Formal Description of Trust-based Access control

Ma Xiaoning

School of Computer Science and Technology, Civil Aviation University of China, Tianjin, China

xnma@cauc.edu.cn

Abstract

Different from traditional access control technologies, such as discretionary access control, mandatory access control, role-based access control, trust-based access control can solve the problem of uncertainty, risk and vulnerability coming from authorization. In this paper, strict definition and formal description of trust-based access control is defined.

© 2012 Published by Elsevier B.V. Selection and/or peer review under responsibility of ICMPBE International Committee.

Keywords: Access Control, Trust, Formalization

1. Introduce

Compared with traditional access control technologies, just as, discretionary access control, compulsory access control and access control base on role, it is different that access control base on trust can solve uncertainty, risk and vulnerability problem for provide authorized party. It has been strictly defined and formal described in this paper for access control base on trust. First of all, the important concept has been strictly defined, and then a formal description has been given by authorization policy mechanism, finally, the algorithm of access control base on trust gives a formal description.

2. Definition

Def 1 Resource: a useful entity. It can be owned by one, but needed by others. It is expressed by signal \( \text{Res} \), \( ResourceSet = \{ \text{Res}_1, \text{Res}_2, ..., \text{Res}_n \} \), in this set, \( \text{Res}_i \) is one kind of resource.

Def 2 Action: the behavior for resource. For example, using and query. It is expressed by signal
Act. ActionSet = \{Act_1, Act_2, ..., Act_n\}, in this Set Act_i is an action.

Def 3 Permission: it means qualification of performing one kind of resource. It is expressed by two elements group Per ≜ <Res, Act> PermissionSet = \{Per_1, Per_2, ..., Per_n\}, in this set, Per_i is a permission, and \(\text{PermissionSet} \leq \text{ResourceSet} \times \text{ActionSet}\).

Def 4 Provider: provider owned permission and provides option for resource. It is expressed by signal Pr. ProviderSet = \{Pr_1, Pr_2, ..., Pr_n\}, in this set, Pr_i means a provider.

Def 5 Requestor: requestor requests for permission for performing one kind of resource. It is expressed by signal Re. RequestorSet = \{Re_1, Re_2, ..., Re_n\}, in this set, Re_i means a requestor.

Def 6 TrustAttribute: the Provider’s assessment of results for the Requestor. It is expressed by signal TrA, and it is usually expressed by Trust-Value that has been evaluated.

Def 7 TrustCondition: it is a condition that must be satisfied by the TrustAttribute of Requestor. If the TrustAttribute of Requestor can satisfy TrustCondition, Provider can trust Requestor; or else untrusted. Usually, TrustCondition is a trust threshold, when the trusting degree greater than or equal to trust threshold, the Requestor is worthy of confidence for Provider; and when the trust degree less than the confidence threshold, Provider does not trust the Requestor. It is expressed by signal TrC. TrustConditionSet = \{TrC_1, TrC_2, ..., TrC_n\}, in this set, TrC_i means a TrustCondition.

Def 8 Req is a two-elements group: Req ≜ <Re, <Res, Act>>, and Re is the Requestor. Re \in RequestorSet, <Res, Act> means the permission requested by requestor. Res is the resource requested by Re, Res \in ResourceSet. Act is an action for Res, Act \in ActionSet.

Def 9 Pol: it is made by provider for managing permission. Pol is expressed by two-elements group Pol ≜ <Res, Act>, TrC > and <Res, Act> is a permission for Pol. TrC is TrustCondition for permission, TrC \in TrustConditionSet.

PolicySet = \{Pol_1, Pol_2, ..., Pol_n\}, and Pol_i means one Pol.

Provider may own many kinds of resource forming ResourceSet, and every resource may have many kinds of operations that mean bigger PermissionSet. Many kinds of Pol are needed by provider for managing permission.

3. The authorization decision

Authorization decision mechanisms ensure that users can get their own privileges, and won’t get
privileges that should not be enjoyed. Following the Web service, the authorization decision method of access control base on trust is formally described.

Assertion 1  PolicyAccord: if $\langle Req, <Res, Act, >>$ is a request, and $Pol = \langle<Res, Act, >, TrC \rangle$ is the authorization policy.


That is to say, in the authorization strategy sets, if there is an authorization policy meeting the following two conditions, then the request is consistent with the policy.


2. $Pol.Act = Req.Act$

The definition of PolicyAccord: in all authorization policies, there is a policy that can satisfy current access request.

$$PolicyAccord(Req, Pol) = \{0, 1\}$$

Assertion 2  NotPolicyAccord: if $\langle Req, <Res, Act, >>$ is a request, and $Pol = \langle<Res, Act, >, TrC \rangle$ is the authorization policy.


$$= (\forall Pol)((Pol.Res \neq Req.Res) \lor (Pol.Act \neq Req.Act))$$

That is to say, if each authorization policy meets any one of the following two conditions, then the request and the strategy does not conform to each other.

1. $Pol.Res \neq Req.Res$

2. $Pol.Act \neq Req.Act$

The definition of NotPolicyAccord: in all authorization policies, there is no one that can satisfy current access request.
\[ PolicyAccord(Req, Pol) = \{0, 1\} \]
\[ \text{NotPolicyAccord}(Req, Pol) = \{0, 1\} \quad \text{and} \quad \neg \text{NotPolicyAccord}(Req, Pol). \]

Assertion3  
**Authorization:** if \( Req = \langle Re, < Res, Act, >> \) is a request, and \( Pol = \langle< Res, Act >, TrC > \) is the authorization policy.

\[ Authorization(Req, Pol) = (\exists Pol)(PolicyAccord(Req, Pol) \land Compliance(Req.Re.Tra, Pol.TrC)) \]

That is to say, in the authorization policy set, if there is a authorization policy can meet the following two conditions, then the request is authorized.

1. \( \exists Pol(\text{PolicyAccord}(Req, Pol)) \);
2. \( \text{Compliance}(Req.Re.Tra, Pol.TrC) \)

The definition of **Authorization:** in all authorization policies, there is one can satisfy current access request. The trustAttribute of requestor which is sending its request can satisfy the conditions of the policy, then the request is authorized.

\[ Authorization(Req, Pol) = \{0, 1\} \]

Assertion4  
**RejectAuthorization:** if \( Req = \langle Re, < Res, Act > >> \) is a request, and \( Pol = \langle< Res, Act >, TrC > \) is the authorization policy.

\[ RejectAuthorization(Req, Pol) = (\exists Pol)(PolicyAccord(Req, Pol) \land \neg Compliance(Req.Re.Tra, Pol.TrC)) \]

That is to say, in the authorization policy set, if there is a policy can meet the following two conditions, the request will be unauthorized.

1. \( \exists Pol(\text{PolicyAccord}(Req, Pol)) \);
2. \( \neg \text{Compliance}(Req.Re.Tra, Pol.TrC) \)
The definition of RejectAuthorization: in all authorization policies, there is one can meet current access request. The trustAttribute of requestor which is sending its request can not satisfy any of the conditions of the policy, the request is unauthorized.

\[
\text{RejectAuthorization}(\text{Req}, \text{Pol}) = \{0,1\} \quad \text{and} \quad (\text{PolicyAccord}(\text{Pol}, \text{Req}) = 1) \rightarrow \left(\text{Authorization}(\text{Req}, \text{Pol}) = \neg \text{RejectAuthorization}(\text{Req}, \text{Pol})\right)
\]

4. The algorithms of access control base on trust

The algorithm is just as following:

Input:
1. \(\text{PolicySet} = \{\text{Pol}_1, \text{Pol}_2, \ldots, \text{Pol}_n\}\),

   And \(\text{Pol}_i = \langle \text{Res}, \text{Act} >, \text{TrC} >\);

2. \(\text{Req} = \langle \text{Re}, \langle \text{Res}, \text{Act} \rangle >\),

Output:
\(\text{Authorization}(\text{Req}, \text{Pol})\) or \(\text{RejectAuthorization}(\text{Req}, \text{Pol})\)

1. if \(\text{PolicySet} = \emptyset\) then
2. goto \(\text{NotPolicyAccord}(\text{Req}, \text{Pol})\)
3. else \(i=1\);
4. while \(\{i<n, \text{and,} \ \text{NotPolicyAccord}(\text{Req}, \text{Pol}_i)\}\)
5. \(\{i=i+1;\}\)
6. if \(\text{NotPolicyAccord}(\text{Req}, \text{Pol}_i)\) then
7. goto \(\text{NotPolicyAccord}(\text{Req}, \text{Pol})\)
8. else
9. if \(\text{Compliance}(\text{Req.Re.TrA}, \text{Pol.TrC}) = 1\) then
   \(\text{Authorization}(\text{Req}, \text{Pol})\)
10. else
11. \( \text{RejectAuthorization}(\text{Req}, \text{Pol}) \) \\
12. \( \text{NotPolicyAccord}(\text{Req}, \text{Pol}) \) \\
return \( \text{RejectAuthorization}(\text{Req}, \text{Pol}) \)

5. Conclusion

The strict definition of access control base on trust is given in this paper, and also the formal description for authorization policy mechanism and algorithms. It will be very important for the research of the access control in the future.

Reference:


