Automated and Efficient Analysis of Role-Based Access Control with Attributes

Alessandro Armando[1,2] and Silvio Ranise[2]

[1] DIST, Università degli Studi di Genova, Italia
[2] Security and Trust Unit, FBK-Irst, Trento, Italia

Abstract. We consider an extension of the Role-Based Access Control model in which rules assign users to roles based on attributes. We consider an open (allow-by-default) policy approach in which rules can assign users negated roles thus preventing access to the permissions associated to the role. The problems of detecting redundancies and inconsistencies are formally stated. By expressing the conditions on the attributes in the rules with formulae of theories that can be efficiently decided by Satisfiability Modulo Theories (SMT) solvers, we characterize the decidability and complexity of the problems of detecting redundancies and inconsistencies. The proof of the result is constructive and based on an algorithm that repeatedly solves SMT problems. An experimental evaluation with synthetic benchmark problems shows the practical viability of our technique.

1 Introduction

Role Based Access Control (RBAC) [27] is one of the most widely adopted model for information security. It regulates access by assigning users to roles which, in turn, are granted permissions to perform certain operations. Despite several advantages (e.g., reduction of the complexity of security administration), it has been observed that RBAC suffers some inflexibility in adapting to rapidly evolving domains in particular when there is a need to take into account dynamic attributes to determine the permissions of a user. The problem is the explosion in the number of roles that should be considered in order to specify the various sets of permissions that a user may acquire depending on the values of the dynamic attributes.

To overcome this problem, [16] proposes to add attributes to RBAC and overviews several approaches to do this. Among these, the one called *Dynamic Roles (DRs)* determines the user's role depending on his/her attributes. The interest of DRs is that they retain the structure of the RBAC model while providing additional flexibility to cope with attributes. A similar approach have already been investigated in [2,3] and problems concerning redundancies in the rules that assign users to roles and possible inconsistencies with the standard RBAC role hierarchy are studied. Furthermore, in [4], negative roles are introduced to widen the scope of applicability of the RBAC model to those situations in which an open (allow-by-default) policy approach is adopted (i.e. access is

N. Cuppens-Boulahia et al. (Eds.): DBSec 2012, LNCS 7371, pp. 25–40, 2012.

denied if there exists a corresponding negative authorization and is permitted otherwise [26]). In this context, conflicts may arise among rules that assign users to a role and, at the same time, to the negated role.

Clearly, simply authoring a set of rules that assign users to roles is not sufficient: an organization must also be able to analyse it in order to avoid the kind of problems sketched above. Testing, while useful, is not exhaustive and, as organizations grow, their rule set can become very large, which places particular burden on the quality of testing. Designers of RBAC policies with DRs would thus benefit from complementing testing with more exhaustive and automatic, formal analysis techniques. In this respect, the paper makes three contributions.

First, we give an abstract definition for policies with DRs that naturally extends the RBAC model (Section 2). Negative authorization (via negated roles) can be easily accommodated in the proposed framework (Section 2.3).

Second, we provide a rule-based characterization of the association between users' attributes and roles (Section 2) that allows us to formally state two crucial problems for the design of RBAC policies with DRs: detection of redundancies (Section 2.1) and detection of conflicts (Section 2.3).

Third (Section 4.1), we show how conditions on the users' attributes of authorization rules can be expressed by theories, whose satisfiability problems can be efficiently decided by Satisfiability Modulo Theories (SMT) solvers (Section 3) and characterize the decidability and the complexity of the problems of detecting redundancies and conflicts under natural assumptions on the theories of the attributes (Sections 4.2 and 4.3). An experimental evaluation with a prototype implementation of the algorithm (Figure 1) used for the decidability results on a synthetic benchmark confirms the viability of the approach (Section 4.4).

Related work and conclusions are also discussed (Section 5). Proofs can be found in the extended version of this paper available at `http://st.fbk.eu/SilvioRanise#Papers`.

2 RBAC with Dynamic Roles

Preliminarily, we recall some basic notions concerning the RBAC model [27]. RBAC regulates access through roles. Roles in a set R associate permissions in a set P to users in a set U by using the following two relations: $UA \subseteq U \times R$ and $PA \subseteq R \times P$. Roles are structured hierarchically so as to permit permission inheritance. Formally, a role hierarchy is a partial order RH on R, where $(r_1, r_2) \in RH$ means that r_1 is *more senior than* r_2 for $r_1, r_2 \in R$. A user u is an *explicit member* of role r when $(u, r) \in UA$, u is an *implicit member* of r if there exists $r' \in R$ such that $(r', r) \in RH$ and $(u, r') \in UA$, and u is a *member* of role r if he/she is either an implicit or explicit member of r. Given UA and PA, a user u *has permission* p if there exists a role $r \in R$ such that $(p, r) \in PA$ and u is a member of r. A *RBAC policy* is a tuple (U, R, P, UA, PA, RH).

When extending RBAC with dynamic roles, the relation UA is not given explicitly but it is defined in terms of pairs (attribute name, attribute value). A *RBAC policy with Dynamic Roles (RBAC-DR)* is a tuple $(U, R, P, \underline{a}, \underline{D_a}, \underline{s_a}, AR,$

PA, RH) where U, R, P, PA, and RH are as in the RBAC model above, $\underline{a}$ is a (finite) sequence of *attributes*, $\underline{D}_{\underline{a}}$ is a (finite) sequence of domains associated to $\underline{a}$, $\underline{s}_{\underline{a}}$ is a sequence of *user-attribute mappings associated to* $\underline{a}$ (the sequences $\underline{a}$, $\underline{D}_{\underline{a}}$, and $\underline{s}_{\underline{a}}$ have equal length) such that for each a in $\underline{a}$ (written $a \in \underline{a}$) s_a is a function from U to 2^{D_a}, and AR is the *attribute-role relation* that contains tuples of the form (C, r) where C is a set of pairs (a, e_a) with $e_a \in D_a$ for $a \in \underline{a}$ and $r \in R$. (In the following, the sub-script of $\underline{D}_{\underline{a}}$ and $\underline{s}_{\underline{a}}$ will be dropped to simplify the notation.) Given a RBAC-DR policy $(U, R, P, \underline{a}, \underline{D}, \underline{s}, AR, PA, RH)$, a user $u \in U$ *is an explicit member of role* $r \in R$ *under the user-attribute mapping* $\underline{s}$ iff there exist a pair $(C, r) \in AR$ and a set $S_u \subseteq C$ such that $e_a \in s_a(u)$ for every $(a, e_a) \in S_u$. The notions of "being an implicit member of a role" and "having a permission" are defined as those of RBAC policies above.

2.1 Rule-Based Authorization Rules

The most difficult part in the design of a RBAC-DR policy is the definition of the attribute-role relation. To do this, following [2], we use authorization rules that associate a role to a user provided that his/her attribute values satisfy certain conditions. Let $\underline{a} = a_1, ..., a_n$ be a sequence of attributes, a pair $(\underline{b}, C)$ is a *condition* C *on a subsequence* $\underline{b} = b_1, ..., b_m$ of $\underline{a}$ (i.e. for each b_j there exists a_k such that $b_j = a_k$ for $j = 1, ..., m$, $k = 1, ..., n$, and $m \leq n$), also written as $C(\underline{b})$, where C is a sub-set of $D_{b_1} \times \cdots \times D_{b_m}$. A *(rule-based) authorization rule* is a pair $(C(\underline{b}), r)$, also written as $C(\underline{b}) \rightsquigarrow r$, such that $C(\underline{b})$ is a condition and r is a role in R. The *attribute-role relation* AR *associated to the set* AU *of authorization rules* is $\{(\{(b_1, e_1), ..., (b_m, e_m)\}, r) \mid (C(b_1, ..., b_m) \rightsquigarrow r) \in AU \text{ and } (e_1, ..., e_m) \in C\}$. By abuse of notation, a tuple $(U, R, P, \underline{a}, \underline{D}, \underline{s}, AU, PA, RH)$, where AU is a finite set of authorization rules and all the other components are as in a RBAC-DR policy, will also be called a RBAC-DR policy. Given a RBAC-DR policy $(U, R, P, \underline{a}, \underline{D}, \underline{s}, AU, PA, RH)$, a user u *satisfies the condition* $C(b_1, ..., b_m)$ of an authorization rule in AU *under the user-attribute mapping* $\underline{s}$, in symbols $u, \underline{s} \vdash C(b_1, ..., b_m)$, iff $(e_1, ...e_m) \in C$ and $e_j \in s_{b_j}(u)$ for $j = 1, ..., m$. The *implicit user-role relation* $IUA \subseteq U \times R$ is defined as $IUA := \{(u, r) \mid \text{there exists } (C \rightsquigarrow r) \in AU \text{ such that } u, \underline{s} \vdash C\}$. A user $u \in U$ is a member of role $r \in R$ under the user-attribute mapping $\underline{s}$ (via the notion of attribute-role relation associated to the set AU of authorization rules) iff $(u, r) \in IUA$.

For effectiveness, we assume the computability of (a) the user-attribute assignments and of (b) the membership to the conditions of the authorization rules as well as to (c) the relations RH and PA. Requirement (a) is reasonable as the notion of user-attribute assignment is an abstraction of the mechanism that associates users with attributes (e.g., an LDAP). Requirement (b) is necessary for the effectiveness of the satisfaction relation $\vdash$. Concerning (c), we observe that since the sets R and P are finite in several applications, then membership to RH and PA is obviously computable.

Example 1. We consider a refined version of the example in [2] for an on-line entertainment store streaming movies to users. The store needs to enforce an access control policy that is based on the user's age and the country where

the user lives that may have different regulations for considering someone as an adult, a teen, or a child. E.g., for the legislation of Italy and France, one is considered adult when he/she is 18 or older while in Japan the adult age is 20.

For simplicity, we assume there are only three users Alice, Bob, and Charlie. Alice is 12 years old and lives in Italy, Bob is 39 and lives in Japan, and Charlie is 17 and lives in France. We leave unspecified the permissions, the role hierarchy, and the role-permission assignment as we want to focus on the user-role assignment based on the age and country attributes. In the corresponding RBAC-DR policy $(U, R, P, \underline{a}, \underline{D}, \underline{s}, AU, PA, RH)$, we have that $U := \{Alice, Bob, Charlie\}$, $R := \{Adult, Teen, Child\}$, $\underline{a} = age, country$, $\underline{D} = \mathbb{N}, WC$ where WC is an enumerated set containing all the countries in the world, $\underline{s} = s_{age}, s_{country}$ where $s_{age} = \{Alice \mapsto 12, Bob \mapsto 39, Charlie \mapsto 17\}$, $s_{country} = \{Alice \mapsto Italy, Bob \mapsto Japan, Charlie \mapsto France\}$, and AU contains, among others, the following authorization rules:

$$\rho_1 : \{(a, c) \in \mathbb{N} \times WC | a \geq 20 \text{ and } c \in \{Japan, Indonesia\}\} \rightsquigarrow Adult,$$
$$\rho_2 : \{(a, c) \in \mathbb{N} \times WC | a \geq 18 \text{ and } c \in \{Italy, France, ...\}\} \rightsquigarrow Adult,$$
$$\rho_3 : \{(a, c) \in \mathbb{N} \times WC | 13 \leq a \leq 17 \text{ and } c \in \{Italy, France, ...\}\} \rightsquigarrow Teen, \text{ and}$$
$$\rho_4 : \{(a, c) \in \mathbb{N} \times WC | a \leq 12 \text{ and } c \in \{Italy, France, ...\}\} \rightsquigarrow Child$$

that correspond to the policy informally described above (a and c abbreviate *age* and *country*, respectively). It is not difficult to see that $Bob, \underline{s} \vdash \rho_1$ but $Bob, \underline{s} \nvdash \rho_2$, that $Charlie, \underline{s} \vdash \rho_3$ but $Charlie, \underline{s} \nvdash \rho_i$ for $i = 1, 2, 4$, and that $Alice, \underline{s} \vdash \rho_4$ but $Alice, \underline{s} \nvdash \rho_i$ for $i = 1, 2, 3$. As a consequence, we have that, e.g., $(Alice, Child) \in IUA$, $(Bob, Adult) \in IUA$, and $(Charlie, Teen) \in IUA$. □

2.2 Redundancies in RBAC-DR Policies

As observed in [3], for RBAC-DR policies, besides the usual role hierarchy RH, it is possible to define another hierarchy IRH induced by the authorization rules in AU. We follow [3] and since IRH is defined for every possible user-attribute mapping, we introduce the notion of *RBAC-DR family of policies* as a tuple $(R, P, \underline{a}, \underline{D}, AU, PA, RH)$ where its components are the same as those of a RBAC-DR policy; in other words, a RBAC-DR family of policies is a RBAC-DR policy where the users and the user-attribute assignment are omitted. For every pair (ρ_1, ρ_2) of authorization rules in AU, ρ_1 is *more senior than* ρ_2 (in symbols, $\rho_1 \sqsupseteq \rho_2$) iff for every user u and every user-attribute assignment $\underline{s}$ if $u, \underline{s} \vdash C_1$ then also $u, \underline{s} \vdash C_2$, where C_i is the condition of ρ_i for $i = 1, 2$, i.e. when the set of users satisfying C_1 is a subset of that satisfying C_2 under every possible user-attribute mapping $\underline{s}$. Then, we define $(r_1, r_2) \in IRH$, i.e. the pair (r_1, r_2) is in the *induced role hierarchy* IRH, iff for each rule $(C_1 \rightsquigarrow r_1) \in AU$ there exists a rule $(C_2 \rightsquigarrow r_2) \in AU$ such that $(C_1 \rightsquigarrow r_1) \sqsupseteq (C_2 \rightsquigarrow r_2)$.

The key difference between RH and IRH is that the former is designed so that seniority among roles reflects inheritance of permissions associated to them whereas the latter characterizes seniority according to the sets of users associated to them by the set AU of authorization rules. It may happen that the induced role hierarchy IRH is such that both $(r, r') \in IRH$ and $(r', r) \in IRH$ for two

distinct roles $r, r' \in R$, i.e. the set of users associated to r and that associated to r' are identical. This is so because IRH is a quasi-order (i.e. it is reflexive and transitive) and not a partial order as RH (that is also anti-symmetric). This implies a redundancy in the definition of IRH since the two roles r_1 and r_2 can be considered equivalent as both $(r_1, r_2) \in IRH$ and $(r_2, r_1) \in IRH$. This does not necessarily imply a problem in the authorization rules but it is desirable that designers become aware of all the redundancies in IRH (see [3] for more on this issue). Notice that once the "more senior than" relation among the authorization rules in AU is known, detecting redundancies in IRH becomes obvious. As a consequence, we define the *problem of computing the "more senior than relation" over a set AU of authorization rules* (*MS-AU problem*, for short) as follows: given a RBAC-DR family $(R, P, \underline{a}, \underline{D}, AU, PA, RH)$ of policies, the MS-AU problem amounts to checking whether $\rho_1 \sqsupseteq \rho_2$ for every pair ρ_1 and ρ_2 of authorization rules in AU. Notice that this problem is stated for a RBAC-DR family of policies and thus must be solved regardless of the set of users and the user-attribute mapping.

Solving the MS-AU problem allows one to eliminate the redundancies in IRH. Formally, this amounts to first turning $\sqsupseteq$ into a partial order in such a way that also IRH is so. This is a crucial pre-requisite to apply the techniques in [3] to detect (and eliminate) any "disagreement" between RH and IRH, i.e. to identify those situations in which the security policies encoded by the authorization rules and the business practices captured by RH do not match. Although we do not explore this problem further, we observe that the automated technique to solve MS-AU problems in Section 4.2 facilitates the application of the techniques in [3].

2.3 RBAC-DR with Negative Authorization

In [4], negative authorizations are added to the RBAC-DR model by allowing negated roles in authorization rules. The intuition is that an authorization rule with a negative role r denies access to the permissions associated to r via the permission-assignment relation. This characterization of negative roles allows for the adoption of the so-called *open* (allow-by-default) *policy* semantics [13,26] in an extension of the RBAC model, that is based on positive permissions conferring the ability to do something on the holders of the permissions [27]. This is crucial to widen the scope of applicability of the RBAC model extended with dynamic roles to applications where the set of users is not known a priori, e.g., web-services. Unfortunately, as pointed out in [4], the addition of negative authorizations introduces conflicts that need to be detected and then resolved.

Formally, we define the set $SR = \{+, -\} \times R$ of *signed roles* and write the pairs $(+, r)$ and $(-, r)$ as $+r$ and $-r$, respectively, for a role $r \in R$. We extend the definition of authorization rule as follows: a pair $C \rightsquigarrow sr$ is a *signed authorization rule* where C is a condition and $sr \in SR$. Intuitively, a user u satisfying the condition C of a signed authorization rule of the form $C \rightsquigarrow -r$ under a certain user-attribute mapping is forbidden to be assigned the role r. A *RBAC-DR policy with negative roles* (RBAC-NDR) is a tuple $(U, R, P, \underline{a}, \underline{D}, \underline{s}, AU, PA, RH)$ where $U, R, P, \underline{a}, \underline{D}, \underline{s}, PA$ and RH are as in a RBAC-DR policy and AU is a

set of signed authorization rules. A RBAC-NDR family of policies is the tuple $(R, P, \underline{a}, \underline{D}, AU, PA, RH)$

To understand the kind of conflicts that may arise from the use of signed authorization rules, consider $\rho_1 := (C_1 \rightsquigarrow +r)$ and $\rho_2 := (C_2 \rightsquigarrow -r)$ such that a user u satisfies both C_1 and C_2 under a certain user-attribute mapping. At this point, we are faced with the problem of deciding whether the user u should be assigned or not the role r. As discussed in [4], there are several strategies to resolve the problem, e.g., by adopting a "deny-override" strategy where a user u can be assigned a role r provided that u satisfies the condition C of a rule of the form $C \rightsquigarrow +r$ in AU and there is no rule in AU of the form $C' \rightsquigarrow -r$ such that u satisfies the condition C' that forbids the assignment of r to u. However, this or alternative conflict elimination strategies may be unsatisfactory for certain applications; see again [4] for details. Furthermore, certain conflicts may arise from errors in the design of the authorization rules and must be identified in order to correct them. As a consequence, we define the *problem of detecting conflicts in the set AU of signed authorization rules* (*CD-SAU problem*, for short) as follows: given a RBAC-NDR family $(R, P, \underline{a}, \underline{D}, AU, PA, RH)$ of policies, the CD-SAU problem amounts to checking whether for each pair $(C \rightsquigarrow +r, C' \rightsquigarrow -r)$ in AU, there is no user u satisfying both C and C' under a user-attribute mapping $\underline{s}$. As for the MS-AU problem, also the CD-SAU problem is stated for a family of policies and thus must be solved for any set of users and any user-attribute mapping.

3 Satisfiability Modulo Theories Solving

We assume the usual syntactic (e.g., signature, variable, term, atom, literal) and semantic (e.g., structure, truth) notions of many-sorted first-order logic with equality (see, e.g., [10]). A *constraint* (*clause*, resp.) is a conjunction (disjunction, resp.) of literals (i.e. atoms or their negations) and a *quantifier-free formula* is an arbitrary Boolean combination of atoms.

According to [24], a *theory* T is a pair $(\Sigma, \mathcal{C})$, where Σ is a signature and $\mathcal{C}$ is a class of Σ-structures; the structures in $\mathcal{C}$ are the *models* of T. Given a theory $T = (\Sigma, \mathcal{C})$, a quantifier-free Σ-formula $\phi(\underline{v})$—i.e. a quantifier-free formula built out of the symbols in Σ and the variables in the sequence $\underline{v}$—is *T-satisfiable* if there exists a Σ-structure $\mathcal{M}$ in $\mathcal{C}$ and a valuation μ mapping the variables in $\underline{v}$ to values of the domain of $\mathcal{M}$ such that $\phi(\underline{v})$ is true in $\mathcal{M}$ (in symbols, $\mathcal{M}, \mu \models \phi$); it is *$T$-valid* (in symbols, $T \models \varphi$) if $\mathcal{M}, \mu \models \varphi(\underline{v})$ for every $\mathcal{M} \in \mathcal{C}$ and every valuation μ. The quantifier-free formula $\varphi(\underline{v})$ is T-valid iff its negation $\neg\varphi(\underline{v})$ is T-unsatisfiable. For example, if φ is the implication $\varphi_1 \rightarrow \varphi_2$, then its T-validity is equivalent to the T-unsatisfiability of the conjunction of φ_1 with the negation of φ_2 ($\varphi_1 \wedge \neg\varphi_2$).

The *satisfiability modulo the theory T* ($SMT(T)$) *problem* amounts to establishing the T-satisfiability of quantifier-free Σ-formulae. Many state-of-the-art SMT solvers, in their simplest form, tackle the SMT(T) as follows. Initially, each atom occurring in the input quantifier-free formula φ is considered simply

as a propositional letter (in other words, the theory T is forgotten). Then, the "propositional abstraction" of φ is sent to a SAT solver. If this reports propositional unsatisfiability, then φ is also T-unsatisfiable. Otherwise, an assignment of truth values to the atoms in φ is returned that makes its propositional abstraction true. Such an assignment can be seen as a conjunction of literals and checked by a specialized *theory solver* for T that can only deal with constraints (i.e. conjunctions of literals). If the theory solver establishes the T-satisfiability of the constraint then φ is also T-satisfiable. Otherwise, the theory solver computes a clause that, once added to the SAT solver, precludes the assignment of truth values that has been considered. The SAT solver is then started again. This process is repeated until the theory solver reports the T-satisfiability of one assignment or all the assignments returned by the SAT solver are found T-unsatisfiable so that also φ is reported to be T-unsatisfiable. Various refinements of this basic schema have been proposed for efficiency, the interested reader is pointed to, e.g., [28,9].

The SMT(T) problem is NP-hard as it subsumes the SAT problem. To evaluate the additional complexity due to the theory T, we focus on theory solvers and consider the complexity of checking the T-satisfiability of constraints, called the *constraint T-satisfiability problem* for some theories that have been found useful for the declarative specifications of authorization policies (see, e.g., [18]). The theory $\mathcal{EUF}$ of *equality with uninterpreted function symbols* is the theory interpreting the symbol $=$ as an equivalence relation that is also a congruence. The constraint $\mathcal{EUF}$-satisfiability problem is decidable and polynomial. The theory $\mathcal{ED}(\{v_1, ..., v_n\}, S)$ of the *enumerated data-type S with values* $\{v_1, ..., v_n\}$ (for $n \geq 1$) is the theory whose signature consists of the sort S, the constant symbols $v_1, ..., v_n$ of sort S, and its class of models contains all the structures whose domain contains exactly n distinct elements. The constraint $\mathcal{ED}$-satisfiability problem is decidable and NP-complete. Linear Arithmetic on the Rationals (Integers, resp.) $\mathcal{LAR}$ ($\mathcal{LAI}$, resp.) is the theory whose class of models is the singleton containing the usual structure $\mathbb{R}$ of the Reals ($\mathbb{Z}$ of the integers, resp.) and whose atoms are equalities, disequalities, and inequalities of linear polynomials whose coefficients are integers and the variables take values over the Reals (integers, resp.). The constraint satisfiability problems for $\mathcal{LAR}$ and $\mathcal{LAI}$ are decidable, the former is polynomial and the latter is NP-complete. The theory $\mathcal{RDL}$ of *real difference logic* is the sub-theory of $\mathcal{LAR}$ whose atoms are written as $x - y \bowtie c$ where $\bowtie$ is an arithmetic operator, x and y are variables, and $c \in \mathbb{R}$. The theory $\mathcal{IDL}$ of *integer difference logic* is defined as $\mathcal{RDL}$ but it is a sub-theory of $\mathcal{LAI}$. Both the constraint $\mathcal{RDL}$- and $\mathcal{IDL}$-satisfiability problems are decidable and polynomial. For more details about these and other theories, see, e.g., [28,9].

Many applications require to reason about conjunctions of constraints coming from several theories $T_1, ..., T_n$. In this situation, it is easy to build a theory solver by reusing those available for the theories $T_1, ..., T_n$ as follows. Let $\alpha = \alpha_1(\underline{v}_1) \wedge \cdots \wedge \alpha_n(\underline{v}_n)$ be a *composite constraint*, i.e. α_i is a literal over the signature of T_i for $i = 1, ..., n$ whose variables $\underline{v}_i$ are disjoint with those of the other

constraints. Then, α is satisfiable iff each α_i is T_i-satisfiable and it is unsatisfiable otherwise, i.e. there exists $i \in \{1, ..., n\}$ such that α_i is T_i-unsatisfiable. Thus, if the constraint T_i-satisfiability problem is decidable and polynomial for each $i = 1, ..., n$, then it is also decidable and polynomial to check the satisfiability of composite constraints.

4 Solving the MS-AU and CD-SAU Problems

We explain how RBAC-DR and RBAC-NDR policies can be specified by using theories and how this allows us to prove the decidability of the MS-AU and CD-SAU problems. The proof is constructive and reduces both problems to a sequence of SMT problems that can be efficiently solved by invoking state-of-the-art SMT solvers.

4.1 Specifying RBAC-DR and RBAC-NDR Policies with Theories

The idea is to use quantifier-free formulae of a suitable theory to specify the conditions of (signed) authorization rules. Formally, we assume the availability of a *theory* $T_A = (\Sigma_A, \mathcal{C}_A)$ *of attributes* and define a *RBAC-DR policy with background theory* T_A *of attributes* as a tuple $(U, R, P, \underline{a}, \underline{s}, AU_{QF}, PA, RH)$ where U, R, P, PA, and RH are as specified above, and the following conditions hold: **(C1)** each $a \in \underline{a}$ is a first-order variable of sort S_a in Σ_A and each $s_a \in \underline{s}$ is a mapping $s_a : U \to 2^{S_a^{\mathcal{M}_A}}$ for some $\mathcal{M}_A \in \mathcal{C}_A$ (where $\sigma^{\mathcal{M}_A}$ is the interpretation in $\mathcal{M}_A$ of the symbol $\sigma \in \Sigma$; e.g., $S_a^{\mathcal{M}_A}$ is a subset of the domain of $\mathcal{M}_A$), **(C2)** for every $e \in S^{\mathcal{M}_A}$ there exists a constant c of sort S_a in Σ_A such that $c^{\mathcal{M}_A} = e$ for some $\mathcal{M}_A \in \mathcal{C}_A$,[1] and **(C3)** the set AU_{QF} contains finitely many pairs of the form $\varphi(\underline{b}) \rightsquigarrow r$, called *syntactic authorization rules*, where φ is a quantifier-free formula built out of the symbols of Σ_A and the variables in $\underline{b}$, that is a sub-sequence of $\underline{a}$.

The crucial observation is that each quantifier-free formula $\varphi(\underline{b})$ in a syntactic authorization rule $\varphi(\underline{b}) \rightsquigarrow r$ in AU_{QF} defines a condition $C(\underline{b})$, in the sense of Section 2.2, as follows: $(e_1, ..., e_m) \in C(\underline{b})$ iff there exists $\mathcal{M}_A \in \mathcal{C}_A$ and a valuation μ such that $\mu(b_j) = e_j \in S_{b_j}^{\mathcal{M}_A}$ for $j = 1, ..., m$ and $\mathcal{M}_A, \mu \models \varphi(\underline{b})$. In the following, the condition $C(\underline{b})$ associated to the quantifier-free formula $\varphi(\underline{b})$ is written as $[[\varphi(\underline{b})]]$. Thus, a RBAC-DR policy $(U, R, P, \underline{a}, \underline{s}, AU_{QF}, PA, RH)$ with background theory $T_A = (\Sigma_A, \mathcal{C}_A)$ of the attributes defines the following RBAC-DR policy: $(U, R, P, \underline{a}, \underline{D}, \underline{s}, AU, PA, RH)$ where $D_a = S_a^{\mathcal{M}_A}$ for each $a \in \underline{a}$ and $AU = \{[[\varphi(\underline{b})]] \rightsquigarrow r \mid (\varphi(\underline{b}) \rightsquigarrow r) \in AU_{QF}\}$ for some $\mathcal{M}_A \in \mathcal{C}_A$. Similar definitions can be given for RBAC-NDR policies and RBAC-DR (RBAC-NDR) family of policies with background theory T_A of attributes.

[1] This condition is not restrictive as it is always possible to add a constant c_e for each element e in the domain whose interpretation is e itself. The addition of these constants does not change the set of satisfiable quantifier-free formulae (see, e.g., [10]).

Example 2. For Example 1, we consider composite constraints from the enumerated data-type theory $\mathcal{ED}(\{Italy, France, Japan, Indonesia, ...\}, C)$ for the set WC of world countries and the theory $\mathcal{IDL}$ for the constraints on the age (see Section 3). The attribute *country* is also seen as a variable of sort C and the attribute *age* as a variable of sort $\mathbb{Z}$. As a consequence, the syntactic authorization rules corresponding to the authorization rules in Example 1 are:

$\rho_1 : -age \leq -20 \wedge (country = Japan \vee country = Indonesia) \rightsquigarrow Adult,$
$\rho_2 : -age \leq -18 \wedge (country = Italy \vee country = France \vee \cdots) \rightsquigarrow Adult,$
$\rho_3 : -age \leq -13 \wedge age \leq 17 \wedge (country = Italy \vee country = France \vee \cdots) \rightsquigarrow Teen,$
$\rho_4 : age \leq 12 \wedge (country = Italy \vee country = France \vee \cdots) \rightsquigarrow Child$

Notice that $country \in \{c_1, ..., c_n\}$ has been translated as $\bigvee_{i=1}^{n} country = c_i$. □

By generalizing the observations in the example, it is not difficult to see that the language to express authorization rules introduced in [2] can be translated to syntactic authorization rules provided that a suitable background theory of attributes is used.

Proposition 1. *Given a RBAC-DR policy* $(U, R, P, \underline{a}, \underline{s}, AU_{QF}, PA, RH)$ *with background theory* $T_A = (\Sigma_A, \mathcal{C}_A)$ *of the attributes. If the SMT(T_A) problem is decidable, then it is decidable to check if a user* $u \in U$ *is a member of a role* $r \in R$.

An obvious corollary of this proposition and the fact that the membership to RH and PA is computable is that checking whether a user is an implicit member of a role or he/she has a permission are also decidable.

In the following, without loss of generality, we assume that **(A1)** any condition of an authorization rule is T_A-satisfiable but not T_A-valid (i.e. its negation is also T_A-satisfiable) and **(A2)** the conditions of the syntactic authorization rules in AU_{QF} are constraints and not arbitrary Boolean combinations of atoms. The two situations ruled out by assumption **(A1)** can be automatically identified under the assumption that the SMT(T_A) problem is decidable and can be safely discarded as uninteresting: those that are T_A-satisfiable never assigns a user to a role while those that are T_A-valid assigns any user to a role (this kind of problems have been considered in the context of a similar rule-based specification framework for access control policies [25]). To see why assumption **(A2)** is without loss of generality, consider a rule $\varphi \rightsquigarrow r$ in AU_{QF} where φ is a quantifier-free formula (and $r \in R$). Since it is possible to transform φ into disjunctive normal form, i.e. into a formula of the form $\bigvee_{j=1}^{d} \alpha_j$ where α_j is a constraint (for $j = 1, ..., d$ and $d \geq 1$), we can consider the set $AU'_{QF} := AU_{QF} \setminus \{\varphi \rightsquigarrow r\} \cup \{\alpha_j \rightsquigarrow r \mid j = 1, ..., d\}$. Clearly, the user-role assignment relations induced by AU_{QF} and by AU'_{QF} are the same. By iterating the process, we derive a set of syntactic authorization rules whose conditions are constraints only.

4.2 The MS-AU Problem

We consider the MS-AU problem (recall its definition at the end of Section 2.2) for RBAC-DR policies with a background theory of the attributes, i.e. given a

```
Input: c: array[0..N-1] of constraints
       r: array[0..N-1] of (signed) roles
1: S := ∅;
2: FOR i=0 TO N-2 DO
3:  FOR j=i+1 TO N-1 DO
4:   f1 := opposite(r[i],r[j]); f2 := related(c[i],c[j])
5:   IF f2 THEN
6:    IF find(i)≠find(j) THEN
7:     i2j := check_{T_A}(c[i] ∧ ¬ c[j])
8:     IF i2j=unsat THEN S := S∪{(i,j,f1,f2)};
9:     j2i := check_{T_A}(c[j] ∧ ¬ c[i])
10:    IF j2i=unsat THEN S := S∪{(j,i,f1,f2)};
11:    IF (i2j=unsat AND j2i=unsat) THEN union(i,j)
12:   ELSE S := S∪{(i,j,f1,f2),(j,i,f1,f2)};
13:  ELSE S := S∪{(i,j,f1,f2)}
```

Fig. 1. Solving the MS-AU and CD-SAU problems

RBAC-DR family of policies $(U, R, P, \underline{a}, AU_{QF}, PA, RH)$ with background theory T_A of the attribute, the MS-AU problem amounts to checking whether $\rho_1 \sqsupseteq \rho_2$ for every pair ρ_1 and ρ_2 of syntactic authorization rules in AU_{QF}.

The key observation is that we can reduce the problem of establishing whether $\rho_1 \sqsupseteq \rho_2$ to an SMT(T_A) problem involving the conditions of the rules ρ_1 and ρ_2. Before being able to formally state this, we need to introduce the following notion. Two rules $(\alpha_1(\underline{b}_1) \rightsquigarrow r_1)$ and $(\alpha_2(\underline{b}_2) \rightsquigarrow r_2)$ are *syntactically related* if $\underline{b}_1 \cap \underline{b}_2 \neq \emptyset$ (by abuse of notation, we consider the sequence $\underline{b}_i$ as a set, $i = 1, 2$); otherwise, they are *syntactically unrelated*.

Proposition 2. *Let $(R, P, \underline{a}, AU_{QF}, PA, RH)$ be a RBAC-RD family of policies with background theory T_A of the attributes and $\rho_i = (\alpha_i(\underline{b}_i) \rightsquigarrow r_i)$ be an authorization rule in AU_{QF} for $i = 1, 2$. Then, the following facts hold:*

(a) *if ρ_1 and ρ_2 are syntactically unrelated, then $\rho_1 \not\sqsupseteq \rho_2$ and $\rho_2 \not\sqsupseteq \rho_1$ and*
(b) *if ρ_1 and ρ_2 are syntactically related, then $\rho_1 \sqsupseteq \rho_2$ iff $\alpha_1(\underline{b}_1) \rightarrow \alpha_2(\underline{b}_2)$ is T_A-valid (or, equivalently, $\alpha_1(\underline{b}_1) \wedge \neg\alpha_2(\underline{b}_2)$ is T_A-unsatisfiable).*

Assuming that the SMT(T_A) problem is decidable, it is possible to automatically check whether a rule is more senior than the other by Proposition 2. This is the idea underlying the algorithm in Figure 1 for automatically solving the MS-AU problem. Let $(R, P, \underline{a}, AU_{QF}, PA, RH)$ be a RBAC-RD family of policies with background theory T_A where $|AU_{QF}| = \mathtt{N}$. The input to the algorithm are two arrays `c` and `r` such that for each rule $\alpha \rightsquigarrow r$ in AU_{QF} there exists `i` in the range `[0..N-1]` such that `c[i]` $= \alpha$ and `r[i]` $= r$ (the latter will contain signed roles only when solving CD-SAU problems, see Section 4.3 below). The variable `S` $\subseteq [0, N-1] \times [0, N-1] \times \{true, false\} \times \{true, false\}$ stores increasingly precise approximations of the relation $\sqsupseteq$ in a sense to be made precise below. The Boolean function `related(c[i], c[j])` returns `true` iff the rules `c[i]` $\rightsquigarrow$ `r[i]` and `c[j]` $\rightsquigarrow$ `r[j]` are syntactically related. The Boolean function `opposite` is the constant function `false` so that the third component of the tuples in `S` is always set

to `false` (this will be important only when solving CD-SAU problems, see Section 4.3 below). The function `find` and the procedure `union` form the interface to a union-find data structure (see, e.g., [30]) to maintain a set of equivalence classes: `find(i)` returns the representant of the equivalence class to which the rule $\mathtt{c[i]} \rightsquigarrow \mathtt{r[i]}$ belongs to and `union(i, j)` merges the equivalence classes to which the rules $\mathtt{c[i]} \rightsquigarrow \mathtt{r[i]}$ and $\mathtt{c[j]} \rightsquigarrow \mathtt{r[j]}$ belong to. The function $\mathtt{check}_{T_A}(\varphi)$ returns `sat` (`unsat`, resp.) iff the quantifier-free Σ_A-formula φ is T_A-satisfiable (T_A-unsatisfiable, resp.) and it is implemented by invoking an available solver supporting the solution of SMT(T_A) problems.

The algorithm works by enumerating pairs of rules (the two nested loops at lines 2 and 3) to establish whether they are syntactically related (flag `f2` line 4). In case they are not, the tuple $(\mathtt{i}, \mathtt{j}, \mathit{false}, \mathit{false})$ is added to `S` (line 13) meaning that the pair $(\mathtt{i}, \mathtt{j})$ of rules cannot be compared with respect to $\sqsupseteq$. Otherwise, the union-find data structure is queried (line 6) in order to establish if the rules identified by `i` and `j` are already in the same equivalence class. If this is the case, the tuples $(\mathtt{i}, \mathtt{j}, \mathit{false}, \mathit{true})$ and $(\mathtt{j}, \mathtt{i}, \mathit{false}, \mathit{true})$ are added to `S` (line 12) meaning that both $\mathtt{i} \sqsupseteq \mathtt{j}$ and $\mathtt{j} \sqsupseteq \mathtt{i}$. Otherwise, it is checked if $\mathtt{i} \sqsupseteq \mathtt{j}$ (line 7, by testing if $\mathtt{c[i]}$ implies $\mathtt{c[j]}$ modulo T_A or, equivalently, the conjunction of $\mathtt{c[i]}$ with the negation of $\mathtt{c[j]}$ is T_A-unsatisfiable) and the tuple $(\mathtt{i}, \mathtt{j}, \mathit{false}, \mathit{true})$ is added to `S` (line 8). The same is done to establish if $\mathtt{j} \sqsupseteq \mathtt{i}$ (lines 9 and 10). If both previous tests (lines 8 and 10) have been successful, then the two equivalence classes to which `i` and `j` belong to are merged (line 11).

Lemma 1. *Let $(R, P, \underline{a}, AU_{QF}, PA, RH)$ be a family of RBAC-RD policies with background theory T_A of the attributes whose SMT(T_A) problem is decidable and $|AU_{QF}| = \mathtt{N}$. Then, the following facts hold after the execution of the algorithm in Figure 1:* ***(a)*** *$(\mathtt{i}, \mathtt{j}, \mathit{false}, \mathit{true}) \in \mathtt{S}$ iff $\mathtt{c[i]} \rightsquigarrow \mathtt{r[i]} \sqsupseteq \mathtt{c[j]} \rightsquigarrow \mathtt{r[j]}$,* ***(b)*** *$\mathtt{find(i)} = \mathtt{find(j)}$ iff $\mathtt{c[i]} \rightsquigarrow \mathtt{r[i]} \sqsupseteq \mathtt{c[j]} \rightsquigarrow \mathtt{r[j]}$ and $\mathtt{c[j]} \rightsquigarrow \mathtt{r[j]} \sqsupseteq \mathtt{c[i]} \rightsquigarrow \mathtt{r[i]}$, i.e. the union-find data structure stores the equivalence classes of the "more senior than" relation over AU_{QF}, and* ***(c)*** *the number of invocations to $\mathtt{check}_{T_A}$ is at most $\mathtt{N}(\mathtt{N}-1)$.*

The cost of invoking the function `opposite` is constant (recall that this is a constant function returning `false` when solving the MS-AU problem) while that of `related` is linear in $\underline{a}$ by using bit-strings to represent sequences of variables occurring in the conditions of the rules. The union-find algorithm in [30] takes almost constant (amortized) time for invoking both the `find` and `union` operations (more precisely, it takes $O(A^{-1}(N))$ where A^{-1} is the inverse of the Ackermann function; since for any practical values of N, $A^{-1}(N)$ is bounded by 5, each invocation to `find` and `union` can be considered as constant). Thus, the complexity of the algorithm is clearly dominated by the number of invocations to $\mathtt{check}_{T_A}$. Notice that the function is invoked on quantifier-free formulae obtained by conjoining a constraint α_i with the negation of a constraint $\neg\alpha_j$ (i.e. a clause). This allows us to characterize the complexity of the MS-AU problem. Formally, we introduce the following notion. A theory T is *convex* (see, e.g., [23]) iff whenever a constraint implies a clause, it also implies one of the literals in the clause. Examples of convex theories are $\mathcal{EUF}$, $\mathcal{ED}$, $\mathcal{LAR}$, and $\mathcal{RDL}$ whose

constraint satisfiability problem is polynomial (convexity of $\mathcal{LAR}$ derives from the convexity of linear algebra). Instead, both $\mathcal{LAI}$ and its sub-theory $\mathcal{IDL}$ are not convex (e.g., $x \geq 0 \wedge x \leq 1 \wedge \neg(x \neq 0 \wedge x \neq 1)$ is $\mathcal{LAI}$-unsatisfiable but neither $x \geq 0 \wedge x \leq 1 \wedge x = 0$ nor $x \geq 0 \wedge x \leq 1 \wedge x = 1$ is $\mathcal{LAI}$-unsatisfiable).

Theorem 1. *If the SMT problem of the background theory T_A of the attributes is decidable, then the MS-AU problem is decidable. If furthermore T_A is convex and the constraint T_A-satisfiability problem is polynomial, then the MS-AU problem is also polynomial.*

When the background theory of the attributes is $\mathcal{EQ}$, $\mathcal{LAR}$, $\mathcal{RDL}$, or we consider composite constraints of these theories, the MS-AU problem is decidable and polynomial since all these theories are decidable and polynomial (see Section 3). When the background theory of the attributes is $\mathcal{ED}$, $\mathcal{LAI}$, or $\mathcal{IDL}$, the MS-AU problem is decidable and NP-complete. While this is obvious for $\mathcal{ED}$ and $\mathcal{LAI}$ (whose constraint satisfiability problem is already NP-complete), it is less so for $\mathcal{IDL}$ since its constraint satisfiability problem is polynomial. Since $\mathcal{IDL}$ is not convex, the problem of checking the satisfiability of a constraint with a clause becomes NP-complete [17]. Thus, besides polynomial constraint satisfiability, convexity is crucial to have a polynomial MS-AU problem.

4.3 The CD-SAU Problem

We consider the CD-SAU problem (recall its definition at the end of Section 2.3) for RBAC-NDR policies with a background theory of the attributes, i.e. given a RBAC-NDR family of policies $(R, P, \underline{a}, AU_{QF}, PA, RH)$ with background theory T_A of the attributes, the CD-SAU problem amounts to checking whether for each signed authorization rule $\alpha_1(\underline{b}_1) \rightsquigarrow +r$ in AU_{QF}, there is no rule $\alpha_2(\underline{b}_2) \rightsquigarrow -r$ in AU_{QF} such that a user u satisfies both $\alpha_1(\underline{b}_1)$ and $\alpha_2(\underline{b}_2)$ under some user-attribute mapping $\underline{s}$. We say that there is a *conflict* between rule $\alpha_1(\underline{b}_1) \rightsquigarrow +r$ and $\alpha_2(\underline{b}_2) \rightsquigarrow -r$ if there exists a user u satisfying both $\alpha_1(\underline{b}_1)$ and $\alpha_2(\underline{b}_2)$ under some user-attribute mapping $\underline{s}$. Interestingly, it is possible to distinguish two types of conflicts depending on the fact that the rules can or cannot be compared by the "more senior than" relation. Formally, the rules ρ_1 and ρ_2 are *relevant* iff $\rho_1 \sqsupseteq \rho_2$ or $\rho_1 \sqsupseteq \rho_2$, and are *irrelevant* otherwise. If two rules are syntactically unrelated, they are also irrelevant by Proposition 2 **(a)**.

We solve the CD-SAU problem by using again the algorithm in Figure 1 where the array `r` stores signed roles, the auxiliary function `opposite(r[i], r[j])` returns `true` iff `r[i]` $= +r$ and `r[j]` $= -r$ or `r[i]` $= -r$ and `r[j]` $= +r$, and all the other data structures and functions are as described in Section 4.2. As anticipated in Section 4.2, the third component of the relation `S` is important and distinguishes between relevant and irrelevant pairs of rules.

As before, the algorithm enumerates pairs of rules and establishes whether they have signed roles $+r$ and $-r$ and they are syntactically related (flags `f1` and `f2` at line 4, respectively). In case the two rules are not syntactically related, the tuple $(\mathtt{i}, \mathtt{j}, \mathtt{f}_1, \mathit{false})$ is added to `S` (line 13) meaning that the rules identified by `i` and `j` are irrelevant and there is a conflict between them when $\mathtt{f}_1$ is *true*.

Otherwise, the union-find data structure is queried (line 6) in order to establish if the rules identified by $\mathtt{i}$ and $\mathtt{j}$ are already in the same equivalence class. If this is the case, the tuples $(\mathtt{i}, \mathtt{j}, \mathtt{f}_1, \mathit{true})$ and $(\mathtt{j}, \mathtt{i}, \mathtt{f}_1, \mathit{true})$ are added to $\mathtt{S}$ (line 12) meaning than the rules identified by $\mathtt{i}$ and $\mathtt{j}$ are relevant and there is a conflict between them when $\mathtt{f}_1$ is *true*. Otherwise, it is checked if $\mathtt{c}[\mathtt{i}] \rightsquigarrow \mathtt{r}[\mathtt{i}] \sqsupseteq \mathtt{c}[\mathtt{j}] \rightsquigarrow \mathtt{r}[\mathtt{j}]$ (line 7) and the tuple $(\mathtt{i}, \mathtt{j}, \mathtt{f}_1, \mathit{true})$ is added to $\mathtt{S}$ (line 8), the same is done to establish if $\mathtt{c}[\mathtt{j}] \rightsquigarrow \mathtt{r}[\mathtt{j}] \sqsupseteq \mathtt{c}[\mathtt{i}] \rightsquigarrow \mathtt{r}[\mathtt{i}]$ (lines 9 and 10): the rules are relevant and there is a conflict between them when $\mathtt{f}_1$ is *true*. If both previous tests (lines 8 and 10) have been successful, then the two equivalence classes to which $\mathtt{i}$ and $\mathtt{j}$ belong to are merged (line 11).

Lemma 2. *Let $(R, P, \underline{a}, AU_{QF}, PA, RH)$ be a family of RBAC-NDR policies with background theory T_A of the attributes whose SMT(T_A) problem is decidable and $|AU_{QF}| = \mathtt{N}$. Then, after the execution of the algorithm in Figure 1 the following holds: if $(i, j, \tau, \mathit{true}) \in \mathtt{S}$, then there is a conflict between rules $\mathtt{i}$ and $\mathtt{j}$ and the two rules are relevant (irrelevant, resp.) when τ is true (false, resp.) and the number of invocations to $\mathtt{check}_{T_A}$ is at most $\mathtt{N}(\mathtt{N}-1)$.*

We state the second main result of the paper.

Theorem 2. *If the SMT problem of the background theory T_A of the attributes is decidable, then the CD-SAU problem is decidable. If furthermore T_A is convex and the constraint T_A-satisfiability problem is polynomial, then the CD-SAU problem is also polynomial.*

4.4 Experiments

To test the practical viability of our techniques, we have implemented the algorithm of Figure 1 in C. To implement $\mathtt{check}_{T_A}$, we have chosen Yices [31] for its easy-to-use API, although many other state-of-the-art SMT solvers (e.g., Z3 [34]) can be used. We have also implemented a generator of synthetic MS-AU problems; we expect similar performances on CD-SAU problems since the algorithm of Figure 1 can solve also these problems with minimal variations. The starting point is the policy of Example 1. The idea is to randomly generate composite constraints for the conditions of the rules taken from an enumerated data-type theory $\mathcal{ED}(\{v_1, ..., v_V\}, E)$ and $\mathcal{IDL}$: the former corresponds to the set of countries in the world and the latter to the age of users. We randomly generate composite constraints whose literals are $e = v_i$, $e \neq v_i$ (for $i \in \{1, ..., V\}$), $a = k$, $a > k$, and $a < k$ (for $k \in \mathbb{Z}$) where e and a are variables of sort E and $\mathbb{Z}$, respectively. According to Theorem 1, the resulting MS-AU problem is NP-complete as the constraint $\mathcal{ED}$-satisfiability problem is NP-complete and that of $\mathcal{IDL}$ is polynomial but $\mathcal{IDL}$ is not convex. Thus the invocations to $\mathtt{check}_{T_A}$ can be computationally expensive. The random generation of literals in the composite constraints is inspired to Gorrilla [15] that is known to generate difficult problems for many state-of-the-art SMT solvers. The method of [15] has been adapted to satisfy assumption **(A1)** of Section 4.1 and to generate constraints that can be considered as "realistic" conditions of authorization rules, i.e. roughly similar to those in Example 2.

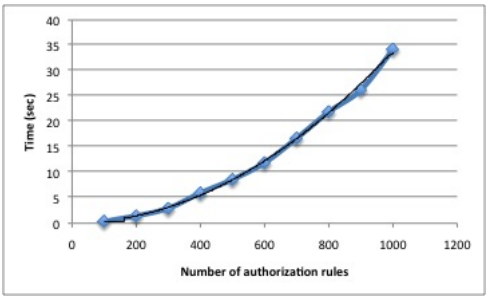

Fig. 2. Experimental results obtained on a MacBook with Intel Core i5 2.53 GHz and 4 GB of RAM running Mac OS X 10.6.6. Timings are averages over 10 random instances

There are three inputs to the benchmark generator: the number V of values of $\mathcal{ED}$, a positive integer M bounding the constant k occurring in the arithmetic literals, and the number N of conditions to be considered. The first two parameters V and M have negligible influence on the performance of our implementation of the algorithm in Figure 1. As a consequence, Figure 2 shows the plot of the time taken to solve an MS-AU problem for the last parameter, i.e. an increasing number N of rules (ranging from 100 to 1,000).[2] The behaviour of the algorithm is clearly quadratic. This is possible because we invoke the SMT solver only (around) 1/4 of the potential N^2 calls, thanks to the notion of two rules being syntactically unrelated (see before Proposition 2 for the definition) and the use of the union-find data structure that provides us with a computationally cheap method to deduce the transitive chains of the "more senior than relation." We believe that this gives a first important evidence of the practical viability of our technique.

5 Related Work and Discussion

The problem of combining RBAC with more flexible methods of assigning users to roles has been considered several times (see, e.g., [21,6]). In this line of work, to the best of our knowledge, no automated analysis technique has been proposed to assist designers to detect redundancies and inconsistencies of policies as we do here. We have presented an abstract rule-based framework for the integration of attributes in RBAC inspired to [2]. The MS-AU and the CD-SAU problems are inspired to problems informally characterized in [3] and [4]. The idea of using SMT solvers for their efficient solution together with their decidability and complexity results are new; except for [7,5] that exploit SMT solvers to solve administrative RBAC problems. In [32,14], techniques to solve (what we call) CD-SAU problems by using description logic and tableaux reasoners, respectively, are described. Neither a complexity characterization nor an extensive experimental evaluation are proposed and the techniques do not support

[2] The sources of the program used to perform the experiments are available at `http://st.fbk.eu/SilvioRanise#Papers`.

rich background theories of the attributes as we do here. The policy specification languages in [33,19] can express authorization policies with attributes. They support policies that go beyond those considered in this paper but do not propose automated techniques for detecting inconsistencies and redundancies as we do here. Furthermore, they adopt a closed (deny-by-default) policy model (where access is allowed if there exists a corresponding (positive) authorization and is denied otherwise [13,26]) rather than an open (allow-by-default) policy as we do in this paper. We leave it to future work the study of the impact of using a closed policy model on the decidability and complexity of the MS-AU and CD-SAU problems. We notice that our framework (including Theorems 1 and 2) can be easily extended to cope with conditions about the environment (e.g., time of the day) as done in [33].

Several works [29,1,22,12,11,8,20] have proposed techniques to detect inconsistencies and redundancies in XACML or extensions of RBAC policies by leveraging a variety of verification engines. None of these works provides decidability and complexity results of the analysis techniques as we do in this paper.

Acknowledgements. This work was partially supported by the "Automated Security Analysis of Identity and Access Management Systems (SIAM)" project funded by Provincia Autonoma di Trento in the context of the "team 2009 - Incoming" COFUND action of the European Commission (FP7).

References

1. Adi, K., Bouzida, Y., Hattak, I., Logrippo, L., Mankovskii, S.: Typing for Conflict Detection in Access Control Policies. In: Babin, G., Kropf, P., Weiss, M. (eds.) MCETECH 2009. LNBIP, vol. 26, pp. 212–226. Springer, Heidelberg (2009)
2. Al-Kahtani, M., Sandhu, R.: A Model for Attribute-Based User-Role Assignment. In: Proc. of 18th Annual Comp. Sec. App. Conf., Las Vegas, Nevada (2002)
3. Al-Kahtani, M., Sandhu, R.: Induced Role Hierarchies with Attribute-Based RBAC. In: Proc. of 8th ACM SACMAT (2003)
4. Al-Kahtani, M., Sandhu, R.: Rule-based RBAC with negative authorization. In: Proc. of 20th Annual Comp. Sec. App. Conf., pp. 405–415 (2004)
5. Alberti, F., Armando, A., Ranise, S.: Efficient Symbolic Automated Analysis of Administrative Role Based Access Control Policies. In: Proc. of 6th ACM Symp. on Info., Computer and Comm. Security, ASIACCS 2011 (2011)
6. Ardagna, C., De Capitani di Vimercati, S., Paraboschi, S., Pedrini, E., Samarati, P., Verdicchio, M.: Expressive and Deployable Access Control in Open Web Service Applications. IEEE Trans. on Serv. Comp. (TSC) 4(2), 96–109 (2011)
7. Armando, A., Ranise, S.: Automated Symbolic Analysis of ARBAC-Policies. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM 2010. LNCS, vol. 6710, pp. 17–34. Springer, Heidelberg (2011)
8. Autrel, F., Cuppens, F., Cuppens, N., Coma, C.: MotOrBAC 2: a security policy tool. In: 3rd Conf. SARSSI, pp. 13–17 (2008)
9. De Moura, L., Bjørner, N.: Satisfiability modulo theories: introduction and applications. Commun. ACM 54, 69–77 (2011)
10. Enderton, H.B.: A Mathematical Introduction to Logic. Academic Press, New York (1972)

11. Fisler, K., Krishnamurthi, S., Meyerovich, L.A., Tschantz, M.C.: Verification and change-impact analysis of access control policies. In: Int. Conf. on Sw Eng. (ICSE), pp. 196–206 (2005)
12. Hughes, G., Bultan, T.: Automated Verification of Access Control Policies Using a SAT Solver. Int. J. on Sw Tools for Tech. Trandf. (STTT) 10(6), 473–534 (2008)
13. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible support for multiple access control policies. ACM Trans. DB Syst. 26, 214–260 (2001)
14. Kamoda, H., Yamaoka, M., Matsuda, S., Broda, K., Sloman, M.: Access Control Policy Analysis Using Free Variable Tableaux. Trans. of Inform. Proc. Soc. of Japan, 207–221 (2006)
15. Korovin, K., Voronkov, A.: GoRRiLA and Hard Reality. In: Clarke, E., Virbitskaite, I., Voronkov, A. (eds.) PSI 2011. LNCS, vol. 7162, pp. 243–250. Springer, Heidelberg (2012)
16. Kuhn, D.R., Coyne, E.J., Weil, T.R.: Adding Attributes to Role Based Access Control. IEEE Computer 43(6), 79–81 (2010)
17. Lahiri, S.K., Musuvathi, M.: An Efficient Decision Procedure for UTVPI Constraints. In: Gramlich, B. (ed.) FroCos 2005. LNCS (LNAI), vol. 3717, pp. 168–183. Springer, Heidelberg (2005)
18. Li, N., Mitchell, J.C.: DATALOG with Constraints: A Foundation for Trust Management Languages. In: Dahl, V. (ed.) PADL 2003. LNCS, vol. 2562, pp. 58–73. Springer, Heidelberg (2003)
19. Li, N., Mitchell, J.C.: RT: A Role-based Trust-management Framework. In: 3rd DARPA Infor. Surv. Conf. and Exp. (DISCEX III), pp. 201–212 (2003)
20. Lin, D., Rao, P., Bertino, E., Li, N., Lobo, K.: EXAM: a comprehensive environment for the analysis of access control policies. IJIS 9, 253–273 (2010)
21. Lupu, E., Sloman, M.: Reconciling Role Based Management and Role Based Access Control. In: 2nd ACM Ws. on Role Based Acc. Contr., pp. 135–142 (1997)
22. Mankai, M., Logrippo, L.: Access Control Policies: Modeling and Validation. In: Proc. of NOTERE, pp. 85–91 (2005)
23. Nelson, C.G., Oppen, D.: Simplification by Cooperating Decision Procedures. ACM Trans. on Programming Languages and Systems 1(2), 245–257 (1979)
24. Ranise, S., Tinelli, C.: The SMT-LIB Standard: Version 1.2, `http://goedel.cs.uiowa.edu/smtlib/papers/format-v1.2-r06.08.30.pdf`
25. Ribeiro, C., Zúquete, A., Ferreira, P., Guedes, P.: Security Policy Consistency. In: 1st Ws. on Rule-Based Constr. Reas. and Progr. CoRR cs.LO/0006045 (2000)
26. Samarati, P., De Capitani di Vimercati, S.: Access Control: Policies, Models, and Mechanisms. In: Focardi, R., Gorrieri, R. (eds.) FOSAD 2000. LNCS, vol. 2171, pp. 137–196. Springer, Heidelberg (2001)
27. Sandhu, R., Coyne, E., Feinstein, H., Youmann, C.: Role-Based Access Control Models. IEEE Computer 2(29), 38–47 (1996)
28. Sebastiani, R.: Lazy Satisfiability Modulo Theories. Journal on Satisfiability, Boolean Modeling and Computation, JSAT 3, 141–224 (2007)
29. Shaikh, R., Adi, K., Logrippo, L., Mankovski, S.: Inconsistency Detection Method for Access Control Policies. In: IEEE 6th IAS, pp. 204–209 (2010)
30. Tarjan, R.E.: Efficiency of a Good But Not Linear Set Union Algorithm. Journal of the ACM 22(2), 215–225 (1975)
31. Yices, `http://yices.csl.sri.com/`
32. Yu, H., Xie, Q., Che, H.: Research on Description Logic Based Conflict Detection Methods for RB-RBAC Model. In: 4th Int. Conf. on AMT, pp. 335–339 (2006)
33. Yuan, E., Tong, J.: Attributed Based Access Control (ABAC) for Web Services. In: Proc. of IEEE ICWS, pp. 561–569 (2005)
34. Z3, `http://research.microsoft.com/en-us/um/redmond/projects/z3`