Distributed Role-based Access Control for Coalition Application

HONG Fan ZHU Xian XING Guanglin

ABSTRACT Access control in multi-domain environments is one of the important questions of building coalition between domains. On the basis of RBAC access control model, the concepts of role delegation and role mapping are proposed, which support the third-party authorization. Then, a distributed RBAC model is presented. Finally the implementation issues are discussed.

KEYWORDS role; access control; multi-domain; delegation; mapping

CLC NUMBER TP309

Introduction

With the development of distributed systems, access control in multi-domain environments is required in more and more applications. Many new requirements are proposed in References [1-3]. RBAC96 has many excellent features to support access control in distributed environments [1]. It has been suggested that such models provide an attractive theoretical framework for multi-domain and distributed systems [1]. However, traditional RBAC systems depend upon a central trusted computing base administered by a single authority, which contains all of the security policy for the entire organization. It cannot be scaled for large number of mutually anonymous users such as one might encounter in coalition settings. On the basis of RBAC96 and related researches, we developed an access control model called distributed role-based access control model for multi-domain environments (MDDRBAC). The model extends RBAC96 with the concepts of role mapping and role delegation, which supports third-party authorization. With these mechanisms, the resources in the coalition can be shared securely between domains. And it has the advantages of decentralized management and scalability.

1 MDDRBAC and related concepts

MDDRBAC is based on RBAC96 and inherits the related concepts and definition in RBAC96. In addition, the following concepts presented informally in this section are included.

A secure domain is an administrative scope that is managed with a single RBAC secure policy. For example, every RBAC system in the distributed environments is a secure domain [1]. We called it domain in this paper for abbreviation.

The core concept of RBAC is role. In MDRBAC, we also take this approach. The user in foreign domain can acquire the role in local domain when it is permitted by the security policy. The user can access the local resource securely.

The permissions are transmitted separately as administrative permissions and usage permissions with the RBAC policy. Administrative permissions are the permissions to assign user to the role, and usage permissions are the permis-
sions authorized to the role. In essence, the interaction between multi-domains is the issues that the user in foreign domain can get the usage permissions that the role in local domain has. To achieve this, the best solution is to map the foreign role to the local roles, which can be utilized by specifying the mapping relation directly between the two domains, or by granting a trusted third-party to specify the mapping relation via a delegation mechanism. We define two kinds of relations as follows.

1. Role delegation relations allow a user who has the administrative permissions over a role $A. R_1$, denoting the role $R_1$ in domain $A$, to delegate the administrative permissions to another role or user, which grants the latter to specify the role mapping relation on $A. R_1$ and grant the latter to further delegate the administrative permissions to another role or user. If the latter is a role, then all users authorized to the role get the administrative permissions. Delegation must be limited by depth.

2. Role mapping relations allow a user who has the administrative permissions over a role $A. R_1$ to map the role $B. R_2$ in other domain to $A. R_1$, which make all users authorized to $B. R_2$ get the usage permissions authorized to $A. R_1$. The mapping relations are not transitive, which means the role attained from a role mapping relation can not be used to do another role mapping.

Delegation is necessary for the scalability of a distributed system. Delegation expresses the trust among different entities. As role delegation is transitive but trust is not, the role delegation must be limited by depth. The two kinds of relations are implemented as certificates, which prevents the data from forging and tampering when transmitted over the network.

Every domain has the administrative permissions over its local roles, so every domain can issue the certificate related to its local roles discretionarily. The third-party can also issue certificate $X$ related to role $A. R_1$, while it is valid only if there is another certificate $Y$ proving that the $X$’s issuer has the administrative permissions over $A. R_1$. We call $Y$ the supporting certificate of $X$. Supporting certificate will need other supporting certificate sometimes. The sequence of certificate and some supporting certificates is called certificate chain.

As for the role delegation relations, because delegating to a role or an arbitrary user would cause some complex problems such as certificate revocation, discovery and verification, the model’s current version will only allow delegating a role to another domain (i.e. to the domain’s administrator).

2 Formal definition of MDDRBAC

**Definition 1** $D$ is a set of domains. An arbitrary element of $D$, say $d_i$, is a 6-tuple $d_i = (U_i, R_i, P_i, UA_i, PA_i, RH_i)$, where $RH_i \subseteq RH$. $U_i$, $R_i$, $P_i$, $UA_i$, $PA_i$ and $RH_i$ are the global data, respectively, and $U_i \subseteq U$, $R_i \subseteq R$, $P_i \subseteq P$, $UA_i \subseteq UA$, $PA_i \subseteq PA$.

**Definition 2** In MDDRBAC, domain must possess the following properties:
1) The set $\{R_1, R_2, \ldots, R_n\}$ is a partition of $R$, where $n = \#D$.
2) Property 1) holds with $R$ being substituted for $P$, $UA$, $PA$ and $RH$, respectively.
3) Sometimes, $U_i \cap U_j$ $(i \neq j) \neq \emptyset$, which means user can belong to two or more domains.

**Definition 3** says the relation between domains in MDDRBAC is peer-to-peer.

**Definition 3**
1) $Domain_R: R \rightarrow D$ is a function that maps each role to a single domain it belongs to.
2) $Domain_P: P \rightarrow D$ is a function that maps each permission to a single domain it belongs to.
3) $Domain_UA: UA \rightarrow D$ is a function that maps each $UA$ to a single domain it belongs to.
4) $Domain_PA: PA \rightarrow D$ is a function that maps each $PA$ to a single domain it belongs to.
5) $Domain_RH: RH \rightarrow D$ is a function that maps each $RH$ to a single domain it belongs to.
6) $Domains_U: U \rightarrow 2^D$ is a function that maps each user to a set of domains it belongs to.
7) $Domain_S: S \rightarrow D$ is a function that maps each session to a single domain it has logged in or it had logged in.
Definition 4 In MDDRBAC, the following properties must hold:
1) \( \forall (u,r) \in UA \rightarrow \text{Domain}_{UA}(u,r) = \text{Domain}_R(r) \);
2) \( \forall (p,r) \in PA \rightarrow \text{Domain}_{PA}(p,r) = \text{Domain}_R(r) \);
3) \( \forall (r_1,r_2) \in RH \rightarrow \text{Domain}_{RH}(r_1,r_2) = \text{Domain}_R(r_1) = \text{Domain}_R(r_2) \);

Definition 4 dictates that the relations including UA, PA and RH must not be built crossing domain in MDDRBAC.

Definition 5
1) Time set \( T = \{ t \mid t \geq 0, t \in R^1 \} \), where \( R^1 \) is real number set;
2) Time interval set \( I = \{ (t_1,t_2) \mid t_1 \leq t_2, t_1 \in T, t_2 \in T \} \), where \( t_1 \) is the start time and \( t_2 \) is the end time;
3) We say a time \( t_i \) is within an interval \( (t_3,t_4) \) if \( t_3 \leq t_i \leq t_4 \), which can also be written as \( t_i \in (t_3,t_4) \);
4) We say an interval \( (t_1,t_2) \) is within \( (t_3,t_4) \), if \( t_1 \geq t_3 \) and \( t_2 \leq t_4 \), which can also be written as \( (t_1,t_2) \subseteq (t_3,t_4) \);

Definition 6 There is a role delegation relation \( \text{DLGT} \subseteq R \times D \times T \times I \times N \times N \) between domains, where \( N \) is natural number set. A tuple \( (d_1,r,d_2,t,i,n_1,n) \) in DLGT means that a domain \( d_1 \) delegates the role \( r \) to domain \( d_2 \) at time \( t \) with depth \( n_1 \); if \( n_1 > 0 \), then \( d_1 \) can further delegate \( r \); the delegation relation tuple is valid within interval \( i \) and is identified by number \( n \).

We have the following rule for DLGT:
\( \forall (d_1,r,d_2,t,i,n_1,n) \in \text{DLGT} \rightarrow d_1 \neq d_2 \)

Definition 6 says that:
1) a domain must not delegate a role to itself, it is of no sense.
2) the start time of a valid interval must be after the creating time of delegation.

When \( \text{Domain}_R(r) = d_2 \), it denotes that domain \( d \) delegate its local role to another domain, which we call root delegation, otherwise third-party delegation.

The trust represented by role delegation relation is transitive.

Definition 7 There is a role mapping relation \( \text{MAP} \subseteq R \times R \times D \times T \times I \times N \) between domains, where \( N \) is natural number set. A tuple \( (r_1,r_2,\text{d_1},t,i,n) \) in MAP means that a domain \( d \) map role \( r_1 \) to role \( r_2 \) at time \( t \), the mapping relation tuple is valid within interval \( i \) and is identified by number \( n \).

We have the following rule for MAP:
\( \forall (r_1,r_2,\text{d_1},t,i,n) \in \text{DLGT} \rightarrow \text{Domain}_R(r_1) \neq \text{Domain}_R(r_2) \)

Definition 7 says that:
1) a domain must not map a role in a domain to another role in the same domain. There would rather use the RH than use the role mapping relation to represent the relation among roles in the same domain.
2) the start time of valid interval must be later than the creating time of mapping.

Definition 8
1) Prior1: \( \text{DLGT} \rightarrow 2^{\text{DLGT}} \) is a function that maps each DLGT to a set of DLGTs.
\( \text{prior1}(d_1,r,d_2,t,i,n_1,n) = \{ (d_1,r_1,d_1,t_1,i_1,n_1',n_1') \mid r_1 \neq r_2, d_1 = d_2, t_1 \in i_1, i_1 \subseteq i \} \)
2) Prior2: \( \text{MAP} \rightarrow 2^{\text{DLGT}} \) is a function that maps each MAP to a set of DLGTs.
\( \text{prior2}(r_1,r_2,d,t,i,n) = \{ (d_1,r_1,d_1,t_1,i_1,n_1') \mid r_1 \neq r_2, d_1 = d, t_1 \in i_1, i_1 \subseteq i \} \)
3) We use the prior1 and prior2 functions to define the supporting relationship between certificates, which represent the mapping and delegation relation tuples.

if \( \forall c_1,c_2 \in \text{DLGT} \) \( (c_1 \in \text{prior1}(c_2)) \), or \( \forall c_1 \in \text{DLGT}, c_2 \in \text{MAP} \) \( (c_1 \in \text{prior2}(c_2)) \), then we say certificate \( c_1 \) supports certificate \( c_2 \), written as \( c_1 \vdash c_2 \).

4) As to sequence \( (c_1,c_2,\cdots,c_n) \), where \( c_i \in \text{DLGT} \) or \( c_i \in \text{MAP} \), if \( c_i \vdash c_{i+1} \) for \( 0 \leq i < n \), then we call it certificate chain. The certificate chain is called rooted if \( c_1 \) is a root delegation or self mapping.

Definition 8 says that:
1) the depth field \( n_1 \) in a certificate must smaller than \( n_1 \) in the previous certificate in a certificate chain, which restricts the chain’s length effectively.
2) The role field \( r_1 \) or \( r_2 \) must be dominated by
the role field \( r \) in the previous certificate in a certificate chain, which demonstrates the consideration of role hierarchy.

2) Only the last certificate may be a role mapping certificate in a certificate chain, and other certificate must be role delegation certificate.

**Definition 9** There is a revocation relation \( \text{REVOKE} \subseteq D \times N \times T \times I \) between domains, where \( N \) is a natural number set. A tuple \((d,n,t,i)\) in \( \text{REVOKE} \) means that; a domain \( d \) revokes the tuple identified by number \( n \) in DLGT or MAP at time \( t \), the tuple being revoked is invalid within interval \( i \).

We have the following rule for \( \text{REVOKE} \):

\[ \forall (d,n,t,i) \in \text{REVOKE} \rightarrow (\exists (r_1, r_2, d, t_1, i, n_1, n) \in \text{MAP} \wedge d_1 = d) \wedge (\exists (d_1, r, d_2, t_1, i, n_1, n) \in \text{DLGT} \wedge d_2 = d) \]

Definition 9 says that only the domain having issued a certificate can revoke the certificate. As to \( \text{REVOKE} \), there are no constraints between \( T \) and \( I \).

**Definition 10**

1) Valid1: \( \text{DLGT} \times T \rightarrow \{\text{TRUE}, \text{FALSE}\} \) is a function that represents each DLGT's validity. When \( \text{valid1}(d_1, r, d_2, t_1, i, n_1, n) = \text{TRUE} \), it denotes that \((d_1, r, d_2, t_1, i, n_1, n)\) is valid at time \( t_1 \), otherwise is invalid.

2) Valid2: \( \text{MAP} \times T \rightarrow \{\text{TRUE}, \text{FALSE}\} \) is a function that represents each MAP's validity. When \( \text{valid2}(r_1, r_2, d, t_1, i, n, t_2) = \text{TRUE} \), it denotes that \((r_1, r_2, d, t_1, i, n)\) is valid at time \( t_1 \), otherwise is invalid.

**Definition 11** In MDDRBC, we have the following rules:

1) \( \text{valid1}(d_1, r, d_2, t_1, i, n_1, n) = \text{TRUE} \), if \((t_1 \in i) \wedge (\neg \exists (d_1, n, t', i') \in \text{REVOKE} (t_1 \in i')) \wedge (\text{Domain}_R(r) = d_2) \wedge (\exists (d_1, r, d_2, t, i, n_1, n) \in \text{prior1}(d_1, r, d_2, t, i, n_1, n) \wedge \text{valid1}(d_1, r, d_2, t, i, n_1, n) = \text{TRUE}) \)

2) \( \text{valid2}(r_1, r_2, d, t_1, i, n, t_2) = \text{TRUE} \), if \((t_2 \in i) \wedge (\neg \exists (d, n, t', i') \in \text{REVOKE}(t_2 \in i')) \wedge (\text{Domain}_R(r) = d) \wedge (\exists (d, r, d', t', i', i, n, n) \in \text{prior2}(r_1, r_2, d, t_1, i, n) \wedge \text{valid1}(d_1, r_2, d_2, t_1, i, n_1, n) = \text{TRUE}) \)

In the definition 11, 1) says that \((d_1, r, d_2, t, i, n)\) is valid at some time, if and only if the time is within the certificate's valid interval. the time is not within the revocation interval if the certificate is revoked and it is a root delegation or it has a supporting certificate which is valid at the certificate's creating time. 1) specifies the delegation certificate's validity rule recursively. The value of function valid1 is TRUE only if the recursion is ended with a root delegation. 2) expresses the similar meaning to 1).

When a certificate's prior is null, the certificate is a root delegation, a self mapping or an invalid certificate.

**Definition 12** As for a certificate chain \( (c_1, c_2, \ldots, c_n) \), if the following conditions are satisfied:

- certificate \( c_k \) is valid at time \( t \), and
- certificate \( c_k \) is valid at the issuing time of certificate \( c_{k+1} \), where \( 0 < k < n \),

then we say the certificate chain is valid at time \( t \).

As for third-party authorization, there must be a rooted and valid certificate chain to support the issuer to issue a certificate.

**Definition 13**

1) \( \text{DL}(r,t) \rightarrow 2^\text{MAP} \) is a function that maps a role and a time to a set of MAPs \( \text{DL}(r,t) = \{(r_1, r, d, t_1, i, n) \in \text{MAP} | r_1 \leq r \wedge \text{valid2}(r_1, r_2, d, t_1, i, n) = \text{TRUE}\} \).

\( \text{DL}(r,t) \) dictates all valid role mapping certificates that role \( r \) has at time \( t \).

2) \( \text{Roles1}: R \times D \times T \rightarrow 2^R \) is a function that maps a role, a domain and a time to a set of roles.

\( \text{Roles1}(r, d, t) = \{r_1 | (r_1, r_2, d, t_1, i, n) \in \text{DL}(r, t_1) \wedge \text{Domain}_R(r_2) = d\} \).

\( \text{Roles1}(r, d, t) \) dictates the role set that role \( r \) can attain directly in domain \( d \) with role mapping certificates at time \( t \).

3) \( \text{Roles2}: S \rightarrow 2^R \) is a function that maps a session to a set of roles.

\( \text{Roles2}(s) \subseteq \{r | (\exists r_1 \geq r)(r_1 \in \text{role1}(r_2, d, t_1)) \wedge d = \text{Domain}_S(s) \wedge \text{Domain}_UA(user(s), r_2) \in \text{Domains}_U(user(s)) \} \cup \text{Roles}(s), \) where \( t \) is the session's creating time, and session \( s \), has the permissions \( \cup_{r \in \text{roles}(s)} \{p | (\exists r_1 \leq r)(p, r_1) \in PA\} \).

\( \text{Roles2}(s) \) dictates that the role set can be obtained.
Definition 13 says that:

1. The role mapping is not transitive. From \((r_1, r_2, d, t, i, n)\) and \((r_2, r_3, d', t', i', n')\) we cannot draw the conclusion that user in role \(r_1\) can attain role \(r_3\).

2. With role mapping relations, a user assigned to a role (or a role senior to the role) in foreign domain can get some role \(r\) and all roles junior to \(r\) in local domain.

3. When a user logs in a domain, the user can acquire the roles authorized directly to the user in the domain, and the user can also acquire some roles by the authorization data between domains and by the roles assigned to the user in other domains.

3 Implementation

We have implemented the MDDRBAC with Java RMI. It is a distributed access control infrastructure that supports the coalition between multi-domain. The tuples in DLGT and MAP are implemented as certificates, which are signed by the issuer’s private key and are transmitted to the login domain to be verified.

3.1 System components

Fig. 1 is the MDDRBAC component architecture.

Every domain has a MDDRBAC engine, which is composed of MDDRBAC component and RBAC component. The two components also include a GUI, which is developed as Java application. RBAC component is responsible for RBAC function such as user addition, user-role assignment etc. MDDRBAC component provides GUI to issue role mapping and role delegation certificate. The message exchanged between MDDRBAC components is done via Java RMI. The certificates and RBAC data are stored in the MDDRBAC DB. The information about domains which user belongs to is stored in the LDAP server, so the related algorithm need not traverse all domains to collect the information. MDDRBAC utilizes the PKI to identify all the entity engaged in trust-sensitive operators and to validate certificates. OpenCA and OpenLDAP are used in our implementation as PKI and LDAP server.

The MDDRBAC infrastructure can be used to manage most security issues except the access control enforcement function. MDDRBAC will give advice about whether a proposed action is permitted, but the enforcement is executed by the application.

We have also developed a sample application called train ticket subscription system with JSP. The system can be used by different train station to sell their tickets. In order to sell the linked tickets which can be used in different stations for customers, a booking clerk in one station often need to get the ticket from other stations. Since every station is a domain, it has its own role hierarchy and the security manager can configure the role mapping or role delegation relation between related stations. The foreign station’s booking clerk can be configured to have some junior roles that the local booking clerk has. So the booking clerk can enter other station’s system to get the ticket online if he can get the corresponding role.

3.2 Certificate distribution, discovery and verification

When a user logs in a domain, how to calculate the set of roles which a user can acquire with role mapping relations is the core algorithm in MDDRBAC. In accordance with the definitions of MDDRBAC, we have developed the following algorithm, where the notation role_set denotes the set of roles that the user can attain via role mapping relations.

\[ \text{role_set} \]
in: role_set ← ϕ;

02: d_set ← a set of Domains to which user is belonged;

03: while d_set ≠ ϕ do

04: d ← get a member from d_set; d_set ← d_set - {d};

05: r_set ← a set of roles to which the user is assigned in domain d;

06: while r_set ≠ ϕ do

07: r ← get a member from r_set; r_set ← r_set - {r};

08: MAP_set ← a set of tuples which satisfy the conditions Domain_R(r) = d and r_i ≤ r;

09: while MAP_set ≠ ϕ do

10: c ← get a member from MAP_set; MAP_set ← MAP_set - {c};

11: if valid2(c) then role_set ← ϕ role_set ∪ \{field r_i, in c\};

12: end;

13: end;

14: end;

15: role_set ← role_set ∪ \{all roles dominated by any element in role_set in domain d\};

The algorithm is distributed, which is started from the domain d when the user is logging in. Then the step from 05 to 13 is executed by each domain d, called by d and the results are returned to d. The certificate which represents the tuple in role mapping relation is distributed to domain Domain_R(r), so the data can be collected directly in d, on the step 08. This decreases the traffic among domains. The results from d include the set of roles attained by role mapping certificates and the corresponding certificate chains, which are used by the domain d to verify whether the user can actually acquire all the roles in the role set.

Domain d gets the certificate chains with the algorithm valid2 (the algorithm's description is omitted in this paper due to space limitation, which is designed in accordance with the function valid2 defined in the model). The certificate that represents the tuple in role delegation relation is distributed to domain d, so data can be collected recursively in valid2. In order to increase the discovery efficiency and decrease the algorithm's complexity, the certificate and its supporting certificate chain are also distributed together.

4 Conclusions

On the basis of RBAC96 and related research work, we have developed an access control model called distributed role-based access control model for multi-domain environments. The model relies on the role delegation and role mapping to achieve the interoperating between systems and a root organization is not required. The authorization revocation is managed with the field I in the REVOKE relation. Finally, some implementation issues are discussed. The distributed access control in multi-domain environments is a complex issue. When the number of roles is huge in every domain, an administrative model is required for MDDRBAC. In addition, if we want to delegate the administrative permission over a role to another role or an arbitrary user, how to solve the problems related to certificate revocation, discovery and verification are our future work.

REFERENCES

1. Sandhu R, Conyne E J, I. Feinstein H, et al. (1996) Role based access control models. IEEE Computer, 29(2): 38-47

2. Cohen E, Thomas R K, Winsborough W, et al. (2002) Models for coalition-based access control (CBAC). The 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002), Monterey, California.

3. Freudenthal E, Pesin T, Port L, et al. (2001) DRBAC: distributed role-based access control for dynamic coalition environments. Technical Report TR2001-819, New York University.

4. Joshi J, Ghafoor A, Aref W, et al. (2001) Digital government security infrastructure design challenges. IEEE Computer, 34(2): 66-72

5. Hong F, Wu M (2003) Role-based access control hierarchy model in distributed system. Journal of Huazhong Univ. of Sci. & Tech. (Nature Science Edition), 31(5): 9-10