Automated Analysis of Semantic-Aware Access Control Policies: a Logic-Based Approach

Alessandro Armando*†, Roberto Carbone†, and Silvio Ranise†

*AI-Lab, DIST – Università di Genova
Viale Causa 13, 16145 Genova, Italy
armando@dist.unige.it
†Security and Trust Unit, FBK
Via Sommarive 18, 38123 Trento, Italy
{armando, carbone, ranise}@fbk.eu

Abstract—As the number and sophistication of on-line applications increase, there is a growing concern on how access to sensitive resources (e.g., personal health records) is regulated. Since ontologies can support the definition of fine-grained policies as well as the combination of heterogeneous policies, semantic technologies are expected to play an important role in this context. But understanding the implications of the access control policies of the needed complexity goes beyond the ability of a security administrator. Automatic support to the analysis of access control policies is therefore needed. In this paper we present an automatic analysis technique for access control policies that reduces the reachability problem for access control policies to satisfiability problems in a decidable fragment of first-order logic for which efficient solvers exist. We illustrate the application of our technique on an access control model inspired by a Personal Health Application of real-world complexity.

Keywords-semantic-aware access control; automatic analysis; logic-based methods;

I. Introduction

An increasing number of security-sensitive applications for e-business, e-health, and e-government are nowadays run over the Internet. A serious concern is how to regulate access to sensitive resources (e.g., health records) handled by these applications. Classical access control models are unsatisfactory for a number of reasons:

Policy Administration. Traditional access control models, e.g., Role-Based Access Control (RBAC) [1], assume that roles are assigned to users statically by administrators. Although changes to an RBAC policy are possible, they are assumed to happen outside the system normal operation and be carried out by administrators (possibly following an additional, administrative policy). This is undoubtedly a severe limitation for those applications that are intended for the general public, where the set of possible users is unknown, changes to the access control policy can be carried out by the users themselves, and privileges can even be delegated. The problem is particularly acute for Personal Health Applications (PHAs) (e.g., Google Health, Microsoft

HealthVault), where users own their Personal Health Records (PHRs) and retain the exclusive right to define fine-grained access control policies prescribing who (e.g., doctor, relative, friend) can access their PHRs and how (i.e. which operations should be allowed).

Policy Integration. With the advent of the Software-as-a-Service paradigm, resources are no longer stored and managed by a single application and external applications (e.g. applications developed and maintained by different service providers) can be given access to these resources in order to obtain additional functionalities. The security policy governing the access to sensitive resources may thus span several applications and the combination of different policies (possibly designed and administered by different service providers) must thus be taken into account. For instance, Google Health allows the users to give access to their PHRs to 38 tools offered by third parties providing a variety services ranging from medication reminder to on-line medical assistance.

Semantic technologies can be used to mitigate these problems. For instance, available ontologies for agents and resources can be profitably used to simplify and disambiguate the specification and management of fine-grained access control policies. Ontologies can also provide the foundation for the interoperability of services (see, e.g. [2]) and the combination of the respective access control policies. But understanding the implications of the resulting access control policies goes beyond the ability of a security administrator, let alone an average user. The need for automatic analysis techniques for semantic-aware access control policies is therefore urgent.

In previous work we have proposed a symbolic analysis technique for Administrative RBAC policies [3], [4]. The technique reduces the problem of finding a violation of an authorization constraint (e.g., an untrusted user can get the right to perform a critical action on a shared resource) to first-order satisfiability problems. To this end, we use a decidable fragment of first-order logic, called the Bernays-

Schönfinkel-Ramsey (BSR) fragment [5]. The key advantage of using BSR is two-fold. On the one hand, BSR is related to Datalog, which have been widely used for the specification of access control policies (see, e.g., [6]). On the other hand, the use of BSR allows us to leverage state-of-the-art automated reasoners (e.g. SPASS [7], [8]) or Satisfiability Modulo Theories (SMT) solvers (e.g. Z3 [9]) for efficient satisfiability solving and thus to automate the analysis. As we will see, our technique is capable to analyze security problems where the number of users is finite but unknown. This is an important feature for applications working in open environments where the set of users is unknown a priori.

In this paper we propose an extension to our technique [3], [4] that supports the analysis of a family of fine-grained access control policies for PHRs. The family of policies we consider encompasses the administration of the policy by the user as well as fine-grained access control rules based on hierarchies of agents, actions, and resources.

Structure of the paper: In the next section (Section II) we present an access control model for PHRs of real-world complexity. In Section III we describe our automatic analysis technique and illustrate its application on the family of policies introduced in Section II. We conclude in Section IV with some final remarks.

II. AN ACCESS CONTROL MODEL FOR PERSONAL HEALTH RECORDS

To illustrate our ideas we consider an access control model for PHRs inspired by the one used in the Project HealthDesign Common Platform [10]. Project HealthDesign (www.projecthealthdesign.org) is a project that aims at promoting innovation in personal health information technology. The Project HealthDesign Common Platform is a set of software components that provide common, shared functions to a variety of PHAs. The components are implemented as web services that PHAs may access via standard web interfaces. Services exist for storing observations and medications, as well as for providing authentication and access-control functions.

We define an access control policy for a PHR (owned by user u_o) as a tuple $\pi = (u_o, U, R, P, UA, PA)$, where:

- U is the set of the user accounts and $u_o \in U$;
- R is a set of roles endowed with the hierarchy relation $\sqsupseteq_R$ defined in Figure 1 (inspired by the ontology in Table 7.5 of [10]);
- $UA \subseteq (U \times R)$ is the user-role assignment relation;
- $P = (Act \times Res)$ is the set of permissions, where
  - Act is the set of actions endowed with the hierarchy relation $\sqsupseteq_{Act}$ of Figure 2 (derived by Table 7.6 in [10]);
  - Res is the set of resources endowed with the hierarchy relation $\sqsupseteq_{Res}$ of Figure 3 (derived by Table 7.7 in [10]). In Figure 3, the resources in

angle brackets are placeholders for sets of categories, e.g. MedicationCategory stands for Central Nervous System Agents, Diabetic Agents, etc.

The hierarchy relations over Act and Res induce a hierarchy relation $\sqsupseteq_P \subseteq P \times P$ defined as follows: $(act, res) \sqsupseteq_P (act', res')$ iff $act \sqsupseteq_{Act} act'$ and $res \sqsupseteq_{Res} res'$. Since $\sqsupseteq_{Act}$ and $\sqsupseteq_{Res}$ are partial orders, it is easy to see that also $\sqsupseteq_P$ is so.

- $PA = ((U \cup R) \times P)$ is the permission assignment relation; it supports the direct assignment of permissions to individual users as well as the indirect assignment through the assignment of permissions to roles.

A *user u can execute act on res in π* iff

1) $(u, p) \in PA$ for some permission p such that $p \sqsupseteq_P (act, res)$, or
2) there exist roles $r, r' \in R$ such that $(u, r) \in UA$, $r \sqsupseteq_R r'$, and $(r', p) \in PA$ for some permission p such that $p \sqsupseteq_P (act, res)$.

For instance, if the access control policy of Bob's PHR is such that $(\text{Alice}, p) \in PA$ with $p = (\text{RecordViewing}, \text{MedicalEvents})$, then Alice can view Bob's MedicalEvents (including his MedicalAppointments and MedicationAdministrations because of the resource hierarchy of Figure 3). Notice that Alice has the same privileges (and possibly more) if the policy is such that $(\text{Alice}, p) \notin PA$, but Alice is a Physician and $(\text{Physician}, p) \in PA$.

It must be noted that the policy allows also to specify administration privileges. For instance, if $(\text{Alice}, p) \in PA$ with $p = (\text{Grant}(\text{RecordViewing}), \text{MedicalEvents})$, then Alice can grant the privilege to viewing Bob's MedicalEvents to any other user. This means that Alice can change the current policy, say $\pi = (u_o, U, R, P, UA, PA)$, into a policy $\pi' = (u_o, U, R, P, UA, PA \cup \{(u, (\text{RecordViewing}, \text{MedicalEvents}))\})$ for some arbitrary user $u \in U$.

Delegation is very useful in practice as it allows Alice to grant her privileges on Bob's MedicalEvents to a colleague when, for instance, she is on vacation. However this form of delegation is so liberal that the user may feel uneasy to (or even refrain from) using it. For instance, Bob might be willing to delegate Alice the permission to grant privileges on his MedicalEvents to any Physician, but he may additionally want to be sure that his relatives cannot possibly access to his MedicalEvents (thereby denying access to those relatives of his that are physicians, if any). To support this type of delegation, we assume that Grant/Revoke actions are associated with a finite set Pre of preconditions of the form $+ur$ or $-ur$, where either $ur \in U$ or $ur \in R$. For instance, if $(\text{Alice}, p) \in PA$ with $p = (\text{Grant}(\text{RecordViewing}) \text{ to } \{+\text{Physician}, -\text{FamilyMember}\}, \text{MedicalEvents})$, then Alice can grant the privilege to viewing Bob's MedicalEvents only to users who are physicians but are not Bob's relatives. In other words, Alice can change $\pi = (u_o, U, R, P, UA, PA)$ into $\pi' = (u_o, U, R, P, UA,$
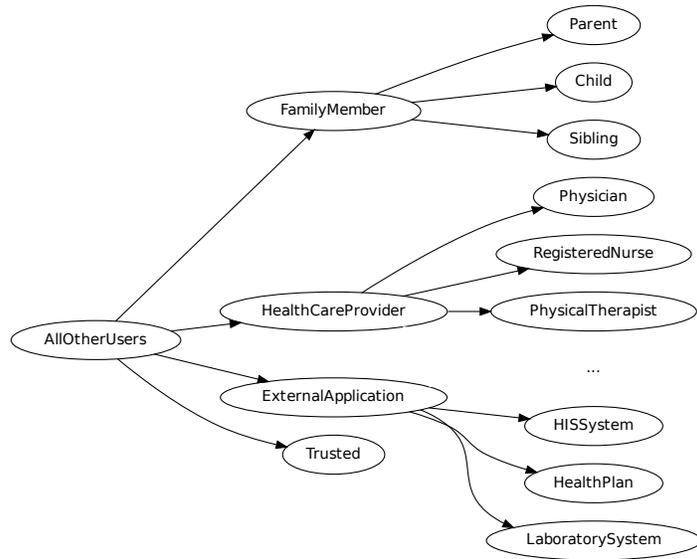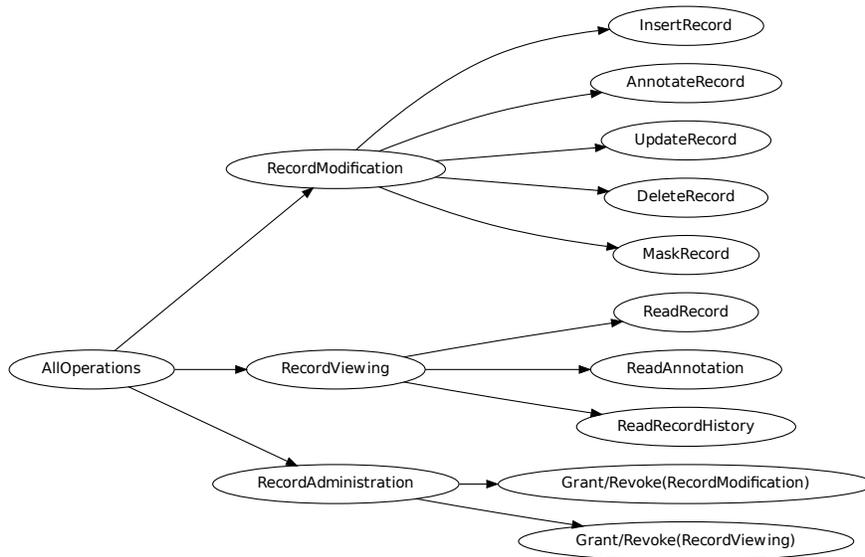
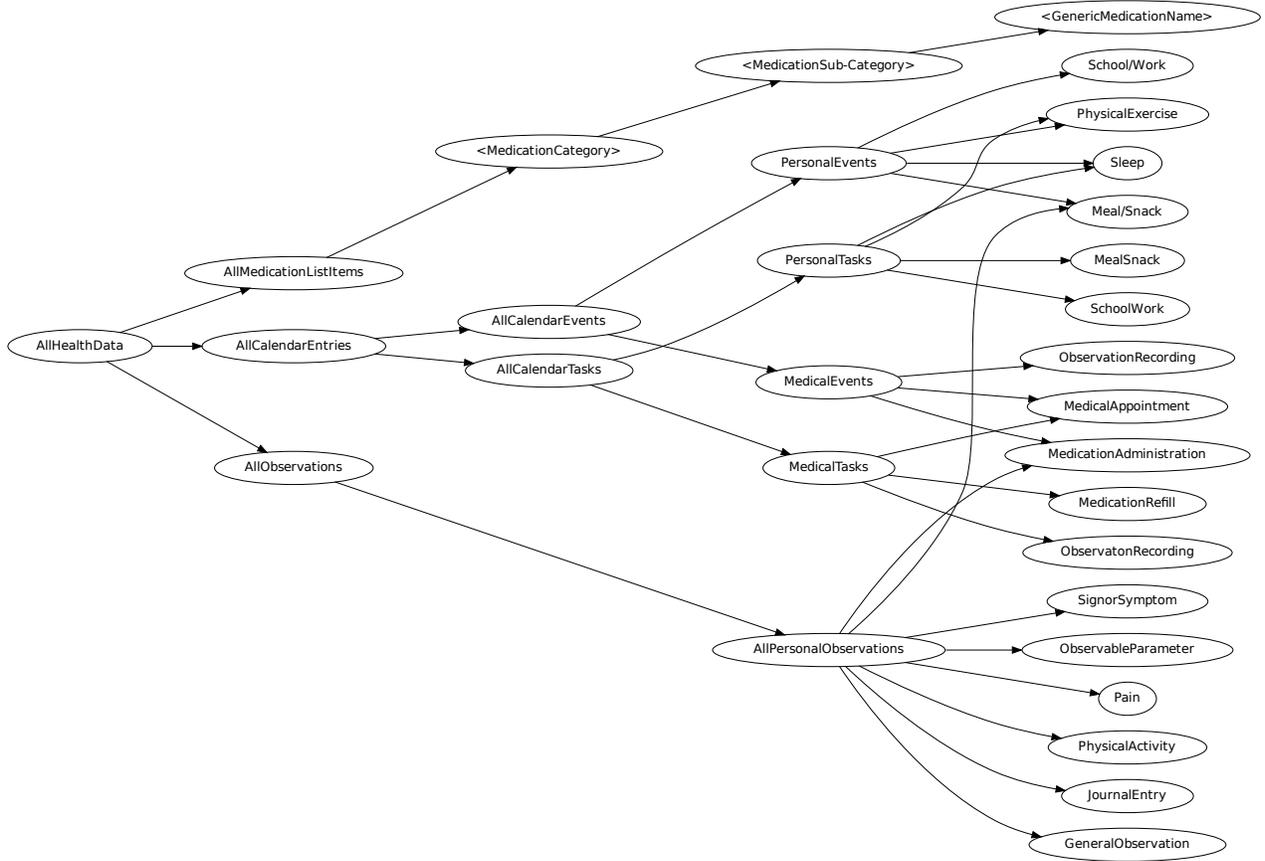Figure 1.   The Role Hierarchy

Figure 2.   The Action Hierarchy

Figure 3. The Resource Hierarchy

$PA \cup \{(u, (\text{RecordViewing}, \text{MedicalEvents}))\})$ if $u \in U$ is such that $(u, r) \in UA$ for some $r \in R$ such that Physician $\sqsupseteq_R r$ and $(u, r) \notin UA$ for all $r \in R$ such that FamilyMember $\sqsupseteq_R r$.

Consider now the situation in which Bob wants to delegate Alice the right to grant viewing privileges only to physicians he trusts. By defining the policy in such a way that $(\text{Alice}, p) \in PA$ with $p = (\text{Grant}(\text{Record-Modification})$ to $\{+\text{Physician}, +\text{Trusted}\}$, MedicalEvents), Bob could conclude that only physicians he trusts could access his MedicalEvents. But this is not necessarily the case. In fact, if Charlie was trusted by Bob, then Alice might have granted Charlie the right to modify Bob's MedicalEvents, and Charlie can keep this privilege even if he is no longer trusted by Bob. As this example illustrates, it can be difficult to predict the effect of delegating privileges and this may lead the user to draw wrong conclusions and ultimately stop trusting and using the PHR management system.

In conclusion, a policy $\pi = (u_o, U, R, P, UA, PA)$ includes its own administrative policy, i.e. the policy prescribing who can change the policy and what are the policies

$\pi' = (u_o, U, R, P, UA, PA')$ that can be obtained by abiding by the policy:

- if $(u, p) \in PA$ with $p = (\text{Grant}(act)$ to $Pre, res)$, then $\pi' = (u_o, U, R, P, UA, PA \cup \{(u', p)\})$ for any $p \in P$ such that $(act, res) \sqsupseteq_P p$ provided that $u' \in U$, and for all $r \in R$
  - if $+r \in Pre$, then $(u', r') \in UA$ for some $r' \in R$ such that $r \sqsupseteq_R r'$,
  - if $-r \in Pre$, then $(u', r') \notin UA$ for all $r' \in R$ we have $r \sqsupseteq_R r'$.

If π' can be reached from π by applying one of these two steps, then we write $\pi \to \pi'$. We say that π_n *is reachable from* π_0 iff there exists a sequence of policies $\pi_0, \pi_1, \ldots, \pi_{n-1}, \pi_n$ such that $\pi_i \to \pi_{i+1}$ for $i = 0, \ldots, n-1$ for some $n \geq 0$.

A *query* is triple of the form (u, act, res) where $u \in U$, $act \in Act$ and $res \in Res$. Given a query (u, act, res) and a policy π, we are interested in determining whether from π it is possible to reach a policy π' such that u can execute act on res in π'. This is an instance of the *PHR reachability problem*. In the next section we will present a procedure that

tackles this problem.

III. Automatic Analysis of Semantic-Aware Access Control Policies

We assume the reader to be familiar with the basic notions of first-order logic with equality (see, e.g., [11]). Following [3], [4], we make use of a particular class of formulae, called Bernays-Schönfinkel-Ramsey (BSR) [5], of the form

$$\exists x_1, ..., x_n. \forall y_1, ..., y_m. \varphi(x_1, ..., x_n, y_1, ..., y_m)$$

where φ is a quantifier-free formula containing only individual constants and predicate symbols (i.e. not containing function symbols) and such that if x is a variable occurring in φ, then $x \in \{x_1, ..., x_n, y_1, ..., y_m\}$. It is well-known that the validity and satisfiability problems of BSR formulae are decidable [5].

A. Symbolic representation of PHR policies

We assume to have countably many constant symbols to represent elements of the sets U, R, Act, Res, and P (i.e. users, roles, actions, resources, and permissions, respectively). To simplify notation, we do not make distinction between the elements of the sets with the constants representing them. We also assume to have unary predicate symbols U, R, Act, Res, and P such that $X(e)$ iff $e \in X$ for every element e and for each $X \in \{U, R, Act, Res, P\}$. As for constants, here and below, we confuse sets with the predicates representing them. Consider the role hierarchy of Figure 1. It is clear that the set of roles is known to contain exactly the elements: FamilyMember, HealthCareProvider, AllOtherUsers, etc. Thus, we assume that the predicate R also satisfies the following formulae:

$$\forall r. R(r) \Leftrightarrow (r = \text{FamilyMember} \lor r = \text{HealthCareProvider} \lor$$
$$r = \text{AllOtherUsers} \lor \cdots)$$
$$\text{FamilyMember} \neq \text{HealthCareProvider},$$
$$\text{FamilyMember} \neq \text{AllOtherUsers},$$
$$\text{HealthCareProvider} \neq \text{AllOtherUsers}, \dots,$$

which constrain the interpretation of R to contain "at most" (first formula above) and "at least" (remaining distinctions) the elements shown in the hierarchy of Figure 1. Similar sets of formulae constrain the interpretations of Act and Res to contain exactly the elements in Figure 2 and Figure 3, respectively. The fact that we leave the interpretation of U unconstrained means that it contains a fixed but unknown number of users.

Furthermore, we assume to have a binary predicate $\sqsupseteq_R$ such that

$$\forall r_1, r_2. r_1 \sqsupseteq_R r_2 \Rightarrow (R(r_1) \land R(r_2))$$
$$\forall r. r \sqsupseteq_R r$$
$$\forall r_1, r_2. (r_1 \sqsupseteq_R r_2 \land r_2 \sqsupseteq_R r_1) \Rightarrow r_1 = r_2$$
$$\forall r_1, r_2, r_3. (r_1 \sqsupseteq_R r_2 \land r_2 \sqsupseteq_R r_3) \Rightarrow r_1 \sqsupseteq_R r_3,$$

a binary predicate UA such that $\forall u, r. UA(u, r) \Rightarrow (U(u) \land R(r))$, two binary predicates $\sqsupseteq_{Act}$ and $\sqsupseteq_{Res}$ satisfying formulae similar to those for $\sqsupseteq_R$, a ternary predicate ARP such that

$$\forall a, s, p. ARP(a, s, p) \Rightarrow (Act(a) \land Res(s) \land P(p))$$
$$\forall a, s, p, p'. (ARP(a, s, p) \land ARP(a, s, p')) \Rightarrow p = p',$$

a binary predicate $\sqsupseteq_P$ such that

$$\forall a, s, p, a', s', p'. (ARP(a, s, p) \land ARP(a', s', p')) \Rightarrow$$
$$(p \sqsupseteq_P p' \Leftrightarrow (a \sqsupseteq_{Act} a' \land s \sqsupseteq_{Res} s')),$$

and finally a binary predicate PA such that
$$\forall x, p. (PA(x, p) \Rightarrow ((U(x) \lor R(x)) \land P(p))).$$

The formulae above, called *axioms*, constrain the predicates of arity larger than 1 to be well-typed (e.g., the interpretation of $\sqsupseteq_R$ must be a sub-set of the Cartesian product of the interpretation of R with itself), $\sqsupseteq_R, \sqsupseteq_{Act}$, and $\sqsupseteq_{Res}$ are partial orderings (i.e. reflexive, antisymmetric, and transitive relations), ARP is a partial function mapping pairs composed of actions and resources to permissions, and $\sqsupseteq_P$ is defined on permissions by using the partial orders on actions and resources. Indeed, we still need to constrain $\sqsupseteq_R, \sqsupseteq_{Act}$, and $\sqsupseteq_{Res}$ so as to satisfy the hierarchies in Figures 1, 2, and 3, respectively. This is particularly easy as it is sufficient to add an axiom $s \sqsupseteq_X d$ for every edge $s \to d$ in a hierarchy $X \in \{R, Act, Res\}$. For example, $\sqsupseteq_{Act}$ (Figure 2) should be such that AllOperations $\sqsupseteq_{Act}$ RecordModification, AllOperations $\sqsupseteq_{Act}$ RecordViewing, AllOperations $\sqsupseteq_{Act}$ RecordAdministration, etc. Let Ax_{PHR} be the set of axioms listed above. It is not difficult to see that all the formulae in Ax_{PHR} can be rewritten into logically equivalent BSR formulae.

It is now easy to formalize that a user u can execute an action a on a resource s as follows:

$$ARP(a, s, p) \Rightarrow$$
$$\begin{pmatrix} \exists p'. (PA(u, p') \land p' \sqsupseteq_P p) \ \lor \\ \exists r, r', p'. (R(r) \land R(r') \land UA(u, r) \land \\ r \sqsupseteq_R r' \land PA(r', p') \land p' \sqsupseteq_P p) \end{pmatrix},$$

which will be abbreviated with $can(u, a, s)$. (Notice that this can be also rewritten into a logically equivalent BSR formula.)

Now, recall the situation where the access control policy of Bob's PHR is such that $PA(\text{Alice}, p)$ and $ARP(\text{RecordViewing}, \text{MedicalEvents}, p)$, for some permission $p \in P$. To check that Alice can view Bob's MedicalAppointments (recall the resource hierarchy of Figure 3), it is sufficient to check that $can(\text{Alice}, \text{RecordViewing}, \text{MedicalAppointments})$ is a logical consequence of $Ax_{PHR} \cup \{\exists p. (P(p) \land PA(\text{Alice}, p) \land ARP(\text{RecordViewing}, \text{MedicalEvents}, p))\}$. Reasoning by refutation, this is equivalent to checking the unsatisfiability

of

$$Ax_{PHR} \cup$$
$$\left\{ \begin{array}{l} \exists p. \left(\begin{array}{l} P(p) \land PA(\text{Alice}, p) \land \\ ARP(\text{RecordViewing}, \text{MedicalEvents}, p) \end{array} \right), \\ \neg can(\text{Alice}, \text{RecordViewing}, \text{MedicalAppointments}) \end{array} \right\}.$$

By simple logical manipulations, the conjunction of the formulae in this set is easily rewritten into a logically equivalent BSR formula, whose satisfiability is decidable and can be checked by available automated theorem provers such as Z3 or SPASS. Although already useful for exploring granted or denied privileges of a given PHR policy π, this kind of analysis is not sufficient to explore the whole consequences of π because certain of its tuples contain administrative actions that are able to modify the state of π itself (recall the definition of PHR reachability problem at the end of the previous section).

B. Symbolic representation of Administrative Actions

We now proceed to consider administrative actions (e.g., Grant(RecordViewing) in Figure 2), which come with a set of preconditions to restrict their applicability. To illustrate, consider again the situation in which Alice can grant the privilege to viewing Bob's MedicalEvents only to users who are physicians but are not Bob's relatives, encoded by the pair $(\text{Alice}, p) \in PA$ with $p = $ (Grant(RecordViewing) to $\{+\text{Physician}, -\text{FamilyMember}\}$, MedicalEvents). First, the preconditions require that the user u who will be granted the privilege to view Bob's MedicalEvents must be a Physician but not a relative of Bob. This can be encoded by the formula

$$\exists r_1. (UA(u, r_1) \land \text{Physician} \sqsupseteq_R r_1) \land$$
$$\forall r_2. (\text{FamilyMember} \sqsupseteq_R r_2 \Rightarrow \neg UA(u, r_2)).$$

Second, the effect of u acquiring the privilege of viewing Bob's MedicalEvents can be modeled as the following update of the PA relation:

$$\forall x, y. (PA'(x, y) \Leftrightarrow (PA(x, y) \lor (x = u \land y = p)))$$

where p is such that $ARP(\text{RecordViewing}, \text{MedicalEvents}, p)$ holds, PA denotes the current permission assignment relation, and PA' denotes the permission assignment relation resulting from the execution of the administrative action. Thus, putting things together, the administrative action related to the pair $(\text{Alice}, p) \in PA$ with $p = $ (Grant(RecordViewing) to $\{+\text{Physician}, -\text{Family-Member}\}$, MedicalEvents) can be represented by the following formula:

$$\exists p. (P(p) \land ARP(\text{RecordViewing}, \text{MedicalEvents}, p) \land$$
$$PA(\text{Alice}, p) \Rightarrow$$
$$\exists u.$$
$$\left(\begin{array}{l} \exists r_1. (UA(u, r_1) \land \text{Physician} \sqsupseteq_R r_1) \land \\ \forall r_2. (\text{FamilyMember} \sqsupseteq_R r_2 \Rightarrow \neg UA(u, r_2)) \land \\ \forall x, y. (PA'(x, y) \Leftrightarrow \\ \quad (PA(x, y) \lor (x = u \land y = p))) \end{array} \right).$$

It is not difficult to see that also this formula can be rewritten into a logically equivalent BSR formula. The other administrative operations can be specified in a similar way.

Let $\tau(PA, PA')$ be the disjunction of all the formulae representing the administrative actions. The formula $\tau(PA, PA')$ represents all possible one-step evolutions of the PA relation allowed by the administrative permissions included in PA. It is easy to see that also $\tau(PA, PA')$ can be rewritten into a logically equivalent BSR formula.

C. A symbolic analysis technique for the PHR reachability problem

Recall the definition of the PHR reachability problem: given a PHR policy π and a query (u, a, s), for a user u, an action a, and a resource s, we want to establish whether there exists a policy π' that can be reached from π such that u can execute a on s in π'. Our analysis technique solves an instance of the PHR reachability problem by carrying out the following steps:

1) Compute the set Ax_{PHR} of axioms (BSR formulae) describing users, actions, resources, and the relation UA in π.
2) Write the BSR formula $\varphi(PA)$ corresponding to the initial value of the relation PA in π.
3) Compute the disjunction $\tau(PA, PA')$ of BSR formulae encoding the administrative actions in π.
4) Check whether there exists a value $n \geq 0$ such that the formula

$$\bigwedge Ax_{PHR} \land \varphi(PA_0) \land$$
$$\bigwedge_{k=1}^{n} \tau(PA_{k-1}, PA_k) \land can_{PA_n}(u, a, s) \quad (1)$$

is satisfiable, where $\bigwedge Ax_{PHR}$ stands for the conjunction of the axioms in Ax_{PHR}, $PA_0, PA_1, ..., PA_{n-1}, PA_n$ are uniquely renamed copies of PA, $\varphi(PA_0)$ has been obtained from $\varphi(PA)$ by replacing PA with PA_0, $\tau(PA_{k-1}, PA_k)$ has been obtained from $\tau(PA, PA')$ by replacing PA with PA_{k-1} and PA' with PA_k (for $k = 1, ..., n$), and $can_{PA_n}(u, a, s)$ has been obtained from $can(u, a, s)$ by replacing PA with PA_n.

Because of the observations above, all the conjuncts of (1) can be rewritten into logically equivalent BSR formulae.

Since a (finite) conjunction of BSR formulae can be rewritten into a logically equivalent BSR formula, we conclude that for any given value of n, the satisfiability of (1) is decidable. The (big) conjunction over k with φ in (1) can be seen as a characterization of the set of states *forward reachable* from the initial PHR policy π. Symmetrically, the (big) conjunction over k with $can_{PA_n}(u, a, s)$ in (1) characterizes the set of states *backward reachable* from the query (u, a, s).

The critical step in the procedure above is the last one: how can we determine a bound for n? An idea could be to generate instances of (1) for increasing values of n and check the satisfiability of the resulting BSR formulae. However, this would not guarantee us the termination of the procedure: if the query is satisfiable in some reachable (from π) PHR policy π', then for some value of n, the corresponding instance of (1) will be satisfiable. However, when the query cannot be answered positively in any of the reachable (from π) PHR policies, all the instances of (1) will always be found unsatisfiable and we will be forced to generate an infinite sequence of formulae. Thus, the decidability of the satisfiability of (1) is only a necessary condition for ensuring the decidability of the PHR reachability problem. A sufficient condition to stop enumerating instances of (1) after a certain value of $n = \overline{n}$ is to check when any of the formulae characterizing the set of reachable states for $n > \overline{n}$ implies the one describing the set of reachable states for $n = \overline{n}$. In this case, we say that a *fixed-point* of the set of reachable states has been detected. A general approach to do this is based on computing the set of backward reachable states, as it is well-known that this can be done more easily than establishing the set of forward reachable states. An additional problem is that the class of formulae used to represent sets of backward reachable states—in our case BSR—is not expressive enough to encode all the fix-points and the procedure may not terminate. In [3], [4], we study the reachability problem of administrative RBAC policies in a similar setting that, suitably extended, can be used also to solve the PHR problem considered in this paper.

Fortunately, analysis techniques based on sophisticated fix-point calculations are only required when we need to certify that a certain query cannot be answered positively. The simpler approach of enumerating instances of (1) for increasing values of n up to an *a priori* fixed bound (and checking for satisfiability) can already be an important tool for debugging PHR policies. In fact, it can expose finite sequences of administrative actions that enable an untrusted user to execute some critical action on a shared resource. Such sequences can be scrutinized and used to eliminate the violations of authorization constraints. The corrected versions of the policy can be scrutinized several times by successive refinements, until no more bugs can be detected within the fixed bound on the length of the traces. Only at that point, the more sophisticated techniques based on fix-point calculations can be used for certification.

IV. DISCUSSION

E-health applications need to allow users the capability to specify which other users may access which part of their resources and what kind of operations may perform (see, e.g., [12], [10]). Thus, sophisticated access control policies defined over structurally complex domains are required. For this reason, it is not surprising that Semantic Web languages have been used to specify authorization policies over heterogeneous domain data and to promote sharing among participants who might adopt different information models. In [13], it is discussed how two (general purpose) Semantic Web languages have been successfully used to express authorization policies together with a comparison to an (*ad hoc*) authorization language for distributed applications. In [14], [15], it is described how to specify context information associated to access control management policies using the Web Ontology Language (OWL). In this way, available Description Logic (DL) reasoners can be used to perform various tasks related to policy management such as consistency checking and conflict resolution.

In [16], the Relation Based Access Control (RelBAC) model and logic is introduced whose goal is to deal with the problem of access control in Web 2.0 applications. One of the key feature of RelBAC is the use of Entity-Relationship (ER) diagrams for the design of access control policies. It is not difficult to see that (a substantial part of) ER diagrams can be easily translated to the class of BSR formulae that we use in this paper for modeling the access control policies we are interested in. This naturally leads—in future work— to investigate two questions: (i) is it possible to use our analysis techniques for the policies developed in the RelBAC framework? (ii) Is it possible to use ER diagrams as a front-end for the natural specification of the policies in our framework?

In [17], the relationships between OWL and the RBAC model are studied. In particular, two ways to support the RBAC model (and attribute-based extensions) in OWL are shown and the relative strengths and weaknesses are discussed. The main difference between the two OWL embeddings of RBAC is the representation of roles and of the role hierarchy. One of the two ways is particularly relevant to our work, namely the way in which roles are represented by OWL classes to which individual users can belong. In this way, role hierarchies are described through OWL class hierarchies where the inheritance relation is the inverse of the "more-senior-than" relation over roles (in which a user gets more privileges when going up to the hierarchy of roles). This is similar to our treatment of the role, action, and resource hierarchies (cf. Figures 1, 2, and 3, respectively) in which users get more privileges as they move down the hierarchies, since classes get more attributes while going down.

Interestingly, [17] also discusses the security analysis of administrative policies by saying that, while certain problems can be solved by using available DL reasoners, others cannot and for the latter it proposes to use the analysis techniques developed in the context of the RT authorization languages (see, e.g., [18]). However, as already pointed out in [17], the results of the analysis are sometimes not precise and more general techniques based on model checking are needed. In particular, the analysis techniques we have proposed in this paper achieve a good trade-off between precision and automation while leveraging state-of-the-art automated reasoning methods.

To summarize, we have presented an automatic analysis technique for the PHR reachability problem based on the idea of reducing the problem to a sequence of satisfiability problems for BSR formulae to be solved by state-of-the-art automated reasoners. As discussed above, the class of BSR formulae captures most of the key features of the attempts to use Semantic Web techniques in the context of authorization policies for the web. An interesting line of future research would be to investigate which classes of first-order logic corresponding to DLs can be used while maintaining the decidability of our analysis technique. To assess the effectiveness of our approach, we plan to implement the analysis procedure described in this paper in the ASASP tool [19] and to carry out a systematic experimental analysis.

ACKNOWLEDGMENT

REFERENCES

[1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996.

[2] J. Puustjrvi and L. Puustjrvi, "Personal Health Ontology: towards the interoperation of e-health tools," *Int. J. Electronic Healthcare*, vol. 6, no. 1, pp. 62–75, 2011.

[3] A. Armando and S. Ranise, "Automated Symbolic Analysis of ARBAC Policies," in *6th Int. Workshop on Security and Trust Management (STM)*, 2010.

[4] F. Alberti, A. Armando, and S. Ranise, "Efficient Symbolic Automated Analysis of Administrative Role Based Access Control Policies," in *6th ACM Symp. on Information, Computer, and Communications Security (ASIACCS)*, 2011.

[5] F. P. Ramsey, "On a Problem of Formal Logic," *Proceedings of the London Mathematical Society*, vol. s2-30, no. 1, pp. 264–286, 1930.

[6] N. Li and J. C. Mitchell, "RT: A Role-based Trust-management Framework," in *The 3rd DARPA Information Survivability Conference and Exposition (DISCEX III). IEEE, pp. 201–212*, 2003.

[7] C. Weidenbach, "Combining superposition, sorts and splitting," in *Handbook of Automated Reasoning*, 2001.

[8] M. Suda, C. Weidenbach, and P. Wischnewski, "On the Saturation of YAGO," in *Proc. of IJCAR*, ser. LNCS, vol. 6173, 2010, pp. 441–456.

[9] Z3, "`http://research.microsoft.com/en-us/um/redmond/projects/z3`," 2011.

[10] Sujansky & Associates LLC, "Project HealthDesign - Common Platform Components: Functional Requirements," Tech. Rep., 2007.

[11] H. B. Enderton, *A Mathematical Introduction to Logic*. New York-London: Academic Press, 1972.

[12] W. V. Sujansky, S. A. Faus, E. Stone, and P. F. Brennan, "A method to implement fine-grained access control for personal health records through standard relational database queries," *J. of Biomedical Informatics*, vol. 43, no. 5, Supplement 1, pp. S46 – S50, 2010.

[13] G. Tonti, J. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok, "Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder," in *The Semantic Web - ISWC 2003*, ser. LNCS, 2003, vol. 2870, pp. 419–437.

[14] E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P. Samarati, "Modality-Conflicts in Semantics-Aware Access Control," in *6th Int. Conf. on Web Engineering (ICWE)*. ACM Pres, 2006.

[15] ——, "Extending Context Descriptions in Semantics-Aware Access Control," in *International Conference on Information Systems Security (ICISS)*, ser. LNCS, vol. 4332, 2006, pp. 162–176.

[16] F. Giunchiglia, R. Zhang, and B. Crispo, "RelBAC: Relation Based Access Control," in *Proceedings of the Fourth International Conference on Semantics, Knowledge and Grid*, 2008, pp. 3–11.

[17] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham, "ROWLBAC: representing role based access control in OWL," in *SACMAT'08: Proceedings of the 13th ACM symposium on Access control models and technologies*. (New York, NY, USA): ACM, 2008, pp. 78–32.

[18] N. Li and M. V. Tripunitara, "Security analysis in role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 4, pp. 391–420, 2006.

[19] F. Alberti, A. Armando, and S. Ranise, "ASASP: Automated Symbolic Analysis of Security Policies," in *Proc. of the 23rd Conference on Automated Deduction (CADE)*, 2011.