**Doku – MFA**

Thema:

­­­­­­­­­­Zugriffskontrolle von verschiedenen Services in einer Cloud-Infrastruktur

Vorgehensweise/ Konzept:

1. Grundlage

* Wir haben Services in Docker-Containern laufen, die durch „docker-compose.yml“-Dateien erstellt werden
* Diese Services befinden sich hinter dem Reverse Nginx Proxy
* Vor allen Services soll MFA implementiert werden

1. MFA-Lösung auswählen

* Eine passende MFA-Lösung ausfindig machen
* Optionen wären …

… TOTP (Time-based One-Time Password)

… OTP (One-Time Password) v­­ia SMS

… mobile App-basierten Lösungen wie Authenticator

* Es gibt Hersteller/ Bereitsteller, die MFA per Default als MFA-Service anbieten

1. Sidecar-Container erstellen

* Einen Docker-Container für das MFA-Sidecar erstellen
* Das Sidecar wird flexible MFA-Authentifizierungsanfragen bearbeiten

1. Nginx für MFA konfigurieren

* Die Nginx-Konfiguration aktualisieren, sodass der MFA-Zugang vor den „Backend“-Services geschalten wird
* Das Nginx „auth\_request“-Modul nutzen, um die MFA-Checks zum Sidecar-Container zuzuschalten

1. MFA in den Reverse Proxy integrieren

* Die Nginx-Konfiguration konfigurieren, um den MFA-Sidecar-Container für jede ankommende Anfrage aufzurufen
* Das Sidecar sollte MFA und den Gültigkeitsprozesse/ Prüfung verarbeiten können
* Bei erfolgreicher MFA-Authentifizierung sollte das Sidecar die Anfrage an den entsprechenden Service weiterleiten

1. MFA-Einstellungen konfigurieren

* Das MFA-Sidecar so konfigurieren, dass die ausgewählte MFA-Methode unterstützt, wird
* MFA-Inhalte sicher speichern und einen Sicherheitsmechanismus einführen

1. Die Einrichtung testen

* Die MFA-Implementierung auf Korrektheit testen
* Bestätigen, dass die Benutzer durch die MFA-Authentifizierung aufgefordert werden, wenn es um den Zugriff/ Zugang zu den Services durch den Reverse Proxy geht

1. Überwachung und Protokollierung

* Überwachung und Protokollierung einführen, um das MFA-Sidecar und dessen Versuche, Erfolge und Misserfolge zu messen
* Die MFA-Aktivität durch regelmäßige Alarme einstellen

1. Dokumentation und Training/ Tests

* Die MFA-Einstellung dokumentieren, auch die Anleitungen für die Benutzer wie dessen Einrichtung des MFAs auf die jeweiligen Endgeräte erfolgt
* Training und Unterstützung für Benutzer, die noch keine Berührungspunkte mit MFA hatten, bereitstellen

1. Backup und Wiederherstellung

* Backup- und Wiederherstellungspläne für MFA implementieren, um sicherzustellen, dass die Nutzer den Zugang wieder erhalten, für den Fall, dass diese ihr Endgerät verlieren, etc.

1. Sicherheitsupdates

* Die MFA-Lösung, das Sidecar und den Nginx Proxy auf aktuellem Stand halten

1. Übereinstimmung/ Compliance und Vorschriften

* Sicherstellen, dass die MFA-Implementierung mit den relevanten Vorschriften und Sicherheitsstandards der Umgebung verfügbar und zugreifbar sind

1. Skalierung und Hochverfügbarkeit

* Skalierung und Hochverfügbarkeit für die MFA-Einstellung, besonders für die kritischen Services, beachten

1. Benutzerunterstützung

* Nutzersupport für MFA-bezogene Probleme bereitstellen

Eigentliche Vorgehensweise:

1. Implementierungsmethode ausgewählt: Authelia

* Gründe stehen in der Zusammenfassung, warum Authelia und nicht einfaches MFA, SSO oder Sidecar patterns verwendet wurden

1. Docker-compose.yml für authelia erstellt, um diese auf dem reverse proxy einzuführen

* Grobe Struktur:

*project\_directory/*  
*|-- authelia/*  
*|   |-- docker-compose.yml*  
*|   |-- authelia\_config.yml*  
*|-- reverse\_proxy/*  
*|   |-- docker-compose.yml*  
*|   |-- nginx.conf*  
*|-- next-db/*  
*|   |-- docker-compose.yml*  
*|-- dev\_tools/infection-monkey/*  
*|   |-- // docker-compose.yml ??*  
*|-- dev\_tools/nexus/*  
*|   |-- docker-compose.yml*  
*|-- dev\_tools/sftpgo2/*  
*|   |-- docker-compose.yml*  
*|-- dev\_tools/sonarqube/*  
*|   |-- docker-compose.yml*  
*|-- docker\_host/*  
*|   |-- docker-compose.yml*  
*|-- git\_lab/*  
*|   |-- docker-compose.yml*  
*|-- mail\_server/mailcow-dockerized/*  
*|   |-- docker-compose.yml*  
*|-- monitoring\_tools/healthchecks/*  
*|   |-- docker-compose.yml*  
*|-- monitoring\_tools/prometheus-grafana/*  
*|   |-- docker-compose.yml*  
*|-- monitoring\_tools/uptime-kuma/*  
*|   |-- docker-compose.yml*  
*|-- netbox-docker/*  
*|   |-- docker-compose.yml*  
*|-- wordpress/*  
*|   |-- docker-compose.yml*

* <https://www.authelia.com/integration/proxies/nginx-proxy-manager/>

1. Configuration.yml und users\_databse.yml erstellt

* MariaDB und Redis sind mit angebunden
* <https://thehomelab.wiki/books/dns-reverse-proxy/page/setup-authelia-to-work-with-nginx-proxy-manager>
* In configuration.com: bei redirection: www.google.com statt mfa-test.tal-deloitte.de einfügen

One-factor anstatt two-factor einstellen

1. Login bei Authelia erfolgte erfolgreich
2. Konfigurationsdateien für Authelia und Nginx zusammenbauen

* authelia-location.conf, authelia-authrequest.conf, proxy.conf
* <https://www.authelia.com/integration/proxies/nginx/>
* <https://www.authelia.com/integration/proxies/nginx-proxy-manager/>

1. Evaluieren, ob LDAP oder OpenID Connect

* !! Noch nicht passiert !! Wahrscheinlich LDAP

1. Konfigurationsdateien dem Reverse Proxy hinzugefügt

* Error-Nachricht, als beim Proxy im Advanced-Bereich Content hinzugefügt wurde:

nginx: [emerg] "location" directive is not allowed here in /data/nginx/proxy\_host/authelia-location.conf:4

rev\_prox-app-1 | nginx: configuration file /etc/nginx/nginx.conf test failed

rev\_prox-app-1 |

rev\_prox-app-1 | [10/12/2023] [9:43:02 AM] [SSL ] › ℹ info Renewing SSL certs close to expiry...

rev\_prox-app-1 | [10/12/2023] [9:43:04 AM] [SSL ] › ✖ error Error: Command failed: /usr/sbin/nginx -t -g "error\_log off;"

rev\_prox-app-1 | nginx: [warn] the "user" directive makes sense only if the master process runs with super-user privileges, ignored in /etc/nginx/nginx.

* Kontent:

location / {

include proxy.conf;

proxy\_pass $forward\_scheme://$server:$port;

}

* [NGINX Proxy Manager - Integration - Authelia](https://www.authelia.com/integration/proxies/nginx-proxy-manager/)
* Changed directory from Content to:

location / {

include /snippets/proxy.conf;

proxy\_pass $forward\_scheme://$server:$port;

}

* Content komplett geändert:

set $upstream\_authelia http://192.168.2.2:9091/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-Original-Method $request\_method;

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;

}

1. In Nginx Proxy Manager bei Advanced Custom Nginx Configuration – ganze Configs übernehmen und Cookies setzen

Sidecar

* Wird an eine übergeordnete Anwendung angefügt und stellt unterstützende Funktionen für die Anwendung bereit
* Wird mit der Anwendung erstellt und außer Kraft gesetzt
* Vorteile
  + Gut geeignet für container
  + Zugang zu den gleich Ressourcen in der Hauptanwendung
  + Reduzierung von Code-Duplikationen in der Mikroservice-Architektur
* Disadvantages
  + Lots of communication between the containers which can be difficult to debug or detect issues in between inter process communication
  + Bei vielen/ mehreren Anwendungen kann die Leistung sinken
* Idee: mehrere Sidecars laufen zu lassen – einen auf Proxy, einen auf tal-cloud-infra und einen auf control-node

Authelia

* Open-Source-Authentifizierungs- und Authorisierungsserver, der mit 2-Faktor-Authentifizierung und Single-Sing-On arbeitet und Webanwendungen vor unbefugten Zugriffen schützt
* Arbeitet mit dem Reverse-Proxy zusammen
* Einmalanmeldung bei einem Dienst und automatisiert bei allen weiteren dann
* Nutzt Time-based (TOTP) und Universal-Second-Factor (U2F)
* Advantages
  + Password reset with identity verification
  + Several second factor methods
  + Compatible with Kubernetes ingress-nginx controller out of the box
* Disadvantages
  + Authelia passes Remote User HTTP header to the backend service – It's up to the service to link that to an account

MFA

* Nutzt die Kombination von zwei oder mehr Berechtigungsnachweisen für die Prüfung der Identität
* Sicherheit von Anmeldeverfahren lässt sich dank MFA deutlich erhöhen
* Authentifizierungsverfahren mit den zu verwendeten Faktoren basierend auf biometrischen Merkmalen, speziellem Wissen oder einem mitgeführten Gegenstand
* Vorteile
  + Identitätsdiebstahl wird erschwert, da Zugriff auf zweiten Faktor fehlt
* Nachteil
  + Einschränkung der Usability
  + Je mehr Faktoren bei der Anmeldung zu verwenden sind, desto aufwendiger und komplexer kann der Anmeldevorgang für den User werden

SSO

* Einmalanmeldung – Authentifizierungsverfahren
* Zugangsverfahren für multiple miteinander assoziierte, aber voneinander unabhängige Anwendungen, bei dem sich der Nutzer nur einmal anmelden muss
* Vorteile
  + Datensicherheit
* Nachteile
  + Lückenhafte Datensicherheit – SSO-Passwort falsche Hände, Arbeitsplatz verlassen