Protectors of the Realm: Wie man einen Keycloak sicher hält

Max Maaß, Tim Walter

9.12.2025 | OWASP Stammtisch HH

Es war einmal...

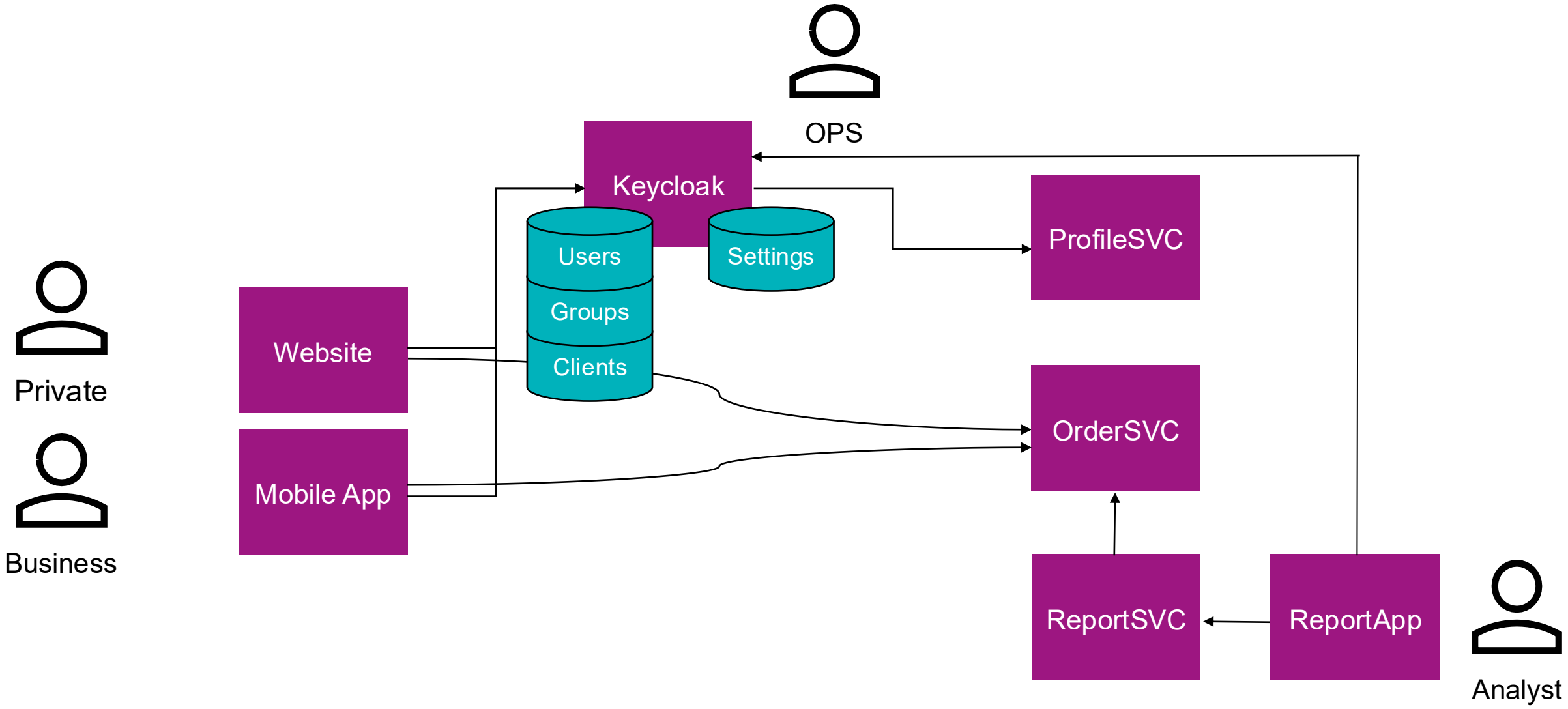
Wer musste schon einen Keycloak konfigurieren?

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

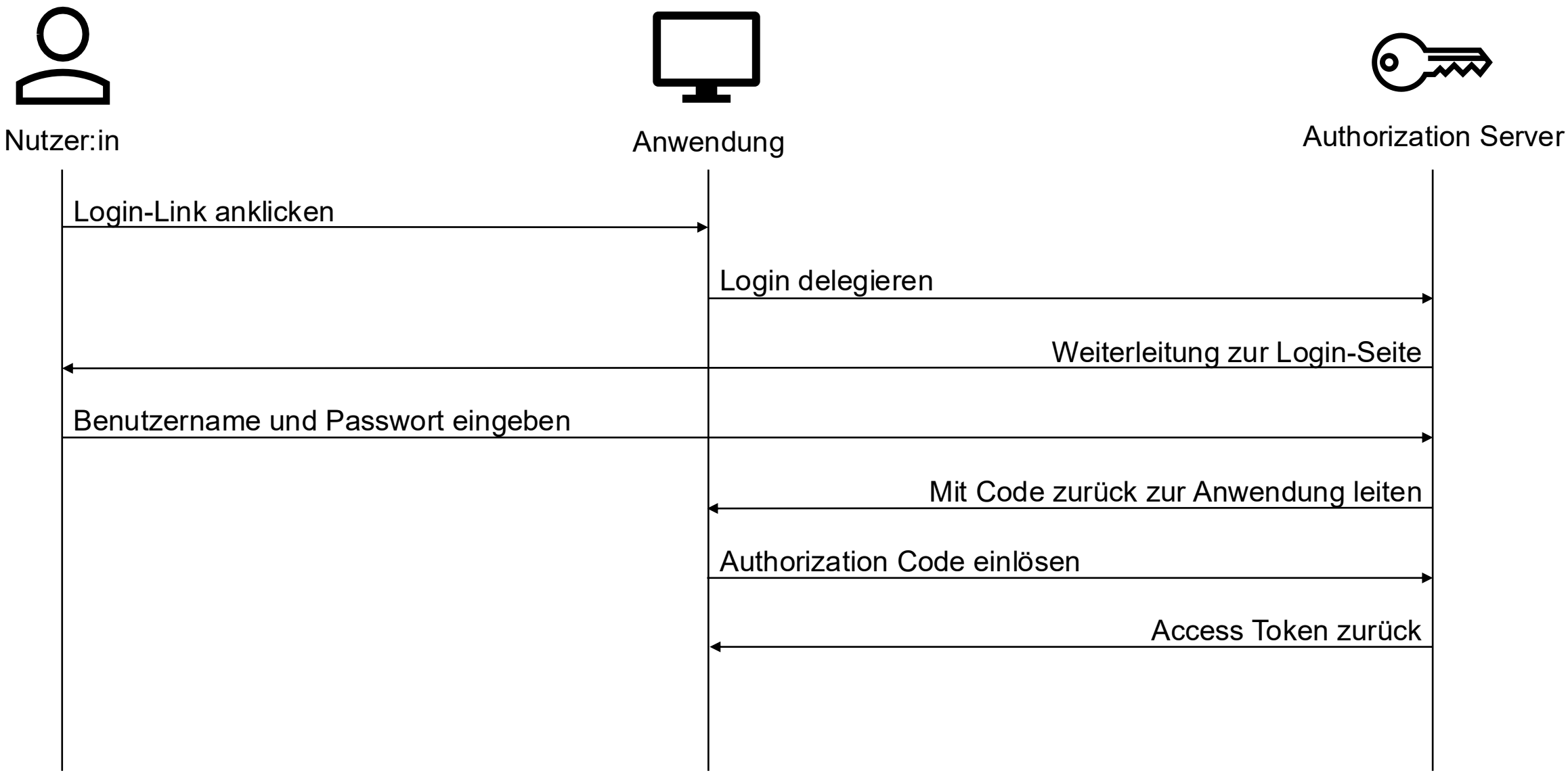


Schritt 2: Das Große zuerst angehen

- Prüfen der Keycloak-Version

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

Hintergrund: OAuth 2

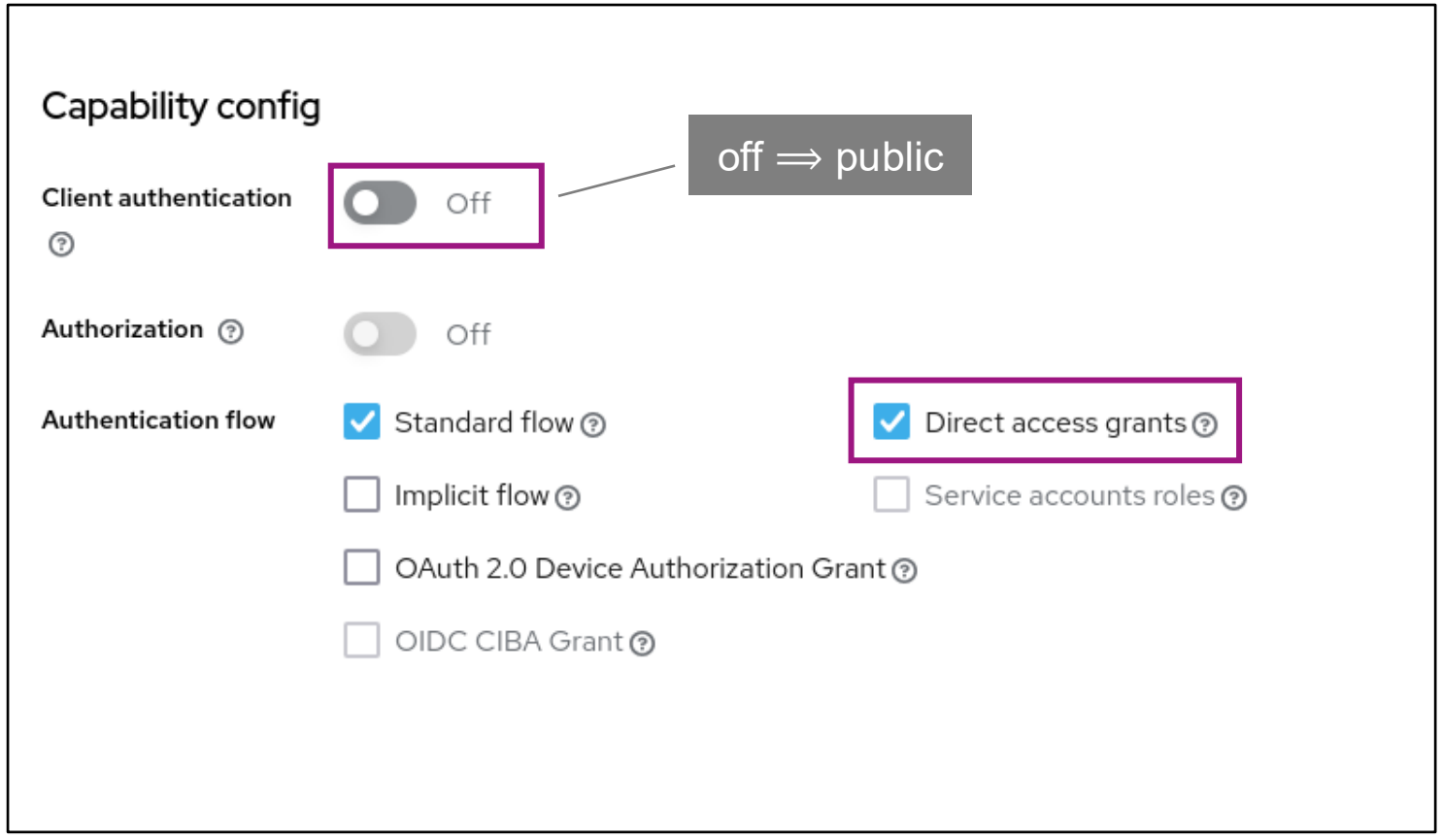


Beispiel #1: Gültige Redirect URIs

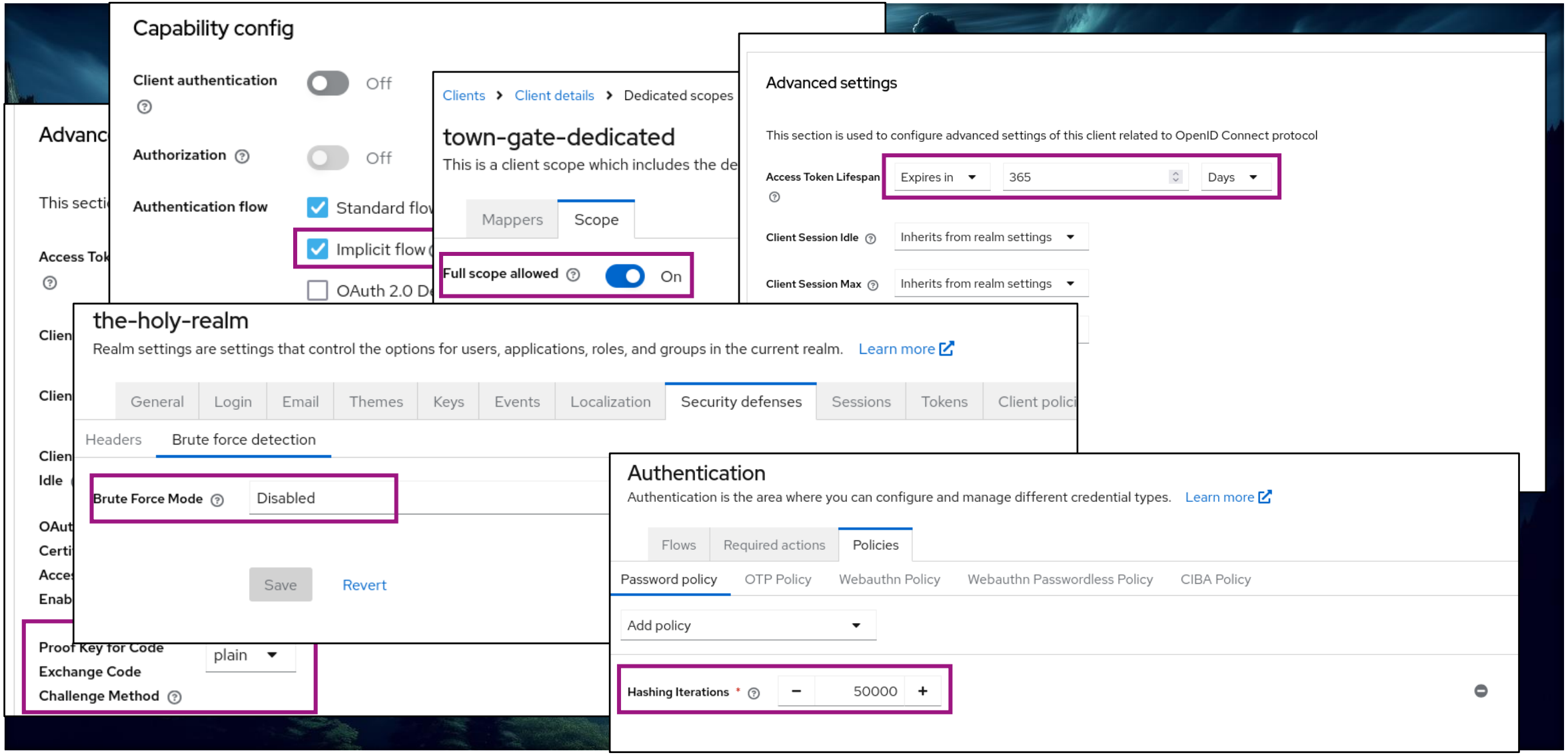
Beispiel #2: Direct Access Grants

- *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-URIs
- Erleichtert Brute-Forcing oder Phishing

Sollte mindestens für public Clients ausgeschaltet sein!



Weitere versteckte Gefahren

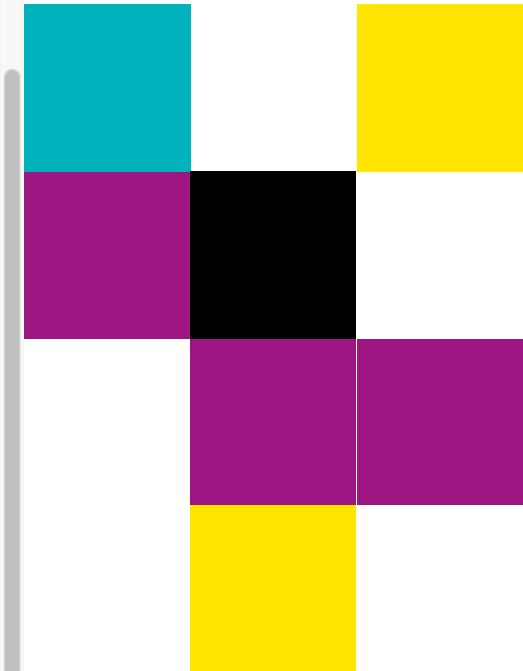


Wer will das per Hand prüfen?

Für jeden Client, jede Gruppe, jeden Scope, …?

kc-audit — -zsh — 141×47

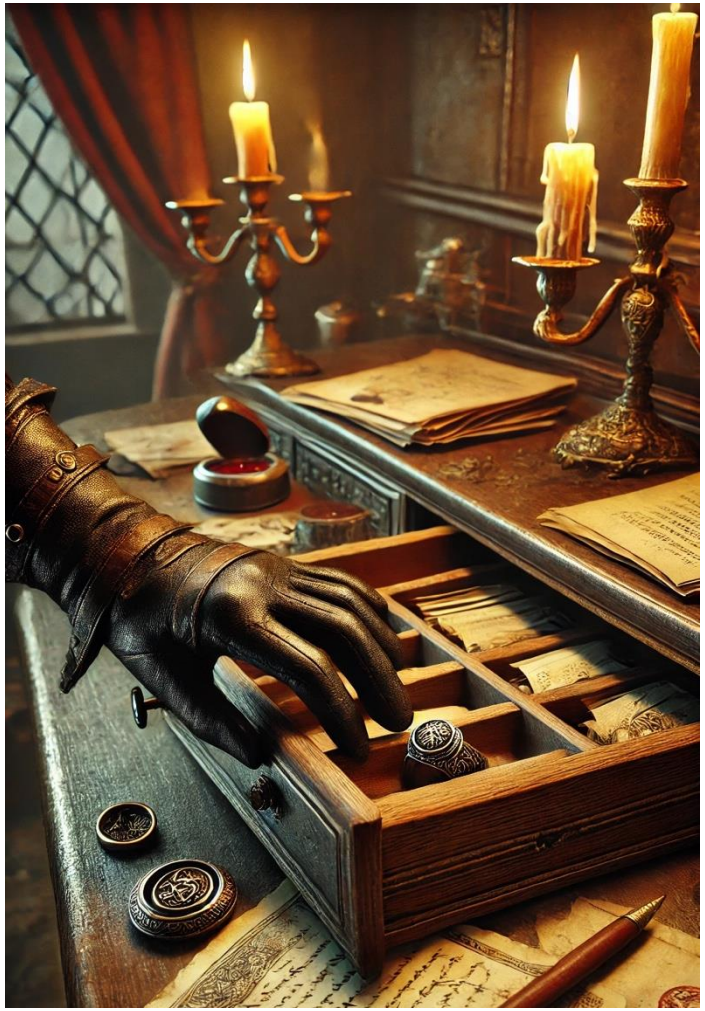
```
$ poetr
```



Unser Tool: kcwarden

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

Schritt 3: Wovor haben wir Angst?

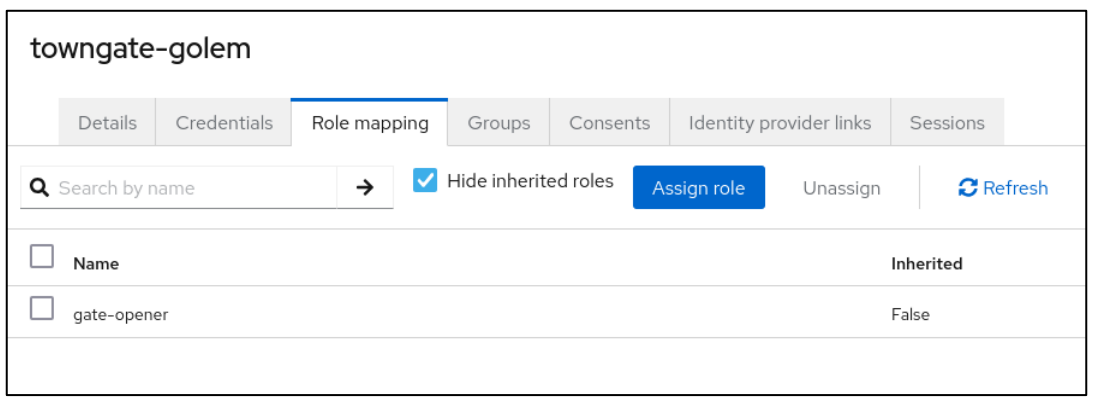


Beispiel #1: M2M – Tech User vs Service Account

Technischer User

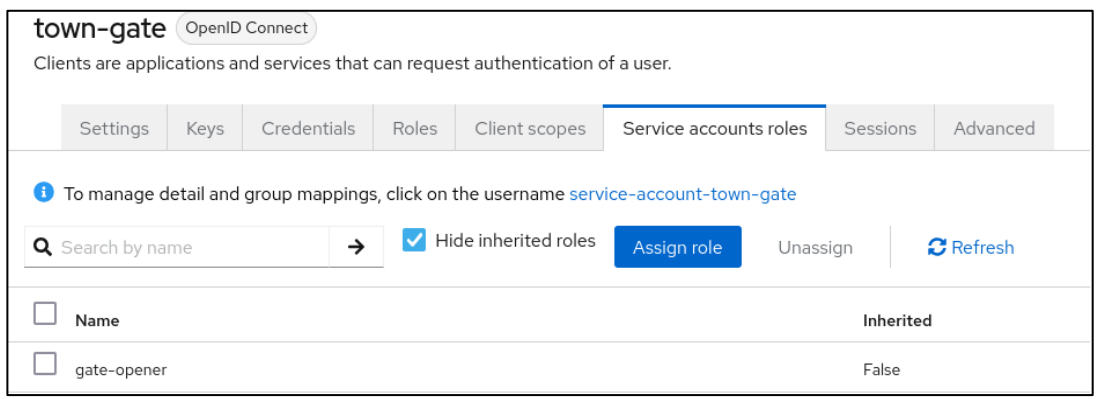
- Normaler Nutzeraccount
- Kann durch das Eingeben falscher Passwörter gesperrt werden
- Benutzt Direct Access Grants

Risiko für die Verfügbarkeit



Service Account

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



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

Schritt 4: Patrouillen aussenden

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

```yaml
- monitor: ServiceAccountWithSensitiveRole
  config:
  - role: ring_carrier
    role-client: realm
    severity: Critical
    allowed:
    - service-account-frodo
    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

```yaml
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