ARP Attacks in Wireless Ad Hoc Networks

Gautam Sadhir
Carnegie Mellon University
`gsadhir@andrew.cmu.edu`

Yih-Chun Hu
UC Berkeley
`yihchun@cs.cmu.edu`

Adrian Perrig
Carnegie Mellon University
`perrig@cmu.edu`

Abstract

Previous research to secure ad hoc network protocols concentrates on key distribution and on securing the routing protocol. Unfortunately, securing the routing layer is not sufficient to secure the ad hoc network, as lower layer and upper layer communication protocols remain vulnerable to attack.

In this paper we illustrate that attacks against the ARP protocol are particularly devastating, even if a secure ad hoc network routing protocol is used. We demonstrate several attacks that paralyze network communication, even if only a small number of attackers are present. We present extensive simulations of a realistic attacker, and design appraoches to defend against ARP attacks.

1. Introduction

In a wireless ad hoc network, network nodes also act as routers, allowing communications between nodes not within direct wireless transmission range. Such networks are useful in construction, exploration, and military situations in which existing infrastructure is unusable, and have recently seen increasing use as a low-cost, incrementally deployable last-mile solution [19, 1, 5]. In military networks, as well as in ad hoc networks used for last-mile connectivity, some nodes are likely to be compromised by malicious users.

It is imperative to secure all communication layers. For example, if we only secure the higher layers, attacks at the lower layers can paralyze the protocol. Similarly, if we only secure the lower layers, attacks against the upper layers are usually still feasible.

A number of *secure* routing protocols have been developed for routing in this situation [3, 9, 10, 14, 15, 16, 18, 20, 21]. These protocols, however, focus only on the routing layer; that is, for each incoming packet, these protocols determine the next-hop destination for that packet. A number of attacks on the physical and MAC layers have been proposed, but the vulnerabilities of the Address Resolution Protocol (ARP) have not yet been explored in this context.

In proposed ad hoc network routing protocols, the next-hop destination of each packet is determined by the routing protocol. ARP then translates the IP address of the next-hop to the MAC address of the next-hop using the node's ARP Cache. ARP is widely known to be vulnerable to cache poisoning. This cache poisoning problem is significantly more dangerous in wireless networks for three reasons. First, many wired links are point-to-point, and ARP is not used on point-to-point links. Secondly, attackers are readily disconnected from wired networks, but in a wireless network disconnecting a node is significantly more difficult. Finally, when two machines establish a peering agreement, significant effort is required to connect those machines; as a result, key distribution is a relatively low cost, incrementally deployable solution in wired networks.

In this paper, we explore attacks against ad hoc network routing focusing on attacks that target the ARP protocol. To build a secure ad hoc network, it is imperative that all communication layers are secure—and as we demonstrate in this paper, the ARP layer is particularly important to secure as an attacker can completely paralyze communication in current ARP implementations.

Contributions To the best of our knowledge, this paper is the first paper to comprehensively treat ARP attacks in ad hoc networks. We present several new ARP attacks and perform simulations to show that these attacks practically paralyze the network. We also present and evaluate countermeasures against these attacks.

Outline Section 2 describes the ARP protocol in the context of ad hoc networks. Section 3 describes our attacks against the ARP protocol, and Section 4 presents potential countermeasures. Finally, we present our conclusions in Section 7.

2. Address Resolution Protocol

In the Internet and most ad hoc networks, routing protocols operate at the network layer. Network nodes are identified by their IP address at the routing layer, and this layer is responsible for establishing routes that a packet uses to reach its destination. Packets are then handed down to a sub-IP layer protocol, which is responsible for determining the MAC address of the next hop towards the destination. After the next-hop IP address is translated into a MAC address, the packet can be sent out at the link layer. The sub-IP layer uses a resolution protocol to provide IP-to-MAC address translations. ARP [17] is a simple protocol that has traditionally been used to provide IP-to-MAC address translations in both wired and wireless domains.

ARP functions as follows. When a node A's routing protocol chooses a next-hop node B for a packet whose MAC address is unknown, ARP buffers this packet and generates an ARP REQUEST message. Traditionally, ARP buffers exactly one packet per destination. The ARP REQUEST contains A's IP address, A's MAC address and B's IP address. This ARP REQUEST is broadcast by node A, and is received by all of A's neighbors. The node whose IP address matches the address in the ARP REQUEST (B in this case) responds to the request by constructing an ARP REPLY message and unicasting it back to A. The returned ARP REPLY contains the MAC address of B. When A receives this ARP REPLY, it makes an entry in its ARP cache, associating B's IP address with the MAC address it learned from the ARP REPLY. The node A can then send out the packet that was initially held for B, since B's MAC address is now available. Whenever the routing protocol chooses B as the next-hop destination for any other packet, ARP would use the entry in its route cache to set the MAC destination.

To fill the ARP cache more rapidly, ARP makes use of information from ARP REQUESTs to fill its cache. In particular, since the ARP REQUEST includes the IP and MAC addresses of the requesting node, any node receiving this broadcast REQUEST can make an entry in its ARP cache, associating the requestor's IP address with its MAC address. Only the neighbor whose IP address matches the one being queried by the ARP REQUEST responds with an ARP REPLY; other nodes simply discard the ARP REQUEST after populating their ARP cache. To ensure freshness of the ARP cache, the entries are typically purged periodically. Most ARP implementations for wired networks have a timeout of 20 minutes, after which aged entries are removed, and another exchange of ARP REQUEST and REPLY packets is necessary for further traffic to use this link.

ARP is a connectionless and stateless protocol. ARP REPLYs that are returned are not matched with ARP REQUESTs that are sent out. This makes APR vulnerable to a number of spoofing and ARP cache poisoning attacks. A malicious node may send out spoofed gratuitous ARP messages, associating its MAC address with some other node's IP address, poisoning caches of all nodes that receive the gratuitous ARP message. A malicious node may also target specific nodes, and poison their caches by sending spoofed ARP REPLY messages unicast to them. Such attacks have been studied and defended against well in wired networks, but they pose a serious threat to wireless networks [4], and in particular ad hoc networks.

2.1. Dynamic Source Routing

DSR is a simple and efficient on-demand ad hoc routing protocol. A node in DSR attempts to find a route to a destination only when it has some data to send— this helps in keeping overheads relatively low in DSR in comparison with other ad hoc network routing protocols. The protocol has been shown to respond quickly to topology changes resulting from node mobility [6]. DSR is composed of two distinct mechanisms, Route Discovery and Route Maintenance, which are used, respectively, to set up paths for data, and to find alternate paths in case of link failures. Route discovery is initiated by a node that wishes to send data to a destination but does not already have a route to that destination. The sending node sends out a Route Request (RREQ) message that is flooded through the network. The RREQ message contains an (initially) empty list of node addresses. Each intermediate hop that receives and processes the broadcast RREQ appends its own address to the list and rebroadcasts the RREQ. Finally, when the intended destination receives the RREQ, it constructs a new message called a Route Reply (RREP), which it sends back to the RREQ originator, either by reversing the path that the received RREQ took, or by initiating a new Route Discovery for the source. To prevent overhead due to excessive re-broadcasting, DSR requires each new RREQ to carry a unique identifier. Intermediate nodes processing RREQs first check to see if they have already processed a RREQ with the same identifier (that is, from the same Route Discovery). Duplicate RREQs from the same Route Discovery are dropped, and are not rebroadcast.

Since a target may receive multiple RREQs through different paths, it usually ends up sending a number of RREPs back to the source. The source caches all the RREPs that it receives. Once the original sender receives a RREP, it can use this route to send data to the destination. Each such data packet that is sent out from the source carries with it the address of each hop along the path to the final destination. It is for this reason that the protocol is called *Source Routing*; the source of each data packet specifies the entire path that the data should take through the network.

Each node forwarding a packet performs Route Maintenance to ensure that the packet reached the next hop. If it cannot determine that a packet it forwarded successfully reached the next hop, it determines that the link from itself to the next hop is broken. This may be caused as a result of the communicating nodes moving beyond their direct communication range. Upon detecting the transmission failure, the node constructs a new message, called a Route Error (RERR), which it sends back to the originator of the data, informing it about the link failure. The source, upon receiving the RERR, removes that link from its cache. It may then look up its cache to find a different route to the destination, and use that route to send subsequent data packets. If the source node does not know of any other route, it may initiate a new Route Discovery to find new paths.

DSR makes use of many techniques to optimize end-to-end throughput and latency. Caching of source routes is employed extensively— not only do the source and destination learn the path between themselves after Route Discovery, but all nodes along the path also learn about each other's presence. Since a node can overhear all transmissions within its communication range, a node, if so configured, can learn about new routes by just overhearing packets received or sent by its neighboring nodes and snooping on their source routes. Similarly, Route Error propagation leads to purging of all cached routes which contain the failed hop, at each node that propagates the route error, or overhears a node transmitting a route error. Because of these optimizations, DSR is able to track topology change information, and disseminate this information quickly throughout the network. Several other optimizations are also present in DSR, the details of which can be found in [11].

Table 1: ARP Requests by Packet Type

Type of packet	Number of ARP REQUESTS
RREQ	0
RREP	465
RERR	32
Total DSR	497
Data Packets	336

3. ARP Attacks

3.1. ARP Reply Spoofing

In an on-demand ad hoc network routing protocol, ARP packets are generally sent in response to routing protocol messages; after routing protocol messages have been exchanged, the ARP cache entries can be used for actual data packets. For instance, in DSR, Route Replies moving towards the requesting wireless node require the use of ARP for IP-to-MAC address translation of next hop nodes. (Route Requests do not invoke ARP since those are flooded through the network as broadcast packets.) Table 1 shows results from a typical ad hoc network simulation, using DSR as the routing protocol, showing that most ARP Requests are sent because of a Route Reply. The parameters used for the simulation run are tabulated in Table 2.

When a packet is handed down to the link layer for a destination whose MAC address in unknown, ARP broadcasts a ARP REQUEST message to query for the unknown MAC address. The packet triggering this request is held in a buffer (one packet per outstanding IP destination) until an ARP REPLY is heard from the destina-

Table 2: Simulation Parameters

Simulation Parameter	Value
Topology	1500x300m
Number of nodes	50
Type of traffic	CBR
Number of connections	20

tion. The node whose IP address matches the address in the ARP REQUEST sends back a unicast ARP REPLY, including its MAC address in the response. When this response is heard at the querying node, the newly discovered MAC address is included in the appropriate headers in the held packet and the packet is removed from the buffer and sent to the lower layer. The destination eventually receives the packet that was held for it.

A malicious node may misuse the ARP mechanism to steal packets that are held for other nodes. Since ARP REQUESTs are broadcast, a malicious node can hear ARP REQUESTs sent by its neighbors. When a malicious node M hears such a request by node A for node B's MAC address, it constructs a spoofed ARP REPLY, associating its own MAC address with B's IP address. It then unicasts this ARP REPLY to A, ignoring delays in the link layer and the MAC layer. The queried node B would also send back a unicast ARP REPLY message to A. If node A receives M's spoofed response before the valid ARP REPLY from B, the held packet would be sent out with M's MAC address and the malicious node would be successful in stealing the packet intended for B. The malicious node M also successfully poisons A's cache, though this poisoning is short-lived. When A receives the legitimate ARP REPLY from B, its ARP cache reverts the IP-MAC association to correct values. The attack can be made more powerful if the malicious node M sends a duplicate spoofed ARP REPLY to A after a sufficient duration to poison the ARP cache again, with the intent of receiving all subsequent packets sent by A for B. Malicious nodes may also collude, and only steal packets intended for non-malicious nodes. Hence, stolen packets fall in two distinct categories— those stolen by the first (rushed) ARP REPLY, and those stolen by the second (poisoning) ARP REPLY. Most of the held packets stolen are routing layer packets, whereas packets stolen in extended steals are generally data packets.

Malicious nodes, with this attack, can adversely affect the route discovery phase in on-demand ad hoc routing protocols. By stealing Route Reply packets in DSR, malicious nodes disrupt path setup through non-malicious nodes. Paths through malicious nodes are thereby favored, giving them a higher share of network traffic. If malicious nodes drop all data packets that they receive, this increased share of network traffic adversely impacts end-to-end throughput. Malicious nodes also receive a large number of data packets due to poisoned ARP caches of their neighbors, and dropping these stolen data packets also contributes to reduced throughput.

3.2. Promiscuous Cache Poisoning

Because of the broadcast nature of the wireless medium, nodes in a wireless network can overhear transmissions made by nodes within wireless communication range. Ad hoc network routing protocols can often improve their performance by using this information, for example by using routing layer headers to learn new routes, updates and topology changes. This optimization allows topology and routing information to quickly spread through the network, making the routing protocol more adaptive to network changes.

However, malicious nodes can misuse the promiscuous mode by implementing a timed ARP spoof attack. When a malicious node M promiscuously receives a data packet sent by a neighboring node A to another neighboring node B, it can infer that data will flow between nodes A and B. It attempts to disrupt communication between the nodes by poisoning A's ARP cache. To do so, it constructs a spoofed ARP REPLY message, associating B's IP address with its own MAC address, and unicasts this ARP REPLY to A. This reply overwrites the existing entry for B in A's cache. This is because ARP is a connectionless and stateless protocol. ARP does not match outgoing ARP REQUESTs with incoming ARP REPLYs, and hence asynchronously sent spoofed ARP REPLYs successfully overwrite existing cached entries. All subsequent packets that are sent by A, addressed for B, would instead be received by the malicious node M because of the poisoned ARP cache entry. The malicious node would continue to receive these data packets as long as it stays within A's transmission range. Without this attack, if B had moved out out of A's transmission range, A's the next transmission to B would have caused MAC layer failures and subsequent routing layer mechanisms to deal with the failures; however, because A's ARP cache is poisoned, M continues to receive data packets for B even though B has moved out of A's transmission range.

Instead of stealing packets destined for other nodes, malicious nodes can attempt to cause those packets to be dropped, by using an invalid MAC address in spoofed ARP REPLY messages. As before, the malicious node M poisons A's cache with a spoofed entry, but instead of associating B's IP address with M's MAC address, it associates B's IP address with an invalid MAC address. Subsequent transmissions by A for B would cause MAC failures, invoking potentially expensive routing layer mechanisms. For instance, in DSR, Route Maintenance would be invoked at A, which causes a Route Error to be generated and sent back to the original source of the data packet. In addition to increasing overheads, Route Errors purge any routes using the A$\rightarrow$B link at any intermediate node on the reverse path from A to the source node. This attack thus causes valid routes to be discarded, which causes the routing protocol to either choose new routes (possibly through malicious nodes) or perform new route discoveries (further increasing routing overhead).

3.3. Spoofing ARP REQUESTS

Previous attacks discussed poison caches of specific nodes, since spoofed ARP REPLYs are unicast to specific target nodes. A more serious attack can result from the misuse of ARP REQUESTs, since ARP REQUESTs cause an IP-to-MAC association of the broadcaster to be entered in ARP caches of all nodes that receive the request. Malicious nodes can link ARP REQUEST spoofing with the Route Discovery phase in on-demand ad hoc routing protocols such as DSR. A node A, that broadcasts or forwards a Route Request, elicits Route Replies from its neighbors. The neighbors may have cached routes to the destination, in which case they may immediately respond to the Route Request with a Route Reply providing that cached route. If the neighbors do not have cached routes to the destination, they would forward the Route Request, appending their own address to a route record kept in the Request, thus allowing the returned reply from the destination to retrace its path to the requesting source. In either case, neighbors of node A, before sending Route Replies to A, would need to query for A's MAC address, if they do not already have an IP-to-MAC association for A in their ARP cache. A malicious node M, also in A's neighborhood, may attempt to steal all the Route Replies returned to A by pro-actively poisoning caches of A's neighbors in response to hearing a Route Request being broadcast (or forwarded) by A. To do this, M constructs an ARP REQUEST message, spoofing A's IP address and broadcasts it as soon as it hears a Route Request from A. It includes its own MAC address in the ARP REQUEST, thereby creating an association between A's IP address and M's MAC address. In the Address Resolution Protocol, all nodes that receive an ARP REQUEST, regardless of whether or not their IP address matches the one being queried, make an entry in their ARP cache reflecting the IP-to-MAC association of the node transmitting the ARP REQUEST. Any node that receives the spoofed ARP REQUEST from M will therefore make an incorrect entry in its ARP cache, which it would use to send packets addressed to A. When performing this attack, the malicious node's ARP REQUEST does not actually solicit an ARP REPLY; as a result, the malicious node can query for a random or invalid IP address. As a result of the poisoning, all packets (including Route Replies) sent by any of A's neighbors addressed for A would instead be received by M. The malicious node can severely impact the Route Discovery process in this way, by stealing and discarding Route Replies from non-malicious nodes. The attack allows malicious nodes to thus favor path setup through themselves or through other malicious nodes. The malicious nodes hence gain a higher percentage traffic share in the network. Also, since many legitimate paths are prevented from being set up, traffic would be routed through non-optimal paths, thus impacting end-to-end latency; which can be critical for real time applications. Further, if malicious nodes choose to drop all data packets that they receive; this attack can severely impact end-to-end throughput. Spoofing ARP REQUESTs as described above is a more powerful attack than spoofed ARP REPLYs for two reasons. Firstly, a single ARP REQUEST poisons caches of all neighbors, as compared to an ARP REPLY which is unicast to a target node. Secondly, by proactively poisoning neighbor caches, the malicious node reduces the number of ARP REQUESTs generated by neighbors of the victim. ARP REQUESTs by neighbors elicits ARP REPLYs from legitimate nodes, which may install correct IP-to-MAC associations, or overwrite existing spoofed associations installed by malicious nodes.

The malicious node can also launch a slightly different attack in which it poisons neighbor caches with invalid MAC address translations. By sending a spoofed ARP REQUEST message associating an invalid MAC address with the Route Request broadcaster's (A's) IP address, the malicious node M causes expensive Route Maintenance mechanisms to be invoked at the neighbors whenever the neighbors have a packet to send to A. This increases packet overhead in the network greatly, causes valid routes to be discarded and non-optimal paths to be setup. Forwarding of data through non-optimal paths impacts latency, while at the same time giving malicious nodes a higher share of network traffic.

3.4. Limitations of ARP Attacks

ARP attacks can often be detected by victim nodes. For example, many operating systems can detect when another system on the same subnet is using the same IP address. However, without a secure ARP protocol, the true owner of the IP address has no way of authenticating its ARP packets to the exclusion of the attacker's ARP packets. One way around this limitation is to use directional antennas aimed away from the victim; however, this also limits the impact of the attack, and particularly attacks that use broadcast ARP messages for cache corruption.

Another way to mitigate the effects of an ARP attack is to operate the network interface in *promiscuous mode*. In promiscuous mode, packets received by the network interface are passed to the IP layer regardless of the MAC destination address. This action can save some of the stolen packets, and the victim can send a gratuitous ARP REPLY to the node that incorrectly sent the packet to the wrong destination. However, this is an inadequate response to the ARP attack, for three reasons. First, the ARP REPLY cannot be authenticated, so the attacker can send another ARP packet to again corrupt the sender's cache. Secondly, broadcast packets are less reliable than unicast packets because of the lack of medium reservation, so the victim is less likely to hear packets destined for it when the wrong MAC destination is used. Finally, when the victim moves outside the sender's wireless transmission range, the malicious node still sends link-layer acknowledgments, which prevents route maintenance from detecting the link breakage. This allows the malicious node to continue stealing packets, and also prevents the victim from recovering the packets since the victim can no longer overhear those packets.

4. Countermeasures

Ad hoc routing protocols can be fortified against ARP related attacks in many ways. One way is to secure the Address Resolution Protocol itself using cryptographic mechanisms. Bruschi et al have developed a secure address resolution protocol called S-ARP [7]. This protocol requires each node to have a public/private key pair used to authenticate ARP messages. All ARP communication that is sent by any node is signed by the sender's private key and verified by the receiver. This ensures source authentication and prevents spoofed ARP REPLYs from polluting cached entries. This scheme, though well-suited to infrastructure networks with centralized administration, is not ideal for use in ad hoc networks because of its use of expensive asymmetric cryptography, and because it requires a trusted certification authority which may not be possible in a decentralized ad hoc network.

Another way of avoiding ARP related attacks is to totally prevent the use of ARP as an address translation mechanism. In *static ARP*, each node in the network adds every other node in the network to its ARP cache with a static IP-to-MAC address association. This ensures that ARP need not send any packets for address resolution. This scheme is frequently used to prevent spoofing of critical routing infrastructure such as routers and switches in wired networks. Any ARP traffic that is observed on the LAN can be labeled as spoofed, and countermeasures can be taken to detect the source of malicious ARP traffic. These ideas, though applicable in subnets and small LANs, can also be difficult to deploy in ad hoc networks, because it is difficult to isolate traffic in a wireless network, and because of the requirement for a single trusted entity to provide each node with these IP-to-MAC address associations.

By integrating an address translation mechanism into the routing protocol itself, the need for an explicit resolution protocol can be nullified. Existing ad hoc routing protocols can be extended to include automatic address resolution in the routing layer. This integration eliminates the need for ARP packets entirely, protecting routing protocols from ARP-related attacks. Secure versions of ad hoc routing protocols, which are also vulnerable to ARP attacks, can also integrate address resolution with the secure routing layer to further fortify the protocol.

Not only does this integration provide increased security in wireless ad hoc networks, but it is also beneficial in terms of performance. ARP message exchange increases the latency of packets triggering that exchange. By proactively populating the ARP cache, some delays in packet delivery can be avoided, and applications can benefit from improved latencies. In addition, packet loss can sometimes be avoided; ARP holds at most one packet for each destination while waiting for ARP REPLYs. Subsequent packets that are handed down to ARP for the same destination overwrite the contents of the single packet buffer. By automatically determining IP-to-MAC address translations within the routing protocol, packets need not be buffered or dropped due to ARP. For example, in an on-demand ad hoc routing protocols such as DSR, a route discovery leads to a reverse broadcast storm of route replies towards the source, with each hop requiring an ARP exchange to resolve next hop addresses. As the route replies converge, they cause congestion around the source node, again resulting in increased packet loss. By tightly integrating address resolution with the routing protocol, these effects can also be avoided.

4.1. Extending DSR to Support Automatic Address Resolution

On-demand ad hoc routing protocols such as DSR can be augmented to include automatic address resolution. Each data packet in DSR contains the route that the packet should take to reach the destination. This list of IP addresses can be augmented (or replaced) by including MAC addresses of each hop in the source route. In this case, routing protocol packets (Route Requests, Route Errors and Route Replies in DSR) are modified to include the necessary MAC addresses. This approach avoids the exchange of ARP messages, at the cost of additional overhead in each packet. In addition, hop-by-hop routing protocols are unable to take advantage of such an approach. To address these problems, we observe that MAC address translation is only needed on a per-hop basis. If the routing protocol maintains a neighbor cache that holds IP-to-MAC translations of all known neighbors, it can populate this neighbor cache based on existing routing packets and can use this cache to eliminate the need for ARP packets.

Automatically obtaining IP-to-MAC address associations by examining routing protocol packets is necessarily protocol-dependent. In this section, we focus on protocol messages used in DSR, though the application of these techniques to other on-demand routing protocols is relatively straightforward.

When a DSR node broadcasts (or rebroadcasts) a Route Request packet, the last node to transmit the packet can be determined from the source route contained in each such Route Request. Nodes hearing this Route Request can associate the IP address from the source route with the source MAC address in the MAC-layer header of the Request. The Route Request is then processed as specified by DSR. When nodes cache IP-to-MAC address associations from Route Requests, all nodes between the source to the destination would know the previous hop's MAC address before route replies retrace the path towards the source node. This avoids the need for ARP packets during the route reply phase.

When data flows over a discovered route, the last hop may not have a IP-to-MAC address mapping because the target does not forward the Route Request. In addition, because of the unreliable nature of broadcast packets in wireless networks, the ith node in a route may not have heard the Request forwarded by the $i+1$st node. To automatically create these IP-to-MAC mappings, we take advantage of the information in the source route and MAC header; in particular, DSR uses a source route to specify the path along which the Route Reply is to be returned to the initiator. When the $i+1$st node in a discovered path forwards the Route Reply to the ith node in that path (since the Route Reply is generally sent by reversing the discovered path), the ith node can retrieve the IP address of the $i+1$st node (from the source route) and the MAC address of that node (from the MAC header). The ith node then places this information in its neighbor table.

This shows that in DSR, after the route discovery phase, all intermediate hops in the discovered route can have IP-MAC mappings for each of their neighbors on the path. These techniques can improve path setup times in DSR.

since route replies avoid performing ARP message exchanges at each hop.

DSR can also promiscuously learn routing information from packets it overhears. However, whenever it learns such a route, it will also hear a source route, and it can therefore determine the IP address of the node that last forwarded that data packet (from the source route) and the MAC address of that node (from the MAC header).

DSR does not require that routes be proactively expired from route caches; as a result, a node can use an old route. If node A and node B used to be neighbors, drifted apart, and then moved back within wireless transmission range of each other, they may have purged the associated neighbor cache entries when they discovered that the other node was no longer directly reachable. In these cases, we can build a routing layer mechanism similar to ARP to perform address resolution. In the next section, we discuss why such a mechanism might be more advantageous than using ARP directly.

4.2. Extending Secure Ad Hoc Routing Protocols with Automatic Address Resolution

The techniques presented in the previous section do not provide much security, since an attacker can overflow the neighbor caches of its neighbors by sending packets with bogus IP addresses and MAC source addresses. The problem is that IP addresses are not authenticated. Even if IP addresses were authenticated, the MAC header is not generally included in the authenticated fields, so an attacker can simply replay legitimate packets to create bogus associations in neighbor nodes. In order to secure the IP-to-MAC address translation, both the IP and MAC addresses need to be authenticated; for example, in the ARP-like protocol described in the previous section, we can add an authenticator that covers the IP and MAC addresses that are to be associated.

We observe that in a secure ad hoc network routing protocol, the problem of key distribution is very similar to the problem of distributing IP-to-MAC associations in Static ARP. In particular, we can treat a node's MAC address as part of its public key that is signed by a certificate authority. Protocols with a built-in way to distribute public-key certificates, such as ARAN [18], can securely distribute IP-to-MAC associations using this mechanism. Otherwise, protocols that use a public key, such as our Ariadne and SEAD protocols [10, 9], would distribute the IP-to-MAC associations together with the necessary public or private keys before network deployment.

5. Evaluation

5.1. Methodology

To evaluate the impact of ARP spoofing, we simulated the attacks described in Section 3. We selected the *ns-2* simulator with the Monarch extensions [6] because it provides a realistic simulation of wireless propagation, and because it correctly models the IEEE 802.11 MAC, ARP, and DSR. We ran our simulations with 50 nodes in an 1500 m $\times$ 300 m space, using a nominal radio transmission range of 250 m. Each node moved according to the random waypoint model with zero pause time; that is, each node started in a random location, chose a random destination and a random speed between 0 and 20 m/s, and moved to the destination at that speed. Once it reached that destination, it repeated the process, choosing a new destination and new speed. Each simulation represents 900 seconds of simulated time. We chose to use constant bit rate (CBR) traffic to best represent the ability of the protocol to deliver a packet at any given time. Usage of a conforming load (such as TCP) would grant even more advantage to the attacker by shutting down the source whenever a path is chosen through the attacker. In our simulation, there were 20 CBR sources, each sending 4 packets per second. The packets each contained 64 bytes of data.

We simulated a varying number of malicious DSR nodes using several attacks. We varied the number of attackers from 0 to 31, and we ran 10 simulations for each data point. In the baseline attack, the malicious nodes simply participate in DSR as normal, but drop all traffic that they are asked to forward. In addition to the baseline attack, we simulated the following spoofing attacks:

- Spoofing ARP REPLYs based on ARP REQUESTs for non-malicious nodes (Section 3.1)
- Using promiscuously overheard data packets to detect data flows, and spoofing ARP REQUESTs in re-

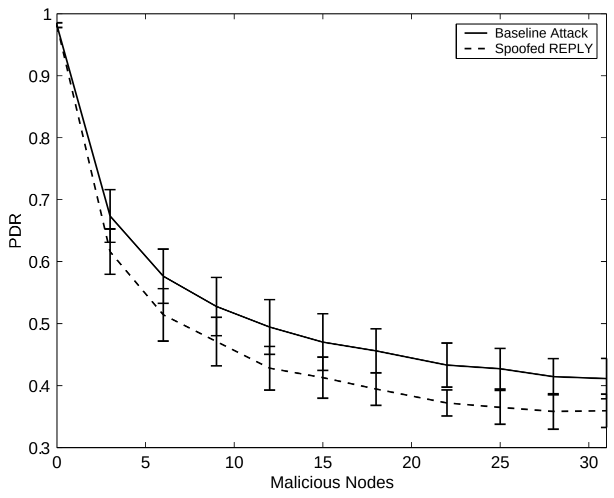

Figure 1: ARP reply snoop rushing.eps

sponse to such flows (Section 3.2)

- Spoofing ARP REQUESTs based on overheard Route Request packets (Section 3.3)

We evaluated performance primarily along the metric of Packet Delivery Ratio (PDR). The packet delivery ratio is the fraction of packets sent by the application layer which are received at the destination.

5.2. Results

Figure 1 shows the results from the simulation of the ARP REPLY spoofing attack (Section 3.1). Packet Delivery Ratio (PDR) is plotted against the total of malicious nodes in the network. Each data point on the graph shows the average over ten simulations, and the error bars show the 95% confidence interval of the PDR. The ARP REPLY spoofing attack is compared against a baseline attack, in which malicious nodes simply discard all data packets that they receive. When the ARP REPLY spoofing attack is performed, PDR drops almost 10% below the baseline performance. This is due to two reasons. First, due to ARP cache poisoning, malicious nodes steal and discard data packets that are intended for other nodes, directly impacting throughput. Secondly, malicious nodes also steal and discard Route Replies from non-malicious nodes,

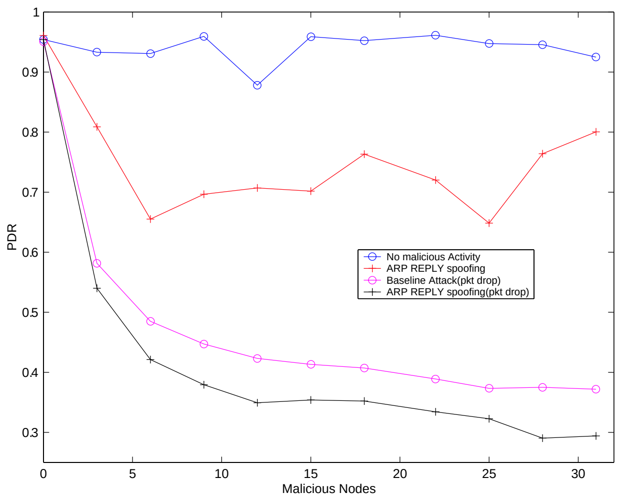

Figure 2: PDR with without pktDrop

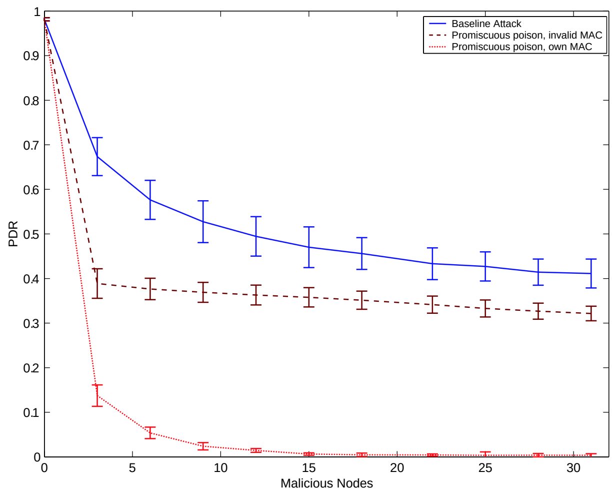

Figure 3: reply cache poisoning

which reduces path setup through non-malicious nodes. By favoring paths through themselves in this way, malicious nodes control (and subsequently discard) a larger percentage of data traffic. Figure 2 shows the impact of ARP REPLY spoofing when malicious nodes do not discard data packets that they receive for forwarding. Relative to a standard DSR simulation run, PDR drops significantly as a result of ARP REPLY spoofing, and con-

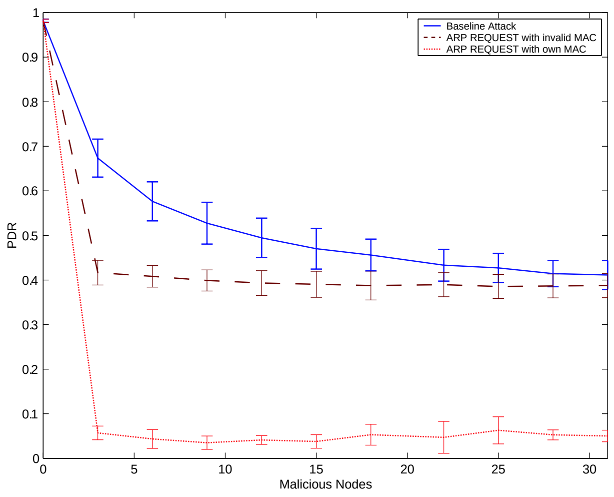

Figure 4: bogus request

sequent ARP cache poisoning. Also shown in the same graph are curves for the baseline attack, and the ARP REPLY spoofing attack, when malicious nodes discard data packets which they are asked to forward.

The impact of promiscuous cache poisoning through spoofed ARP REPLY messages (Section 3.1) is shown in Figure 3. When the attacker uses its own MAC address in the spoofed REPLYs, packet delivery performance is severely impacted even with a very small number of malicious nodes. When the attacker instead sends an invalid MAC address, performance is reduced relative to the baseline attack, but is well above when the attack uses the malicious node's MAC address. This is because an invalid MAC address invokes Route Maintenance in DSR, and nodes are somewhat successful in finding alternate paths to destinations, resulting in a higher PDR. This attack though, does result in increased packet overhead in the network since numerous Route Errors are generated.

Figure 4 shows the results from ARP REQUEST spoofing attacks (Section 3.3). As before, the attack where the malicious node uses its own MAC address is more effective than when the malicious node uses invalid MAC addresses. When the malicious node uses its own MAC address, the PDR drops very quickly, even for small numbers of malicious nodes. When there are few malicious nodes, this attack is more successful than spoofing ARP REPLYs, because a single malicious node transmission affects the ARP caches in many more nodes. However, as the number of attackers increases, spoofing ARP REPLYs becomes more powerful. This is because of the effects of local congestion caused by several malicious nodes spoofing ARP packets: in the ARP REPLY spoofing, the congestive effects equally affect legitimate and attacking ARP packets (and the number of attacking ARP packets far exceeds the number of legitimate ones in order to cause congestion); in REQUEST spoofing, the spoofed packets compete with each other and legitimate Route Requests. As a result, when the spoofed packets collide, ARP will get a chance to find an IP-to-MAC address translation.

6. Related Work

The notion of *Manycast Transactions* in ad hoc networks [8] also brings forth the idea of address resolution integration with the routing protocol. The authors also discuss some of the benefits that can be achieved through this integration. There has also been previous work on Topology Broadcast based on Reverse Path Forwarding (TBRPF) [13], in which the authors provide support for automatic address resolution. However, neither of these approaches discuss the security issues inherent in ARP, and because they do not secure the underlying routing messages, they cannot provide secure IP-to-MAC address translation.

The problem of secure IPv6 Neighbor Discovery is similar to the well-known ARP weaknesses. A number of proposals have been made for securing IPv6 Neighbor Discovery (e.g. [2, 12]), usually based either on a PKI or on cryptographic addresses. We differ from this work in two ways: first, our approaches do not rely on asymmetric cryptography and can often be implemented with no overhead, and second, we demonstrate the damage that ARP spoofing can cause in ad hoc networks.

7. Conclusion

To build a secure ad hoc network, it is imperative that all communication layers are secure, as otherwise an attack targetted against a layer that is not secure could compromise the system.

Prior research on secure ad hoc network protocols has largely ignored attacks against the ARP protocol. As we demonstrate in this paper, the ARP layer is particularly important to secure as an attacker can completely paralyze communication if current ARP implmentations are used. We perform a comprehensive study and present measures to counteract these attacks. We find that a secure ARP protocol can largely mitigate these attacks. However, better security can be achieved by combining ARP functionality with higher layers and securing both layers simultaneously.

References

[1] Daniel Aguayo, John Bicket, Sanjit Biswas, Douglas S. J. De Couto, and Robert Morris. MIT Roofnet Implementation. Technical report, MIT, August 2003. Available at `http://pdos.lcs.mit.edu/roofnet/design/`.

[2] Jari Arkko, Tuomas Aura, James Kempf, Vesa-Matti M antyl a, Pekka Nikander, and Michael Roe. Securing IPv6 Neighbor and Router Discovery. In *ACM Workshop on Wireless Security (WiSe)*, September 2002.

[3] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *ACM Workshop on Wireless Security (WiSe)*, September 2002.

[4] John Bellardo and Stefen Savage. 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In *Proceedings of the USENIX Security Symposium, Washington D.C., August 2003*.

[5] Pravin Bhagwat, Bhaskaran Raman, and Dheeraj Sanghi. Turning 802.11 Inside-Out. In *Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[6] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual International Conference on Mobile Computing and Networking* (MobiCom 1998), pages 85–97, October 1998.

[7] D. Bruschi, A. Ornaghi, and E. Rosti. S-ARP: A Secure Address Resolution Protocol. In *Proceeding of the 19th Annual Computer Security Applications Conference, Las Vegas, Nevada*, December 2003.

[8] C. Carter, S. Yi, and R. Kravets. ARP Considered Harmful: Manycast Transactions in Ad-Hoc Networks. In *Proceedings of the IEEE WCNC*, 2003.

[9] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. *Ad Hoc Networks*, 1(1):175–192, July 2003.

[10] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking* (MobiCom 2002), pages 12–23, September 2002.

[11] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-07.txt, February 2002. Work in progress.

[12] G. Montenegro and C. Castelluccia. Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses. In *Symposium on Network and Distributed Systems Security (NDSS 2002)*, February 2002.

[13] Richard G. Ogier, Fred L. Templin, Bhargav Bellur, and Mark G. Lewis. Topology Broadcast Based on Reverse-Path Forwarding (TBRPF). Internet-Draft, draft-ietf-manet-tbrpf-11.txt, November 2003. Work in progress.

[14] Venkata N. Padmanabhan and Daniel R. Simon. Secure Traceroute to Detect Faulty or Malicious Routing. In *Proceedings of The First Workshop on Hot Topics in Networks (HotNets-I)*, October 2002.

[15] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Routing for Mobile Ad Hoc Networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, January 2002.

[16] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Link State Routing for Mobile Ad Hoc Networks. In *Proceedings of the IEEE Workshop on Security and Assurance in Ad Hoc Networks*, January 2003.

[17] David C. Plummer. An Ethernet Address Resolution Protocol. RFC 826, November 1982.

[18] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Elizabeth Royer, and Clay Shields. A Secure Routing Protocol for Ad hoc Networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols* (ICNP '02), November 2002.

[19] Rudi van Drunen, Dirk-Willem van Gulik, Jasper Koolhaas, Huub Schuurmans, and Marten Vijn. Building a Wireless Community Network in the Netherlands. In *Proceedings of the USENIX 2003 Annual Technical Conference*, pages 219–230, June 2003.

[20] Manel Guerrero Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *Proceedings of the ACM Workshop on Wireless Security* (WiSe), pages 1–10, September 2002.

[21] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6):24–30, November/December 1999.