Role Based Access Control und Identity Management

Treffen des ZKI-AK Verzeichnisdienste,

7.-8.3.2012, Halle

Peter Gietz, Markus Widmer,
DAASI International GmbH
Peter.gietz@daasi.de

Agenda

- Motivationen für rollenbasierte Zugriffskontrolle
- Der RBAC-Standard
- OpenRBAC
- RBAC und Identity Management

Motivation für rollenbasierte Zugriffskontrolle

- Durch die Verwendung von Rollen wird die Zugriffskontrolle wesentlich übersichtlicher
- Wenn ein Benutzer die Rolle wechselt (z.B. von Student zu Systemadmin) bekommt er automatisch die entsprechenden Rechte
- Es werden keine Sonderlösungen eingeführt, die dann nicht dokumentiert sind und vergessen werden
- Die reale, gewachsene Organisationsstruktur wird durch Rollen (z.B. Student, Professor, Sekretärin) abgebildet
- Veränderungen in der Organisationsstruktur können einfach in das RBAC-System übernommen werden
- Es gibt bewährte Standards (RBAC und XACML)

DAASI International

Voraussetzungen

- Klares Rollenkonzept
- Rollen müssen in Benutzerverwaltung abgebildet sein
- Anwendungen müssen entsprechende Informationen verwerten können
- Identity Management ist hierbei sehr hilfreich
- Auch über Föderationsinfrastrukturen (Shibboleth) können Rolleninformationen transportiert werden

Der RBAC-Standard

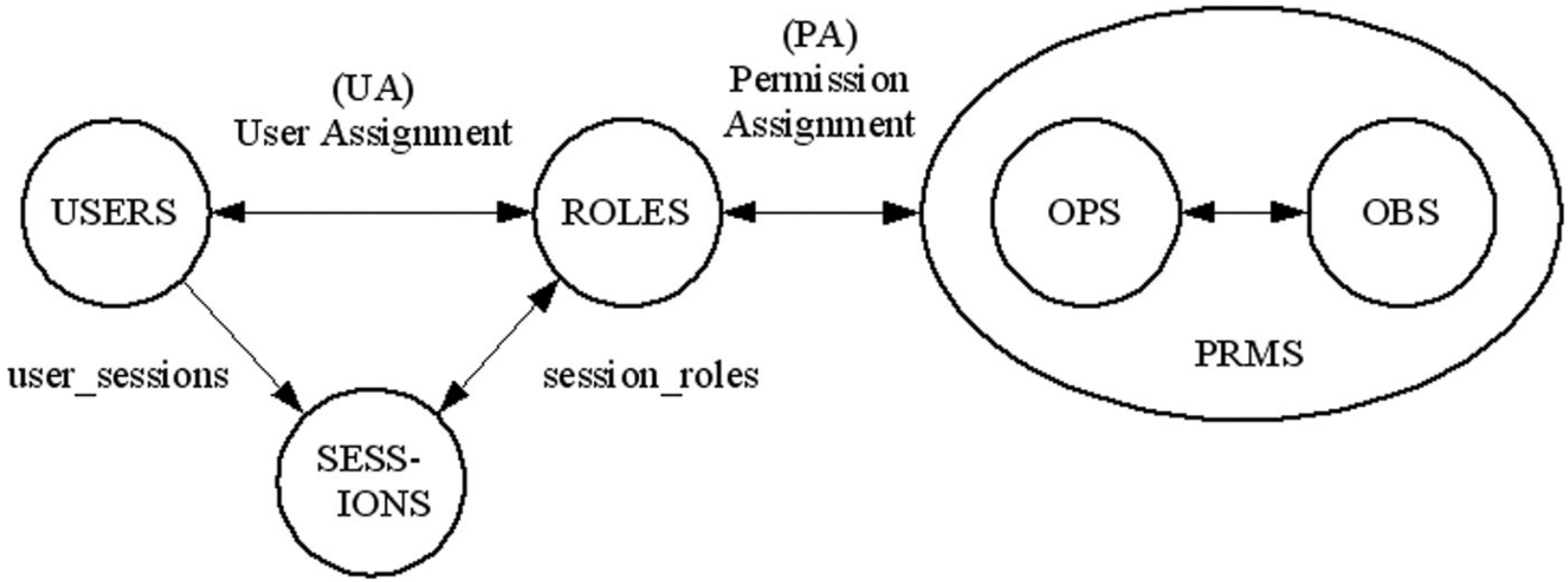
- Der ANSI-Standard RBAC teilt sich auf in drei Bereiche:
  - RBAC Core
  - Hierarchical RBAC (zwei Typen)
  - Separation of Duty (zwei Typen, die im Standard als eigene Bereiche gezählt werden)
- Wichtige Komponenten sind:
  - Benutzer
  - Rolle
  - Session
  - Berechtigung
  - Ressource

RBAC-Core

- Definiert grundlegende Funktionen, die eine RBAC-Implementierung beinhalten muss. Dazu gehören:
  - Anlegen und Löschen eines Benutzers, einer Rolle oder einer Session
  - Hinzufügen und Entfernen von Berechtigungen auf Ressourcen
- Definiert die Funktion checkAccess, mit der eine Zugriffsentscheidung angefordert werden kann
- Definiert weitere Funktionen zum
  - Ändern von Beziehungen der Komponenten untereinander
  - Abfrage von Informationen zu den einzelnen Komponenten des Systems

DAASI International

RBAC-Core



Aus: ANSI: Role Based Access Control
http://csrc.nist.gov/groups/SNS/rbac/documents/draft-rbac-implementation-std-v01.pdf

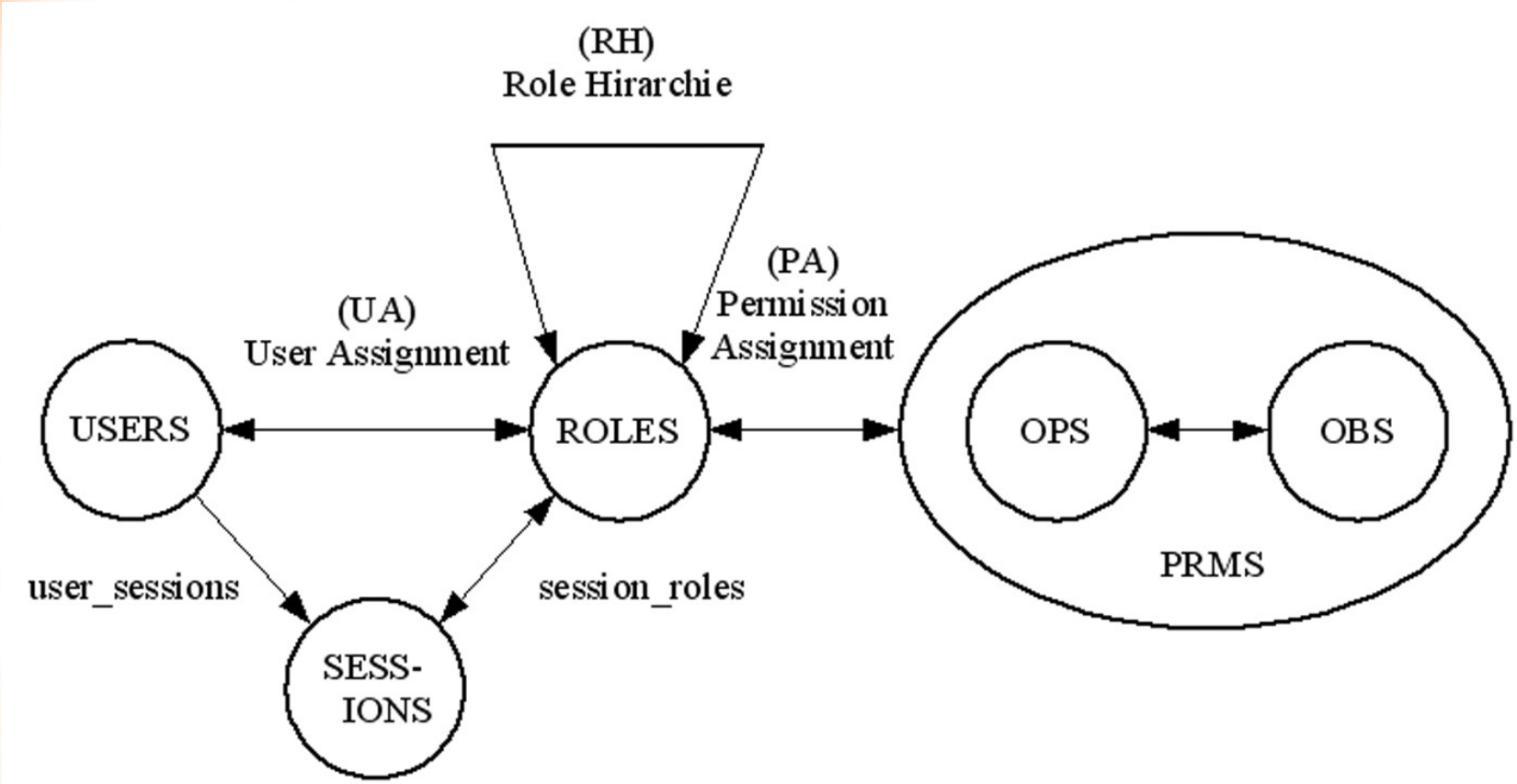
DAASI International

Hierarchical RBAC

- Erweitert die Grundfunktionalität um Rollenhierarchien, wobei es zwei Typen gibt:
  - Limited Role Hierarchy: Rollen sind in Baumstrukturen geordnet (jeweils nur ein Elternknoten)
  - General Role Hierarchy: Rollen können in freien Graphen organisiert sein (beliebig viele Elternknoten)
- Hierbei werden einige im RBAC-Core definierte Funktionen angepasst:
  - z.B. die Funktion addActiveRole, die Rollen in einer Session aktiviert, muss nun auch über die Hierarchie implizit geerbte Rollen berücksichtigen
- Darüber hinaus werden neue Funktionen definiert zum Ändern der Hierarchien von Rollen

DAASI International

Hierarchical RBAC



Aus: ANSI: Role Based Access Control
http://csrc.nist.gov/groups/SNS/rbac/documents/draft-rbac-implementation-std-v01.pdf

DAASI International

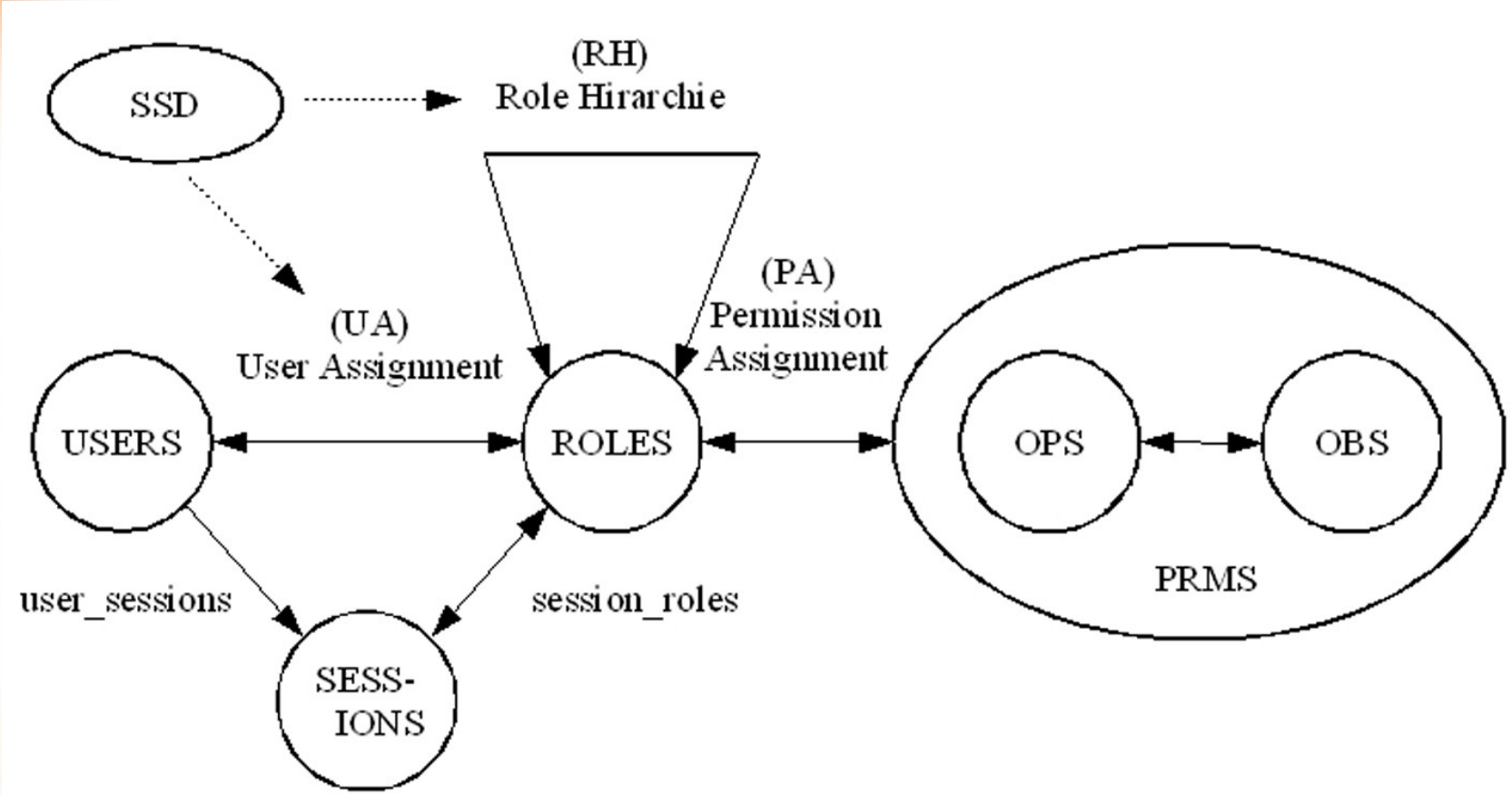
Separation of Duty

- Um zu verhindern, dass ein Benutzer gleichzeitig sehr unterschiedliche Rollen ausübt, wird als weiterer Zusatz eine Pflichtentrennung eingeführt
- Über sogenannte Sets werden sich gegenseitig ausschließende Rollen definiert
- Es wird unterschieden zwischen
  - Static Separation of Duty
  - Dynamic Separation of Duty

Static Separation of Duty (SSD)

- Statisch bedeutet hier: „über die Zeit nahezu unveränderlich“
- SSD-Sets definieren sich grundsätzlich ausschließende Rollen, die kein Benutzer gleichzeitig innehaben darf
- Die in SSD-Sets definierten Einschränkungen werden bei jeder Zuweisung von einer Rolle zu einem Benutzer angewendet

Static Separation of Duty (SSD)



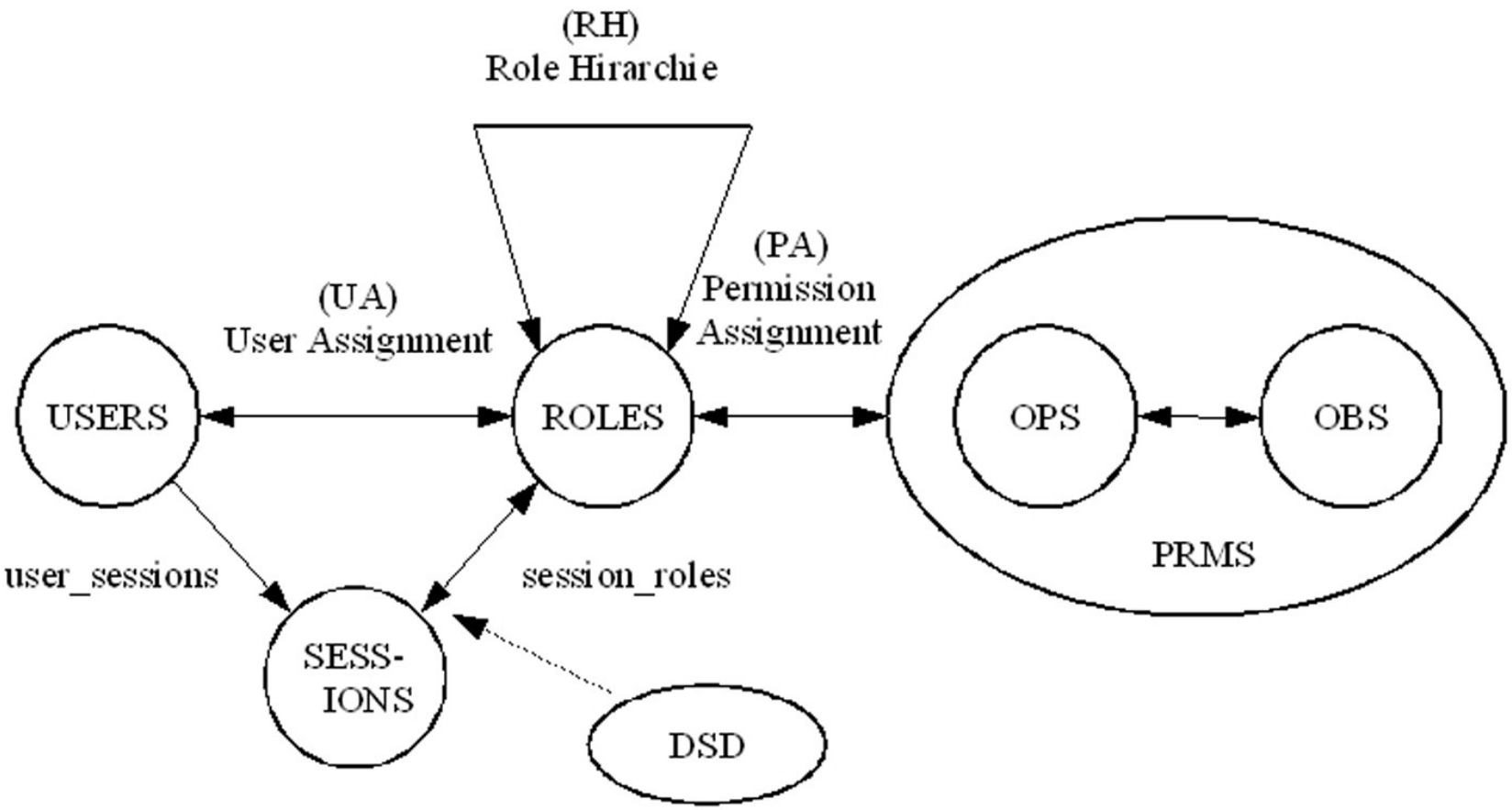
Aus: ANSI: Role Based Access Control
http://csrc.nist.gov/groups/SNS/rbac/documents/draft-rbac-implementation-std-v01.pdf

DAASI International

Dynamic Separation of Duty (DSD)

- Im Unterschied zum SSD werden die Einschränkungen erst beim Aktivieren von Rollen (also „zur Laufzeit“) in einer Session geprüft
- Hierdurch kann es einem Benutzer z.B. möglich sein die Rollen „Antragssteller“ und „Antragsprüfer“ auszufüllen, jedoch nicht gleichzeitig
- Eine aktive Rolle zu haben bedeutet, im Unterschied zu einer nicht aktiven Rolle, dass diese in einer Session eingetragen ist

Dynamic Separation of Duty (DSD)



Aus: ANSI: Role Based Access Control
http://csrc.nist.gov/groups/SNS/rbac/documents/draft-rbac-implementation-std-v01.pdf

DAASI International

Erweiterbarkeit von RBAC

- Das durch den Standard definierte RBAC stellt bereits eine sehr umfangreiche Bibliothek zur Verfügung, mit der viele Autorisierunganforderungen erfüllt werden können
- Diese Funktionalität kann aber auch sehr leicht erweitert werden, da der Standard selbst bereits in einzelne Bereiche gegliedert ist, die aufeinander aufbauen.
- Ein Beispiel hierfür ist „Multi-session Separation of Duties“ (David Chadwick, 2006):
  - Eine Erweiterung, in der sich ausschließende Rollen über alle Sessions eines Benutzers geprüft werden.

OpenRBAC

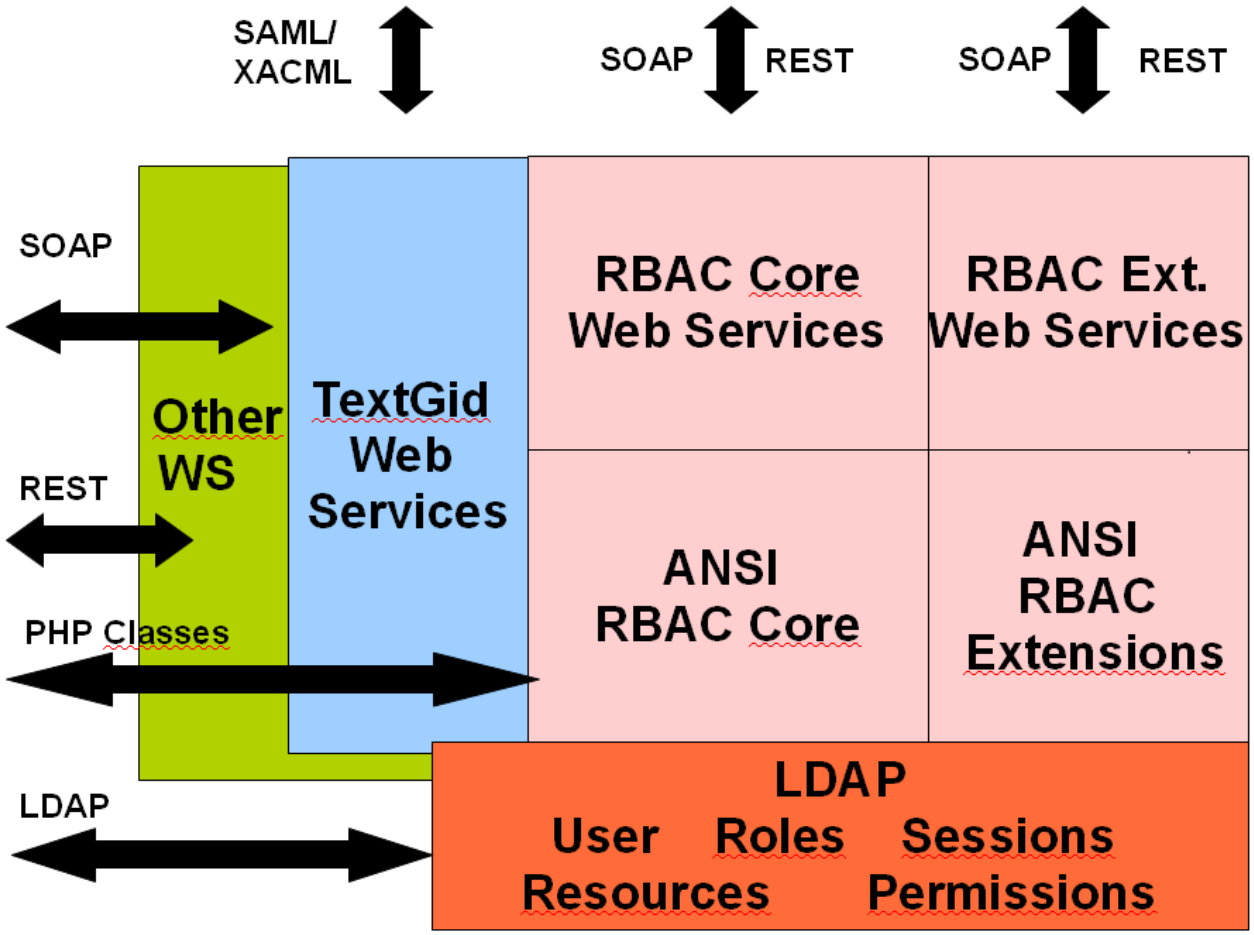
- OpenRBAC ist eine OpenSource-Implementierung des RBAC-Standards
- Im Rahmen einer Diplomarbeit von Markus Widmer bei DAASI International entstanden
- Im Rahmen von mehreren Grid-Forschungsprojekten von DAASI eingesetzt und weiterentwickelt worden
- Implementiert den gesamten Standard außer General Role Hierarchy (Da die Rollenhierarchien im LDAP-Baum abgebildet werden)
- Die RBAC-Funktionen werden von OpenRBAC selbst geschützt
- Stabile Version und Dokumentation unter http://www.openrbac.de

DAASI International

OpenRBAC-Schichten-Modell

- OpenRBAC ist in verschiedenen Schichten implementiert:
  - Kern (Datenbackend) ist ein OpenLDAP-Server mit entsprechend definiertem DIT und Schema
  - Die RBAC-Funktionen sind als PHP-Klassen implementiert, wobei die drei Bereiche Core, Hierarchical und Separation of Duty gekapselt sind
  - Die PHP-Klassen können über Web-Service Wrapper als Web Services aufgerufen werden
  - Erweiterte Webservices können ebenfalls auf die einzelnen RBAC-Methoden zugreifen
  - Die Funktion Check-Access ist auch über das XACML/SAML-Protokoll abfragbar

OpenRBAC Schichten Modell



DAASI International

Warum OpenLDAP als Backend

- Teile der Informationen (über die Benutzer) sind meistens ohnehin schon in einem LDAP-Server
- Die einzelnen Informationen sind in verschiedenen Teilbäumen organisiert (ou=Roles, ou=Resources, ou=Sessions, etc) und können, wo sinnvoll auch auf verschiedenen LDAP-Servern liegen (z.B. User auf dem Authentifizierungsserver und der Rest auf einem eigenen OpenRBAC-Server
- Das entwickelte Datenmodell ist grundsätzlich mit XACML kompatibel

Warum OpenLDAP als Backend

- (Open)LDAP kann sehr schnell Antworten auf Access-Fragen geben
  - CheckAccess ist über einen einzigen LDAP-Filter realisierbar
  - Ein OpenLDAP-Server auf einem starken Rechner (Multi-Core mit 48 GB RAM) kann auch bei großen Datenmengen (z.B. 1 Million Ressourcen) über 60.000 Abfragen pro Sekunde beantworten

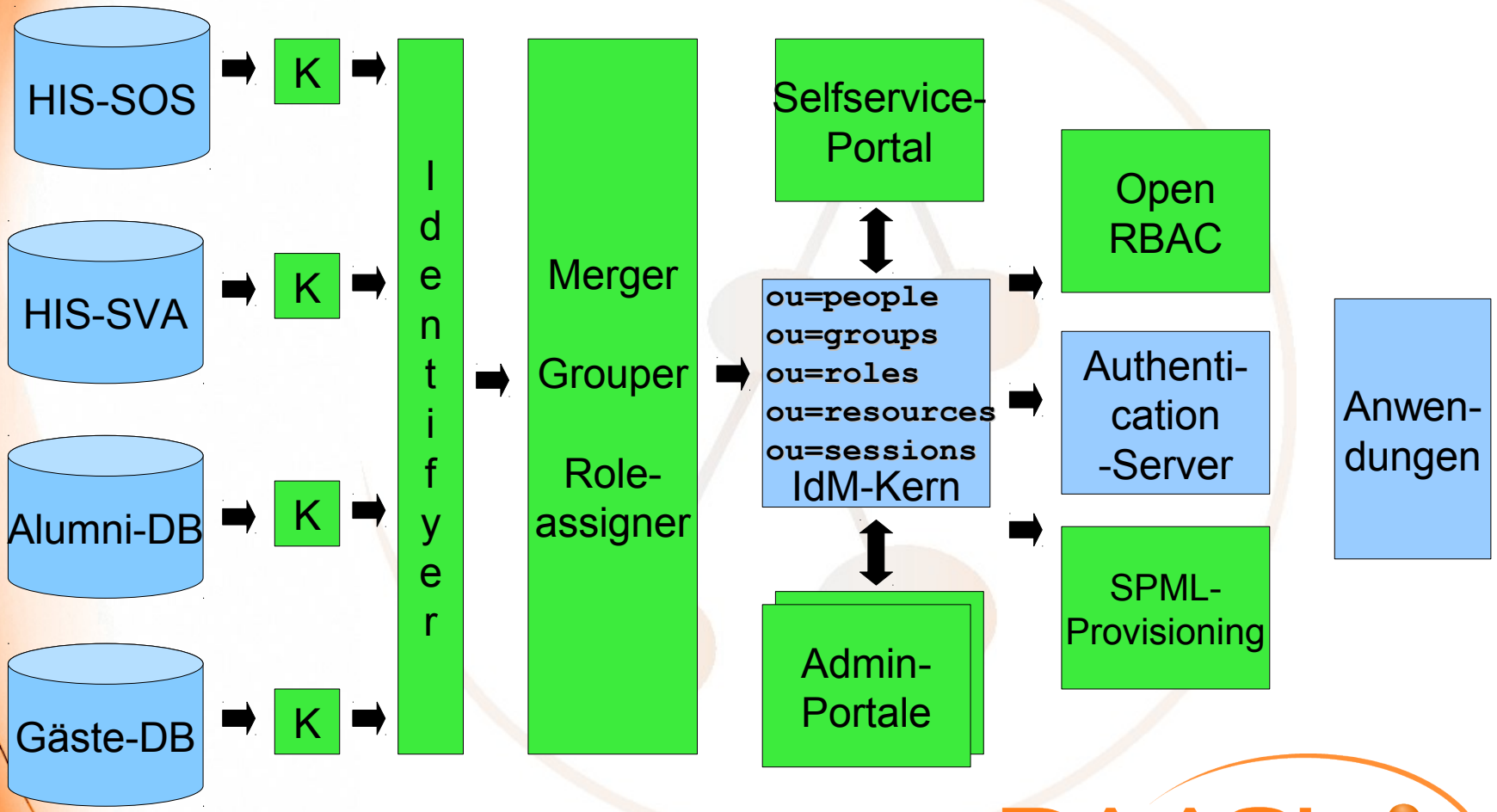
Access Management mit openRBAC

- Erweiterung des Identity Management
  - Bisher: Personenobjekte und Gruppen/Rollen
  - Zusätzlich: Berechtigungen auf Ressourcen
- Zentrale Verwaltung von Rechten für existierende Benutzer
- Ermöglicht einheitliche logische Trennung von:
  - Ressourcendefinition durch Anwendungsentwickler (Name und mögliche Operationen)
  - Berechtigungsvergabe durch Verwaltung (Rolle und erlaubte Operation)
  - Trennung durch verschiedene Administrations-interfaces und ACLs

Access Management mit openRBAC

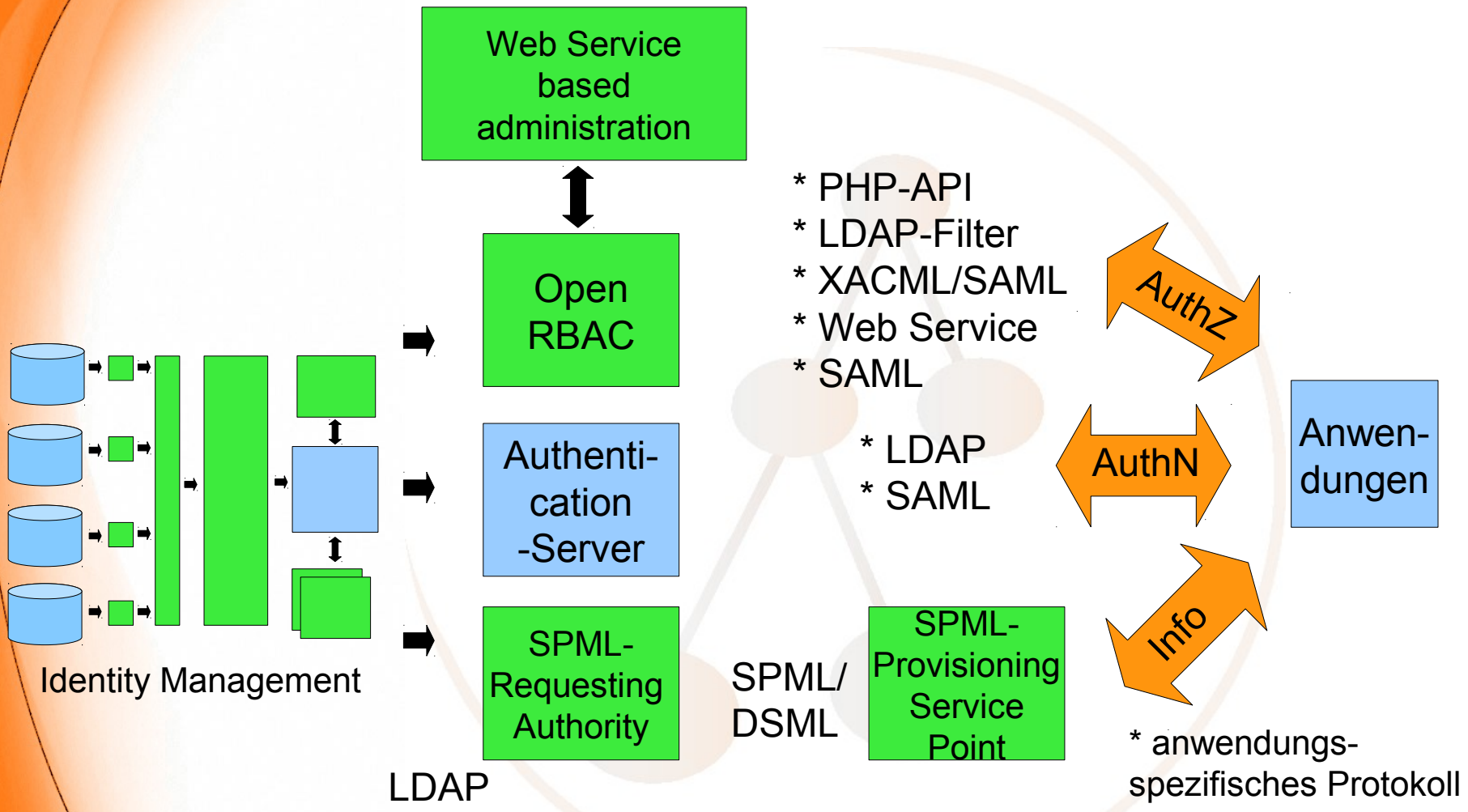
- Rechtemanagement über Anwendungsgrenzen hinweg:
  - Beispiel Multi-Session-SD: Benutzer meldet sich an Anwendung A an und aktiviert eine Rolle. Bei der Anmeldung an Anwendung B kann Benutzer nicht mehr alle Rollen aktivieren.
- Einfache Integration
  - Verwendet Infrastruktur, die für IdM bereits aufgebaut worden ist (LDAP-Server)
  - Stellt PHP-API oder Webservices bereit
  - Bestimmte Rollen können auch automatisch aufgrund von Quellsystem oder Attributen vergeben werden

RBAC und Identity Management



DAASI International

Schnittstellen für Anwendungen

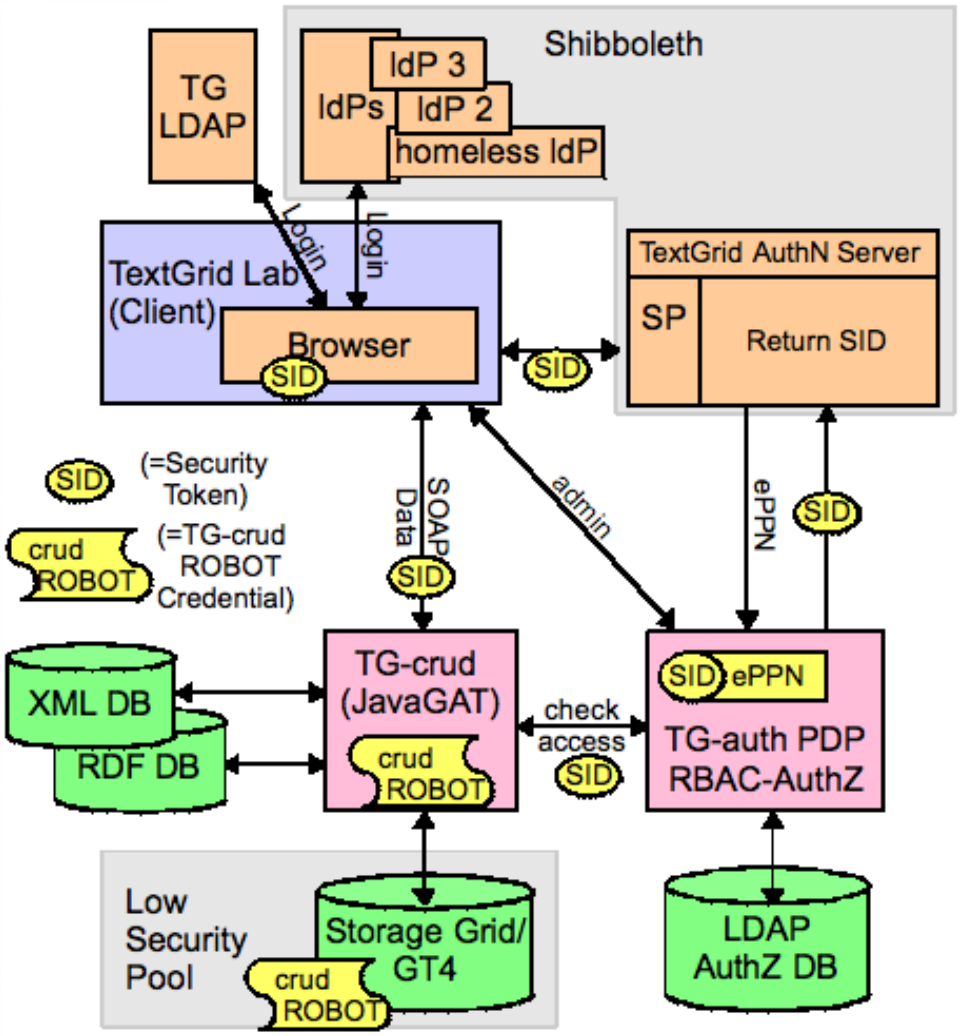


Integration von Shibboleth

- OpenRBAC lässt sich gut mit Shibboleth „verheiraten“
- Konzept der Shibboleth-Session wird nachgenutzt
- Eine Implementierung hierfür wurde im BMBF-Projekt Textgrid gemacht
  - Wobei auch textgridspezifische Web Services geschrieben wurden
  - z.B. Service, der beliebig viele Suchergebnisse bekommt und nur die Liste der Ergebnisse zurückgibt, für die der Benutzer Zugriffsrechte hat

DAASI International

Anwendungsszenario in TextGrid mit Shibboleth



Security-Token für nicht shibbolethisierte Anwendung

- Abfrage von Berechtigung (check_access) ist über Security-Token (=Sessionid, SID) möglich
- Security-Token wird nach Authentifizierung an Benutzer übergeben (als Cookie im Browser)
  - denkbar wäre auch QR-Code für das Handy, App kann dann QR-Code oder zurückgerechnete SID verwenden
  - SID ist Grundlage für Check-Access-Fragen der Anwendungen
- Anonymisierung an Anwendungen möglich, da nur Security-Token von Anwendern an Anwendung gegeben wird

Security-Token für nicht shibbolethisierte Anwendung

- Sicherheits-Problem:
  - Security-Token muss bei der Übertragung immer über verschlüsselte Verbindung geschützt sein
  - jede empfangende Anwendung muss vertrauenswürdig sein
  - Security Token kann zusätzlich symmetrisch verschlüsselt werden und nur vertrauenswürdige Anwendungen haben den Schlüssel.

Vielen Dank für Ihre Aufmerksamkeit!

- Noch Fragen?

- DAASI International GmbH
  - www.daasi.de
  - Info@daasi.de