DEVELOPMENT OF A NEW POLICY EVALUATION PROCEDURE FOR XACML

Jorian van Oostenbrugge
Supervisor: Fatih Turkmen

August 19, 2016

System and Network Engineering
University of Amsterdam

WHY

- Customer data more and more valuable
- Data stored in cloud
- Access control becomes critical

XACML

- eXtensible Access Control Markup Language

- XML-based language

- Also an architecture

- OASIS standard for the expression of security policies

XACML ELEMENTS

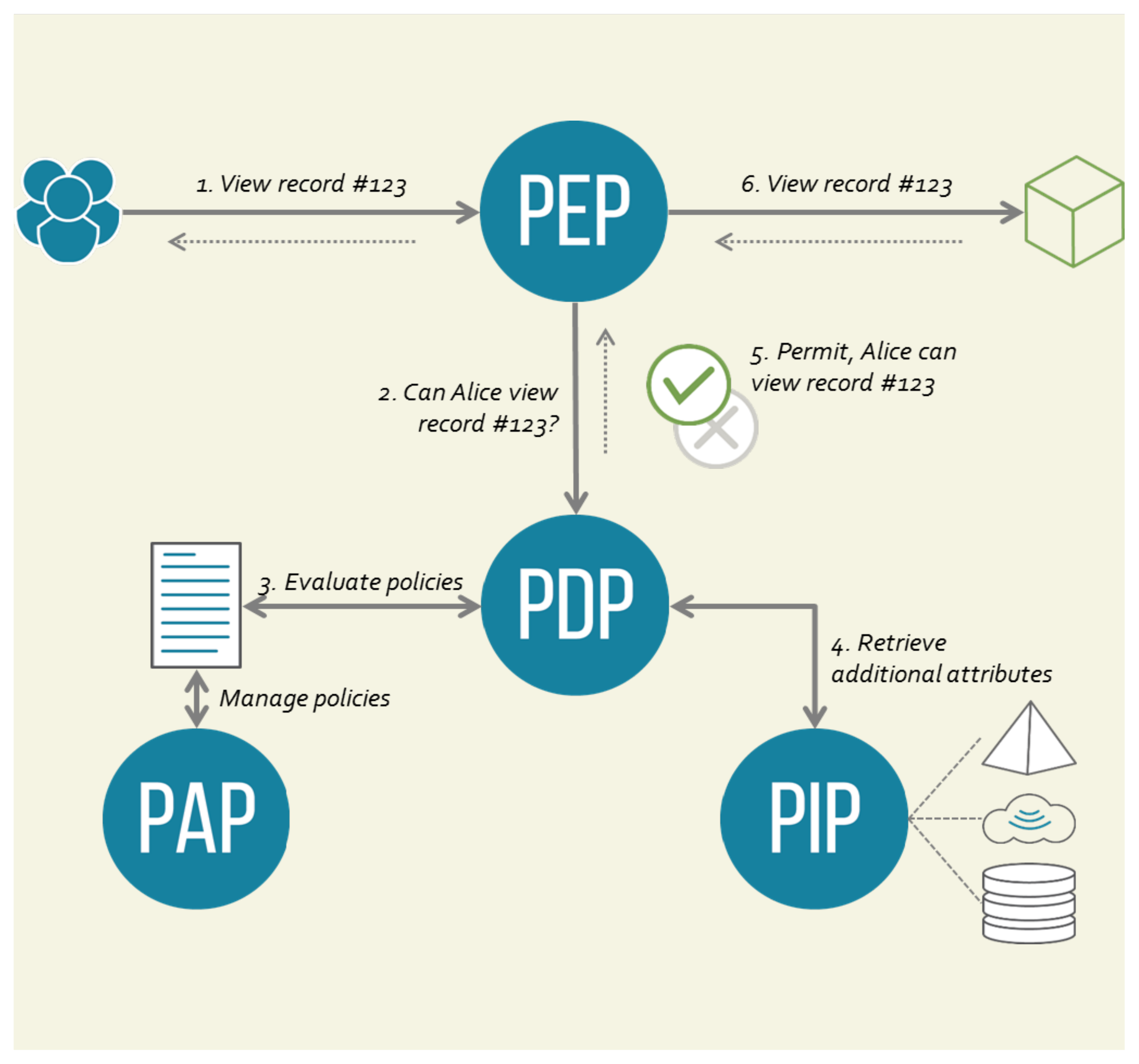
```
<PolicySet>
  <Policy RuleCombiningAlg="..." >
     <Target/>
     <Rule RuleId="..." Effect="Permit">
       <Target/>
       <Condition/>
     </Rule>
     <Rule RuleId="..." Effect="Deny">
       <Target/>
       <Condition/>
     </Rule>
  </Policy>
  <Policy RuleCombiningAlgId="...">
     ...
  </Policy>
<PolicySet>
```

EXAMPLE XACML POLICY

```
<Policy
    RuleCombiningAlgId="identifier:rule-combining-algorithm:permit-overrides">
    <Target/>
    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:SimpleRule1" Effect="Permit">
        <Target>
            <AnyOf>
                <AllOf>
                    <Match MatchId="string-equal">
                        <AttributeValue DataType="string">admin</AttributeValue>
                        <AttributeDesignator AttributeId="role" DataType="string"/>
                    </Match>
                </AllOf>
            </AnyOf>
        </Target>
        <Condition>
            ...
        </Condition>
    </Rule>
    <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:SimpleRule2" Effect="Deny">
        ...
    </Rule>
</Policy>
```

XACML IN ACTION

1. Request intercepted by PEP
2. Request converted to XACML
3. PDP evaluates policy
4. If needed retrieve additional attributes
5. PDP reaches decision and forwards this to PEP
6. Request arrives at resource


1. View record #123
PEP
6. View record #123
2. Can Alice view record #123?
5. Permit, Alice can view record #123
3. Evaluate policies
PDP
4. Retrieve additional attributes
Manage policies
PAP
PIP

Source: Wikipedia

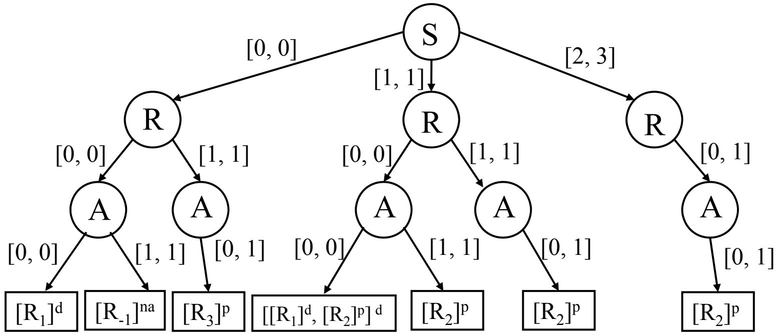
RELATED RESEARCH

(Adaptive) reordering

- Based on statistics and categorization

Decision Diagrams

- XEngine
- Matching Tree (MT) and Combining Tree (CT)
- SNE-XACML with MIDD



Source: XEngine: A Fast and Scalable XACML Policy Evaluation Engine

RESEARCH QUESTION

- Propositional encoding
- PDP

SAT & CNF

- Boolean function: $f(x_1, x_2, \ldots, x_n)$
- Variables, operators and parentheses: $x_1, \wedge, \vee, \neg, ()$
- SAT solvers
- CNF: $(p_1 \vee p_2) \wedge (p_3 \vee p_4) \wedge (p_5 \vee p_6)$
  - Conjunction of clauses
  - Disjunction of literals

ALGORITHM

- Constructing attribute domains
- Policy flattening
- SAT encoding

CONSTRUCTING ATTRIBUTE DOMAINS (1)

- Attributes
  - *AttributeValue*
  - *AttributeDesignator*
  - *AttributeSelector*

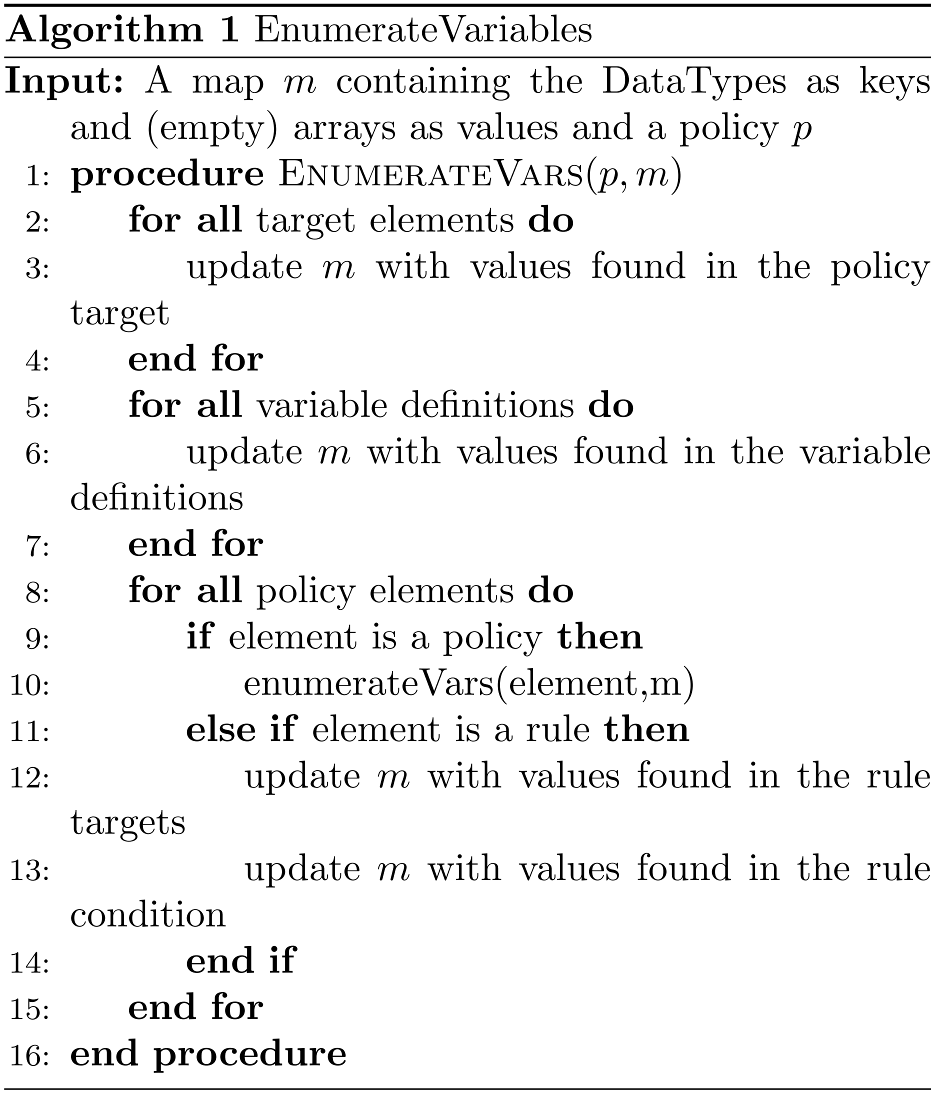
```
<rule Effect=”Permit”>
  ...
  <AttributeValue DataType=”String”>admin</AttributeValue>
  <AttributeDesignator AttributeId=”role” DataType=”String”/>
  ...
</rule>
```

CONSTRUCTING ATTRIBUTE DOMAINS (2)

$$\mathrm{D}_{role} \in \{\mathrm{admin}, \mathrm{manager}, \mathrm{hr}, \mathrm{user}\}$$

$$\mathrm{admin} \in \{\mathrm{admin}, \mathrm{manager}, \mathrm{hr}, \mathrm{user}\}$$

CONSTRUCTING ATTRIBUTE DOMAINS (3)



Algorithm 1 EnumerateVariables

Input: A map m containing the DataTypes as keys and (empty) arrays as values and a policy p

```
procedure EnumerateVars($p, m$)
    for all target elements do
        update $m$ with values found in the policy target
    end for
    for all variable definitions do
        update $m$ with values found in the variable definitions
    end for
    for all policy elements do
        if element is a policy then
            enumerateVars(element,m)
        else if element is a rule then
            update $m$ with values found in the rule targets
            update $m$ with values found in the rule condition
        end if
    end for
end procedure
```

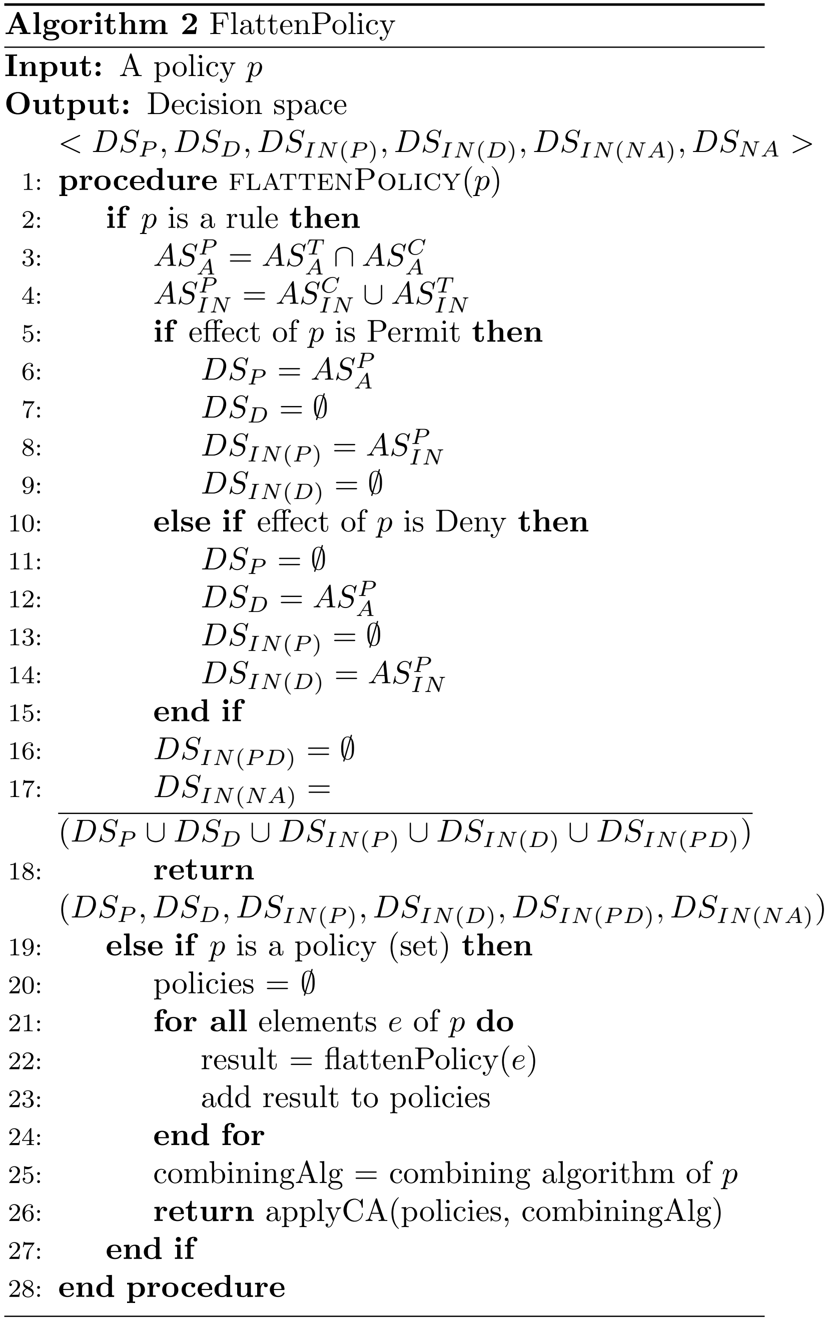
ALGORITHM

- Constructing attribute domains
- Policy flattening
- SAT encoding

Policy Flattening (1)

- Applicability space <AS_A, AS_{IN}, AS_{NA}>

- Decision space <DS_P, DS_D, DS_{IN}, DS_{NA}>



Algorithm 2 FlattenPolicy

Input: A policy p

Output: Decision space $< DS_P, DS_D, DS_{IN(P)}, DS_{IN(D)}, DS_{IN(NA)}, DS_{NA} >$

```
procedure FLATTENPOLICY($p$)
    if $p$ is a rule then
        $AS_A^P = AS_A^T \cap AS_A^C$
        $AS_{IN}^P = AS_{IN}^C \cup AS_{IN}^T$
        if effect of $p$ is Permit then
            $DS_P = AS_A^P$
            $DS_D = \emptyset$
            $DS_{IN(P)} = AS_{IN}^P$
            $DS_{IN(D)} = \emptyset$
        else if effect of $p$ is Deny then
            $DS_P = \emptyset$
            $DS_D = AS_A^P$
            $DS_{IN(P)} = \emptyset$
            $DS_{IN(D)} = AS_{IN}^P$
        end if
        $DS_{IN(PD)} = \emptyset$
        $DS_{IN(NA)} = \overline{(DS_P \cup DS_D \cup DS_{IN(P)} \cup DS_{IN(D)} \cup DS_{IN(PD)})}$
        return $(DS_P, DS_D, DS_{IN(P)}, DS_{IN(D)}, DS_{IN(PD)}, DS_{IN(NA)})$
    else if $p$ is a policy (set) then
        policies = $\emptyset$
        for all elements $e$ of $p$ do
            result = flattenPolicy($e$)
            add result to policies
        end for
        combiningAlg = combining algorithm of $p$
        return applyCA(policies, combiningAlg)
    end if
end procedure
```

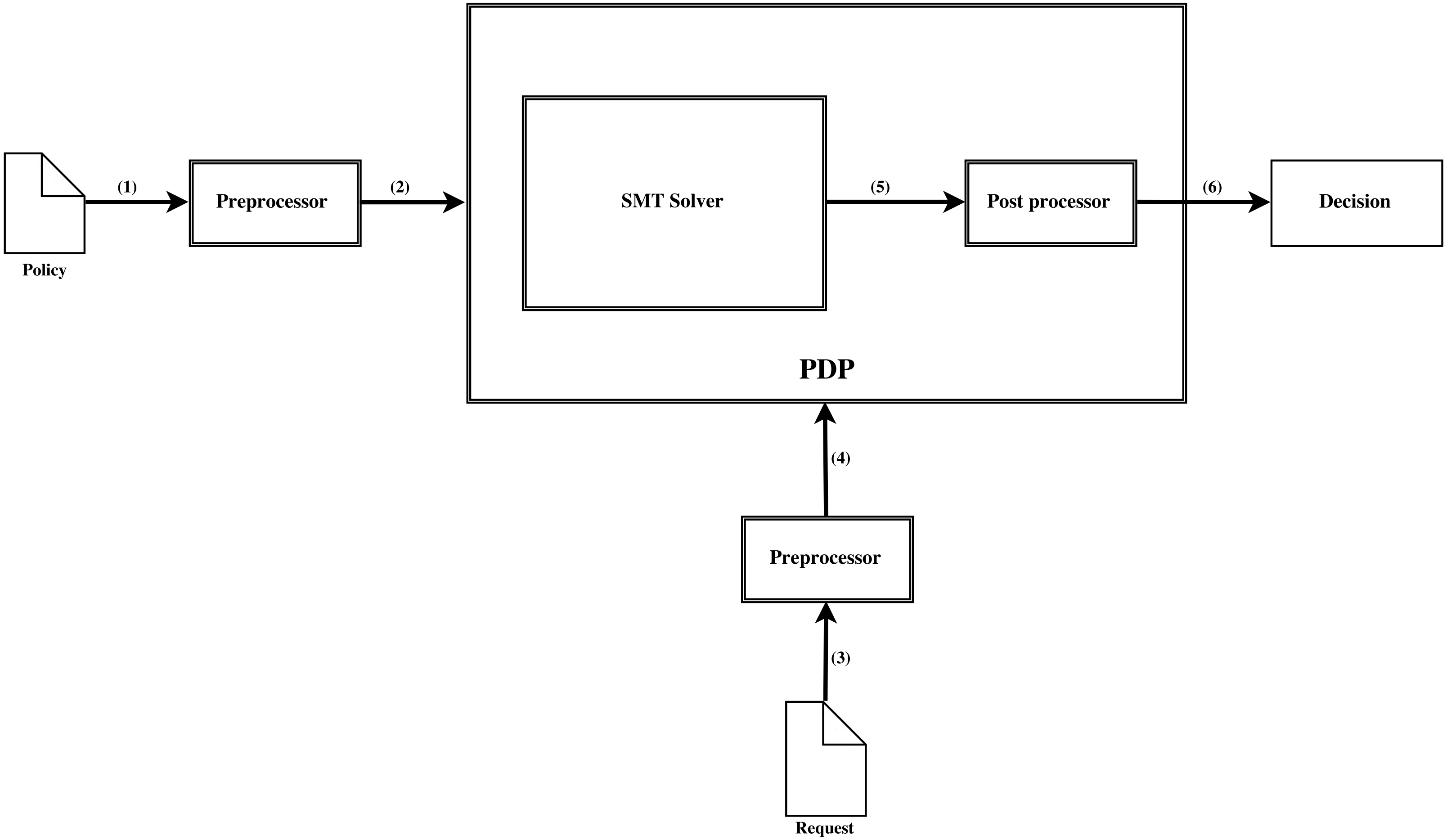
ALGORITHM

- Constructing attribute domains
- Policy flattening
- SAT encoding

SAT ENCODING

$$DS_P \cup DS_D \cup DS_{IN(P)} \cup DS_{IN(D)} \cup DS_{IN(PD)} \cup DS_{NA}$$

FRAMEWORK


(1)
Preprocessor
(2)
SMT Solver
(5)
Post processor
(6)
Decision
Policy
PDP
(4)
Preprocessor
(3)
Request

Conclusion

- Creating SAT formula
- SAT solvers
- No trees
- Experimental validation