PRIVILEGE DELEGATION AND REVOCATION FOR DISTRIBUTED PERVASIVE COMPUTING ENVIRONMENTS

Anh Quan Pham[1]

[1]Faculty of Engineering,
University of Technology Sydney

ABSTRACT: This paper looks at the problem of privilege delegation and revocation in distributed pervasive computing environments. In this paper, the relationship between the privilege delegation and revocation will be discussed as well as how to deploy them on distributed systems. There will be a signature based schemes to achieve privilege delegation with different trust assumptions. The scheme operates by using either public key or secret key based cryptography. Then the issues of revocation of delegations and proposal some schemes to achieve this. Finally, the transitivity of trust in related with privilege delegation and implicated trust assumptions will be briefly mentioned in the conclusion.

INTRODUCTION

In distributed pervasive computing environments which contain multi-users and multi-systems, any entity may be constrained on how it acts upon other entity. In general, one entity or user has a set of privileges for some services that it can access. For traditional systems, a static set of privileges for each user would be adequate. However, with the distributed systems, it is insufficient especially in some circumstances in which it is difficult to anticipate in advance what privileges are needed.

This raises the demand of a more flexible systems of access control in which privileges can be dynamic granted and withdraw. The basic philosophy is that one entity performs some actions on behalf of another entity by asking for authorisation. The main issue is to build a mechanism in which delegation and revocation is verifiable. In such systems, if one entity claims to be delegated some privileges from other entity, the service provider (the end point) would be able to check that it has indeed been delegated.

At the moment, many projects are looking on these issues in terms of architecture design [JKF01]. However, not many of them really propose the generic protocols for the delegation and revocation chains. This paper offers a general formulisation for such protocols.

DISTRIBUTED PERVASIVE COMPUTING SCENARIO

Consider the following scenario. Peter is an employee of the company. He is equipped with a kind of identification device such as Smartcard (or possibly a PDA). The identification device is WiFi-enable. Every time, he comes to his company, he will be automatically authenticated and granted the access to some services (room, printers, etc.). Come along Peter's supervisor, John. He wants Peter to come to the Protection Room which is only accessible for Supervisors, to collect some materials for her.

Of course, Peter could not get the access to this room. Then Peter must request the correct permission from John by sending a request message to John via the company network. Basing on the company security policy, John can delegate access permission to anyone he trusts. So, he delegates his access privilege to Peter for a particular period of time.

John will send a signed and encrypted delegation to Peter's identification device. When Peter tries to re-access the Protection Room, his device will send his certificate and the delegation to the authentication server. Because John is trusted, authenticated and be able to delegate access rights and because the delegation complies with the security policy, Peter now has access to the Protection Room. When the delegation expires, Peter will be informed and he must ask John for another delegation.

This scenario can, to some extent, demonstrate the importance of privilege delegation and trust relationship over the traditional security mechanisms in a distributed pervasive computing

environment. The scheme allows Peter, a non-authenticated user or even an outside user, to access some services without changing his identification and role in the system or opening up and exposing the system in an insecure way.

SECURITY CHALLENGES AND REQUIREMENTS

For each entity involved in the delegation process, there are certain requirements which are required to be met. The initiator or delegator may want to give a part of its set of privileges. Moreover, it may only want to give the privileges a particular period of time. Most importantly, the initiator may want to identify and record each of its delegation so that the initiator can revoke the privilege at some stage in future.

An entity must be able to prove its delegation to the service provider (the end point). It also has to know what privileges are needed and how to get them. It is also important to know the scope of the privileges; what they are for and how long they may last. It is also necessary for the entity, which receives delegation, to set further delegations.

The service provider (the end point), when receiving a request from the entity which claim to have some delegated privileges, must be able to read, authenticate the entity and verify the delegated message. The service provider then can make the decision basing on the entity's privileges and its own access control information. [VAB90]

Due to the scope of this paper, only privilege delegation and revocation between entities are mentioned.

PRIVILEGE DELEGATION

This section will describe the generic protocol for the delegation scheme.

Definition of Signature: Let $<M>_A$ be message M signed by A. Any entity received $<M>_A$ would be able to verify that the message actually come from A at some time in the past and the entity must be able to read M. The additional condition is the algorithm to produce $<M>_A$ must be "strong". It means if $<M>_A$ is captured by some one else, they can not read, edit or produce $<M'>_A$ which is the same as $<M>_A$.

Here is the expression for a delegation from A to B (A delegates its privileges to B):

A ↙ B: $<A,T_A>$, where $\mathbf{T_A = <B,A,Pr_a,r_a,t_a>_A}$

Pr_a: privilege of A, which A delegate to B

r_a: a unique random identifier, used for replay detection

The message contain A's identity, for identification purpose, followed by the delegation token T_A, which is signed by A. The token includes the identity of A, the initiator, the receiver and the privileges. This is to guarantee that the initiator authorises the receiver to have certain privileges of the initiator. In the token, r_a is the unique random identifier which is never used twice by A. The token has t_A to define a finite lifetime for the token, so the message and the delegation. Thus, the above expression can be said "A give privilege Pr_a to B for a limited period of time t_a".

Now if B want to use service S by using privilege Pr_a:

B ↙ S: $<B,<A,T_A>>$

S can understand the message. By the definition of signature S know that A produce the message and S can also check that Pr_a is actually from A.

However, this is not enough for B to use the service. S only lets B access if B can prove that at the time it requests, it is who it claims to be. So B needs to send S a message to claim the delegated privileges

B ↙ S: $<B,T_B>,<A,T_A>$ where T_A as above; $\mathbf{T_B = <S,B,r_b,t_b>_B}$

S then reads the message and if everything is matched, S grants the access for B

Now, if B wants to make some future delegates to C, the message would be:

B ↙ C: <B,T_B>,<A,T_A> where T_A as above; **T_B = <C,B,r_b,t_b>$_B$**

T_B actually says that B passes the privilege Pr_a to C for a duration t_b. T_A is needed in the token to show that B gets the original privileges from A.

Now, if C wants to use S, C do the same as B did:

C ↙ S: C,<B,T_B>,<A,T_A> where T_A and T_B as above

C ↙ S: <C,T_C>,<B,T_B>,<A,T_A> where T_A and T_B as above and **T_c = <S,C,r_c,t_c>$_C$**

Again S can be sure that both B and A are involved in the delegation chain. Please note that

- S must check A's privileges to pass on Pr_a.
- S must check B's privilege to pass on Pr_b.
- Time duration on delegation from B to C relies on the duration of delegation from A to B. So t_b should not greater than t_a.

Now we consider the scheme when implementing using public key system

Public Key Scheme:

As in this scheme, each entity has a pair of public key and private key. For example, entity A has public key PbKA and private key PrKA. In this scheme, everything should be the same, except the signing process will be performed by using the receiver public key.

The following assumptions are necessary:

- public key is really public (it should be kept on a public directory)
- with a public key can not produce a private key
- information encrypted by private key can only decrypted by the public key
- private key is kept securely and integrity uncompromised.

The messages should be:

A ↙ B: <A,T_A>, where **T_A = <B,A,Pr_a,r_a,t_a>$_{PrKA}$**

B ↙ C: <B,T_B>,<A,T_A> where T_A as above; **T_B = <C,B,r_b,t_b>$_{PrKB}$**

C ↙ S: <C,T_C>,<B,T_B>,<A,T_A> where T_A and T_B as above and **T_c = <S,C,r_c,t_c>$_{PrKC}$**

The sending entity identifier is sent in plain text allow the receiver to know which public key to use to decrypt the message. Moreover, from the message, the end point can get the whole chain of delegation.

The delegation can also deploy with Secret Key system. However, there is a big disadvantage because the receiving entity can not read the message as the key is only known by the initiator and the authentication server. So the content of the delegation token should be sent in plain text, which is less secure. It also increases traffic and redundancy of the system by the larger messages and the communication with the authentication server to verify the exchange messages.

A ↙ B: <B,A,Pr_a,r_a,t_a ,<B,A,Pr_a,r_a,t_a>$_{KA}$> where KA is the Secret Key of A

REVOCATION OF DELEGATION

On the previous part, the privilege delegation is discussed. However, very often in the real world, one entity may want to revoke the delegated privileges. In the scenario above, when Peter is on the way to the Protection Room, his boss, John, decides that he does not need the materials any more. So he gives Peter a call to cancel and, in parallel, revoke his delegation from Peter.

In this section, the revocation of delegation, in which an initiator (delegator) can revoke an authorisation, will be discussed. Also, the conditions for revocation and some assumptions, which they must be meet before the revocation operation is processed, will be mentioned.

Depending on the mechanism employed, such as secret key or public key systems, and wether we have access control lists at the service provider (the end point), then we have different possible solutions. These solutions also depend on trustworthiness and the key management protocols.

Basing on different assumption, we propose different solutions. In the solutions, we suppose that messages are differentiable. That is, if an initiator issues more than one delegations, then it can selectively withdraw the one or a set of privileges and not have to revoke all delegations. By providing a nonce, or "random identifier", in the message, the initiator can revoke privileges which have this specific identifier. This nonce may also be used for replay detection.

Solution 1:

The initiator sends a revoke message to the delegator that it initially sends the authorisation to. This revocation message is then passed on to all entity that received that authorisation. For example, in the scenario above, John will send a message to Peter and destroy the delegation.

This solution is good if the initiator trust each entity in the delegation chain and each entity will pass on the revocation message. This solution also required each entity in the delegation chain maintain a list of all other entities that have received the delegation.

However, there are some problems with this solution. What happens if one of the entities in the delegation chain is "offline"? Obviously, the revocation process will be stopped.

Solution 2:

The initiator sends a revoke message to the service provider that its privileges are needed for. This revocation message will tell the service provider the explicit instructions to ignore a specific delegated message.

This is fast, efficient and maintains a high level of security for the system. However, it only works when the initiator knows the service provider (the very end point). For small system, it would be practical to know the end point, but for a large and complex system, the feasibility is low. Moreover, it also required the service provider to have some kind of access control revocation mechanism, not only just validating the revocation message.

Solution 3: The philosophy of this solution is to make the service provider (the end point) unable to verify the delegation message. This can be done by changing the initiator's key for verification process.

This solution is good and efficient as the initiator can easily change the situation. However, the key management mechanism is required and the effectiveness of the system will depend on the key management mechanism. In the PKI, the public key is stored at a directory. Changing the key may make the revocation message can not filter through the end point (service provider). This solution works well with the Secret Key mechanism but if the authentication server changes the key, then same problem happens.

Solution 4:

This solution has the same philosophy with the solution above. But in this case, the key for decryption process is changed. This makes the service provider unable to read the delegation message.

This solution is quite similar with the solution above. So it has similar strength and weakness.

CONCLUSION AND DISCUSSION

The development of the distributed pervasive computing environment rise many challenges in terms of system security. There is an obvious demand for delegation and revocation to make the system become more flexible and maintainable. The nature of delegation and revocation is to be able to securely perform some actions that one entity acts on behalf of another entity. In the system, the access privileges are not static but dynamic and changed based on delegations and revocations. Users are assigned some privilege based on their individual credentials and/or their generic roles.

Users can use their privileges to access some service or delegate to some entities they trusts. The delegated privileges then in turn can be revoked. The security policy is also involved in all delegation and revocation process.

When the initiator passes privileges to an entity, it is saying “I give this entity some certain privileges that I have, it is up to this entity to use the privileges”. This means we take the point of view that passing the privileges assumes responsibilities in the receiver. The initiator should not concern about the way the receiver use the delegated privileges. The received entity is considered trusted in the sense that it does not abuse the trust.

Both Public Key Scheme and Secret Key Scheme can be used to deploy delegation. However, PKI provides better solution as there is no need for a trusted server. Thus there is no need for exchanged message with the authentication server for each delegation and so the traffic intensity is decreased.

Finally, one of the important aspects that would play the vital role in delegation and revocation is trust relationship management. Just image, if another staff, Ben, comes to the Protection Room and he also does not have access privilege. What he can do? He can ask his friend, Peter, and Peter could say “Come to ask John, he has necessary privilege to access”. Then, Ben comes to John. However, there is a question arise which is “How can and How much John can trust Ben so that he can delegate the privilege to Ben?” It forms a transitivity of trust between John, Peter and Ben. Obviously, just by basing on the introduction from Peter is not enough for the transitivity of trust. Thus, the delegation and revocation can not work alone but a trust relationship management mechanism is necessary. This would be the next step which is worth to think about to approach a completed architecture for a privilege delegation and revocation scheme in distributed pervasive environments.

ACKNOWLEDGEMENTS

I would like to thank the following person for their materials, comments, suggestions and the supportive attitude.

Professor Hoang, Doan is the professor at the Faculty of Information Technology, University of Technology, Sydney. He is leading the Advanced Research in Networking (ARN) Group.

Professor Varadharajan, Vijay is the Microsoft Chair in Computing at Department of Computing, Macquarie University, Sydney. He is also the Director of the Information and Network System Security (INSS) Research Group.

Mr. Kumar, Ashok is a research assistant of Professor Varadharajan, Vijay at Information and Network System Security (INSS) Research Group at Department of Computing, Macquarie University, Sydney.

REFERENCES

[ABKL93] M. Abadi, M. Burrows, C. Kaufman and B. Lampson. Authentication and Delegation with Smartcards. *Science of Computer Programming*, 21(2), pp. 91-113, 1993.

[ABLW92] M. Abadi, M. Burrows, E. Wobber and B. Lampson. Authentication in Distributed Systems: Theory and Practice. *ACM Trans. Computer Systems*, 10(4), pp. 265-310, 1992.

[BFL96] M. Blaze, J. Feigenbaum and J. Lacy. Decentralised Trust Management, *Proceeding of the IEEE Conference on Security and Privacy*, 1996.

[CLB02] M. Clifford, C. Levine and M. Bishop. The Solar Trust Model: Authentication without Limitation, [Online] Available at: http://www.acsac.org/1998/abstracts/fri-a-1030-clifford.pdf, 2002.

[CY02] R. Chen and W. Yeager. Palano – A Distributed Trust Model for Peer to Peer Network. *Sun Microsystems Press*, [Online] Available at: http://www.jxta.org/docs/trust.pdf, 2002.

[JKF01] A. Joshi, L. Kagal and T. Finin. Trust-based Security in Pervasive Computing Environments. *IEEE Computer*, 12, pp.154-157, 2001.

[JKFUP04] A. Joshi, L. Kagal, T. Finin, J. Udercoffer and F. Perich. A Security Architecture Based on Trust Management for Pervasive Computing Systems. *IEEE Pervasive Computing*, 2004.

[VAB90] V. Varadharajan, P. Allen and S. Black. An analysis of the Proxy Problem in Distributed Systems. *Proceeding of the IEEE Symposium on Security and Privacy*, 1990.

[VA90] V. Varadharajan and P. Allen. Joint action based Authorisation Schemes. *ACM Operating System Review Journal*, pp. 32-45, 1990.

[Var01] V. Varadharajan, Distributed Authorisation: Principle and Practice, *Coding Theory and Cryptography*, Singapore University Press, pp. 385-433, 2001.

[VCP98] V. Varadharajan, C. Crall and J. Pato. Authorisation for Enterprise Wide Distributed Systems. *Proceeding of the IEEE Computer Security Application Conference*, ACSAC'98, 1998.

[WL92] T.Y.C. Woo and S.S. Lam. Authorisation in Distributed Systems: A Formal Approach. *Proceeding of the IEEE Symposium on Security and Privacy*, pp. 33-50, 1992.

[SCF96] R. Sandhu, E.J. Coyne and H.L. Feinstein. Role based access control model. *IEEE Computer*, 29(2), pp. 202-213, 1996.

[YLYLZ04] H. Yang, H. Luo, F. Ye, S.W. Lu and L. Zhang. Security in Mobile Ad hoc Networks: Challenges and Solutions. *IEEE Wireless Communication*, pp.38-47, 2004.