ON JAMMING IN WIRELESS LOCAL AREA NETWORKS

by

LIYI GU

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

May 2014, Hong Kong

Copyright © by Liyi Gu 2014
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

[Signature]
LIYI GU
ON JAMMING IN WIRELESS LOCAL AREA NETWORKS

by

LIYI GU

This is to certify that I have examined the above M.Phil. thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

PROF. BRAHIM BENSAOU, THESIS SUPERVISOR

PROF. SIU-WING CHENG, ACTING HEAD OF DEPARTMENT

Department of Computer Science and Engineering

13 May 2014
ACKNOWLEDGMENTS

First, I would like to express my deep gratitude to my supervisor, Prof. Brahim Bensaou for his continuous support, kind encouragement and useful advice throughout my MPhil study. From him I have learned the way of reading, thinking, developing ideas, writing and overall, researching. I would like to thank Prof. Sunil Arya and Prof. Mordecai Golin for their help and advice in my future career. Special thanks go to my thesis committee members, Prof. Jogesh Muppala and Prof. Kai Chen for their insightful comments on this work.

I would also like to thank my group members, Jun Zhang, Min Wang, Xiangming Dai, Ying Wang, Amuda James Abu and Ahmed Mohamed Abdelmoniem for their help and opinions on my research. I thank my friends, Jianchang Hu, Mao Mao, Langhuan Pei, Naiyan Wang and many others for making these years a colorful period in my life.

Last but not the least, I cannot say enough thanks to my parents, without whom I could never come to this day.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>i</td>
</tr>
<tr>
<td>Authorization Page</td>
<td>ii</td>
</tr>
<tr>
<td>Signature Page</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>Abstract</td>
<td>x</td>
</tr>
<tr>
<td>Chapter 1  Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Overview</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Thesis Outline</td>
<td>4</td>
</tr>
<tr>
<td>Chapter 2  Related Work</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Jamming attacks</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Detection schemes</td>
<td>8</td>
</tr>
<tr>
<td>Chapter 3  A Stealth Greedy Attack on TCP Traffic in Wireless Local Area Networks</td>
<td>11</td>
</tr>
</tbody>
</table>
3.1 Overview

3.2 Settings and assumptions

3.3 Jamming preparations

3.4 Jamming algorithm
   3.4.1 MAC-ACK to the target
   3.4.2 Jammer’s MAC-ACK times out
   3.4.3 Expiry of the period timer

3.5 Probabilistic model
   3.5.1 Relationship between $p(x)$ and $q(x)$
   3.5.2 Calculation of $p(x)$

3.6 Other issues
   3.6.1 Use of RTS/CTS
   3.6.2 Jamming multiple targets
   3.6.3 Tracing the target’s queue length
   3.6.4 TCP delayed ACK mechanism

3.7 Immunity to detection
   3.7.1 Estimated backoff window
   3.7.2 RTT
   3.7.3 Number of collisions

Chapter 4 Evaluation

4.1 Evaluation settings

4.2 A case in detail

4.3 Overall effect of jamming
   4.3.1 Throughput
   4.3.2 Estimated backoff window
4.3.3 RTT 37
4.3.4 Number of collisions 38

Chapter 5 Conclusion and Future Work 39

5.1 Conclusion 39
5.2 Future work 40

Bibliography 42
# LIST OF FIGURES

1.1 Attack illustration  
2.1 Architecture of DOMINO  
3.1 Layout of the wireless LAN; inner circles are nodes' transmission ranges  
4.1 Throughput of jammer and target nodes, 1st run  
4.2 TCP sender’s window of jammer and target nodes, 1st run  
4.3 Samples of round-trip time of jammer and target nodes, 1st run  
4.4 Throughput of jammer and target nodes, 2nd run  
4.5 TCP sender’s window of jammer and target nodes, 2nd run  
4.6 Samples of round-trip time of jammer and target nodes, 2nd run  
4.7 Jammer’s throughput comparison: for different $q(x)$  
4.8 Jammer’s throughput after taking the outlier cases out
LIST OF TABLES

2.1 Summary of disruptive jamming models: C = Implementation Complexity, E = Energy Efficiency, S = Stealthiness, D = Level of DoS, R = Anti-Jamming Resistance 7

4.1 Simulation parameters 29

4.2 Average throughput simulation results 36

4.3 Estimated backoff window in slots: A=Jammer Backoff window; B=Nominal (AP) backoff window 37

4.4 Average RTT simulation results 37

4.5 Number of collisions α simulation results 38
ON JAMMING IN WIRELESS LOCAL AREA NETWORKS

by

LIYI GU

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

The shared nature of the medium in wireless networks makes them susceptible to all sorts of careless or malicious misbehaviour. Among them, jamming, which involves purposefully trying to interfere with the transmission and reception, is one of the major forms of attacks that can easily be staged in wireless local area networks (WLANs).

In this thesis we propose a new jamming attack on hidden nodes in WLANs. This attack deliberately targets uplink TCP acknowledgement packets (TCP-ACKs) of some downlink TCP flow as a means of increasing the throughput of the attacker(s). The jamming attack is designed in such a way that no modification to the wireless hardware is needed and it can be easily staged on commercial off-the-shelf wireless nodes. The attack consists of a series of well-coordinated stealth attacks on TCP traffic with the collective effect of degrading the target’s throughput by causing occasional time-outs and by increasing its round-trip times (RTTs). In this attack, a rogue node first scans the channel for downlink traffic (from the AP), identifies one or several target nodes, then
relies on our probabilistic estimation model to forecast the time when a transmission of jamming signal has a high likelihood of colliding with the target’s generated TCP-ACKs. Repeating this process intelligently results in a decrease of the average window of the targeted TCP sender and an increase in its round-trip time, in addition to some occasional time-outs due to consecutive TCP-ACK losses. The rogue node and/or its colluding attackers can thus increase their TCP throughput. We conduct ns-2 protocol simulations to demonstrate the effectiveness of such attack and discuss its immunity to detection by existing detection schemes. We also identify some possible parameters that may be used in building future detection mechanisms to counter this attack.
CHAPTER 1

INTRODUCTION

1.1 Background

The shared nature of the medium in wireless networks makes them susceptible to all sorts of careless or malicious misbehaviour. Yet current wireless protocols such as the IEEE802.11 do not have any built-in detection or prevention mechanisms to deal with these aspects. Just like the internet, security has come to wireless as an afterthought. With the proliferation of wireless local area networks (WLANs) and their increased popularity in commercial enterprises, security has become of utmost importance in recent years. Over the years, various forms of attacks have come into being, ranging from the most basic denial-of-service (DoS) attacks of flooding the wireless channel with jamming signals, to the more intelligent and stealthy ways of selectively corrupting specific packets after monitoring the channel. As a response, more and more methods of detecting such jamming attacks are being developed and deployed to address these threats.

1.2 Motivation

In order to enhance network security and be prepared for new forms of attack, it is of great significance to build the attacks in advance. It is well known (Blinn et al., 2005; Afanasyev et al., 2010) that TCP downloads dominate today’s WLAN traffic thanks to the adoption of progressive HTTP as a video streaming mechanism (YouTube, Netflix). Therefore, an attack on TCP traffic could potentially yield some good reward for the attacker in any normal network. Furthermore, TCP is known for its self-clocking and its supposed fairness or lack of it (when parameters such as RTT and TCP flavours differ). Therefore,
targeting TCP-ACKs successfully may result occasionally in some time-outs and most frequently in an increase of the RTT of the target, leading to a reduced throughput. It is also shown in (Proano & Lazos, 2010) that in an exposed scenario, a disruptive jamming attack on TCP-ACKs can significantly decrease the throughput of the TCP. This gives us reason to believe a similar attack on TCP-ACKs on a hidden node could free up the bandwidth for the jammer to monopolize. Meanwhile, staging an attack that does not require any modification to the hardware could make the attack accessible to ordinary users making it more commonly deployable. As such, a jamming attack in WLAN that targets the TCP traffic of nodes hidden from the attacker could be a good option, since jamming exposed nodes requires breaching the rules of the built-in CSMA mechanism and thus needs non-trivial modification to the hardware. Also, an ordinary user normally is more willing to gain benefit from misbehavior than to simply disrupt the network, so we look at designing a greedy jamming attack which increases the jammer’s own download throughput rather than a disruptive one. Consider a daily scenario where a person comes to a public place, like a coffee shop, and starts a download, but is not satisfied with the download speed. She could then turn on the jammer in her own machine, the jammer searches for appropriate hidden nodes that are taking up much bandwidth and jams them. The ease of staging such attack and its versatility could pose a potentially huge threat to WLANs especially with the increasing popularity of such networks and the proliferation of so-called personal hotspots using smart-phones. Our goal in this thesis is to design such attack to demonstrate its feasibility, and by the same way inseminate the area of research on countering such attack by discussing some possible parameters that can be used to detect such attack.

1.3 Overview

In this thesis, we first make a review on the various work previously done on jamming in wireless LANs. Then we focus on constructing a greedy selective jamming attack
model that targets downlink TCP traffic of some node(s) in a WLAN. Although jamming exposed nodes’ frames is more accurate, it requires some non-trivial modifications to the CSMA mechanism and therefore would require some custom-built hardware. As a result, we focus in particular on targeting hidden nodes because of the relative simplicity of staging such attacks with commercial off-the-shelf hardware without modifying the MAC protocol. For the same reason, our jammer does not jam the actual downlink TCP-data from the AP nor the corresponding MAC-ACK frames to the AP (see Fig. 1.1) as this can only be achieved by modifying the hardware to enable custom inter-frame spaces (IFS). Therefore in the basic access mode, TCP-ACK frames from a hidden target node to the AP turn out to be a very good candidate, since the jammer is no longer bound by the MAC IFS with respect to the target’s uplink transmissions. Nevertheless, since the target is now hidden from the jammer, this latter can no longer exploit the listening capability to synchronize its jamming signals with the target transmissions. Furthermore, random jamming fails to achieve one of the fundamental requirements of the greedy attacker – viz. increasing the throughput of the attacker and may fail to remain stealth. To tackle this problem and reduce the frequency of jamming, our jammer makes jamming decisions based on our analytical estimation model that intends to match sporadic attacks with the targets’ transmission times with a high probability.

Figure 1.1: Attack illustration
1.4 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 first introduces some background knowledge, including a brief literature review of jamming attacks in WLANs, as well as methods developed to detect the attack. Chapter 3 presents our new jamming attack, including the attack model, specifications and calculations. In Chapter 4 we exhibit the simulation results on the effectiveness and stealthiness of our jamming attack. We finally conclude the thesis and propose some potential directions for future work in Chapter 5.
CHAPTER 2

RELATED WORK

2.1 Jamming attacks

Xu et al. (Xu et al., 2005) define the jammer as “an entity who is purposefully trying to interfere with the physical transmission and reception of wireless communications”. Based on their intention, jamming attacks can be classified into two categories: disruptive jammers (Bellardo & Savage, 2003), who intend simply to disrupt the network from normal functioning, and greedy jammers (Kyasanur & Vaidya, 2005) whose goal is to profit from their actions by for example grabbing bandwidth from other nodes.

Disruptive jamming attacks share the idea of occupying the channel and corrupting any packet transmitted along with their jamming signals. Basic disruptive jammers, despite its name, may take various forms of different simplicity and effectiveness, exemplified by the following four jamming models (Xu et al., 2005). The simplest basic disruptive attack may be staged by the so-called constant jammer who transmits radio signals continually over the medium, leading either to the sender deferring its transmission on sensing a busy channel or to the receiver being unable to decode the interfered data. Such attack is effective and easy to stage, albeit, it is also very easy to detect. Variations on the constant jammer can make it harder to detect: for example, instead of churning out random bits, the deceptive jammer sends out regular packets continually to deceive the receiver; the random jammer alternates between sleeping and jamming; while the reactive jammer listens to the channel and starts transmitting signals as soon as it senses channel activity. Such naive disruptive attacks can be easily implemented, but can also be combated by simple detection methods such as measuring signal strength, carrier sensing time, etc.
A reactive jammer can also be made to jam on a probabilistic basis, using less energy and becoming harder to detect, but at the same time still yielding good jamming effect (Chinta et al., 2009).

In contrast to the basic jammers which employ physical layer jamming, some more “intelligent” jammers may exploit the higher layer protocol’s design. Depending on the targeted packet, the jammer could perform a CTS Corruption Jamming which waits for SIFS after a transmission of RTS packet, an ACK Corruption Jamming which waits for SIFS after a transmission of DATA packet, or a DATA Corruption Jamming which waits for DIFS after a transmission of CTS packet, and these attacks are shown to be effective in terms of the level of DoS (Acharya & Thuente, 2005). The disruptive corruption attack on TCP traffic that a specific packet of TCP traffic is corrupted is also effective, especially when targeting TCP acknowledgement packets (Proano & Lazos, 2010). The jammer could also end the wait for interframe spaces preemptively to monopolize the channel (Bellardo & Savage, 2003) Such jammers, however, require drastic modifications to their wireless hardware, rendering the attacks less easy to stage. Alternatively, the identity attacks (Bellardo & Savage, 2003), which involves the attacker spoofing messages of the target node in order to cause a deauthentication, disassociation or entrance into sleeping mode on the target node, may be carried out by the jammers. Table 2.1 summarizes the various disruptive jamming models above (Pelechrinis et al., 2011).

Greedy attackers, on the other hand, try to increase their own throughput share via misbehaviour. There are a number of ways for the rogue node to gain more throughput depending on the flaw in 802.11 it takes advantage of. For example, for uplink traffic it could scramble the CTS, ACK and DATA frames like in the disruptive attacks (Raya et al., 2006). In this way the target node’s congestion window increases and consequently its throughput decreases, giving more bandwidth to the attacking node. The attacker could also change the duration value set in the frame header of RTS and DATA frames to increase the length of the busy period for other nodes indicated by the NAV (Bellardo &
<table>
<thead>
<tr>
<th>Jamming Model</th>
<th>C</th>
<th>E</th>
<th>S</th>
<th>D</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Deceptive</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Random</td>
<td>Low</td>
<td>Adjustable</td>
<td>Medium</td>
<td>Adjustable</td>
<td>Medium</td>
</tr>
<tr>
<td>Reactive</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Packet Corruption</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Identity Attacks</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of disruptive jamming models:

C = Implementation Complexity,
E = Energy Efficiency,
S = Stealthiness,
D = Level of DoS,
R = Anti-Jamming Resistance

Savage, 2003). The clear channel assessment (CCA) threshold may be altered upwards as well so that the attacker ignore signals from other nodes and count down its backoff faster. Hence the attacker will have more access to the channel and thus higher throughput (Pelechrinis et al., 2009). Backoff misbehavior, one of the simplest and most effective attacks yet hardest to detect (Lu et al., 2010), is related to the backoff mechanism of 802.11: the attacker reduces its random backoff time to occupy the channel faster than other normal nodes, leading to more frequent access to the channel (Kyasanur & Vaidya, 2003). In the case of downlink traffic, the greedy node can either jam TCP-ACKs from the target to the AP, so that they never reach the sender, which decreases its sending rate, or jam the data frames from the AP to the target node, and forge MAC-ACKs to prevent the AP from retransmitting data, leading the sender to decreasing its rate (Raya et al., 2006). Most of these jamming techniques also require changes to the hardware.
2.2 Detection schemes

Many detection schemes have been proposed to address user misbehavior in IEEE 802.11 networks. The most well-known greedy attack detection system is DOMINO (Detection Of greedy behavior in the MAC layer of IEEE 802.11 public NetwOrks) (Raya et al., 2006). The main advantages of DOMINO are its simplicity and easiness of implementation, that it covers a wide range of misbehaviors and does not need modifications to the MAC protocol. DOMINO is implemented in the AP which is assumed trustworthy, and adopts a three-layer architecture (Fig. 2.1). The first layer, or “module” as it is called, collects traffic traces in one monitoring period and passes them to the second module. DOMINO takes into account four different misbehaviors in 802.11 MAC layer, i.e. scrambled frames, shorter than DIFS, oversized NAV and backoff manipulation, and devises six tests to detect the misbehaviors. The tests are simple comparisons between the average of observed values and the threshold value. The results of the tests are sent to the third module, or the decision making component, where they are aggregated in the aggregation component and analyzed in the behavior classification component. The system informs the operator if the last module classifies the node as misbehaving.

Due to the random nature of the backoff time selection in 802.11, the backoff misbehavior is among the hardest to detect and thus has been the attention of much literature. Kyasanur et al. proposed a modification to the 802.11 protocol to detect backoff misbehavior (Kyasanur & Vaidya, 2005). In their scheme, rather than the sender chooses its backoff timer, the receiver assigns a backoff timer for the sender instead. If the time between consecutive transmissions from the sender does not match the assigned backoff timer, the receiver deems that the sender is potentially misbehaving with its backoff time and will assign a larger backoff to the sender for the next transmission. Repeated deviations will cause the receiver to diagnose the sender as greedy node. However, the modification on the protocol, the assumption that the receiver is always trusted and the vulnerability against colluding attackers are the limitations to this detection scheme. The
second point is alleviated by the work of Cardenas et al. (Cardenas et al., 2004) in which they presented a modification to the previous scheme. In their improved scheme called “ERA-802.11”, the sender and receiver perform a series of cryptographical message exchanging to negotiate a truly random backoff timer for the sender. In this way the sender can now make sure the receiver is not a misbehaving node by deliberately assigning large backoff timers. A similar idea based on modification to 802.11 is proposed in (Guang et al., 2006), where a new access method, Predictable Random Backoff (PRB), is presented. PRB forces each node to generate a predictable backoff timer, so that misbehavior is easily detected.

A wide range of statistical techniques are applied on the detection of backoff misbehavior. The Kolmogorov-Smirnov (K-S) test is used in (Toledo & Wang, 2007; Lopez Toledo & Wang, 2007) to test whether the distribution of sampled traffic pattern is based on the same underlying normal distribution of traffic data or not. The sequential probability
ratio test (SPRT) is used in (Radosavac et al., 2005, 2008; Rong et al., 2006), where misbehavior is detected if a random walk of the likelihood ratio of observations exceeds the threshold. These studies base the observed backoff timer on the successfully received frames at the AP and the channel idle periods, which may not be extremely accurate. The cumulative sum (CUSUM) change point test, which can quickly find abrupt changes in a process without a priori knowledge of the statistics of change occurrences, is used in (Tang et al., 2010, 2011). The test takes the number of successful transmissions of a tagged node in every $M$ successful transmissions over the whole network and monitor the CUSUM statistic of it. The detection event happens if the CUSUM statistic goes beyond the detection threshold. Game-theoretical techniques are also used in backoff misbehavior detection (Cagalj et al., 2005; Konorski, 2001, 2002). These studies modify the standard protocol to try to achieve a Nash equilibrium for all nodes such that a greedy node cannot gain extra throughput independently.

It is worth noting, however, that most of the detection schemes above are based on the assumption of misbehavior in a saturated uplink UDP traffic environment and all nodes should behave the same if acting normally, which is rarely the case in real-world wireless LANs. Moreover, these schemes are all based on the assumption that the nodes in the network are all exposed to each other, thus they do not take into account the complexity and abnormality that WLAN network traffic exhibits even under non-jamming situations due to the existence of hidden nodes.
CHAPTER 3

A STEALTH GREEDY ATTACK ON TCP TRAFFIC IN WIRELESS LOCAL AREA NETWORKS

3.1 Overview

The jamming attack we design enables increasing the jammer’s own downlink TCP throughput in a wireless LAN without drastic changes to the hardware. This section is dedicated to elaborating how the attack functions and the reasoning behind it.

3.2 Settings and assumptions

Consider a WLAN with one AP associated with two or more nodes, one of which is a rogue node, another (or several others) is (are) the target(s) of jamming, and the remaining nodes are ordinary nodes. The targets are hidden from the jammer, and the ordinary nodes can be exposed to the jammer or exposed to the target (see Fig. 3.1). Each node in the network (including the jammer) is supposed to be involved in a TCP download, receiving downlink TCP traffic from a remote sender, in which the TCP download stream to the target is to be cut down in throughput with jamming and thus the TCP download stream to the jammer is to be benefited when the bandwidth is freed. The WLAN link is assumed to be the bottleneck. Furthermore, we assume that the TCP receivers running at the wireless nodes do not use the delayed ACK mechanism, and all nodes use the basic access mode in the MAC transmission (these cases will be discussed later). By setting its network interface card into promiscuous mode, the jammer can capture any frame within its floor and is able to inspect its type and destination. The propagation time in
the WLAN is assumed to be negligible (compared to the TCP RTT), and the nodes are assumed to be near the AP enough so that capture effect in the channel is negligible (i.e., every collision results in a failed transmission of all colliding frames).

Figure 3.1: Layout of the wireless LAN; inner circles are nodes’ transmission ranges

The jammer can disguise its jamming signals by sending out normal-length UDP datagrams (e.g., DNS queries or VoIP packets). We assume that a jamming signal transmission spans \( m \) (MAC) slots, a TCP-ACK spans \( n \) slots, and a MAC-ACK spans \( r \) slots. We further denote by \( \delta \) (respectively \( \sigma \)) the DIFS (respectively the SIFS) duration in number of slots. For ease of exposition, we use one jammer and one target only. The multi-target case is discussed further later in the chapter.

### 3.3 Jamming preparations

Our jammer gains throughput by trying to jam the uplink TCP-ACKs frames from the target to the AP, so that either one of the following two events could take place:

1. similar to the idea of the “shrew” attack (Kuzmanovic & Knightly, 2003), when enough consecutive TCP-ACKs have been jammed, a retransmission timeout (RTO) at the target’s TCP source is caused, and this sender looses its TCP self-clocking and enters into slow start again;

2. even if an RTO is not reached (which is the case here most often as shown later), the jamming process increases the delivery time of the TCP-ACKs, thus increasing the round-
trip time (RTT) of the TCP stream (slowing down the TCP clock that drives window increases) and thus reduces the frequency of rate increases at the target’s TCP sender. Achieving either event is difficult, as the jammer is hidden from the target, and it cannot precisely jam the TCP-ACKs. To improve the accuracy of jamming while keeping its frequency as low as possible (to avoid detection), the jammer needs to know the probability that the target transmits a TCP-ACK at a given time slot and only jam when this probability is large enough.

To this end, the jammer maintains a backoff-slot counter that mimics (and approximates) the counter at the target, and estimates the total time the target has spent counting down over all the backoff stages since its last successful transmission. Using this counter and the probabilistic model of transmission the jammer calculates the probability that the target will transmit a frame, and uses this probability to decide to jam or to defer the jamming. For example, if according to the timer, the target is deemed to having experienced, up till a given slot, 20 slots of backoff, it could have experienced in its first transmission 10 backoff slots drawn from the minimum contention window but failed, then is mid-way through trying 20 backoff slots in the first retry drawn from double the window; or, it could have tried 5 backoff slots and failed in the first transmission, then tried 10 backoff slots and failed again in the first retry, and is mid-way through trying 10 slots in the second retry. Evidently these two possible sample paths occur with different probabilities since the process is not in steady state, as the backoff random variables in the different stages are drawn from different window sizes and are thus not identically distributed. As a result, the probability that the target transmits at the next chance depends on a combination of the probabilities of all possible sample paths that lead to the current state.

To reduce the frequency of jamming, the rogue node maintains another timer, the “period timer”, used to initiate jamming at intervals of \((m + n - 1)\) slots. That is, since the jamming signal is \(m\) slots and the TCP-ACK spans \(n\) slots, any jamming that starts
(n – 1) slots after the TCP-ACK starts is guaranteed to overlap the ACK for at least one slot (further explained below). In addition, the jammer also keeps track of how many of its jamming signals have collided since the target’s last successful transmission (not necessarily jamming the intended frames): this can very simply be achieved by counting the returned MAC-ACKs for the jamming signals sent out.

### 3.4 Jamming algorithm

Algorithm 3.1 shows the flow of the attack on a per-slot-basis in a event-driven algorithm. Events of the type “received” or “overheard” frame normally are asserted at the slot when the frame has just finished full transmission and therefore we need to compensate for the duration of the transmission in the calculations. In the algorithm we denote the backoff-slot counter as $B$, the period timer as $P$ and the collision counter as $C$. In terms of sequencing, the jamming attack can be divided into two stages: pre-jamming and actual jamming. Once the jammer is in the latter stage, it remains in this stage to attempt to jam all targeted frames until the user decides not to jam any more.

In the pre-jamming period, besides initialization of $B, C$ and $P$, the jammer first observes the channel activity for some time to obtain information of the network, particularly the number of nodes and their relative positions in the network. This can be very easily done by overhearing and matching frames from the AP to different MAC addresses with the corresponding MAC-ACKs in monitor mode. If the jammer identifies a viable jamming target, e.g., a node from which no MAC-ACK frame is overheard but that receives many large DATA frames from the AP (i.e., the node is hidden and takes up a large bandwidth), it is ready to start the attack.

Note here that even with an encrypted link layer using WPA2, it is easy to detect TCP packets and TCP ACKs from the length of the frame. For this we have conducted some investigations on our WLAN and noticed that most big downloads that are worth
attacking generate TCP data with an MTU of 1500 bytes and ACKs of 40 bytes including YouTube videos. After pinpointing a target the rogue node is ready to start sending out the first jamming signal. To do this, the jammer listens for a TCP data packet from the AP to the target node, and tries to jam the returning TCP-ACK. In order to align the backoff-slot counter with the actual one at the target, after the AP sends the TCP packet the jammer waits for \((\delta + r + \sigma)\) before starting the jamming process (Line 16-18) in the Algorithm.

During the actual jamming, there are two rules for the jammer. First, when the jammer senses a busy channel, apart from refraining from sending any jamming signals to obey the carrier sensing mechanism, if the period timer \(P\) expires during this period, the jammer does nothing and restarts \(P\) (Line 7-8). Second, when the jammer overhears a frame from the AP, it freezes the backoff-slot counter and the period timer until the end of the transmission plus \((\delta + r + \sigma)\) or \(\delta\) depending on the frame type (Lines 9-15, where the function \(\text{Delay}(x)\) stops the whole attack for the next \(x\) slots). This is because the target also is exposed to the AP and hears the frame and stops its backoff countdown. Then, judging on the destination and the type of the frame overheard by the jammer, or when one of its timers expires, the algorithm may take different actions:

### 3.4.1 MAC-ACK to the target

This event means the target has most probably successfully transmitted a TCP-ACK to the AP, and the previous jamming failed. It corresponds to Lines 21-22 in Algorithm 3.1. The target will now reset its backoff stage and transmit a new frame if applicable, so the jammer should start the jamming process all over again.

### 3.4.2 Jammer’s MAC-ACK times out

This event corresponds to Lines 23-34 and 40 in the algorithm. A timeout of the jammer transmitted MAC-ACK is indication of a collision at the AP. By default, the jamming
Algorithm 3.1 The jamming attack

1: Initialization
2: \( B := 0, P := n - 1, C := 0; \)
3: \( \text{CounterOn} := \text{False}; \) \( \triangleright \) True if first TCP Data is received
4: \( \text{TwoTimers} := \text{False}; \) \( \triangleright \) True if two timers in use
5: Execute the following for each elapsed slot
6: switch Event do
7: \hspace{1em} case Busy channel AND \( P = 0 \)
8: \hspace{2em} \( P := m + n - 1; \)
9: \hspace{1em} case A frame spanning \( \phi \) from AP heard
10: \hspace{2em} \( B := \phi, P := \phi; \) \( \triangleright \) Restore \( B, P \) to values before the frame is heard
11: \hspace{2em} if The frame is \( \text{DATA} \) then
12: \hspace{3em} Delay(\( \delta + r + \sigma \));
13: \hspace{2em} else
14: \hspace{3em} Delay(\( \delta \));
15: \hspace{2em} end if
16: \hspace{1em} case \( \text{CounterOn} = \text{False} \) AND TCP-Data to target heard
17: \hspace{2em} \( \text{CounterOn} := \text{True}; \)
18: \hspace{2em} Delay(\( \delta + r + \sigma \));
19: if \( \text{CounterOn} = \text{True} \) then
20: \hspace{1em} switch Event do
21: \hspace{2em} case \( \text{MAC-ACK} \) to target
22: \hspace{3em} Goto 1;\hspace{2em} 
23: \hspace{2em} case Jammer’s \( \text{MAC-ACK} \) time out
24: \hspace{3em} if \( \text{TwoTimers} = \text{True} \) then
25: \hspace{4em} \( B := B_{\text{old}}, P := P_{\text{old}}, C := C_{\text{old}}; \)
26: \hspace{3em} end if
27: \hspace{3em} if \( C \geq c \) then
28: \hspace{4em} \( C_{\text{old}} := C, B_{\text{old}} := B, P_{\text{old}} := P; \)
29: \hspace{4em} \( \text{TwoTimers} := \text{True}; \)
30: \hspace{3em} Goto 1;
31: \hspace{2em} end if
32: \hspace{2em} case \( \text{MAC-ACK} \) to jammer AND \( \text{TwoTimers} = \text{True} \)
33: \hspace{3em} \( \text{TwoTimers} := \text{False}; \)
34: \hspace{2em} case \( P \leq 0 \)
35: \hspace{3em} Send jamming signal with probability \( q(B); \)
36: \hspace{3em} \( P := m + n - 1; \)
37: end if
38: end if
39: \( B+ := 1, P- := 1; \)
40: if \( \text{TwoTimers} = \text{True} \) then \( B_{\text{odd}}+ := 1, P_{\text{odd}}- := 1; \)
41: end if
signal is to be resent after the collision, which is undesirable considering that the jam- mings are based on the timers. We could take advantage of the Quality of Service (QoS) enhancements for wireless LAN by assigning the jamming frames to an access category (AC) different from the normal frames, and set the retry count of that AC to 0, i.e. a jamming signal is not resent after a collision.

Meanwhile, the number $C$ of collisions suffered by the jamming signals since the target’s last successful transmission is increased by one, and if $C$ exceeds the retry count $c$, the target could have dropped the frame and reset its backoff window to $CW_{min}$. It also might not have done so as some of the collisions experienced by the jamming signals may have been caused by exposed-node collisions at the AP. Adjusting the backoff-slot counter when the target resets its window is crucial since if the target will transmits with $CW_{min}$ while the jammer still jams according to an estimated large window and is highly likely to miss. In order to decide with confidence whether the target has reset or not its window, whenever $C$ exceeds $c$, a new backoff-slot counter is initiated by the jammer, while the old one is also kept running. A jamming is first attempted with this new counter. As initial jamming attempts have very high probabilities of success, if this first jamming is successful then the target has most likely reset its backoff window, and conversely, if the jamming is unsuccessful, then there is a high probability that the target did not reset its window, and is still counting down in a further backoff stage. Hence every time $C$ exceeds $c$, while keeping the old $B$, $P$ and $C$ alive and running, the jammer initiates the new jamming process with the parameters reset, sends out the first jamming signal and waits for a returning MAC-ACK from the AP. If this MAC-ACK is heard, it means the jamming signal went through. Then we assume the target has not reset its backoff window, and the jammer should come back to operate on the old $B$, $P$ and $C$. Conversely if the jammer’s MAC-ACK timer for this signal expires, it means the jamming signal has collided. Under such circumstance the jammer should continue to work on the new jamming attack procedure.
3.4.3 Expiry of the period timer

This event corresponds to Lines 35-37 in the algorithm. Since corrupting one single bit makes the frame fail the CRC check, the jammer does not need to jam the whole TCP-ACK from the target. As a result, it can take advantage of the length of the TCP-ACK to jam less frequently while achieving the same effect. For example, at the start of a jamming process, the jammer can wait until the \((n-1)\)th slot to start emitting jamming signals, so that even if the target sends out its TCP-ACK at the earliest available opportunity, its tail will still collide with the very beginning of the jamming signal.

As the transmission of a jamming signal spans \(m\) slots and the target’s TCP-ACK spans \(n\) slots, the whole jamming period is divided into smaller periods of \((m+n-1)\) slots each, and at the \((n-1)\)th slot in each period, the jammer decides to transmit the jamming signal with probability \(q(B)\) where \(B\) is the current elapsed slots on the backoff-slot counter. The period timer is always reset to \((m+n-1)\) slots when it expires. Probability \(q(B)\) is based on the expected data rate of the target, the potential of being detected and the probability \(p(B)\) that the target’s transmission of the TCP-ACK overlaps with the next \(m\) slots: \(p(B) = \text{Pr}[\text{target starts transmission in } [B-n+1, B+m-1] \text{ slots}]\). At each jamming opportunity, priority is given to normal data frames over jamming signals, and in this case the period timer is reset for the next jamming opportunity.

One prerequisite for accurate jamming is that the MAC layer of the jammer emits the jamming signal as soon as it receives the downward request. Again we can utilize the QoS of 802.11 to set the \(\text{CWmin}\) and \(\text{CWmax}\) of the AC assigned to the jamming signals to 0.
3.5 Probabilistic model

3.5.1 Relationship between \( p(x) \) and \( q(x) \)

The choice of the jamming probability \( q(x) \) should depend on the target’s transmission probability \( p(x) \). This is intuitive since if the target is going to send a TCP-ACK with high probability, the jammer should also try to jam it with increased effort in order to achieve a better jamming effect. Particularly, \( q(x) \to 0 \) when \( p(x) \to 0 \) and \( q(x) \to 1 \) when \( p(x) \to 1 \). Therefore, we could simply choose a power function, an exponential function or a linear function as the transfer function \( f : p(x) \to q(x) \). In the model we choose the power function \( q(x) = p(x)^w \), with \( w \) as a parameter. The function can either be concave, linear or convex depending on the choice of \( w \), which enables us to test several shapes to achieve the best jamming effect.

3.5.2 Calculation of \( p(x) \)

Let, \( A(i, j) \equiv \{ \text{the target transmits at the } i\text{th slot in its } j\text{th backoff stage} \} \), \( B(i, j) \equiv \{ \text{the target does not transmit from the } i\text{th to the } j\text{th slot} \} \), and \( C(i) \equiv \{ \text{no successful transmission happens before the } i\text{th slot} \} \). Then we have:

\[
p(x) = \sum_{i=1}^{m+n-1} \sum_{j=0}^{c} \Pr[A(x-n+i, j), B(x-n+1, x-n+i-1)|C(x-n+1)] \tag{3.1}
\]

Here the index \( i \) iterates through the slots in the interval where a transmission of the TCP-ACK from the target would be hit by the (possible) jamming signal, and \( j \) is the count of backoff stages, upper bounded by the long retry count of \( c \). \( B(x-n+1, x-n+i-1) \) is present in the equation to make the joint event in each term of summation independent of each other.
Define $D(x, i, j) \equiv \{A(x + i, j) \cap B(x + 1, x + i - 1) \cap C(x + 1)\}$. Then from (3.1) by conditional probability we have

$$p(x) = \sum_{i=1}^{m+n-1} \sum_{j=0}^{c} \frac{\Pr[D(x - n, i, j)]}{\Pr[C(x - n + 1)]}.$$  \hspace{1cm} (3.2)

If $j = 0$, i.e., the TCP-ACK is a newly-sent packet, then,

$$
\Pr[D(x, i, 0)] = \Pr[(x + i) \text{ chosen as backoff from the minimum contention window}]
= \begin{cases} 
1/(CW_{\min} + 1) & \text{if } x + i \leq CW_{\min} \\
0 & \text{otherwise}.
\end{cases} \hspace{1cm} (3.3)
$$

Define $E(x, j) \equiv \{A(x, j) \cap C(x)\}$. Since if the target transmits a TCP-ACK that is corrupted on the way, it needs to wait for a MAC-ACK timeout period before realizing the TCP-ACK is lost and tries to resend it, it cannot resend the TCP-ACK immediately after a collision. Assume this MAC-ACK timeout period spans $s$ slots. Then for $j > 0$, expanding on the time of transmission of the previously sent (and failed) TCP-ACK, by the definition of $E(x, j)$,

$$
\Pr[D(x, i, j)] = \sum_{k=1}^{\min\{x-1, x+i-n-s\}} \Pr[E(k, j-1)] \\
\times \Pr[\text{collision at } k] \\
\times b(x + i - k - n - s, j - 1), \hspace{1cm} (3.4)
$$

where $b(i, j)$ in is the probability that the backoff chosen from the $j$th backoff window equals $i$ slots, i.e.,

$$b(i, j) = \begin{cases} 
1/(CW_{j} + 1) & \text{if } i \in [0, CW_{j}] \\
0 & \text{otherwise},
\end{cases} \hspace{1cm} (3.5)$$
with $CW_j$ being the size of the $j$th contention window, i.e. $CW_j = 2^j(CW_{\text{min}} + 1) - 1$.

The probability of the target TCP-ACK colliding with anything at the $x$th slot, $\Pr[\text{collision at } x]$ is approximated as follows:

$$\Pr[\text{collision at } k] = 1 - (1 - q(d(k - 1))) \times (1 - \mathbb{B}(\tau + 1)),$$

where $\mathbb{B}(n)$ is the probability of collision for $n$ nodes in Bianchi’s model (Bianchi, 2000), $\tau$ is the number of stations on the targets’ side and $d(x)$ is the slot where the last jamming decision before the $x$th slot is made,

$$d(x) = \left\lfloor \frac{x}{m+n-1} \right\rfloor \times (m+n-1) + n - 1.$$  

(3.7)

$\Pr[E(x,j)]$ induces on itself in a similar way:

$$\Pr[E(x,j)] = \sum_{k=1}^{x-n-s} \Pr[E(k, j-1)] \times \Pr[\text{collision at } k] \times b(x - k - n - s, j - 1),$$

(3.8)

with the base case being

$$\Pr[E(i,0)] = \begin{cases} 
1/(CW_{\text{min}} + 1) & \text{if } i \leq CW_{\text{min}} \\
0 & \text{otherwise}; 
\end{cases}$$

(3.9)

and $\Pr[E(1,j)] = 0$ for $j > 0$. All previously calculated values of $\Pr[E(\cdot, \cdot)]$ are stored to be reused for later recursions.

By substituting (3.5), (3.6) and (3.8) into (3.4) we have a recursive form on $\Pr[D(x, i, j)]$. The denominator in (3.2), $\Pr[C(x-n+1)]$ can also be represented by $\Pr[E(x,j)]$:  

21
\[ \Pr[C(x)] = \sum_{k=1}^{x-1} \sum_{j=0}^{c} \Pr[E(k,j)] \times \Pr[\text{collision at } k] \times a(x - k - n - s, j) + t(x), \]  

(3.10)

where \( a(i, j) \) is the probability that the backoff chosen from the \( j \)th backoff window is larger than or equal to \( i \) slots, i.e.

\[
a(i, j) = \begin{cases} 
\frac{(CW_j - i + 1)}{(CW_j + 1)} & \text{if } CW_j \geq i \text{ and } i > 0 \\
1 & \text{if } i \leq 0 \\
0 & \text{otherwise},
\end{cases}
\]

(3.11)

and \( t(x) \) is the probability that no transmission occurred before the \( x \)th slot, i.e.

\[
t(x) = \begin{cases} 
\frac{(CW_{min} + 2 - x)}{(CW_{min} + 1)} & \text{if } x \leq CW_{min} + 1 \\
0 & \text{otherwise}.
\end{cases}
\]

(3.12)

By putting (3.3), (3.4) and (3.10) into (3.2) we obtain the value of \( p(x) \). Note that in the calculation of \( p(x) \), previously calculated \( q(x) \) values are used; on the other hand, \( q(x) = f(p(x)) \). This means the calculation of \( p(x) \) and \( q(x) \) are similar to dynamic programming, i.e., first calculate \( p(x) \) for small \( x \), obtain \( q(x) = f(p(x)) \), then calculate \( p(x) \) for further \( x \). This makes sense as the intensity of jamming influences the target’s transmissions at a later time which in turn influences the intensity of jamming then, by the assumption of our model.
3.6 Other issues

3.6.1 Use of RTS/CTS

The use of RTS/CTS mechanism to avoid hidden terminal collisions can be circumvented by targeting the RTS frame for the TCP-ACK instead of the actual TCP-ACK. In this case the jamming model still works with some minor changes to some constants, including the short retry count $c$ and the length of the jammed frame $n$. Now, instead of TCP-ACKs to be transmitted directly, the target node sends out RTS frames first, which are jammed by the jammer with a certain probability. The resulting effect remains similar to that without the use of RTS/CTS.

3.6.2 Jamming multiple targets

It is known that unfair bandwidth allocation exists in a network where the client nodes are partitioned in mutually hidden clusters of nodes with different numbers (Hung & Bensaou, 2011). Therefore when there are multiple nodes hidden from the jammer, jamming only one of them may not result in any gain in throughput, as the other hidden nodes still beat the jammer in grabbing the freed bandwidth, while the jamming signals also occupy some of the jammer’s already little available bandwidth. To address this problem, the jammer can be designed to jam multiple targets at the same time, simply by keeping one copy of the backoff-slot counter, collision counter and period timer per target. When one period timer times out while the jammer is currently jamming, it is reset for the next period. Evidently, there is a relationship between the jamming effort and the jamming effect: the more nodes to jam, the more jamming signals are needed, and these jamming signals may conversely have a negative effect on the jammer’s throughput gain. Hence the jamming attack to jam all nodes will be less effective with the increase in the number of normal nodes on the other hand, and eventually worse than with no jamming. When multiple normal nodes also exist on the jammer’s side cluster however, they could gain a large
additional throughput from the targets even if in this case the jammer itself does not gain much. This opens up the possibility to a scenario with colluding attackers (e.g., running a smart-phone to jam while grabbing the bandwidth with a notebook). The attacker does not need to be redesigned.

3.6.3 Tracing the target’s queue length

In the model, the jammer is always in the jamming process after it is started unless the user forces a stop. Alternatively, the jammer could trace the queue length of the target, so that when the targets queue is presumed empty, there is no need to continue jamming until a new packet arrives at the target, and the jammer can enter stand-by mode. Tracing the queue length can be accomplished by looking into packets directed to the target: a TCP data packet increases the queue length by 1; a MAC-ACK to the target indicating a TCP-ACK received at the AP or a packet dropped due to exceeding retry limit decreases the queue length by 1. This may result in less jamming signals sent, yielding more airtime to jammers traffic and lower probability of detection. However, implementing it may raise some problems to be addressed. First is that the jammer may enter the network later than the target, such that the target could have built up a nonzero queue already. By estimating the RTT (Jiang & Dovrolis, 2002) and carefully matching the data packets received with the TCP-ACKs sent, the jammer could come to be sure at one point that the targets queue is empty, and trace it onwards. Second, the exposed collisions can cause the tracing inaccurate, leading to the possibility that the jammer is still jamming when the target has nothing to send, or the jammer still has something to send but the jammer is not jamming. The latter case is more unwanted as a sequence of unjammed TCP-ACKs received at the TCP source can rapidly recover the TCP sending window, rendering previous jamming efforts useless in terms of causing an RTO. Since the jamming has already started, it now requires very careful analysis of overheard packets in order to adjust the presumed queue length now and then.
3.6.4 TCP delayed ACK mechanism

In some cases the TCP delayed ACK mechanism (Stevens, 1997) may be used. Then it constitutes a problem to identifying when the target will actually send the acknowledgement after receiving a data packet. This can be solved by matching the acknowledgement with the two data packets received just before it. Also during jamming, if the jammer starts jamming on overhearing a packet to the target but the target decides to defer the acknowledgement, when some time later a second data packet comes to the target and it is prepared to acknowledge, the backoff-slot counter of the jammer may have reached a large number, and the jamming is likely to fail in the first transmission. Thus the jammer needs to remember which data packets are going to be acknowledged and which are in between two former ones and not going to be acknowledged. Together with the implementation of tracing targets queue length, the jammer may even benefit from delayed ACK mechanism as it stops jamming when the target is deferring the acknowledgement.

3.7 Immunity to detection

While the effectiveness of jamming is mainly verified by the jammer’s achieved throughput gain, evading misbehaviour detection is also of utmost importance. Here we assume that the detection system is in the AP and can only gather information that the AP can overhear and interpret. Since there is no existing detection scheme that tackles either a jamming attack against hidden nodes or that jams the TCP-ACKs of the target, we cannot directly apply any detection system to it. Hence we will explore the following different aspects of transmission to see if a detection scheme based on any of them is viable.
3.7.1 Estimated backoff window

Because the jamming attack involves manipulation of the backoff since a jamming signal is transmitted as soon as the jammer decides to jam instantaneously, the jamming may become a target of backoff misbehaviour detection schemes such as DOMINO (Raya et al., 2006), in which the station that emits two transmissions with no collision in between is assumed to spend the idle time backing off. This average backoff time is then compared with the nominal backoff value (the average backoff of the AP) times a parameter between 0 and 1. We test this value on our jamming to examine its vulnerability to such detection method.

3.7.2 RTT

Assuming the AP is able to measure the end-to-end TCP RTT, simply estimating this end-to-end RTTs cannot give information on jamming attacks because traffic streams naturally go through paths with different delays. However if in addition the AP can inspect the packets and know the source IP addresses, it may estimate the approximate delay over the distance between the source and itself. As shown previously, the jamming attack greatly increases the RTT of the target traffic, so a huge discrepancy between the estimated RTT and the real RTT values that exceeds the variance of RTT caused by congestions, route changes and so on (Sessini & Mahanti, 2006), could be a sign of the existence of a jammer, albeit not a conclusive one.

3.7.3 Number of collisions

The hidden nodes setting inevitably leads to more collisions than when the nodes are exposed, but as the attack consciously aims at colliding the target frames, the number of collisions could be even higher. Specifically, the ratio between the number of collisions and the total number of frames received at the AP may be higher than the case where
the nodes are still hidden but no jamming is staged. We look into the possibility for this ratio to be a tool for detection.


CHAPTER 4

EVALUATION

4.1 Evaluation settings

We validate the effectiveness of our attack model via network protocol simulation in ns-2 (Fall & Varadhan, 2005) in a wired-cum-wireless network. IEEE 802.11 standard with basic access mode is used in the simulation. The parameters used are shown in Table 4.1. The scenario consists of an AP placed in the center, with two sets of nodes placed on the opposite sides. One set is composed of the jammer plus other normal nodes if any, and on the other side are the target nodes. The transmitters’ range and the receivers’ sensitivity are properly configured such that both sides can hear and transmit to the AP, while remaining hidden from each other. TCP sources are placed in the wired network and are connected to the TCP sink nodes in the wireless network. Each source generates downlink TCP traffic to its corresponding wireless node. The round-trip link delay in the wired link is set to 12 ms. The wireless link bandwidth is 54Mbps with a basic rate of 6Mbps for frame headers and the wired links have 200Mbps bandwidth, making the WLAN link the bottleneck. Each case of the simulation is tested with 100 runs, and each run lasts 100 second, during which the jammer starts jamming after 20 seconds for the network to achieve a steady state before the jamming process operates. Data for assessing jamming performance is collected from 30 to 100 seconds.

We first inspect a particular case in detail to understand what is actually happening in the jamming process, then look at the overall effect of jamming under different circumstances.
**Table 4.1: Simulation parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame payload</td>
<td>1500 bytes</td>
</tr>
<tr>
<td>Data rate</td>
<td>54 Mbps</td>
</tr>
<tr>
<td>Basic rate for frame headers</td>
<td>6 Mbps</td>
</tr>
<tr>
<td>minRTO</td>
<td>1s</td>
</tr>
<tr>
<td>Slot time</td>
<td>9 µs</td>
</tr>
<tr>
<td>m</td>
<td>27 slots</td>
</tr>
<tr>
<td>n</td>
<td>4 slots</td>
</tr>
<tr>
<td>s</td>
<td>7 slots</td>
</tr>
<tr>
<td>r</td>
<td>5 slots</td>
</tr>
<tr>
<td>c</td>
<td>7</td>
</tr>
<tr>
<td>CW&lt;sub&gt;min&lt;/sub&gt;</td>
<td>15</td>
</tr>
<tr>
<td>CW&lt;sub&gt;max&lt;/sub&gt;</td>
<td>1023</td>
</tr>
</tbody>
</table>

### 4.2 A case in detail

For simplicity we consider a 1-on-1 case with \( w = 0.6 \), and look into the throughput, TCP sender’s window and RTT (samples taken every 0.1 second) of both the jammer and the target’s TCP traffic. We choose two representative runs and plot the change of these three variables (Fig. 4.1, Fig. 4.2 and Fig. 4.3 for the first run, Fig. 4.4, Fig. 4.5 and Fig. 4.6 for the second run) with respect to time.

From the data shown here and other runs of simulation, we can arrive at some key observations. First, as shown in Fig. 4.2 between 21s and 26s for the jammer and between 62s and 78s for the target in the first run and in Fig. 4.5 between 48s and 54 for the target in the second run, both the jammer and the target may experience RTOs, but overall the probability is not high for both cases; in some cases there is even no RTO. Observations over all the runs of simulation show that the jammer suffers much fewer RTOs than the target as the jamming constantly delays the transmission of the target’s TCP-ACK, and if the target node reaches an RTO, it almost always takes longer, if ever,
Figure 4.1: Throughput of jammer and target nodes, 1st run

Figure 4.2: TCP sender’s window of jammer and target nodes, 1st run
Figure 4.3: Samples of round-trip time of jammer and target nodes, 1st run

Figure 4.4: Throughput of jammer and target nodes, 2nd run
Figure 4.5: TCP sender's window of jammer and target nodes, 2nd run

Figure 4.6: Samples of round-trip time of jammer and target nodes, 2nd run
than the jammer to recover. When the target times out, the jammer can quickly take over the bandwidth with its own TCP traffic. On the contrary, even when the jammer times out, the jammer cannot occupy the majority of the bandwidth due to consistent jamming from the jammer, and this leaves room for the jammer to make a fast recovery after its RTO.

Second, even when the target does not experience an RTO, due to the delay caused by the jamming its RTT is much longer than that of the jammer (Fig. 4.3, 4.6), making its TCP sender’s window grow at a much slower rate than that of the jammer, as can be observed from the slopes in the growth of the sender’s windows in Fig. 4.2 and 4.5. Especially, when the target times out, the jammer’s window grows particularly fast. This will eventually lead to a low average window and an even lower average throughput for the target.

4.3 Overall effect of jamming

Once we understand how the jamming works, we can evaluate the effectiveness of jamming. First we investigate the validity of basing \( q(x) \) on \( p(x) \) and the influence of parameter \( w \). In Fig. 4.7, we compare the average throughput of the jammer for a constant transfer function \( q(x) = w \) against the power function used in our model \( q(x) = p(x)^w \). From the figure we can see that our model achieves much better throughput, while the throughput for both nodes without jamming is around 10Mbps. Also, the figure, especially the more dramatic line of \( q(x) = w \), agrees with the intuition that the jammer’s throughput should follow a pyramid-like shape with the increase of \( w \). This is because too large a value of \( w \) decreases \( q(x) \), generating too few jamming signals and degrading the jamming effectiveness, while too small a value of \( w \) increases \( q(x) \), generating more jamming signals than needed and taking up jammer’s own bandwidth when it could be counting down the backoff for transmitting its own useful packets, which can even be worse than without
Figure 4.7: Jammer’s throughput comparison: for different $q(x)$

jamming as is shown for $q(x) = w$ at $w = 0.1$ in Fig. 4.7. In case of $w = 0.5$ in $q(x) = p(x)^w$ where the throughput of the jammer drops quite significantly, we further look into the results case by case and find that this is caused by the occasional deviation into a failed jamming or outlier case - where the jammer suffers from jamming. We observe that although the chance of a failed jamming is very low, one such case could have a significant effect of the average throughput. In the case of $q(x) = p(x)^w$ where average throughputs of different $w$ do not vary much, few more failed jamming cases out of 100, which does not truly reflect the low probability of a failed jamming, could make a big difference. Fig. 4.8 shows the throughput comparison for $q(x) = p(x)^w$ when the outlier cases - those with jammer throughput below its average without jamming - are taken out, where the dashed line is the number of outliers for each $w$. We can observe that the number of outliers shows no sign of following any particular trend, and the throughput shape agrees with our intuition.
We then run the simulation of the attack under different settings, all with $w = 0.6$. In the “Setting” rows/columns in the tables, the first number indicates the number of nodes on the jammer’s side, where only one of them is the jammer itself while other nodes, if any, are normal TCP sinks, and the second number is the number of nodes on the targets’ side, where all nodes here are targeted.

### 4.3.1 Throughput

Table 4.2 shows the average throughput achieved under attack with different settings, in Mbps. It shows the throughput of the jammer increases dramatically, in many cases to twice or three times its normal (no jamming) throughput. However as the number of nodes in the network increases, the share of throughput the jammer could grab from other nodes becomes smaller, while the need to jam more nodes exerts a negative influence on the transmission of its own data. Therefore, the throughput gain decreases and would
finally vanish as the number of nodes further increases. Still, as is discussed in 3.6.2, in a multi-on-multi scenario, the jamming throttles the target nodes, earning bandwidth for normal nodes on the jammer’s side that are not sending jamming signals. This suggests the effectiveness of staging this attack by a group of malicious nodes acting normally with a colluding jammer.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Jammer</th>
<th>Jammer gain</th>
<th>Gain Jammer-side nodes</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/1</td>
<td>16.64</td>
<td>58%</td>
<td>N/A</td>
<td>2.03</td>
</tr>
<tr>
<td>1/2</td>
<td>13.43</td>
<td>287%</td>
<td>N/A</td>
<td>2.20</td>
</tr>
<tr>
<td>1/3</td>
<td>6.40</td>
<td>234%</td>
<td>N/A</td>
<td>2.84</td>
</tr>
<tr>
<td>1/4</td>
<td>4.00</td>
<td>153%</td>
<td>N/A</td>
<td>2.43</td>
</tr>
<tr>
<td>1/5</td>
<td>1.58</td>
<td>33%</td>
<td>N/A</td>
<td>4.02</td>
</tr>
<tr>
<td>2/2</td>
<td>6.55</td>
<td>23%</td>
<td>55%</td>
<td>2.14</td>
</tr>
<tr>
<td>2/3</td>
<td>2.07</td>
<td>-30%</td>
<td>155%</td>
<td>2.57</td>
</tr>
<tr>
<td>3/3</td>
<td>1.98</td>
<td>-49%</td>
<td>52%</td>
<td>2.01</td>
</tr>
</tbody>
</table>

Table 4.2: Average throughput simulation results

4.3.2 Estimated backoff window

Table 4.3 shows the estimated backoff window of the jammer and the nominal backoff window in slots, discussed in 3.7.1, the nominal backoff window being the window estimated by the AP. We can observe that even if false positives increase in the presence of hidden nodes (Raya et al., 2006), the average observed backoff window value of the jammer never falls below the nominal backoff window, thus rendering the detection scheme in DOMINO ineffective in this attack. The detection based on the relatively large observed backoff of the targets is also not viable, as the AP cannot distinguish between the backoff time and the idle time in an unsaturated downlink TCP scenario. Further careful analysis based on packet inspection, transmissions and collisions is needed to achieve a valid detection scheme. Another possible misbehavior in (Raya et al., 2006), “Scrambled frames” for uplink traffic is not applicable in our attack, because the authors argue that
the jammer cannot scramble the headers of the DATA frames (the TCP-ACK frames in our case) which contains sequence number, as in the uplink scenario it needs to know whether the frame is destined to itself; in our attack the jammer does not need to worry about this. The other potential misbehaviors in the paper, “Shorter than DIFS” and “Oversized NAV” are also not valid for the attack due to the way the jamming algorithm is designed.

<table>
<thead>
<tr>
<th>Setting</th>
<th>1/1</th>
<th>1/2</th>
<th>1/3</th>
<th>1/4</th>
<th>1/5</th>
<th>2/2</th>
<th>2/3</th>
<th>3/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>16.9</td>
<td>19.1</td>
<td>41.6</td>
<td>46.6</td>
<td>64.4</td>
<td>38.5</td>
<td>92.8</td>
<td>61.9</td>
</tr>
<tr>
<td>B</td>
<td>8.21</td>
<td>8.05</td>
<td>8.02</td>
<td>8.01</td>
<td>8.01</td>
<td>8.23</td>
<td>8.11</td>
<td>8.18</td>
</tr>
</tbody>
</table>

Table 4.3: Estimated backoff window in slots: A=Jammer Backoff window; B=Nominal (AP) backoff window

4.3.3 RTT

Table 4.4 shows the average RTT in milliseconds for both sides in the simulation, displaying a large gap between them. However, the jammer’s TCP traffic also experiences a large delay. Thus it is inconclusive to judge the difference in RTT as an indication of jamming. Also, several conditions must be met before the RTT can be used as a detection standard, including the precise estimation of RTT, the steady routing and link conditions, the use of the optional time-stamp, the synchronization of clocks between the AP and the TCP source, and so on.

<table>
<thead>
<tr>
<th>Setting</th>
<th>1/1</th>
<th>1/2</th>
<th>1/3</th>
<th>1/4</th>
<th>1/5</th>
<th>2/2</th>
<th>2/3</th>
<th>3/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jammer (ms)</td>
<td>57</td>
<td>65</td>
<td>83</td>
<td>95</td>
<td>104</td>
<td>65</td>
<td>85</td>
<td>77</td>
</tr>
<tr>
<td>Target (ms)</td>
<td>145</td>
<td>158</td>
<td>164</td>
<td>172</td>
<td>181</td>
<td>128</td>
<td>135</td>
<td>127</td>
</tr>
</tbody>
</table>

Table 4.4: Average RTT simulation results
4.3.4 Number of collisions

Finally, Table 4.5 compares the number of collisions experienced with and without jamming. $\alpha$ is the number of collisions divided by the number of frames correctly received by the AP. From the comparison we can say that parameter $\alpha$ seems to be a good indicator of jamming, especially when the number of nodes is large and the jammer needs to jam more. Nonetheless, it remains to be seen whether there could be scenarios where no jamming is present yet the ratio is as high as the values shown in the table.

<table>
<thead>
<tr>
<th>Setting</th>
<th>1/1</th>
<th>1/2</th>
<th>1/3</th>
<th>1/4</th>
<th>1/5</th>
<th>2/2</th>
<th>2/3</th>
<th>3/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jamming</td>
<td>0.434</td>
<td>0.731</td>
<td>1.61</td>
<td>2.08</td>
<td>2.65</td>
<td>0.647</td>
<td>1.11</td>
<td>1.04</td>
</tr>
<tr>
<td>No jamming</td>
<td>0.206</td>
<td>0.245</td>
<td>0.258</td>
<td>0.254</td>
<td>0.231</td>
<td>0.409</td>
<td>0.604</td>
<td>0.633</td>
</tr>
</tbody>
</table>

Table 4.5: Number of collisions $\alpha$ simulation results
CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The widely used MAC protocol for wireless LANs, IEEE 802.11 is designed with the assumption that all nodes follow the protocol specifications. However, it is possible that a malicious jamming node deviates from the protocol and brings unfairness to the network. Therefore, jamming forms an important part of WLAN security issues, and many studies have been done in this field. Yet previous research has ignored the case where deliberate jamming occurs between nodes hidden from each other. Because such nodes can exist naturally in WLANs and are not necessarily detectable with classic techniques, the jamming against hidden nodes could pose a new threat to the network security. Motivated by the fact that TCP as well as download traffic forms a major part in today’s Internet, and intending not to change the hardware for the easiness of staging, in this thesis we introduced a new intelligent jamming attack on the TCP-ACKs of downlink TCP traffic against hidden targets. The jammer works by probabilistically estimating the time when the target is expected to transmit TCP-ACKs and transmits its own jamming signal at the proper time to suppress them. Throughput is gained by the jammer by making the target TCP reach a timeout, or simply by increasing its round-trip time. Simulations are conducted in ns-2 to show the effectiveness of the attack and analyse its immunity to detection schemes. Discussion of possible parameters to use to detect such attack is also given.
5.2 Future work

There are a number of aspects in which future work can be done.

• In order to verify its effectiveness in the real world, we are currently focused on implementating and deploying this attack in a real wireless network. We use the libpcap interface to capture packets in the promiscous mode, gather necessary information from the packets and pass them to an analysis module. A decision making module takes the input of the processed results from the analysis module and issues a jamming instruction if it decides to jam according to the probabilistic model. Then a jamming module handles the actual emission of jamming signal.

• An analytical study on the throughput gain is another aspect we are looking into, using the method in (Padhye et al., 1998) and trying to add to it the influence of RTT delay and TCP RTO.

• Currently the attack strategy on multiple targets is quite simple, so we are interested in improving it to achieve better throughput and evade detection. It would also be interesting to further explore the effectiveness of a collusion attack where normal nodes use one dummy jammer to gain throughput for themselves against a large cluster of targets, as well as how and why various parameters have different effects on the jamming performance.

• It is also of our interest to inspect the possibility of the existence of multiple jammers. The scenarios consist of multiple jammers on one side or on both sides. Intuitional reasoning tells us that multiple jammers on one side could ease each other’s jamming efforts when jamming multiple targets on the opposite side while remaining orderly amongst their jamming signals due to the carrier sensing among exposed nodes. Further tweaks in the attack can even yield better jamming effect by assigning specific targets or jamming periods for each jammer. On the other hand, multiple jammers on both sides will try to jam each other’s DATA packets and eventually nobody will be able to have much bandwidth, let alone any increase. More in-depth study on these scenarios are to be done.
• Detection against the attack is the other side of the problem, and although we suggested that the number of collisions could be a potential indicator of jamming, its validity, accuracy, false positive rate and feasibility under real setting are all of necessity for further research, while other more effective way of detecting could also be a direction of future work.
Bibliography


