

Protectors of the Realm: Wie man einen Keycloak sicher hält

Max Maaß, Tim Walter

9.12.2025 | OWASP Stammtisch HH

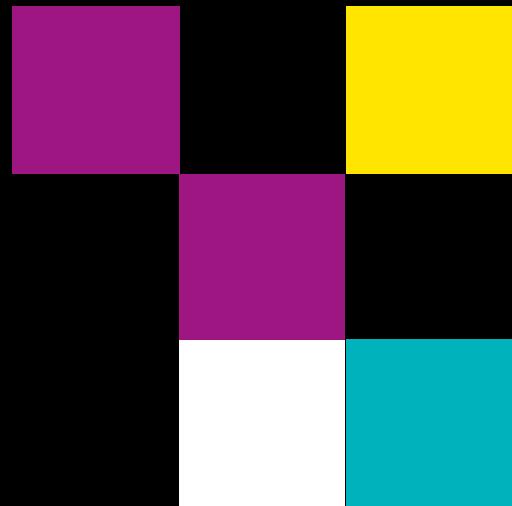


Es war einmal...



**Wer musste schon einen Keycloak
konfigurieren?**

Wer hat sich dabei sicher gefühlt?

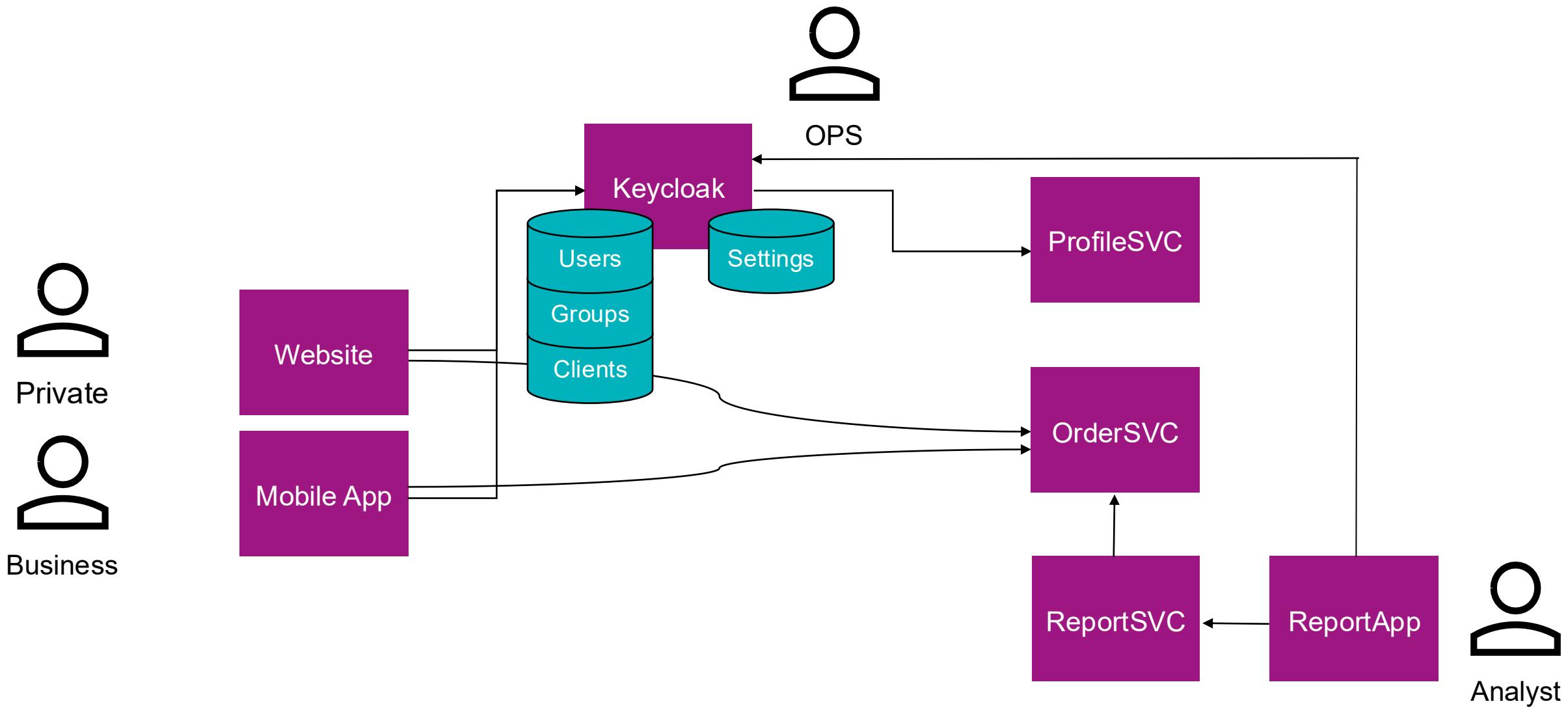


Schritt 1: Einen Überblick bekommen

- Welche Systeme sind beteiligt?
- Welche Keycloak-Features nutzen sie?
- Was für Gruppen von Nutzer:innen gibt es? Interne, externe, ...?
- Was ist der fachliche Kontext? Welche Geschäftsprozesse sind betroffen?
- Wer betreibt und konfiguriert den Server?
- Sind Erweiterungen (SPIs) installiert? Was tun sie?
- ...



Schritt 1: Einen Überblick bekommen

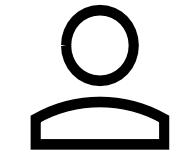


Schritt 2: Das Große zuerst angehen

- Prüfen der Keycloak-Version
- Probleme mit der Konfiguration
 - Viele Funktionen und Konfigurationsmöglichkeiten
 - Unsichere Standardwerte
 - Konfiguration wird meist manuell geändert



Hintergrund: OAuth 2



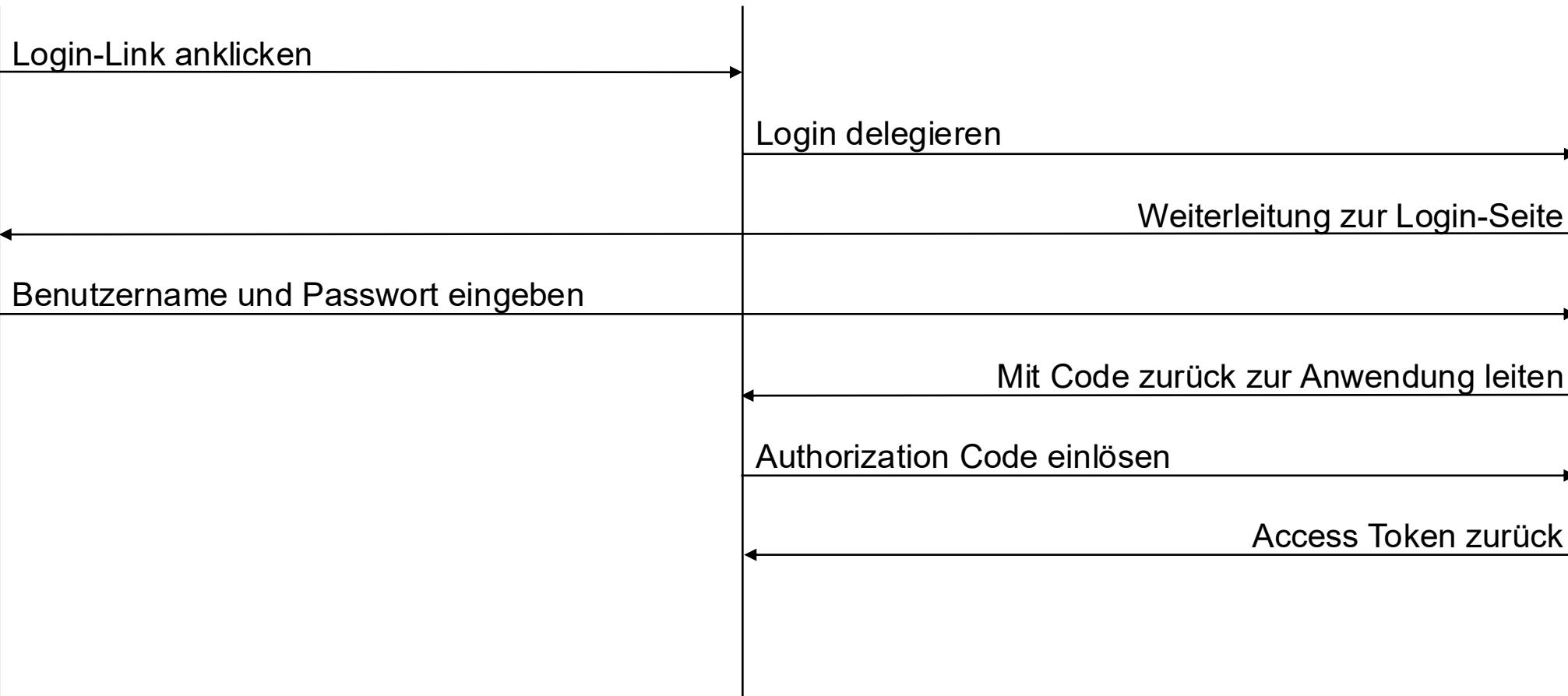
Nutzer:in



Anwendung



Authorization Server



Beispiel #1: Gültige Redirect URIs

Access settings

Access settings

Root URL <https://veilshire.cloakeyrion.kingdom>

Home URL <https://veilshire.cloakeyrion.kingdom/town-gate>

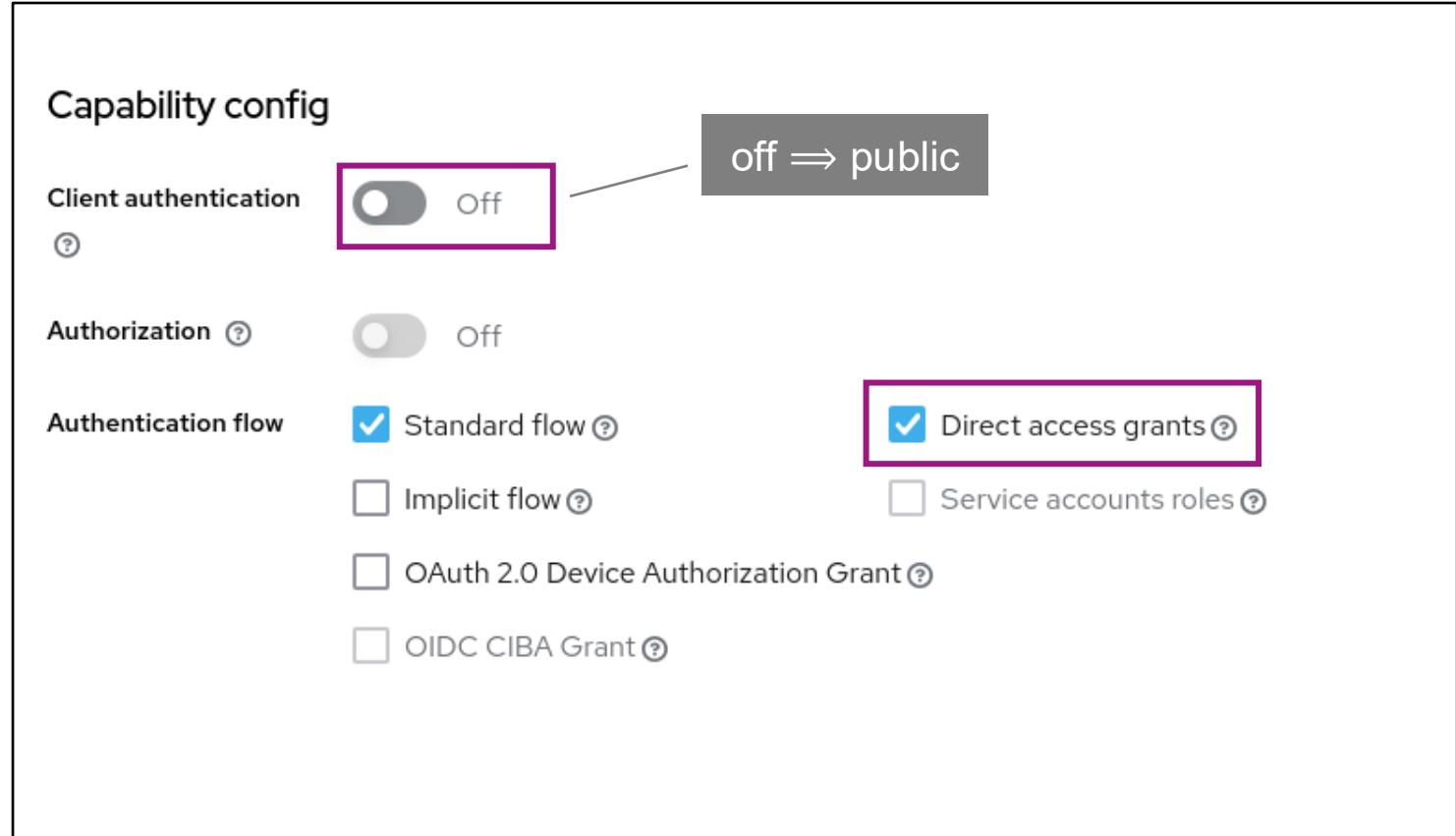
Valid redirect URIs https://veilshire.cloakeyrion.kingdom/* 

<https://veilshire.cloakeyrion.kingdom.villain.evil/trap>

Beispiel #2: Direct Access Grants

- Legacy OAuth 2.0 Password Grant Type
- Direktes Senden von Benutzername und Passwort an Keycloak für einen Access-Token
- Umgehen von Security-Mechanismen wie der Prüfung von Redirect-URLs
- Erleichtert Brute-Forcing oder Phishing

Sollte mindestens für public Clients ausgeschaltet sein!



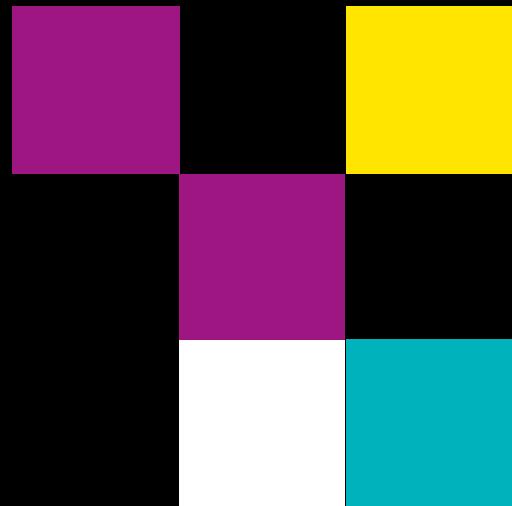
Weitere versteckte Gefahren

The screenshot displays several configuration panels from a web-based management interface:

- Capability config**: Shows client authentication and authorization toggles set to "Off". Under "Authentication flow", "Standard flow" and "Implicit flow" are selected. A purple box highlights the "Implicit flow" checkbox and the "Full scope allowed" toggle switch, both of which are turned on.
- Advanced settings**: A section for configuring advanced settings related to the OpenID Connect protocol. It includes fields for "Access Token Lifespan" (set to "Expires in 365 Days"), "Client Session Idle" (set to "Inherits from realm settings"), and "Client Session Max" (set to "Inherits from realm settings").
- the-holy-realm**: A realm settings page. The "Security defenses" tab is active. Under "Brute force detection", the "Brute Force Mode" is set to "Disabled".
- Authentication**: A policy configuration page. The "Policies" tab is active. It shows tabs for "Flows", "Required actions", and "Policies". Under "Policies", there are tabs for "Password policy", "OTP Policy", "Webauthn Policy", "Webauthn Passwordless Policy", and "CIBA Policy". A dropdown menu "Add policy" is open. A purple box highlights the "Hashing Iterations" input field, which is set to "50000".

Wer will das per Hand prüfen?

**Für jeden Client, jede Gruppe, jeden
Scope, ...?**





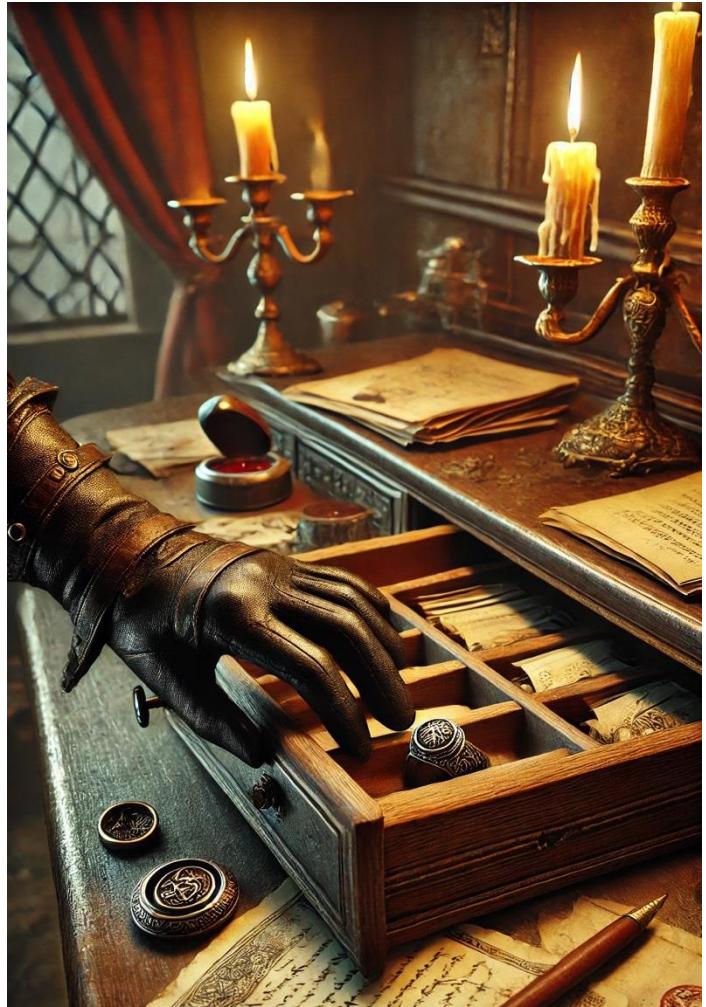
\$ poetr

Unser Tool: kcwarden

- Open Source
- Untersucht Keycloak-Konfigurationen automatisch auf Sicherheitsprobleme
- Konfigurierbar – einzelne Regeln können jederzeit stummgeschaltet werden



Schritt 3: Wovor haben wir Angst?



Beispiel #1: M2M – Tech User vs Service Account

Technischer User

- Normaler Nutzeraccount
- Kann durch das Eingeben falscher Passwörter gesperrt werden
- Benutzt Direct Access Grants

Risiko für die Verfügbarkeit

towngate-golem

Details Credentials Role mapping Groups Consents Identity provider links Sessions

Search by name → Hide inherited roles Assign role Unassign Refresh

<input type="checkbox"/> Name	Inherited
<input type="checkbox"/> gate-opener	False

Service Account

- Spezieller Account, der an einem Client hängt
- Benötigt confidential Client
- Access Token wird via Client ID + Secret abgerufen

town-gate (OpenID Connect)

Clients are applications and services that can request authentication of a user.

Settings Keys Credentials Roles Client scopes Service accounts roles Sessions Advanced

To manage detail and group mappings, click on the username service-account-town-gate

Search by name → Hide inherited roles Assign role Unassign Refresh

<input type="checkbox"/> Name	Inherited
<input type="checkbox"/> gate-opener	False

Beispiel #2: User Attributes

- Zusätzliche Metadaten, die am Account hängen
- Z.B. Referenzen auf IDs in anderen Systemen
- Je nach Konfiguration: Ggf. Durch die Nutzer:innen selbst editierbar!

Accountübernahme möglich

The screenshot shows a user profile page with the following details:

- Header:** Users > User details
- User Email:** lyra-the-bard@veil.shire
- Tab Navigation:** Details (selected), Attributes, Credentials, Role mapping, Groups, Consents, Identity provider links, Sessions
- Attribute Table:**

Key	Value	Remove
soulmark	13	-
craft	bardic arts	-
- Buttons:** + Add attributes, Save, Revert

Schritt 4: Patrouillen aussenden

- Konfigurationen ändern sich mit der Zeit
- Leitplanken können gegen ungewünschte Konfigurationsänderungen (z.B. Rollen-Zuweisungen) helfen
- kcwarden unterstützt dies nativ:

```
1 - monitor: ServiceAccountWithSensitiveRole
2   config:
3     - role: ring_carrier
4       role-client: realm
5       severity: Critical
6       allowed:
7         - service-account-frodo
8       note: Do not give it to Boromir under any circumstances!
```

- Details in der Dokumentation



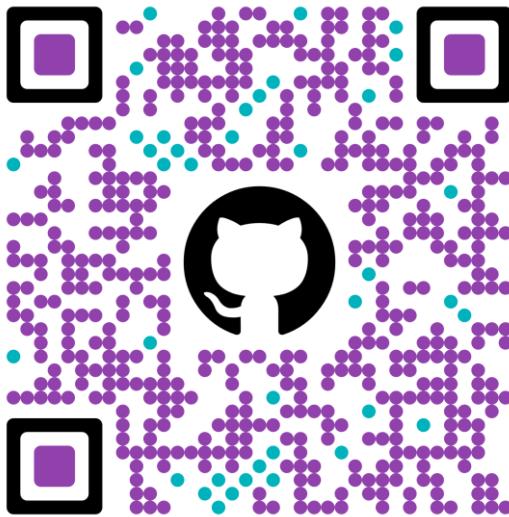
Anwendungsbeispiel: Kontinuierliches Monitoring

- Erstellt eine Konfiguration mit projektspezifischen Regeln
- Lasst diese Regeln automatisch regelmäßig prüfen
- Z.B. in einer scheduled Pipeline in GitLab

```
kcwarden:  
  stage: audit  
  image: ghcr.io/iteratec/kcwarden:latest  
  
  variables:  
    KCWARDEN_KEYCLOAK_PASSWORD: ${KEYCLOAK_ADMIN_PASSWORD}  
  
  script:  
    - echo "🚀 Running kcwarden"  
    - >-  
      kcwarden download -r "${KEYCLOAK_REALM}" -m password  
      -u "${KEYCLOAK_ADMIN_USERNAME}"  
      "${KEYCLOAK_URL}" -o "${KEYCLOAK_CONFIG_FILE}"  
    - >-  
      kcwarden audit --ignore-disabled-clients --fail-on-findings  
      --config "./kcwarden-config.yaml" "${KEYCLOAK_CONFIG_FILE}"
```

Fazit

- Es ist kompliziert, einen Keycloak sicher zu halten
- Die Probleme können in den grundlegenden Einstellungen, aber auch in komplexen Interaktionen mit Umsystemen liegen
- kcwarden hilft dabei, eine sichere Baseline zu schaffen, und erlaubt einen Fokus auf die komplizierteren Fälle
- Das System ist erweiterbar – schickt gerne Pull Requests mit neuen Regeln :)



kcwarden auf GitHub:

 /iteratec/kcwarden



Dr. Max Maaß

max.maass@iteratec.com
@hacksilon@infosec.exchange



Tim Walter

tim.walter@iteratec.com
@twwd@infosec.exchange