

Secure Ad Hoc Networks Against Injecting Traffic Attacks

N.Krishnaiah

Asst.Professor

B.V.C Engg college, Odalarevu, A.P

Email: nkrishna520@gmail.com

B.S N Murthy

Asst.prof in CSE

B.V.C.Engg College, Odalarevu

E-mail: murthy_asst@yahoo.com

Abstract: While wireless communication has dramatically changed the way people work and interact, the wireless era continues to be plagued by insufficient security. The inherent cooperative nature of these networks makes them extremely vulnerable to insider attacks. Without necessary counter measures, even a few attackers can break down the whole network. This paper presents the system model and investigations regarding the possible types of injecting traffic attacks and ways of securing these networks. We presented two methods for security. First method is a set of fully distributed defense mechanism which can effectively detect injecting data packet attacks and the other, a centralized defense mechanism with de-centralized implementation which has an added advantage of reduced overhead.

1. INTRODUCTION:

A wireless ad hoc network is a group of nodes without requiring centralized administration or fixed network infrastructure, in which nodes can communicate with other nodes out of their direct transmission ranges by cooperatively forwarding packets for each other through wireless connections.

Coming to the threats posed to these networks we have routing disruption attacks, injecting traffic attacks, black holes, routing table overflow, sleep deprivation and many others. We are focusing in this paper on one of the powerful attacks: Injecting traffic attacks. Specifically, attackers inject an overwhelming amount of traffic into the network in attempt to consume valuable network resources, and consequently degrade the network performance. Since in ad hoc networks, nodes need to cooperatively forward packets for other nodes, such networks are extremely vulnerable to injecting traffic attacks, especially those launched by insider attackers.

Cooperative ad hoc networks, where nodes are classified into two types: good and malicious. We focus on the scenario that nodes use omnidirectional transmission techniques, such as omnidirectional antennas. However, attackers are allowed to use directional transmission techniques, such as directional antennas or adaptive beam forming, to improve their attacking capabilities. According to the system goal, each

node may schedule to generate and send a sequence of packets to certain destinations. We call a source-destination pair to $T_{s,d}(t)$ to be legitimate if this pair is needed to achieve the system goal. For each legitimate source-destination pair (s,d) in the network, we assume that the number of packets that is scheduled to inject by this pair until time t is $T_{s,d}(t)$. In general, the exact value of $T_{s,d}(t)$ may not be known a priori by other nodes in the network. We assume that a loose upper-bound of $T_{s,d}(t)$, denoted by $f_{s,d}(t)$, can be estimated *traffic injection upper-bounds associated to pair (s, d)* .

II. TYPES OF ATTACKS:

As mentioned before, our focus is on defending against injecting Traffic attacks. Injecting traffic attacks can be classified into two types: query-flooding attack and *injecting data packets attack (IDPA)*.

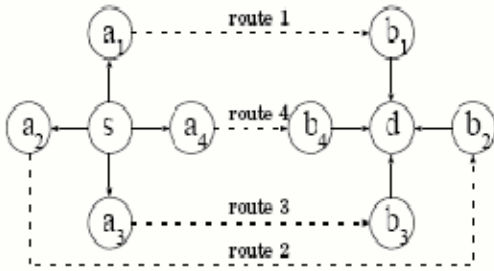
Due to the changing topology or traffic pattern, nodes in ad hoc networks may need to frequently perform route updates, which may require broadcasting query messages. Then attackers can broadcast query message with a very high frequency to consume valuable network resources. Such attack is called as query-flooding attack

Besides query-flooding attacks, attackers can also inject an overwhelming amount of data packets into the network to request other nodes to forward. When other nodes process and forward these packets, their resources (e.g., energy) are wasted. We call such attack as injecting data packets attack (IDPA). Since in general the size of data packet is much larger than the size of query messages, and the injection rate of data packets is much higher than the injection rate of query message, the damage that can be caused by injecting data packets attack is usually more severe than by query-flooding attacks

We first consider the possible ways that IDPA can be launched by attacker's s and d with s being the source and d being the destination.

The simplest way, which is called simple IDPA, is that s picks a route R to d and injects an overwhelming amount of packets into the network, which is much higher than the legitimate bound $f_{s,d}(t)$.

In the second way, which is called long route IDPA, s



picks a very long route to inject data packets into the network. For example, in the fig1, s can pick the route Swcbahefgd to send packets from s to d with the number of injected packets conforming to the legitimate bound $f_{s,d}(t)$. By doing this way, s and d can achieve the same effect as increasing its traffic injecting rate.

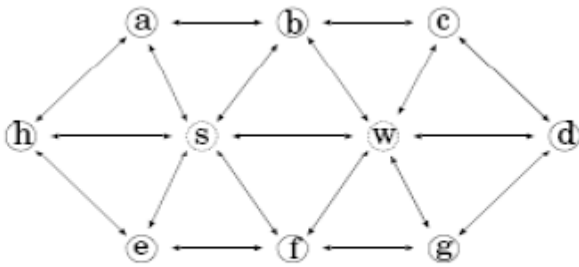


Figure 1: An example of long route attack

In the third and advanced way, which is called multiple routes IDPA, s picks multiple routes to d and simultaneously injects traffic into the network via these routes. For example, as shown in fig2, s uses four routes Sa1 b1d", Sa2 . . . b2d", Sa3 ... b3d and Sa4 b4d_ to inject packets into the network. By doing in this way, the traffic can be distributed among multiple routes such that for each route the packet injection rate conforms to the legitimate bound $f_{s,d}(t)$ though the total number of injected packets can be much higher than $f_{s,d}(t)$. Moreover, the attackers can also take advantage of advanced transmission techniques, such as directional antenna and beam forming, to avoid being detected

Figure 2: An example of multiple route attack

III Defense Mechanisms

In order to detect whether a node has launched injecting traffic attacks, we have to base on the observed behavior of that node. In other words, a node can be marked as launching injecting traffic attacks only if it has been observed by some other nodes that it has injected

too much traffic (higher than the legitimate bound), or it has sent traffic to illegitimate destinations. Therefore, the following mechanisms will be required by the defense system:

- A packet delivery mechanism where for any packet injected by node x, x cannot deny that this packet is from it and no other nodes can generate the same packet without colluding with x.
- A traffic monitoring mechanism to count the number of packets injected by each node in the network.
- A detection mechanism to detect injecting traffic attacks based on observed information.

A. Route Discovery and Packet Delivery

We adopt DSR (Dynamic Source Routing) as the underlying routing protocol to perform route discovery. Meanwhile, to defend against possible attacks, the following security enhancements will also be incorporated into the DSR protocol:

- When node s initiates a route discovery to destination d, besides the source destination pair, the query packet should also include a unique ID associated to this query and the sequence number corresponding to the last data packet that s has sent to d

The following format is used for each query packet:

$$\{s, d, ids(s, d), seq_x(s, d), sign_x(s, d, ids(s, d), seq_x(s, d))\}$$

Here $ids(s, d)$ is the sequence number of this query packet, which has an initial value of 1 and is required to be increased by 1 after each query has been issued by the pair (s, d). $seq_x(s, d)$ is the sequence number of the last packet that has been injected into the network by the pair (s, d). $Sign_x(s, d, ids(s, d), seq_x(s, d))$ is the signature generated by s based on the message $\{s, d, ids(s, d), seq_x(s, d)\}$.

When a good node x receives a route request packet with s being the source and d being the destination, x first checks whether the following conditions can be satisfied:

1. The source-destination pair (s, d) is legitimate;
2. All signatures are valid;
3. $idx(s, d) < ids(s, d)$, where $idx(s, d)$ is the maximum query sequence number corresponding to the pair (s, d) that x has seen before;
4. $seq_x(s, d) \leq seq_s(s, d)$, where $seq_x(s, d)$ is the maximum data packet sequence number corresponding to the pair (s, d) that x has seen before;
5. No nodes appended to the request packet have been detected as malicious by x;
6. less than T maxhop relay nodes have been appended to the query packet, where T maxhop is a system-level parameter indicating the maximum number of relays that a route can have.
7. x has not forwarded any request for the pair (s, d) in the last $T_x(s, d)$ interval, where $T_x(s, d)$ is the minimum query forwarding interval specified by x to indicate that x will

If all of the above conditions can be satisfied, we call such a request as a valid request. In this situation, x will assign the value of $ids(s, d)$ to $idx(s, d)$, assign the value of *the new query*. If only the conditions from 1 to 4 are satisfied, x will only assign the value of $ids(s, d)$ to $idx(s, d)$, assign the value of $seqs(s, d)$ to $seqx(s, d)$. In all other situations, x will discard this route request, and perform necessary attacker detection. Assume request is the received valid query message that x has decided to forward, and then the following format will be used for x to append its own address:

$\{(request, x, signx(request, x))\}$.

Once a source has decided to send a packet to a certain destination using a certain route, a data packet delivery transaction should be started. The proposed data packet delivery mechanism works as follows. Suppose that node s is to send a packet with payload msg and sequence number $seqs(s, d)$ to destination d through the route R . s first generates two signatures $sigh$ and $sigb$, with $sigh$ being generated based on the message $\{R, seqs(s, d)\}$ and $sigb$ being generated based on the message $\{R, seqs(s, d), MD(msg)\}$ where $MD()$ is a digest function. The format of the packet to be sent is as follows: $\{R, seqs(s, d), sigh, msg, sigb\}$

We refer to $\{R, seqs(s,d),sigh\}$ as the header of the packet, and refer to $\{msg,sigb\}$ as the body of the packet. Next, s transmits this packet to the next node on route R , and is required to increase the value of $seqs(s, d)$ by 1. When a node (e.g., x) detects that a certain packet is to be transmitted by a certain node (e.g., y), 'a' first decodes and checks the header of the packet. Assume $\{R, seqs(s, d), sigh\}$ is the header of the transmitted packet, 'a' needs to continue receiving and decoding the body of the packet only if all of the following conditions are satisfied:

1. the signature $sigh$ is valid;
2. x is on the route R and is the target of this transmission;
3. no nodes on route R has been detected as malicious by x
4. $seqs(s, d) > seqx(s, d)$;
5. route R has no more than $Tmaxhop$ relays;
6. x has agreed to participate on this route before and the route has not expired, where each route will be set an expiration time.

If all of the above conditions are satisfied, x will continue receiving and decoding the body of the packet, assuming it is $\{msg, sigb\}$. If the signature $sigb$ is valid, x will forward the packet to the next node on the route, and assign the value of $seqs(s, d)$ to $seqx(s, d)$.

B. Traffic Monitoring

Traffic monitoring is an indispensable component to detect possible injecting traffic attacks. In this paper we present a method where each node will keep monitoring its neighbors' transmission activities using the proposed

header watcher mechanism. Specifically, when a node x detects that a neighbor y is transmitting a data packet, no matter whether x is the receiver of this transmission or not, x will try to receive and decode the packet header sent by y . Actually this is needed by most wireless networks: without decoding the header, how can a node know whether this packet target at it or not?

The mechanism lays in that each node will also check the validity of the signature for the packet header. If the signature of the packet header is valid, x will put the packet header into the set $List(s, d, x)$ in x 's records, which will be used later to detect whether s has launched injecting traffic attacks.

Unlike the "atcdog" mechanism which requires a node to buffer all the packets that it has sent or forwarded and to keep monitoring its neighbors' transmission activities in order to check whether those packets have been forwarded by them, the header watcher_ mechanism proposed in this paper only requires a node to monitor the packet headers around its neighborhood. Since only packet header needs to be received and decoded, and the header of a packet is much shorter than the body of a packet, a lot of effort can be saved comparing to the watchdog mechanism which requires receiving, decoding, and comparing the whole packets

In general, if all packet headers received by node x are recorded, with the increase of x 's staying time in the network, more and more storage will be required. Actually, in our scheme, for each legitimate source-destination pair (s, d) , only those packet headers received after the last valid route request issued by (s, d) need to be recorded by x ; in another words, only those packet headers whose sequence numbers are larger than the sequence number broadcast by s in its last valid route request packet. With this modification, the storage requirement become very small and does not increase over x 's staying time in the network. In Section 6, we will also show how to further decrease the storage requirement.

C. Injecting Traffic Attack Detection

Here we take that each good node in the network will perform injecting traffic attack detection based on the observed behaviors. Specifically, for each source-destination pair (s,d) with $List(s, d,x)$ being non-empty in good node x 's records, the following detection rules will be used by x to check whether s has launched injecting traffic attacks.

- Rule 1: If $List(s, d,x)$ is not empty and the source-destination pair (s, d) is illegitimate, x will mark s as malicious.
- Rule 2: x received a request issued by an illegitimate source-destination (s, d) , x will mark s as malicious.

- Rule 3: For any packet header $\{R, seqs(s, d), sigh\}$ which belongs to $List(s, d, x)$, if route R has more than T_{maxhop} relays, x will mark s as malicious.
- Rule 4: If x detects that there exist two valid packet headers $\{R, seqs(s, d), sigh\}$ and $\{R0, seq0s(s, d), sig0h\}$ in the set $List(s, d, x)$ with $seqs(s, d) = seq0s(s, d)$ but R not equal to $R0$, x will mark s as malicious.
- Rule 5: Let $seqmax(s, d)$ be the maximum possible sequence number corresponding to the source-destination pair (s, d) at time t , that is, there exists a sequence number $seqs(s, d)$ in

The first two rules imply that only legitimate source-destination pairs can inject packets into the network. Rule 3 implies that no routes should have more than T_{maxhop} relays. Rule 4 handles multiple route attack. Rule 5 handles attackers which inject more packets than they should. In summary, rule 4 and 5 are used to prevent attackers from injecting more packets than they are

number. That is, no any two packets for the same trafficpair should have the same sequence number, and the sequence number has to be increased monotonically

Once x detects that s is launching injecting traffic attacks, x will also inform the other nodes in the network by broadcasting an ALERT message which includes evidence such as the corresponding packet headers. When other good nodes have received the ALERT message, after necessary verification (i.e., signatures are valid), they will also mark s as malicious.

Next we analyze the effects of possible impersonationattacks that can be launched by attackers. In the proposed mechanism, the only way that an attacker m can impersonate a good node s whose has not been compromised is to first record the packets that s has transmitted, then later forwards/broadcasts these packets. Specifically, there are two situations:

Situation 1: m recorded a query packet issued by s and rebroadcast it later.

However, since this query packet has been seen by all other nodes in the network due to the flooding nature of query message, no nodes will further process this query packet.

Situation 2: m recorded a data packet issued by s and forwarded it later.

However, since nodes on the route associated to this data packet will only process this packet at most one time, forwarding this packet at time $t1$ by m cannot cause damage to other nodes.

In summary, impersonation attack cannot cause further damage to good nodes in the network. Furthermore, it can

be readily checked that as long as s is good and has not been compromised, the probability that x will mark s as

malicious is 0. That is, the false alarm ratio of the above detection rules is 0.

IV. OVERHEAD ANALYSIS

Now we analyze the overhead associated with the above defense mechanism. According to the above description, there is no extra communication overhead. The extra computation overhead comes from generating and verifying the signatures for packet headers. Comparing to packet body, the length of packet header is much smaller, so the extra computation overhead is also small.

Now we analyze the storage overhead. In the proposed defense mechanism, each node needs to store the set of packet headers between two consecutive route query requests. In mobile ad hoc networks, due to dynamic topology change, the time interval between two consecutive route query requests is usually short. Therefore, the number of packet headers that need to be stored is also small. Now we present a method which further reduces the storage overhead yet following the detecting rules related to distributed defense system.

V. CENTRALIZED DETECTION WITH DE-CENTRALIZED IMPLEMENTATION

In the modified version, instead of performing attacker detection by itself, each good node will report the observed information to certain nodes which we called centralized detectors, then the centralized detector will perform attacker detection based on the collected traffic information.

The detailed description of the modified defense system is as follows:

First, the route discovery and packet delivery procedure is the same as described in Section 3.1.

Second, the monitoring mechanism is still header watcher as described in Section 3.2. with the following modification: for each good node, instead of storing all listened valid packet headers locally, most time it does not need to any packet headers locally, but only needs to store the following three-tuple (traffic pair, sequence number, route) that is associated to each listened valid packet headers. A good node needs to record the full packet headers only if it has been notified by the centralized detectors to do so, as explained next. Furthermore, instead of reporting each listened packet header information separately, each good node will report the listened packet header information in a batch mode, that is, each report consists of many listened packet header information.

For the centralized detector, its job is to perform injecting traffic attack detection by applying similar detection rules as described in Section 3.3. The major difference lies in that when the centralized detector performs injecting traffic attack detection, the procedure is

two steps. In the first step, the detector will check whether a node has injected two packets with the same sequence number or whether a sequence number is larger than specified upper-bound based only on the collected partial packet header information, that is, without checking the packet header signatures. If any of the two conditions has been satisfied, the detector then will request those nodes who report such information to submit full packet headers. That is, the centralized detector needs concrete evidence to charge the attacker.

Now we analyze the detection performance of the modified defense system. It is easy to see that either simple IDPA or long-route IDPA can be easily detected. Meanwhile, for the multi-route IDPA, requiring packets sent via different route to use different sequence number has not gain from the attacker's point of view, and allowing packets sent via different route to use same sequence number will be detected immediately when omnidirectional transmission technique is used.

Comparing to the fully distributed defense system described in Section 3, the storage overhead of the modified defense system can be dramatically reduced, but some extra communication overhead is introduced due to that each node needs to report to the centralized detector. However, since the size of each report is very small comparing to the data packet, the extra communication

Until now we have assumed that each good node will keep listening all the packet transmission in its neighborhood. Next we show how to further decrease the overhead by letting nodes selectively listen packet transmissions, with negligible degradation of the detection performance. Specifically, each node can selectively listen its neighbors' transmission with a certain probability p , which we called probabilistic monitoring. That is, a packet transmission event happens in a good node's neighborhood, with only probability p this node will monitor this transmission and report the observation to the centralized detector. Now when an attacker has injected $n \geq 2$ packets with the same sequence number via n node-disjoint routes, with no more than probability equal to $p(n) = (1 - p)^n + p(1 - p)^{(n-1)}$, the attacker can avoid being detected. Furthermore, after the attacker has injected k packets, the probability that it will not be detected will be decreased to $p(n)^k$, which goes to 0 with the increase of k . By applying probabilistic monitoring, the communication overhead can be further decreased by $(1 - p)$, while the detection performance only suffers negligible degradation.

One possible drawback of such centralized detection mechanism is that the detector itself can also become attackers' target. Besides increasing the protection level, one can also increase the number of centralized detectors. For example, if there are 2 detectors in the network, even one has been compromised, the other still work well. In

this case, for each node, it can either submit report to both detectors, or each time randomly pick one to submit, where the later is equivalent to reducing p by half

CONCLUSION

One of the major threats to wireless ad hoc networks, the injecting traffic attack have been discussed. We have also presented the above methods which have high immunity to these attacks. The first method depicted the major distributed defense system and the second method utilized the concept of centralized detection to achieve overhead reduction. We hope that by utilizing these methods the attacks can be pinpointed and necessary actions could be taken against malicious nodes. This is very essential since the ad hoc networks have been proliferating in the recent times and are being used as the means of sharing confidential data by firms etc. which incur heavy losses in case of spoofing the networks. We conclude that by using the proposed methods, the wireless ad hoc networks would be fortified against the injecting traffic attacks.

REFERENCES

- 1] Wei Yu securing Wireless ad hoc Networks under Noise and Imperfect Monitoring
- [2] Jong Youl Choi security Problems for ad hoc routing protocols, New York University. overhead is negligible.