Access Policy Transfer Between Active Nodes Using Identities

Youngsoo Kim, Jongwook Han, Dongil Seo, Seungwon Sohn*

* Network Security Department, Information Security Technology Division, ETRI, Daejeon, Korea
(Tel : +82-42-860-5856; E-mail: blitzkrieg@etri.re.kr)

Abstract: Active networks allow active node's functionality to be extended dynamically through the use of active extensions. This flexible architecture facilitates the deployment of new network protocols and services. However, the active nature of a network also raises serious safety and security concerns. These concerns must be addressed before active networks can be deployed. In this paper we look at how we can control active extension's access to different active nodes. Specifically, the authentication between active nodes is very important in this case. We use unique identity each node has for transferring access policies between active nodes. In this paper, we suggest a new method of transferring access policies performing authentications using identities between active nodes.

Keywords: Active networks, Active Extension, Identity, Cryptography

1. INTRODUCTION

Active networks allow active node(router or switch)'s functionality to be extended dynamically through the use of active extensions. Namely, we will call software modules that extend active node functionality "extensions". This flexible architecture facilitates the deployment of new network protocols and services. However, the active nature of a network also raises serious safety and security concerns [1][2]. One particular security question is how we can limit what resources and data active extensions can access on the active node. There exist some solutions like using access control policy. On the other hand, it can be the security issue that how we can control active extension's access to different active nodes. Specifically, the authentication between active nodes is very important in this case. This paper is dealing with the latter one. We suggest a new way of transferring access policies performing authentications using identities, not using the Kerberos[3] mechanism traditionally.

The rest of the paper is organized as follows. In Section II, we deal with security issues for active node. In Section III, we present a traditional way of solving this problem, Kerberos mechanism for transferring access policies. We suggest a new method for transferring access policies improving conventional one and conclude in Section IV.

2. SECURITY ISSUES FOR ACTIVE NODES

While an active node architecture increases flexibility, it also raises safety and security concerns. Safety centers around the question of how we can safely execute extensions that may be faulty, e.g., they could cause the node or node components to fail. These problems are addressed by isolating the extension code from the rest of system, either using runtime mechanisms (e.g., Java sand boxing, virtual memory) or compile time mechanisms (e.g., Proof carrying code [4]). We divide security issues into two categories in this paper. One is the internal security issue and the other is the external security issue.

The internal security issue is focusing on active node architecture itself. We want to prevent active extensions from disrupting the network service received by other users, for example by using their resources or by reading or writing their data. This problem is similar to the problem addressed by a traditional operating system, except that nodes have a very different task. Their primary responsibility is forwarding and processing packets, not general-purpose data processing, storage management, or user interface support. This means that node operating systems face a different set of security concerns. In order to perform their tasks (e.g., implementing QoS, selecting nodes, or encrypting data), node extensions must be able to control critical node resources such as link bandwidth and access critical data structures such as the routing table. Extensions must also be able to manipulate data traffic, e.g., dropping packets or modifying packet contents. It is easy to see that without proper security mechanisms, extensions can use these operations to harm other users. For example, malicious or faulty extensions can steal bandwidth by making invalid reservations, can corrupt the routing table, or can manipulate data traffic that belongs to other users.

On the other hand, the external security issue is focusing on communication between active nodes. For example, active extensions can implement specialized routing protocols or a customized network-monitoring infrastructure. However, given their ability to access critical node resources and affect data flows, extensions can pose serious threats to the active network and other users. The threats range from risks local to one node to risks that can span the whole network. For example, without proper access control, an extension can steal bandwidth for its flows by either increasing its bandwidth reservation parameters, or by associating its flows with resource nodes of other users[5]. Alternatively, a malicious extension can reroute random flows to disrupt other users ' traffic, or issue a Denial-of-Service (DoS) attack by tunneling flows to a victim server or network segment. So, we need some policies to control authority of access. Furthermore, to exchange those policies between active nodes, they should have authentication step between them.

3. CONVENTIONAL METHOD FOR TRANSFERRING ACCESS POLICIES

Figure 1 depicts our system architecture. The user is the entity that injects extension code into nodes; it is also known as the Extension Initiator (EI). We designed a protocol for the User, Policy Manager (PM) and Active Node (R) to securely exchange security policy information and extension code.

The communications between the users, PM and active nodes must be secure to ensure the security of the system. We designed a secure communication protocol based on Kerberos that allows (1) users to authenticate with nodes through the use of the Policy Manager; (2) secure transportation of extension code and corresponding access policy information to active nodes.

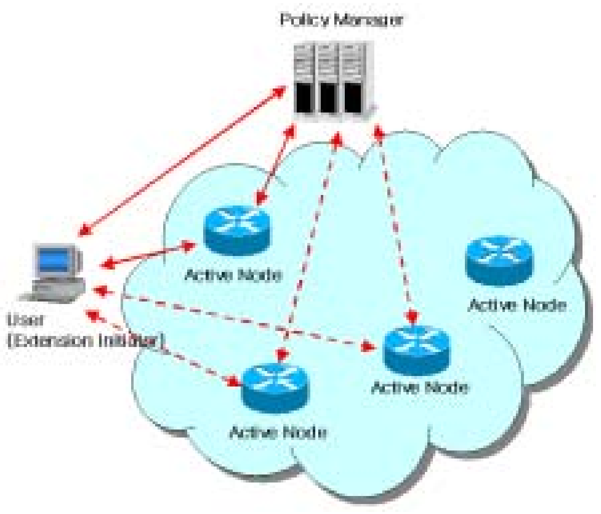


Fig. 1 System Architecture

We assume there is one Policy Manager (PM) operated by a trusted authority within a domain. It has full control and knowledge of the active nodes and links of this domain. A network manager will typically be responsible for defining the policy for extensions in the network using a policy definition language. Users within the domain must register with the PM before trying to install extensions on the nodes of this domain. The primary task of the PM is policy enforcement, i.e., assigning access permissions to extensions based on the policy specified by the network manager. Besides acting as a policy server, the PM also serves as the authentication center and session key distribution center in the security protocol we present below. The protocol allows policies and extension code segment to be transferred to active nodes securely by guaranteeing message integrity and confidentiality, and it also allows the PM and nodes to authenticate users.

The PM decides on an access policy for a user based on the identity of the user and the operations the extension would like to perform on the selected node. An access policy must specify the following information:

- The amount of resources, e.g., bandwidth, can be allocated to the extension.
- The filter envelope that defines the traffic that can be accessed. For example, the PM may only allow a user' s extension to install .Filters that have a source address field equal to the user' s host address. This way, the PM limits the extension to only access traffic that is initiated by the original user.
- Access rights that regulate operation on bandwidth. For example, the PM may only allow a extension to monitor the bandwidth usage of a resource node and disallow other operations on the node.
- Control over traffic processing. The PM may specify what type of processing modules can be applied to certain flows. For the processing that has network-wide effect, the PM must specify some extra parameters to prevent network-wide security violations. For example, the PM specifies the acceptable encapsulation source and destination addresses for tunneling. The PM determines this information based on the concept of a " virtual mesh" , which identities all the network resources a user is allowed to use [6]. The idea is that users should only be allowed to redirect traffic within their virtual mesh, so that they cannot affect other parts of the network.

We use *EI*, extension initiator, to represent the user who wants to install an extension. Suppose there are *n EI*s in the domain, and we number them *EI*1, ..., *EIn*. Suppose there are *m* active nodes in the domain and they are numbered *R*1, ..., *Rm*. We examine the case of *EIi* installing an extension on *Rj*. Before doing it, we need a priori registration and key distribution process. *EIi* shares a secret key with PM, *KEIi* and *Rj* shares a secret key with PM, *KRj*. Figure 2 and Figure 3 show the registration/key distribution process and main process (Extension-installing process) between *PM*, *EIi* and *Rj*.

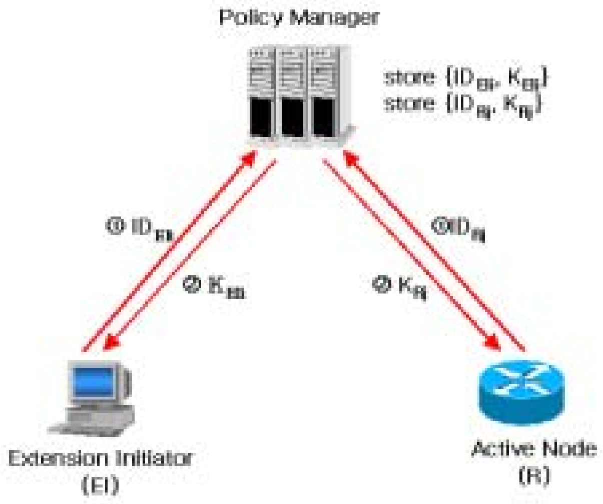


Fig. 2 Registration & Key Distribution Process

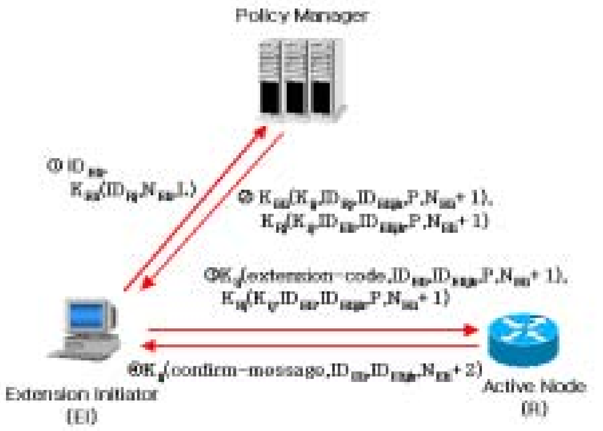
The registration and key distribution process contains the following message exchanges in order.

1. *EI* $\rightarrow$ *PM*: $\underline{ID_{EIi}}$, / *R* $\rightarrow$ *PM*: ID_{Rj}

EIi sends ID_{EIi} to PM for registering its identity and getting private key. *Rj* also sends ID_{Rj} to PM for the same reason.

2. *PM* $\rightarrow$ *EI*: $\underline{K_{EIi}}$, / *PM* $\rightarrow$ *R*: K_{Rj}

Receiving identities of *EIi* and *Rj*, PM generates their private keys, stores {ID_{EIi}, K_{EIi}} and {ID_{Rj}, K_{Rj}} in its database and sends each private key to *EIi* and *Rj*.


Policy Manager
① ID_EIi, K_EIi(ID_Rj,N_EIi1,)
② K_EIi(K_ij,ID_Rj,ID_EIi,P,N_EIi+1), K_Rj(K_ij,ID_EIi,ID_EIi,P,N_EIi+1)
③K_ij(extension-code,ID_EIi,ID_EIi,P,N_EIi+1), K_Rj(K_ij,ID_EIi,ID_EIi,P,N_EIi+1)
④K_ij(confirm-message,ID_Rj,ID_EIi,N_EIi+2)
Extension Initiator
(EI)
Active Node
(R)
Fig. 3 Extension-installing Process

We examine the case of *EIi* installing an extension on *Rj*. The protocol contains the following message exchanges in order.

1. $EI \rightarrow PM$: $\underline{ID_{EIi}, KEIi\ (ID_{Rj}, NEIi, L)}$

ID_{EIi} is the identifier of delegate *EIi*; ID_{Rj} is the identifier of target node *Rj* ; *NEIi* is a random sequence number chosen by *EIi* as a nonce to prevent replay; and *L* is a " proposal" that contains the list of resources that the extension to be installed will access on node *Rj*.

This message serves three purposes. First, this is the way that *EIi* authenticates itself to the PM. Upon receiving this message, PM uses the key corresponding to *EIi* to decrypt the second half of the message. If the message is successfully decrypted, the PM is then sure about the authenticity of the message and the freshness can be verified with the nonce. Second, *EIi* uses this message to request a session key to communicate with *Rj* ; Third, *EIi* requests an access policy, i.e., the set of access permissions for the extension it is installing.

After receiving the above message, the PM executes the following steps: (1) the PM creates a globally unique identity for the extension to be installed ID_{EIijk}, which means the *k*th extension *EIi* creates on *Rj* ; (2) the PM generates a session key *Kij* for communication between *EIi* and *Rj* ; (3) the PM produces an access policy, *P*, for this extension based on the identify of *EIi* and the proposal *L*. It then sends the following reply message.

2. $PM \rightarrow EI$:
$\underline{KEIi\ (Kij, ID_{Rj}, ID_{EIijk}, P, NEIi + 1), KRj\ (Kij, ID_{EIi}, ID_{EIijk}, P, NEIi + 1)}$

This reply message has two parts. Part 1 is encrypted with *EIi*' s secret key. *EIi* decrypts it to retrieve the session key, the identity for the extension, the access policy and a nounce. Part 2 is encrypted with *Rj*' s secret key; it contains the session key, extension identity and the policy. ID_{EIijk} and *P* together are called a credential.

3. $EI \rightarrow R$:
$\underline{Kij(extension\text{-}\ code, ID_{EIi}, ID_{EIijk}, P, NEIi + 1),}$
$\underline{KRj\ (Kij, ID_{EIi}, ID_{EIijk}, P, NEIi + 1)}$

This message also has two parts. The first part is the EI' s identity, the extension code, the credential and the nounce encrypted with the session key. For performance concern, the code part can be replaced with a message digest and the code itself can be sent in clear message if no secrecy of the code is required. An alternative to including the code with the request is to replace it with a reference to a secure code server. The second part is the same as the second part in the previous message.

When *Rj* receives the message, it decrypts the second part of the message to reveal the shared key *Kij* and the credential for this extension. At this stage, *Rj* believes that this part of the message is from PM because PM is the only other entity that knows *KRj* . *Rj* uses key *Kij* to decrypt the first part of the message. If successful, *Rj* now knows that this message is from *EIi*, since it is the only party, other than the PM, that knows *Kij* .

The EI identifiers and the sequence numbers in these two message parts must match to prevent tampering and replay attack.

4. $R \rightarrow EI$: $\underline{Kij(confirm\text{-}message, ID_{EIi}, ID_{EIijk}, NEIi + 2)}$

Rj sends this message to *EIi* to report the outcome of the extension installation: success or failure.

4. THE PROPOSED SCHEME

We suggest an improved method of transferring access policies. We use identity-based public key cryptosystem instead of private key cryptosystem like Kerberos[7][8]. The proposed scheme is a transformed public key cryptosystem. Instead of general public key cryptosystem that generates public/private key pairs randomly and opens public key to the public, identity, each entity (e.g., Extension Initiator or Active Node) has uniquely, plays a role of public key in identity-based public key cryptosystem. In our case, we use IP address as an identity. The private key corresponds to the public key is computed and distributed by PM at the key distribution process.

We use *EI*, extension initiator, to represent the user who wants to install an extension. Suppose there are *n EI*s in the domain, and we number them *EI*1, ..., *EIn*. Suppose there are *m* active nodes in the domain and they are numbered *R*1, ..., *Rm*. We examine the case of *EIi* installing an extension on *Rj*. Before doing it, we need a priori registration and key distribution process. *EIi* shares a secret key with PM, *KEIi* and *Rj* shares a secret key with PM, *KRj*. Figure 4 and Figure 5 show the registration/key distribution process and main process (Extension-installing process) between *PM*, *EIi* and *Rj*.
The registration and key distribution process contains the following message exchanges in order.

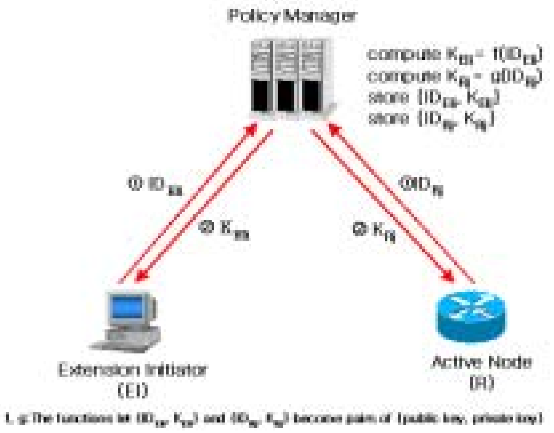

Policy Manager
compute $K_{EIi} = f(ID_{EIi})$
compute $K_{Rj} = g(ID_{Rj})$
store $\{ID_{EIi}, K_{EIi}\}$
store $\{ID_{Rj}, K_{Rj}\}$
① ID_{EIi}
② K_{EIi}
①ID_{Rj}
② K_{Rj}
Extension Initiator (EI)
Active Node (R)
f, g: The functions let $\{ID_{EIi}, K_{EIi}\}$ and $\{ID_{Rj}, K_{Rj}\}$ become pairs of {public key, private key}

Fig. 4 Registration and Key Distribution Process

1. $EI \rightarrow PM$: $\underline{ID_{EIi}, / R \rightarrow PM: ID_{Rj}}$

EIi sends to register identity ID_{Eii} to PM. *Rj* also sends to register ID_{Rj} to PM. Each identity will be used as a public key at main process.

2. $PM \rightarrow EI$: $\underline{K_{EIi}, / PM \rightarrow R: K_{Rj}}$

Receiving identities of *EIi* and *Rj*, PM generates their private keys as follows, stores $\{ID_{EIi}, K_{EIi}\}$ and $\{ID_{Rj}, K_{Rj}\}$ in its database and sends each private key to *EIi* and *Rj*. (f and g are functions let $\{ID_{EIi}, K_{EIi}\}$ and $\{ID_{Rj}, K_{Rj}\}$ become pairs of {public key, private key})

$$K_{EIi} = f(ID_{EIi}),\ K_{Rj} = g(ID_{Rj}) \qquad (1)$$

We examine the case of *EIi* installing an extension on *Rj*. The protocol contains the following message exchanges in order.

1. $EI \rightarrow PM$: $\underline{ID_{EIi}, KEIi\ (ID_{Rj}, NEIi, L)}$

ID_{EIi} is the identifier of delegate *EIi*; ID_{Rj} is the identifier of target node *Rj* ; *NEIi* is a random sequence number chosen by *EIi* as a nounce to prevent replay; and *L* is a “ proposal” that contains the list of resources that the extension to be installed will access on node *Rj*.

This message serves three purposes. First, this is the way that *EIi* authenticates itself to the PM. Upon receiving this message, PM uses the key corresponding to *EIi* to decrypt the second half of the message. If the message is successfully decrypted, the PM is then sure about the authenticity of the message and the freshness can be verified with the nounce. Second, *EIi* uses this message to request a session key to communicate with *Rj* ; Third, *EIi* requests an access policy, i.e., the set of access permissions for the extension it is installing.

After receiving the above message, the PM executes the following steps: (1) the PM creates a globally unique identity for the extension to be installed ID_{EIijk}, which means the *k*th extension *EIi* creates on *Rj* ; (2) the PM generates a session key *Kij* for communication between *EIi* and *Rj* ; (3) the PM produces an access policy, *P*, for this extension based on the identify of *EIi* and the proposal *L*. It then sends the following reply message.

2. *PM* $\rightarrow$ *EI*: $\underline{KEIi\ (ID_{EIijk}, P, NEIi + 1)}$

This message is encrypted using *EIi*’s private key K_{EIi} PM has. *EIi* decrypts this message to get identity, P, and nounce.

3. *EI* $\rightarrow$ *R*: $\underline{ID_{Rj}(extension\text{-}\ code, ID_{EIi}, ID_{EIijk},\ P, NEIi + 1)}$

This message consists of *EIi*’s identity, extension code, credential, and nounce and encrypted using active node *Rj*’s public key ID_{Rj}.

4. *R* $\rightarrow$ *EI*: $\underline{ID_{EIi}(confirm\text{-}message, ID_{EIi}, ID_{EIijk}, NEIi + 2)}$

Rj sends this message, encrypted confirm-message using *EIi*’s public key ID_{EIi}, to *EIi* to report the outcome of the extension installation: success or failure.

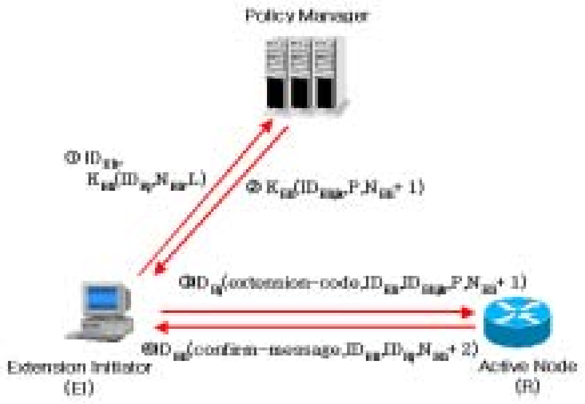

Policy Manager
① ID_EIi, K_EIi(ID_Rj, N_EIi, L)
② K_EIi(ID_EIijk, P, N_EIi + 1)
③ID_Rj(extension-code, ID_EIi, ID_EIijk, P, N_EIi + 1)
④ID_EIi(confirm-message, ID_EIi, ID_Rj, N_EIi + 2)
Extension Initiator (EI)
Active Node (R)

Fig. 5 Extension-installing Process

5. CONCLUSION

We suggested an improved method of transferring access policy between active nodes in active network environment. This proposed scheme adopts public key cryptosystem, but it is secure, lightweight and need not manage public key directory. Therefore, it can be used in real world.

REFERENCES

[1] K.Psounis, “Active networks: Applications, Security, Safety, and Architectures”, *IEEE Communications Surveys*, 1999.

[2] Y.S.Kim, J.C.Na, S.W.Sohn, “A Secure Active Packet Transfer using Cryptographic Techniques”, *Journal of The Korean Institute of Information Security and Cryptology*, pp.135-145, 2002.

[3] J.G.Steiner, B.C.Neuman, and J.I.Schiller, “Kerberos: An Authentication Service for Open Network Systems”, *Proceedings of Winter USENIX Conference*, pp.191-201, 1988

[4] George Necula and Peter Lee, “Safe Kernel Extensions Without Run-Time Checking”, *In Proceedings 2nd Symposium on Operating Systems Design and Implementation (OSDI’96)*, pp.229-243, 1996

[5] I.Stoica, H.Zhang, and T.S.Eugen, “A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Service”, *Proceedings of the SIGCOMM’97 Symposium on Communications Architectures and Protocols*, pp.249-262, 1998

[6] Prashant Chandra, Allan Fisher, Corey Kosak, T. S. Eugene Ng, Peter Steenkiste, Eduardo Takahashi, and Hui Zhang, “Darwin: Customizable Resource Management for Value-Added Network Services”, *In Sixth International Conference on Network Protocols*, pp.177-188, Austin, October 1997

[7] B.Schneier, “Applied Cryptography: Second Edition”, *Wiley*, 1996..

[8] A.Shamir, “Identity-Based Cryptosystems and Signature Schemes”, *Proceedings of CRYPTO ’84*, Springer-Verlag, pp.47-53, 1985.