Interactive Access Control for Autonomic Systems: Theory and Implementation

HRISTO KOSHUTANSKI[†] and FABIO MASSACCI
University of Trento, Italy

Autonomic communication and computing is a new paradigm for dynamic service integration over a network. An autonomic network crosses organizational and management boundaries and is provided by entities that see each other just as partners. For many services no autonomic partner may guess a priori what will be sent by clients nor clients know a priori what credentials are required to access a service.

To address this problem we propose a new *interactive access control*: servers should interact with clients asking for missing credentials necessary to grant access, whereas clients may supply or decline the requested credentials. Servers evaluate their policies and interact with clients until a decision of grant or deny is taken.

This proposal is grounded in a formal model on policy-based access control. It identifies the formal reasoning services of deduction, abduction and consistency checking. Based on them, the work proposes a comprehensive access control framework for autonomic systems.

An implementation of the interactive model is given followed by system performance evaluation outlining its practical relevance.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection—*Access controls, Information flow controls*

General Terms:

Additional Key Words and Phrases:

1. INTRODUCTION

Controlling access to services is a key aspect of networking and the last few years have seen the domination of policy-based access control. Indeed, the paradigm is broader than simple access control and one may speak of *policy-based self-management networks* (see [Sloman and Lupu 1999; Lymberopoulos et al. 2003] or the IEEE Policy Workshop series[1]). The intuition is that actions of nodes controlling access to services are automatically derived from policies. The nodes look at events, requested actions and credentials presented to them, evaluate the policy rules according to those new facts and derive the actions [Sloman and Lupu 1999; Smirnov 2003]. Policies can be "simple" `iptables` configuration rules for Linux firewalls[2] or complex logical policies expressed in languages such as Ponder [Damianou et al. 2001] or a combination of policies across heterogeneous systems as in OASIS XACML[3] framework.

Dynamic coalitions and autonomic communication add new challenges: an auto-

[1] http://www.policy-workshop.org
[2] See http://www.netfilter.org
[3] http://www.oasis-open.org/committees/xacml

[†] Corresponding author: University of Malaga, Spain. E-mail: hristo@lcc.uma.es

nomic network comprises nodes that are no longer within the boundary of a single enterprise, which could deploy its policies on each and every node and guarantee interoperability. An autonomic network is characterized by properties of self-awareness, self-management and self-configuration of its constituent nodes. In an autonomic network, nodes are like partners that offer services and lightly integrate their efforts into one (hopefully coherent) network.

Since access to network services is offered by autonomic nodes on their own and to potentially unknown clients, the decision to grant or deny access can only be made on the basis of credentials sent by a client. Credentials could be issued by certificate authorities belonging to distinct domains and could be located on distributed repositories. This aspect has emerged as the notion of distributed access control, also referred as trust management [Weeks 2001], and has become a major issue for any existing information system.

Credentials themselves convey sensitive information and often become subject of unauthorized misused and disclosure. Recent several years have seen the emergence of a new concept called trust negotiation [Seamons and Winsborough 2002; Winslett et al. 2002]. It is a policy-based technique that provides clients with the right to protect their own credentials and to negotiate with servers access to those credentials. Thus trust negotiation allows two network entities (nodes) to mutually establish requirements to access a resource by requesting each other sensitive credentials until a sufficient trust is established.

Although several efficient and powerful systems have been developed so far [Bonatti and Samarati 2002; Yu et al. 2003; Bertino et al. 2004; Nejdl et al. 2004; Constandache et al. 2007] none of them provides a clear execution framework for access control, the key missing component presented in our paper. The logical model presented in this paper fills an important gap between policy specification and policy evaluation and enforcement.

In an autonomic communication scenario a client might have all the necessary credentials to access a service but may simply not know it. Equally, it is unrealistic to assume that servers will publish their security policies on the web so that clients can do policy evaluations themselves. So, it should be possible for a server to ask a client on the fly for additional credentials whereas the client may disclose or decline to provide them. Next, the server re-evaluates the client's request, considering the newly submitted credentials, and computes an access decision. The process iterates between the server and the client until a final decision of grant or deny is taken. We call this modality *interactive access control.*

Part of these challenges can be solved by using policy-based self-management of networks but not all of them. Indeed, if we abstract away the details of the policy implementation, one can observe that the only reasoning service that is actually used by policy-based approaches is *deduction*: given a policy and a set of additional facts find out all consequences (actions or obligations) from the policy according to the facts. We simply look whether granting the request can be deduced from the policy and the current facts. Policies could be different [Bertino et al. 2001; Li et al. 2003; Bonatti and Samarati 2002; Winslett et al. 2005; Li et al. 2005] but the kernel reasoning is the same.

Access control for autonomic communications needs another reasoning service:

abduction [Shanahan 1989]. Loosely speaking, we could say that abduction is deduction in reverse: given a policy and a request to access a network service we want to know what are the credentials (facts) that would grant access. Logically, we want to know whether there is a (possibly minimal) set of facts that added to the policy would entail (deduce) the request.

If we look again at our intuitive description of the interactive access control, it is immediate to realize that abduction is the core service needed by the policy-based autonomic servers to reason for missing credentials.

1.1 Paper Contribution

We present a model for reasoning about access control for autonomic communication. The model abstracts from a specific policy language and provides an algorithm based on deduction and abduction reasoning services for policy evaluation and enforcement. The key aspect of the algorithm is that if a client does not have enough access rights the algorithm computes on the fly missing credentials necessary for the client to get access. Thus a server interacts with a client asking for possible solutions that unlock a service.

Further the paper explores the interactive access control model from theoretical and practical aspects. It analyzes the behavior of the model when applied on monotonic and non-monotonic policies and against cooperative and malicious clients. Technical guarantees of correctness and completeness are proven against monotonic and a set of non-monotonic policies, called well-behaved policies.

An implementation of the model is given together with its performance evaluation.

The paper follows by introducing the starting point of interactive access control and comparing it with existing approaches. Section 3 presents semantics of the logical model and identifies the different reasoning services used in the model. Section 4 introduces the interactive access control algorithm followed by a comprehensive example of how the algorithm works (Section 5). Section 6 formally defines and proves the model's guarantees of correctness and completeness. Section 7 describes in details the interactive access control prototype, called iAccess. Next, Section 8 presents the prototype performance evaluation. Section 9 concludes the paper.

Appendix A shows proofs of theorems stated in the paper.

2. FROM ACCESS CONTROL TO INTERACTION

We will introduce the concept of interactive access control by evolving existing access control frameworks.

Let us start with the traditional access control. A server has a *security policy for access control* $\mathcal{P}_{\mathcal{A}}$ that is used when taking decisions about the usage of services offered by a service provider. A user submits a set of credentials $\mathcal{C}_p$ and a service request r in order to execute a service. We say that policy $\mathcal{P}_{\mathcal{A}}$ and credentials $\mathcal{C}_p$ entail r meaning that request r should be granted by the policy $\mathcal{P}_{\mathcal{A}}$ and the presented credentials $\mathcal{C}_p$.

Figure 1 shows the "traditional" access control decision process [De Capitani di Vimercati and Samarati 2001]. Whether the decision process uses Role-Based Access Control [Sandhu et al. 1996], Simple Public Key Infrastructure [SPKI 1999], RT framework [Li and Mitchell 2003] or other trust management framework it is

immaterial at this stage: they can be captured by suitably defining $\mathcal{P}_\mathcal{A}$, $\mathcal{C}_p$ and the entailment operator.

---

(1) check whether $\mathcal{P}_\mathcal{A}$ and $\mathcal{C}_p$ entail r,
(2) if the check succeeds then **grant** access
(3) else **deny** access.

Fig. 1. Traditional Access Control

A number of works has deemed such blunt denials unsatisfactory. Bonatti and Samarati [2002] and Yu et al. [2003] proposed to send back to clients some of the policy rules that are necessary to gain additional access. Subsequent promising approaches [Bertino et al. 2004; Nejdl et al. 2004; Kapadia et al. 2004; Constandache et al. 2007] have developed their own polices and mechanisms for deriving and negotiating policy rules. Figure 2 shows the essence of the approaches.

---

(1) check whether $\mathcal{P}_\mathcal{A}$ and $\mathcal{C}_p$ entail r,
(2) if the check succeeds then **grant** access else
  (a) find a rule $r \leftarrow p \in \mathcal{P}_\mathcal{A} \cup \mathcal{C}_p$, where p is a partial policy protecting r,
  (b) if such a rule exists then send it back to the client else deny access.

Fig. 2. Disclosable Access Control

If we look at the underlying policy models one can find that either of the approaches requires policies to be flat, i.e. a policy protecting a resource must contain all credentials needed to allow access to that resource. As a result, it calls for structuring of policy rules counter-intuitive from access control point of view. For instance, a policy rule may say that for access to the full text of an on-line journal article a requester must satisfy the requirements for browsing the journal's table of contents plus some additional credentials. A rule detailing access to the table of contents could then specify another set of credentials. Even this simple scenario is not intuitive in either formalisms.

Further, constraints that would make policy reasoning non-monotone (such as separation of duties) require to look at more than one rule at a time. So, if the policy is not flat, it has constraints on the credentials that can be presented at the same time or a more complex role hierarchy is used, these systems would not be complete.

If we abstract the above approaches, the only reasoning service used for access control is *deduction* – check whether the request follows from the resource's policy and the presented credentials.

2.1 Intuition 1: Advanced Reasoning Service Abduction

We need another, less known, reasoning service, called *abduction* – check what missing credentials are necessary so that the request can follow from the policy and the presented credentials. Thereupon, we present the basic idea of interactive access control shown in Figure 3.

---

(1) check whether $\mathcal{P}_\mathcal{A}$ and $\mathcal{C}_p$ entail r,
(2) if the check succeeds then `grant` access else
  (a) compute a set $\mathcal{C}_\mathcal{M}$ such that:
    – $\mathcal{P}_\mathcal{A}$ together with $\mathcal{C}_p$ and $\mathcal{C}_\mathcal{M}$ entail r, and
    – $\mathcal{P}_\mathcal{A}$ together with $\mathcal{C}_p$ and $\mathcal{C}_\mathcal{M}$ preserve consistency.
  (b) if $\mathcal{C}_\mathcal{M}$ exists then ask the client for $\mathcal{C}_\mathcal{M}$ and iterate
  (c) else `deny` access.

Fig. 3. Basic Idea of Interactive Access Control

The "compute a set $\mathcal{C}_\mathcal{M}$ such that ..." (step 2a) is exactly the operation of abduction. Section 3 formally defines the basic reasoning services used in the algorithm. An essential part of the abduction reasoning is the computation of a set of missing credentials that is a solution for the request and, at the same time, is consistent with the policy state. The consistency property give us strong guarantees for the missing set of credentials when applying the algorithm on non-monotonic policies. Section 6 examines in details algorithm's properties.

CHALLENGE: *Having abduction as a tool for finding missing credentials we face a new challenge: how do we decide the potential set of missing credentials?*
It is clearly undesirable to disclose all credentials occurring in $\mathcal{P}_\mathcal{A}$ and, therefore, we need a way to define how to control the disclosure of such a set.

Bonatti and Samarati's approach defines that governing access to services is composed in two parts: prerequisite rules and requisite rules. Prerequisite rules specify the requirements that a client should satisfy before being considered for the requirements stated by the requisite rules, which in turn grant access to services. Their approach does not decouple policy disclosure from policy satisfaction, as noted in [Yu and Winslett 2003].

Yu and Winslett [2003] overcome this limitation and propose to treat policies as first class resources, i.e., each policy protecting a resource is considered as a sensitive resource itself whose disclosure is recursively protected by another policy, called meta-policy.

The meta-policy approach is also evidenced in [Bertino et al. 2004]. In this work, each resource is protected by one or more disclosure policies. Each disclosure policy has its policy preconditions and the policy content is revealed if one of the preconditions is satisfied. Each policy precondition has its own policy preconditions that at least one of them must be satisfied before the policy content is released.

The work by [Kapadia et al. 2004] proposes to infer possible alternatives from failed requests (sets of missing credentials) based on the policy scheme of Yu and Winslett. The work uses ordered binary decision diagrams (OBDDs) to represent a resource's policy and its meta-policies. Root of the diagram is the resource itself

and its successors are the policy requirements protecting the resource. The policy requirements have further successors which are the meta-policy requirements and so on. On failure of a request the approach traverses the diagram from the root to its true state (representing grant status) and finds all alternatives that would satisfy the request.

However, the work inherits the monotonicity limitations of Yu and Winslett's settings. The scheme reasons on a single rule specifying a resource's policy and its relevant meta-policies rules in order to find missing credentials satisfying the request. As so, the approach cannot scale to non-monotonic access policies because they require to look at more than one rule (often the entire policy) at any time.

Protecting resources' policies with meta-policies in the access policy itself allows a system to have control on when a policy can be disclosed from when a policy is satisfied.

However, this is not really satisfactory as it does not decouple the decision about access from the decision about disclosure. Resource access is decided by the business logic whereas policy disclosure is due to security and privacy considerations.

2.2 Intuition 2: Disclosure Control Policy

We need *two* policies: one for granting access to one's own resources and one for disclosing the need of foreign (someone else's) credentials. Therefore, we introduce a *security policy for disclosure control* $\mathcal{P}_\mathcal{D}$. The policy for disclosure control identifies the credentials whose need can be potentially disclosed to a client. In other words, $\mathcal{P}_\mathcal{A}$ protects partner's resources by stipulating what credentials a requestor must satisfy to be authorized for a particular resource while, in contrast, $\mathcal{P}_\mathcal{D}$ defines which credentials among those occurring in $\mathcal{P}_\mathcal{A}$ are disclosable (i.e. can be asked) to the requestor.

Figure 4 shows the refined algorithm for interactive access control with controlled disclosure.

---

(1) check whether $\mathcal{P}_\mathcal{A}$ and $\mathcal{C}_p$ entail r,
(2) if the check succeeds then **grant** access else
  (a) compute a set of disclosable credentials $\mathcal{C}_\mathcal{D}$ entailed by $\mathcal{P}_\mathcal{D}$ and $\mathcal{C}_p$,
  (b) compute a set of missing credentials $\mathcal{C}_\mathcal{M}$ out of the disclosable ones ($\mathcal{C}_\mathcal{M} \subseteq \mathcal{C}_\mathcal{D}$) such that
    – $\mathcal{P}_\mathcal{A}$ together with $\mathcal{C}_p$ and $\mathcal{C}_\mathcal{M}$ entail r, and
    – $\mathcal{P}_\mathcal{A}$ together with $\mathcal{C}_p$ and $\mathcal{C}_\mathcal{M}$ preserve consistency.
  (c) if $\mathcal{C}_\mathcal{M}$ exists then ask the client for $\mathcal{C}_\mathcal{M}$ and iterate
  (d) else **deny** access.

Fig. 4. Interactive Access Control with Controlled Disclosure

Yu and Winslett's policy scheme determines whether a client is authorized to be informed of the need to satisfy a given policy. While, in our case, having a separate disclosure control policy allows us to determine whether a client is authorized to see the need of single credentials. Thus we approach a fine-grained disclosure control by defining credentials as single units of disclosure. We can control not only

the disclosure of (entire) policies as a single unit but also the disclosure of single credentials composing those policies.

Let us look at Yu and Winslett's own example [Yu and Winslett 2003, page 4].

EXAMPLE 2.1. *(McKinley Clinic)*

[Access Scenario] McKinley clinic makes its patient records available for online access. Let r be Alice's record. To gain access to r a requester must either present Alice's patient ID for McKinley clinic ($C_{AliceID}$), or present a California social worker license (C_{CSWL}) and a release-of-information credential (C_{RoI}) issued to the requester by Alice.

[Disclosure Scenario] Alice wants to keep the latter constraint inaccessible to strangers as it may help them to infer that Alice has mental or emotional problem. However, employees of McKinley clinic ($C_{McKinleyEmployee}$) should be allowed to see the contents of this policy.

[Security Policy]

$\mathcal{P}$:
- $r : P$
- $P \leftrightarrow P_1 \vee P_2$ *and* P : *true*
- $P_1 \leftrightarrow C_{AliceID}$ *and* P_1 : *true*
- $P_2 \leftrightarrow C_{CSWL} \wedge C_{RoI}$ *and* P_2 : *true*
- $P_2 : P_3$
- $P_3 \leftrightarrow C_{McKinleyEmployee}$ *and* P_3 : *true*

P, P_1, P_2 and P_3 are policy identifiers. r and P_2 are protected sensitive resources. □

EXAMPLE 2.2. *(Example 2.1 formalized as two logic programs)*

$\mathcal{P}_\mathcal{A}$:
- $r \leftarrow C_{AliceID}.$
- $r \leftarrow C_{CSWL}, C_{RoI}.$

$\mathcal{P}_\mathcal{D}$:
- $C_{AliceID}.$
- $C_{CSWL} \leftarrow C_{McKinleyEmployee}.$
- $C_{RoI} \leftarrow C_{McKinleyEmployee}.$

$\mathcal{P}_\mathcal{A}$ states that access to r is granted either to Alice or to California social workers that have a release-of-information credential issued by Alice.

$\mathcal{P}_\mathcal{D}$ states that the disclosure of Alice's ID is not protected and potentially released to anybody. The need for credentials California social worker license C_{CSWL} and release-of-information C_{RoI} is disclosed only to users who have already presented their McKinley employee certificate $C_{McKinleyEmployee}$.

We point out that having $C_{McKinleyEmployee}$ does not allow access to r but rather unlocks more information on how to access r. □

A question that would come up is whether the disclosure policy is resource aware, i.e. whether we could have an association between resources and credentials protecting resources within $\mathcal{P}_\mathcal{D}$. For example, let us have a second resource r_2 protected by credential C_{CSWL} the disclosure of which does not depend on the presence of $C_{McKinleyEmployee}$, but which is the case for r.

To make the interactive algorithm behave as expected we have to include the service request as a fact when computing the disclosable credentials. Thus, step 2a becomes "compute a set of disclosable credentials $\mathcal{C}_\mathcal{D}$ entailed by $\mathcal{P}_\mathcal{D}$, r and $\mathcal{C}_p$."

In the rest of the paper, without lost of generality, we assume that the disclosure policy is resource aware and whenever $\mathcal{P}_\mathcal{D}$ is involved it implies $\mathcal{P}_\mathcal{D} \cup \{r\}$.

To control the disclosure of resources' policies as single units one can also accommodate the notion of policy identifiers as in [Yu and Winslett 2003] (refer also to

Example 2.1). Thus, taking a full advantage of that, one can transform the unified policy schemes of Yu and Winslett or Bertino et al. [2004] to the two-policy model.

2.2.1 *Stepwise Disclosure Control.* Having a separate disclosure policy allows us to have an additional reasoning on the policy that helps us to disclose credentials in a stepwise fashion. The basic intuition is that the logical policy structure itself tells us which credentials must be disclosed to obtain the information that other credentials are missing. The model presented in [Koshutanski and Massacci 2007] extends the interactive access control algorithm with an additional functionality of computing stepwise sets of missing credentials (by observing $\mathcal{P}_\mathcal{D}$) and requesting them gradually until a solution is agreed that grants a resource.

The example below summarizes the policy disclosure functionalities we have presented so far.

EXAMPLE 2.3. *(Stepwise disclosure control)*

$\mathcal{P}_\mathcal{A}$:
$r \leftarrow C_{AliceID}.$
$r \leftarrow C_{CSWL}, C_{RoI}.$
$r_2 \leftarrow C_{CSWL}.$

$\mathcal{P}_\mathcal{D}$:
$C_{AliceID} \leftarrow r.$
$C_{McKinleyEmployee} \leftarrow r.$
$P_1 \leftarrow C_{McKinleyEmployee}, r.$
$C_{RoI} \leftarrow P_1.$
$C_{CSWL} \leftarrow P_1.$
$C_{CSWL} \leftarrow r_2.$

$\mathcal{P}_\mathcal{D}$ states that the disclosure of Alice's ID is potentially released to anybody having requested resource r. The need for credential $C_{McKinleyEmployee}$ is released to anybody requested r. P_1 is a policy identifier encapsulating disclosure of C_{CSWL} and C_{RoI}. The need for credentials C_{CSWL} and C_{RoI} is disclosed only to those who have already presented $C_{McKinleyEmployee}$ and requested resource r. Alternatively, the need for C_{CSWL} is released to those clients requested r_2. □

Now, using the stepwise approach the interactive algorithm will return the need of either $C_{AliceID}$ or $C_{McKinleyEmployee}$ when r is requested. The algorithm would further disclose the need for C_{CSWL} and C_{RoI} only if the client has presented $C_{McKinleyEmployee}$. As already discussed, this holds for the policy settings in [Yu and Winslett 2003].

However, if we add the two rules $\{C_{RoI} \leftarrow r.\ C_{CSWL} \leftarrow C_{RoI}, r.\}$ to $\mathcal{P}_\mathcal{D}$ then we are able to provide another solution to the example problem that is not intuitive in the policy scheme [Yu and Winslett 2003]. We allow a client to know for the need of release-of-information credential issued by Alice without revealing the need of C_{CSWL} (inferring Alice has a mental problem). On the other side, C_{CSWL} is disclosed only if a client has C_{RoI} presented to the system. Thus ensuring only clients evidenced by Alice will know the need for C_{CSWL} required to get access to r.

EXAMPLE 2.4. *(Rental car service [Bertino et al. 2004] page 832)*

$\mathcal{P}$:
$pol_1 = (\{\}, Rental\_Car \leftarrow C_{corrier\_employee}, C_{ID\_card}).$
$pol_2 = (\{\}, Rental\_Car \leftarrow C_{driving\_license}).$
$pol_3 = (\{pol_2\}, Rental\_Car \leftarrow C_{credit\_card}).$
$pol_4 = (\{pol_3, pol_1\}, Rental\_Car \leftarrow DELIV).$

Policy pol_4 says that either pol_3 or pol_1 must be satisfied before granting (delivering) the service $Rental\_Car$. pol_3 states that the release of the need for a credential for

a credit card will be shown to those who satisfy pol_2. pol_2 has no preconditions and releases its content to anybody, i.e. the need for a driving license credential. pol_1 has also no preconditions and discloses the need for an employee certificate at Corrier company and credential for an ID card potentially to any client.

Below we show how the rental car service can be formalized as two logic programs so that using the stepwise approach one can achieve the same policy protection functionality.

EXAMPLE 2.5.

$\mathcal{P}_{\mathcal{A}}$:
$Rental\_Car \leftarrow C_{driving\_license}, C_{credit\_card}.$
$Rental\_Car \leftarrow C_{corrier\_employee}, C_{ID\_card}.$

$\mathcal{P}_{\mathcal{D}}$:
$C_{corrier\_employee} \leftarrow Rental\_Car.$
$C_{ID\_card} \leftarrow Rental\_Car.$
$C_{driving\_license} \leftarrow Rental\_Car.$
$C_{credit\_card} \leftarrow C_{driving\_license}, Rental\_Car.$

If we consider dynamic environments where the network (privacy) settings are continuously changing one can even define a set of disclosure policies each corresponding to a particular network/system state. For example a disclosure policy may consider a set of environment factors like time, work loads or other system conditions that could enforce additional disclosure control. These dynamic conditions are likely to be the case in autonomic communication networks. Once the access control algorithm is run it could dynamically select a disclosure policy best suited current system state. Having multiple disclosure policies for particular access control requirements is not scalable in the scheme of [Yu and Winslett 2003; Bertino et al. 2004] because they tie access and disclosure requirements in one security policy setting.

2.3 Intuition 3: Extension to Automated Trust Negotiation

A question that would come up when introducing the interactive access control is what happens on the client side once the computed missing credentials are requested by a server. The work in [Koshutanski and Massacci 2007] extends the interactive model to function on client and server sides. The intuition is that by mirroring the access control algorithm on the client side, the client is also able to abduce what missing credentials are needed (to be requested to a server) in order to disclose its own credentials.

The work proposes a negotiation scheme that builds on top of the interactive algorithm a negotiation protocol. The negotiation protocol allows two entities in a network to mutually establish sufficient access rights needed to grant a resource. The protocol runs on two sides so that entities understand each other and automatically interoperate.

This paper examines in details the guarantees the interactive model provides when applied to monotonic and non-monotonic access policies. So one could run a trust negotiation over a non-monotonic policy domain and still guarantee completeness and correctness of the negotiation process.

Since the negotiation protocol is driven by abduction and deduction reasonings one can accommodate negotiation strategies on top of it so that missing sets of

requirements are released (negotiated) according to some high-level goals. The work by [Baselice et al. 2007] proposes a general framework for specifying high-level negotiation strategies abstracting from a specific policy language.

3. POLICY SYNTAX AND SEMANTICS

Policies are written as normal logic programs [Apt 1990]. These are sets of *rules* of the form:

$$A \leftarrow B_1, \ldots, B_n,\ not\, C_1, \ldots,\ not\, C_m \tag{1}$$

where A, B_i and C_i are (possibly ground) predicates. A is called the *head* of the rule, each B_i is called a *positive literal* and each $not\, C_j$ is a *negative literal*, whereas the conjunction of the B_i and $not\, C_j$ is called the *body* of the rule. If the body is empty the rule is called a *fact*. A normal logic program is a set of rules.

In our framework, we also need *constraints* that are rules with an empty head.

$$\leftarrow B_1, \ldots, B_n,\ not\, C_1, \ldots,\ not\, C_m \tag{2}$$

One of the most prominent semantics for normal logic programs is the *stable model semantics* proposed by Gelfond and Lifschitz [1988]. The intuition is to interpret the rules of a program P as constraints on a solution set S (a set of ground atoms) for the program itself. So, if S is a set of atoms, rule (1) is a constraint on S stating that if all B_i are in S and none of C_j are in it, then A must be in S. A constraint (2) is used to rule out from the set of acceptable models situations in which all B_i are true and all C_j are false.

Below we give the formal definitions for the basic reasoning services.

DEFINITION 3.1 LOGICAL CONSEQUENCE AND CONSISTENCY. *Let $\mathcal{P}$ be a logic program and L be a positive ground literal. L is a* logical consequence *of $\mathcal{P}$, symbolically $\mathcal{P} \models L$, if L is true in every stable model of $\mathcal{P}$. $\mathcal{P}$ is* consistent, *$\mathcal{P} \not\models \bot$, if there is a stable model for $\mathcal{P}$.*

DEFINITION 3.2 SECURITY CONSEQUENCE. *A request r is a security consequence of a policy $\mathcal{P}$ if (i) $\mathcal{P}$ is logically consistent and (ii) r is a logical consequence of $\mathcal{P}$.*

DEFINITION 3.3 ABDUCTION. *Let $\mathcal{P}$ be a logic program, H a set of ground atoms (called hypotheses or abducibles), L a ground literal (called observation), and $\prec$ a partial order over subsets of H. A solution of the abduction problem $\langle L, H, \mathcal{P}\rangle$ is a set of ground atoms E such that:*

(i) $E \subseteq H$,
(ii) $\mathcal{P} \cup E \models L$,
(iii) $\mathcal{P} \cup E \not\models \bot$,
(iv) any set $E' \prec E$ does not satisfy all conditions above.

Traditional partial orders are subset containment or set cardinality.

DEFINITION 3.4 SOLUTION SET FOR A RESOURCE. *Let $\mathcal{P}$ is an access policy and r be a resource. A set of credentials $\mathcal{C}_S$ is a* solution set *for r according to $\mathcal{P}$ if r is a security consequence of $\mathcal{P}$ and $\mathcal{C}_S$, i.e. $\mathcal{P} \cup \mathcal{C}_S \models r$ and $\mathcal{P} \cup \mathcal{C}_S \not\models \bot$.*

DEFINITION 3.5 MONOTONIC AND NON-MONOTONIC POLICY. *A policy $\mathcal{P}$ is* monotonic *if whenever a set of statements $\mathcal{C}$ is a solution set for r according to $\mathcal{P}$ ($\mathcal{P} \cup \mathcal{C} \models r$) then any superset $\mathcal{C}' \supset \mathcal{C}$ is also a solution set for r according to $\mathcal{P}$ ($\mathcal{P} \cup \mathcal{C}' \models r$).*

In contrast, a non-monotonic *policy is a logic program in which if $\mathcal{C}$ is a solution for r it may exists $\mathcal{C}' \supset \mathcal{C}$ that is not a solution for r, i.e. $\mathcal{P} \cup \mathcal{C}' \not\models r$*

DEFINITION 3.6 RESOURCE r ADDITIVE POLICY. *A policy $\mathcal{P}$ is* a resource r additive *if for any two solution sets $\mathcal{C}_\mathcal{S}$ and $\mathcal{C}_\mathcal{S}'$ for r where $\mathcal{C}_\mathcal{S} \not\subset \mathcal{C}_\mathcal{S}'$ and $\mathcal{C}_\mathcal{S}' \not\subset \mathcal{C}_\mathcal{S}$ then also $\mathcal{C}_\mathcal{S} \cup \mathcal{C}_\mathcal{S}'$ is a solution set for r according to $\mathcal{P}$.*

DEFINITION 3.7 RESOURCE r SUBSET CONSISTENT POLICY. *A policy $\mathcal{P}$ is a* resource r subset consistent *if for every solution set $\mathcal{C}_\mathcal{S}$ for r it holds that any $\mathcal{C} \subseteq \mathcal{C}_\mathcal{S}$ preserves consistency in $\mathcal{P}$, i.e. $\mathcal{P} \cup \mathcal{C} \not\models \bot$.*

DEFINITION 3.8 WELL-BEHAVED POLICY. *A policy $\mathcal{P}$ is* well-behaved *if for all resources $r \in \mathcal{P}$*

(i) $\mathcal{P}$ is resource r additive *and*
(ii) $\mathcal{P}$ is resource r subset consistent.

Section 6 shows how the interactive access control model guarantees completeness and correctness when applied on well-behaved policies.

4. INTERACTIVE ACCESS CONTROL ALGORITHM

Below we summarize all the information we needed to state the interactive access control algorithm, shown in Figure 5.

$\mathcal{P}_\mathcal{A}$ – security policy governing access to resources,

$\mathcal{P}_\mathcal{D}$ – security policy controlling the disclosure of foreign (missing) credentials,

$\mathcal{C}_p$ – set of credentials presented by a client in a single interaction,

$\mathcal{C}_\mathcal{P}$ – set of active credentials presented by a client during an interactive access control process,

$\mathcal{C}_\mathcal{N}$ – set of credentials that a client has declined to present during an interactive process.

When a client initially requests a service the server creates a client's profile corresponding to a new session. The client's profile consists of $\mathcal{C}_\mathcal{P}$ and $\mathcal{C}_\mathcal{N}$ set up to empty sets initially. A client requests for a service by submitting a request r and a set of presented credentials $\mathcal{C}_p$. $\mathcal{C}_p$ is optional and could be an empty set.

Steps 1 and 2 update client's profile with newly presented credentials as follows. Active credentials $\mathcal{C}_\mathcal{P}$ are updated with $\mathcal{C}_p$. Declined credentials $\mathcal{C}_\mathcal{N}$ are updated with the the missing credentials ($\mathcal{C}_\mathcal{M}$) the client was asked in the last interaction set difference with the newly presented ones. $\mathcal{C}_\mathcal{M}$ is set up to an empty set initially.

Once the profile is updated, the algorithm checks whether the request r is granted by $\mathcal{P}_\mathcal{A}$ according to $\mathcal{C}_\mathcal{P}$ (step 3). If the the client does not have enough access rights then the algorithm computes all credentials disclosable from $\mathcal{P}_\mathcal{D}$ according to $\mathcal{C}_\mathcal{P}$ and from the resulting set removes all already declined and presented credentials (step 4a). The latter step is used to avoid dead loops of asking something already declined or presented.

Input: r, $\mathcal{C}_p$;
Output: grant/deny/ask($\mathcal{C}_\mathcal{M}$);

(1) update $\mathcal{C}_\mathcal{P} = \mathcal{C}_\mathcal{P} \cup \mathcal{C}_p$,
(2) update $\mathcal{C}_\mathcal{N} = \mathcal{C}_\mathcal{N} \cup (\mathcal{C}_\mathcal{M} \setminus \mathcal{C}_p)$, where $\mathcal{C}_\mathcal{M}$ is from the last interaction,
(3) verify whether the request r is a security consequence of $\mathcal{P}_\mathcal{A}$ and $\mathcal{C}_\mathcal{P}$, namely $\mathcal{P}_\mathcal{A} \cup \mathcal{C}_\mathcal{P} \models r$ and $\mathcal{P}_\mathcal{A} \cup \mathcal{C}_\mathcal{P} \not\models \bot$,
(4) if the check succeeds then return ***grant*** else
  (a) compute a set of disclosable credentials $\mathcal{C}_\mathcal{D}$ as
  $\mathcal{C}_\mathcal{D} = \{c \mid c \text{ credential that } \mathcal{P}_\mathcal{D} \cup \mathcal{C}_\mathcal{P} \models c\} \setminus (\mathcal{C}_\mathcal{N} \cup \mathcal{C}_\mathcal{P})$,
  (b) compute a minimal set of missing credentials $\mathcal{C}_\mathcal{M} \subseteq \mathcal{C}_\mathcal{D}$ such that
  – $\mathcal{P}_\mathcal{A} \cup \mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{M} \models r$ and
  – $\mathcal{P}_\mathcal{A} \cup \mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{M} \not\models \bot$,
  (c) if $\mathcal{C}_\mathcal{M}$ exists then return ***ask($\mathcal{C}_\mathcal{M}$)*** and iterate,
  (d) else return ***deny***.

Fig. 5. Interactive Access Control Algorithm

Next, the algorithm computes all subsets of $\mathcal{C}_\mathcal{D}$ that are consistent with $\mathcal{P}_\mathcal{A}$ and satisfy r. Out of all these sets (if any) the algorithm selects the minimal one (step 4c) to be asked to a client.

REMARK 4.1. *Using declined credentials is essential to avoid dead loops in the process and to guarantee successful interactions in presence of disjunctive information.*

We point out that minimality criteria play an important role when selecting a missing set of credentials as we shall see in the example below. Set cardinality criterion does not fully apply mainly because credential sensitiveness plays the more important role than the cardinality. The role minimality criterion is more adequate when dealing with role-based access control models. If we have attribute-based access control one can associate values to attributes (e.g. level of sensitiveness) so that can perform minimality reasoning on them. Additionally, one can accommodate sequences of criteria depending on particular access control settings.

In cases of more than one equally minimal solutions computed the algorithm picks one of them as a solution to be asked to a client. However, one can slightly modify the algorithm so that it returns all equally minimal solutions as disjunctive information. Then, on next interaction the declined credentials are computed as the union of all missing sets asked in the last interaction set difference with presented ones in the current step.

5. A COMPREHENSIVE EXAMPLE

Let us consider a shared overlay network Planet-Lab between Italian universities and German research institutions. For the sake of simplicity assume that there are three main access types to resources: *disk* – read access to data residing on Planet-Lab machines; *run* – read access to data and run processes on the machines; and *conf* – configure access to data including the previous two types of access plus the possibility of configuring network services on machines.

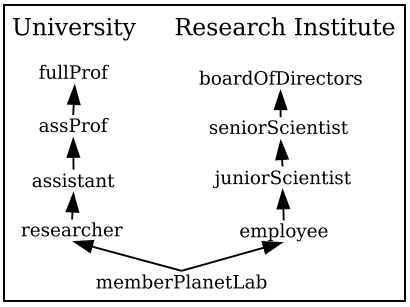


Fig. 6. Planet-Lab hierarchy model

Figure 6 shows the hierarchy and granularity of roles considered for universities and research institutions, respectively. The partial order of roles is indicated by arcs, where higher the role in the hierarchy, more powerful it is. A role dominates another role if it is higher in the hierarchy and there is a direct path between them.

The access policy of the Planet-Lab network specifies that:

- `disk` access is allowed to any role of the Planet-Lab hierarchy,
- `run` access is allowed to any employee or higher roles at a German research institute, or to any researcher or higher roles at an Italian university,
- `conf` access is allowed to junior scientists or higher roles at a German research institute, or to any assistant or higher roles at an Italian university.

There is a preprocessing step that validates and transforms certificates to predicates suitable for the formal model: credential $(HolderID, AttrName, IssuerID)$ if attribute certificate; certificate $(SubjectID, IssuerID)$ if identity certificate. There is a mapping from the trusted public keys of SOAs (Source of Authorities) or CAs (Certificate Authorities) to their internal policy identifiers represented by $IssuerID$ value. Using a second predicate IssuerType $(IssuerID)$ one can classify what SOAs and CAs are considered trusted to issue particular type of attributes and identity certificates.

We represent variables staring with a capital letter (e.g. Holder, Attr, Issuer) while constants starting with a small case letter (e.g., planetLab_Class1SOA, accredited, juniorScientist). A variable indicates any value in its field and is valid within the rule it appears.

Figure 7 shows the formalization of Planet-Lab access and disclosure policies. Following is the functional explanation of the policies.
Access policy:

—Rule (1) defines the trusted SOA issuing Planet-Lab membership certificates. Rule (2) defines the trusted SOA accrediting German research institutes, rule (3) defines the trusted SOA accrediting Italian universities. Rules (4) and (5) specify trusted CAs certifying identities of Italian universities and German research institutions, respectively.

—Rule (6) grants `disk` access to the shared network to any entity (holder *Hol*) presented *memberPlanetLab* role credential issued by the trusted Planet-Lab SOA.

—Rule (7) grants `disk` access to anybody who has `run` access permission.

Access Policy:

(1) issuerPlanetLab ($planetLab\_Class1SOA$).
(2) issuerAccredInstDE ($deutschAkkred\_Class1SOA$).
(3) issuerAccredUnivIT ($crui\_class1SOA$).
(4) issuerUnivIT ($govitaliane\_class1CA$).
(5) issuerInstDE ($govdeutsch\_class1CA$).
(6) assign ($disk$) $\leftarrow$ credential ($Hol, memberPlanetLab, Iss$), issuerPlanetLab ($Iss$).
(7) assign ($disk$) $\leftarrow$ assign (run).
(8) assign (run) $\leftarrow$ credential ($Hol, Attr, Univ$), $Attr \succeq researcher$, certificate ($Univ, IssUniv$), issuerUnivIT ($IssUniv$), credential ($Univ, accredited, IssAcc$), issuerAccredUnivIT ($IssAcc$).
(9) assign (run) $\leftarrow$ credential ($Hol, Attr, Inst$), $Attr \succeq employee$, certificate ($Inst, IssInst$), issuerInstDE ($IssInst$), credential ($Inst, accredited, IssAcc$), issuerAccredInstDE ($IssAcc$).
(10) assign (run) $\leftarrow$ assign ($conf$).
(11) assign ($conf$) $\leftarrow$ credential ($Hol, Attr, Univ$), $Attr \succeq assistant$, certificate ($Univ, IssUniv$), issuerUnivIT ($IssUniv$), credential ($Univ, accredited, IssAcc$), issuerAccredUnivIT ($IssAcc$).
(12) assign ($conf$) $\leftarrow$ credential ($Hol, Attr, Inst$), $Attr \succeq juniorScientist$, certificate ($Inst, IssInst$), issuerInstDE ($IssInst$), credential ($Inst, accredited, IssAcc$), issuerAccredInstDE ($IssAcc$).

Disclosure Policy:

(1) credential ($Hol, memberPlanetLab, planetLab\_Class1SOA$).
(2) credential ($Hol, Attr, Inst$) $\leftarrow Attr \succeq employee$.
(3) credential ($Hol, Attr, Univ$) $\leftarrow Attr \succeq researcher$.
(4) certificate ($Univ, govitaliane\_class1CA$).
(5) certificate ($Inst, govdeutsch\_class1CA$).
(6) credential ($Univ, accredited, crui\_class1SOA$).
(7) credential ($Inst, accredited, deutschAkkred\_class1SOA$).

Fig. 7. Planet-Lab Access and Disclosure Control Policies

—Rule (8) grants `run` access to any holder of an attribute higher or equal to a researcher position issued by an Italian university. To validate an italian university, Planet-Lab policy requires two additional certificates: an identity certificate identifying the university entity as a legal key holder and an attribute certificate attesting the university entity as accredited Italian university. The former case is validated by the two predicates certificate ($Univ, IssUniv$) and issuerUnivIT ($IssUniv$) while the latter case by the two predicates credential ($Univ, accredited, IssAcc$) and issuerAccredUnivIT ($IssAcc$). Together all the predicates in the body of rule (8) validate an Italian university and implicitly delegate the right to it to state who has what position at the university.

—Rule (9) grants `run` access to any holder of a credential certificate with an attribute role higher or equal to an employee position at any German research institute. Planet-Lab policy requires two additional certificates to validate a research institution: an identity certificate identifying the institution as a legal key holder and an attribute certificate attesting the institution as accredited one.

—Rule (10) grants `run` access to anybody who has `conf` access permission.

—Rule (11) grants `conf` access to any holder of a credential with a position equal or higher to assistant and issued by an Italian university. University validation follows analogously of rule (8).

—Rule (12) grants `conf` access to any holder of a credential certificate with a role higher or equal to a junior scientist position issued by a German research

institute. Institute validation follows analogously of rule (9).

Disclosure policy:

—Rule (1) discloses the need for a Planet-Lab membership credential and specifies the intended credential issuer. Rule (2) discloses the need for credentials attesting employee, junior scientist, senior scientist and board of directors, respectively. Rule (3) discloses the need for credentials attesting roles researcher, assistant, associate and full professor, respectively.

—Rule (4) discloses the need for a certificate identifying Italian university entities and specifies the intended certificate issuer. Rule (5) discloses the need for a certificate identifying German research institutes and specifies the intended certificate issuer.

—Rule (6) discloses the need for a credential certifying Italian universities as accredited institutions and specifies the intended credential issuer. Rule (7) discloses the need for a credential certifying German institutions as accredited and specifies the intended credential issuer.

Access Control Scenario 1. Alice is a senior scientist at Fraunhofer institute in Berlin. She has been issued two certificates one for an employee at the research institute and another one attesting her as a senior scientist, either of them issued by a Fraunhofer certificate authority.

Now, Alice wants to run a service located at the Planet-Lab network. For doing so she presents her employee certificate at access time

$$\mathsf{credential}\,(alice\_milburk, employee, fraunhofer\_Inst\_Berlin)$$

presuming it is enough as she knows that Planet-Lab is a joint network between German and Italian institutions.

According to the access policy (rule 9) any employee at a German research institute is allowed *run* access to the network but additionally there must be presented certificates identifying Fraunhofer as a legal key holder and a attesting Fraunhofer as a legal (accredited) German research institute.

Alice's credentials are not enough to get `run` access and the request would be denied. Then, the interactive algorithm (step 4a) computes the set of disclosable credentials as all credentials disclosed from rules (1) to (7) of the disclosure policy set difference with Alice's presented credentials, i.e. credential for an employee.

Next, abduction computation (subset minimal) finds the following missing set that satisfies the request:

$$\{\mathsf{certificate}\,(Inst, govdeutsch\_class1CA),\\ \mathsf{credential}\,(Inst, accredited, deutschAkkred\_class1SOA)\}$$

There is a post processing step that maps the internal policy identifiers of SOAs and CAs (like *govdeutsch_class1CA*) to their high level description that is to be returned back to the client.

Alice receives the missing set of credentials, then she consults the Fraunhofer authority that issued her employee certificate in order to obtain the missing credentials. Next interaction, she requests the service presenting the missing set of credentials and the system grants her access.

Access Control Scenario 2. Alice wants to configure an online system for paper submissions of a workshop. She submits her employee certificate together with the two certificates identifying Fraunhofer institute as legitimate key holder and as accredited institution. Formally speaking, the initial set of credentials is:

$\{\mathsf{credential}\,(alice\_milburk, employee, fraunhofer\_Inst\_Berlin),$
$\mathsf{certificate}\,(fraunhofer\_Inst\_Berlin, govdeutsch\_class1CA),$
$\mathsf{credential}\,(fraunhofer\_Inst\_Berlin, accredited, deutschAkkred\_class1SOA)\}$

Looking at the access policy rule (12), configure access is allowed to junior scientists or higher role positions. The algorithm computes the set of disclosable credentials and out of them removes all what has been already presented by Alice. Next, abduction reasoning finds the following sets of missing credentials:

$\{\mathsf{credential}\,(Hol, juniorScientist, Inst)\}$
$\{\mathsf{credential}\,(Hol, seniorScientist, Inst)\}$
$\{\mathsf{credential}\,(Hol, boardOfDirectors, Inst)\}$

Now, using role minimality criterion the algorithm selects the set $\{\mathsf{credential}\,(Hol, juniorScientist, Inst)\}$ as the minimal one and returns it back to the client.

Since Alice is a senior scientist she declines to present the requested credential and returns the access request but with no entry for presented credentials. The algorithm updates Alice's profile marking the requested credential as declined. The difference comes when the algorithm recomputes the disclosable credentials as all disclosable credentials from the disclosure policy set difference Alice's presented and declined credentials. Out of those abduction finds the following missing sets:

$\{\mathsf{credential}\,(Hol, seniorScientist, Inst)\}$
$\{\mathsf{credential}\,(Hol, boardOfDirectors, Inst)\}$

The algorithm selects the need for senior scientist and returns it back to Alice. On next interaction, Alice presents her senior scientist credential and gets granted the service request.

The interactive steps in access scenario 1 can be leveraged by two ways. First one by using credential chain discovery algorithm as a pre-processing step to the interactive algorithm, i.e. discovering all relevant credentials before getting a decision. Second way is by using a post-processing step that returns to a client the need for credentials only relevant to subject attributes and gathering the remaining credentials directly from predefined certificate authorities' public repositories. The second approach outlines potential future work on extending the model to automatic credential discovery.

6. ACCESS CONTROL GUARANTEES

We define below the main guarantees the access control framework provides.

DEFINITION 6.1 SOUNDNESS. *If a client gets grant then he has a solution for the request.*

DEFINITION 6.2 COMPLETENESS. *If a client has a solution for a service then he gets grant the service.*

To prove the guarantees first we have to look at the policies underlying our model and especially what would be reasonable access and disclosure policies that support the above guarantees.

DEFINITION 6.3 FAIR ACCESS. *Let $\mathcal{P}_\mathcal{A}$ be an access control policy and let $\mathcal{C}_{\mathcal{P}_\mathcal{A}}$ be the set of ground instances of all credentials occurring in $\mathcal{P}_\mathcal{A}$. The policy $\mathcal{P}_\mathcal{A}$* guarantees fair access *if for any request r there exists a set $\mathcal{C}_\mathcal{S} \subseteq \mathcal{C}_{\mathcal{P}_\mathcal{A}}$ that is a solution for r.*

DEFINITION 6.4 FAIR INTERACTION. *Let $\mathcal{P}_\mathcal{A}$ and $\mathcal{P}_\mathcal{D}$ be access and disclosure control policies, respectively. The policies* guarantee fair interaction *if*

(1) $\mathcal{P}_\mathcal{A}$ guarantees fair access and
(2) if $\mathcal{C}_\mathcal{S}$ is a solution for a request r then $\mathcal{C}_\mathcal{S}$ is disclosable by $\mathcal{P}_\mathcal{D}$, i.e. $\forall c \in \mathcal{C}_\mathcal{S}$, $\mathcal{P}_\mathcal{D} \models c$.

The intuition of fair interaction is that any solution for a request should be potentially visible to clients. This property essentially defines the ability of granting services to (good) clients.

The property itself does not imply that a service disclosure policy is trivial, rather it is evident in all trust negotiation models: if there is a policy protecting a resource then the policy (requirements) should be negotiated with an opponent in order to provide access. How the policy (a solution set) is negotiated (disclosed) is a matter of concrete trust negotiation settings/strategies.

On top of the property one can protect a solution set by means of stepwise disclosure control (ref. [Koshutanski and Massacci 2007]) or by means of hidden credentials described later in the section.

One would argue that if fair interaction then why not simply request all credentials allowed by the disclosure policy instead of abducing missing ones. First, the disclosure policy controls the disclosure of all resources' policies under a partner's domain and so we need to abduce only the relevant information (credentials). Second, by abducing the missing credentials we ensure that at any moment the missing set of credentials is an actual solution, i.e. it is consistent with the access policy state.

We make the following policy assumptions.

REMARK 6.1 POLICY ASSUMPTIONS. *Hereinafter all $\mathcal{P}_\mathcal{A}$ are* well-behaved *policies and all $\mathcal{P}_\mathcal{D}$ are* monotonic *policies.*

The well-behaved property ensures that if we have two solutions for a service we can "add" them and we will still get the service, and if we have a solution for a service any subset of this solution is consistent with the policy. With the latter requirement we avoid situations where lack of information makes a policy inconsistent.

The set of well-behaved policies reside between monotonic and arbitrary policies.

PROPOSITION 6.1. *All monotonic policies are well-behaved but the converse is not true.*

PROOF. In one direction the property is immediate (refer to Definition 3.5). For the other direction we show a counter-example:

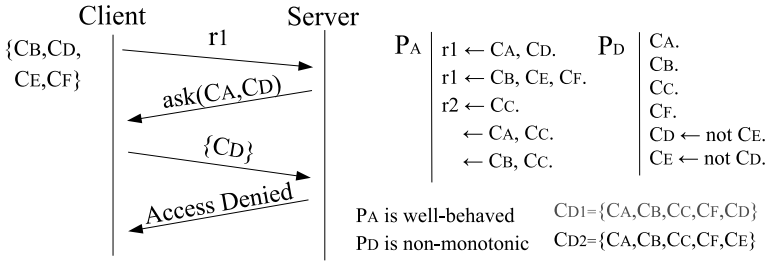
$$r_1 \leftarrow C_A.$$

$$
\begin{aligned}
r_1 &\leftarrow C_B.\\
r_2 &\leftarrow C_C.\\
&\leftarrow C_A, C_C.\\
&\leftarrow C_B, C_C.
\end{aligned}
$$

In our case having $\{C_A, C_B, C_C\}$ bans the client to get either of the services, which clearly shows that the example is a non-monotonic policy. At the same time, for each of the services we have additive and subset consistent properties so that the policy is well-behaved. □

Monotonicity on $\mathcal{P}_\mathcal{D}$ ensures that if a solution is visible to a client then that solution should remain visible during interaction steps. Consider the following example.

EXAMPLE 6.1 NON-MONOTONIC DISCLOSURE.



There are two sets of disclosable credentials and either of them contains a solution set for r_1. If the interactive algorithm selects $\mathcal{C}_{\mathcal{D}1}$ then the solution abduction finds is $\{C_A, C_D\}$. On the next interaction step the client supplies C_D (as he does not have in possession C_A) and gets denial because the presence of C_D bans the disclosure of C_E and abduction cannot find any solution in the new set of disclosable credentials. □

Let us look at a client side and define what would be a reasonable client that our framework aims to provide the guarantees for.

DEFINITION 6.5 COOPERATIVE CLIENT. *A client with a set of credentials (ability) $\mathcal{C}$ is a* cooperative client *if whenever receives* `ask`$(\mathcal{C}_\mathcal{M})$ *returns* $\mathcal{C}_\mathcal{M} \cap \mathcal{C}$.

The definition captures the practical and intuitive aspect of client's behavior: A client who has the right set of credentials and who is willing to send them to a server will be granted access. We notice that it is fairly difficult to model and prove any results for non-cooperative clients.

The work in [Koshutanski and Massacci 2007] empowers cooperative clients with a negotiation model that allows them to negotiate with a server additional requirements before presenting own credentials. Thus, a client and a server become cooperative on those set of credentials on which they have mutually satisfiable requirements.

We assume that a client initiates a service request with an empty set of presented credentials. The assumption is important to avoid initial inconsistency and to ensure a successful first step.

THEOREM 6.1 SOUNDNESS. *Let $\mathcal{P}_\mathcal{A}$ be an access policy, $\mathcal{P}_\mathcal{D}$ be a disclosure policy and r a request. If a client gets grant r with the access control algorithm then he has a solution set $\mathcal{C}_\mathcal{S}$ that unlocks r according to $\mathcal{P}_\mathcal{A}$.*

PROOF. Refer to Appendix A. □

THEOREM 6.2 TERMINATION. *The access control algorithm always terminates.*

PROOF. Refer to Appendix A. □

THEOREM 6.3 COMPLETENESS FOR A COOPERATIVE CLIENT. *Let $\mathcal{P}_\mathcal{A}$ be an access policy, $\mathcal{P}_\mathcal{D}$ be a disclosure policy and r a request. If $\mathcal{P}_\mathcal{A}$ and $\mathcal{P}_\mathcal{D}$ guarantee fair access and interaction then if a cooperative client has a set of credentials $\mathcal{C}_\mathcal{S}$ that is a solution for r according to $\mathcal{P}_\mathcal{A}$ then the client always gets grant r with the access control algorithm.*

PROOF. Refer to Appendix A. □

DEFINITION 6.6 DISCLOSABLE AND HIDDEN CREDENTIALS. *Let $\mathcal{P}_\mathcal{D}$ be a disclosure policy, credential c is* disclosable *if there is a set of credentials $\mathcal{C}$ such that $c \notin \mathcal{C}$ and $\mathcal{C}$ together with the disclosure policy $\mathcal{P}_\mathcal{D}$ entails c, namely $\mathcal{P}_\mathcal{D} \cup \mathcal{C} \models c$. A credential c is* hidden *if it is not disclosable.*

The intuition behind hidden credentials is that the system *does not ask* for them but *expects* them from clients. So, the information for hidden credentials is obtained by out-of-band sources. Hidden credentials are used either to unlock more credentials needed to grant access or used directly to unlock a resource. So, a client must provide them when initially requests for a service.

DEFINITION 6.7 HIDDEN CREDENTIALS FOR A RESOURCE r. *Let $\mathcal{P}_\mathcal{A}$ be an access control policy and $\mathcal{P}_\mathcal{D}$ be a disclosure control policy. A* set of hidden credentials for a resource r *is the set $\mathcal{C}_\mathcal{H}$ such that:*

(1) there exists a solution set $\mathcal{C}_\mathcal{S}$ for r according to $\mathcal{P}_\mathcal{A}$ such that $\mathcal{C}_\mathcal{S} \supseteq \mathcal{C}_\mathcal{H}$ and
(2) all hidden credentials in $\mathcal{C}_\mathcal{S}$ with respect to $\mathcal{P}_\mathcal{D}$ are in $\mathcal{C}_\mathcal{H}$.

To protect a solution set from unnecessary disclosure one can specify which credentials from the solution set should be hidden so that only selected clients can access a resource.

Taking it to extreme, all solution sets for a resource could be hidden, i.e. for any solution $\mathcal{C}_\mathcal{S}$ and its set of hidden credentials $\mathcal{C}_\mathcal{H}$ holds $\mathcal{C}_\mathcal{H} = \mathcal{C}_\mathcal{S}$, and we fall back in the standard classical access control framework of having only grant/deny decisions.

DEFINITION 6.8 CLIENT WITH HIDDEN CREDENTIALS FOR A RESOURCE r. *A* client with hidden credentials for a resource r *is any client that has the set of hidden credentials $\mathcal{C}_\mathcal{H}$ of a solution set for r and whenever requests r he sends $\mathcal{C}_\mathcal{H}$ initially.*

Below, we refine the fair interaction property for hidden credentials.

DEFINITION 6.9 FAIR INTERACTION WITH HIDDEN CREDENTIALS. *Let $\mathcal{P}_\mathcal{A}$ and $\mathcal{P}_\mathcal{D}$ be access and disclosure control policies, respectively. The policies* guarantee fair interaction *if*

(1) *$\mathcal{P}_\mathcal{A}$ guarantees fair access and*

(2) *if $\mathcal{C}_\mathcal{S}$ is a solution for a request r and $\mathcal{C}_\mathcal{H}$ is the set of hidden credentials for $\mathcal{C}_\mathcal{S}$ then the visible part of $\mathcal{C}_\mathcal{S}$ is disclosable by $\mathcal{P}_\mathcal{D} \cup \mathcal{C}_\mathcal{H}$, i.e. $\forall c \in (\mathcal{C}_\mathcal{S} \setminus \mathcal{C}_\mathcal{H})$, $\mathcal{P}_\mathcal{D} \cup \mathcal{C}_\mathcal{H} \models c$.*

The intuition of fair interaction with hidden credentials is that hidden credentials in a solution set are all that is needed to obtain the disclosure of the remaining credentials.

THEOREM 6.4. COMPLETENESS FOR A COOPERATIVE CLIENT WITH HIDDEN CREDENTIALS. *Let $\mathcal{P}_\mathcal{A}$ be an access policy, $\mathcal{P}_\mathcal{D}$ be a disclosure policy and r a request. If $\mathcal{P}_\mathcal{A}$ and $\mathcal{P}_\mathcal{D}$ guarantee fair access and interaction then if a cooperative client with hidden credentials $\mathcal{C}_\mathcal{H}$ for r has a solution set $\mathcal{C}_\mathcal{S}$ with respect to $\mathcal{C}_\mathcal{H}$ then the client always gets grant r with the access control algorithm.*

PROOF. Refer to Appendix A. □

7. IMPLEMENTING THE ACCESS CONTROL FRAMEWORK: IACCESS SYSTEM

This section describes an implementation of the access control framework called iAccess.

7.1 Use of answer set programming solvers (ASP)

With the increase of computational power over the last decade, ASP solvers have become very efficient tools[4] that can compute results in few seconds even with several thousands of atoms and rules. It makes them highly suitable for access control engines especially when scaling to hundreds of access rules, constraints, and user roles and tasks.

We use DLV[5] system as a back-end engine for the basic computations of deduction and abduction. DLV is a disjunctive datalog system with negations and constraints under the stable model semantics. DLV has variety of front-ends facilitating different computations. Among them the two front-ends relevant to our purposes are: the disjunctive datalog front-end (the default one) used for deductive computations and the diagnosis front-end used for abduction computations.

We use DLV's default front-end with input a service request marked as a query over the second argument the access policy to check whether the request is granted by the policy. DLV output of this step is whether the request is true or false in all stable models of the policy.

We use DLV's default front-end with input a disclosure policy union a set of presented credentials in order to compute all disclosable credentials.

We use the abduction diagnosis front-end with input a service request stored in a temporary file with extension *obs*, a set of disclosable credentials stored in a temporary file with extension *hyp* and, the third argument, an access policy union a set of presented credentials. The file with extension *hyp* points to a set of hypotheses and the file with *obs* points to a set of observations. The DLV output of that step

[4]See benchmarks of ASP solvers http://asparagus.cs.uni-potsdam.de

[5]www.dlvsystem.com

are all subsets (subset minimal) of the hypotheses that satisfy the observations. In that way we find all missing sets of credentials satisfying the request.

On top of the missing sets we filter them according to some minimality criteria. We have adopted the sequence first role minimality then set cardinality. For doing so, we include extra information to the credentials in the hypotheses, specifying a credential weight or its position in the role-lattice hierarchy. Then we select the set(s) with the lowest role-position values. After obtaining the minimal set we drop the extra information from the credentials that are to be sent to a client.

7.2 Integration with X.509 standard

We adopted X.509 [X.509 2005] certificate standard for attesting participants identities and attributes. There are two certificate types considered by the standard: identity and attribute certificates. Figure 8 shows the structures of the two certificates.

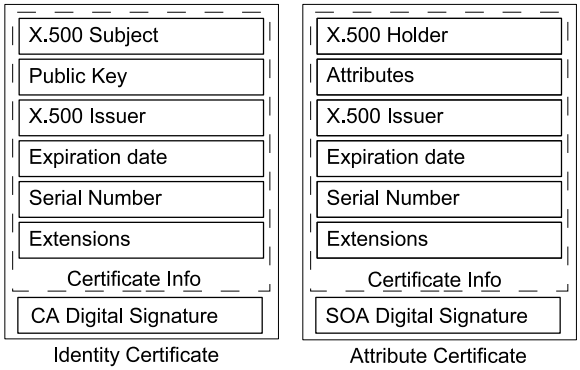


Fig. 8. X.509 Identity and Attribute Certificates Structure

X.509 identity certificate is used to identify entities in a network. The main fields of the certificate's structure are the subject information, the public key identifying the subject (corresponding to the subject's private key), the issuer information and the digital signature on the document, signed by the issuer (with its private key). X.509 attribute certificate has the same structure like the identity one with the difference that instead of a public key field there is a field for listing attributes and the subject field is called holder (of the attributes).

As already noted in Section 5, we need a way to semantically convert X.509 certificates to internal policy compliant representation. We adopted the following transformations:

—An identity certificate to `certificate`(subject, `Issuer`: i) predicate identifying entity subject stated by authority `i`.

—An attribute certificate to `credential`(holder, `Attr`: a, `Issuer`: i) predicate attesting that the holder has an attribute `a` issued by authority `i`.

The logical model has the following two sets of predefined identifiers regarding credential transformations: `Attr` for attribute identifiers and `Issuer` for certificate authority identifiers. We developed a semantic conversion module that has a predefined database specifying certificate to credential conversions including transformations between public keys of trusted CAs and SOAs and their logical identifiers as well as transformations between attribute values and their logical representation.

The semantic conversion module has also the responsibility to properly convert internal credential values to values compliant with the external domain a request comes from.

7.3 Integration with SAML standard

We have adopted OASIS SAML[6] standard for having standard semantics of authorization statements among participants in an autonomic network. SAML offers a standard way of exchanging authentication and authorization information between on-line partners. The basic SAML data objects are assertions. Assertions contain information that determines whether users can be authenticated or authorized to use resources. The SAML framework also defines a protocol for requesting assertions and responding to them, which makes it suitable when modeling interactive communications between entities in a distributed environment.

A client uses SAML Authorization Decision Query statement to specify a resource name and a resource action when requesting a service. Once a SAML request is received iAccess extracts the Authorization Decision Query and invokes the semantic conversion module for transforming it to a predicate of the logical model. We have adopted the transformation Authorization Decision Query to `grant(Resource`: r, `Action`: p) where `Resource` and `Action` are predefined sets of identifiers considered in the logical model.

Once an access decision is taken, iAccess generates a SAML response part incapsulating a SAML Authorization Decision assertion. The authorization decision assertion has three types of decision values: permit, deny and indeterminate.

—Permit or Deny decision is used when the access control algorithm explicitly returns grant or deny.

—Indeterminate decision is used when `ask(`$\mathcal{C}_{\mathcal{M}}$`)` is returned.

iAccess uses the SAML standard attribute assertions to list the set of missing credentials. For each certificate(subject, issuer) the semantic conversion module generates a SAML assertion with an authentication statement detailing subject and issuer fields. In case of credential(holder, attribute, issuer) the semantic conversion module generates a SAML assertion with an attribute statement. The inconvenience with SAML v1.1 is the fact that issuer field is designed as an attribute to an assertion element and not as separate element having its own generalized name identifier. The standard issuer field of a SAML assertion has the semantics of the entity issued the assertion, in our case it is the iAccess system. As so we had to provide the information for a potential issuer of an attribute by means of an additional attribute field in the attribute statement listing the missing credential.

[6] http://www.oasis-open.org/committees/security

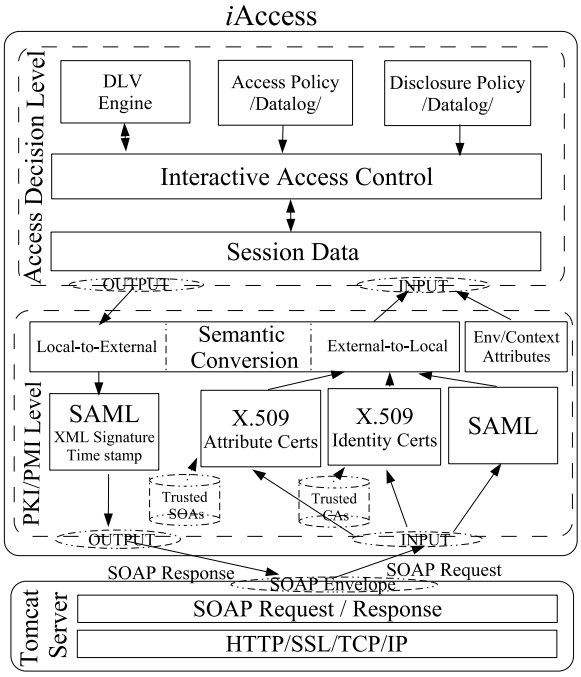


Fig. 9. *i*Access Architecture

7.4 iAccess Architecture

To make the access decision engine Web Services compatible we also adopted W3C SOAP[7] as a main transport layer protocol. SOAP is a lightweight protocol for exchanging structured information in a distributed environment. It has an optional Header element and a required Body element. Informally, in the body we specify what information is directly associated with the service request and in the header additional information that should be considered by the end-point server.

To request for an access decision on a message level one has to:

(1) place SAML Request in the SOAP Body thus making it an input to the decision engine being invoked and

(2) attach X.509 Certificates in the SOAP Header using WS-Security[8] specification for that.

Figure 9 shows iAccess system architecture. The bottom most layer comprises the integration of the prototype with the Tomcat[9] application server. To ensure message confidentiality on the transport layer one can perform all interactions over an SSL connection.

When the Tomcat server receives an access request it invokes the iAccess engine

[7] http://www.w3.org/TR/soap
[8] http://www.oasis-open.org/committees/wss
[9] http://tomcat.apache.org

for an access decision. iAccess parses the SOAP envelope, containing the body and the header elements, extracts X.509[10] identity and attribute certificates, and the SAML[11] request protocol.

Next, iAccess validates and verifies the certificates and invokes the semantic conversion module with input the certificates and SAML authorization request. The verification against trusted CAs and SOAs is to identify internally those authorities that are known and trusted by a server. Those authorities unknown to a server are internally represented according to some default criteria, e.g. by using authorities' common name (CN) of the X.500 structure.

We note that the semantic conversion database is dynamically allocated and loaded depending on the domain the request comes from.

Once an access decision is taken iAccess invokes the conversion module for transforming grant, deny or additional credentials onto a SAML decision assertion that is then wrapped in a SAML Response. Next, iAccess places a time-stamp for validity period on the decision statement and digitally signs it to ensure integrity of the information. Tomcat server returns the SAML decision to the entity requested it. When the SAML assertion is received it becomes an authorization certificate that is to be presented to an application enforcement module for providing access to a resource.

Envr/Context module provides the environment and context attributes that access and disclosure policies are sensitive against (e.g. system time and state, network endpoint address etc).

8. IACCESS PERFORMANCE EVALUATION

We will present iAccess time response evaluation in three parts: PKI/PMI part, access decision part and total time response. The PKI/PMI part covers the extraction of X.509 certificates and SAML request, certificate validation and verification, and logical predicates transformation. The second part corresponds to the actual interactive access control algorithm functionality. The overall time response includes the time response of PKI/PMI part, the access decision part and the time response of the conversion module for generating a digitally signed SAML response element containing the access decision.

We run the iAccess system on the access control policies described in Section 5. All the tests have been run on a PC with Windows XP, Intel Pentium 4 processor on 2 Ghz and 512 Mb of RAM.

Figure 10 shows the first set of trials. The performance has been measured in milliseconds and rounded to seconds when displayed on the diagram. We have invoked iAccess server 11 times with different number of X.509 certificates. For that purpose, we generated X.509 attribute certificates with roles fraunhofer *employee01* to fraunhofer *employee97*. Each trial has been done with increase of 10 certificates and the last one with input of 100 certificates.

Each trial had an input the three certificates for Alice Milburk employee, Fraunhofer identity certificate and Fraunhofer accredited institution, and the remaining certificates from the newly generated ones, i.e. first trial with 2 of the new certifi-

[10] X.509 technology provider: http://www.bouncycastle.org
[11] SAML technology provider: http://www.opensaml.org

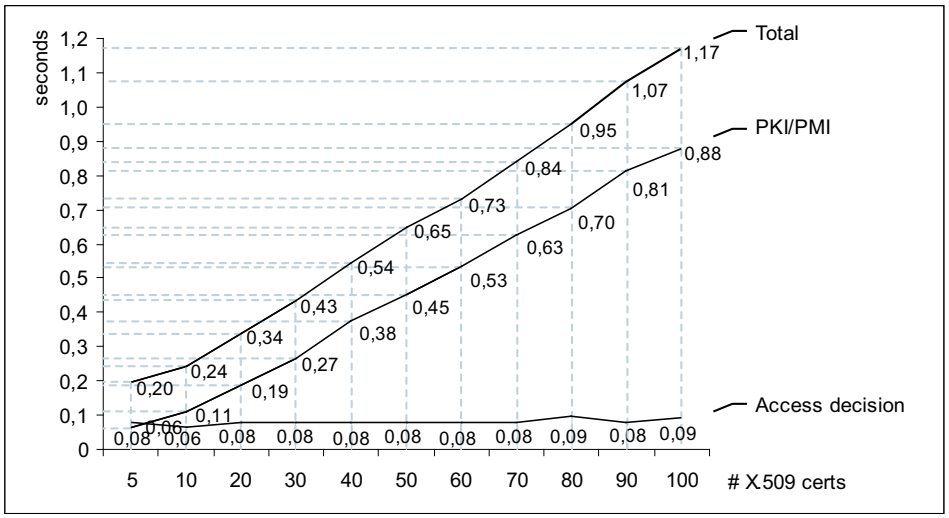


Fig. 10. iAccess performance with increasing input of X.509 certificates

cates, second with 7, third with 17 and so on and the last one with 97. Each trial specified access request for *conf* permission so that the system had to compute a set of missing credentials (in our case *juniorScientist*) for every request, thus obtaining the maximum system load.

The positive outcome of the first set of trials is that even with 100 certificates still the iAccess system response to take a decision is around 1 second. We conclude that the number of certificates and its respective cryptographic operations are not a bottleneck for the iAccess timely decision. The generation of a digitally signed SAML response remained within the range of 70-150 milliseconds for all the trials.

However, what we observed was that the access decision time remained less that 100 milliseconds for all the 11 trials while only the PKI/PMI part increased with the increasing number of certificates. So, the total time response has been influenced mainly by the PKI/PMI part.

The explanation for that is the way the access decision algorithm functions. In case of not enough access rights the algorithm computes the set of disclosable credentials and then invokes the abduction reasoning with input the set of credentials marked as hypotheses. If we look at the disclosure policy (Fig. 7) we can immediately find out that whatever credentials we input the disclosure policy releases only the 9 roles we have in the hierarchy and the 4 certificates of trusted CAs and SOAs. Thus with the increased number of client's certificates the number of hypotheses to the abduction engine remained unchanged – 10 facts – and so the engine took the same time to compute a set of missing credentials. We remind that from the disclosable credentials we remove all presented credentials, in our case the three certificates for an employee, legal key holder and accredited organization.

Figure 11 shows the second set of trials but this time with increasing number of hypotheses on each trial. We modified the disclosure policy (by adding new rules) such that for each presented credential in the range *employee01* to *employee97* the

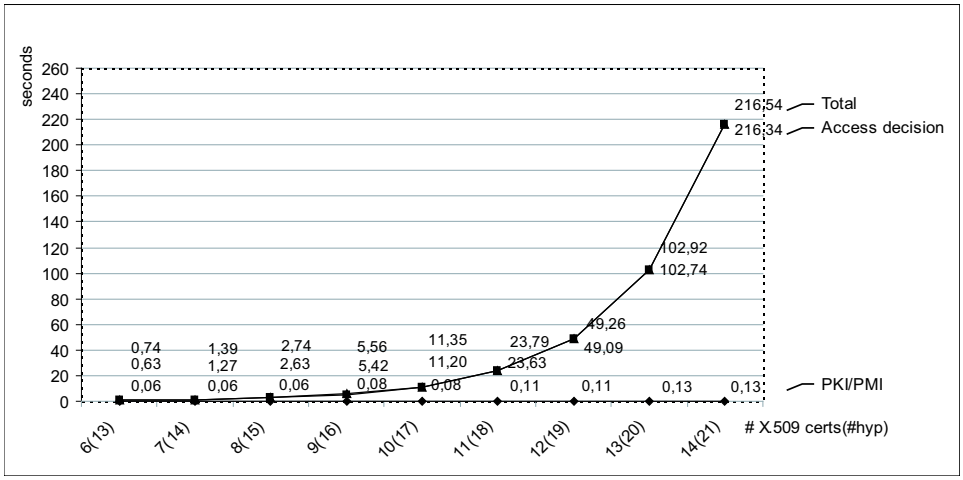


Fig. 11. iAccess performance with increasing input of X.509 certificates and dynamic hypotheses

policy discloses a new credential *juniorScientist01* to *juniorScientist97*, respectively. Additionally, we specified that each *juniorScientistXX* dominates the basic role *juniorScientist*. In this way with increase number of input certificates we increase the number of hypotheses to the abduction engine.

We have done 9 trials but this time staring with 6 certificates and increasing with 1 on each trial. On the horizontal axis we denote the number of input certificates and in brackets the number of hypotheses feed to the abduction engine. With 13 hypotheses the access decision time is 630 milliseconds and with 21 hypotheses (14 input certificates) the time response is approximately 3 and a half minutes. The PKI/PMI part remained imperceptible with respect to the access decision time. We refer to [Eiter and Gottlob 1995] for a comprehensive reading of abduction complexity.

However, the conclusion from the second set of trials is that with 16 possible roles in an organization the iAccess system would give access decision results within a reasonable amount of time (approx. 6 seconds) taking into account that for any request potentially all roles are disclosable. As of now iAccess could be suitable for small and medium size enterprises (SMEs).

The main factor for the potential number of rounds needed to grant a service is the complexity of the access policy and particularly how many possible solutions exist for a service.

9. CONCLUSIONS

We have proposed a framework for access control for autonomic communication. The key idea is that in an autonomic network a client may have the right set of credentials but may not know it and an autonomic server needs a way to interact and communicate with the client missing credentials that grant access.

We have proposed a solution to this problem by extending classical policy-based access control models with an advanced reasoning service: abduction. Built on top of it, we have presented the interactive access control algorithm that computes on

the fly missing credentials that entail a request.

We enriched the framework over the existing policy-based approaches for access control by introducing the difference between monotonic and well-behaved policies and between disclosable and hidden credentials.

The first distinction extends our work on a wider set of policy languages with respect to the already existing approaches [Bonatti and Samarati 2002; Yu et al. 2003; Kapadia et al. 2004; Bertino et al. 2004; Constandache et al. 2007]. The latter distinction addresses the behavior of an autonomic server by allowing it to dynamically protect the privacy of its policies by specifying which credentials are hidden and which are not.

We have shown that the access control framework is sound and correct. We have presented an implementation of the model called iAccess and its performance evaluation.

Future work is to extend the interactive model to cope with automated credential discovery. The aim is to leverage the interactive access control process and reduce the number of credentials requested to a client.

Another direction of future work is researching what guarantees the interactive framework offers in terms of interoperability when applied to existing negotiation systems such as TrustBuilder[Winslett et al. 2002], Trust-X [Bertino et al. 2004] and PeerTrust [Nejdl et al. 2004].

REFERENCES

Apt, K. 1990. Logic programming. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Elsevier.

Baselice, S., Bonatti, P. A., and Faella, M. 2007. On interoperable trust negotiation strategies. In *Proceedings of IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*. IEEE Computer Society, 39–50.

Bertino, E., Catania, B., Ferrari, E., and Perlasca, P. 2001. A logical framework for reasoning about access control models. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM Press, 41–52.

Bertino, E., Ferrari, E., and Squicciarini, A. C. 2004. Trust-X: A peer-to-peer framework for trust establishment. *IEEE Transactions on Knowledge and Data Engineering 16,* 7, 827–842.

Bonatti, P. and Samarati, P. 2002. A unified framework for regulating access and information release on the web. *Journal of Computer Security 10,* 3, 241–272.

Constandache, I., Olmedilla, D., and Siebenlist, F. 2007. Policy-driven negotiation for authorization in the grid. In *Proceedings of IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*. IEEE Computer Society, 211–220.

Damianou, N., Dulay, N., Lupu, E., and Sloman, M. 2001. The Ponder policy specification language. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks (POLICY'01)*. IEEE Computer Society, 18–38.

De Capitani di Vimercati, S. and Samarati, P. 2001. Access control: Policies, models, and mechanism. In *Foundations of Security Analysis and Design - Tutorial Lectures*, R. Focardi and F. Gorrieri, Eds. LNCS, vol. 2171. Springer-Verlag Press.

Eiter, T. and Gottlob, G. 1995. The complexity of logic-based abduction. *Journal of the ACM 42,* 1, 3–42.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference on Logic Programming (ICLP'88)*, R. Kowalski and K. Bowen, Eds. MIT-Press, 1070–1080.

Kapadia, A., Sampemane, G., and Campbell, R. H. 2004. Know why your access was denied: regulating feedback for usable security. In *Proceedings of the 11th ACM conference on Computer and Communications Security*. ACM Press, New York, NY, USA, 52–61.

KOSHUTANSKI, H. AND MASSACCI, F. 2007. A negotiation scheme for access rights establishment in autonomic communication. *Journal of Network and System Management (JNSM) 15,* 1, 117–136. Springer press.

LI, J., LI, N., AND WINSBOROUGH, W. H. 2005. Automated trust negotiation using cryptographic credentials. In *Proceedings of the 12th ACM conference on Computer and Communications Security*. ACM Press, New York, NY, USA, 46–57.

LI, N., GROSOF, B. N., AND FEIGENBAUM, J. 2003. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security (TISSEC) 6,* 1, 128–171.

LI, N. AND MITCHELL, J. C. 2003. RT: A role-based trust-management framework. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX III)*. IEEE press, Los Alamitos, California, 201–212.

LYMBEROPOULOS, L., LUPU, E., AND SLOMAN, M. 2003. An adaptive policy based framework for network services management. *Journal of Network and Systems Management 11,* 3 (September), 277–303.

NEJDL, W., OLMEDILLA, D., AND WINSLETT, M. 2004. PeerTrust: Automated trust negotiation for peers on the semantic web. In *VLDB Workshop on Secure Data Management (SDM)*. Lecture Notes in Computer Science, vol. 3178. Springer, 118–132.

SANDHU, R., COYNE, E., FEINSTEIN, H., AND YOUMAN, C. 1996. Role-based access control models. *IEEE Computer 39,* 2 (February), 38–47.

SEAMONS, K. AND WINSBOROUGH, W. 2002. Automated trust negotiation. US Patent and Trademark Office. IBM Corporation, patent application filed March 7, 2000.

SHANAHAN, M. 1989. Prediction is deduction but explanation is abduction. In *Proceedings of IJCAI'89*. Morgan Kaufmann, 1055–1060.

SLOMAN, M. AND LUPU, E. 1999. Policy specification for programmable networks. In *Proceedings of the First International Working Conference on Active Networks*. Springer-Verlag, 73–84.

SMIRNOV, M. 2003. Rule-based systems security model. In *Proceedings of the Second International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS)*. Springer-Verlag Press, 135–146.

SPKI. 1999. SPKI certificate theory. IETF RFC 2693.

WEEKS, S. 2001. Understanding trust management systems. In *IEEE Symposium on Security and Privacy (SS&P)*. IEEE Press.

WINSLETT, M., YU, T., SEAMONS, K., HESS, A., JACOBSON, J., JARVIS, R., SMITH, B., AND YU, L. 2002. Negotiating trust in the Web. *IEEE Internet Computing 6,* 6 (Nov/Dec), 30–37.

WINSLETT, M., ZHANG, C. C., AND BONATTI, P. A. 2005. PeerAccess: a logic for distributed authorization. In *Proceedings of the 12th ACM CCS Conference*. 168–179.

X.509. 2005. The directory: Public-key and attribute certificate frameworks. ITU-T Recommendation X.509:2005 | ISO/IEC 9594-8:2005.

YU, T. AND WINSLETT, M. 2003. A unified scheme for resource protection in automated trust negotiation. In *Proceedings of the IEEE Symposium on Security and Privacy*. 110–122.

YU, T., WINSLETT, M., AND SEAMONS, K. E. 2003. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security (TISSEC) 6,* 1, 1–42.

A. FORMAL PROOFS

PROOF THEOREM 6.1. This proof is rather straightforward. The only way to introduce a credential in $\mathcal{C}_\mathcal{P}$ is by step 1 of the algorithm. Since initially $\mathcal{C}_\mathcal{P} = \emptyset$ so the client has sent a sequence of sets of credentials $\mathcal{C}_{p_1}, \ldots, \mathcal{C}_{p_n}$ such that $\bigcup_{i=1}^{n} \mathcal{C}_{p_i} = \mathcal{C}_\mathcal{P}$. So, the client has a set of credentials that unlocks r. □

PROOF THEOREM 6.2. At each interaction the union of presented and declined credentials always increases. Because at each interaction abduction finds different solution with respect to the preceding ones then the union sets increases always with new credentials occurring in the access policy.

Since the union set is bound by the credentials occurring in the policy then there is always a stage in which either grant (enough presented credentials) or deny (too many declined credentials) is given. □

PROOF THEOREM 6.3. We proof the theorem in two parts. First part proves that in a single interaction if a cooperative client does not get grant r he gets `ask(`$\mathcal{C}_\mathcal{M}$`)`, i.e. a cooperative client will not be denied access by the algorithm. Second part (rather straightforward) shows that since the access policy is finite then a cooperative client with a solution set for r will get grant r.

Part 1.

Proof by induction on interaction steps:

Interaction 1. Client requests service r together with an initial set of presented credentials $\mathcal{C}_p = \emptyset$. Fair access and interaction properties guarantee that: (i) a solution for r exists according to the access policy $\mathcal{P}_\mathcal{A}$ and (ii) that solution is disclosable by the disclosure policy $\mathcal{P}_\mathcal{D}$. Therefore, abduction reasoning finds a solution for r and the algorithm returns it back to the client.

Interaction N. Here we use the induction hypothesis that the client fails to get grant r and gets `ask(`$\mathcal{C}_\mathcal{M}$`)` at interaction step N-1. Now, suppose that the client fails to get grant r at interaction N. There are two reasons to fail: either there is no solution in the set of active credentials $\mathcal{C}_\mathcal{P}$ that unlocks the request or $\mathcal{C}_\mathcal{P}$ makes the access policy state inconsistent so that any solution set in $\mathcal{C}_\mathcal{P}$ does not entail the request.

The set of active credentials $\mathcal{C}_\mathcal{P}$ increases only with credentials that are part of other solutions for r, i.e. $\mathcal{C}_\mathcal{P} \subset (\mathcal{C}^1_\mathcal{M} \cup \ldots \cup \mathcal{C}^{N\text{-}1}_\mathcal{M})$ where $\mathcal{C}^i_\mathcal{M}$ denotes the set of missing credentials returned at each interaction preceding the current one. Here we use the assumption that access policy $\mathcal{P}_\mathcal{A}$ is *well-behaved*. According to Definitions 3.8, 3.7 and 3.6 follows that $\mathcal{C}_\mathcal{P}$ is subset of a solution set for r (using the additive property) and is consistent with the access policy.

Therefore, $\mathcal{C}_\mathcal{P}$ preserves consistency in $\mathcal{P}_\mathcal{A}$ and the only reason the client fails to get grant is that there is no solution for r in $\mathcal{C}_\mathcal{P}$.

In step 3 the algorithm computes the set of disclosable credentials $\mathcal{C}_\mathcal{D}$.

Since $\mathcal{P}_\mathcal{A}$ and $\mathcal{P}_\mathcal{D}$ guarantee fair access and interaction then the solution set $\mathcal{C}_\mathcal{S}$ that the client has is disclosable – $\mathcal{C}_\mathcal{S} \subseteq (\mathcal{C}_\mathcal{D} \cup \mathcal{C}_\mathcal{P})$ – and not yet presented – $\mathcal{C}_\mathcal{S} \not\subseteq \mathcal{C}_\mathcal{P}$.

Following that, the abduction reasoning will find a solution for r in step 4b and that solution is guaranteed by the existence of the non-empty set $\mathcal{C}_\mathcal{S} \setminus \mathcal{C}_\mathcal{P}$:

(i) $(\mathcal{C}_\mathcal{S} \setminus \mathcal{C}_\mathcal{P}) \subseteq \mathcal{C}_\mathcal{D}$ and

(ii) $(\mathcal{C}_{\mathcal{S}} \cup \mathcal{C}_{\mathcal{P}}) \subset (\mathcal{C}^{1}_{\mathcal{M}} \cup \ldots \cup \mathcal{C}^{N\text{-}1}_{\mathcal{M}} \cup \mathcal{C}_{\mathcal{S}})$

(iii) Since $\mathcal{P}_{\mathcal{A}}$ is well-behaved follows that $\mathcal{C}_{\mathcal{S}} \cup \mathcal{C}_{\mathcal{P}}$ does not make $\mathcal{P}_{\mathcal{A}}$ inconsistent, i.e. $\mathcal{P}_{\mathcal{A}} \cup \mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{S}} \not\models \bot$.

Following the bullets above the client gets $\texttt{ask}(\mathcal{C}_{\mathcal{M}})$ at interaction step N.

Part 2.
We proved in Part 1 that in a single interaction step if a cooperative client does not get grant r he gets $\texttt{ask}(\mathcal{C}_{\mathcal{M}})$.

There are a finite number of solutions for each request r simply because $\mathcal{P}_{\mathcal{A}}$ consists of a finite number of access rules. The abduction reasoning service at each interaction computes different solution with respect to the solutions computed in previous interactions because from the disclosable credentials we remove all credentials that have been already requested to a client (ref. step 4a).

Since there are finite solution sets for r and since the client has one of them therefore the client in a finite number of interaction steps will be asked to present $\mathcal{C}_{\mathcal{S}}$, i.e. $\mathcal{C}_{\mathcal{S}} \subseteq \mathcal{C}_{\mathcal{P}}$, and will get grant r. □

PROOF THEOREM 6.4. Analogously of theorem 6.3 we proof the theorem in two parts. First part proves that in a single interaction if the client does not get grant r he gets $\texttt{ask}(\mathcal{C}_{\mathcal{M}})$. Second part shows that in a finite number of interactions the client will be asked the solution he has and gets grant.
Part 1. Proof by induction on the interaction steps:

Interaction 1. Client requests service r together with an initial set of presented credentials $\mathcal{C}_p = \mathcal{C}_{\mathcal{H}}$. We apply Definition 6.8 for a client with hidden credentials. Next, using Definition 6.9 fair interaction with hidden credentials follows that the solution $\mathcal{C}_{\mathcal{S}}$ the client has is disclosable by the disclosure policy and the set $\mathcal{C}_{\mathcal{H}}$. Next, abduction step will find a solution for r (at least $\mathcal{C}_{\mathcal{S}}$ is a potential solution) and will return $\texttt{ask}(\mathcal{C}_{\mathcal{M}})$.

Interaction N. Here we use the induction hypothesis that the client fails to get grant r and gets $\texttt{ask}(\mathcal{C}_{\mathcal{M}})$ at interaction step N-1.
Now, suppose that the client fails to get grant r at interaction N. The proof of this step becomes identical to the respective interaction step in the proof of Theorem 6.3 taking into account that the client's solution set $\mathcal{C}_{\mathcal{S}}$ is disclosable by $\mathcal{P}_{\mathcal{D}}$ and $\mathcal{C}_{\mathcal{P}}$, because $\mathcal{C}_{\mathcal{H}}$ is already in the set $\mathcal{C}_{\mathcal{P}}$.
Then since $\mathcal{P}_{\mathcal{A}}$ is well-behaved and $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{D}}$ satisfy fair access and interaction property follows that the non-empty set $\mathcal{C}_{\mathcal{S}} \setminus \mathcal{C}_{\mathcal{P}}$ will guarantee that the abduction reasoning will find a solution for r and the client will get $\texttt{ask}(\mathcal{C}_{\mathcal{M}})$.

Part 2.
The rest of the proof follows the lines of the proof of Theorem 6.3. □