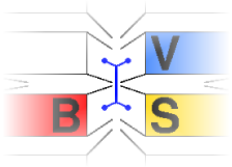
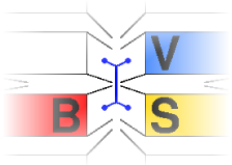


An Implementation of a Privacy Enforcement Scheme based on the Java Security Framework using XACML Policies

Thomas Scheffler, Stefan Geiss, Bettina Schnor
{scheffler, schnor}@cs.uni-potsdam.de

Department of Computer Science, University of Potsdam
Potsdam, Germany

September 8th, 2008

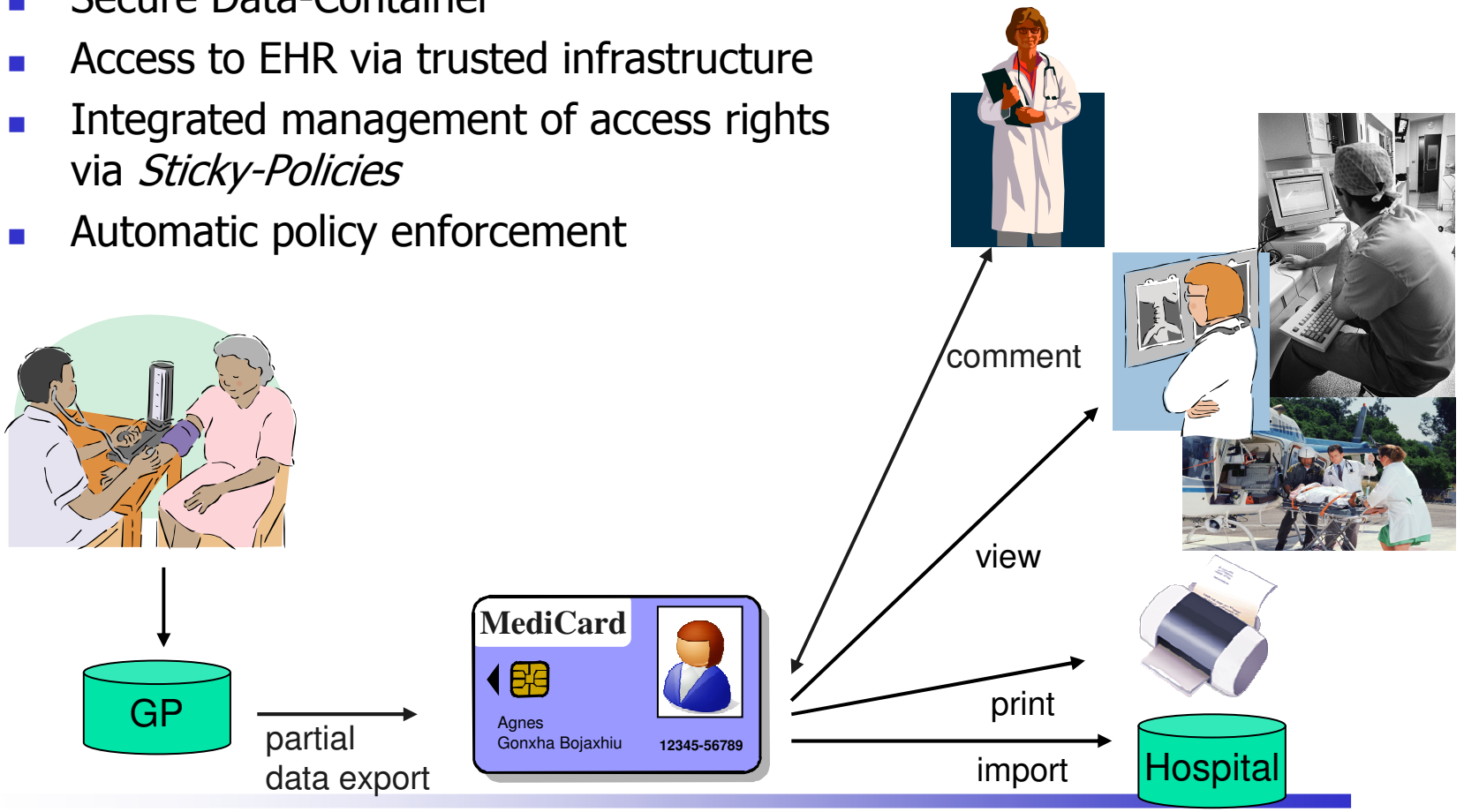


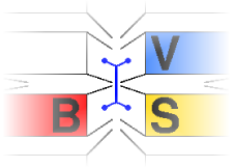
Overview

- Motivation and Idea
- XACML Access Policies
- Privacy Enforcement Scheme
- Conclusion and Outlook

Distributed Electronic Health Records

- Storage of patients medical history in Mobile EHRs
- Secure Data-Container
- Access to EHR via trusted infrastructure
- Integrated management of access rights via *Sticky-Policies*
- Automatic policy enforcement





Data Privacy for Medical Records

Question: How can the sensitive private data of a medical record be protected in the presence of different actors?

- Sensitive data stored as semi-structured XML-Documents
- Distributed Access Control Framework
  - Requests to resources must be evaluated at real time
  - Deployment of trusted infrastructure
- Automated enforcement of authorisations

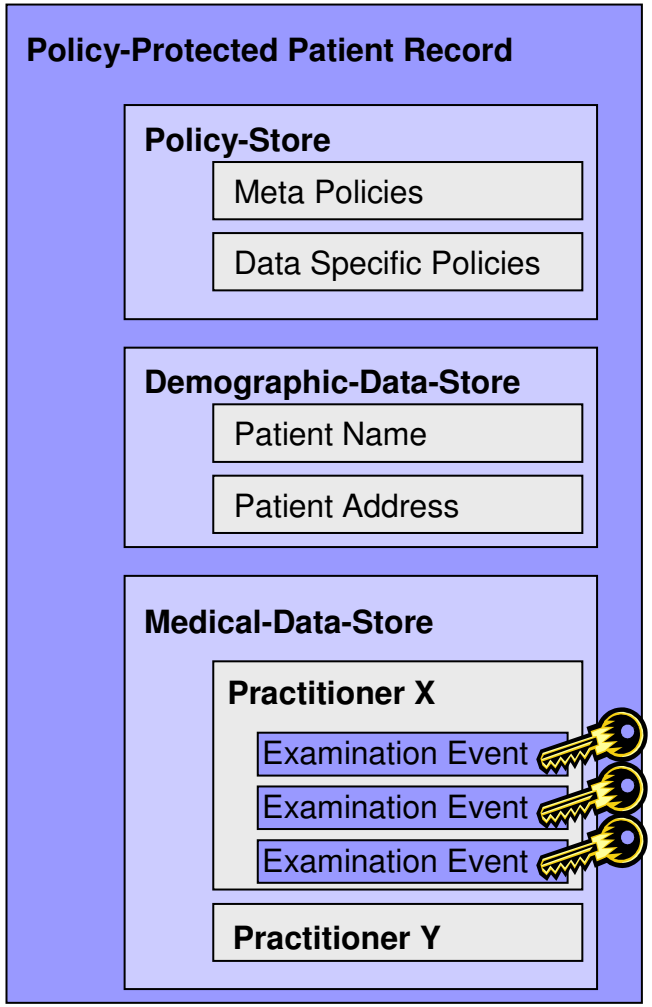
Data Privacy = Access Control + Usage Control

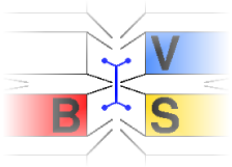
Owner Controlled Data Access Policies

- The Data Owner specifies the access policy for data. The Data User is bound to follow this policy.
- The protected data are stored together with the usage policies as a Sticky Policy Object and can be referenced anytime and anywhere by different data users.

Sticky Policies

- The Policy-Store holds
  - Generic Policies
  - User-generated Policies
- Demographic data about the patient will be stored in the Demographic-Data-Store
- The Medical-Data-Store contains medical data about examinations and treatments of the patient

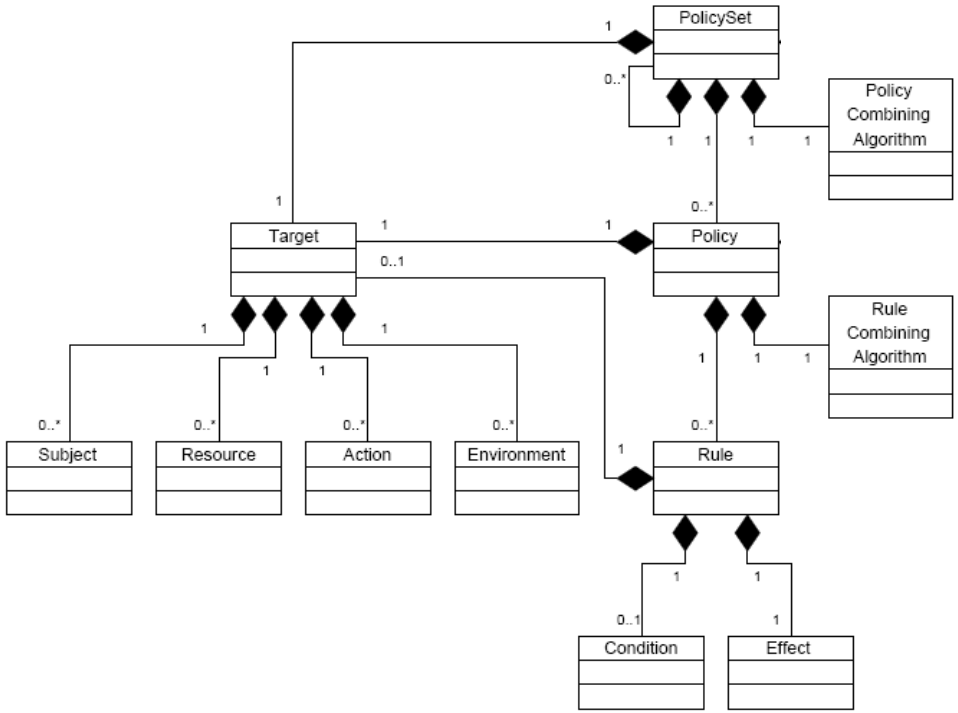




XACML Access Policy

XACML

- eXtensible Access Control Markup Language (XACML) developed by OASIS, current version 2.0
- Policy Language and Request/Response Language



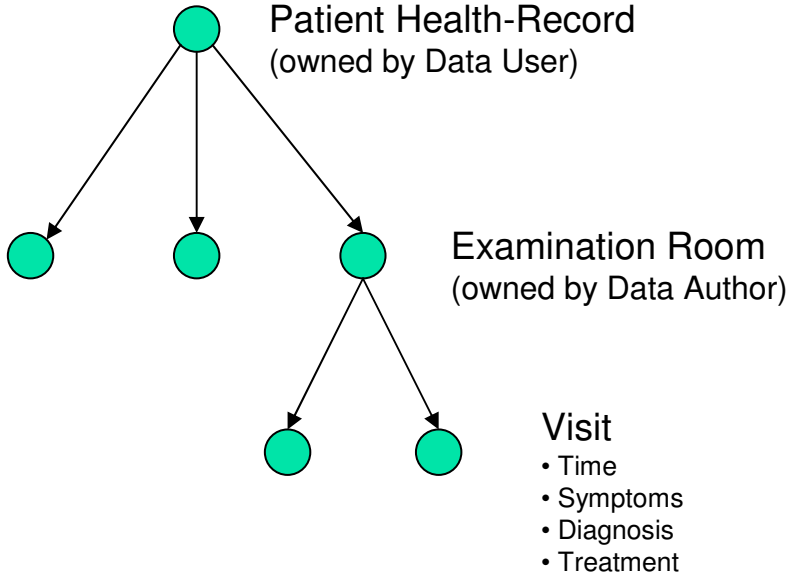
Source: OASIS, XACMLv2

Rights-Expression Languages

- Common security semantics
  - Greater expressability than simple ACL (conditional access, time based function support, inference mechanism)
  - Better suited for changing policies
  - Higher level of abstraction
- Separation of policy expression from enforcement mechanisms

HealthRecord-Example

```
<HealthRecord>
  <Policy PolicyId="1"
    RuleCombiningAlgId="deny-overrides">
     <Rule Effect="Permit">
        <Target> ...
        <Condition> ... </Condition>
     </Rule>
  </Policy>
  <fileData>
     <demographicData> ... </demographicData>
     <insuranceData> ... </insuranceData>
  </fileData>
  <medicalHistory>
     <examinationRoom>
        <practitioner>
           <practitioner.id>CN=Nic Riviera,
                      ...</practitioner.id>
        </practitioner>
        <examinations>
           <visit date="1988-06-01 16:35:27">
             <diagnosis>Cancer</diagnosis>
             <treatment>Chemo
                       therapy</treatment>
           </visit>
           <visit date="2005-08-30 10:35:27">
             ...
           </visit>
        </examinations>
      </examinationRoom>
  </medicalHistory>
</HealthRecord>
```



Problem: Generic XACML Policy Rules

Generic Rule Example: Data Authors should be able to access their own data entries

Implementation in XACML requires:

- Determination of current requestor via an XACML **SubjectAttributeDesignator**
- Dynamic referencing of data author/owner of requested resource via XPATH expression from the supplied **ResourceAttributeDesignator**

Problem: XACML v2.0 currently only supports the evaluation of static XPATH expressions.

Solution: Generic XACML Policy Rules

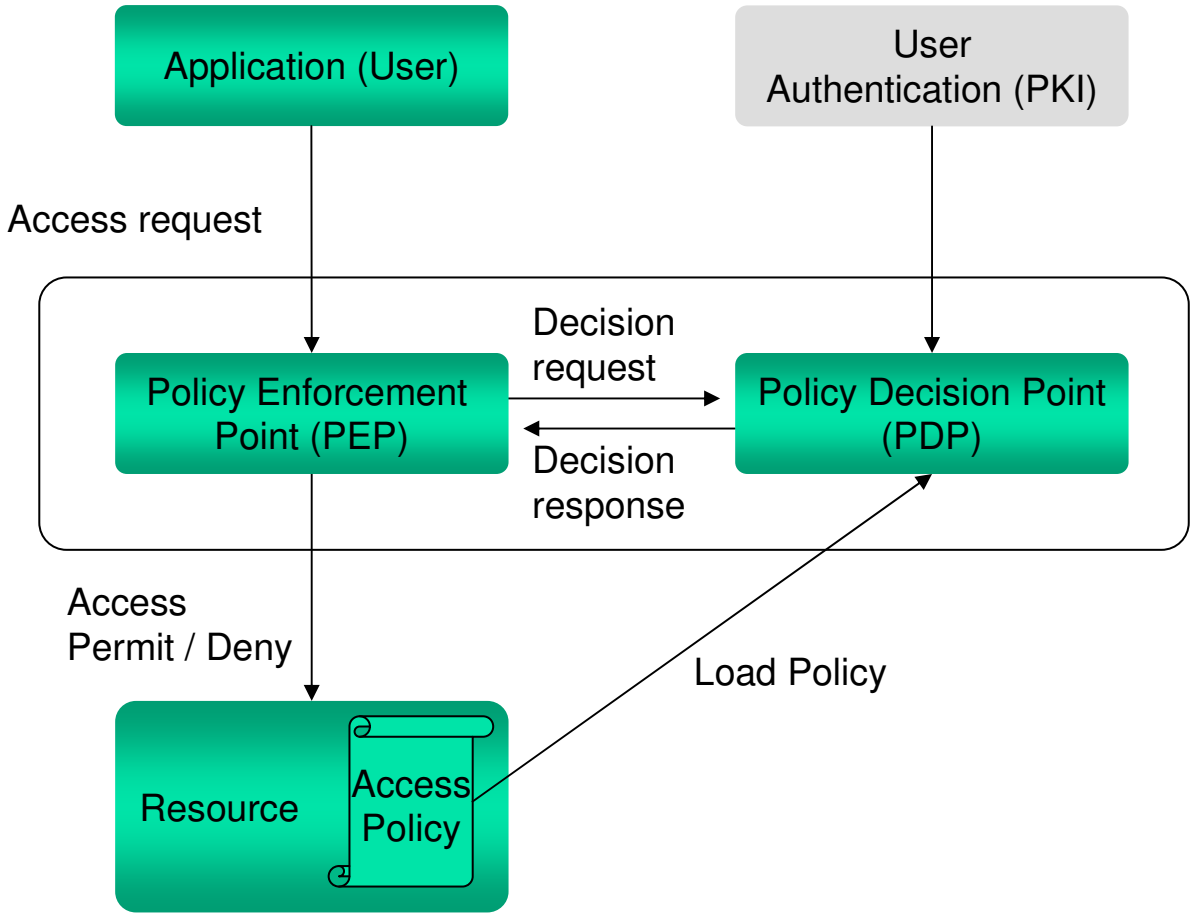
Necessary implementation of a **new** XACML comparison function:

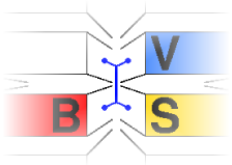
- Compare a X500 name with string type, evaluated as another X500 name
- Allow the dynamic referencing of owner names for arbitrary resources through XACML string concatenation

```
<Condition FunctionId="function:xpath-node-element-x500-compare">
    <Apply FunctionId="x500Name-one-and-only">
        <SubjectAttributeDesignator DataType="x500Name" AttributeId="subject-id" />
    </Apply>
    <Apply FunctionId="string-concatenate">
        <Apply FunctionId="string-one-and-only">
               <ResourceAttributeDesignator AttributeId="resource-id"
                DataType="http://www.w3.org/2001/XMLSchema#string" />
        </Apply>
        <AttributeValue DataType="string">
              /parent::visit/parent::examinationRoom/parent::practitioner/@id
         </AttributeValue>
    </Apply>
</Condition>
```

Privacy Enforcement Scheme

Policy Enforcement Architecture





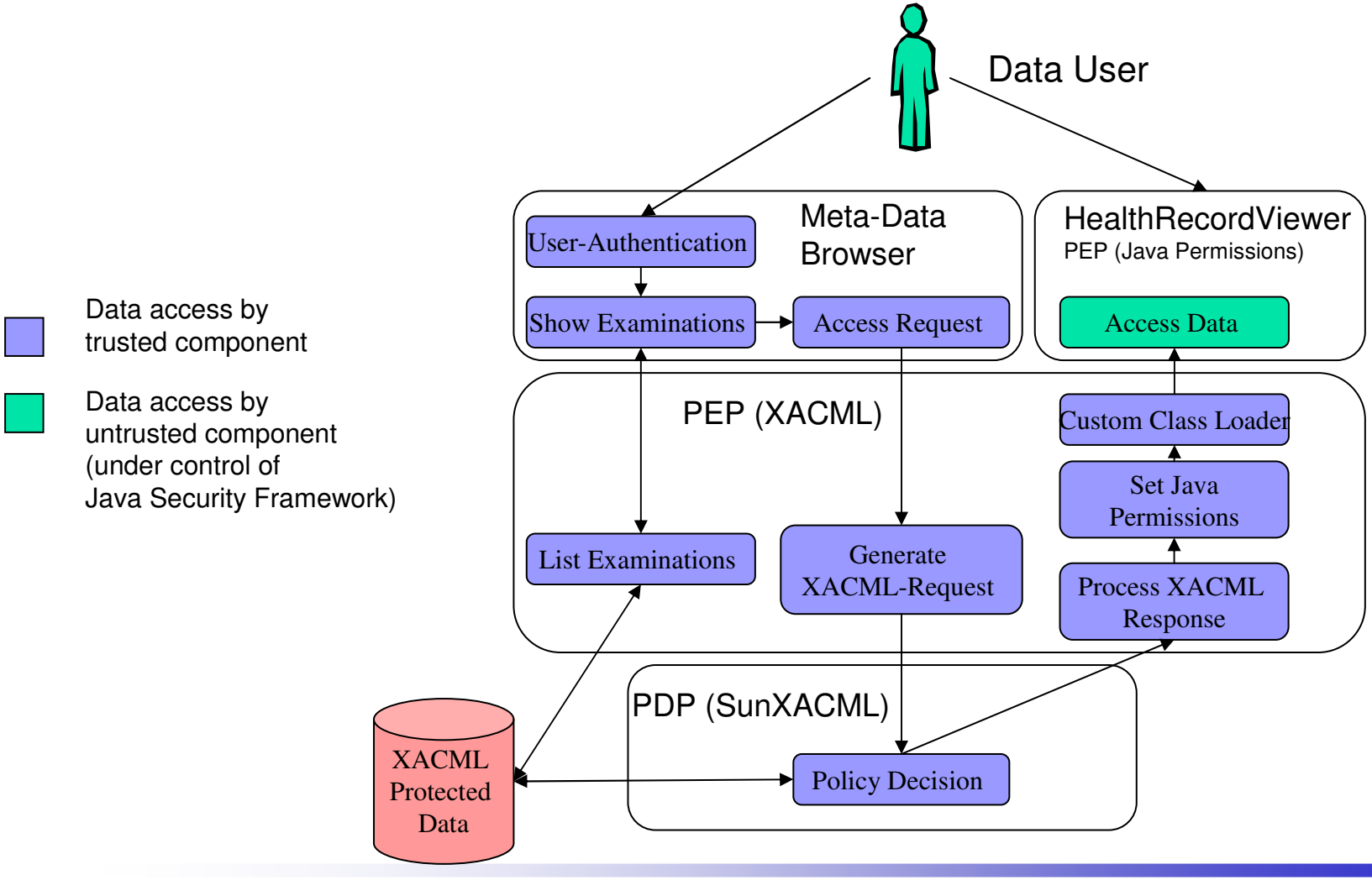
PEP Implementation

Enforcement process:

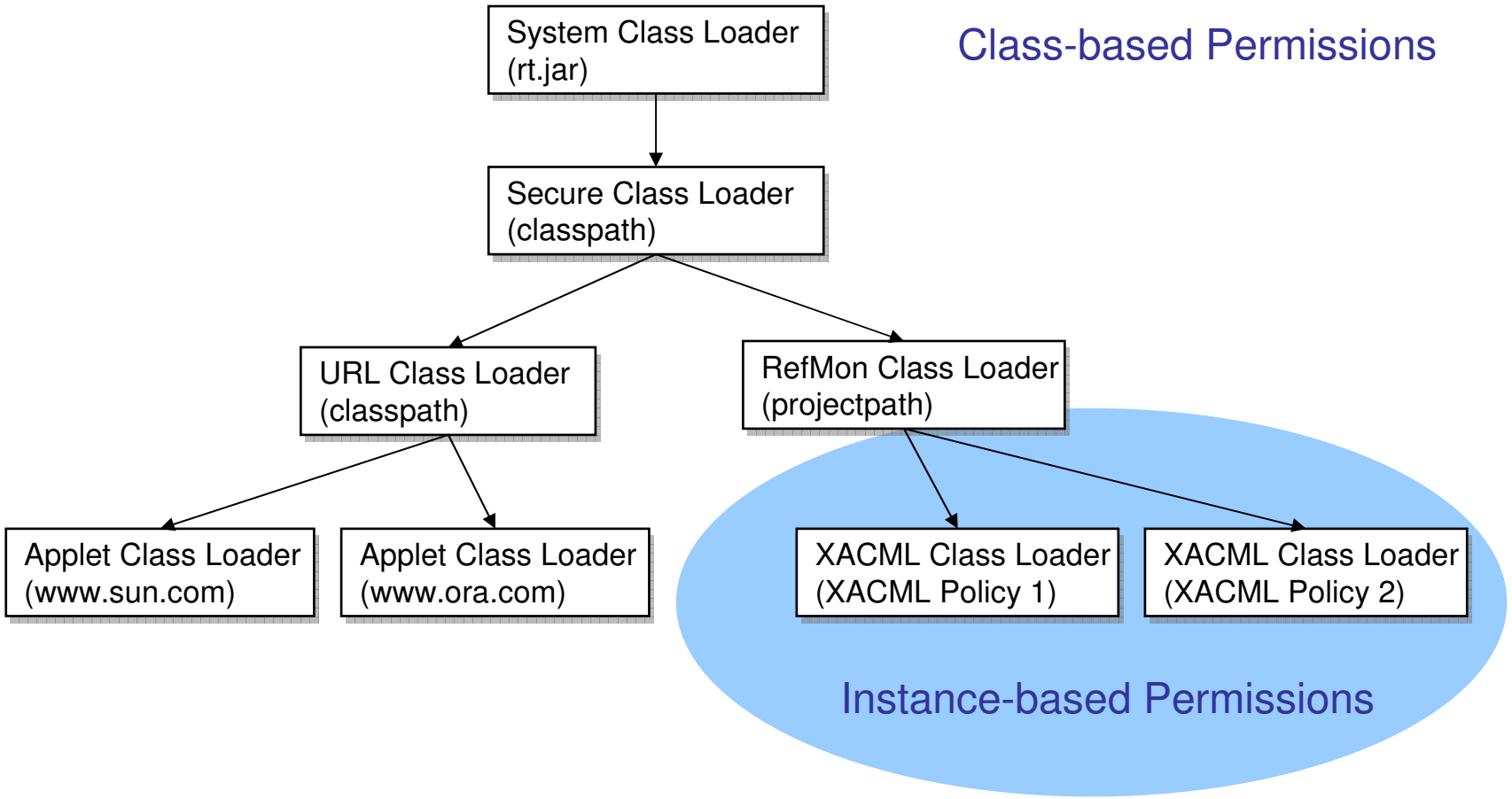
1. XACML Policies are translated into Java Permissions
2. A Custom Class Loader loads the application class with appropriate PermissionCollection
3. Permissions are monitored and enforced by the Java SecurityManager at application run time

Arbitrary applications can be started and data access can be controlled via this mechanism

Policy Decision Process



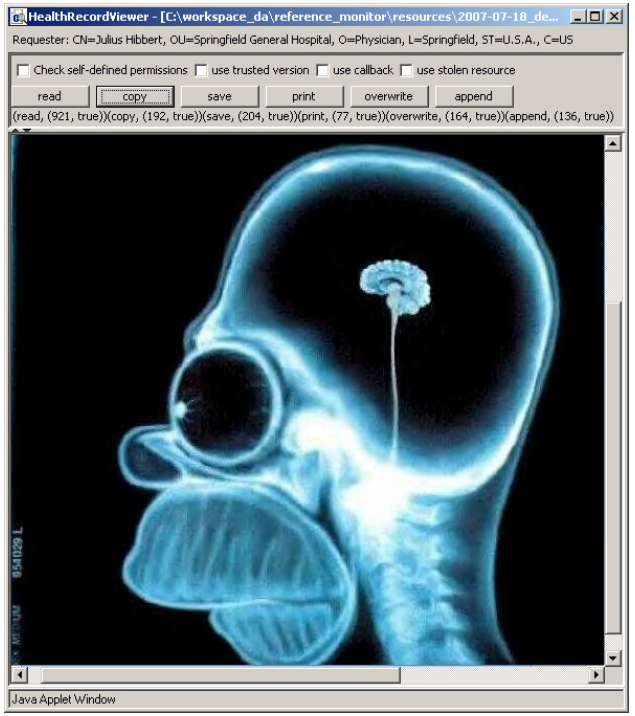
Java Class Loading Hierarchy



Usage control via the Java Security Framework

The following actions might be specified via the data policy and can be directly enforced using Java Permissions:

- **Read**: Accessing the data object
- **Copy**: Controlling access to the OS clipboard
- **Save**: Restricting general file system access prohibits storage of data copies outside the protected XML-Containers
- **Print**: Controlling access to the OS Print-Queue for launching of print jobs



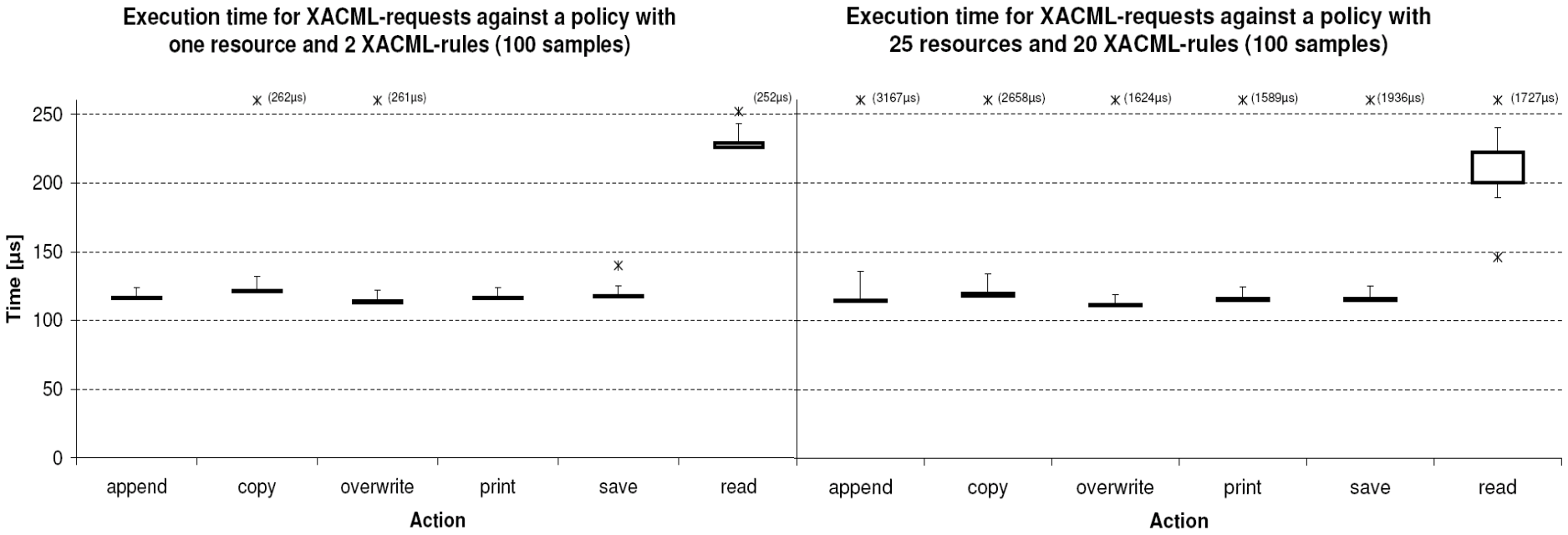
Mapping XACML -> JavaPermissions

- Actions that can be directly enforced in Java:

Action	Read	Copy	Save	Append	Print	Delete
AWT:accessClipboard		☑				
Runtime:queuePrintJob					☑	
FilePermission:read	☑					
FilePermission:write			☑	☑		
FilePermission:delete						☑

- Actions, that require extra cooperation from the Reference Monitor:
  - Append, Delete (selective addition or deletion of data in existing repository)
  - Timing and other environmental restrictions

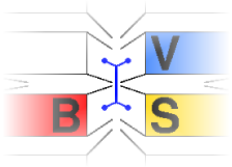
Measurements (XACML Policy Evaluation)



- Java 1.5.0_14, Pentium-M III, 1,73 GHz, 1Gb RAM

Conclusion

- Implementation of Owner Controlled Data Access policies possible
  - XACML policies are able to express data owner policies
- Enforcement requires trusted enforcement infrastructure
  - Java Security Framework can be adapted to automatically enforce certain usage restrictions without cooperation of the application
- Working prototype available



Thomas Scheffler
University of Potdam, Germany

scheffler@cs.uni-potsdam.de
www.cs.uni-potsdam.de