A Theory of Role Composition

Jeffrey Fischer Rupak Majumdar
University of California, Los Angeles
{fischer,rupak}@cs.ucla.edu

***Abstract*—We study the access control integration problem for web services. Organizations frequently use many services, each with its own access control policies, that must interoperate while maintaining secure access to information. The integration problem is to take the set of such services and to find a globally consistent access control policy that ensures no authorization failures or information disclosures for the system composed from the services. Currently, this is performed manually, and incurs high administrative overhead and high risks for errors. We give a sound and complete algorithm for access control integration by reducing the problem to Boolean constraint solving. We have implemented ROLEMATCHER, a tool to infer global role-based access control schemas for a set of services, and show on examples that it can quickly infer global roles for composed systems, or determine the absence of a globally consistent role schema.**

I. INTRODUCTION

A key function in any enterprise security infrastructure is *access control*, which specifies the valid ways in which users can access resources and perform operations. *Role-based access control* (RBAC) is a frequently-used access control mechanism for enterprise systems [10]. In RBAC, *roles* represent functions within a given organization; authorizations for resource access are granted to roles rather than to individual users. Users "play" the roles, acquiring the privileges associated with the roles, and administrators grant or revoke role memberships.

While RBAC provides an elegant and scalable access control mechanism for a single system, organizations frequently deploy many interacting applications, each with its own RBAC policy. These individual policies must be integrated into a global policy for the organization. For example, web portal applications combine content from several back-end services in a single web interface. Portal frameworks generally provide their own RBAC policy enforcement, which must be configured to be consistent with the individual applications displayed on the portal. Currently, this global access control policy is designed manually, and this is both labor intensive and error-prone. The goal of our work is to automate this process.

In particular, we develop an algorithm that computes, if possible, a global RBAC policy from the RBAC policies of the different applications, such that the following incompatibilities cannot occur:

- In the processing of a request, a service in one component may call other services in different components. Even though the user had access rights to the original service, they might not have access to the called services, leading to *indirect authorization errors*. Such errors are difficult to prevent through testing, since access rights are usually assigned uniquely to each user.
- When confidential data is passed between services, the receiver could potentially disclose that data to users and services which do not have the necessary access rights in the source system. This may violate the intent of the source system's security policy.

We call this the *global schema inference* problem. A *global schema* is a system-wide notion of roles, and a mapping from local roles to (possibly multiple) global roles, such that with these global assignments of roles to users, there are no indirect authorization errors (the global roles are *sufficient*) or information leaks (the global roles are *non-disclosing*). Our main result is a constraint-based algorithm to infer, given the local role assignments for a set of components, whether there exists a sufficient and non-disclosing global schema that is consistent with the local roles in each component.

We make the following contributions in this paper.

First, we study the integration problem abstractly by defining an interface formalism for components that specify both the roles required to access each service and the other services that may be invoked by a service, as well as a simple language of service invocations. In an associated technical report [11], we define the operational semantics of this language, formalizing indirect authorization errors. Given two components, each with their independent notion of roles, we define *global role schemas* that group together roles from each service such that there are no indirect authorization errors. Global role schemas are inferred using a constraint-based algorithm that reduces the inference problem to Boolean constraint solving. In addition to requiring sufficiency, the constraints also ensure other desired properties: *separation*, *role ascription*, and *minimality*. Separation ensures that two different roles from the same component are never merged in a global role (i.e., local role semantics are maintained). Role ascription allows the administrator to specify that certain roles across components should be merged. Minimality ensures the principle of least privilege: each global role must contain only required local roles and no extra roles. As might be expected, the global schema inference problem is NP-complete.

Next, we augment service components with *information flow* that additionally models information flow between services. We extend the global schema inference algorithm to produce minimal global role schemas that are sufficient as well as non-disclosing.

A possible alternative to global compatibility is to have one global role scheme managed and enforced centrally in the organization. In practice, this can be difficult, or impossible, since each application can be locally administered by different

Application	Service	Roles
Clinical Management	Scheduling	C:Receptionist, C:Nurse, C:Doctor
		C:Doctor
	Vitals	C:Nurse, C:Doctor
	CareOrders	C:Doctor
Laboratory	TestOrders	L:Clinician, L:Billing
	TestResults	L:Clinician
Patient Records	PatientHistory	P:Clinician

Fig. 1. Services used in our examples.

groups, or even be legacy applications where the notion of access control is built into the system. Also, external enforcement approaches may not necessarily capture all the places where an application discloses sensitive data (e.g., data included in other pages or disclosed through integrations).

Alternately, one can implement *ad hoc* mechanisms to ensure interoperability, for example by guessing sets of local roles for users based on past access history. Unfortunately, these mechanisms usually do not provide correctness guarantees, and may lead either to breaches of confidentiality, when an access not permitted within an individual system becomes permitted through indirect service calls, or to violation of least privilege, when an user is assigned more authority than is strictly required.

We have implemented ROLEMATCHER, a tool for global role schema inference. Our implementation shows that our algorithm infers global role schemes within a few seconds for small examples, and is expected to scale well for large collections of systems. In addition, we performed a case study inspired by an industrial problem where an IT management company wanted to integrate two applications, one (App A) with a fixed RBAC scheme, and another (App B) in which the customer builds site-specific roles. For this problem, ROLEMATCHER can produce, for each customer site, a globally consistent view for the entire system that unifies customer-specified roles in App B, the fixed roles in App A, and the role schemes of any site-specific applications that the customer decides to install.

II. EXAMPLE

We demonstrate our techniques on a hypothetical healthcare information system at a medical clinic. The clinic has three applications:

- Clinical management: this application manages the scheduling of patients and captures the actions performed by doctors and nurses.
- Laboratory information system: this application tracks the tests to be performed and their results.
- Patient records: this application maintains historical data about each patient's health.

Each system provides one or more web services, which expose a set of callable methods and encapsulates access to the underlying data and application functionality. These services are protected by Role-Based Access Control (RBAC). A set of roles is associated with each service and with each user. To access a service, the set of active roles for the current user must include at least one of the roles required by the service.

Unfortunately, each application has its own RBAC schema, so integrating these systems requires reconciling multiple schemas. Figure 1 lists the services provided by each application and the roles required for accessing these services. We prefix application-local role names with a unique letter for each application (C, L, or P). We call the roles defined within a given application *local roles*.

Suppose the clinic adds a new web portal which provides convenient web access across the other three applications (Figure 2(a)). The application does not store any confidential data locally. Instead, when the user requests a page, the portal makes service calls to the other applications using the requesting user's login.

Consider the PatientMedData service of the portal which permits doctors and nurses to see the relevant medical information for a patient. Figure 2(a) shows the services called by PatientMedData: the Vitals and CareOrders services of Clinical Management, the TestResults service of Laboratory, and the PatientHistory service of Patient Records. The CareOrders service, in turn, calls the TestOrders service to retrieve details of tests that have been ordered. This service is accessible to users with either the W:Doctor or W:Nurse roles.

Global Roles. Since each application has its own notion of roles, it is difficult to determine whether a given user will have all the access permissions needed to retrieve the data for each page. This is the *global role compatibility* problem: does there exist some global set of roles that represent sets of local roles from the different applications that can be used to maintain consistent user to role mappings? The global role compatibility problem takes as input the applications and their role requirements, and produces, if possible, a set of global roles and a mapping from each global role to a set of local roles such that the following constraints are satisfied:

1) **[Separation]** No two local roles from the same application should be mapped to the same global role. This ensures that the semantics of authorization within an application is not changed by the global map. Otherwise, admistrators will be unable to independently assign the two roles to users. Also, this restriction prevents degenerate solutions, such as assigning all local roles from a given application to the same group. In section VIII, we discuss how we might selectively relax this restriction.
2) **[Sufficiency]** For each service of the web portal application, services that it (transitively) calls must include all of the global roles required by the web portal service. Thus, if a user has any one of the required roles

Fig. 2. (a) Services of the web portal application. Boxes denote applications with their own notions of roles. Inner boxes show services available for each application. Directed arrows show other services that are called by a service. (b) Data synchronization between applications. Wide arrows denote direction of information flow.

for the PatientMedData service, they will a required role for each called service. This ensures no indirect authorization failures.

3. **[Ascriptions]** Administrators may optionally specify sets of local roles that must map to the same global role. This permits the representation of semantic constraints specific to an application domain.

A global role mapping is the *minimal* mapping which satisfies the constraints: if there are no calls combining two roles on separate systems, then the roles should be mapped to two separate global roles. For example, the C:Receptionist and L:Billing roles are unrelated to each other or any other roles through calls. Thus, they should be mapped to unique global roles. Minimality ensures a form of least privilege — a global role mapping should not give users any more access rights than strictly necessary to accomplish their objectives.

Note that a given local role can map to more than one global role. This may occur when one system has a less precise security model than the others. In our example, the Laboratory system has only a clinician role for both doctors and nurses, while the Clinical Management application distinguishes between doctors and nurses. By requirement 1 above, we need to maintain two separate global roles for doctors and nurses. If the Vitals service of Clinical Management is used in conjunction with the TestResults service of Laboratory, we need to map Clinician to both of these global roles.

Global Schema Inference. We solve the global schema inference problem by formulating it as a Boolean constraint satisfaction problem. Notice that requirement 1 constrains W:Doctor and W:Nurse to be in different global roles (Constraint (1)), and similarly C:Doctor and C:Nurse must be in different global roles (Constraint (2)). Since PatientMedData invokes CareOrders, the set of global roles for PatientMedData must be included in the set of global roles for CareOrders (Constraint (3)). To reflect the idea that doctor and nurse roles should be common across applications, we add an ascription to force the roles W:Doctor and C:Doctor to map to the same global role (Constraint (4)) and, similarly, an ascription to force the roles W:Nurse and C:Nurse to the same global role (Constraint (5)).

Surprisingly, it is not possible to find a mapping which satisfies all these constraints. From Constraints (1) and (2), the Doctor and Nurse roles within each service must be mapped to different global roles. Constraint (3) forces C:Doctor to map to both W:Doctor and W:Nurse. However, this violates the ascription constraints, as W:Nurse should be mapped to role C:Nurse. Thus, our initial portal design does not admit global role schemas. With a tool to infer global role schemas, we can catch such problems at design time rather when users are assigned to roles and attempt to use the system (as common with ad hoc approaches to access control integration).

Now consider an alternative design of the web portal where we split the PatientMedData service into two services: PatientMedDataD, which contains all the data from the original PatientMedData, but is only accessible to the W:Doctor role, and PatientMedDataN, which does not include data from the CareOrders service, and is accessible to the W:Nurse role. In this case, we obtain the (global) role mappings from Figure 3(a) that maintains consistent global authorization across applications.

Roles and Information Flow. We now extend the role mapping algorithm in the presence of information flow. If one application keeps a copy of protected data from another application, it must control access to this copied data. Otherwise, a user that does not have access to the original application may be able to retrieve the same data through the target application. In our example, the Patient Records application maintains an archive of data from the Clinical Management

Global Role	Local Roles
G:Doctor	W:Doctor, C:Doctor, L:Clinician, P:Clinician
G:Nurse	W:Nurse, C:Nurse, L:Clinician, P:Clinician

(a)

Global Role	Local Roles
G:Doctor	C:Doctor, P:Clinician, L:Clinician
G:Nurse	C:Nurse, L:Clinician

(b)

Fig. 3. Role Mappings: (a) Separation, sufficiency, and ascription constraints (b) With information flow

and Laboratory applications. Thus, Patient Records must deny access to any users which do not have access to both the Clinical Management and Laboratory applications.

As shown in figure 2(b), the Patient Records application is populated with data in the following manner. The DataSync process periodically calls the Vitals and CareOrders services to retrieve patient clinical data older than a certain age, saves this data to the PatientHistory service, and then deletes the original copies from Clinical Management. This DataSync process runs as a super user on both systems, and thus does not have any issues with accessing the appropriate services. The TestResults service periodically connects directly to PatientHistory as a super user and saves any new test results since the last update. Finally, as with the web portal example, the CareOrders service makes a call to the TestOrders, but only relays the resulting data to its caller, without saving it locally.

When computing the global roles for this scenario, we enforce the same properties as listed before. In addition, we want to ensure that the users which can access the target service of a data synchronization are always a subset of the users which can access the source service. To implement this, we use sets of roles as a proxy for sets of users and thus add a new requirement:

4. **[Information Flow]** Whenever data flows from one service to another, the target service's global roles must be a subset of the source service's roles.

This ensures that the target service provides at least the same level of access control for the data as its originating service. Note that information may flow in the opposite direction as a call (e.g., the calls to Vitals and CareOrders by the DataSync process).

For the data synchronization scenario, we obtain the following additional constraints for the doctor and nurse related roles from information flow considerations:

1) From the information flow from Vitals to PatientHistory, the global roles for P:Clinician must be a subset of the global roles for the set $\{$ C:Nurse,C:Doctor $\}$.
2) From the information flow from CareOrders to PatientHistory, the global roles for P:Clinician must be a subset of the global roles for C:Doctor.
3) From the information flow from TestResults to PatientHistory, the global roles for P:Clinician must be a subset of the global roles for L:Clinician.

If we solve these constraints to find a set of global roles, we obtain the mapping from Figure 3(b). Note that this mapping shuts nurses out from accessing the PatientHistory service (by excluding role P:Clinician) — it exposes data from CareOrders, which is only visible to doctors. A naive mapping of roles that allows both doctors and nurses to access PatientHistory would subvert the access controls applied to CareOrders. This could be a serious issue, violating privacy regulations (such as HIPAA) and opening the system to abuse.

III. Semantics of Roles

We now define an interface describing an application's services, the roles which may access each service, and the possible outbound calls made by each service. In the associated technical report [11], we formally describe the runtime behavior of these interfaces using a small-step operational semantics.

A. Services

Let Names be a set of *web service names* and Users a set of *users*.

A *web application* $A = (\mathsf{Roles}, \mathsf{Services}, \mathsf{Perm})$ consists of a set of *roles* Roles, a set of *services* Services, and a *user permission mapping* $\mathsf{Perm} : \mathsf{Users} \rightarrow 2^{\mathsf{Roles}}$ from the (global) set of users to subsets of roles in Roles. A *(web) service* $S = (n, R, C, M)$ in Services consists of a name $n \in \mathsf{Names}$, a subset of roles $R \subseteq \mathsf{Roles}$ denoting the required permissions to call n, a set of called service names $C \subseteq \mathsf{Names}$, and a mapping $M : C \rightarrow 2^R$ from the services in C to subsets of R. We write A.Roles, A.Services, and A.Perm to refer to the roles, services, and user maps of A, respectively, and for a service S, we write $S.n$, $S.R$, $S.C$, and $S.M$ to reference the components of a service. We assume that there is exactly one service with name n, and we write Svc.n for that service. With abuse of notation, we identify a service S with its name $S.n$, and say, e.g., that a service n is in an application A.

The required roles are disjunctive — one of the roles must be satisfied to call the service. The mapping M represents a more precise subset of the roles known to be active when calling a service.

A *system* Sys consists of a set of applications, where we assume that the services and roles of the applications are pairwise disjoint. With abuse of notation, we speak of a service or role in a system for a service or role in an application in the system. Thus, we speak of a service $S \in \mathsf{Sys}$ if there is an application $A \in \mathsf{Sys}$ such that $S \in A.\mathsf{Services}$. We write $\mathsf{AllRoles} = \cup_{A \in \mathsf{Sys}} A.\mathsf{Roles}$ for the set of all (local) roles in system Sys.

We say that a system Sys is *well-formed* if, (a) [service names are unique] for each $n \in \mathsf{Names}$, there is at most one service S in Sys with $S.n = n$, (b) [all called services exist] for each application $A \in \mathsf{Sys}$, each service $S \in A.\mathsf{Services}$, and each called service name $n \in S.C$, there exists an application

$A' \in \mathsf{Sys}$ and a service $S' \in A'.S$ such that $S'.n = n$. We say that a system has *non-redundant* roles if no two roles are assigned to the same subset of the services, formally, if there does not exist an application A and roles $r_1, r_2 \in A.\mathsf{Roles}$ such that for all services $S \in A.\mathcal{S}$, we have $r_1 \in S.R$ iff $r_2 \in S.R$. Well-formedness and non-redundancy are syntactic checks, and henceforth we assume all systems have both properties.

Accessibility and Sufficiency. Let Sys be a system, S a service in Sys, and $u \in \mathsf{Users}$ a user. A call to S is *accepted* by Sys if $S.R \cap A_i.\mathsf{Perm}.u \neq \emptyset$. Otherwise, the call is *rejected*. Intuitively, a call to service S by user u is accepted if the user has at least one of the roles required to execute the service.

For a system Sys, the function $\mathsf{rolesOf} : \mathsf{Users} \to 2^{\mathsf{AllRoles}}$ maps each user u to the set of roles available to u, that is, $\mathsf{rolesOf}.u = \cup_{A \in \mathsf{Sys}} A.\mathsf{Perm}.u$. A set of services $\mathsf{Services}$ is *accessible* to a set of roles R if for each service $S \in \mathsf{Services}$, there exists an $r \in R$ such that $r \in S.R$. Similarly, a set of services $\mathsf{Services}$ is *accessible* to a user $u \in \mathsf{Users}$ if $\mathsf{Services}$ is accessible to $\mathsf{rolesOf}.u$. In this case, all calls by u to any service in $\mathsf{Services}$ will be accepted; however, transitive calls made by these services may cause authentication failures.

We wish to ensure that if a call is accepted by a system, no further authorization errors can occur. This is provided by the stronger notion of *sufficiency*. We say that a set of roles R is *sufficient* if, for every user u with $\mathsf{rolesOf}.u = R$ and service S accessible to R, a call by user u to service S will not result in an authorization error. A system is sufficient if for all users $u \in \mathsf{Users}$, we have that $\mathsf{rolesOf}.u$ is sufficient.

B. Role Compatibility

For systems with a single application, sufficiency can be checked by a dataflow analysis [19]. In general though, each application in a system comes with its own notion of local roles, and a call in application A_1 to service S in application A_2 only provides information about roles in A_1 held at the call point, not the roles in application A_2. Thus, in order to check sufficiency, we must somehow “convert” the local roles in each application to a global set of roles. We introduce *global role schemas* to do this.

Global Role Schema. Let Sys be a system. A *global role schema* $\mathsf{Grs} = (\mathcal{R}, G)$ consists of a set $\mathcal{R}$ of *global role names* and a mapping $G : \mathcal{R} \mapsto 2^{\mathsf{AllRoles}}$ that maps global role names to sets of local roles in the system Sys. This schema guides role assignments for individual users: if a user is assigned to a global role $g \in \mathcal{R}$, then that user must also be assigned all local roles in the set $G.g$.

We can also take an existing set of user assignments and see whether it corresponds to our global role schema. We say that the assignment of roles $\{A.\mathsf{Perm} \mid A \in \mathsf{Sys}\}$ *conforms* to a global role schema Grs if there exists a user to global role assignment $\mathsf{Perm} : \mathsf{Users} \to 2^{\mathcal{R}}$ such that for all users $u \in \mathsf{Users}$

$$\bigcup_{g \in \mathsf{Perm}.u} G.g = \mathsf{rolesOf}.u$$

That is, there is a mapping of users to global roles such that the set of local roles designated by the global role schema to each user u is exactly the set of local roles $\mathsf{rolesOf}.u$ assigned to the user.

Sufficiency. A global role schema Grs is *fully sufficient* if, for each global role $g \in \mathcal{R}$, the set of local roles $G.g$ is sufficient. Given a user-role assignment that conforms to a fully sufficient role schema, any service call by an arbitrary user will either be rejected or execute without authentication failure.

Separation. A global role schema $\mathsf{Grs} = (\mathcal{R}, G)$ has *role separation* if no two roles from the same application map to the same global role, that is, for all $g \in \mathcal{R}$ and $A \in \mathsf{Sys}$, we have $|G.g \cap A.\mathsf{Roles}| \leq 1$.

Role separation ensures that the roles of each application can be assigned to users independently. If multiple roles of an application appear in the same global role, these roles are effectively combined, potentially violating the intent of the original roles (e.g., allowing users access to data they should not see).

Minimality. Minimality encodes the requirement that a global role schema should not grant access to more services than necessary to ensure sufficiency. A set of local roles R is *minimal* if it is sufficient and there exists an $l \in R$ such that any subset of R containing l is not sufficient. We extend minimality to global role schemas as follows: a global role schema is *minimal* if there exists an injective mapping $\mu : \mathcal{R} \to \mathsf{AllRoles}$ from global roles to local roles $\mathsf{AllRoles}$ such that (a) for all $g \in \mathcal{R}$ we have $\mu.g \in G.g$, and (b) any subset of $G.g$ containing $\mu.g$ is not sufficient. These conditions ensure that each global role g has unique local role which requires the local role set $G.g$ for sufficiency. Note that there may be more than one minimal global role schema.

Global Schema Inference. The *global schema inference problem* takes as an input a system Sys and finds a minimal global schema Grs which has separation and is fully sufficient, if one exists.

Theorem 1: The global schema inference problem is NP-complete.

Given a global schema and a witness for minimality, one can check the properties in polynomial time. The hardness is by a reduction from one-in-three 3SAT. The details may be found in our technical report [11].

IV. Constraint-Based Inference

We solve the global schema inference problem through Boolean constraint solving. First, notice that, due to our minimality requirement, the number of global roles is at most the total number of roles in $\mathsf{AllRoles}$. Although each local role can be in one or more global roles, each global role must be sufficient for at least one local role. If the number of global roles is larger than the number of local roles $\mathsf{AllRoles}$, then global roles can be eliminated while still ensuring a sufficient global role for each local role.

We generate a set of global roles that include a local role in the following way. Fix a global role g. For each local role $r \in \mathsf{AllRoles}$, we define an atomic predicate r^g which is `true` if the role r is included in the global role g and `false` otherwise. The predicates r^g must satisfy the following constraints.

1) **[Separation Constraints]** No two local roles from the same application should be mapped to the same global role. That is, for each $A \in \mathsf{Sys}$, at most one local role $r \in A.\mathsf{Roles}$ can be in g. Thus, for each application $A \in \mathsf{Sys}$, we have (considering each r^g to be a 0-1 variable):
$$\sum_{r \in A.\mathsf{Roles}} r^g \leq 1 \tag{1}$$
or, equivalently,
$$\bigwedge_{A \in \mathsf{Sys}} \bigwedge_{r_1, r_2 \in A.\mathsf{Roles}, r_1 \neq r_2} (\neg r_1^g \vee \neg r_2^g)$$
2) **[Sufficiency Constraints]** The sufficiency constraints dictate that for each service S and each service $c \in S.C$ called from S, if one of the roles in $S.M.c$ is mapped to the global roles g, then one of the roles in $\mathsf{Svc}.c.R$ must also be mapped to g. That is,
$$\bigwedge_{A \in \mathsf{Sys}} \bigwedge_{S \in A.\mathsf{Services}} \Big(\bigvee_{r \in S.M.c} r^g \Big) \rightarrow \Big(\bigvee_{\hat{r} \in \mathsf{Svc}.c.R} \hat{r}^g \Big) \tag{2}$$

Let ϕ_{Sys} be the conjunction of the constraints from Equation 1 and Equation 2. Clearly, ϕ_{Sys} is polynomial in the size of Sys. A satisfying assignment for ϕ is a function mapping each r^g to $true$ or $false$ such that ϕ evaluates to $true$.

Theorem 2: Let ρ be a satisfying assignment to ϕ_{Sys}. Then the set of roles $\{r \mid \rho.r^g = true\}$ is a global role which is fully sufficient and has role separation.

Given the constraints, we can find a global group containing local role r by conjoining ϕ_{Sys} with r^g. We place in the group the set of local roles which are assigned `true` in a satisfysing assignment for this constraint. This set of local roles $\mathcal{R}_l$ may include roles which are not required for sufficiency. We can then compute a new set of local roles $\mathcal{R}_l'$, which does not include extraneous roles, in polynomial time. We starting with $\mathcal{R}_l' = \{r^g\}$ and iteratively add roles from $\mathcal{R}_l$ back into the group, as needed ensure sufficiency of the services accessible by the current $\mathcal{R}_{\updownarrow}'$.

To construct a global role schema G, we iterate through the set of local roles AllRoles, finding a global role group for each local role. If there is no satisfying assignment to $\phi_{\mathsf{Sys}} \wedge r^g$, where r is one of the local roles in AllRoles, we stop with `no_solution`. The complete algorithm for global schema inference may be found in the technical report [11].

Theorem 3: If global schema inference returns a global role schema G for Sys, then G has role separation, is fully sufficient, is minimal with respect to the role signatures of Sys, and each local role appears in at least one global role. If global schema inference terminates with `no_solution` for Sys, then no such global role schema exists for Sys.

Solving ascribed roles. The administrator can specify a subset R' of local roles such that there must be a global role g with $R' \subseteq G.g$. The above algorithm does not address these *role ascriptions*. To extend it for ascribed roles, we define the following constraints.

- **[Ascription Constraints]** For each ascription $\{r_1, \ldots, r_k\}$, we have $r_1^g \leftrightarrow r_2^g \leftrightarrow \ldots \leftrightarrow r_k^g$.

We first solve for each of the ascribed roles, conjoining the associated ascription constraint with ϕ_{Sys}. When we remove redundant roles from ascribed groups, we start with an $\mathcal{R}_l'$ which includes all the roles associated with the ascription. After solutions are found for each ascribed group, we then solve for the remaining roles without any ascription constraints. Note that we permit ascribed groups to be extended as needed to achieve sufficiency. If all local roles are ascribed, then the problem is reduced to *global schema checking*, rather than *global schema inference*.

V. Services with Information Flow

We now extend our results to services and systems where we model flow of sensitive data between applications. We must now ensure that information that can only be accessed under some role constraints are not "disclosed" to applications that do not hold the required roles.

To model information flow, we extend the services to include a directed *information flow graph*. Thus, a service is now a 5-tuple (n, R, C, M, I), where (n, R, C, M) are as before, and $I \subseteq (C \cup \{n, \mathsf{Caller}_{in}\}) \times (C \cup \{n, \mathsf{Caller}_{out}\})$ is a set of pairs of service names (or the special symbols Caller_{in} and Caller_{out} denoting the entry and exit points respectively of a caller of the service). A pair $(n_1, n_2) \in I$ represents an information flow from n_1 to n_2, which may occur when the service is called, and self-links of the form (c, c) are not permitted. The information flow graph models two forms of information flow: *synchronization*, where data from one service is saved in another service, and *disclosure*, where data from a service is made available to callers of a (potentially different) service.

Given a service S and callee $c \in S.C$, the callee-to-self link $(c, S.n)$, represents a synchronization of data from service c to S. The callee-to-caller link $(c, \mathsf{Caller}_{out})$ represents a disclosure of data from c by S.

Synchronization and disclosure are distinguished from benign *non-disclosing transfers*, where a service moves data between two other services without saving or disclosing it. Callee-to-callee and caller-to-callee links, where there is no additional link from the source to the current service or its caller, are all non-disclosing.

Information Flow and Non-disclosing Global Schema. Informally, we say that a global role schema Grs is *non-disclosing* for a conforming user assignment, if it does not permit the disclosure to a user $u \in \mathsf{Users}$ of data originating at a service S for which the user does not have access. Informally, this is the requirement that a user cannot subvert access control rules by exploiting information flow between services.

A global role schema Grs is *non-disclosing* if there does not exist services S and S', a global role g, and a user u with $\mathsf{rolesOf}.u = G.g$ such that (a) role g does not have access to service S: $G.g \cap S.R = \emptyset$, (b) role g has access to service S': $G.g \cap S'.R \neq \emptyset$, and (c) $\mathsf{Flow}(S.n, S'.n)$ is true.

System	Num svcs	Num calls	Num grps	Max pred	Time (ms)
portal1	8	5	N/A	96	4
portal2	9	8	5	92	5
data_sync	8	5	6	69	5
it_mgt	7	6	5	94	5

Fig. 4. Performance results for ROLEMATCHER

To compute global role schemas that are non-disclosing, we conjoin additional constraints with ϕ_{Sys} to ensure only permitted information flow. Details may be found in the associated technical report [11].

VI. EXPERIENCES

We have implemented ROLEMATCHER, a tool to infer global role schemas. Our tool takes as input a textual representation of the Sys definition described in section III. It produces a global role schema via our constraint generation algorithm, using the MiniSat [8] satisfiability solver to resolve the boolean constraints.

Figure 4 summarizes the results of running ROLEMATCHER on several small examples, using a Dell PowerEdge 1800 with two 3.6Ghz Xeon processors and 5 GB of memory. The "Num svcs" and "Num calls" columns represent the total number of services in the system description and the total number of service calls, respectively. "Num grps" lists the number of groups in the inferred schema (or N/A if no solution was possible). "Max pred" is the size of the largest predicate passed to the solver and "Time" the elapsed time in milliseconds to find and print the global role schema.

We ran our tool on four different system definitions. `portal1` and `portal2` correspond to the clinic web portal of Figure 2(a), `data_sync` corresponds to the data synchronization example of Figure 2(b), and `it_mgt` represents the case study described below. Since the problem is NP-complete, the use of an exponential procedure is inevitable. Even though the algorithm involves SAT solving, this has not been a bottleneck. This is because the constraints are a combination of 2-literal clauses (for separation) and Horn clauses (for sufficiency), and while this combination is already NP-complete, Boolean constraint propagation and unit resolution heuristics in a modern SAT solver are particularly tuned for these types of clauses.

Case Study. To further evaluate our approach to role interoperability, we considered a real-world scenario described to us by an industrial colleague. An IT management applications company had several independently-developed products from companies it had acquired. The company wished to integrate these applications (including their security models) in order for customers to use them as an end-to-end solution to their IT management needs.

We focused on two applications in the company's product line. The *IT System Management* (ITSM) application includes modules for incident, problem, and change management, as well as a database to track a company's hardware and software assets. The *Patch Management* application gathers an inventory of the patches currently installed on the company's computers and manages the application of new patches.

The integrations between these systems are straightforward: the patch inventory data should be included in the ITSM asset database (the arrow from *Discovery Data* to *Asset Management*) and the application of patches should be controlled via the ITSM change management module (the arrow from *Change Management* to *Apply Patch*.

Both systems use role-based access control, but with very different role models. The Patch Management application has a very simple model with three fixed roles: *User*, *PowerUser*, and *Admin*. The ITSM application allows system administrators at the customer to define their own roles and mappings to data/service access permissions. Thus, it is not feasible for the application vendor to ship a fixed role mapping. A better solution is to build a tool to extract role interfaces from the ITSM system's role metadata and infer a global role schema as a part of application deployment and configuration.

VII. RELATED WORK

Access control for web services. The eXtensible Access Control Markup Language (XACML) [2] defines an access control policy language for web services which is flexible enough to express many access control models, including RBAC [1]. However, XACML policy definition and enforcement are not tied to the underlying access policies of individual services. Thus, while clearly a useful tool for defining security policies, XACML does not address the basic issues addressed by global role schema inference. On the other hand, XACML policies could be generated from a global role schema. This would enable centralized enforcement while avoiding the problems associated with two independent policy layers.

[20] proposes an access policy language which can reference the past history of service invocations, using pure-past linear temporal logic (PPLTL). Like XACML, it uses a centralized approach to policy specification and enforcement. In order to reason about access policies across systems, the administrator must provide a role mapping. Thus, it can be used as a layer on top of global schema inference.

Other approaches to ensuring access control constraints are possible. For example, in situations where multiple providers for a service are available, a broker can dynamically select among the available providers to satisfy security requirements [5].

Other web service security standards. Other standards address orthogonal issues to access control: Security Assertions Markup Language (SAML) provides a framework for querying authentication and authorization statements across security domains, WS-Security defines how encryption and digital signatures may be applied to web service messages, and WS-Policy establishes a format for services to advertise their security requirements.

Theory of access control policy interoperation. [12] investigates the theory behind interoperability of ACL (Access

Control List) based security models. The desired interoperation between systems is specified as a set of accessibility links between principals of the individual systems and a set of restrictions between principals. It is shown that finding a maximal subset of the accessibility links, such that the security constraints of the individual systems are not violated, is NP-Complete. [4] shows a similar result for the interoperation of partial order based security models.

Role mapping. A role-based access control policy may be seen as an abstraction of an underlying ACL security policy. *Role mapping* [21], [22] attempts to find this abstraction automatically by finding a minimal set of roles for a single system which captures the underlying relationship between users and the resources they may access. [21] shows that this problem is NP-Complete and provides algorithms for both full and approximate solutions.

Role mapping may be extended to address the interoperability of RBAC systems. In this context, it is assumed that an inter-system call will include the set of underlying permissions required on the target system. *Inter-domain role mapping* (IDRM) attempts to find the minimal set of target system roles which satisfy the requested permissions [18], [7], [6]. [3] presents a static version of IDRM where roles are directly mapped between systems. Links between roles are determined by grouping similar objects across systems (e.g., accounts, insurance claims, etc.) and then linking their underlying permissions (e.g., access to accounts on system A implies access to accounts on system B). Mappings between roles attempt to satisfy as many of these links as possible while avoiding the subversion of the individual systems' access policies. This problem is formulated as a system of integer programming constraints.

When inferring a global role schema, we use each service interface's set of called services as a proxy for required access permissions. This is appropriate and necessary for systems whose internals are encapsulated by services. In addition, our approach globally optimizes the role mappings to minimize the number of mappings needed while preserving full interoperation.

Static analysis of RBAC systems. [19] describes a static analysis for roles within a single Java Enterprise Edition application. This analysis checks for three types of errors: indirect authorization errors, redundant role definitions, and the subversion of an RBAC policy by exploiting unchecked intra-component calls. Our work can be viewed as extending these static checks across systems.

Information flow. [14] describes a static program analysis to compute information flow in a modular fashion. This work has inspired the use of decentralized information flow security in programming languages [15], operating systems [9], [13], and web service compositions [16]. Rather than use information flow as *the* access control mechanism, we use information flow to inform a standard access control policy, leveraging the use of existing RBAC infrastructure.

Information flow has also been studied in the context of RBAC. [17] computes the information flow for a single system's RBAC policy due to two causes: 1) the ability to pass data between two objects protected by the same role, and 2) the ability to pass data between objects protected by different roles, when those roles may be simultaneously activated.

VIII. Discussions

Access control integration for systems interacting via service calls is an important industrial problem. Current solutions either perform expensive and error-prone manual integration, or ad hoc mining of access control rules from logs (e.g., Securent Entitlement Management (acquired by Cisco) or Vaau RBACx (acquired by Sun)) that are then centrally managed and enforced. In contrast, our global constraint analysis, together with our precise formulation of the semantics, allows us to make precise claims of correctness with respect to sufficiency and confidentiality.

Our approach does have limitations —theoretical and pragmatic— that we are addressing in current work. In the event that no global role schema satisfying our constraints exists for a given system, we currently just report an error. Going forward, we would like to provide more guidance to the user in this situation. One approach is to provide more feedback about which services and constraints actually cause the infeasibility. This information can be obtained as an *unsatisfiable core* from the SAT solver.

We can also attempt to relax constraints or drop inter-system calls and return to the user a solution which satisfies as many of the constraints as possible. This approach has been proposed for other access control interoperability situations [12]. For example, upon finding that no full solution is possible, one might selectively relax the role separation constraints until a sufficient global schema is found.

In a large collection of systems, the appropriate mappings between roles may not be obvious. If several candidate roles can satisfy the constraints for a given group, the system arbitrary picks one of them. The administrator may influence this process by using role ascriptions. In fact, our approach permits the administrator to pick any point on the spectrum from full role inference to the validation of a completely specified global role schema. However, manually providing ascriptions for all the roles in a large enterprise may be quite tedious. One solution is to use existing role mining techniques [21], [22], [3], to automatically compute an initial set of ascriptions and run our algorithm using this initial guess. If these ascriptions do not admit a global role schema, the solver could relax them and consider other role mappings as needed.

References

[1] XACML Profile for Role Based Access Control (RBAC), February 2004.

[2] eXtensible Access Control Markup Language (XACML) Version 2.03. OASIS Standard, February 2005.

[3] E. Bertino, A. Ghafoor, J. Joshi, and B. Shafiq. Secure interoperation in a multidomain environment employing rbac policies. *IEEE Trans. on Kn. and Data Eng.*, 17(11):1557–1577, November 2005.

[4] P. Bonatti, M. Sapino, and V. Subrahmanian. Merging heterogeneous security orderings. In *EOSRICS '96)*, pages 183–197, 1996.

[5] B. Carminati, E. Ferrari, and P. Hung. Security conscious web service composition. In *ICWS '06*, pages 489–496, 2006.

[6] L. Chen and J. Crampton. Inter-domain role mapping and least privilege. In *SACMAT '07*, pages 157–162. ACM, 2007.
[7] S. Du and J. Joshi. Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. In *SACMAT '06*, pages 228–236. ACM Press, 2006.
[8] N. Een and N. Sörensson. An extensible SAT-solver. In *SAT '03*, pages 502–518, 2003.
[9] P. Efstathopoulos et al. Labels and event processes in the asbestos operating system. In *SOSP '05*, pages 17–30. ACM, 2005.
[10] D. Ferraiolo and R. Kuhn. Role-based access control. In *15th National Computer Security Conference*, 1992.
[11] J. Fischer and R. Majumdar. A theory of role composition. Technical Report 080012, UCLA Computer Science Dept., April 2008. http://cs.ucla.edu:˜fischer/papers/tr080012.pdf.
[12] L. Gong and X. Qian. Computational issues in secure interoperation. *IEEE Trans. on Softw. Eng.*, 22(1):43–52, 1996.
[13] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. Kaashoek, Eddie Kohler, and R. Morris. Information flow control for standard os abstractions. In *SOSP '07*, pages 321–334. ACM, 2007.
[14] A. Myers and B. Liskov. A decentralized model for information flow control. In *SOSP '97*, pages 129–142. ACM, 1997.
[15] A. Myers and B. Liskov. Protecting privacy using the decentralized label model. *ACM Trans. Softw. Eng. Methodol.*, 9(4):410–442, 2000.
[16] K. Ono, Y. Nakamura, F. Satoh, and T. Tateishi. Verifying the consistency of security policies by abstracting into security types. In *ICWS '07*, pages 497–504, 2007.
[17] S. Osborn. Information flow analysis of an rbac system. In *SACMAT '02*, pages 163–168. ACM, 2002.
[18] S. Piromruen and J. Joshi. An rbac framework for time constrained secure interoperation in multi-domain environments. In *WORDS '05*, pages 36–58. IEEE, 2005.
[19] Marco Pistoia, Stephen J. Fink, Robert J. Flynn, and Eran Yahav. When role models have flaws: Static validation of enterprise security policies. In *ICSE '07*, pages 478–488. IEEE Computer Society, 2007.
[20] M. Srivatsa, A. Iyengar, T. Mikalsen, I. Rouvellou, and J. Yin. An access control system for web service compositions. In *ICWS '07*, pages 1–8, 2007.
[21] Jaideep V., Vijayalakshmi A., and Q. Guo. The role mining problem: finding a minimal descriptive set of roles. In *SACMAT '07*, pages 175–184. ACM Press, 2007.
[22] D. Zhang, K. Ramamohanarao, and T. Ebringer. Role engineering using graph optimisation. In *SACMAT '07*, pages 139–144. ACM Press, 2007.