On Classifying Access Control Implementations for Distributed Systems

Kevin Kane and James C. Browne
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233 USA
kane@cs.utexas.edu, browne@cs.utexas.edu

ABSTRACT

This paper presents a classification of implementations of access control systems based on a lattice taxonomy where the axes are properties of the implementation. The current taxonomy has six axes representing: partitioning of control over sharing of access control credentials, distribution of the state relevant to access control decisions, fidelity of policy enforcement, the identity resolution mechanism, local versus centralized decisions, and static or adaptive trust management. Analysis of implemented systems in terms of these properties sheds insight on tradeoffs between performance, scalability and potential vulnerability to specified attacks. The taxonomy reveals that distributed systems for several points on the lattice with interesting access control characteristics have not yet been implemented. The relationship of this classification to conventional classifications by type (for instance, role-based access control or mandatory access control) and mechanism (for instance, access control list or capabilities) is briefly discussed. Several implementations of access control are classified by their values for these properties. The roles of access control in formulation and operation of distributed systems are discussed.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access (e.g., hacking, phreaking)*

General Terms

Security, Design, Theory

Keywords

Access control, distributed systems, taxonomies, classification, analysis

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SACMAT'06, June 7–9, 2006, Lake Tahoe, California, USA.
Copyright 2006 ACM 1-59593-354-9/06/0006 ...$5.00.

1. INTRODUCTION

The rise of Internet based systems and systems with distributed control such as peer to peer systems has brought access control back to the front line of both theoretical and practical computer systems research.

An access control policy is typically a declarative specification of allowed accesses to resources[1]. An access control scheme, as defined by [19], is a state transition system in which access control decisions are specified as changes of state which conform to the stated access control policy in an appropriate representation such as an access control matrix. An access control implementation is a realization of a scheme on a networked system of services. Access control schemes are typically formulated under the assumption that decisions are made using complete and consistent state. This assumption may not be valid for a distributed system and is certainly not valid for distributed systems without central control, such as peer to peer networks. Therefore, implementations of access control for distributed systems may not faithfully conform to the access control scheme. There is a need to understand the trade-offs among fidelity of an implementation to a scheme, performance and scalability of the implementation and possible vulnerability of the implementation to attacks. This goal of this paper is to contribute to understanding these trade-offs.

Access control systems are commonly classified by the abstraction used to design the policy, such as user-based, role-based, or mandatory access control; and/or by an implementation mechanism such as access control lists or capabilities. These classifications do not differentiate between centralized and distributed implementations and do not offer insight into the relative robustness of different implementations in the context of distributed systems nor do they define the space of possible distributed implementations. This paper presents an analysis of the properties of implementations of access control, particularly for distributed systems. The classification and analysis is based on identification of properties of access control system implementations and constructing a lattice taxonomy where each axis represents a property and the points on the axis are determined by possible values of the property. The systems analyzed are those given as examples in Section 5: Akenti [18], dRBAC [11], CRISIS [2], Kraft-Schäfer [13], BitTorrent [7] and WPA [21]. The current taxonomy has six axes representing:

[1]Please refer to section 3 for more detailed definitions of access control policies, schemes, models, and implementations.

partitioning of control over sharing of access control credentials, distribution of the state relevant to access control decisions, fidelity of policy enforcement, the identity resolution mechanism, local versus centralized access control decisions and static or adaptive trust management. Each of these properties impacts the faithfulness with which an implementation implements a scheme and thus indirectly the vulnerability of the implementation to certain attacks. The current set of axes is not totally orthogonal but is still useful for classification, analysis of potential vulnerability to some forms of attacks, and suggestions for implementations with interesting properties which have not yet been analyzed or implemented.

The remainder of the paper is organized as follows. We discuss other classification efforts and related work in section 2. We present definitions for access control in section 3. We then present the six classifying axes, each representing a particular property of access control implementations, and a number of discrete points on each that represent design or algorithm choices in section 4. We then classify a number of existing implementations in section 5. We discuss these results and note some unoccupied space in section 6, and conclude with future work in section 7.

2. RELATED WORK

We divide related research into two categories: surveys and descriptions of access control policies and models, and comparison of expressiveness among schemes and models.

2.1 Surveys and Descriptions

di Vimercati, Paraboschi, and Samarati [8] survey current systems in the single-host case and their access control implementations. They list desirable goals for access control systems, and then discuss the implementation of access control in Linux, Windows, Database Management Systems, TCPD, the Apache web server, and Java 2.

Sandhu and Samarati [17] discuss the Access Control Matrix, and implementations of both access control lists (ACLs) and capabilities. This discussion focuses mostly on the single-system case. They describe Discretionary Access Control (DAC) and Mandatory Access Control (MAC), both official standards from the U. S. Department of Defense, and introduce role-based policies which later led to role-based access control models [16]. Sandhu revisits these topics in [15], further developing the role-based access control model and introducing the Task Based Access Control model.

These papers provide useful descriptions of the space of policies, models and schemes and mechanisms for implementation but assume complete faithfulness of implementations with policies and/or schemes and do not enable comparison of implementations with respect to performance, scalability or security properties.

2.2 Comparison of Expressiveness of Schemes and Models

Tripunitara and Li [19] give a theory for comparing the expressiveness of access control models in terms of *simulatability*. It is their definition of an access control scheme that we have used in this paper. The existence or non-existence of a simulatability relation between schemes shows the relationship between the relative expressiveness of schemes.

Bertino, et. al. [3] formulate a logical framework for comparing the expressiveness of access control models for data based management systems where each instance of the framework corresponds to a program in a logic programming language. They give a classification of models in terms of structural equivalence and in terms of reachability for a given state of a scheme and consistency among reachable states.

Comparisons of expressiveness among policies/models does not address differentiation among or comparison of possible distributed implementations of the policies and schemes which is the subject of this paper.

3. DEFINITIONS

3.1 System

A *networked system* is a set of nodes connected by a communications medium that allows bi-directional communication between any two nodes. We assume that the nodes are aware of each other and share a common model of interaction, and wish to offer services to one another in a way that enables attainment of common goals and does not put them at risk for attack. This method of acting to prevent or mitigate such harmful behavior is *access control*. We abstract human operators as nodes, which provide no services to other nodes. A *user* is a node that has been granted an authorization to access the services of another node. The *service* is the node providing those services. For the remainder of this paper, we will refer to these nodes as services, except in the particular case when a service is acting as a user, in which case it will be referred to as a user.

3.2 Access Control Policies, Schemes, and Implementations

The development of access control goes through three stages: formulation of a *policy*, representation of that policy as a *scheme*, and finally realization of that model in an *implementation*.

- An *access control policy* is a definition of how a system should provide or deny access which can range from an abstract statement like, "only users on this list should have access," or "only users who have given me service in the past should have access," to *policy languages* with executable (operational) semantics [4, 5, 6, 14].

- An *access control scheme* [19] is a state transition system $\langle \Gamma, Q, \vdash, \Psi \rangle$ that embodies the access control policy. Γ is a set of states, Q is a set of queries that include privileged requests considered by the system, $\vdash : \Gamma \times Q \rightarrow \{true, false\}$ is the entailment relation that determines whether a given query is true or not in a given state, and Ψ is the set of state-transition rules[2]. $\gamma \vdash q$ means that the query q is true in state γ, and $\gamma \nvdash q$ means that it is not. In particular, when q is a privileged request, this means the request is allowed in state γ.

- An *access control implementation* is the realization of such a scheme on actual hardware on a service i. In the distributed systems we consider here, each state $\gamma \in \Gamma$ is the global state of the entire system, made up of the *local states* $(s_1, s_2, \ldots, s_n)$ for each service

[2][19] also defines an access control model as a collection of schemes. This definition of an access control model is not relevant for the purposes of this paper.

in the system, where each $s_i \in S_i$, the set of all possible states for a particular service i. Global state may be replicated across multiple local states, and some of these replicas may become outdated. A local state is always dependent on the global state, but for brevity we will assume local states are in the context of the currently-mentioned global state. When more than one global state is mentioned, we will annotate the local states with their corresponding global state: for example, $s_{i(\gamma_1)}$ is the local state of service i in global state γ_1, and $s_{i(\gamma_2)}$ is the local state of service i in global state γ_2.

The implementation computes a function $\vDash$: $S_i \times R \rightarrow \{true, false\}$, where S_i is the set of local states of the service, and R is the set of specific requests being considered. We specifically note that $R \subseteq Q$: a specific request for service is always a type of query, but it is possible the scheme can answer more general kinds of queries. Without loss of generality, we also assume that a request considered by a service is one where the service is requested to perform a service, as services will not process queries unrelated to them. The goal of an implementation is to follow the scheme as closely as possible. Typically, the queries in a scheme are formulated so that they can be implemented perfectly, or nearly perfectly, if the relevant system state is complete and consistent. For some systems with distributed control, the nature of the system prevents perfect correspondence, which gives rise to this distinction between a scheme and its implementation.

3.3 Implementations versus Schemes

We now address the case where a scheme, which can consider the state of an entire networked system, and an implementation in a distributed system, which may have to make a decision on the basis of partial and/or inconsistent state, can differ. We say that a request r is *improperly serviced* if for a given service i, and its local state s_i in a particular global state γ, $\gamma \nvdash r$ but $s_i \vDash r$, and so the service grants the request when it should be denied.

Conversely, we say that a request is *improperly denied* when $\gamma \vdash r$ but $s_i \nvDash r$, and so the service denies the request when it should be granted. We call either of these situations *improper*. An improper request is *detectable* if the implementation can transition to a state such that the impropriety of the request becomes known. Such a request is *always detectable* if the implementation is guaranteed to transition to such a state.

3.4 Access Control Credential

An access control *credential* is a bitstring used in access control decisions. It is protected from undetectable modifications through a mechanism such as cryptography or kernel-imposed restrictions.

3.5 Lattice Taxonomy

We define a lattice to be a set of points on a set of axes. Each axis represents an enumerated type. An ordering metric can be defined for each axis. A lattice taxonomy classifies systems by assigning each system a value for each of the axes of the lattice. An ordered lattice taxonomy has one or more metrics which order points on the lattice globally.

4. CLASSIFICATION AXES

We now present the axes we have identified from access control implementations. We identified these axes by formulating implementation of access control as a workflow, identifying the steps in the workflow and then analyzing a number of access control implementations for networked systems, and extracting properties of the implementations of each step for these systems. The workflow formulation we used has the steps: (1) authentication of identity, (2) acquisition of system state information, (3) the access control decision process and (4) enforcement of the decision.

4.1 Control over Sharing of Access Credentials

The holder of an access control credential may be able to share access to a service by delegating that credential to another user while the service which granted the access credential may be able to revoke the access authorized by the credential. Consider two users i, j and a service k (with local states s_i, s_j, s_k respectively), and requests r_i that states "i requests service from k", and r_j that states "j requests service from k". Suppose there is a reachable global state γ such that $\gamma \vdash r_i$, but $\gamma \nvdash r_j$, and therefore $s_k \vDash r_i$ but $s_k \nvDash r_j$. Let C be the credential(s) used by i to make the request r_i, and let the statement "i delegates C to j" indicate the invocation of a state transition ψ such that s_j now contains C. Let $\gamma \mapsto_\psi \gamma_1$ denote ψ causes a transition from global state γ to γ_1.

An access control credential is:

1. *Delegatable*, if ψ also causes a change of state $\gamma \mapsto_\psi \gamma_1$ such that $\gamma_1 \vdash r_j$, and $s_{k(\gamma_1)} \vDash r_j$, meaning that such a delegation then permits r_j,
2. *Revocable*, if from γ where $\gamma \vdash r_j$, there exists a reachable state γ_2 such that $\gamma_2 \nvdash r_j \Rightarrow s_{k(\gamma_2)} \nvDash r_j$, meaning the implementation can support the revocation of an authorization, and
3. *Fine-grained revocable*, if from γ where i and j are both authorized, there exists a reachable state γ_2 such that $\gamma_2 \nvdash r_j$ but still $\gamma_2 \vdash r_i$, the implementation can ensure that $s_{k(\gamma_2)} \nvDash r_j$ while still $s_{k(\gamma_2)} \vDash r_i$. The absence of this third property means that i's authority must be revoked in order to revoke j's, assuming such authority can be revoked at all. This can only apply where property 2 applies.

We identify the following points on this axis, ordered by increasing amount of control given to the service:

1. **User control.** The user can delegate credentials at will once the service provides it, and that credential cannot be invalidated. Therefore, delegation causes a state change that grants access to the delegate j. Credentials are neither revocable nor fine-grained revocable.
2. **Shared control without recorded delegations.** Delegations do cause the state change, so (1) applies, but the service can transition to a state where the access is revoked. Users still can freely delegate their authorizations. The service is able to revoke that authorization and all delegations made of it. Access credentials are revocable but not fine-grained revocable.

3. **Shared control with recorded delegations.** The state change ψ caused by the delegation also causes some state to be added that tracks delegation. This is typically achieved by requiring an addition to the credential to delegate it. This state may not be recorded on the service itself, but eventually becomes available to the service. This now gives the service additional information, which may allow it to revoke authorizations in a more fine-grained manner, possibly by invalidating all credentials and re-issuing credentials to only the desired recipients.

4. **Service control.** Credentials are not delegatable at all. This allows them to be completely revocable and fine-grained revocable, as there is a 1:1 correspondence between credentials and authorized users.

The points on this access define both a metric for control of access and a trade-off between complexity of processing of queries/requests and ability to support complex collaborations among users. Shared control with recorded delegation may require complex processing to validate a delegated credential.

We are unaware of any implementations at the user control point, but it does represent the absolute minimum of control given to the service. We will see examples of services at the other three points in section 5. We specifically ignore the possibility of sharing an identity credential for the purposes of impersonating another user as a means of delegation; we instead will address this issue in section 4.4.

4.2 State Distribution

Any access control implementation must maintain information relevant to access control decisions. Credentials transmitted across a network are just one type of such state. This information can also be access lists, capability lists, historical data, etc. This axis represents how the global state γ is distributed across the local states $s_1, \ldots, s_n$. We specifically examine the case where for a particular request r "i requests service from k", how much state information is kept in s_i versus the amount kept in s_k for only that particular request. We are therefore not considering the aggregate state of k for all requests it considers. We also do not count any service-wide state it maintains, such as its own identity certificate, or list of operations provided.

We present the following points on this axis, listed in increasing amount of state per authorization stored by the user i. When state is stored across multiple physical nodes on the serving side, such as when a service queries a trusted server for the access control decision, we group all of this state under the management of "the service." Let $\mathcal{S}$ be the space requirement of the service, and $\mathcal{U}$ that of the user.

1. **Centralized.** $\mathcal{S}$ is unbounded, and $\mathcal{U} = 0$. The service maintains all state information, and the user stores zero state – not even a username or password.

2. **Service managed.** $\mathcal{S}$ is unbounded, and $\mathcal{U} = O(1)$. The service maintains that part of the access control-related state, which may grow to arbitrary size, while the user stores constant-sized state, such as a username/password pair.

3. **Equal sharing.** $|\mathcal{S}| \approx |\mathcal{U}|$. The service and the user each maintain arbitrary per-authorization state. There may be differences in size or content.

4. **User managed.** $\mathcal{S} = O(1)$, and $\mathcal{U}$ is unbounded. The service stores constant state for each authorization, but the user's storage requirement may grow arbitrarily large.

5. **Decentralized.** $\mathcal{S} = 0$, and $\mathcal{U}$ is unbounded. The service stores zero state, and so the user stores all access-control state.

This axis is related to vulnerabilities to different attacks and to the cost of maintaining consistency between local and global state. State kept on users minimizes the impact of a successful theft of data but increases the probability that an attack will be successful since users may be less well protected than services. State kept on users also requires that more state data be communicated for each access. If a service maintains access control state data on its users then an attack may result in widespread damage.

The only centralized system of which we are aware is one that is anonymous, such as anonymous FTP, where not even a username and password are stored by the user to get access. We will consider systems at the other four points in section 5.

4.3 Fidelity of Enforcement

A distributed implementation of an access control scheme may not implement the behavior of the access control scheme it implements since maintenance of local states which are complete with respect to all queries and strongly consistency with global system state may be prohibitively expensive. We quantify deviation from complete fidelity of an implementation to its scheme by introducing the enforcement axis, which measures how well an implementation follows the decisions of its scheme. We present the following points in decreasing amount of fidelity of the implementation to the scheme, as we expect the earlier points will be more familiar to the reader.

Recall that $\gamma \in \Gamma$ is the global state of the system, and s_i is the local state of a particular service i in that state. Further, recall that $\vdash$ is the query result according to the scheme, which has perfect knowledge of global state, and $\models$ is the query result according to the implementation, which has only the local state as maintained by the implementation to consider. Let r be a request under consideration. These properties are taken for all states γ and the corresponding s_i in that global state.

1. **Total fidelity.** In this case, $s_i \models r \Leftrightarrow \gamma \vdash r$. No improper requests can occur.

2. **Total prevention.** A slightly weaker case, $\gamma \nvdash r \Rightarrow s_i \not\models r$. A request may still be denied when it should be granted, but a request will never be granted when it should be denied.

3. **Total detection.** In this case, it is possible that $\gamma \vdash r$ but $s_i \not\models r$, or $\gamma \nvdash r$ but $s_i \models r$, meaning that the service makes an incorrect decision due to its incomplete state data, so we have only partial fidelity. However, improper requests are *always detectable*, so a service eventually discovers when a deviation from the scheme has occurred.

4. **Partial detection.** There is still only partial fidelity, but now improper requests are not always detectable:

a service is not guaranteed to later detect an incorrect decision. Improper requests still are detectable, so there is a chance the violation will be discovered.

5. **No detection.** In this situation, improper accesses are not even detectable; a service makes a decision and cannot later tell whether or not it was correct.

6. **No fidelity.** In this situation, $(\forall r) s_i \models r$, regardless of $\vdash$. This is the absence of an access control mechanism.

This axis is a direct metric for strength of compliance to the scheme being implemented. However, the complexity and cost of attainment of each level of fidelity in a distributed implementation depends upon the queries permitted in the scheme and the cost and complexity of maintaining complete and consistent local state information. In the Kraft-Schäfer [13] mobile ad-hoc routing system (see section 5.4) queries concern forwarding messages in an ad-hoc network. The dynamic network structure causes transient incompleteness of state information inducing a window of possible loss of fidelity. BitTorrent [7] (see section 5.5) has a similar window of fidelity loss between when a user decides not to cooperate, and when that lack of cooperation is detected.

4.4 Identity Resolution

Although identity does not have to be used in an access control decision, it is done with such frequency that the mechanisms by which identity is established should be a part of any characterization of access control implementations. At some point in the operation of such a system, a service must make a connection between a user outside of itself and a local representation. In simple systems, it may be the connection between a user and a particular set of permissions. In role-based systems [11, 16], it may be the connection between a user and a role it is attempting to assert. The connection that spans the gap between outside the service and inside the service is called the user's *identity* and is usually represented by an *identity credential*. While a scheme assumes that all identity credentials which are presented are valid, this cannot be assumed in an implementation. The confidence in this connection measures the likelihood of convincing an outside party, such as an impartial judge, of the strength of the binding between that representation and the outside party. We present the following points in increasing order of confidence given to this binding.

1. **No identity mechanism.** This is the lack of an identification, where all users are classified exactly the same.

2. **Shared secret.** In this mechanism, users are given a shared secret to distinguish them as members of a selected class, distinguishing them from users who do not possess the secret. There is no differentiation amongst individuals in the group; merely a classification into "in the group" and "not in the group."

3. **Known name and shared secret.** This adds to the shared secret by differentiating amongst individual users, which allows more fine-grained delegation of authorizations amongst those individuals.

4. **Known name and certificate.** A certificate is defined as a document attesting to the truth of some fact. An identity certificate specifically attests to the identity of the bearer – the very connection we wish to make when establishing identity. This certificate adds to the confidence by adding the testimony of a witness or authority.

5. **Unforgeable identity.** There is a fool-proof guarantee of identity, or identity is established through a mechanism whose probability of error can be safely neglected, such as biometric identification of human operators.

This axis is a metric for vulnerability to attacks since many attacks are based on an attacker successfully impersonating some other entity.

4.5 Decision Mode

In a distributed system, the ultimate authority of whether or not to provide a service is the service itself. Services may consult other services, or defer their decision to a chosen authority. For any request r "i requests service from k" under consideration, we define two subsets of the services' state machines: r_{IN} which is the set of local states that contribute input to the decision on r, and r_{OUT}, which is the set of state machines that have control over whether or not $s_k \models r$.

We present the following points in increasing order of how globally such decisions are made.

1. **Local.** The simplest case, where an access control decision is made based entirely on the local state of the service. $r_{IN} = r_{OUT} = \{s_k\}$.

2. **Advised.** In this case, the ultimate decision is still made by the service itself, but it reaches out to other services to collect information, to get a better view of the global system state and thus make a more well-informed decision. $r_{OUT} = \{s_k\}$, but now $|r_{IN}| > 1$. Generally, $s_k \in r_{IN}$ as well, but this is not required.

3. **Consensus.** In this case, not only are the opinions of other services involved, but some group collaborates following an agreed-upon protocol and reaches a common decision, with none having ultimate authority. In this case, both $|r_{IN}| > 1$ and $|r_{OUT}| > 1$. s_k is often, but not necessarily, in each set.

4. **Centralized.** A special case of consensus, and the most global decision, where the "group" is a single node, that is not the service itself. Here, the central authority A has the best possible knowledge of the global state, and so can make the best decisions. There is no restriction in the size or membership of r_{IN}, although typically it consists of just s_A. However, $r_{OUT} = \{s_A\}$, and also, $A \neq k$.

Schemes are defined on the assumption that decisions are made on the basis of complete and consistent global state. Any distributed implementation is a trade-off between cost of state maintenance and fidelity. The points on this axis represent trade-offs between fidelity, scalability and performance and cost of maintenance of complete and consistent state.

4.6 Static or Adaptive Trust Management

"Trust" is shorthand for a standard of expected behavior. The choices of whether to incorporate trust in an access control decision and whether or not trust is statically assigned or dynamically computed on the basis of runtime behavior is a property of a policy and thus the scheme representing the policy. We include static or adaptive trust management as a property of an implementation since dynamic computation of trust is a source of loss of fidelity of an implementation with respect to a scheme. For most systems with centralized access control, trust relationships do not change without the outside intervention of an operator. However, with increasingly distributed control, a service may need to revisit its trust relationships during execution to respond to changing conditions, such as evidence that an entity has altered its behavior. We specifically exclude the case where an administrator or outside influence causes changes to trust relationships, as we instead view this as the state machine itself being altered; here we address only when there the service can transition between states that reflect a change in certain trust relationships as a result of run-time events. There are two cases:

1. **Static** (or manually changed). Trust relationships never change as a result of run-time relationships, and remain static throughout the operation of the system, excluding changes effected externally. Therefore, if $s_i \vDash r$ in *any* global state γ, then $s_i \vDash r$ in *all* γ; and similarly for $s_i \nvDash r$.
2. **Adaptive.** $(\exists r, \gamma_1, \gamma_2)\left[s_{i(\gamma_1)} \vDash r\right] \wedge \left[s_{i(\gamma_2)} \nvDash r\right]$. The service can transition to states that carry an altered trust relationship, as a result of run-time events, such as internal state changes, the behavior of users, or other aspects of interactions with other services.

Implementations utilizing adaptive trust relationships can constructively respond to certain forms of attack or access abuse.

5. EXAMPLE CLASSIFICATIONS

This sections gives a brief characterization of each system in terms of what kinds of requests its scheme and implementation support, followed by its classification under each axis. For brevity, we do not give a complete description of the access control scheme, but instead give a brief characterization of each system in terms of what kinds of requests its scheme and implementation support, followed by its classification under each axis.

5.1 Akenti

Akenti [18] is a system for access control in grid environments. Each user has an identity certificate, and *stakeholders* who control resources express their rules for access in certificates they sign and provide. In this system, servers providing resources look to a trusted Akenti policy server to make the decision, rather than making its decision locally. Requests in this system concern only accessing grid resources, subject to those restrictions. The properties of Akenti are as follows.

- Control over Sharing: **Service control.** Users cannot delegate any authority to other users. Authority must come directly from the stakeholder's access control description.
- State Distribution: **Service managed.** Users each store an X.509 certificate, and that is all. The resources and policy servers, which we consider both part of the service side, store all the stakeholder-provided access control restrictions.
- Fidelity of Enforcement: **Total prevention.** Requests are always evaluated as to whether or not they follow the provided stakeholder certificate. A request will only be granted if the resource server can contact a policy server and get a positive answer. If a policy server cannot be contacted, the request will be denied, which may be at odds with the intended behavior. But, a request that should be denied will always be denied.
- Identity Resolution: **Known name and certificate.** An X.509 certificate, that must be signed by a recognized Certifying Authority, is used for identification. Akenti uses commercial vendors such as Netscape, Entrust, or Verisign for signed identity certificates. Akenti uses TLS [9] as the authenticating transport.
- Decision Mode: **Centralized.** Decisions are always deferred to Akenti policy servers. The policy server may be replicated, and so there may be several, but it is in effect one single decision-maker.
- Trust Management: **Static.** All access control restrictions are statically assigned by stakeholders. This will never be changed by the system itself.

5.2 CRISIS

CRISIS [2] is the security architecture of the WebOS [20] distributed operating system, which uses many of the ideas of the shared-memory multiprocessor operating system TAOS [1]. CRISIS introduces an architecture of *principals*, *objects*, and *resource monitors* to correspond to users, services, and access control decision-makers, respectively. Users have identity certificates, and can construct *transfer certificates* to give to other users to delegate portions of their access. Transfer certificates contain predicates that limit the access delegated. Access is granted if a user presents an identity that is explicitly authorized for a resource (by appearing on the resource's access list), or presents a chain of transfer certificates proving delegation from such an authorized user and satisfaction of all limiting predicates on all transfer certificates. Requests concern access to arbitrary services available in the network. The states of the system incorporate transfer certificates which record delegations that have been made. The properties of CRISIS are as follows.

- Control over Sharing: **Shared control with recorded delegations.** A user makes transfer certificates to transfer a subset of its privileges to another. This results in a chain of certificates refining a set of capabilities to the point where the principal attempting to use them is found, and are used as proof of access rights. When presented for invocation, this information becomes available to the service, which can be used for later changes in access control policy or system auditing.
- State Distribution: **User managed.** A service must store an entry in an access control list for each authorization, with a list of access rights. A user wishing to

make use of a resource must store a certificate chain, which can grow to arbitrary size depending on the number of delegations made, and present it to make an invocation.

- Fidelity of Enforcement: **Total fidelity.** As the access control policy is encoded directly into the reference monitor, and the possible queries are all permitted or not directly based on that policy, the reference monitor can perfectly implement the scheme.
- Identity Resolution: **Known name and certificate.** CRISIS uses two kinds of certificates: identity certificates and transfer certificates. The identity certificates are signed by both a trusted certifying authority and an on-line authority.
- Decision Mode: **Local.** The resource monitor needs only verify the chain of signatures, and then verify the root of the chain is an explicitly-authorized user on its local access control list.
- Trust Management: **Static.** Changes to access control restrictions are only ever made by explicit alteration of access control lists by an operator.

5.3 dRBAC

Distributed Role-based Access Control (dRBAC) [11] is a distributed access control system for managing systems which have components in multiple administrative domains. It incorporates the RBAC model in [16], and includes delegation chains similar to those in CRISIS. They abstract both users and resources in the system as *entities*, each of whom has its own local namespace. It reconciles these two features by introducing delegation semantics that delegate roles to other roles or entities. Only roles can be delegated, not individual rights. These delegations can be extended with *valued attributes* which modulates the access level. The right to delegate is itself a right that must be delegated separately; being delegated a right does not allow one to delegate it further unless the right to delegate has been delegated as well.

In dRBAC, an entity is the only certifying authority for entries in its namespace, which corresponds to resources being the only ultimate authority for operations they export, and so each chain must end with a delegation by the entity itself. These delegation chains are then verified by *proof monitors*, which is part of the dRBAC infrastructure, present at each entity. In this scheme, requests concern access to specific resources, and the pertinent states of the system concern what delegations have been made between users. The properties of dRBAC are as follows.

- Control over Sharing: **Shared control with recorded delegations.** dRBAC uses a certificate chain where each link in the chain is a statement of the form `[Subject` → `Object] Issuer`, where an Issuer asserts that the Subject has the role of Object. Subject can be a role or an entity, but Object is always a role. This allows assigning roles to entities, as well as roles to other roles.
- State Distribution: **Decentralized.** These certificate chains are maintained by the users and presented when service is requested. The proof monitor needs only to be able to verify the chain of signatures, which it can do with just the submitted chain, and it needs no per-authorization state to do so. Once the delegation is extended, it can then be forgotten by the service.
- Fidelity of Enforcement: **Total fidelity.** As the requests are straightforward enough that they can be programmed definitively into a service, the service can follow the policy with total fidelity.
- Identity Resolution: **Known name and certificate.** Each certificate chain starts with an identity certificate, and then has zero or more delegation certificates. dRBAC assumes that when a delegation certificate is signed, the issuer is also attesting to the identity of the entity or role to which authority is delegated, eliminating the need for a separate certifying authority.
- Decision Mode: **Local.** All certificates can be validated without any external sources, and the final entry must be issued by the servicing entity, which can naturally verify its own signature.
- Trust Management: **Static.**

5.4 Kraft-Schäfer Mobile Ad-Hoc Networks

Kraft and Schäfer [13] give a system for the exclusive purpose of granting or denying relay access in a mobile ad-hoc network. To adhere to their nomenclature, in this section we will specifically refer to "nodes" rather than services. The scheme adopts a trust metric to decide whether or not another node has been sufficiently cooperative and trustworthy to forward its packets; in a mobile ad-hoc network, the primary improper behavior is greedily using the network, and not forwarding the packets of others. Each node generates its own identity certificate, and can be vouched for by other nodes in a "web of trust." Both opinions of other nodes and observed behavior affects a node's local rating. The requests supported by this system are to request a packet be forwarded onward in the network, and the states concern historical data, and whether or not a particular node is trying to be "rogue." The properties of Kraft-Schäfer are as follows.

- Control over Sharing: **Shared control with recorded delegations.** If one node decides it no longer can trust another node, and so no longer wants to pass traffic on that node's behalf, it can immediately cease. For the purpose of introducing new nodes to the network, established nodes can provide "warrants" to new nodes, to delegate part of their good rating to the new node.
- State Distribution: **Equal sharing.** All nodes in the network retain the same amount of information about each other, in tables of trust values.
- Fidelity of Enforcement: **Partial detection.** This system uses a heuristic to decide beyond what point a node is considered sufficiently uncooperative to warrant future denial of access. Although in the global state we can tell when a node decides to become rogue and not cooperate, it is not guaranteed that an individual node will detect this ever, depending on how uncooperative the rogue node becomes. At least one improper request can certainly be accomplished by the rogue node, as behavioral information is not available until after the improper request has taken place.

- Identity Resolution: **Known name and certificate**. In this case, to avoid Sybil [10] attacks but not require a globally-recognized certificate authority, certificates must be signed by another node in the network. The trust given such a certificate depends on its signer.

- Decision Mode: **Advised.** An individual node decides whether or not to forward traffic at the behest of another node. It may use input from other nodes under a form of opinion sharing [12], but the decision to forward or not is entirely its own.

- Trust Management: **Adaptive.** Trust relationships are constantly re-evaluated by the system itself based on previous cooperation.

5.5 BitTorrent

BitTorrent [7] is a file distribution system used to spread the bandwidth cost of disseminating large files across a large number of nodes. It uses access control to enforce collaborative behavior. A node grants access to those who are providing data, and denies access to those who are attempting to consume without providing. In this analysis we consider only the peer-to-peer file transfer protocol, and do not consider the "trackers" which are used for bootstrapping. The requests are for downloading "chunks" of files, and the states of the system are the recorded history made by each service in its recent interactions with a particular user. The properties of BitTorrent are as follows.

- Control over Sharing: **Service control.** A BitTorrent peer serving a file, the service, decides directly what access to give a client. That access cannot be then shared with other users.

- State Distribution: **Centralized.** A service maintains historical data on users, but indexes them by network address as opposed to any independent identity. A downloading user has to find the service, but needs store no access control-related state.

- Fidelity of Enforcement: **Total detection.** A service can detect when a downloading user is not giving "tit-for-tat," and deny access after the fact. However, such a lack of cooperation will always be detected in time for the service to cease its own cooperation.

- Identity Resolution: **Unforgeable identity.** BitTorrent identifies a peer exclusively by its endpoint IP address and port number for the duration of its connection. It is unforgeable because as soon as a peer disconnects, its identity and history is forgotten, in a way that makes it unforgeable by design.[3]

- Decision Mode: **Local.** The access control function is evaluated solely on information observed by the service.

- Trust Management: **Adaptive.** BitTorrent regularly re-evaluates the access control function to consider new observations on downloading peer behavior. It quickly chokes off uncooperative peers, and gradually increases cooperation with cooperative peers.

[3]It is theoretically possible for an intruder host on the same subnetwork or even the same host as a downloading user to simultaneously shut the peer out while taking its place, thus assuming its identity and its (presumably) favorable history. We are unaware of any such attack in existence and expect it is highly impractical to mount.

5.6 Wi-Fi Protected Access in IEEE 802.11 Wireless Networks

Wi-Fi Protected Access [21], or WPA, and its successor, WPA2, are the latest access control mechanism for 802.11a/b/g wireless networks. The requests handled by this scheme are to gain access to a wireless network for communication. It incorporates a Temporal Key Integrity Protocol (TKIP) which dynamically changes keys as the system is used.

The differences between WPA and WPA2 are not relevant for this discussion. However, WPA and WPA2 each operate in two primary modes: WPA-Enterprise (or WPA2-Enterprise), for use in a large-scale networks with an IEEE 802.1X-compliant authentication server, and WPA-Personal (or WPA2-Personal). Each mode has different properties under our classification, and so we will present each one separately.

Enterprise Mode

In the enterprise environment, a user possesses a set of credentials in the form of a digital certificate, username/password pair, smart card, or any other identity mechanism desired by the administrator. A user then authenticates to an authentication server, which is a single point for all wireless access points, which are the services. After this point, a shared session key is negotiated to allow access. The properties of WPA in this mode are as follows. The requests are the association/authentication protocol, and a request to forward messages. The relevant state is the authentication information programmed into the central server, and the state stored in the wireless access points to track what users are authenticated at a particular time.

- Control over Sharing: **Service control.** Users cannot delegate their authorization to other users by any means.

- State Distribution: **Service managed.** A user must store an identification credential of some kind, but the authentication server can store any number of access restrictions, such as to which access points a user may associate, time-of-day usage restrictions, or validity periods.

- Fidelity of Enforcement: **Total prevention.** Access to the wireless network is only granted when the authentication server returns a positive response, and the authentication server is programmed with exactly the desired access control policy. However, if the authentication server is unavailable, requests may be denied when they should be granted.

- Identity Resolution: **At least known username and shared secret.** Part of the WPA-Enterprise standard allows choosing the identification mechanism. Such a mechanism is at least a username and password pair, but can be a username and digital certificate, smart card, or any other secure identity mechanism. We are unaware of any application of WPA-Enterprise that uses less confident measures.

- Decision Mode: **Centralized.** Each service (the wireless access point) defers its decision to a central authentication server, which makes global decisions on what users to allow and what users to deny.
- Trust Management: **Static.** Although it is possible for an authentication server to be designed to automatically update its trust relationships based on run-time events, we are unaware of any such implementations.

Personal Mode

For "SOHO" (small office/home) users who do not have such an elaborate authentication infrastructure, WPA can also operate in the "personal" mode. Also known as *pre-shared key* (PSK) mode, an individual access point and any authorized users are given a passphrase to access the network, much the same way as with WEP, although the implementation of the cryptography has been improved to avoid WEP's aforementioned vulnerabilities. In this case, there is no central authentication server, and wireless access points do not coordinate with one another (although they can share the same passphrase). The properties of WPA in this mode are as follows.

- Control over Sharing: **Shared control with unrecorded delegations.** The passphrase is now a credential that can be delegated, and there is no tracking of its being passed from user to user. Access by all authorized users can be revoked by changing the passphrase and delegating it again to authorized users, but there is no method of more fine-grained revocation.
- State Distribution: **Equal sharing.** The service and the user each need only now store the passphrase, and any intermediary keys negotiated as part of the authentication protocol.
- Fidelity of Enforcement: **Total fidelity.** The query in this system is whether or not a user knows the passphrase. If so, access is granted. This query is accurately implemented by providing the passphrase to the access point as proof of knowledge.
- Identity Resolution: **Shared secret.** As previously mentioned, the only differentiation amongst users are those that are authorized by virtue of knowing the passphrase, and those that are not.
- Decision Mode: **Local.** The access point is manually programmed with the passphrase, and needs no outside communication to determine the access control decision.
- Trust Management: **Static.**

6. EVALUATION

The insights which arose from development of the classification scheme and analysis of the several systems include:

1. Access control is conventionally thought of as implementing security and management of resources. But in fact, access control is being used to play several different roles in the systems we have characterized:
   - Security and control over resources: Akenti, WPA, CRISIS, dRBAC
   - Construction of virtual systems: Akenti
   - Enabling collaboration: CRISIS, dRBAC
   - Enforcement of collaboration: BitTorrent, Kraft-Schäfer

   These broader definitions of access control arise primarily from the intrinsic requirements of distributed control for collaboration to accomplish goals.
2. Positions in the taxonomy where we did not find implemented systems but where one might expect to find future distributed systems include those where Control over Sharing is at least Shared Control with Recorded Delegation, where Decision Mode is Advised or Consensus and where Trust Management is Adaptive.
3. In systems with more distributed control, such as Kraft-Schäfer in section 5.4 and BitTorrent in section 5.5, implementation decisions are made which appear to make the enforcement weaker, but these implementations are, in fact, the strongest implementations for the given scheme devised thus far. This dichotomy stems not from poor design choices or implementation, but that the access control scheme now must support queries that prove difficult to implement with fidelity in a way that gives acceptable performance. Enforcement with total fidelity or total prevention requires maintenance of up-to-date global state information that is infeasible for networked systems of arbitrary size. Just as the legal system contends with illegal acts in a very large population by employing what prevention is feasible, detection where it is not feasible, and then incentive to comply in the form of punishments, this analysis suggests these new frontiers in distributed systems with fully distributed control will require such hybridized approaches as we are seeing emerge.

7. CONCLUSIONS AND FUTURE WORK

We have presented a classification for access control implementations, and observed that new systems, with more distributed control, have access control schemes that are more difficult to implement efficiently with complete fidelity, and have led to the necessity of less "perfect" mechanisms that still function. We give this as a basis for comparing access control implementations, and as a beginning to exploring the deeper unifying concepts amongst access control implementations, and the interplay between a scheme and a corresponding implementation.

There may exist further properties of implementations that can define additional axes. There may exist additional points representing additional implementation mechanisms on the axes discovered so far. In particular, the Trust Management axis, may have mechanisms for implementation of adaptation which lead to different degrees of fidelity of enforcement. We will continue to look for both of these as future work. Further, in an ideal taxonomy the axes of the lattice would be orthogonal from each other, but we can see that our current axes are not. In particular, we have seen an interplay between the Decision Mode and the Enforcement axis: in the systems with Centralized decision mode, there exists the potential for a network failure where the central server is inaccessible, and so a service will default to

"fail-closed" rather than possibly improperly service a request that limits the implementation to total prevention on the Enforcement axis. We continue to look for orthogonal decompositions of this space, in which some of these non-orthogonal axes may instead be better expressed as combinations of more fundamental, orthogonal ones.

Finally, although a taxonomy such as this is interesting as a way of formalizing properties that so far have been treated informally, it has potential its possible use as a means of comparing implementations of a particular access control scheme for properties such as vulnerability to a given form of attack, or motivating exploration of alternative implementation choices for it. Specifically, we are looking into creating a set of metrics that give ordered sub-lattices of this taxonomy with respect to how well an implementation guards a system against particular kinds of threats, like denial of service, unintended provision of service, unintended information flow, or collusion amongst a group of attackers. We have only seen one access control implementation so far attempt to take advantage of decentralized state distribution: dRBAC. Solutions requiring no state storage on the side of the server have shown resilience to Denial of Service attacks. These solutions often employ cryptography to store state in an immutable way on the user, just as the certificate chains in dRBAC, the one system we have found at this state distribution point. Conversely, such increased distribution could lead to unintended information flow, where access rights or trust relationship information is exposed. In systems where colluding groups of attackers is an issue, adaptive trust management and more global decision modes can lead to quicker detection of such groups. We will explore formalizing these metrics in terms of our axes.

Acknowledgements

This work has been supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract No. NBCH30390004 and by NSF grant number ACI-0305644, "Montage: An Integrated End-to-End Design and Development Framework for Wireless Networks."

8. REFERENCES

[1] M. Abadi, E. Wobber, M. Burrows, and B. Lampson. Authentication in the Taos operating system. In *Proceedings of the 14th ACM Symposium on Operating System Principles*, pages 256–269, The Grove Park Inn and Country Club, Asheville, NC, 1993. ACM Press.

[2] E. Belani, A. Vahdat, T. Anderson, and M. Dahlin. The CRISIS wide area security architecture. In *USENIX Security Symposium*, pages 15–30, Jan 1998.

[3] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. *ACM Transactions on Information and System Security*, 6(1):71–127, Feb 2003.

[4] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote trust-management system, version 2. IETF RFC 2704, 1999.

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.

[6] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss. Referee: Trust management for web applications. *World Wide Web Journal*, 2:706–734, 1997.

[7] B. Cohen. Incentives build robustness in BitTorrent. `http://www.bittorrent.com/bittorrentecon.pdf`, May 2003.

[8] S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Access control: principles and solutions. *Software: Practice and Experience*, 33(5):397–421, Apr 2003.

[9] T. Dierks and C. Allen. The TLS protocol, version 1.0 (RFC 2246). `http://www.ietf.org/rfc/rfc2246.txt`, Jan 1999.

[10] J. R. Douceur. The Sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Mar 2002.

[11] E. Freudenthal, T. Pesin, L. Port, E. Keenan, and V. Karamcheti. dRBAC: Distributed role-based access control for dynamic coalition environments. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002)*, Jul 2002.

[12] A. Jøsang. A subjective metric of authentication. In *Proceedings of the 5th European Symposium on Research in Computer Security (ESORICS'98)*, 1998.

[13] D. Kraft and G. Schäfer. Distributed access control for consumer operated mobile ad-hoc networks. In *Proceedings of the First IEEE Consumer Communications and Networking Conference (CCNC'2004)*, Jan 2004.

[14] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transactions of Information and System Security (TISSEC)*, 6(1):128–171, Feb 2003.

[15] R. S. Sandhu. Access control: The neglected frontier. In *First Australian Conference on Information Security and Privacy*, 1996.

[16] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, Feb 1996.

[17] R. S. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.

[18] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of the Eighth Usenix Security Symposium*, pages 215–228, Aug 1999.

[19] M. V. Tripunitara and N. Li. Comparing the expressive power of access control models. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, Oct 2004.

[20] A. Vahdat, E. Belani, P. Eastham, C. Yoshikawa, T. Anderson, D. Culler, and M. Dahlin. WebOS: Operating system services for wide area applications. In *Proceedings of the Seventh Symposium on High Performance Distributed Computing*, 1997.

[21] Wi-Fi Alliance. Wi-Fi Protected Access: Strong, standards-based, interoperable security for today's Wi-Fi networks (white paper). `http://www.wifialliance.com/OpenSection/pdf/Whitepaper_Wi-Fi_Security4-%29-03.pdf`.