

Security and Cryptography 2020

V. ACCESS CONTROL

Mirosław Kutylowski

ACCESS CONTROL (AC)

- decision whether a subject (user, etc) is allowed to carry out a specific action (operation) on an object (resource)
- policy = set of rules of (axioms + derivation rules) for making the decision
- model: there are
 - Principals: users, processes, etc.
 - Operations performed by Principals: read, write, append, execute, delete, delegate...
 - Objects: the objects on which the principals perform operations: files, hardware execution (opening door lock)...

- **goals:**
 - availability (false rejection is a disaster in some cases)
 - no unauthorized access
 - simplicity (users may have problems to understand what is going on, complicated systems are prone to be faulty)
- **practical problems:**
 - **diversity of application scenarios**, no unique generalizations
 - **no trusted party**: administrator/system provider/product provider/user might be malicious
 - **safety** issues (access control to SCADA systems for industrial installations, ...)
 - **future**: constraints must respect future states
 - **no test environment**: designing a system much harder: hardware, software design are much easier to test, in access control one cannot create a complete environment for testing purposes
 - **cultural differences**: the users may react differently, this may lead to severe misunderstanding the situation by the AC designer

Formal description

- **goal:** define secure and insecure states:
 - **secure state** is such that the principals get access to objects as claimed by some predefined axiomatic properties,
 - **insecure states:** when the things go explicitly wrong
 - **gray zone:** states that are neither secure nor insecure: AC policy violated but still no catastrophe
- AC defines rules define a **dynamic system** – changing access rights by principals
- **basic security question:**
given: a secure state
safety problem: **is it possible to change the state to an insecure one through a number of moves that are admissible according to a given AC policy**

Theorem – Undecidability

the problem whether from a given secure state we can reach an insecure state is **undecidable**

Proof idea

- a Turing machine can be regarded as an access control system:
 - a head may be treated as a principal
 - the tape represents objects (each entry is a data)
 - transition function of TM specifies what is allowed concerning data and security status
- ask if a Turing machine can reach some state+tape contents (which is declared as insecure)
- the problem is in general undecidable

□

Corollary

It is necessary to restrict the freedom to define Access Control systems to avoid undecidability issues.

It is hard as many rewriting systems are equivalent to Turing machines (i.e. Post Correspondence Problem).

- restrictions must be strong in order to get low complexity of formal safety
- too strict restrictions make the system less flexible and hard to use

Main Models

- approaches:
 - discretionary – the objects owner defines the access rules
 - mandatory – the system takes care
- main models:
 - Access Control Matrix or Access Matrix (AM)
 - Access Control List (ACL)
 - Role-Based Access Control (RBAC)
 - Attribute-Based Access Control (ABAC)
 - lattice models
- main decisions:
 - principals: **least privilege** given to achieve what is needed
 - objects: options are A) **explicit permissions needed**
B) **admitted unless prohibited**
 - rule administration: this is the **core part** of the system

AM (Access Control Matrix)

- Objects, Subjects , Access function
- Access Function depicted as a Matrix: each entry defines allowed operations
(e.g. read, write, delete, execute, ...)

application areas: database systems, operating systems, ... the most common case: the access rights of apps in Android, iOS, ...

problems of AM:

- i. lack of scalability, no flexibility
- ii. enormous effort, unless an AC system is tiny
- iii. central administration, single point of failure (attack the administrator to get unrestricted control over the system / confuse the user of an app)

ACL (Access Control List)

- object centric: for each object a list of admitted principals
- there are file systems based on ACL, Red Hat Linux: default ACL in a directory
- relational database systems, SQL systems, network administration

advantages of ACL:

close to business models, simple, easy to implement in small and highly distributed systems

problems of ACL:

- i. management of principals is hard (blacklisting a certain principal means scanning all ACL's)
- ii. poor scalability unless the number of principals is small and certain “locality” properties
- iii. safety problems: lack of coordination between ACL may lead to problems
- iv. impossible as a global access control

so ACL is not much useful for large and complicated systems

Systems based on ACL concept

Kerberos:

- early system based on tickets:
 - 1 user authentication against a central server, service request
 - 2 cryptographic ticket issued for the user and a given service
 - 3 ticket presented by the user to a server
- symmetric crypto used: secret shared between the central server and the service servers

SPKI Simple PKI

- certificates issued by objects, certificate confirms a public key
- private key based authentication
- delegation of access rights
- local naming with the link to a public key
- concepts: short time certificates, "certificate of health", "suicide note"

RBAC (Role-Based Access Control)

- **organization:**
 - *subjects* assigned to *roles*
 - *permissions* assigned to *roles*

in a **matrix representation**

- **textbook example:** a hospital system with different roles: doctor, nurse, patient, accounting staff, family members of a patient, insurance company controller, ... :
 - a patient may read own record but not write or erase it
 - a doctor can write/read a record (but not erase)
 - a nurse can write records specifying activities, but not diagnosis part (reserved for the doctor's role)
 - ...

Problems: even in this example we get into troubles: how to meet the requirements of GDPR: how to formalize “patient’s own medical record”? Impossible with roles only.

RBAC1 standard

- hierarchy of roles (a directed acyclic graph)
- a role gets all permissions from its lower roles in the hierarchy,

advantages: description might be much smaller, logic of the system follows a strict military fashion

disadvantages: expressive power the same as for RBAC

RBAC2 standard: RBAC0+constraints for RBAC matrices:

- **mutual exclusion:** a user can get at most one role in a set, permission can be granted to only one role in a set, ...
- **cardinality:** the maximum number of subjects with a given role might be set
- **prerequisites:** necessary to have role A before getting role B
- **RBAC3=RBAC1 consolidated with RBAC2**

advantages of RBAC:

- simple management of users
- suitable for large systems with a simple “military” architecture

problems of RBAC:

- no fine tuning of access rights
 - all users with the same role have the same permissions
- problems with dynamic changes
 - splitting roles?
 - adjustments must be global
- limited use in large global systems
 - the global assignment of roles to users is an utopia

LATTICE models: many security levels organized in a directed acyclic graph

– **confidentiality policy Bell-LaPadula model:**

- information gathering like in army:
- **read-down:** read allowed only if the subject's security level dominates the object's security level
- **write-up:** write allowed only if the subject's security level is dominated by the object's security level (write-up)
- **tranquility property:** changing the security level of an object concurrent to its use is forbidden

– **Integrity Policy: Biba model**

- dual to BLP, like assigning orders in an army
- write-down, read-up

– **Ring model**

- a version of Biba model for an operating system, focus on integrity
- reading up is allowed, writing down is allowed. But within a specified ring (a, b) (between layers a and b)

Denning's Lattice Model

- a general model for information flow
- a partial order of different levels, not always a linear order
- lub operation: least upper bound of two nodes
- takes into account implicit and explicit data flows
- static and dynamic binding to levels may be considered

Explicit and implicit data flows

- explicit: e.g. via an assignment $b := a$
- implicit: via conditional: if $a = 0$ then $b := c$
information on a is leaked via a change/no change in b
- programs considered composed of assignments via composing sequences and conditionals

Security rules

- explicit flows according to the rules
- for a sequence each single element must be secure
- a conditional is secure if
 - the flows in the body are secure
 - the implicit flow by conditional is secure

Binding to levels

- static: binding to levels is fixed (Biba, BLP, ...)
 - security checked during runtime
 - Data Mark Machine: mark data in the program, mark of a depends on its location
- dynamic:
 - huge problems with implicit data flows – the fact what is at a given level leaks data
 - High Water Mark technique: after executing $b := a$ the level of b is raised to the least upper bound of a and b (in case of BLP, for Biba use Low Water Mark)

Decentralized Label Model

- any entity has a list of labels
- operation allowed if the lists of subject and object intersect
- semantics:
 - constructions such as “if appropriate” for exceptions

Chinese Wall (the name is misleading)

- focused on conflicts of interest
- Conflict of Interest Classes (CIC) – from each class access to ≤ 1 dataset allowed
- In each CIC some number of datasets, datasets belong to companies
- each object in some dataset
- an extra CIC of one dataset of sanitized objects (data freely available)
- allowed access:
 - S can read O only if
 - S has already read an object from the same dataset as O
 - S has not read any object from the CIC containing O
 - writing more complicated: user A has access to dataset CompanyA in CIC x_1 , user B has access to dataset CompanyB in CIC x_1 , both have access to dataset BankA in CIC x_2 . Unconstrained writing in objects from BankA would enable leakage from CompanyA to CompanyB
 - S can write into O only if
 - the same condition as for read (“simple security rule”), and

- "no object can be read which is in a different company dataset to the one for which write access is requested" - consequence is that no writing if a subject has read datasets of two companies

- an axiomatized approach enabling formal verification is possible

AC for group of principals

- examples:
 - exporting a secret key from HSM and enabling reconstruction iff k shares are used
 - access granted when a smart card and a smartphone of a user are involved in identification
- in general:
 - access granted iff a certain number of subjects involved simultaneously
 - the simplest scenario: threshold scheme – at least k out of n in a group request an access
 - general case: *access structure*:
 - a family \mathcal{F} of subsets of a group which is an ideal: if $A \in \mathcal{F}$ and $A \subset B$, then $B \in \mathcal{F}$
 - it might be difficult to provide a compact description of an ideal, however the practical cases should enable formulation as a simple Boolean formula using \wedge and \vee operators (and no negation \neg)
 - given such a formula it is easy to create an access system based on secret sharing and authentication with the key reconstructed from secret sharing: \wedge corresponds to secret sharing based on XOR, \vee corresponds to Shamir's scheme 1 out of k based on polynomial of degree 1,

ABAC – Attribute Based Access Control

OASIS specification with some examples:

see <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf>

recommended architecture:

- **language:** – standard: eXtensible Access Control Markup Language (XACML)
- **policies:**
 - **rule** (stating when to allow or deny access) → **policy** as a set of rules → **policy sets** composed of policies
 - policies might be distributed, PIP (policy information point) keeps track where to find policies
 - unique policy names within one PIP
 - **policy evaluation engine** provides an **automatic evaluation** of a request based on policy sets and attributes — the engine might be very complex!

- **policy evaluation:** process answering access queries, computation based on entities, attributes and current policies – ADF (access decision function):
 - i identify policies applicable to the current request,
 - ii the request evaluated against each rule from these policies – it may require evaluation of attributes and pattern matching,
 - iii the results are combined within a policy,
 - iv the results combined between policies
 - v a decision revealed together with obligations (compulsory) and advices (optional to follow)

- **subjects:**

- entities asking for access

- **attributes:**

- e.g. name, ID, network domain, email address
- numeric: age, and other multivalued

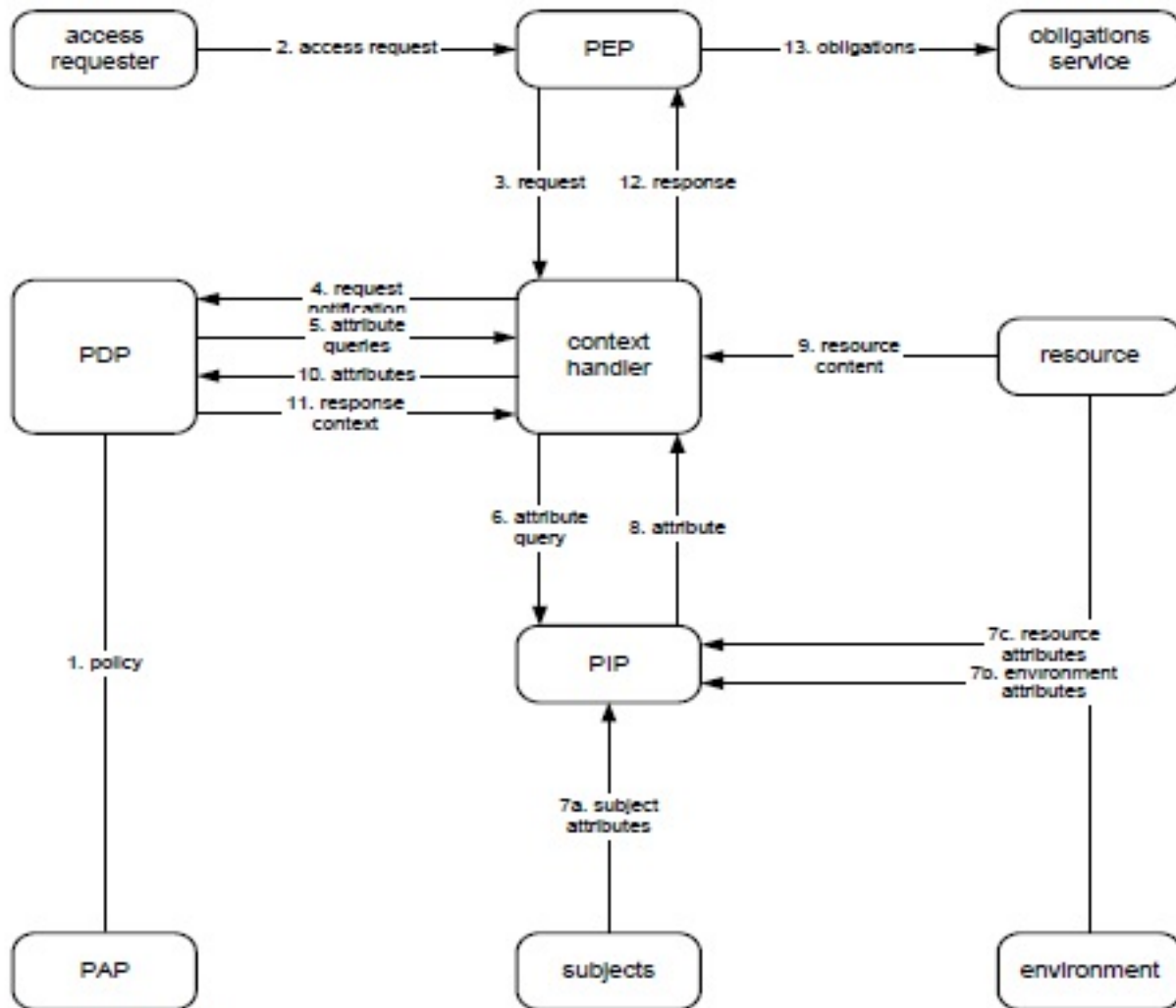
- **resource:**

- actions on resources to be denied or permitted (or left undecided)

- **data form:** tree structures (or forest) typical for XML
- **environment:**
 - it provides a set of logical and mathematical operators on attributes of the subject, resource and environment.

– **recommended architecture:**

- **Policy Enforcement Point (PEP)** – handles different syntax from external systems (not everything in XACML, so a translation might be needed)
- **Policy Information Point (PIP)** – responsible for gathering data on attributes, environment,
- **Policy Administration Point (PAP)** - provides to PDP everything concerning policies
- **Policy Decision Point (PDP)** - makes actual evaluation



9
n

Figure 1 - Data-flow diagram

(figure from the ABAC manual)

steps :

- access request goes to PEP
- access request goes to context handler
- request for attributes value to PIP
- PIP gets attributes evaluation for environment, resources and subjects
- attribute queries sent to context handler
- data on resources from resources to context handler
- everything to PDP
- response from PDP with a decision
- response from context handler to PEP
- obligations from PEP to obligations service (not really within the standard what and how to do)

major problems:

- creating policies is a complicated task, errors are likely
- trade-off between expresiveness and complexity of evaluation

Rule

components: target, effect, condition, obligation, advice

- **target:** defines where the rule should be applied
 - a **if no target** given in the rule, then the (more general) **target from the policy** applies
 - b **subject, resource and action defined by target should match** the **subject, resource and action from the request**
 - c **if not matched**, then the **rule is not applicable**
 - d **shortcuts**, e.g.: <AnySubject />, <AnyResource />, <AnyAction />
 - e **subjects** interpreted as a subtree starting in a node;
 - matching of the whole tree
 - f **resources:** different **options** for matching:
 - i the contents of the identified node only,
 - ii the contents of the identified node and the contents of its immediate child nodes
 - iii the contents of the identified node and all its descendant nodes
 - g **XML pointers** may be used, values fetched during evaluation

Further components of a rule

- **effect:** "Permit" or "Deny" (to be applied if the rule is applicable)
- **condition:** Boolean expression to be satisfied (so in order to apply the rule we must have both matching with the target and condition evaluated to true)
- **obligation expression:** it is mandatory to be fulfilled by PEP
 - e.g. when giving access to an account in online banking to send an SMS to the client
- **advice expression:** may be ignored by PEP

Important feature of ABAC

- the rules need not to be consistent with each other
- it will be the job of a policy to resolve the conflicts

```
<Rule RuleId= "urn:oasis:names:tc:xacml:3.0:example:SimpleRule1" Effect="Permit">
  <Description>
    Any subject with an e-mail name in the med.example.com domain can perform any action on any resource.
  </Description>
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">
            med.example.com</AttributeValue>
          <AttributeDesignator
            MustBePresent="false"
            Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
            AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
            DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>
```

Policy components:

- **target:** one of more patterns, different approaches for combining definitions of targets (must be in one of targets from policy rules or in all targets of the rules)
- **rule-combining algorithm identifier** determines how to combine the rules' effects (e.g. all rules MUST have the effect Permit, or at least one Permit), some algorithms are predefined but one can define own ones
- **set of rules**
- **obligation expressions** and **advice expressions**

decisions of a policy:

- permit
- deny
- not applicable
- indeterminate

access granted only if **“permit” and obligations fulfilled**

<?xml version="1.0" encoding="UTF-8"?>

<Policy

xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17

http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"

PolicyId="urn:oasis:names:tc:xacml:3.0:example:SimplePolicy1"

Version="1.0"

RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">

<Description> Medi Corp access control policy </Description>

<Target/>

<Rule ...>

...

</Rule>

</Policy>

attributes evaluation

a few flexible options:

- string matching algorithms
- XPath selection in an XML structure
- result might be a *bag* of attributes - all matchings according to the selection specified

target evaluation

- AnyOf : match if all options match, if one does not match then “ no match”
- AllOf: it suffices to match one option, but in this option all components must be matched

rule evaluation:

target:	condition:	rule value:
match or no target	true	effect
match or no target	false	not applicable
match or no target	indeterminate	indeterminate
no match	*	not applicable
indeterminate	*	not applicable

Table 1.

policy evaluation:

target:	rule value:	policy value:
match	at least one rule with Effect	according to rule-combining algorithm
match	all rules with NotApplicable	not applicable
match	at least one rule Indeterminate	according to rule-combining algorithm
no match	*	not applicable
indeterminate	*	indeterminate

standard rule combining algorithms:

- a) deny-overrides
- b) permit-overrides
- c) first-applicable
- d) only-one-applicable

security issues:

- i. replay attack – works unless some freshness safeguards
- ii. message insertion – it may create a lot of harm, message authentication needed
- iii. message deletion – the system should prevent permitting access if some message undelivered
- iv. message modification - obviously one could change the decision (authentication required)
- v. NotApplicable - dangerous since sometimes automatically converted to Permit (webservers...)
- vi. negative rules - not always effective: in some cases evaluation result is undeterminite, so there is no False and access is granted
- vii. DoS - via loops in evaluating policies

further **example from OASIS specification:**

medical database, form of a record

[1] <?xml version="1.0" encoding="UTF-8"?>

[2] <record xmlns="urn:example:med:schemas:record"

[3] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

[4] <patient>

[5] <patientName>

[6] <first>Bartholomew</first>

[7] <last>Simpson</last>

[8] </patientName>

[9] <patientContact>

[10] <street>27 Shelbyville Road</street>

[11] <city>Springfield</city>

[12] <state>MA</state>

[13] <zip>12345</zip>

[14] <phone>555.123.4567</phone>

[15] <fax/>

[16] <email/>

[17] </patientContact>

[18] <patientDoB>1992-03-21</patientDoB>

[19] <patientGender>male</patientGender>

[20] <patient-number>555555</patient-number>
[21] </patient>
[22] <parentGuardian>
[23] <parentGuardianId>HS001</parentGuardianId>
[24] <parentGuardianName>
[25] <first>Homer</first>
[26] <last>Simpson</last>
[27] </parentGuardianName>
[28] <parentGuardianContact>
[29] <street>27 Shelbyville Road</street>
[30] <city>Springfield</city>
[31] <state>MA</state>
[32] <zip>12345</zip>
[33] <phone>555.123.4567</phone>
[34] <fax/>
[35] <email>homers@aol.com</email>
[36] </parentGuardianContact>
[37] </parentGuardian>
[38] <primaryCarePhysician>
[39] <physicianName>
[40] <first>Julius</first>

[41] <last>Hibbert</last>
[42] </physicianName>
[43] <physicianContact>
[44] <street>1 First St</street>
[45] <city>Springfield</city>
[46] <state>MA</state>
[47] <zip>12345</zip>
[48] <phone>555.123.9012</phone>
[49] <fax>555.123.9013</fax>
[50] <email/>
[51] </physicianContact>
[52] <registrationID>ABC123</registrationID>
[53] </primaryCarePhysician>
[54] <insurer>
[55] <name>Blue Cross</name>
[56] <street>1234 Main St</street>
[57] <city>Springfield</city>
[58] <state>MA</state>
[59] <zip>12345</zip>
[60] <phone>555.123.5678</phone>
[61] <fax>555.123.5679</fax>

[62] <email/>
[63] </insurer>
[64] <medical>
[65] <treatment>
[66] <drug>
[67] <name>methylphenidate hydrochloride</name>
[68] <dailyDosage>30mgs</dailyDosage>
[69] <startDate>1999-01-12</startDate>
[70] </drug>
[71] <comment>
[72] patient exhibits side-effects of skin coloration and carpal degeneration
[73] </comment>
[74] </treatment>
[75] <result>
[76] <test>blood pressure</test>
[77] <value>120/80</value>
[78] <date>2001-06-09</date>
[79] <performedBy>Nurse Betty</performedBy>
[80] </result>
[81] </medical>
[82] </record>

Request example

```
[1] <?xml version="1.0" encoding="UTF-8"?>
[2] <Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
[3] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[4] xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
    http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
[5] ReturnPolicyIdList="false">
[6] <Attributes
[7] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
[8] <Attribute IncludeInResult="false"
[9] AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[10] Issuer="med.example.com">
[11] <AttributeValue
[12] DataType="http://www.w3.org/2001/XMLSchema#string">CN=Julius
    Hibbert</AttributeValue>
[13] </Attribute>
[14] <Attribute IncludeInResult="false"
[15] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
```

```
[16] Issuer="med.example.com">
[17] <AttributeValue
[18] DataType="http://www.w3.org/2001/XMLSchema#string"
[19] >physician</AttributeValue>
[20] </Attribute>
[21] <Attribute IncludeInResult="false"
[22] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id"
[23] Issuer="med.example.com">
[24] <AttributeValue
[25] DataType="http://www.w3.org/2001/XMLSchema#string">jh1234</AttributeValue>
[26] </Attribute>
[27] </Attributes>
[28] <Attributes
[29] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
[30] <Content>
[31] <md:record xmlns:md="urn:example:med:schemas:record"
[32] xsi:schemaLocation="urn:example:med:schemas:record
[33] http://www.med.example.com/schemas/record.xsd">
[34] <md:patient>
[35] <md:patientDoB>1992-03-21</md:patientDoB>
```

[36] <md:patient-number>555555</md:patient-number>
[37] <md:patientContact>
[38] <md:email>b.simpson@example.com</md:email>
[39] </md:patientContact>
[40] </md:patient>
[41] </md:record>
[42] </Content>
[43] <Attribute IncludeInResult="false"
[44] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector" >
[45] <AttributeValue
[46] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[47] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
[48] >md:record/md:patient/md:patientDoB</AttributeValue>
[49] </Attribute>
[50] <Attribute IncludeInResult="false"
[51] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace" >
[52] <AttributeValue
[53] DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[54] >urn:example:med:schemas:record</AttributeValue>
[56] </Attributes>


```
[57] <Attributes
[58] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
[59] <Attribute IncludeInResult="false"
[60] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" >
[61] <AttributeValue
[62] DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
[63] </Attribute>
[64] </Attributes>
[65] <Attributes
[66] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment">
[67] <Attribute IncludeInResult="false"
[68] AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date" >
[69] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date"
[70] >2010-01-11</AttributeValue>
[71] </Attribute>
[72] </Attributes>
[73] </Request>
```

Constructing a policy

Rule 1: A patient may read records concerning him or her.

Rule 2: The same concerns a parent or guardian if a patient is under 16 years of age.

Rule 3: The designated primary care physician can write, provided that an email is sent to the patient.

Rule 4: An administrator shall not be permitted to read or write any record.

How to express it as a policy? One should be careful:

- how to check that the record corresponds to the requester?
- what about a parent? How to check the current date? (no access in the day of 16th birthday and later), can a child enter own medical record?
- what if the administrator gets sick? Then Rule 1 and 4 are contradictory
- can a doctor be an administrator?

Rule 1

[f1] <?xml version="1.0" encoding="UTF-8"?>

[f2] <Policy

[f3] xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"

[4] xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"

[5] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

[6] xmlns:md="http://www.med.example.com/schemas/record.xsd"

[7] PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:1"

[8] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides"

Version="1.0">

[10] <PolicyDefaults>

[11] <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>

[12] </PolicyDefaults>

[13] <Target/>

[14] <VariableDefinition VariableId="17590034">

[15] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

[16] <Apply

[17] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">

[18] <AttributeDesignator

[19] MustBePresent="false"

[20] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"

[21] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:patient-number"

[22] DataType="http://www.w3.org/2001/XMLSchema#string"/>

[23] </Apply>

[24] <Apply

1054 [25] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">

[26] <AttributeSelector

[27] MustBePresent="false"

[28] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"

[29] Path="md:record/md:patient/md:patient-number/text()"

[30] DataType="http://www.w3.org/2001/XMLSchema#string"/>

[31] </Apply>

[32] </Apply>

[33] </VariableDefinition>

[34] <Rule

[35] RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"

[36] Effect="Permit">

[37] <Description>

[38] A person may read any medical record in the

<http://www.med.example.com/schemas/record.xsd> namespace

for which he or she is the designated patient

[41] </Description>

[42] <Target>

[43] <AnyOf>

[44] <AllOf>

[45] <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">

[46] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"

[47] >urn:example:med:schemas:record</AttributeValue>

[48] <AttributeDesignator

[49] MustBePresent="false"

[50] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"

[51] `AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"`

[52] `DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>`

[53] `</Match>`

[54] `<Match`

[55] `MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">`

[56] `<AttributeValue`

[57] `DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"`

[58] `XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"`

[59] `>md:record</AttributeValue>`

[60] `<AttributeDesignator`

[61] `MustBePresent="false"`

[62] `Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"`

[63] `AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"`

[64] `DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>`

[65] `</Match>`

[66] `</AllOf>`

[67] `</AnyOf>`

[68] <AnyOf>
[69] <AllOf>
[70] <Match
[71] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[72] <AttributeValue
[73] DataType="http://www.w3.org/2001/XMLSchema#string"
[74] >read</AttributeValue>
[75] <AttributeDesignator
[76] MustBePresent="false"
[77] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
[78] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[79] DataType="http://www.w3.org/2001/XMLSchema#string"/>
[80] </Match>
[81] </AllOf>
[82] </AnyOf>
[83] </Target>
[84] <Condition>
[85] <VariableReference VariableId="17590034"/>
[86] </Condition>
[87] </Rule>
[88] </Policy>

Rule 2

[1] <?xml version="1.0" encoding="UTF-8"?>

[2] <Policy

[3] xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"

[4] xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"

[5] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

[6] xmlns:xf="http://www.w3.org/2005/xpath-functions"

[7] xmlns:md="http://www.med.example.com/schemas/record.xsd"

[8] PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"

[9] Version="1.0"

[10] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining- algorithm:deny-overrides">

[11] <PolicyDefaults>

[12] <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>

[13] </PolicyDefaults>

[14] <Target/>

[15] <VariableDefinition VariableId="17590035">

[16] <Apply

[17] FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal">

[18] <Apply

[19] FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">

[20] <AttributeDesignator

[21] MustBePresent="false"

[22] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"

[23] AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"

[24] DataType="http://www.w3.org/2001/XMLSchema#date"/>

[25] </Apply>

[26] <Apply

[27] FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration">

[28] <Apply

[29] FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">

[30] <AttributeSelector

[31] MustBePresent="false"

```
[32]     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
[33]     Path="md:record/md:patient/md:patientDoB/text()"
[34]     DataType="http://www.w3.org/2001/XMLSchema#date" />
[35] </Apply>
[36] <AttributeValue
[37] DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration"
[38] >P16Y</AttributeValue>
[39] </Apply>
[40] </Apply>
[41] </VariableDefinition>
```

[42] <Rule

[43] RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"

[44] Effect="Permit">

[45] <Description>

[46] A person may read any medical record in the

[47] <http://www.med.example.com/records.xsd> namespace

[48] for which he or she is the designated parent or guardian,

[49] and for which the patient is under 16 years of age

[50] </Description>

[51] <Target>

[52] <AnyOf>

[53] <AllOf>

[54] <Match

[55] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">

[56] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"

[57] >urn:example:med:schemas:record</AttributeValue>

[58] <AttributeDesignator

[59] MustBePresent="false"

[60] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"

[61] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"

[62] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>

[63] </Match>

[64] <Match

[65] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">

[66] <AttributeValue

[67] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"

[68] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"

[69] >md:record</AttributeValue>

[70] <AttributeDesignator

[71] MustBePresent="false"

[72] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"

[73] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"

[74] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>

[75] </Match>

[76] </AllOf>

[77] </AnyOf>

[78] <AnyOf>

[79] <AllOf>

[80] <Match

[81] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

[82] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"

[83] >read</AttributeValue>

[84] <AttributeDesignator

[85] MustBePresent="false"

[86] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"

[87] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

[88] DataType="http://www.w3.org/2001/XMLSchema#string"/>

[89] </Match>

[90] </AllOf>

[91] </AnyOf>

[92] </Target>

[93] <Condition>

[94] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">

[95] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

[96] <Apply

[97] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">

[98] <AttributeDesignator

[99] MustBePresent="false"

[100] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"

[101] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id"

[102] DataType="http://www.w3.org/2001/XMLSchema#string"/>

[103] </Apply>

[104] <Apply

[105] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">

[106] <AttributeSelector

[107] MustBePresent="false"

[108] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"

[109] Path="md:record/md:parentGuardian/md:parentGuardianId/text()"

[110] `DataType="http://www.w3.org/2001/XMLSchema#string" />`

[111] `</Apply>`

[112] `</Apply>`

[113] `<VariableReference VariableId="17590035" />`

[114] `</Apply>`

[115] `</Condition>`

[116] `</Rule>`

[117] `</Policy>`

Policy Set

[1] <?xml version="1.0" encoding="UTF-8"?>

[2] <PolicySet

[3] xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"

[4] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

[5] PolicySetId="urn:oasis:names:tc:xacml:3.0:example:policysetid:1"

[6] Version="1.0"

[7] PolicyCombiningAlgId=

[8] "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">

[9] <Description>

[10] Example policy set.

[11] </Description>

[12] <Target>

[13] <AnyOf>

[14] <AllOf>

[15] <Match

[16] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

[17] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"

[18] >urn:example:med:schema:records</AttributeValue>

[19] <AttributeDesignator

[20] MustBePresent="false"

[21] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"

[22] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"

[23] DataType="http://www.w3.org/2001/XMLSchema#string"/>

[24] </Match>

[25] </AllOf>

[26] </AnyOf>

[27] </Target>

[28] <PolicyIdReference>

[29] urn:oasis:names:tc:xacml:3.0:example:policyid:3

[30] </PolicyIdReference>

[31] <Policy

[32] PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"

[33] RuleCombiningAlgId=

[34] "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides"

[35] Version="1.0">

[36] <Target/>

[37] <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"

[38] Effect="Permit">

[39] </Rule>

[40] <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"

[41] Effect="Permit">

[42] </Rule>

[43] <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"

[44] Effect="Deny">

[45] </Rule>

[46] </Policy>

[47] </PolicySet>