Jamming and Sensing of Encrypted Wireless Ad Hoc Networks

Timothy X Brown

Jesse E. James

Amita Sethi

University of Colorado at Boulder

Technical Report CU-CS-1005-06
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309

Jamming and Sensing of Encrypted Wireless Ad Hoc Networks

Timothy X Brown, Jesse E. James, Amita Sethi
Electrical and Computer Engineering
Interdisciplinary Telecommunications Program
University of Colorado, Boulder CO 80309-0530
{timxb,jesse.james,amitha.sethi}@colorado.edu

ABSTRACT

This paper considers the problem of an attacker disrupting an encrypted victim wireless ad hoc network through jamming. Jamming is broken down into layers and this paper focuses on jamming at the Transport/Network layer. Jamming at this layer exploits AODV and TCP protocols and is shown to be very effective in simulated and real networks when it can sense victim packet types, but the encryption is assumed to mask the entire header and contents of the packet so that only packet size, timing, and sequence is available to the attacker for sensing. A sensor is developed that consists of four components. The first is a probabilistic model of the sizes and inter-packet timing of different packet types. The second is a historical method for detecting known protocol sequences that is used to develop the probabilistic models, the third is an active jamming mechanism to force the victim network to produce known sequences for the historical analyzer, and the fourth is the online classifier that makes packet type classification decisions. The method is tested on live data and found that for many packet types the classification is highly reliable. The relative roles of size, timing, and sequence are discussed along with the implications for making networks more secure.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Architecture and Design – *Network communications, wireless communication.*

General Terms

Algorithms, Design, Experimentation, Security,

Keywords

Ad hoc networks, jamming, sensing, encryption.

1. INTRODUCTION

Ad hoc networks are envisioned as playing a significant role in mission critical communication for the military, utilities, and industry. An adversary may attempt to attack a victim ad hoc network to prevent some or all victim communication. Such denial-of-service (DoS) attacks have been considered in ad hoc wireless networks at several levels. A number of researchers have considered DoS where the attackers are internal participants in the victim ad hoc network (see e.g. [8]). Ad hoc networks require the cooperation of peer nodes for their operation and are especially susceptible to such peer-based attacks. In this paper we consider encrypted victim networks in which the entire packet including headers and payload are encrypted and thus the attacker can not directly manipulate any of the victim communication. In this case, the attacker must resort to external physical-layer-based DoS, also known as *jamming*.

Jamming can be as simple as sending out a strong noise signal in order to prevent packets in the victim network from being received. This method of jamming is not the subject of this paper. This paper attempts to exploit the protocols at various layers to get three advantages: jamming gain; targeted jamming; and reduced probability of detection. Jamming gain is the increase in efficiency from exploiting features of the victim network relative to continuous jamming. More precisely, it is the amount of energy[1] used to achieve a desired effect relative to the amount of energy used to achieve the same effect with continuous jamming. This gain translates directly into reduced energy requirements for the attacker. At the link level, corrupting a single bit in a packet will cause the packet to fail its checksum and be discarded. For a 10,000 bit packet (1250 bytes) it implies that jamming gains as high as 40dB are possible. Further, typical wireless packet networks are lightly loaded so that jamming only when packets are present has further jamming gains. These examples make clear that there are significant jamming

[1] or power as appropriate

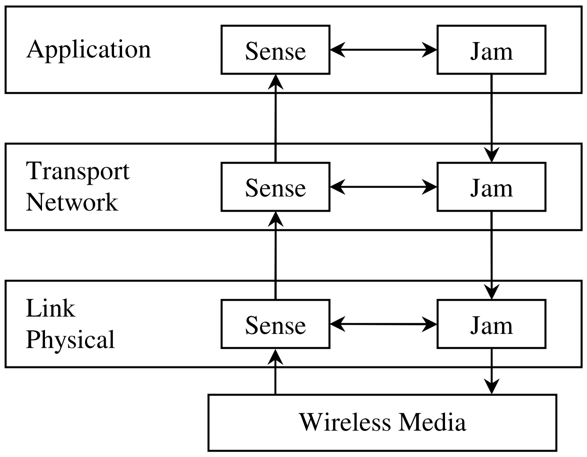

Figure 1: The sensing and jamming layered model

gains possible. This concept is more fully explored later in the paper.

Targeted jamming refers to jamming only specific victim nodes, links, or flows. The attacker may be interested in only certain parts of the victim network, and attacking only these parts can lead to further jamming gains.

With reduced probability of detection, the victim network may not realize that jamming countermeasures are necessary. Targeting some TCP-DATA packets will cause the TCP window to collapse and poor connection performance that a user might attribute to network congestion or a low quality wireless connection. Further, if ICMP packets are not blocked the victim users will have contradictory views of the network state. If jamming is discovered, lower probability of detection jamming will be harder to detect, localize, and suppress.

Jamming is not a transmit-only activity. It requires an ability to detect and identify victim network activity, which we denote as *sensing*. At the physical layer a sensor needs to identify the presence of packets. Since the network is encrypted, only the start time and size of the packet can be measured. At higher layers a sensor needs to classify packets using protocol information. In 802.11 for instance, whether a packet is successfully jammed or not can be seen by whether or not a node sends a short packet (i.e. the ACK) within 10μsec.

The key insight in this paper is that encryption only provides bit level protection of the data. This protection is in the form of bit level operations to remove any exploitable data structure. A packet network running protocols at multiple layers reimposes structure on the data. Any transmission follows specific patterns of DNS lookup, TCP connection set up, IP ARP, AODV route requests, and 802.11 atomic data exchanges. While these do not necessarily expose the bit-level data, they provide multiple avenues for DoS attack.

1.1 A Layered Model for Jamming

Together jamming and sensing can be broken down into a layered model similar to the OSI stack. We break it down into three levels for convenience as shown in Figure 1. The Link/Physical layer directly interacts with the media. If a higher layer requests a packet to be jammed, then this lower layer generates the physical signal and ensures that a packet and each of its link layer retries are jammed. This layer also provides the basic sensing capability of packet duration and timing. If sophisticated enough it could shield the upper layer from Link, MAC, and Physical layer control packets such as RTS/CTS and only report the higher OSI layer packets to the higher layer sensing and jamming.

The Transport/Network Layer interacts with the corresponding Ad Hoc, IP, TCP, and UDP protocols. This layer senses packet types and traffic flows which can then be targeted by jamming.

The Application layer senses HTTP sessions, VoIP set up and the like and targets specific user activities for jamming. It also sets higher level policies that define when jamming should take place and what targets in the victim network should be jammed. Further, the goal may be purely to sense the kind of network activity.

Each of these layers contributes to the overall performance of the system so that each layer can provide its own contribution to jamming gain, targeted jamming, and low probability of detection.

This paper discusses exclusively the role of jamming at the Transport/Network layer. The Link/Physical layer provides a sensing and jamming service. The jamming service is defined as jamming for a specified period, jamming a specified number of packets, or to start jamming continuously until a stop jamming request is made. This protocol is described in more detail in Section 3. The sensing provides a report on each packet observed at the link layer. This report could conceivably include the following information:

Size: The physical layer could measure the transmission start and stop times or use other signal processing techniques to estimate the packet size in bytes.

Timing: Similarly the packet start time can be estimated.

Source Token: While the actual address of the transmitter source may not be known. Analysis of the transmitter signal could distinguish different transmitters so that each transmitter could be assigned a unique token.

Destination Token: As noted before, receiver ACKs can be identified in many protocols by the unique timing. Similarly by analysis of ACK transmitters the destination might also be identified.

Unicast vs. Broadcast: In many MAC and Link protocols, broadcast packets are not acknowledged while unicast packets are acknowledged. This could be used to identify whether a packet is unicast or broadcast.

While all of these are possible, only the first two Size and Timing are assumed available since these make the fewest assumptions about the underlying network.

The Transport/Network in turn provides jamming and sensing services to the higher layers. The jamming service can be as simple as to attack a target node at the greatest jamming gain possible while avoiding detection. The sensing service is to report on each packet seen adding to the Link/Physical layer attributes a broad packet classification into Data or Control and a narrow classification into specific Data (TCP or UDP) and Control (TCP-ACK, TCP-SYN, etc.) types.

It should be emphasized that this layered model applies to the particular type of external DoS attack that is the subject of this paper. As in the OSI model, the choice of layers is not absolute and different architectures might have greater or fewer layers. This layering provides the usual benefits of decomposing the problem into manageable modules that define layers in terms of services between layers and also by allowing a layer to be combined interchangeably with different layers. The modularity is in the sense that a single Transport/Network layer might be reused with many different Link/Physical layers to attack networks build on protocols such as 802.11a or 802.16.

1.2 Sensing & Jamming in Ad Hoc Networks

In network protocols, certain critical packets are necessary for operation. Jamming TCP-SYN, or TCP-SYN-ACK packets will prevent a TCP connection from being established. Jamming ARP-REQUEST or ARP-RESPONSE packets will prevent IP from associating IP and MAC addresses. Ad hoc networks add another protocol that can be attacked. Jamming AODV-RREQ or AODV-RREP packets will prevent ad hoc routes from ever being established. Jamming a few protocol control packets can prevent or delay connections; preventing the connection when the goal is to shut the connection down and delaying the connection when the goal is to inhibit communication without being detected.

As suggested from the above, knowing which packet to jam is the key to getting significant jamming gains. A sensor needs to identify the key control packets from different protocols. Ad hoc network protocols add additional packet types that can be detected. Sensing can be online or offline. In online sensing packets are identified as they are received. This can be difficult since in some cases a packet is identified within a protocol sequence that has not yet completed. Offline sensing is allowed to classify packets received in the past based on packets received both before and after the packet in question. These jamming and sensing ideas are explored more in a later section.

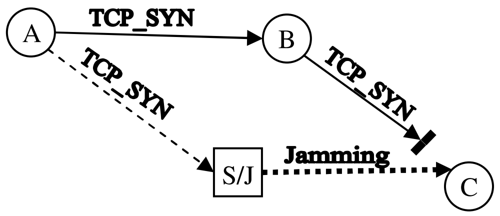

Figure 2: Exploiting multi-hop ad hoc routing. Ad hoc node A is communicating with C through B. The Sensor/Jammer identifies the target packet on the first hop and jams it on the second hop.

By the time a sensor classifies a packet it is too late to be jammed. Any jamming signal in response to the sensor classification would arrive after the packet is received by its intended receiver. This leads to the third role played by ad hoc networks. In a multi-hop path, a packet is transmitted and retransmitted several times. This provides an opportunity for a packet to be identified on one hop and jammed on the second hop. This idea is shown in Figure 2.

Finally, ad hoc networking could support a network of attackers sharing sensing information and jamming attacks. In this paper only a single attacker is considered

1.3 The Role of Encryption

MAC protocols can have various levels of encryption. 802.11 Wired Equivalent Privacy (WEP) and WiFi Protected Access (WPA) both are designed to protect the contents of the packet but not the control information in the MAC header [17]. Some implementations go further and also encrypt the entire MAC header [10]. In this paper, we assume that the entire packet is encrypted and only size and packet timing information can be measured. The main difference then is that encryption may change the packet size by an unknown amount and disrupt the Transport/Network layer sensing. The above encryption schemes add a fixed offset that, as we will see, do not impose serious difficulties on the sensing. Another type of encryption is exemplified by the 802.11i WPA2 protocol. This protocol uses a block encryption so that all packet sizes are rounded up to the nearest multiple of 128bits. This tends to reduce the fidelity of the sensing since similar size packets get clumped to the same size. It is assumed that none of these schemes has any significant effect on the timing of packets.

In the simulated and emulated experiments in this paper, no actual encryption takes place. The encryption is modeled as an offset to the size according to one of the above models and the packet size and timing information are passed to the Transport/Network layer sensor. The sensor is assumed not to know the encryption scheme and must adaptively estimate its effect.

1.4 Prior Work

Some attention has been given to attacks on the physical layer [16] of wireless networks. While much more consideration has been placed on attacks against the protocols that control these networks [7][8][9]. In [16], Stahlberg describes techniques to jam 802.11 networks by attacking the physical layer characteristics. Stahlberg describes jamming efficiency that can be attained by focusing jam efforts at specific transmission timeframes. He does not describe any intelligent methods of jamming a specific protocol nor does he mention any method of determining how jam periods are specified.

Negi and Perrig propose that an intelligent jammer could exploit MAC layer semantics to carry out jamming of specific MAC packet types [13] which they argue would cause a cascading effect due to the use of random back-off algorithms. Other papers propose attacks against the MAC and transport layers from the perspective of either a network participant [8][9] or as a node that creates pockets of congestion [7].

It should be noted that besides aiding jamming, sensing has other uses. Cryptanalysis attacks on encrypted data benefit from knowing the plaintext bits [3]. For known protocols, if packets can be identified then this allows bits such as the protocol value, version number, and length fields to be inferred. In some attack applications, the goal is to identify user activity. For instance, websites can be identified by the pattern of packets exchanged [4][18]. Traffic analysis can be used to attack user privacy [6][15]. The sensing described in this paper can provide more detailed pattern information that can refine such pattern and traffic analysis.

1.5 Paper Overview

Within the framework defined so far this paper provides seven contributions. First it demonstrates the potential Transport/Network layer jamming gains within a simulated environment. Second a simulated jamming protocol is developed that allows testing on an ad hoc network of lap top computers. Third the potential jamming gains are demonstrated on a live network using the simulated jamming protocol. Fourth a sensor is developed that uses packet size, timing, and sequence. It uses off-line sensing to adapt an online sensor to the current network conditions and a probabilistic model of the sizes and inter-packet timing of different packet types. A historical method for detecting known protocol sequences is used to develop the probabilistic models. The fifth is an active jamming mechanism to force the victim network to produce known sequences for the historical analyzer. The sixth is the online classifier that makes packet type classification decisions. The method is tested on live data and found that for many packet types the classification is highly reliable. Finally the relative roles of size, timing, and sequence are discussed along with the implications for making networks more secure.

2. POTENTIAL JAMMING GAINS

To see the potential for jamming we designed a simple modification to the network simulator, ns2, that enabled us to run jamming “recipes” that would jam specific packets. A typical recipe is shown in Figure 3. The goal is to slow the connection without causing the connection to fail. The TCP sender (left) has an established connection with the receiver (right). At time 305 sec, a 10 sec Jam signal causes the TCP window size to shrink to 1. Due to the TCP exponential back-off, the first TCP packet is seen 10 seconds after the noise signal. TCP forces an AODV route lookup. The attacker then jams 6 of the 7 RREP retries to obtain a 4 sec timing delay. Jamming the seventh would cause AODV to give up and alert the user, so the seventh is let through. The following TCP Ack is jammed to force the RTO to back-off further. This eventually triggers another AODV route lookup, and so on.

To put a number to the jamming gain, we use the following model. We assume that the cost of jamming a single packet is equivalent to 10msec of jamming. At the MAC layer a packet and any retries may need jammed and the 10msec represents the total of this effort. In reality the Link/Physical layer attacker may be more or less efficient than this, but this is a function of the Link/Physical layer jamming gain and outside the scope of this paper.

Applying this to simulated jamming attack, one cycle of AODV and TCP jamming consists of 7 jammed packets over 20 seconds. Each cycle admits one TCP-DATA packet, but, since it is never acknowledged the transfer never progresses. At 10msec per packet jammed, this implies that 70msec of jamming is equivalent to 20 seconds of continuous jamming. The net result is a sustainable jamming gain of 20sec/70msec = 286. This jamming gain is produced by a combination of jamming between multiple protocols. The simulator is just one implementation of these protocols and so we developed a test bed for simulating jamming recipes against protocols implemented in real networks.

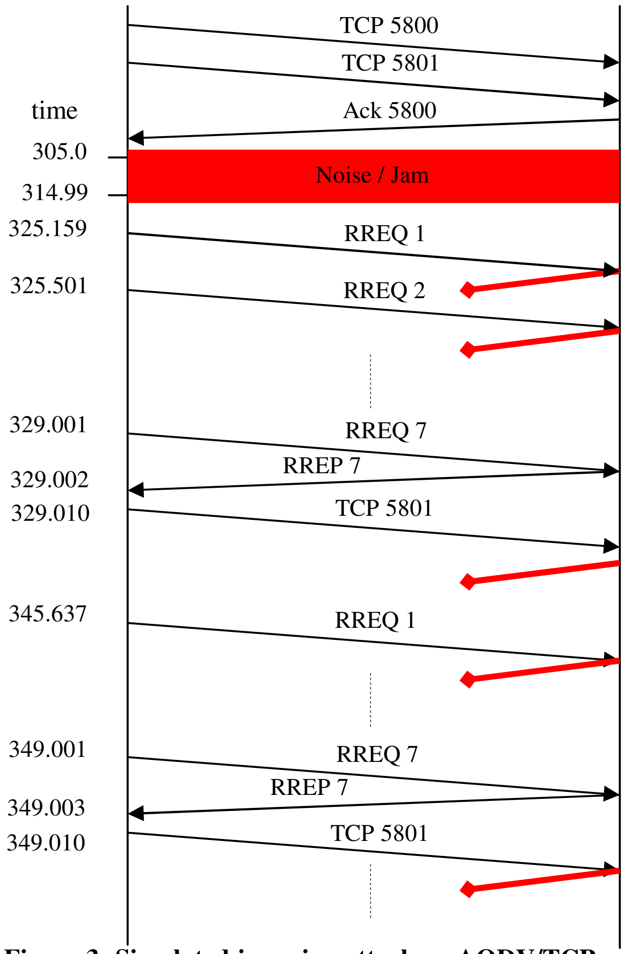

Figure 3: Simulated jamming attack on AODV/TCP

3. TEST BED

A test bed was constructed for testing the sensing and jamming. It consisted of Linux laptops [11] running the AODV-UU [14][19] protocol. The APE Mackill [2][20] allowed specific topologies to be set up on the desktop. The sensing and jamming was focused on the Transport/Network layer of this paper. For sensing the 802.11b the attacker used an Atheros 802.11 card in monitor mode. This passed all packets to the sensor with only the 14 byte Ethernet header. The Jamming used the so-called Simulated Jamming Protocol (SJP). Every victim node in the SJP filters all packets according to a signal sent by the attacker. The filter was written using the Click Modular Router [5]. When running in kernel level, the Click software assumes the operating system's role of packet receiver. When a packet enters through the wireless interface, it is given exclusively to the router software. The software then decides to either give it to the OS or to perform some act upon it. The architecture is shown in Figure 4. The Attacker sends jamming signal packets to a Jam Receiver module in the victim node. The packets are one of the following four instructions: Jam for a specified period of time; jam a specified number of packets; jam all packets indefinitely; or stop jamming. These instructions define jam periods. When a packet arrives over the wireless interface, the jam receiver either discards the packet or forwards it to the kernel depending on whether it is in a jam period or not. The attacker sends its instructions over a separate wired interface to avoid any contention on the wireless interface. The attacker can address the jamming to individual nodes or broadcast to all nodes.

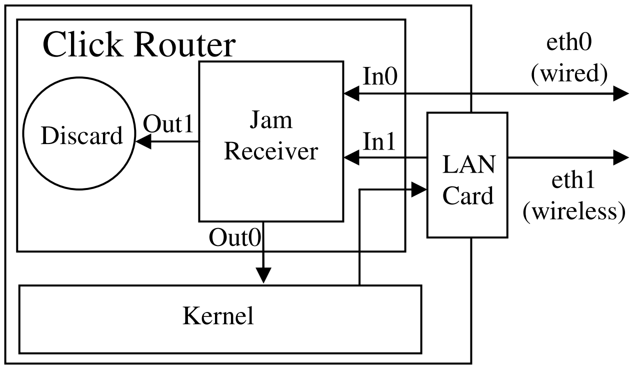

Figure 4: Simulated Jamming Protocol

We note that the emulation is not completely realistic. It does not model the interaction between jamming and the wireless transmitter's carrier sensing and MAC layer RTS/CTS/ACK packets. The SJP is designed to work above any MAC protocol and could be customized to interact with a specific MAC protocol to be more realistic.

The test bed used three configurations:

C1: The first is a pair of victim laptops one running the Apache server [1] and the other running a web browser application. The attacker node is placed so that it can receive the exchange of wireless packets between the nodes. A wired Ethernet hub connects the three computers so that SJP packets can be sent from the attacker to victim nodes.

C2: The second is identical to the first, except that the server node is connected to the Internet.

C3: The third is similar to Figure 5 with APE used to force a two hop path between nodes 1 and 4. The attacker node is placed so that it can hear traffic to and from node 4.

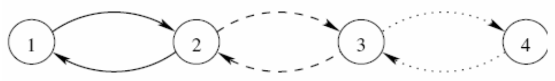

Figure 5: Multi-hop scenario simulated with Mackill tool of APE

4. TEST BED JAMMING GAINS

This section provides some insights into the potential jamming gains that are possible. The first results show what is possible when a TCP startup sequence is attacked.

Configuration C1 was used and the two 802.11 interfaces were in ad hoc mode. An HTTP connection was established between them to create a valid ARP table entry. The connection was terminated and then the client was jammed. Normally a series of UDP packets (DNS Lookup) followed by TCP-SYN, TCP-SYN-ACK, TCP-ACK are exchanged in the initial 3-way handshake. With jamming the client never receives the TCP-SYN. It retries four times with the result that it aborts the connection setup. Further the victim assumes something is wrong with the ARP table and it starts broadcasting ARP requests. The resulting times are shown in Table 1. The timing in this sequence is remarkably precise and was similarly precise across multiple runs which suggest that packet transmissions are predictable. As verification, the same experiment was replicated in Windows XP [12] with similar results except the time period between the 3rd and 4th TCP-SYN was 24 seconds. The predictable sequence timing suggests that precision jamming is possible. Using the model of 10msec of jamming per packet jammed, jamming the first TCP-SYN yields a 3 sec delay. The jamming gain is 3sec/10mses = 300. Similarly jamming the first and second yields a 9 sec delay and the jamming gain is 9sec/20msec = 450. Eventually TCP gives up and the jamming would need to jam one ARP-REQ per second (jamming gain is 100) to continue blocking the connection. Though not shown here, the timers on the TCP-SYN-ACK have similar backoff steps and yield similar delays. So, a more aggressive attack would jam the TCP-SYNs followed by the TCP-SYN-ACKs to yield a connection setup delay approaching one minute. This scenario shows that large jamming gains over 100 are easily obtained with Transport/Network layer jamming.

The next chart examines a similar scenario using AODV-UU for the routing in configuration C3. The attacker first jams five route request packets and then lets the sixth through to establish the route. Jamming the sixth would cause the connection to fail and notify the user. Next the TCP-SYN packets are jammed. The results are shown in Table 2. As can be seen, AODV-UU aggressively sends route request packets over the first 0.8 second. This time does not add to the delay to the subsequent TCP-SYNs which appear 3, 9, and 21 seconds after the start the same as in Table 1. Thus, the additional effort to jam the AODV-RREQ does not provide additional jamming gain.

This result and the simulation in Section 2 show that the attacker should directly jam TCP startups when possible or use a combination of AODV and TCP for an ongoing connection.

5. SENSING

The simulation and experimental results show that jamming has the potential for large gains, *if* the packet types are identified. This section describes the approach to sensing packet types. There are two approaches to classifying packets into types. The first classifies packets as they arrive (so-called *online* classification). The second is allowed to collect more observations before making the decision on packet type (so-called *offline* classification). Online classification is the preferred approach, but as will be shown in the following subsections, both online and offline classification have a role.

5.1 The Role of Size, Timing, and Sequence

The Link/Physical Layer reports on the timing and size of packets. These measurements do not necessarily need to be accurate, and the approach in this paper can detail with measurement variations, but for simplicity we will assume that they are reported without errors. We also assume that the lower layer reports all packets in the correct sequence. Though measured accurately, packet sizes vary across encryption as described in Section 1.3 and also because of protocol implementation variations, and variations in how

Table 1: Test bed TCP-SYN jamming gain

Packet Jammed	Total time (µsec)	Δt (µsec)	Cumulative Jam Gain
TCP-SYN	0	0	300
TCP-SYN	2991910	2991910	450
TCP-SYN	8991910	6000000	700
TCP-SYN	20991910	12000000	650
ARP-REQ	25991900	4999990	540
ARP-REQ	26991900	1000000	467
ARP-REQ	27991900	1000000	414
ARP-REQ	28991900	1000000	375
ARP-REQ	29991900	1000000	-

Table 2: Test bed AODV/TCP-SYN jamming gain

Packet Jammed	Total time (µsec)	Δt (µsec)	Cumulative Jam Gain
AODV-RREQ	0	0	1
AODV-RREQ	1350	1350	16
AODV-RREQ	323240	321890	11
AODV-RREQ	324440	1200	20
AODV-RREQ	813240	488800	16
TCP-SYN	819140	5900	50
TCP-SYN	2993010	2173870	129
TCP-SYN	9000120	6007110	262
TCP-SYN	21002540	12002420	-

the size might be reported at lower layers. Packet timing, and in particular, inter-packet spacing varies for the above reasons plus variations caused by network congestion. Protocol sequence does not vary (an ACK can only occur after a DATA packet) but multiple overlapping data streams require deconfliction. Therefore, the identification of packet types is statistical.

The sensor observes over time a sequence of packet sizes with known packet spacing. From this observation, O, it chooses the packet type T with the maximum a posteriori probability (MAP):

$$T = \arg\max_{T'} \{P(T' \mid O)\},$$

where $P(T|O)$ is the probability of packet type T given observation O. Using Bayes rule:

$$P(T \mid O) = \frac{P(O \mid T)P(T)}{P(O)}.$$

We note that $P(O)$ is independent of T, so that the MAP decision simplifies to

$$T = \arg\max_{T'} \{P(O \mid T')P(T')\}. \qquad (1)$$

So, classification requires an a priori estimate of the probability of each packet type and an estimate of the probability of an observation given each packet type. These are described in the next section.

5.2 Probabilistic Model of Size

What if our only observation is the size, S, of the current packet? How useful is size to determining packet type? Table 3 categorizes 945 packets captured between two laptops in configuration C1. It consisted of downloading a simple website with pictures and scp transfers.

Table 3: Distribution of packet sizes (bytes) in an ad hoc network.

Packet Type	# of Packets	Sizes utilized	$P(T)$
ARP-REQ	2	42	0.002
ARP-RESP	2	42	0.002
TCP-ACK	342	66	0.362
TCP-DATA	529	(all >74)	0.560
TCP-FIN	5	66	0.005
TCP-SYN	3	74	0.003
TCP-SYN-ACK	3	74	0.003
TCP-KEEP-ALIVE	12	60	0.013
AODV-RREQ	6	66	0.006
AODV-RREP(unicst)	6	62	0.006
AODV-RREP(brdcst)	35	62	0.037

Table 4: Distribution of Packet Sizes (bytes) across four different WLAN networks.

Packet Type	# of Packets	Sizes utilized	$P(T)$
ARP REQ	6	42(6)	0.003
ARP RESP	10	42(10)	0.005
TCP-ACK	714	54(113), 66(601)	0.360
TCP-DATA	1120	Various (95%>100)	0.565
TCP-FIN	39	66(39)	0.020
TCP-SYN	30	74(30)	0.015
TCP-SYN-ACK	31	58(4), 74(21), 78(6)	0.016
UDP-DATA	34	Various	0.017

Some sizes correspond to a unique type. Any packet over 74 bytes corresponds to TCP-DATA and 60 bytescorresponds to TCP-KEEP-ALIVE. For sizes, S, which are used by only one packet type T, $P(T|S) = 1$. The exceptions are 42 bytes (ARP-REQ and ARP-RESP), 66 bytes (TCP-ACK, TCP-FIN, and AODV-RREQ), 74 bytes (TCP_SYN_ACK and TCP_SYN), and 62 bytes (AODV-RREP(unicast), and AODV-RREP(broadcast))[2]. In these cases $P(T|S) < 1$ and the most likely packet given S is chosen (e.g. TCP-ACK). Using MAP classification on this data would yield 98% accuracy. Of course 92% of the packets are the easy to identify TCP-DATA and TCP-ACK. Results in Section 7 provide more detailed analysis.

Table 4 lists aggregate packet size statistics from four packet captures of a WLAN network[3], a total of 1984 packets sent between a client and server over a wireless link in configuration C2. The size variations in the third column for TCP-ACK and of TCP-SYN-ACK are from different packet captures on different networks. Within a capture they were a consistent size. Further we expect the encryption algorithms to add a consistent modification to each of these packet sizes. With the incorrect packet size model, the MAP classifier will not achieve high classification accuracy. The problem, then, is to develop a model for size that captures these variations initially and can adapt to the specific sizes present in the victim network. This model can be used in the MAP classifier. For this we use the Bayes equivalent MAP classifier in (1). The key to this approach is that $P(S|T)$ is independent of the other packet types and

[2] In AODV-UU, broadcast AODV-RREP are used as HELLO packets. The careful observation of network activity in this research has been an exercise in identifying anomalous non-standard network behavior across many protocols.

[3] An ordinary WLAN was used because it allowed different combinations of operating systems and servers.

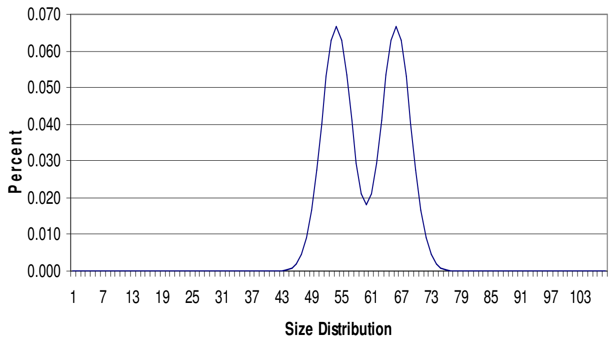

Figure 6: Initial size distribution of TCP_ACK packet.

so it allows independent estimation of the size distribution for each type. The coupling between types is given by the a priori type probability $P(T)$ given in the last column of Table 4. The data in the third column can be the basis of the initial $P(S|T)$. In order to capture the uncertainty the distribution is initially set to a broad distribution. Figure 6 shows an example for TCP-ACK.

The next section describes a method for getting samples of packet sizes for different traffic types. These samples are used to modify the distribution as follows. For each sample (S,T), we set $P(S|T) = P(S|T) + \varepsilon_T$ and then renormalize the distribution to have total probability 1. The constant ε_T sets the rate that the distribution adapts with new samples. The subscript indicates it can be different for different packet types. Generally, ε_T is larger for packet types that are more rare to speed their adaptation. It should be noted that when most sizes correspond to a unique packet type as is the case in Table 4 and would be true for any size transformation that is a simple additive offset, the distribution only needs to start to converge on the correct distribution for the MAP classifier to be correct. Thus, any reasonably accurate (S,T) samples will yield a high accuracy MAP classification. Some packet types, like TCP-DATA can be classified accurately with no training at all. Large packets are simply data. This long packet is data, short packet is control is the basis for bootstrapping identification of samples (S,T) as described in the next section.

5.3 Historical Analyzer

In order to derive samples of (S,T), we use the full size, timing, and sequence information over a historical packet window of W packets. Protocols introduce distinctive sequences, Q, such as data exchange (TCP-DATA, TCP-ACK) and TCP startup (TCP-SYN, TCP-SYN-ACK, TCP-ACK). A large packet followed shortly by small packet is likely a characteristic TCP DATA and ACK exchange.[4] An L packet sequence is defined by a structure $Q = (T_1, \tau_2, T_2, \ldots, \tau_L, T_L)$ where T_i is the ith packet in the sequence and τ_i is the distribution of time between packet T_{i-1} and T_i in the sequence. What is actually observed is $O = (S_1, t_2, S_2, \ldots, t_L, S_L)$ where S_i is the size of the ith packet and t_i is the time gap between adjacent packets.

The timing distributions are defined by a mean and standard deviation, (μ, σ). Given τ, the probability of inter-packet interval t is defined by a Gaussian distribution:

$$P(t \mid \tau) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2},$$

where $z = (t - \tau)/\sigma$ is the normalized Gaussian variable. Thus we compute the probability of a sequence Q given observation O as[5]

$$P(Q \mid O) = \prod_{i=1}^{L} P(t_i \mid \tau_i) P(S_i \mid T_i) P(T_i) K$$

where we define $P(t_1|\tau_1) = 1$ and $K = P(\tau_i)/P(t_i)P(S_i)$. The value of K requires distributions across all target sequences, inter-packet times and packet sizes. For simplicity at this stage, we use $K = 10$. Longer sequences multiply more probabilities together and thus tend to be smaller. This choice compensates for this effect with minimal assumptions about the network.

Each packet S_i, $i = 1,\ldots, W$ in the history window is a potential sequence starting point. The sequence Q that maximizes $P(Q|O)$ is computed and S_i is assigned type T_1 from the sequence Q. The goal of the historical analyzer is to provide accurate samples (S,T) for adapting the packet size distributions. In many cases even the maximum probability sequence Q is still unlikely. Thus, a minimum threshold θ is defined and the classified packet is not used unless $P(Q|O) > \theta$.

The history analyzer focuses on the sequences in Table 5. These sequences we refer to as lower protocol sequences since they depend on the communication transport and network layers. These can be composed into upper protocol sequences derived from application activity. For instance, an FTP session, consists of an AODV, ARP, DNS-lookup, and TCP-Startup sequence followed by multiple Data-Ack sequences. Recognizing these upper protocol sequences while potentially important would be in the attacker's Application layer and so outside the scope of this paper.

5.4 Online Classifier

[4] Recall that in our sensing model we can not distinguish different senders, so the sequence is large packet, small packet. If the senders can be distinguished then the sequence would be large packet from one device, small packet from a different device. For the purposes of this paper, we assume this extra fidelity is unavailable.

[5] This makes the independence assumption between the different sequence components.

Table 5: Types of sequence

Sequence Name	Packets in Sequence
Data-Ack	TCP-DATA, TCP-ACK
ARP	ARP-REQ, ARP-RESP
TCP-Startup	TCP-SYN, TCP-SYN-ACK, TCP-ACK
AODV	AODV-RREQ, AODV-RREP(unicast)
DNS-Lookup	UDP-DATA, UDP-DATA

The online classifier differs from the history analyzer in that it must make a packet classification on each packet as it arrives in order to feed information to the jamming module in a timely manner. In this scenario, a sequence may or may not be helpful depending on where a packet type appears in a sequence. The TCP-Startup sequence is not useful for online classification of TCP-SYN packets. $L = 1$ is a valid sequence and equates to classifying a single packet by size. Each sequence with $L > 1$ implicitly defines initial subsequences by using less than L packets. Thus the online classifier uses whatever sequence or sequence fragments it can in order to classify the current packet.

In this paper, we simplify the use of sequence by limiting the activity to one connection at a time. A large number of co-mingled connections would produce sequences with other packets in between. In principle the search for sequence matches can be extended to skip over intermediate packets. Future work will investigate this more fully. Another alternative that was explored was to use jamming to “reset” the network and simplify the traffic seen by the attacker. This is one of several potential uses of jamming in sensing discussed in the next section.

6. JAMMING FOR ACTIVE SENSING

Jamming has been discussed solely as a mechanism for disrupting the victim network. Sensing has been discussed as a passive observation activity. But, jamming can play a role in sensing. We describe three roles: discerning different traffic types, resetting the network, forcing the victim to produce certain packet types.

Different protocols react differently to lost packets. In this way, jamming provides one method for distinguishing protocols. TCP will back off if packets are lost, while UDP will continue to push out packets. Experiments were carried out that demonstrated this phenomenon. The protocols do react as expected to a burst of jamming, TCP throughput temporarily dips when jammed while UDP does not react. This response was found to be more difficult to classify than passively sensing the Data-Ack sequence of TCP. This approach would be worth pursuing in special cases such as distinguishing UDP and TCP streams that consist of small data packets similar in size to the TCP-ACK.

When an attacker turns on, the victim network may have many connections in progress that impedes sensing. Further, the vulnerable connection setup phase has passed. A second concept that was tested was to jam until connections fail so that the sensor has more distinct flows to work with and jamming could be targeted. To test this, we used configuration C1 and started a large HTTP download. The client was then jammed for increasing periods of time and then observed to see if the TCP connection continued or TCP had capitulated. Through this testing it was determined that the Apache Web server would hang its connection after about 18 seconds of jamming. After this failure, the user would likely retry the link starting a new connection setup that could be delayed arbitrarily through jamming.[6] An 18 second burst of jamming in order to take control of the victim network is a reasonable tradeoff.

To confirm whether this was a general result or unique to Apache, we attempted the same experiment with configuration C2 accessing large data files from common public sites. In some cases the behavior was similar to Apache, in others such as www.google.com, the TCP session persisted after more than 10 minutes of jamming. This is detrimental to getting good jamming gains, but useful for low probability of detection since it shows that long jamming periods may never result in a user notification.

Sensing requires examples of each packet type in order to adapt. Certain packet types such as ARP packets are rare. A third use of jamming in sensing is to force the victim network to produce rare packet types. A naïve approach would be to jam a network for the 5-20minutes it takes for the ARP cache to timeout. A more practical approach is required that does not entail extensive jamming periods. As noted in Section 4 a failure to establish a TCP connection results in a series of ARP requests. This requires far less jamming to produce.

The results in this session suggest that jamming may have a role in sensing. Further work is required to formalize what is possible.

7. TEST BED SENSING

To test the sensing we use configuration C2 between two laptops with an 802.11 interface in ad hoc mode. The experiment is to show how the sensing performs classifying ARP, TCP, and UDP packets in three scenarios since these packets are part of the attack with the highest jamming gain. In each scenario, a monitor records each packet as it is received and reports the timing and a size that depends on

[6] If this is performed only once at attacker power up it may not alert the victim to the attacker’s presence.

the encryption model. The scenarios differ in the encryption model:

Scenario S1: reports the actual packet size.

Scenario S2: reports the packet size with an offset of 10 bytes.

Scenario S3: reports the packet size padded to the next nearest multiple of 16 bytes.

In each scenario two sensors are applied. The first sensor only uses size in estimating the packet and does not use timing or sequence nor does it use the historical analyzer to adapt the size distribution. The other sensor is the adaptive sensor defined in this paper. The adaptive sensor first observes 2000 packets to adapt the size distributions using the historical analyzer and then proceeds to the online classification based on sequence and size. The results are shown in Figure 7 tabulated in a so called confusion matrix $\{c(T,T')\}$ that counts for each packet type, T, how often it was classified into packet type T'. An ideal classifier would have $c(T,T') = 0$ except on the diagonal when $T = T'$. Packets on the highlighted diagonal are correctly classified. Since these were based on actual traffic captures, the number of packets in each confusion matrix is not the same. But, the performance is still comparable. In (a), using only size the classifier correctly classifies ARP-REQ, but the ARP-RESP are incorrectly classified also as ARP-REQ. Both of these packets are the same size and so there is no way to distinguish the two packet types based on size. A similar phenomenon occurs between TCP-FIN and TCP-ACK and between TCP-SYN and TCP-SYN-ACK. When an offset is introduced, as in (c), the size-only classifier makes many more mistakes. When the padding is introduced, as in (e), the classifier is unusable for jamming.

The performance when the adaptive algorithm is applied is shown in Figure 7b. Compared to (a), the classifier can distinguish correctly between ARP-REQ and ARP-RESP, and between TCP-SYN and TCP-SYN-ACK. TCP-FIN is incorrectly classified, but, this is expected because no sequence for TCP-FIN was defined that can be used to adapt its distribution. In (d) the results are equally good. A simple offset does not change the ability to distinguish between different packet types. In (f) the padding clumps all the TCP control packets to the same size. The TCP-ACK packet has a much higher prior distribution compared to other control packet so that sequence can not separate out the other control packets as it has been implemented here. Further work is investigating this issue.

8. LESSONS LEARNED: MAKING NETWORKS MORE SECURE

The attacks in this paper are based on carefully exploiting protocol patterns and consistencies across size, timing and sequence. This suggests that to make networks more secure these consistencies should be removed wherever possible. For size, padding control packets so that they are all the same size will make it difficult to discern different packet types. Padding all packets including control so that they have the same minimum size (say 100 bytes) will further remove size as useful metric. For wireless MAC protocols such as 802.11, every packet has substantial overhead so that small packets already consist mostly of this overhead. Additional padding will have minimum effect on throughputs.

Timing in these protocols is overly precise. In TCP, the receiver does not use the three second back off time between the first and second TCP-SYN. Indeed, if the first one has been jammed it is not even expecting the second. Similarly, the precise timing between many packets in the sequence can be varied by significant factors so that it is difficult to precisely jam the packets. The timing of some packets such as TCP-ACKs is used by protocols for estimating aspects of the network. But, it is conceivable that these protocols could be modified to allow for added delays. For instance, the header could indicate any additional delay that was added for security reasons so that this could be factored into RTT calculations.

Sequence for the protocols is immutable. But, it also can be foiled. One approach is to aggregate multiple packets. This will affect both timing and size of packets as well as potentially hiding the precise number of packets that are exchanged. Another attack is what we refer to as the zebra defense in which a single connection is striped across multiple TCP connections so that the attacker has difficulty separating and attacking individual victim connections.

9. CONCLUSIONS

Jamming and sensing are two related functions in physical-layer-based denial of service attacks against an encrypted wireless ad hoc networks. These functions are complex and the layered approach developed in this paper showed how they could be broken down into a manageable design problem. This paper presented initial results in designing such a layered attacker for the Transport/Network layer. Jamming can get significant jamming gains, well over 100, when it knows the packet type and timing. Interestingly most of these gains were produced by attacking packets above the ad hoc network layer. Protocols introduce highly predictable timing that can be exploited. The limited information of packet size, timing, and sequence is enough to accurately predict packet types. Using a combination of offline historical analysis of sequence to provide training data for the online models, a packet classifier was developed that adapts to variations across networks and across different encryption models. The development in this paper suggests simple methods for making victim networks less vulnerable to these kinds of attacks. That said, wireless TCP/IP based networks are ubiquitous and a complete legacy backhaul is unlikely leaving a significant number of vulnerable networks.

The research presented here is ongoing. Future work will fully connect and test the jamming and sensing which were treated separately. The statistical sensing tools continue to be refined. A few representative attacks were presented and the test bed tools described here are being used to methodically evaluate other attacks. Scaling to larger ad hoc networks and networked attackers is the long term goal.

10. REFERENCES

[1] The Apache HTTP Server Project, release 2.0, downloaded Sep. 2004. http://httpd.apache.org/

[2] APE Project, *How to build, install and run the APE testbed*, Uppsala University, Nov. 8, 2002 http://apetestbed.sourceforge.net/ape-testbed.pdf

[3] Bellovin, S.M., Probable plaintext cryptanalysis of the IP security protocols, In *Proc. 1997 Symposium on Network and Distributed System Security.* Feb. 1997 pp. 52–59

[4] Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N., Privacy Vulnerabilities in Encrypted HTTP Streams, In *Proc. Privacy Enhancing Technologies Workshop* (PET 2005).

[5] Click Modular Router Project, MIT, release 1.4.3, downloaded Dec. 2004 http://pdos.csail.mit.edu/click/

[6] Fu, X., Graham, B., Bettati, R., Zhao, W. Active Traffic Analysis Attacks and Countermeasures. In *Proc. of the 2003 International Conference on Computer Networks and Mobile Computing*, 2003.

[7] Gupta, V., Krishnamurthy, S., Faloutsos, M. Denial of Service Attacks at the MAC Layer in Wireless Ad Hoc Networks. In *Proc. of Milcom*, 2002.

[8] Hu, Y.-C., Perrig, A. A survey of secure wireless ad hoc routing. *IEEE Security & Privacy Magazine*. v. 02, n. 3, (May–Jun. 2004), pp. 28–39.

[9] Joncheray, L. A Simple Active Attack Against TCP. In *Proc. Fifth Usenix UNIX Security Symposium*, 1995

[10] Landeta, D., *Secure Wireless LAN SecNet 11 & SecNet 54*, in Information Assurance Solutions Working Symposium, Aug. 2005. See also, http://www.govcomm.harris.com/secure-comm/

[11] Linux, *The linux homepage*, the 2.4.27 kernel, downloaded Nov. 2005, http://www.linux.org

[12] Microsoft Corporation, Microsoft Windows XP Home Edition Version 2002 Service Pack 2.

[13] Negi, R., Perrig, A. *Jamming analysis of MAC protocols.* Carnegie Mellon Technical Memo, 2003.

[14] Perkins, C., Royer, E., Das, S., *Ad hoc On-demand Distance Vector (AODV) Routing*, Internet Draft, draft-ietf-manet-aodv-11.txt, *work in progress*, Aug 2002.

[15] Raymond, J. Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems. In H. Federrath, ed., *Designing Privacy Enhancing Technologies*, v. 2009 of LNCS, pp. 10–29. Springer-Verlag, 2001

[16] Stahlberg, M.. Radio jamming attacks against two popular mobile networks. In H. Lipmaa and H. Pehu-Lehtonen, ed., *Proc. of the Helsinki University of Technology Seminar on Network Security.* Fall 2000.

[17] Stallings, W., *Wireless Communications and Networks*, 2nd Ed., Prentice Hall, 2005.

[18] Sun, Q., Simon, D.R., Wang, Y., Russell, W., Padmanabhan, V.N., Qiu, L., Statistical identification of encrypted web browsing traffic. *IEEE Symposium on Security and Privacy*, 2002.

[19] Uppsala University, *The AODV-UU implementation* , version 0.8.1, downloaded Nov. 2005 http://core.it.uu.se/AdHoc/AodvUUImpl

[20] Uppsala University, The Ad hoc Protocol Evaluation (APE) testbed, release 0.3, downloaded Nov. 2005 http://apetestbed.sourceforge.net

Classified Packets		Actual Packets						
		ARP		TCP				
		RESP	REQ	FIN	ACK	SYN	SYN ACK	DATA
TCP	DATA			5				158
	SYNACK				1		5	
	SYN					20	14	4
	ACK			26	161			1
	FIN							
ARP	REQ	2	2					
	RESP							

(a) Actual packet size, size-only sensor

Classified Packets		Actual Packets						
		ARP		TCP				
		RESP	REQ	FIN	ACK	SYN	SYN ACK	DATA
TCP	DATA			28				963
	SYNACK						64	
	SYN					60		
	ACK			66	876	2		
	FIN							
ARP	REQ		2					
	RESP	2						

(b) Actual packet size, adaptive sensor

Classified Packets		Actual Packets						
		ARP		TCP				
		RESP	REQ	FIN	ACK	SYN	SYN ACK	DATA
TCP	DATA			4				251
	SYNACK						6	
	SYN			28	138	23	20	4
	ACK	2	2	8	118		3	2
	FIN							
ARP	REQ							
	RESP							

(c) 10 byte size offset, size-only sensor

Classified Packets		Actual Packets						
		ARP		TCP				
		RESP	REQ	FIN	ACK	SYN	SYN ACK	DATA
TCP	DATA			10				422
	SYNACK						34	
	SYN					30		
	ACK			35	436			
	FIN							
ARP	REQ		6					
	RESP	6						

(d) 10 byte size offset, adaptive sensor

Classified Packets		Actual Packets						
		ARP		TCP				
		RESP	REQ	FIN	ACK	SYN	SYN ACK	DATA
TCP	DATA			28				212
	SYNACK							
	SYN	1	1					
	ACK			17	150	15	15	3
	FIN							
ARP	REQ							
	RESP							

(e) Padding to nearest 16 bytes, size-only sensor

Classified Packets		Actual Packets						
		ARP		TCP				
		RESP	REQ	FIN	ACK	SYN	SYN ACK	DATA
TCP	DATA			9				274
	SYNACK							
	SYN			2				
	ACK			25	283	23	23	
	FIN							
ARP	REQ		4					
	RESP	4						

(f) Padding to nearest 16 bytes, adaptive sensor

Figure 7: Test bed results for sensing packets with a size only sensor or the adaptive sensor. The results on the shaded diagonal are the number of packets of the associated type classified correctly. The off-diagonal counts incorrectly classified packets with the column indicating the true packet type and the row the incorrect classification.