Tracing Anonymous Mobile Attackers in Wireless Network

Ming Hour Yang[*1], Shiuhpyng Shieh[*2]
[*1]*Department of Information and Computer Engineering*
Chung Yuan Christian University
Chung Li City, Taoyuan County 32054, Taiwan
[*2]*Department of Computer Science & Information Engineering*
National Chiao Tung University
Hsinchu City 300, Taiwan
mhyang@cycu.edu.tw, ssp@csie.nctu.edu.tw

Abstract

In a flooding-based distributed denial-of-service (DDoS) attack, an adversary attempts to exhaust a target's computing resource. To detect DDoS attacks in a network environment, IP traceback methods are deployed to determine the origin of attack packets. With the increase in bandwidth of wireless networks, attackers may choose this medium from which to launch attacks. However, tracing the attackers in wireless networks is more difficult since intermediate nodes may move or attackers may change their location to hide themselves. Thus, conventional traceback schemes for wired networks cannot apply directly to the wireless network infrastructure because the intermediate routers could be compromised by an adversary. Therefore, we propose a flexible and lightweight traceback scheme to determine the source of attack packets in a mobile ad-hoc environment. We will demonstrate our method's ability to track the movement of an attacker and recognize attack traffic launched by the same attacker from different locations.

Keywords: *Wireless traceback, Distributed Denial of Service Attack, Mobile*

1. Introduction

With the decreasing cost of Internet access and its increasing availability from a plethora of devices and mediums, the impact of attacks on the network is becoming more significant and detrimental. In a distributed denial-of-service (DDoS) attack, an adversary comprises a large number of hosts and uses them as zombies to generate large traffic flows against a target host. By consuming the victim's resources, such as computational power, memory usage, and network bandwidth, the attacker can severely degrade or completely deny normal services [1]. Attack traffic may comprise of SYN packets, ICMP packets, or other [1]types of packets [2][3]. Nowadays, many well-known websites, such as Yahoo! and Amazon, are under the threat of DDoS attack, resulting in millions of dollars in damage.

The problem of DDoS stems from the nature of the IP protocol design [4]. Because the IP protocol routes packets by the destination address in the IP header without regard to the source address, an adversary may insert any address in the source address field. Depending upon the requested service, the victim may allocate resources for a host that is not expecting any service.

To combat DDoS attacks, RFC 2827 [5] proposed ingress filters to check that inbound traffic originates from a valid IP address of an authorized network. Researchers have also suggested various IP traceback methods, such as i-trace [6], packet logging [7][8] and packet marking [9][10][11], to detect the source of the attacks. In the i-trace approach, Bellovin utilizes the ICMP packet to probabilistically send traceback information. Unfortunately, routers frequently block ICMP messages

This work was supported in part by National Science Council under the grants NSC-98-2221-E-003-041-, ITRI, Institute for Information Industry, Chung Shan Institute of Science and Technology, Chunghwa Telecomm., Investigation Bureau, D-link, the International Collaboration for Advancing Security Technology (iCAST) and Taiwan Information Security Center (TWISC), respectively.

because of their security risks. Packet logging requires extra storage overhead in routers along the route path to store traceback data. In contrast, packet marking utilizes the 16-bit ID field of the IP header to save traceback information. Due to the limited amount of available bits in the header, packet marking schemes probabilistically mark packets with partial traceback information. Therefore, the victim must receive a sufficient amount of packets to accurately reconstruct the attack path. To succeed in the wireless environment, however, these schemes require modifications to adapt to the mobility of nodes.

Wireless networks consist of infrastructure mode and ad-hoc mode. In the infrastructure mode, the wireless nodes connect directly to the access point (AP), which acts as a mediator between the wired and wireless network. Conversely, nodes self-organize in the ad-hoc mode to create routing paths with neighboring nodes.

Depending upon how ad-hoc routing protocols maintain basic routing information, they can be divided into proactive routing and reactive routing. Proactive routing protocols periodically update the routing table even if there is no network traffic. In contrast, reactive routing protocols only create routing paths as needed. Dynamic Source Routing (DSR) [13] is a reactive routing protocol that keeps the full routing path in the header. Thus, intermediate nodes can forward packets with the header information.

With the growth of wireless device usage and the increase in wireless bandwidth, adversaries may opt to generate attacks from the wireless ad-hoc network. The ability to perform IP traceback in an ad-hoc network will become more important. Huang and Lee [14] developed a traceback scheme for ad-hoc networks based upon the SPIE scheme [7]. SPIE makes use of a special data structure, a bloom filter [15], to reduce the packet logging overhead on each router. Similarly, Huang and Lee's scheme utilizes a tagged bloom filter, which decreases the data overhead used to construct the link graph. With the link graph and neighbor list, they can detect and locate a hotspot.

In designing a traceback scheme, we want the ability to trace an attack packet to its source in the ad-hoc network. Since intermediate and malicious nodes within the wireless network can join, leave, or move around, we would like our scheme to be able to continue tracking the attacker. That is, we would not consider that node as a new attacker. To the best of our knowledge, this is the first paper that achieves these goals.

We organize the remainder of this paper as follows. In the next section we elaborate on the network architecture and attack model. Next, we describe our proposed scheme in Section 3 and evaluate its ability to detect attackers in Section 4. Finally, we conclude our findings in Section 5.

2. Preliminaries

Before we begin, we discuss our network architecture, assumptions, and threats. As depicted in Figure 1, we focus on the attack model in which an adversary within an ad-hoc network executes a DDoS attack against a remote server. When the server's Intrusion Detection System detects a DDoS attack, it initiates an IP traceback on the wired network to find the gateway AP. There exists numerous traceback methods [7][8][9][10][11] that can determine the attack route from the server to the end AP. To ferret out the adversary, the AP then performs a trace on the wireless network by utilizing our proposed scheme. Our method employs a combination of logging and packet marking, which it sends out of band (ICMP) to avoid interference with traditional wired traceback.

To access the internet in a wireless ad-hoc network, one node acts as an AP and forwards packets on behalf of member nodes. Thus, this node, which we refer to as the AP, provides a bridge between the wired and wireless domains. Our ad-hoc network implements the Dynamic Source Routing (DSR) protocol to route messages amongst member nodes.

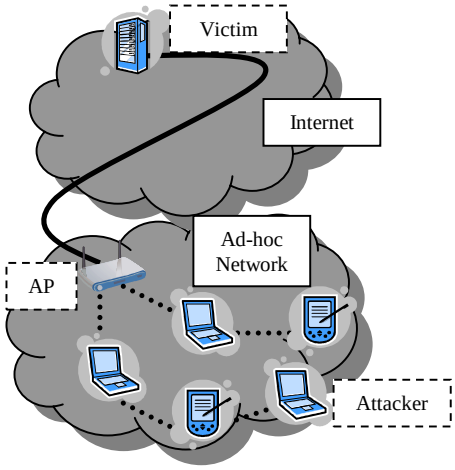

Figure 1. Architecture

.

2.1. Assumptions

We make the following assumptions:

The AP must possess enough computation, storage, and energy resources to perform logging and traceback services.

The AP cannot be compromised. Since the AP handles logging operations, an AP under the control of an adversary cannot provide useful traceback information.

Intermediate nodes within the ad-hoc network can be compromised.

Adversaries may change locations within the ad-hoc network or associate with other networks.

2.2. Threats

An adversary may attack the network with a variety of methods. In the simplest approach, the adversary captures a node and proceeds with a DDoS attack against the server. In addition, the adversary may insert randomly generated source IP addresses into attack packets. Furthermore, the attacker may relocate within ad-hoc network to alter the routing path.

3. Wireless IP traceback Scheme

We propose an IP traceback scheme (WiTrack) to locate mobile attackers in an ad-hoc network. First, we describe how to create a Route Graph History table. Next we show how an AP marks packets. Last, we detail the process of how we traceback mobile adversaries.

3.1. Routing Graph History

To pinpoint attackers in a wireless network, we need to deal with the mobility issues. When an AP receives a notice from a victim's IDS, the wireless network topology changes when a victim is under attack. Thus, we introduce Route Graph History (RGH) to log the changes of the network topology. With an AP's RGH table, we can have a snapshot of the network topology when malicious packets invade the wireless network. In the following paragraphs we will detail how to create a RGH table.

In a distributed system, time synchronization is vital in ensuring the transaction agreements and events between nodes. However, time synchronization of all nodes requires higher costs. To loosely

synchronize time in a distributed system, we propose a RECORD_ROUTE() scheme to collect the records and establish a virtual clock T_k that represents the k_{th} change of the network topology in an AP.

Algorithm RECORD_ROUTE() :

Assume the Current Timestamp at the AP is T_k

$P(N_i, LR_i) = \text{RGH}(N_i, LT_i)$

$CT = T_k$

IF $(P(N_i, LR_i) == \phi$ or $P(N_i, LR_i) \neq P(N_i, R_j))\{$

$CT = T_{k+1}\}$

IF$(\text{RGH}(N_i, CT) == \phi)\{$

$\text{RGH}(N_i, CT) = P(N_i, R_j)\}$

Our ad-hoc network model contains n nodes N_1, N_2,…, N_n with access to the internet through a gateway AP. $LR_i=\{LT_1, LT_2,\ldots,LT_n\}$ represents a list of last updated timestamps for node N_i. $P(N_i,R_j)$ symbolize the routing path starting from ith node N_i that goes through route R_j to the AP, where R_j consists of an ordered set of nodes. When the AP receives a packet originating from N_i at T_k, it checks whether the last updated path of node N_i differs from the new packet's path. If $P(N_i,R_j)$ changes or does not exist, then the AP increases the virtual clock T_k to T_{k+1} and records the new path $P(N_i,R_j)$ with the modified timestamp T_{k+1} in RGH(N_i, T_{k+1}), where RGH(N_i, T_{k+1}) means i-th column and $k+1$-th row in RGH table; otherwise, the clock maintains the same value and the AP adds the unchanged path with the current clock $P(N_i,R_j)$ to the RGH(N_i, T_k). Note that actual time intervals between timestamps are not necessarily the same. That is, timestamp changes only when a route changes.

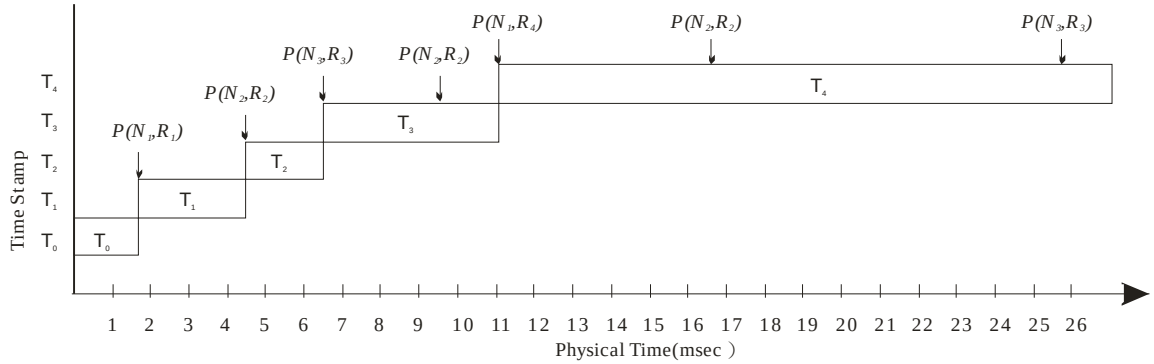

Figure 2. Elapse of Timestamp on an AP

Figure 2 shows an example of maintaining timestamp on an AP. Initially, timestamp is set to T_0. As Figure 3(a) shows, when AP receives the first packet originating from N_1 *at* 2msec, it increases the timestamp to T_1 and records the path $P(N_1,R_1)=\{N_1, N_8, N_5, N_4\}$on RGH table (First row of Table 1). Since the node N_1 did not sends any packets to this AP before receiving packet $P(N_1,R_1)$. As Figure 3(b) shows, the AP receives second packet $P(N_2,R_1)$ at 4.5msec. Although the route from N_2 is same with the previous route from N_1, the AP needs to increase the virtual clock to T_2 and records the path $P(N_2,R_1)=\{N_2, N_8, N_5, N_4\}$ on RGH table. Since the AP's RGH does not contain a previous route for N_2. It follows the same procedure for the third packet from node N_3 respectively.

For the fourth packet from node N_2, the AP determines the path of this packet is equivalent to the last updated path $P(N_2,R_2)$. Thus, it does not increase the virtual clock at T_3; however, the AP must log this path $P(N_2,R_2)$ since this route does not exist in the RGH table at time T_3. As Figure 3(d) shows, when the AP receives the fifth packet from N_1, it increases the timestamp to T_4 because the last updated route of node N_1, i.e. $R_1=\{N_8, N_5, N_4\}$, is different

from route $R_4=\{N_5, N_4\}$, and records the path $P(N_1,R_4)$ on RGH table. Finally, the log of packets received by this AP is illustrated in Table 1.

Table 1. Example RGH table

Time \ Node	N_1	N_2	N_3
T_1	$P(N_1,R_1)$		
T_2		$P(N_2,R_2)$	
T_3		$P(N_2,R_2)$	$P(N_3,R_3)$
T_4	$P(N_1,R_4)$	$P(N_2,R_2)$	$P(N_3,R_3)$

We define the graph G_i as the set of all routing paths during time T_i. Thus, $G_i = \{ RGH(N_1,T_i) \cup RGH(N_2,T_i) \cup \ldots \cup RGH(N_n,T_i) \}$ is the network topology at time T_i. Figure 3 shows examples of network topology constructed by using the records of RGH table. On the third row of the table, two disjoint paths are recorded at T_i. And the network topology is drawn accordingly as Figure 3(c) shows. At T_4, node N_1 follows different route to send a packet to the AP, and nodes N_2 and N_3 follow identical routes as before to route packet to the AP. Then, the network topology on the AP at T_4 will look like Figure 3(d).

Initially, the network will experience more updates and changes to the global clock as the AP populates the empty RGH table with routing paths. When the network stabilizes, there should be fewer updates. Nodes whose routing paths remain null for an extended period of time may have left the network. During an attack, however, the AP will have to go through many updates as the adversary moves on the network or spoofs many source routes. If we try to trace DDoS attackers from victim to AP, we need to mark the packet on an AP.

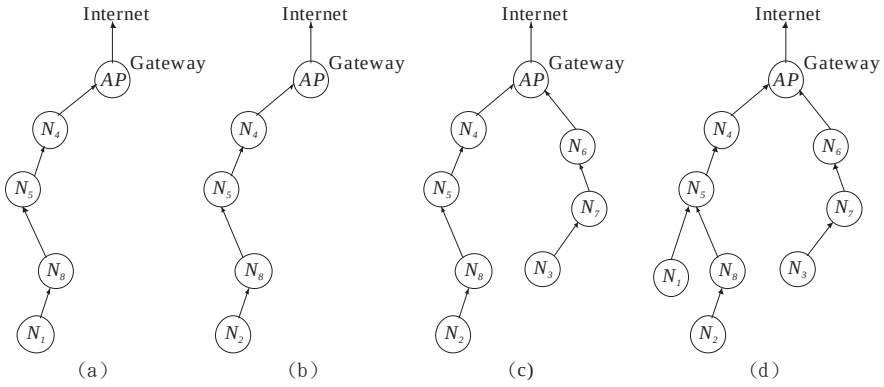

Figure 3. History Network Topology in an AP
(a) At timestamp T_1 (b) At timestamp T_2 (c) At timestamp T_3 (d) At timestamp T_4

3.2. Packet Marking

Since the AP extracts the source route from each packet's DSR header, it has a complete view of the ad-hoc network's topology. However, forwarding a complete source route to the server during normal operations is inefficient. To reduce network bandwidth, the AP generates a hash to mark a packet. The AP sends the packet's marked data (MD) to the server with the original packet out of band. The reason we send the MD out of band is to prevent the MD from being erased by wired traceback marks.

MD relates a packet with its routing path and timestamp. It contains the packet's timestamp, a hash of its routing path, and a hash of the packet's payload. That is,

$\mathrm{MD} = T_j \,|\, \mathrm{HASH}(\, RGH(N_i,T_j)) \,|\, \mathrm{HASH}(\, Payload\,)$, where | is a stream concatenation operator.

The mark includes three parts and the first is a timestamp, which helps the receiver know when the AP receives this packet. The second is the hash value of the wireless routing path, helping the victim trace from the AP to the

attacker. The third part of the packet includes its hash value. Since the mark is sent out of band, the hash value is used to connect attack packet and the mark. As we assume the attack to be DDoS, the false positive can be eliminated as more and more packets come in. We, therefore, recommend the digest of packet use the packet header and some bytes of payload as the input of a universal hash table.

When the server receives the MD, it should retain the information for a limited amount of time. For immediate look-up, the server puts the MD in a hash table, using the payload hash as the index, and the timestamp and routing path hash as the value. Once detecting a DDoS attack, the server returns the MD to the AP, so that it can hunt down the attacker accordingly.

3.3. Mobile Attacker Traceback Scheme

When an adversary initiates a DoS attack on a server from wireless ad-hoc network, the adversary may insert randomly generated source IP addresses into the attack packets and then relocate within ad-hoc network to alter the routing path. Therefore, the attack traffic that comes from different places could be launched by the same attacker. If a tracing algorithm cannot recognize the attack traffic which probably comes from the same attacker, a number of "attackers" will be recorded on a server. Also, it will be difficult for a victim to identify the current location of an attacker. However, the conventional tracing algorithms that are designed for wired network cannot trace mobile attackers. Hence, an algorithm MERGE_PATH is introduced to trace mobile attackers and identify two attack traffic that comes from two different places but might be launched by same attacker.

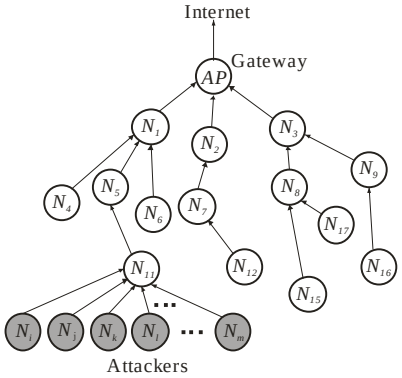

Figure 4. Attack graph of Spoofed IP address attack. Gray nodes are attack nodes notified by victims

When the AP receives an attack notice, it utilizes the algorithm MERGE_PATH() to verify whether the attacker is new or has just simply moved; therefore, it can reduce the number of suspected attackers because some of them do come from the same source. After receiving an attack notice, an AP extracts from the notice the MD with $P(N_i,R_j)$ *and* T_k, and checks if it matches the entry in its RGH table. Since an adversary may spoof the address of the last hop along the wired network, a mismatch is possible. As a result, the AP simply discards this attack notice. On the contrary, if the MD has ever appeared in the RGH table, it means the server has successfully located the originating AP. Therefore, the AP commences a search for the attacker within its network. When the AP tracks down the attacker, it adds this node to the Attacker List (AL), a log of all malicious nodes at each interval. The AP maintains and forwards the list to the server. $AL=\{AL_1, AL_2, AL_3, \ldots, AL_n\}$, where $AL_i=(N_m,T_n)$ is *i*-th attack node and timestamp received and recorded in the AP. The notation $AL_i\_N= N_m$ represents the attack node of the record AL_i, and $AL_i\_T= T_n$ represents the timestamp of the record AL_i. Furthermore, the MERGE_PATH() merges attackers by tracing movement in the attack graph. An attack graph at time T_k, denoted AG_k, represents a partial graph of G_k. That is, $AG_k = \{ RGH(N_1,T_k), RGH(N_2,T_k), \ldots, RGH(N_n,T_k) \}$, where $N_n \in \mathrm{AL}$.

The constructed graph AG_j is loop free because DSR is a loop free routing algorithm. If a spoofing tool inserts a randomly generated source IP address into attack packets, as depicted in Figure 4, lots of attack nodes route their traffic through the same intermediate node. The proposed scheme MERGE_PATH() merges any two malicious nodes N_i from time T_m and N_j from time T_n if the two nodes have the same parent in a tree structure, so as to verify

whether the attacks are launched by the same adversary. Three attack models and their tracing examples will be detailed below.

If an adversary does not spoof the source address of its attack packets, the AP can trace the adversary by following the attack graph. Even though the attacker changes locations, he must execute DSR's route discovery to find a valid path to the AP. Since any change of routes from the attack source is recorded on the RGH table, attackers can be located because of attack graph tracing. As previous attacking notices are generated for the same attacker, the AP can remove all those notices from AL to reduce the list's size. For example, as Figure 5(a) shows, when the AP receives a new attack notice from a victim (Figure 5(b)), we merge the original attack graph AG with the new attack path into a new attack graph. Since the moving adversary is attacking a victim, the merged attack graph, see Figure 5(c), shows the attack traffic is launched by the source node N_{11}, which moves from its neighboring node N_{10} to node N_5. Then, the old N_{11} is no longer needed and will be removed from the attack graph. After AP removing the duplicated nodes, the attack graph is shown as Figure 5(d).

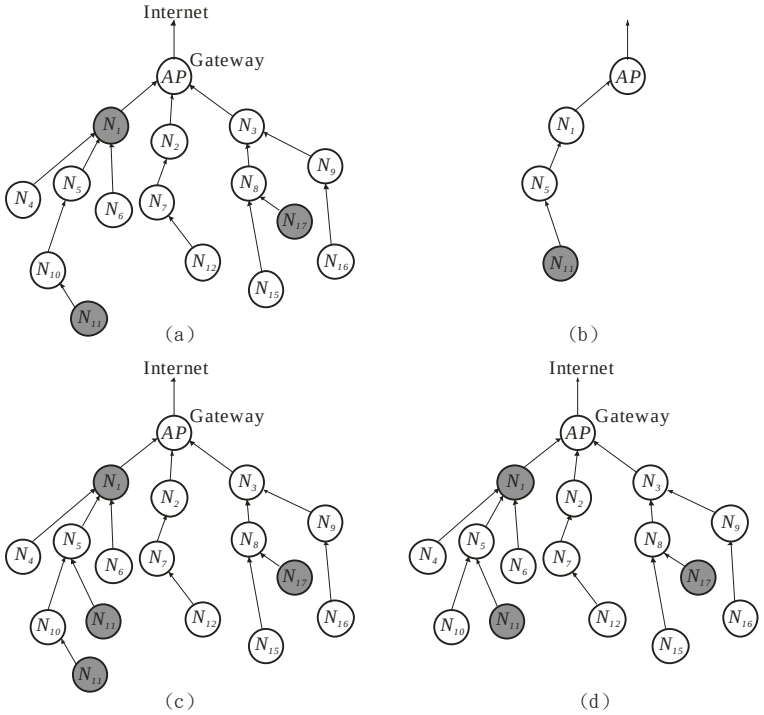

Figure 5. Example of merged attack graphs, the gray nodes are identified attackers. (a) The attack graph before new attack notice. (b) AP receiving an attack notice. (c) The attack graph after the attack notice. (d) Attack graph after the merging of two attackers N_{11}.

As for those attackers who spoof their source addresses, the AP must generate an attack graph (AG) after getting an attack notice because attack traffic that comes from different sources may be launched by the same attacker. AG consists of the merged set of all MD that the AP receives, i.e. $AG = \{ AG_1, AG_2, ..., AG_n \}$. Since the routing path may change as intermediate nodes join, move, or leave the network, a routing loop may happen when AP merging attack graphs from different time intervals. If there is a loop in an attack graph, a tracing algorithm may fall into the loop and make it difficult for the algorithm to trace the source of the attack traffic. In order to remove a loop from a graph, a path has to be removed from the graph to keep the graph loop free. We will define the to-be-removed path in the following.

We assume two attack graphs AG_i at T_i and AG_j at T_j, where $T_i<T_j$. If a sequence of nodes $\{N_s, N_p, N_q, \ldots, N_r, N_e\}$ exists in the graph $AG=AG_i \cup AG_j$, where nodes N_s and N_e exist in the attack graphs AG_i and AG_j.

$SubP_i(AG_i,AG_j)=\{N_p, N_q,\ldots, N_r\}$ is a set of nodes in AG_i, not in AG_j. $SubP_j(AG_i,AG_j)=\{N_p, N_q,\ldots, N_r\}$ is a set of nodes not in AG_i but in AG_j. The nodes $\{N_s, N_p, N_q,\ldots, N_r, N_e\}$ will be a loop in the merged graph AG.

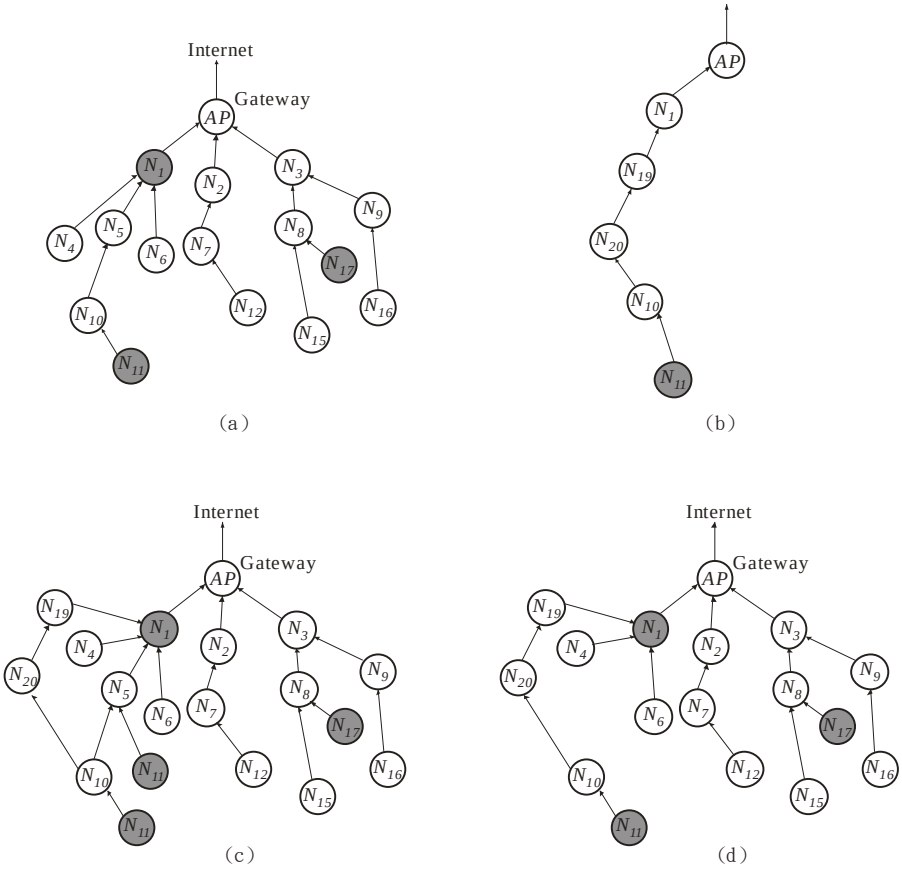

Figure 6. Example of removing loops from an attack graph. (a) The attack graph before new attack notice. (b) AP receiving an attack notice. (c) The attack graph after the merging of two attack graphs. (d) The attack graph after AP removing a loop from the graph.

Since the traffic moves from node N_e to node N_s, it will not choose the route $SubP_i(AG_i,AG_j)$ at T_j; instead, it uses the alternative path $SubP_j(AG_i,AG_j)$ to route packets. Therefore, MERGE_PATH() removes the path $SubP_i(AG_i,AG_j)$ from the merged graph AG so as to remove the loop from the graph $AG_i \cup AG_j$. Thus, the MERGE_PATH() traces an attacker from the root of a tree to leaf nodes to locate the attacker. Figure 6 exemplifies the removing of a loop from a merged attack graph. When an AP receives an attack notice, its rout is shown in Figure 6(b). Because the moving of intermediate nodes, the routing path will begin to change from the last transmission. We find the merged graph has a loop $\{N_1, N_{19}, N_{20}, N_{10}, N_5\}$ as Figure 6(c) shows. Also, the nodes that send messages through node N_{10} do not follow the routing sequence (N_{10},N_5,N_1). Instead, the message will follow the route $(N_{10},N_{20},N_{19},N_1)$. Because the packets do not route through nodes (N_{10},N_5,N_1), we can remove the unwanted path (N_{10},N_5,N_1) from the graph. Besides, the tracing algorithm can trace the attacker on the new loop-free attack graph.

Algorithm MERGE_PATH :

Let N_m be an attacking node sent from a victim

IF $(\exists(N_m, T_k) \in RGH)$ {

IF AL $= \phi$ {

$AL = AL + (N_m, T_k)$

$AG_k = P(N_m, R_j)$

NumberOfAttackers = 1}

NumberOfAttackers = NumberOfAttackers + 1

$Diff_k = P(N_m, R_n) - (P(N_m, R_n) \cap AG) - SubP_j(AG, P(N_m, R_n))$

IF $(N_m, T_k) \in AL$

NumberOfAttackers = NumberOfAttackers − 1

ELSE {

While (i ≤ |AL| and no match){

IF $((AL_i \_ N, T_k) \notin AL$ and (

$((|Diff_k| == 1)$ and $(AL_i \_ N\text{->parent} == N_m\text{->parent}))$ or

$((|Diff_k| == 1)$ and $(AL_i \_ N\text{->parent->parent} == N_m\text{->parent}))$ or

$((|Diff_k| == 2)$ and $(AL_i \_ N\text{->parent->parent} == N_m\text{->parent->parent}))))${

NumberOfAttackers = NumberOfAttackers − 1

$AL = AL - AL_i\}\}$

$AL = AL + (N_m, T_k)\}$

$AG_k = AG_k \cup P(N_i, R_j) - SubP_i(AG_i P(N_i, R_j))$

} discard the notice

Since the nodes in an ad-hoc network can be compromised by an adversary, these compromised nodes will be stepping stone for attackers. Sometimes an attacker compromises a host in the middle of its routing path and uses the compromised nodes to remove all tracing clues from the transmitted packets. When MERGE_PATH() traces the source of attack traffic, it will identify the attack node as malicious users' stepping stone, and the attack node will not be a leaf of the attack graph. If the routing path from the real attacker to the stepping stone is steady, the attacker can hide himself behind the stepping stone and our scheme cannot locate him. Then, he will be logged on the AP and identified as another attacker, i.e. his stepping stone actually, in the attack graph. N_1 in Figure 6(a) exemplifies a stepping stone in an attack graph.

In the previous Mobile Attacker Tracing Scheme, we focus on single access point traceback. However, it is possible for a mobile attacker to roam from one AP to another. When an attacker roams between two APs, they should exchange their RGH records with each other to maintain the correctness of packet transmission, like registration information in Mobile IP. That is to say the AP is able to know the address of the roaming nodes. When an attacker tries to generate attack packets from one AP to another, the two APs' RGH should be combined to trace the attacker.

For instance, when an attacker roams from AP1 to AP2, the victim can get RGH_{AP1} and RGH_{AP2} from the traced AP. However, the problem of distributed RGH is that the two RGHs are not synchronized. That is to say, when RGHs are not synchronized, the only way to merge two attackers is to scan all the tables for known attackers and time slices.

Hence, when we exchange RGH_{AP1} and RGH_{AP2}, the two APs can synchronize the two RGHs by matching the two tables' T_{max}. With regular exchange of RGH_{AP1} and RGH_{AP2}, we can create the mapping of T_{max} for AP1 and AP2 respectively. Then, we can limit the scan size of the table and synchronize the two RGHs. Figure 7 illustrates how two RGHs keep their own counters and synchronize each other.

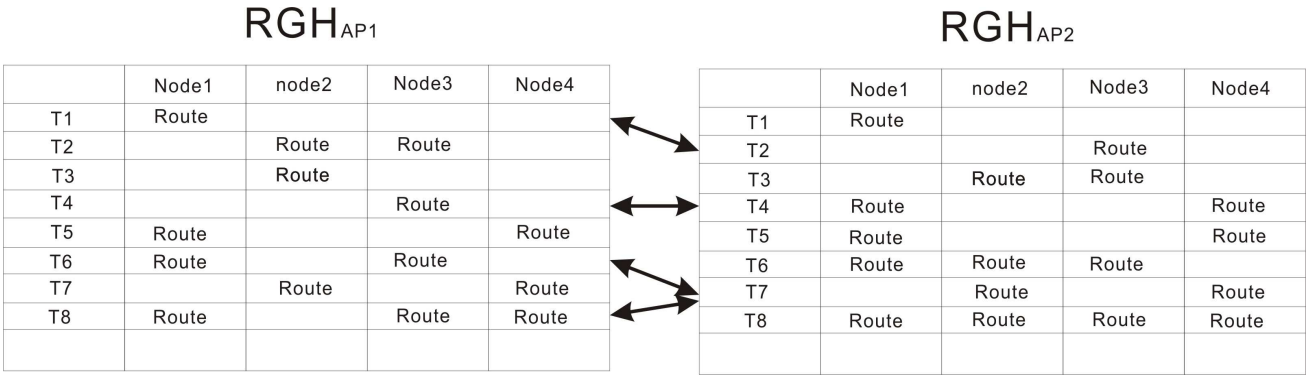

Figure 7. Two Route Graph History and synchronization

To trace an attacker, we can use the original approach, but the scan range will have to extend from a table to two and the timestamps for the two tables need to be remapped. For instance, as illustrated in Figure 7, after the remapping, we find: T_1 on RGH_{AP1} ↔ T_2 on RGH_{AP2}; T_4 on RGH_{AP1} ↔ T_4 on RGH_{AP2}; T_6 and T_8 on RGH_{AP1} ↔ T_7 on RGH_{AP2}.

4. Simulation

We use the ns2 simulator [17] to test our scheme's ability in locating an adversary. In general, we setup the simulations with the default wireless environment configuration for channel, propagation, link layer type, interface queue type, Media Access Control, and antenna. For the DSR agent, we configure the flow state and send buffer. Next, we center the AP in our 600 m^2 test grid and enable it to forward packets to the Internet. Then, we generate 50 mobile nodes using the ns2's indep-utils/cmu-scen-gen/setdest and log DSR packets with CMUTrace. Every node is able to send valid UDP packets to AP. Malicious nodes may move through the network at a faster but reasonable rate.

Intermediate nodes may move along a routing path. In the first simulation, we demonstrate our scheme's ability to identify a mobile DoS attack by merging attack graphs even though intermediate nodes moves during attacks. In Figure 8, we find that the network size does not affect the accuracy of our scheme when we verify that the attack traffic sent from different locations is, in fact, generated from the same source. Then, we identify those attacks as one and merge them into one. By considering network of up to 50 nodes, we are able to merge 90% of all attack traffic launched by the same attackers.

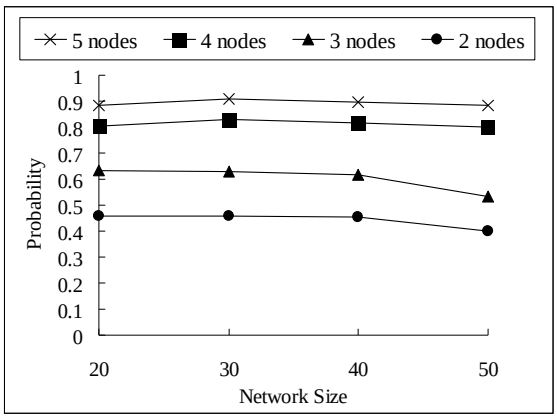

Figure 8. Effect of network size

We find that examining a larger number of moved intermediate nodes yields higher merging rate. Since the moving of intermediate nodes make the change of network topology, it provides more information for us to analyze the movement of an attacker.

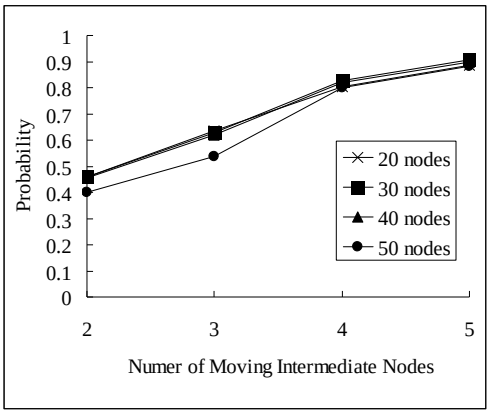

Figure 9. Effect of moving routers

Next, we will prove that our scheme is able to detect spoofed source IP addresses. As illustrated in Figure 10, we find that our scheme can successfully trace over 99% of all randomly generated source IP addresses.

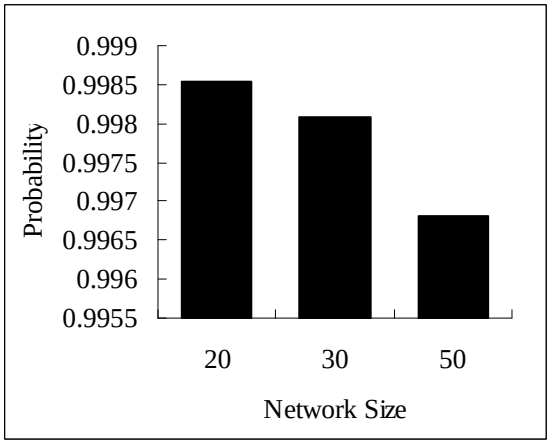

Figure 10. Spoofed IP address detection rate

Finally, we test if our scheme yields the correct number of attackers between time intervals. That is, our scheme will not double count a moving adversary even though it tries to spoof its source address. Figure 11 illustrates that our scheme can successfully hunt down adversaries with 99% accuracy.

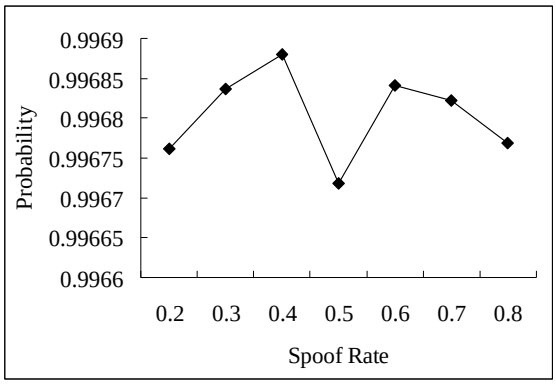

Figure 11. True positive rate

5. Conclusion

In this paper, we introduce a traceback scheme for wireless networks running Dynamic Source Routing. Compared to existing traceback schemes, our method can trace an adversary to its source, even if he changes locations. Furthermore, our scheme can determine if a set of attack packets originate from the adversary. Our simulations show that our scheme can detect duplicate attackers with 99% effectiveness.

References

[1] Vern Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," ACM Comp. Comm. Review, vol. 31, no. 3, 2001.

[2] David Moore, GeoRrey M. Voelker, and Stefan Savage, "Inferring internet Denial-of-Service activity," in In proceedings of the 2001 USENIX Security Symposium, 2001, pp. 9-22.

[3] Microsoft Corporation, "Stop 0A in tcpip.sys when receiving out of band(OOB) data," http://support.microsoft.com/support/kb/articles/Q143/4/78.asp.

[4] J. Postel, "Ineternet Protocol," RFC 791, IETF, Spe 1981.

[5] Paul Ferguson and Daniel Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing," RFC 2827, IETF, May 2000.

[6] Steven M. Bellovin, "Icmp traceback messages," IETF Draft, 2000, http://www.research.att.com/smb/papers/draft-bellovin-itrace-00.txt.

[7] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayers, "Hash-Based IP Traceback," in ACM SIGCOMM '01, August 2001.

[8] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayers, "Single-Packet IP Traceback," IEEE/ACM Transactions on Networking, vol. 10, no. 6, pp. 721-734, December 2002.

[9] Stefan Savage, DavidWetherall, Anna Karlin, and Tom Anderson, "Network Support for IP Traceback," IEEE/ACM Transactions on Networking, vol. 9, no. 3, pp. 226-237, June 2001.

[10] Hal Burch and Bill Cheswick, "Tracing anonymous packets to their approximate source," In Proc. USENIX LISA '00, December 2000.

[11] D. X. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback Messages," in Proc. IEEE Infocom '01, April 2001.

[12] C. Perkins, "Ad hoc on demand distance vector (aodv) routing," 1997.

[13] David B. Johnson, David A. Maltz, and Yih-Chun Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," IETF Draft, 2004, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt.

[14] Yi and Huang and Wenke Lee, "Hotspot-based traceback for mobile ad hoc networks," in WiSe '05: Proceedings of the 4th ACM workshop on Wireless security, New York, NY, USA, 2005, pp. 43-54, ACM Press.
[15] Burton H. Bloom, "Space/Time Trade-ORs in Hash Coding with Allowable Errors," Communication of ACM, vol. 13, no. 7, pp. 422-426, July 1970.
[16] J. Reynolds J. Postel, "TELNET PROTOCOL SPECIFICATION," RFC 854, IETF, May 1983.
[17] Steven McCanne and Sally Floyd, "ns Network Simulator," http://www.isi.edu/nsnam/ns/