

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?

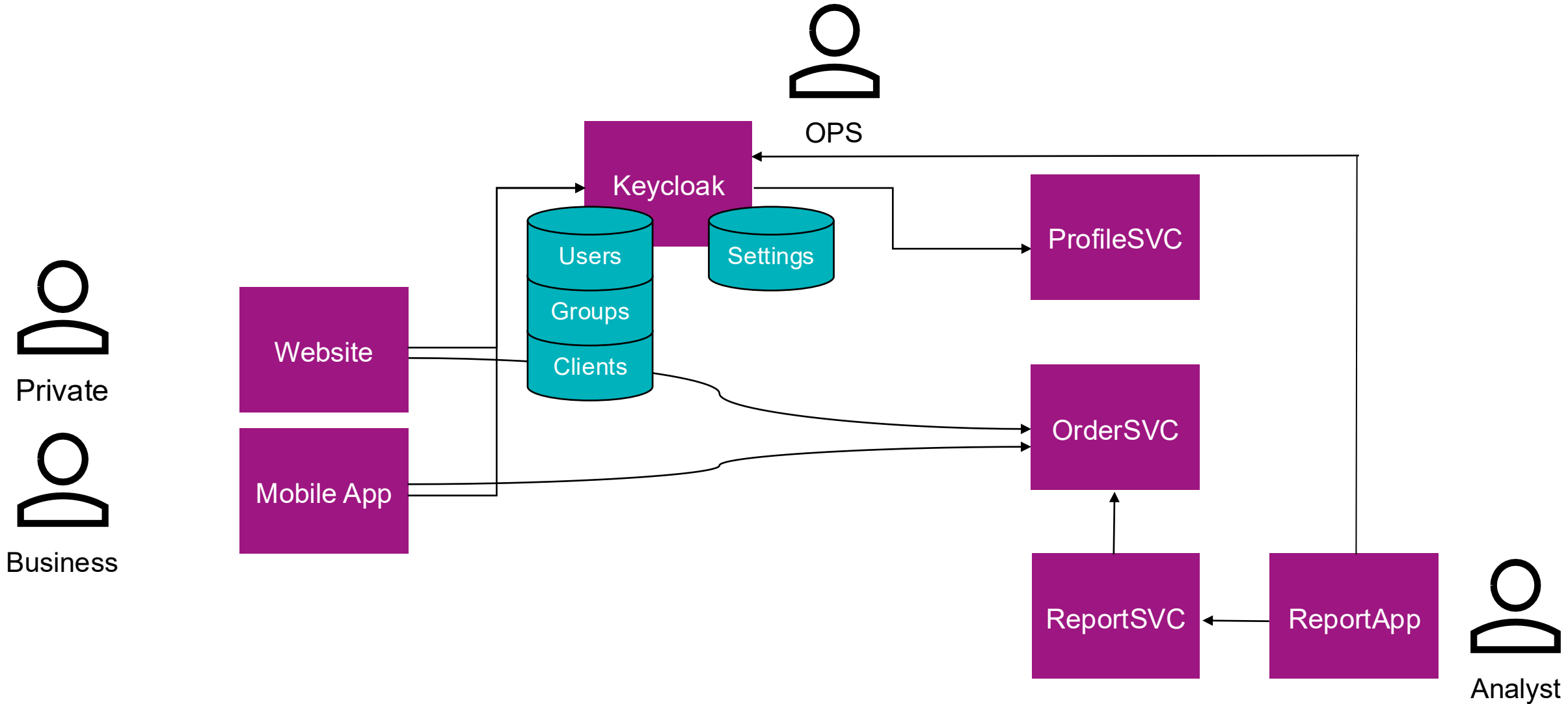


Schritt 1: Einen Überblick bekommen

- 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

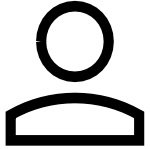


Schritt 2: Das Große zuerst angehen

- 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

Login-Link anklicken

Login delegieren

Weiterleitung zur Login-Seite

Benutzername und Passwort eingeben

Mit Code zurück zur Anwendung leiten

Authorization Code einlösen

Access Token zurück

Beispiel #1: Gültige Redirect URIs

Access settings

Access settings

Root URL 

Home URL 

Valid redirect URIs 



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

Beispiel #2: Direct Access Grants

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

Sollte mindestens für public Clients ausgeschaltet sein!

Capability config

Client authentication ? ☐ Off off ⇒ public

Authorization ? ☐ Off

Authentication flow

☒ Standard flow ? ☒ Direct access grants ?

☐ Implicit flow ? ☐ Service accounts roles ?

☐ OAuth 2.0 Device Authorization Grant ?

☐ OIDC CIBA Grant ?

Weitere versteckte Gefahren

Capability config

Client authentication ☐ Off

Authorization ☐ Off

Authentication flow ☒ Standard flow ☒ Implicit flow ☐ OAuth 2.0 De

town-gate-dedicated

This is a client scope which includes the de

Mappers Scope

Full scope allowed ☒ On

Advanced settings

This section is used to configure advanced settings of this client related to OpenID Connect protocol

Access Token Lifespan

Client Session Idle

Client Session Max

the-holy-realm

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more](#)

General Login Email Themes Keys Events Localization Security defenses Sessions Tokens Client polici

Headers Brute force detection

Brute Force Mode

Save Revert

Proof Key for Code Exchange Code Challenge Method

Authentication

Authentication is the area where you can configure and manage different credential types. [Learn more](#)

Flows Required actions Policies

Password policy OTP Policy Webauthn Policy Webauthn Passwordless Policy CIBA Policy

Add policy

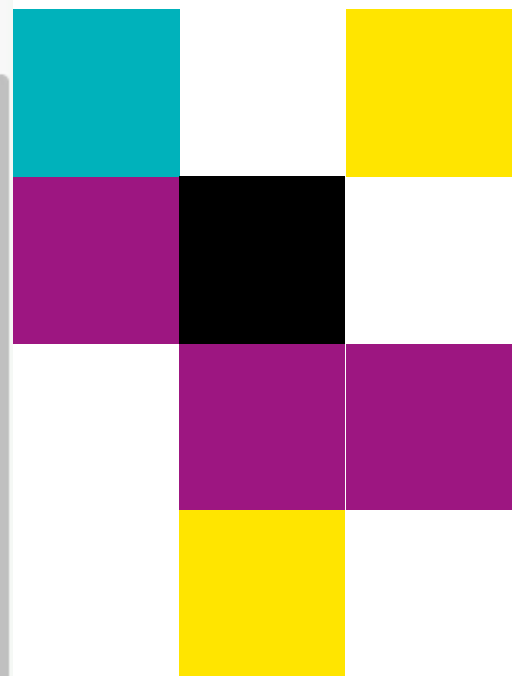
Hashing Iterations *

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

[i](#) 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**

[Users](#) > User details

lyra-the-bard@veil.shire

Details

Attributes

Credentials

Role mapping

Groups

Consents

Identity provider links

Sessions

Key	Value	
soulmark	13	⊖
craft	bardic arts	⊖

[+ Add attributes](#)

Save

Revert

Schritt 4: Patrouillen aussenden

- Konfigurationen ändern sich mit der Zeit
- Leitplanken können gegen ungewünschte Konfigurationsänderungen (z.B. Rollen-Zuweisungen) helfen
- kwarden 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

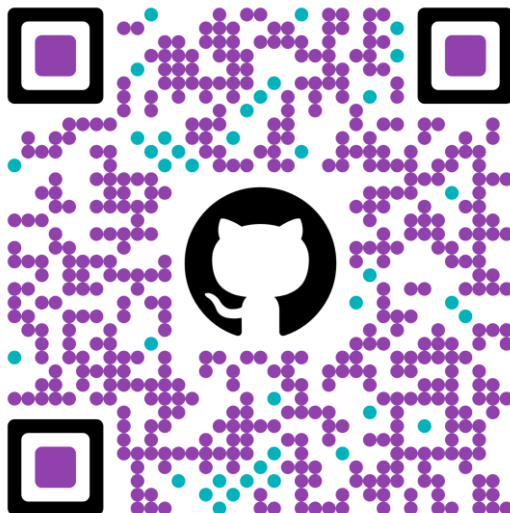
```
kcwarden:
  stage: audit
  image: ghcr.io/iteratec/kcwarden:latest

  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 Umssystemen 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