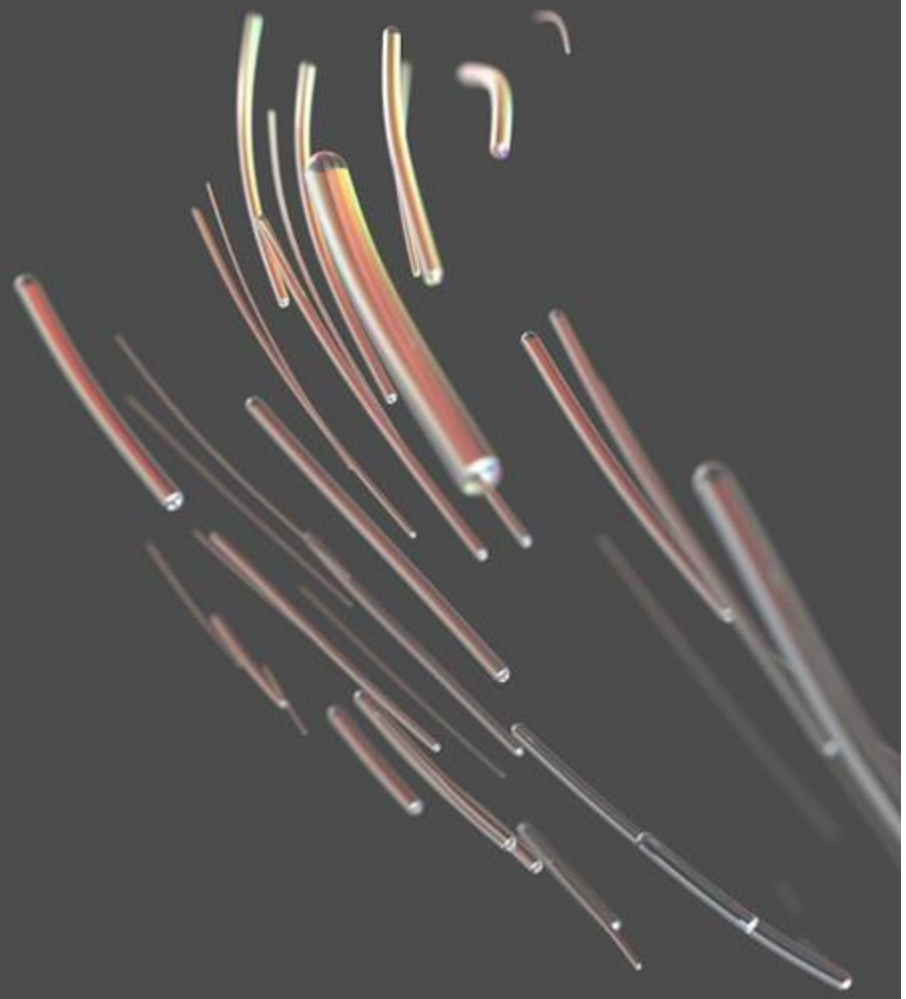


INNOQ TECHNOLOGY DAY / 5.12.2024

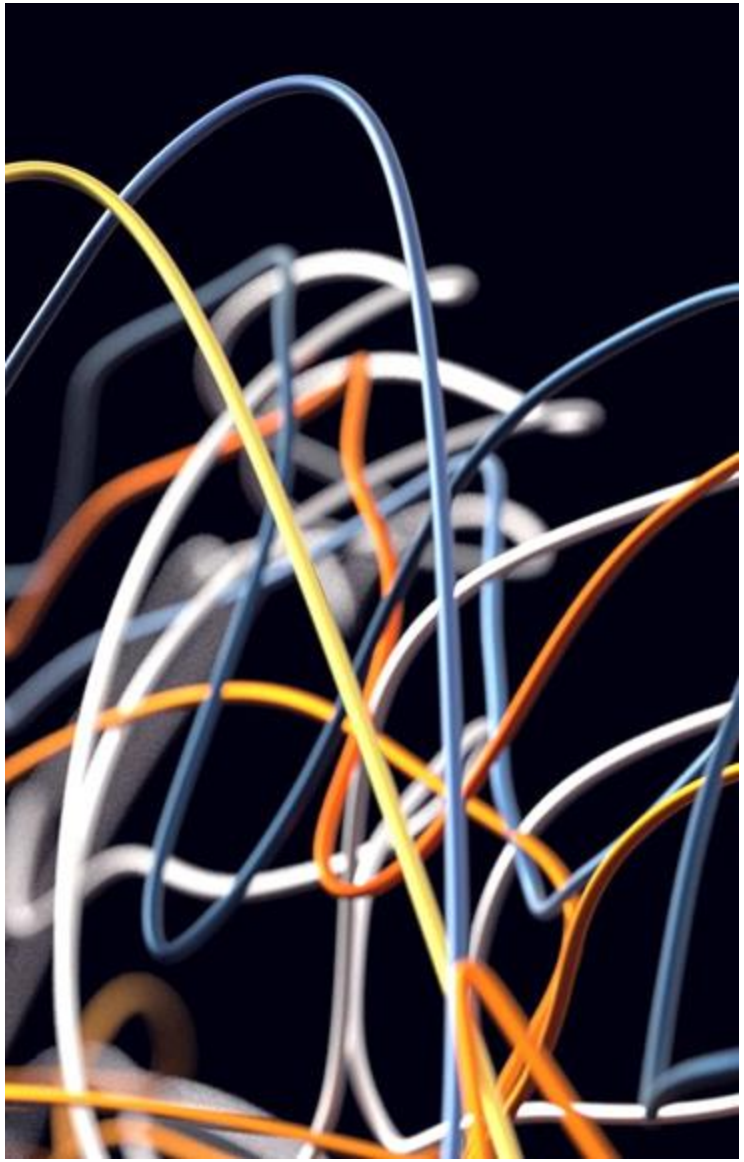
RBAC, ABAC, PBAC, REBAC - WHAT THE HECK?

INNOQ

Dominik Guhr
Senior Consultant

Agenda

- Warum?!
- Was ist Autorisierung?
- Autorisierungskonzepte
- Anforderungen
- Autorisierungsmodelle
  - RBAC
  - ABAC
  - Urlaub
  - ReBAC
- Demo



Warum sollte mich das jetzt interessieren?!

OWASP Top 10 2021 & API Top 10 2023

- **A01:2021-Broken Access Control** moves up from the fifth position; 94% of applications were tested for some form of broken access control. The 34 Common Weakness Enumerations (CWEs) mapped to Broken Access Control had more occurrences in applications than any other category.
- **API1:2023 - Broken Object Level Authorization** - APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface of Object Level Access Control issues. Object level authorization checks should be considered in every function that accesses a data source using an ID from the user.
- **API3:2023 - Broken Object Property Level Authorization** - This category combines API3:2019 Excessive Data Exposure and API6:2019 - Mass Assignment, focusing on the root cause: the lack of or improper authorization validation at the object property level. This leads to information exposure or manipulation by unauthorized parties.
- **API5:2023 - Broken Function Level Authorization** - Complex access control policies with different hierarchies, groups, and roles, and an unclear separation between administrative and regular functions, tend to lead to authorization flaws. By exploiting these issues, attackers can gain access to other users' resources and/or administrative functions.
- **API6:2023 - Unrestricted Access to Sensitive Business Flows** - APIs vulnerable to this risk expose a business flow - such as buying a ticket, or posting a comment - without compensating for how the functionality could harm the business if used excessively in an automated manner. This doesn't necessarily come from implementation bugs.

Und wo ist jetzt das Problem?

Wir haben 100 Leute gefragt…

“Super einfach bei uns!?”

“Wir haben ein organisatorisches Problem mit dem Berechtigungskonzept”

“Wir haben gerade ein Jahr daran gearbeitet, einen Autorisierungs-Service zu bauen”

“Änderungen fürs (Identity- &) Access-Management fressen ca 20% unserer Zeit.”

Wer hat denn jetzt recht?!

Was ist überhaupt Autorisierung?

- tl;dr: Der Prozess, jemandem oder etwas Zugriff auf etwas anderes zu gewähren (oder eben nicht).

Abgrenzung zu Authentisierung:

- Authentisierung beantwortet die Identitätsfrage “Wer bist du?”
- Autorisierung beantwortet “Ok, jetzt wo wir wissen wer du bist: Was darfst du?”

Autorisierungskonzepte

- **Subjekt**

  Entitäten, die Zugriff auf *Objekte* haben können, um Aktionen an ihnen auszuführen. Beispiel: Ein Nutzeraccount, oder Prozesse in einem IT-System

- **Objekt**

  Ressourcen innerhalb eines IT-Systems, die durch Zugriffskontrolle geschützt sind.

- **Recht**

  Die Art des Zugriffs, die einem Subjekt auf ein Objekt gewährt wird - z.B. lesen, schreiben, löschen, …

- **Kontext**

  Bestimmte Bedingungen, unter denen Zugriff gewährt oder verweigert wird, z.B. Zeit, Ort, Netzwerk, …

Autorisierung: Die richtigen Fragen

- **Der Klassiker: Binäre Ja/Nein-Fragen:**
  - “Kann Bob drucken?”
  - “Hat Alice Zugriff auf Dokument 123?”
- **Kontextabhängige Fragen (“Ja, aber”-Fragen):**
  - “Hat Bob **heute** Zugriff auf die Dokumente?”
  - “Hat Alice Zugriff auf die Dokumente, auch wenn sie gerade **vom Privatrechner aus ohne VPN** arbeitet?”
- **Non-binäre Fragen:**
  - “Auf **welche** Dokumente hat Alice Zugriff?”
  - “**Worauf** hat Alice alles Zugriff?”
  - “**Wer** hat alles Zugriff auf mein Dokument?”

Autorisierung: Fachliche Anforderungen

- **Konfigurierbarkeit**
- **Übertragbarkeit**
- **Evolvierbarkeit**
- **Regulatorik & Transparenz:**
  - Before the fact-Audit
    - “Auf welche <Objekte> hat <Subjekt> Zugriff”?
    - “Welche <Subjekte> haben Zugriff auf welche <Objekte>”?
  - After the fact-Audit
    - Entscheidungs-Log
- Granularität

Autorisierung: Low-Level-Anforderungen

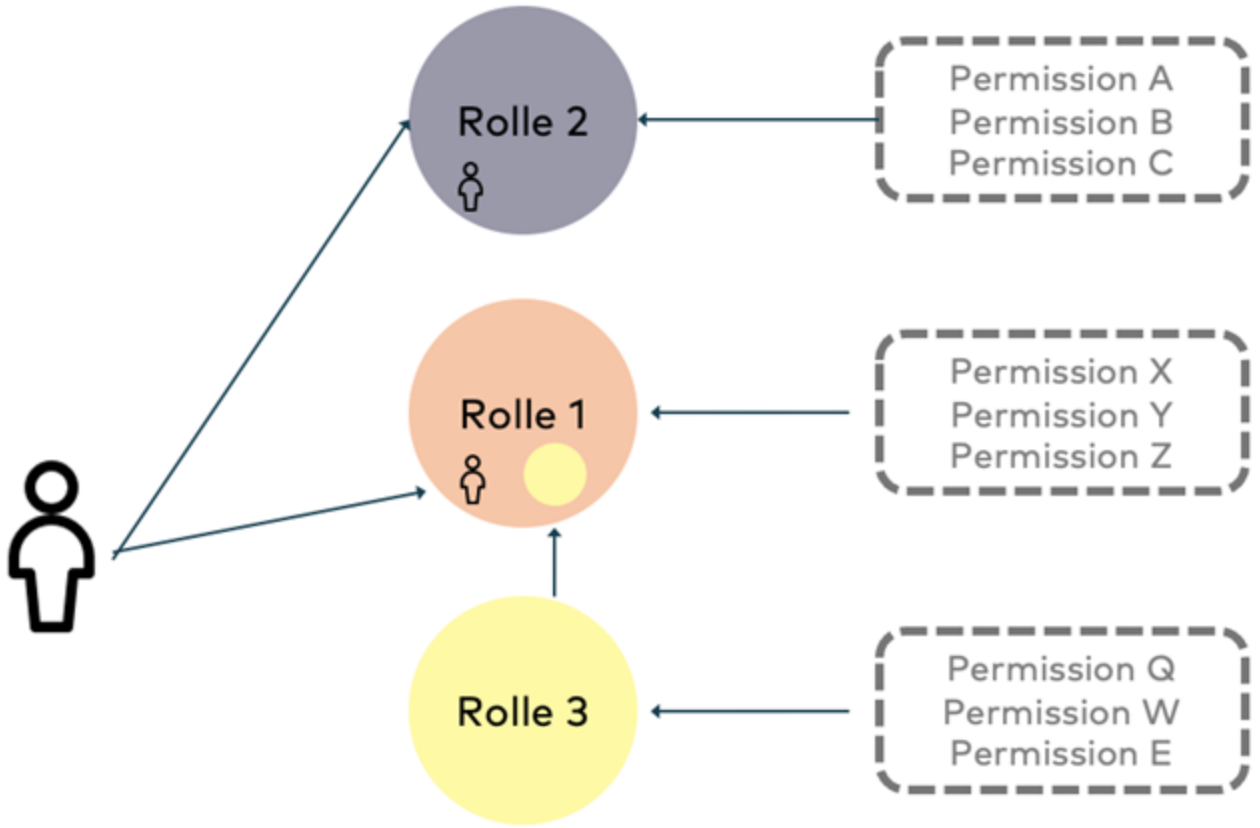
- **Performance:** <1ms, < ca. 75ms >=100ms
- **Zuverlässigkeit:** 100% ;-)
- **Konsistenz**
  - “New Enemy Problem” - strong consistency vs eventual consistency

Autorisierungsmodelle

Autorisierungsmodelle sind Modelle, die definieren, anhand welcher Merkmale und wie Zugriffsregeln abgebildet und Zugriffsentscheidungen Getroffen werden.

Role Based Access Control (RBAC)

Beispiel:
“Dominik hat die Rolle Speaker, die ihn berechtigt, einen Talk zu geben und Slides zu zeigen.”



- Zugriffsentscheidungen werden über Rechte (permissions) getroffen.
- Ein Set an Rechten wird in eine oder mehrere Rollen gekapselt, die meist Arbeitsprozesse oder -Strukturen abstrahieren.
- Rollen selbst können andere Rollen beinhalten.

RBAC - Vorteile der Implementierung

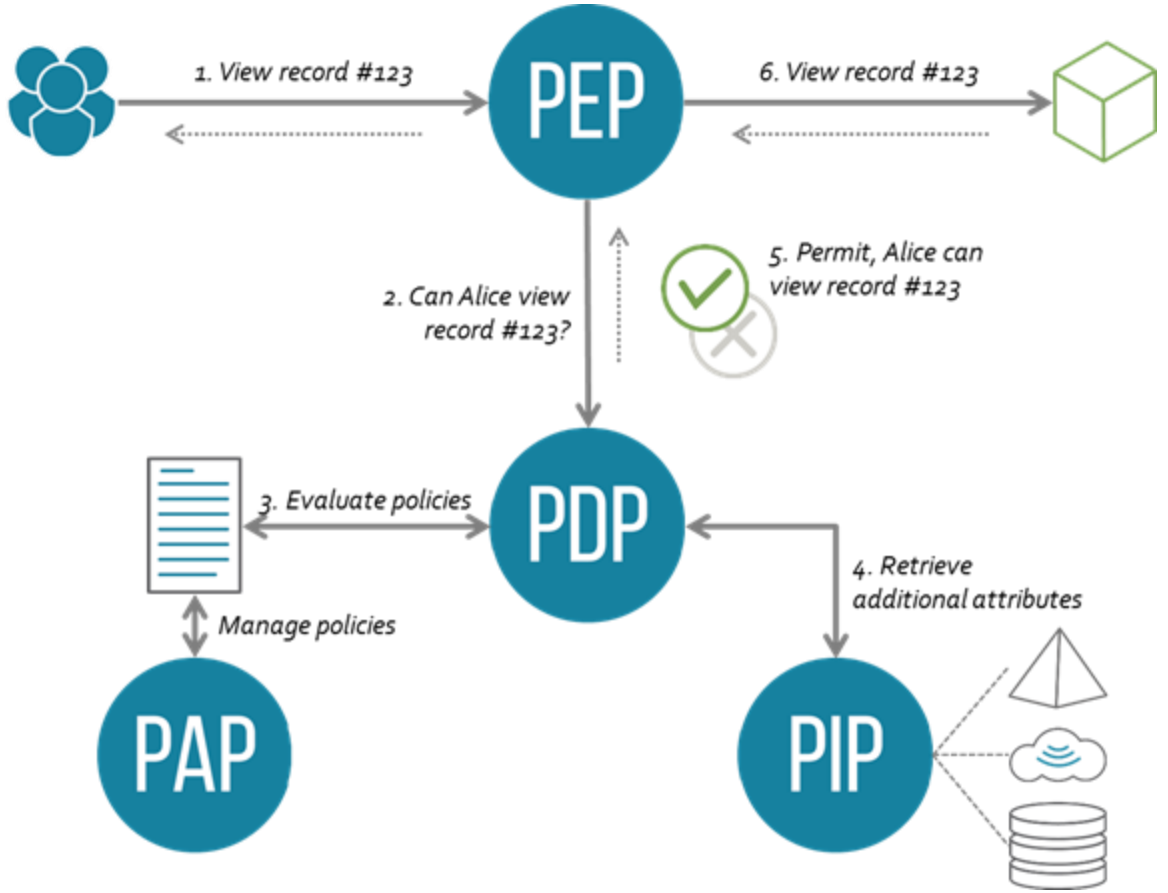
- Status Quo
  - Weite Verbreitung
  - vorhandenes Wissen
  - Gutes Ökosystem an Implementierungen
- Gut passend für einfache Anwendungsfälle
- Performance & Einfachheit der Implementierung in monolithischen Systemen gut.

RBAC - Nachteile der Implementierung

- Grobgranular
- Rollenexplosion
- Duplikation der Implementierung pro Service
- Meist “direkt im Code”, oft mit eigenen Erweiterungen.
- Keine Offenen oder kontextabhängige Fragen stellbar.
- Modellierung skaliert nicht

Attribute Based Access Control (ABAC)

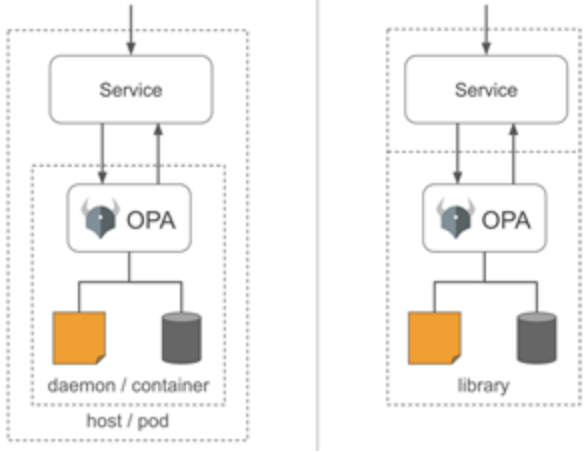
Beispiel:
“Wenn ein Bankmitarbeiter bei der Bank arbeitet, und innerhalb der Arbeitszeit auf das Kundenkonto zugreifen will, erlaube den Zugriff.”

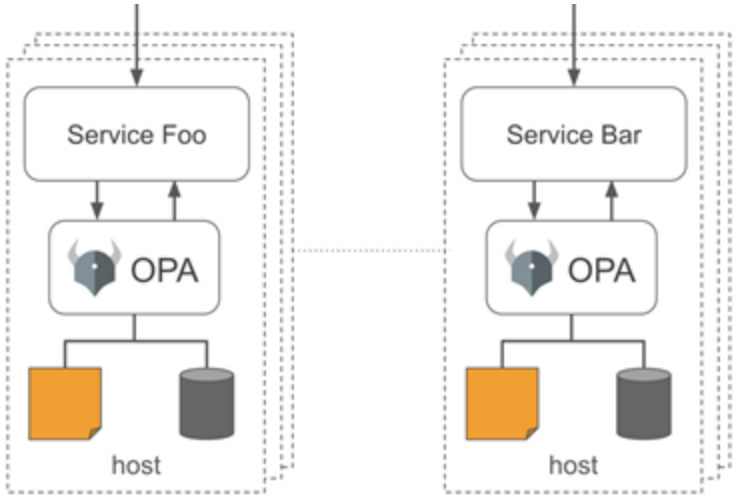


- Zugriffsentscheidungen werden anhand einer Reihe von Attributen getroffen.
- Attributtypen sind nicht limitiert, sie können sich z.B. auf Subjekt, Objekt, Umgebung, Kontext, … beziehen
- Attribute sind via boolescher Logik kombinierbar

PBAC - Policy Based Access Control

- PBAC == eine nutzbare Implementierung von ABAC
  - Policy Engine
  - Spezialisierte DSL für schreiben von Policies
  - unterstützendes Tooling
- z.B. Open Policy Agent



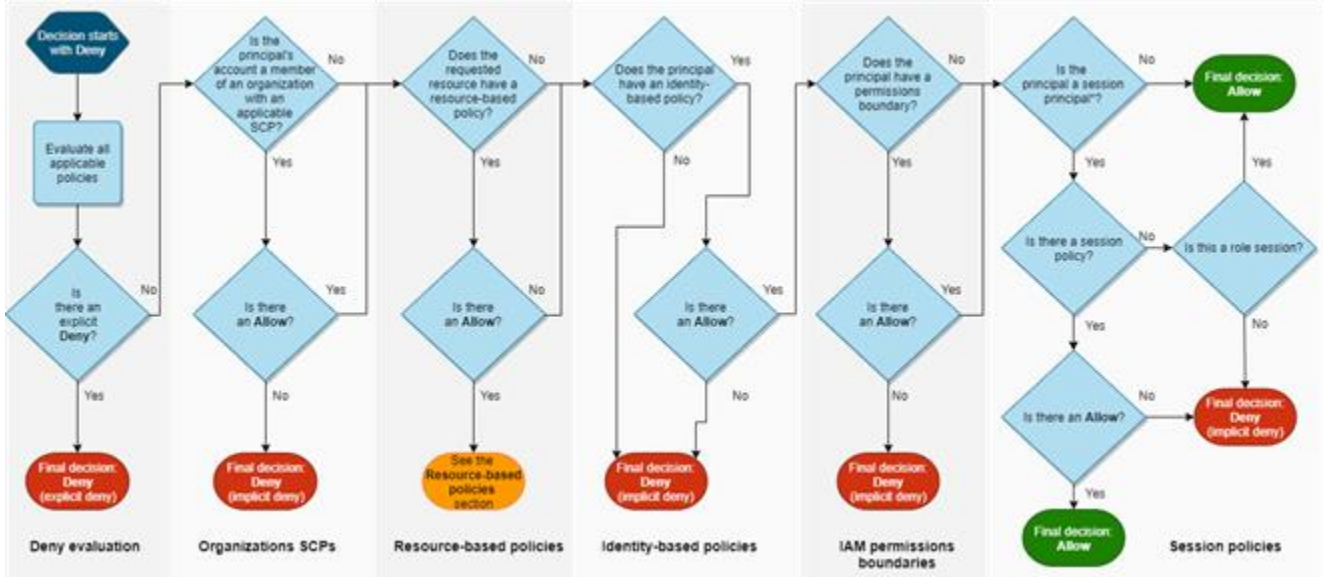


ABAC - Vorteile

- Sehr flexibel
- Quasi beliebig skalierbar
- Feingranular
- Kontextbezogene Fragen Toplevel-Konzept

ABAC - Nachteile

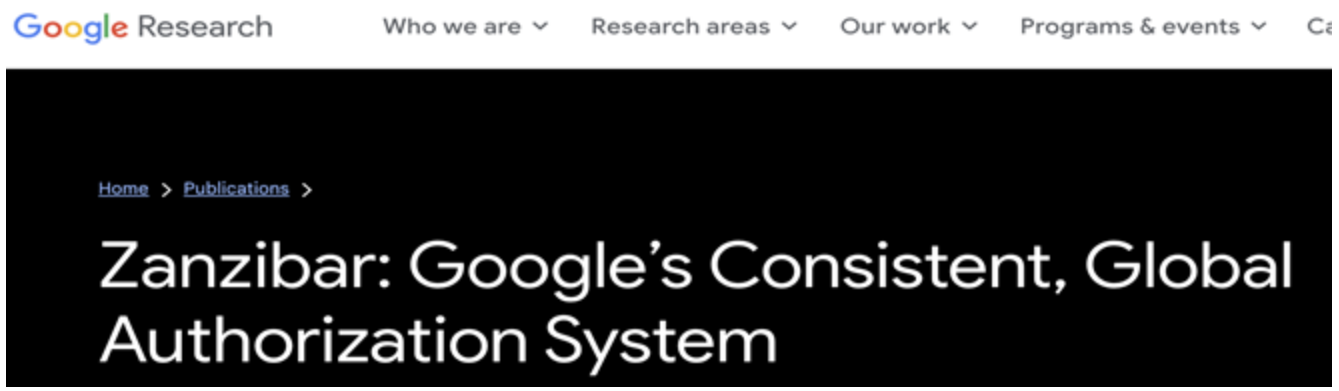
- Administrative Komplexität
- Provisioning der benötigten Daten, Datenkonsistenz
- Lernkurve (meist komplizierte “Sprache” rego, xacml/alfa)
- Offene Fragen: Nope.
- Wird schnell komplex



Ok - Ich brauch Urlaub

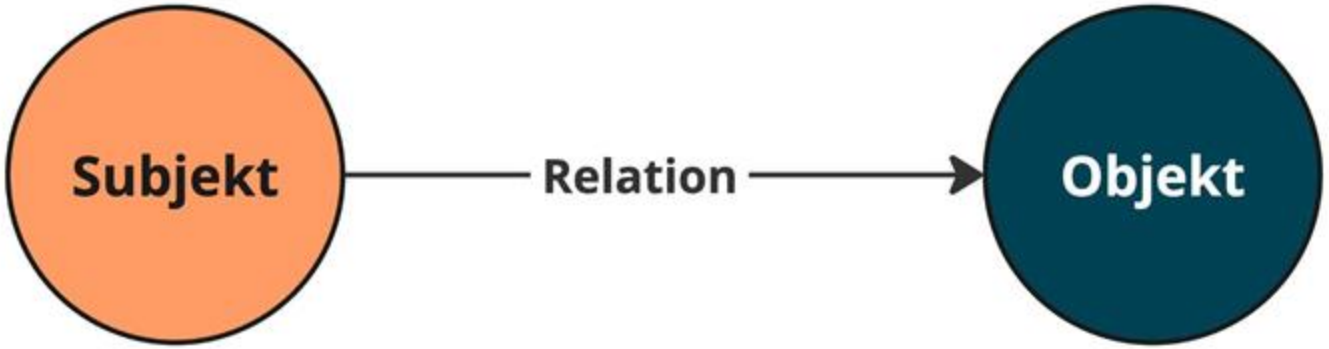
2019: Zanzibar-Paper

- Google-Paper über Access Management, basierend auf ACLs
- Skaliert auf Milliarden von Tupeln ⟨tuple⟩::=⟨object⟩‘#’⟨relation⟩‘@’⟨user⟩
- mehr als 99.999% Verfügbarkeit für über 3 Jahre (~8.75h p.A.)
- rückwärts-indizierbare Semantik
- zentrales System
- p95 Latenz <=10ms



Relationship Based Access Control (ReBAC)

Beispiel:
“Alice kriegt Leseugriff auf ein bestimmtes Dokument, weil sie entweder die Erstellerin ist, oder im Team der Erstellerin ist, oder weil die Erstellerin ihr Lesezugriff gegeben hat.”



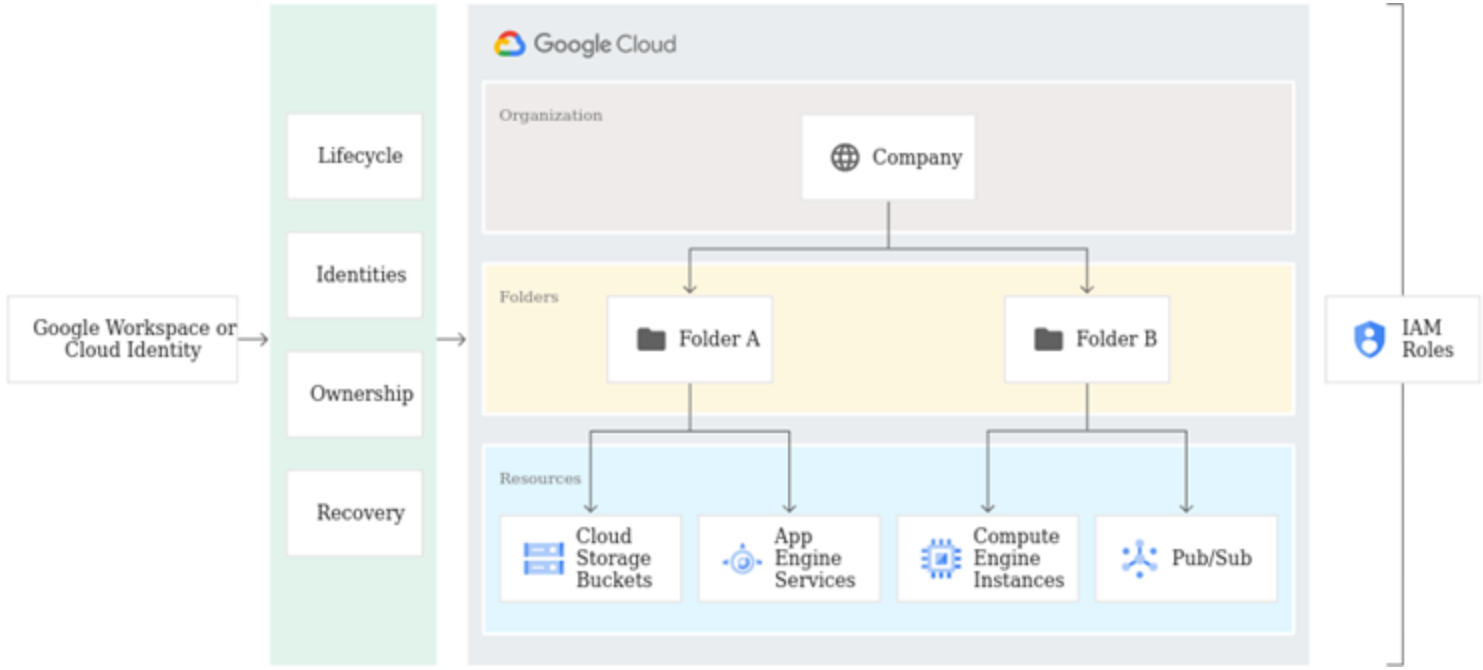
<Tupel> ::= <objekt>'#'<relation>'@'<user>

<Tupel> ::= <Dokument1>'#'<owner>'@'<alice>

- **basiert auf Relationen zwischen Subjekten und Objekten**
- **visualisierbar als Graph**
- **zentralisiertes System**
- **einfache DSL**

ReBAC - Vorteile

- Feingranulare Autorisierung möglich
- Provisionierung in zentralem System einfacher
- Offene Fragen beantwortbar
- Starke Konsistenz



ReBAC - Nachteile

- zentralisierte, kritische Komponente
- Keine <1ms-Anwendungsfälle möglich
- Kontextabhängige Fragen nur bedingt beantwortbar*
- Bisher kaum kostenloses Tooling für Schema-Provisionierung

* Aber Kombinationen mit ABAC verfügbar.

Demo Time

Fazit

- Autorisierung != RBAC
- Externalisieren von Autorisierung erlaubt getrennte, flexible Evolution von Businesscode und Autorisierungscode
- ABAC am flexibelsten, aber auch schnell am kompliziertesten in der Implementierung.
- ReBAC bietet gute Schnittmenge zwischen einfacher Nutzbarkeit und feingranularer Flexibilität, gerade mit ABAC-Kombination.
- Zukunft: Einfach zu nutzende, flexible Kombinationen der Modelle, samt Integration von AuthN (z.B. via **Heimdall**)

DANKE! FRAGEN?

https://linkedin.com/in/dguhr oder im Chat