Distributed Enforcement of Unlinkability Policies: Looking Beyond the Chinese Wall

Apu Kapadia*
Institute for Security Technology Studies (ISTS)
Dartmouth College, USA

Prasad Naldurg
Microsoft Research, India

Roy H. Campbell
Department of Computer Science
University of Illinois at Urbana-Champaign, USA

Abstract

We present a discretionary access control framework that can be used to control a principal's ability to link information from two or more audit records and compromise a user's privacy. While the traditional Chinese Wall (CW) access control model is sufficient to enforce this type of unlinkability, in distributed environments CW is inefficient because its semantics requires knowledge of a user's access history. We propose a restricted version of the CW model in which policies are easy to enforce in a decentralized manner without the need for an access history. Our architecture analyzes system policies for potential linkability conflicts. Users can identify specific threats to their privacy, typically in terms of trusted and untrusted roles in the context of RBAC (role based access control), following which the system attaches automatically generated policy constraints to the audit records. When these constraints are enforced appropriately, they implement unlinkability policies that are provably secure and precise for a fixed protection state. We extend the model with a versioning scheme that can handle evolving protection state, including changing roles and permissions, trading precision to maintain the security of deployed policies.

*Apu Kapadia was funded in part by the U.S. Dept. of Energy's High-Performance Computer Science Fellowship through Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and Sandia National Laboratories, and in part by the Bureau of Justice Assistance, under grant 2005-DD-BX-1091. The views and conclusions do not necessarily reflect the views of the sponsors.

1 Introduction

We address the privacy concerns of users accessing services within an organization such as a university or corporate network. Local and external administrators from across various departments may monitor a user's accesses to various resources through audit records. In the case where "linking" these accesses is not explicitly sanctioned by a mandatory policy, such linkages can compromise the privacy of users. We present a model that allows users to express and refine unlinkability concerns and have them enforced by the system without the need for explicit coordination such as maintaining a user's access history.

In the context of our problem, *unlinkability* is defined as the infeasibility of an adversary to correlate two transactions initiated by the same user. To address this problem, researchers have proposed a number of cryptographic mechanisms to construct anonymous credentials [3, 1, 2, 7] that make it computationally infeasible for a server to link the use of these credentials. There are systems, however, in which users simply cannot be anonymous. For example, an organization may be required to keep detailed audit records about who accessed payroll information by law. In such systems, it becomes important to provide unlinkability through access control, allowing for linkability in only certain cases, e.g., legal subpoenas, or only by certain trusted individuals in the organization. In this paper we focus on providing unlinkability through access control, with the observation that denying access to related audit records prevents the possibility of linking the

Eighth IEEE International Workshop on
Policies for Distributed Systems and Networks (POLICY'07)
0-7695-2767-1/07 $20.00 © 2007 IEEE

contents of the audit records.[1]

We consider the setting in which a variety of databases with independent access control mechanisms often store the audit records of users' actions and service invocation requests across departmental boundaries. This makes enforcement of unlinkability a difficult task. In theory, centralized mechanisms based on the Chinese Wall (CW) model can solve this problem. For example, a CW policy could ensure that administrators can access at most one dataset within a "conflict of interest" (COI) class. By grouping datasets corresponding to users actions into a COI class, the CW model will ensure that no administrator can access two or more such datasets, thereby providing unlinkability. The implementation of such a policy, however, must maintain a history of administrators' accesses. In a distributed setting, maintaining this access history across departments may render this approach infeasible or impractical. Centralized solutions present a bottleneck for distributed access to resources and act as a single point of failure for access control. Distributed enforcement of CW policies requires the propagation of history information that must be kept consistent across different databases, incurring high communication and computational overheads.

In this context, we introduce an access control model for **policy-based unlinkability**, which is a restricted form of the CW model. Our model addresses the problem of restricting accesses by a single domain administrator to multiple audit records belonging to the same user as defined by a "session" and does not require an implementation to maintain the access history of users in the system. We outline an enforcement framework based on this model that can analyze the system protection state for unlinkability threats, and change the authorizations based on the user's requirements (except when they are explicitly required by system policy) to counter these threats. For example, given a set of services as defined by Alice's *session*, the system may inform Alice that network administrators can access audit records of her accesses to multiple services within her session, and thereby track her actions in the organization. Alice can negotiate a set of constraints to prevent certain administrative users from linking her transactions, while allowing certain trusted administrators to do so. These constraints are attached to individual audit records and local access control decisions can be made based on these constraints, allowing for the distributed enforcement of users' unlinkability policies.

We prove that our framework is both secure and precise with respect to enforcing the negotiated policies. We first prove these results under the strong tranquility assumption where the protection state (role and permission assignments) of the system does not change over a session, i.e., the administrators' access rights remain fixed. Subsequently, we show how we can relax these assumptions and present an approach that uses versioning to handle changes in the authorizations under a weak tranquility assumption, sacrificing precision for the ability to change protection state. Using versioning we can always identify the set of users for which the policies are secure and precise. In both cases we show how users can add new flows to their existing sessions, refining their unlinkability requirements iteratively. In other words, it is not necessary for a user Alice to specify all her possible accesses beforehand.

We briefly summarize our contributions:

1. We introduce an access control model for scalable and decentralized enforcement of unlinkability policies without the need for maintaining the access history of users (Section 2).

2. We outline an enforcement architecture based on our access control model and prove that it is secure and precise for fixed protection state (Section 3).

3. To address evolving protection state, we present an approach based on versioning. We prove that our approach maintains the security of deployed unlinkability policies by trading off precision for evolving protection state (Section 4).

We present discussion, related work and conclusions in Sections 5, 6 and 7 respectively.

2 Access control model

In this section, we provide some background on the Chinese Wall model and present our model in relation to its simple security condition.

2.1 Chinese Wall Model

Policies in the Chinese Wall (CW) Model group objects belonging to a company into a *Company Dataset* (CD). CDs can be grouped into *Conflict*

[1] Denying access to a record does not leak information about a user's access. Administrators issue queries for records such as "Alice's access to the wireless network," and feedback of the form "access denied or record does not exist" does not reveal whether Alice accessed the wireless network.

Eighth IEEE International Workshop on
Policies for Distributed Systems and Networks (POLICY'07)
0-7695-2767-1/07 $20.00 © 2007 IEEE

of Interest (COI) classes. The overall goal of this model is that no subject can read objects from two or more CDs within the same COI class. Once a subject reads from a particular CD, future accesses to other CDs within the COI class are denied. In particular, the semantics of CW policies allow an individual to access objects from any one CD of their choosing in a COI class, and prevents further accesses to objects in other CDs in that class. Typically, Chinese Wall policies are enforced using centralized history-based approaches [10] or require explicit coordination [8], which is expensive in practice.

Let $CD(o)$ be the CD of object o (similarly $COI(o)$) and $PR(u)$ be the set of objects that u has read. Each object belongs to exactly one COI class. This behavior is formalized in the following security condition:

Definition 1. CW-Simple Security Condition: *A subject u can read an object o if and only if any of the following holds:*

1. *There is an object o' such that u has accessed o' and $CD(o') = CD(o)$.*
2. *For all objects o', $o' \in PR(u) \Rightarrow COI(o') \neq COI(o)$.*

Since our system addresses only *read* accesses to objects, we omit discussion of the CW-*-Property that governs *write* accesses.

2.2 Our model

In this subsection, we show how we can modify the simple security condition in the CW model to capture the notion of unlinkability in our context. We assume a distributed system for sharing resources that allows us to specify and enforce system-wide access control policies. Users in the system access services by presenting credentials resulting in an *access transaction*. Objects (audit records) related to an access transaction are stored in one or more databases.

Definition 2. *An* **audit flow** *for a given access transaction is the set of audit records $I = \{o_1, o_2, \ldots, o_m\}$ related to that access transaction. A collection of audit flows $S = \{I_1, I_2, \ldots, I_n\}$ that a user desires to keep unlinkable, is called a* **session**.

Sessions are associated with individual users and may be open-ended, i.e., they last for the lifetime of the system and users are allowed to update the list of transactions in their session.

It is possible to enforce unlinkability policies using the Chinese Wall model. The user's session can be specified as a COI class, where each audit flow is equivalent to a CD. The CW model would prevent administrators from accessing audit records of two or more audit flows within the specified session. The distributed enforcement of CW policies, however, requires a mechanism to record and disseminate access history across the system. We present an alternative access control model that *does not maintain the access history of users*, making distributed enforcement of linkability policies more efficient.

We assume that users and access permissions are organized into roles using RBAC. The underlying RBAC system governs read accesses to audit records in the absence of unlinkability policies. We refer to these permissions as "static read access":

Definition 3. *If in the underlying RBAC system a user* u *has read access to a database* d*, then we say that* u *has* **static read access** *to all the audit records in database* d*. Similarly,* u *has static read access to an audit flow I if* u *has static read access to any audit record in I. Static read access can be overridden by linkability policy constraints.*

Our unlinkability specification will guarantee that subjects with static read access to two or more different audit flows within a session are denied access to *any* object within that session. Note that the set of behaviors allowed in this model is more restrictive than CW, which allows access to one object of the subject's choosing. We show that our model's semantics allows the enforcement of policies in a decentralized setting, where access decisions can be made local to the object being accessed.

Recall, that an object o is an audit record for some transaction $T(o)$. Let $I(o)$ be the audit flow for transaction $T(o)$. Let $Session(o)$ be the session that contains $I(o)$ (we assume that an audit flow can be part of only one session).

Users supply a set of roles called the "deny set" that the unlinkability policies should apply to. For example, a student Alice may not want other students (e.g., part-time student administrators) to link her transactions, and would add the Student role to her deny set. Subjects not in the deny set are not explicitly denied from linking audit flows within the session. We explain this in more detail in Section 3. Let $DenySet(o)$ be the deny set associated with $Session(o)$.

We define the simple security condition for Unlinkability as follows:

Eighth IEEE International Workshop on
Policies for Distributed Systems and Networks (POLICY'07)
0-7695-2767-1/07 $20.00 © 2007 IEEE

Definition 4. Simple Security Condition (SSC1) for Unlinkability: *A subject u is granted read access to an object o if and only if the following hold:*

1. *u has static read access to o.*
2. *if $u \in DenySet(o)$ and there is no object o' such that u has static read access to o', and o and o' belong to different flows in the same session ($Session(o') = Session(o)$ and $I(o) \neq I(o')$).*

This means that subjects in the deny set of a session with static read access to two or more audit flows within that session are denied access to any objects within the session. This semantics does not give subjects the ability to access exactly one flow of their choosing within a session (as in the CW model) and is applied only to subjects identified as unlinkability threats by the user (as defined by the user's deny set for that session). SSC1 can be enforced easily if all the audit flows in a session are specified in advance. For "open-ended" sessions where all the audit flows are not known in advance (a more realistic assumption), however, we modify SSC1 to allow read access to at most one audit flow within a session, but access to the flow of the subject's choosing is not guaranteed. We call this SSC2:

Definition 5. Simple Security Condition 2 for Unlinkability (SSC2): *A subject u is granted read access to an object o if the following hold:*

1. *u has static read access to o.*
2. *if $u \in DenySet(o)$ and there is no object o' such that u has static read access to o', and o and o' belong to different flows in the same session ($Session(o') = Session(o)$ and $I(o) \neq I(o')$).*

Furthermore, if a subject u is granted read access to an object o then the following hold

1. *u has static read access to o*
2. *u will not be granted read access to any object o' such that $I(o) \neq I(o')$ and $Session(o') = Session(o)$*

In short, a subject with static read access to two or more audit flows in a session may be able to access zero or one audit flow in the session, but access to a particular audit flow of the subject's choosing cannot be guaranteed.

In the remainder of this paper we show that our access control model can be realized in a distributed environment by attaching policies to data, which are enforced locally. Since evolving protection state can result in a violation of the simple security condition, we present a system that uses versioning to maintain the security of deployed policies.

3 Enforcement architecture

Throughout this paper we will refer to three types of policies. *Flow policies* are explicit representations of data flows between databases. The policy (d_1, d_2) allows database d_1 to propagate copies or transformations of data to d_2. The system can use these flow policies to construct graphical representations of audit flows throughout the system. *Access policies* are Permission-Role assignments (d, r), where role r may access database d. Lastly *linkability policy constraints* are described in Section 3.2, and are attached to audit records. Access to an audit record is granted to users based on the access policy for that database, and the linkability policy constraints of that audit flow, which can override the former.

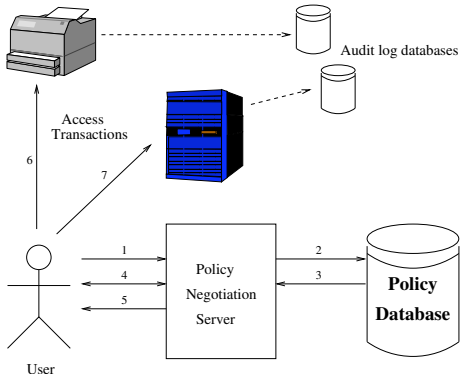

Figure 1. System Architecture

We now present a high level overview of our enforcement architecture using an end-user example scenario as shown in Figure 1. (1) In the first step, a concerned user Alice sends her session information to the policy negotiation server (PNS). This is a set of identifiers (or unique types) corresponding to access transactions to unique servers. In steps (2)-(3), the PNS looks up relevant information for each service including access policies and flow policies, builds the audit flows $I_1, \cdots, I_n$, and analyzes them for unlinkability conflicts. The PNS presents Alice with a set of roles whose users can access her audit-record information from two or more audit flows, e.g., *Security Officer* and *Student Administrator*.

Eighth IEEE International Workshop on
Policies for Distributed Systems and Networks (POLICY'07)
0-7695-2767-1/07 $20.00 © 2007 IEEE

In Step (4) Alice identifies her discretionary unlinkability requirements in terms of roles (we call this Alice's "deny set") whose users she wants to prevent from linking her audit information, e.g., *Student Administrator*. The PNS may not be able to enforce some of Alice's choices if there are mandatory access requirements, e.g., the request to add *Security Officer* to Alice's deny set may be disallowed by mandatory system policy. After Alice and the PNS agree on Alice's deny set, (5) the PNS sends Alice a certificate with policy constraints for her audit records. This certificate is digitally signed and is tagged to Alice's audit data. The PNS also stores Alice's discretionary policies and session information in the policy database because Alice may later want to add more transactions to her existing session. In Steps (6)-(7), for each access transaction, Alice presents these certificates during authentication, which are attached to audit records that make up the audit flows. Access to an audit flow is allowed only if the accessing user's role is not precluded by the policy constraints. We assume that all interactions are cryptographically secured for authenticity, confidentiality, and integrity and that policy constraints are known only to Alice, the PNS, and the reference monitors that enforce those constraints.

3.1 Construction

We now propose an approach to enforce unlinkability for Alice's session by analyzing the roles that are explicitly granted static read access to each audit flow. In our construction, a PNS examines all the users in this set of roles and constructs a set of *conflicting roles* for that session.

Definition 6. *A role* c *is a* **conflicting role** *for a session S if there exists a user* u *assigned to role* c *who has static read access to two or more audit flows in session S.*

For example, say *Network Admin* has static read access to I_1, *Local Admin* has static read access to I_2, and there are users in the *Student* role that also belong to both *Local Admin* and *Network Admin*. In this case *Student* is a conflicting role for I_1 and I_2, since there exists a *Student* user with static read access to both audit flows. The PNS identifies the set of conflicting roles and presents this set to the user, who picks a subset of these conflicting roles as the "deny set." Alice may decide that *Student* administrators are indeed potential threats to her privacy and add *Student* to her deny set. Linkability policy constraints are generated that will ensure that all read accesses to audit flows satisfy SSC1. We also assume that for the purposes of accessing audit records, the system has access to all the roles that a user *can* activate, not only those that the user has activated currently.

Note that in our enforcement architecture students who are not linkability threats (i.e., those with static read access to only one flow), will still be allowed to access Alice's audit records. A reference monitor enforcing access to the audit-record database will check the policy constraints and allow or deny access appropriately. We formalize these concepts, and show how we can provide users with unlinkability with respect to audit flows. The key idea here is that Alice can specifically deny users of certain roles from linking her information. We believe that our system will provide users with a usable presentation of potential threats to their privacy in the form of conflicting roles, and allow users to easily select roles that they consider to be privacy threats.

In our technical report [5], we describe how the PNS constructs a user's *session graph* by analyzing flow and access policies in the system to identify conflicting roles (the description includes a detailed time complexity analysis). Briefly, the session graph models audit flows by representing databases and roles as vertices, and information flows as edges. This graph can be analyzed to identify conflicting roles. We assume that access transactions are defined only periodically in a system, and the PNS can reuse audit flow graphs to construct session graphs for users, thereby reducing the computational burden at the PNS. For simplicity, we avoid the details in this paper and focus on how policy constraints are generated and enforced once the conflicting roles have been identified.

3.2 Generating and enforcing policy constraints

Using the session graph, PNS returns to Alice the set of conflicting roles C in S. Alice picks a subset of these roles C_{Alice} as her *deny set*.

The members in Alice's deny set should be prevented from linking Alice's flows. Note that not all users in the deny set are linkability threats, and hence we need to make sure that only the users who can link Alice's flows must be denied access. We define Alice's policy constraints $\mathcal{P}_S$ for session S as the tuple $\langle C_{Alice}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$, where $\mathcal{R}_i$ is the set of roles with static read permission to information flow I_i (for compactness, $\mathcal{R}_i$ includes only those roles

that have a common user with some role in C_{Alice}).

Records for audit flows in session S are tagged with $\mathcal{P}_S$. When a user u attempts to access an audit record, the database's reference monitor first checks to see if u has static read access for that database. If so, it then checks the attached $\mathcal{P}_S$ to see if any of u's roles are in C_{Alice}. If so, the reference monitor checks to see if u's role-set $URA(u)$ has a non-empty intersection with at least two different sets in $\{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$. If so, the user has static read access to two or more flows in S, and the user is denied access by the reference monitor. In the worst case, for users with static read access to the database, the reference monitor needs to compute $n+1$ intersections, where each intersection takes $O(|URA(u)|+|\Gamma|)$ operations (using hash-tables), which is $O(|\Gamma|)$. Γ is the set of roles in the system. Hence the time complexity for evaluating $\mathcal{P}_S$ is $O(n|\Gamma|)$ if u is in Alice's deny set. If not, the time complexity is $O(|\Gamma|)$, the cost of computing the intersection $URA(u) \cap C_{Alice}$. As mentioned earlier, $\mathcal{P}_S$ is known only to Alice, the PNS, and the reference monitors at each database.

At this point, a valid question is why not generate policy constraints with user IDs. There are two reasons for this. Firstly, if a user u was identified to be a linkability threat, then adding u to the policy constraints will prevent u from accessing two or more flows. If u is removed from a particular role and is no longer a linkability threat, however, u will *still* be denied access. Our scheme adds more precision to the system by allowing users who are no longer linkability threats to access audit records. And secondly, in large systems we expect a role based formalism to be a more compact representation of linkability conflicts.

We now prove that our system is secure, sound, and precise under certain assumptions.

Definition 7. Strong Tranquility *asserts that the static access permissions associated with the users of the system (i.e., the user-role assignment (URA) and the permission-role assignment (PRA)) do not change by system operation.*

Policy constraints are generated based on the current protection state of the system (i.e., the URA and the PRA). Changes to the protection state can result in policy constraints that are "out of date." We first prove that our constraints are *secure*, *sound*, and *precise* with under the strong tranquility assumption. We relax this assumption in Section 4 and show how we can trade precision for security when the protection state and the session information are allowed to change. The following theorems trivially hold because of the strong tranquility assumption, which makes the properties hold by construction of session graph S and policy constraints $\mathcal{P}_S$.

Due to space limitations, we omit the proofs of the following theorems. Please refer to our technical report [5] for the proofs of theorems in this paper.

Theorem 1. (SSC1 Security) *Assuming strong tranquility, if a user* u *with a role in Alice's deny set C_{Alice}, has static read access to two or more audit flows in Alice's session $I_1, \cdots, I_n$, the policy constraints will prevent* u *from accessing these flows. Furthermore, Alice was presented with all of u's roles as conflicting roles.*

Theorem 2. (Soundness) *Assuming strong tranquility, if a user* u *is denied access to a flow I_i by the policy constraints, then the user has static read access to two or more audit flows in the session S.*

The following theorem is simply the contrapositive of Theorem 2. In the following sections we will only refer to security and precision, since precision follows from soundness.

Theorem 3. (Precision) *Assuming strong tranquility, if a user* u *has static read access to exactly one audit flow within a session, then* u *is not denied access by the policy constraints.*

3.3 Open-ended sessions

Our construction maintains security and precision for a predefined session. Consider the case when user Alice does not know all her transactions *a priori*. Alice would like to dynamically generate constraints for new audit flows, without invalidating her constraints to older audit flows. We extend our algorithm to allow users to add audit flows to existing sessions and generate new constraints appropriately.

Constraints for a new flow I_{n+1} can be generated by extending Alice's existing session graph for $I_1, I_2, \ldots, I_n$. We prove that enforcing the incrementally-generated constraints for each flow in an open-ended session provides security and precision with respect to SSC2.

Theorem 4. (SSC2 Security)

Assuming strong tranquility, if a user u *with a role in Alice's deny set C_{Alice}, has static read access to two or more audit flows in Alice's session $I_1, \cdots, I_{n+1}$, then the policy constraints will prevent* u *from accessing* **two or more** *of these flows.*

Eighth IEEE International Workshop on
Policies for Distributed Systems and Networks (POLICY'07)
0-7695-2767-1/07 $20.00 © 2007 IEEE

Theorem 5. (Precision)
Assuming strong tranquility, if a user u *has static read access to exactly one audit flow within a session, then* u *is not denied access by the policy constraints.*

3.4 Mandatory audit flows

The PNS may consider access by certain conflicting roles to be mandatory. For example, the PNS may mandate that student administrators cannot be denied access, and are exempt for policy constraints. If there are exempted users that can access two or more audit flows, the user is informed of this. Our goal is to make the privacy implications of sensitive information explicit to the user. Users will have complete information of who can link the user's information, and will proceed only if they agree to the PNS's mandatory policy.

In the next section, we relax the strong tranquility assumption and present a discussion of what policies we can enforce when the permissions are allowed to change and investigate the trade-off between security and precision.

4 Security under weak tranquility

Our strong tranquility assumption in Section 3.2 is restrictive since the users, roles, and permissions, which define the protection state in any organization will change over time. Once the protection state changes, it may not be possible to enforce some of the unlinkability requirements. New conflicts may emerge that may invalidate existing policy constraints.

In this section, we extend our results to model the effect of changing the protection state. Our proposed solution uses versioning to localize the impact of these updates. Since our policy enforcement mechanisms are decentralized, i.e., records belonging to a particular flow in a database are tagged with access restrictions, it is important to guarantee the security of these access restrictions under evolving protection state without requiring updates to deployed policies. Similar to maintaining consistent access histories for CW policies, we consider updating policies throughout the system to be infeasible.

We define the notion of weak tranquility which captures the effect of changing permissions on the satisfaction of unlinkability properties.

Definition 8. Weak Tranquility for user u **with respect to policy constraint** $\mathcal{P}_S$ *states that the access permissions (i.e., the URA and the PRA) associated with a user* u *of the system do not change in such a way that it violates the security and precision of the enforcement of* $\mathcal{P}_S$ *for user* u.

Our goal is to guarantee that changes to the protection state can preserve the weak tranquility property for as many users as possible during the lifetime of the system and identify such users for each policy constraint in the system.

When a policy is agreed upon by the user and the PNS, the policy constraints certificate is stamped with the current *system version number* maintained by the PNS. When users are added to the system, they are also assigned the current system version number. The user's version number will be updated when certain changes are made to the protection state. A user u can access an audit record belonging to flow I only if $Version(u) \leq Version(I)$, which implies that weak tranquility holds for u with respect to the policy constraint for I. We assume that reference monitors have access to the current version number for a user (e.g., policy database or a revocation-based certificate approach). We prove Lemma 1 based on the following update rules for a user's version number.

Lemma 1. *Consider audit flows* $I_1, \ldots, I_n$ *in a session* S. *After any change to URA or PRA, if for a user* u, $Version(u) \leq Version(I_i)$ *for all* $i = 1, \ldots, n$, *then weak tranquility holds for user* u *with respect to the policy constraints* $\mathcal{P}_S$.

Proof. We prove this for each possible update to the protection state, and hence the lemma holds by induction on the number of updates to the protection state. For the base case, there are no updates to the protection state, and the lemma trivially holds by strong tranquility, which implies weak tranquility.

New User u **Created:** No change to system version number. Assign current system version number to user u. u has not been granted any static read access and weak tranquility trivially holds for u with respect to $\mathcal{P}_S$. Weak tranquility for other users with respect to $\mathcal{P}_S$ is not affected by this change.

New Role r **Added:** No change to system version number. No permissions have changed in the system, and weak tranquility holds for all users with respect to $\mathcal{P}_S$.

User-Role (u, r) **Assignment Added:** When a User-Role assignment (u, r) is added, it is possible that u now has static read access to two or more flows in session S, but will not be denied access

Eighth IEEE International Workshop on
Policies for Distributed Systems and Networks (POLICY'07)
0-7695-2767-1/07 $20.00 © 2007 IEEE

to two or more flows by the policy constraints. To maintain the security property of $\mathcal{P}_S$ with respect to u, the system version number is incremented, and u is assigned the new version number. Hence existing policies $\mathcal{P}_S$ will deny access to u based on u's version number. Weak tranquility for other users with respect to $\mathcal{P}_S$ is not affected by this change.

User-Role Assignment (u, r) **Deleted:** No change in version number. We only need to examine the case when u had static read access to two or more flows in S before the user-role assignment was deleted. If u continues to have static read access to two or more flows in S, then u must activate roles other than r, which must appear in the original policy constraints. Hence u will be prevented access by the policy constraints if u has a role in the deny list of the constraints (security property). If u does not have any roles on the deny list (see discussion for *privilege escalation* for the case when $r \in C_{Alice}$), then u is allowed access. If it is the case that u no longer has static read access to two or more audit flows, then r was necessary for access to two or more flows. Hence $r \in URA(u)$ is a necessary condition for being denied access by the policy constraints. Since now $r \notin URA(u)$, the policy constraints will allow u to access flows in S (precision). Weak tranquility for other users with respect to $\mathcal{P}_S$ is not affected by this change.

User u **Deleted:** Version number does not change. Equivalent to iteratively removing all User-Role assignments for u. Delete all the User-Role assignments.

Role r **Deleted:** Equivalent to iteratively removing all User-Role assignments for r followed by removing all $PRA(r)$. Note that after this operation, the system version number remains unchanged.

Permission-Role (d, r) **Assignment Added:** This means that a role r has been granted static read access to some database d. Since this role may not have been included in the session graph, it is possible that some users in r can now access two or more audit flows, and will not be denied access by the policy constraints, violating the security of the policy constraints, and weak tranquility does not hold for users in r with respect to existing policy constraints $\mathcal{P}_S$. If there are any users assigned to role r, the system version number is incremented, and all users in r are assigned the new version number. Weak tranquility for other users with respect to $\mathcal{P}_S$ is not affected by this change.

Permission-Role (d, r) **Assignment Deleted:** This means that the static read access to database d has been removed for a role r. It is possible that users in r are no longer a threat to linkability, but will still be denied access by policy constraints, violating the precision of the policy constraints. Hence weak tranquility does not hold for users in r. If there are any users assigned role r, the system version number is incremented, and all users in r are assigned the new version number. Weak tranquility for other users with respect to $\mathcal{P}_S$ is not affected by this change. Note that the security of policy constraints is not affected by adding the assignment (d, r). For every policy, however, we would like to maintain the set of users for which weak tranquility holds, which is why we update the version numbers for affected users.

Privilege Escalation: Consider the situation when a user has access to only one flow in a session. After accessing this information, the user is removed from a particular role, and then added to a new role, giving the user access to another flow in the session, violating the unlinkability requirement. The version number of the user, however, is incremented when a new user-role assignment is added, which will prevent this kind of privilege escalation. Similarly, incrementing the version number on the addition of a new permission-role assignment prevents privilege escalation due to changing permission-role changes. More generally, privilege escalation is prevented by the fact that a user's version number is incremented whenever the user's static permission set increases. It is important to note that if a role r is removed from a user's role-set, it is possible that r is on the deny list of some policy constraint, and that the user will now be able to link flows in that session, which was disallowed before this removal. With cooperation from the security officer, a user can remove, and subsequently add, r to his/her role-set resulting in one form of privilege escalation. We assume that the security officer is trusted, and that privilege escalation from the removal of a conflicting role is semantically correct and secure. An alternative approach would be to define this type of privilege escalation as not secure, and increment the version number when a user-role assignment is removed.

□

Under versioning, the following theorems follow easily from Lemma 1. Please refer to our technical report for the proofs [5].

Theorem 6. (Secure) *If a user u with a role in Alice's deny-set C_{Alice}, has static read access to two or more audit flows in Alice's session $I_1, \cdots, I_{n+1}$,*

then the policy constraints will prevent u *from accessing* **two or more** *of these flows.*

Theorem 7. (Precise up to Versioning) *If a user u has static read access to exactly one audit flow within a session* $S = \{I_1, \ldots, I_n\}$*, then u is not denied access by the policy constraints if* $Version(u) \leq Version(I_i)$ *for all* $i = 1, \ldots, n$.

After the policy constraints have been generated, previously deployed policy constraints gradually lose precision by being overly restrictive to users affected by evolving system permissions. This imprecision, however, is restricted only to users who gain new permissions, and users of roles for which database permissions change. We argue that the latter case is rare and can be performed at predefined system epochs. To cope with degrading precision, the PNS can choose to honor the policy constraints for a certain time period called *unlinkability window*. This window can either be a static parameter in the system, or can be negotiated with the user. As mentioned earlier, changes in flow policies are considered to be non-trivial changes. These changes can take place in epochs that honor the unlinkability window. When this is not possible, all data along the new flow is tagged as sensitive and is only allowed access by designated administrators. Users can be informed in general that changes in flow policy are possible, and that certain designated administrators will have access to audit flows in the session.

5 Discussion

Guaranteeing the unlinkability of a user's accesses is a hard problem in general because of other channels of observation outside the scope of anonymizing protocols or an access control system. Motivated adversaries can physically observe a user accessing a printer, room, etc. We have made an initial attempt at characterizing the semantics of unlinkability in a distributed setting as a confidentiality property. We assume that users will take the necessary physical safeguards for their privacy and our model provides the user with the specific guarantee that two or more records within the user's session will not be accessible by certain individuals, preventing the linkability of the contents within the records. Our model does not address context-sensitive authorizations. A more sophisticated model could allow users to specify circumstances in which their records can be linked (for example, in the case of a medical emergency).

In our model, the user specifies a deny set based on the conflicting roles identified by the PNS. Depending on the system, the set of conflicting roles could be quite large. To simplify the process of specifying privacy constraints, therefore, a usable interface would be needed in practice. Perhaps the system, or its users, can maintain lists of untrusted roles used in previous deny sets and thereby reduce the number of decisions that need to be made by users.

Our enforcement architecture uses versioning to maintain the security of policies, but these policies lose their precision as protection state evolves. Our solution does not claim to achieve optimality. Better enforcement mechanisms (e.g., a more sophisticated versioning scheme) may yield better precision under evolving protection state. Ideally, distributed reference monitors would identify exactly whether a user violates the security and precision of a policy and deny access accordingly. Such approaches, however, may require a more sophisticated versioning scheme, which could increase the complexity of the system. Our solution takes the conservative approach of identifying users for which the system cannot guarantee security and precision. Further study is required to measure the rate of degradation of precision in realistic systems.

6 Related Work

Research on the privacy of a user's accesses has focused on cryptographic mechanisms for anonymous authorization. To provide unlinkability, researchers have explored the construction of credential systems that satisfy the *multi-show* property whereby the owner of a certificate can construct two or more credentials with the same attributes that are unlinkable[12, 9]. *Anonymous credentials* presented by Chaum [3] relies on the interaction with a trusted third party for unlinkability. Camenisch, Lysyanskaya et al. [2, 7] extend this unlinkability based on computational zero-knowledge proofs, and the credential system proposed in [9] defines "Chameleon certificates" that provide a user complete control over the amount of information revealed as well as computational zero-knowledge proofs for unlinkability of credentials. In contrast, our research focuses on enforcing unlinkability policies using access control mechanisms. As mentioned earlier, in some systems it may not be feasible to allow anonymous access to resources.

The Separation of Duty (SoD) problem is traditionally viewed as preventing a single user from

Eighth IEEE International Workshop on
Policies for Distributed Systems and Networks (POLICY'07)
0-7695-2767-1/07 $20.00 © 2007 IEEE

performing different actions on the *same* object in the course of a workflow to protect transactional integrity [11]. In contrast, we prevent a user from accessing *different* audit records associated with different information flows initiated by a single user. Sandhu's work on Transaction Control Expressions (TCE [10]) shows how dynamic SoD constraints can be enforced adequately using history if the information about each transaction is annotated with the object itself. Simon and Zurko [11] argue that such history is essential to enforce general SoD constraints, violating our requirement of not maintaining an access history. Gligor et al. [4] formalize the relationship between SoD and RBAC and show how RBAC is not sufficient to enforce all types of SoD properties, especially dynamic SoD constraints. More recently, Li et al. [6] show how directly enforcing static SoD policies is intractable, let alone dynamic SoD policies, and show how statically mutually exclusive roles can be engineered to enforce these constraints on a best-effort basis. In the context of our unlinkability problem, annotating audit records in different databases with history information does not provide us with a mechanism to enforce unlinkability as these data objects are independent and local history cannot be used to enforce global constraints. Instead, our proposed solution annotates different audit records with policy constraints to enforce unlinkability. Furthermore, we show how unlinkability constraints can be computed in polynomial time, and show that it is tractable to enforce such policies under relaxed semantics.

7 Conclusions

We explore the problem of unlinkability in the context of administrators accessing a user's audit records in a distributed environment. Since Chinese Wall policies are difficult to enforce in a distributed environment, we propose new semantics that allow for the enforcement of unlinkability policies without having to maintain any access history. We present an enforcement architecture for our access control model and show how audit flows for different access transactions can be analyzed to generate policy constraints for unlinkability. With appropriate tranquility assumptions on the underlying authorizations, we prove that these constraints can guarantee unlinkability when enforced locally. To maintain the security of deployed policy constraints under evolving protection state, we propose a solution based on versioning that maintains security by trading precision for evolving protection state.

Acknowledgments

We thank Marianne Winslett and the anonymous reviewers for their helpful comments.

References

[1] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy*. MIT Press, 2000.

[2] J. Camenisch and A. Lysyanskaya. An efficient non-transferable anonymous multishow credential system with optional anonymity revocation. In *EUROCRYPT*, 2001.

[3] D. Chaum and J.-H. Evertse. A secure privacy preserving protocol for transmitting personal information between organizations. In *CRYPTO*, 1986.

[4] V. D. Gligor, S. I. Gavrila, and D. F. Ferraiolo. On the formal definition of seperation-of-duty policies and their composition. In *In Proceedings of the IEEE Symposium on Research in Security and Privacy. (Oakland, CA.), 172–183*, 1998.

[5] A. Kapadia, P. Naldurg, and R. H. Campbell. Distributed Enforcement of Unlinkability Policies: Looking Beyond the Chinese Wall. *Technical Report, University of Illinois, UIUCDCS-R-2006-2689*, 2006.

[6] N. Li, Z. Bizri, and M. V. Tripunitara. On Mutually-Exclusive Roles and Separation of Duty. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS), October*, 2004.

[7] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas of Cryptography, Volume 1758 LNCS*, 1999.

[8] N. H. Minsky. A Decentralized Treatment of a Highly Distributed Chinese-Wall Policy. In *Proceedings IEEE 5th International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, pages 181–184, June 2004.

[9] P. Persiano and I. Visconti. An Anonymous Credential System and a Privacy-Aware PKI. In *R. Safavi-Naini and J. Seberry, editors, Information Security and Privacy, 8th Australasian Conference, ACISP 2003, volume 2727 of Lecture Notes in Computer Science. Springer Verlag*, 2003.

[10] R. Sandhu. Transaction control expressions for separation of duties. In *Proceedings of the 4th Aerospace Computer Security Applications Conference*, 1998.

[11] R. T. Simon and M. E. Zurko. Separation of duty in role-based environments. In *IEEE Computer Security Foundations Workshop*, pages 183–194, 1997.

[12] E. R. Verheul. Self-Blindable Credential Certificates from the Weil Pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 533–551. Springer-Verlag, 2001.