ports to the ports on the host)
      - '80:80'
      - '443:443'
    networks:
      management_net:
        aliases:
          - 'auth.domain.com'
      authelia: {}
    volumes:
      # So that Traefik can listen to the Docker events (read-only)
      - '/var/run/docker.sock:/var/run/docker.sock:ro'
      # LetsEncrypt ACME Configuration
      - '/mnt/glustermount/data/traefik_data/acme.json:/le/acme.json'
      # Mount for Traefik AccessLog
      - '/mnt/glustermount/data/traefik_data/access.log:/access.log'
      # (STATIC CONFIG)
```

```
    - './traefik/config/traefik.yml:/traefik.yml:ro'
    # (DYNAMIC CONFIG)
    - './traefik/config/dynamic.yml:/dynamic.yml:ro'

  environment:
    - TZ=Europe/Zurich
  deploy:
    mode: replicated
    replicas: 1
    labels:
      - 'traefik.enable=true'
      - 'traefik.http.routers.traefik.rule=Host(`traefik.domain.com`)'
      - 'traefik.http.routers.traefik.service=api@internal'
      - 'traefik.http.services.traefik.loadbalancer.server.port=8080'
      - 'traefik.http.routers.traefik.tls.certresolver=leresolver'
      - 'traefik.http.routers.traefik.entrypoints=websecure'
      - 'traefik.http.routers.http-catchall.rule=hostregexp(`{host:.+}`)'
      - 'traefik.http.routers.http-catchall.entrypoints=web'
      - 'traefik.http.routers.http-catchall.middlewares=redirect-to-https'
      - 'traefik.http.middlewares.redirect-to-https.redirectscheme.scheme=https'
      #Authelia Integration
      - 'traefik.http.routers.dashboard.middlewares=authelia@docker'

authelia:
  image: 'authelia/authelia:4.38'
  container_name: 'authelia'
  volumes:
    - '/mnt/glustermount/data/authelia_data/secrets:/secrets:ro'
    - '/mnt/glustermount/data/authelia_data/config:/config'
    - '/mnt/glustermount/data/authelia_data/logs:/var/log/authelia/'
  networks:
    authelia: {}
    management_net: {}
  labels:
    ## Expose Authelia through Traefik
    traefik.enable: 'true'
    traefik.docker.network: 'authelia'
    traefik.http.routers.authelia.rule: 'Host(`auth.domain.com`)'
    traefik.http.routers.authelia.entrypoints: 'websecure'
    traefik.http.routers.authelia.tls.certresolver: 'leresolver'
```

```
      traefik.http.services.authelia.loadbalancer.server.port: '9091'
      ## Setup Authelia ForwardAuth Middlewares
      traefik.http.middlewares.authelia.forwardAuth.address: 'http://traefik_authelia:9091/api/authz/forward-auth'
      traefik.http.middlewares.authelia.forwardAuth.trustForwardHeader: 'true'
      traefik.http.middlewares.authelia.forwardAuth.authResponseHeaders: 'Remote-User,Remote-
Groups,Remote-Name,Remote-Email'
    environment:
      TZ: 'Europe/Zurich'
      X_AUTHELIA_CONFIG_FILTERS: 'template'

  whoami-secure:
    image: 'traefik/whoami'
    restart: 'unless-stopped'
    container_name: 'whoami-secure'
    depends_on:
      - authelia
    labels:
      traefik.enable: 'true'
      traefik.http.routers.whoami-secure.rule: 'Host(`whoami-secure.domain.com`)'
      traefik.http.routers.whoami-secure.entrypoints: 'websecure'
      traefik.http.routers.whoami-secure.middlewares: 'authelia@docker'
      traefik.http.services.whoami-secure.loadbalancer.server.port: '80'
      traefik.http.routers.whoami-secure.tls.certresolver: 'leresolver'
    networks:
      management_net: {}
networks:
  management_net:
    external: true  # Primary network for management
  authelia:
```

Step 7: Start the Stack

You can either do that with the provided Command or start the Stack with Portainer.

```
docker compose up -d
```

Step 8: Verify Setup

- Check container status: `docker compose ps`
- Access Traefik dashboard: https://traefik.domain.com
- Test authentication: https://whoami-secure.domain.com

> Now you are ready to Setup with further Configuration.
>
> **It is possible to add Authelia Middleware to your Custom Applications by adding: traefik.http.routers.YOUR-APPLICATION.middlewares: 'authelia@docker'**

Troubleshooting

- Check logs: `docker logs authelia`
- Ensure secret files exist and have correct permissions.
- Check Official Docs: https://www.authelia.com/configuration/prologue/introduction/

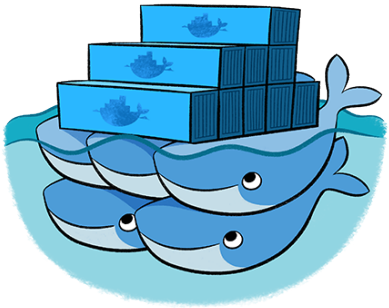
Further Configuration

Check out other Posts in my Wiki about setting up SMTP for example.

https://wiki.aeoneros.com/books/authelia/chapter/configuration

Configuration (Notifications)

Setup SMTP



How to Set Up SMTP Notifications for Authelia

Authelia supports sending email notifications via an SMTP server, which is essential for account management and security events like password recovery and login alerts. This guide will walk you through configuring SMTP notifications for Authelia.

Prerequisites

- An SMTP server (e.g., Gmail, custom domain SMTP server)
- Access to your `configuration.yml` file
- Basic knowledge of YAML and email server parameters

Configuration File Example

Below is a basic example of the `notifier` section in the `configuration.yml` file:

```
notifier:
  disable_startup_check: false
  smtp:
    address: 'smtp://127.0.0.1:25'
    timeout: '5s'
    username: 'test'
    password: 'password'
    sender: "Authelia "
    identifier: 'localhost'
    subject: "[Authelia] {title}"
    startup_check_address: 'test@aeoneros.com'
    disable_require_tls: false
    disable_starttls: false
    disable_html_emails: false
    tls:
      server_name: 'smtp.aeoneros.com'
      skip_verify: false
      minimum_version: 'TLS1.2'
      maximum_version: 'TLS1.3'
      certificate_chain: |
        -----BEGIN CERTIFICATE-----
        ...
        -----END CERTIFICATE-----
```

```
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  private_key: |
    -----BEGIN RSA PRIVATE KEY-----
    ...
    -----END RSA PRIVATE KEY-----
```

Key Configuration Options

- **address:** The SMTP server's address. Must include the protocol (`smtp`, `submission`, or `submissions`).
- **username:** The username for SMTP authentication. Pair it with a password.
- **password:** The password for SMTP authentication. It is strongly recommended to use a secret for containerized environments.
- **sender:** The email address used for the "From" field. Must follow RFC5322 format.
- **identifier:** The identifier sent with HELO/EHLO commands. Avoid using `localhost` for external SMTP services.
- **subject:** The subject template for emails, supporting the `{title}` placeholder.
- **tls:** Optional TLS settings, including minimum/maximum versions and custom certificate chains.

Using Gmail

If you are using Gmail as your SMTP server, you must generate an *App Password*. Configure the `notifier` section as follows:

```
notifier:
  smtp:
    address: 'submission://smtp.gmail.com:587'
    username: 'your-email@gmail.com'
    password: 'your-app-password'
    sender: "Admin "
```

Follow Google's documentation to generate an app password: Generate App Password.

Testing the Configuration

To test your configuration, restart Authelia and check the logs:

```
docker logs authelia
```

Ensure no errors related to the SMTP connection appear. Use the `startup_check_address` to validate the SMTP setup without sending actual emails.

Troubleshooting

- Ensure the SMTP server address and port are correct.
- Verify credentials are correct and have sufficient permissions.
- Check the logs for specific error messages.
- Review Authelia's official documentation for advanced troubleshooting tips.

With the correct configuration, SMTP notifications enhance user experience and provide critical security alerts seamlessly.