TrustBAC - Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems

Sudip Chakraborty
Computer Science Department
Colorado State University
Fort Collins, CO 80523, USA
sudip@cs.colostate.edu

Indrajit Ray
Computer Science Department
Colorado State University
Fort Collins, CO 80523, USA
indrajit@cs.colostate.edu

ABSTRACT

Conventional access control models like role based access control are suitable for regulating access to resources by known users. However, these models have often found to be inadequate for open and decentralized multi-centric systems where the user population is dynamic and the identity of all users are not known in advance. For such systems, credential based access control has been proposed. Credential based systems achieve access control by implementing a binary notion of trust. If a user is trusted by virtue of successful evaluation of its credentials it is allowed access, otherwise not. However, such credential based models have also been found to be lacking because of certain inherent drawbacks with the notion of credentials. In this work, we propose a trust based access control model called TrustBAC. It extends the conventional role based access control model with the notion of trust levels. Users are assigned to trust levels instead of roles based on a number of factors like user credentials, user behavior history, user recommendation etc. Trust levels are assigned to roles which are assigned to permissions as in role based access control. The TrustBAC model thus incorporates the advantages of both the role based access control model and credential based access control models.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; H.1 [**Models and Principles**]: Miscellaneous; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Management, Security, Theory

Keywords

Access control, Trust, RBAC

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

1. INTRODUCTION

Proper access control to resources is one of the major concerns for any organization. Different models of access control have been proposed over the years, for example, discretionary and mandatory access control models, Clark-Wilson model, Task based models and Role Based Access Control model. Among these, role based access control (RBAC) [15] is gradually emerging as the standard for access control. The main advantage of RBAC over other access control models is the ease of security administration. In the RBAC model access permissions are not assigned directly to the users but to abstractions known as "roles". Roles correspond to different job descriptions within an organization. Users are assigned to different roles and, thus, indirectly receive the relevant permissions. Thus, with RBAC, security is managed at a level corresponding to an organization's human resource structure.

Notwithstanding the success of the RBAC model, researchers have often found the model to be inadequate for open and decentralized multi-centric systems where the user population is dynamic and the identity of all users are not known in advance. Examples of such systems are service providers operating over open systems like the Internet. It is almost impossible to know beforehand all the users that will request services in these systems. Assigning appropriate roles to these users thus becomes an irrational and ad-hoc exercise. To overcome the shortcomings of RBAC for such systems, researchers have proposed credential-based access control models [8, 7, 20]. Credentials implement a notion of binary trust. Here the user has to produce a predetermined set of credentials (for example, credit card numbers or proof of membership to certain groups etc.) to gain specific access privileges. The credential provides information about the rights, qualifications, responsibilities and other characteristics attributable to its bearer by one or more trusted authorities. In addition, it provides trust information about the authorities themselves. Researchers have also integrated credential based access control with role-based access control to facilitate security administration [22, 10, 21, 23].

Although credential based models solve the problem of access control in open systems to a great extent, it still has a number of shortcomings. A credential, strictly speaking, does not bind a user to its purported behavior or actions. It does not guarantee that its bearer really satisfies the claims in the credential. It does not convey any information about the behavior of the bearer between the time the credential was issued and its use. A credential does not reveal whether it was obtained via devious means. In real life some or all

such information may play crucial parts in access control decisions. Additionally, credential based access control does not keep track of the user's behavior history. Access permission is given on the basis of the credential presented for a particular session. Either the user's credentials are accepted and required privileges are allowed, or the credentials are rejected and the user does not get the access rights. Thus, good behavior by the user cannot be rewarded with enhanced privileges nor bad behavior be punished.

The above observations motivate us to revisit the problem of access control in decentralized and multi-centric open systems. We believe credential based access control is a step in the right direction. However, we would like to enhance the binary trust paradigm in these models with a much richer multi-level trust model. In this new trust model, trust levels in the users can be determined not only by using the credentials presented by the user but also from the results of past interactions with the user, from recommendations about the user and/or knowledge about other characteristics of the user. A user is mapped to different trust levels based on these information. Trust levels (and not users, unlike in conventional RBAC) are then mapped to roles of RBAC. Thus our access control model is an enhanced RBAC (TrustBAC). Changes in the trust level of user changes the roles that the user has in the system and thus the user's privileges. The system can define as many trust levels as it wants and can assign each level to a specific set of resources tied with a specific set of access privileges. The system just needs to monitor the trust level of the user and the regulation of access is automatically achieved.

The rest of the paper is organized as follows. Section 2 gives an overview of some of the related works on access control. There is a plethora of works in access control mechanisms. Here we present some of the works that are related to role-based access control model. We also present works that address authorization issues in open environments, like the Internet. In section 3 we formally present the core TrustBAC model. We define the various components of the model and the inter-relationships among the different components. In the TrustBAC model, trust relationship between the user and service provider is evaluated based on a vector model of trust that we had proposed earlier [25]. We present the relevant portions of the trust model in section 4 and discuss the modifications to the basic model for adoption in TrustBAC. Next, in section 5, we discuss how the trust model works in TrustBAC to perform access control. Finally section 6 concludes the paper.

2. RELATED WORK

Role-based Access Control (RBAC) was first introduced by Ferraiolo and Kuhn in [14] to address the limitations of discretionary access control model (DAC). Sandhu et. al [26] introduce four reference models to provide systematic approach to understand RBAC model. Their framework separates administration of RBAC from its use for controlling access. They also categorize the implementation of RBAC in different systems. Finally, after a series of modifications, the NIST standard for RBAC is proposed in [15]. The standard specifies RBAC reference model which defines the scope of features that comprise the standard and provides terminologies to support specification. The standard also specifies system and administrative functional specification which defines functional requirements for administrative operations and system level functionalities. We have followed the approach of the standard to formalize our TrustBAC model.

A main feature of the TrustBAC model is the use of user behavior history is determining access privileges. Similar concept of using execution history in access control can be found in literature. In [13] the authors present a history-based access control mechanism which is suitable for controlling access from mobile code. The scheme maintains a selective history of access requests made by individual program and use this history to measure the degree of safeness of a request. Another history based access control for codes is presented in [1]. In this scheme the access privileges of a code is determined in runtime by examining the attributes of the pieces of code that have run before. The pieces of code that have run includes the codes on stack as well as the codes that have been called and returned. Also, ours is not the only work which uses the concept of trust in access control. Sandhu et. al in a recent paper [27] present a trusted computing architecture to enforce access control policies in peer-to-peer environment.

In [28], Thomas introduces the notion of TeaM-based Access Control (TMAC) as an approach to applying role-based access control in collaborative environments. The "team" is an abstract container that encapsulates a set of users with specific roles. A team is formed with the objective of accomplishing a specific task. One advantage of TMAC model is it allows role-based permissions across object types as well as fine-grained, identity-based control on individual users in certain roles and to individual object instances. Georgiadis et. al [16] extend TMAC model by integrating it with RBAC and using it for general contextual information like time of access, location from which access is requested, location of the object for which access is requested etc. Somewhat similar idea is presented in the YGuard access control model [29]. YGuard employs a set-based access control list where a group of subjects is authorized to access sets of objects. That is, for those objects, every member of the group has same access privileges. Another similar approach is coalition-based access control (CBAC) model [11] where a coalition (group of members) shares data with their partners while ensuring that their resources are safe from inappropriate access. They define the protection state of a system, which provides the semantics of CBAC-based access policies. Researches are still continuing on issues and applications of RBAC model. Some recent works includes [30] which presents a multilayered, distributed and location-dependent approach to RBAC; Park and Hwang [24] present a scheme in P2P environment where RBAC mechanisms are dynamically supported based on each peer's current context; in GEO-RBAC [4], Bertino et. al extend the RBAC model to deal with spatial and location-based information.

Digital library is one of the major application areas of access control in open environments. There are quite a few work on authorization issues in digital library domain. In one of the early work on access control in digital libraries, H.M. Gladney proposes a scheme called DACM (Document Access Control Methods) in [18]. The basic idea is biased towards discretionary access control with some extensions to handle mandatory access control. Winslett et. al [31] propose a mechanism to assure security and privacy for digital library transactions. This is basically a credential-based system where both client and server can specify their own policies regarding credential disclosure and security. Client

uses a personal security assistant (PSA) module and the server uses server security assistant (SSA) to manage credentials and credential acceptance policies. In [2], Adam et. al propose a content-based authorization model for digital library environments. Authorization is specified based on positive and negative qualifications and characteristics of the user. Credentials are associated with each user and represent qualifications and characteristics of the user.

There are a number of works that specifically address access control on the Internet. Bonatti and Samarati [9] propose a uniform formal framework to regulate service access and information disclosure on the Internet. The regulation is based on credentials. The framework includes mechanism to treat information which are not in the form of a certificate and is needed for required access. It has comprises a language for expressing access and release policies. The framework also has a policy filtering mechanism which helps the entities involved in the communication to exchange their requirements in a concise and privacy preserving way. In [3] Bauer et. al describe the design, implementation, and performance of a system to control access on web. The system is based on proof-carrying authorization (PCA). The access control model provides locating and using pieces of the security policy that have been distributed across hosts and keeping the policies hidden from unauthorized clients. It also provides iterative authorization by which a server can require a browser to prove a series of challenges. In [17] the authors address the problems of access control in large open systems where the authenticated identity of an entity does not provide any information regarding the likely behavior of that entity. Their scheme, called cryptographic access control, is based on cryptography to guarantee confidentiality and integrity of objects stored in potentially untrusted servers in the system.

In [6], the authors present X-RBAC, an XML-based RBAC policy specification framework to deal with access control issues in dynamic XML-based web services. They extend X-RBAC to a trust-enhanced version in [5] where the role assignment is based on trust. The authors define 'trust' as "the level of confidence associated with a user based on certain certified attributes". Unlike our model, this trust level is not quantitatively measured. Instead, the authors use the trust management approach of trusted third parties (e.g., public key certification authority) and use the certificates provided by the third party to assign roles to the users. The authors argue that traditional access control schemes following identity or capability-based approach for authorization do not scale well to the distributed web services architecture. To overcome this limitation they describe a mechanism to configure X-GTRBAC to provide context-aware trust-based access control in Web services. X-GTRBAC is a framework based on Generalized Temporal Role Based Access Control – GTRBAC [19] It provides a generalized mechanism to express a wide range of temporal constraints including periodic as well as duration constraints on roles, user-role assignments, and role-permission assignments. The X-GTRBAC extends GTRBAC with XML to allow policy enforcement in heterogeneous and distributed environment.

In a current survey [12], the present state and future trends in the access control are discussed. The survey shows that the new trend in access control are part of future communication infrastructure supporting mobile computing.

3. TRUSTBAC MODEL

The TrustBAC model is defined in terms of a set of elements and relations among those elements. The elements are of the following types: *user, user_properties, session_instance, session_type, session, session_history, trust_level, role, object, action, permissions* and *constraint*. The corresponding sets are USERS, USER_PROPERTIES, SESSION_INSTANCES, SESSION_TYPES, SESSIONS, SESSION_HISTORY, TRUST_LEVELS, ROLES, OBJECTS, ACTIONS, PERMISSIONS and CONSTRAINTS. The TrustBAC model is illustrated in figure 1(we use one-directional arrows to represent one-to-many relationships, two directional arrows to denote many-to-many relationships and plain lines to denote one-to-one relationships). We define the different elements as follows.

user A user $\in$ USERS is defined as a human being. The notion of user can be extended to include systems, or intelligent agents, but for simplicity we choose to limit a *user* to a human entity.

user_properties Each user u has certain set of properties $\mathscr{P}_u$, called *user_properties*. The set USER_PROPERTIES $= \bigcup_{u \in USERS} \mathscr{P}_u$. A user can manifest any subset P of $\mathscr{P}_u$ (i.e., $P \in 2^{\mathscr{P}_u}$) at a particular session.

session_instance A session_instance $\in$ SESSION_INSTANCES is a 'login' instance of an user. A user can instantiate multiple login thereby initiating multiple session_instances at the same time. A session_instance is uniquely identified by a system generated id.

session-type A session_instance is identified with a type which is determined by the set of properties manifested in that session_instance by the user invoking that session_instance. For a session_instance s invoked by a user u with P ($P \subseteq \mathscr{P}_u$) properties, has the session_type P. Formally, the set SESSION_TYPES $= 2^{USER\_PROPERTIES}$.

session A session $\in$ SESSIONS is identified by a session_instance with a session_type. A session with session_instance s of type P is denoted by the symbol s^P. Formally, $SESSIONS = SESSION\_INSTANCES \times SESSION\_TYPES$.

session_history A session_history $\in$ SESSION_HISTORY is a set of information regarding the user's behavior and trust level in a previous use of a session of that type.

trust_level A trust_level is a set of real number between -1 and +1. A user, at some instant of time with a particular session has a trust_level. The set TRUST_LEVELS is the set of possible subsets of [-1, 1]. That is, TRUST_LEVELS $= \{S \mid S \subseteq [-1, 1]$. Thus TRUST_LEVELS becomes an infinite set where each member S can be either discrete or continuous.

role The concept of role is same as in the RBAC model. A role $\in$ ROLES is a job function with some associated semantics regarding the responsibilities conferred to a user assigned to the role.

object An object $\in$ OBJECTS is a data resource as well as a system resource. It can be thought of as a *container* that contains information.

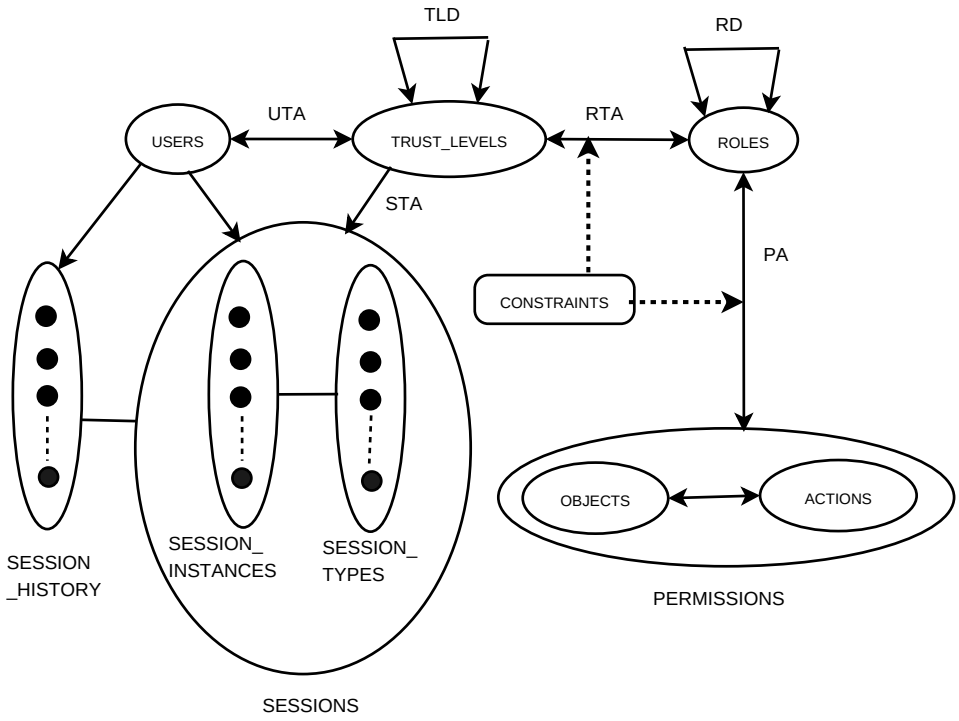

Figure 1: TrustBAC model

action An action ∈ ACTIONS is an executable image of a program. 'read', 'write', 'execute' are examples of a typical action.

permission A permission ∈ PERMISSIONS is an authorization to perform certain task within the system. It is defined as a subset of OBJECTS × ACTIONS i.e., $PERMISSIONS = 2^{(OBJECTS \times ACTIONS)}$. Therefore, a permission = $\{(o, a) \mid o \in OBJECTS, a \in ACTIONS\}$. Permissions are assigned to a role. The type of a permission depends on the nature of the system. The model does not dictate anything about the type.

constraint We borrow the concept of constraint from RBAC model. Therefore, a constraint ∈ CONSTRAINTS is defined as a predicate which applied to a relation between two TrustBAC elements returns a value of "acceptable" or "not-acceptable". Constraints can be viewed as conditions imposed on the relationships and assignments.

Association between any two of the above elements are specified by mathematical relations. TrustBAC has the follwing relations.

1. sua : USERS × SESSION_INSTANCES × SESSION_TYPES → SESSIONS defines the *user-session* assignment relation. $sua(u, s, P) = s^P$ for $u \in$ USERS, $s \in$ SESSION_INSTANCES, $P \in$ SESSION_TYPES, and $s^P \in$ SESSIONS shows that a single session s^P of type P is associated with a single user u with certain properties P. A user can invoke multiple sessions of different types simultaneously.

2. UTA ⊆ USERS × TRUST_LEVELS defines the *user-trust_level* assignment relation. It is a many-to-many relation where a user can have multiple trust levels. Since a user can invoke many sessions at a time, she can have different trust levels, one for each invoked session. A single trust_level can be assigned to many users. The restriction on a member $(u, L) \in$ UTA is L must be a singleton member of TRUST_LEVELS i.e., $L = \{l\},\ l \in [-1, 1]$.

3. STA ⊆ SESSIONS × TRUST_LEVELS defines the *session-trust_level* assignment. It is a one-to-many relation where a session can have only one trust value. That is, the trust_level L corresponding to that session is a singleton member of TRUST_LEVELS. But many sessions can have the same trust_level.

4. RTA ⊆ ROLES × TRUST_LEVELS defines the *role-trust_level* assignment relation. It is also a many-to-many relationship where a trust_level can be associated with many roles and same role can be performed with different trust_levels.

5. The function ush: USERS × SESSIONS_TYPES → SESSION_HISTORY defines a three-way relation between a user, a session_type and the trust history of the user in an earlier use of a session of that type. $ush(u, P) = {}_uh^P$, where $u \in$ USERS and $P \in$ SESSION_TYPES. A session_history ${}_uh^P$ is associated with a single user u and any session s^P of type P. A user can have many session_histories as a user can invoke many sessions of different types.

6. PA ⊆ PERMISSIONS × ROLES is a many-to-many permission to role assignment relation. An element in

PA is of type (p, r) where $p \in$ PERMISSIONS and $r \in$ ROLES.

7. The function $Assigned\_Roles : TRUST\_LEVELS \rightarrow 2^{ROLES}$ specifies the mapping of a trust_level $L(\subseteq [-1, 1])$ onto a set of roles. Formally, $Assigned\_Roles(L) = \{r \in ROLES \mid (r, L) \in RTA\}$. It implies, for any $l \in L, Assigned\_Roles(\{l\}) = Assigned\_Roles(L)$.

8. The function $Assigned\_Permission : ROLES \rightarrow 2^{PERMISSIONS}$ specifies the mapping of a role r onto a set of permissions. Formally, $Assigned\_Permission(r) = \{p \in PERMISSIONS \mid (p, r) \in PA\}$. This function is same as *assigned_permissions* function of RBAC model.

The constraints are applied on the above assignment functions depending on the access control policies of the system. Constraints on *Assigned_Roles* are similar to the constraints on user-role assignment in RBAC model. It specifies which roles are 'permitted' to be assigned to a certain trust_level. Constraints on *Assigned_Permissions* determines the assignment of permissions to a specific role. RBAC model suggests different constraints like *mutually exclusive role, prerequisite roles, cardinality constraints, static separation of duty, dynamic separation of duty* etc. But we prefer not to specify any particular constraint on these functions. Rather we leave it as general to give finer control in defining access control policies depending on the requirements of a system.

We also introduce a concept of **role dominance** among roles in our model. *Role dominance* is similar to the concept of *role hierarchies* in RBAC model. A role dominance relation, denoted by RD, defines a dominance relation between two roles. The dominance is described in terms of permissions. We define role dominance as,

Definition 1. Role dominance $RD \subseteq ROLES \times ROLES$ is a partial order on ROLES where the partial order is called a *Dominance* relation, denoted by $\preceq$. For any $(r_1, r_2) \in RD$, we say r_2 'dominates' r_1 only if all permissions assigned to r_1 are also permissions of r_2. Formally, $(r_1, r_2) \in RD \Rightarrow r_1 \preceq r_2$ and $r_1 \preceq r_2 \Rightarrow Assigned\_Permissions(r_1) \subseteq Assigned\_Permission(r_2)$.

The above definition implies that any user u having a role r_2 can have all the privileges of a user with role r_1.

The relation RD induces a similar relation called **trust_level dominance** among trust_levels in our model. Whenever there is a role dominance between two roles, there is a trust_level dominance between the corresponding trust_levels. Trust_level dominance, denoted by TLD is defined as follows:

Definition 2. Trust_level dominance, $TLD \subseteq TRUST\_LEVELS \times TRUST\_LEVELS$ is a partial order relation on TRUST_LEVELS and is denoted by $\leq'$. For any $(L_1, L_2) \in TLD$, we say L_2 'dominates' L_1 only if $L_1 \subseteq L_2$. If L_2 is a singleton set $\{l_2\}$, then dominance is defined as, $sup\{L_1\} \leq l_2$ that is, l_2 is greater than or equal to the maximum element of L_1. If both $L_1 = \{l_1\}$ and $L_2 = \{l_2\}$ are singletons then $L_1 \leq' L_2 \Rightarrow l_1 \leq l_2$ (the $\leq$ is the usual 'less equal to' relation of number theory).

The relation TLD is induced by RD. That is, for any $(r_1, r_2) \in RD, \exists (L_1, L_2) \in TLD$ such that $r_1 \in Assigned\_Roles(L_1)$ and $r_2 \in Assigned\_Roles(L_2)$. That is, the trust degree of a user with role r_2 is greater than that of a user with role r_1.

4. MODEL FOR EVALUATING TRUST RELATIONSHIPS

We adopt the vector model of trust that we had introduced earlier [25] for purpose of evaluating trust values of users. In this discussion we include the changes we have made. The interested reader is referred to [25] for the core model.

Definition 3. Trust is defined to be the firm belief in the competence of an entity to act dependably and securely within a specific context.

Definition 4. Distrust is defined as the firm belief in the incompetence of an entity to act dependably and securely within a specified context.

Although we define trust and distrust separately in our model, we allow the possibility of a neutral position where there is neither trust nor distrust.

We specify trust in the form of a trust relationship between two entities – the truster – an entity that trusts the target entity – and the trustee – the target entity that is trusted. This trust is always related to a particular context. An entity A needs not trust another entity B completely. A only needs to calculate the trust associated with B in some context pertinent to a situation. The specific context will depend on the nature of application and can be defined accordingly. Based on our current model, trust is evaluated under one context c only. The simple trust relationship $(A \xrightarrow{c} B)_t$ is a vector with three components – *experience*, *knowledge*, and *recommendation*. It is represented by $(A \xrightarrow{c} B)_t = [{}_AE_B^c, {}_AK_B^c, {}_\psi R_B^c]$, where ${}_AE_B^c$ represents the magnitude of A's experience about B in context c, ${}_AK_B^c$ represents A's knowledge and ${}_\psi R_B^c$ represents the cumulative effect of all B's recommendations to A from different sources.

To compute a trust relationship we assume that each of these three factors is expressed in terms of a numeric value in the range $[-1, 1] \cup \{\perp\}$. A negative value for the component is used to indicate the *trust-negative* type for the component, whereas a positive value for the component is used to indicate the *trust-positive* type of the component. A 0 (zero) value for the component indicates *trust-neutral*. To indicate a lack of value due to insufficient information for any component we use the special symbol $\perp$.

4.1 Computing the experience component

We model experience in terms of the number of events encountered by a truster A regarding a trustee B in the context c within a specified period of time $[t_0, t_n]$. An event can be trust-positive, trust-negative or, trust-neutral depending on whether it contributes towards a trust-positive experience, a trust-negative experience or, a trust-neutral experience. Intuitively, events far back in time does not count as strongly as very recent events for computing trust values. Hence we introduce the concept of *experience policy* which specifies a length of time interval subdivided into non-overlapping intervals. It is defined as follows.

Definition 5. An *experience policy* specifies a totally ordered set of non-overlapping time intervals together with a

set of non-negative weights corresponding to each element in the set of time intervals.

Recent intervals in the experience policy are given more weight than those far back. The whole time period $[t_0, t_n]$ is divided in such intervals and the truster A keeps a log of events occurring in these intervals.

If e_k^i denote the k^{th} event in the i^{th} interval, then we denote the value associated with e_k^i as v_k^i. This value is assigned according to relative importance of the event e_k^i. Then, $v_k^i \in [-10, 0)$ if $e_k^i \in \mathcal{Q}$, $v_k^i \in (0, 10]$ if $e_k^i \in \mathcal{P}$ and $v_k^i = 0$ if $e_k^i \in \mathcal{N}$ where, $\mathcal{P}$ = set of all trust-positive events, $\mathcal{Q}$ = set of all trust-negative events and $\mathcal{N}$ = set of all trust-neutral events. Here we modify our original definition of v_k^i a little. In [25] we did not distinguish between two trust-positive event or two-trust negative events. For all trust-positive events, we had $v_k^i = +1$ and for all trust-negative events $v_k^i = -1$. For this paper we choose to assign different weights to differenet events. This allows the truster to define relative importance of events.

The *incidents* I_j, corresponding to the j^{th} time interval is the normalized sum of the values of all the events, trust-positive, trust-negative, or neutral for the time interval. The normalization is done in such a way that $I_j \in [-1, 1]$. If n_j is the number of events that occurred in the j^{th} time interval, then

$$I_j = \begin{cases} \perp \quad \text{, if } \nexists\ e_k \in\ [t_{j-1}, t_j] \text{ for any } k \\ \frac{\sum_{k=1}^{n_j} v_k^j}{\sum_{k=1}^{n_j} |v_k^j|} \quad \text{, otherwise} \end{cases}$$

The *experience* of A with regards to B for a particular context c is given by

$$_A\mathrm{E}_B^c = \sum_{i=1}^{n} w_i I_i \tag{1}$$

where, w_i is a non-negative weight assigned to i^{th} interval.

4.2 Computing the knowledge component

The knowledge component has two parts - *direct knowledge* and *indirect knowledge (or, reputation)*. The truster A assigns two values to these two parts. Her *knowledge policy* regarding B in context c determines the weights to express relative importance between these two. Sum of the product of values and weights for the parts gives us a value for knowledge.

The *knowledge* of A with regards to B for a particular context c is given by

$$_AK_B^c = \begin{cases} d, \text{if } r = \perp \\ r, \text{if } d = \perp \\ w_d \cdot d + w_r \cdot r, \text{ if } d \neq \perp, r \neq \perp \\ \perp, \text{if } d = r = \perp \end{cases}$$

where $d, r \in [-1, 1] \cup \{\perp\}$ and $w_d + w_r = 1$. d and r are the values to direct and indirect knowledge respectively and w_d and w_r are the corresponding non-negative weights.

4.3 Computing the recommendation component

Recommendation is evaluated on the basis of a *recommendation value* returned by a recommender to A about B. Truster A uses the "level of trust" he has on the recommender in the context "to provide a recommendation" as a weight to the value returned. This weight multiplied by the former value gives the actual *recommendation score* for trustee B in context c.

The *recommendation* of A with regards to B for a particular context c is given by

$$_\Psi R_B^c = \frac{\sum_{j=1}^{n} (\mathbf{v}(A \xrightarrow{rec} j)_t^N) \cdot V_j}{\sum_{j=1}^{n} (\mathbf{v}(A \xrightarrow{rec} j)_t^N)} \tag{2}$$

where Ψ is a group of n recommenders, $\mathbf{v}(A \xrightarrow{rec} j)_t^N)$ = trust-value of j^{th} recommender and $V_j = j^{th}$ recommender's recommendation value about the trustee B.

4.4 Trust vector

We next observe that given the same set of values for the factors that influence trust, two trusters may come up with two different trust values for the same trustee. We believe that there are two main reasons for this. First, during evaluation of a trust value, a truster may assign different weights to the different factors that influence trust. The weights will depend on the trust evaluation policy of the truster. So if two different trusters assign two different sets of weights, then the resulting trust value will be different. The second reason is applicable only when the truster is a human being and is completely subjective in nature – one person may be more trusting than another. We believe that this latter concept is extremely difficult to model. At this stage we choose to disregard this feature in our model and assume that all trusters are trusting to the same extent. We capture the first factor using the concept of a *normalization policy*. The normalization policy is a vector of same dimension as of $(A \xrightarrow{c} B)_t$; the components are weights that are determined by the corresponding trust evaluation policy of the truster and assigned to experience, knowledge, and recommendation components of $(A \xrightarrow{c} B)_t$. The normalization policy together with the experience policy and the knowledge policy form the truster's *trust evaluation policy*.

We use the notation $(A \xrightarrow{c} B)_t^N$, called *normalized* trust relationship to specify a trust relationship. It specifies A's *normalized* trust on B at a given time t for a particular context c. This relationship is obtained from the simple trust relationship – $(A \xrightarrow{c} B)_t$ – after combining the former with the normalizing policy. It is given by $(A \xrightarrow{c} B)_t^N = \mathbf{W} \odot (A \xrightarrow{c} B)_t$.

The $\odot$ operator represents the normalization operator. Let $(A \xrightarrow{c} B)_t = [_AE_B^c, _AK_B^c, _\psi R_B^c]$ be a trust vector such that $_AE_B^c,\ _AK_B^c,\ _\psi R_B^c \in [-1, 1] \cup \{\perp\}$. Let also $\mathbf{W} = [W_E,\ W_K,\ W_R]$ be the corresponding trust policy vector such that $W_E + W_K + W_R = 1$ and $W_E, W_K, W_R \in [0, 1]$.

The $\odot$ operator generates the normalized trust relationship as

$$\begin{aligned} (A \xrightarrow{c} B)_t^N &= \mathbf{W} \odot (A \xrightarrow{c} B)_t \\ &= [W_E,\ W_K,\ W_R] \odot [_AE_B^c,\ _AK_B^c,\ _\psi R_B^c] \\ &= [W_E \cdot {}_AE_B^c,\ W_K \cdot {}_AE_B^c,\ W_R \cdot {}_\psi R_B^c] \\ &= [_A\hat{E}_B^c,\ _A\hat{K}_B^c,\ _\psi \hat{R}_B^c] \end{aligned}$$

We next introduce a concept called the *value* of a trust relationship. This is denoted by the expression $\mathbf{v}(A \xrightarrow{c} B)_t^N$ and is a number in $[-1, 1] \cup \{\perp\}$ that is associated with the normalized trust relationship. The special symbol $\perp$ is used to denote the value when there is not enough

information to decide about trust, distrust, or neutrality. This value now represents the trustee's trust degree.

Definition 6. The *value* of a normalized trust relationship $(A \xrightarrow{c} B)_t^N = [{}_A\hat{E}_B^c,\ {}_A\hat{K}_B^c,\ {}_\Psi\hat{R}_B^c]$ is a number in the range $[-1,1] \cup \{\perp\}$ and is defined as $\mathbf{v}(A \xrightarrow{c} B)_t^N = {}_A\hat{E}_B^c + {}_A\hat{K}_B^c + {}_\Psi\hat{R}_B^c$.

The value for a normalized trust relationship allows us to revise the terms trust and distrust as follows:

$$\mathbf{v}(A \xrightarrow{c} B)_t^N = \begin{cases} [-1,0) \ \Rightarrow \text{ it is } \textbf{distrust}. \\ 0 \ \Rightarrow \text{ it is } \textbf{neutral} \text{ (i.e., neither trust nor distrust).} \\ (0,1] \ \Rightarrow \text{ it is } \textbf{trust}. \\ \perp \ \Rightarrow \text{ it is } \textbf{undefined}. \end{cases}$$

Finally, we investigate the dynamic nature of trust – how trust (or distrust) changes over time. We make a couple of observations. First, trust depends on trust itself; that is a trust relationship established at some point of time in the past influences the computation of trust at the current time. If an agent is positively trusted to begin with then negative factors are often overlooked (that is given less weightage) when trust is re-evaluated in the agent. Second, trust decays with time. This is owing to the effect of forgetfulness of the human mind. The second idea is captured by the equation

$$\mathbf{v}(\vec{T_{t_n}}) = \mathbf{v}(\vec{T_{t_i}})e^{-(\mathbf{v}(\vec{T_{t_i}})\Delta t)^{2k}} \tag{3}$$

where, $\mathbf{v}(\vec{T_{t_i}})$, be the value of a trust relationship, $\vec{T_{t_i}}$, at time t_i and $\mathbf{v}(\vec{T_{t_n}})$ be the decayed value of the same at time t_n. We have developed a method to obtain a vector of same dimension as of $(A \xrightarrow{c} B)_t^N$ from this value $\mathbf{v}(\vec{T_{t_n}})$. The effect of time is captured by the parameter k which is determined by the truster A's *dynamic policy* regarding the trustee B in context c. The current normalized vector together with this time-affected vector are combined according to their relative importance. Relative importance is determined by truster's *history_weight policy* which specifies two values α and β in $[0,1]$ (where, $\alpha + \beta = 1$) as weights to current vector and the vector obtained from previous trust value. The new vector thus obtained gives the actual normalized trust vector at time t for the trust relationship between truster A and trustee B in context c. This is represented by the following equation

$$(A \xrightarrow{c} B)_{t_n}^N = \begin{cases} [{}_A\hat{E}_B^c, {}_A\hat{K}_B^c, {}_\Psi\hat{R}_B^c], & \text{if } t_n = 0 \\ [\frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}], & \text{if } t_n \neq 0 \text{ and } {}_A\hat{E}_B^c = {}_A\hat{K}_B^c = {}_\Psi\hat{R}_B^c = \perp \\ \alpha \cdot [{}_A\hat{E}_B^c, {}_A\hat{K}_B^c, {}_\Psi\hat{R}_B^c] + \beta \cdot [\frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}], & \text{if } t_n \neq 0 \text{ and at least one of } {}_A\hat{E}_B^c, {}_A\hat{K}_B^c, {}_\Psi\hat{R}_B^c \neq \perp \end{cases}$$

where $[\frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}, \frac{\mathbf{v}(\hat{T})}{3}]$ is the time-effected vector and $\mathbf{v}(\hat{T}) = \mathbf{v}(\vec{T_{t_n}})$.

5. ACCESS CONTROL USING TRUSTBAC

Basic purpose of an access control mechanism is to protect system resources by restricting the user's activities on them. A user's authorization to perform certain tasks on specific resources is specified by the access control policy of the system. When using TrustBAC for access control, a *user* invokes a *session_instance* of a particular type at an instant of time. During this session the user has a *trust_level* which allows her to use the *roles* associated with that trust_level. That is, a user can be a member of a role. Also a single role can be exercised by many users. For each of these roles, the user has a set of *permissions*. Therefore, the user is restricted to perform a set operations on a particular set of resources as specified by the set of permissions obtained as a member of those roles.

A first time user u registers with the system and logs in which instantiates a session_instance s of the user. Depending on the set of disclosed properties P, the system invokes the function *sua* with arguments u, s, and P to start a session s^P. The system initiates a trust relationship $(SYS \xrightarrow{P} u)_t^N$ with the user in that session. The underlying context of this trust relationship is identified by the session_type P. This relationship does not change, but gets updated for any other session of same type P invoked by the same user u. If the user invokes another session_instance of type P' at time t, then the system creates another trust relationship $(SYS \xrightarrow{P'} u)_t^N$. The value of the trust relationship $(SYS \xrightarrow{P} u)_t^N$ is evaluated for the session s^P. Let $\mathbf{v}(SYS \xrightarrow{P} u)_t^N = l,\ l \in [-1,1]$. The system invokes the function *Assigned_Roles* to determine the roles that the user u can execute. Let $Assigned\_Roles(\{l\}) = \{r_1, r_2, \ldots, r_n\}$. u can choose to execute more than one of these n roles. With each r_i, u has a set of p_js where $\forall j, (p_j, r_i) \in PA$. Therefore, in a session s^P, the user u has the set of permissions given by, $\bigcup_i Assigned\_Permissions(r_i) = \bigcup_{1 \leq i \leq n}\{p_{ji} \mid (p_j, r_i) \in PA\}$. Hence, the user u is restricted to perform actions A on a set of objects O where $\exists$ a $p_{ji} \in \bigcup_{1 \leq i \leq n}\{p_{ji} \mid (p_j, r_i) \in PA\}$, such that, for any $(o,a) \in O \times A,\ (o,a) \in p_{ji}$. The user executes these actions on the allowed objects and each activity during that session s^P is stored as the session_history ${}_uh^P$ for that session. Whenever the trust_level is re-evaluated (within s^P or, at the start of next instance s' of a session of type P), the *events* in ${}_uh^P$ are evaluated. The evaluated trust_level, say l' overwrites l in ${}_uh^P$. The subsequent events also overwrite the previous event log.

We assume that for a registered user u in a session s^P, the trust relationship $(SYS \xrightarrow{P} u)$ is managed by a diligent system-administrator who is outside the scope of this access control framework. We denote this system-admin by the symbol SYS. We also assume that the system has two predefined policies – an access control policy $\mathcal{A}_{sys}$ and a trust evaluation policy $\mathcal{T}_{sys}$ which are not independent. $\mathcal{A}_{sys}$ defines the functions *Assigned_Roles* and *Assigned_Permissions* together with the 'constraints' on them. The components are evaluated as

Computing *knowledge* The user u initiates the session s^P by disclosing a set of properties P, which includes information (e.g., name, address, affiliation, etc.) as well as some credentials. Credentials are in the form of typical digital certificates. The system assign a value within $[-1,1]$ as weights to the information and the credentials. The assignment is done as specified by $\mathcal{T}_{sys}$ and ${}_{SYS}K_u^P$ is computed according to the equation 4.2. The next instance of a session of type P, the values assign to members of P may change due

to change in values in P. For example, the user disclose the same type of certificate, but with a different certifying authority.

Computing *experience* As mentioned in section 4.1, experience is computed from the *events* occurred during some intervals. Our model does not dictate about the length of an interval. It depends on implementation – the system may choose to identify a whole session as an interval. Independent of the length of an interval, any *action* performed by the user is identified as an 'event'. This record is kept in session_history ${}_uh^P$ till the next instant of trust evaluation. Formally, let l be the trust_level of u in a session s^P. Let $Assigned\_Roles(l) = \{r_1, r_2, \ldots, r_n\}$ of which u activate r_1, r_2, r_3. These are the *active roles* of u in session s^P. The events are the set of actions $\mathbb{A}$ where for any $a \in \mathbb{A}$, $\exists\ p \in \bigcup_{1 \le i \le 3} Assigned\_Permissions(r_i)$. The weight to the result of a particular action is assigned according to $\mathcal{T}_{sys}$ and the experience ${}_{SYS}E_u^P$ is computed as specified by equations 1.

Computing *recommendation* The system may take *role-specific* and *role-independent* input from other users about u in a session. These information constitute u's recommendation and the component ${}_{\Psi}R_u^P$ is calculated using equation in 2. Ψ is the set of other users who provide recommendation for u to SYS. However, we choose not to specify how these information are collected.

After computing the components, the system calculates the normalized trust by combining $(SYS \xrightarrow{P} u)_t$ and the normalization policy of $\mathcal{T}_{sys}$. Then the previous trust_level is fetched from ${}_uh^P$ and final $(SYS \xrightarrow{P} u)_t^N$ is calculated using the equation 4.4. The corresponding value $\mathbf{v}(SYS \xrightarrow{P} u)_t^N$ is calculated as specified in the section 4.4. This value denotes the current trust_level of u in a session of type P and gets stored in corresponding session_history ${}_uh^P$.

Note, TrustBAC does not verify the credentials disclosed by the user. The TrustBAC module includes a trust evaluation module which computes and stores all relevant trust information including session_history. It interacts with two policy specifier modules which stores $\mathcal{A}_sys$ and $\mathcal{T}_sys$. Authenticity and veracity of credentials are checked by a suitable module outside the framework. It passes the necessary information to trust evaluation module. A compliance checker and access controller module can be implemented to enforce the access privileges according to the trust decisions. It interacts with the access specifier and the trust evaluation module to enforce the proper access privileges. These modules are outside TrustBAC framework and a part of the application which work in conjunction with TrustBAC.

5.1 Example

Let us consider an example to show how the TrustBAC framework works. For this purpose we assume that a digital library system DL uses TrustBAC to control access privileges of its users for the resource present in that DL. The DL has an access control policy $\mathcal{A}_{DL}$ and a trust evaluation policy $\mathcal{T}_{DL}$. Let *basic user* and *privilege user* be two roles in the ROLES set of the digital library. We assume $\mathcal{A}_{DL}$ specifies the following: $Assigned\_Roles([0.05, 0.4]) = basicuser$ and $Assigned\_Roles([0.35, 0.6]) = privilegeuser$. Let a user u log in to the system and manifest a set of credentials c (for simplicity we assume that user properties are expressed in terms of credentials). The system initiates a session s^c and the trust relationship $(DL \xrightarrow{c} u)_t^N$ is considered. The credentials are verified and evaluated and the corresponding value is stored in ${}_{DL}K_u^c$. The session_history ${}_uh^c$ is consulted and the trust is evaluated as $\mathbf{v}(DL \xrightarrow{c} u)_t^N = 0.45$. Therefore, according to $Assigned\_Roles$ the user at this stage is allowed to act as a *privilege user* as well as a *basic user*. Let the user select the role of *privilege user*. Let the privilege users of DL be allowed to write comment about the articles present in the database as well as can upload digital copies of articles that are not present in the database. Let the system consider abusive/irrelevant comments as negative events and upload of a corrupted or inauthentic file as negative event. Let u during the session s^c write several bad comments and upload a few inauthentic files. Each of these activities get reported in the session_history ${}_uh^c$. To handle recommendation we assume that DL is a part of a digital library consortium where the member DLs are linked to one another. During the session the DL system sends messages requesting for recommendation from other members of the consortium about u. Let $\mathcal{T}_B$ evaluate trust periodically within a session. Let at some evaluation point $\mathbf{v}(DL \xrightarrow{c} u)_t^N = 0.345$. This shows that u is no longer 'trustworthy' to the system as a *privilege user*. The system automatically withold the role of *privilege user* for u. During the remaining time in this session u can no longer act as a *privilege user*. So if there is a section of articles in the DL which is only available to *privilege users* then u can not access those articles anymore. However, u can continue to act as a *basic user*. Otherwise she may logout. The next time u logs in with properties c, u can only perform the role of a *basic user*. Good actions and good recommendations can increase the trust_level for u and when it reaches 0.35, u is again able to act as *privilege user*. Alternatively, u can produce some extra credential (something like a special permission from the digital library authority) in the new session to raise her trust_level. However, the set of extra credential alone may not be sufficient to raise the trust_level. u may still need to behave well. How the trust_level decreases for bad behavior or increases with extra set of credentials depends on $\mathcal{T}_{DL}$. u can deliberately perform malicious actions as a *privilege user* to get personal benefit without caring about her trust_level. For example she may want to decrease the rating of a article written by someone she hates by putting bad comments about it. When she is restricted to perform as a *basic user* only then she starts behaving well to increase her trust_level to act again as a *privilege user* and repeats the cycle. To prevent this type building trust and then milking the system can be prevented by "slow-to-increase' and 'fast-to-decrease' policy. The $\mathcal{T}_{DL}$ can be so configured that every bad action is heavily penalized to lower the trust_level rapidly. Every good action adds only a little amount to the trust_level. So it will either need extra credentials and set of good recommendation or consistent good behavior over a series of sessions.

6. CONCLUSION

In this work, we introduce the TrustBAC model for access control in open systems. The model extends the RBAC model by introducing the notion of trust levels. Instead of users being assigned to roles as in traditional RBAC,

users are assigned to trust levels. The user to trust level assignment is determined by three factors – user's past behavior, knowledge about the user (for example, credentials presented by the user) and recommendation provided by others about the user. The system may choose one or all of these factors in deciding on the trust level. Trust levels are assigned to roles according to organizational policies. Roles are assigned to permissions as in the traditional RBAC model.

The TrustBAC model being an extension of the RBAC model has all the latter's advantages. In addition, it borrows from credential based access control models in the sense that TrustBAC relies on evaluation of user's trustworthiness for access control. The model does not preclude use of credentials for such evaluation of trustworthiness. Thus the model is well suited for open systems like the Internet.

A lot of work remains to be done. To begin with, we plan to incorporate recent extensions to the basic RBAC model to give the TrustBAC model much richer semantics. Next we plan to develop a policy language for expressing access control policies in the TrustBAC model. We plan to culminate our efforts by developing a permission management infrastructure built around this model much in the same manner as the PERMIS project (http://sec.isi.salford.ac.uk/permis/) is doing for traditional RBAC.

Acknowledgment

This work was partially supported by the U.S. Air Force Research Laboratory (AFRL) and the Federal Aviation Administration (FAA) under contract F30602-03-1-0101. The views presented here are solely that of the authors and do not necessarily represent those of the AFRL or the FAA. The authors would like to thank Mr. Robert Vaeth of the AFRL and Mr. Ernest Lucier of the FAA for their valuable comments and their support for this work.

7. REFERENCES

[1] M. Abadi and C. Fournet. History-based Access Control for Mobile Code. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium*, pages 107–121, San Diego, California, USA, February 2003.

[2] N. R. Adam, V. Atluri, E. Bertino, and E. Ferrari. A Content-Based Authorization Model for Digital Libraries. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):296–315, March 2002.

[3] L. Bauer, M. A. Schneider, and E. W. Felten. A General and Flexible Access-Control System for the Web. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, California, USA, August 2002.

[4] E. Bertino, B. Catania, and M. L. Damiani. GEO-RBAC: A Spatially Aware RBAC. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT'05)*, pages 29–37, Stockholm, Sweeden, June 2005.

[5] R. Bhatti, E. Bertino, and A. Ghafoor. A Trust-based Context-Aware Access Control Model for Web-Services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 184–191, San Diego, California, USA, June 2004.

[6] R. Bhatti, J. Joshi, E. Bertino, and A. Ghafoor. Access Control in Dynamic XM-based Web-Services with X-RBAC. In *Proceedings of the 1st International Conference on Web Services*, Las Vegas, Nevada, USA, June 2003.

[7] M. Blaze, J. Feigenbaum, and J. Ioannidis. The KeyNote Trust Management System Version 2. Internet Society, Network Working Group. RFC 2704, 1999.

[8] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proceedings of 17th IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, California, USA, May 1996.

[9] P. Bonatti and P. Samarati. Regulating Service Access and Information Release on the Web. In *Proceedings of the 7th ACM COnference on Computer and Communication Security*, pages 134–143, Athens, Greece, November 2000.

[10] D. Chadwick, A. Otenko, and E. Ball. Role-Based Access Control with X.509 Attribute Certificates. *IEEE Internet Computing*, 7(2):62–69, March/April 2003.

[11] E. Cohen, R. K. Thomas, W. Winsborough, and D. Shands. Models for Coalition-based Access Control (CBAC). In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT'02)*, pages 97–106, Monterey, California, USA, June 2002.

[12] E. Damiani, S. Vimercati, and P. Samarati. New Paradigm for Access Control in Open Environments. In *Proceedings of the 5th IEEE Symposium on Signal Processing and Information Technology (ISSPIT'05)*, Athens, Greece, December 2005.

[13] G. Edjlali, A. Acharya, and V. Chaudhary. History-based Access Control for Mobile Code. In *Proceedings of the 5th ACM Conference on Computer and Communication Security (CCS'98)*, pages 38–48, San Francisco, California, USA, November 1998.

[14] D. Ferraiolo and R. Kuhn. Role-Based Access Controls. In *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, pages 554–563, Bultimore, Maryland, USA, October 1992.

[15] D. Ferraiolo, R. Sandhu, S. Gavrila, R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and Systems Security*, 4(3):224–274, August 2001.

[16] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible Team-based Access Control Using Contexts. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies (SACMAT'01)*, pages 21–27, Chantily, Virginia, USA, May 2001.

[17] A. Harrington and C. Jensen. Cryptographic Access Control in a Distributed File System. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT'03)*, pages 158–165, Como, Italy, June 2003.

[18] H.M.Gladney. Access Control for Large Collections. *ACM Transactions on Information Systems*, 15(2):154–194, April 1997.

[19] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A Generalized Temporal Role-Based Access Control Model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, January 2005.

[20] N. Li and J. Mitchell. Datalog with Constraints: A Foundation for Trust-management Languages. In *Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, New Orleans, Louisiana, January 2003.

[21] N. Li and J. Mitchell. RT: A Role-based Trust Management Framework. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition*, Washington D.C., April 2003.

[22] N. Li, J. Mitchell, and W. Winsborough. Design of a Role-Based Trust-Management Framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130, Oakland, California, May 2002.

[23] N. Li, W. Winsborough, and J. Mitchell. Beyond Proof-of-Compliance: Safety and Availability Analysis in Trust Management. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Oakland, California, May 2003.

[24] J. S. Park and J. Hwang. Role-based Access Control for Collaborative Enterprise In Peer-to-Peer Computing Environments. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT'03)*, pages 93–99, Como, Italy, June 2003.

[25] I. Ray and S. Chakraborty. A Vector Model of Trust for Developing Trustworthy Systems. In *Proceedings of the 9th European Symposium of Research in Computer Security (ESORICS 2004)*, volume 3193 of *Lecture Notes in Computer Science*, pages 260–275, Sophia Antipolis, France, September 2004.

[26] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, February 1996.

[27] R. Sandhu and X. Zhang. Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT'05)*, pages 147–158, Stockholm, Sweeden, June 2005.

[28] R. K. Thomas. Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments. In *Proceedings of the Second ACM Workshop on Role-based Access Control (RBAC'97)*, pages 13–19, Fairfax, Virginia, USA, November 1997.

[29] T. van den Akker, Q. O. Snell, and M. J. Clement. The YGuard Access Control Model: Set-based Access Control. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT'01)*, pages 75–84, Chantily, Virginia, USA, May 2001.

[30] H. F. Wedde and M. Lischka. Role-Based Access Control in Ambient and Remote Space. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT'04)*, pages 21–30, Yorktown Heights, New York, USA, June 2004.

[31] M. Winslett, N. Ching, V. Jones, and I. Slepchin. Assuring security and privacy for digital library transactions on the Web: client and server security policies. In *Proceedings of the IEEE international forum on Research and Technology Advances in Digital Libraries*, pages 140–151, Washington, DC, USA, May 1997.