Towards Secure Delegation with Chinese Wall Security Policy (CWSP)

Chunxiao Ye  
College of Computer Science, Chongqing University, Chongqing, China  
Email: yecx@cqu.edu.cn

Haibo Hu and Hong Xiang  
School of Software Engineering, Chongqing University, Chongqing, China  
Email: {oram, xianghong}@cqu.edu.cn

Abstract—Chinese Wall Security Policy (CWSP) is a widely applied access control policy in many fields, especially in commercial world. Delegation is one of the hot topics of access control technologies. Delegation with CWSP means delegation must satisfy not only delegation constrains but CWSP as well. There exist many delegation models, such as RBDM, RDM2000 and PBDM et al, but few focus on it. This paper proposed an approach of how to delegate permission with the restriction of CWSP. Although CWSP is part of delegation constraint, it does not mean that existing delegation models can be easy applied to this kind of delegation. In our approach, we first define two types of delegation constraints consisting of CWSP. Then we discussed different types of revocation and found that automatic revocation can make delegation safer than user revocation. Also, we found that there exists security vulnerability in multi-step delegation and gave some feasible solutions. Finally, this paper gave system implementation architecture and some examples to show how our approach works properly in a situation with CWSP.

Keywords— Role Based Access Control, Delegation, Chinese Wall Security Policy, CWSP

I. INTRODUCTION

Role-Based Access Control (RBAC) [1] has emerged as a new security technology which is an alternative to DAC and MAC. RBAC has recently received considerable attention. In RBAC, permissions are associated with roles, and users are assigned to appropriate roles to acquire permissions.

Delegation means that a user (delegator) can give her permissions to other person (delegatee). There are three types of situations in which delegation can take place: backup of roles, decentralization of authority and collaboration of work. Human-to human delegation has received considerable discussion recently [2] ~ [7].

RBDM [3] is the first delegation model based on role. RBDM deals with flat and hierarchical role, multi-step delegation. RBDM also deals with grant-dependent cascading and grant-independent cascading revocation. In RBDM, revocation can be manually performed by delegator and system administrator. Revocation also can be automatic triggered by delegation duration.

RDM2000 [4] is an extension of RBDM. A rule-based declarative language has been proposed to specify and enforce policies in delegation. Only user revocation is considered in RDM2000 and there are two types of user revocation: grant-dependent and grant-independent revocation. User revocation in RDM2000 has two options: non-cascading and cascading revocation. RDM2000 proposes three rules as enforcement of delegation revocation policies.

PBDM [5] is a flexible delegation model that supports multi-step delegation and revocation in role and permission level. PBDM includes three sub-models:PBDM0, PBDM1, and PBDM2. Revocation in PBDM is seen as the reverse process of delegation and a delegator can remove his/her own delegation at any time. PBDM supports multi-step revocation but revocation is discussed shortly.

RPRDM [6] only addresses repeated and partial delegation. There are four types of revocations in RPRDM. An original delegator, system administrator or a delegated delegator can revoke delegated permissions from delegatees. But there is only a very short discussion on automatic revocation.

We have proposed a new delegation model named Attribute-Based Delegation Model (ABDM) in [7]. Delegation constraint in our delegation model includes delegation prerequisite condition (CR) and delegation attribute expression (DAE). ABDM is a strict and more secured delegation model both in temporary and permanent delegation.

Chinese Wall Security Policy model (CWSP model) [8, 9] was first defined by Brewer and Nash in 1989. It has
been widely applied in commercial world where there exists conflict of interest among different companies. In this model, a company's data is called an object and belongs to one and only one dataset. Datasets are divided into Conflict of Interest (COI) classes. At first, a user has the right to choose which dataset she wants to access, but once she accessed a dataset of a COI, the system will isolate her from other datasets of the same COI. With CWSP, a user can access to only one dataset of a COI. That means, user can only access to datasets which belong to different COI. In figure 1, assume a user has accessed to Bank A and Airline B, then she cannot access to Bank B and Airline A in the future. Similarly, she cannot access to Oil Company A if she has accessed to Oil Company B. But she can access to Bank A and Oil Company A simultaneously, for they belong to different COI. CWSP can be seen as a mixture of DAC and MAC and is seem as a history-based access control policy.

In a commercial environment, delegation occurs frequently, especially when someone is on a business trip. That means a user (delegator) can transfer her permissions to other person (delegatee) and delegatee can perform these transferred permissions just like the delegator herself. Of course, delegation is constrained by system security policy. Above researches have focus on delegation constraints, but few have elaborated delegation with CWSP.

The rest of the paper is organized as follows. In section 2, we give the motivation of our work. In section 3, we propose how delegation works with CWSP. Section 4 gives the system implementation and some examples. Finally, we summarize the contributions of the paper and discuss some future works.

II. MOTIVATION

Let us consider the example shown in figure 2: Alex can access to data of Bank A, Oil Company B and Airline A, and Mike can access to data of Bank B and Oil Company A. Because Bank A and B, Oil Company A and B are in different Conflict of interest classes (COI) respectively, according to CWSP, there is a “wall” between Alex and Mike. In figure 2, the “wall” is denoted by dashed line and different individual company datasets accessed by different user are denoted by filled and unfilled boxes respectively.

Figure 3 illustrates examples of delegation among different persons. P_{Bank A}, P_{Oil Company B} and P_{Airline A} stand for access right to Bank A’s, Oil Company B’s and Airline A’s datasets respectively. According to CWSP, Alex cannot choose Mike and delegate permissions to him. So, Alex can choose another person, say Tom, and perform delegation. Assume Alex delegates P_{Bank A} to Tom, as shown in figure 3 (a). If Tom has accessed to

![Figure 2. User access to different datasets](image)

![Figure 3. Example of single-step delegation](image)
those datasets that Mike is able to access to, the delegation will be prohibited; otherwise the delegation will succeed. In this case, because Tom has no permissions before delegation, the delegation succeeds.

Let us consider another situation: assume Alex and Mike delegate some of their permissions to Tom separately, as shown in figure 3(b). According to CWSP, Tom can accept permissions within one COI, say Bank A and B, and once he choose to access to one bank’s dataset and a “wall” will prevent him from the other’s. So, delegator does not know whether the delegation is success after the delegation finished, for it depends on delegatee’s selection of permissions. For example, if Tom chooses to access to bank A’s data, then Alex’s delegation is success and Mike’s failed. This is a kind of “uncertain delegation” but is actually allowed according to CWSP. Thus we have two choices: to make delegation easy, we can prohibit it, or to make delegation flexible, we can allow it. The former is easy to implement and less safe and the latter is safer and makes the implementation somewhat difficult. In the former case, CWSP can be expressed by delegation constrains, such as prerequisite condition (CR) and in the later by activation constrains. If “uncertain delegation” is allowed, for the purpose of not violating CWSP, once a delegatee activates some permissions, the delegatee is prohibited from activating the corresponding conflict permissions for ever or those permissions will be revoked immediately.

There is also a choice for multi-step and single-step delegation. For example, Alex delegates some permissions to Tom and Tom is not able to activate those permissions because of CWSP, there are two solutions for this situation: one is that the permission can be revoked and delegated to other person by Alex; the other is that Tom is allowed to further the permission to other person. The former is a single-step delegation and is easy to implement but less flexible. The later is a multi-step delegation and is more useful and flexible. But the later has potential security vulnerability: let us consider figure 2(b), if Tom is able to perform P_{Bank A}, of course it is harmless for Tom to further it to other person. Because of little attentions have been paid to delegation with CWSP, base on existing delegation model, we developed some mechanisms to make delegation safe and flexible.

III. DELEGATION WITH CWSP

As we know, RBAC is a widely used access control model and is easy to implement. RBDM is the first delegation model based on RBAC but it does not support partial delegation. The unit of a delegation in RBDM is role. In PBDM, a delegator can delegate part of her permissions to delegatee. Obviously, PBDM is more useful than RBDM in many situations. Based on PBDM, we discuss how delegation is performed under the constraint of CWSP.

Delegator can perform delegation as follows: 1) delegator creates a temporary delegation role and assigns delegation permissions or roles to it; 2) delegator chooses a delegatee and assigns the temporary delegation role to her; 3) if the delegation satisfies all delegation constraints, it will succeed; otherwise it failed.

CWSP can be regarded as a kind of SOD and looks like dynamic SOD [10]. We also believe CWSP is an object-based separation of duty as mentioned in [11], but the difference is that we use role-based separation of duty to describe CWSP.

A. Concepts

Here we give some concepts definitions:

Definition 1

- T, U, AR, RR, TDR, P, O, OBJ and R are set of time, user, administrative role, regular role (created by system administrator), temporary delegation role (created by delegator), permission, operation, object and role respectively.
- \( t_0, t_c \in T \): \( t_0 \) is the initial time point and \( t_c \) is current time point of system.
- \( P \subseteq O \times OBJ \)
- \( R = RR \cup TDR \)
- \( PRA \subseteq R \times P \)
- \( URA \subseteq U \times RR \): a relation of user-regular role assignment.
- \( UDA \subseteq U \times TDR \): a relation of user-temporary delegation role assignment.
delegation can be defined as two different types: $p_1 \leftrightarrow p_2$.

For example, if $p_1 = (\text{read, Bank A's account})$ and $p_2 = (\text{read, Bank B's account})$, we say $p_1$ conflict with $p_2$ for $\exists \text{ other person, thus cause some security risk in multi-step delegation.}$

Other concepts’ definitions can be found in [5] and [7].

Definition 2 for two permissions $p_1 = (o_1, \text{obj}_1)$ and $p_2 = (o_2, \text{obj}_2)$, if $\text{obj}_1$ and $\text{obj}_2$ belong to the same COI, we call $p_1$ and $p_2$ are conflict with each other, denoted as $p_1 \leftrightarrow p_2$.

For example, if $p_1 = (\text{read, Bank A’s account})$ and $p_2 = (\text{read, Bank B’s account})$, we say $p_1$ conflict with $p_2$ for Bank A and Bank B’s datasets are in the same COI.

Definition 3 we call $r$ and $r'$ are conflict with each other, if $(\exists p, p' \in P \land (r, p) \in \text{PRA} \land (r', p') \in \text{PRA} \land p \leftrightarrow p'$, denoted as $r \leftrightarrow r'$. A role $r$’s conflict roles can be denoted as $\text{R}_r$, where $\forall r' \in \text{R}_r, r \leftrightarrow r'$.

Definition 3 means if two conflict permissions belongs to two different roles, then the roles are conflict with each other. For example, if $r_1 = (p_1)$ and $r_2 = (p_2, p_3)$, where $p_1 = (\text{read, Bank A’s account})$, $p_2 = (\text{read, Bank B’s account})$ and $p_3 = (\text{read, Oil A’s account})$, we call $r_1 \leftrightarrow r_2$ for $p_1 \leftrightarrow p_2$.

Definition 4

\[
\text{hasAct}(u, r) = \begin{cases} 
\text{true} & (u, r) \in \text{UAH} \land u \text{ has activated/ is activating}, \\
\text{false} & \text{otherwise}. 
\end{cases}
\]

Definition 4 defines the function to judge whether a role has been activated or being activated.

B. Delegation

Delegation constraint is vital for delegation security. In our work, delegation constraint consists of three parts: the first one is prerequisite role (CR), the second is SOD and the third is CWSP. Although CWSP looks like SOD, we separate it from SOD for there are two types of CWSPs: weak and strong CWSP.

- Weak CWSP can be seen as a normal delegation constraint. This means a user can keep two conflict roles at the same time but she can only activate one role and lost the right of activate the other one for ever. So, if a delegatee has roles that conflict with delegation role but she has not activated or is activating these roles, delegation will be allowed.

- Strong CWSP is stricter than weak CWSP. This means if delegatee already has roles that conflict with delegation role, delegation will not be allowed even those roles has not been activated before. Strong CWSP beyond the basic security restrictions of CWSP and is useful in a situation of multi-step delegation.

Accordingly, to obtain different delegation security level, delegation can be defined as two different types: delegation with weak CWSP and strong CWSP. Delegation relations are defined as follows:

Definition 5 a delegation with weak CWSP can be denoted as relation $\text{can-delegateW} \subseteq U \times R \times U$, where $U$ and $R$ are user and role set respectively. A delegation that a user $u$ delegates a role $r$ to $u'$ succeed means $(u, r, u') \in \text{can-delegateW} \Rightarrow (u, r) \in \text{UA} \land (\forall r_1 \in \text{R}_u, (u', r_1) \in \text{UAH} \Rightarrow \text{hasAct}(u', r_1) = \text{false})$.

A successful delegation means it satisfies all delegation constraints, such as CR, SOD and CWSP.

Here we only address CWSP in definition 5 and do not give the details of CR and SOD, which can found in [5]. For example, (Alex, rBank A, Annie) $\in \text{can-delegateW}$ means Alex can delegates rBank A to Annie only if Annie has not activated or is not activating rBank B. Alex can also delegate rBank C to Annie only if Annie has not activated rBank A and rBank B or is no activating rBank A or rBank B.

Definition 6 a delegation with strong CWSP can be denoted as relation $\text{can-delegateSC} \subseteq U \times R \times U$. A delegation that a user $u$ delegates a role $r$ to $u'$ succeed means $(u, r, u') \in \text{can-delegateSC} \Rightarrow (u, r) \in \text{UA} \land (\forall r_1 \in \text{R}_u, (u', r_1) \in \text{UAH})$.

For example, (Alex, rBank A, Annie) $\in \text{can-delegateSC}$ means Alex delegates rBank A to Annie only if Annie has not been assigned rBank B.

C. Revocation

We propose two types of revocation: user and automatic revocation. User revocation means revocation only can be performed by delegator or system administrator manually, while automatic revocation means revocation is triggered and performed by system automatically. User revocation can be found in most delegation model, but few support automatic revocation.

Definition 7

- Revocation performed by delegator is denoted as relation $\text{can-revokeD} \subseteq U \times R \times U$. $(u, r, u') \in \text{can-revokeD} \Rightarrow (u, r, u') \in \text{can-delegateD}$ and $(u, u') \in \text{can-delegateW}$.

- Revocation performed by system administrator is denoted as relation $\text{can-revokeS} \subseteq U \times \text{AR} \times U \times R \times U$. $(u, r, u', u'') \in \text{can-revokeS} \Rightarrow (u, r) \in \text{URA} \land ((u', r, u'') \in \text{can-delegateS} \lor (u', r, u'') \in \text{can-delegateW})$.

For example, Alex has delegated rBank A to Annie and he can revoke it from Annie whenever he wants to. A system administrator can also revoke the role from Annie if necessary.

As discussed in above, if a user who has been delegated two conflict roles by can-delegateW, once she choose to activate one role and the conflict role will not allowed to be activated. This is the fundamental principal of CWSP. There are two choices for the conflict role: to be revoked manually or automatically. We believe the latter is important because if the conflict role has not been revoked in time and delegatee may further it to other person, thus cause some security risk in multi-step delegation.
delegation, as mentioned in section 2. Automatic revocation can make delegation safer.

We define automatic revocation as:

**Definition 8** revocation triggered and performed by system automatically can be denoted as relation can-auto-revoke \( \subseteq U \times R \times U \), \((u, r, u') \in \text{can-auto-revoke} \iff (u, r, u') \in \text{can-delegateW} \land (\exists r_i \in R_r, (u', r_i) \in \text{UAH}) \land \text{hasAct}(u', r_i) = \text{true})\).

As we can see in definition 8, automatic revocation can only be triggered and performed in the situation of delegation with weak CWSP. The reason is that in a delegation with strong CWSP, a role cannot be delegated to a user who has activated or is activating the role’s conflict roles. For example, if \((\text{Alex}, r_{\text{Bank A}}, \text{Annie}) \in \text{can-delegateW}\) and \((\text{Alex}, r_{\text{Bank B}}, \text{Annie}) \in \text{can-delegateW}\), once Annie choose to activate \(r_{\text{Bank A}}\), \(r_{\text{Bank B}}\) will be revoked automatically by system. Or, if Annie has activated \(r_{\text{Bank C}}\) before, \(r_{\text{Bank A}}\) can also be automatic revoked.

D. Multi-step delegation

As we have discussed in section II, there exists security risks in multi-step delegation with weak CWSP. Here we discuss how to avoid these risks:

- If a delegatee has activated some delegation roles, then she is not able to delegate roles that conflict with activated roles to others. For example, assume Annie has two delegation roles \(r_{\text{Bank A}}\) and \(r_{\text{Bank B}}\). If Annie has activated \(r_{\text{Bank A}}\), then she cannot delegate \(r_{\text{Bank B}}\) to Roy. As we know, delegation is regarded as that the delegatee acts on behalf of the delegator itself. Although Annie delegates \(r_{\text{Bank B}}\) to Roy, it is still regarded as that it is Annie “herself” executes \(r_{\text{Bank B}}\).
- A delegatee cannot further different conflict roles to different person. For example, Annie cannot delegate \(r_{\text{Bank A}}\) to Roy and \(r_{\text{Bank B}}\) to Jacob separately, for she can get results from the executions of two conflict roles.
- If a delegatee furthers some delegation roles to others, then she is not able to activate those conflict roles of delegated roles for ever. For example, if Annie delegates \(r_{\text{Bank A}}\) to Roy, then Annie lost the power to activate \(r_{\text{Bank A}}\) for ever, because if Roy returns the results of execution of \(r_{\text{Bank A}}\) to Annie, this will violates CWSP.

Based on above discussion, we give some definitions for role activation and delegation relation. Because definition 5 has not taken multi-step delegation into consideration, here we give its modification:

**Definition 9** \(\text{can-delegateW} \subseteq U \times R \times U\), \((u, r, u') \in \text{can-delegateW} \Rightarrow ((u, r) \in \text{UA}) \land (\forall r_i \in R_r, (u', r_i) \in \text{UAH}) \land \text{hasAct}(u', r_i) = \text{false}) \land (\forall r_j \in R_r, (u, r_j) \in \text{UAH}) \land \text{hasAct}(u, r_j) = \text{false}) \land (\neg \exists u'', \neg \exists r' \in R_r, (u'', r') \in \text{UAH}) \land \text{own}(u))\).

In definition 9, \((\forall r_i \in R_r, (u', r_i) \in \text{UAH}) \land \text{hasAct}(u', r_i) = \text{false})\) is the constraint on delegatee and \((\forall r_j \in R_r, (u, r_j) \in \text{UAH}) \land \text{hasAct}(u, r_j) = \text{false})\) is the constraint on delegator. The later means delegator cannot activate roles

---

**Figure 5. Delegation and Revocation Components**

© 2011 ACADEMY PUBLISHER
that conflict with roles which she wants to further to other person. \(\neg \exists u'', r' \in \text{UAH} \land (u'', r') \in \text{can-delegateW or (u', r', u') \in \text{can-delegateS}}.

**Definition 10** a role activation must satisfy relation can-activate \(\subseteq \text{U} \times \text{R}, (u, r) \in \text{can-activate} \iff ((u, r) \in \text{UAH}) \land ((\neg \exists r' \in \text{R} \land (u, r') \in \text{UAH}) \land (\neg \exists u', (u', r', u') \in \text{can-delegateW or (u', r', u') \in \text{can-delegateS}}.

In definition 10, \((u, r) \in \text{UAH}) \land (\neg \exists r' \in \text{R} \land (u, r') \in \text{UAH})\) means a user cannot activate conflict roles simultaneously or separately, and \(\neg \exists u', (u', r', u') \in \text{can-delegateW or (u', r', u') \in \text{can-delegateS}}\) means the user cannot delegate conflict roles to other person.

IV. IMPLEMENTATION

A. System Architecture

Figure 5 shows the main components of delegation and revocation with CWSP. The solid arrow lines denote data flow between components and databases, while dashed arrow lines denote data flow among components.

Each component’s function is listed in table 1. We can divide these components into four different groups: temporary delegation role and delegatee generation, delegation constraints judgment, user and auto revocation and access control data and results (consists of PRA, User, Role and Permission, Prerequisite Condition (CR), SOD, CWSP and URA). The first group generates temporary delegation role and delegatee, the second group performs delegation by judging CR, SOD and CWSP, the third group deals with user and auto revocation and the fourth group stores basic access control data and delegation and revocation results.

B. Delegation and Revocation Steps

We give some sequence diagrams to show how delegation and revocation works, including single and multi-step delegation, user and automatic revocation. Involved components and their sequence in those processes are also given.

1. Single-step delegation

Figure 6 is the sequence diagram of single and multi-step delegation. We summary delegation steps as follows:

- 1) Delegator chooses which permissions or roles she will delegates to other person (Delegator \(\rightarrow\)CDP);
- 2) Delegator creates a temporary delegation role (CDP\(\rightarrow\)ACD, CDP\(\rightarrow\)ACD, CD\(\rightarrow\)ACD, CD\(\rightarrow\)ACD);
- 3) Delegator chooses a user and delegates a tdr to her (Delegator\(\rightarrow\)CD, CD\(\rightarrow\)ACD, CD\(\rightarrow\)ACD);
- 4) tdr and user’s information are forward to CR Judgement (GTDR\(\rightarrow\)CRJ, CD\(\rightarrow\)CRJ);
- 5) CRJ judges whether delegation satisfies CR (CRJ\(\rightarrow\)ACD, CRJ\(\rightarrow\)ACD). If not, delegation failed (Delegator \(\leftarrow\)CRJ; otherwise CRJ forwards delegation to SODJ (CRJ\(\rightarrow\)SODJ).
- 6) SODJ judges whether delegation satisfies SODJ (SODJ\(\rightarrow\)ACD, SODJ\(\rightarrow\)ACD). If not, delegation failed (Delegator \(\leftarrow\)SODJ); otherwise SODJ forwards delegation to CWSPJ (SODJ\(\rightarrow\)CWSPJ).
- 7) CWSPJ judges whether delegation satisfies CWSPJ (CWSPJ\(\rightarrow\)ACD, CWSPJ\(\rightarrow\)ACD). If not, delegation failed (Delegator \(\leftarrow\)CWSPJ); otherwise CWSPJ writes delegation results into ACD (CWSPJ\(\rightarrow\)ACD, Delegator \(\leftarrow\)CWSPJ).

2. Multi-step delegation

The main differences between multi-step and single-step delegation is that in multi-step delegation, EM will determines at the end of sequences whether environmental conditions allow this kind of delegation after the judgments of CRJ, SODJ and CWSPJ. Steps of multi-step delegation are similar to those of single-step delegation, except that there are two different steps. Here we only give the different steps as follows:

Steps 1) to 6) are the same as those of single-step delegation.

- 1) CWSPJ judges whether delegation satisfies CWSPJ (CWSPJ\(\rightarrow\)ACD, CWSPJ\(\rightarrow\)ACD).

<table>
<thead>
<tr>
<th>Component</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRA, User, Role and Permission, Prerequisite Condition (CR), SOD, CWSP and URA (ACD)</td>
<td>Store Role and Permission, PRA, CR, SOD, CWSP and URA data.</td>
</tr>
<tr>
<td>Choose Delegation Permission (CDP)</td>
<td>Delegator choose which permissions (including roles) she will delegates to other person.</td>
</tr>
<tr>
<td>Choose Delegatee (CD)</td>
<td>Delegator chooses a delegatee.</td>
</tr>
<tr>
<td>Generate TDR (GTDR)</td>
<td>Generates temporary delegation role.</td>
</tr>
<tr>
<td>CR Judgement (CRJ)</td>
<td>Judge whether a delegation satisfies CR.</td>
</tr>
<tr>
<td>SOD Judgement (SODJ)</td>
<td>Judge whether a delegation satisfies SOD.</td>
</tr>
<tr>
<td>CWSP Judgement (CWSPJ)</td>
<td>Judge whether a delegation satisfies weak or strong CWSP according to system configuration.</td>
</tr>
<tr>
<td>User Revocation (UR)</td>
<td>Delegator or system administrator performs user revocation.</td>
</tr>
<tr>
<td>Auto Revocation (AR)</td>
<td>System performs automatic revocation.</td>
</tr>
<tr>
<td>Event Monitor (EM)</td>
<td>Monitor occurrence of auto revocation or role activation.</td>
</tr>
</tbody>
</table>
delegation failed (Delegator ← CWSPJ); otherwise CWSPJ forwards delegation to EM (CWSPJ→EM).

2) EM determines whether the tdr’s conflict roles has been activated, being activated or been furthered to other person by delegate (EM→ACD, EM←ACD). If not, delegation failed (Delegator ← EM); otherwise EM writes delegation results into ACD (EM→ACD, Delegator ← EM).

Figure 7 is the sequence diagram of user ((a)) and auto revocation ((b)). We give two types of revocation steps as follows:

3. User revocation
1) Delegator decides to revoke tdr from delegatee, she calls UR (Delegator→UR);
2) UR determines whether revocation succeeds (UR→ACD, UR←ACD). If it is successful, UR

© 2011 ACADEMY PUBLISHER
saves results into ACD (UR→ACD, Delegator→UR); otherwise it returns revocation failed message to delegator (Delegator→UR).

4. Automatic revocation

1) EM detects an event that one of a delegatee’s delegation roles which are conflict with each other is now activating (EM→ACD, EM→ACD);
2) EM forwards this message including delegatee, conflict roles, et al. to AR (EM→AR);
3) AR determines whether revocation succeeds and saves results into ACD (AR→AR, AR→ACD) or returns revocation failed message to delegator accordingly (AR→AR).

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we firstly give an introduction and overview of Chinese Wall Security Policy and delegation model and find that few works are focus on delegation with CWSP. Then, we show the motivation of our work and find that delegation with CWSP is different from regular delegation. Current delegation models cannot deal with these issues efficiently. So, we propose some mechanisms to meet these delegation requirements, such as single and multi-step delegation, user and automatic revocation and activation rule. Thus ensure security of delegation with CWSP. Finally, we give the system implementation architecture and some examples to show how our mechanisms work. In this paper, permissions of a temporary delegation role are only consisting of the whole or part of a role’s permissions. It is difficult to delegates permissions coming from different roles, especially in multi-step delegation, for weak and strong CWSP will be more complicated. In future work we would like to deal with this issue.

ACKNOWLEDGMENT

This research is supported by National Natural Science Foundation of China (Project No. 60803027), the Natural Science Foundation of Chongqing, China (Project No.CSTC, 2008BB2320) and the National High Technology Research and Development Program of China (Project No. 2007AA01Z445).

We acknowledge the generous help of Dr. Jason Crampton, who working in the Information Security Group (ISG), Royal Holloway, University of London.

REFERENCES


Chunxiao Ye was born in Chongqing, China, on April 12th, 1973. He received his BEng in computer software from Sichuan University, China in 1995 and MEng in computer architecture from Chongqing University, China in 2002. In 2005, he received his PhD in computer science from Chongqing University, China.

He is now an associate professor at the College of Computer Science, Chongqing University, China. He has published more than 50 journal papers and conference articles in access control, software engineering and database. His research interests include access control, grid, software engineering and database.

Dr. Ye is now a senior member of China Computer Society and a member of The Institution of Engineering and Technology (IET).

Haibo Hu obtained BSc degree and MEng from University of Chongqing in 1999 and 2004 respectively. He is lecturer at the School of Software Engineering, University of Chongqing, China, and Ph.D. student of Computer Science. His current research interests include software quality, access control and semantic web technology.

Hong Xiang is professor at the School of Software Engineering, University of Chongqing, China, and consultant of security for e-government in Chongqing. He was previously researcher for Nortel Networks in Canada. He obtained Ph.D. degree from the University of Alberta, Edmonton, Canada, in 1998. His current research interests include security engineering for software system, security verification and evaluation.