Security Analysis of Relationship-Based Access Control Policies

Amirreza Masoumzadeh
University at Albany – SUNY
Albany, NY
amasoumzadeh@albany.edu

ABSTRACT

Relationship-based access control (ReBAC) policies can express intricate protection requirements in terms of relationships among users and resources (which can be modeled as a graph). Such policies are useful in domains beyond online social networks. However, given the updating graph of user and resources in a system and expressive conditions in access control policy rules, it can be very challenging for security administrators to envision what can (or cannot) happen as the protection system evolves.

In this paper, we introduce the *security analysis* problem for this class of policies, where we seek to answer security queries about future states of the system graph and authorizations that are decided accordingly. Towards achieving this goal, we propose a state-transition model of a ReBAC protection system, called RePM. We discuss about formulation of security analysis queries in RePM and present our initial results for a limited version of this model.

CCS CONCEPTS

• **Security and privacy** → **Access control**; **Authorization**;

KEYWORDS

relationship-based access control; security analysis; safety

ACM Reference Format:
Amirreza Masoumzadeh. 2018. Security Analysis of Relationship-Based Access Control Policies. In *CODASPY '18: Eighth ACM Conference on Data and Application Security and Privacy, March 19–21, 2018, Tempe, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3176258.3176323

CODASPY '18, March 19–21, 2018, Tempe, AZ, USA

ACM ISBN 978-1-4503-5632-9/18/03…$15.00
https://doi.org/10.1145/3176258.3176323

1 INTRODUCTION

Several relationship-based access control (ReBAC[1]) models have been developed in recent years [5, 12, 13, 9, 7]. While they were initially inspired by online social networks (OSNs) and Web 2.0 applications, it has been also shown that their expressive power is useful in other domains such as healthcare [25]. In ReBAC models, authorizations are specified based on relationship patterns between users and resources that are involved in an access control scenario. Such patterns are typically captured as paths or topological constraints in a graph structure of entities and relationships among them. For example, access control policy options in an OSN, which allows a user to share with "friends" or "friends of friends," are expressed as paths of different lengths in the OSN graph based on friendship relationships between users.

ReBAC policies are very expressive (e.g., see various intricate examples expressible in Fong's ReBAC [13]). However, such expressiveness also complicates understanding a ReBAC policy. For example, consider this question: given a set of actions that users can perform and policies that they can configure in an OSN, can a certain leak of information happen? (e.g., can a stalker obtain access to a victim's information after performing a series of actions?) There are several factors that makes answering such a question challenging in ReBAC environments, such as interaction of rules, updates to graph and policies, and distributed administration. A ReBAC policy is composed of individual rules, each authorizing some action(s) according to relationship-based conditions in a system. In any sizable ReBAC policy, it will be hard to comprehend and ensure that protection requirements are met as interactions between rules may lead to unknown situations and some security gaps might be even left undetected. Even if protection requirements are confirmed at one moment, future updates to the policy may result in unexpected situations. The problem is even more complicated since as the underlying graph of relationships is updated it will also affect the authorization decisions in a ReBAC environment. We need to also consider that often times in modern systems, updates happen in a distributed fashion (not by a central security administrator). For example, each user is in charge of specifying her own access control options (privacy settings) in an OSN, and the policy of an OSN as a whole is a combination of such policies (along with other rules set by the OSN provider).

In this paper, we investigate the *security analysis* problem [19] in the context of ReBAC models. The goal of security analysis is to verify properties of a protection system as it

[1] We use the term *ReBAC* to refer to the class of access control models based on relationships. In order to avoid confusion, we refer to the seminal work proposed in [12, 13] as Fong's ReBAC.

may evolve: i.e., starting from a given secure state, and based on a given set of state transitions (or constraints that control them), whether an (undesirable) state is reachable (i.e., existential query) or a (desirable) property is always held (i.e., universal query). The existential flavor of this problem, known as *safety* problem, has been studied whether leakage of a specific right can happen in a discretionary access control model [14, 28, 26]. It was then extended to include universal questions as well, and thus, called *security analysis* [19] in the context of trust management systems. The security analysis problem has also been investigated in the context of administrative role-based access control [18, 29]. It is worthwhile to note that security analysis is essentially a more complex problem than verifying properties of a protection system at a given state, e.g., whether certain undesirable authorization exists. The latter problem can be referred to as policy analysis [23, 21] or proof of compliance [19].

While there have been formal approaches to the specification/enforcement (e.g, [13, 9]), and more recently administration [30, 8] of ReBAC policies, to the best of our knowledge, this is the first work to study the security analysis problem in this context. In particular we make the following contributions in this paper:

- We propose a formal relationship-based protection model, called RePM, that captures the essence of ReBAC policies as a state-transition system. The model captures an underlying graph of entities, ReBAC policy that controls authorizations on the graph, and administrative ReBAC policy that governs changes to the ReBAC policy itself. In addition to providing a generalizable model, RePM addresses a major gap in previous formal ReBAC models [13, 9] by supporting authorization policies for "viewing" edges in the system graph.
- We present, for the first time, the security analysis problem for ReBAC policies, formulated in the context of the proposed protection model. We propose two versions of the problem which have different security analysis query targets (graph and authorization).
- We study the safety problem in a limited version of our model and prove its decidability.

The rest of the paper is organized as follows. In Section 2, we briefly review the related work on ReBAC policies and security analysis. We begin Section 3 with presentation of a running example and the design considerations for our protection model. We then propose a formal model of ReBAC protection system, called RePM. We formulate the security analysis in the context of RePM in Section 4, and finally illustrate decidability result for a simplified model in Section 5.

2 RELATED WORK

Our work is closely related to two bodies of work, namely, relationship-based access control and safety/security analysis. Among them, we review the most relevant work to this paper in the followings.

2.1 Relationship-Based Access Control

The early work on ReBAC models focused on specifying simple path conditions based on relationships among users in OSNs and other web applications [17] which may be also composed with trust evaluation along such paths [4]. Other models involved including more rich relational semantics available in Semantic Web [3, 22], conflict resolution on shared resources by multiple parties [16, 15], and including objects in addition to users in the network [6, 22]. Fong et al. [13, 12, 2] proposed a series of access control policy models (named "ReBAC") which are capable of expressing sophisticated topological relationships among users based on modal/hybrid logics. While the models consider only positive authorization rules, exceptions can be handled using negative conditions. Inspired by the notion of principals in Unix and ReBAC, the RPPM model [9, 10] proposed a two-step authorization process for an access request. Access subjects are first matched to the known principals in the policy based on path-based conditions. Then, further path-based conditional rules determine authorizations (after resolving conflicts if needed). Finally, a recent work shows how conversion of Fong's ReBAC to Datalog programs [24] can help extending it to include negative authorizations and temporal policies, and additionally support simple policy analysis (in the same protection state). Unlike the above mentioned models, the focus of our work is to provide a clear formal model of a ReBAC protection system as a state transition machine which can be used for security analysis. In other words, we are not seeking to provide a better expressiveness power for ReBAC policies. In fact, our goal is to keep the model compatible (to the extent possible) with existing condition specification approaches. In describing our model, we use path expressions similar to RPPM [9].

Almost all of the work mentioned so far, consider only using topology-based conditions in policies without addressing how such topology itself might evolve. The recent ReBAC administrative models [30, 8] based on RPPM [9] have focused on this missing aspect. While we also adopt ideas from these recent models in formalizing our protection system, a clarification is needed with regards to the use of terminology in our model versus the previous work. While both RPPM2 [30] and ARPPM [8] consider changes to graph of entities by users as "administration," we consider those as user-level changes. We believe that only updates to ReBAC policy rules should be considered as administrative actions, and provide a full justification as we present our model in Section 3.

2.2 Safety and Security Analysis

The safety problem was first introduced in the context of the access matrix protection system [14]. It was shown that for a given access matrix (protection state) and set of commands that make changes to the matrix (transition to new protection states), it is generally undecidable whether a certain right will be entered in the matrix in future (i.e., leaking). Here, leakage of the right represents an undesirable situation that should be avoided in a safe and secure system. Also, it was shown

while there are decidability results for restricted versions of the problem, they are still hard problems (to solve in polynomial time). Subsequently, other protection models were developed with improved decidability results [28] and it was shown that considering entity types in an access matrix model will lead to decidable cases for safety [26]. The problem was revisited later in the context of trust management (a distributed approach to access control based on roles) by Li and Winsborough [19]. They showed that universal queries can be verified beyond existential queries such as "simple safety" (e.g., is there a reachable state in which a user has certain access?). For example one can query about "simple availability" (e.g., can a certain access happen in all reachable states?), or "containment" (e.g., will every principal having certain access will have certain property in all reachable states?) Therefore, the problem was renamed to "security analysis" to include all such queries. They also showed that, in the context of a simplified version of RT language [20], simple safety, availability, and containment (only when language contains simple delegations) are answerable in polynomial time. In the context of RT, protection state is described by a set of policy statements (membership to a role, delegation of a role, etc.), and state transitions involve addition and removal of such statements while the analysis prevents addition or removal of statements about certain roles. Security analysis has been also studied in the context of role-based access control (RBAC) [18, 29]. Here, the protection state includes the RBAC policy configuration (i.e., current assignments) while transitions can be performed based on *can_assign* and *can_remove* policies in administrative model, ARBAC [27].

3 RePM: RELATIONSHIP-BASED PROTECTION MODEL

We propose a formal model of a ReBAC protection system, which we call *RePM*. Following the construction of safety/security analysis problems in the literature [14, 19], our model will be a state-transition system. Before discussing the model, we present a running example and our design considerations in order to facilitate discussions in the rest of the paper. Then, we define the notion of protection state in RePM which captures the changing components of the system. And finally, we show how the protection system model is formed together with the non-changing parts.

3.1 Running Example

ReBAC policies are useful in many application domains. In order to facilitate a focused discussion as we present the model and discuss its analysis, we choose a simple running example in a healthcare information environment. Suppose Alice is being treated for cancer. She is receiving treatment by her oncologist, Jane, at M-Hospital. Jane happens to be Alice's primary physician as well. Alice has specified Bob and Carol as her contacts and has specifically designated Bob as her emergency contact. Figure 1 portrays the entities and relationships as a graph in such a scenario. We consider

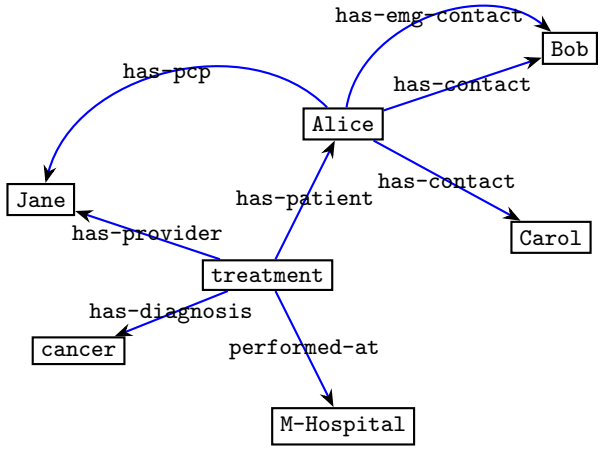

Figure 1: Running Example – A Healthcare Scenario

the following access control policy statements applied in this scenario:

(a) a doctor and a patient should know everything about a treatment in which they're involved
(b) a patient's provider in a treatment case should know about the patient's primary care doctor
(c) a patient can add contacts
(d) a patient can designate contacts as emergency contact
(e) a facility at which a patient is being treated can inform a patient's emergency contact about the patient's diagnosis when needed

The above rules may not capture all security requirements. However, we note that the number of rules even for such a simple scenario increases fast, which demonstrates the need for policy and security analysis.

In the rest of the paper we refer to the above scenario as *the running example* and omit an explicit reference to this section.

3.2 Design Considerations

Our main goal in the design of the RePM model is to formalize the necessary elements and structure in a ReBAC environment that are important in reasoning about its security properties. Therefore, RePM is not supposed to be yet another model to capture more expressive authorization or administrative policies. It is rather designed to be a comprehensive model that can incorporate or be easily extended based on various proposals in the literature. As a fundamental design principle, we intend to keep the proposed model as simple as possible, mainly to avoid overcomplicated presentation and to facilitate compatibility with the existing ReBAC models in the literature. For example, as we define policies, we avoid using keywords such as "*" that may match with any values. While such notions are useful for convenience purpose, they are not necessary for analysis purpose as long as policies can be equivalently expressed without them.

We also intend to maintain some separation between functional and security aspects of a system. Since ReBAC policies can benefit from a rich amount of information about entities and their relationships in the system, typically, there will be a significant overlap between information model from a functional perspective (for providing application functions) and information model from a security perspective (for deciding about authorizations). However, we intentionally avoid inclusion of any non-security requirement in our protection model in order to keep the model simple. For example, some data integrity requirements such as functional validity of a specific type of relationship between two types of entities may be accommodated by mechanisms other than access control (e.g., in a database environment) and so they will be excluded from our model. However, we acknowledge that any restrictions enforced on evolution of a system may affect its security analysis and we plan to address such considerations as future work.

3.3 System Graph

A protection state in RePM needs to capture the current state of user authorizations. Authorizations in ReBAC are derived based on the authorization rules which test some specified conditions about entity relationships in the system. Entities include users as well other logical resources in the application that need to be represented in the protection state. Therefore, a protection state needs to capture entities, relationships among them, and authorization policy.

A natural way to capture entities and their relationships is a graph structure. For this purpose, we follow a similar formulation as in RPPM [9, 10].

Definition 3.1 (System Graph). Given a set of relationship labels L_R, a system graph is a pair of vertices and edges $\langle V, E\rangle$, where V is the set of entities, and $E \subseteq L_R \times V \times V$ is the set of labeled relationships between entities.

We previously illustrated the system graph for our running example in Figure 1. We should note that in the rest of the paper we use dot notation to refer to elements within a concept. For instance, $G.V$ refers to the set of vertices in system graph G.

The system graph, as defined above, is limited to directed and typed edges. We note that it can be extended by notions such as symmetric relationships and typed vertices [10] among other possibilities (e.g., see the variations explored in [1]). However, for simplicity we limit our model as above. Also, it is noteworthy that unlike RPPM [9] we do not consider a *system model* alongside an instantiated system graph. From a logics point of view, a model and its instantiated version can be seen as TBox and ABox, respectively. The integrity constraints that need to be enforced according to a TBox (e.g., a *has-pcp* relationship is only meaningful between two users and not between a user and a resource) will be trivial to enforce and will be outside of the scope of our model.

One of the features of applications that benefit from ReBAC policies is the overlap between application and protection state. For instance, while the system graph for the running example (Figure 1) is part of the protection state it also captures data that will be utilized in the application (e.g., in order to look up contacts of a patient). In RePM, we assume the overlap is complete in the way that the application resources that need to be protected have been already represented in the system graph. This assumption has been validated in application areas beyond online social networks such as in an open-source medical record system [25].

Moreover, we consider edges (or relationships) as the main protected resources. This allows RePM to be able to specify authorizations for "viewing" edges; for instance, that Jane is Alice's PCP in the running example (i.e., seeing relationship `has-pcp` between Alice and Jane). Existing formal ReBAC models such as Fong's ReBAC [12] and RPPM [9] consider only vertices as access objects while specifying policies according to relationship-based conditions. Such models are not be able to model authorization policies for edges as mentioned in the above example. It should be noted that while recent ReBAC administrative models such as RPPM2 [30] and ARPPM [8] include update to edges as part of administration, the administrative actions naturally include "insertion"/ "removal" of edges and exclude "viewing" edges.

3.4 Authorization Policy

Since the system graph captures the protected resources, we consider any access attempt to the graph (including modification to it) as an access request. Modification to system graph has been termed as "administrative action" by recent works such as RPPM2 [30] and ARPPM [8]. However, we see it as modification of the application resources which also happens to be part of the information used to decide about authorization (known as ADI or access decision information in standard access control frameworks [11]). We consider six *primitive operations* that capture the basic modes of access to the graph. They include viewing, inserting, or deleting a vertex or an edge. In order to simplify presentation, following the approach in ARPPM [8], we focus on the operations on edges only (although ARPPM only considers them in the context of "administrative actions"). We assume that a vertex is inserted upon creation of its first edge and removed when all of its edges have been removed. Furthermore, we assume that viewing a vertex without its edges would not reveal any sensitive information.

Definition 3.2 (Access Request). An access request is a triple $\langle s, \langle rl, v_s, v_d\rangle, op\rangle$, where subject $s \in V$ is requesting to perform operation *op* (`view`/`insert`/`remove`) on an edge expressed as a tuple of type $rl \in L_R$ between $v_s, v_d \in V$.

A sample access request in the running example is as follows.

$$\langle \texttt{M-Hospital}, \langle \texttt{has-emg-contact}, \texttt{Alice}, \texttt{Bob}\rangle, \texttt{view}\rangle$$

We clarify that an access request is not directly formulated by a subject in the system, but rather by the reference monitor that authorizes the request. The mention of Bob in the above request does not mean that M-Hospital already knows about Alice's emergency contact; the reference monitor has

formulated the request to authorize revealing the relationship to M-Hospital.

An access request needs to be evaluated against an authorization policy comprising of authorization rules. Authorization rules in ReBAC test relationship-based conditions based on an incoming access request.

Definition 3.3 (Authorization Rule). An authorization rule is a tuple $\langle \phi, rl, op, d \rangle$, where ϕ is a matching condition based on an access request, rl is the relationship label, op is a primitive operation on system graph (`view`/`insert`/`remove`), and d is an authorization decision (`grant`/`deny`).

Relationship label rl and primitive operation op in an authorization rule are matched against their counterparts in an incoming access request in order to determine the applicability of the rule. Various proposals exist in the literature for expressing the matching condition ϕ stated above, including modal/hybrid logic approach of Fong's ReBAC [12, 2] and path expressions in ARPPM [8]. While we envision RePM to be neutral in the choice of matching conditions, the choice of language is important in determining the complexity of its security analysis. In this paper, we will adopt a slightly simplified version of path expressions of ARPPM [8] in order to formulate our examples and discuss results further below.

Definition 3.4 (Path Condition). A path is constructed using the following syntax:

$$\pi ::= \top \mid \diamond \mid rl \mid rl; \pi \mid \overline{\pi}$$

where $rl \in L_R$ is a relationship label. A path condition is a a logical conjunction of one or more expressions of format $u.\pi.v$ where given a system graph G, an access request q, $u, v \in G.V$ we have:

- $G, q \models \top$;
- $G, q \models u. \diamond .v$ iff $u = v$;
- $G, q \models u.rl.v$ iff $\langle rl, u, v \rangle \in G.E$;
- $G, q \models u.rl; \pi.v$ iff there exists $w \in G.V$ such that $G, q \models u.rl.w$ and $G, q \models w.\pi.v$;
- $G, q \models u.\overline{\pi}.v$ iff $G, q \models v.\pi.u$.

Informally, the expressions above, respectively model the always-true condition, test for same vertices, a relationship, concatenation of a relationship and a path, and the inverse of a path.

In the above definition, u and v can either represent specific vertices (e.g., `Alice`) or vertex elements of access request tuple q (i.e., s, v_s, and v_d). For instance, the following captures the authorization rule corresponding to the policy statement (c) in the running example. Here, $s. \diamond .v_s$ means that subject s is the same as the source vertex for the edge:

$$\langle s. \diamond .v_s, \texttt{has-contact}, \texttt{insert}, \texttt{grant} \rangle$$

Given a set of authorization rules, an authorization policy is defined as follows.

Definition 3.5 (Authorization Policy). An authorization policy is a pair $\langle R, \sigma \rangle$, where R is a set of authorization rules and σ is a conflict resolution strategy.

When multiple rules within set R with conflicting decisions become applicable to an access request, we use strategy σ to resolve the conflict. For example, *deny-takes-precedence* is a well-known access control conflict resolution approach that adheres to the principle of fail-safe defaults in security.

3.5 Protection State

We formally define a protection state based on system graph and authorization policy as follows.

Definition 3.6 (Protection State). A protection state is a pair $\langle G, P \rangle$, where G is the system graph capturing entities and relationships among them, and P is an authorization policy which controls operations on G.

Note that the protection state as defined above encompasses both authorization decision information (system graph) and authorization policy. The authorization policy needs to be enforced when changes to the system graph are requested. Furthermore, the authorization policy may be updated over time by performing administrative operations.

3.6 Administrative Policy

The RePM protection system comprises of the protection state and an administrative policy that allows modifications to the authorization policy (a component in the protection state). In terms of modifications to authorization policy, we define two *primitive administrative operations*: namely, insertion and removal of an authorization rule. We assume the administration is distributed and performed by entities represented in the system graph.

Definition 3.7 (Admin Request). An admin (access) request is a triple $\langle s, pr, op \rangle$, where subject $s \in V$ is requesting to perform primitive administrative operation op (`insert`/`remove`) on authorization rule pr, which is specified as a tuple as in Definition 3.3.

We define the administrative authorization rules and policy analogous to their non-administrative counterparts.

Definition 3.8 (Administrative Rule). An administrative rule is a tuple $\langle \phi, \phi_{pr}, rl_{pr}, op_{pr}, d_{pr}, op, d \rangle$, where ϕ is a matching condition based on an admin request, op is a primitive administrative operation (`insert`/`remove`), and d is an administrative decision (`grant`/`deny`). The elements with pr subscript indicate matching component in the authorization rule pr provided in the administrative request.

Based on an incoming administrative request $\langle s, pr, op \rangle$, four sets of checks need to be completed in order to determine applicability of an administrative rule to the request. First, the primitive operation of an administrative rule should be the same as in the request. Second, condition ϕ is tested which may include tests on relationships about subject s in the admin request. Third, components rl_{pr}, op_{pr}, and d_{pr} are checked for equivalence against their counterparts within the requested pr rule in the request. Finally, ϕ_{pr} is tested to establish if it is less than or equally specific to its counterpart in the request's pr. Here, the idea is that an

administrative request is allowed as long as it is more specific than what an administrative rule authorizes. In other words, ϕ_{pr} specifies the most generic authorization rule (based on relationship-based conditions) that can be inserted/removed from the policy by the specified administrators. This captures the *strictness order* proposed in RPPM2 [30].

The policy statement (e) in the running example is an administrative policy which updates the authorization policy by allowing emergency contact of a patient to see the diagnosis. For instance, when needed, M-Hospital needs to add the following authorization rule to the policy so that Alice's emergency contact, Bob, can view her diagnosis:

$$\langle \texttt{Alice.has-emg-contact}.s \ \wedge \ v_s.\texttt{has-patient.Alice}, \texttt{has-diagnosis}, \texttt{view}, \texttt{grant} \rangle$$

Note that the above rule is an authorization rule according to Definition 3.3 which ensures that subject s is Alice's emergency contact, and the starting vertex v_s of the protected resource (`has-diagnosis` edge) is Alice's treatment. Now, an administration rule that authorizes addition of the above rule to the authorization policy is as follows.

$$\langle s.\overline{\texttt{performed-at}}; \texttt{has-patient}.\$x, \$x.\texttt{has-emg-contact}.s \ \wedge \ v_s.\texttt{has-patient}.\$x, \texttt{has-diagnosis}, \texttt{view}, \texttt{grant}, \texttt{insert}, \texttt{grant} \rangle$$

Note that we consider free variables such as $\$x$ in the syntax for administrative rules in order to support flexible administrative policies. It should be clear that when $\$x$ in the above administrative rule is bound to `Alice`, M-Hospital will be able to insert the previous authorization rule.

Finally, given a set of administrative rules, an administrative policy is defined as follows.

Definition 3.9 (Administrative Policy). An administrative authorization policy is a pair $\langle AR, \sigma \rangle$, where AR is a set of administrative rules and σ is a conflict resolution strategy.

3.7 Protection System

We now formally define a RePM protection system as follows.

Definition 3.10 (Protection System). A RePM protection system is defined by a pair $\langle PS, AP \rangle$, where PS represents the possible protection states and AP is an administrative policy. AP authorizes administrative operations applicable to the authorization policy at a state ps, i.e., $ps.P$.

In each state of the system ps, exercise of an operation on system graph $ps.G$ is authorized by $ps.P$, and if allowed, will result in transition to a new protection state (with an updated $ps.G$). At the same time, each exercise of an administrative action authorized by AP will also result in transition to a new protection state (with an updated $ps.P$).

4 SECURITY ANALYSIS IN RePM

In this section, we formulate security analysis problem in the context of RePM. In order to facilitate discussions about security analysis, we first provide an illustration of a the RePM protection model. Figure 2 depicts a fairly complex state-transition system of RePM using a small synthetic example. In the figure, each node (ellipse shape) represents a protection state, which includes both system graph G and authorization policy P. Each solid arrow indicates possibility of transition from one state to another. The dashed circles show logical groupings of states based on same authorization policy p_i. The transitions in the protection system are controlled by the enforced policies. While transitions within a logical group are restricted by authorization policy p_i, transitions across logical groups are restricted by administrative policy AP. Note that administrative policy AP is considered constant (or only updated by trusted users). However, authorization policy can be updated according to AP, and for that reason, it is considered constant only within each logical circle in the figure.

4.1 Factors Affecting Analysis

In this section, we informally discuss about the security analysis problem in the context of RePM and contributing factors to its expressiveness and complexity.

Query target A security analysis query should be about properties of protection states. In RePM, given the structure of a protection state $\gamma = \langle G, P \rangle$ the query can be about system graph G, or valid authorizations based on authorization policy P. We call them *graph queries* and *authorization queries*, respectively. In the context of the running example, "whether a person can be a patient's PCP and her emergency contact at the same time" is a graph query. An example of authorization query is that whether a regular contact will be able to learn about a patient's illness.

Query formulation One of the strengths of ReBAC is its ability to specify powerful conditions based on system graph. This feature can be utilized in security analysis queries to precisely characterize an entity (group of entities) of interest based on its (their) relationships at the desired level of specificity. For example, one can ask if at any point any doctor will have more access to treatment information than the involved patients. While this enables much more flexible queries compared to the existing security analysis approaches (for example in case of ARBAC [18]), it adds to the complexity of answering such queries.

Existential vs. universal query We expect both existential and universal queries to be useful. Such quantifiers are applicable both to system states and queries.

Policy syntax The richness of both authorization and administrative policies in RePM will affect the complexity of security analysis. That includes support for expressions based on system graph in the conditions, and whether rules will be positive-only or a mixture of positive and negative rules accompanied by a conflict resolution strategy.

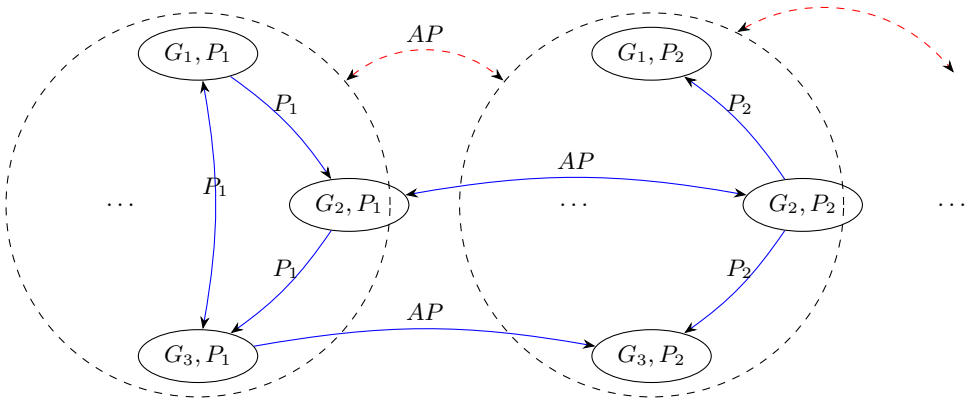

Figure 2: RePM State Transition System

Authorization policy update RePM supports modification of the authorization policy using the administrative policy, which is assumed to be constant itself (or modified only by trusted users). This is natural in ReBAC environments. For example, OSN users may specify their own authorization policies for their data (privacy settings). At the same time, they only have a fixed amount of administrative power on specifying such policies. We also envision ReBAC systems which may not need such administrative capability. We therefore, consider and analyze a class of systems with no administrative policy as well.

4.2 Security Analysis Problem

In order to formulate the security analysis problem in the context of RePM, we adopt the following two definitions of access control scheme and security analysis, originally presented in the context of RBAC [19].

Definition 4.1 (Access Control Scheme [19]). An access control scheme is a state-transition $\langle \Gamma, Q, \vdash, \Psi \rangle$, where Γ is set of states, Q is set of queries, $\vdash$ is an entailment relation $\vdash: \Gamma \times Q \to \{\texttt{true}, \texttt{false}\}$ that determines whether a query is true or not in a given state, and Ψ is the set of the possible schemes for state transitions.

In the above definition, $\psi \in \Psi$ is a state-change rule, i.e., a policy. We write $\gamma \overset{*}{\mapsto}_{\psi} \gamma_1$ if starting from state γ we reach state γ_1 after zero or more transitions allowed by ψ.

Definition 4.2 (Security Analysis [19]). Given an access control scheme $\langle \Gamma, Q, \vdash, \Psi \rangle$ a security analysis instance takes the form $\langle \gamma, q, \psi, \Pi \rangle$, where $\gamma \in \Gamma$ is a starting state, $q \in Q$ is a query, $\psi \in \Psi$ is a state-change rule, and $\Pi \in \{\exists, \forall\}$ is a quantifier. An instance $\langle \gamma, q, \psi, \exists \rangle$ asks whether there exists γ_1 where $\gamma \overset{*}{\mapsto}_{\psi} \gamma_1$ and $\gamma_1 \vdash q$. An instance $\langle \gamma, q, \psi, \forall \rangle$ asks whether for every γ_1 such that $\gamma \overset{*}{\mapsto}_{\psi} \gamma_1$ and $\gamma_1 \vdash q$.

As discussed in Section 4.1, there are many contributing factors to security analysis problem for ReBAC policies which can lead different formulations of the problem. In the followings, we introduce RePM-specific security analysis problems by specifying elements of an access control scheme (Definition 4.1). In particular, we define two classes of security analysis problems in the context of RePM based on different query targets. The first class uses graph queries as targets.

Definition 4.3 (RePM Security Analysis with Graph Query Target). A RePM security analysis based on a graph query is defined according to the following access control scheme:

states A state γ is a pair $\langle G, P \rangle$ of a system graph and an authorization policy.

state-transition rules A state transition ψ includes both authorization policy P and administrative policy AP.

queries A graph query is a tuple $q_g = \langle \phi, rl, \Pi' \rangle$, where ϕ is a matching condition based on Definition 3.4, rl is a relationship label, and $\Pi' \in \{\exists, \forall\}$ is a quantifier. Variables v_s and v_d can be used in ϕ to express beginning and ending vertices of an edge with label rl.

entailment An *existential graph query* (where $\Pi' = \exists$) entails true if and only if there exists an edge $\langle rl, v_s, v_d \rangle$ in the system graph that satisfies matching condition ϕ. A *universal graph query* (where $\Pi' = \forall$) entails true if and only if all edges $\langle rl, v_s, v_d \rangle$ in the system graph satisfy matching condition ϕ.

Note the distinction between quantifiers Π and Π' in Definitions 4.2 and 4.3, respectively. While the former quantifies future states, the latter quantifies edges in the system graph at a given state. The combination of the two quantifiers leads to four different types of analysis.

As an example of an existential security analysis ($\Pi = \exists$), based on an existential graph query ($\Pi' = \exists$), we may ask: "Would it be possible that some patient's PCP be her emergency contact as well?" Such an analysis can be formulated as follows.

$$\langle \gamma, \langle v_s.\texttt{has-emg-contact}.v_d, \texttt{has-pcp}, \exists \rangle, \psi, \exists \rangle$$

The other variations of the above analysis that can be formulated are as follows. For brevity, we avoid repeating the analysis expressions.

- ($\Pi = \forall; \Pi' = \exists$) Is it always the case that some patient's PCP is her emergency contact as well?
- ($\Pi = \exists; \Pi' = \forall$) Would it be possible that every patient's PCP be their emergency contacts as well?
- ($\Pi = \forall; \Pi' = \forall$) Is it always the case that every patient's PCP are their emergency contacts as well?

We present our second RePM-specific security analysis problem based on authorization queries.

Definition 4.4 (RePM Security Analysis with Authorization Query Target). A RePM security analysis based on an authorization query is defined according to the following access control scheme:

states A state γ is a pair $\langle G, P\rangle$ of a system graph and an authorization policy.

state-transition rules A state transition ψ includes both authorization policy P and administrative policy AP.

queries An authorization query is a tuple $q_a = \langle \phi, rl, op, d, \Pi'\rangle$, where ϕ, rl, op, and d are same as in an authorization rule (Definition 3.3), and $\Pi' \in \{\exists, \forall\}$ is a quantifier.

entailment An *existential authorization query* q_a entails true if and only if $\gamma_1.P$ (authorization policy in state γ_1) and corresponding authorization elements in q_a result in consistent authorization decision for an applicable access request. A *universal authorization query* q_a entails true if and only if $\gamma_1.P$ and q_a result in consistent authorization decisions for every applicable access request.
Consistent decisions for a policy and q_a means that either both grant or both deny the applicable access(es).

As an example of a universal analysis, based on a universal authorization query, we may ask: "Will patients always be able to see their diagnosis?" The analysis will be formulated as follows.

$\langle \gamma, \langle v_s.\texttt{has-patient}.s, \texttt{has-diagnosis}, \texttt{view}, \texttt{grant}, \forall\rangle, \psi, \forall\rangle$

5 SAFETY ANALYSIS IN RePM$_G$

As a sample security analysis problem for ReBAC policies, we study a safety problem in a limited version of RePM, which we call RePM$_G$. In RePM$_G$, there is no administrative policy, i.e., $AP = \emptyset$. This means that authorization policy is never updated or only updated by trusted administrators. Actions taken by trusted administrators are excluded from security analysis as they are assumed to avoid any breach of security.

The safety problem in RePM$_G$ that we consider in this section is a graph query that asks whether at any future state a certain edge will exist in the system graph. We define the problem as follows, as a subclass of the problem in Definition 4.3.

Definition 5.1 (Edge Safety Problem in RePM$_G$). Edge safety problem in RePM$_G$ is a simplified version of RePM security analysis problem with graph query target according to the following access control scheme:

states A state γ is modeled as system graph G.

state-transition rules A state transition ψ is modeled using authorization policy P.

queries The edge safety query q_g can be formulated as the following graph query: $\langle \top, rl, \exists\rangle$

THEOREM 5.2. *The edge safety problem in RePM$_G$ is decidable.*

We can prove the above theorem by reducing the safety problem in RePM$_G$ to the safety in the mono-operational case of the HRU model [14]. A sketch of the proof is as follows.

PROOF SKETCH. Given an instance of safety problem in RePM$_G$, we build an HRU protection system by converting system graph G to an access control matrix and each authorization rule with `insert`/`remove` primitive operations to a command in the context of HRU model. We note that an authorization rule with a `view` primitive operation does not result in changing the protection state. It is sufficient to represent system graph G as its adjacency matrix in order to convert it to its access matrix equivalent. The result will be a square matrix M, where $l \in M[x, y]$ if and only if $\langle l, x, y\rangle \in G.E$. Note that in terms of access matrix this means subject x will have right l on object y. Given the structure of an access request $\langle s, \langle rl, v_s, v_d\rangle, op\rangle$, an authorization rule $pr = \langle \phi, rl, op, \texttt{grant}\rangle \in P$ is converted to the following command:

```
command auth_cmd_pr(s, rl, v_s, v_d, w_1,...,w_t)
{
  if Φ then
    OP(rl, v_s, v_d)
}
```

In the above, primitive operation OP will correspond to op in the authorization rule. The condition Φ will translate path condition ϕ in the authorization rule into a series of tests for presence of rights in the access matrix as follows. Let $\phi_i = u_i.\pi_i.v_i$ be the ith operand of the conjunction in ϕ. According to Definition 3.4, ϕ_i has format $u.l_1; l_2; \ldots; l_k.v$, which can be unfolded into a conjunction

$$u.l_1.w_1 \wedge w_1.l_2.w_2 \ldots \wedge w_{k-1}.l_k.v$$

We create a conjunctive expression consisting of tests on access matrix corresponding to each operand $x.l_i.y$ as follows:

- $x = y$ if $l_i = \diamond$;
- $l_i \in M[x, y]$ if $l_i \in L_R$;
- $l_i \in M[y, x]$ if $l_i = \overline{l'_i}$ and $l'_i \in L_R$.

This will capture ϕ_i in terms of access matrix. Finally, we consider Φ to be the conjunctive statement containing all such corresponding ϕ_is. Arguments $w_1 \ldots w_t$ in the command correspond to placeholder vertices that are used in the construction of individual tests described above. The count of them will depend on the length of the paths tested in the path condition. Now, test Φ on access matrix is equivalent to matching condition ϕ in RePM$_G$.

In this setting, leaking a right in the context of the constructed HRU model is equivalent to insertion of the equivalent edge in system graph G. The constructed HRU model

is a mono-operational HRU protection system, therefore the same process that can decide about it [14] can decide about safety in RePM_G. □

Note that, in the above proof sketch, we have slightly deviated from the syntax of commands in the HRU model. The HRU commands do not consider tests for equivalency of arguments. However, including such tests will not compromise the original proof of decidability presented in [14]. We recall that the main argument of the original proof is that by considering the creation of only one new subject instead of many possible creations, the leak can be still achieved (if possible at all). If we substitute the newly created subject s_1 in place of any other subject that might have been potentially created, say s_2: as long as tests for an equivalency test $x = s_2$ is replaced with $x = s_1$, if $x = s_2$ was able to a leak a right, $x = s_1$ will lead to a leak too.

6 CONCLUSION

It is critical to formally analyze safety and security of an access control policy system in order to ensure that it provides the expected security. Reasoning about security of complex ReBAC policy systems as they evolve can mitigate many challenges with which such systems are already facing. For example, it can potentially resolve uncertainties that both users and provider of an online social network may have about future security and privacy behaviors in the system. In this work, we presented a first formulation of security analysis problem in the context of ReBAC systems by formalizing a ReBAC protection system, RePM. We also showed the decidability of edge safety problem in a limited version of RePM.

There are many exciting challenges to be addressed as future work including improving generalizability of the RePM model, investigating various subclasses of the security analysis problem in this context, and studying application of such analysis on real-world systems.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments and helpful suggestions that guided us in improving the final manuscript.

REFERENCES

[1] T. Ahmed, R. Sandhu, and J. Park. Classifying and Comparing Attribute-Based and Relationship-Based Access Control. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, CODASPY '17, pages 59–70, New York, NY, USA. ACM, 2017.

[2] G. Bruns, P. Fong, I. Siahaan, and M. Huth. Relationship-based Access Control: Its Expression and Enforcement Through Hybrid Logic. In *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*, CODASPY '12, pages 117–124, New York, NY, USA. ACM, 2012.

[3] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. A semantic web based framework for social network access control. In *Proc. 14th ACM Symposium on Access Control Models and Technologies*, pages 177–186. ACM, 2009.

[4] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in Web-based social networks. *ACM Trans. Inf. Syst. Secur.*, 13(1):1–38, Nov. 2009. ISSN: 1094-9224.

[5] B. Carminati, E. Ferrari, and A. Perego. Rule-Based Access Control for Social Networks. In R. Meersman, Z. Tari, and P. Herrero, editors, *Proc. OTM 2006 Workshops (On the Move to Meaningful Internet Systems)*, volume 4278 of *LNCS*, pages 1734–1744. Springer, Oct. 2006.

[6] Y. Cheng, J. Park, and R. Sandhu. Relationship-Based Access Control for Online Social Networks: Beyond User-to-User Relationships. In *Proc. 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 646–655, Sept. 2012.

[7] M. Cramer, J. Pang, and Y. Zhang. A Logical Approach to Restricting Access in Online Social Networks. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, SACMAT '15, pages 75–86, New York, NY, USA. ACM, 2015.

[8] J. Crampton and J. Sellwood. ARPPM: Administration in the RPPM Model. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pages 219–230, New York, NY, USA. ACM, 2016.

[9] J. Crampton and J. Sellwood. Path Conditions and Principal Matching: A New Approach to Access Control. In *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies*, SACMAT '14, pages 187–198, New York, NY, USA. ACM, 2014.

[10] J. Crampton and J. Sellwood. Relationships, Paths and Principal Matching: A New Approach to Access Control. arXiv:1505.07945 [cs], May 29, 2015. arXiv: 1505.07945;.

[11] eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS, 2013.

[12] P. W. Fong. Relationship-based access control: protection model and policy language. In *Proc. CODASPY '11*, pages 191–202, San Antonio, TX, USA. ACM, 2011.

[13] P. W. Fong and I. Siahaan. Relationship-based access control policies and their policy languages. In *Proc. 16th ACM Symposium on Access Control Models and Technologies*, SACMAT '11, pages 51–60, Innsbruck, Austria. ACM, 2011.

[14] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, Aug. 1976. ISSN: 0001-0782.

[15] H. Hu, G. J. Ahn, and K. Kulkarni. Discovery and Resolution of Anomalies in Web Access Control Policies. *IEEE Transactions on Dependable and Secure Computing*, 10(6):341–354, Nov. 2013. ISSN: 1545-5971.

[16] H. Hu and G.-j. Ahn. Multiparty Authorization Framework for Data Sharing in Online Social Networks. In Y. Li, editor, *Proceedings of the 25th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy*, volume 6818 of *Lecture Notes in Computer Science*, pages 29–43. Springer Berlin / Heidelberg, 2011.

[17] S. R. Kruk. FOAF-Realm: control your friends access to the resource. In Workshop on Friend of a Friend, Social Networking and the Semantic Web, 2004.

[18] N. Li and M. V. Tripunitara. Security Analysis in Role-based Access Control. In *Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies*, SACMAT '04, pages 126–135, New York, NY, USA. ACM, 2004.

[19] N. Li and W. H. Winsborough. Beyond proof-of-compliance: safety and availability analysis in trust management. In *Proceedings of the 2003 Symposium on Security and Privacy*, pages 123–139, May 2003.

[20] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, Jan. 1, 2003. ISSN: 0926-227X.

[21] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo. EXAM: a comprehensive environment for the analysis of access control policies. *International Journal of Information Security*, 9(4):253–273, Aug. 2010. ISSN: 1615-5262.

[22] A. Masoumzadeh and J. Joshi. OSNAC: An Ontology-based Access Control Model for Social Networking Systems. In *Proc. 2nd IEEE Int'l Conference on Information Privacy, Security, Risk and Trust (PASSAT 2010)*, pages 751–759, Minneapolis, MN, USA, Aug. 2010.

[23] T. Nelson, D. J. Dougherty, C. Barratt, and K. Fisler. The Margrave Tool for Firewall Analysis. In *Proceedings of the 24th USENIX Large Installation System Administration Conference (LISA 2010)*, 2010.

[24] E. Pasarella and J. Lobo. A Datalog Framework for Modeling Relationship-based Access Control Policies. In *Proceedings of the 22Nd ACM on Symposium on Access Control Models and Technologies*, pages 91–102, New York, NY, USA. ACM, 2017.

[25] S. Z. R. Rizvi, P. W. Fong, J. Crampton, and J. Sellwood. Relationship-Based Access Control for an Open-Source Medical Records System. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, SACMAT '15, pages 113–124, New York, NY, USA. ACM, 2015.

[26] R. S. Sandhu. The typed access matrix model. In *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 122–136, May 1992.

[27] R. S. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 Model for Role-Based Administration of Roles. *ACM Transactions on Information and Systems Security*, 2(1):105–135, 1999.

[28] R. S. Sandhu. The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes. *J. ACM*, 35(2):404–432, Apr. 1988. ISSN: 0004-5411.

[29] A. Sasturkar, P. Yang, S. D. Stoller, and C. R. Ramakrishnan. Policy analysis for administrative role based access control. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, 13 pp.–138, 2006.

[30] S. D. Stoller. An Administrative Model for Relationship-Based Access Control. In *SpringerLink*. IFIP Annual Conference on Data and Applications Security and Privacy, pages 53–68. Springer, Cham, July 13, 2015.