EXTENSIBLE ACCESS CONTROL Markup Language

Author

Francisco Alexandre de Gouveia

Coordinator

Doutor Diogo Gomes

Collaborator

Engenheiro Ricardo Azevedo





Universidade de Aveiro Instituto de Telecomunicações Portugal Telecom Inovação

XACML – What is it?

- Standard defined by OASIS for extensible and generic access control
- Consists on:
 - Extensible policy language in XML
 - Extensible request-response language in XML
 - Distributed architecture based on:
 - Policy Enforcement Point
 - Policy Decision Point
 - Policy Information Point
 - Policy Administration Point





XACML – What is it used for?

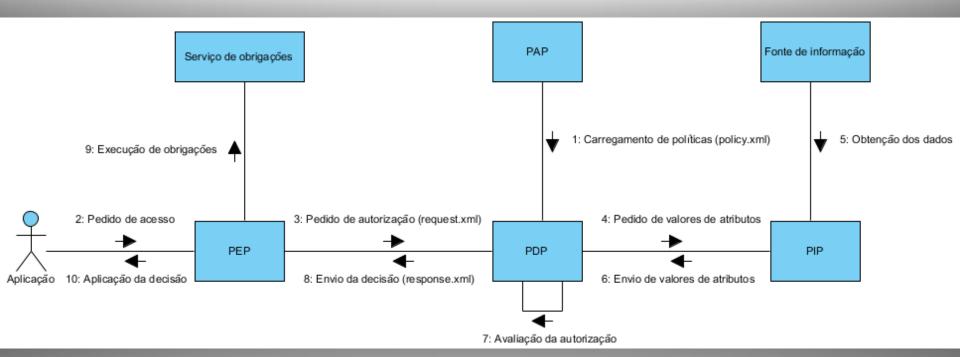
- Access control
- As it is generic and extensible, can be applied in any context:
 - Door access control
 - Web page access control
 - Service access control
- Only takes decisions!
 - Doesn't tell you which accesses do you have





XACML – How it works?

- Request-response system
- Distributed architecture





XACML - Request

- Can a subject make an action on the resource in some environment?
- Keywords:
 - Subject The one who wants to interact
 - Action Kind of interaction
 - Resource Interaction's destiny
 - Environment Anything that cannot be included in the last three categories

XACML - Request

 Can a subject make an action on the resource in some environment?

```
<Request>
                                                                                XACMLv2 example
     <Subject>
          <a href="http://www.w3.org/2001/XMLSchema#string">
                <a href="#"><AttributeValue>Anonymous</a></attributeValue>
                                                                            Subject
          </Attribute>
     </Subject>
     <Action>
          <a href="mailto:</a></a></a><a href="https://www.action.id">Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"</a>
          DataType="http://www.w3.org/2001/XMLSchema#string">
                <a href="#"><AttributeValue>Read</a></attributeValue>
                                                                                      Action
          </Attribute>
     </Action>
     <Resource>
          <a href="http://www.w3.org/2001/XMLSchema#string">
                <a href="#"><AttributeValue>Topic</a>/AttributeValue>
          </Attribute>
     </Resource>
</Request>
                                                                            Resource
```

XACML – Request

 Can a subject make an action on the resource in some environment?



XACML – Response

- Response types:
 - Access granted (Permit)
 - Access denied (Deny)
 - Indeterminate decision (Indeterminate)
 - No policies applicable (Not applicable)
- Together with:
 - Tasks to be run before granting access *





XACML – Response

- * In the 2nd version of XACML, there is an element Obligation
- Policy Enforcement Point must do all the requested tasks described in Obligations

 So, what happens when a task is not relevant for the decision but Policy Enforcement Point cannot do it?

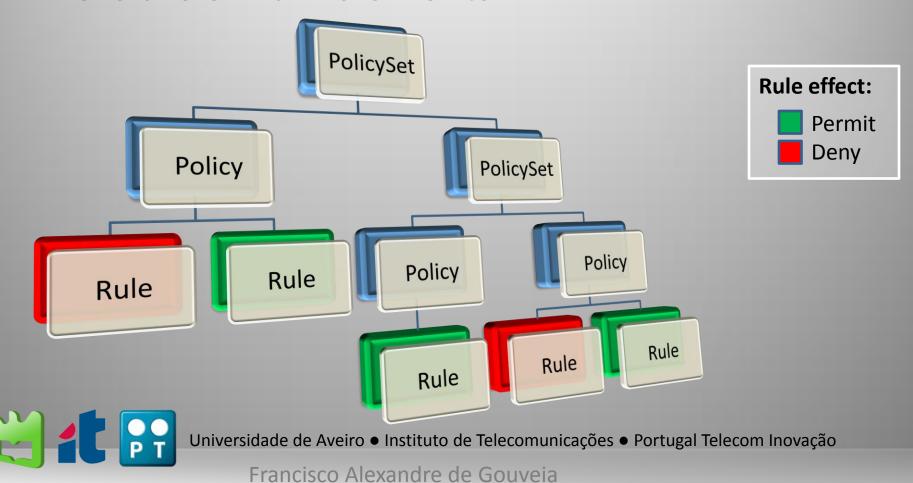
XACML – Response

- * In the 3rd version of XACML, there are Obligation and Advice elements
- Policy Enforcement Point:
 - Must run all the tasks described in Obligations
 - Should run all the tasks described in Advices

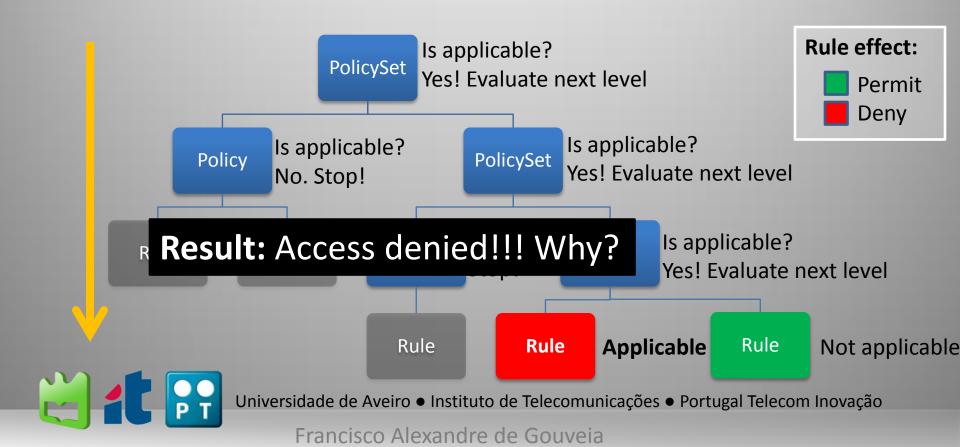
If PEP fails to do a task from Advice elements,
 the final decision is not changed



- How are policies structured?
- There are 3 main elements



- How are elements evaluated?
- Each element has a "Target"
- Evaluation is made from the top

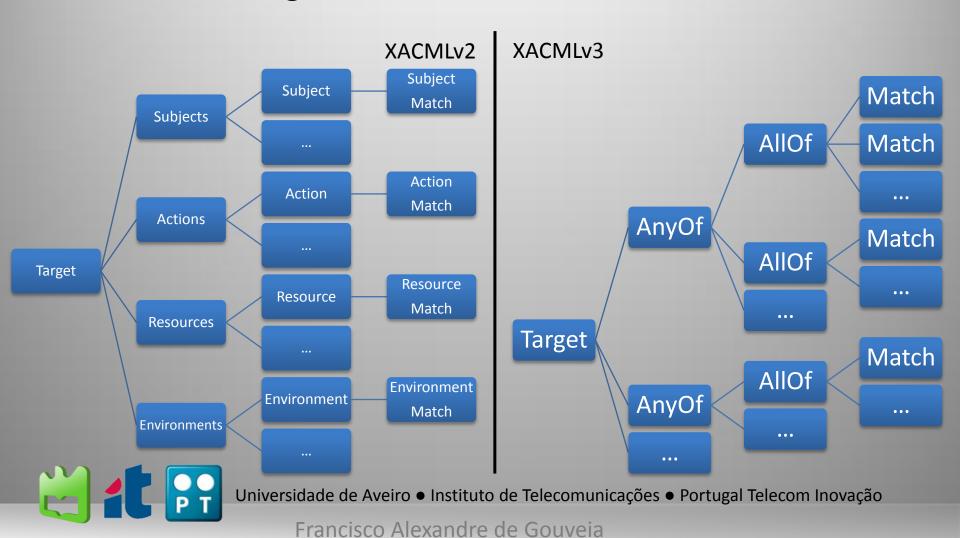


- What if more than one rule or policy is applicable?
- Answer: Combining algorithms
 - Permit-overrides
 - Deny-overrides
 - Only-one-applicable
 - First-applicable
 - ... (more can be created)





How is a Target element made?

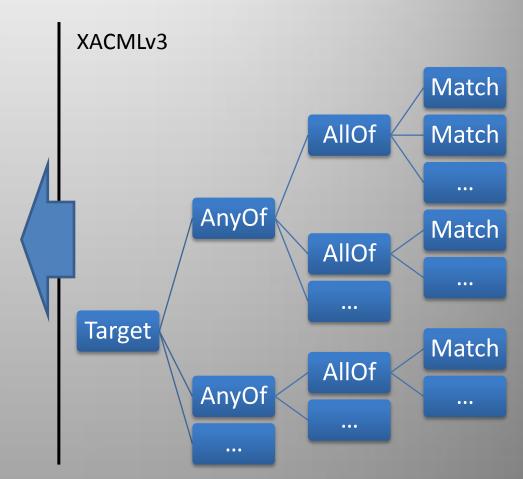


How is a Target element made?

- Uniform model for all the categories
- •Allows to define intersections and unions of Matches

But...

- •AnyOf and AllOf elements are not identified. Managing policies with such elements turns out to be a problem that can be solved by:
 - Analysing deeply element values to know where to modify
 - •Or recreating the Target element each time a change is made







PAP XACMLv3 Policy Administration Point

Author

Francisco Alexandre de Gouveia

Coordinator

Doutor Diogo Gomes

Collaborator

Engenheiro Ricardo Azevedo





Universidade de Aveiro Instituto de Telecomunicações Portugal Telecom Inovação

- Project objectives
 - Extensible information system (able to import modules without recompilation)
 - User interface that abstracts the complexity of XACMLv3
 - Creation of policies that respect the standard

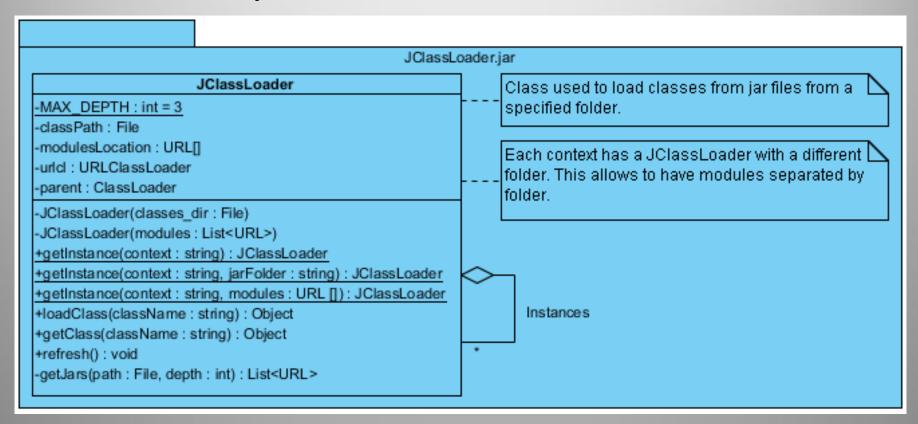




- Extensibility
 - Allows importation of classes implementing the defined interfaces for:
 - Retrieving information from Policy Information Point
 - Persistence and retrieving policies
- Used solution:
 - URLClassLoader loads classes from *.jar files in run-time



Extensibility (Class loader)





Extensibility (Interfaces)

<<Interface>> **IPolicyRetreiver** +getRootPolicy(depth : int) : Node +getPolicyTreeElement(id : string, depth : int) : Node +getPolicySet(policySetId : string, depth : int) : Node +getPolicy(policyId: string, depth:int): Node +getRule(ruleId: string, int depth): Node +insertElementIntoPolicySetAsFirst(policySetId: string, element: Node): OperationResult +insertElementIntoPolicySetAsLast(policySetId: string, element: Node): OperationResult +insertElementIntoPolicySetAfterElement(policySetId: string, elementId: string, element: node): OperationResult +insertElementIntoPolicyAsFirst(policyId: string, element: Node): OperationResult +insertElementIntoPolicyAsLast(policyId: string): OperationResult +insertElementIntoPolicyAfterElement(policyId: string, elementId: string, element: Node): OperationResult +removeElementFromPolicyTreeElement(elementId: string, elementName: string): OperationResult +removePolicySet(policySetId: string): OperationResult +removePolicy(policyld: string): OperationResult +removeRule(ruleId : string) : OperationResult +policySetExist(policySetId: string): boolean +policyExist(policyld : string) : boolean +ruleExist(ruleId: string): boolean

<<Interface>>

+getResourceDescription(id : string) : string +getResourceShortName(id : string) : string

+listResources(): Set<string>

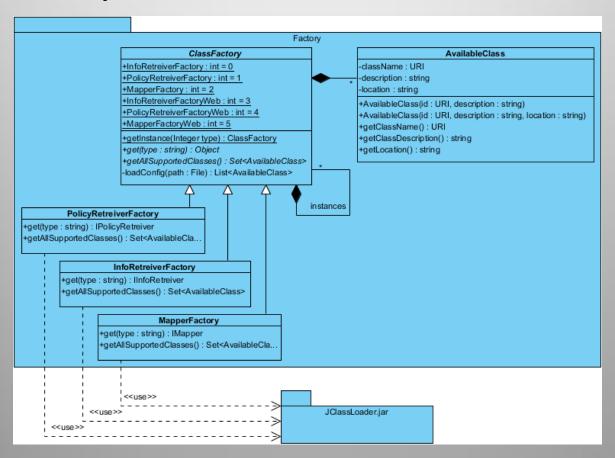
+listResources(category : string) : Set<string>

+doesMapping(): boolean

+setMapper(mapping : IMapper) : OperationResult



Extensibility (Factory)







Interfaces Policy **IPolicyRetreiver** +getRoatPalicy(depth:int):Node +getPolicyTreeElement(id: string, depth:int): Node +getPolicySet(policySetId: string, depth: int): Node w3.org.dom +getPolicy(policyld:string,depth:int): Node +getRule(ruleId: string, int depth): Node «Interface» «Interface» +insertElementIntoPalicySetAsFirst(palicySetId: string, element: Node): OperationResult NamedNodeMap Node +insertElement IntoPolicySetAsLast(policySetId: string, element: Node): OperationResult +insertElement IntoPolicySetAfterElement(policySetId: string, elementId: string, element: node): OperationResult +insertElementIntoPalicyAsFirst(palicyId: string, element: Node): OperationResult +insertElementIntoPolicyAsLast(policyId:string): OperationResult +insertElement IntoPalicyAfterElement(policyId: string, elementId: string, element: Node): OperationResult Exter +removeElementFromPolicyTreeElement(elementId: string, elementName: string): OperationResult +removePolicySet(policySetId:string): OperationResult +removePolicy(policyld: string): OperationResult +removeRule(ruleId: string): OperationResult +policySetExist(policySetId : string) : boolean +policyExist(policyld:string):boolean +ruleExist(ruleId : string) : boolean IPolicyRetreiver etRootPalicy(depth:int):Node etPalicyTreeElement(id:string,depth:int):Node etPalicySet(palicySetId:string,depth:int):Node - ugshi-olyciset(schios/selfd stating, despin :ni); Node - ugshi-olyciset(stating, selfd stating, despin :ni); Node - ugshi-olyciset(stating, despin :ni); Node - ugshi-olyciset(stating, usin despin; Node - ugshi-olyciset(stating, dement : Node - unsettlimentative) sociose(stating, dement : Node - unsettlimentative) sociose(stating, dement : Node - unsettlimentative) sociose(stating, dementative) sociose(stating, dementative) sociose(stating, dementative) sociose(stating, dementative) - unsettlimentative) sociose(stating, dementative) - unsettlimentative) sociose(stating, dementative) - unsettlimentative) - unse PolicyRetreiver XACMLNamedNodeMap PolicyRetreiver +getRootPolicy(depth:int):Node +getPolicyTreeElement(id: string, depth:int): Node +getPolicySet(policySetId: string, depth: int): Node +getPolicy(policyld: string, depth: int): Node +getRule(ruleId : string, int depth) : Node +insertElement IntoPolicySetAsFirst(policySetId: string, element: Node): OperationResult +insertElementIntoPolicySetAsLast(policySetIId: string, element: Node): OperationResult B XML +insertElement IntoPolicySetAfterElement(policySetId: string, elementId: string, element: node): OperationResult +insertElementIntoPalicyAsFirst(palicyld:string,element:Node):OperationResult ««useo» // +insertElementIntoPolicyAsLast(policyId: string): OperationResult XACMLNodeAdaptor Root Policy(death : int) : Node +insertElement IntoPalicyAfterElement(policyId: string, elementId: string, element : Node): Operation Result petPolicyTree Element(id : string, depth : int): Node petPolicySe(policySe)(d : string, depth : int): Node petPolicy(policyId : string, depth : int): Node petPolicy(policyId : string, depth : int): Node +removeElementFromPolicyTreeElement(elementId: string, elementName: string): OperationResult +removePolicySet(policySetId : string): OperationResult. geffkelerledt string, int depth; Noder insefflement in Große Aufreit in String, element : Node) - insefflement in Große Auf Finispilicity Staff : string, element : Node) - insefflement in Große Aufreit : Node) - Operant in Große Aufreit in Große Aufreit : Node : Operant in Große Aufreit in Große Aufreit in Große : string, element in Große Aufreit in Große : string; element in Große Aufreit in Große : string; element in Große : in strick in Große : string; element in Große : in strick in Große : in string : Große : in Große : in string : Große : in Große : in gereit : in string : Große : in +removePolicy(policyld: string): OperationResult +removeRule(ruleId: string): OperationResult +policySetExist(policySetId : string) : boolean +policyExist(policyld : string) : boolean COLUMNO +ruleExist(ruleId : string) : boolean dards noveRule(ruleld : string) : OperationResult licySetExist(policySetId: string): boolear nicyExist(policyld:string):boolean leExist(ruleId:string):boolean BDB XmiHelper tions BDBXmlDatabas eHelper +addNamespace(prefix : string, namespace : string) : void BDB XmlDsta +addNode(nodes : string, where : string) : XmlResults addNamespace(prefix : string, na +addNodeAfter(nodes : string, which : string) : XmlResults addNode/nodes : string, where : s addNodeAfter(nodes : string, white addNodeAsFirst/nodes : string, w +addNodeAsFirst(nodes: string, where: string): XmlResults +addNodeAsLast(nodes : string, where : string) : XmlResults arktNorteAsLast(nodes : string, v dearName spaces(): void +dearName spaces(): void ecuseo <<usess +dase(): vaid getEnvironment() : Environment +createDocument(name: string, content: string): XmlDocument +getEnvironment(): Environment dXmiContained1 - XmiContain KmilManager() : XmilManager KmilManagerConfig() : XmilMa +getEnvironmentConfig(): EnvironmentConfig +getXmiContainer(): XmiContainer query(xQuery: string): XmlResu emoveDocument(docName: stri +getXmlManager():XmlManager +getXmlManagerConfig(): XmlManagerConfig moveNamespace(prefix : string moveNode(which : string) : Xm +putDocument(uniqueName : string, fileName : string) : boolean odateNode/newNodes - string -+query(xQuery: string): XmiResuits +removeDocument(docName : string) : boolean +removeNamespace(prefix : string) : void

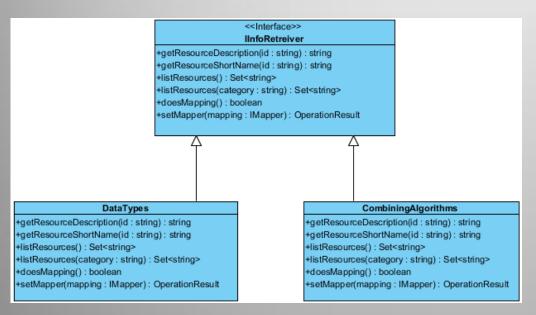
Francisco Alexandre de Gouveia

vação

+removeNode (which : string) : XmResults

+updateNode(newNodes:string,where:string):XmlResults

Extensibility (Implemented modules)



- Info Retriever
 - Reads xml file
- Returns:
 - Data types
 - Functions
 - Combining algorithms





XACML complexity abstraction on the user interface

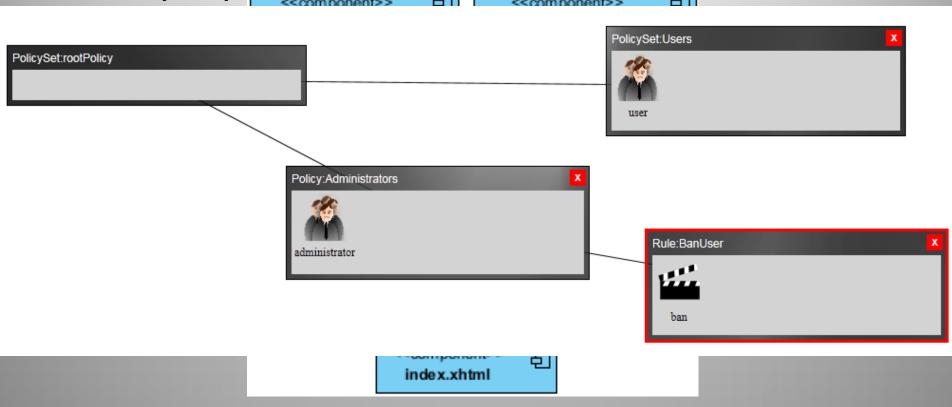
Used solution:

- Web interface with:
 - Policy representation in nodes with connections between them
 - Complex name abstractions with images and simplified names
 - XACML standard abstraction, showing only the possible options per element





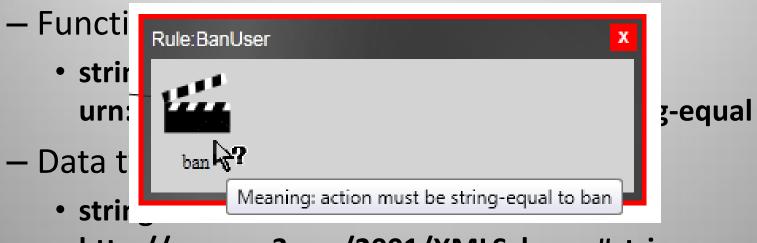
Policy representation with nodes connected







Complex names abstraction with images and simplified names



http://www.w3.org/2001/XMLSchema#string





- Creation of policies that obey to standard rules
 - Behind all abstraction, there is an effort to save the policies in respect to the standard
 - Options change in relation to the element selected, accordingly to the possibilities of this element
 - E.g.: There is a toolbox whose buttons are displayed depending on the element selected

- Implementation: J2EE (problems occurred)
 - ClassLoader didn't work as expected when hosted in an application server
 - Single instance classes weren't single instance





Java Class Loader

Works hierarchically

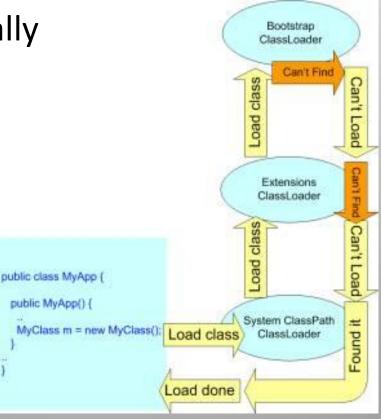


Image taken from http://www.objectsource.com/j2eechapters/Ch21-ClassLoaders_and_J2EE.htm







- Web server instances
 - Application is stored in more than one container
 - For performance purposes, load is balanced between containers

 For single instance classes, a Session Bean Singleton was used



Conclusions

- XACML allows you to make fine-grained and generic access control
- 3rd version brought improvements in relation to the 2nd version
 - Added advice element
 - Unions and intersections of targets
 - Multi-request
 - But it's still a draft...
- There are not many implementations
- Hard to administrate Targets





Questions?



