# A Framework for Managing and Analyzing Changes of Security Policies

Achim D. Brucker *Helmut Petritsch*

{achim.brucker, helmut.petritsch}@sap.com

SAP Research Karlsruhe
Germany

IEEE International Symposium on Policies for Distributed Systems and Networks
POLICY 2011
Pisa, Italy, 7th June 2011

# Motivation

Lots of regulations, e. g.,

- Financial market, Basel II (EU), SOX (US), PCI (credit cards)
- Health care, e. g., HIPAA in the US

Hard to enforce legal regulations

- Problem: translation from legal documents to policies

Audits needed to assess compliance

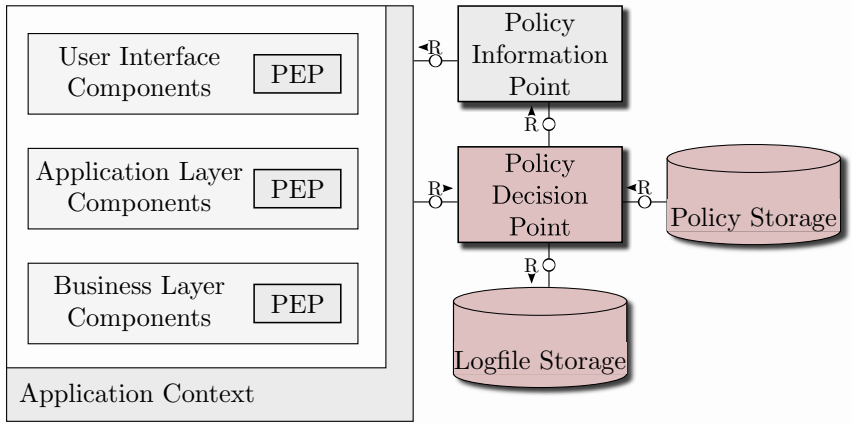
- Vast amount of log information
- Increasing costs for audits

General idea:

- Support for writing policies
- Support for auditing log traces

# Standard Architecture (Distributed Systems)

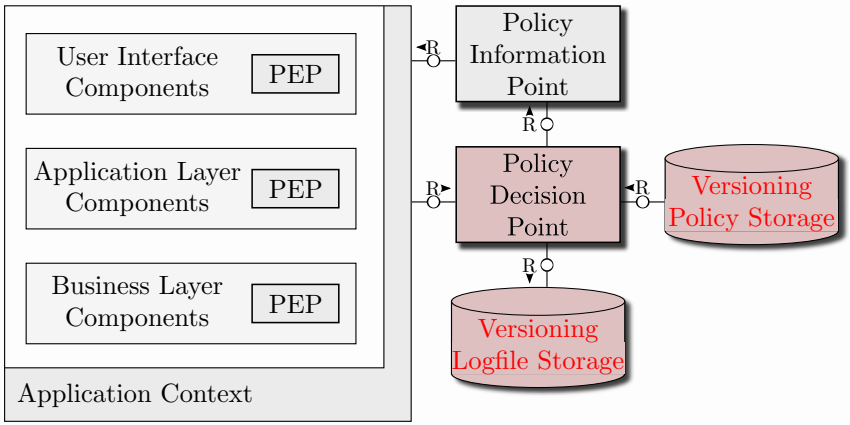
- Multiple PEPs accessing a central PDP
- Policy Information Point (PIP) resolves information from the application context
- Policies are loaded from a Policy Storage
- Access control requests and results are stored in a Logfile Storage



# Versioning

Based on XACML

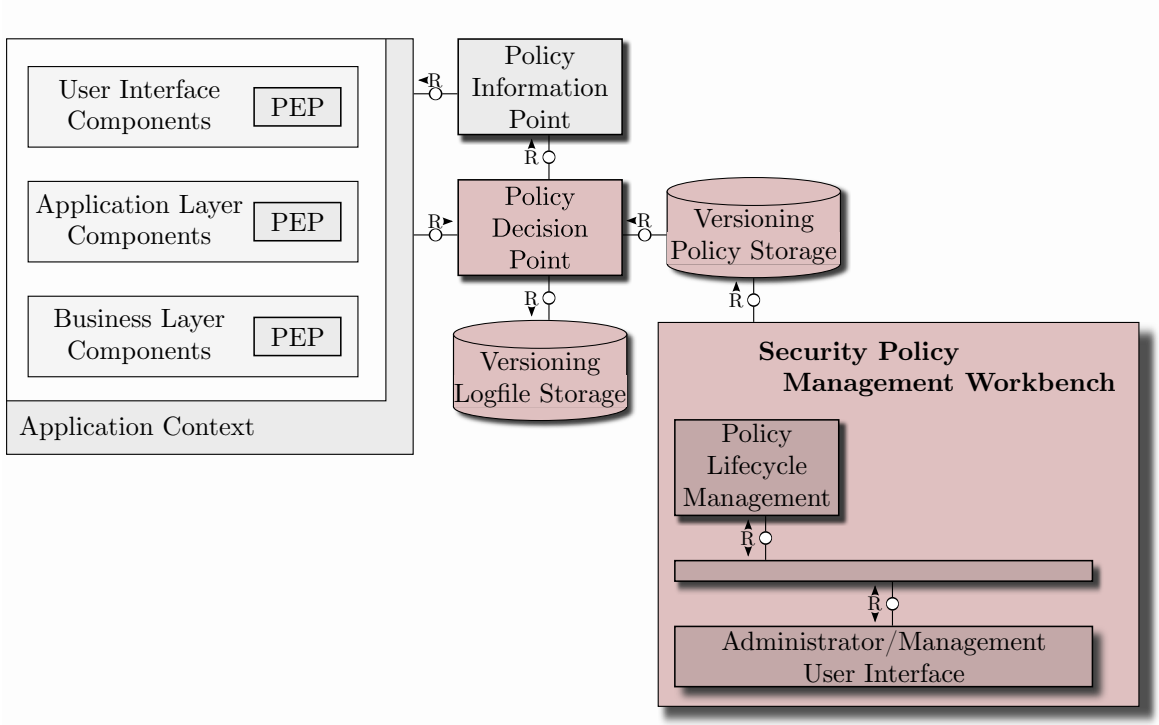
- Store policies in a Versioning Policy Storage
  (e. g., SVN for XACML policies)
- Save all PIP-resolved data in a Versioning Logfile Storage
  - XACML: resolved attributes
  - Save the current "state" of the system as seen by the PDP
- Interface for clients and PIP remains the same



# Administration

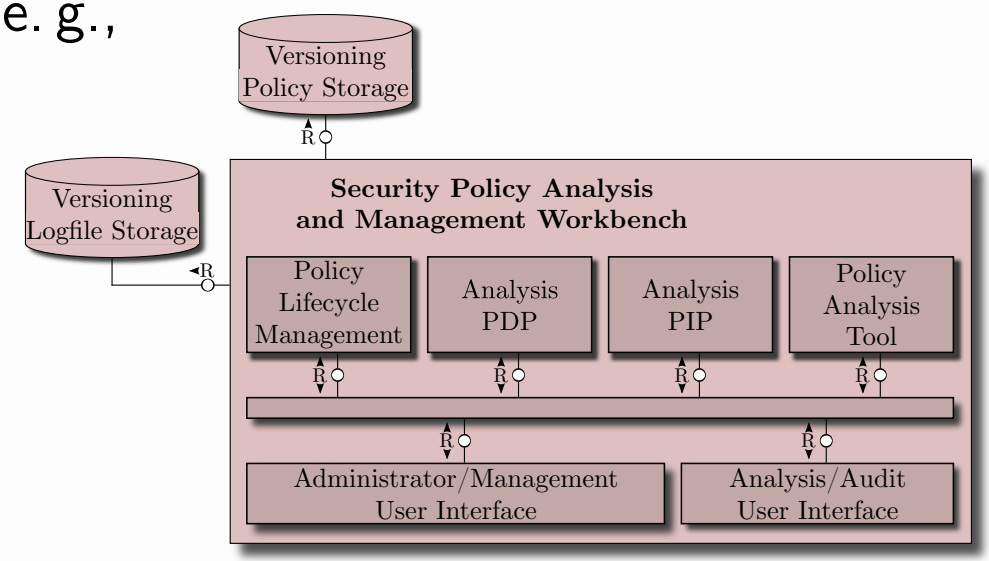
Administration and Management Interface

- Policy Administration Point (PAP)
- Adopt or provide a wrapper with versioning support for standard tools, e. g., XACML editor



# Analysis Workbench

- Analysis PDPs load any policy version from the policy store
- Analysis Policy Information Point (PIP) as context provider
  - Analysis PIP retrieves attributes from log store
  - Simulated runtime environment for analysis
- Replay (re-evaluate) recorded (or new) access control requests
- Analysis PDP enhancements allow for advanced analysis methods, e. g.,
  - Debugging of Policies
  - Abstract evaluation



# Replay Access Control Requests

To replay an access control request

- Select log entry from the log store
- Instantiate an Analysis PDP with a policy version
- Replay request on Analysis PDP
- Analysis PDP retrieves attributes as recorded for this request via Analysis PIP from the log store

Support for understanding policies changes, e. g.,

- Replaying incidents or suspicious requests with different policy versions
  - Does a change in the policy lead to a different result?

# Analysis PDP Enhancements

Enhancement of the Analysis PDP with analysis features

- Generation of "evaluation events" for every XACML element
- Injection of runtime information into XACML objects
  - Allows to, e. g., access the call stack (Java objects representing XACML elements) at runtime

Allows to

- Debug policies
- Provide runtime information about evaluation state to users and analysis tools

# Abstract Evaluation

- Evaluate policies with abstract attribute values instead of concrete values
- Results relying on abstract values have to be treated as abstract itself
- Reimplement (parts of) functions and combining algorithms: no lazy-evaluation based on abstract results
- Evaluate those parts of the policy which could be reached with any arbitrary configuration of the attribute

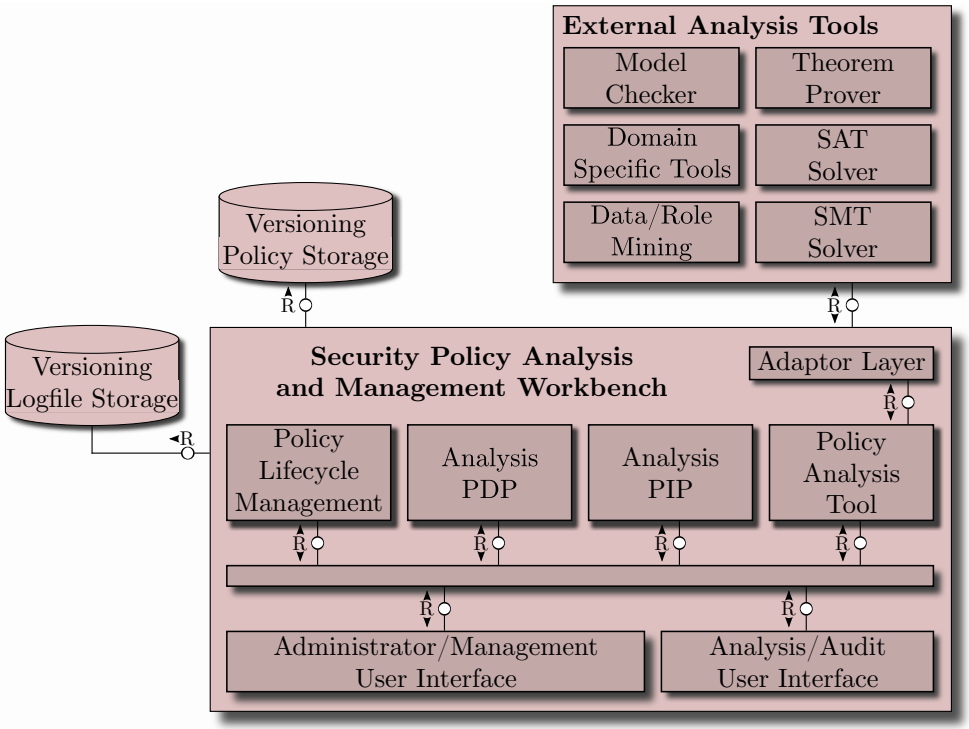
# Abstract Evaluation

Allows to obtain those parts of a policy, which are relevant for a specific request

- Use efficient evaluation to cut off non relevant parts
- Obtain "sub policy" which is relevant for specific request (and analysis question thereon)
- Complex analysis has only to deal with remaining sub policy

# External Analysis Tools

- Integrate existing and new developed tools
- Provide interfaces to access policy and log store
- Load and use Analysis PDPs
  - Define or modify the simulated runtime environment
  - Retrieve evaluation events from the Analysis PDP
  - Browse the evaluation state



# Policy Animation

During replay, attributes may be missing

- Policy version at runtime did not require attribute (and the attribute was therefore not recorded)
- Intentionally removed from the request
- Manually defined request, e. g., for testing

Resolution strategies:

- Ask user for value
- Policy Animation: computation of equivalence classes

# Policy Animation

What are suitable values for the attribute *current*

- For a nurse, when patient and subject department are known?

```
<!-- PolicySet: match HealthRecord -->
<PolicySet PolicyComb="first-applicable">
 <Target><Resource>HealthRecord</Resource></Target>
 <!-- Policy: rules for nurses -->
 <Policy RuleCombAlg="first-applicable">
  <Target><Role>Nurse</Role></Target>
  <!-- Deny in non-working-hours (i.e., at night) -->
  <Rule Id="1" Effect="Deny"><Target/>
   <Condition> 20:00 <= current <= 06:00<Condition>
  </Rule>
  <!-- permit read, if patient is on the same department -->
  <Rule Id="2" Effect="Permit">
   <Target><Action>read</Action></Target>
   <Condition> pat-dep == subj-dep</Condition>
 </Rule></Policy>
 <!-- Policy: rules for doctors: Permit during working hours -->
 <Policy RuleCombAlg="first-applicable">
  <Target><Role>Doctor</Role></Target>
  <Rule Id="3" Effect="Permit"><Target/>
   <Condition> 05:30 <= current <= 19:00<Condition>
  </Rule></Policy>
 <!-- final policy for HealthRecords -->
 <Policy><Target/>
  <Rule Id="final" Effect="Deny"/>
</Policy></PolicySet>
```

# Policy Animation

Observation: similar to input partitioning for test-case generation

- Use abstract evaluation to extract relevant sub policy
- Translate sub policy as test specification for HOL-TestGen

For a request, where the *current* attribute is missing:

```
firstApplicable {
  firstApplicable {
    timeInRange ($current, 20, 6) → deny
    true                          → permit
  }
  firstApplicable {
    true                          → deny
  }
}
```

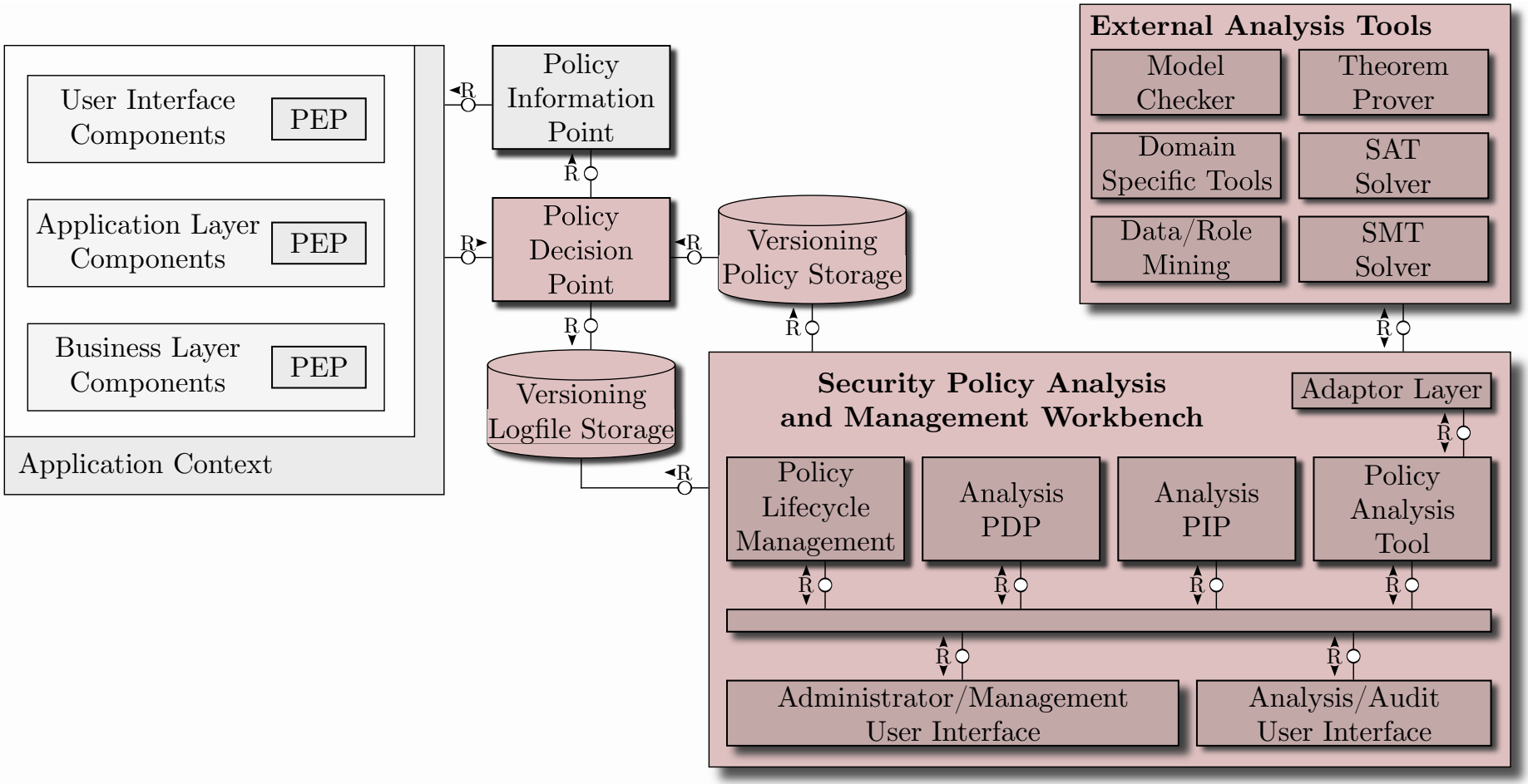
HOL-TestGen computes three equivalence classes

- $0 \leq \mathit{current} \leq 6$ results in a deny
- $6 < \mathit{current} < 20$ results in a permit
- $20 \leq \mathit{current} \leq 24$ results in a deny

# Conclusion

- Policies are considered to be static
  - Only a few concepts exist how to handle changes in policies
- Our Framework supports both policy author and auditor
  - Similar techniques to support both jobs
- Provide a platform to integrate (existing) analysis tools
  - Also tools from foreign domains can provide value

Thank you for your attention!

# Questions and Remarks?

# Full Architecture