Redirection Policies for Mission-Based Information Sharing

David Keppler
dkeppler@mitre.org

Vipin Swarup
swarup@mitre.org

Sushil Jajodia
jajodia@mitre.org

The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102

ABSTRACT

When an access decision function denies a data access request by a mission participant in a mission-critical situation, the mission often suffers. In this paper, we propose a sharing control mechanism that computes and executes requests that are mission-related to denied requests. We extend the Flexible Authorization Framework (FAF) with predicates and hierarchies that permit us to specify authorization rules over denied requests and mission-specific relationships. We illustrate our techniques using a prototypical information sharing scenario, namely an emergency first-responder scenario.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Security, Integrity, and Protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Management, Security

Keywords

Data sharing, information sharing, access control policy, authorization

1. INTRODUCTION

Access control policies specify whether requests by principals to perform actions on resources should be authorized. The access decisions can rely on a wide variety of information including the principals' attributes, the resources' attributes, the actions being requested, a history of previous events, and environmental attributes [4].

If an access decision function denies an access request, most current systems will reject that request and take no further action. Our investigations have revealed that in mission-critical situations, humans and organizations violate such strict rejection policies routinely, although in an ad hoc manner. We address this problem by proposing a general sharing control framework that, rather than merely denying requests, determines a range of other possible actions to take. Our approach is based on our observation that the very fact that a principal made a request is interesting and may contain useful information based on the context in which the request was made. To the best of our knowledge, our sharing framework is the first systematic approach for expressing policies with alternate actions.

There are two broad categories of alternate actions that we consider. First, if a requestor is denied access, the sharing control function determines whether some other principal (who is related to the requestor and his mission) is authorized and should be granted access, even if that principal did not request it. For instance, if Joe requests read-access to a data object but is denied, the sharing control system would check whether the data is critical to Joe's current mission, and if so, would send the data to another principal (e.g., Joe's manager) who is related to Joe's role in the mission and who is authorized to access the data. A filtering mechanism ensures that such sharing does not impose an undue burden on either the network or any individual principal.

Second, if a requestor is denied access, the sharing control function considers whether this might be due to the access decision function not having adequate privileges or information to make the decision, and determines whether other authorities may be able to make more informed access decisions. If so, the original request (and optionally the data object itself) is forwarded to those authorities for further handling. For instance, some requests may be denied since an individual access decision function is not authorized to receive all the credentials relevant to making the access decisions. Trust negotiation policies and mechanisms have been proposed to address this situation, but there are many possible scenarios where trust negotiation fails [13]. We observe that trusted intermediaries (i.e., trusted third parties) may make more informed access decisions in such cases. Our policy language includes several such options for redirecting access requests.

In this paper, we present a formal sharing policy language for expressing policies with the above categories of alternate actions. Our language is based on the Flexible Authorization Framework (FAF) [8] which is a general, logic-based access control framework. A sharing policy includes both a

SACMAT'06, June 7–9, 2006, Lake Tahoe, California, USA.

conventional access control policy and a *redirection policy*. A sharing control system enforces a sharing policy by first applying its access control policy to an access request. If the request is denied by the access control policy, then the redirection policy is evaluated to determine a set of alternate actions to grant instead of the original requested action.

The remainder of this paper is organized as follows. Section 2 presents a running, motivational example and Section 3 describes desirable sharing actions. Section 4 summarizes the Flexibile Authorization Framework (FAF), while Section 5 presents our Sharing Policy Language (SPL) that extends FAF. Section 6 describes related work and Section 7 concludes this paper.

2. MOTIVATING EXAMPLE

Throughout this paper, we shall use a prototypical example that exemplifies the information sharing problem. Consider a hypothetical situation where the Federal Bureau of Investigation (FBI) suspects that a terrorist cell is stockpiling hazardous material (e.g., chemical and biological agents) in a building. FBI agents are in contact with other agencies (e.g., the Centers for Disease Control and Prevention (CDC) and the United States Army Medical Research Institute of Infectious Diseases (USAMRIID)) that have detailed knowledge about the hazardous materials. A fire suddenly breaks out in a warehouse adjacent to the building, and local firefighters respond to the emergency. The fire grows rapidly and is about to spread to the adjacent building. The firefighters do not have adequate security clearances for terrorist threat information, and they may even be unauthorized to know general information about the specific hazardous materials suspected of being in the building. However, this information is relevant to their current mission and the information may save lives. The sharing infrastructure must determine how (and with whom) it should share this information so that: (i) missions are not compromised, and (ii) confidentiality is maintained both of the information and of the sources and methods by which the information was collected. Similar scenarios occur daily in numerous environments including military operations, law enforcement, counter-terrorism, border control, etc.

3. ACCESS REQUEST REDIRECTION

The motivating example highlights two key issues. First, while individual firefighters might not have adequate privileges to receive critical information, other principals (e.g., their fire chief or an FBI agent at the scene) might have the required privileges and might be in a position to influence the firefighting mission. Second, CDC and USAMRIID might not have adequate information to make informed access control decisions regarding local firefighters, whereas a regional counter-terrorism center or the local fire chief may be better situated to do so. These considerations motivate several alternate actions that the system might grant when a principal's information access request is denied:

Data Redirection: Suppose that a principal P makes a request to read some data object O but the access control policy is unable to authorize the action. There is some other principal Q (or multiple principals Q_i) which is authorized to read O. Further, P and Q are related such that providing O to Q instead of P will yield a sufficient utility gain. A redirection policy might grant the push action of sending O to Q in response to P's read access request. Note that this assumes an environment with both push and pull style data access methods, strong relationships between principals, and some measure of the utility gained by one principal when data objects are sent to certain other related principals. Additionally, the data recipient Q is assumed to make full use of the data pushed to it to achieve the expected utility gain for the transaction.

Data object redirection results in indirect information flows, i.e. a provider cannot directly grant a task-related request from Alice, but it provides the data instead to Bob who works with Alice on the task. This sort of action is desirable in situations where the requestor and data provider are in separate organizations. The provider will likely not have sufficient context to make decisions about every member of the consumer organization, but may have some direct knowledge of a certain subset of members of the other organization and some information about its structure.

Two practical considerations arise from data redirection that an implementation ought to address: 1) request originators may receive some notification indicating when redirection was used to fulfill requests; and 2) some filtering or rate-limiting mechanism must exist to prevent overloading recipients with redirections and creating a denial of service situation. Any response notifications delivered to requestors must be subject to concerns of leaking information about the existence, sensitivity, and contents of data objects as well as the privileges of other principals.

Request Redirection: A principal P submits a request to perform an action a on data object O and the system makes an authorization request to the access decision function α. α is unable to grant P's request, but a redirection policy sends the request to access decision function β instead. β's access control policy might either directly grant the requested action, or else grant the action of sending α additional information which might enable α to grant the requested action. In the former case, β serves as an "override authority", i.e., it overrides α's access control policy. Note that this applies to decentralized systems with many principals and many access decision authorities, each with limited knowledge. Further, it requires that α has knowledge of other decision authorities, including relations between decision authorities and objects, and between decision authorities and principals.

Redirection of Data and Request: This is a generalization and combination of the data-object and request redirection schemes. A principal P makes a request for action a on data object O to decision authority α. α is a responsible authority for the data O but does not have the context to decide whether to grant P's request. α is aware of several trusted intermediary decision authorities or has the means to discover them. A *trusted intermediary* is an especially trusted and knowledgeable decision authority to whom other decision authorities can delegate responsibility over a particular request and associated data object. One or more trusted intermediaries that are authorized to

access the object O and have the most appropriate context for making decisions about P are selected. Then, the data object O and the original request are forwarded to the selected trusted intermediaries who may eventually grant access to the object to P. This assumes a distributed network of principals and decision authorities in which the participants in any particular request/decision cycle may not have the proper context and relations to complete the transaction. It requires that decision authorities trust certain other decision authorities as being trusted intermediaries.

Other alternate actions include temporarily elevating a principal's privileges by 'reading in' the principal to the policies and penalties associated with the information, and downgrading a sanitized version of the information so that it can be released to the principal. Both involve manual processes that require complex risk management decisions and they are used primarily for information with short-lived sensitivity. In the remainder of this paper, we focus on redirection actions and present a general logic-based framework for expressing policies regarding redirection actions.

4. FLEXIBLE AUTHORIZATION FRAMEWORK

Our sharing control policy language is based on the Flexible Authorization Framework (FAF) [9, 10, 8]. FAF is a logic-based framework for managing access to data by users. The framework permits system administrators to specify multiple access control policies that can be enforced within a single system. FAF provides a data system that consists of users, groups, and objects, together with the roles they may play, the access modes they may use, and three classification hierarchies (user-group, object-type, and role hierarchies). Our summary of FAF is adapted from [15].

Authorization subjects (AS) denote those entities for which accesses are authorized (users, groups, and roles), and authorization objects (AO) denote those entities on which accesses are authorized (objects, types, and roles). Access modes (A) are actions that authorization subjects wish to execute on authorization objects. An authorization is a triple $(o, s, \langle sign \rangle a)$ where $o \in AO$, $s \in AS$, "sign" is either "+" or "−", and $a \in A$. The triple $(o, s, +a)$ means that subject s is authorized to execute action a on object o, while $(o, s, -a)$ means that s cannot execute a on o. The set of all authorizations is called AUTH, while $AUTH^+$ and $AUTH^-$ denote, respectively, the sets of positive and negative authorizations.

Formally, FAF's Data System (DS) is the five-tuple $\langle Rel, OTH, UGH, RH, A \rangle$ where:

- Rel is a set of (first order) predicates. In particular, it contains the unary predicate symbols `isuser`, `isgroup`, `isrole`, `isobject`, and `istype`, which are true if their argument is a user identifier, group identifier, role, object identifier, and object type, respectively. These predicates define the sets of users (U), groups (G), roles (R), objects (Obj), and object types (T), respectively. Rel also includes predicates which define relations between users, groups, roles, and objects. The definition of these will typically be specific to a given environment and policy. One such predicate is defined in Section 5 as an integral part of our sharing policy language.
- OTH = $(Obj, T, \leq_{OT})$ is an object-type hierarchy where Obj is a set of object identifiers, T is a set of object types, and $\leq_{OT}$ is a partial order such that for all $o \in Obj$ and $ot \in T$, $o \leq_{OT} ot$ iff o is an object of type ot.
- UGH = $(U, G, \leq_U G)$ is a user-group hierarchy where U is a set of user identifiers, G is a set of group identifiers, and $\leq_{UG}$ is a partial order such that for all $u \in U$ and $g \in G$, $u \leq_{UG} g$ iff u is a member of group g.
- RH = $(\phi, R, \leq_R)$ is a role hierarchy where R is a set of roles, and $\leq_R$ is a partial order such that for all $x, y \in R$, $x \leq_R y$ iff x is a specialization of y.
- A is a set of access modes. Access modes are application specific, and may include, for instance, read, write, etc.

FAF uses a locally stratified logic programming language, Authorization Specification Language (ASL), to specify the set of authorizations (AUTH). ASL defines the following predicate symbols:

- `cando`$(s, o, \langle sign \rangle a)$: represents particular authorizations directly defined by a system administrator.
- `dercando`$(s, o, \langle sign \rangle a)$: represents authorizations derived through propagation policy rules.
- `do`$(s, o, \langle sign \rangle a)$: represents the affirmative action of granting a specific authorization, rather than the ability to grant it as `cando` and `dercando` provide. The `do` predicate is used in conflict resolution and decision rules to resolve multiple cando and dercando terms that might apply to a particular request.
- `done`(s, o, r, a, t) : represents that subject s playing role r has executed action a on object o at time t.
- `over`$_{AS}(o1, o2, s, \langle sign \rangle a)$: represents an overriding policy.
- `over`$_{AS}(o, s1, s2, \langle sign \rangle a)$: represents an overriding policy.
- `error` : a propositional symbol that represents violation of integrity constraints.
- `in`(x, y, H) : represents that $x \leq y$ in hierarchy H.

FAF specifications consist of propagation policies that specify basic authorization facts as well as rules to derive additional authorizations, conflict resolution policies that eliminate contradictory authorizations, decision policies that ensure the completeness of authorizations, and integrity policies that check integrity constraints. Propagation policies consist of authorization rules of the form:

$$cando(s, o, a) \longleftarrow L_1 \& \ldots \& L_n$$

where $o \in AO$, $s \in AS$, $a \in A$, and each L_i is a done, hie-, or a rel- literal; and

$$dercando(s, o, a) \longleftarrow L_1 \& \ldots \& L_n$$

where $o \in AO$, $s \in AS$, $a \in A$, and each L_i is a cando, dercando, done, hie-, or a rel- literal. Conflict resolution policies use the override predicates `over` to resolve contradictory

(i.e., both positive and negative) authorizations, while integrity policies remove authorizations that violate integrity constraints. Then, the authorization set AUTH consists of a consistent subset of the derivable cando and dercando literals that satisfy all integrity constraints.

5. SPL: SHARING POLICY LANGUAGE

The Sharing Policy Language (SPL) extends FAF with several new hierarchies, predicates, and rules. Conceptually, sharing policies reside in a new layer over FAF access control policies. Requests are processed through the access-control and sharing policy layers as follows:

1. A request is received
2. The request is evaluated against the access-control policy
3. If the request is directly granted by the access-control policy, according to the standard FAF method, evaluation completes; otherwise the request is passed to the sharing policy module.
4. The request is evaluated against the sharing policy rule set possibly resulting in a direct grant of access, one or more redirections, or a denial if no other recourse is available.

Note that a sharing policy is evaluated only when the underlying FAF access control policy denies a request. So by default, in sharing policies, requests imply both that a principal has made the request and that the access control policy was unable to grant it. We also make the materialized view of the underlying FAF access control policy available to a sharing policy.

Much like FAF, the sharing data system consists of users, groups, objects, roles, and hierarchies. The concept of *purpose* or *missions* is added to requests in the system as an extra data point for making grant decisions in response to requests. The set of mission purposes will be known as **MP**. A mission purpose is distinct from both groups and roles in that it encodes the context in which a request is made or the cause of that request. On the other hand, groups and roles encode information about users identity or privileges, not their intent.

While FAF policies operate in isolation, i.e. there is a single access control policy and a single enforcer of that policy in the data system, sharing and redirection imply multiple entities implementing multiple sharing policies. We will term an implementer of a sharing policy a *decision point*, and the set of all decision points will be known as **DP**. A certain subset, $\mathbf{TI} \subseteq \mathbf{DP}$, of decision points may be considered trusted intermediaries. A *trusted intermediary* is a decision authority that is not necessarily responsible for any data objects itself, but is rather tasked especially to facilitate data sharing. These nodes are strategically picked such that they have a useful intersection between the set of data objects that can be granted to the intermediary and the group of principals the intermediary knows well enough to make decisions for. Since we seek to share data objects with decision points in addition to the normal FAF authentication subjects, we must extend the definition of the authentication subject set to be $\mathbf{AS} = \mathbf{U} \cup \mathbf{G} \cup \mathbf{R} \cup \mathbf{DP}$.

5.1 Hierarchies and Relations for Sharing

A number of new hierarchies and relation structures are necessary to describe the various redirection scenarios outlined above. The redirection of data objects to decision points requires knowledge of relationships among decision points, the data objects they hold responsibility over, and the users that reside within their domains. This entails defining hierarchies relating authentication objects to decision points (ODPH) and authentication subjects to decision points (SDPH) where:

- ODPH = $(AO, DP, \leq_{ODP})$ where AO is the set of authorization objects Obj $\cup$ T $\cup$ R, DP is the set of decision points, and $\leq_{ODP}$ is a partial order such that for all $o \in AO$ and $d \in DP$, $o \leq_{ODP} d$ iff d has the authority to decide access requests for the object o.
- SDPH = $(AS, DP \leq_{SDP})$ where AS is the set of authorization subjects U $\cup$ G $\cup$ R $\cup$ DP, DP is the set of decision points, and $\leq_{SDP}$ is a partial order such that for all $s \in AS$ and $d \in DP$, $s \leq_{SDP} d$ iff d has the authority to decide access requests initiated by subject s.

Redirection of data objects necessitates describing relations between authentication subjects. These relations will be modeled as a directed graph of authentication subjects (ASG) where:

- ASG = $(AS, \preceq_{AS})$ where AS is the authorization subject set and $\preceq_{AS}$ is a binary relation for all $s, t \in AS$. $s \preceq_{AS} t$ iff t is related to s in such a way that fulfilling a request, made by s, by positively responding to t results in a net utility gain, in the economic sense, on the part of s.

An example of such a subject relation graph, which we will use later, appears in strongly hierarchical organizations, such as the military. Each member of a *chain of command* in such an organization, other than the highest ranking, has a superior that issues them orders or tasking. Assuming at least the root of this authority tree has sufficient privileges to successfully request some subset $X \subseteq AO$ of objects, then any subordinate member of the chain of command can request $x \in X$ with the result of at least one superior receiving x. Subsequently, the recipient of x is guaranteed to be in a position to issue orders based on x that will eventually have an effect on the original requesting subject. Formally, the relation $\prec_{COC}$ is defined as a strict partial ordering on AS such that for all $p, q \in AS$, $p \prec_{COC} q$ iff p is a subordinate of q in the tree-structured ASG. We will refer to an ASG that uses the $\prec_{COC}$ relation as COC.

Missions are organized from a number of individual users, groups, and roles. For example, a first-responder mission may consist of various individuals, groups representing involved organizations, and roles describing the responsibilities and authority of those involved in the mission. This structure can be represented by a hierarchy which we will call the mission-purpose hierarchy (MPH):

- MPH = $(MM, MP, \leq_{MM})$ where MM = $U \cup G \cup R$, MP is the set of mission-purposes, and $\leq_{MM}$ is a binary relation such that for all mission members $m \in MM$ and missions $p \in MP$, $m \leq_{MM} p$ iff the user, group, or role m is included in the mission p.

Finally, relations between decision points are modeled. Like the inter-subject relation above, a directed graph is used, which we will term DPG.

- DPG = $(DP, \Rightarrow_{DP})$ where DP is the set of decision points and $\Rightarrow_{DP}$ is a binary relation over all $x, y \in DP$. $x \Rightarrow_{DP} y$ iff x is willing to perform redirections with decision point y as the target.

Depending upon how the edges of the DPG graph are assigned, it can be used to represent concepts such as visibility between decision points or trust between decision points, among other possibilities.

5.2 Predicate Extensions for Sharing

The sharing policy language defines several logical predicate symbols in addition to those FAF provides:

- $\texttt{request}(o, p, m, a)$: where $o \in \mathbf{AO}$, $p \in \mathbf{AS}$, and a is an action. True when a request by principal p to perform action a on object o for the purposes of mission m has been received.
- $\texttt{grant}(o, p, m, a)$: where $o \in \mathbf{AO}$, $p \in \mathbf{AS}$, $m \in MP$ and a is an action. True if the materialized FAF access control policy authorizes p to perform action a on object o, and if the predicate $\texttt{in}(p, m, MPH)$ is true. This predicate is used to describe situations that require a request with certain parameters to be grantable before conducting a redirection, no matter the way in which it was granted.
- $\texttt{redirect}_{\texttt{data}}(o, q, m, a)$: Has the same arguments as the request and grant predicates. This predicate represents an outcome to a sharing policy evaluation for which $\texttt{request}(o, p, m, a)$ is true. The statement $\texttt{redirect}_{\texttt{data}}(o, q, m, a)$ dictates that the data object o be sent to the principal q rather than the original requestor p, where $p \neq q$ and $a =$ 'read'.
- $\texttt{redirect}_{\texttt{req}}(\beta, \langle o, p, m, a\rangle)$: where $\beta \in \mathbf{DP}$ and the second argument is a tuple of $o \in \mathbf{AO}$, $p \in \mathbf{AS}$, $m \in MP$, and action a, representing a data access request. $\texttt{redirect}_{\texttt{req}}(b, \langle o, p, m, a\rangle)$ represents an outcome to a sharing policy evaluation for which $\texttt{request}(o, p, m, a)$ is true. The request is forwarded to a decision point β from α, the decision authority implementing the forwarding rule. $\alpha \neq \beta$.
- $\texttt{redirect}_{\texttt{ti}}(\beta, \langle o, p, m, a\rangle, o)$: The first and second arguments are the same as those of $\texttt{redirect}_{\texttt{req}}$, and the third argument is $o \in \mathbf{AO}$. The predicate represents an outcome to a sharing policy evaluation of the request represented by the tuple $\langle o, p, m, a\rangle$, indicating that both the request and the object of the request be forwarded from this decision point, α, to the decision point β. β will then decide whether to grant access to the requested object o to the requesting subject p.
- $\texttt{trusted-intermediary}(\beta)$: where $\beta \in \mathbf{TI}$. True when β is allowed to perform the role of a *trusted intermediary*.
- $\text{says}(\beta, L)$: This predicate asserts that the decision point β has asserted the statement L, where L is any statement in the sharing policy language. If another decision point α trusts β to make such statements (e.g., if $\alpha \Rightarrow_{DP} \beta$), then α's sharing policy can derive L.

5.3 Sharing Rules

A number of sharing rules can be expressed given the above defined predicates. Note that `request` predicates represent access requests, while `grant` predicates represent the materialized view of a FAF policy. The sharing rules described below are only applied when a request is not granted by the FAF policy and they attempt to redirect the request itself, the requested data, or both.

5.3.1 Data Redirection Rules

Data redirection rules are triggered by denied requests and proactively push data to authorized principals that are related to the original requestors.

DEFINITION 5.3.1. Data Redirection Rules *are specified as follows:*

$$\texttt{redirect}_{\texttt{data}}(o, q, m, read) \longleftarrow \texttt{request}(o, p, m, read) \wedge \texttt{grant}(o, q, m, read) \wedge L_1 \wedge \cdots \wedge L_n$$

where o is an object identifier, p and q are authorization subjects, and m is a mission identifier. The L_i are hierarchy or relation literals that establish a connection between p and q for mission m.

Data redirection rules must encode useful relationships between principals. In particular, data should only be redirected to principals that: a) are authorized to read the data themselves; and b) are mission-related to the original requestors so that they can propagate some (or all) of the benefit of the data to the mission without breaching confidentiality. Inter-user relations that fulfill the necessary conditions for data redirection can be found in well-defined, hierarchical organizations and mission tasking structures. Note that data redirection rules require that the underlying data sharing network that implements the sharing policy is capable of unsolicited, push-style messaging.

EXAMPLE 5.3.1. *Consider the firefighter example of Section 2. Suppose that firefighter Joe is part of the mission FM to fight the fire in the warehouse. Fire Chief Bob is the commander of this mission as well as Joe's superior. Joe is a member of the firefighter role ff and Bob is a member of the fire-chief role fc. Joe requests information on any hazardous material known to be in the warehouse or adjacent buildings, and an information broker matches his request with an object bc.*

$$\texttt{request}(bc, ff, FM, read).$$

However, bc's access decision function denies this request. On the other hand, Bob does have permission to view the desired information.

$$\texttt{grant}(bc, fc, FM, read).$$

Joe's request may be partially fulfilled by sending the data to Bob instead. This can be specified via use of the chain-of-command relation for the fire department, which specifies Bob as Joe's superior:

$$\texttt{redirect}_{\texttt{data}}(bc, fc, FM, read) \longleftarrow request(bc, ff, FM, read) \wedge in(ff, fc, COC_{fd}) \wedge grant(bc, fc, FM, read)$$

or by leveraging mission membership, which relates Joe and Bob via their participation in the mission, FM:

$$\begin{aligned}\texttt{redirect}_{\texttt{data}}(bc, fc, FM, read) \longleftarrow \\ \texttt{request}(bc, ff, FM, read) \\ \wedge\, \texttt{in}(ff, FM, MPH) \\ \wedge\, \texttt{in}(fc, FM, MPH) \\ \wedge\, \texttt{grant}(bc, fc, FM, read)\end{aligned}$$

5.3.2 Request Redirection Rules

If a decision point's access control policy does not grant an access request, the decision point might determine that another decision point is better suited to make the access decision. For instance, that other decision point might have a different version of the object (with a different authorization policy), or it may be authorized to receive additional credentials that are required to grant access. Request redirection rules capture such situations and specify that an access request be redirected to another decision point.

DEFINITION 5.3.2. Request Redirection Rules *take the form:*

$$\begin{aligned}\texttt{redirect}_{\texttt{req}}(\beta, \langle o, p, m, a\rangle) \longleftarrow \texttt{request}(o, p, m, a) \\ \wedge\, L_1 \wedge \cdots \wedge L_n\end{aligned}$$

where o is an object identifier, a is an action, p is an authorization subject, m is a mission identifier, and β is a remote decision authority different from the decision authority α implementing this rule. The L_i terms are hierarchy or relation literals that represent relations between the decision point β and either decision point α or principal p.

For request redirection to be feasible, the set of decision authorities must fulfill certain properties. First, there must be an intersection between the subset, $\mathbf{DP_S}$, of authorities that can make decisions about requests from a particular principal and the subset, $\mathbf{DP_O}$, of authorities that can make decisions concerning a particular data object. We will label this intersection as the subset $\mathbf{DP_I} = \mathbf{DP_S} \cap \mathbf{DP_O}$. Second, there must be a mechanism for any given decision authority to find a member of $\mathbf{DP_I}$ for a particular request. Finally, there must be a mechanism to prevent request redirection loops.

EXAMPLE 5.3.2. *Continuing with the firefighter example, suppose that the fire departments server had a local limited cache of terrorist threat and biohazard information and also had a local access decision point DP_{FD}. Suppose also that the authoritative repository of threat information is on an FBI server with access decision point DP_{FBI}. If a firefighter's data access request cannot be granted by the local decision point, the request can still be fulfilled by a request redirection rule implemented at DP_{FD}:*

$$\begin{aligned}\texttt{redirect}_{\texttt{req}}(DP_{FBI}, \langle bc, ff, FM, read\rangle) \longleftarrow \\ \texttt{request}(bc, ff, FM, read) \\ \wedge\, \texttt{in}(bc, DP_{FBI}, ODPH). \\ \texttt{request}(bc, ff, FM, read).\end{aligned}$$

Another type of request redirection rule in the sharing policy language embodies the concept of delegation of authorization authority from one decision point to another. Delegation allows for the separation of policy decisions from policy enforcement which can make the management of large-scale sharing policies more manageable. A logic-based trust management framework (e.g., Binder [5] or RT [11]) may be used for such derivations. We present one such rule here:

DEFINITION 5.3.3. *A* Decision Delegation Rule *is a rule of the form:*

$$\begin{aligned}\texttt{grant}(o, p, m, a) \longleftarrow \texttt{request}(o, p, m, a) \\ \wedge\, \beta \text{ says } \texttt{grant}(o, p, m, a) \wedge L_i \wedge \cdots \wedge L_n\end{aligned}$$

where o, p, m, a, and β are members of the sets of data objects, principals, missions, actions, and decision authorities, respectively, and $n \geq 0$. The relation and hierarchy terms, L_i, define relations between β, o, p, m, a, and the decision authority, α, implementing this rule, that constitute reasons why α trusts β to make this decision for it, e.g. the logical representation of the result of some out-of-band trust establishment protocol.

The existence of a strong trust relationship between the decision points involved in a delegation is paramount. Trust can be formulated in two ways in the rule set, namely implicitly and explicitly. The very existence of an unconditional delegation rule implies that the delegate decision point is trusted implicitly; otherwise the rule would not have been inserted in the rule set. An out-of-band establishment of trust between decision points is necessary. Acceptance of a delegation may also depend on some explicitly stated conditions that, taken together, dictate a required level of trust in the delegate. The explicit conditions are represented by application-specific predicates symbolized by the L_i terms in the rule formulation above. For example:

EXAMPLE 5.3.3. *Consider the following example decision redirection rules:*

$$\begin{aligned}\texttt{grant}(\texttt{SecretDoc}, p, m, a) \longleftarrow \\ \texttt{Pres} \text{ says } \texttt{grant}(\texttt{SecretDoc}, p, m, a) \\ \texttt{grant}(o, p, m, a) \longleftarrow \beta \text{ says } \texttt{grant}(o, p, m, a) \\ \wedge\, \texttt{certified}(\beta)\end{aligned}$$

The first rule is an explicit override of the local sharing access policy for the `SecretDoc` *object. It delegates access control to a remote authority, in this case the* `President`*. The very act of adding this rule to a sharing policy rule-set means that the system security officer of this decision point has declared the* `President` *decision authority as trusted. The second rule declares a remote authority β to be trusted, and delegates decisions to it, if β has been verified and accredited by some certifying authority that the local security administrator trusts.*

5.3.3 Data and Request Redirection Rule

The third type of redirection rule combines aspects of forwarding data objects and forwarding requests. This double forwarding makes use of special decision authorities called *trusted intermediaries.*

DEFINITION 5.3.4. Trusted Intermediary Rules *have the following form:*

$$\begin{aligned}\texttt{redirect}_{\texttt{ti}}(\beta, \langle o, p, m, a\rangle, o) \longleftarrow\ & \texttt{request}(o, p, m, a)\\ & \wedge \texttt{grant}(o, \beta, \texttt{read})\\ & \wedge \texttt{trusted-intermediary}(\beta)\\ & \wedge L_1 \wedge \cdots \wedge L_n\end{aligned}$$

where o, p, m, a, and β are members of the sets of data objects, principals, missions, actions, and decision authorities, respectively, and $n \geq 0$. Decision point β must be considered a suitable trusted intermediary *from the point of view of decision point α, the node implementing this policy rule. Further, β must be authorized to receive the data-object o. The L_i terms represent relations between the decision authorities α and β, the principal p, the mission m, and the data object o.*

EXAMPLE 5.3.4. *Let us revisit the firefighter example and assume that the FBI has no knowledge of local firefighting missions or the local firefighters participating in those missions. However, a trust relationship is established between the FBI decision point and the fire department decision point such that DP_{FBI} considers DP_{FD} to be a trusted intermediary and grants data access to DP_{FD}:*

$$\begin{aligned}&DP_{FBI} : \texttt{trusted-intermediary}(DP_{FD}).\\ &DP_{FBI} : \texttt{grant}(bc, DP_{FD}, FM, read).\end{aligned}$$

The fire department decision point contains rules to redirect requests it receives from firefighters for the building contents data to DP_{FBI}:

$$\begin{aligned}DP_{FD} : \texttt{redirect}_{\texttt{req}}(DP_{FBI}, \langle bc, ff, FM, read\rangle) \longleftarrow\ & \texttt{request}(bc, ff, FM, read)\\ & \wedge \texttt{in}(bc, DP_{FBI}, ODPH).\end{aligned}$$

The FBI decision point is configured to trust DP_{FD} to make the appropriate access decisions for firefighters so its rule set contains a trusted intermediary rule. Note that the FBI returns the current state of the data object back to the fire department:

$$\begin{aligned}DP_{FBI} : \texttt{redirect}_{\texttt{ti}}(DP_{FD}, \langle bc, ff, FM, read\rangle, bc.data) \longleftarrow\ & \texttt{request}(bc, ff, FM, read)\\ & \wedge \texttt{grant}(bc, DP_{FD}, FM, read)\\ & \wedge \texttt{trusted-intermediary}(DP_{FD})\\ & \wedge \texttt{in}(ff, DP_{FD}, SDPH).\end{aligned}$$

Finally, the fire department decision point contains access rules that state the data should be redirected to the chief instead of the individual fire fighter who does not have access:

$$\begin{aligned}&DP_{FD} : \texttt{grant}(bc, fc, FM, read).\\ &DP_{FD} : \texttt{redirect}_{\texttt{data}}(bc, fc, FM, read) \longleftarrow\\ &\qquad \texttt{redirect}_{\texttt{ti}}(DP_{FD}, \langle bc, ff, FM, read\rangle, bc.data)\\ &\qquad \wedge \texttt{in}(ff, FM, MPH)\\ &\qquad \wedge \texttt{in}(fc, FM, MPH)\\ &\qquad \wedge \texttt{grant}(bc, fc, FM, read).\end{aligned}$$

The data flow proceeds as follows:

1. *a request is submitted to DP_{FD}:* $\texttt{request}(bc, ff, FM, read)$
2. *the request is redirected to DP_{FBI}*
3. *DP_{FBI} sends the request and data back to DP_{FD} for a final decision*
4. *DP_{FD} evaluates the request against its rule set and redirects the data to the fire chief*

5.4 Redirection Filtering

A rule set containing many grants and redirection rules will not necessarily yield a single action for a given input request. In fact, a single redirection rule may yield a large set of possible results. Below we describe methodologies for choosing the most appropriate action or set of actions in response to a request. A sharing policy, by definition, will do its best to maximally share data. In pursuit of this goal, a large number of possible outcomes may be generated from a given request. While this maximizes the number of paths by which information can flow, and thus maximizing the probability that useful data objects will find their way to those users who need them most, it can lead to a great deal of unnecessary transmission of data, which is wasteful and taxing to both the distributed data system and its users.

FAF contains mechanisms for resolving conflicting authorization results. These are necessary because both positive and negative authorizations can be specified or derived which may cause contradictions. Sharing policies only deal in the positive but yield a set of results rather than a discrete decision. The analogous problem to conflict resolution is one of filtering this result set to maintain scalability in the system and relevance of the data that is shared. For example, a data redirection rule making use of the chain-of-command relation will necessarily result in a set of potential recipients that includes the entirety of the chain of command above the original requestor. That is, if a low ranking soldier were to request some informational document in such a system, the data might be returned to every single individual from his or her immediate superior all the way up to the commanding general. Obviously this situation is not scalable and will cause more harm than good as users become overwhelmed with unsolicited data. The optimal result would be to forward the document to one or a handful of the individuals immediately above the requestor in the chain of command.

A selection strategy must satisfy the following properties:

1. Maximize the value of sharing data
2. Minimize the overloading of any particular user

The second goal will often dominate and constrain the pervasiveness of sharing. This effect will be especially apparent when the sharing decision path is not automated end-to-end from the information provider to the initial requesting consumer, as is the case with redirection of data-object rules. We now propose a basic mechanism that fulfills the necessary properties.

Assume that the set of possible redirections resulting from the sharing policy can be preferentially ordered in some fashion. Incomparable redirections are considered to be equally preferable. The result is a set orderings that includes all possible redirections generated by the sharing policy. The most preferred redirections from all the orderings are then executed. The measure of value or preference in comparable redirections is assumed to be additive, i.e. of two comparable redirections, the more preferred will confer all of the

value of the less preferred and then some. A pair of incomparable redirections may provide value to different, disjoint groups of data consumers. Therefore, both redirections must be done to achieve maximal value. This scheme fulfills the first property of a selection strategy.

While the above scheme maximizes the benefit of data sharing, it will not prevent degenerate cases which lead to overloads and bottlenecks. To fulfill the second property of a selection strategy, the targets of redirection must be provided with a mechanism to limit their intake of data-objects to an acceptable amount. The described data-sharing environment is a message-based push-style system much like email, and so the same filtering strategies that apply to managing large volumes of email are applicable here. Let each sharing system node that accepts redirections define a filtering list of either redirections they are willing to accept, a white-list approach, or a black-list of redirections they will not accept. In either case, the filter is advertised such that a decision point initiating a set of redirections per the algorithm described above can discover and apply the filter. The filter is iteratively applied to each ordered list in the set of ordered redirections. The most preferred redirection of each list that is passed by the filter is then executed. The complete algorithm proceeds as follows:

```
Let R be the set of possible redirections
Let p be the preference relation
Let F_α be the filter for decision point α
Let RO be the set of ordered lists resulting from sorting
the members of R using p
for all elements RO_i of RO do
  while RO_i ≠ ∅ do
    r ← head element of RO_i
    α ← the target node of the redirection r
    if r is allowable by F_α then
      execute the redirection r
      RO_i ← ∅
    else
      RO_i ← RO_i − r
    end if
  end while
end for
```

6. RELATED WORK

Numerous access control models (e.g., RBAC, trust management, etc.) have been proposed in the literature. We are unaware of any prior access control model that grants actions other than those requested by principals. In contrast, our sharing framework extends the Flexible Authorization Framework [8] with a redirection policy layer that determines alternate actions to grant if a requested action cannot be granted.

Automated trust negotiation [14] refers to techniques that permit principals to bootstrap mutual trust by exchanging credentials iteratively. Push authorization [12] has been proposed as a way of actively pushing credentials to access decision points that might need them. Our redirection policies also involve active ("push") actions. However, our actions involve pushing data objects and requests, not credentials.

We have used the term *mission* to refer to the context within which a request is made. A mission includes a set of mission goals, mission participants (principals) and their roles and relationships, and the current mission state including organizational structure and workflow state. This is similar to the notion of "context" in Team-Based Access Control [7] and "purpose" in Purpose-Based Access Control [3]. In this paper, we have not used the structure of contexts but have only referred to the relationships between principals, data objects, and contexts.

Publish-subscribe systems [6] are event-based systems in which providers publish data while consumers subscribe to published data with specified attributes. In such systems, publish and subscribe actions are matched and data objects routed from publishers to appropriate consumers. Access control mechanisms for publish/subscribe systems typically control the publish and subscribe actions. Only principals with specific roles or other attributes are authorized to execute those actions. This involves standard access control models (e.g., RBAC) applied to a push-style data dissemination system [2, 1]. In contrast, our approach proactively routes data objects and access requests based on requests that are denied by access control policies.

7. CONCLUSIONS AND FUTURE WORK

If an access decision function denies a principal's data access request, most current systems will reject that request and take no further action. In this paper, we have presented a general sharing control framework for expressing access policies that, before denying unauthorized requests, determine other possible actions to take. These actions include sending the requested data to authorized principals which are related to the original requestor, sending the request to an overriding authority, and sending both the request and the requested data to a trusted intermediary. Our investigations have revealed that such actions are performed routinely, in an ad hoc manner, by humans and organizations in mission-critical situations. To the best of our knowledge, our sharing framework is the first systematic approach for specifying policies that incorporate such alternate actions.

Our future work includes the implementation of a redirection mechanism, based on our sharing policy language, within a data sharing system. In this paper, we have focussed on redirection policies for requests that are denied by access control policies; however, the concept of redirection is equally applicable to successful requests and we are exploring this further. Another direction for future work is to explore other alternate actions such as returning a similar, less sensitive version of a data object (e.g., an older version or a sanitized version of the object) in response to a request rather than denying the request.

8. REFERENCES

[1] J. Bacon, D. M. Eyers, K. Moody, and L. Pesonen. Securing publish/subscribe for multi-domain systems. In *Middleware 2005, ACM/IFIP/USENIX, 6th International Middleware Conference*, volume 3790 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2005.

[2] A. Belokosztolszki, D. M. Eyers, P. R. Pietzuch, J. Bacon, and K. Moody. Role-based access control for publish/subscribe middleware architectures. In *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems, DEBS 2003*, 2003.

[3] J.-W. Byun, E. Bertino, and N. Li. Purpose based access control of complex data for privacy protection.

In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 102–110, New York, NY, USA, 2005. ACM Press.

[4] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder policy specification language. In *POLICY '01: Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, pages 18–38, London, UK, 2001. Springer-Verlag.

[5] J. DeTreville. Binder, a logic-based security language. In *IEEE Symposium on Security and Privacy*, pages 105–113, 2002.

[6] P. T. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[7] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible team-based access control using contexts. In *SACMAT*, pages 21–27, 2001.

[8] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.

[9] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *IEEE Symposium on Security and Privacy*, pages 31–42, 1997.

[10] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 474–485, 1997.

[11] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[12] T. B. Quillinan and S. N. Foley. Synchronisation in trust management using push authorisation. In *First International Workshop on Security and Trust Management (STM'05)*. Elsevier Science B. V., September 2005.

[13] W. H. Winsborough and N. Li. Safety in automated trust negotiation. In *IEEE Symposium on Security and Privacy*, pages 147–160, 2004.

[14] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition (DISCEX '2000)*, volume 1, pages 88–102. IEEE Press, 2000.

[15] N. Zannone, S. Jajodia, F. Massacci, and D. Wijesekera. Maintaining privacy on derived objects. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 10–19, New York, NY, USA, 2005. ACM Press.