Preventing Sybil Attacks by Privilege Attenuation: A Design Principle for Social Network Systems

UC CPSC Technical Report 2010-984-33

Philip W. L. Fong
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
pwlfong@ucalgary.ca

December 1, 2010

Abstract

In Facebook-style Social Network Systems (FSNSs), which are a generalization of the access control model of Facebook, an access control policy specifies a graph-theoretic relationship between the resource owner and resource accessor that must hold in the social graph in order for access to be granted. Pseudonymous identities may collude to alter the topology of the social graph and gain access that would otherwise be forbidden. We formalize Denning's Principle of Privilege Attenuation (POPA) as a run-time property, and demonstrate that it is a necessary and sufficient condition for preventing the above form of Sybil attacks. A static policy analysis is then devised for verifying that an FSNS is POPA compliant (and thus Sybil free). The static analysis is proven to be both sound and complete. We also extend our analysis to cover a peculiar feature of FSNS, namely, what Fong *et al.* dubbed as Stage-I Authorization. We discuss the anomalies resulted from this extension, and point out the need to redesign Stage-I Authorization to support a rational POPA-compliance analysis.

Keywords: Access control, policy analysis, social network systems, Sybil attacks, Principle of Privilege Attenuation, soundness and completeness of static analysis.

1 Introduction

One way of gauging the effectiveness of an access control scheme is by carefully articulating the security property that is enforced by the scheme. For example, by showing that a protection mechanism has the noninterference property [21, 31], one knows that the high-level information flow policy is indeed enforced by the mechanism. When the target security

property is ambiguous, the access control mechanism adds guards and mediations with no clear security goal.

The rise of social computing introduces the world to a new paradigm of access control, in which the inter-personal relationships between users are explicitly articulated in a social graph that forms the basis of authorization decisions. Authors have begun to refer to this emerging paradigm as Relationship-Based Access Control (ReBAC) [20, 6, 18]. For instance, Fong *et al.* [19, 1] generalized the access control mechanism of the popular Social Network System (SNS), Facebook, into a family of access control systems known as Facebook-style Social Network Systems (FSNSs). A distinctive feature of FSNSs is that every access control policy specifies a graph-theoretic relationship between the resource owner and the resource accessor (e.g., the owner and accessor share k or more common friends). Access is granted when the stated relationship is realized in the social graph. A question one must ask is the following: Exactly what is the security goal of such a protection paradigm? That is, what security property is enforced by this protection paradigm? This work is an attempt to answer the questions above.

We demonstrate that some FSNSs, when improperly configured, are amenable to ***Sybil attacks*** [17]. In a classical Sybil attack, a malicious user of a peer-to-peer system creates multiple pseudonymous identities, and uses their combined influence to bypass the reputation system. In an FSNS, because authorization decision is a function of the current topology of the social graph, it is possible that a group of pseudonymous identities may collude to manipulate the topology of the social graph, with the effect of gaining privileges that would otherwise be forbidden. (In this work, we concern ourselves with "formal" Sybil attacks, in which malicious users add or remove edges in the social graph without employing social engineering. Sybil attacks involving social engineering, such as Profile Cloning attacks [2], are beyond the scope of this work.) The prevention of Sybil attacks is taken to be a security goal of FSNSs in this work.

The articulation of the above security goal begs the question of whether Sybil attacks can be prevented by a properly designed FSNS. In early work on access control systems, Denning formulated a design principle, the ***Principle of Privilege Attenuation (POPA)***, to prevent privilege escalation caused by the collusion of unprivileged users. In short, POPA demands that a user may confer a privilege only when she already has that privilege. It turns out that POPA can be adopted in the design of FSNSs to effectively prevent Sybil attacks. Specifically, this work is the first one to propose a policy analysis that will allow an FSNS designer to statically certify that the FSNS complies to POPA, and thus provably prevent Sybil attacks at run time. Specifically, the following are the contributions of this work:

- We formalize POPA as a property of the set of execution traces generated by an FSNS, in a manner akin to non-interference [21, 31]. We show that this formulation of POPA is a necessary and sufficient condition for preventing Sybil attacks in FSNSs.

- We define a static analysis that allows one to examine only the current configuration (i.e., privacy settings) of the FSNS, and determine if the FSNS complies to POPA. We prove that the static analysis is sound: if the current configuration of the FSNS passes the static analysis, then Sybil attacks will not occur at run time. This result can be further exploited so that an FSNS designer can statically analyze the vocabulary of

policies supported by the FSNS, and certify the FSNS to be POPA compliant under every possible configuration.

- We show that the above static analysis is complete: if the configuration of the FSNS complies to POPA, then the static analysis will recognize it as such. This means the analysis is precise.

- We extend the static analysis to cover a peculiar feature of FSNS. In an FSNS, a necessary condition for access is that the accessor must "reach" the "search listing" of the profile owner (see Section 2.1 for more details). We illustrate the complexities and anomalies that will arise from widening the scope of the policy analysis to cover this feature. This motivates a need to redesign the access control mechanism of FSNSs to facilitate POPA-compliance analysis.

2 Overview

The kind of Sybil attacks studied in this work is distinct from its counterparts in peer-to-peer systems [39, 38], reputation systems [13] and recommendation systems [40]. Sybil attack prevention in FSNSs is a novel research challenge that has not been studied in the past. This section gives an overview of this research problem and our solution approach.

2.1 Facebook-style Social Network Systems

We review here an analysis of Facebook's access control mechanism that was reported by Fong *et al.* [19]. We then outline a family of SNSs that they identified to be Facebook-style Social Network Systems.

2.1.1 Profile and Profile Items

Facebook allows each user to construct a representation of herself in the form of a ***profile***. A profile displays such ***profile items*** as personal information (e.g., favorite books), multimedia contents (e.g., pictures), activity logs (e.g., status), or other user-authored contents (e.g., blog-like postings). Facebook users may grant one another access to the profile items they own.

2.1.2 Search Listings and their Reachability

Access to profile items is authorized in two stages. In ***Stage I***, the accessing user must ***reach*** the ***search listing*** of the profile owner. Then in ***Stage II***, the accessing user requests access to the profile, and the profile items are selectively displayed. The search listing of a user could be seen as a "capability" [16, 27] of the user in the system, through which access is mediated. There are two means by which a search listing may be reached in Stage I — ***global name search*** and ***social graph traversal***.

2.1.3 Global Name Search

A successful global name search produces for the accessing user the search listing of the target user. A profile owner may specify a ***search policy*** to permit only a subset of users to reach her search listing through a global name search.

2.1.4 Social Graph Traversal

A second means to reach a search listing is by traversing the ***social graph***. Facebook allows users to articulate their relationships with one another through the construction of ***friend lists***. Every user may list a set of other users as her ***friends***. As friendship is an irreflexive, symmetric binary relation, it induces a simple graph known as the social graph, in which users are vertices and relationships are edges. A user may traverse this graph by examining the friend lists of other users. A user u may restrict traversal by specifying a ***traversal policy***, which specifies the set of users v who are allowed to examine u's friend list after v has reached the search listing of u.

2.1.5 Profile Access

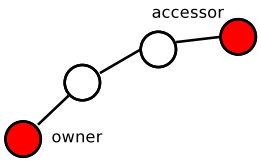
Once the search listing of a profile owner is reached, the accessing user may elect to access the profile, thereby initiating Stage II of authorization. Not every accessing user sees the same profile items when a profile is displayed. The owner may assign an ***access policy*** to each profile item, dictating who can see that profile item when the profile is accessed. This is the means through which a user may project different representations of herself to different groups of users.

2.1.6 Friendship Articulation

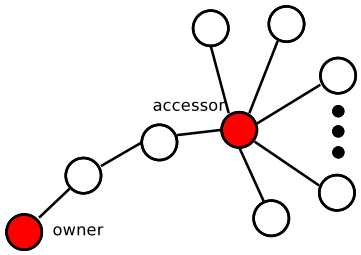
Articulating friendship involves a consent protocol, whereby a user sends a friendship invitation to another user, who may then accept or ignore the invitation. Once a mutual consent is reached, that friendship is recorded by Facebook as an edge in the social graph. Such an operation is called ***befriending***. Befriending invitation is guarded in Facebook. To initiate a befriending invitation, the inviting user must first reach the search listing of the invited user (i.e., Stage I). Then, the ***acquaintance policy*** of the invited user will determine if the inviting user can extend the befriending invitation. On the other hand, ***defriending***, that is, the dissolution of friendship, is not guarded. It can be carried out unilaterally by either end of the relationship.

2.1.7 Policies

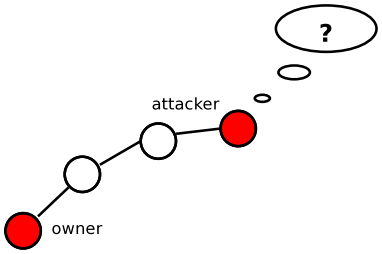
We have seen in the above discussion that various aspects of user activities are controlled by user-specified policies (e.g., search policy, access policy, etc). This is typical of a discretionary access control system [22, 26], in which a user may grant access privileges to other users. Facebook offers a fixed vocabulary of predefined policies for users to choose from when they are to identify sets of privileged users. These predefined policies identify user sets not in terms of user identities, but rather in terms of how the accessor and the owner are related



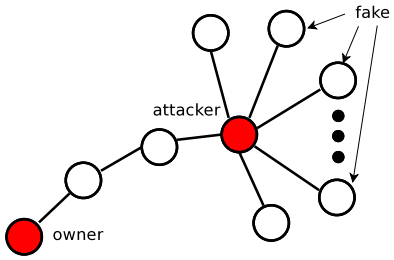
(a) Access policy: "somewhat related".



(b) Access policy: "somewhat related" and "popular".



(c) An attacker who is "somewhat related" but not "popular".



(d) Attacker befriends fake accounts to gain popularity.

Figure 1: A Sybil attack example.

to one another in the social graph. For example, one may specify that a certain profile item is accessible only by "friends" of the profile owner, or that befriending invitation is only available to "friends of friends".

2.1.8 Facebook-style Social Network Systems

Fong *et al.* [19] proposed a generalization of the above access control mechanism, whereby the policy vocabulary may contain policies that specify complex graph-theoretic relationships between the owner and the accessor. For example, the policy vocabulary of such an SNS may contain a policy that grants access when the owner and the accessor share k common friends (i.e., "friends of friends" is a special case with $k = 1$). As another example, consider a policy that grants access when the owner and the accessor belong to the same k-clique (i.e., "friends" is a special case when $k = 2$). A ***Facebook-style Social Network System (FSNS)*** is an SNS that shares with Facebook the same access control mechanism as described above, but with its own vocabulary of policies. The family of FSNSs effectively specifies a design space for Facebook-like SNSs, so that the design space is parameterized by the choice of policy vocabulary, where Facebook is but one point of the design space. The goal of this work is to identify a design criterion for selecting a "good" policy vocabulary.

2.2 Sybil Attacks in FSNSs

The normal assumption of a SNS is that every user is represented by exactly one user account (i.e., one identity), and thus each vertex of the social graph is controlled by a distinct user.

Typically this assumption is enforced by imposing terms of use, by CAPTCHA, and by cross-referencing personal data collected from users. There is, however, no "hard" mechanism for strictly enforcing such a policy. It is entirely possible for a user to assume multiple identities [2]. The result is that the same human user could offer befriending or defriending consent on behalf of multiple social graph vertices.

In FSNSs, accessibility is a function of the topology of the social graph. By taking control of multiple user accounts, and strategically initiating befriending and defriending actions, a user can manipulate the topology of the social graph neighborhood that is dominated by the vertices she controls. A Sybil attack [17] is resulted when a malicious user, through setting up multiple accounts, coordinates the addition and deletion of edges in the social graph, in such a manner that allows her to gain privileges that otherwise she would not possess.

Example 1. *Suppose an owner allows access to her dating-ready photo only if the accessor is "*somewhat related*", and is a "*popular*" person. What is an access policy that appropriately represents this privacy need? To ensure the accessor is "somewhat related", let us demand the accessor to be of distance no further than three from the owner (Figure 1a). To ensure the accessor is "popular", let us further demand the accessor to have a vertex degree of one hundred or more (Figure 1b).*

Now suppose there is an attacker who is "somewhat related" (distance no more than three), but not "popular" enough (degree less than a hundred). How can the attacker launch a Sybil attack to gain access to the owner's photo (Figure 1c)? All she needs to do is to create enough fake accounts, and befriend them accordingly (Figure 1d). As she controls both her original account and these fake accounts, she could offer consent to any befriending requests.

Notice that the attack above can be launched in a slightly different manner. Suppose the attacker does not create fake accounts, but instead colludes with an appropriate number of strangers by befriending them. She will then be able to increase her vertex degree, and thus gain access as a result. This observation leads to a core insight we would like to highlight:

> *In FSNSs, there is no material difference between Sybil attacks and the collusion of unprivileged users to gain access.*

Creation of pseudonymous identities is *not* the crux of the problem. The crux of the problem is whether an SNS can be configured to prevent unprivileged users from manipulating the access control system into conferring to them privileges that they do not rightly possess, an anomaly traditionally known as the ***Confused Deputy*** [23].

2.3 Principle of Privilege Attenuation

In early work on access control systems, Denning formulated a design principle to avoid exactly this kind of problems. Specifically, to prevent unprivileged users from colluding with one another to gain access, Denning advocates the ***Principle of Privilege Attenuation (POPA)*** [15, page 372]. A modern paraphrase of POPA by Bishop [3, page 43] is the following:

> *A subject may not give rights it does not possess to another.*

In other words, the granter of a right must possess the right.

In FSNSs, there is no explicit granting of rights. Authorization is the consequence of the social graph possessing a particular topology. Granting and revocation of access occur implicitly when befriending and defriending actions alter the topology of the social graph. We adapt POPA to the context of FSNSs, and devise below a formulation of POPA that prevents Sybil attacks:

> *Every action (e.g., befriending or defriending) that could cause an unprivileged subject to eventually gain access must be initiated by (i.e., have the consent of) a subject who already have access.*

To see how POPA prevents Sybil attacks, note that both the attacker's original account and the new accounts she creates lack the privilege to access the target profile item. Actions initiated solely within this group of user accounts, according to POPA, could never cause any of them to eventually gain access. To gain access, they need the consent of users who already possess the required access. (This informal argument connecting POPA to Sybil attack prevention will be made formal in Section 4.3.)

2.4 Our Approach

Promising as it is, the above statement of POPA leads to two intellectual challenges. First, what precisely do we mean when we say "an action *causes* a subject to eventually gain access." How does one formalize such a notion of causality, which is foundational to POPA itself? Second, how does one design or configure a FSNS to ensure it realizes POPA? To the very least, is there a way for us to validate that an FSNS is POPA compliant? The main contributions of this work are concrete means to address these two challenges. We will capture causality by a noninterference-style formulation of POPA, and devise a static analysis to check if a given FSNS design or configuration is POPA compliant. Specifically, our plan is the following:

1. We will formalize POPA as a noninterference-style property [21, 31] of the set of execution traces of a given FSNS (i.e., a run-time property).

2. We will identify a structural property that can be imposed on the policy vocabulary of the FSNS to guarantee POPA (i.e., a static property).

3. We will prove a soundness result: If the policy vocabulary satisfies the static property in Step 2, then the set of execution traces generated by the FSNS will satisfy the run-time property of Step 1.

3 Modelling FSNSs

This work is built on a formal model of FSNSs and a theory of FSNS policies that were previously proposed in [19, 1]. In the following, we review, adapt and extend these results.

3.1 Basic Notations

We write $\mathbb{B}$ to denote the set $\{0, 1\}$ of Boolean values. Given a set S, 2^S is the power set of S, $[S]^k$ is the set of all size-k subsets of S, and, when S is finite and non-empty, $\mathcal{G}(S)$ is the set of all simple graphs with S as the vertex set (i.e., $\mathcal{G}(S) = \{\, \langle S, E\rangle \mid E \subseteq [S]^2 \,\}$). The (open) neighbourhood $N_G(u)$ of vertex u is the set of vertices adjacent to u (i.e., $\{v \in V(G) \mid \{u, v\} \in E(G)\}$). Given $e \in [V(G)]^2$, we write $G + e$ to denote the graph $\langle V(G), E(G) \cup \{e\}\rangle$. We write $G_1 \subseteq G_2$ whenever graph G_1 is a subgraph of graph G_2 (i.e., $V(G_1) \subseteq V(G_2)$ and $E(G_1) \subseteq E(G_2)$). We also write $G_1 \cup G_2$ to denote the graph $\langle V(G_1) \cup V(G_2), E(G_1) \cup V(G_2)\rangle$. Fixing a vertex set V, we write $G^{\emptyset}$ for the empty graph $\langle V, \emptyset\rangle$.

If there is a path between vertices u and v, then we say that the two vertices are ***connected***. Otherwise, they are ***disconnected***. A graph is ***connected*** iff every pair of vertices are connected. Otherwise the graph is ***disconnected***. A ***component*** of a graph is a maximal connected subgraph. A vertex v is a ***cut vertex*** in graph G if and only if the component to which v belongs will become disconnected when v is removed from G. A vertex set V is a ***vertex cover*** of an edge set E iff $V \cap e \neq \emptyset$ for every $e \in E$.

3.2 A Model of FSNSs

3.2.1 Policies

Fixing a set Sub of users, we model a policy as a predicate (i.e., a Boolean function) with the signature $Sub \times Sub \times \mathcal{G}(Sub) \to \mathbb{B}$. Given a profile owner $u \in Sub$, a profile accessor $v \in Sub$, and the current social graph $G \in \mathcal{G}(Sub)$, a policy predicate returns a Boolean value indicating if access should be authorized.

Example 2. *The policy predicates dist_k, cf_k and clique_k are defined as follows: $\mathsf{dist}_k(u, v, G)$ holds whenever there is a path of length k or less between u and v in G; $\mathsf{cf}_k(u, v, G)$ holds iff either $\mathsf{dist}_1(u, v, G)$ or else u and v share at least k distinct common friends in G; $\mathsf{clique}_k(u, v, G)$ holds iff either $u = v$ or else u and v belong to a common clique of size k in G. We write me, friend and fof as shorthands for dist_0, dist_1 and dist_2 respectively. We also write $\top$ and $\bot$ to denote the constant predicates that always return true and false respectively.*

We write $\mathcal{PP}(Sub)$ to denote the set of policy predicates sharing the signature $Sub \times Sub \times \mathcal{G}(Sub) \to \mathbb{B}$.

3.2.2 Systems

An FSNS (or simply a ***system***) N is a triple $\langle Sub, PV, Pol\rangle$. The first component, Sub, is a finite set of users[1]. The second component, PV, called the ***policy vocabulary***, specifies the set of legitimate policy predicates supported by the FSNS. Let $\mathcal{PT}$ be the set $\{\mathsf{sch}, \mathsf{tra}, \mathsf{acc}\}$ of identifiers denoting the three types of policies (i.e., search, traversal and access). PV :

[1]Rather than modelling a system with an evolving number of users and thus complicating the model unnecessarily, we fix the number of users in a system. User addition could be seen as certain dormant users becoming active. User removal can be modelled in the opposite way.

$\mathcal{PT} \to 2^{\mathcal{PP}(Sub)}$ is a family of policy sets indexed by the three policy types. Specifically, $PV(\mathsf{acc})$ is the set of policy predicates that a user can legitimately use as an access policy, and the interpretation of $PV(\mathsf{sch})$ and $PV(\mathsf{tra})$ is similar. Lastly, Pol, representing the ***privacy settings*** of the users, is a function with signature $Sub \times \mathcal{PT} \to \mathcal{PP}(Sub)$, with the requirement that $Pol(u,t) \in PV(t)$. That is, $Pol(u,\mathsf{sch})$, $Pol(u,\mathsf{tra})$ and $Pol(u,\mathsf{acc})$ are respectively the search, traversal and access policy of user u. In this work, we no not consider acquaintance policies.

3.2.3 States

A state of an FSNS is simply a social graph. We write $\mathcal{S}_N$ to denote the set $\mathcal{G}(Sub)$ of all states of N. In this work, we do not consider the management aspect of an FSNS, and thus we do not model user-initiated change of policies.

3.2.4 Queries

We model access as queries over states. Specifically, we write the sequent $G \vdash_N q$ whenever the social graph G satisfies the query q. There are two types of query:

$$q ::= v \ \mathsf{reads}\ u \mid v \ \mathsf{finds}\ u$$

The sequent "$G \vdash_N v \ \mathsf{finds}\ u$" holds whenever accessor v can reach the search listing of owner u in state G of system N. The sequent "$G \vdash_N v \ \mathsf{reads}\ u$" holds whenever accessor v is authorized to access the profile of owner u in state G. Formally, the sequents are defined as follows:

$$\frac{N = \langle \_, \_, Pol \rangle \qquad Pol(u,\mathsf{sch})(u,v,G)}{G \vdash_N v \ \mathsf{finds}\ u} \quad \text{(F-Sch)}$$

$$\frac{u = v}{G \vdash_N v \ \mathsf{finds}\ u} \quad \text{(F-Slf)}$$

$$\frac{\{u,v\} \in E(G)}{G \vdash_N v \ \mathsf{finds}\ u} \quad \text{(F-Frd)}$$

$$\frac{G \vdash_N v \ \mathsf{finds}\ u' \qquad \{u,u'\} \in E(G) \qquad N = \langle \_, \_, Pol \rangle \qquad Pol(u,\mathsf{tra})(u',v,G)}{G \vdash_N v \ \mathsf{finds}\ u} \quad \text{(F-Trv)}$$

$$\frac{G \vdash_N v \ \mathsf{finds}\ u \qquad N = \langle \_, \_, Pol \rangle \qquad Pol(u,\mathsf{acc})(u,v,G)}{G \vdash_N v \ \mathsf{reads}\ u} \quad \text{(R-Acc)}$$

In each of the queries "$v \ \mathsf{reads}\ u$" and "$v \ \mathsf{finds}\ u$", we call v the ***accessor*** of the query. Given query q, we write $q(v)$ to denote the query obtained from q by substituting v for the accessor of q. For example, if q is "$v \ \mathsf{reads}\ u$", then $q(v')$ is the query "$v' \ \mathsf{reads}\ u$". Accessor substitution constructs a query that exercises the same right as the original query, but now the right is exercised by a different user.

3.2.5 Transitions

There are only two types of transitions: befriending and defriending, identified respectively by ***transition identifiers*** of the forms e and $\overline{e}$, where e is the edge in the social graph affected by the transition. A befriending transition, identified by e, requires the consent of both parties involved. A defriending transition, identified by $\overline{e}$, can be initiated unilaterally by either end of the edge e. Let $\mathcal{T}_N$ be the set of all transition identifiers.

The transition relation $\cdot \xrightarrow{\cdot}_N \cdot : \mathcal{S}_N \times \mathcal{T}_N \times \mathcal{S}_N$ is defined as follows:

$$G \xrightarrow{e}_N G + e \qquad \text{if } e \notin E(G)$$
$$G \xrightarrow{\overline{e}}_N G - e \qquad \text{if } e \in E(G).$$

Notice that befriending is unmediated, meaning that everyone can befriend everyone else. It is no longer necessary to reach the search listing of the target user and seek authorization by her acquaintance policy in order to initiate befriending (see Section 2.1). We call this simplifying assumption ***Unmediated Befriending***.

3.2.6 Traces

A finite sequence from $(\mathcal{T}_N)^*$ is called a ***trace***. We generalize the transition relation to traces. Specifically, we write $G \xrightarrow{\epsilon}_N G$, and, given $\tau, \pi \in (\mathcal{T}_N)^*$, we write $G \xrightarrow{\tau \cdot \pi}_N G'$ whenever $G \xrightarrow{\tau}_N G''$ and $G'' \xrightarrow{\pi}_N G'$ for some social graph G''. A trace τ is ***feasible in*** G iff $G \xrightarrow{\tau}_N G'$ for some G'. This feasible trace τ ***establishes*** q ***in*** G whenever "$G' \vdash_N q$" holds but "$G \vdash_N q$" does not.

Discussion We have significantly simplified the FSNS model previously proposed in [19]. The simplifications, which are summarized below, have been instrumental in making the results in this work feasible.

- A policy predicate is parameterized only by the owner, the accessor and the present social graph. In previous work, we allow a limited form of history information to affect authorization decisions.
- All policies (i.e., traversal, search, and access) are fixed: i.e., the privacy settings of users remain constant at run-time. Our previous model captures the management aspect of the access control system, and thus the change of policies at run time.
- We use a single access policy to protect the entire profile, thereby removing the need to track a different access policy for each profile item. Extending the theory to handing multiple profile items is a straightforward exercise.
- The only user actions in the model are befriending and defriending. Our previous work permits a more complex consenting protocol. Furthermore, the two actions are atomic. Specifically, we model the multi-step protocol of obtaining consent as a single transition in our model.
- Befriending is unmediated: everyone can befriend everyone else.

3.3 A Theory of Policies

Recall that a policy predicate is a member of $\mathcal{PP}(Sub)$, the set of predicates with the signature $Sub \times Sub \times \mathcal{G}(Sub) \to \mathbb{B}$. In our previous work [1], we provided characterization of certain families of policy predicates. provide here a review of these results, which will be used in the sequel. We begin with the notion of birooted graphs, on which the characterization results are based.

Definition 3. *A **birooted graph** $G_{(u,v)}$ is a triple $\langle G, u, v\rangle$ such that G is a simple graph and $u, v \in V(G)$. The **roots** u and v need not be distinct. We write $\mathcal{B}(S) = \{G_{(u,v)} \mid G \in \mathcal{G}(S)\}$ to denote the set of all birooted graphs with vertex set S.*

*$G'_{(u,v)}$ is a **(birooted) subgraph** of $G_{(u,v)}$, written as $G'_{(u,v)} \subseteq G_{(u,v)}$, iff G' is a subgraph of G. (Note the matching roots.) We say that $G_{(u,v)}$ and $G'_{(u',v')}$ are **isomorphic** iff there exists a graph isomorphism $\phi : V(G) \to V(G')$ between G and G', such that $\phi(u) = u'$ and $\phi(v) = v'$. In this case we write $G_{(u,v)} \cong G'_{(u',v')}$. We also write $G'_{(u',v')} \lesssim G_{(u,v)}$ whenever there is a birooted graph $G''_{(u,v)}$ such that $G''_{(u,v)} \subseteq G_{(u,v)}$ and $G'_{(u',v')} \cong G''_{(u,v)}$.*

Other than the above cases, when we apply graph-theoretic languages to birooted graphs, we are essentially referring to the underlying simple graphs.

Not all policies are equal. Some exhibit characteristics that are security relevant.

Definition 4. *A policy P is **monotonic** iff $G_{(u,v)} \subseteq G'_{(u,v)}$ implies $P(u, v, G) \Rightarrow P(u, v, G')$.*

*Policy P is **topology-based** iff $G_{(u,v)} \cong G'_{(u',v')}$ implies $P(u, v, G) = P(u', v', G')$.*

*Policy P is **positive** iff it is both monotonic and topology-based.*

*Policy P is **local** iff $P(u, v, G) \neq P(u, v, G+e)$ implies vertices u, v and edge e all belong to the same component in G.*

With a monotonic policy, adding edges to the social graph never diminishes accessibility. Similarly, removing edges from the social graph never increases accessiblity. With a topology-based policy, topologically equivalent access scenarios, represented by isomorphic birooted graphs, produce identical authorization decisions. In short, the individual identity of the users involved are not considered in the authorization decision. Topology-based policies are therefore the birooted version of graph properties. A local policy captures a relationship between the owner and the accessor, rather than a property of the owner or a property of the accessor.

Example 5. *All policy predicates in Example 2 are monotonic, topology-based, positive and local.*

There are alternative ways to characterize monotonic and positive policies.

Definition 6. *Let $\mathcal{B} \in \mathcal{B}(Sub)$ be a set of birooted graphs. The policy **monotonically induced** by $\mathcal{B}$ is the predicate $P^M_{\mathcal{B}}$ for which*

$$P^M_{\mathcal{B}}(u, v, G) \text{ iff } \exists G'_{(u,v)} \in \mathcal{B} \,.\, G'_{(u,v)} \subseteq G_{(u,v)}$$

*The policy **positively induced** by $\mathcal{B}$ is the predicate $P^+_{\mathcal{B}}$ for which*

$$P^+_{\mathcal{B}}(u, v, G) \text{ iff } \exists G'_{(u,v)} \in \mathcal{B} \,.\, G'_{(u,v)} \lesssim G_{(u,v)}$$

The following theorem provides an alternative characterization to policies that are respectively monotonic and positive.

Theorem 7. *Every monotonic (resp. positive) policy predicate is monotonically (resp. positively) induced by a set of birooted graphs. The minimal set of birooted graphs to monotonically (resp. positively) induce a given predicate P is defined to be the set $\mathcal{B}$ for which there exists no proper subset of $\mathcal{B}$ that also monotonically (resp. positively) induces P. This minimal set does not contain a pair of distinct birooted graphs $G_{(u,v)}$ and $G'_{(u',v')}$ such that $G_{(u,v)} \subseteq G'_{(u',v')}$ (resp. $G_{(u,v)} \lesssim G'_{(u',v')}$). Such a minimal set always exists, and is unique (resp. unique up to birooted graph ismorphism). Moreover, the prediate is also local iff the minimal set contains only birooted graphs for which the ends of every edge are connected to each of the two roots.*

The theorem has two versions, one regarding monotonic policy predicates, the other regarding positive predicates. The positive version of the theorem has been previously proved in [1]. The proof of the monotonic version follows that of the positive version closely.

Example 8. *Let $Sub = \{u_0, u_1, \ldots, u_n\}$, where $n \geq 2$. Consider the sets $\mathcal{B}_{\mathsf{dist}_k}$, $\mathcal{B}_{\mathsf{cf}_k}$, $\mathcal{B}_\bot$ and $\mathcal{B}_\top$ to positively induce dist_k, cf_k, $\bot$ and $\top$ respectively (assuming $1 \leq k \leq n$).*

1. $\mathcal{B}_{\mathsf{dist}_k} = \{G^i{}_{(u_0,u_i)} \mid 0 \leq i \leq k\}$, *where the graph G^i contains exactly the edges that form the path $u_0u_1 \ldots u_i$.*
2. $\mathcal{B}_{\mathsf{cf}_k} = \{G^\emptyset{}_{(u_0,u_0)}, G^\flat{}_{(u_0,u_1)}, G^\sharp{}_{(u_0,u_{k+1})}\}$, *where* $E(G^\flat) = \{\{u_0, u_1\}\}$, $E(G^\sharp) = \{\{u_0, u_i\} \mid 1 \leq i \leq k\} \cup \{\{u_i, u_{k+1}\} \mid 1 \leq i \leq k\}$.
3. $\mathcal{B}_\bot = \emptyset$.
4. $\mathcal{B}_\top = \{G^\emptyset{}_{(u_0,u_0)}, G^\emptyset{}_{(u_0,u_1)}\}$

On top of the policy classes in Definition 4, which were originally introduced in [1], we add the following classes.

Definition 9. *A monotonic policy predicate P is said to be **proper** iff the minimal set to monotonically induce P does not contain a birooted graph $G_{(u,v)}$ for which $u \neq v$ and v is an isolated vertex. A proper predicate is **properly local** iff it is also local.*

Example 10. *The constant predicate $\top$ is the only policy in Example 2 that is not proper.*

Corollary 11. *The minimal set of birooted graphs to monotonically induce a properly local predicate contains a birooted graphs $G_{(u,v)}$ only if u, v and every edge of G belong to the same component of G.*

We define policy combinators that allow us to create complex policies from primitive ones (the combinators $\vee$ and $\wedge$ were defined in [19, 1]; the combinators $\circ$, $\cdot^{+1}$ and $\cdot(\!|\cdot|\!)$ are new).

Definition 12. *Given policy predicates P, P_1 and P_2, and vertex u_0, we define the composite predicates $P_1 \vee P_2$, $P_1 \wedge P_2$, $P_1 \circ P_2$, P^{+1} and $P(\!|u_0|\!)$ as follows:*

$$
\begin{aligned}
(P_1 \vee P_2)(u, v, G) &= P_1(u, v, G) \vee P_2(u, v, G) \\
(P_1 \wedge P_2)(u, v, G) &= P_1(u, v, G) \wedge P_2(u, v, G) \\
(P_1 \circ P_2)(u, v, G) &= (\exists u' \in Sub \,.\, P_1(u, u', G) \wedge P_2(u', v, G)) \\
P^{+1}(u, v, G) &= (u = v) \vee (\{u, v\} \in E(G)) \vee \left(\bigvee_{u' \in Sub} ((\{u, u'\} \in E(G)) \wedge P(u', v, G)) \right) \\
P(\!|u_0|\!)(u, v, G) &= (u = u_0) \wedge P(u, v, G)
\end{aligned}
$$

The following proposition is new.

Proposition 13. *Suppose policy predicates P, P_1 and P_2 are monotonically induced by $\mathcal{B}$, $\mathcal{B}_1$ and $\mathcal{B}_2$, and u_0 is a vertex. Then $P_1 \vee P_2$, $P_1 \wedge P_2$, $P_1 \circ P_2$, P^{+1} and $P(\!|u_0|\!)$ are monotonically induced by the following sets respectively:*

$$\mathcal{B}_1 \cup \mathcal{B}_2 \tag{1}$$

$$\{(G_1 \cup G_2)_{(u,v)} \mid G_{1(u,v)} \in \mathcal{B}_1 \wedge G_{2(u,v)} \in \mathcal{B}_2\} \tag{2}$$

$$\{(G_1 \cup G_2)_{(u,v)} \mid \exists u' \in Sub \,.\, G_{1(u,u')} \in \mathcal{B}_1 \wedge G_{2(u',v)} \in \mathcal{B}_2\} \tag{3}$$

$$\{G^{\emptyset}{}_{(u,u)} \mid u \in Sub\} \cup \{(G^{\emptyset} + \{u, v\})_{(u,v)} \mid \{u, v\} \in [Sub]^2\} \cup \{(G + \{u, u'\})_{(u,v)} \mid G_{(u',v)} \in \mathcal{B}\} \tag{4}$$

$$\{G_{(u_0,v)} \mid G_{(u_0,v)} \in \mathcal{B}\} \tag{5}$$

While the proof of the proposition is elementary, there are two important corollaries that we will rely on in the sequel. The first and obvious one is that monotonicity is preserved by all the combinators. The second corollary is not as direct. Suppose $\mathcal{B}$, $\mathcal{B}_1$ and $\mathcal{B}_2$ are not just any sets to monotonically induce P, P_1 and P_2, but the minimal ones to do so. Then, by the proposition above, the minimal sets to monotonically induce $P_1 \vee P_2$, $P_1 \wedge P_2$, $P_1 \circ P_2$, P^{+1} and $P(\!|u_0|\!)$ are respectively subsets of the five sets identified above. This, in turn, means that we know the "shape" of the members of the minimal sets to monotonically induce the six composite policies. For example, every member of the minimal set to monotonically induce $P_1 \wedge P_2$ must be of the form $(G_1 \cup G_2)_{(u,v)}$, where $G_{1(u,v)}$ and $G_{2(u,v)}$ are respectively members of $\mathcal{B}_1$ and $\mathcal{B}_2$.

4 POPA Formalized

We follow a two-step plan to devise a formal statement of POPA. First, we will define the notion of "reduction", which allows us to "blame" certain transitions in a trace as the causal enablers of access granting. Second, we will define the notion of "rationalization", which allows us to "explain", after the fact, how authority is conferred among the causal enablers identified above, ultimately leading to the granting of access. POPA is then expressed as a statement asserting that it is possible to explain authority conferral among the causal

enablers in such a way that only privileged users may confer privilege to unprivileged users. Lastly, we demonstrate that this definition of POPA eliminates Sybil attacks.

In this section and in the rest of this paper, we will make the following assumption.

Assumption 14 (Monotonicity)**.** *All policy predicates considered are monotonic.*

In particular, this implies that each of $PV(\mathsf{sch})$, $PV(\mathsf{tra})$ and $PV(\mathsf{acc})$ contains only monotonic policies. Proposition 13 guarantees that policies constructed out of the policy vocabulary using the policy combinators will remain monotonic.

4.1 Reduction

In a trace that establishes access, not every transition in the trace is causally relevant to the eventual granting of access. We want to be able to identify those that are relevant, and impose on them constraints regarding conferral of privileges. To this end, we define a notion of reduction in the style of noninterference.

Definition 15. *Suppose τ establishes q in G. A subsequence*[2] *π of τ is a (G, q)-**reduction** (or simply a **reduction**) of τ iff π also establishes q in G. Trace τ is **minimally** (G, q)-**reduced** (or simply **minimally reduced**) iff τ is the only (G, q)-reduction of itself. We call π a **minimal** (G, q)-**reduction** (or simply **minimal reduction**) of τ if π is a (G, q)-reduction of τ that is minimally (G, q)-reduced.*

By definition, reducibility is reflexive, transitive and anti-symmetric. That is, reduction imposes a partial ordering over the set of traces.

Intuitively, a reduction π is a "simplified" version of τ, with the transitions inessential for establishing q removed. A minimal reduction of τ is a reduction π that does not contain another reduction as a proper subsequence. That is, it contains the transitions that are essential for establishing q. These transitions are considered to have "caused" q to be established. Note that a feasible trace may have more than one minimal reductions. Under Assumption 14 (Monotonicity), the following observation can be made.

Proposition 16. *Every transition of a minimally reduced trace is a befriending transition (i.e., with a transition identifier e).*

In short, defriending is never necessary for establishing access when all policies are monotonic. By the observation above, a minimal reduction of length n has the form $e_0 e_1 \dots e_{n-1}$.

Under the assumption of Unmediated Befriending, every user can befriend every other user at will (so long as consent is granted by the other party). A consequence is the following observation, which can be proved easily by contradiction.

Proposition 17. *Every permutation of a minimally reduced trace is also feasible and minimally reduced.*

[2] E.g., *acb* is a subsequence of *abcba*. As special cases, every sequence is its own subsequence, and the empty sequence is a subsequence of every sequence.

4.2 Rationalization

We define what it means for a mimimally reduced trace to properly observe the requirement of POPA.

Definition 18. *Suppose $\tau = e_0 e_1 \ldots e_{n-1}$ is a minimally (G, q)-reduced trace for system N. Then τ is **admissible** whenever:*

1. *$G = G_0 \xrightarrow{e_0}_N G_1 \xrightarrow{e_1}_N \ldots\ G_{n-1} \xrightarrow{e_{n-1}}_N G_n = G''$ for some social graph $G'' \in \mathcal{S}_N$, and*

2. *there is a sequence of vertices v_0, v_1, ..., v_{n-1} such that $v_i \in e_i$ and $G_i \vdash_N q(v_i)$ for $1 \leq i < n$. Note that the sequence may contain repeated vertices.*

A trace π satisfying requirements 1 and 2 above establishes q in a principled manner. Since befriending transition e_i furthers the establishment of q, we interpret the addition of e_i as a privilege conferral event. In accordance with the intuition of POPA, such a privilege conferral event must be initiated by someone who already possesses the conferred privilege. As a befriending action e_i is jointly initiated by both ends of the edge, we therefore require that one of its ends (i.e., $v_i \in e_i$) to already possess the conferred privilege. That is, we require that, in the state G_i in which the privilege is conferred, the initiator above already possess the privilege: i.e., $G_i \vdash_N q(v_i)$. The sequence π therefore honors POPA in the way it incrementally establishes q.

According to Proposition 17, every permutation of a minimally reduced trace is also minimally reduced, and thus every such permutation also establishes the query in question. Some of these permutations may be admissible.

Definition 19. *Suppose τ is a minimally (G, q)-reduced trace. Then a permutation[3] π of τ is a **rationalization** of τ whenever π is admissible.*

If a minimally reduced trace has a rationalization, it means that we can explain, after the fact, by rearranging the transitions in the trace, how privilege could have been incrementally conferred in accordance to POPA. Not every minimally reduced trace has a rationalization.

Definition 20. *A system N is **POPA-compliant** iff, for every $G \in \mathcal{S}_N$, every query q of the form "v reads u", and every $\tau \in (\mathcal{T}_N)^*$ that establishes q in G, it is the case that every minimal (G, q)-reduction of τ has a rationalization.*

Note that POPA compliance is not a property of individual traces, but rather an assertion regarding the correlation of traces generated by the system (i.e., "if such a trace is feasible then such other traces are also feasible" [14]). It employs a noninterference-style condition [21, 31] to filter away inessential transitions in a given trace, and then assert the existence of an explanation of how individual transitions could have been arranged such that at least one initiator of each transition is authorized for access when the transition occurs.

[3]E.g., *bcad* is a permutation of *abcd*. As a special case, every sequence is a permutation of itself.

4.3 Sybil Attack Prevention

POPA compliance is a strong condition: it requires *every* conferral of privilege to be initiated by a user possessing the conferred privilege. An interesting question is whether such a strong requirement is actually necessary for preventing Sybil attacks. Intuitively, to prevent Sybil attacks, all we need is that a group of unprivileged users cannot collude to gain privilege (Section 2.2). That is, the establishment of privilege requires the cooperation of *at least one* privileged user. In the following, we demonstrate that the two conditions are in fact equivalent, and thus POPA compliance is not only sufficient but also necessary for preventing Sybil attacks.

The following scenario constitutes a Sybil attack in system $N = \langle Sub, PV, Pol \rangle$. Suppose q is a query of the form "v reads u", $G \in \mathcal{S}_N$ is a state in which $G \vdash_N q$ does not hold, and $\tau \in (\mathcal{T}_N)^*$ is a trace that establishes q in G. Under Assumption 14, defriending transitions in τ are not significant. We focus on the befriending transitions. A Sybil attack has occurred if, for every befriending transition e in τ, and every initiator $v' \in e$, $G \vdash_N q(v')$ does not hold. That is, none of those contributing to the establishment of q possesses the established privilege in the initial state G. The following definition captures the condition under which a system is not susceptible to Sybil attacks.

Definition 21. *A system $N = \langle Sub, PV, Pol \rangle$ is **Sybil free** iff the following holds: For every query q of the form "v reads u," for every state $G \in \mathcal{S}_N$ in which $G \vdash_N q$ does not hold, and for every trace $\tau \in (\mathcal{T}_N)^*$ that establishes q in G, there exists a befriending transition e in τ, with an initiator $v' \in e$, such that $G \vdash_N q(v')$.*

Intuitively, the condition above requires that every trace that establishes a privilege must contain a transition initiated by a user who possesses the privilege before the trace is executed. That is, it is impossible for a group of users who do not possess a privilege to collude and cause one of them to gain privilege.

Theorem 22. *A system is Sybil free iff it is POPA compliant.*

This result is surprising, because POPA compliance is apparently stronger than Sybil freedom. Yet it turns out that the two are equivalent. This means POPA compliance is the not only sufficient for preventing Sybil attacks, but also necessary. As such POPA compliance gives us a tight condition for Sybil attack prevention.

Proof. Let $N = \langle Sub, PV, Pol \rangle$ be the system in question. We begin with the "if" direction. Suppose N is POPA compliant. Consider a query q of the form "v reads u", a state $G \in \mathcal{S}_N$ such that $G \vdash_N q$ does not hold, and a trace $\tau \in (\mathcal{T}_N)^*$ that establishes q in G. POPA compliance (Definition 20) guarantees that there exists a rationalization π for every minimal (G, q)-reduction of τ. We know that π contains at least one transition, and that every transition in π is a befriending transition (Proposition 16). Let e be the first transition in π. Now, e also occurs in τ, and by Definition 18 we know $G \vdash_N q(v')$ for some initiator v' of e. Therefore, N is Sybil free.

We now prove the "only-if" direction. Suppose N is Sybil free. Consider a query q of the form "v reads u", states $G, G' \in \mathcal{S}_N$, and a trace $\tau \in (\mathcal{T}_N)^*$ such that $G \xrightarrow{\tau}_N G'$. Suppose further $G \vdash_N q$ does not hold but $G' \vdash_N q$ does. Let π be any minimal (G, q)-reduction of τ.

Our goal is to construct a rationalization π' for π. We describe in the following an inductive procedure for constructing π'.

We begin with the base case, in which π contains only one transition e_0. Sybil freedom guarantees that one of the initiator $v_0 \in e_0$ is such that $G \vdash_N q(v_0)$. Thus we can simply take $\pi' = \pi$ to satisfy the requirement of admissibility (Definition 18).

In the induction step, consider $\pi = e_0 e_1 \dots e_{n-1}$, for some $n > 1$, such that:

$$G = G_0 \xrightarrow{e_0}_N G_1 \xrightarrow{e_1}_N \dots \; G_{n-1} \xrightarrow{e_{n-1}}_N G_n = G''$$

for some social graph $G'' \in \mathcal{S}_N$. Sybil freedom guarantees that there exists a transition e_i, $0 \leq i < n$, such that one of its initiators $v_i \in e_i$ satisfies $G \vdash_N q(v_i)$. Let $\pi = \pi_1 e_i \pi_2$ for some $\pi_1, \pi_2 \in (\mathcal{T}_N)^*$. Now consider the trace $\pi^\sharp = e_i \pi_1 \pi_2$ obtained by moving e_i to the front of the sequence, leaving all the other transitions as is. By Proposition 17 the trace $\pi^\sharp$ is also a minimal (G, q)-reduction. Suppose

$$G \xrightarrow{e_i}_N G^\sharp \xrightarrow{\pi_1 \pi_2}_N G''$$

We know that $G^\sharp \vdash_N q$ does not hold, or else $\pi^\sharp$ would not have been a minimally reduced trace. This means $\pi_1 \pi_2$ is also a minimal (G, q)-reduction. By induction, we can construct a rationalization $\pi^\flat$ of $\pi_1 \pi_2$. We take $\pi' = e_i \pi^\flat$ to be the desired rationalization of π. □

5 A Static Analysis

POPA compliance is a property of the set of feasible traces generated by an FSNS. As such it is a dynamic property. Our goal in this section is to devise a static analysis so that one can ascertain that a given FSNS is POPA compliant without having to examine all feasible traces of the system. To this end we will formulate a static property regarding the policy vocabulary of an FSNS. We will establish a soundness theorem, which asserts that, if an FSNS has a policy vocabulary conforming to the said static property, then the system is POPA compliant. One does not generally expect such a static property to be complete: i.e., every POPA-compliant system possesses such a static property. To our surprise, the static property we are to formulate is indeed complete. Lastly, we also provide an alternative characterization of the static property that is more convenient to use when the policies in question are proper.

5.1 Assumptions

In this section, we target our analysis on the last line of defense of an FSNS: the access policies. This means we do not consider Stage-I Authorization to be a proper component of access authorization, but rather, merely as an idiosyncratic means for, say, protecting the privacy of the users' friend lists. This narrowing of focus is captured in the following assumption.

Assumption 23 (Unmediated Access)**.** $PV(\mathsf{sch}) = \{\top\}$.

With this assumption, the only search policy that can be adopted will be $\top$. That is, global name search will always succeed. The effect is that profile access is now guarded only by the access policy, but not by the reachability of the owner's search listing, because the search listing is now rendered always reachable. Essentially, we are replacing (R-Acc) by the following inference rule:

$$\frac{N = \langle \_, \_, Pol \rangle \qquad Pol(u, \mathsf{acc})(u, v, G)}{G \vdash_N v \ \mathsf{reads}\ u} \qquad \text{(R-Acc}^{-}\text{)}$$

The significance of this assumption can be understood in terms of constraining the scope of "blaming". Recall that POPA compliance is defined in terms of reduction, which is a means for discarding from consideration the set of transitions that do not contribute to the establishment of access. By removing from (R-Acc) the precondition of reaching the owner's search listing, we are essentially refusing to "blame" a transition if it contributes only to making the owner's search listing reachable, but not to satisfying the owner's access policy.

5.2 The Substructure Property

We state the Substructure Property, which can be employed as the basis for a static analysis that verifies if a given FSNS is POPA compliant.

Definition 24. *Suppose $N = \langle Sub, PV, Pol \rangle$ is an FSNS, and $\mathcal{B}$ is the minimal set of birooted graphs to monotonically induce a predicate $P \in \mathcal{PP}(Sub)$. Then P is said to have the* ***Substructure Property (SP) at*** *$u \in Sub$ iff the following holds:*

> *For every $G_{(u,v)} \in \mathcal{B}$, there exists a non-empty, finite sequence of distinct vertices $v_0, v_1, \ldots, v_n = v$ from Sub, and a corresponding sequence of graphs $G_0, G_1, G_2, \ldots, G_n = G$ from $\mathcal{G}(Sub)$, such that, for $0 \leq i \leq n$, we have: (a) $G_i \subseteq G$, (b) $G_{i(u,v_i)}$ is a member of $\mathcal{B}$, and (c) the vertex set V_i is a vertex cover for $E(G_i)$, where $V_0 = \emptyset$, and $V_i = \{v_0, \ldots, v_{i-1}\}$ for $1 \leq i \leq n$.*

We say that P has SP iff P has SP at every $u \in Sub$.

Definition 25. *An FSNS $N = \langle Sub, PV, Pol \rangle$ has SP iff, for every $u \in Sub$, the policy predicate $Pol(u, \mathsf{acc})$ has SP at u.*

In the sequel, we will show that an FSNS that has SP is also POPA compliant, meaning that SP can serve as a static analysis for verifying POPA compliance. In practice, it would be inefficient to verify SP at every vertex, as the number of users can be very large. The following corollary prescribes a more tractable procedure for static analysis: one examines the policy vocabulary of the system rather than the individual privacy settings of users.

Corollary 26. *An FSNS $N = \langle Sub, PV, Pol \rangle$ has SP if every policy predicate in $PV(\mathsf{acc})$ has SP.*

Note that the procedure above checks whether each access policy as a whole has SP, rather than checking if some access policy has SP at a specific vertex. Still, checking whether a monotonic predicate has SP can be tedious, as there are potentially many birooted graphs to consider in the minimal set to monotonically induce the predicate. Fortunately, if a policy is positive, then checking can be performed much more efficiently, as there are usually only a few birooted graphs to consider.

Corollary 27. *Suppose $N = \langle Sub, PV, Pol\rangle$ is an FSNS, and $\mathcal{B}$ is the minimal set of birooted graphs to* positively *induce a predicate $P \in \mathcal{PP}(Sub)$. Then P has SP iff the following holds:*

> *For every $G_{(u,v)} \in \mathcal{B}$, there exists a non-empty, finite sequence of distinct vertices $v_0, v_1, \ldots, v_n = v$ from Sub, and a corresponding sequence of graphs $G_0, G_1, G_2, \ldots, G_n = G$ from $\mathcal{G}(Sub)$, such that, for $0 \leq i \leq n$, we have: (a) $G_i \subseteq G$, (b) $G_{i(u,v_i)}$ is* isomorphic to *a member of $\mathcal{B}$, and (c) the vertex set V_i is a vertex cover for $E(G_i)$, where $V_0 = \emptyset$, and $V_i = \{v_0, \ldots, v_{i-1}\}$ for $1 \leq i \leq n$.*

As we saw in Section 3.3, birooted graphs could be used as a policy language for specifying the minimal set of "patterns" to positively induce a policy predicate. Corollary 27 is therefore the preferred form to use when the static analysis in Corollary 26 is to be performed. Note, however, the definition of SP is based on the more primitive notion of monotonic policies (instead of positive policies). As we shall see in the sequel, this minimalist approach will pay dividend when we work with policy predicates that are monotonic but not topology based.

Example 28. *Let $Sub = \{u_0, u_1, \ldots, u_n\}$. The following are examples and counter-examples of SP.*

1. *The policy predicate dist_k has SP. To see this, recall from Example 8 that the minimal set to positively induce dist_k is $\mathcal{B}_{\mathsf{dist}_k} = \{G^i{}_{(u_0,u_i)} \mid 0 \leq i \leq k\}$, where the graph G^i contains exactly the edges that form the path $u_0u_1 \ldots u_i$. For each $G^i{}_{(u_0,u_i)}$, the vertex sequence $u_0, u_1, \ldots, u_i$ and graph sequence $G^0, G^1, \ldots, G^i$ satisfy the requirements of Corollary 27.*

2. *The policy predicate cf_k has SP. Recall from Example 8 that the minimal set to positively induce cf_k is $\mathcal{B}_{\mathsf{cf}_k} = \{G^{\emptyset}{}_{(u_0,u_0)}, G^{\flat}{}_{(u_0,u_1)}, G^{\sharp}{}_{(u_0,u_{k+1})}\}$, where $G^{\flat}$ and $G^{\sharp}$ are defined in Example 8. That $G^{\emptyset}{}_{(u_0,u_0)}$ and $G^{\flat}{}_{(u_0,u_1)}$ satisfy the three requirements of Corollary 27 is obvious. For $G^{\sharp}{}_{(u_0,u_{k+1})}$, the three requirements of Corollary 27 are satisfied by the vertex sequence $u_0, u_1, \ldots, u_{k+1}$ and the graph sequence $G^0, G^1, \ldots, G^{k+1}$, where $G^0 = G^{\emptyset}$, $G^{k+1} = G^{\sharp}$, and $G^i = \langle Sub, \{\{u_0, u_i\}\}\rangle$ for $1 \leq i \leq k$.*

3. *The topology-based policies used in Facebook, including $\bot$, me, friend, fof and $\top$, have SP. To see this, note that me, friend and fof are but shorthands for dist_k policies, and $\bot$ and $\top$ have SP by virtue of Proposition 30.1.*

4. *The policy predicate clique_k does not have SP at u for every $u \in Sub$, nor does the policy in Figure 1b*

The following observation is immediate.

Lemma 29. *In Definition 24, G_0 is necessarily an empty graph.*

Proof. Recall that $V_0 = \emptyset$. The only way V_0 can be a vertex cover for $E(G_0)$ is for G_0 to be an empty graph. □

Additional facts about SP are the following:

Proposition 30. *The following statements hold:*

1. *Both constant policies $\top$ and $\bot$ satisfy SP.*
2. *If P_1 and P_2 both have SP (at u), then $P_1 \vee P_2$ also has SP (at u).*
3. *If P has SP (at u), then $P(\!|u_0|\!)$ has SP (at u).*
4. *If P has SP, then P^{+1} has SP.*

Statement 1 gives boundary examples of SP. Statements 2–4 offer ways to compose complex SP-satisfying policies from simpler ones. There are two versions to each of Statements 2 and 3: one with the "at u" clauses, the other without. Note that SP is not preserved by $\wedge$ and $\circ$.

Proof. We prove the statements in turn.

1. With constant policies, no query can be established in any state. SP is therefore satisfied vacuously.

2. Let $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_\vee$ be respectively the minimal set of birooted graphs to monotonically induce P_1, P_2 and $P_1 \vee P_2$. Note the following: (i) by Proposition 13, every member of $\mathcal{B}_\vee$ is either a member of $\mathcal{B}_1$ or $\mathcal{B}_2$ (or both); (ii) by the minimality of $\mathcal{B}_\vee$ and the fact that $P_i \Rightarrow (P_1 \vee P_2)$, for every $G_{(u,v)} \in \mathcal{B}_i$, there exists $G'_{(u,v)} \in \mathcal{B}_\vee$ such that $G'_{(u,v)} \subseteq G_{(u,v)}$.

   Assuming that P_1 and P_2 has SP at u, we demonstrate that $P_1 \vee P_2$ also has SP at u. Consider $G_{(u,v)} \in \mathcal{B}_\vee$. We know from (i) above that $G_{(u,v)}$ is a member of either $\mathcal{B}_1$ or $\mathcal{B}_2$. Without loss of generality, say it is $\mathcal{B}_1$. Since P_1 satisfies SP at u, there is a sequence of vertices $v_0, \ldots, v_n = v$ and a sequence of graphs $G_0, \ldots, G_n = G$ as promised by Definition 24 with respect to $\mathcal{B}_1$. For each $G_{i(u,v_i)}$, $0 \leq i < n$, condition (ii) above ensures there exists $G'_{i(u,v_i)} \in \mathcal{B}_\vee$ such that $G'_{i(u,v_i)} \subseteq G_{i(u,v_i)}$. It is easy to check that the sequence of vertices $v_0, \ldots, v_n = v$ and sequence of graphs $G'_0, G'_1, \ldots, G'_{n-1}, G$ satisfy requirements (a), (b) and (c) in Definition 24 with respect to $\mathcal{B}_\vee$.

3. Suppose P has SP at u. We show that $P(\!|u_0|\!)$ has SP at u. If $u \neq u_0$ then the requirement is satisfied vacuously as $P(\!|u_0|\!)(u, v, G) = 0$ for all v and G. If $u = u_0$ then we can reuse the vertex and graph sequences promised by Definition 24 to satisfy the requirement.

4. Suppose P has SP and u is a arbitrary vertex. We show that P^{+1} has SP at u. Let $\mathcal{B}$ and $\mathcal{B}_{+1}$ be the minimal sets of birooted graphs to monotonically induce P and P^{+1} respectively. Consider a member $G_{(u,v)}$ of $\mathcal{B}_{+1}$. We know from Proposition 13 that there are three cases for $G_{(u,v)}$.

Case 1: $u = v$, $G = G^{\emptyset}$, and thus $G_{(u,v)} = {G^{\emptyset}}_{(u,u)}$.

The singleton vertex sequence $v_0 = v$ and graph sequence $G_0 = G^{\emptyset}$ satisfy the three requirements of Definition 24.

Case 2: $u \neq v$, and $G = G^{\emptyset} + \{u, v\}$.

The vertex sequence $u = v_0, v_1 = v$ and the graph sequence $G^{\emptyset} = G_0, G_1 = G^{\emptyset} + \{u, v\}$ satisfy the three requirements of Definition 24.

Case 3: $u \neq v$, and $G = G' + \{u, u'\}$ for some vertex u' ($u' \neq v$) and graph G' such that ${G'}_{(u',v)} \in \mathcal{B}$.

Since P has SP and thus also has SP at u', by Definition 24 there exists a vertex sequence $v_0, v_1, \ldots, v_n = v$ and graph sequence $G_0, G_1, \ldots, G_n = G'$ with respect to ${G'}_{(u',v)} \in \mathcal{B}$.

Without loss of generality, we assume that u is not one of $v_0, v_1, \ldots, v_{n-1}$. If $u = v_i$, $0 \leq i < n$, then we simply remove v_i and G_i from the above sequence without affecting the following construction.

Construct the vertex sequence $u, v_0, v_1, \ldots, v_n$ and graph sequence $G^{\emptyset}, G_0^{\star}, G_1^{\star}, \ldots, G_{n-1}^{\star}, G_n + \{u, u'\} = G' + \{u, u'\} = G$, where, for $0 \leq i < n$, $G_i^{\star} \in$ is a subgraph of $G_i + \{u, u'\}$ that belongs to $\mathcal{B}_{+1}$. It is mechanical to check that the vertex sequence and the graph sequence above satisfy the requirements of Definition 24 with respect to $G_{(u,v)} \in \mathcal{B}_{+1}$.

□

5.3 Soundness and Completeness

The connection between POPA as dynamic property and SP as a static analysis is captured in a pair of theorems: Soundness and Completeness. We begin with the main result of this work:

Theorem 31 (Soundness)**.** *If an FSNS has SP, then it is POPA compliant.*

The soundness theorem allows us to examine the privacy settings of an FSNS, or more often its policy vocabulary (Corollary 26), to determine if the system is POPA compliant.

Proof. Suppose the FSNS $N = \langle Sub, PV, Pol \rangle$ has SP. We show that N is POPA compliant.

Consider a social graph $G \in \mathcal{S}_N$, vertices $u, v \in Sub$, and a query q of the form "v reads u". Let $P = Pol(u, \mathsf{acc})$. Let $\mathcal{B}$ be the minimal set of birooted graphs to monotonically induce P. Now, suppose $\neg P(u, v, G)$, and $G \xrightarrow{\tau}_N G'$ for some trace $\tau \in (\mathcal{T}_N)^*$ and social graph G', such that $P(u, v, G')$. That is, τ establishes q in G. We construct in the following the rationalization π required by the definition of POPA compliance (Definition 20).

Because $P(u, v, G')$ but $\neg P(u, v, G)$, there is at least one birooted graph ${G^{\star}}_{(u,v)} \in \mathcal{B}$ such that ${G^{\star}}_{(u,v)} \subseteq {G'}_{(u,v)}$ but not ${G^{\star}}_{(u,v)} \subseteq G_{(u,v)}$. Let $E = E(G^{\star}) \backslash E(G)$. That is, E is a minimal set of edges that can be added to G to establish q. (The set E is minimal because $\mathcal{B}$ is assumed to be minimal.) The trace τ must contain a subsequence τ' that correspond to a permutation of the edges in E. The subsequence τ' is a minimal (G, q)-reduction of τ. In fact, every (G, q)-reduction τ' of τ corresponds to some permutation of an E constructed as

above from an appropriate choice of $G^\star$ (in the general case there may be multiple candidates for $G^\star$). Fixing one choice of $G^\star$, E and τ', we now show that there is a permutation π of the edges in E that is admissible. This permutation π is a rationalization of τ', as required by the theorem.

We construct π as follows. As N has SP, P has SP at u. Let v_0, v_1, ..., $v_n = v$ and G_0, G_1, ..., $G_n = G^\star$ be the vertices and graphs promised by Definition 24. We start with π being an empty trace. We will go through $n+1$ rounds of construction. In each round, we draw a number of (possibly zero) edges from E, and add them to the end of π. Specifically, no edge is selected in round 0. Then, in round i, $i \in \{1, \ldots, n\}$, an edge e is selected from E iff one end of e is v_{i-1}, and the other end of e is one of v_i, ..., v_n. Multiple edges may be selected in the same round. In such a case, we add the edges to the end of π in any arbitrary order. Note that every edge from E is selected in exactly one round. Thus all edges in E will have been added into π by the end of the $n+1$ rounds.

We argue that π is admissible. Let π_i be the trace we have constructed by the end of round i. This trace π_i is feasible in state G, as every transition in π_i is a distinct befriending transition e where e is by construction absent from $E(G)$. Suppose $G \xrightarrow{\pi_i}_N G_i^\sharp$ for some social graph $G_i^\sharp$. We claim that $P(u, v_i, G_i^\sharp)$ is an invariant for the construction process. We prove this claim by first considering the case for $i = 0$. Since $G_0^\sharp$ is obviously a supergraph of the empty graph, monotonicity and Lemma 29 guaranees that $P(u, v_0, G_0^\sharp)$ holds. We now consider the case when $i > 0$. By monotonicity, the claim holds if $G_i \subseteq G_i^\sharp$. By SP, the vertex set $\{v_0, \ldots, v_{i-1}\}$ is a vertex cover of $E(G_i)$. This means every edge in G_i either (a) is already in G, or (b) has already been added into π_i in round i or earlier. We thus have $G_i \subseteq G_i^\sharp$. So the claim holds. Now, we use the invariant to prove admissibility. Let e be an arbitrary befriending transition in π, and let i be the round in which e is added into π. We know $i > 0$ because no edge is selected in round 0. By construction, one end of e is v_{i-1}. We know from the invariant that $P(u, v_{i-1}, G_{i-1}^\sharp)$. When π is executed from G, in the social graph $G^\flat$ immediately before e is added, we shall have $P(u, v_{i-1}, G^\flat)$ by monotonicity (because $G_{i-1}^\sharp \subseteq G^\flat$). Admissibility of π is therefore guaranteed. □

The second theorem to connect SP to POPA compliance is the completeness theorem.

Theorem 32 (Completeness)**.** *If an FSNS is POPA compliant, then it has SP.*

Completeness asserts that the static analysis corresponding to SP is precise, in the sense that it does not miss out any opportunity to declare a system to be POPA compliant.

Proof. Suppose the FSNS $N = \langle Sub, PV, Pol \rangle$ is POPA compliant. We show that N has SP. That is, we show that, for every user $u \in Sub$, the policy predicate $Pol(u, \mathsf{acc})$ has SP at u.

Let P be $Pol(u, \mathsf{acc})$ for some arbitrary $u \in Sub$. Let $\mathcal{B}$ be the minimal set of birooted graphs to monotonically induce P. Consider an arbitrary member $G_{(u,v)}$ in $\mathcal{B}$. We construct for $G_{(u,v)}$ the sequences v_0, v_1, ..., $v_n = v$ and G_0, G_1, ..., $G_n = G$ as required by Definition 24.

If G is an empty graph, then the sequence can be constructed trivially by taking $n = 0$.

Suppose G contains at least one edge. Let q be the query "v reads u". Because P is monotonic, and $\mathcal{B}$ is minimal, we know that no proper subgraph G' of G_0 can satisfy

$P(u, v, G')$. Therefore, we have $G \vdash_N q$ but not $G^\emptyset \vdash_N q$. Let τ be a trace such that $G^\emptyset \xrightarrow{\tau}_N G$. The definition of POPA compliance (Definition 20) guarantees that every minimal (G, q)-reduction of τ has a rationalization. It is always possible to select τ so that τ is that rationalization.

Let $\tau = e_0 e_1 \dots e_{m-1}$, where $m \geq 1$. Definition 18, together with the choice of v and G, guarantee the existence of the vertex sequence $u_0, u_1, \dots, u_{m-1}, u_m = v$, and the graph sequence $G^\emptyset = G_0^\sharp, G_1^\sharp, \dots, G_{m-1}^\sharp, G_m^\sharp = G$ such that $G_i^\sharp \vdash_N u_i$ reads u for $0 \leq i \leq m$. Note that the above vertex sequence may contain repeated occurrences of the same vertex, but the vertex v occurs exactly once as the last vertex of the sequence (because τ is minimally reduced).

We construct in the following the sequence of distinct vertices $v_0, v_1, \dots, v_n$ required by Definition 24. Suppose there are $n + 1$ distinct vertices in the sequence $u_0, u_1, \dots, u_{m-1}, u_m$ (we know that $n \geq 1$, because u_m is distinct from each of $u_0, \dots, u_{m-1}$ and thus the sequence $u_0, u_1, \dots, u_{m-1}, u_m$ contains at least two distinct vertices). We set v_i to be u_j such that u_j is the i'th vertex to occur for the first time in the sequence. For example, if the sequence $u_0, u_1, \dots, u_m$ is a, a, c, a, b, c, a, d, then $v_0 = a$, $v_1 = c$, $v_2 = b$, and $v_3 = d$. This choice is also consistent with the requirement that $v_n = v$. Let $G_i^\flat$ be the graph $G_j^\sharp$ such that j is the smallest index for which $v_i = u_j$. Observe that the vertex set $V_i = \{v_0, \dots, v_{i-1}\}$ is a vertex cover for $E(G_i^\flat)$. This is because Definition 18 guarantees that every edge in $G_i^\flat$ is added by a befriending transition initiated by some vertex in V_i. Since P is monotonically induced by $\mathcal{B}$, there is a birooted graph $G_{i(u,v_i)} \in \mathcal{B}$ such that $G_i \subseteq G_i^\flat \subseteq G$. The vertex set V_i must therefore be a vertex cover for G_i as well. □

Notice that the static analysis corresponding to Corollary 26, which examines the policy vocabulary rather than the actual access policies, is conservative (i.e., incomplete). A system with a policy vocabulary that contains SP-violating policies, can be POPA compliant so long as none of those SP-violating policies are actually adopted as access policies. In practice, however, Corollary 26 is the appropriate means for static analysis, as a system designer cannot anticipate which of the policies in the vocabulary are actually adopted.

5.4 Proper Predicates

The formulation of SP presented in this Section is very general, accommodating even policies that are not natural. Naturally occurring policy predicates are usually proper. We describe here results that apply to proper predicates.

Let us begin with a simple observation.

Lemma 33. *In Definition 24, if P is proper, then $v_0 = u$.*

Proof. Follows immediately from 29 and Definition 9. □

A proper predicate that has SP possesses certain properties.

Proposition 34. *The following statements hold:*

1. *If a proper predicate P has SP, then P is (properly) local.*

2. *Suppose a proper predicate P has SP, and $\mathcal{B}$ is the minimal set of birooted graphs to monotonically induce P. For every $G_{(u,v)} \in \mathcal{B}$, v is not a cut vertex in G.*

Statements 1 and 2 provide quick tests to identify policies that do not have SP. For example, consider the policy that grants access if the accessor is the owner, or else the accessor has a vertex degree of one hundred or more. Suppose $Sub = \{u_0, u_1, \ldots, u_n\}$. The minimal set of birooted graphs to positively induce this policy is $\{{G^{\emptyset}}_{(u_0,u_0)}, {G^{\star}}_{(u_0,u_1)}\}$, where $G^{\star} \in \mathcal{G}(Sub)$ contains exactly the edges $\{u_1, u_i\}$ for $2 \leq i \leq 101$. In the birooted graph ${G^{\star}}_{(u_0,u_1)}$, the owner root u_0 and the accessor root u_1 belong to two distinct components. This policy, however, is proper. If the policy had SP, then Statement 1 would imply it is properly local. The fact that u_0 and u_1 belong to two components contradicts Corollary 11. Thus the policy does not have SP. As another example, consider the policy of Figure 1b. This policy is properly local, but the accessor root of the birooted graph shown in Figure 1b is a cut vertex, thereby violating the requirement of Statement 2. We can thus safely conclude that such a policy does not have SP.

Proof. We prove the statements in turn.

1. Consider the set-up in the statement of Definition 24. We claim that the following is an invariant: The graph $G_0 \cup G_1 \cup \ldots \cup G_i$ contains a component that includes all of $v_0, \ldots, v_n$ as well as every edge of that graph. Because the policy P is proper, Lemma 33 implies that the invariant holds for the base case ($i = 0$). Now consider v_i. Since P is proper, v_i is not an isolated vertex in G_i. Every edge that incident on v_i joins v_i to one of $v_0, \ldots, v_{i-1}$ (because the latter form a vertex cover for v_i). And we know there is at least one such edge. Therefore, the graph $G_0 \cup G_1 \cup \ldots \cup G_i$ contains a component that includes $v_0, \ldots, v_i$ and every edge of the graph so long as the same can be said about $G_0 \cup G_1 \cup \ldots \cup G_{i-1}$.

2. Consider the set-up in the statement of Definition 24. Because P is proper, we know from Lemma 33 that $v_0 = u$. Again, since P is proper, we know from Statement 1 and Corollary 11 that $v_n = v$ is connected to u in $G_n = G$. Suppose v is a cut vertex. There must be a v_i for which v_i is adjacent to v and the removal of v from G disconnects v_i from u. Select v_i with the smallest possible i. Because neither v_n nor v_i belongs to v_0, $\ldots v_{i-1}$, the edge $\{v_n, v_i\}$ is not covered by $v_0, \ldots v_{i-1}$. This means the edge $\{v_n, v_i\}$ is not in G_i. In fact, by the choice of v_i, the vertex is disconnected from u in G_i. By Statement 1 and Corollary 11, this contradicts with the assumption that P is local.

□

Verifying whether a predicate P has SP can be labor-intensive. It involves, for every birooted graph in the minimal set to monotonically (positively) induce P, the construction of a vertex and a subgraph sequence (Definition 24 and Corollary 26). Fortunately, if P is proper, the check can be constrained to the neighborhood of the accessor root. In the cases when the accessor root has few neighbors, this could translate to a much less labor-intensive check. We present here an alternative characterization of SP for proper predicates.

Theorem 35. *Suppose $N = \langle Sub, PV, Pol\rangle$ is an FSNS, and $\mathcal{B}$ is the minimal set of birooted graphs to monotonically induce a* proper *predicate $P \in \mathcal{PP}(Sub)$. Then P has SP at u iff the following holds:*

> *For every $G_{(u,v)} \in \mathcal{B}$, there exists a non-empty, finite sequence of distinct vertices $v_0, v_1, \ldots, v_n = v$ from the vertex set $\{v\} \cup N_G(v)$, and a corresponding sequence of graphs $G_0, G_1, G_2, \ldots, G_n = G$ from $\mathcal{G}(Sub)$, such that, for $0 \leq i \leq n$, we have: (a) $G_i \subseteq G$, (b) $G_{i(u,v_i)}$ is a member of $\mathcal{B}$, and (c) the vertex set V_i is a vertex cover for the edge set E_i, where $V_0 = \emptyset$, $V_i = \{v_0, \ldots, v_{i-1}\}$ for $0 < i \leq n$, and $E_i = \{e \in E(G_i) \mid e \subseteq \{v\} \cup N_G(v)\}$ for $0 \leq i \leq n$.*

The structure of Theorem 35 is very similar to the definition of SP (Definition 24). We highlight the differences between the two. Firstly, the predicate P is proper in Theorem 35, but not necessarily so in Definition 24. That is, this theorem provides an alternative characterization of SP only when the predicate in question is proper. Secondly, the vertex sequence is taken from the set $\{v\} \cup N_G(v)$ rather than Sub. This alternative characterization is more convenient because it allows us to examine only the closed neighborhood of the accessor root. Thirdly, one only needs to test if the vertex set V_i is a vertex cover of those edges of G_i that lie strictly within the closed neighborhood of v in G.

Proof. We sketch the outline of the proof. The "only-if" direction follows immediately from Definition 24. The "if" direction can be established using induction. In the "if" direction, we need the requirement that P is proper (via Lemma 33, Proposition 34.1, and Corollary 11). □

One can easily produce a corollary of Theorem 35 to account for positive predicates that are proper.

Before we move on, the reader should be reminded that the results presented in this section apply only to an FSNS under Assumption 23. This means the results apply when we do not "blame" transitions in a trace that are responsible solely for enabling reachability of the owner's search listing, but not responsible for satisfying the owner's access policy.

6 Mediated Access

In this section, we relax Assumption 23 (Unmediated Access), and devise a static analysis for the full model presented in Section 3. That is, we use (R-Acc) rather than (R-Acc$^-$) to model access, and we no longer assume that $PV(\mathsf{sch}) = \{\top\}$. Consequently, Stage-I authorization comes into full effect. In order to access the profile of user u, the accessor v must first reach the search listing of u. That is, the query "v finds u" must be satisfied in order for access to be granted.

Incorporating Stage-I authorization into the analysis amounts to widening the scope of "blaming". Befriending transitions that establish the reachability of the owner's search listing, but otherwise play no role in satisfying the owner's access policy, are also "blamed" for enabling access. An important contribution of this section is to point out the analytical complexities and anomalies that could arise from this widening of "blaming" scope.

Let us formulate a high-level strategy of adapting the analysis of the previous section to this more general situation. A key observation is that, once the system N is fixed, the sequent "$G \vdash_N v$ finds u", as defined by the inference rules (F-Sch), (F-Slf), (F-Frd) and (F-Trv), induces a policy predicate that takes an owner u, an accessor v and a social graph G, and returns a boolean value indicating reachability. Denote this predicate by P_{finds}. (Note that P_{finds} is monotonic but unlikely to be topology based, as each user may adopt a different search and traversal policy. This is one of the reasons our previous analyses are formulated in terms of monotonic rather than positive policies.) By (R-Acc), the sequent "$G \vdash_N v$ reads u" holds whenever the policy predicate $P_{\mathsf{finds}} \wedge Pol(u, \mathsf{acc})$ is satisfied by u, v and G. This is equivalent to the set-up in which Unmediated Access is assumed (i.e., (R-Acc$^-$)), but the access policy of u is $P_{\mathsf{finds}} \wedge Pol(u, \mathsf{acc})$ instead. In other words, if one can show, through an appropriate static analysis, that $P_{\mathsf{finds}} \wedge Pol(u, \mathsf{acc})$ has SP at u, for each $u \in Sub$, then Theorem 31 guarantees that the system N is POPA compliant. In addition, Theorem 32 guarantees that this analysis is precise. This is the approach to be taken in this section.

Corollary 36. *Suppose $N = \langle Sub, PV, Pol\rangle$ is an FSNS. Then N is POPA compliant iff the policy predicate $(P_{\mathsf{finds}} \wedge P_{\mathsf{acc}})$ has SP, where the predicate $P_{\mathsf{finds}}(u, v, G)$ holds whenever $G \vdash_N v$ finds u, and the predicate P_{acc} is defined as follows:*

$$P_{\mathsf{acc}} = \bigvee_{u \in Sub} Pol(u, \mathsf{acc})(\!|u|\!) \tag{6}$$

There are two subproblems involved in assessing whether $(P_{\mathsf{finds}} \wedge P_{\mathsf{acc}})$ has SP. The first is to express P_{finds} in terms of the search and traversal policies of N. We address Subproblem 1 in Section 6.1. The second subproblem is to show that the conjunction $(P_{\mathsf{finds}} \wedge P_{\mathsf{acc}})$ has SP at every $u \in Sub$. Part of the problem has to do with identifying the condition under which conjunction preserves SP. It turns out that neither subproblem can always be solved, meaning that once we begin to "blame" transitions that are executed solely for the sake of establishing reachability as causal enablers of access, POPA compliance can no longer be guaranteed in many cases. In Sections 6.2 and 6.3, we examine two instantiations of our model to illustrate how the above analytical approach unfolds. We then reflect on the proper interpretation of negative results in Section 6.4.

6.1 Formulating P_{finds}

We formulate P_{finds} by stating a set of recurrence equations:

$$P^{[0]}(u, v, G) = 0 \tag{7}$$

$$P^{[i+1]}(u, v, G) = Pol(u, \mathsf{sch})(u, v, G) \vee (u = v) \vee (\{u, v\} \in E(G)) \vee \left(\bigvee_{u' \in Sub} \left(P^{[i]}(u', v, G) \wedge (\{u, u'\} \in E(G)) \wedge Pol(u', \mathsf{tra})(u', v, G) \right) \right) \tag{8}$$

The equations defines a sequence of policy predicates $P^{[i]}$ for natural numbers i. Intuitively, $P^{[i]}(u, v, G)$ holds iff "$G \vdash_N v$ finds u" can be established using no more than i applications

of the inference rules (F-Sch), (F-Slf), (F-Frd) or (F-Trv). Equation (7) is obvious because "$G \vdash_N v$ finds u" cannot be established without applying any of the inference rules. Consider the definition of $P^{[i+1]}$ in (8). The disjunct $Pol(u, \mathsf{sch})(u, v, G)$ corresponds to the premises of (F-Sch). Similarly the inference rules (F-Slf) and (F-Frd) are represented respectively by the disjuncts $(u = v)$ and $(\{u, v\} \in E(G))$. The last disjunct:

$$\left(\bigvee_{u' \in Sub} \left(P^{[i]}(u', v, G) \wedge (\{u, u'\} \in E(G)) \wedge Pol(u', \mathsf{tra})(u', v, G) \right) \right)$$

corresponds to the premises of (F-Trv).

Observe that the sequence $P^{[i]}$ is monotonic (i.e., $i \leq j$ implies $P^{[i]}(u, v, G) \Rightarrow P^{[j]}(u, v, G)$), Sub is finite, and $\mathbb{B}$ is a complete lattice with a finite height. By Taski's Fixed Point Theorem [36, 25], the sequence converges to a least fixed point after finitely many iterations. That is, for some $m \in \mathbb{N}$, $i \geq m$ implies $P^{[i]} = P^{[i+1]}$. (We even know that $m = O(|Sub|^2)$.) This $P^{[m]}$ is the policy predicate P_{finds}. Furthermore, because m and Sub are both finite, we can in principle state a closed-form formula for P_{finds}, although the closed form may be enormously long. This is quite acceptable because our interest for now is to demonstrate that the sequence converges and that a closed form exists.

We will simplify the recurrence equations (7) and (8) into a more manageable form. Doing so allows us to analyze when P_{finds} has SP. A technical lemma is needed for the simplification.

Lemma 37. *Suppose $\{P_u\}_{u \in Sub}$ is a family of policy predicates. Then we have the following:*

$$\left(\bigvee_{u' \in Sub} P_{u'}(\!|u'|\!) \right) (u, v, G) = P_u(u, v, G) \tag{9}$$

We are now ready to present a manageable simplification of the recurrence equations.

Lemma 38. *The recurrence equations (7) and (8) define the same predicate sequence as the following recurrence equations.*

$$P^{[0]} = \bot \tag{10}$$

$$P^{[i+1]} = P_{\mathsf{sch}} \vee \left(P^{[i]} \wedge P_{\mathsf{tra}} \right)^{+1} \tag{11}$$

where

$$P_{\mathsf{sch}} = \bigvee_{u \in Sub} Pol(u, \mathsf{sch})(\!|u|\!) \tag{12}$$

$$P_{\mathsf{tra}} = \bigvee_{u \in Sub} Pol(u, \mathsf{tra})(\!|u|\!) \tag{13}$$

Proof. That (10) is equivalent to (7) is immediate. To derive (11) from (8), consider the

following:

$$
\begin{aligned}
&\left(P^{[i]} \wedge P_{\mathsf{tra}}\right)^{+1}(u, v, G) \\
&= (u = v) \vee (\{u, v\} \in E(G)) \vee && \text{by Def. 12} \\
&\quad \bigvee_{u' \in Sub} \left((\{u, u'\} \in E(G)) \wedge P^{[i]}(u', v, G) \wedge P_{\mathsf{tra}}(u', v, G)\right) \\
&= (u = v) \vee (\{u, v\} \in E(G)) \vee && \text{by (13)} \\
&\quad \bigvee_{u' \in Sub} \left((\{u, u'\} \in E(G)) \wedge P^{[i]}(u', v, G) \wedge \left(\bigvee_{u'' \in Sub} Pol(u'', \mathsf{tra})(\!|u''|\!)\right)(u', v, G)\right) \\
&= (u = v) \vee (\{u, v\} \in E(G)) \vee && \text{by (9)} \\
&\quad \bigvee_{u' \in Sub} \left((\{u, u'\} \in E(G)) \wedge P^{[i]}(u', v, G) \wedge Pol(u', \mathsf{tra})(u', v, G)\right)
\end{aligned}
$$

Lastly, we have:

$$
\begin{aligned}
&P_{\mathsf{sch}}(u, v, G) \\
&= \left(\bigvee_{u' \in Sub} Pol(u', \mathsf{sch})(\!|u'|\!)\right)(u, v, G) && \text{by (12)} \\
&= Pol(u, \mathsf{sch})(u, v, G) && \text{by (9)}
\end{aligned}
$$

□

Let us pause and summarize what we have achieved so far. We know that if, for every u, $P_{\mathsf{finds}} \wedge Pol(u, \mathsf{acc})$ has SP at u, then the system is POPA compliant. We further know that P_{finds} is the least fixed point of the following set of recurrence equations:

$$P^{[0]} = \bot \qquad\qquad P^{[i+1]} = P_{\mathsf{sch}} \vee \left(P^{[i]} \wedge P_{\mathsf{tra}}\right)^{+1}$$

Now, suppose we know that $Pol(u, \mathsf{acc})$, $Pol(u, \mathsf{sch})$ and $Pol(u, \mathsf{tra})$ all have SP at u, can we conclude that $P_{\mathsf{finds}} \wedge Pol(u, \mathsf{acc})$ has SP at u? Well, SP is preserved by $\vee$, $\cdot(\!|\cdot|\!)$ and $\cdot^{+1}$. If we can further show that SP is preserved by $\wedge$, then $P_{\mathsf{finds}} \wedge Pol(u, \mathsf{acc})$ will also have SP, and the system is POPA compliant. Yet, $\wedge$ does not preserve SP in the general case. This complication is examined next.

6.2 Traversal Forbidden

The first instantiation of the model that we consider corresponds to the following assumption.

Assumption 39. $PV(\mathit{tra}) = \{\bot\}$.

Under this assumption, the only traversal policy that can be adopted by any user is $\bot$. That is, the inference rule (F-Trv) is essentially rendered inapplicable. An accessor may

reach the search listing of herself, a friend of hers, or someone reachable via global name search. Applying this assumption to (10) and (11), we get the following:

$$\begin{aligned} P^{[0]} &= \bot \\ P^{[1]} &= P_{\mathsf{sch}} \vee (\bot \wedge \bot)^{+1} \\ &= P_{\mathsf{sch}} \vee \mathsf{friend} \\ P^{[2]} &= P_{\mathsf{sch}} \vee ((P_{\mathsf{sch}} \vee \mathsf{friend}) \wedge \bot)^{+1} \\ &= P_{\mathsf{sch}} \vee \bot^{+1} \\ &= P_{\mathsf{sch}} \vee \mathsf{friend} \end{aligned}$$

Therefore, the sequence converges to the following:

$$P_{\mathsf{finds}} = P^{[1]} = P_{\mathsf{sch}} \vee \mathsf{friend}$$

That is, the FSNS is POPA compliant if we can demonstrate that, for every $u \in Sub$, the policy predicate $(P_{\mathsf{sch}} \vee \mathsf{friend}) \wedge Pol(u, \mathsf{acc})$ has SP at u. By Proposition 30.2, this holds when both $(P_{\mathsf{sch}} \wedge Pol(u, \mathsf{acc}))$ and $(\mathsf{friend} \wedge Pol(u, \mathsf{acc}))$ have SP at u. We know friend has SP. Suppose we know also that all search and access policies have SP, can we then conclude that the two conjunctive policies above have SP?

As we noted before, conjunction does not preserve SP. A sufficient condition for SP to be preserved by conjunction is the following.

Definition 40. *Let $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}$ be the minimal sets of birooted graphs to monotonically induce P_1, P_2 and $P_1 \wedge P_2$ respectively. Suppose further that P_1 and P_2 both have SP at u. Then P_1 and P_2, jointly, are said to have the* ***Shared Substructure Property (SSP) at*** *u iff the following requirement holds:*

> *Suppose $G_{(u,v)} \in \mathcal{B}_i$ is a birooted graph such that there is a corresponding birooted graph $G'_{(u,v)} \in \mathcal{B}$ for which $G_{(u,v)} \subseteq G'_{(u,v)}$. (Such a $G_{(u,v)}$ is said to be* ***realized****.) The definition of SP (Definition 24) guarantees the existence of a certain vertex sequence $v_0, v_1, \ldots, v_n = v$ and graph sequence $G_0, G_1, \ldots, G_n = G$. It is required that $G_{j_{(u,v_j)}} \in \mathcal{B}$ for $0 \leq j < n$.*

In addition, if P_1 and P_2 have SP, then we say that they (jointly) have SSP when P_1 and P_2 have SSP at u for every $u \in Sub$.

SSP is a property of an unordered pair of policy predicates. A few subtleties of the definition shall be highlighted. First, a birooted graph $G_{(u,v)}$ belonging to either $\mathcal{B}_1$ or $\mathcal{B}_2$ may not be contained in a member of $\mathcal{B}$ at all. This means that a birooted graph satisfying $P_1 \wedge P_2$ may not contain $G_{(u,v)}$ as a birooted subgraph. Those that do are the ones that are "realized", and those are the ones that further requirements are imposed. Second, say $G_{(u,v)}$ belongs to one of $\mathcal{B}_1$ or $\mathcal{B}_2$. Because P_1 and P_2 have SP at u, the definition of SP guarantees that there is a vertex sequence $v_0, v_1, \ldots, v_n = v$ and graph sequence $G_0, G_1, \ldots, G_n = G$ that satisfy requirements (a), (b) and (c) of Definition 24. We now demand that the birooted graph $G_{j_{(u,v_j)}}$ is not only a member of $\mathcal{B}_i$ (as required by Definition 24(b)), but also a member of $\mathcal{B}$, the minimal set of birooted graphs to monotonically induce $P_1 \wedge P_2$. This is demanded only for $0 \leq j < n$, meaning that we do not demand $G_{(u,v)}$ to be a member of $\mathcal{B}$.

Theorem 41. *If policies P_1 and P_2 each has SP (at u) and jointly have SSP (at u), then $P_1 \wedge P_2$ has SP (at u).*

Note that the theorem asserts two statements. One with the "at u" clauses present, and the other without. The theorem states that SP is preserved by conjunction when the two policies involved have SSP.

Proof. We prove the version of the theorem with the "at u" clauses. The version without the "at u" clauses follow immediately.

Suppose P_1 and P_2 each has SP at u and jointly have SSP at u. Let $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}$ be the minimal sets of birooted graphs to monotonically induce P_1, P_2 and $P_1 \wedge P_2$ respectively. Our goal is to show that $P_1 \wedge P_2$ has SP at u.

Consider $G_{(u,v)} \in \mathcal{B}$. By Proposition 13, $G_{(u,v)}$ can be expressed as $(G^1 \cup G^2)_{(u,v)}$, where ${G^1}_{(u,v)} \in \mathcal{B}_1$ and ${G^2}_{(u,v)} \in \mathcal{B}_2$. Note that both ${G^1}_{(u,v)}$ and ${G^2}_{(u,v)}$ are realized. Definition 24 guarantees that there exists, for each $i \in \{1, 2\}$, a vertex sequence $v^i_0, v^i_1, \ldots, v^i_{n_i} = v$ and a graph sequence $G^i_0, G^i_1, \ldots, G^i_{n_i} = G^i$, such that, for $0 \leq j \leq n_i$, we have: (a) $G^i_j \subseteq G^i$, (b) $G^i_{j_{(u,v^i_j)}} \in \mathcal{B}_i$, and (c) the vertex set V^i_j is a vertex cover for $E(G^i_j)$, where $V^i_0 = \emptyset$, and $V^i_j = \{v^i_0, v^i_1, \ldots, v^i_{j-1}\}$ for $0 < j \leq n$.

We now construct the required vertex sequence and graph sequence for $G_{(u,v)}$. Let $U = \{v^1_0, v^1_1, \ldots, v^1_{n_1}\} \cup \{v^2_0, v^2_1, \ldots, v^2_{n_2}\}$. (Recall that $v^1_{n_1} = v^2_{n_2} = v$.) Let $n = |U|$. We construct a vertex sequence $v_0, v_1, \ldots, v_n$ as follows. First, $v_0, v_1, \ldots, v_{n_1-1}$ are simply $v^1_0, v^1_1, \ldots, v^1_{n_1-1}$. Next, $v_{n_1}, v_{n_1+1}, \ldots, v_{n-1}$ are the vertices taken from $U \backslash \{v^1_0, v^1_1, \ldots, v^1_{n_1}\}$, in the order they appear in the vertex sequence $v^2_0, v^2_1, \ldots, v^2_{n_2}$. Lastly, $v_n = v$. We define the graph sequence $G_0, G_1, \ldots, G_n$ accordingly. First, the graphs $G_0, G_1, \ldots, G_{n_1-1}$ are $G^1_0, G^1_1, \ldots, G^1_{n_1-1}$. Next, the graphs $G_{n_1}, G_{n_1+1}, \ldots, G_{n-1}$ are those graphs among $G^2_0, G^2_1, \ldots, G^2_{n_2-1}$ that correspond to the vertices $v_{n_1}, v_{n_1+1}, \ldots, v_{n-1}$ selected out of $v^2_0, v^2_1, \ldots, v^2_{n_2-1}$. Lastly, we set G_n to G. The following can be observed. First, $G_j \subseteq G$. This is because $G_n = G$ by construction, and $G_j \subseteq G^i \subseteq G^1 \cup G^2 = G$ for $0 \leq j < n$. Second, $G_{j_{(u,v_j)}} \in \mathcal{B}$. In the case of $j = n$, $G_n = G \in \mathcal{B}$ by construction. In the case of $0 \leq j < n$, since G^i_n is a realized member of $\mathcal{B}_i$, thus by SSP $G_j \in \mathcal{B}$. Third, the vertex set V_j is a vertex cover for $E(G_j)$, where $V_0 = \emptyset$, and, for $0 < j \leq n$, $V_j = \{v_0, v_1, \ldots, v_{j-1}\}$. This follows immediately from the fact that V^i_j is a vertex cover of $E(G^i_j)$. □

Example 42. *Suppose $PV(\mathsf{acc}) = PV(\mathsf{sch}) = \{\, \mathsf{dist}_k \mid k \in \mathbb{N} \,\}$. (Recall friend is but a shorthand for dist_1.) Then every pair of such policies have both SP and SSP. According to Theorem 41, the accessibility predicate $(P_{\mathsf{sch}} \vee \mathsf{friend}) \wedge Pol(u, \mathsf{acc})$ has SP at u, for every $u \in Sub$. By Corollary 36, the FSNS is POPA compliant.*

While the instantiation above motivates SSP, the next instantiation illustrates the fact that we do not even need complete information of P_{finds} in order to determine if the system is POPA compliant.

6.3 Boolean Traversal Policies

Let us apply our analysis to a slightly more complex FSNS, through a relaxation of Assumption 39.

Assumption 43. $PV(\mathit{tra}) = \{\bot, \top\}$.

Under this assumption, a user either allows every user to examine her friend list, or deny friend list access categorically. Although Assumption 43 appears to be a minor relaxation of Assumption 39, we shall see that it results in a major increase in the complexity of the analysis.

Our first order of business is to obtain a closed form for P_{finds} [4]. Suppose the FSNS is in state G. If there is a path $v_0 v_1 \ldots v_n$ in G ($n \geq 0$) for which $Pol(v_i, \mathsf{tra}) = \top$ holds for $0 < i \leq n$, then we say that the path is a $\top$-path from v_0 to v_n. Note that we do not require that $Pol(v_0, \mathsf{tra}) = \top$. Also, a length-zero path (i.e., $n = 0$) is always a $\top$-path. Suppose a user can reach the search listing of v. Then that user can further reach the search listing of u through n applications of (F-TRV) if there is a $\top$-path from u to v. At the heart of the above observation is that, under Assumption 43, what determines whether v can traverse through the friend list of u is not the whereabout of v in relation to u, but the setting of $Pol(u, \mathsf{tra})$ as $\bot$ or $\top$. Let the policy predicate $\mathsf{traverse}_k$ be defined such that $\mathsf{traverse}_k(u, v, G)$ holds whenever there is a $\top$-path of length no more than k from u to v in G. Define also policy predicate $\mathsf{traverse}$ such that $\mathsf{traverse}(u, v, G)$ holds whenever there is a $\top$-path (of any length) from u to v in G. Formally, $\mathsf{traverse}_k$ is defined as follows:

$$\mathsf{traverse}_0(u, v, G) = (u = v) \tag{15}$$

$$\mathsf{traverse}_{k+1}(u, v, G) = (u = v) \vee \bigvee_{u' \in Sub} \left(\begin{array}{r} (\{u, u'\} \in E(G)) \wedge \mathsf{traverse}_k(u', v, G) \\ \wedge\, P_{\mathsf{tra}}(u', v, G) \end{array} \right) \tag{16}$$

When $k \geq |Sub|$, $\mathsf{traverse}_k = \mathsf{traverse}_{k+1}$. Thus, the following holds when $k \geq |Sub|$:

$$\mathsf{traverse} = \mathsf{traverse}_k \tag{17}$$

Note that both $\mathsf{traverse}_k$ and $\mathsf{traverse}$ are monotonic predicates. More importantly, since every subpath of a $\top$-path is also a $\top$-path, it is easy to show that both $\mathsf{traverse}_k$ and $\mathsf{traverse}$ have SP.

[4] In a previous version of this manuscript that we made available on the internet, the definitions of $\top$-path and $\mathsf{traverse}$ are different from the ones adopted in this technical report. The change is mainly for improving the presentation of the materials, making the correctness of the derivation more obvious, and correcting a minor technical error. The conclusions are the same as the previous version. We detail the difference in the following. (We write $\top$-path^{old}, $\mathsf{traverse}^{old}$, $\top$-path^{new} and $\mathsf{traverse}^{new}$ to differentiate the two versions within this footnote.)

- A $\top$-path^{old} was originally defined as a path $v_0 v_1 \ldots v_n$ in which all interior vertices v_i, $0 < i < n$, are such that $Pol(v_i, \mathsf{tra})$ holds. A $\top$-path^{new} adds the additional requirement that $Pol(v_i, \mathsf{tra})$ holds if $n > 0$.
- The predicate $\mathsf{traverse}^{old}$ and $\mathsf{traverse}^{new}$ are based on two different notions of $\top$-paths. Specifically, we have:
$$\mathsf{traverse}^{old} = \mathsf{traverse}^{new} \circ \mathsf{friend} \tag{14}$$
- Given equation (14), the conclusions of Example 46 are identical in both versions.

Lemma 44. *Let P be a monotonic policy predicate. The following holds:*

$$(\mathit{traverse}_{k+1} \circ P)(u, v, G) = P(u, v, G) \vee \bigvee_{u' \in Sub} \left(\begin{array}{l} (\{u, u'\} \in E(G)) \wedge \\ ((\mathit{traverse}_k \circ P) \wedge P_{\mathit{tra}})(u', v, G) \end{array} \right) \quad (18)$$

Proof.

$$\begin{array}{ll}
& (\mathsf{traverse}_{k+1} \circ P)(u, v, G) \\
= & \bigvee_{v' \in Sub} (\mathsf{traverse}_{k+1}(u, v', G) \wedge P(v', v, G)) \\
= & \bigvee_{v' \in Sub} \left(\begin{array}{l} \left(\begin{array}{l} (u = v') \vee \\ \bigvee_{u' \in Sub} (\{u, u'\} \in E(G)) \wedge \mathsf{traverse}_k(u', v', G) \wedge P_{\mathsf{tra}}(u', v', G)) \end{array} \right) \\ \wedge P(v', v, G) \end{array} \right) \\
= & \bigvee_{v' \in Sub} \left(\begin{array}{l} ((u = v') \wedge P(v', v, G)) \vee \\ \left(\begin{array}{l} \left(\bigvee_{u' \in Sub} (\{u, u'\} \in E(G)) \wedge \mathsf{traverse}_k(u', v', G) \wedge P_{\mathsf{tra}}(u', v', G))\right) \\ \wedge P(v', v, G) \end{array} \right) \end{array} \right) \\
= & P(u, v, G) \vee \\
& \bigvee_{v' \in Sub} \left(\begin{array}{l} \left(\bigvee_{u' \in Sub} (\{u, u'\} \in E(G)) \wedge \mathsf{traverse}_k(u', v', G) \wedge P_{\mathsf{tra}}(u', v', G))\right) \\ \wedge P(v', v, G) \end{array} \right) \\
= & P(u, v, G) \vee \\
& \bigvee_{v' \in Sub} \bigvee_{u' \in Sub} (\{u, u'\} \in E(G)) \wedge \mathsf{traverse}_k(u', v', G) \wedge P_{\mathsf{tra}}(u', v', G) \wedge P(v', v, G)) \\
= & P(u, v, G) \vee \\
& \bigvee_{v' \in Sub} \bigvee_{u' \in Sub} (\{u, u'\} \in E(G)) \wedge \mathsf{traverse}_k(u', v', G) \wedge P_{\mathsf{tra}}(u', v, G) \wedge P(v', v, G)) \\
& [\text{Note: The above step is due to Assumption 43, which implies that} \\
& \ Pol(u', \mathsf{tra})(u', v', G) = Pol(u', \mathsf{tra})(u', v, G).] \\
= & P(u, v, G) \vee \\
& \bigvee_{u' \in Sub} \bigvee_{v' \in Sub} (\{u, u'\} \in E(G)) \wedge \mathsf{traverse}_k(u', v', G) \wedge P_{\mathsf{tra}}(u', v, G) \wedge P(v', v, G)) \\
= & P(u, v, G) \vee \\
& \bigvee_{u' \in Sub} \left(\{u, u'\} \in E(G)) \wedge \left(\bigvee_{v' \in Sub} (\mathsf{traverse}_k(u', v', G) \wedge P(v', v, G))\right) \wedge P_{\mathsf{tra}}(u', v, G)\right) \\
= & P(u, v, G) \vee \\
& \bigvee_{u' \in Sub} (\{u, u'\} \in E(G)) \wedge (\mathsf{traverse}_k \circ P)(u', v, G) \wedge P_{\mathsf{tra}}(u', v, G)) \\
= & P(u, v, G) \vee \\
& \bigvee_{u' \in Sub} (\{u, u'\} \in E(G)) \wedge ((\mathsf{traverse}_k \circ P) \wedge P_{\mathsf{tra}})(u', v, G))
\end{array}$$

□

Theorem 45. *The following holds.*

$$P_{\mathit{finds}} = \mathit{traverse} \circ (P_{\mathit{sch}} \vee \mathit{friend})$$

Intuitively, a search listing can be reached by applying one of (F-Sch), (F-Slf) or (F-Frd), followed by applying (F-Trv) for zero or more times.

Proof. We claim that the following hold:

$$P^{[0]} = \bot \tag{19}$$

$$P^{[k+1]} = \mathsf{traverse}_k \circ (P_{\mathsf{sch}} \vee \mathsf{friend}) \qquad \text{for } k \geq 0 \tag{20}$$

(19) is just a restatement of (10). By (17), (20) implies that $P^{[k]}$ converges to $\mathsf{traverse} \circ (P_{\mathsf{sch}} \vee \mathsf{friend})$ as required by the theorem. We prove (20) by induction.

Base case: By (11), we have:

$$\begin{aligned} P^{[1]} &= P_{\mathsf{sch}} \vee (P^{[0]} \wedge P_{\mathsf{tra}})^{+1} \\ &= P_{\mathsf{sch}} \vee (\bot \wedge P_{\mathsf{tra}})^{+1} \\ &= P_{\mathsf{sch}} \vee \mathsf{friend} \\ &= \mathsf{traverse}_0 \circ (P_{\mathsf{sch}} \vee \mathsf{friend}) \end{aligned}$$

Induction step: Again, by (11), we have:

$$\begin{aligned} P^{[k+2]} &= P_{\mathsf{sch}} \vee (P^{[k+1]} \wedge P_{\mathsf{tra}})^{+1} \\ &= P_{\mathsf{sch}} \vee ((\mathsf{traverse}_k \circ (P_{\mathsf{sch}} \vee \mathsf{friend})) \wedge P_{\mathsf{tra}})^{+1} && \text{by Ind. Hyp.} \\ &= \mathsf{traverse}_{k+1} \circ (P_{\mathsf{sch}} \vee \mathsf{friend}) \end{aligned}$$

To see the last step,

$$\begin{aligned} & (\mathsf{traverse}_{k+1} \circ (P_{\mathsf{sch}} \vee \mathsf{friend}))(u, v, G) \\ &= (P_{\mathsf{sch}} \vee \mathsf{friend})(u, v, G) \vee \\ &\qquad \bigvee_{u' \in Sub} \left(\begin{array}{l} (\{u, u'\} \in E(G)) \wedge \\ \qquad ((\mathsf{traverse}_k \circ (P_{\mathsf{sch}} \vee \mathsf{friend})) \wedge P_{\mathsf{tra}})(u', v, G) \end{array} \right) && \text{by (18)} \\ &= P_{\mathsf{sch}}(u, v, G) \vee (u = v) \vee (\{u, v\} \in E(G)) \vee \\ &\qquad \bigvee_{u' \in Sub} \left(\begin{array}{l} (\{u, u'\} \in E(G)) \wedge \\ \qquad ((\mathsf{traverse}_k \circ (P_{\mathsf{sch}} \vee \mathsf{friend})) \wedge P_{\mathsf{tra}})(u', v, G) \end{array} \right) \\ &= P_{\mathsf{sch}}(u, v, G) \vee ((\mathsf{traverse}_k \circ (P_{\mathsf{sch}} \vee \mathsf{friend})) \wedge P_{\mathsf{tra}})^{+1}(u, v, G) \\ &= (P_{\mathsf{sch}} \vee ((\mathsf{traverse}_k \circ (P_{\mathsf{sch}} \vee \mathsf{friend})) \wedge P_{\mathsf{tra}})^{+1})(u, v, G) \end{aligned}$$

□

Suppose we know that $PV(\mathsf{sch})$ contains only policies that have SP, and thus P_{sch} has SP. We still cannot conclude that $P_{\mathsf{finds}} = \mathsf{traverse} \circ (P_{\mathsf{sch}} \vee \mathsf{friend})$ has SP because SP is not preserved by the $\circ$ combinator. The following example shows us that in some cases, we may still show that P_{finds} has SP, but this is not true in general.

Example 46. *Suppose $PV(\mathsf{sch}) = \{\bot, \mathsf{connected}, \top\}$, where the predicate $\mathsf{connected}$ holds whenever there is a path between the owner and the accessor. (Note that $\mathsf{connected}$ has SP.) Consider $P_{\mathsf{finds}}(\!|u|\!)$. There are three cases. First, if there is a $\top$-path from u to a vertex u' for*

which $Pol(u', \mathsf{sch}) = \top$*, then every user* v *can reach the search listing of* u *by first reaching* u' *by global name search (which will always succeed), and then traversing the* $\top$*-path to reach* u*. Therefore,* $P_{\mathsf{finds}}(\!|u|\!) = \top(\!|u|\!)$*. Second, if the first case does not apply, but there is a* $\top$*-path from* u *to a vertex* u' *for which* $Pol(u', \mathsf{sch}) = \mathsf{connected}$*, then every user* v *which is connected to* u' *can reach the search listing of* u*. But these* v *are exactly those that are connected to* u *itself. Therefore,* $P_{\mathsf{finds}}(\!|u|\!) = \mathsf{connected}(\!|u|\!)$*. Third, if neither the first nor the second case applies, then every vertex* u' *reachable from* u *via a* $\top$*-path is such that* $Pol(u', \mathsf{sch}) = \bot$*. In this case, either* u' *or a friend of* u' *can reach the search listing of* u*. Thus,* $P_{\mathsf{finds}}(\!|u|\!) = \mathsf{traverse} \circ \mathsf{friend}$*. In summary,* $P_{\mathsf{finds}}(\!|u|\!) \in \{ (\mathsf{traverse} \circ \mathsf{friend})(\!|u|\!), \mathsf{connected}(\!|u|\!), \top(\!|u|\!) \}$*.*

Let us consider two instantiations of this FSNS.

1. $PV(\mathsf{acc}) = \{\bot, \mathsf{connected}, \top\}$*. Then it can be shown that* $P_{\mathsf{finds}}(\!|u|\!) \wedge Pol(u, \mathsf{acc}) \in \{ \bot, (\mathsf{traverse} \circ \mathsf{friend})(\!|u|\!), \mathsf{connected}(\!|u|\!), \top(\!|u|\!) \}$ *has SP. Thus the FSNS is POPA compliant.*

2. $PV(\mathsf{acc}) = \{\mathsf{dist}_k \mid k \in \mathbb{N}\}$*. Then it is possible that* $P_{\mathsf{finds}}(\!|u|\!) \wedge Pol(u, \mathsf{acc}) = \mathsf{traverse}(\!|u|\!) \wedge \mathsf{dist}_k$ *for some* $k \in \mathbb{N}$*, which can be shown not to have SP. (It is easy to show that* $\mathsf{traverse}$ *and* dist_k *do not have SSP.) Thus the FSNS is not POPA compliant.*

Observe that our goal above is not to obtain a precise closed form for $(P_{\mathsf{finds}} \wedge P_{\mathsf{acc}})$: we do not even know what traverse looks like. Instead, the main goal is to identify the major subexpressions in $(P_{\mathsf{finds}} \wedge P_{\mathsf{acc}})$, so that we know to which policies and policy pairs we shall impose SP and SSP respectively.

6.4 Interpretation of Negative Results

The second instantiation of Example 46 illustrates that there are in fact FSNSs for which all search, traversal and access policies have SP, but the FSNSs themselves are not POPA compliant. The reason for this is that we now demand the conjunction $(P_{\mathsf{finds}} \wedge P_{\mathsf{acc}})$ to have SP (Corollary 36), rather than requiring simply that P_{acc} has SP (Theorem 31). The failure of the conjunction to satisfy the requirements of SP may be caused by at least two reasons. First, if the reachability predicate P_{finds} has SP, the requirement of SSP may not be satisfied by the relevant policy pairs (e.g., second instantiation of Example 46). Second, P_{finds} may not have SP. The structure of Stage-I authorization is designed in such a way that SP is not a condition we can expect to hold by P_{finds} except for a few special cases. In fact, as the next example illustrates, so long as $PV(\mathsf{tra})$ contains non-constant predicates, it is unlikely that P_{finds} has SP.

Example 47. *Suppose* $PV(\mathsf{sch}) = PV(\mathsf{tra}) = PV(\mathsf{acc}) = \{\bot, \mathsf{me}, \mathsf{friend}, \mathsf{fof}, \top\}$*. (Note the resemblance between this policy vocabulary and that of Facebook.)* P_{acc} *has SP (Example 28), but neither* P_{finds} *nor* $P_{\mathsf{finds}} \wedge P_{\mathsf{acc}}$ *has SP.*

This situation demands us to be very careful in interpreting negative results. Suppose N is an FSNS for which P_{acc} has SP, but $(P_{\mathsf{finds}} \wedge P_{\mathsf{acc}})$ does not have SP. By Corollary 36, we would conclude that N is not POPA compliant, and thus Sybil attacks are possible in N (Theorem 22). But then exactly what is the nature of these Sybil attacks? It turns out such a Sybil attack is caused by befriending transitions which (i) establish the reachability

of the owner's search listing, (ii) play no role in satisfying the owner's access policy, and (iii) involve no initiator that already has access. Yet the assumption is that the access policies have SP. This means in the start state of the Sybil attack, the accessor and her colluding partners already satisfy the access policy of the owner. The reason they do not have access to begin with is because they cannot reach the owner's search listing in the start state. That means all that the offending transitions achieve is to turn the owner's search listing from unreachable to reachable by the accessor and her colluding partners.

The crux of the problem is how we are to perceive Stage-I Authorization. Suppose we do *not* consider Stage-I Authorization to be a proper component of access authorization (Section 5.1). That is, friend lists are no different than other profile items, and thus traversal policies are merely access policies for friend lists. Then so long as we demand that the traversal policies have SP, the FSNS is POPA compliant, and the above anomaly disappears. Suppose, however, we consider Stage-I Authorization to be a proper component of access authorization, such that the search listings are capability-like entities, the reachability of which is a necessary condition for access. The analysis of this section implies that, except in a few special cases, POPA compliance is not achievable. Our conclusion is that the current design of Stage-I Authorization does not support a rational static analysis for POPA compliance. As we will point out in Section 8, a redesign of Stage-I Authorization to facilitate POPA-compliance analysis is a pressing research challenge.

7 Related Work

Recent years have seen the proposal of novel authorization schemes for Relationship-Based Access Control (ReBAC) [20, 6, 18]. In the access control models of Kurk *et al.* [24] and Carminati *et al.* [8, 10], access control policies identify the existence of a directed path between the owner and the accessor in the social network, consisting of edges of a particular relationship type, within a certain length, and with aggregate trust weighting above a certain threshold. Carminati *et al.* [9, 5, 11] later refined their model to use a trust metric for decentralized authorization. In [7], Carminati *et al* employed Semantic Web technologies to encode access control information of an ReBAC system. Fong *et al.* [19, 1] proposed a formal model for Facebook-style Social Network Systems (FSNSs), which is a generalization of the access control model behind Facebook. Although this model does not support directed relationships, relationship types, and trust levels, it supports a uniquely novel feature missing from previously proposed models, namely, the use of arbitrary graph-theoretic relations as access control policies. Fong [18] later devised an ReBAC model that adds support for relationship types and access contexts on top of graph-theoretic policies. A modal logic is employed as its policy language, rather than birooted graphs. Squicciarini *et al.* [33] proposed a scheme for automatically deriving access control policies for the users of an SNS. The present work is an attempt to identify a security property, namely, Principle of Privilege Attenuation, that the above authorization schemes shall strive to guarantee.

To the best of our knowledge, this work is the first to propose a policy analysis for verifying that an SNS complies to a global security property. Comparable to our goal is the work of Squicciarini *et al.* [34, 35], which studied the collaborative management (or co-ownership) of resources in SNSs. Game-theoretic analysis is employed to ensure fairness

and other global properties in their protocol.

Traditional defences against Sybil attacks begin with (direct) identity validation [17, 12, 4, 30], the goal of which is to prevent large-scale injection of pseudonymous identities by a small number of malicious users. Douceur [17] proposed the use of computational challenges to validate that the user behind an identity indeed possesses certain computational resources (e.g., communication, storage, computing power). A malicious user forging multiple, pseudonymous identities will fail to meet these challenges. Similar use of client puzzles for admission control were found in [12, 4, 30]. CAPTCHA [37] is a popular variation of this approach to tell human users apart from computational agents. In the light of the feasibility of Profile Cloning attacks [2] in popular SNSs (including Facebook), partly made possible by circumventing CAPTCHA, this work accepts as an axiom that pseudonymous identities are not avoidable in FSNS. Instead, the goal of this work is to explore how an FSNS can be designed to prevent Sybil attacks even in the presence of pseudonymous identities.

In recent years, major headway has been made to develop near-optimal or optimal defence mechanisms against Sybil attacks in peer-to-peer networks [39, 38] and recommendation systems [40]. Particularly notable are SybilGuard [39] and its successor SybilLimit [38], which provide randomized protocols for a node (the verifier) in a peer-to-peer system to determine if it should engage in a transaction with another node (the suspect). The verification step of both protocols involves performing randomized routing in a stationary social graph defined over all nodes. Exploiting a certain property ("fast-mixing") of social graphs, one can guarantee that, with arbitrarily high probability, the verifiers will accept only logarithmically many Sybil nodes, and turn away arbitrary few honest nodes. A recommendation system rates objects based on the previous assessment of these objects by a population of users. Sybil attacks occur when a number of malicious users or pseudonymous identities collude to influence object ratings. DSybil [40] is a recommendation system that comes with recommendation quality guarantee even in the presence of Sybil attacks. Specifically, an upper bound for bad recommendations made by the system in the worst case is established alongside a matching lower bound. The above works are related to our work in that they all accept pseudonymous identities as an unavoidable reality, and provide protection against Sybil attacks, with provable guarantee in the quality of protection. The similarity, however, ends here. First, our work studies Sybil attacks in the novel context of a Relationship-Based Access Control system (i.e., FSNS), rather than peer-to-peer or recommendation systems. Our goal is to prevent pseudonymous identities from manipulating the topology of the social graph in order to gain access. Second, our solution approach is a static analysis on the policy vocabulary, from which users may freely adopt their own policies, rather than the imposition of a single policy. Third, our guarantee (Sybil freedom via a soundness theorem) is absolute rather than probabilistic, and is not based on assumptions regarding the topology of social graphs.

Cheng and Friedman [13] studied characteristics of reputation functions that are Sybilproof. A reputation function maps an edge-weighted graph and a vertex to a reputation value for that vertex. They showed that Sybilproof reputation functions cannot be "symmetric", meaning that the reputation value is invariant under graph isomorphism. Their result does not apply to our topology-based policies, for which the authorization decision is

invariant under *birooted-graph isomorphism*[5].

Sybil attacks have also been studied in the context of sensor networks [28] (a malicious device taking on multiple identities), alongside with the much related node replication attacks [29] (multiple malicious devices assuming an existing identity). Defence mechanisms include improved key predistribution schemes or attack detection schemes [28, 29]. Again, the focus is on admission control and identity validation.

8 Summary and Future Work

In this work, we examined the challenge of preventing Sybil attacks in Facebook-style Social Network Systems. We formulated a version of the Principle of Privilege Attenuation for FSNSs, and showed that it is the necessary and sufficient condition for preventing Sybil attacks. We then devised a static policy analysis for verifying if an FSNS is POPA compliant. The analysis is both sound and complete. Lastly, we demonstrated the complexities and anomalies that would arise from incorporating Stage-I Authorization into the analysis.

To address the anomalies surrounding Stage-I Authorization, two research questions present themselves. The first is to devise a better characterization of those policy vocabularies that guarantee POPA compliance even in the presence of Stage-I Authorization. The second is to redesign Stage-I Authorization to facilitate a rational POPA-compliance analysis.

The results in this work apply only to monotonic policies. A future direction is to explore how the analysis can be extended to cover policies that may not be monotonic. Also implicit in the current work is the assumption of a particular consent protocol: befriending requires the consent of both parties, but defriending can be initiated unilaterally. Our preliminary experience with nonmonotonic policies suggests that alternative consent protocols may be required in order for the static analysis to become feasible in the midst of nonmonotonic policies.

While this work focuses on static policy analysis, a natural question is whether POPA compliance can be achieved by coupling static analysis of policy vocabulary with dynamic checks performed at run time. A challenge will be to minimize the run-time and storage overhead required for such a scheme.

The definition of Sybil attacks and the formulation of POPA in Section 4 are in fact domain independent. It is interesting to find out if this formulation of POPA is applicable to other discretionary access control systems.

In this work, Sybil attack prevention is taken as a security goal of ReBAC. It is worthwhile to explore alternative security goals for ReBAC, or relaxations of the present security goal. An example of the latter option is to impose lower bound for the "cost" of launching a Sybil attack (e.g., the number of colluding users required [32], etc).

[5]It is also worthwhile to point out that there is a flaw in the proof of their Theorem 1, which incorrectly claims that the social graphs before and after a Sybil attack are isomorphic.

Acknowledgment

This work is partly funded by an NSERC Strategic Project Grant.

References

[1] Mohd Anwar, Zhen Zhao, and Philip W. L. Fong. An access control model for Facebook-style social network systems. Technical Report 2010-959-08, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, July 2010. Submitted for review.

[2] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: Automated identity theft attacks on social networks. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*, pages 551–560, Madrid, Spain, April 2009.

[3] Matt Bishop. *Computer Security*. Addison Wesley, 2002.

[4] Nikita Borisov. Computational puzzles as sybil defenses. In *Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing (P2P'06)*, pages 171–176, Cambridge, UK, September 2006.

[5] Barbara Carminati and Elena Ferrari. Privacy-aware collaborative access control in web-based social networks. In *Proceedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DAS'08)*, volume 5094 of *LNCS*, pages 81–96, London, UK, July 2008. Springer.

[6] Barbara Carminati and Elena Ferrari. Enforcing relationships privacy through collaborative access control in web-based social networks. In *Proceedings of the 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'09)*, Washington DC, USA, November 2009.

[7] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. A semantic web based framework for social network access control. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies (SACMAT'09)*, pages 177–186, Stresa, Italy, June 2009.

[8] Barbara Carminati, Elena Ferrari, and Andrea Perego. Rule-based access control for social networks. In *Proceedings of the OTM 2006 Workshops*, volume 4278 of *LNCS*, pages 1734–1744. Springer, October 2006.

[9] Barbara Carminati, Elena Ferrari, and Andrea Perego. Private relationships in social networks. In *Proceedings of Workshops in Conjunction with the International Conference on Data Engineering – ICDE'07*, pages 163–171, Istanbul, Turkey, April 2007.

[10] Barbara Carminati, Elena Ferrari, and Andrea Perego. A decentralized security framework for web-based social networks. *International Journal of Information Security and Privacy*, 2(4):22–53, October 2008.

[11] Barbara Carminati, Elena Ferrari, and Andrea Perego. Enforcing access control in web-based social networks. *ACM Transactions on Information and System Security*, 13(1), October 2009.

[12] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, USA, December 2002.

[13] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of peer-to-peer systems (P2PEcon'05)*, pages 128–132, Philadelphia, PA, USA, August 2005.

[14] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 51–65, Pittsburch, Pennsylvania, USA, June 2008.

[15] Peter J. Denning. Fault tolerant operating systems. *ACM Computing Surveys*, 8(4):359–389, December 1976.

[16] Jack B. Dennis and Earl C. Van Horn. Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9(3):143–155, March 1966.

[17] John R. Douceur. The Sybil attack. In *Proceedings for the First International Workshop on Peer-to-Peer Systems (IPTPS'02)*, volume 2429 of *LNCS*, pages 251–260, Cambridge, MA, USA, March 2002. Springer.

[18] Philip W. L. Fong. Relationship-based access control: Protection modeal and policy language. Technical Report 2010-974-23, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, September 2010.

[19] Philip W. L. Fong, Mohd Anwar, and Zhen Zhao. A privacy preservation model for Facebook-style social network systems. In *Proceedings of the 14th European Symposium on Research In Computer Security (ESORICS'09)*, volume 5789 of *LNCS*, pages 303–320, Saint Malo, France, September. Springer.

[20] Carrie E. Gates. Access control requirements for Web 2.0 security and privacy. In *IEEE Web 2.0 privacy and security workship (W2SP'07)*, Oakland, California, USA, May 2007.

[21] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy (S&P'82)*, pages 11–20, 1982.

[22] G. Scott Graham and Peter J. Denning. Protection: Principles and practices. In *Proceedings of the 1972 AFIPS Spring Joint Computer Conference*, volume 40, pages 417–429, Alantic City, New Jersey, USA, May 1972.

[23] Norm Hardy. The confused deputy (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review*, 22(4):36–38, October 1988.

[24] Sebastian Ryszard Kruk, Slawomir Grzonkowski, Adam Gzella, Tomasz Woroniecki, and Hee-Chul Choi. D-FOAF: Distributed identity management with access rights delegation. In *Proceedings of the First Asian Semantic Web Conference (ASWC'06)*, volume 4185 of *LNCS*, pages 140–154, Beijing, China, September 2006. Springer.

[25] J. L. Lassez, V. L. Nguyen, and E. A. Sonenberg. Fixed point theorems and semantics: A folk tale. *Information Processing Letters*, 14(3):112–116, 1982.

[26] Ninghui Li and Mahesh V. Tripunitara. On safety in discretionary access control. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 96–109, Oakland, California, USA, May 2005.

[27] Mark S. Miller, Ka-Ping Yee, and Jonathan Shapiro. Capability myths demolished. Technical Report SRL2003-02, System Research Lab, Department of Computer Science, The John Hopkins University, Baltimore, Maryland, USA, 2003.

[28] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The Sybil attack in sensor networks: Analysis & defenses. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*, pages 259–268, Berkeley, CA, USA, April 2004. ACM.

[29] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 49–63, Oakland, CA, USA, May 2005.

[30] Hosam Rowaihy, William Enck, Patrick McDaniel, and Thomas La Porta. Limiting Sybil attacks in structured P2P networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*, pages 2596–2600, Anchorage, Alaska, USA, May 2007.

[31] John Rushby. Noninterference, transitivity, and channel-control security policies. Technical Report CSL 92-02, Computer Science Laboratory, SRI International, 1992.

[32] Lawrence Snyder. Theft and conspiracy in the take-grant protection model. *Journal of Computer and System Sciences*, 23(3):333–347, 1981.

[33] Anna Squicciarini, Federica Paci, and Smitha Sundareswaran. PriMa: An effective privacy protection mechanism for social networks. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS'10)*, pages 320–323, Beijing, China, April 2010.

[34] Anna C. Squicciarini, Mohamed Shehab, and Federica Paci. Collective privacy management in social networks. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*, pages 521–530, Madrid, Spain, April 2009.

[35] Anna C. Squicciarini, Mohamed Shehab, and Joshua Wede. Privacy policies for shared content in social network sites. *The VLDB Journal*, 2010. To appear.

[36] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, pages 285–309, 1955.

[37] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of the 22nd International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'03)*, volume 2656 of *LNCS*, pages 294–311, Warsaw, Poland, May 2003. Springer.

[38] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. SybilLimit: A near-optimal social network defense against Sybil attacks. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy (S&P'08)*, pages 3–17, Oakland, CA, USA, May 2008.

[39] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham D. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. *IEEE/ACM Transactions on Networking*, 16(3):576–589, June 2008.

[40] Haifeng Yu, Chenwei Shi, Michael Kaminsky, Phillip B. Gibbons, and Feng Xiao. DSybil: Optimal Sybil-resistance for recommendation systems. In *Proceedings of the 2009 IEEE Symposium on Security and Privacy (S&P'09)*, pages 283–298, Berkeley, CA, USA, May 2009.