An Automated Model-based Test Oracle for Access Control Systems

Antonia Bertolino[1], Said Daoudagh[1,2], Francesca Lonetti[1], Eda Marchetti[1]

[1]ISTI-CNR

[2]University of Pisa

Agenda

Introduction

- Access Control Systems
- XACML policies
- XACML testing

XACMET approach

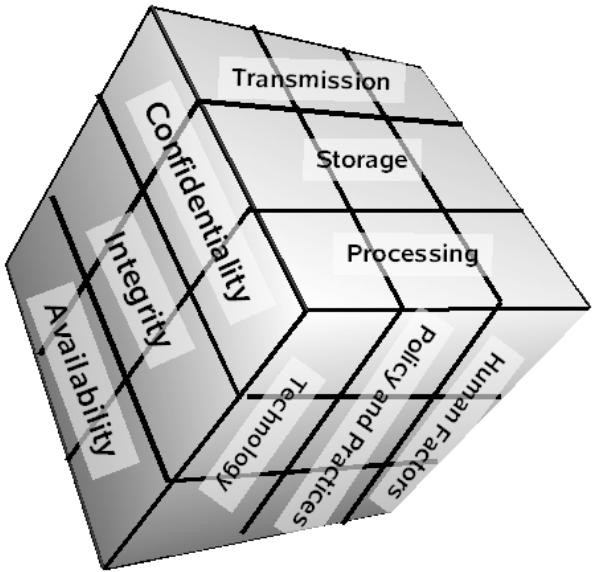
- XACML oracle
- XAC-tree, XAC-graph and XAC-paths

Empirical Evaluation

- Study 1: OASIS conformance test suite
- Study 2: comparison against multiple PDPs

Conclusions and Future Work

Introduction



Security is a primary concern in modern interconnected distributed software systems

It is made of the CIA Triad:

- Confidentiality
- Integrity
- Availability

Access control

For data and resources security, we need to ensure that only the intended subjects can access them and that these intended users are only given the level of access required to accomplish their tasks.

An access control system provides a decision to an authorization request, typically based on predefined policies

Defining security policies

A security policy states what is and what is not allowed

XACML Standard

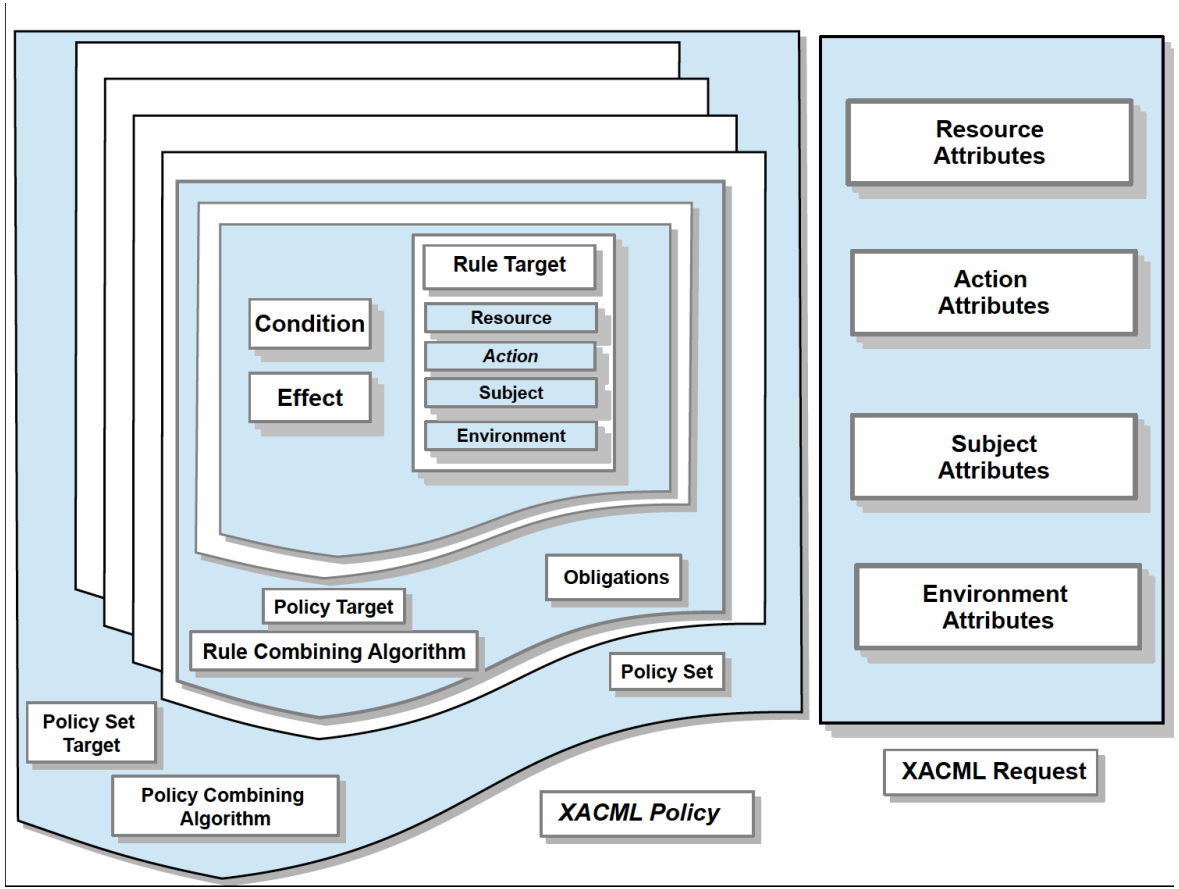
eXtensible Access Control Markup Language

OASIS
Advancing open standards for the information society

www.oasis-open.org

- XACML is the OASIS standard for specifying Access Control Policy
- It is a general-purpose language for access control policies. It provides an XML-based syntax for managing access to resources

XACML languages



XACML policy example

```
<Policy RuleCombiningAlgId="deny-overrides" PolicyId="policyExample">
<Target></Target>
 <Rule RuleId="rule1" Effect="Deny">
  <Target>
   <Resource>
      <AttributeValue >documentEntry</AttributeValue>
   </Resource>
   <Action>
      <AttributeValue >Write</AttributeValue>
   </ Action >
  </Target>
 </Rule>
 <Rule RuleId="rule2" Effect="Permit">
 <Target>
   <Subject>
      <AttributeValue >Julius</AttributeValue>
   </Resource>
   < Resource >
      <AttributeValue >book</AttributeValue>
   </ Resource >
  </Target>
 </Rule>
</Policy>
```

XACML policy example

```
<Policy RuleCombiningAlgId="deny-overrides" PolicyId="policyExample">
<Target></Target>
 <Rule RuleId="rule1" Effect="Deny">
  <Target>
   <Resource>
      <AttributeValue >documentEntry</AttributeValue>
   </Resource>
   <Action>
      <AttributeValue >Write</AttributeValue>
   </ Action >
  </Target>
 </Rule>
 <Rule RuleId="rule2" Effect="Permit">
 <Target>
   <Subject>
      <AttributeValue >Julius</AttributeValue>
   </Resource>
   < Resource >
      <AttributeValue >book</AttributeValue>
   </ Resource >
  </Target>
 </Rule>
</Policy>
```

Rule1

XACML policy example

```
<Policy RuleCombiningAlgId="deny-overrides" PolicyId="policyExample">
<Target></Target>
 <Rule RuleId="rule1" Effect="Deny">
  <Target>
   <Resource>
      <AttributeValue >documentEntry</AttributeValue>
   </Resource>
   <Action>
      <AttributeValue >Write</AttributeValue>
   </ Action >
  </Target>
</Rule>
<Rule RuleId="rule2" Effect="Permit">
<Target>
   <Subject>
      <AttributeValue >Julius</AttributeValue>
   </Resource>
   < Resource >
      <AttributeValue >book</AttributeValue>
   </ Resource >
  </Target>
</Rule>
</Policy>
```

Rule1

Rule2

XACML policy example

```
<Policy RuleCombiningAlgId="deny-overrides" PolicyId="policyExample">
<Target></Target>
 <Rule RuleId="rule1" Effect="Deny">
  <Target>
   <Resource>
      <AttributeValue >documentEntry</AttributeValue>
   </Resource>
   <Action>
      <AttributeValue >Write</AttributeValue>
   </ Action >
  </Target>
 </Rule>
 <Rule RuleId="rule2" Effect="Permit">
  <Target>
   <Subject>
      <AttributeValue >Julius</AttributeValue>
   </Resource>
   < Resource >
      <AttributeValue >book</AttributeValue>
   </ Resource >
  </Target>
 </Rule>
</Policy>
```

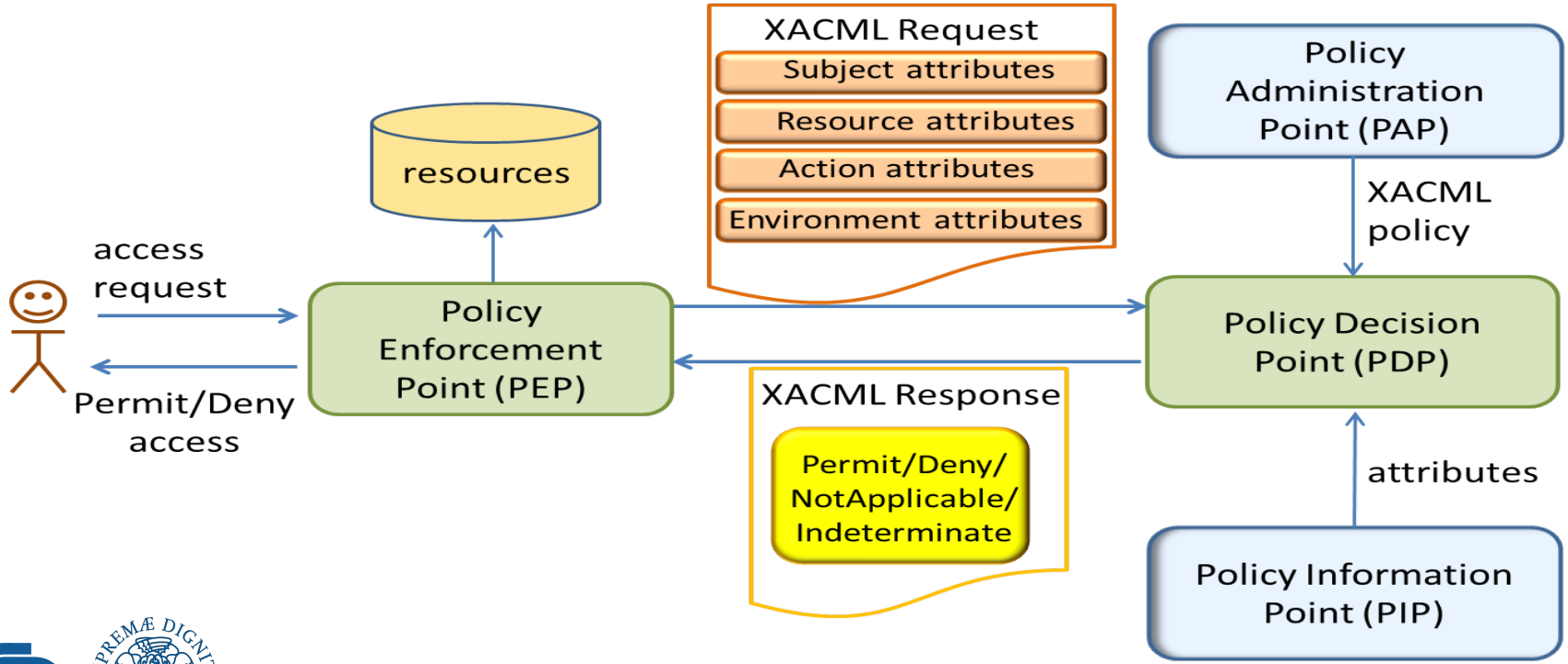
Rule1

Target

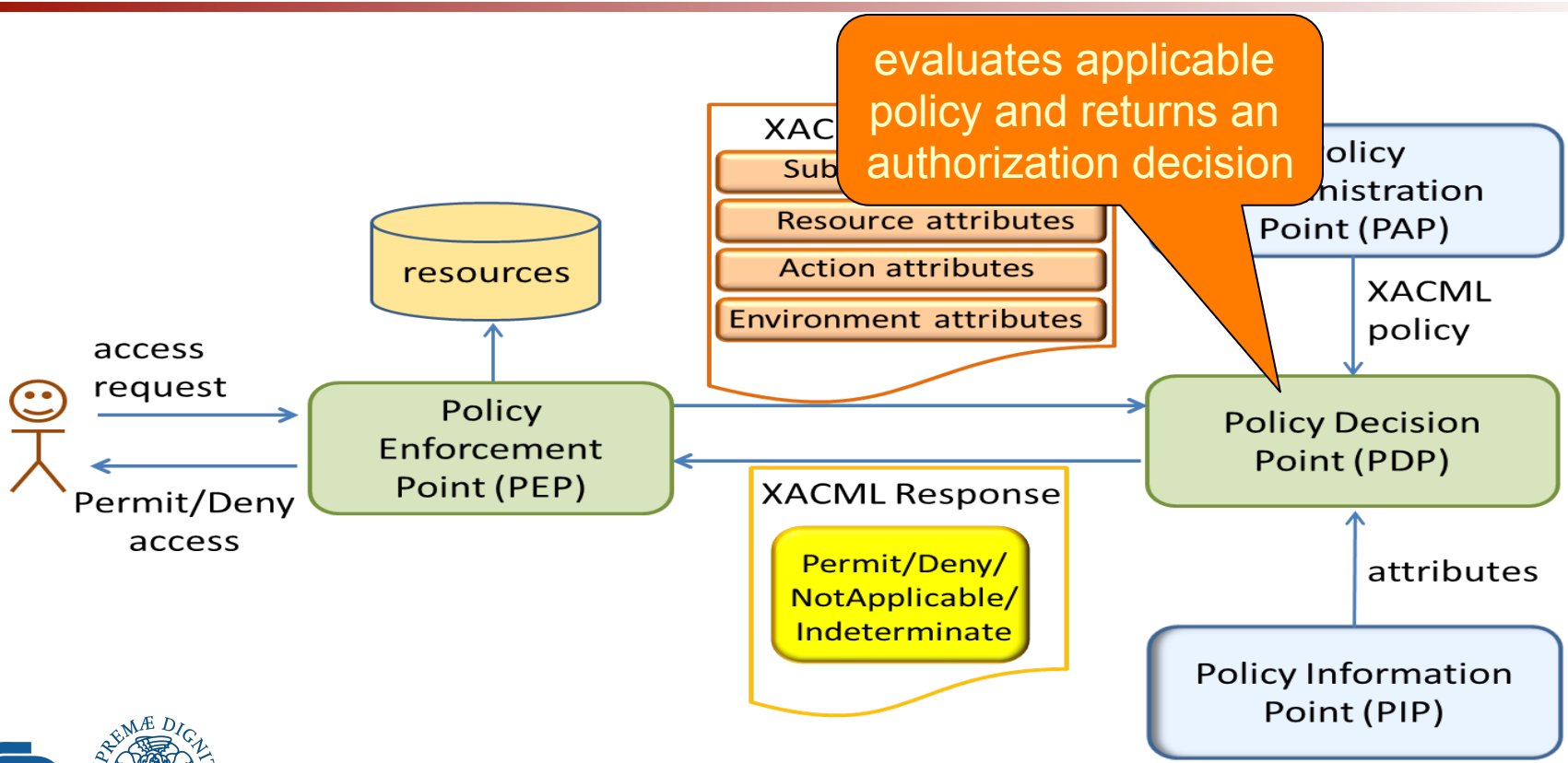
Rule2

Target

XACML architecture



XACML architecture



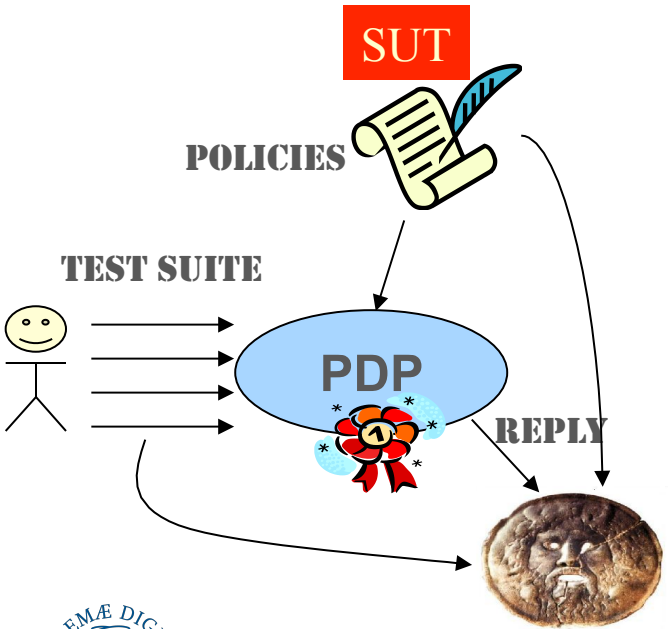
How do we validate the access control system?

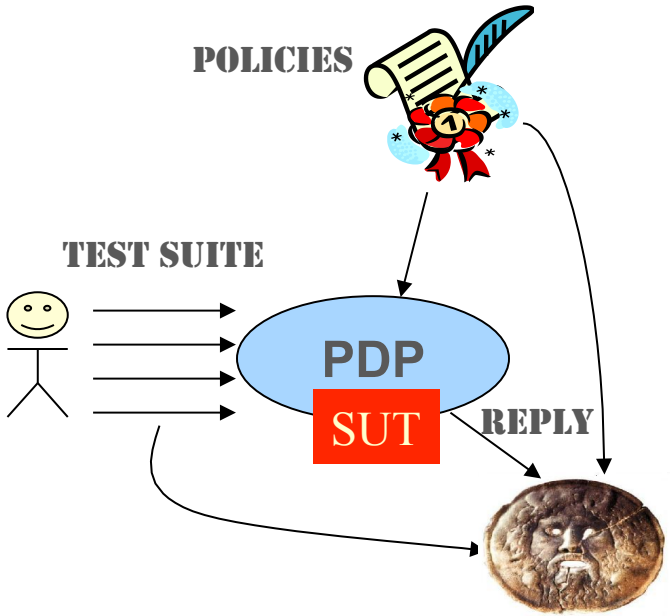
XACML properties of interoperability, extensibility, distribution are paid in terms of complexity and verbosity

Policies can be deceiving and need to be carefully tested

Two testing purposes

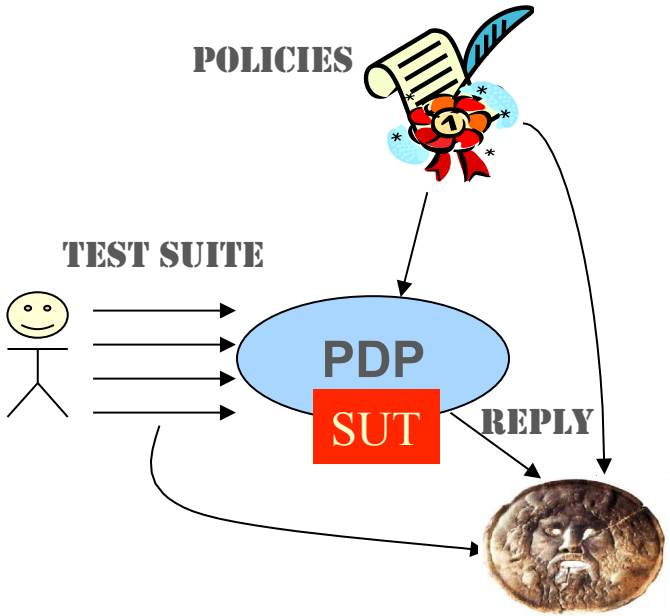
Testing the policies ← vs. → Testing the PDP





Two testing purposes

Testing the PDP



Motivation

Several proposals for automating PDP testing, including:

- Mutation;
- Coverage;
- Random;
- Combinatorial;
- Model-based techniques.

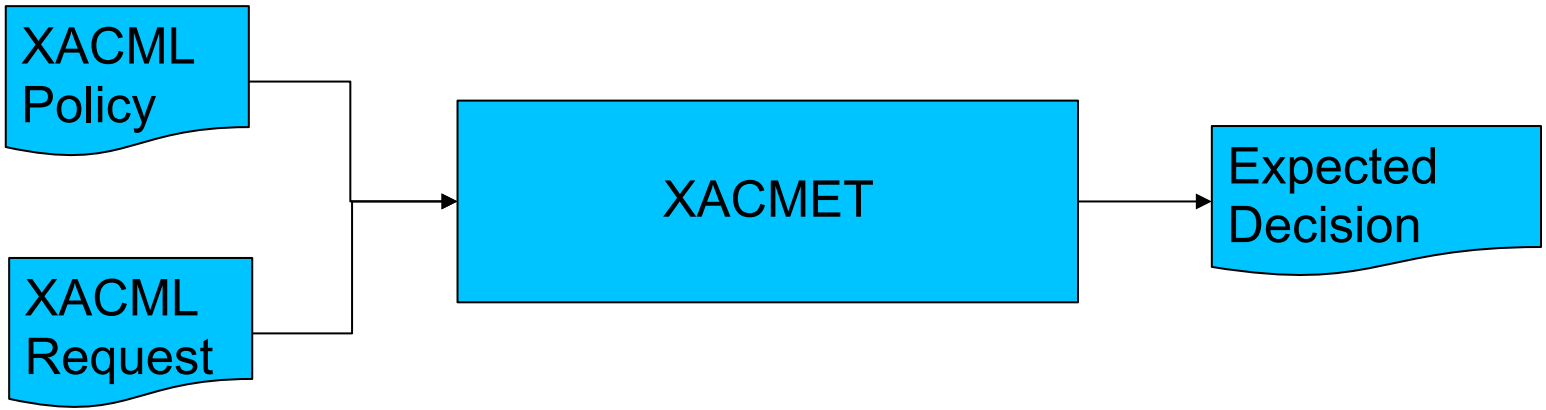
They all share an important drawback: **the lack of the oracle**

- i.e., for the generated requests the expected PDP decision is not provided;
- an important limitation, especially when test suites are large and manual inspection of results is unfeasible.

XACML oracle

Given a generic request, the result of the evaluation of an XACML policy with that request depends on:

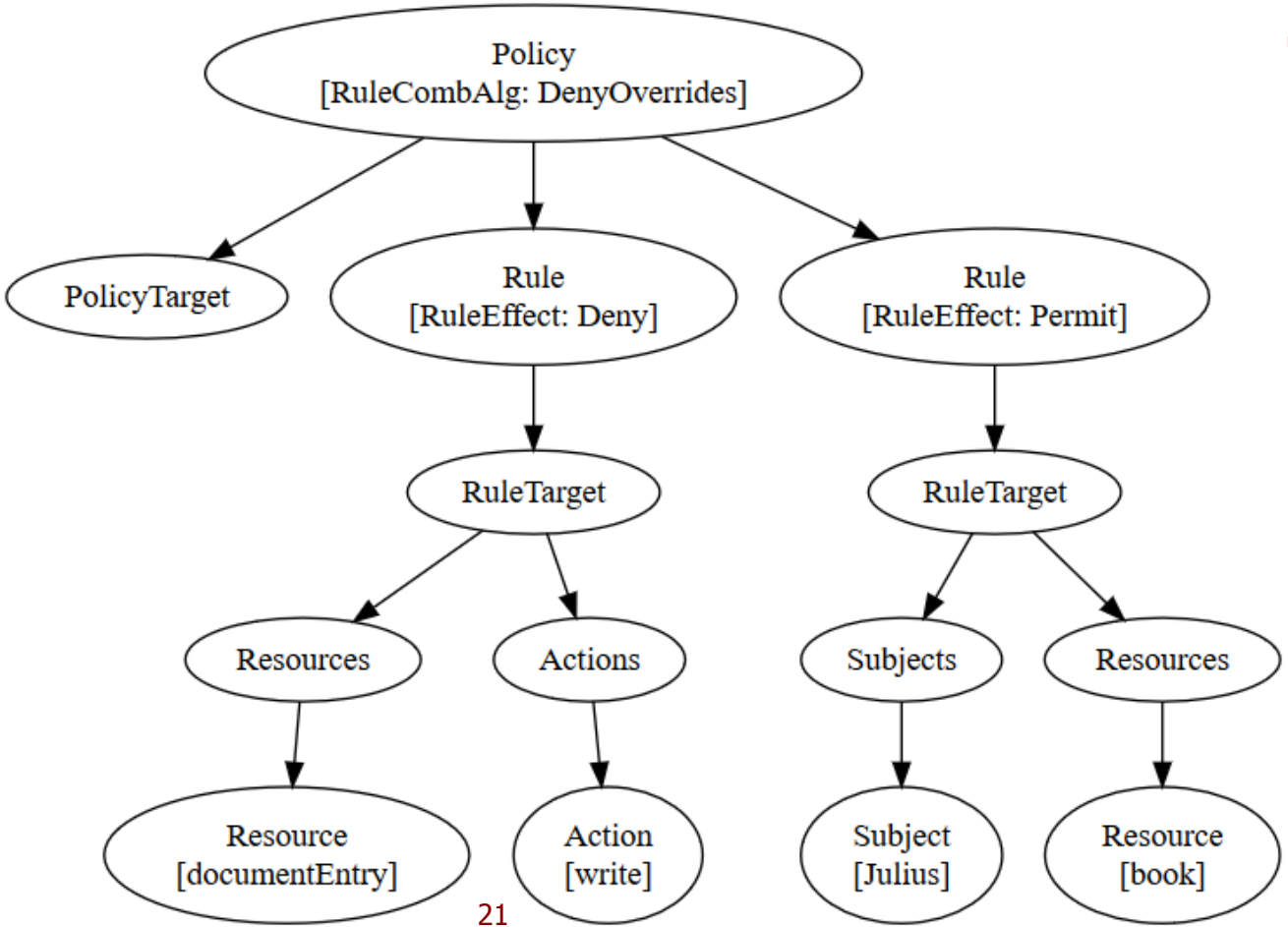
- the request values;
- the policy constraints;
- as well as the combining algorithm that prioritizes the evaluation of the policy rules.


XACML Policy
XACML Request
XACMET
Expected Decision

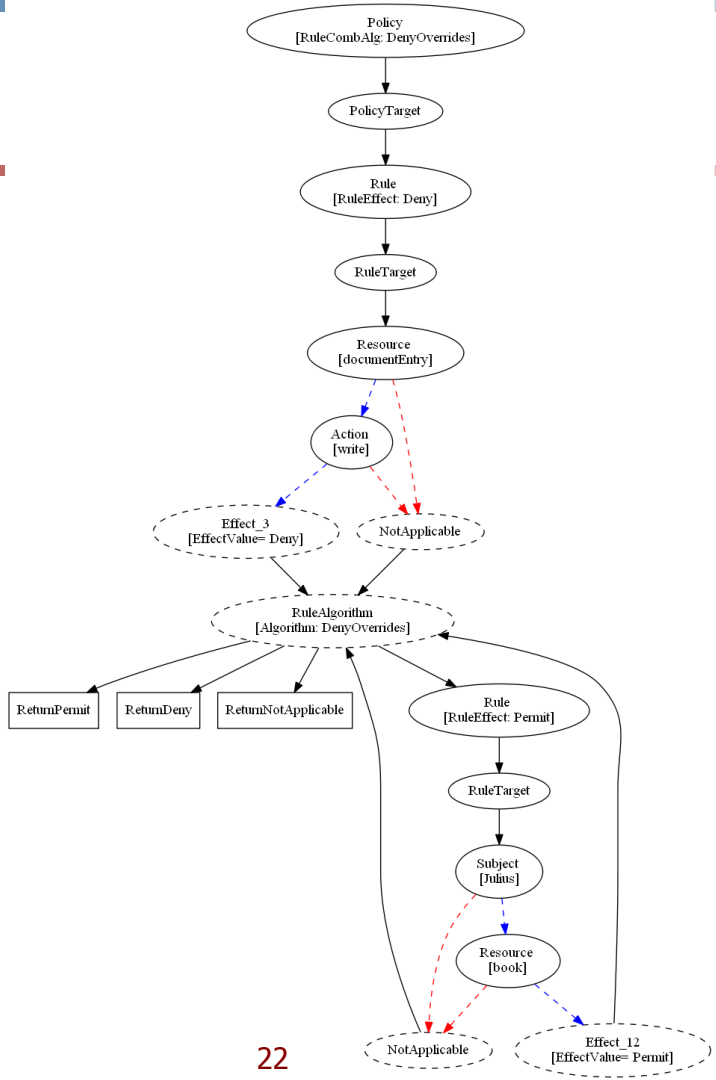
XACMET oracle derivation

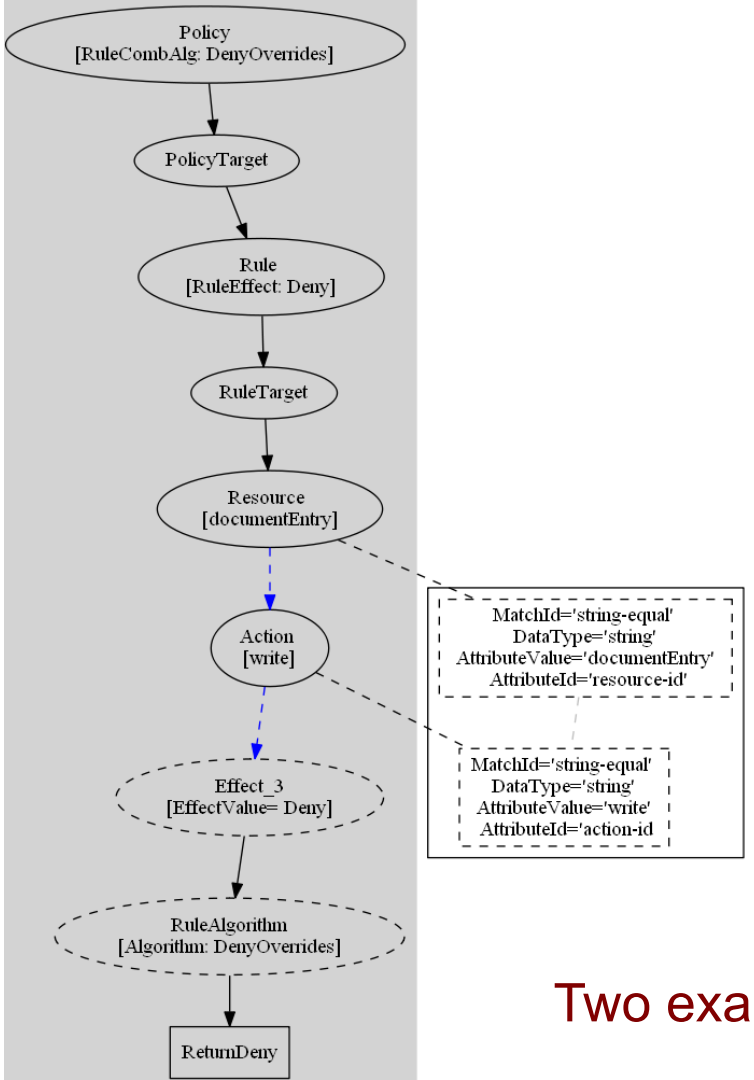
1. The XACML policy is represented as a XAC-Tree
2. The XAC-Tree is transformed into a XAC-Graph
3. The paths over the XAC-Graph are derived
4. For each path, a verdict (the oracle) is obtained

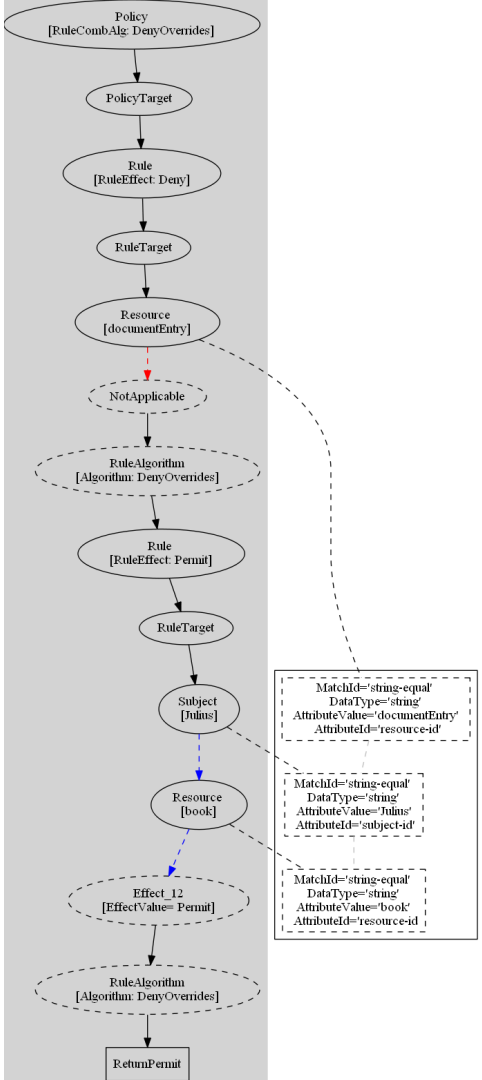
XAC-Tree example



XAC-Graph


Policy
[RuleCombAlg: DenyOverrides]
PolicyTarget
Rule
[RuleEffect: Deny]
RuleTarget
Resource
[documentEntry]
Action
[write]
Effect_3
[EffectValue= Deny]
NotApplicable
RuleAlgorithm
[Algorithm: DenyOverrides]
ReturnPermit
ReturnDeny
ReturnNotApplicable
Rule
[RuleEffect: Permit]
RuleTarget
Subject
[Julius]
Resource
[book]
NotApplicable
Effect_12
[EffectValue= Permit]





Two examples of XAC-Paths

Evaluation

We conducted two studies:

1. Compliance with XACML conformance test suite
2. Comparison against one BB existing approach:

> Nuo Li, JeeHyun Hwang, and Tao Xie. 2008. Multiple-implementation testing for XACML implementations. TAV-WEB '08

Study 1

- For each test case, we derived XAC-Graph associated to the XACML policy and an ordered set of paths.
- Then, we evaluated the XACML request against the obtained set of paths, we identified the first covered path and derived the verdict associated to that path.
- Finally, we compared this verdict with the decision value specified in the response belonging to the test case.

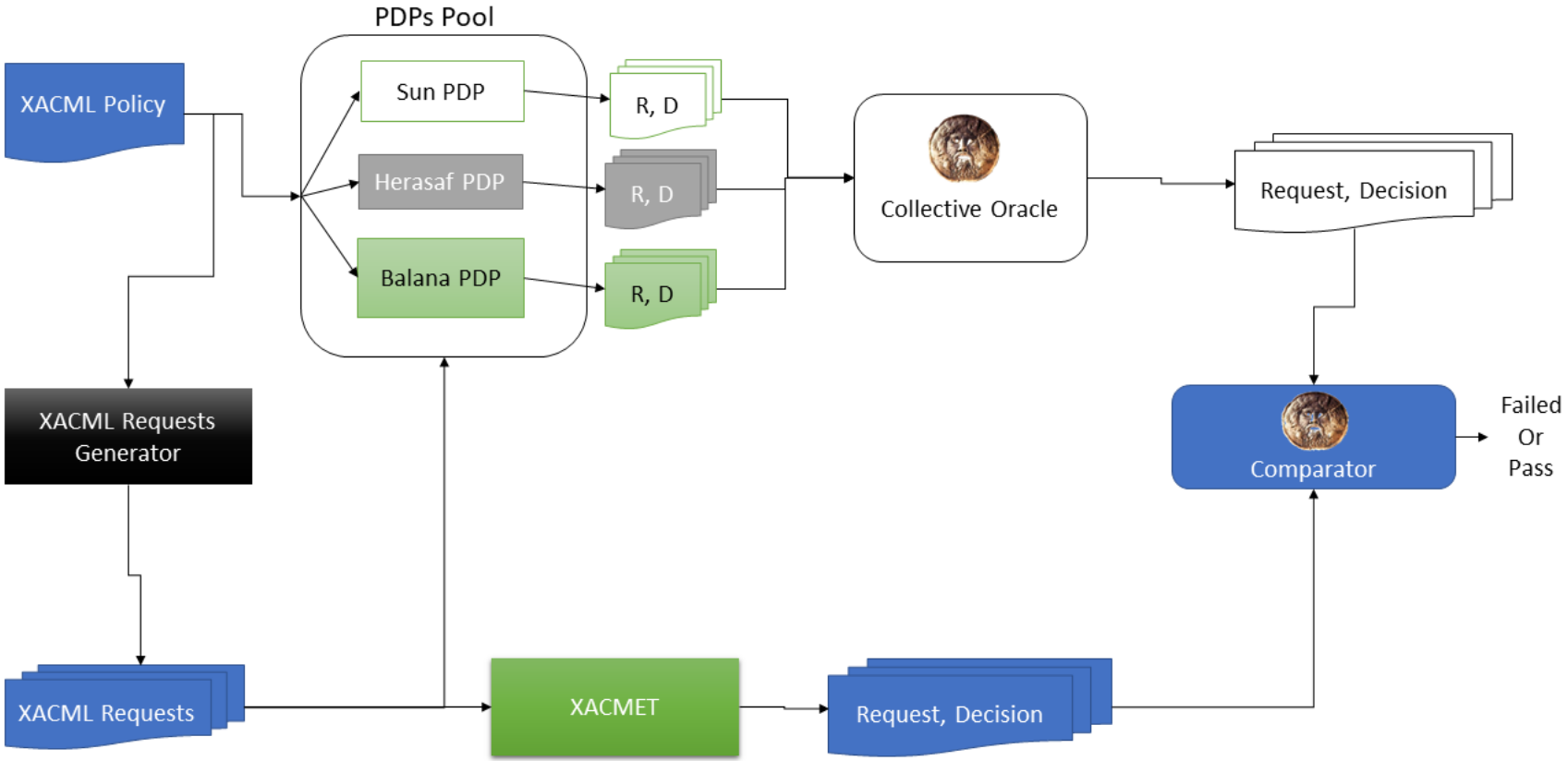
Study 1

XACML Policy	Functionality							XACML Request
	#Policy	#Rule	#Cond	#Sub	#Res	#Act	#Funct	
Conformance Test Suite XACML Policies								
II A (90 %)	18	18	12	18	8	16	112	18
II B (100 %)	53	53	6	51	50	98	410	53
II C (10 %)	22	22	22	18	3	1	102	22
II D (17 %)	5	13	7	13	-	-	60	5

A **Conformance Test Case** consists of **three elements**:
XACML **policy**, XACML **request**, and XACML **response**
We focused on the subset of tests implementing the mandatory functionalities

For all tests, the XACMET verdict coincided with the expected access decision.

Study 2


PDPs Pool
XACML Policy
Sun PDP
Herasaf PDP
Balana PDP
R, D
R, D
R, D
Collective Oracle
Request, Decision
XACML Requests Generator
Comparator
Failed
Or
Pass
XACML Requests
XACMET
Request, Decision

Study 2

XACML Policy	Functionality							XACML Request
	#Policy	#Rule	#Cond	#Sub	#Res	#Act	#Funct	
Real world XACML Policies								
2_73020419964_2	1	6	5	3	3	0	4	8
create-document	1	3	2	1	2	1	3	5
demo-5	1	3	2	2	3	2	4	13
demo-11	1	3	2	2	3	1	5	8
demo-26	1	2	1	1	3	1	4	16
read-document	1	4	3	2	4	1	3	6
read-informationunit	1	2	1	0	2	1	2	4
read-patient	1	4	3	2	4	1	3	6
Xacml-Nottingham-1	1	3	0	24	3	3	2	18

For all requests the XACMET oracle verdict coincided with the one from the multiple PDPs

Conclusions

- We have introduced a novel model-based approach to automatic generation of XACML oracle for testing policy evaluation engines.
- The XACMET approach fully automatically derives a verdict for each XACML request by considering the expected behavior of the PDP.
- Experimental results so far evidence the effectiveness of our proposal with respect to the oracle provided in the XACML conformance tests.

Future Work

- We plan to extend our automated oracle in order to consider more functionalities of the XACML conformance policies
- The XACMET approach is being extended to be compliant with the latest version of the XACML standard
- The XACMET approach can also be used for (not shown here):
  - Automatically generating a test suite
  - Measuring the coverage over the XAC-Graph
- Future work will also include further experimentation of XACMET, and its comparison with other model-based approaches.

Thank you for your attention!

Antonia Bertolino, Said Daoudagh, Francesca Lonetti, Eda Marchetti: ***An Automated Model-based Test Oracle for Access Control Systems***. AST@ICSE, Gothenburg, Sweden. May 28-29, 2018.

For XACMET details please contact: said.daoudagh@di.unipi.it