Security and Access Control:

Access And Delegation Control For Distributed Services

Ulf Schreier

Overview

- **Introduction: Identity and Access Management (IAM)**
- Delegation Control
  - OAuth and OIDC for RESTful services
  - SAML and Web SSO for web services
- Access control
- Conclusions (for IAM)

Literature

- [RS 16] Richer, Sanso: *OAuth2 in Action*, Manning Early Access Program 2016
- [OAuth 12] Hardt (Ed.): *The OAuth 2.0 Authorization Framework*, IETF 6749
- [OIDC 14] Sakimura et al. (Ed.): *Open ID Connect Core Version 1.0*, OpenID Foundation, 2014
- [SAML 08] Ragouzis, et al. (Ed): *Security Assertion Markup Language (SAML) V2.0 Technical Overview*, OASIS Committee Draft 02, 2008
- [XACML 13] OASIS (Ed.): *eXtensible Access Control Markup Language (XACML), Version 3.0*, OASIS 2013
- [ALFA 15] Giambiagi et al. (Ed.): *Abbreviated Language for Authorization Version 1.0*, OASIS Working Draft 01, https://www.oasis-open.org/committees/download.php/55228/alfa-for-xacml-v1.0-wd01.doc (see also documentation at axiomatics.com)
- [UMA 15] Hardjono et al. (Ed.): *User-Managed Access (UMA) Profile of OAuth 2.0*, Recommendation Kantara Initiative, 2015

Technology Overview

Category	JSON/RESTful API based solutions	XML/web service based solutions
Protocol	OAuth	SAML
Identity protocol	Open ID Connect (OIDC)	SAML Web SSO
Access control language		XACML

Technology Overview

Category	JSON/RESTful API based solutions	XML/web service based solutions
Protocol	OAuth	SAML
Identity protocol	Open ID Connect (OIDC)	SAML Web SSO
Rule language	RESTACL (own work)	XACML

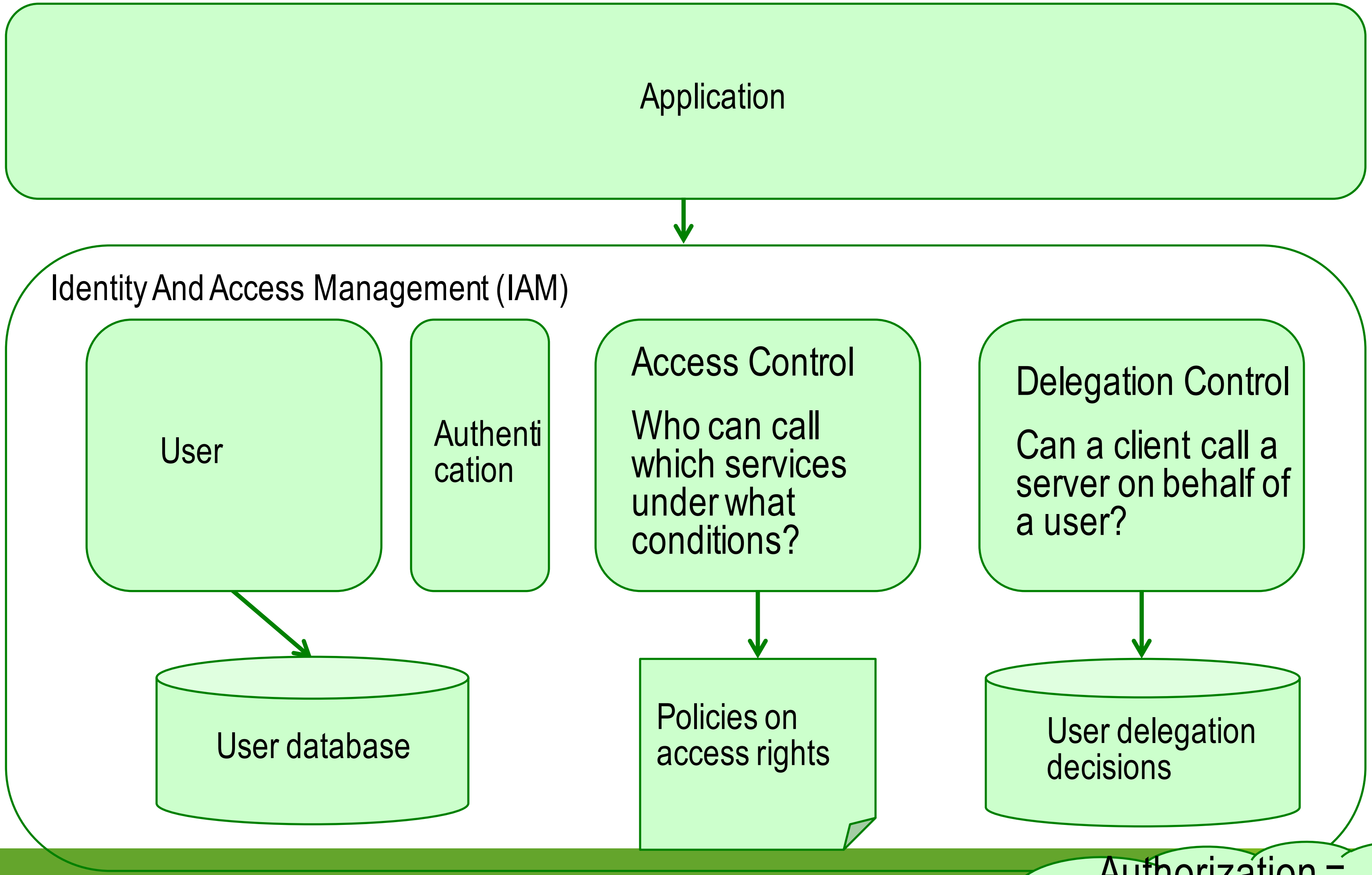
Access Control Tailored For REST API

- [HS 15] Hüffmeyer, Schreier: *An Attribute Based Access Control Model for RESTful Services*, SummerSOC 2015
- [HS 16a] Hüffmeyer, Schreier: Designing Efficient XACML Policies for RESTful Services, in: Hildebrandt et al. (Eds.): *Web Services, Formal Methods, and Behavioral Types, Revised Selected Papers,* Springer 2016
- [HS 16b] Hüffmeyer, Schreier: *Analysis of an Access Control System for RESTful Services*, ICWE 2016
- [HS 16c] Hüffmeyer, Schreier: *Formal Comparison of an Attribute Based Access Control Language for RESTful Services with XACML*, ACM SACMAT 2016

Main Question Of This Tutorial

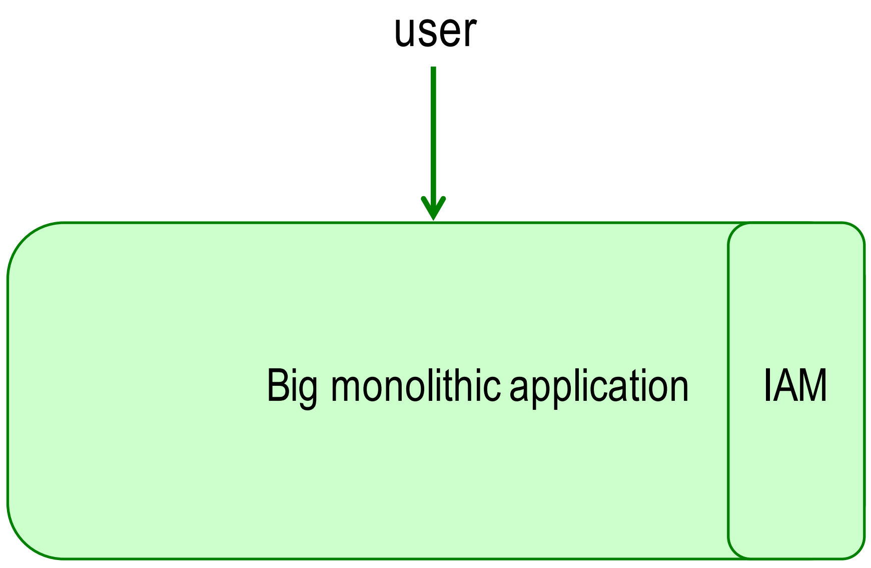
- What is the common ground
- of Identity-and-Access-Management-related standards
- and what are the differences?

Basic Architecture of Identity And Access Management


Application
Identity And Access Management (IAM)
User
Authenti cation
Access Control
Who can call which services under what conditions?
Delegation Control
Can a client call a server on behalf of a user?
User database
Policies on access rights
User delegation decisions

Authorization = Access + Delegation Control

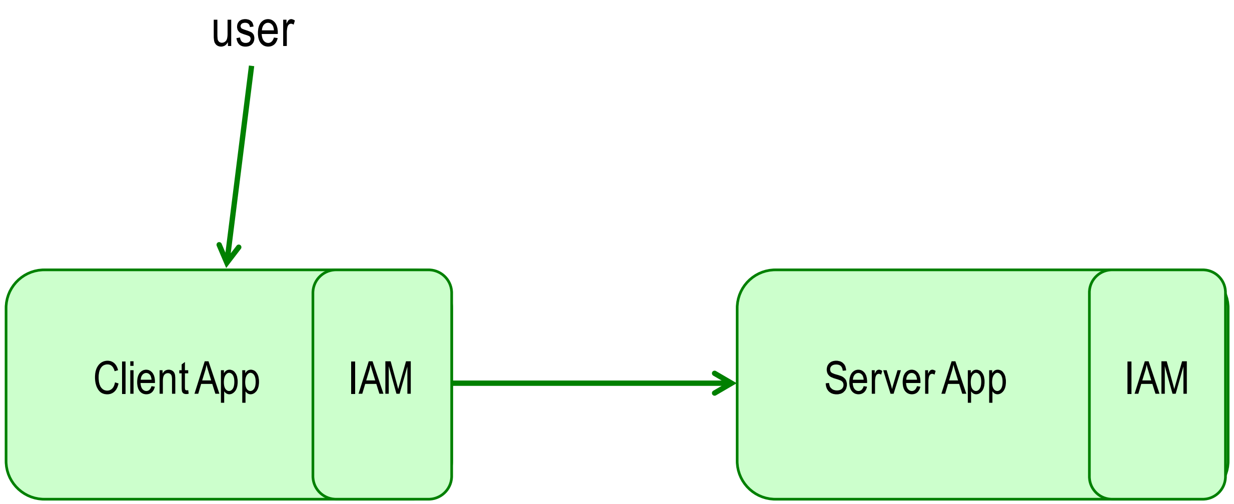
Delegation Control: from big monolithic applications ...



Potential problems:

- Management
- Reuse
- Scalability
- Enterprise View: more than one application

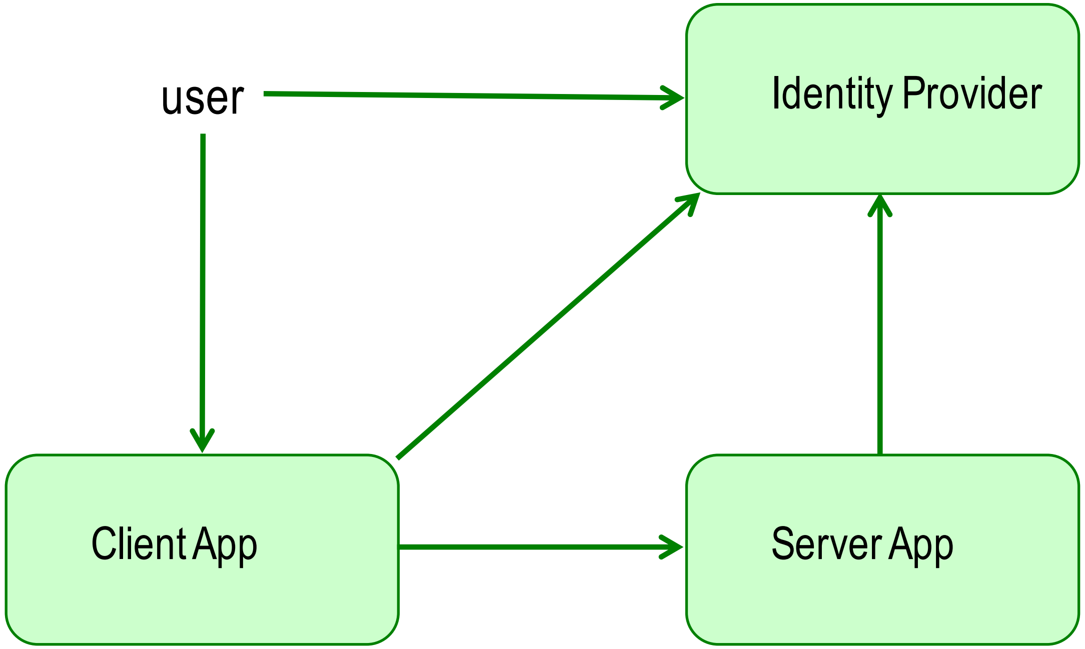
Delegation Control: ... over microservices ...



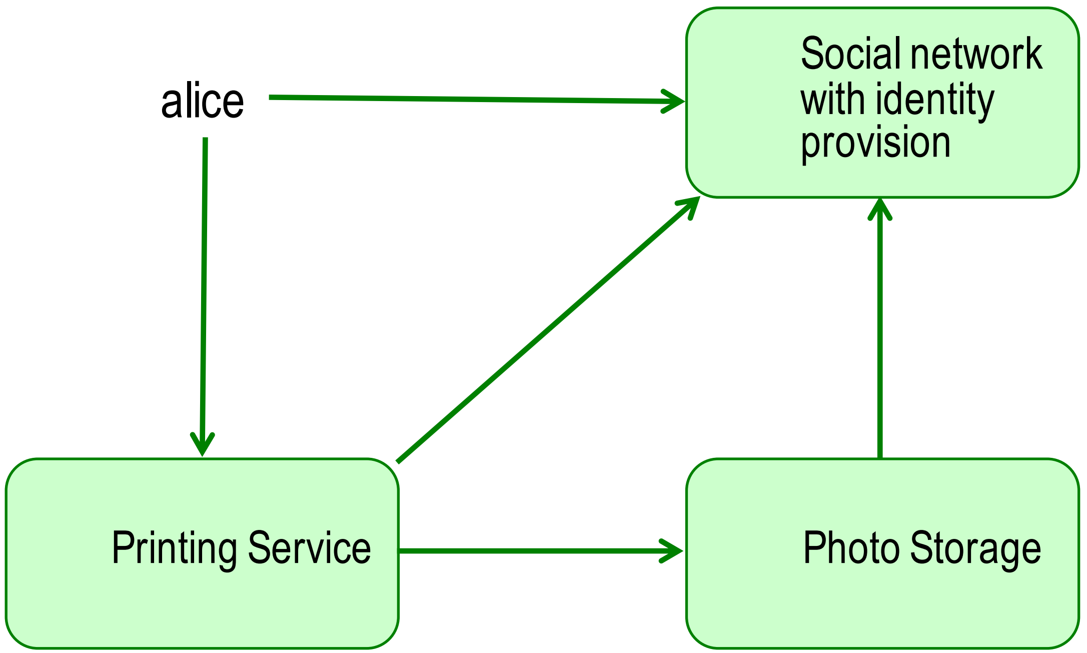
Potential problem:

- Cross cutting concerns
- In particular: doubled IAM

... to identity provision



Example



Google,
Microsoft,
Facebook,
Etc.

Overview

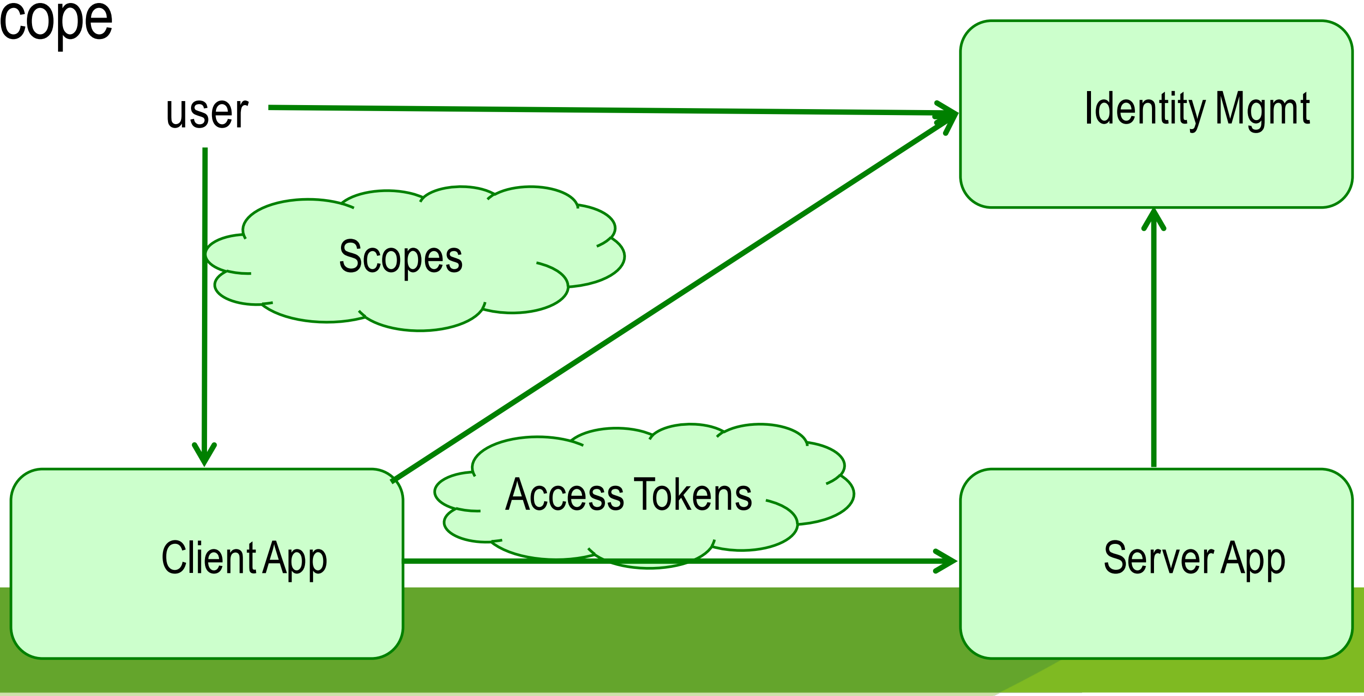
- ✧ Introduction: Identity and Access Management (IAM)
- ✧ Delegation Control
  - **OAuth and OIDC for RESTful services**
  - SAML and Web SSO for web services
- ✧ Access control
- ✧ Conclusions (relationships access / delegation control)

OAuth 2.0 For Delegation Control

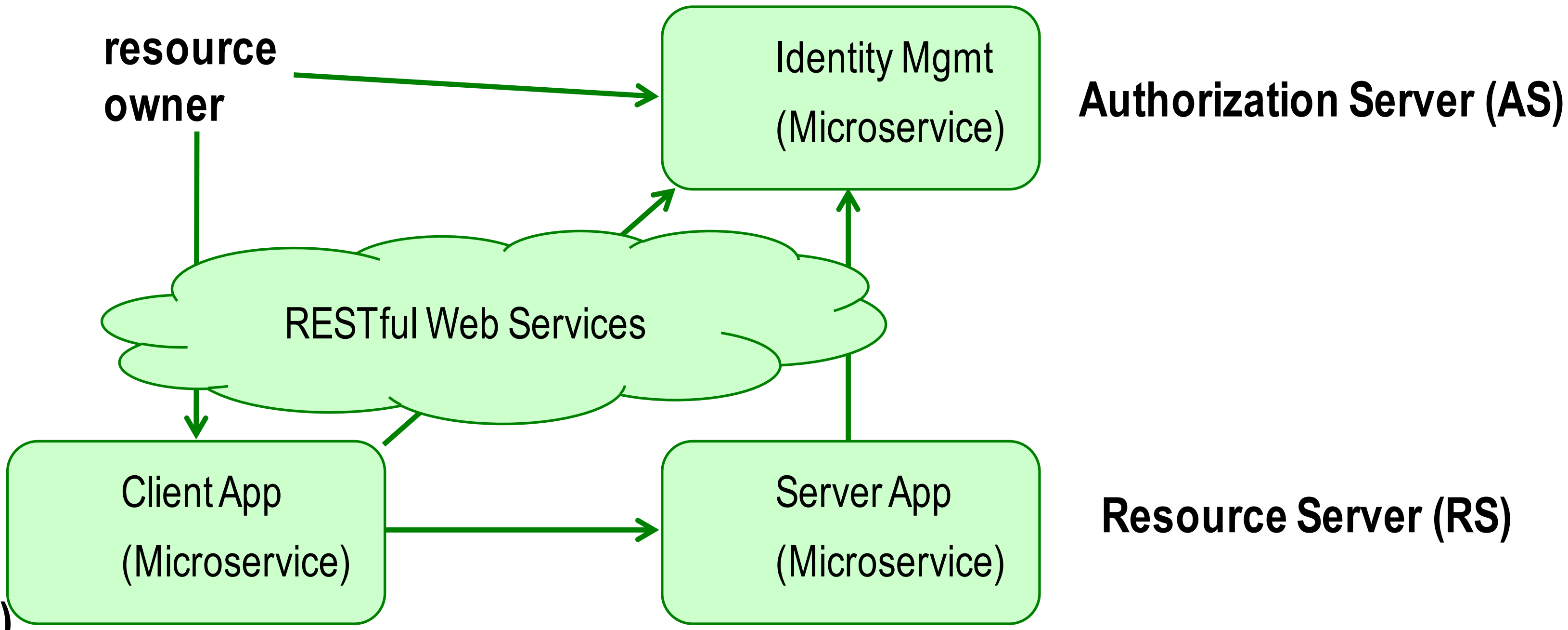
✧ ... answers new authorization questions:

- How to give power to delegates (clients) without revealing too much (passwords)? → access tokens

- How to **restrict** the power of delegates on server apps? → scope



Oauth 2.0 Terminology



resource owner

Identity Mgmt (Microservice)

Authorization Server (AS)

RESTful Web Services

Client App (Microservice)

Server App (Microservice)

Resource Server (RS)

Client (C) on

- **Web server**
- **User-Agent (Web browser)**
- **Native app (Device)**

OAuth 2.0 Basics

- Main ideas
  - OAuth = Open Authorization Framework
  - Token-based protocol
  - Security as simple as possible
    - https for encryption as minimum
    - JWT (JSON Web Token) could be used (e.g. OIDC ID token)

- References: IETF standard [OAuth 12], Text book [RS 16]

Tasks Standardized by OAuth 2.0

- Client registration at AS (see literature for details)
- **Authorization**
- Refreshing tokens (see literature for details)
- Token introspection (another IETF RFC: 7662)
- Revoke of trust in client by user (by deleting access token)

4 Application Areas Of OAuth 2.0

1. Authorization Code : **web server apps**
2. Implicit : **web browser apps** (e.g. JavaScript app)
3. Resource Owner Password Credentials : **native apps** (e.g. smartphone with Objective-C implementation)
4. Client Credentials: **no-user clients** (e.g. client owns resources)

Scope: Hint To Access Right

- Just a string
- RS permits access depending on scope
  - No definition how to map scope to access right
- AS has a list of scopes
- User assigns permitted scopes to client
- AS stores user/scope in a database

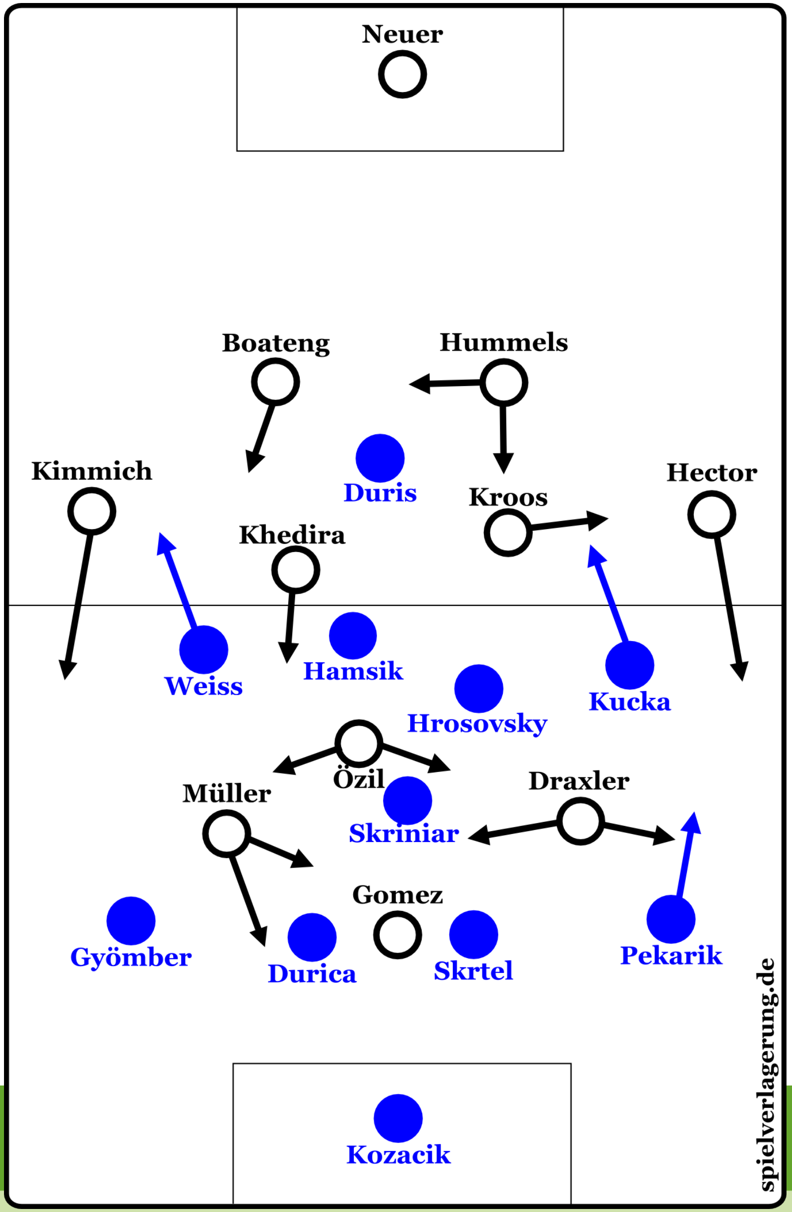
Google examples for scopes

- Drive API v3
  - *View and manage the files in your Google Drive*
    - *https://www.googleapis.com/auth/drive*
  - *View and manage its own configuration data in your Google Drive*
    - *https://www.googleapis.com/auth/drive.appdata*
  - *View and manage Google Drive files and folders that you have opened or created with this app*
    - *https://www.googleapis.com/auth/drive.file*
  - *View and manage metadata of files in your Google Drive*
    - *https://www.googleapis.com/auth/drive.metadata*
  - *View metadata for files in your Google Drive*
    - *https://www.googleapis.com/auth/drive.metadata.readonly*
  - *View the photos, videos and albums in your Google Photos*
    - *https://www.googleapis.com/auth/drive.photos.readonly*
  - *View the files in your Google Drive*
    - *https://www.googleapis.com/auth/drive.readonly*
  - *Modify your Google Apps Script scripts' behavior*
    - *https://www.googleapis.com/auth/drive.scripts*

Best practice:
use URL format
for scope strings

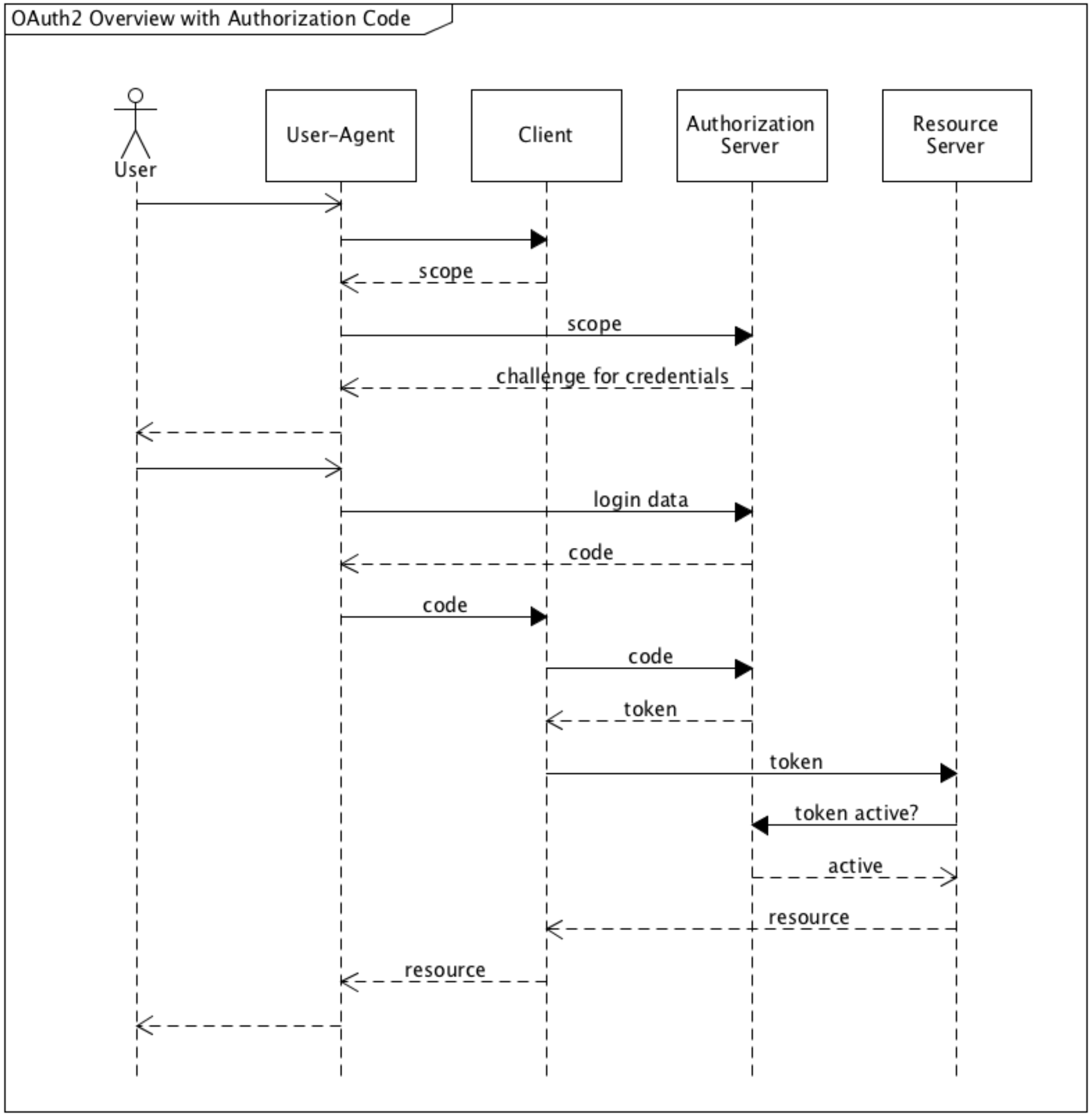
https://developers.google.com/oauthplayground/

Soccer Protocols: Passing Game, Double Passes And Routes



Analysis Chart
European Soccer
Championship 2016
Germany - Slovakia

Complete Passing Game of the Team



Assumptions

- Client registered
- Success case
- Web server app
- Authentication by name/password
- Token validation by introspection

Authorization Request

- Send as redirect request to user agent with query parameters
  - response_type = ”code” REQUIRED
  - client_id REQUIRED
  - redirect_uri (client endpoint called after authorization) OPTIONAL
  - scope OPTIONAL
  - state (against cross-site request forgery) RECOMMENDED
- Example http request of web browser [OAuth 12]

  -
    ```
    GET /authorize
      ?response_type=code
      &client_id=s6BhdRkqt3
      &state=xyz
      &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
    Host: server.example.com
    ```

Access Token Response

- Content with
  - Access token REQUIRED
  - Refresh information OPTIONAL
  - Additional application specific information OPTIONAL
- Example http response [OAuth 12]

  -
    ```
    HTTP/1.1 200 OK
    Content-Type: application/json;charset=UTF-8
    Cache-Control: no-store
    Pragma: no-cache

    {
      "access_token":"2YotnFZFEjr1zCsicMWpAA",
      "token_type":"example",
      "expires_in":3600,
      "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
      "example_parameter":"example_value”
    }
    ```

Resource Server Workflow

1. Validate access token
   - Alternatives:
     a) **Check signature of JWT**
        - Is access token really from AS?
        - Requires PKI
        - No real-time revoke possible
        - fast

     b) **Retrieve access token directly from AS database**
        - Real-time revoke possible
        - Slower than a)
        - Not in control of AS
        - AS and RS should be local neighbours

     c) **Request response from AS Introspection Point**
        - Real-time revoke possible
        - Slower than b)
2. Is client correct? No hacking?
3. Interpret scope of access token
4. Interpret additional information, e.g. user-id and resource-id
5. Depending on interpretation, decide what kind of information to return (e.g. a photo of user alice)

Resource Request From C To RS

Example [OAuth 12]

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

What an Authorization Server is doing

Authorization Endpoint

1. Receive scope from C
2. Ask user in a dialog to authenticate and to agree
3. **Generate authorization code**
4. **Store (code, client) in database**
5. Give C as redirect over user-agent an authorization code

Token Endpoint

1. Receive code from C
2. **Check with database**
3. **Store (access token, client, scope, expiration, user, ...) in database**
4. Respond access token to C

Introspection Endpoint

1. Receive access token from RS
2. **Check with database**
3. Return true/false and (access token, client, scope, expiration, user, ...) to RS

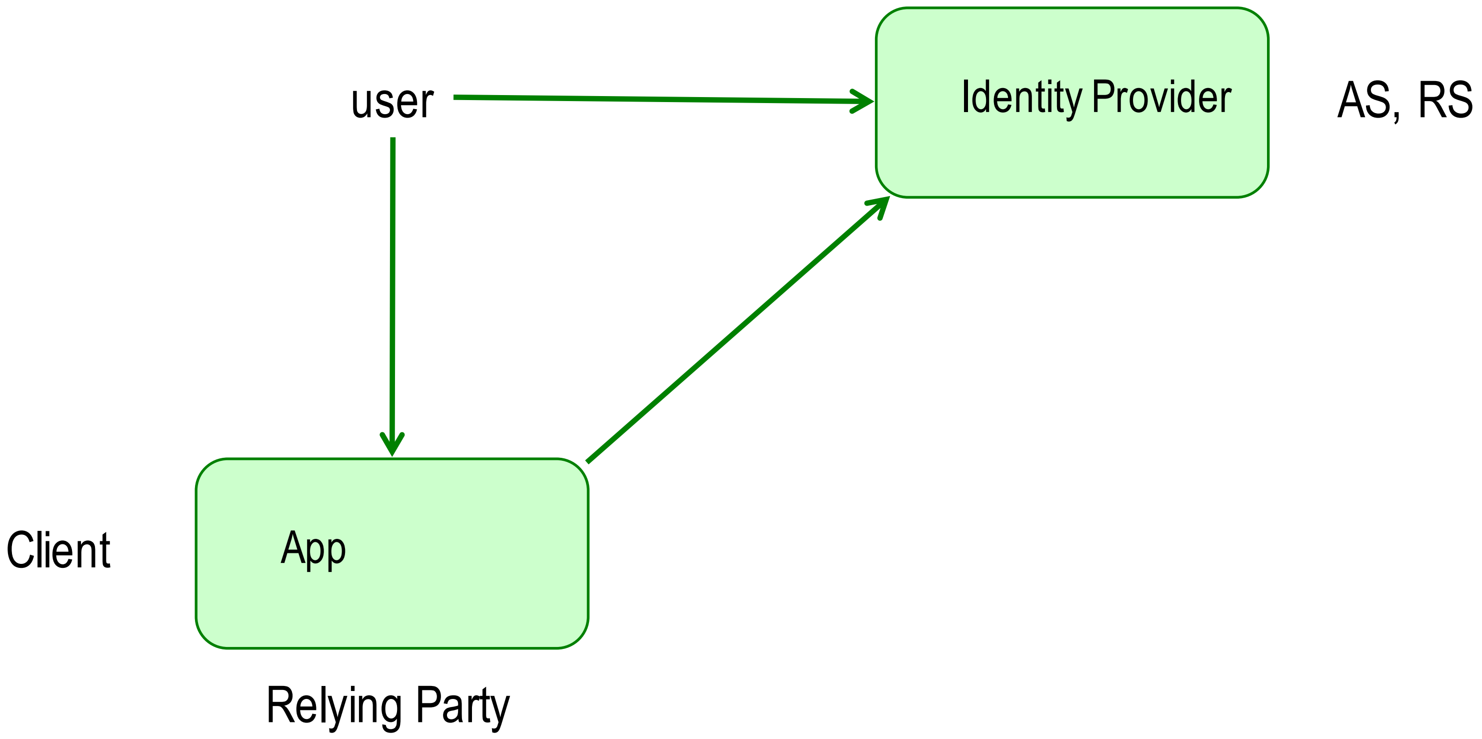
Open ID Connect (OIDC)

- Industry standard of OpenID Foundation
- Many offerings of OpenID: Google, Microsoft, Facebook, etc.
- Goal: Support of **authentication**
- OAuth 2.0 Profile (i.e. specialization of OAuth protocol)

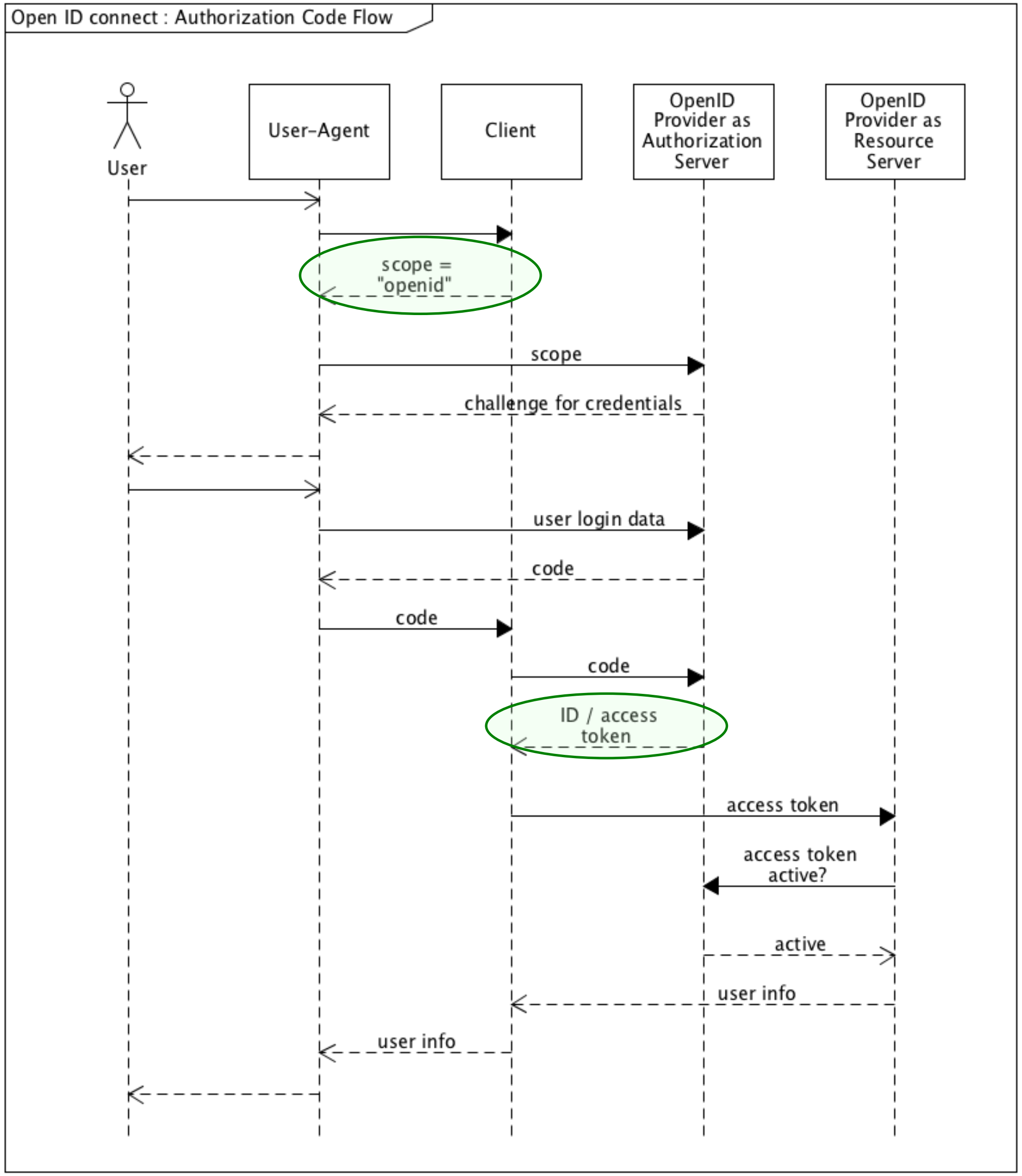
OIDC As OAuth 2.0 Profile

- Specialization of OAuth
  - User (nothing else) as scope and resource
- Variations
  - Authorization code flow (**web server app**)
  - Implicit Flow (**user-agent-based app**)
  - Hybrid Flow

OIDC big picture



OIDC Authorization Code: Passing Game and Routes



OIDC as OAuth Profile: Specialities

- ✧ Special scope: “openid”
- ✧ 2 tokens
  - ID token
    - **Confirmation**: authentication event and its context
    - JSON web token (IETC RFC 7519)
Format: Header . Payload . Signature
  - Access token
    - access of user info (could expire later than ID token)
    - OAuth delegation to access protected resource

Successful OIDC Token Response

```
HTTP/1.1 200 OK
 Content-Type: application/json
 Cache-Control: no-store
 Pragma: no-cache

 {
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "refresh_token": "8xLOxBtZp8",
  "expires_in": 3600,
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
    yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
    NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
    fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
    AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
    Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
    NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
    QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
    K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
    XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
 }
```

Example [OIDC 14]

Example ID Token (enclosed in JWT) [OIDC 14]

```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "auth_time": 1311280969,
}
```

Unique id, second login delivers same id

- ✧ issuer (OpenID provider)
- ✧ subject (string for end user)
- ✧ audience (Oauth 2.0 client_id)
- ✧ (against replay attacks)
- ✧ expiration time
- ✧ issue time
- ✧ authentification time

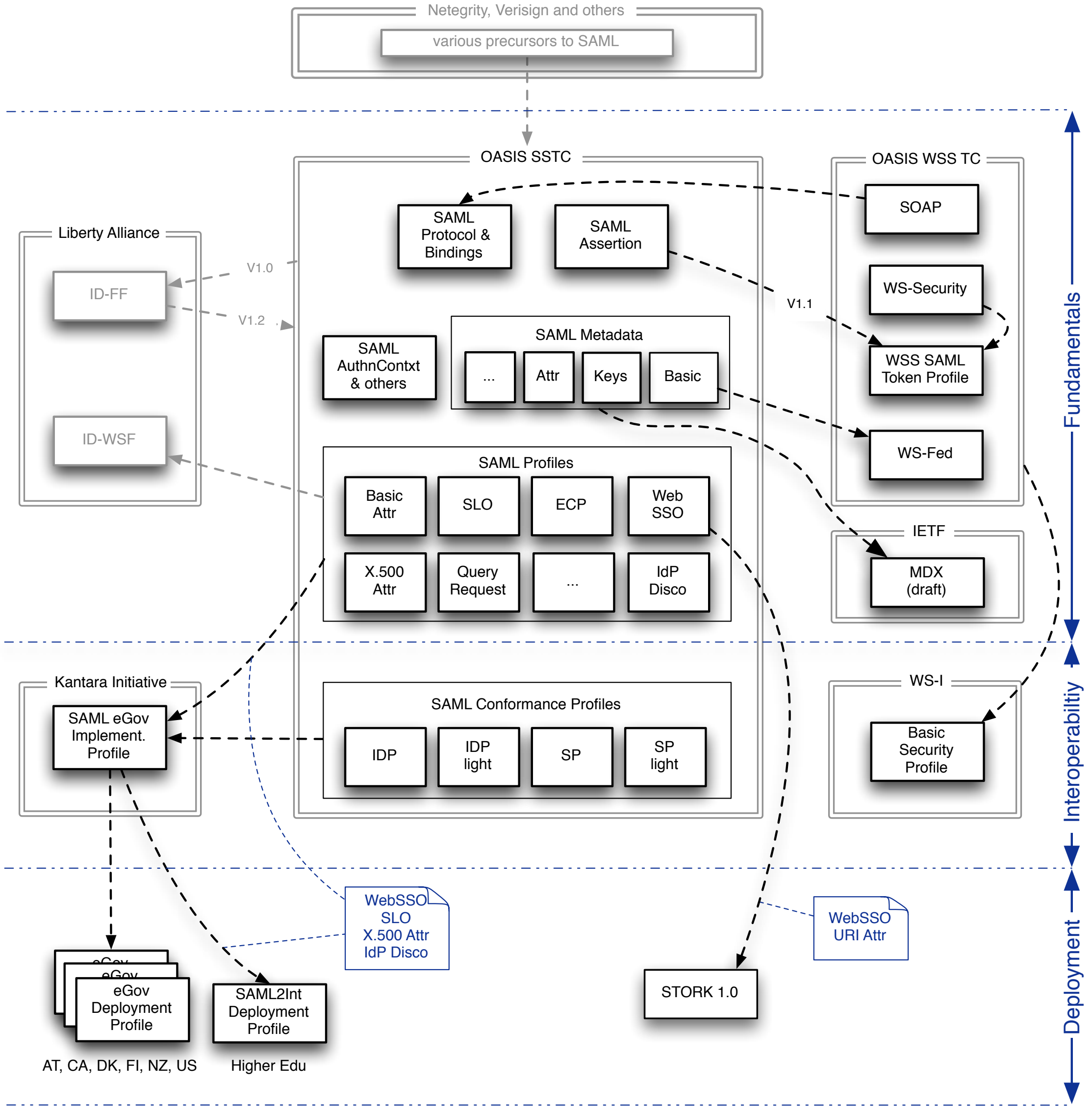
Overview

- ✧ Introduction: Identity and Access Management (IAM)
- ✧ Delegation Control
  - OAuth and OIDC for RESTful services
  - **SAML and Web SSO for web services**
- ✧ Access control
- ✧ Conclusions (relationships access / delegation control)

SAML 2.0 Basics

- Security Assertion Markup Language [SAML 08]
- Industry standard from OASIS
- Set of XML Schema definitions
  - Extensible framework of super/subtypes
  - Processing rules in addition
- Concepts
  - Security Assertions
  - Protocol (request/response formats, used by profiles)
  - Many SOAP and HTTP based bindings
  - Profiles

SAML Specification Family



Kantrara Federation Interoperability WG
CC BY-SA 23-Aug-2012

Source:
kantarainitiative.org
WG Federation Interoperability
SAML Interoperability
and Dependencies
CC BY-SA

3 Major SAML Profiles (Complete protocols)

- Single-Sign On (Authentication) with 5 subprofiles
- Assertion Query/Request

Getting general security information

- SAML Attribute
  - X.500/LDAP Profile (User Info)
  - XACML Profile (give access? → permit/deny)
  - Etc.

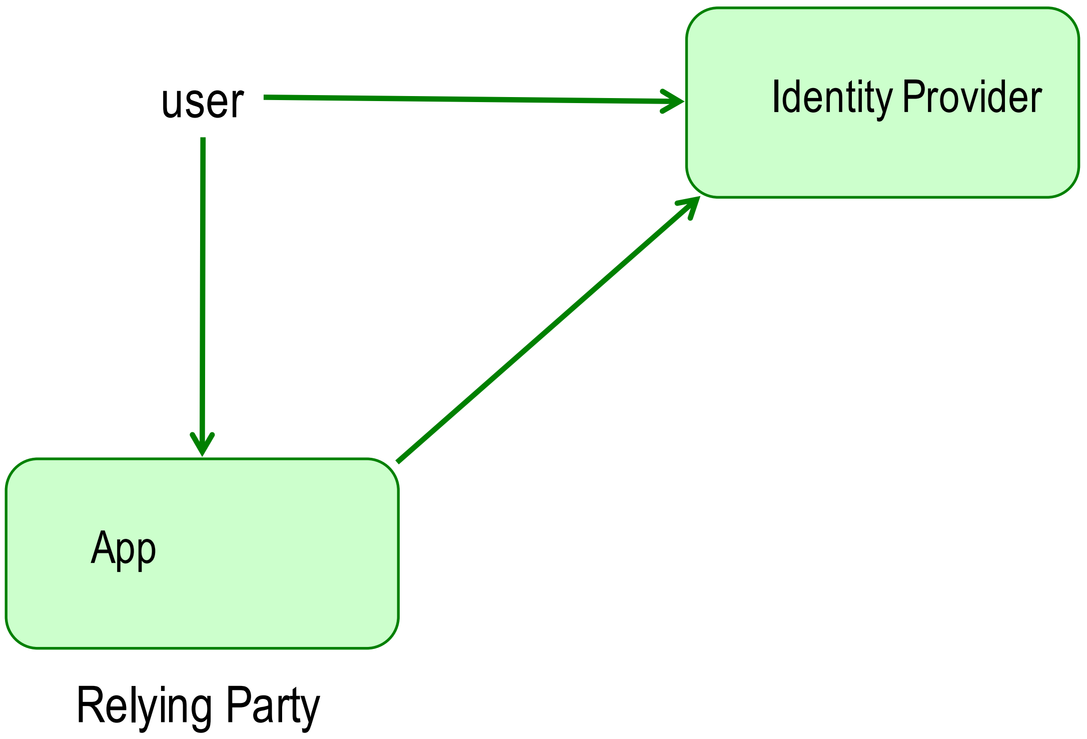
SAML Assertions

- Format for response data
- May be signed/encrypted using **XML Signature/Encryption standard**
- XML elements
  - Id REQUIRED
  - Issuer REQUIRED
  - Signature OPTIONAL
  - Subject OPTIONAL
  - Conditions OPTIONAL
  - Advice OPTIONAL
  - Statement (zero, one ore many)

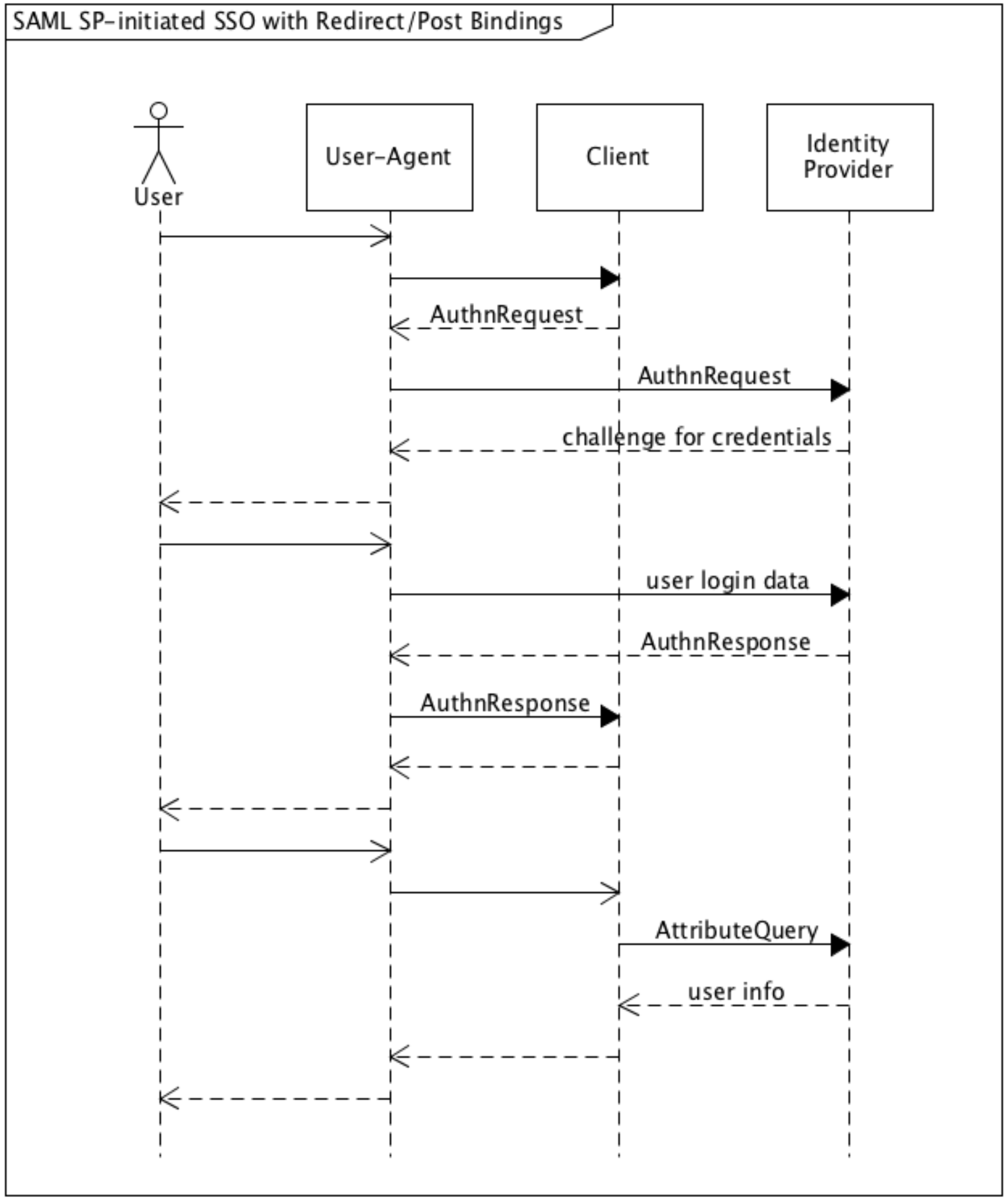
SAML Statement Types

- Authentication
- Authorization Decision (permit/deny access)
- Attribute (Security information, e.g. user data)

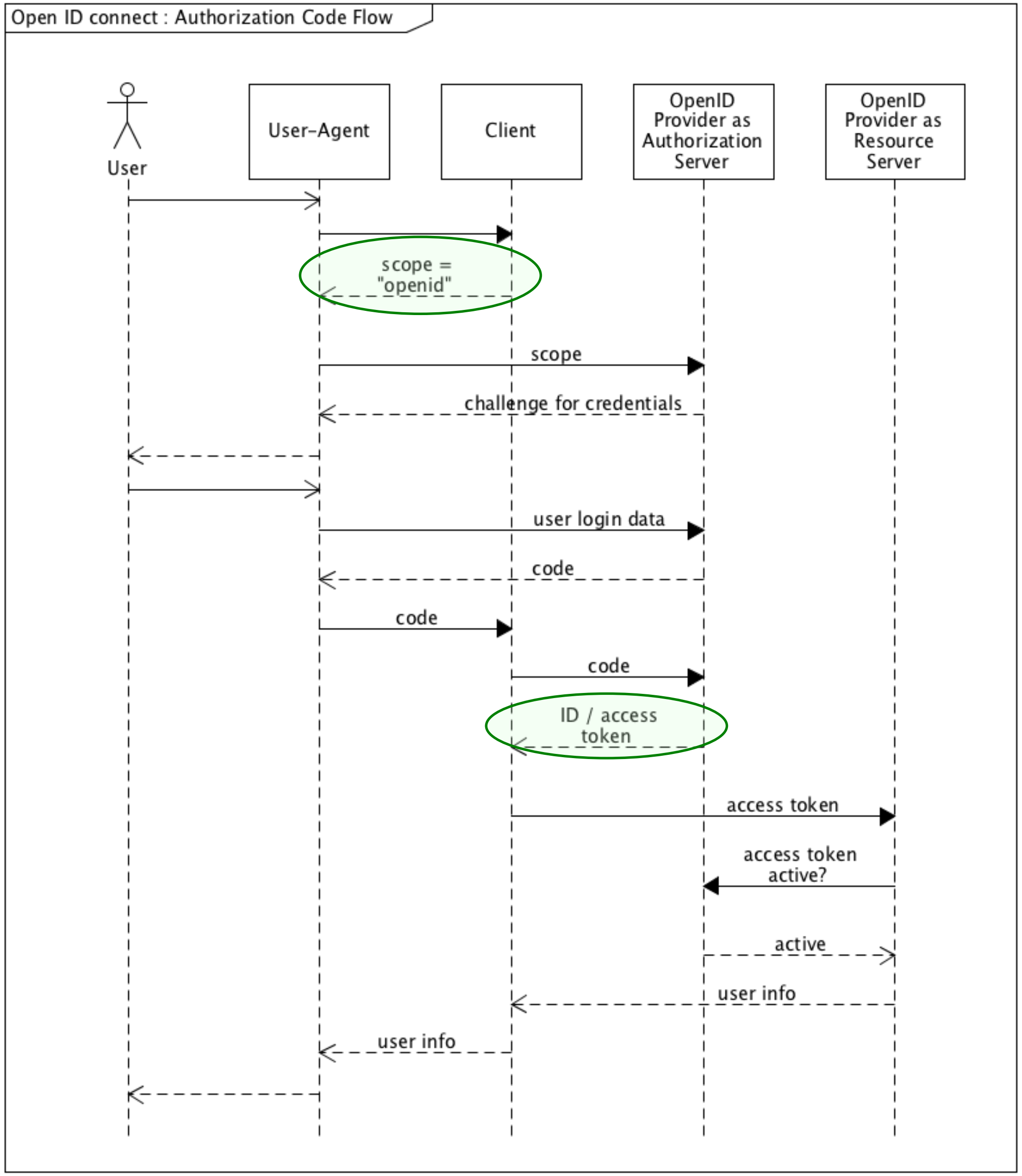
Big Picture SAML Web Single-Sign On



Service Provider Initiated Web Single-Sign On



Reminder: OIDC Authorization Code Flow



SAML Authentication Request [SAML 08]

```
<samlp:AuthnRequest
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol“
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion“
  ID="identifier_1" Version="2.0"
  IssueInstant="2004-12-05T09:21:59Z"
  AssertionConsumerServiceIndex="1">
    <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
   <samlp:NameIDPolicy  AllowCreate="true"
     Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
</samlp:AuthnRequest>
```

SAML Authentication Response [SAML 08]

```
<samlp:Response
 xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
 ID="identifier_2"  InResponseTo="identifier_1"  Version="2.0"
 IssueInstant="2004-12-05T09:22:05Z"
 Destination="https://sp.example.com/SAML2/SSO/POST">
 <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
 <samlp:Status>  <samlp:StatusCode  Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>  </samlp:Status>
 <saml:Assertion  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  ID="identifier_3"  Version="2.0"
 IssueInstant="2004-12-05T09:22:05Z">
  <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
  <!-- a POSTed assertion  MUST  be signed  -->
  <ds:Signature  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
  <saml:Subject>
    <saml:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">  3f7b3dcf-1674-4ecd-92c8-1544f346
   <saml:SubjectConfirmation  Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
     <saml:SubjectConfirmationData  InResponseTo="identifier_1"
      Recipient="https://sp.example.com/SAML2/SSO/POST"
      NotOnOrAfter="2004-12-05T09:27:05Z"/>
   </saml:SubjectConfirmation>
  </saml:Subject>
  <saml:Conditions  NotBefore="2004-12-05T09:17:05Z"  NotOnOrAfter="2004-12-05T09:27:05Z">
   <saml:AudienceRestriction>  <saml:Audience>https://sp.example.com/SAML2</saml:Audience>  </saml:AudienceRestriction>
  </saml:Conditions>
  <saml:AuthnStatement  AuthnInstant="2004-12-05T09:22:00Z"  SessionIndex="identifier_3">
   <saml:AuthnContext> <saml:AuthnContextClassRef>  urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport  </saml:AuthnContextClassRef>  </saml:AuthnContext>
 </saml:AuthnStatement>
</saml:Assertion>
</samlp:Response>
```

Assertion

Subject Identifier

Lifetime etc.

Authentication Statement

Reminder: Successful OIDC Token Response [OAuth 12]

```
HTTP/1.1 200 OK
 Content-Type: application/json
 Cache-Control: no-store
 Pragma: no-cache

 {
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "refresh_token": "8xLOxBtZp8",
  "expires_in": 3600,
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
    yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
    NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
    fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
    AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
    Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
    NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
    QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
    K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
    XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
 }
```

Reminder: Example ID Token (enclosed in JWT) [OAuth 12]

```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "auth_time": 1311280969,
 }
```

Summary: SAML Web SSO vs. OIDC

	SAML	**OIDC**
Message Format	XML	JSON
Security	Message Level (XML Signature & Encryption)	Message (JWT) & Transport Level (TLS)
Typical Client Apps	Web	Web, Mobile, Desktop, Embeddded

Complex, Flexible

Simple, Restricted

Summary: OAuth vs. SAML

	OAuth	**SAML**
Message Format	JSON	XML
Number of use cases	small	large
Architectural constraint	Keep it as simple as possible	Complex , flexible generic framework with many specializations
Basic idea	Protocol for user consenting delegation of power (authorization)	Data formats to exchange security assertions

Overview

- ✧ Introduction: Identity and Access Management (IAM)
- ✧ Delegation Control
  - OAuth and OIDC for RESTful services
  - SAML and Web SSO for web services
- ✧ **Access control**
- ✧ Conclusions (relationships access / delegation control)

Access Control Solutions

- Role based access control (RBAC)
- Access control list (ACL)
- Procedural access control
  - inside of service implementation
  - e.g. many business apps extending RBAC in this way
- **Attribute based access control (ABAC)**

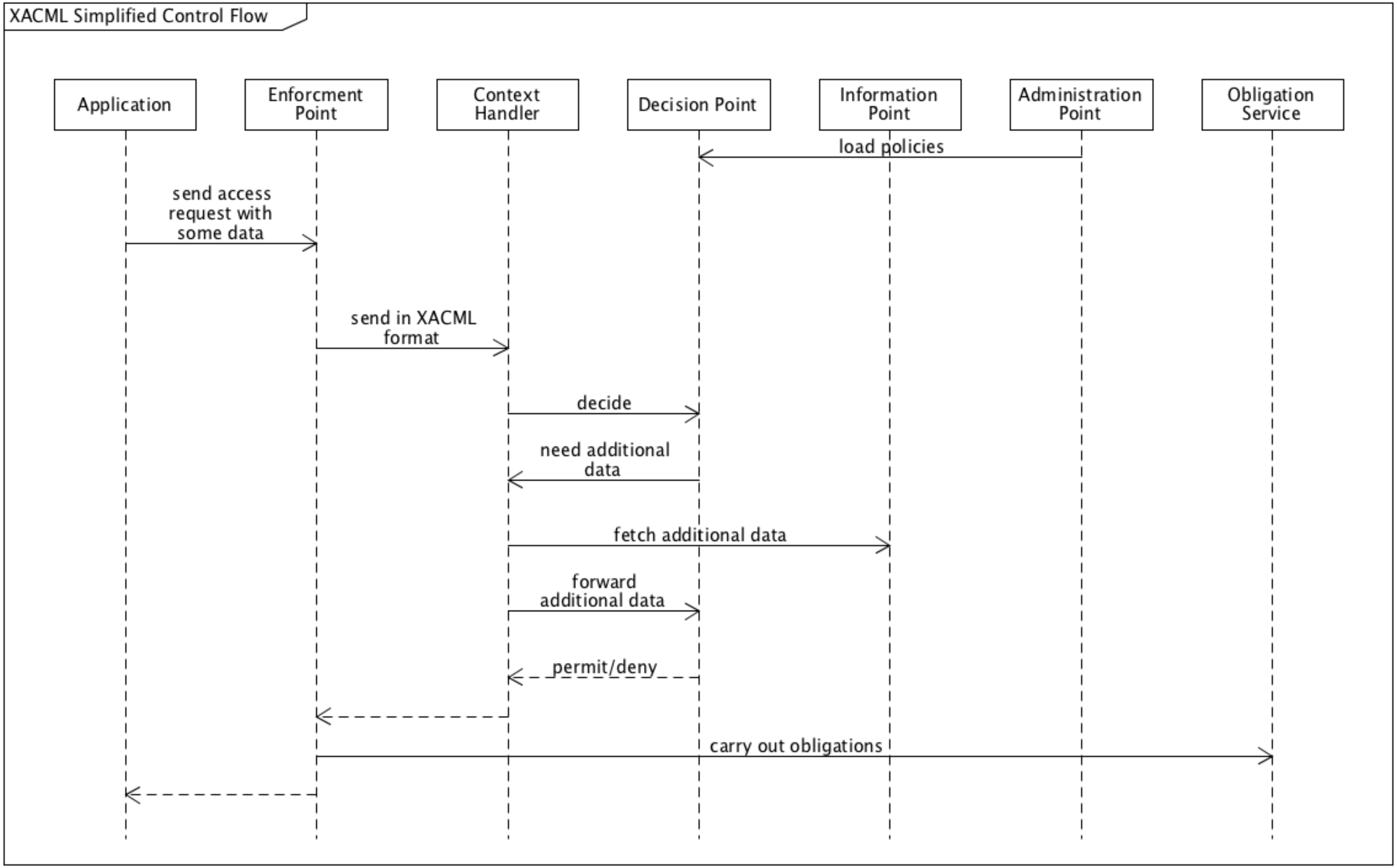
Attribute based access control (ABAC)

- Rules
  - Conditions of attributes about
    - User
    - Resources
    - Environment (time, device, etc.)
- ABAC
  - IAM with XACML,
  - Microsoft dynamic access control

XACML Basics

- OASIS standard version 3.0 [XACML 13]
- Complex XML syntax, hard to read
- ALFA: Abbreviated Language for Authorization (easy to read syntax) [ALFA 15]
- Many additional profiles for special variations
- Components
  - Rule language
  - Protocol (communication flow)
  - Message formats
    - Access request and decision response

XACML Protocol



XACML Components and Messages

- PEP: Policy Enforcement Point
  - Transforms request/response from application format to XACML
  - Calls obligation services
- Context Handler: Coordination Point
- PDP: Policy Decision Point
  - Rule engine
- PAP: Policy Administration Point
  - Policy storage
- PIP: Policy Information Point
  - Mediator retrieving additional data

Request/Response PEP <-> PDP

- Access
  - Request format with (some) attribute values
  - Responses with decisions
    - Permit / Deny
    - Indeterminate, NotApplicable
    - Obligations (access control actions after a decision, e.g. logging)
    - Advices (returning messages)
- Formats for access messages (not rules)
  - XACML basic XML format
  - **SAML profile: SAML syntax**
  - **JSON profile: JSON**
  - REST profile: endpoint definitions

XACML Rules

- 4 parts
  1. attributes (conditions)
     - on data of
       - resources,
       - user,
       - environment (device, time, …)
       - other information sources
  2. Action: permit / deny
  3. Obligations
  4. Advices

XACML Policies, Sets And Combining Strategies

- ✧ Nested policy sets
  - Controlled by target conditions
- ✧ Complex nested combining strategies
  - **Deny-override**: first deny rule evaluation decides
  - **Permit-override**: first permit decides
  - **First-applicable**: first applicable rule decides (either permit or deny)
  - **Only-one-applicable**: if not only one, result is “indeterminate“
  - Each set has its own combining

XACML Example (using ALFA Syntax)

```
policyset {
  apply denyOverrides
  target clause URLresource == "https://example.com/physicians"
    policy {
      apply permitOverrides
      target clause actionMethod == "GET“
        rule {
          target clause subjectName == ”Alice”
          permit
        }
    }
}
```

PolicySet:

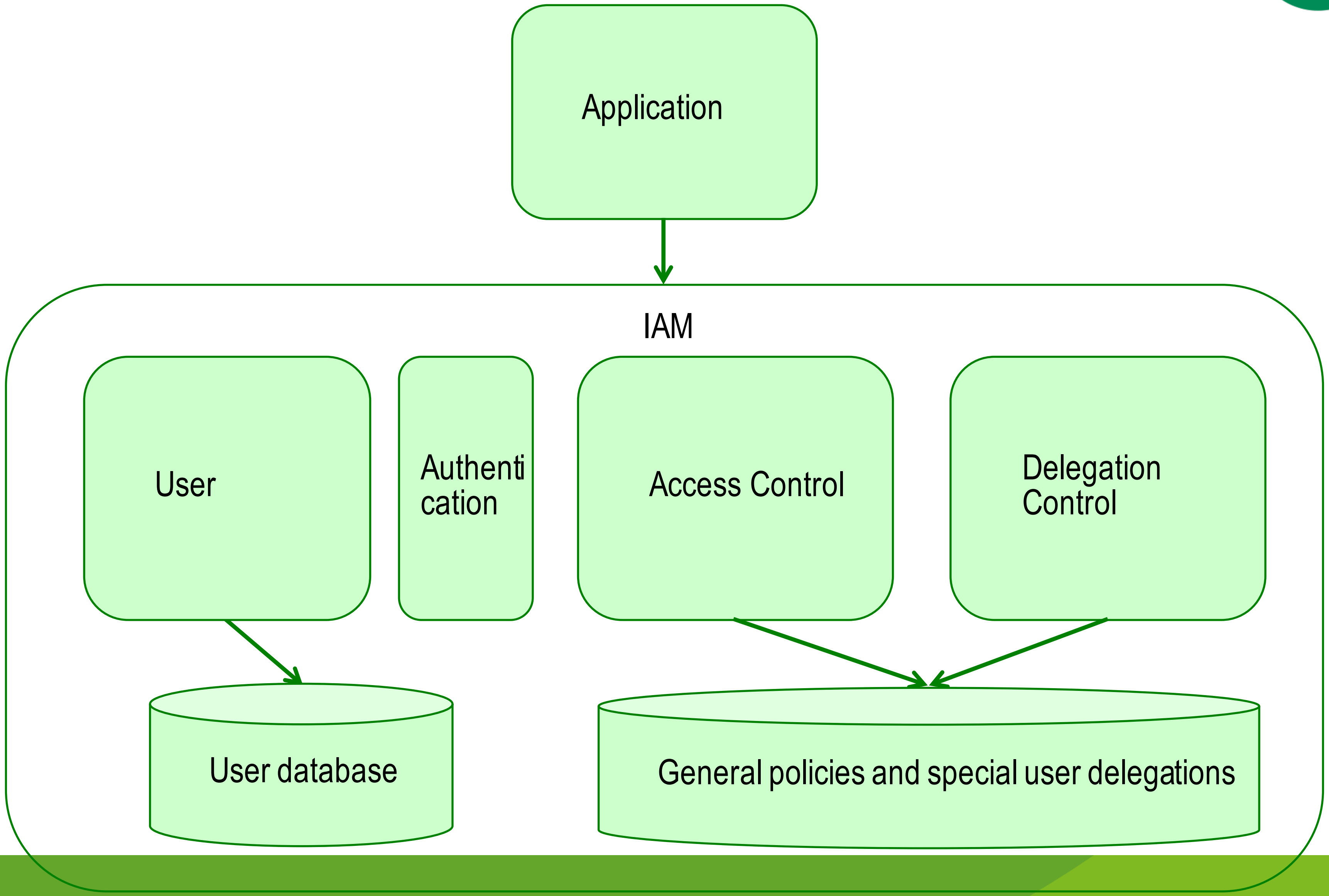
Nested sequence of

If-statements

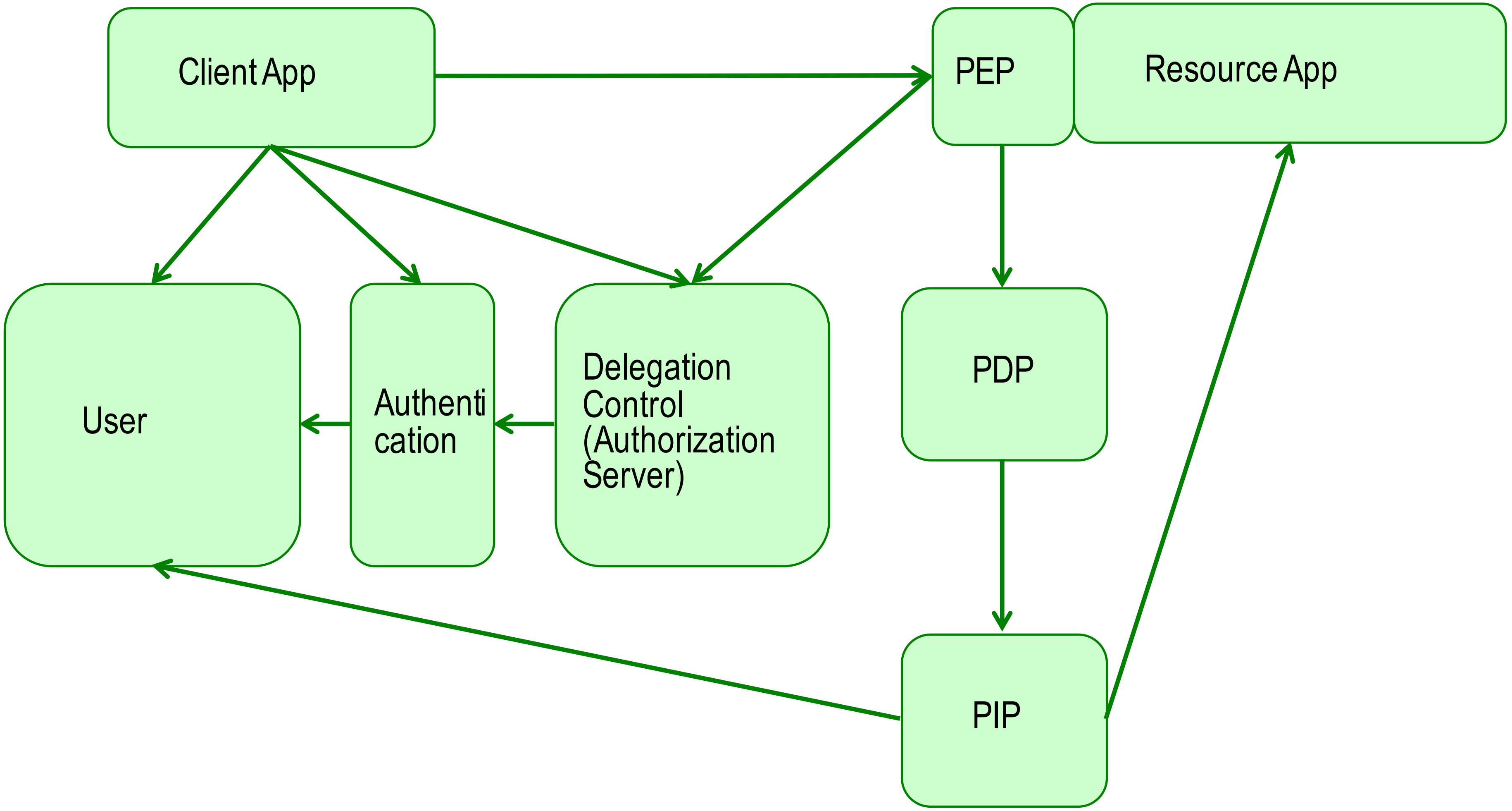
Overview

- ✧ Introduction: Identity and Access Management (IAM)
- ✧ Delegation Control
  - OAuth and OIDC for RESTful services
  - SAML and Web SSO for web services
- ✧ Access control
- ✧ **Conclusions (relationships access / delegation control)**

Basic Architecture of IAM reconsidered



Relationships IAM Components



Evolving Issues

- Coordinated access and delegation control
  - Who calls the PDP?
  - AS: token as cache solution
    - New OAuth profile: User-Managed Access [UMA 15]
  - RS: latest possible point
  - Best method for token validation?
- Integration of Policies
  - Black/Whitelist: General policies
  - Greylist: Ad hoc user decisions
- Attribute based access control tailored for REST API [HS 15], [HS 16a], [HS 16b], [HS 16c],

Thank You!