SAPL Access Control Lessons

Training Series

Dominic Heutelbeck

Lessons Overview

- **Lesson 1:** Access Control Goals and Terminology
- **Lesson 2:** Access Control Models
- **Lesson 3:** ABAC Access Control Mechanisms
- **Lesson 4:** ASBAC and SAPL Fundamentals
- **Lesson 5:** Applying ASBAC and SAPL

SAPL Access Control Lessons

Lesson #01 - Access Control Goals and Terminology

Dominic Heutelbeck

SAPL Webinar #01

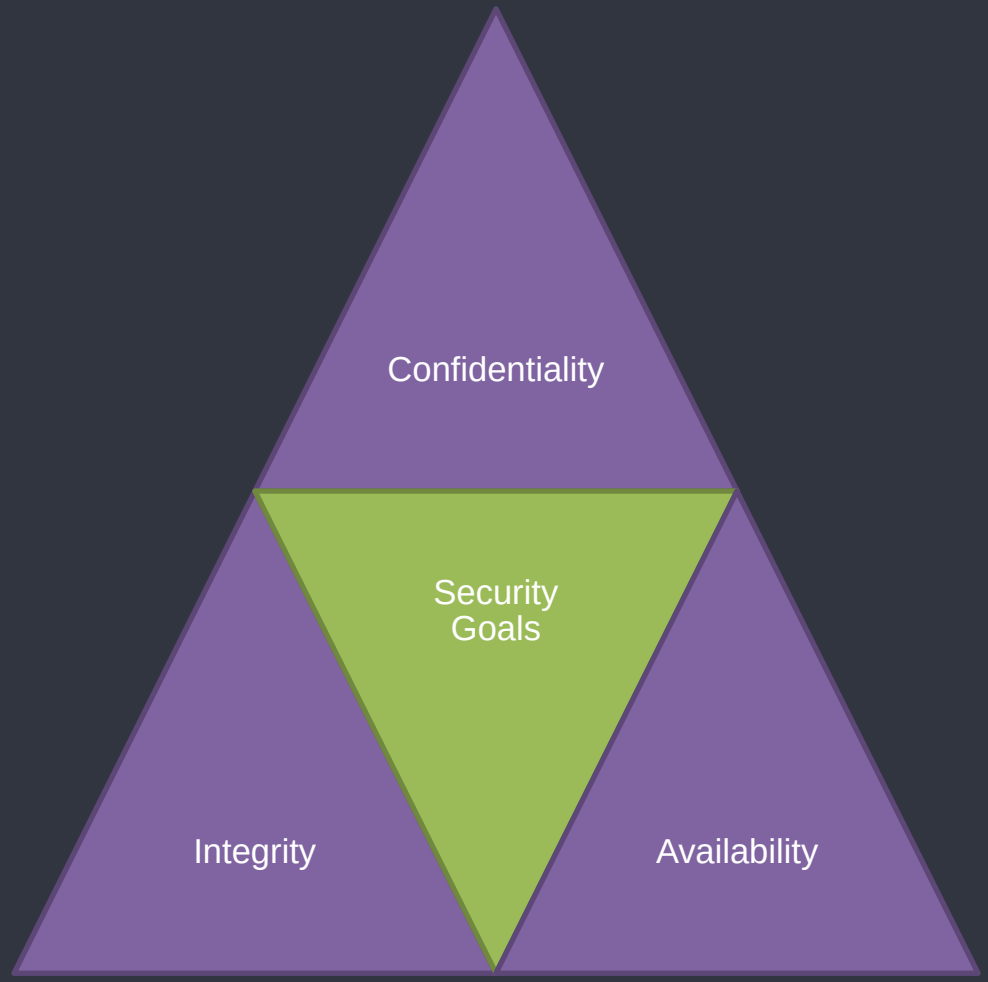
Goal of the Lesson

Challenge - The domain of Access Control is full of jargon.

- Provide an in depth understanding of the underlying principles.
- Gain the competence to map the terminology to the matching principles.
- Know the overall process of access control and some key Access Control Models

Information Security Objectives

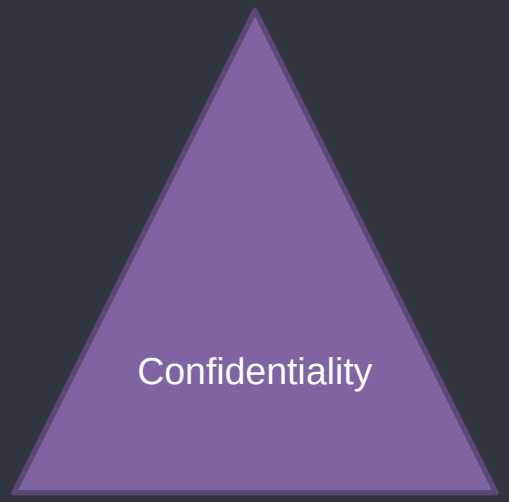
The CIA Triad



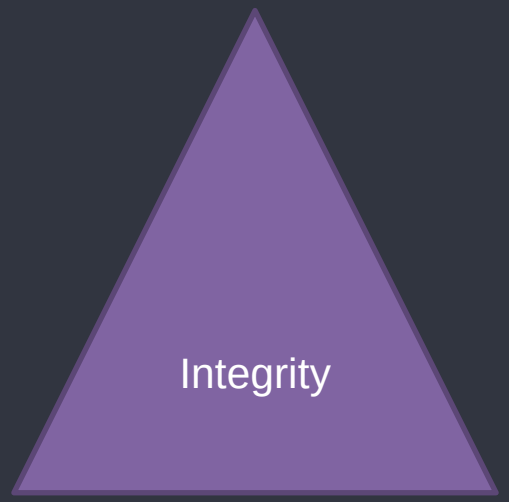
Various extensions do exist.
This is the common core.

E.g., see [ISO27001]

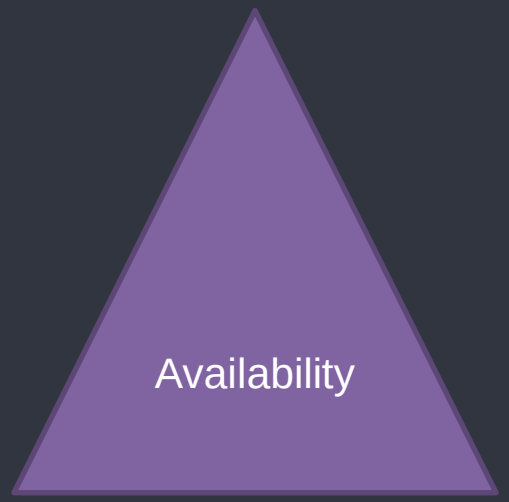
Confidentiality



The property that data is not disclosed to system entities unless they have been **authorized** to know the data. [RFC4949]



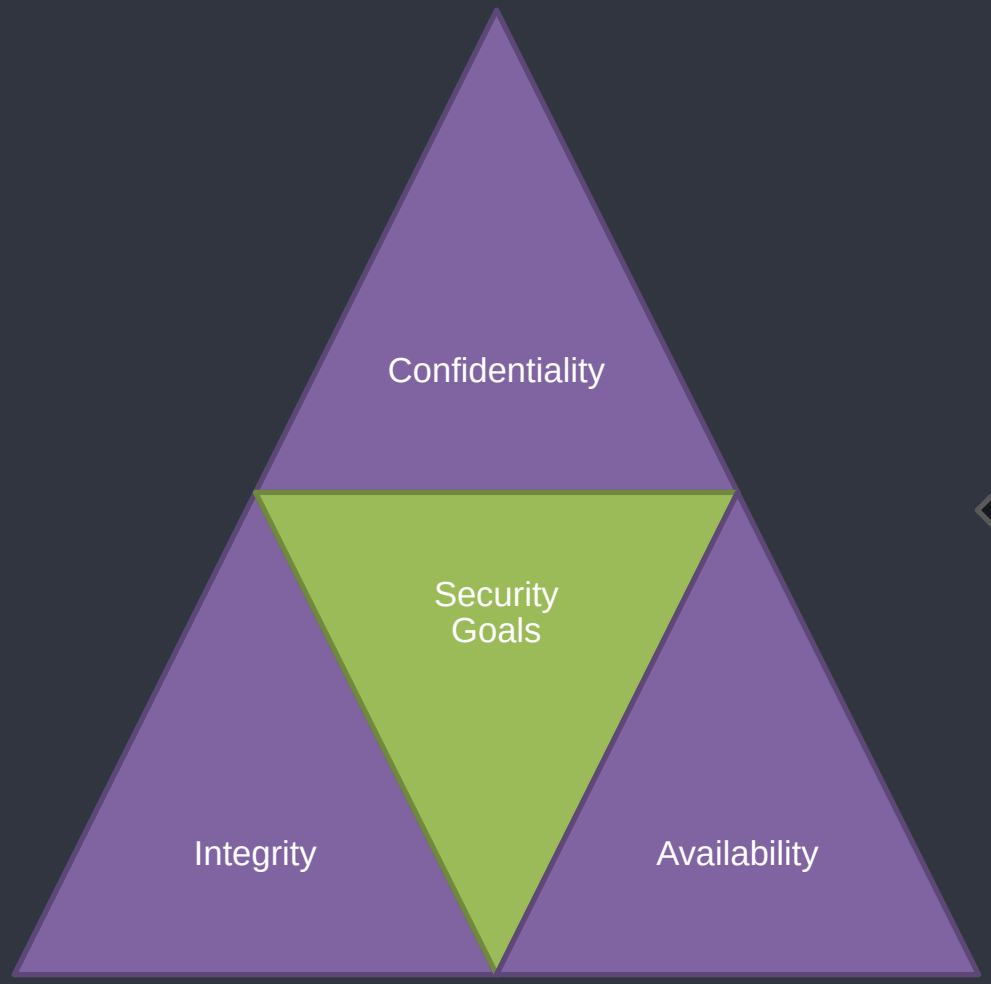
The property that data has not been changed, destroyed, or lost in an **unauthorized** or accidental manner. [RFC4949]



The property of a system or a system resource being accessible, or usable or operational upon demand, by an **authorized system entity,** according to performance specifications for the system; i.e., a system is available if it provides services according to the system design whenever users request them. [RFC4949]

Information Security Objectives

The CIA Triad



Authentication and authorization are important in achieving security goals

Principle of Least Privilege (POLP)

Principle of Least Privilege :

> "Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job." [Saltzer74]

→ You get what you need to do the job, but not more!

Need to Know Principle

Need to Know Principle:

To make information available as needed at the time of need.

→ may be implemented by compartmentalization
- persons know what they must do but not why
- operatives in country A do not need to know details for country B

→ privileges can/should be retracted when the information is no longer needed.

Declare a “need to know” labels/attributes to subjects → basis for lattice models

Separation Of Duty (SoD)

Synonym: Segregation of Duty

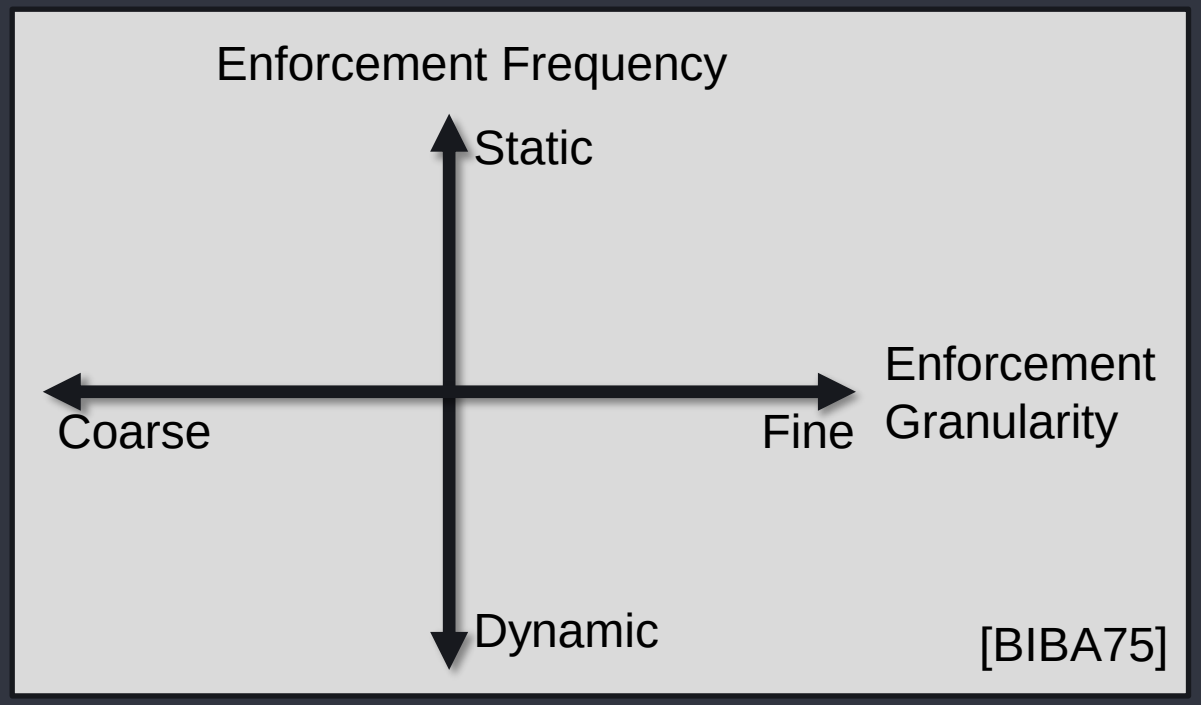
"A basic internal control that prevents or detects errors and irregularities by assigning to separate individuals the responsibility for initiating and recording transactions and for the custody of assets.

Scope Notes: Segregation/separation of duties is commonly used in large IT organizations so that no single person is in a position to introduce fraudulent or malicious code without detection." [ISACTA_SOD]

- Prevent conflict of interest (real or apparent), wrongful acts, frauds, abuse, or errors.
- Detection of security incidents

Example: Ensure the person implementing information security measures is not the same as the one auditing the security measures.

Access Control Granularity and Frequency



"Frequency refers to the time at which access control enforcement occurs." (e.g., once vs. at every access)

"Granularity refers to the size and resolution of the protected system elements"

[BIBA75]

Two key perspectives on granularity [Bossard11] :

- "the expressiveness of the grammar used to express access control rules"
  I.e., the expressiveness of the Access Control Model.
- "the ability of the 'agents' to see more or less information"
  I.e., the ability to tailor information presented to the subject at runtime. Like deciding which components to present in an application or filtering or blackening fields in a dataset.

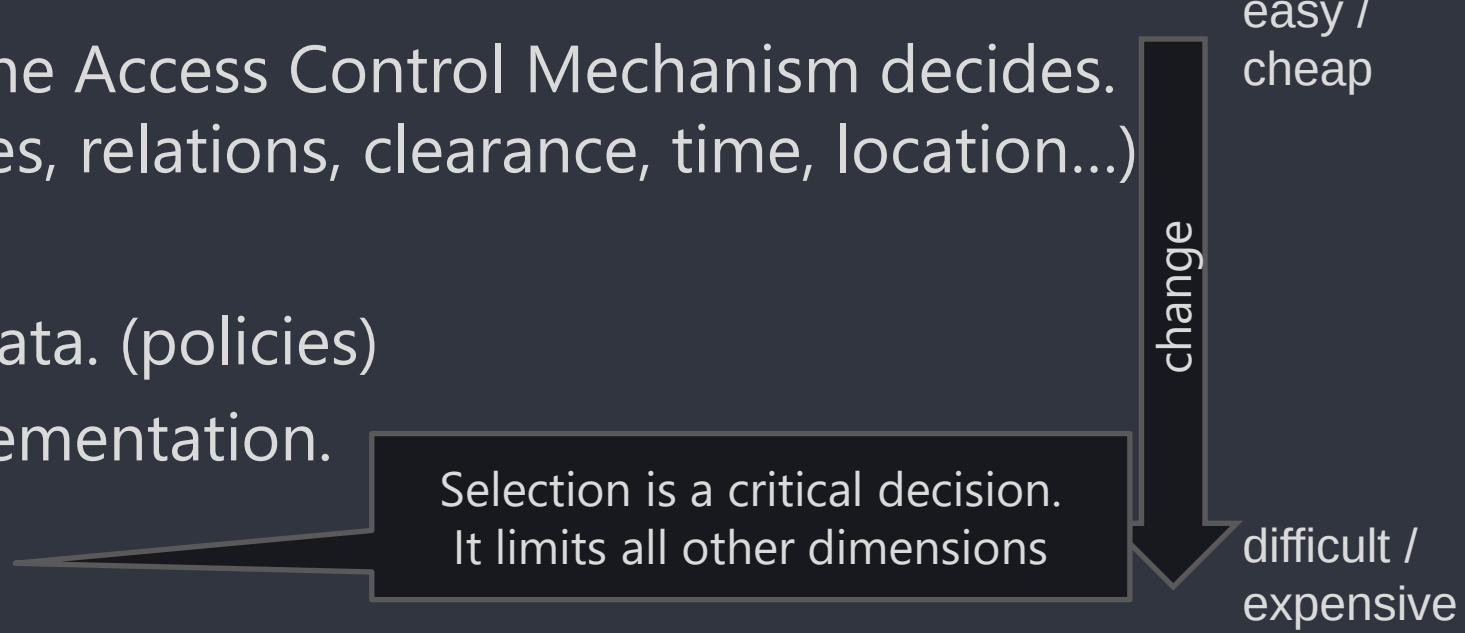
Dynamic Assignment of Privilege

Dynamic Assignment of Privilege:

> "Protection regimes are not constant during the life of a process. They may change as the work proceeds, and in a fully general discussion they should be allowed to change arbitrarily." [Needham72]

Where and what can be dynamic ?

- The values of data upon which the Access Control Mechanism decides. (roles, group, privileges, attributes, relations, clearance, time, location…)
- Which data to use for decisions.
- The rules used to interpret the data. (policies)
- Access Control Mechanism implementation.
- The Access Control Mechanism.



Continuous Access Control

Continuous Access Control: (→ Frequency)

Access should be controlled not only upon initiating the access, but also during access of the resource.

For long-running access the likelihood of access rights changing over time may be significant.
Depending on the nature of the resource unauthorized access may occur.

Example: Only checking permission for opening a data stream for read access.

Scenarios:

- Session-based applications
- IoT Data Streams
- Collaborative tools (CSCW)

Dynamic Authorization Management:

- Refers more to a deployment and administrative style
- Externalization of authorization decision from the application
- Decisions are made at runtime (vs. stored permissions like in DAC)
- Potential centralization of authorization
- Typically referring to an ABAC model such as implemented by a SAPL or XACML-based infrastructure (policy-based)

More marketing terminology than introducing a new concept.

Access (1/2)

General understanding

Access: „The ability and means to communicate with or otherwise interact with a system to use system resources either to handle information or to gain knowledge of the information the system contains.“ [RFC4949]

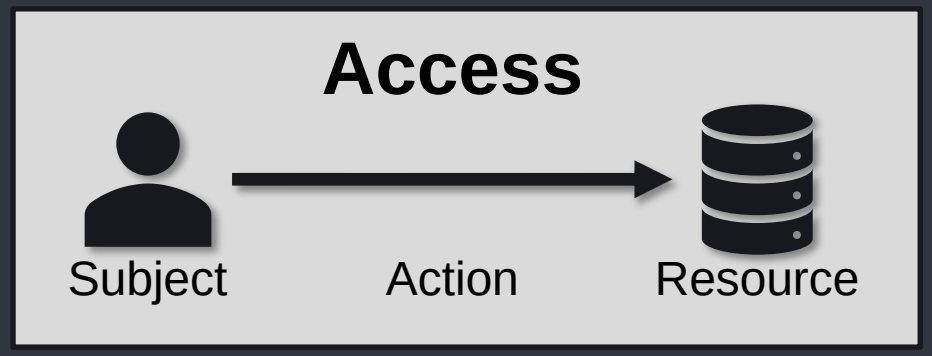
To handle: „Perform processing operations on data, such as receive and transmit, collect and disseminate, create and delete, store and retrieve, read and write, and compare.“ [RFC4949]

To access: Execute upon the ability to access a system. I.e., manipulating its state or gaining knowledge.

„someone has access vs. someone does access“

Access (2/2)

Terminology typically used in access contol models



Resource: "A logical object [...] an entity to be protected from unauthorized use." [NIST800162]

Subject: "[...] the entity requesting to perform an operation upon the object. [...] sometimes referred to as a requestor." [NIST800162]

Action: „An operation on a resource"[XACML] Synonym: operation
Generic: CRUD (create, read, update, delete), search, publish, subscribe
Domain specific: approve plan, onboard machine, perform maintenance, audit

[To] Access: „Performing an action" [XACML]

more precisely:

A subject performing an action on a resource.

Access Control:

„1. Protection of system resources against unauthorized access.
2. A process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy.“ [RFC4949]

“The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.” [ITUTX.800]

Scope here:

- Information Systems
- However, there is strong overlap is certain areas with physical access control.

Authentication &. Authorization (1/2)

Authentication: "Verify (i.e., establish the truth of) an attribute value claimed by or for a system entity or system resource." [RFC4949]

Typically in the context of access control: establishing the identity of the subject.

Authorization: "An **approval** that is granted to a system entity to access a system resource." [RFC4949] (often synonymous: **permission, privilege, access right**)

"A **process** for granting approval to a system entity to access a system resource." [RFC4949]

Authorization Decision:

The outcome of the authorization process.
Typically: "permit", "deny", "permit, and do something", or some error.

Access Control Mechanism:

Some (sub-)system/code section/service performing the authorization.

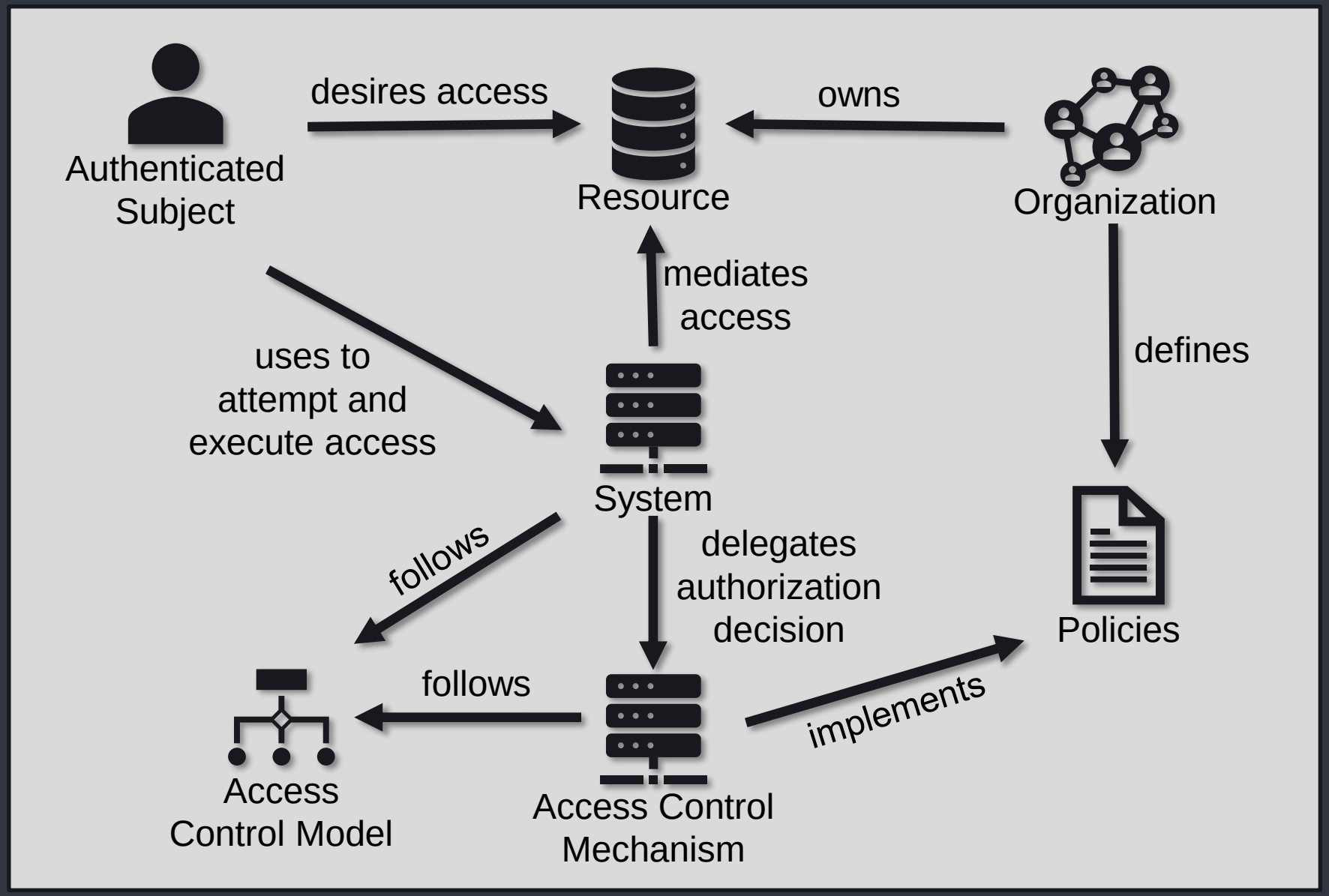
Authentication & Authorization (2/2)

Remarks

- In practice access control covers often both authentication and authorization.
- Typically **authentication of the subject is mandatory before authorization**.
- Anonymity or being “unauthenticated” can be considered an authentication in its own right: It is a truthfully verified predicate of the entity.
- Be precise when discussing Authentication and Authorization!
  The abbreviation “auth” as it is ambiguous.
  Example: OAuth for “Open Authorization” often mistaken as “Open Authentication”
  Better:
  - Authentication → auth**n**
  - Authorization → auth**z**

Authorization In Context

Birds Eye Perspective



- Very general view
- Resource, System, and Authorization Component may collapse into one or two physically deployed units.
- In specific code paths developers have to insert code to enforce access control. Domain specific!
- Still internally this structure prevails.
- Specific models are more fine grained.

- Even more terminology:
  - Access Control Model
  - Policies

Policies (1/2)

Organizations (Enterprises) have specific requirements on how they want or must manage access to resources.

- application domain
- compliance with laws, regulations, contracts, standards

IP

These requirements are the **Inherent Policies** (IP) of the organization/domain.

Documenting the requirements (IP) results in **Natural Language Policies (NLP)**. “Statements gov[illegible]rprise objects. NLPs are human expres[illegible]-enforceable access control policies.” More general: “…transla[illegible]

Attention:
Potential cause of confusion.
There are often „Inherent Policies (IP) govenrning the access to Intellectual Property (IP)“.
When talking with stakeholders, IP is primarily used for Intellectual Property.

Policies (2/2)

Through an appropriate management process, NLPs are translated into **Digital Policies (DP) "**Access control rules that compile directly into machine executable codes or signals" [NIST800162].

NLPs should include requirements on DP usage, management, and evolution. Such policies about policies are called **Meta Policies (MP)** "A policy about policies, or policy for managing policies, such as assignment of priorities and resolution of conflicts between DPs or other MPs." [NIST800162] In consequence, both NLPs and DPs may contain MPs.

SAPL Access Control Lessons

Q&A

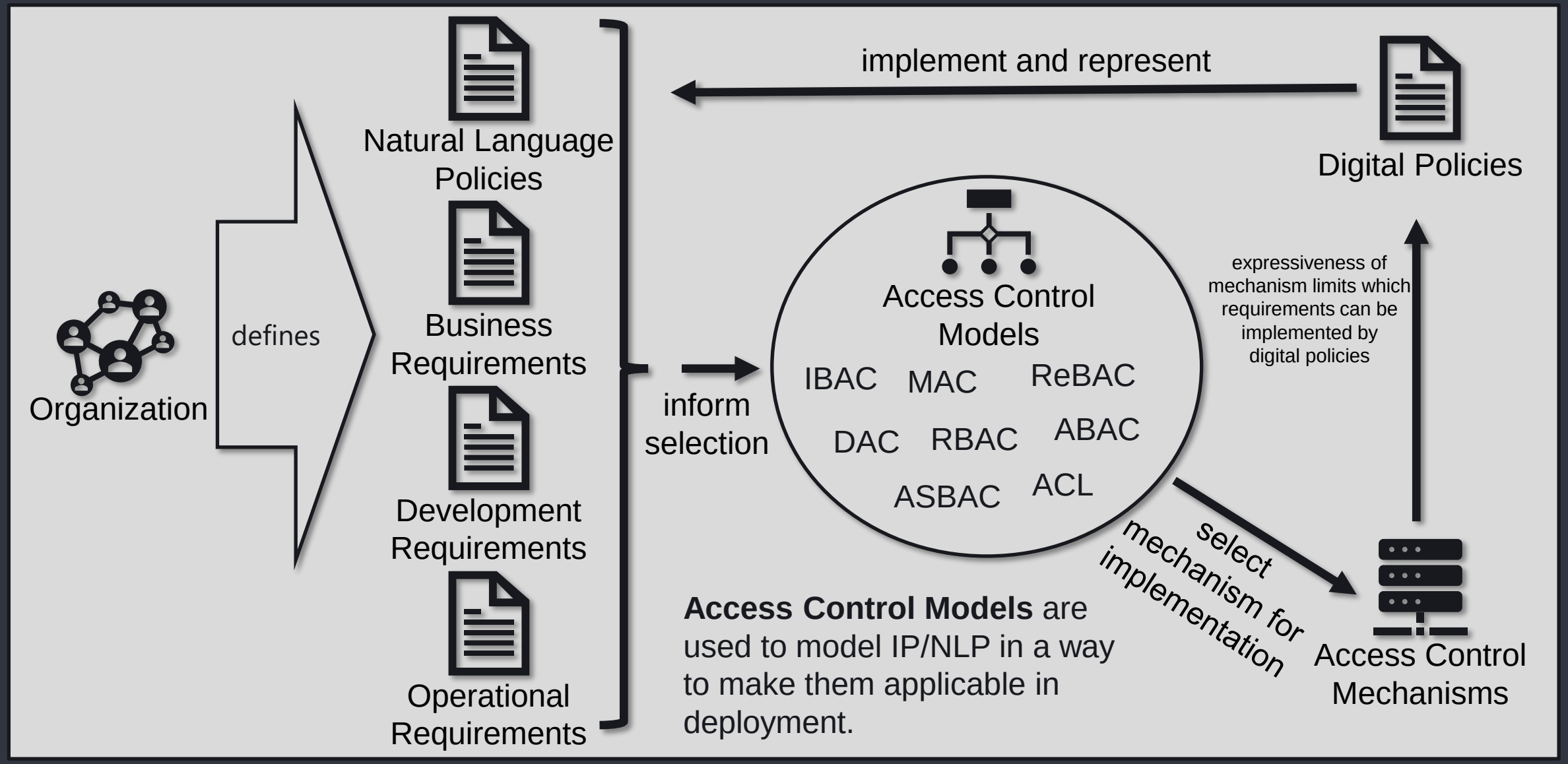
SAPL Access Control Lessons

Lesson #02 – Access Control Models

Dominic Heutelbeck

SAPL Webinar #02

Access Control Models



Discretionary Access Control (DAC) (1/2)

"An access control service that

(a) enforces a security policy based on the identity of system entities and the authorizations associated with the identities and

(b) incorporates a concept of ownership in which access rights for a system resource may be granted and revoked by the entity that owns the resource.

Derivation: This service is termed "discretionary" because an entity can be granted access rights to a resource such that the entity can by its own volition enable other entities to access the resource." [RFC4949]

- The "owner" of a resource decides how to share.
- Based on the identity of the subject
- Often extended by a group concept

Typical application: Operating Systems, Databases

- Also: Identity-based Access Control (IBAC) → Decision primarily based on the identity of the subject

Discretionary Access Control (DAC) (2/2)

Access Control Matrix

resources

$$\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{c} \begin{array}{ccc} R_1 & R_2 & R_3 \end{array} \\ \begin{bmatrix} orw & & x \\ & rx & \\ & & rwx \end{bmatrix} \end{array}$$

subjects

permissions

o - owner
r - read
w - write
x - execute

representations

Access Control Lists

$R_1 \rightarrow [\ (S_1, [o, r, w])]$
$R_2 \rightarrow [\ (S_2, [r, x])]$
$R_3 \rightarrow [\ (S_1, [x]), (S_3, [r, w, x])]$

Capability Lists

$S_1 \rightarrow [\ (R_1, [o, r, w]), (R_3, [x])]$
$S_2 \rightarrow [\ (R_2, [r, x])]$
$S_3 \rightarrow [(R_3, [r, w, x])]$

DAC Example (1/2) - Posix permissions

```
subject1@ubuntu01:~$ ls -la
total 32
drwxr-xr-x 4 subject1 subject1 4096 Nov 26 11:51 .
drwxr-xr-x 5 root     root     4096 Nov 26 11:51 ..
-rw------- 1 subject1 subject1   45 Nov 26 11:51 .bash_history
-rw-r--r-- 1 subject1 subject1  220 Nov 26 11:33 .bash_logout
-rw-r--r-- 1 subject1 subject1 3771 Nov 26 11:33 .bashrc
drwx------ 2 subject1 subject1 4096 Nov 26 11:34 .cache
drwx------ 3 subject1 subject1 4096 Nov 26 11:34 .gnupg
-rw-r--r-- 1 subject1 subject1  807 Nov 26 11:33 .profile
```

owner permissions | group permissions | “others” permissions | owner | group

Equivalent to Access Control Lists (pseudo notation)

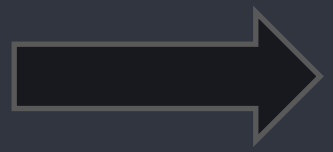
```
/home/subject1                → [(user:subject1,[o,r,w,x]), (group:subject1, [o,r,x]), (others, [r,x])]
/home                         → [(user:root,[o,r,w,x]), (group:root, [o,r,x]),
                                 (others, [r,x])]
/home/subject1/.bash_history  → [(user:subject1,[o,r,w]), (group:subject1, [o])]
…
/home/subject1/.gnupg         → [(user:subject1,[o,r,w,x]), (group:subject1, [o])]
/home/subject1/.profile       → [(user:subject1,[o,r,w]), (group:subject1, [o,r]), (others, [r])]
```

Warning: Ubuntu “`touch file`” in home directory → default permissions include `(others, [r])`

DAC Example (2/2) - Posix ACLs

```
subject1@ubuntu01:~$ ls -la
total 32
drwxr-xr-x 4 subject1 subject1 4096 Nov 26 11:51 .
drwxr-xr-x 5 root     root     4096 Nov 26 11:51 ..
-rw------- 1 subject1 subject1   45 Nov 26 11:51 .bash_history
-rw-r--r-- 1 subject1 subject1  220 Nov 26 11:33 .bash_logout
-rw-r--r-- 1 subject1 subject1 3771 Nov 26 11:33 .bashrc
drwx------ 2 subject1 subject1 4096 Nov 26 11:34 .cache
drwx------ 3 subject1 subject1 4096 Nov 26 11:34 .gnupg
-rw-r--r-- 1 subject1 subject1  807 Nov 26 11:33 .profile
```

```
subject1@ubuntu01:~$ getfacl .gnupg
# file: .gnupg
# owner: subject1
# group: subject1
user::rwx
group::---
other::---
```



```
subject1@ubuntu01:~$ setfacl -m g:accounting:rwx .gnupg
subject1@ubuntu01:~$ getfacl .gnupg
# file: .gnupg
# owner: subject1
# group: subject1
user::rwx
group::---
group:accounting:rwx
mask::rwx
other::---
```

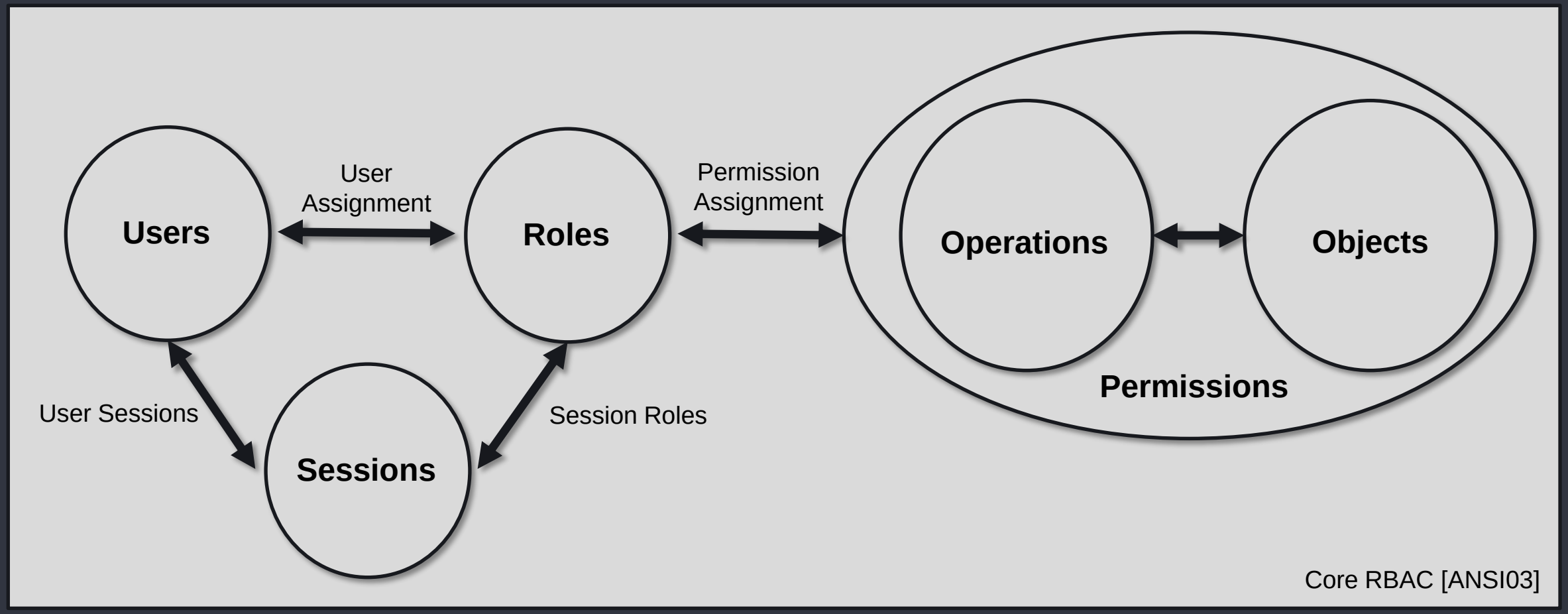
New ACL entry

```
subject1@ubuntu01:~$ ls -la
total 32
drwxr-xr-x  4 subject1 subject1 4096 Nov 26 11:51 .
drwxr-xr-x  5 root     root     4096 Nov 26 11:51 ..
-rw-------  1 subject1 subject1  309 Nov 26 13:11 .bash_history
-rw-r--r--  1 subject1 subject1  220 Nov 26 11:33 .bash_logout
-rw-r--r--  1 subj                  Nov 26 11:33 .bashrc
drwx------  2 su                    Nov 26 11:34 .cache
drwxrwx---+ 3 subject1 subject1 4096 Nov 26 11:34 .gnupg
-rw-r--r--  1 subject1 subject1  807 Nov 26 11:33 .profile
```

Indicator for ACL

Also see: [Gruenbacher03]

Role: "A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role." [ANSI03]



Hierarchichal RBAC:

Adds a hierarchy with permission inheritance to the role definitions

Static Separation of Duty:

Enforce constraints on the assignment of roles.

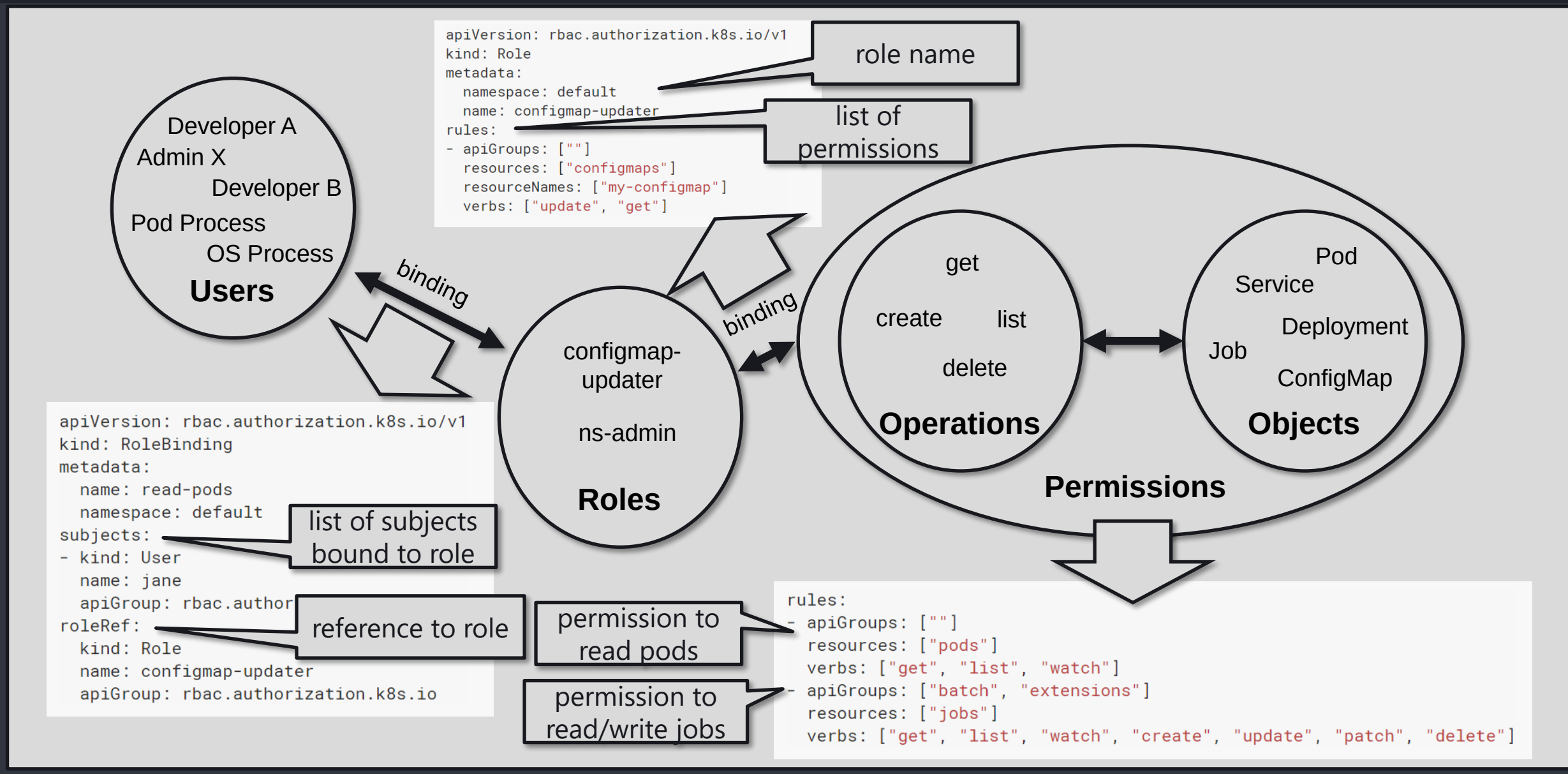
Assignment of one role may prohibit the assignment of another.

Dynamic Separation of Duty:

Add constraints on the simultaneously activated roles within the sessions.

Constraints on cardinality and role assignment conditions.

RBAC Example - Kubernetes



RBAC Problem: Role Explosion

Role Explosion refers to practical administrative challenges in the application of RBAC.

→ increasingly difficult to manage number of roles in the organization

“**Symptom 1:** An enterprise organization requires employees to access several IM systems and most (or all) of the systems autonomously manage their own set of role (or group) information.

[…]

Symptom 2: An enterprise organization has one or more IM systems where the total number of users approaches or surpasses the total number of roles.” [ELLIOTT&K10]

Access Control Model: Mandatory Access Control (MAC)

Origin: 1960s and 1970s, US Department of Defense

"The information system enforces [...] policy [...] over defined subjects and objects where the policy specifies that a subject that has been granted access to information can do one or more of the following:

(a) Pass the information to any other subjects or objects;
(b) Grant its privileges to other subjects;
(c) Change security attributes on subjects, objects, the information system, or the information system's components;
(d) Choose the security attributes to be associated with newly created or revised objects; or
(e) Change the rules governing access control." [NIST80053]

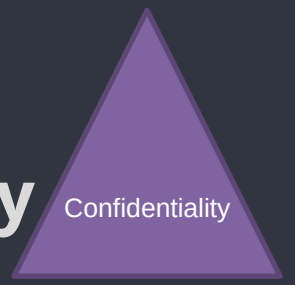
- Users have no discretion to change access rights by default.
- Policies are predefined.

MAC - Bell LaPadula Model (1/2)

- Based on security (clearance) levels:
  E.g. NATO [BMIA007]:
  - COSMIC TOP SECRET (CTS) (Germany: STRENG GEHEIM)
  - NATO SECRET (NS) (Germany: GEHEIM)
  - NATO CONFIDENTIAL (NC) (Germany: VS-VERTRAULICH)
  - NATO RESTRICTED (NR) (Germany: VS-NUR FÜR DEN DIENSTGEBRAUCH)

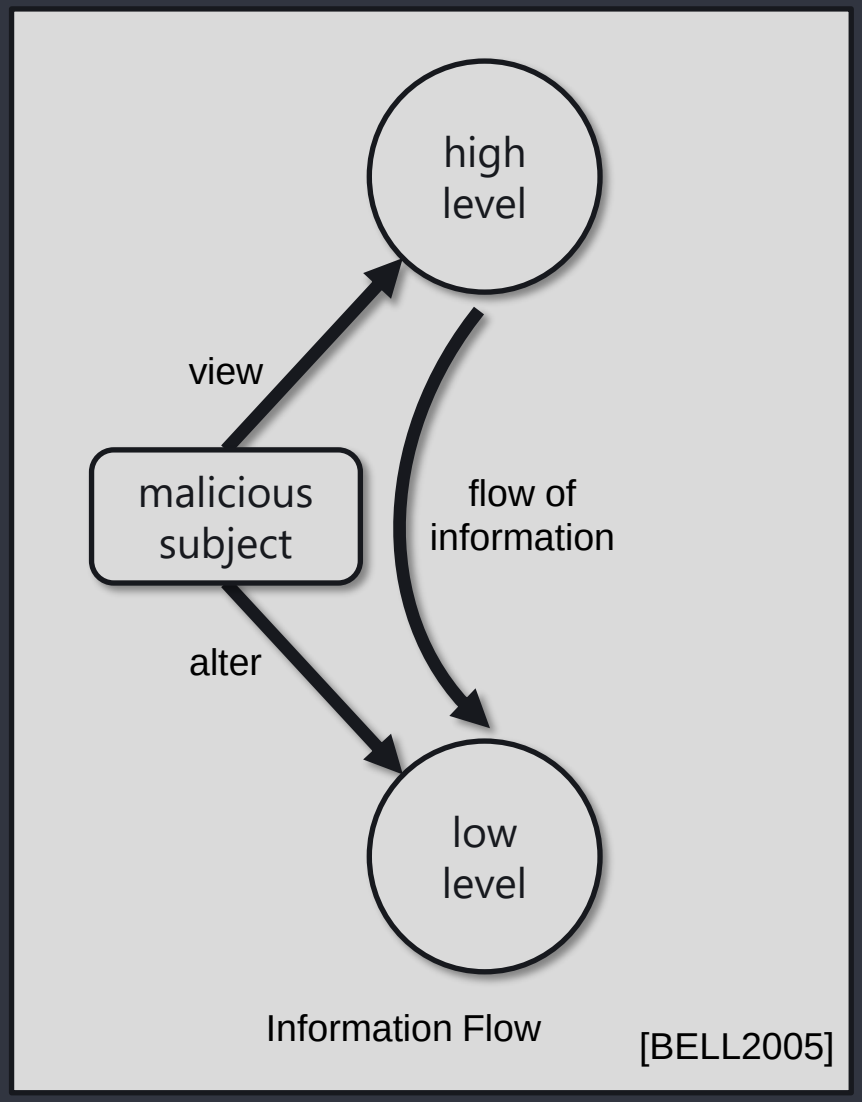
more secret

Primary goal of Bell LaPadula is confidentiality

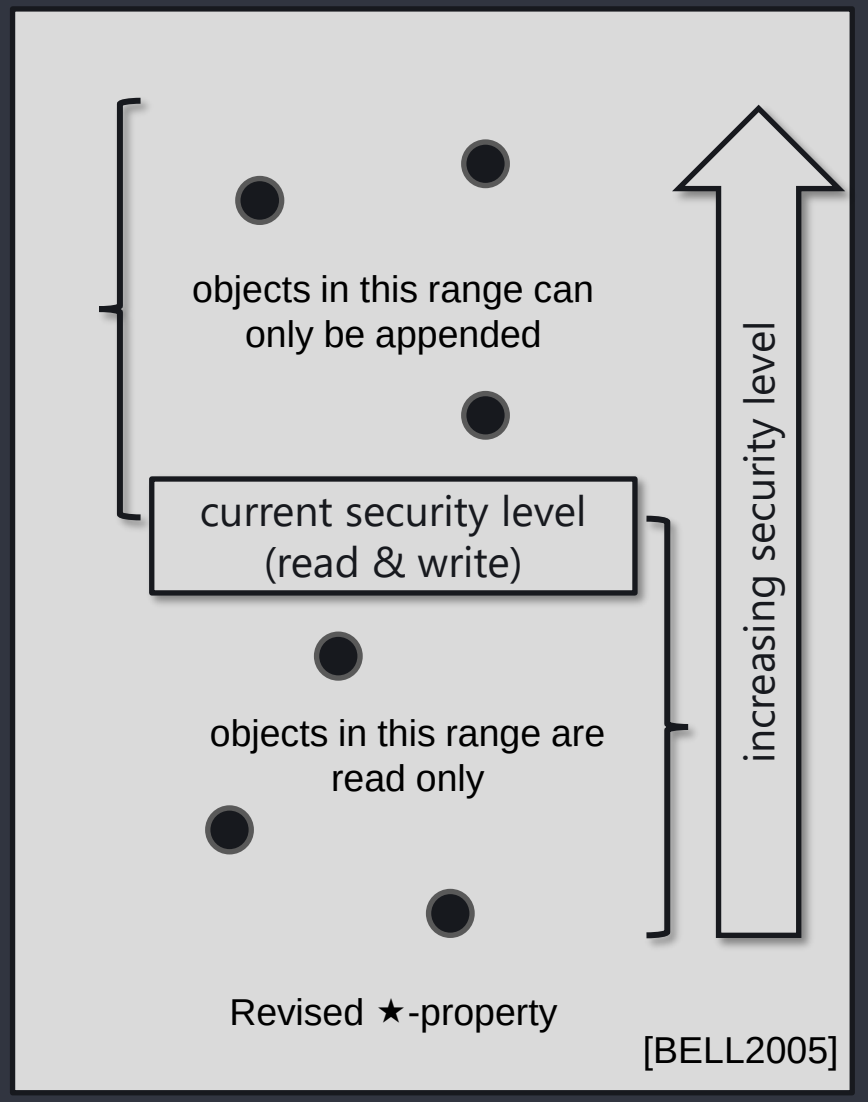


- Subjects and Resources have assigned security levels.
  (security labels/security attributes/clearance)
- Security levels can only be altered by designated administrators

MAC - Bell LaPadula Model (2/2)



Information Flow [BELL2005]



Revised ★-property [BELL2005]

Principle:
- No write down
- No read up

"*-Property"

Formalization:
Bell-LaPadula Model [BELL05]

Problem:
"write up" → risk to integrity

Clearance and Vetting

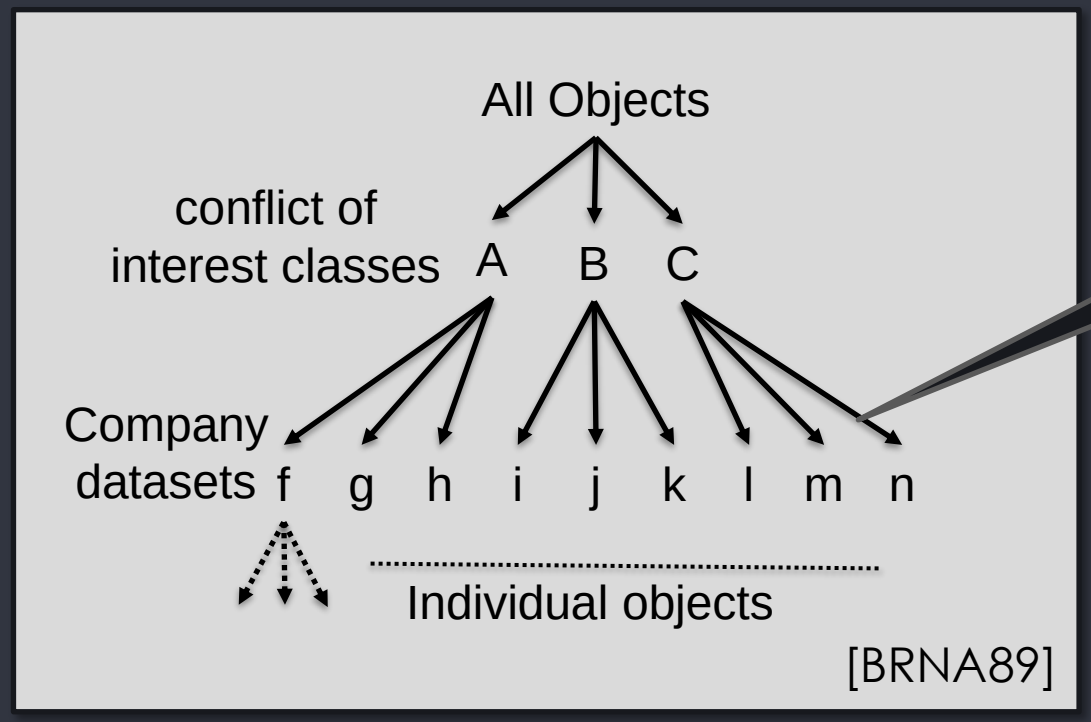
- Vetting → requirement → clearance levels in national security
- Vetting → access to information & available occupations within organization.
- Vetting → beyond national security
  - → work with vulnerable (e.g. children)
  - → policy work (criminal records)

Germany “Sicherheitsüberprüfungsgesetz”

- “Einfache Sicherheitsüberprüfung” [SÜG § 8]
  - access to VS-VERTRAULICH (CONFIDENTIAL) information
- “Erweiterte Sicherheitsüberprüfung” [SÜG § 9]
  - access to a high number of VS-VERTRAULICH (CONFIDENTIAL) information
  - access to GEHEIM (SECRET) information
- “Erweiterte Sicherheitsüberprüfung mit Sicherheitsermittlungen” [SÜG § 10]
  - access to a high number of GEHEIM (SECRET) information
  - access to STRENG GEHEIM (TOP SECRET) information

Domain: Financial sector

Goal: Confidenciality, i.e., limit insider knowledge in stocks trading etc.

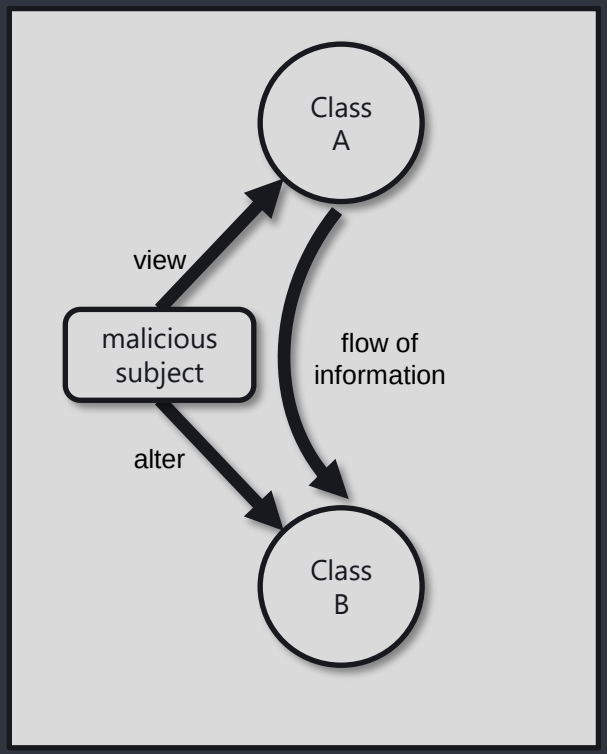


If two datasets are within the same conflict of interst class, a subject may only have access to exactly one of them

Model tracks access to datasets to limit subsequent access to other classes.

"*-Property"
Write permitted only, if

- Read Access granted
- No object can be read which is in a different company dataset than the one where write access is requested

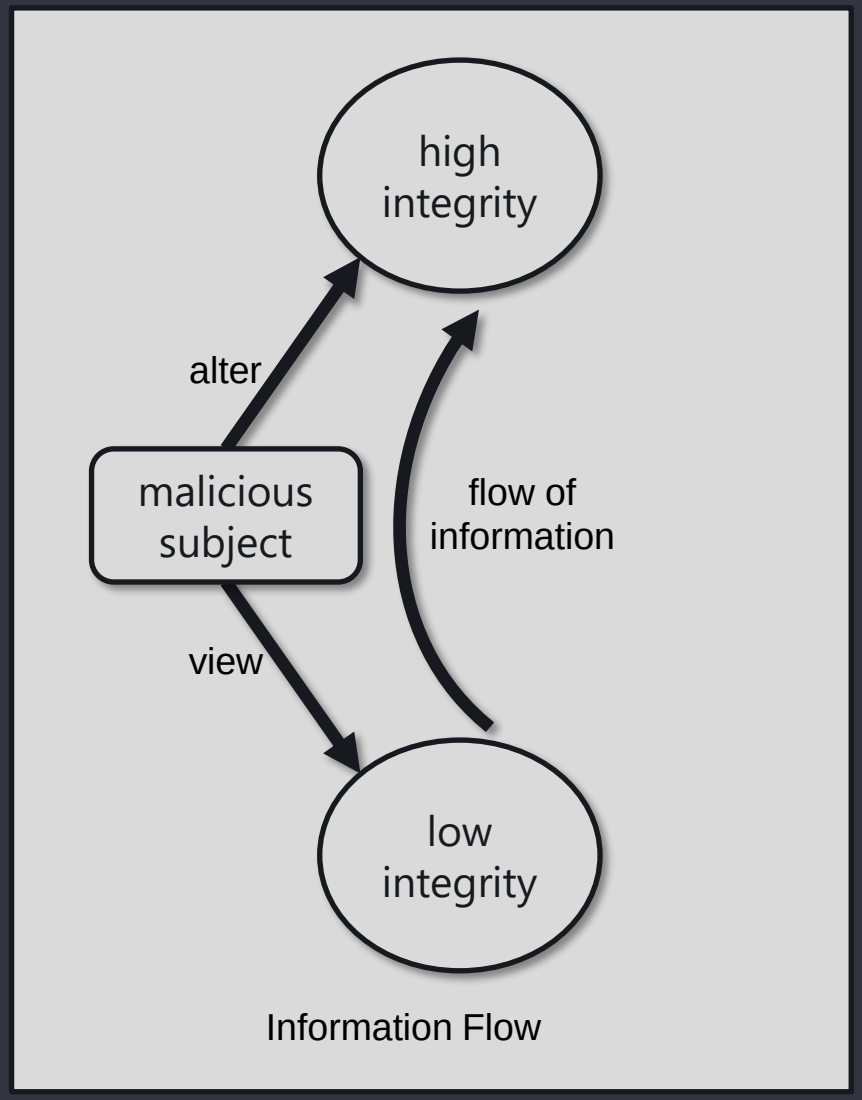


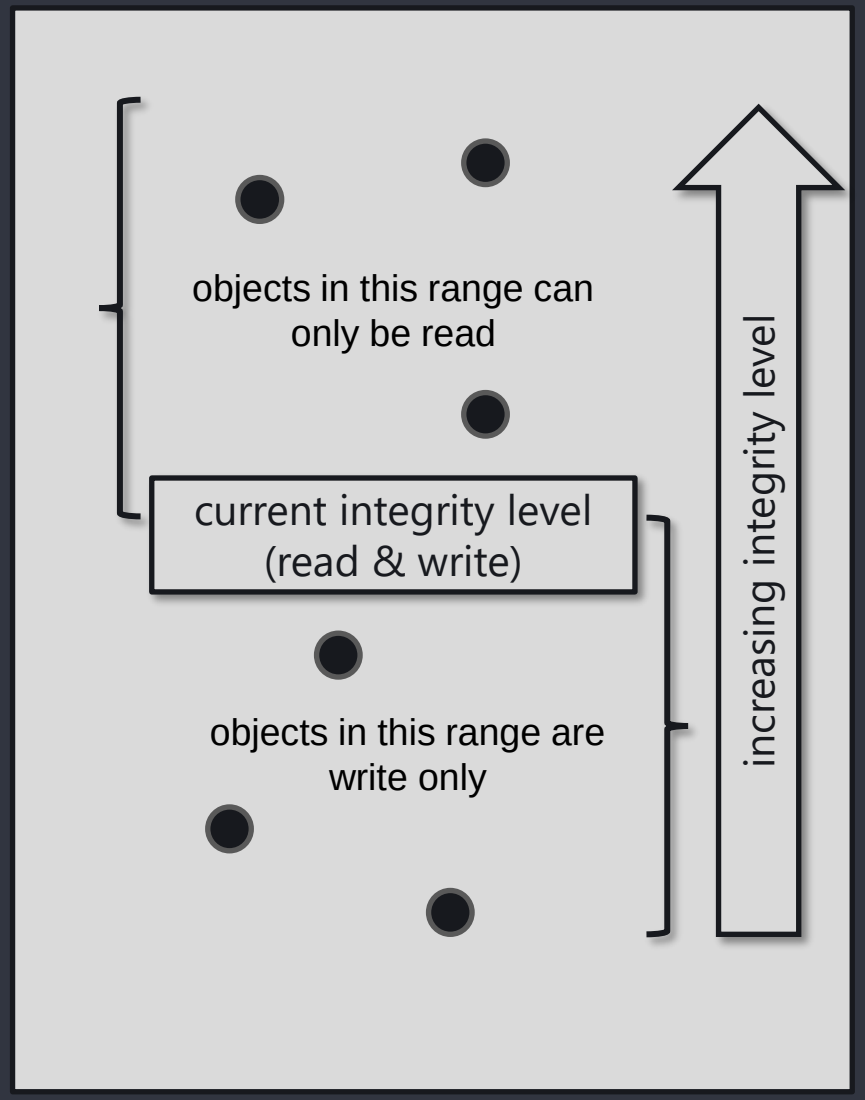
Goals

- Unauthorized users must not be able to make changes within a system
- Authorized users must not be able to make unauthorized changes
  i.e., a change that changes the integrity
- Internal and external consistency is maintained
  - internal: internal transactions actually do what they are supposed to do (2+2=4)
  - external: if a customer bought 1000 shares of stock this is actually done and not only 900 shares are bought and a man in the middle got the difference

MAC - Biba Model (2/2)

Also Low-Watermark Mandatory Access Control (LoMAC)





Inversion of Bell LaPadula

Principle:

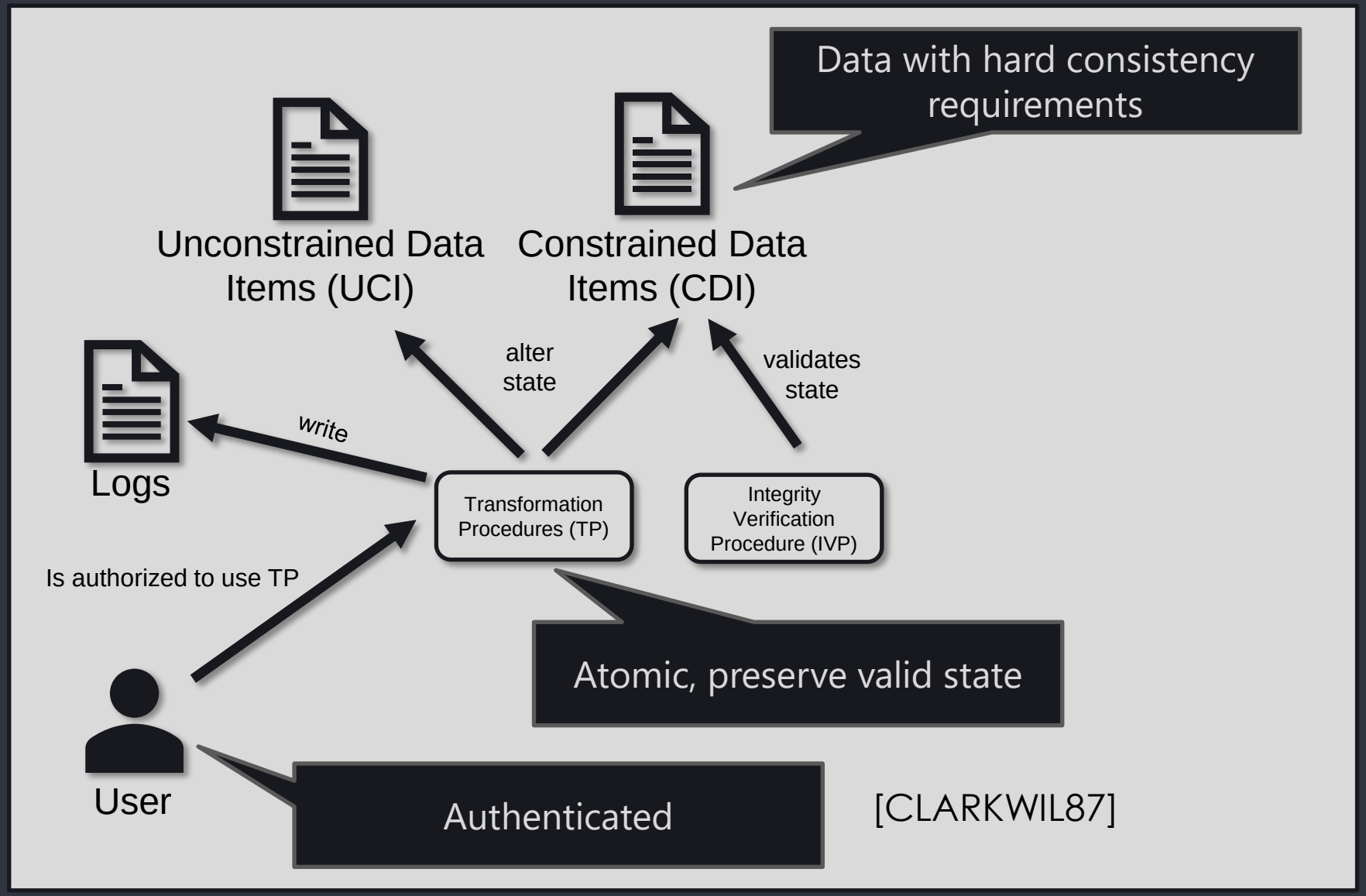
- No read down
- No write up

Origin: [BIBA75]

Primary goal of Biba is integrity

Application:
E.g., Honeypots

MAC – Clark Wilson Model



[CLARKWIL87]

Not a pure Access Control Model

Concepts found in many modern software architectures

E.g.:

- Domain Layer is multi-tiered architectures with Database transactions,
- Aggregates in Domain Driven Designs as consistency boundaries contain TP

Meta Access Control

Controlling who can modify access rights

- Graham–Denning Model [GrDe72]
- Harrison–Ruzzo–Ullman Model [HaWaUl76]
- NGAC [INCITS 565-2020]

SAPL Access Control Lessons

Q&A

SAPL Access Control Lessons

Lesson #03 – ABAC Access Control Mechanisms

Dominic Heutelbeck

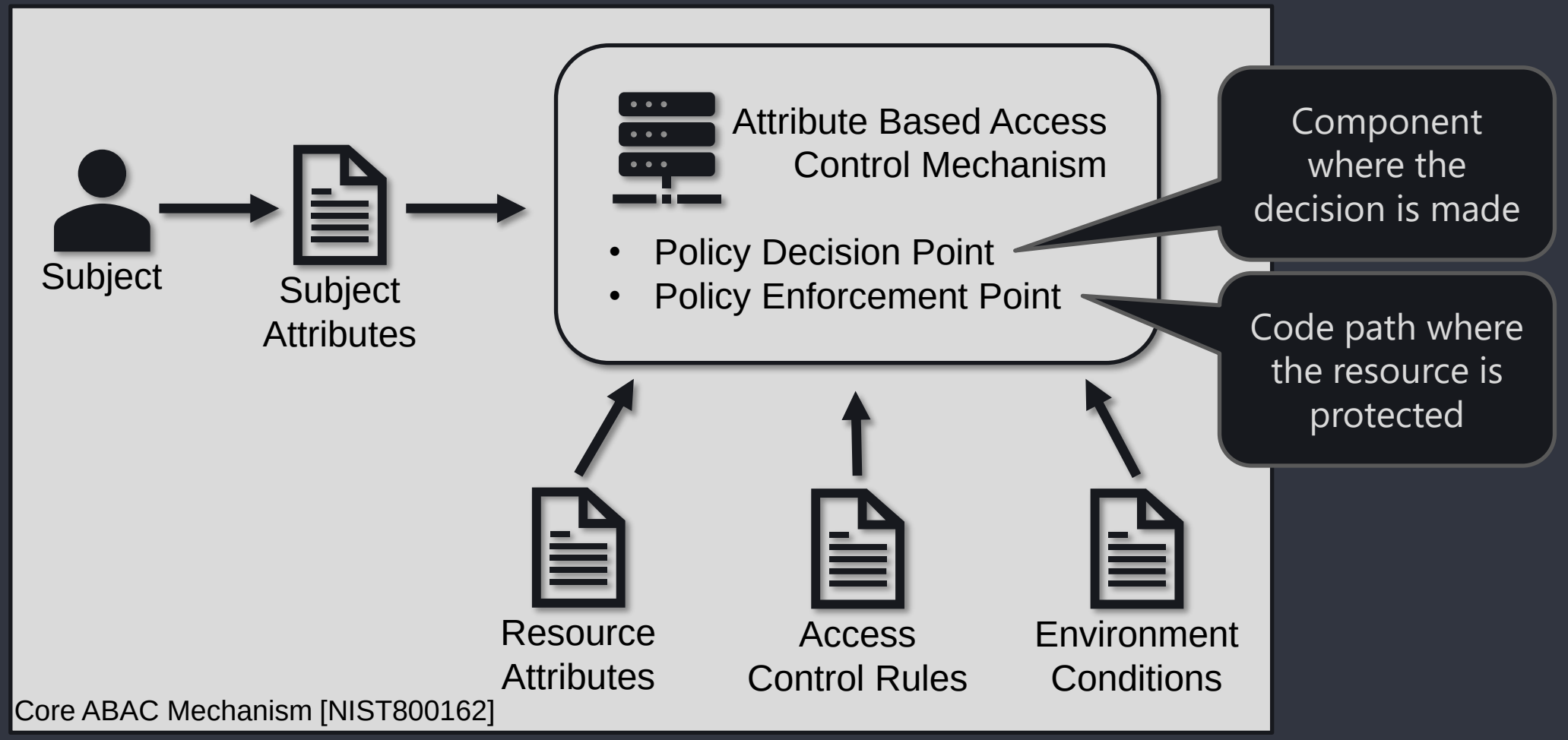
SAPL Webinar #03

Limitations in Expressiveness (DAC, MAC, RBAC)

- ACLs, RBAC, MAC provide limited expressiveness wrt. AC requirements (NLPs).
- Certain concepts cannot be easily expressed in ACLs, RBAC, MAC:
  - Location. E.g.,:
    - "access only on premise of library"
    - "access only within 50m of machine"
    - "access only on airport premise"
    - "access only after subject passed through physical access control system and did not yet exit"
  - Environmental information:
    - "only during rainy weather"
    - "if machine temperature exceeds .."
    - "if defence condition > DEFCON 3"
  - Time constraints. E.g.:
    - "may access only during work hours according to schedule"
    - "[...] no matter how much he cries, or how much he begs, never, *never* feed him after midnight" [Gremlins, 1984].

Attribute Based Access Control (ABAC)

- ABAC uses characteristics (or **attributes**) of subjects, resources, or the environment and Access Control Rules (DPs) to come to authorization decisions.



Attributes

- In most concrete ABAC implementations:
  - attributes are key-value pairs associated with subject, resource, or environment

- DAC, RBAC, MAC can be expressed using ABAC (subset relation).
  Attributes used:
  - DAC → identity and permissions
  - RBAC → roles as attributes
  - MAC → security level

Corollary: DAC, RBAC, MAC can be implemented using an ABAC implementation

- ABAC is in practice primarily used in information system development.

- Typically not part of OS like DAC, MAC

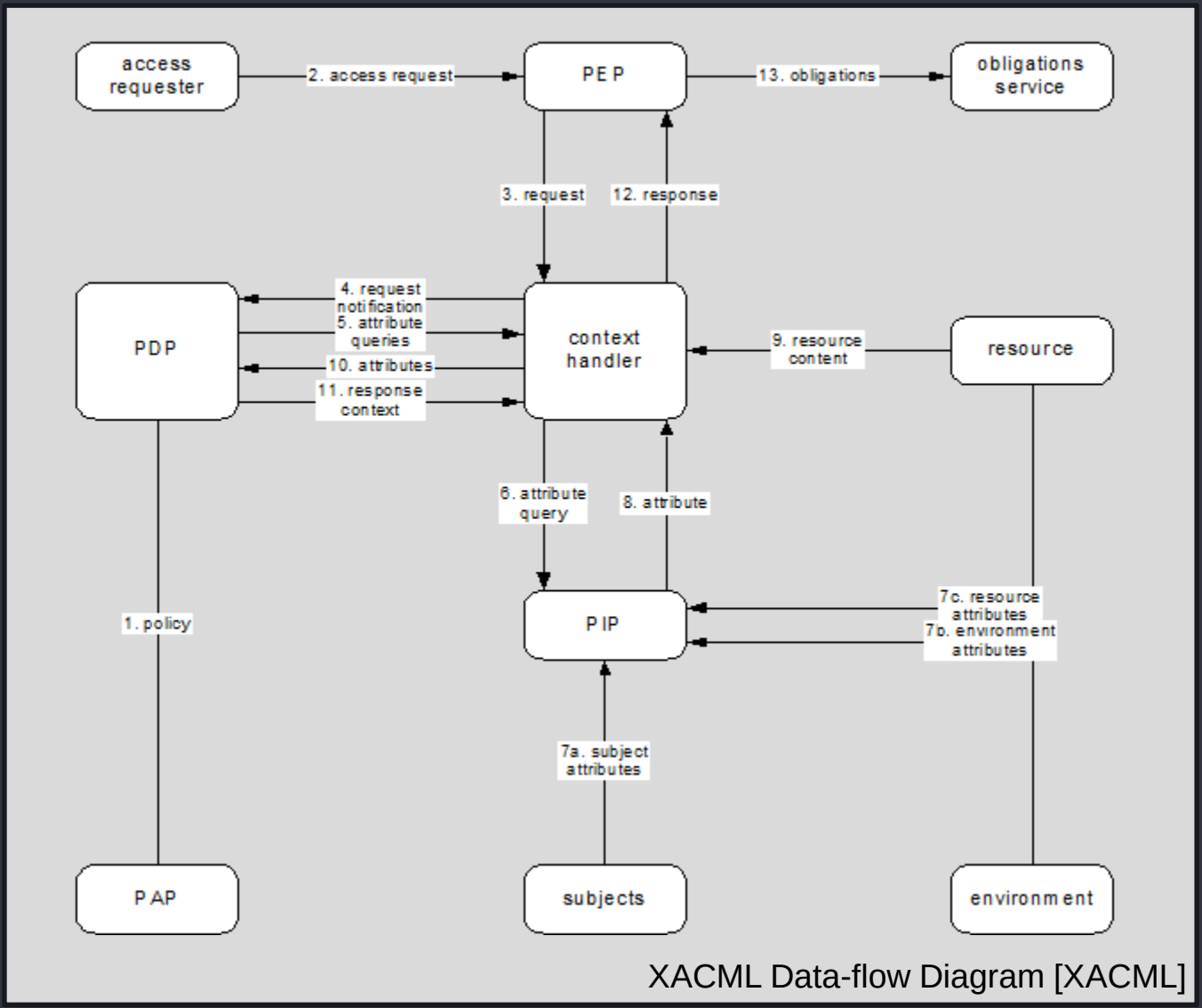
eXtensible Access Control Markup Language (XACML)

- XACML [XACML] is an OASIS standard for ABAC implementation.
  - Concrete functional architecture
  - Policy Language
  - Protocol
  - Various extensions
  - XML-based language and data model

- Open Source and Commercial implementations available:
  AuthzForce (OW2), Axiomatics Policy Server, Balana, ndg-xacml, NextLabs, OpenAZ, Oracle Entitlements Server, Security Policy Tool, SunXACML, ViewDS Access Sentinel, XEngine

- Implementations dominated by Java

XACML – Architecture



XACML Data-flow Diagram [XACML]

PIP: Policy Information Point
retrieves attributes from repositories

PAP: Policy Administration Point
component for managing policies to be considered by the PDP

obligation service:
XACML can require or recommend the PEP to perform certain actions.
This service performs these.

XACML – Protocol

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
    ReturnPolicyIdList="false">

    <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
        <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
            <AttributeValue DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
                bs@simpsons.com
            </AttributeValue>
        </Attribute>
    </Attributes>

    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
        <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
                file://example/med/record/patient/BartSimpson
            </AttributeValue>
        </Attribute>
    </Attributes>

    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read
            </AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

- requests contain key-value pairs assigned to subject, action and resource (categories)
- values are typed
- multi-requests supported
- JSON request variant available

subject:subject-id	bs@simpsons.com
resource:resource-id	file://example/med/record/patient/BartSimpson
action:action-id	read

XACML 3.0 Specification: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
XML Schema: http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd

XACML – Protocol

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os">
  <Result>
    <Decision>Permit</Decision>
    <Obligations>
      <Obligation FulfillOn="Permit" <ObligationId="sendEmail">
        <AttributeAssignment AttributeId="email" DataType="http://www.w3.org/2001/XMLSchema#string">
          some@example.org
        </AttributeAssignment>
      </Obligation>
    </Obligations>
  </Result>
</Response>
```

Permit:	The access request is permitted.
Deny:	Access is denied
NotApplicable:	There was no policy applicable (i.e., matching) the request
Indeterminate:	The PDP is unable to evaluate the requested access. E.g., missing attribute, network error, division by zero, unresolved conflicts

XACML Policies

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
            http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
        PolicyId="urn:oasis:names:tc:xacml:3.0:example:SimplePolicy1"
        Version="1.0"
        RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
  <Description>
     Medi Corp access control policy
  </Description>
  <Target/>
    <Rule RuleId= "urn:oasis:names:tc:xacml:3.0:example:SimpleRule1"
          Effect="Permit">
    <Description>
      Any subject with an e-mail name in the med.example.com domain
      can perform any action on any resource.
    </Description>
    <Target>
      <AnyOf>
        <AllOf>
          <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
              med.example.com
            </AttributeValue>
            <AttributeDesignator MustBePresent="false"
                                 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
                                 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                                 DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>
</Policy>
```

How to resolve multiple applicable rules

Limit the applicability of the policy

Define a rule, which if fulfilled yields permission

Limit the applicability of the rule

rfc822Name-match(med.example.com, subject-id) == true ?

<Target> := AND of all contained <AnyOf>
<AnyOf> := OR of all contained <AllOf>
<AllOf> := AND of all contained <Match>
<AnyOf> → think Disjunctive Normal Form over predicates
<Target> → Overall not DNF/CNF

XACML Combining Algorithms

- Deny-overrides
- Ordered-deny-overrides
- Permit-overrides
- Ordered-permit-overrides
- Deny-unless-permit
- Permit-unless-deny
- First-applicable
- Only-one-applicable

Combining algorithms resolve if multiple policies/rules yield a decision, or if no decision can be made based on policies

„Grant access, if subject and resource are in a common group of type A and the subject has role B within the group“.

a) You cannot provide a parameter in an XACML policy for an attribute.
   There is no way to express an attribute:
   “`urn:groupsOfTypeAndRoleInhabited`”(AttributeId) with two parameters for the group type and the role type.
b) Solution: Have the PIP calculate all permutaions of group types and roles and encode the parameters in the AttributeId:
   “`urn:groupsOfTypeFinanceAndRoleAccountant`”
   then do bag intersection calculations in the policy.

→ Unnatural, wasteful, actually potentially subject to change, and potential need to restart the PDP (WSO2IS in the case study)

- To use domain information as attributes in XACML, all required Information must be reachable via exactly one attribute access step.

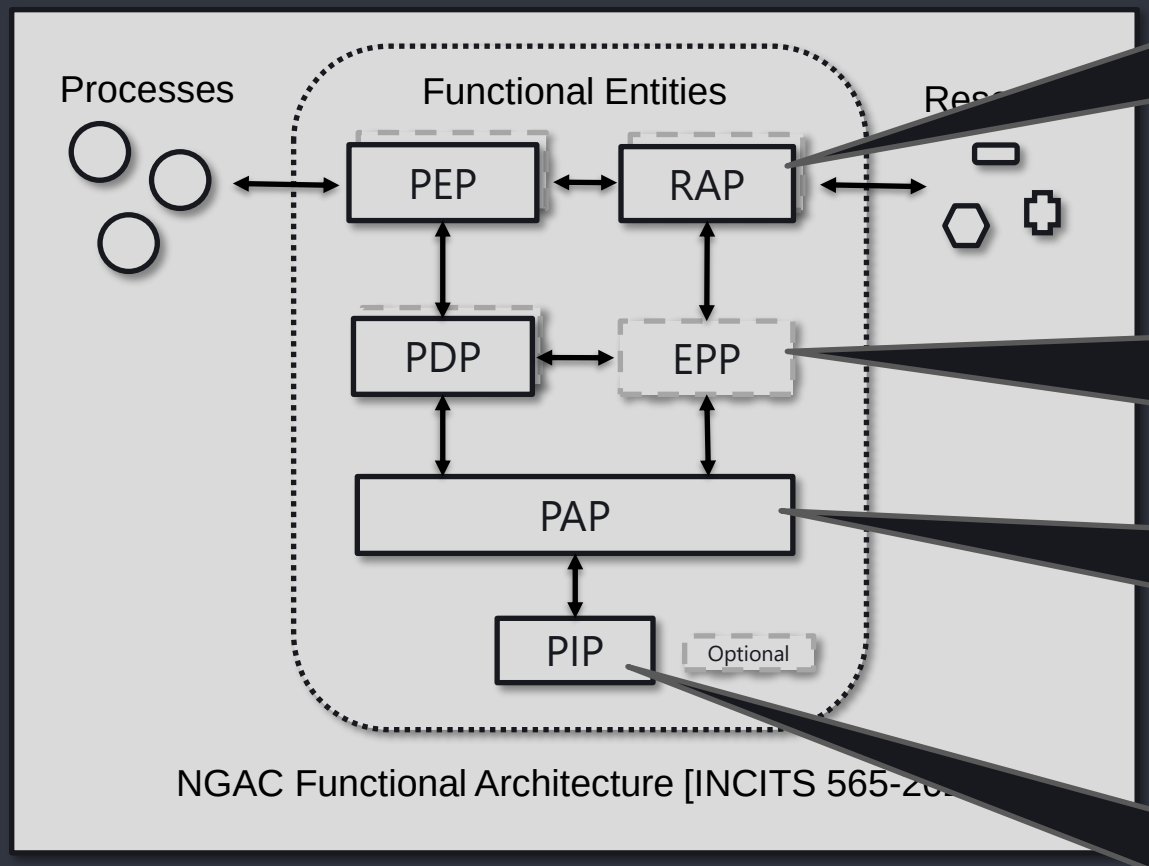
Example for Problem:

1. Given: Subject has Id in Request
2. Domain logic indexes attribute matching to subject by other Id (e.g., in a DDD modelled scenario with disjunct sub-domains used as PIPs)
3. Map Id to alternate Id (external access attribute of subject)
4. Get attribute using new Id (result of previous PIP access)

- Results in additional work for application developers (separation of concerns) and synchronization issues.

Next Generation Access control - NGAC

Functional Model



NGAC Functional Architecture [INCITS 565-20

Resource Access Point
The only means of access to a resource

Event Processing Point
handles events potentially triggering obligations

Policy Administration Point
PDP accesses policy information via the PAP

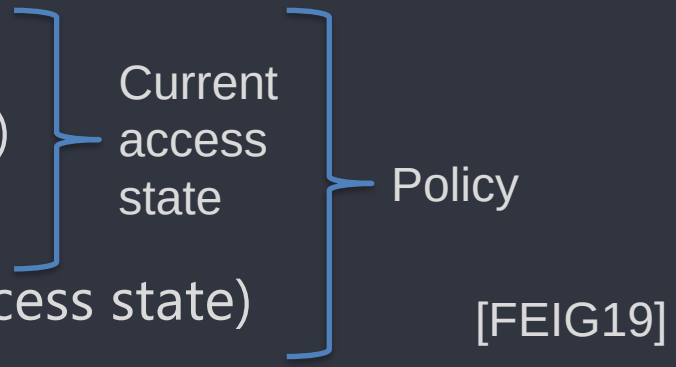
Policy Information Point
Contains the actual policies.
ACID transactions.
NOT: additional attributes as in XACML/SAPL

Flows

- resource access flow
- administration access flow
- optional event context flow

On Access, PEP and PDP can issue an event context to the EPP.
Obligations are retrieved via PAP from the PIP and mathching handling is triggered.

- Basic elements
  - Users, access rights (resource and admin), and resource objects
- Containers
  - User attributes, object attributes, and policy classes
- Relations
- Assignments (define membership in containers)
  - Associations (with assignments, used for deriving privileges)
  - Prohibitions (denies for users access capabilities)
  - Event-pattern/admin-response (for dynamically alter the access state)

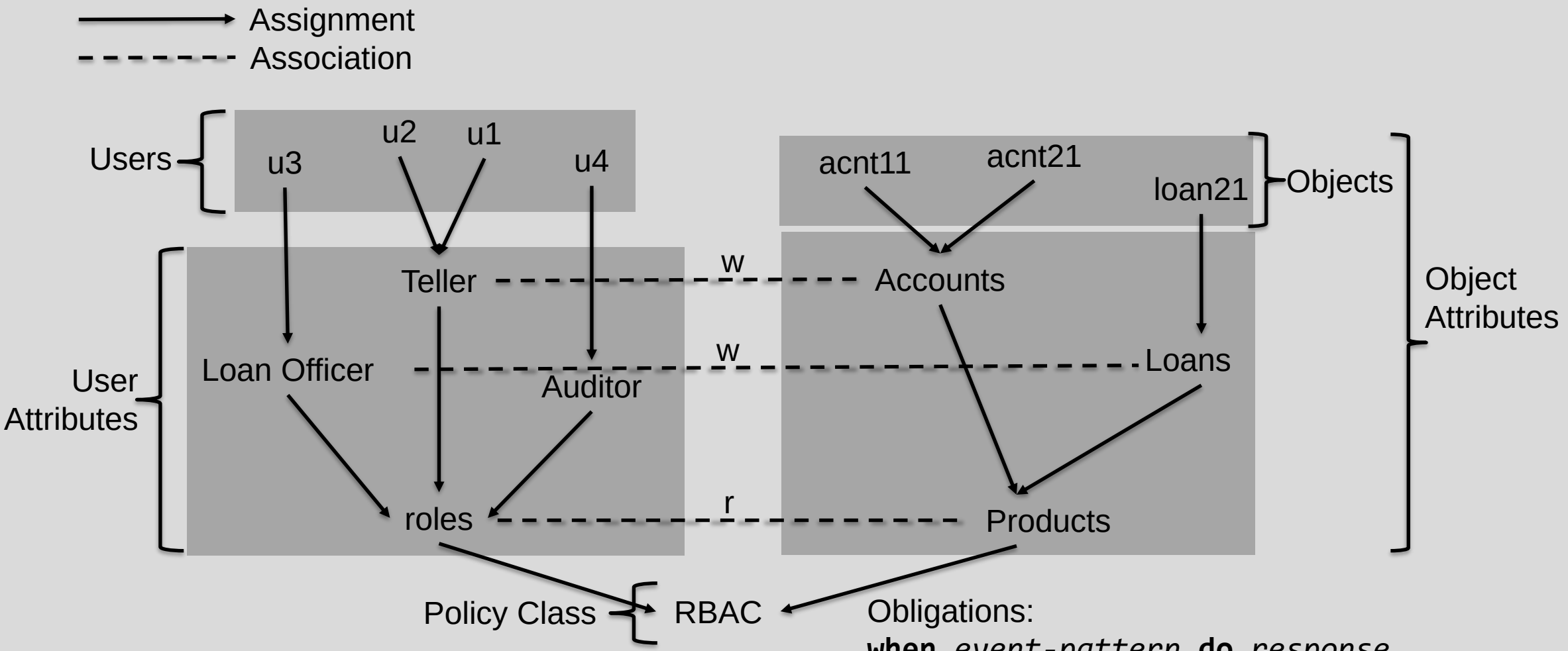


[FEIG19]

Note:
NGAC does not have the notion of individual policies.
It requires complete knowledge of subjects and resources.
The entirety of assignments within the network of entities defines ist global policy. However, sometimes not used consistently in NGAC context

NGAC Security Model

Assignments, Associations, Prohibitions



Obligations:

when *event-pattern* **do** *response*

```
when: 5:00PM, do: create ua-deny(Teller, {r,w}, Accounts)
when: 9:00AM, do: delete ua-deny(Teller, {r,w}, Accounts)
```

Prohibition: `u-deny(u1, w, acnt21)`

Advantages:

- No IO for attributes
- Complete Knowledge
- Auditability
- Linear cost

Disadvantages

- Complete Knowledge required (synchronization problems with application data)
- Optimized for machine readability and processing. Not well suited for communication with stakeholders (e.g., Domain Driven Design context)
- Event-driven self modification not really considered under the view of auditability.
- Event-driven self modification to model time-based policies not very elegant (subjective)
- Not easily extensible
- Location-based policies limited to semantic labels in the attribute graph, no geometry operations. Fine-grained geofencing not possible.

SAPL Access Control Lessons

Q&A

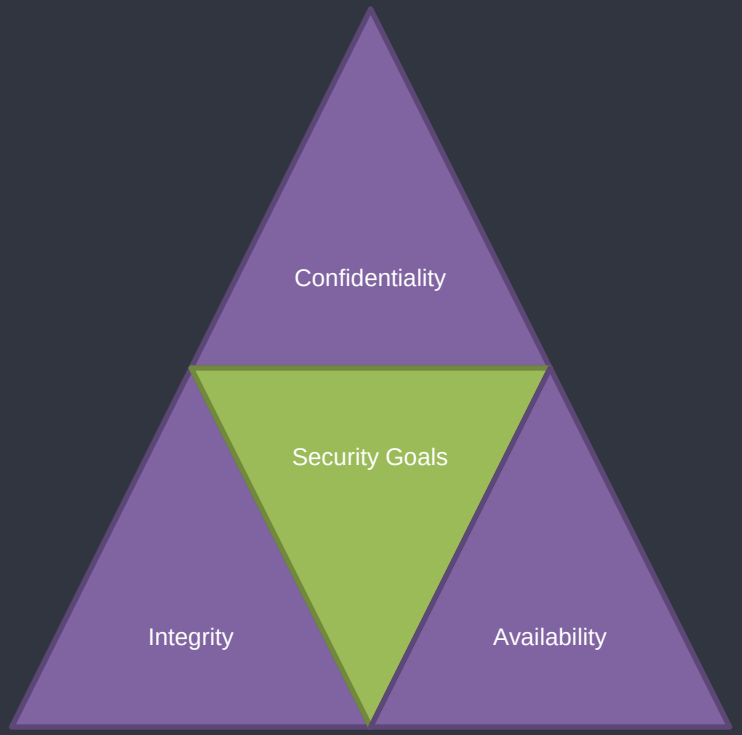
SAPL Access Control Lessons

Lesson #04 – ASBAC and SAPL Fundamentals

Dominic Heutelbeck

SAPL Webinar #04

Attribute Stream Based Access Control - ASBAC



When applying standard request-response-based Access Control Models threats to the security goals confidentiality and availability can be identified....

Threat

Current systems exclusively check access right **before** granting access.

Once access is established and not interrupted, and not explicitly checked regularly the client may **potentially access resources indefinitely.** Threatening **confidentiality** of information in the context of information security.

Threat significant in stateful, session-based systems, connection-oriented protocols, (collaborative) web applications, data stream management (IoT sensor data, MQTT, etc.), web sockets, server sent events…

Threat insignificant in stateless systems, database queries, other request-response based systems, very short-lived sessions…

Threat

Once access is denied, users are not actively informed when previously denied access would be permitted. Users or systems must **actively re-request access rights** and potentially **use side channels** to find out if they should. Threatening **availability** of information in the context of information security. An reducing the overall user experience.

Threat significant in applications with spatio-temporal access policies, policies based on the state of the application domain, collaborative applications

Takeaway:

Applying (potentially blocking) request-response patterns for authorization (security in general) in increasingly event-driven systems:

Probably a bad idea!

Current solution

- Add **explicit handling** in domain code or
- by **polling** the policy decision point in

Problems

- Direct relation between accepted **latency** and **polling frequency**
- Polling frequency directly related to **system load** on authorization infrastructure

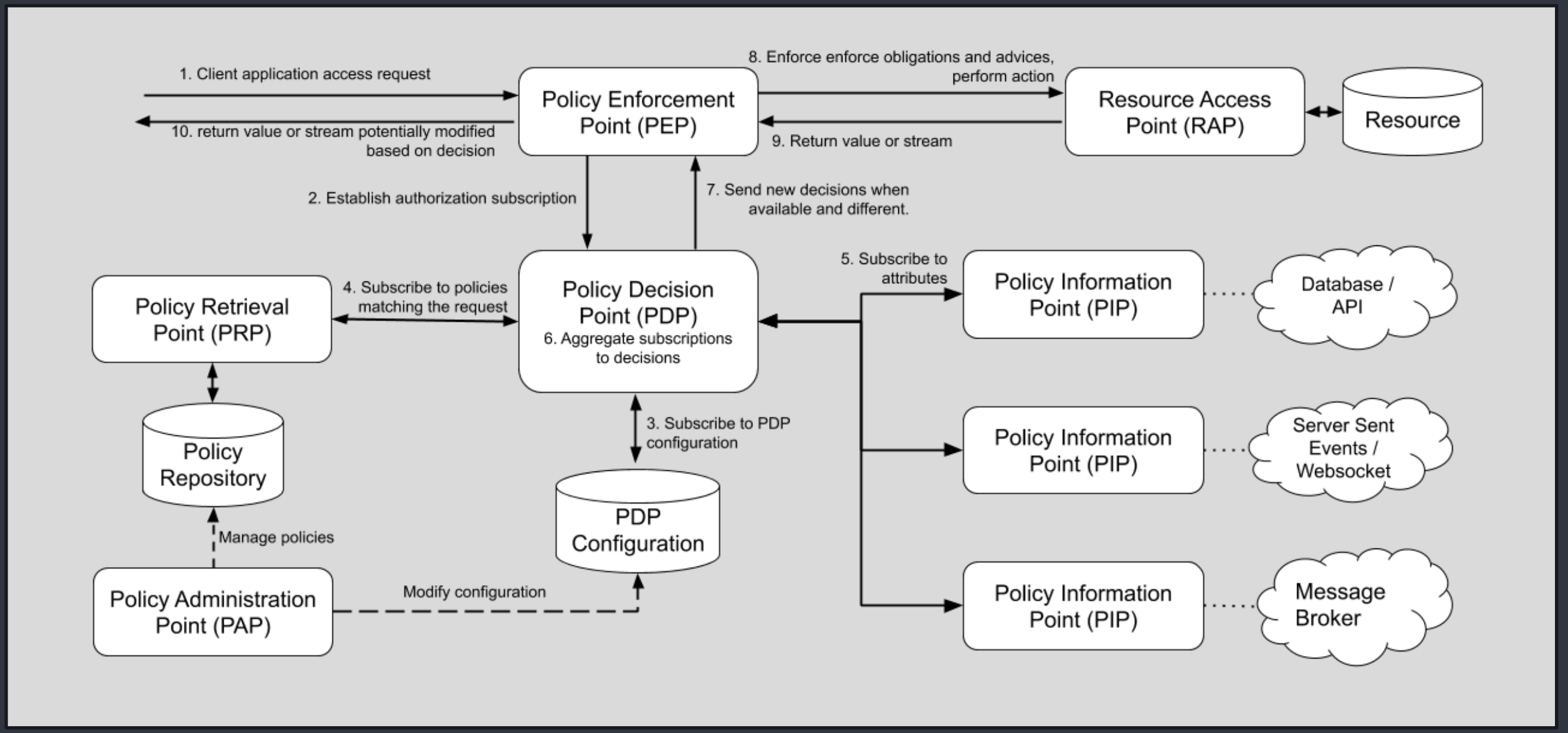
The solution must be acceptable with regards to the remaining risks and latency and introduced load.

- **Key Question**
  How to support session-oriented applications with dynamic low-latency attribute-based authorization decisions without polling?

- **Proposed solution**
  Switch the communication model of authorization infrastructures **from request-response to publish-subscribe**.

  # “Attribute Stream-Based Access Control (ASBAC)”

Functional Architecture

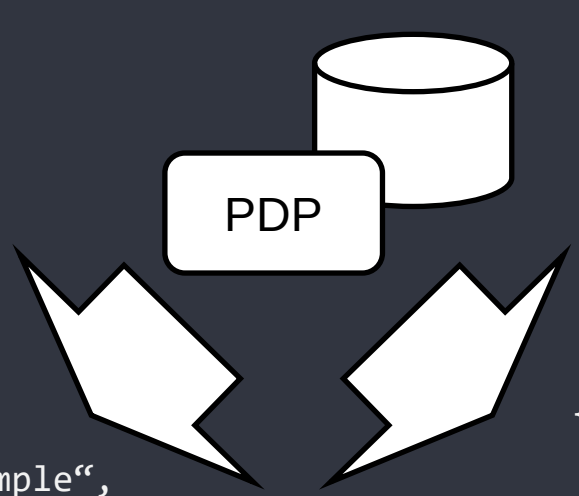


Additional Use-Cases/Requirements

- Support JSON natively
- Support filtering and transformation. E.g.:

```
{
   patientId: „035nd3dc“,
   name: „Alice Example“,
   birthDate: „27.01.1995“,
   diagnosis: „Leukemia“
}
```

```
{
   age: „20-30“,
   diagnosis: „Leukemia“
}
```




```
{
   id: „abc“,
   cardHolder: „Alice Example“,
   cardNumber: „1234 1234 1234 1234“
}
```

Payment Microservice

```
{
   id: „abc“,
   cardHolder: „Alice Example“,
   cardNumber: „1234 1234 1234 1234“
}
```

Web UI

What are true new requirements?

- Support of **publish-subscribe** model, throughout the infrastructure.
- Handling **transformations and filtering**
- Attribute access **parametrization and cascading**

Not new but desired:

- Usability improvements (Authoring and PEP development)
- Native JSON support
- Extensible
- Provide an open testbed for testing more AC paradigms (e.g., ReBAC)

Baseline:

- Support similar use cases like XAML, i.e., breaking the glass, obligations and advices, policy sets, multi request, geofencing

The Streaming Attribute Policy Language

Subscriptions:

```
{
   subject: JSON Value,
   action:  JSON Value,
   resource: JSON Value,
   environment: JSON Value
     (optional)
}
```

Multi-Subscriptions:

```
{
   "subjects"  : ["bs@simpsons.com", "ms@simpsons.com"],
   "actions"   : ["read"],
   "resources" : ["file://example/med/record/patient/BartSimpson",
                  "file://example/med/record/patient/MaggieSimpson"],
   "environments" : [],
   "requests" : {
     "req-1" : { "subjectId": 0, "actionId": 0, "resourceId": 0 },
     "req-2" : { "subjectId": 1, "actionId": 0, "resourceId": 1 }
   }
}
```

Decision:

```
{
   decision: „PERMIT“,
   obligations: [],(optional)
   advices: [], (optional)
   resource: JSON Value
     (optional)
}
```

Multi-Decision:

```
{
  "responses" : {
    "req-1" : {
      "decision" : "PERMIT",
      "resource" : { ... }
    },
    "req-2" : {
      "decision" : "DENY"
    }
  }
}
```

Comment:
Vocabulary is being updated ro „decision“ from „response“

A SAPL policy document generally consists of:

- Optional `import` statement for PIPs, Functions
- the keyword `policy`, declaring that the document contains a policy
- a unique policy name (uniqueness at publishing time in the PDP/PRP)
- the entitlement (decision returned on successful evaluation): `permit` or `deny`
- optional target expression for indexing and policy selection
- optional `where` clause containing the rules
- optional `advice` and `obligation` to be sent to the PEP upon successful evaluation
- optional `transformation` clause for defining a transformed/filtered resource

```
// Import the filter library, so that 'blacken' can be used directly
// instead of using the absolute name 'filter.blacken'.
import filter.*
/*
 * Administrators read access patients, however the
 * classification and diagnosis are blackened in parts
 * also administrator access is to be documented.
 */
policy "administrator access to patient data"
permit
       action.java.name == "findById"
where
       "ROLE_ADMIN" in subject..authority;
obligation
          {
            "type" : "logAccess",
            "message" : subject.name +
                        " has accessed patient data (id=“
                        +resource.id+") as an administrator."
          }
transform
          // filtering with text blackening
          resource |- {
                        @.icd11Code : blacken(2,0,"\u2588"),
                        @.diagnosisText : blacken(0,0,"\u2588")
                      }
```

A SAPL policy set consists of:

- Optional **`import`** statement for PIPs, Functions
- the keyword **`set`**, declaring that the document contains a policy set
- a unique policy name (uniqueness at publishing time in the PDP/PRP)
- The combining algorithm:
  - `deny-unless-permit`
  - `permit-unless-deny`
  - `only-one-applicable`
  - `deny-overrides`
  - `permit-overrides`
  - `first-applicable` (exclusive for policy sets)
- An optional target expression for indexing and policy selection with a leading **`for`** keyword
- Any number (>=1) of policies

```
import filter.*
set "PatientRepository"
  first-applicable
for "PatientRepository" in action.java.instanceof..simpleName

/*
 * All doctors and nurses have full read access on all patient records.
 */
policy "doctor and nurse access to patient data"
permit
      action.java.name == "findById"
where
      "ROLE_DOCTOR" in subject..authority ||
      "ROLE_NURSE" in subject..authority;

/*
 * This policy enables all authenticated users to see the patient list.
 */
policy "all authenticated users may see patient list"
permit
      action.java.name == "findAll"
where
      !("ROLE_ANONYMOUS" in subject..authority);
```

Policy Evaluation

```
/*
 * Permit attending doctors to delete patients
 */
policy "attending doctors may delete patient data"
permit
       action.java.name == "deleteById"
where
    ("ROLE_DOCTOR" in subject..authority);
    subject.name == action.arguments[0].<patient.patientRecord>.attendingDoctor;
```

Only evaluate the rest if this holds true

Where where block is a conjuction of its lines

Return the decision indicated by the entitlement `permit`, when all expressions listed evaluate to `true`.

Access an external policy information point.
`value.<attribute>`
uses value as the input for the attribute finder implementing `attribute`
`value` may be an arbritary JSON value,
Allowing for parametrization of access.

Example: Integration with Spring Data Repositories

The previous examples declare policies for a Spring Data Repository:

Declarative annotations to Repository APIs automatically generate policy enforcement points.

Subscriptions/Requests are Generated via reflections or can be manually configured using the Spring expression language.

Secures arbitrary classes at runtime.

```
public interface PatientRepository {

        @PostEnforce(resource = "returnObject")
        Optional<Patient> findById(Long id);

        @PreEnforce
        Optional<Patient> findByName(String name);

        @PreEnforce
        List<Patient> findAll();

        @PreEnforce
        Patient save(Patient patient);
```

- Value Expression: JSON value or `undefined`
- Identifier Expression: the name of a variable or of a request attribute (`subject`, `resource`, `action` or `environment`)
- Function Expression: a function call (e.g. `simple.get_minimum(resource.array)`)
- Typical arithmetic, logic, string and array operators

Query JSON Objects using JSON Path

Examples

```
{
   "key"    : "value1",
   "array1" : [
        { "key" : "value2" },
        { "key" : "value3" }
    ],
    "array2" : [ 1, 2, 3, 4, 5 ]
}
```

`object.*` `object[*]`	`[ "value1", [ { "key" : "value2" }, { "key" : "value3" } ], [ 1, 2, 3, 4, 5 ] ]`	**Wildcard step** applied to an object returns an array with the value of each attribute - applied to an array it returns the array itself
`object.array2[0:-2:2]`	`[ 1, 3 ]`	**Array slicing step** starting from first to second last element with a step size of two

`object..key`
`object..['key']`
`object..["key"]`
	`[ "value1", "value2", "value3" ]`	**Recursive descent step** looking for an attribute

`object..[0]`	`[ { "key" : "value2" }, 1 ]`	**Recursive descent step** looking for an array index
`object.array2[(3+1)]`	`5`	**Expression step** that evaluates to a number (index) - can also evaluate to an attribute name
`object.array2[?(@>2)]`	`[ 3, 4, 5 ]`	**Condition step** that evaluates to true/false, `@` is a reference to the currently examined item - can also be applied to an object
`object.array2[2,3]`	`[ 3 , 4 ]`	**Union step** for more than one array index
`object["key","array2"]`	`[ "value1", [ 1, 2, 3, 4, 5 ] ]`	**Union step** for more than one attribute

```
/*
 * Visitors which are relatives may see the name, phone number and room number.
 */
policy "visiting relatives access patient data"
permit
       action.java.name == "findById"
where
       "ROLE_VISITOR" in subject..authority;
        /*
         * The next condition invokes the "patient" policy information point and
         * determines the "relatives" attribute of id of the patient.
         * The policy information policy point accesses the database to determine
         * the relatives of the patient and it is checked if the subject is in the
         * list of relatives.
         */
       subject.name in resource.id.<patient.relatives>;
transform
          // Subtractive template with filters removing content
          resource |- {
                              @.medicalRecordNumber      : remove,
                              @.icd11Code                : remove,
                              @.diagnosisText       : remove,
                              @.attendingDoctor          : remove,
                              @.attendingNurse           : remove
                          }
```

SAPL allows to construct arbitary JSON values by constructing objects, or by removing from objects.

Whitelist vs. Blacklist approaches

Static vs. flexible

The filter operator

Policy: time-based

```
policy "permit doctor and nurse read time restricted blood pressure data"
permit
  resource == "bloodPressureData"
where
  "ROLE_DOCTOR" in subject..authority || "ROLE_NURSE" in subject..authority;
  var interval = 1;
  time.localSecond(interval.<clock.ticker>) < 31 || time.localSecond(interval.<clock.ticker>) > 35;
```

Time as Stream

Policy: location-based

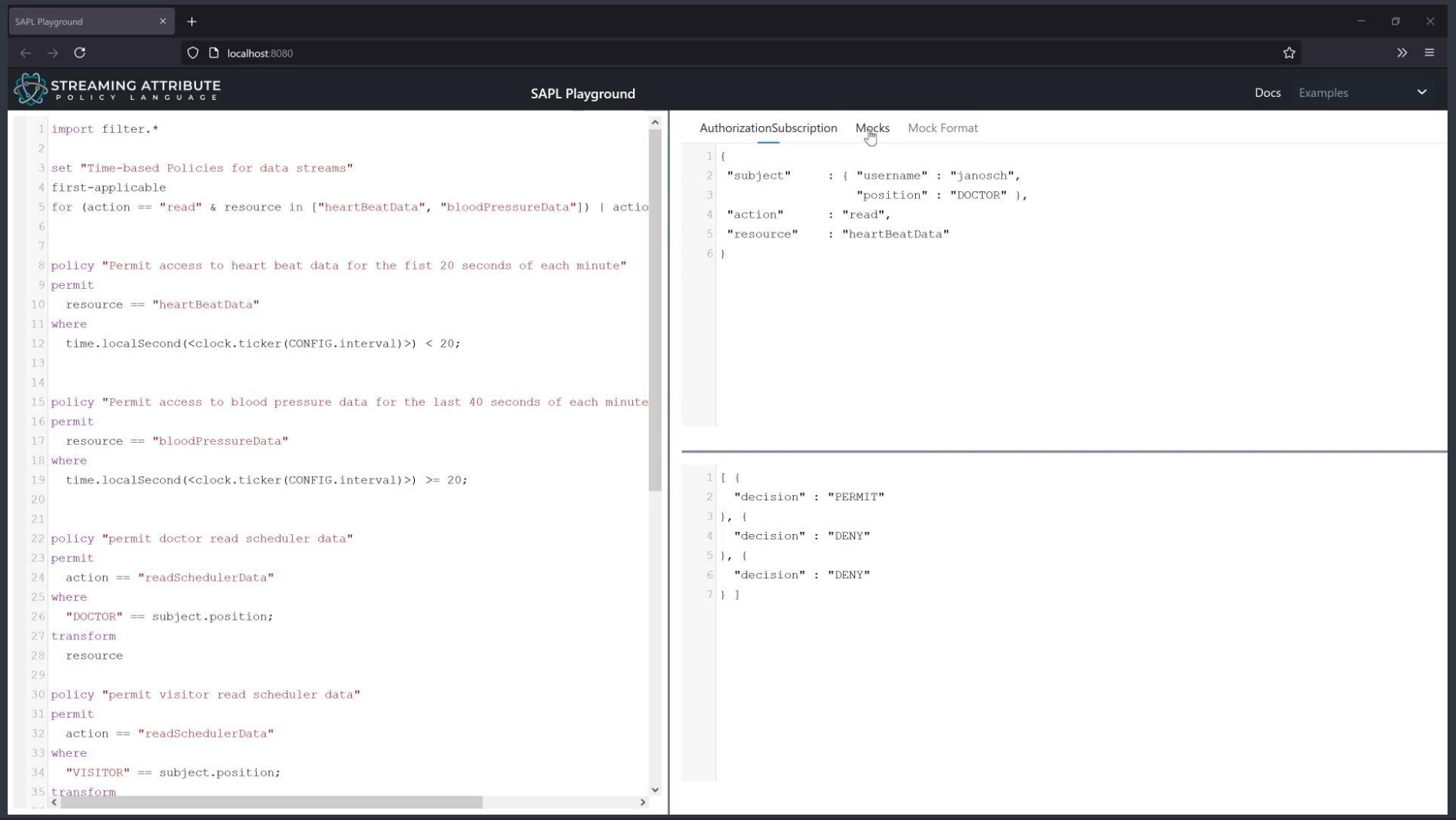
```
policy "restrict_restricted_data"
deny action.httpMethod == "GET" & action.httpUri == "/pil" & resource.classification >= param.restricted
where
        var trackedDevice = deviceTrack.<io.sapl.pip.geo.traccar>;
        var stations = stationsDB.<io.sapl.pip.geo.postgis>.geofences;
        var airports = airportsDB.<io.sapl.pip.geo.postgis>.geofences;
        var proj = getProjection(param.wgs84, param.webMercator);
        var invProj = getProjection(param.webMercator, param.wgs84);
        var airportArea = project(buffer(project(airports[(resource.dep)], proj), param.airportBuffer), invProj);


        var allowedArea = union(airportArea, stations[(resource.dep)]);
        !contains(allowedArea, trackedDevice.position);
```

Location as Stream

Geofences as Streams

SAPL Playground

http://playground.sapl.io



What is actually different beyond syntax?

- Conflict resolution: transformation uncertainty
  I.e., ensure that transformations are unambiguous in decisions

- Handling of data streams and subscriptions.
- require explicit semantics.
- Handling of lazy evaluation, i.e., only subscribe to upstream PIPs which are required given other PIP streams.

Example:

- `subject.<attributeA> || subject.<attributeB>`
- Only have the PDP subscribe to `subject.<attributeB>` as long as `subject.<attributeA> is false`

Full open source implementation of ASBAC policy engine:

Engine: https://github.com/heutelbeck/sapl-policy-engine

Demos: https://github.com/heutelbeck/sapl-demos

- Apache 2.0 License
- Feature rich policy language (SAPL)
  DSL, not expressed on XML/JSON
- Deep integration into the Java Framework Spring Boot
- Declarative API for PEP implementation
- Libraries for Geofencing

SAPL Access Control Lessons

Q&A

SAPL Access Control Lessons

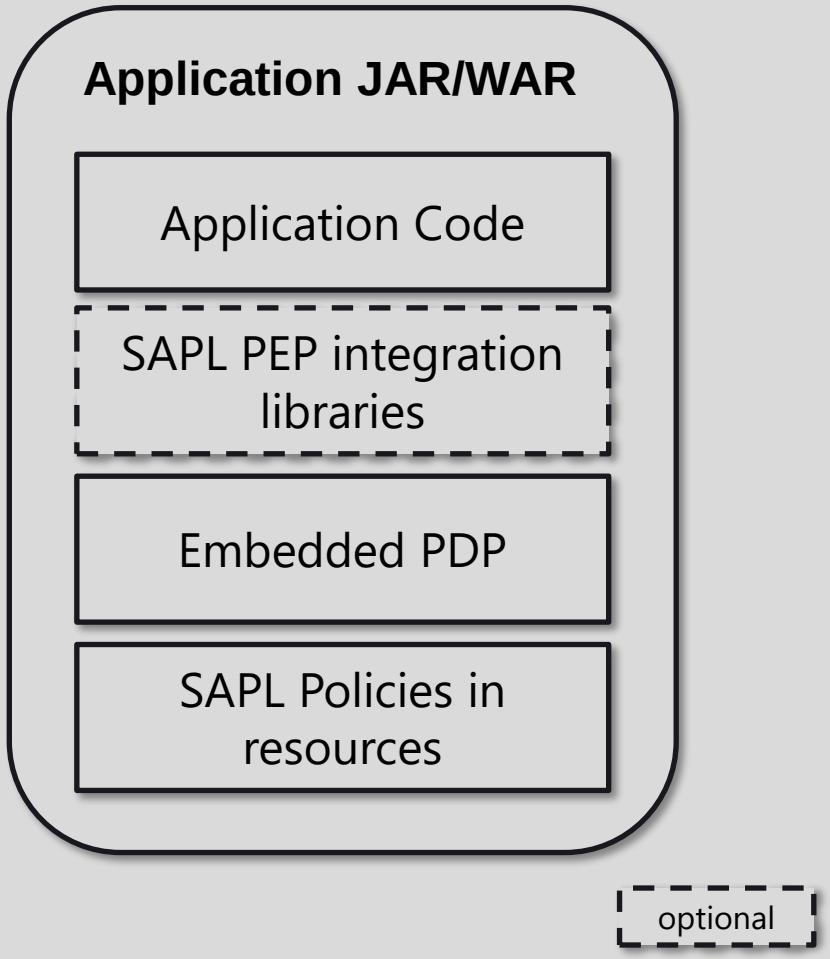
Lesson #05 – Applying ASBAC and SAPL

Dominic Heutelbeck

SAPL Webinar #05

SAPL Deployment (1/3)

Embedded PDP with bundled policies



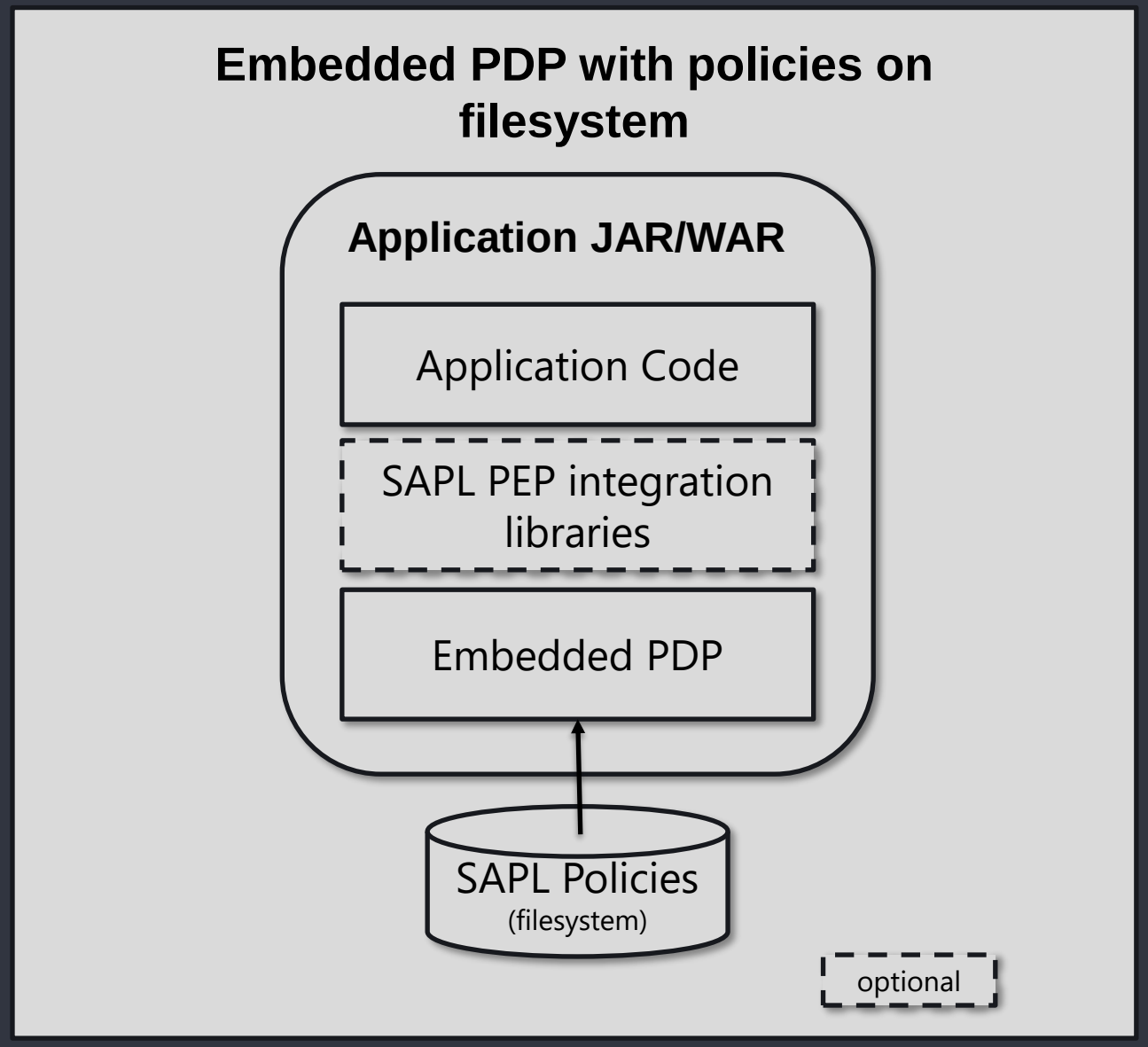
- The SAPL Policy Engine is implemented in Java
- Bundling of Engine with JVM Languages possible

Bundle PDP and policies in application binary (JAR/WAR)

- Optimal performance (no network IO with PDP)
- Eliminates the need for additional infrastructure
- Develop polices alongside the application
- Integrate testing of policies into overall application test suite
- SAPL PEP libraries for Spring Boot allow for seamless integration

SAPL Deployment (2/3)

Bundle PDP and policies on filesystem

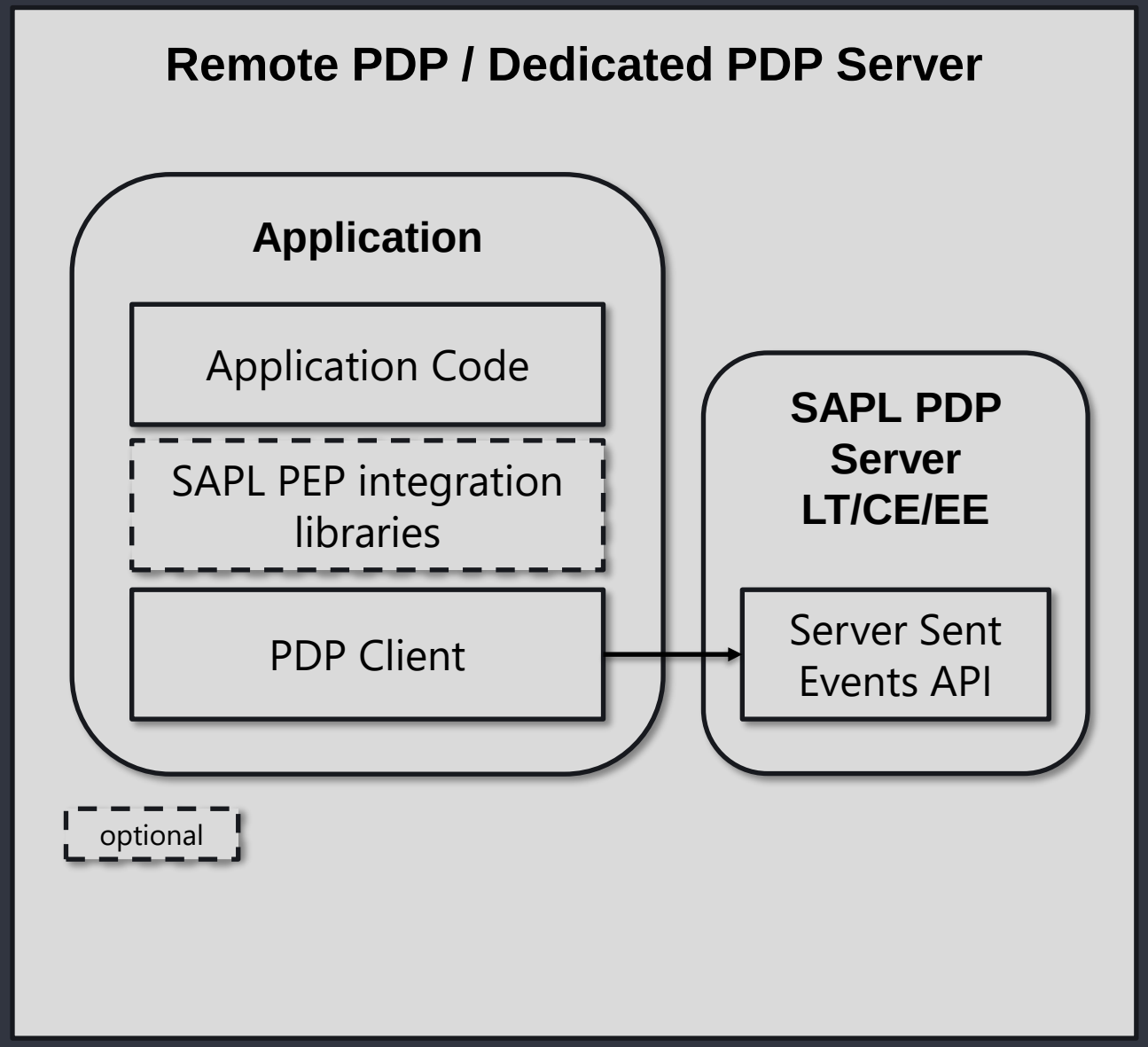


- Optimal performance (no network IO with PDP)
- Eliminates the need for additional infrastructure
- Allows for dynamic policies
- Policy changes without disruption of application
- Polices can be developed independently
- Separation of concerns code vs. policy
- Can be integrated with policy administration workflows

- SAPL PEP libraries for Spring Boot allow for seamless integration

SAPL Deployment (3/3)

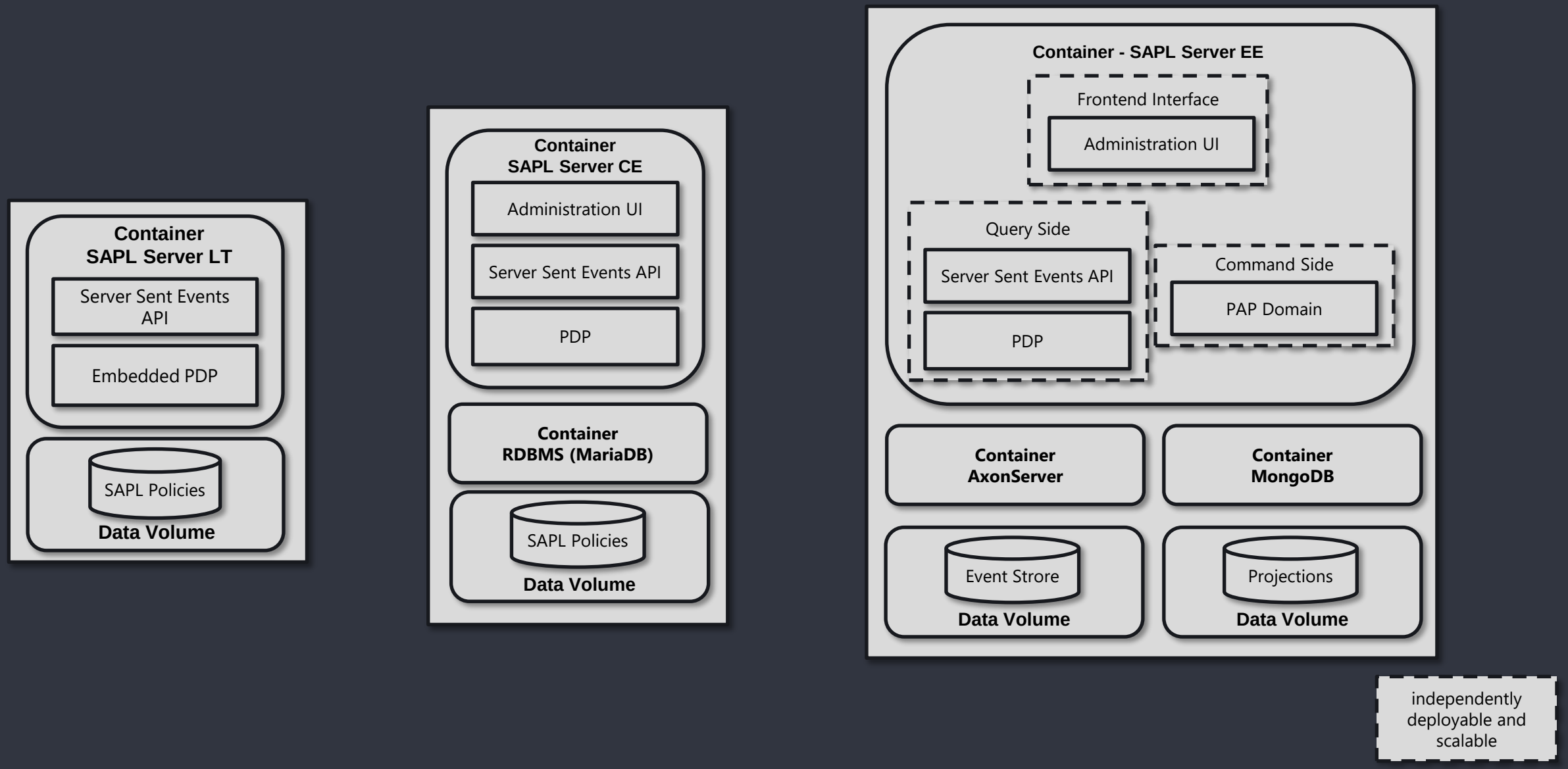
Dedicated PDP server



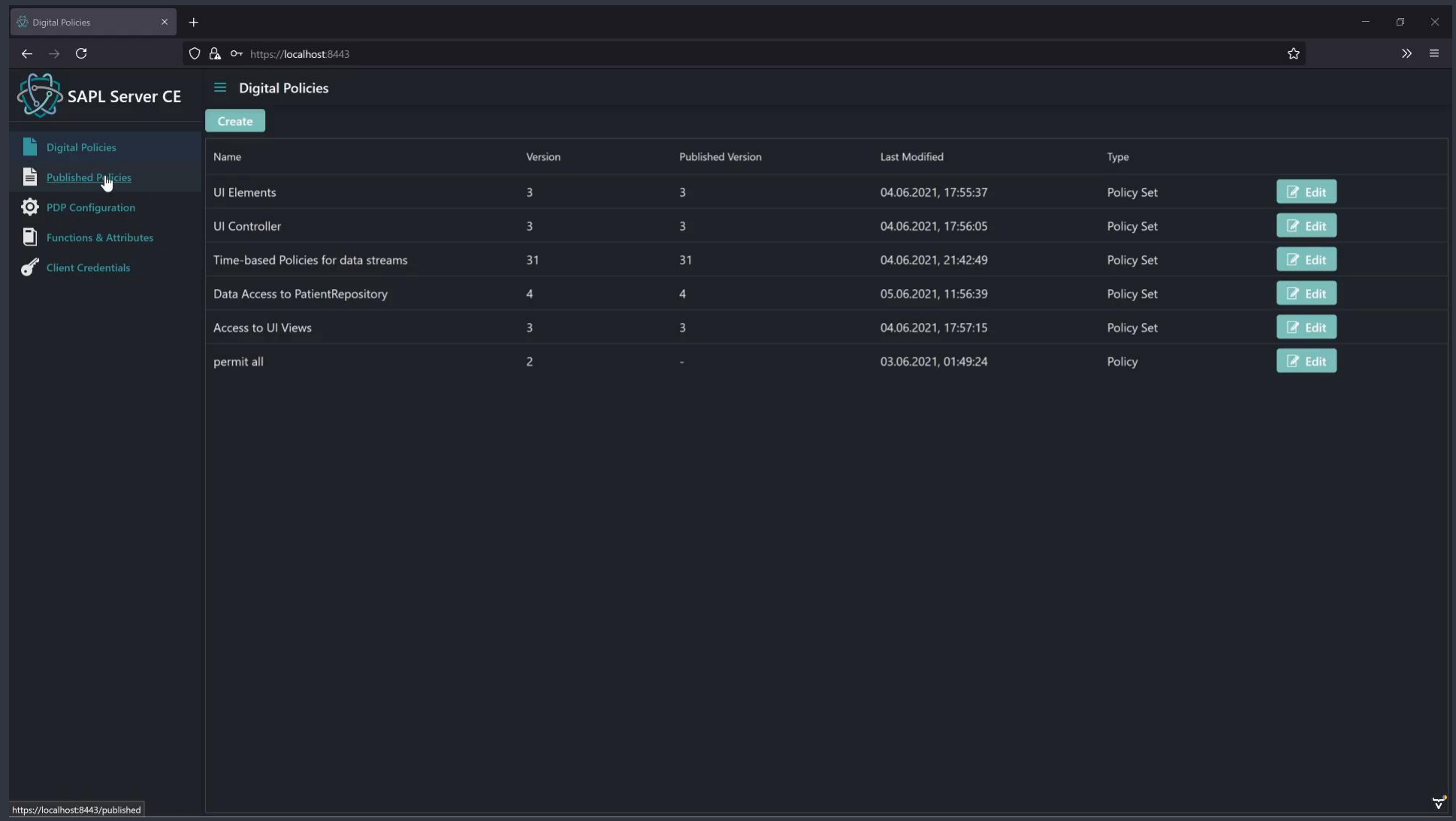
- Application in any language or runtime environment
- Allows for dynamic policies
- Policy changes without disruption of application
- Polices can be developed independently
- Support for different policy administration workflows
- Three server implementations:
  - LT (light) – headless filesystem based
  - CE (Community Edition) – Administration UI, Policy Editor, RDBMS backed
  - EE (Enterprise Edition) - Administration UI, Policy Editor, Axon and MongoDB backed

- SAPL PEP libraries for Spring Boot allow for seamless integration
- SAPL PEP libraries for Python/Django under development
- Simple Server Sent Events (SSE) API easy to consume

SAPL Servers



Policy Administration Frontent (Server CE/EE)



Policy Testing

Scenario : Continuous Integration and Delivery of policies
Problem : Quality Control and validation of policies before deployment
Approach: Treat policies like code. Apply tests and check policy code coverage.
Only deploy on successful tests and upon meeting quality criteria.

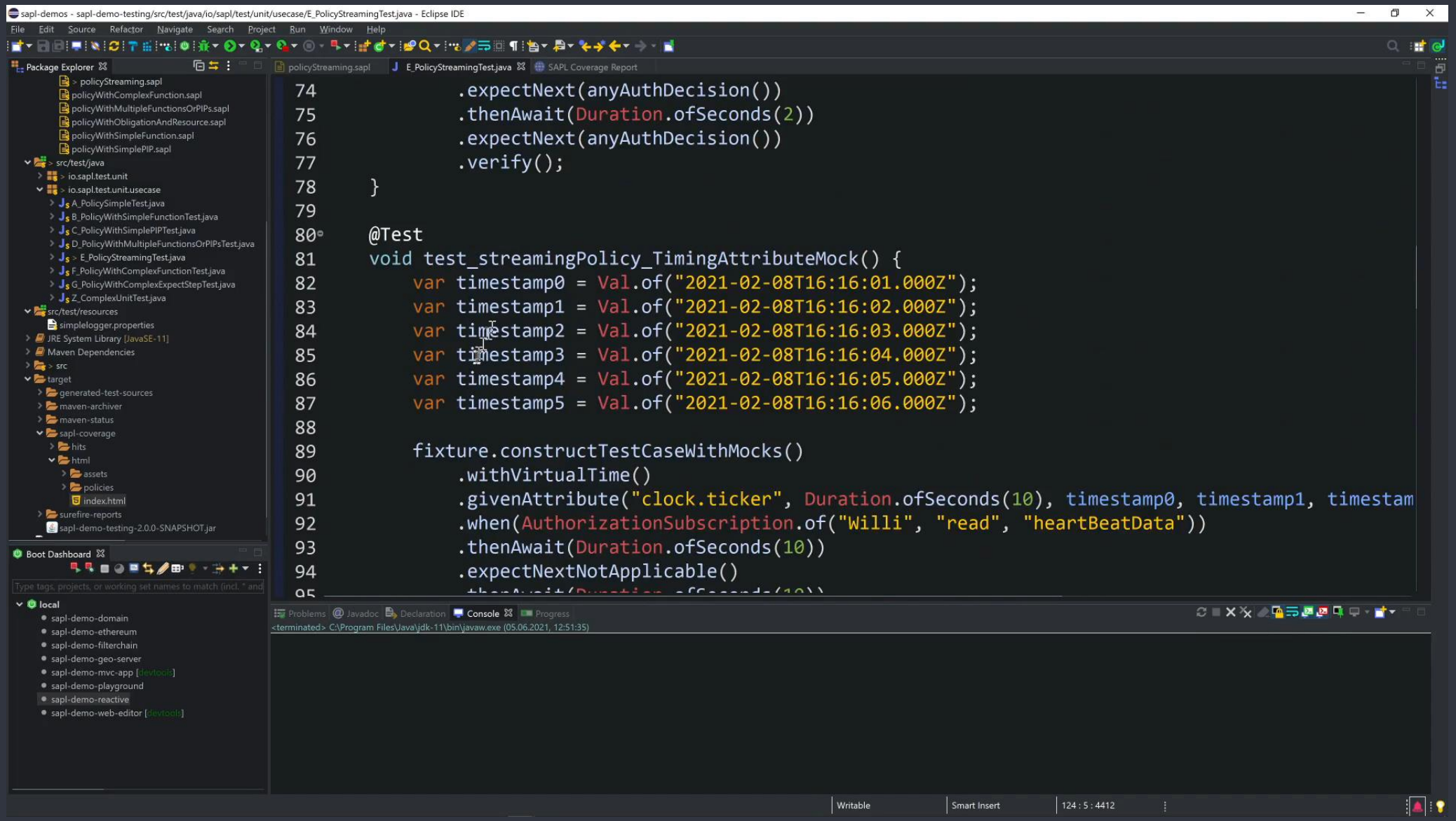
SAPL Approach:

- Write Policy tests in Java
- Test library support
- Code coverage reports
- Maven Plugin
- Quality gate
- Use existing CI/CD pipelines

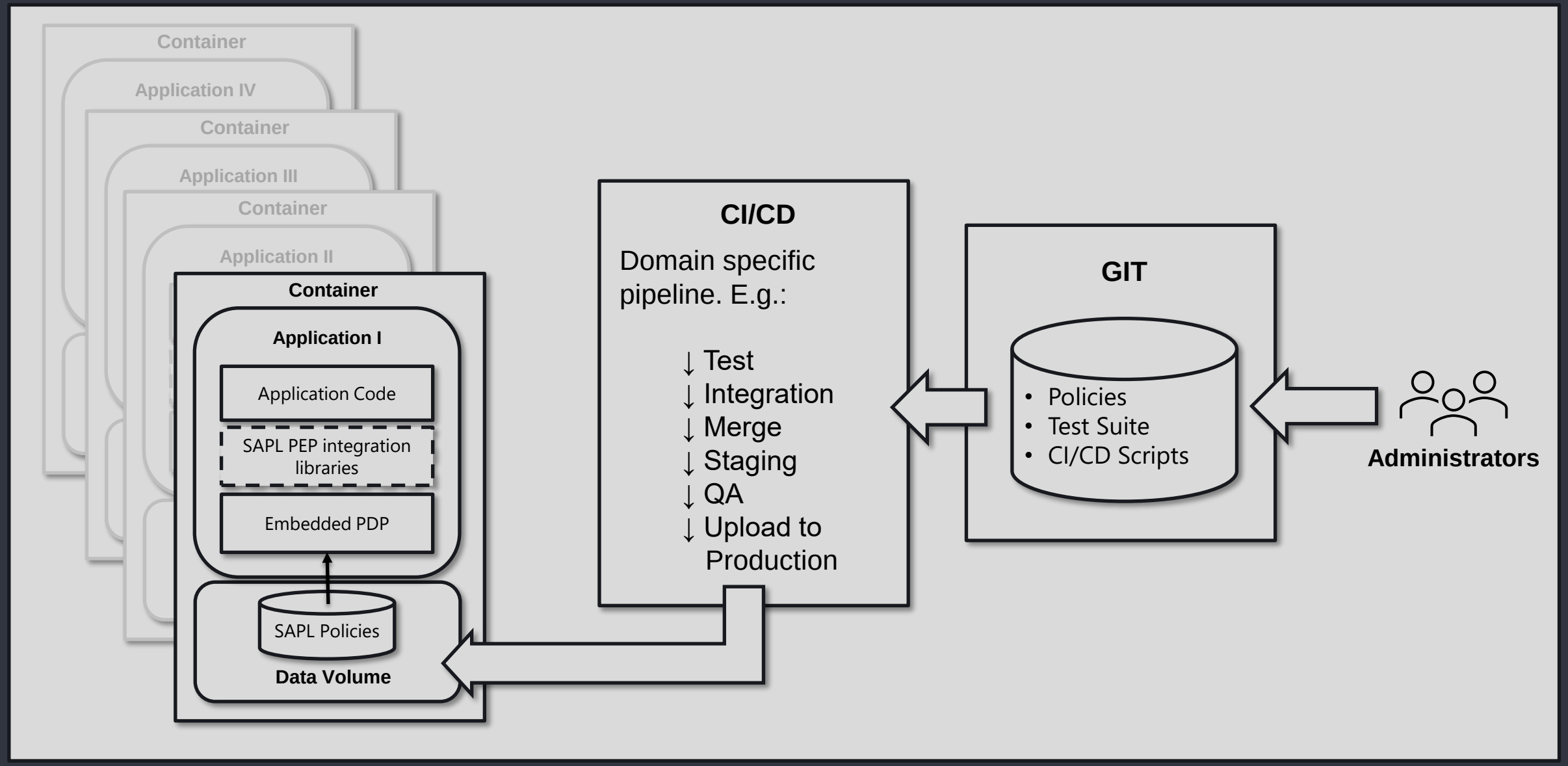
Documentation:
https://sapl.io/docs/sapl-reference.html#testing-your-sapl-policies

Policy Testing

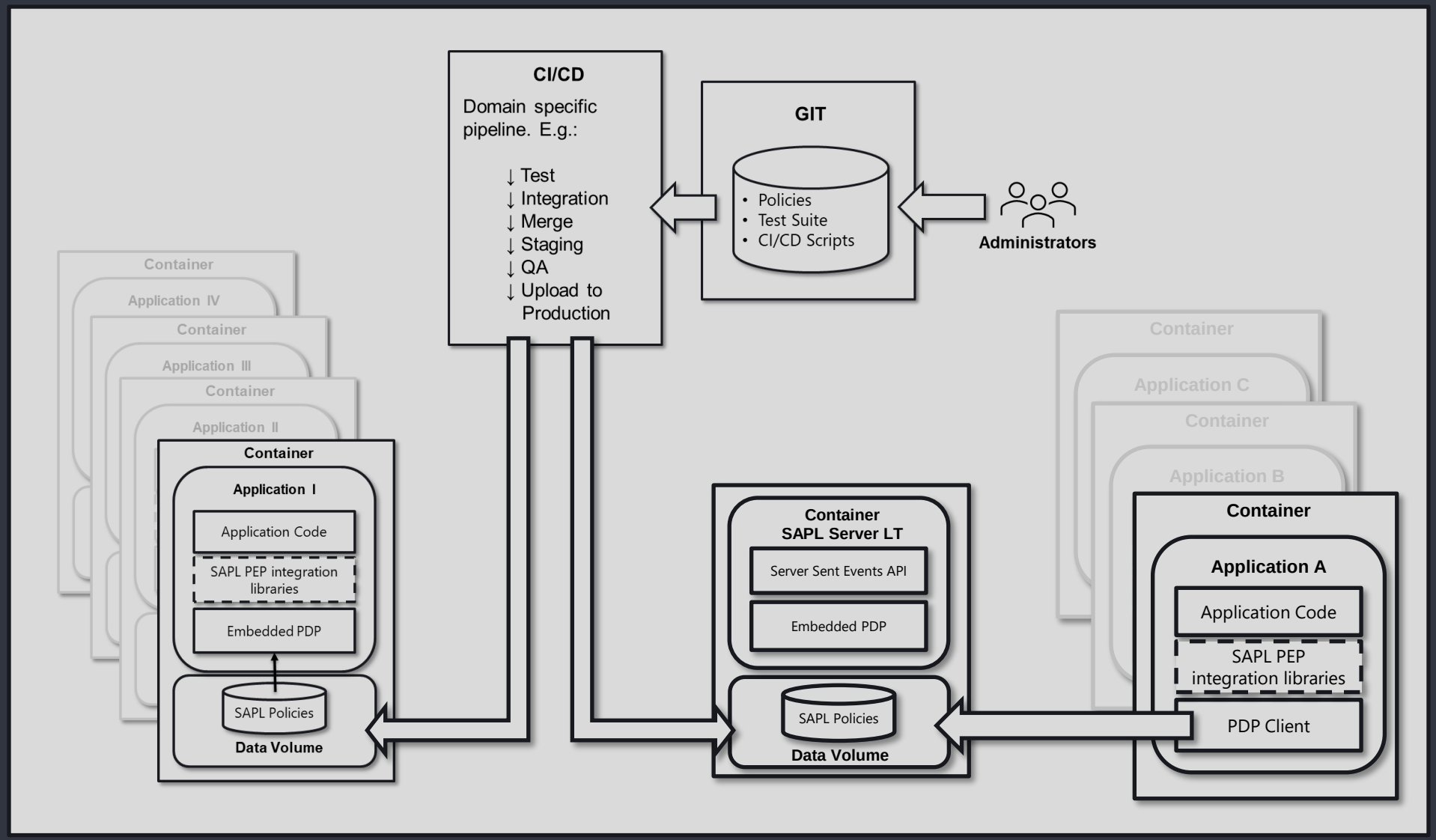
Demo



GIT-Driven Infrastructure (1/2)



GIT-Driven Infrastructure (2/2)



Gather Requirements

1. Identifiy Subjects
2. Identify Resources in the system
3. Identify actions applicable to different resources
4. Describe Natural Language Policies

For each type of resource.

- Under which conditions may what information be shared with which subjects.
- Which subjects may trigger which state change

Recommendation: Follow Domain Driven Design principles and apply the ubiquituous language from the overall DDD process.

2. Identify code paths for PEPs

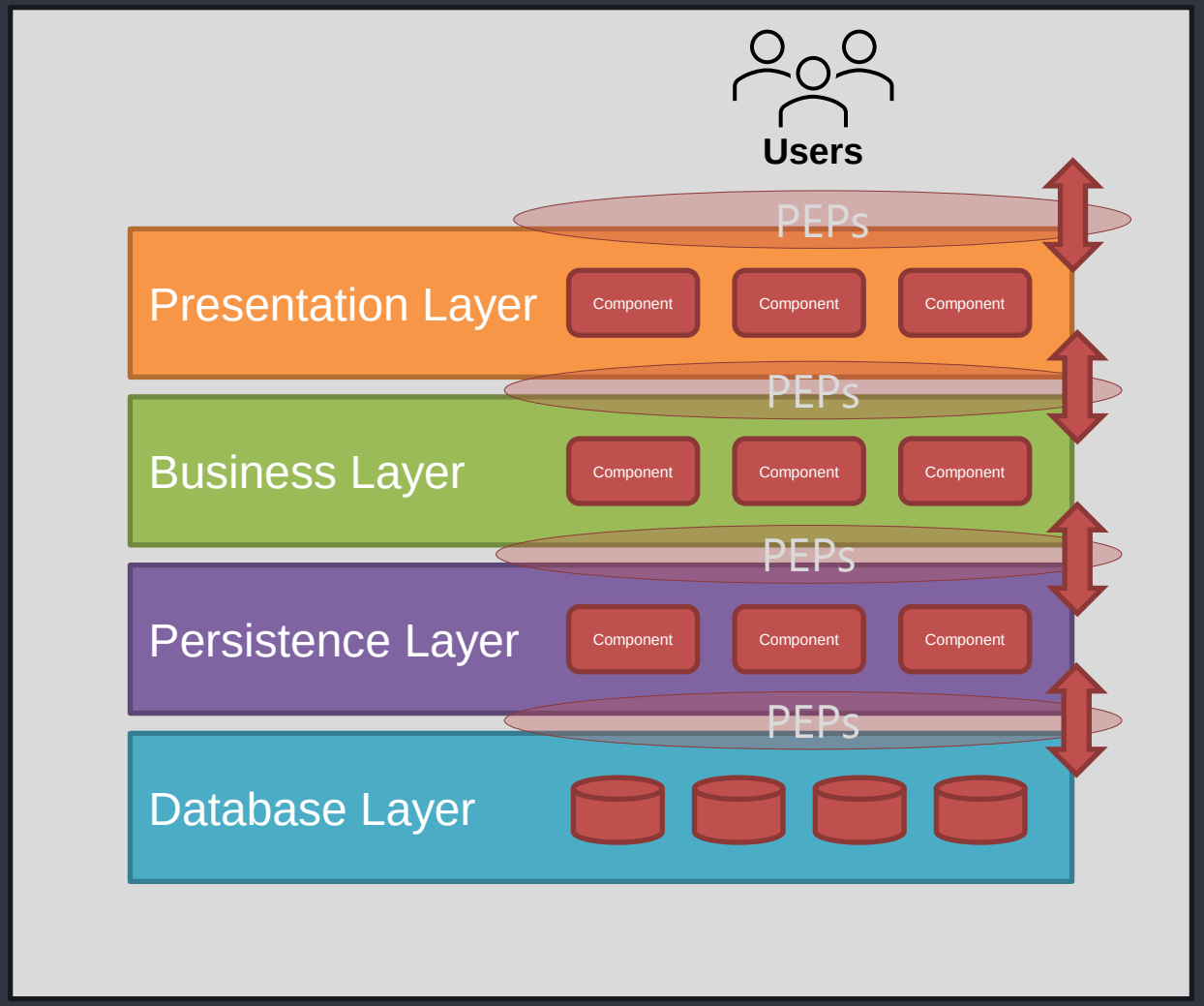
Depends on: Domain and Architecture

Domain: What has to be protected according to which policies

Architecture: Where in the system to protect the resources and how

Example: Layered/Tiered Architecture

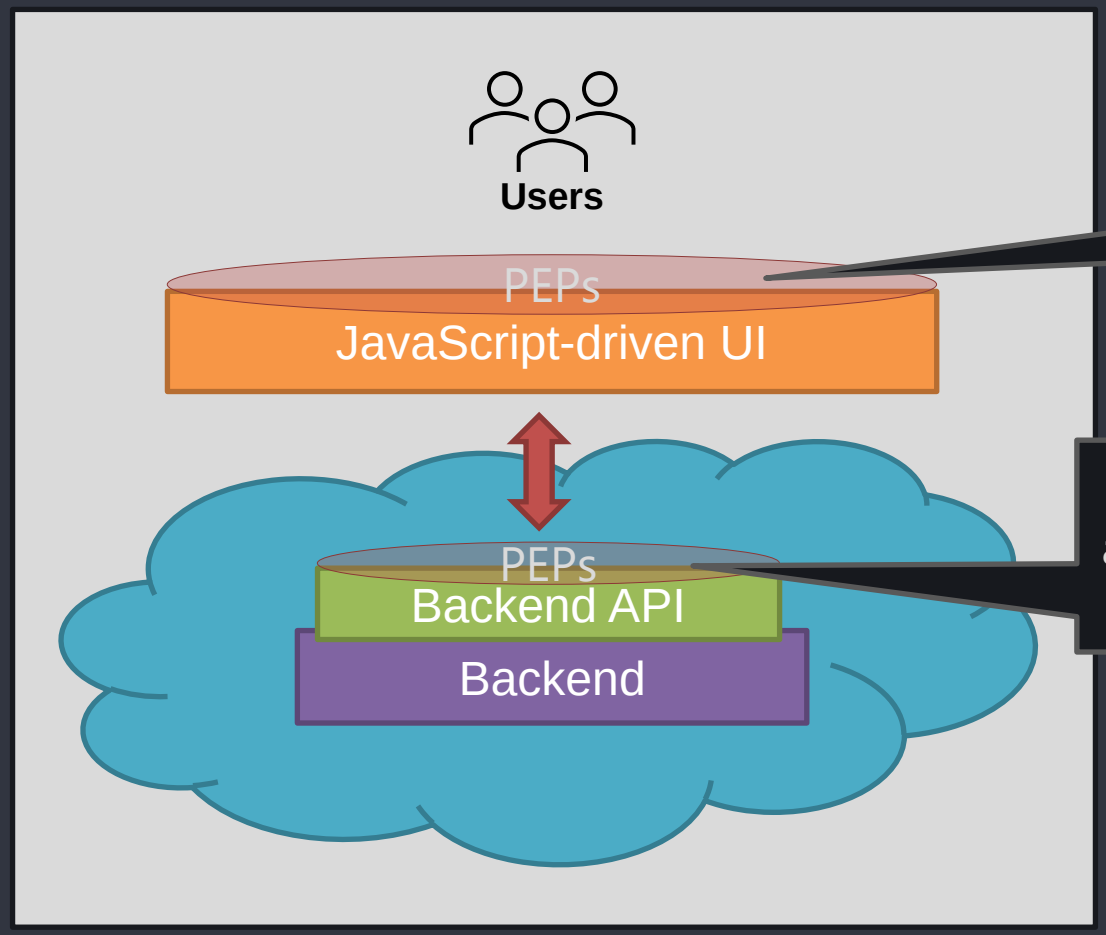
Simplified View



PEPs can be established on every layer of the Architecture

Example Client Side vs. Server Side

Simplified View

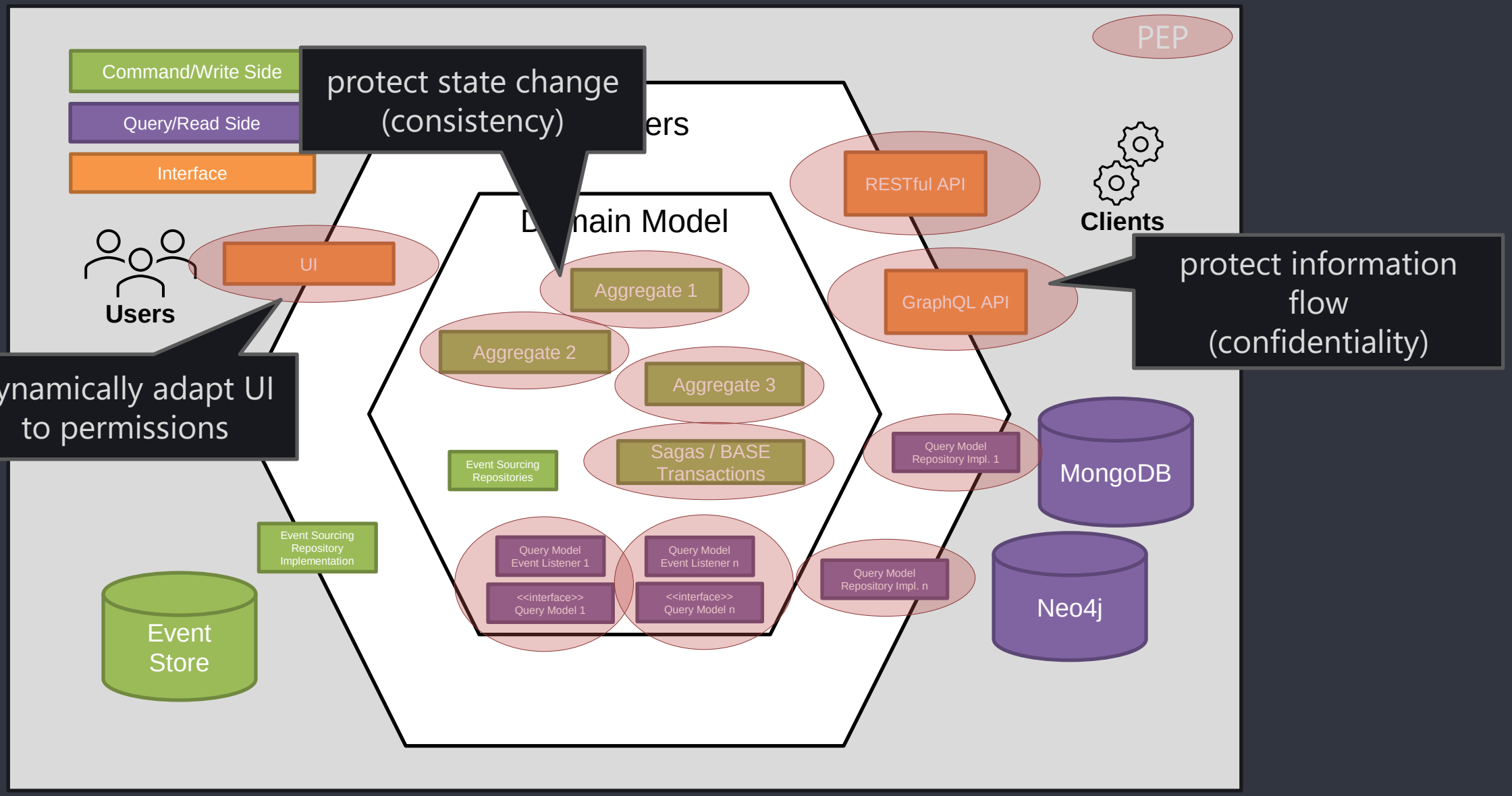


Dynamically adapt UI to permissions

Control flow of information and state change triggered by users

Example: Hexagonal/Onion Architecture

Simplified View



Agree on Schema for Authorization Subscriptions

- technical nature:
  - Derived from technical environment and protocols
  - Contain information like: class/type, method, id, parameters, protocol method (GET, POST,…)
  - Results in policies from developers for developers. Hard to communicate with stakeholders and to link to NLPs
  - Can often be generated automatically
    (potentially large request objects, containig everything the infrastructure could gather)
- domain nature:
  - Derived from ubiquitous (domain) language
  - Uses terms from the language stakeholders use like: account, withdraw, lock account, read balance
  - Typically defined manually
  - Result in policies resembling NLPs
- hybrid: mixing approaches

Agree on Schema for Authorization Subscriptions

- Identify attributes required by NLPs to make decisions
- Identify/decide which attributes are added to the subscription, and which ones have to be retrieved by the PDP using custom PIPs
- Attributes which are subject to change over time must be retrived via PIPs
- Future work:
  SAPL will support JSON schema to assist in policy authoring and validation.

Implement Custom PIPs

- Implement any PIPs required by your domain to make decisions
  - Use publish/subscribe protocols whenever possible
  - Legacy sources may have to be polled

Implement Policy Enforcement Points

- A Policy Enforcement Point must:
  - Enforce the decisions made by the PDP
  - handling of any constraints (advice/obligation)
  - Resource data transformation

- Typical PEPs:

- Wrap a method/function access
  - Before executing the method
  - After executing the method -> enables resource transformation, use of return value in subscription.
  - Only consume the 1st decision from the PDP, as the method access does not span a longer time/session

Implement Policy Enforcement Points

Simple Method Warapping

Java Spring-Boot Application:

```java
public interface PatientRepository {

    @PostEnforce(resource = "returnObject")
    Optional<Patient> findById(Long id);

    @PreEnforce
    Optional<Patient> findByName(String name);

    @PreEnforce
    List<Patient> findAll();

    @PreEnforce
    Patient save(Patient patient);

    @PreEnforce
    void deleteById(Long id);

    @Modifying
    @PreEnforce
    @Transactional
    @Query("update Patient p set p.name = ?1 where p.id = ?2")
    void updateNameById(String name, Long id);

    @Modifying
    @PreEnforce
    @Transactional
    @Query("update Patient p set p.diagnosisText = ?1 where p.id = ?2")
    void updateDiagnosisTextById(String diagnosisText, Long id);
```

Python Django Application:

```python
@pre_enforce
def patients(request):
    all_patients = Patient.objects.all()
    return render(request, "medical/patients.html", {"patients": all_patients})
```

PEP integration libraries can generate these PEPs automatically.

By default subscriptions will be of technical nature and verbose

Annotations allow for full customization of subscriptions to be domain driven

Includes constraint handling APIs

Example from Python Django Application:

```
{% enforce action="'secret.view'" %}
    <p>You are permitted to see this secret message.</p>
    {% denied %}
    <p>You are not permitted to see something secret here.</p>
{% endenforce %}
```

Templating engines are traditionally request/response-based.

Like method wrapping:

only consume the 1st decision from the PDP

The pre and post enforce pattern can me applied to asynchronous data sources as well.

In the simple case, again only consume one decision, but enforce constraints and continuously throughout the lifecycle of the asynchronous data stream.

```
@PreEnforce
public Mono<String> getMonoString() {
    return Mono.just("data returnded by Mono");
}

@PostEnforce(resource = "returnObject")
public Mono<String> getMonoStringWithPreAndPost() {
    return Mono.just("I will be decorated with * on the left and right, because the policy said so");
}

@PreEnforce
public Flux<Integer> getFluxNumbers() {
    return Flux.just(0, 1, 2, 3, 4, 5, 6, 7, 8, 9).delayElements(Duration.ofMillis(500L));
}
```

Fully Streaming Oriented PEPs

Subscribe to all decisions made by the PDP.
Update the enforcement strategy based on the incoming decisions.

There are three basic enforcement strategies:

- **EnforceTillDeny**
- **EnforceDropWhileDeny**
- **EnforceRevcoverableIfDeny**

Enforce Till Deny

If first decision is PERMIT, then grant access until the first non-PERMIT decision is sent by the PDP. Then cancel the subscription to the stream.

Enforce Drop While Deny

Subscribe to the resource after the first decision, make it a hot source. Filter out all events from the data stream wile the most recent decision is not PERMIT.

Keep the subscription alive as long as the client does.

The client is not aware of access denied events.

Enforce Revcoverable If Deny

Subscribe to the resource after the first decision, make it a hot source. Filter out all events from the data stream wile the most recent decision is not PERMIT. However, on a non-permit signal an Access Denied downstream. Enable the client to recover and wait for the resource to become available again.

Keep the subscription alive as long as the client does.

The client is aware of access denied events.

Fully Streaming Oriented PEPs

Automatic PEP creation is under development for the Spring integration and will be delivered with the 2.0.0 release.

Conclusion

- Implementing SAPL PEPs using integration libraries is straight forward.
- SAPL introduces a some complexity into the development process.
- SAPL is currently the only engine implementing ASBAC while also implementing a user-friendy ABAC developer experience.
- Deployment is flexible
- Results in potentially polling-free reactive applications. E.g., real-time collaborative applications with dynamically changing access rights depending on process state.

SAPL Access Control Lessons

Q&A

Speaker

Prof. Dr. Dominic Heutelbeck
Geschäftsführender Vorstand
FTK - Forschungsinstitut für Telekommunikation und Kooperation e. V.
Wandweg 3
44149 Dortmund

Web: https://ftk.de
Mail: dheutelbeck@ftk.de

Full open source implementation of ASBAC policy engine:
Engine: https://github.com/heutelbeck/sapl-policy-engine
Demos: https://github.com/heutelbeck/sapl-demos

References

[RFC4949] R. Shirey (2007), „Internet Security Glossary, Version 2". https://tools.ietf.org/html/rfc4949 (accessed 21.11.2019)

[NIST80053] Joint Task Force Transformation Initiative (2013), "Security and Privacy Controls for Federal Information Systems and Organizations", NIST Special Publication 800-53 Revision 4, NIST National Institute of Standards and Technology, (including updates 02-22-2015) https://doi.org/10.6028/NIST.SP.800-53r4 (accessed 21.11.2019)

[NIST800162] Hu et. Al (2014), "Guide to Attribute Based Access Control (ABAC) Definition and Considerations". NIST National Institute of Standards and Technology, NIST Special Publication 800-162, (including updates as of 08-02-2019) https://doi.org/10.6028/NIST.SP.800-162 (accessed 21.11.2019)

[XACML] eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS Standard, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html (accessed 21.11.2019)

[ITUTX.800] ITU-T Recommendations X.800 "Security architecture for Open System Interconnection" https://www.itu.int/rec/T-REC-X.800-199103-I (accessed 21.11.2019)

[Saltzer74] Saltzer, Jerome H. (1974). "Protection and the control of information sharing in multics". Communications of the ACM. 17 (7): 388–402. doi:10.1145/361011.361067

[Needham72] Needham, R.M., (1972) 'Protection systems and protection implementations,' Proc. 1972 Fall Joint Computer Conference, AFIPS Conf. Proc., vol. 41, pt. 1, pp. 571-578.

[Bossard11] Bossard, D. (2011), "Coarse-grained vs. fine-grained access control – part I", https://www.webfarmr.eu/2011/05/coarse-grained-vs-fine-grained-access-control-part-i/ (accessed 21.11.2019)

[Gruenbacher03] Grünbacher, A. (2003) "POSIX Access Control Lists on Linux", 2003 USENIX Annual Technical Conference, https://www.usenix.org/legacy/events/usenix03/tech/freenix03/full_papers/gruenbacher/gruenbacher.pdf (accessed 21.11.2019)

[ANSI03] American National Standards Institute, "Role Based Access Control", Secretariat, Information Technology Industry Council, BSR INCITS 359, DRAFT, 10 November 2003. https://profsandhu.com/journals/tissec/ANSI+INCITS+359-2004.pdf (accessed 21.11.2019)

References

[ISACTA_SOD] ISACA Glossary https://www.isaca.org/Pages/Glossary.aspx?tid=1835&char=S (accessed 21.11.2019)

[ELLIOTT&K10] Elliott, A., & Knight, S. (2010, July). "Role Explosion: Acknowledging the Problem". In Software Engineering Research and Practice (pp. 349-355).

[BELL2005] Bell, D. E. (2005). "Looking Back at the Bell-LaPadula Model" Proceedings of the 21st Annual Computer Security Applications Conference. Tucson, Arizona, USA. pp. 337–351. doi:10.1109/CSAC.2005.37

[BMIA007] Bundesministerium des Innern, "Anlage VII Verschlusssachenanweisung Hinweise zur Handhabung von Verschlusssachen ausländischer Staaten sowie über- oder zwischenstaatlicher Organisationen"

[ISO27001] ISO/IEC 27001 — Information technology - Security Techniques – Information security management systems — Requirements.

[SÜG] „Gesetz über die Voraussetzungen und das Verfahren von Sicherheitsüberprüfungen des Bundes und den Schutz von Verschlusssachen" https://www.gesetze-im-internet.de/s_g/

[BIBA75] Biba, K. J. "Integrity Considerations for Secure Computer Systems", MTR-3153, Mitre Corporation, Juni 1975

[CLARKWIL87] David D. Clark, David R. Wilson: A Comparison of Commercial and Military Computer Security Policies. IEEE Symposium on Security and Privacy, S. 184, 1987.

[FEFEWI16] Ferraiolo, D. F., Feldman, L., & Witte, G. A. (2016). Exploring the next generation of access control methodologies.

[INCITS 565-2020] American National Standard for Information Technology – Next Generation Access Control (NGAC)

[FEIG19] David Ferraiolo (NIST) & Ignasi Barrera (Tetrate), NGAC talk at the Service Mesh Day 2019 https://www.youtube.com/watch?v=-4D8N0M-syA

[BRNA89] Dr. David F.C. Brewer and Dr. Michael J. Nash: The Chinese Wall Security Policy. In: IEEE (Hrsg.): Proceedings of IEEE Symposium on Security and Privacy. 1989, S. 206–214

[HaWaUl76] Harrison, Michael A.; Ruzzo, Walter L.; Ullman, Jeffrey D. (August 1976). "Protection in Operating Systems". Communications of the ACM. 19 (8): 461–471

[GrDe72] Graham, G. Scott, and Peter J. Denning. "Protection: principles and practice." Proceedings of the May 16-18, 1972, spring joint computer conference. 1971.