CHAPTER 3

SECURITY ARCHITECTURE

3.1 INTRODUCTION

A distributed system is a major concern in the today’s network world for fast accessing. The development of distributed systems is affected by the need of flexibility, adaptability and autonomy (Aderounumu et al 2006). Mobile agent technology is an active solution to fulfill the needs of flexibility, adaptability and autonomy in distributed systems, to make them smart. For effective mobility, agent platforms use the Agent Transfer Protocol (ATP) (Lange and Aridor 1997) to transfer the mobile agent between the hosts effectively. Nevertheless, security is a major issue in mobile agent technology that degrades its performance. As per the Yao (2004) report, security falls mainly into two areas:

(a) Protecting a Server from Malicious Agents: The server in a mobile agent environment can be attacked by malicious agents with illegal codes. Attacks may be the denial of service or Unauthorized Access or Masquerade. To protect the servers from the malicious agent, authentication and byte code verification are the major security requirements.

(b) Protecting a Mobile agent from Malicious Servers: A mobile agent roaming in the distributed network can be attacked by malicious platforms. A malicious platform may modify or kill the agent. Protecting the mobile agent from the malicious
server is a more critical issue than protecting the server because the agent dispatched to remote servers is fully under the control of the remote server. The major security requirements for protecting the mobile agent environment are confidentiality and integrity.

To protect the agent and platform from alternating attacks, a security architecture is proposed with multiple protection techniques.

3.2 SECURITY PROPERTIES

The mobile agent and its environment are protected by having the various security properties against the malicious attackers. The security properties are:

i) Confidentiality: A Mobile agent will roam in the network to collect the information. The information collected from the remote machines should be kept secret from other remote machines during the travel. For example, Subbu wants to fly to Australia and he wants to know the cost details from various travel agencies. An agent is dispatched to collect the information on behalf of Subbu. It will visit multiple agencies and gather the information. The information gathered from one agency should be kept secret from the other agency. Otherwise, an agency will fine-tune its cost details based on the other agency to give preference to choose its flight.

ii) Integrity: Accuracy is the essential thing in all the scenarios. In the mobile agent environment, the agent code and its information should be original (it has to be always the same), not modified by any of the malicious servers.
iii) Non-repudiation: The server which sends the Information to the owner of the agent or to the remote server should not deny that he is not the owner of the particular information and agent.

iv) Authentication: The mobile agent or its information from the remote host must be authenticated before it is allowed to execute or perform execution on that. The server, which receives the agent should validate whether the data or agent is from the valid user or not.

v) Byte Code Verification: The agent server is protected by verifying the code of the migrated agent from the remote host. The code is verified before the execution of the agent starts by the server. The verifier should scan the code of the agent to identify whether it has any illegal operation.

3.3 REQUIREMENTS FOR SECURITY PROPERTIES

Security properties to the mobile agent environment are provided by incorporating various cryptography requirements. The requirements to fulfill the security properties of the mobile agent environment are:

i) Digital Signature: It is used to prove that the message was generated by a particular domain. It is particularly used for the non-repudiation purposes.

ii) Hash function: The hash function or message digest is a mathematical transformation that takes the arbitrary length of the message as the input and computes a fixed length of the hexadecimal number. It is a one-way function because it is not able to figure out what input corresponds to a given output.
iii) Key Cryptography: It is used to send information between participants in a way that prevents others from reading or modifying it.

iv) Scanner: It is used to scan the incoming agent for identifying the illegal (malicious codes) activities in the agent byte code to attack the mobile agent platform.

3.4 MOBILE AGENT SERVER SECURITY ARCHITECTURE

The mobile agent in a distributed network will roam among remote hosts to collect the information by computing some task. To protect the agent and the platform during migration, each platform should perform some verification and computation based on previous works. The mobile agent server security architecture proposed in this thesis also requires verification and computation as shown in Figure 3.1. In the design of the mobile agent security architecture, the decision of the next host (Next Host Decider) is added to give smartness to the multi-hop mobile agent with a dynamic order.

Initially, the agent from its originator consists of the byte code, state, itinerary, credential (identification) and dummy information. In addition to this, the encrypted hash code is added into the credential to detect any modification in the agent code. The host which receives the agent will start the protection models in the architecture to verify the integrity and malicious activity of the agent. The proposed mobile agent server security architecture has five models, namely: (i) Malicious Identification Police (MIP) Scanner, (ii) Code Integrity Checker (CIC), (iii) Information Verifier and Generator (IVG), (iv) Next Host Decider and (v) Recovery.

The process of the security architecture after receiving the agent is: every platform should initially start the MIP to scan the incoming agent to identify malicious codes. If the MIP did not find any malicious activity in the agent code, it will send the control to the Code Integrity Checker to start its
execution. It will check the integrity of the agent code and transfer the control to the Information Verifier and Generator (IVG) to check the integrity of the information. Then the Next Host Decider will take control to choose the succeeding remote host from the static itinerary or dynamic itinerary and then the clone is taken by the recovery model before dispatching the mobile agent to the succeeding host. The sequence diagram of the mobile agent server architecture is shown in Figure 3.2 and the verification sequence of the architecture is shown in Figure 3.3.

Figure 3.1 Mobile Agent Server Security Architecture
Figure 3.2  Sequence Diagram of the Mobile Agent Server Security Architecture

The verification process of the architecture after receiving an agent by the hosts is shown in Figure 3.3. The MIP will take the agent code to detect the platform attack instructions. If it finds any malicious agent, it will kill it and inform to the authority. In the case of a legitimate agent, the code integrity is verified by the CIC, and waits for the clone in the case of a modified code; otherwise, the offer (data) integrity is verified by IVG. If it finds that the data is modified, then it will intimate it to the agent owner and wait for the clone to continue with the legitimate agent. Otherwise, the information required by the agent is provided or it allows the agent to perform its computation.
If the MIP or Code Integrity Checker or Information Verifier finds any malicious activity with the agent, it will immediately intimate it to the administrator to take necessary action, or the host itself may take some immediate action. The descriptions of the protection and recovery models in the security architecture are as follows.

**Figure 3.3 Verification Process of the Mobile Agent Server Security Architecture**
3.4.1 MIP Scanner

The purpose of the Malicious Identification Police (MIP) scanner is to protect the mobile agent platform from a malicious host or agent. Its process is like the security (person) having a scanner or detector to scan people before allowing them to enter a sensitive place. The Malicious Identification Police is used to scan the byte code of the agent to detect whether the agent consists of any illegal code or not. Today’s Java has a code verification mechanism to identify the object initialization, stack overflow and underflow (Xavier 2003), but it will not prevent malicious access, exiting the platform, denial of service, etc. The MIP with the Attack Identification Scanner (AIS) will prevent all types of attacks generated by the cruel agents, and the scanning is based on the privileges given to the agent originator.

3.4.2 Code Integrity Checker (CIC)

Apart from platform protection, the proposed Code Integrity Checker (CIC) serves to protect the agent code alteration with the Root canal (RC) and the eXtended Root Canal algorithm (XRC). The RC algorithm is the same as the code signing model (Joseph and Luis 1996) and the self-protection scheme (Pedro et al 2006) with the minimum number of cryptographic mechanisms, but it will not prevent a false malicious claim. To prevent the false malicious claim, the algorithm is modified as the eXtended Root Canal (XRC) algorithm. The functions of the XRC are different for the originator, first remote host and the forthcoming succeeding remote hosts.

- Originator: The originator should sign the agent code using its private key and dispatch the mobile agent to the remote host.
- First Remote Host: It should verify the signature of the originator with the originator’s public key while receiving the
agent and it should sign on the originator’s signature using its private key.

- Forthcoming Remote Host: It should verify the signature of the preceding host first and then the signature of the originator. The first verification is to prevent a false malicious claim, and the second verification is to detect the alteration of the agent code. After that it will discard the signature of the preceding host and sign on the originator’s signature using its private key.

3.4.3 Information Verifier and Generator (IVG)

In addition to the agent code protection model, the newly proposed 3-ID verification algorithm in the Information Verifier and Generator aims to protect the agent data (offer or information) from multiple colluded attackers in adjacent places and also the attacks protected on the one hop backward and two hop forward (Xu et al 2006) mechanisms. The 3-ID verification algorithm consists of a chain and each entity of the chain consists of three identities (preceding host, current host and the succeeding host). To protect the information, every host should verify the identity integrity of the chain. As per the 3-ID algorithm, an alteration in the information leads to an alteration in the identity and it is detected. The purpose of the Information Generator is to provide the required information to the agent.

3.4.4 Next Host Decider

The Next Host Decider of the architecture is a non-security model to route the multi-hop mobile agent with a dynamic order. The Next Host Decider has two functions: (i) one is to generate the routing table periodically
to improve the journey smartness of the agent and (ii) the other one is to decide the succeeding host for the agent.

(i) Generating the routing table: Every host in the environment should have the agent to calculate the link cost among the hosts and to generate the routing table and update it periodically. The routing table consists of the remote host address and the link cost (time to reach the respective remote host). The link cost is calculated, based on the agent’s dispatch time from the home \(T_D\) and the arrival time of agent \(T_A\) at the remote host, as follows:

\[
\text{Link Cost} = T_A - T_D
\]

(ii) Deciding Next Host: An agent roaming in the network with a static itinerary dynamic order depends on the remote hosts to decide the succeeding host. The decision of the next host is based on the condition that may be based on the shortest path or the network traffic. This work concentrates on the shortest path because the network traffic is known by every host in the network. The shortest path details of all the hosts are available with the routing table to decide the next nearest host. The routing table available with the host is not constant all the time, for it will be modified periodically considering the network traffic. The link cost has to be calculated periodically to update the table from time to time because of the network traffic. Hence, no other host can identify the decision of the succeeding host.
### 3.4.5 Recovery Model

Apart from detecting the attack on the agent, recovery is an important factor to bring the original agent back. The proposed $K$-response in the recovery model is to recover the agent after the single host or multiple host colluding attack. As per this model, every host should maintain the agent clone until the response comes from the $K$ number of the succeeding remote hosts. The value of $K$ is computed based on the empirical formula derived from the total number of nodes to be visited by the agent ($N$) and the number of nodes yet to be visited by the agent ($Y$).

### 3.5 PROOF OF ARCHITECTURE

The formal proof of recovery of the agent by the recovery model and the identification and prevention of attacks by the protection models embedded in the security architecture are given below:

**Lemma 1**: Any agent with a malicious code can be identified and prevented by the MIP.

**Proof**: According to the MIP model, the agent from the remote host has to be scanned by the AIS scanner before allowing the agent to perform its process. This is to identify the malicious code with the agent. The agent with the $M_{Code}$ can be identified by the MIP with the AIS scanner.

\[
f(A) = \{A_{code}\} \cap \{M_{code}\} = \begin{cases} \emptyset, & \text{Genuine agent} \\ M_{Code}, & \text{Malicious Agent} \end{cases}
\]  

(3.2)

From the equation (3.2), any agent with the malicious code can be identified and killed by the hosts. Hence, lemma is proved for attack identification and prevention by the MIP.
Lemma 2: In the RC algorithm, an active attack on the agent code is identifiable.

Proof: According to the algorithm, the agent migrating from the owner consists of the encrypted hash value for the agent code. This is because, every remote host should decrypt the hash value using the owner’s public key and also generate the hash code for the received agent code and compare the two (hash code from the agent owner and the generated hash code).

\[ f(A) = (H_{\text{code}}, \oplus RH_{\text{code}}) = \begin{cases} 
= 0, & \text{No modification} \\
= 1, & \text{Modification in the Agent} 
\end{cases} \tag{3.3} \]

If the result of equation (3.3) is zero, then there is a contradiction between the hash codes. Hence, the active attack on the agent code is identifiable by the RC algorithm and the lemma is proved.

Lemma 3: In the 3-ID algorithm, offer alteration is identifiable.

Proof: According to the algorithm, offers of every host should be encapsulated with its credentials. The part of the credential consists of the identity of the previous host, current host (who is the owner of the particular encapsulated offer) and the succeeding hosts. This is because; the succeeding host \( h_i \) receiving the agent should compare the last two identities of the preceding host \( h_{i-2} \) with the first two identities of the preceding host \( h_{i-1} \) for integrity. In the case of an alteration, the encapsulated offer reflects the contradiction of the identity similarity. Hence, lemma is proved for the 3-ID algorithm identification of the offer alteration by the single host or set of hosts.
Lemma 4: In the $K$-response model, a killed or damaged agent is recoverable.

Proof: According to the $K$-response model, every host should have the clone of the agent before dispatching it to the succeeding host and every host should send at least one response to the preceding host. This is because, if the host does not get the response from the $K$-succeeding host within the time out period of the clone (in case of agent failure), the preceding host has to resend the clone as the primary agent to the network. Hence, the failed agent is recoverable with the $K$-response model and the lemma is proved.

Theorem 1: Any malicious agent can be identified and the original agent can be recovered by the proposed security architecture.

Proof: From Lemma “1”, the MIP scanner will prevent the execution of an incoming malicious agent. Lemma “2” proves that the agent alteration in the middle of the itinerary can be identified. Lemma “3” proves that offer alteration is identifiable. Lemma “4” proves that the $K$-response model will recover the failed agent. By Lemmas “1” to “4”, all malicious agents can be identified and the original agent recovered by the proposed security architecture.

3.6 SUMMARY

In this chapter, a new security architecture is proposed to protect the mobile agent environments. The agent who migrated from one server to another server will immediately validate its integrity. After the validation is successful, the agent is allowed to continue its computation in the environment. It is easy to identify the illegal modification in the information,
and the agent code or illegal code available in the agent to assail the agent platform or server.

Any illegal modification or change identified in the information or code of the agent can be intimated to the agent owner or administrator of the distributed environment to take necessary action against the malicious servers. This mobile agent server security architecture is useful for incorporating the mobile agent in any environment like e-Voting (Robles 1999), ad hoc networks (Levy et al 2005), sensor networks (Wang and Qi 2004), Intrusion Detection System (Hijazi and Nasser 2005), etc.