



WEB-SECURITY RINGS UM DIE ANMELDUNG

How to log in

Dr. Stefan Schlott (BeOne Stuttgart GmbH)

 [@_skyr](#)  @skyr@chaos.social

ABOUT.TXT

Be1ne
s t u t t g a r t

Stefan Schlott, BeOne Stuttgart GmbH

Java-Entwickler, statische-Sprachen-Fan, Linux-Jünger

Seit jeher begeistert für Security und Privacy

Aktuell im Projekt: Übersetzer Team ↔ Security

✉ stefan.schlott@beone-stuttgart.de

🐦 ~~@_skyr~~ 📺 @skyr@chaos.social





TEIL 1

Die Probleme



LOGINVORGANG

DER KLASSIKER

Login mit Username und Password

Bitte ans UI-Team: Passwortmanager-friendly!

ANBINDUNG ACCOUNTVERWALTUNG

Eigene Accountverwaltung

LDAP

OAuth / OIDC

E-MAIL \neq UID

Leute wechseln ihren (Nach)namen

Leute wollen ggfs. ihren Userhandle wechseln

Leute wechseln ihre Mailadresse

→ als UID intern synthetischen Identifier benutzen

BRUTEFORCE-ANGRIFFE

Systematisches Durchprobieren (geleakte Passwörter,
Dictionaries, etc.)

Account nach X versuchem sperren = Denial of Service

Rate Limiting (pro Username)

Ggfs. zusätzlich Captcha nach gewisser Anzahl Fehlversuche

PASSWORD POLICIES

Mindestlänge, Mindestkomplexität
(Länge sticht Sonderzeichen!)

Zwang zu häufigem Wechsel: Kein Sicherheitsgewinn

Abgleich mit "Top 100.000 Passwörtern"

Abgleich mit Infos über Leaks (z.B. haveibeenpwned.com)

ZWEITER FAKTOR



Besser als superstrikte Passwort-Regeln

Populärster Vertreter: TOTP (Time-based One-Time Password)

Eingabe einer (üblicherweise) sechsstelligen Zahl
Gültig für bestimmtes Intervall

Berechnung: $\text{Hash}(\text{Secret} \oplus \text{Start akt. Intervall})$

Erfordert (in etwa) synchrone Uhren

Übliche Toleranz: Eingabe aus vorigem und nächstem Intervall
auch akzeptieren

SPEICHERN VON PASSWÖRTERN

Prepare worst case (database leak)

→ nie im Klartext, nie ungesichert

Erster Schritt: Hash(Salt \oplus Password)

→ Identische Passwörter sehen „unterschiedlich“ aus

→ verhindert Angriff mit Rainbow Tables

Zweiter Schritt: Key derivation functions (z.B. scrypt, argon2id)

Wiederholtes Anwenden des Hashings, bremst Berechnung bei
Dictionary Attacks

PASSWORT-ALTERNATIVEN

(oder alternativer zweiter Faktor)

WebAuthn („Passkeys“)

Pairing: Hardware Token (oder Passwortmanager) generiert
Public-/Private-Key-Paar *für diese Domain*

Server erhält nur Public Key

Login: Server sendet Nonce und prüft Korrektheit der Signatur



SESSIONMANAGEMENT

SESSIONMANAGEMENT

Session: Verknüpfung zwischen ID und Login-Status, etc.

Vom Browser bei jedem Request mitgeliefert

Verschiedene Umsetzungsmöglichkeiten:

- Session-ID (typischerweise Cookie; assoziierte Info auf Server)
- Client-side sessions + HMAC (z.B. Play Framework)
- Json Web Tokens (JWT) - standardisiertes Format; client-side; Signatur per HMAC oder Public Key

ANGRIFFE AUF SESSIONS

Bei jeder Implementierung: Zugriff auf Session-Daten =
Zugriffsmöglichkeit auf aktuellen Zustand

Abgreifen der Session im Client via XSS: Cookie HttpOnly

XSRF-Angriffe (Aufrufe in andere Webseite einbetten): Cookie
SameSite=Strict

ANGRIFFE AUF SERVER-SIDE SESSIONS



Session-ID: Nur ein opaker Identifier (Rest liegt auf Server)

Vorhersage der Session-ID: „Guter“ Zufall für die Erzeugung

SESSION FIXATION ATTACK: VORGABE DER SESSION-ID

Manchmal Fallback ohne Cookies: Übergabe in URL

```
<a href="./helloservlet;jsessionid=F3F89FD58687DF41C218831E843AB4E2">...
```

User klickt (Phishing/Social engineering) und loggt sich dann ein: Angreifer kennt Session-ID!

Abhilfe: Deaktivieren oder bei Login neue Session erzeugen

MEHRFACH-LOGINS

Naive Implementierung: Ist möglich. Ist das gewünscht?

Was mit vergessenem Logout?

Best practice / ideal wäre:

- Liste aller offenen Sessions
- Gezieltes Beenden einzelner oder aller Sessions

LOGOUT

Server-Side Session: Einfach invalidieren

Problem Client-Side Sessions: Ablaufdatum fix „eingebakken“

Sofortiger Logout? → Liste offener Sessions + Blacklist führen

Account deaktivieren? → Blacklist führen



PASSWORTWECHSEL UND -RECOVERY

PASSWORTWECHSEL

Beendet nicht automatisch alle anderen Sitzungen

Hat keine Auswirkungen auf „Login-Tokens“ (generierte
Passwörter für Anwendungen, die keine 2FA durchführen
können)

→ sollte das gewünscht sein: Muss gezielt implementiert
werden

PASSWORT-RECOVERY

Klassiker: Passwort-Rücksetzungs-Link per Mail

Nie Passwörter per Mail verschicken!

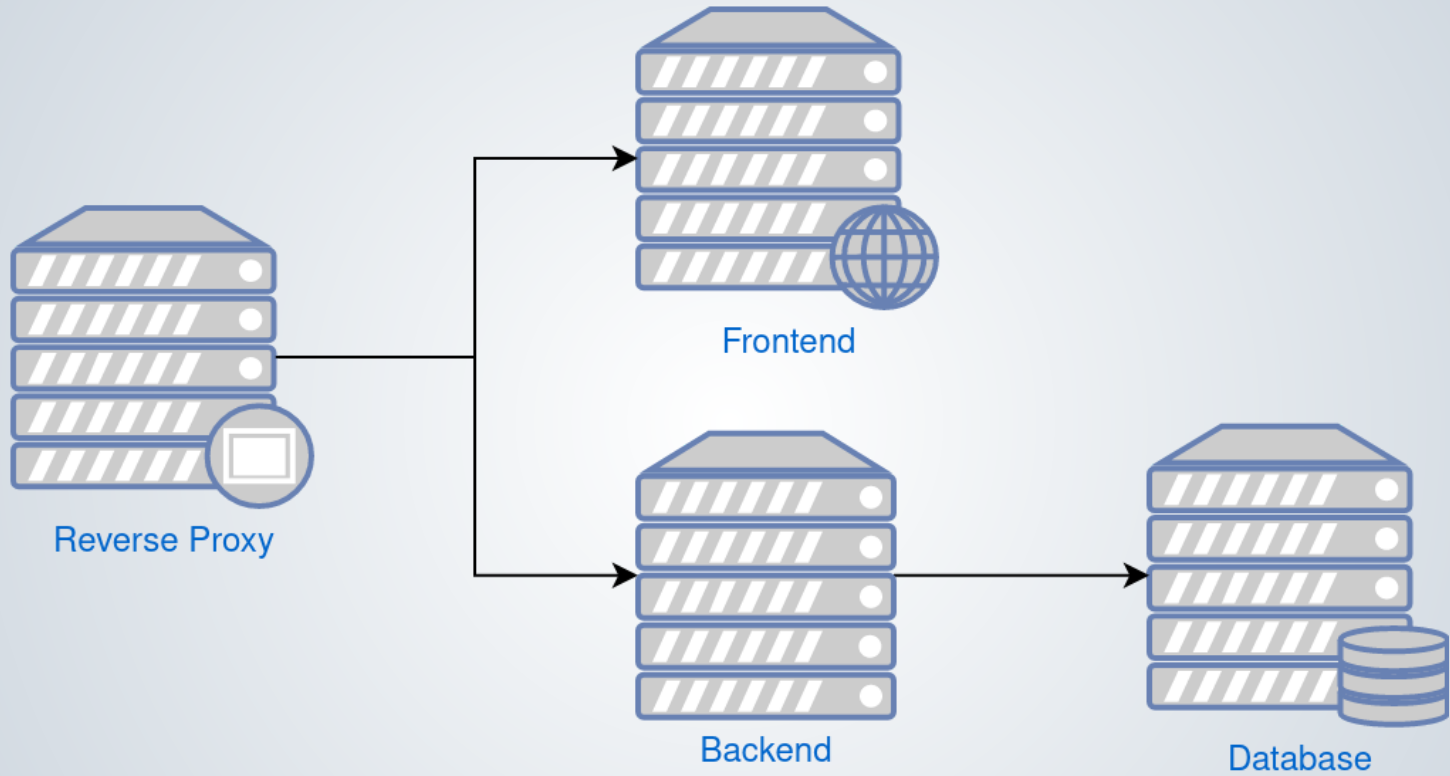
Problem: Lädt die Sicherheit auf den Mailzugriff ab. Zugang zum Postfach + Rücksetzungs-Link = Account-Übernahme

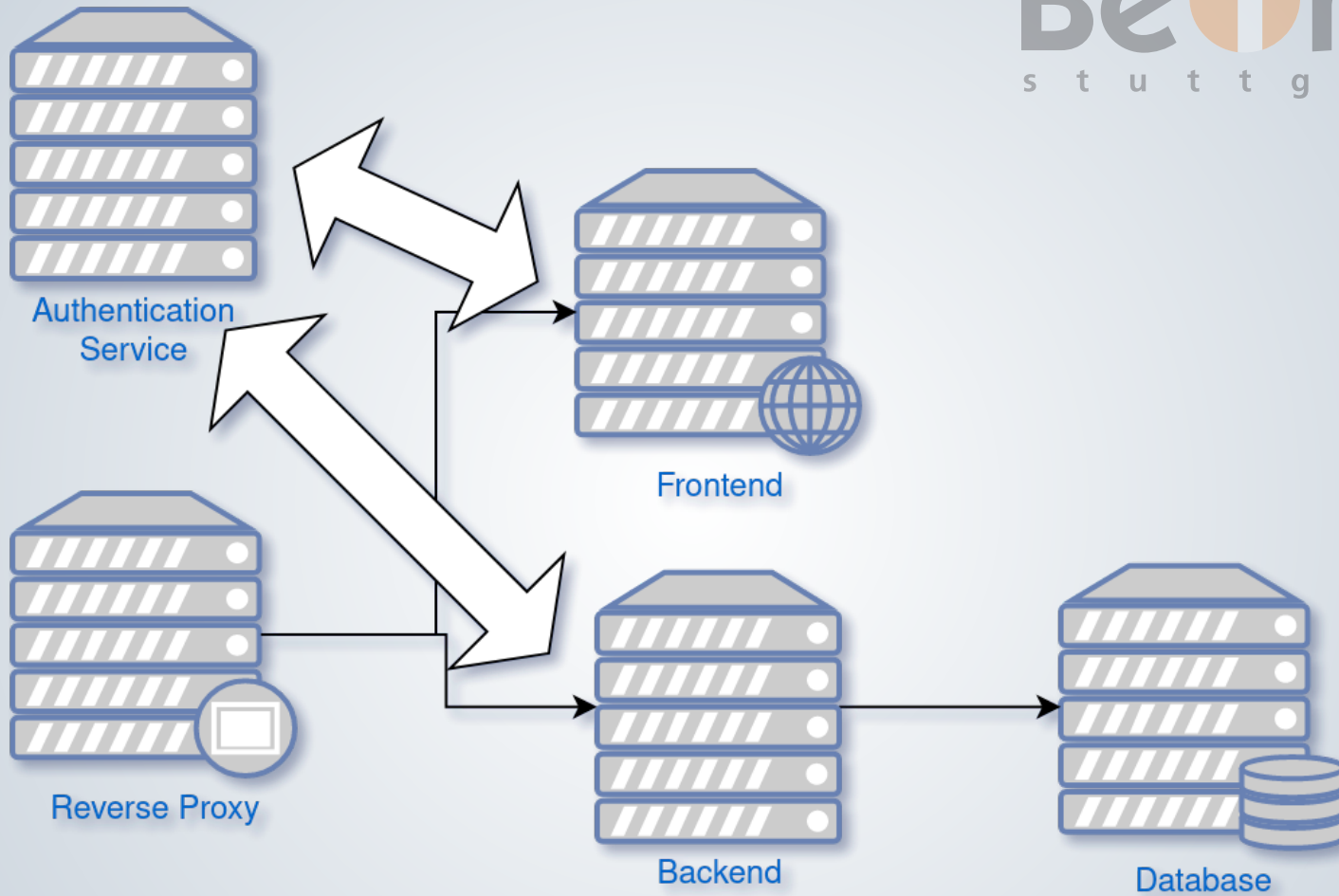
Tradeoff zu Hotline-Aufwand (auch hier muss die Prozedur sauber definiert werden!)




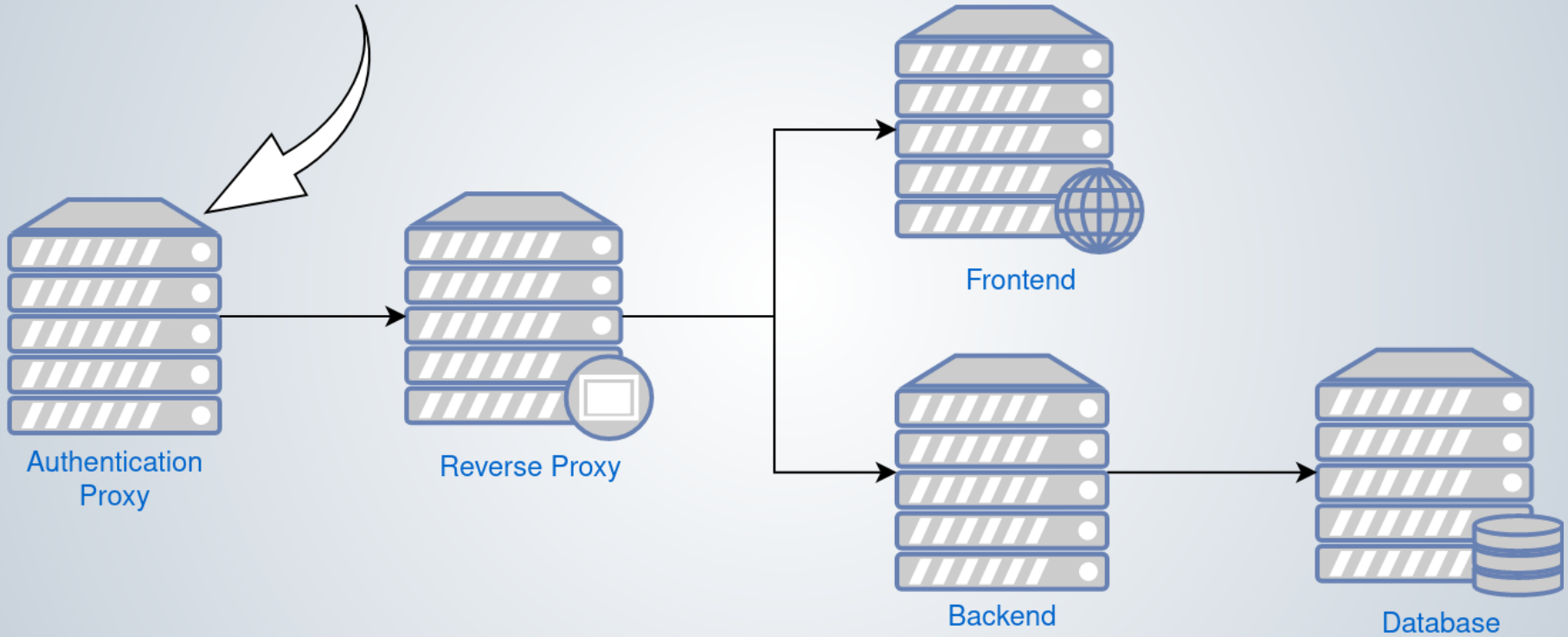
TEIL 2

Alternativen zur
Eigenimplementierung





Authentication info
added here 





IDENTITY PROVIDER

IDENTITY PROVIDER

„All-in-one“-Lösung für Authentisierung

- Management User-Accounts (eigene DB oder Backend-Anbindung)
- Passwort-Policies
- Login, Passwort-Rücksetzung
- Gruppen-Zuordnung
- Versch. Integrations-Möglichkeiten: OIDC, OAuth, Proxy, ...

Produkte: z.B. Keycloak, Authentik, Authelia, Zitadel, ...

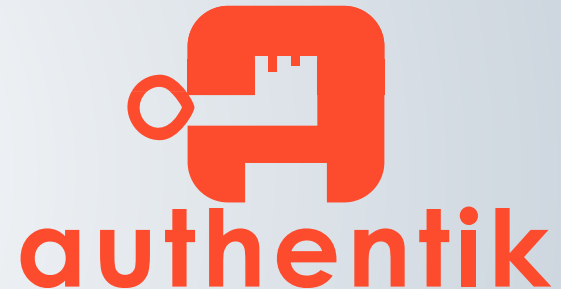
BEISPIEL: AUTHENTIK

Alternative zum „Platzhirsch“ Keycloak

Python-Implementierung

Opensource-Version: Authentisierung via
OIDC, SAML, LDAP, SCIM, Radius, Proxy

„Outposts“: Ausgelagerter Proxy vor Anwendung



Anwendung mittels Wizard hinzufügen

New application

Create a new application

- 1 Application Details
- 2 Provider Type
- 3 Provider Configuration
- 4 Submit Application

Name *
Application's display Name.

Slug *
Internal application name used in URLs.

Group
Optionally enter a group name. Applications with identical groups are shown grouped together.

Policy engine mode * any
Any policy must match to grant access

all
All policies must match to grant access

> UI Settings

New application

Create a new application

- 1 Application Details
- 2 Provider Type**
- 3 Provider Configuration
- 4 Submit Application

- OAuth2/OIDC (Open Authorization/OpenID Connect)
Modern applications, APIs and Single-page applications.
- LDAP (Lightweight Directory Access Protocol)
Provide an LDAP interface for applications and users to authenticate against.
- Transparent Reverse Proxy
For transparent reverse proxies with required authentication
- Forward Auth (Single Application)
For nginx's auth_request or traefik's forwardAuth
- Forward Auth (Domain Level)
For nginx's auth_request or traefik's forwardAuth per root domain
- Remote Access Provider
Remotely access computers/servers via RDP/SSH/VNC

! This feature requires an enterprise license. [Learn more](#)
- SAML (Security Assertion Markup Language)
Configure SAML provider manually

Back

Next

Cancel

New application

Create a new application

- 1 Application Details
- 2 Provider Type
- 3 Provider Configuration
- 4 Submit Application

Configure Proxy Provider

This provider will behave like a transparent reverse-proxy, except requests must be authenticated. If your upstream application uses HTTPS, make sure to connect to the outpost using HTTPS as well.

Name *

Authentication flow
Flow used when a user access this provider and is not authenticated.

Authorization flow *
Flow used when authorizing this provider.

External host *
The external URL you'll access the application at. Include any non-standard port.

Internal host *
Upstream host that the requests are forwarded to.

Internal host SSL Validation

Anwendung in Outpost aktivieren

Update Outpost

Name *

authentik Embedded Outpost

Type *

Proxy

Integration

Selecting an integration enables the management of the outpost by authentik.

See [documentation](#).

Applications

Available Applications



http echo ✓
Provider for http echo

Selected Applications



> 1 item(s) selected.

>> http echo
Provider for http echo



> Advanced settings

Update

Cancel

Falls noch nicht geschehen: Hostname von Authentik selbst setzen

Update Outpost ×

▼ Advanced settings

Configuration

```
1 log_level: info
2 docker_labels: null
3 authentik_host: http://authentik.localhost:9000
4 docker_network: null
5 container_image: null
6 docker_map_ports: true
```

← → ↻ 🔍 httppecho.localhost:9000/foo/bar

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
path: "/foo/bar"
▼ headers:
  host: "httppecho.localhost:9000"
  ▶ user-agent: "Mozilla/5.0 (X11; Linux .../20100101 Firefox/126.0)"
  ▶ accept: "text/html,application/xhtml+xml,image/webp,*/*;q=0.8"
    accept-language: "de-DE,en-US;q=0.7,en;q=0.3"
  ▶ cookie: "authentik_proxy_btctbhVL...PTFUGY0GKL3ANHXAA0HVHEY"
  dnt: "1"
  priority: "u=1"
  sec-fetch-dest: "document"
  sec-fetch-mode: "navigate"
  sec-fetch-site: "none"
  sec-fetch-user: "?1"
  sec-gpc: "1"
  ▼ sentry-trace: "c90f1a2ffa14c257542cf8d260c0eb95-8061ce4ebde9c710-0"
  upgrade-insecure-requests: "1"
  x-authentik-email: "stefan.schlott@beone-stuttgart.de"
  x-authentik-groups: "authentik Admins"
  ▼ x-authentik-jwt: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwOi8vYXV0aGVudGlrLmxvY2FsaG9zd...
  x-authentik-meta-app: "http-echo"
  ▼ x-authentik-meta-jwks: "http://authentik.localhost:9000/application/o/http-echo/iwks/"
  x-authentik-meta-outpost: "authentik Embedded Outpost"
  x-authentik-meta-provider: "Provider for http echo"
  x-authentik-meta-version: "goauthentik.io/outpost/2024.2.2"
  x-authentik-name: "authentik Default Admin"
  ▼ x-authentik-uid: "28abc7065d32a126d6a9289501b7fe1c26a65d444662daa54fb194e7b3d13489"
  x-authentik-username: "akadmin"
```

INTEGRATION ANWENDUNG

Auswertung der Header des Proxys

Mapping auf Authentisierungs-Framework der Anwendung

Prüfen, dass Header nicht von außen gefälscht werden kann ;-)

BEISPIEL QUARKUS



```
IdentityAuthenticationMechanism: HttpAuthenticationMechanism {
  authenticate(context: RoutingContext?, identityProviderManager: IdentityProviderManager?
context?.request()?.headers()?.contains("X-Authentik-Username") == true) {
  username = context.request().getHeader("X-Authentik-Username")
  header is present: Create non-anonymous identity
  builder = QuarkusSecurityIdentity.builder()
    .setAnonymous(false)
    .setPrincipal(QuarkusPrincipal(username))

  add groups as roles
  groups = context.request().getHeader("X-Authentik-Groups")?: ""
  groups.isNotEmpty()) {
  groups.split('|').forEach { group ->
    builder.addRole(group)

  return Uni.createFrom().item(builder.build())

  return Uni.createFrom().optional(Optional.empty())
```

BEISPIEL QUARKUS

```
@Path("/hello")
class GreetingResource {
    @Path("/user")
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    @RolesAllowed("quarkus-users")
    fun helloUser() = "Hello quarkus user!"
}
```



AUTHENTICATION PROXY

AUTHENTICATION PROXY

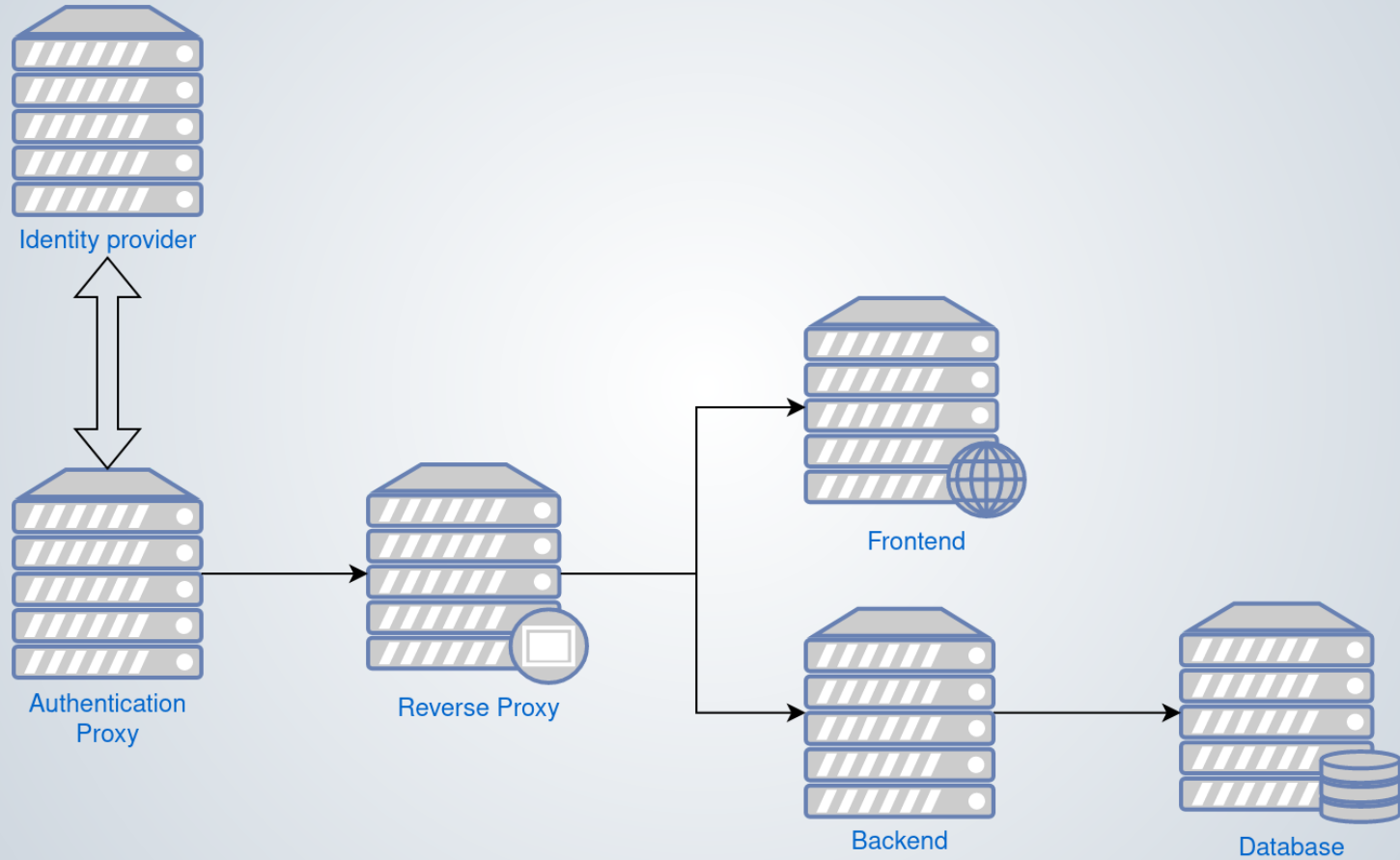
„One-trick pony“: Erledigt die Authentisierung
(bei Identity Provider)

Ergebnis wird an dahinterliegende Anwendung weitergereicht

Produkte erledigen häufig noch mehr: API rate limiting,
Logging/Metrics, etc.

z.B. Apache APISIX, Kong, Krakend, Apiman, Tyk, ...

AUTHENTICATION PROXY



BEISPIEL: APACHE APISIX

API Gateway: Mehr als nur Authentisierung

Load balancing, Canary release, Metriken, ...

Lua-Implementierung

„Spricht“ OIDC (im „Platzhirsch“ Kong erst im Bezahl-Tier oder mit einigem Basteln)



PROVIDER IN AUTHENTIK ANLEGEN

New provider ✕

Create a new provider.

1 Select type

2 Create OAuth2/
OpenID Provider

- LDAP Provider
Allow applications to authenticate against authentik's users using LDAP.
- OAuth2/OpenID Provider
OAuth2 Provider for generic OAuth and OpenID Connect Applications.
- Proxy Provider
Protect applications that don't support any of the other Protocols by using a Reverse-Proxy.

PAC Provider

New provider

Create a new provider.



1 Select type

2 Create OAuth2/
OpenID Provider

Name *

Authentication flow

Flow used when a user access this provider and is not authenticated.

Authorization flow *

Flow used when authorizing this provider.

▼ Protocol settings

Client type *

Confidential

Confidential clients are capable of maintaining the confidentiality of their credentials such as client secrets

Public

Public clients are incapable of maintaining the confidentiality and should use methods like PKCE.

Client ID *

Client Secret

Finish

Back

Cancel

ZUGEHÖRIGE APPLIKATION

Be1ne

s t u t t g a r t

Create Application

Name *

Application's display Name.

Slug *

Internal application name used in URLs.

Group

Optionally enter a group name. Applications with identical groups are shown grouped together.

Provider

Select a provider that this application should use.

Backchannel Providers **+**

Select backchannel providers which augment the functionality of the main provider.

Policy engine mode * any

Any policy must match to grant access

all

All policies must match to grant access

> UI settings

Create

Cancel

APISIX-KONFIGURATION



```
authecho.localhost
*
am:
s:
httpecho:8080": 1
: roundrobin
s:
oid-connect:
ient_id: ELWlePl9ZEK4A8kFGelyd7k8fch6ZkGULfHkKFah
ient_secret: ss9KJoAPf99m55H5S6toHwAPxx22...
scovery:
http://authentik-server-1:9000/application/o/apisix-httpecho/.well-known/openid-configuration
direct_uri: http://authecho.localhost:9080/success
ope: openid email profile
ccess_token_in_authorization_header: true
ssion:
secret: verysecretsessionkey
```

access_token_in_authorization_header leitet JWT-Token an Anwendung weiter



FAZIT

FAZIT

Thema mit hoher Komplexität

„Outsourcing“ in Produkt: Komplexität verschwindet nicht -
Korrektheit prüfen!

Themen „abseits der Technik“ nicht vernachlässigen

Interessante Alternative zu integrierten Authentisierungs-
Frameworks