A SURVEY OF SECURE ADDRESS AUTO-CONFIGURATION IN MANET

Majid Taghiloo1,2, Jamshid Taghiloo3, Mehdi Dehghan1

1- Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
2- Network Security Department, Nasr Electronics Research Center, Tehran, Iran
3- Computer Engineering Department, Iran University of Science and Technology, Tehran, Iran

ABSTRACT

Mobile ad hoc networking offers convenient infrastructure-less communication over the shared wireless channels. The major research efforts in the area of Mobile Ad hoc Network (MANET) focus on developing efficient routing protocols assuming that IP addresses of mobile hosts have been already configured before joining the network. Providing security for IP address auto-configuration in MANET is still an open problem. Existing solutions for IP address auto-configuration in MANET do not address security issues. However, it may disserve network security if there is not a reliable authentication mechanism, e.g. an attacker may spoof other nodes and hijack their traffic. Unlike their wired counterpart, infrastructure-less ad hoc networks do not have a clear line of defense, and every node must be prepared for encountering with an adversary. Therefore, a centralized or hierarchical network security solution does not work well. This paper describes the state of research in secure address auto-configuration problem and analyzes the current existing solutions.

1. INTRODUCTION

An ad-hoc network is a collection of temporary nodes that are capable of dynamically forming a temporary network without the support of any centralized fixed infrastructure. These networks can be formed, merged or partitioned into separate networks on the fly, without necessarily relying on a fixed infrastructure to manage the operation.

Two important properties of an ad-hoc network are that it is self-organized and adaptive. ‘Self organizing’ implies that a network can be formed on the fly and then changes its topology without the presence of system administration entities. The term ‘adaptive’ simply implies that an ad-hoc network can take different forms and has highly variable mobile characteristics such as power and transmission conditions, traffic distribution variations, and load balancing. In fixed infrastructure networks, The IPv6 address configuration procedures [1] allow a more dynamic distribution and discovery of the routing information, simplify mobility and network management. Although these mechanisms bring known benefits, they also open the door for new security threads, many of which have been identified during the design of security solutions for Mobile IP [2]. Wireless links are vulnerable to both passive and active attacks, such as eavesdropping and denial-of-service attacks. Potential damage includes compromise of transmitted secret information, interfering with messages, and impersonating nodes. Limited bandwidth of wireless connections also gives a target for denial-of-service attacks. The lack of central servers and user administration in a zero configuration setting also raises serious issues about security. Where there is no centralized management or user intervention, key generation, distribution and maintenance become very difficult. As a result, providing security services such as access control, data integrity and authentication of nodes, which require cryptographic keys, is difficult. Ubiquitous computing environments can continuously change over time because of the movement of devices. Any security solution involving static configuration of a node is therefore inappropriate because of the dynamic topology of the network. To achieve high availability, a distributed architecture without reliance on central management entity is needed.

For security reasons, each node must be able to prove its membership to the network. Moreover, a node should not be able to join the network with its own IP address. On the contrary, the network must impose it. That makes it possible to avoid some basic addresses conflicts. Lastly, the protocol must take care as far as possible not to offer possibility of Denial of service attack. For example, one should not allow a malicious node to monopolize all the addresses in the network.

The rest of this paper is organized as follow. In section 2, we list the major security threads associated with address auto-configuration in MANET. In section 3, we present secure addressing issue, requirements and solutions in MANET. Finally, after analyzing these solutions in section 4, a brief conclusion is provided in section 5.
2. MANET AUTO-CONFIGURATION ATTACKS

Potential damage includes compromise of transmitted secret information, interfering with messages, and impersonating nodes. Wireless links are vulnerable to both passive and active attacks, such as eavesdropping and denial-of-service attacks. In address auto-configuration, active attack can affect but also passive attack can cause to doing active attack. For listing possible address auto-configuration attack, we use well attack classification proposed in [3], for security analyzing:

- Address Spoofing Attack: Without an authentication mechanism, a malicious node can freely choose any configured node as a victim, spoof its IP address, and hijack its traffic.
- False Address Conflict Attack: An attack may purposely transmit a false address conflict message to a targeted victim. Since the victim cannot verify the authenticity of the supposed address conflict, it may have to give up its current address and seek a new one.
- Denial of service Attack: An attacker could maliciously claim as many IP addresses as possible. If all valid IP addresses is exhausted by the attacker, a newly arrived node will not be able to get an IP address.
- Negative Reply Attack: In some previous work, the assignment of a new address requires an approval of all configured nodes. An attacker therefore may continuously send negative replies to prevent a newly arrived node from getting an address.

3. CURRENT SOLUTIONS CLASSIFICATION

The research on MANET address Auto-Configuration security is still in its early stage. Currently, few papers presented in this area [3]-[5]. In this section, we briefly describe some of the existing proposals for Secure IP address auto-configuration, classifying them into three different categories: Self-Authentication, Challenge Response, and Trust Modeling.

3.1. Self-Authentication

This scheme uses self-authentication method for doing secure address auto-configuration. By using one-way hash function, it binds a node’s address with a public key. Thus, the owner of an address can use the corresponding public key to unilaterally authenticate itself. This solution uses all address space for auto-configuration. It assumes the nodes in MANET keep loose time synchronization. New node (like A) first randomly generates a public/private key pair \((k_p, k_s)\) and a secret key \((q)\), and immediately saves these keys to a secure local storage. Then, node A calculates a 32-bit (in IPv4) or a 128-bit (in IPv6) hash value of \(k_p\), i.e., \(a = H(k_p)\), where \(H\) is a secure one-way hash function. Next, node A temporarily uses the resulting value \(a\) as its IP address, initiates a timer, and broadcasts a Duplicate Address Prob (DAP) message. This message is illustrated below, where \(t^r\) is a time-stamp that generated by node A, \(E(t^q, q)\) means the encryption of \(t^q\) using the secret key \(q\), and \(\|\) means the conjunction. \(DAP = \{IP, k_p, t^r, E(t^q, q), Sig\{0 \||\| t^q \| E(t^q, q)\}\}\)

If a configured node (call it B), finds that the IP address in a received DAP message is same as its own, it will verify the authenticity of this DAP message. Node B first tests whether \(IP = H(k_p^B)\) holds. If it does, node B then verifies the signature using the received \(k_p^B\). If the signature proves correct, node B further verifies whether \(k_s^B = k_s^A\) holds, and whether \(E(t^q_B, q) = t^q_A\) holds, to prevent the possibility of replay attack. \(D\{M\}\) means the decryption of a message M with the secret key \(q\). If at least one of these two equalities does not hold, node B then broadcasts an Address Conflict Notice (ACN) message as shown below, to inform the corresponding node A of the address conflict, where \(k_p^B\) is a time stamp generated by node B. Otherwise, node B simply discards the received DAP message.

\(ACN = \{IP, 1, k_p^B, t^r_B, E(t^q_B, q), Sig\{1 \||\| t^r_B \| E(t^q_B, q)\}\}\)

If no ACN message is received before the timer expires, node A assumes that the IP address \(a\) is not in use and begins using this address. If node A receives an ACN message from another node, e.g. node B, it first verifies this ACN message before trying another IP address. Node A checks the embedded time stamp \(t^r\) to see whether the received ACN is expired. If not, node A further checks whether \(IP = H(k_p^A)\) holds and whether the signature is correct based on the embedded \(k_p^B\). If all the verification results are positive, node A is sure that IP address is in use by another node. Then, it has to repeat the auto-configuration procedure with another pair of public/private keys. Otherwise, node A simply discards the received ACN message.

It is possible that two configured nodes happen to have the same IP address after a merger of partitions. Here, when the address conflict is detected, e.g. by employing the passive detection method proposed in [6], one of these two nodes (e.g. node A) broadcasts an ACN message. Once receiving the ACN message, the other node (e.g. node B) first verifies the authenticity of this ACN as was discussed earlier. If the verification result is positive, node B then checks whether it is the source of this ACN message to prevent the reply attack. Node B checks whether the embedded public key, \(k_p\), is same as its own,
deciphers the cipher-text \( E(t) \) with its secret key \( q^B \), and checks whether the resulting value matches the embedded time stamp \( t \). If all results are positive, node \( B \) treats this ACN message as a replayed one and simply discards it. Otherwise, node \( B \) gives up its current IP address and starts a procedure of address auto-configuration.

3.2. Challenge Response

This algorithm employs the concept of challenge which obliges a node to answer a question to prove its identity. Concretely, a node wishing to join the network sends a request with its public key and, as a temporary identifier, its MAC address. Afterward, the neighbors calculate separately a nonce that they will return to the requester, after having ciphered it with the public key. The mission of the requester is then to return this nonce incremented to the concerned nodes, after having ciphered it this time with its private key.

The address allocation protocol in [4] reuses the buddy systems technique which was already employed in [7]. However, it differs by the fact that the nodes here do not have any more responsibility of entire addresses ranges but just point on (and are pointed by) their neighbors of address. The term point indicates here the fact of maintaining in memory a reference on the next node which was present during the IP address allocation.

For doing authentication, each nodes have separate role and behavior. When node wishes to join the ad hoc network, it must broadcast a request with its MAC address as a temporary address. In this intention, it creates a message to which it joins its public key. It initializes a timeout in the case where it would be the first node of a new network. It broadcast the whole to all its neighbors. The neighbors receive a request message and they must answer by a challenge message. Thus, they randomly generate a nonce coded on 16 bits, which they cipher with the public key of the requester. They store the co-ordinates of the requester (MAC address, nonce, public key) in a buffer. When the requester receives the challenge, it decipheres the messages using its private key. For each one of them, it extracts the public key of the transmitter, the nonce, and increments this last one. In parallel, it generates itself, for each certifier, an ID-nonce which will be used for identifying it until it obtains its IP address. Then, it re-ciphers the whole with the public key corresponding to each certifier and replies it with an answer message. After receiving the answer by neighbor, the neighbor node records in a special table, an association (public key / ID-nonce) for a limited period.

In address allocation phase the requester sends a unicast request for address message to one of its randomly chosen neighbors, by pointing out the nonce that was chosen in the preceding phase and by indicating its public key. When a neighbor receive request for address message, it checks that the nonce corresponds really to a node which it has just identified and that the timeout was not reached. If the nonce does not correspond to that present in its table, it discards the request. In the contrary case, it can begin the address prospection phase: it calculates the address range which separates itself from the node on which it points and divides it by two. It keeps the first part and gives the other one to the requester, by informing it by an address allocation message. Then, it launches a timeout and waits for an acknowledgement from the requester. When this one arrives, the node updates its references by pointing from now on the claiming node. When the requesters receive its address, it sends an acknowledgement to the node from which it received the address. It updates its references because it is from now on, pointed by its tutor. It must then obtain certificates from its neighbors. In this intention, it broadcasts a request for certificate, by always specifying the nonce that it had initially chosen, by joining its public key and by registering its IP address as the destination.

A node when receive a request for certificate message, It checks whether the nonce always corresponds to that present in its table. Then, it generates a certificate which must authenticate the association public key / IP address of the requester, which will be able from now, to join it to all its messages in order to prove his identity.

When a node sends a connection request and does not receive any response at the end of a fixed time, it deduces from it that it is alone. In this case, it allows itself the first address of the network and initializes its references compared to itself. Consequently, when a second node arrives, it allows to it the address which corresponds to the half of the available space of addressing, and updates its pointers.

The responsibility for address allocation is entrusted to an already configured node. Each one is authenticated by one or more other participants. The security level depends on the number of different certificates that a node is able to present.

3.2. Trust Modeling

In this method, trust reflects the degree of belief that one entity has in the correctness of the behavior of another entity. It is dynamic, and reduces if an entity misbehaves, and vice versa. The level of trust \( A \) has in \( B \), is a rational (floating point) value denoted by \( TA(B) \). Every node \( A \) has a threshold trust value, denoted by \( TA^* \). That is, \( A \) deems \( B \) as trustable if and only if the trust value that \( A \) currently assigns to \( B \) is at least its threshold trust value, i.e. \( TA(B) \geq TA^* \); otherwise node \( A \) will regard node \( B \) as a potentially malicious node. Different nodes may choose different trust thresholds. Hence, the definition of malicious node may vary from node to node, depending on local policy. Every node also keeps a blacklist.
Whenever it finds another node for which its trust value is lower than its threshold trust value, it adds this node to its blacklist. Except for the messages used for calculating trust values, it will ignore all other messages from nodes in its blacklist and will not route any other messages to these nodes. Any nodes in a passive mode can gather information and use this information to make a direct trust judgment on its neighbor nodes. If one of its neighbor nodes is malicious, it can detect the misbehavior of this malicious node. It maintains trust values for all its neighbor nodes and regularly updates them. In addition, any node is able to calculate the trust values of non-neighbor nodes based on the trust values kept by itself and/or other nodes. It assumes that a majority of the nodes in the network are honest, the probability that a malicious node will be chosen as an initiator is low. The trust model can be used to discover possible malicious nodes for preventing DoS attack. If initiator receives a reply message from the other node like B that indicate address conflict, if B is not in initiator blacklist, then initiator either maintains a trust value for B or can calculate one (if B is not a neighbor node). If initiator’s trust value for B is greater than or equal to TA*, then A believes that the candidate IP address is already being used. Otherwise, initiator deems B a malicious node. And adds B to its blacklist and ignores this address conflict message. Initiator also broadcast malicious suspect message about B to all other node.

4. SECURITY AND PROTOCOL ANALYZING

4.1. Self-Authentication Analyzing

This approach doesn’t entirely solve the key-setup problem. It forces an attacker to find a public key whose hash value equals the victim’s IP address before being able to launch an attack. To find such a key requires a number of computations or storage of key/address pairs. The cost for an attacker to break this protocol relies on the size of the address allocation space. The security performance is relatively poor for IPv4. An attacker may relatively easily find a public key whose hash value equals a given 32-bit number. This protocol use idea from [8], which uses some additional information combined with IP address to uniquely identify a node.

When Address Spoofing Attack occurred, the attacker should replay DAP message for spoiling the IP address. The time stamp generated by owner of IP address can prevent to replay attack. In purposely transmit a false address conflict message to a target victim, the attacker should send ACN message. The victim first verifies this ACN message before trying another IP address. After checking time stamp, victim node further checks whether \( IP = H(k^P_B) \) holds and whether the signature is correct based on the embedded \( k^B_P \). Unauthorized node can not pass these checks. Similar mechanism will prevent from occurring Denial of service Attack and Negative Reply Attack. This mechanism use cryptographic functions like hash, cipher and signature, to do self-authentication. Therefore, it can not protect auto-configuration protocol from resource consumption attack. Each node to verifying DAP and ACN messages should use two encryption and one signature operations.

4.2. Challenge response analyzing

The use of a stateful protocol presents the advantage to limit the addresses conflicts probability. Hence, there is no need for duplicate address detection mechanism and costful periodical update to maintain coherent the global address space. Then, since the nodes do not have anymore the responsibility of whose address pool, the brutal departure of a node becomes less significant for the management of global address space. Indeed In this approach, such event has only as a consequence to modify the references of the nodes pointing and be pointed by the leaver.

Chaining references allow increasing significantly the search speed for free address. Each node keeps references on two neighbors. Consequently, when a node does not have any more available address, it can either seek the free address two by two while making use of its references or relay the request to its authenticator. In this approach, as soon as the requesting node was identified, it can immediately obtain its address without needing to flood the entire network. One of the drawbacks of this protocol is the cost of reconnections in the case of networks merger. Such events oblige a lot of nodes to repeat the address allocation phase. This may also have significant impact on the network traffic and break all connections at the transport layer. In addition, network mergers may lead to security problems since one of the networks could be only composed of colluded malicious nodes. Moreover, it is also very hazardous to base on the sender address of a packet to punish or exclude a malicious node since we cannot guarantee the origin of messages. The algorithm presented in this scheme makes it possible to prevent the spoofing attack. Indeed, before accepting a node in the network, the protocol takes care to checking the validity of its public key and the identity of the node is then certified by several other already authenticated nodes. Thus, even if a node wants to corrupt the network by providing to the requester already allotted addresses; it will always be possible to find its trace through the address contained in the pointers, to eventually exclude it from the network by refusing to validate its certificates. On the other hand, the protocol does not make it possible to prevent a malicious behavior from the participants.
after their authentication. Once that a node is authenticated and has obtained an address it can perform other attacks like Denial of service.

4.3. Trust modeling analyzing

In this protocol, only trusted nodes will be chosen as initiators for address allocation. Malicious nodes will be detected and isolated with the cooperation of other network nodes. Only minor computational costs will be incurred if the number of malicious nodes is small. However, when a malicious node acts as a requester and simultaneously asks many initiators for IP address allocation, each initiator will treat this malicious node as a new node and will not be able to calculate a trust value for it. This kind of attack cannot be prevented by using the trust model approach described in this scheme. Alternatively, a node can behave honestly for all network interactions, and only behave maliciously with respect to trust model functionality. In this scheme, malicious nodes of this type will not be detected by other nodes, and the calculation of trust values will potentially be affected by these nodes. This scheme assumes that misbehaving node will be detected by its neighbor nodes. The number of malicious nodes that this trust model can tolerate varies depending on the network topology. This model requires at least one valid route from A to B in order to calculate the trust value of A for B. If no valid route from A to B and A regards B as an honest node, a malicious node B can attack this trust model by modifying the route record. Table 1 presents a comparison of the aforementioned methods. The first six rows are a characteristics summary of the three secure allocation algorithms. The last three rows focus on the security or the aforementioned methods. The time between the points when a node initiates auto-configuration and the one when it is assigned a free IP address is referred as latency. Process Overhead related to resource consumption.

5. CONCLUSIONS

This paper discusses several security protocols in implementing secure address auto-configuration for a MANET. Optimistic approaches can provide a better trade off between security and performance. It would be interesting to work on possible optimizations like allowing the merging of network or using less resource greedy cryptographic mechanisms.

6. REFERENCES


<table>
<thead>
<tr>
<th>Protocols</th>
<th>Self-Authentication</th>
<th>Challenge response</th>
<th>Trusted modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Maintainence</td>
<td>Partially stateful</td>
<td>Partially stateful</td>
<td>Partially stateful</td>
</tr>
<tr>
<td>Address reclamation</td>
<td>Unneeded</td>
<td>Needed</td>
<td>Unneeded</td>
</tr>
<tr>
<td>Complexity</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Process Overhead</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Evenness of distribution</td>
<td>Even</td>
<td>Possibly uneven</td>
<td>Even</td>
</tr>
<tr>
<td>Latency</td>
<td>High</td>
<td>low</td>
<td>High</td>
</tr>
<tr>
<td>Address Spoofing Attack</td>
<td>Protected</td>
<td>protected</td>
<td>unprotected</td>
</tr>
<tr>
<td>False Address Conflict Attack</td>
<td>Protected</td>
<td>protected</td>
<td>Protected</td>
</tr>
<tr>
<td>Negative Reply Attack</td>
<td>Protected</td>
<td>protected</td>
<td>Protected</td>
</tr>
<tr>
<td>Denial of service Attack</td>
<td>Protected</td>
<td>protected</td>
<td>unprotected</td>
</tr>
</tbody>
</table>