Unleashing the Shrew: a Stealth Greedy Targeted Attack on TCP Traffic in Wireless LANs*

Liyi Gu[1], Jun Zhang[2], Brahim Bensaou[1]
[1]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology
[2]Network and Computer Science Department, Telecom ParisTech
lgu@cse.ust.hk, jun.zhang@telecom-paristech.fr, brahim@cse.ust.hk

***Abstract*—This paper presents a new jamming attack in wireless LANs that deliberately targets uplink TCP acknowledgements (TCP-ACKs) of downlink TCP flows. To ensure immunity to detection with existing schemes, in this attack, the attacker does not jam the target constantly; instead, it relies on our probabilistic estimation model to forecast the time when its transmission has the highest likelihood of colliding with the target's generated TCP-ACKs. Repeating this process results in a decrease of the average congestion window of the targeted due to an increased round-trip time (RTT). The rogue node and/or its colluding attackers can grab this freed bandwidth and increase their throughput. We demonstrate via ns-2 simulation the effectiveness of such attack and show how easy it is to deploy without hardware modification. We also discuss its immunity to detection by existing detection schemes and investigate some parameters that may be used in building future detection mechanisms.**

***Index Terms*—Jamming, wireless LAN, hidden node, TCP**

I. Introduction

The shared nature of their medium makes wireless networks 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 LANs (WLANs) and their increased popularity in commercial enterprises, security in general has become of utmost importance in recent years.

Jamming in general is one of the major forms of attacks that can easily be staged in WLANs. Jammers can be divided into two categories: disruptive jammers, who intend simply to disrupt the normal functioning of a network, and greedy jammers whose goal is to profit from their actions by grabbing more bandwidth for example. A disruptive attack may be staged by sending out packets continually to deceive the receiver, or by coordinating such transmissions with channel activity. Such attacks have been shown to be very effective [1], [2], nevertheless, such jammers require drastic modifications to their wireless hardware, rendering such attacks less easy to stage by ordinary users of a WLAN. Greedy attackers, on the other hand, try to increase their own throughput share via misbehaviour. For uplink traffic, the greedy node can selectively scramble frames from other nodes, leading them to increasing their contention window [3]. The greedy node can also manipulate the protocol parameters, including the inter-frame space (IFS) duration, the duration value in the MAC frame header to induce longer NAVs for other nodes, and/or the binary exponential backoff time [4]. For downlink traffic, the greedy node can jam TCP-ACKs from the target to the AP, so that they never reach the sender, which decreases its sending rate, or it can jam the data frames from the AP to the target node and forge MAC-ACKs to prevent the AP from retransmitting the data, leading the sender to decreasing its rate [3]. Many of these jamming techniques require also changes to the hardware, making them inaccessible to "user lambda".

Many detection schemes have been proposed to address user misbehaviour in IEEE802.11 networks [3], [5]–[9]. While some detection schemes [3], [5] assume the backoff duration of each node to be observable at the detection terminal, other alternative methods [6]–[9] take advantage of successfully received frames at the AP and the channel idle period to infer the backoff value indirectly. However, these detection 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.

Motivation. It is well known [10], [11] that TCP downloads dominate today's WLAN traffic thanks to the wide 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 or TCP flavours differ). Therefore, successfully targeting TCP-ACKs may result in some occasional time-outs and most frequently in an increased RTT for the target, leading to a lower throughput. This is corroborated by the results in [12]. Meanwhile, staging an attack that does not require hardware modification could make the attack accessible to ordinary users making it more commonly stageable. As such, a jamming attack in WLAN that targets TCP traffic of nodes that are hidden from the attacker could be a good option because jamming exposed nodes requires breaching the rules of the built-in CSMA mechanism and thus needs non-trivial modification to

*This work is supported in part under grant: RGC GRF 610411
The work of Dr Jun Zhang was done while he was with the Hong Kong University of Science and Technology

978-1-4799-3780-6/14/$31.00 ©2014 IEEE

the hardware. A typical scenario for such attack is a malicious node joining a public hotspot turning on the jamming process that first monitors the traffic for a few seconds in search for appropriate hidden nodes targeting particularly those with high bandwidth and jams them. The simplicity of such attack and its versatility could pose a potentially threat to the viability of WLANs in public places. Our goal in this paper 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.

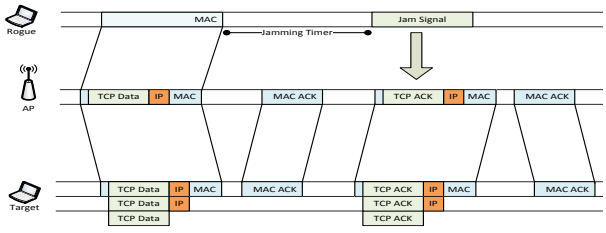

Fig. 1: Attack illustration

Overview. In this paper, 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 is more accurate, it requires 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) as both can only be achieved by modifying the hardware to enable custom inter-frame spaces. In the basic access mode, TCP-ACK frames from a hidden target node to the AP turn out to be a very good candidate for jamming, 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's transmissions. Randomly jamming such TCP-ACKs may improve the success rate of jamming but it fails to achieve two of the fundamental requirements of the greedy attacker – viz. increasing its throughput (as now it occupies its share of bandwidth with jamming packets) and remaining 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 its sporadic attacks with the targets' transmission times with a high probability.

Outline. The rest of the paper is organized as follows; the description of the attack model and the detailed event-driven algorithm for deriving the jamming times are given in Section II. Simulation results of the proposed strategy are presented in Section III, and we conclude the paper in Section IV. For ease of exposition, the actual analytical model is detailed in the appendix.

II. Jamming Attack

A. 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. 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, and 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 (the use of RTS/CTS will be discussed later). By setting its network interface card into monitor mode, the jammer can capture any frame within its floor and is able to inspect its type and destination (see Fig. 1). 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 for all colliding frames).

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 δ (respect. σ) the *DIFS* (respect. 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 section and is included in our probabilistic model.

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: *i)* similar to the idea of the "shrew" attack [13], 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; *ii)* even if the RTO is not caused (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 will transmit a TCP-ACK at a given time slot and only jams it 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 (see Appendix) 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.

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. 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.

B. Jamming algorithm

Algorithm 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 in practice 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 network information, in particular, 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[1]. 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 δ depending on the frame type (Lines 9-15, where the function $Delay(x)$ 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:

Algorithm 1 The jamming attack

```
1:  Initialization
2:  B := 0, P := n − 1, C := 0;
3:  CounterOn := False;        ▷ True if first TCP Data is received
4:  TwoTimers := False;        ▷ True if two timers in use
5:  Execute the following for each elapsed slot
6:  switch Event do
7:      case Busy channel AND P = 0
8:          P := m + n − 1;
9:      case A frame spanning φ from AP heard
10:         B− := φ, P+ := φ; ▷ Restore B, P to values before the
        frame is heard
11:         if The frame is DATA then
12:             Delay(δ + r + σ);
13:         else
14:             Delay(δ);
15:         end if
16:     case CounterOn = False AND TCP-Data to target heard
17:         CounterOn := True;
18:         Delay(δ + r + σ);
19: if CounterOn = True then
20:     switch Event do
21:         case MAC-ACK to target
22:             Goto 1;
23:         case Jammer's MAC-ACK time out
24:             if TwoTimers = True then
25:                 B := B_old, P := P_old, C := C_old;
26:             end if
27:             C := C + 1;
28:             if C ≥ c then
29:                 C_old := C, B_old := B, P_old := P;
30:                 TwoTimers := True;
31:                 Goto 1;
32:             end if
33:         case MAC-ACK to jammer AND TwoTimers = True
34:             TwoTimers := False;
35:         case P ≤ 0
36:             Send jamming signal with probability q(B);
37:             P := m + n − 1;
38: end if
39: B+ := 1, P− := 1;
40: if TwoTimers = True then B_old+ := 1, P_old− := 1;
41: end if
```

MAC-ACK to the target (Line 21-22): means the target has most probably successfully transmitted a TCP-ACK to the AP, and the previous jamming failed. 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.

Jammer's MAC-ACK times out (Line 23-34, 40): A time-out of the jammer is indication of a collision at the AP (i.e.,

[1]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 constantly generate TCP data with an MTU of 1500 bytes and ACKs of 40 bytes, including YouTube videos.

the AP did not send the MAC-ACK). By default, the jamming signal is to be resent after the collision, which is undesirable considering that the jamming is 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 timer, and conversely, if the jamming is unsuccessful, then there is a high probability that the target did not reset its timer, and is still counting down in a further backoff stage. Hence every time C exceeds c, while keeping the old values 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, in which case, the jammer should continue to work on the new jamming attack procedure.

Expiry of the period timer (Line 35-37): Since corrupting one single bit is sufficient for the frame to 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 jamming, so that even if the target sends out its TCP-ACK at the earliest available opportunity, the tail of the ACK 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: $p(B) = \Pr[\text{target starts transmission in } [B-n+1, B+m-1] \text{ slots}]$. Although the calculation of this probability is central to the effectiveness of our jamming attack, for ease of exposition, we relinquish the derivation of $p(B)$ to the appendix. In each period, priority is given to normal data frames over jamming signals, 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. This can be achieved by the QoS of 802.11 to set the CWmin and CWmax of the AC assigned to the jamming signals to 1.

C. Calculation of $q(x)$

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. The details of the derivation of $p(x)$ and $q(x)$ are shown in the Appendix.

D. Other issues

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 the same.

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 [14]. 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 and the jamming signals also occupy some of the jammer's 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 side, and eventually worse than with no jamming. When multiple normal nodes also exist on the jammer's side, 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 attack does not need to be redesigned to achieve this.

E. 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 against TCP-ACKs, 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:

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 [3], 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 (average backoff of the AP) times a parameter less than 1. We test this value on our jamming to examine its vulnerability to such detection method.

RTT: Assuming the AP is able to measure the end-to-end TCP RTT, simply estimating this 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 discussed 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 [15], could be a sign of the existence of a jammer, albeit not a conclusive one.

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. This may be invoked as a tool for detection.

TABLE I: Simulation parameters

Frame payload	1500 bytes
Basic rate for ACK and MAC header	6 Mbps
minRTO	1s
Slot time	9μs
m	27 slots
n	4 slots
s	7 slots
r	5 slots
c	7
CW_{min}	15
CW_{max}	1023

We will examine all these aspects in our numerical study.

III. Simulation

We validate the effectiveness of our attack model via network protocol simulation in ns-2 [16] in a wired-cum-wireless network (we are currently building the attacker in a real WLAN node). IEEE802.11 standard with basic access mode is used in the simulation. The parameters used are shown in Table I. 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 and the wired links have 200Mbps bandwidth, making the WLAN link be the bottleneck. Each run of the simulation lasts 100 second, and 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.

A. A case in detail

For simplicity we consider a 1-on-1 case with $w = 0.6$, and examine the throughput (Fig. 2), TCP sender's window (Fig. 3) and RTT (Fig. 4, samples taken every 0.1 second) of both the jammer and the target's TCP traffic.

From these figures and other runs of simulation, we can arrive at some key observations. First, as shown in Fig. 3 between 21s and 26s for the jammer and between 62s and 78s for the target, both the jammer and the target may experience an RTO, but the chance is not high for either one. In fact there are more cases with no RTOs than those with RTOs. Also with many runs, we observed that the jammer suffers much less RTOs than the target as the jamming constantly delays the

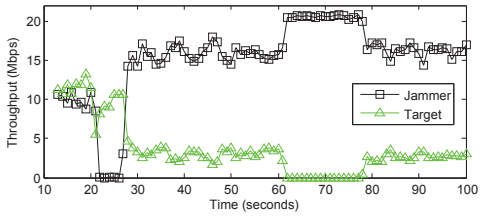

Fig. 2: Throughput of jammer and target nodes

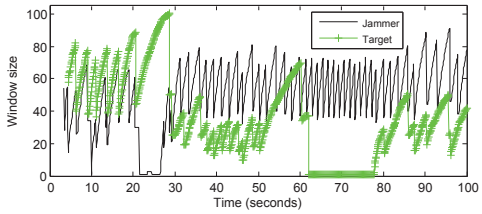

Fig. 3: TCP sender's window of jammer and target nodes

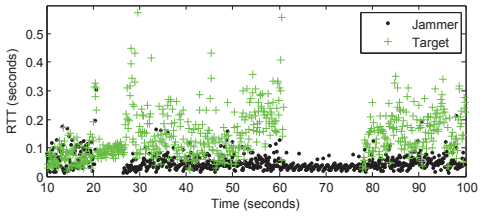

Fig. 4: Samples of round-trip time of jammer and target nodes

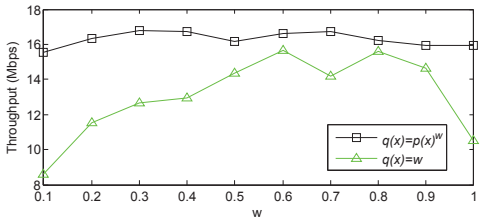

Fig. 5: Jammer's throughput comparison: for different $q(x)$

transmission of the target's TCP-ACK, and if the target reaches an RTO, on average it takes longer, if ever, than the jammer to recover. When the target times out, the jammer can quickly take over the bandwidth with its own TCP traffic. Second, even when the target does not experience an RTO, its RTT, due to the delay caused by the jamming, is much longer than that of the jammer, making its TCP sender's window grow at a much slower rate than that of the jammer. This can be observed from the slopes in the growth of the sender's windows in Fig. 3. Especially, when the target times out between 62s and 78s, the jammer's window grows particularly fast. This will eventually lead to a low average window and a low average throughput for the target.

B. Overall effect of jamming

Once we understand how the jamming works, we can evaluate the effectiveness of jamming. Let us first investigate the validity of basing $q(x)$ on $p(x)$ and the influence of parameter w. In Fig. 5, we compare the average throughput of the jammer for a constant transfer function $q(x) = w$ vs. the power function used in our model $q(x) = p(x)^w$. Each case consists of 100 averaged simulation runs. From the figure we can see that our model achieves much better throughput.

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 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.

Throughput: Table II shows the throughput achieved under attack with different settings, in Mbps. 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, 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 normal nodes with a colluding jammer.

TABLE II: Throughput simulation results

Setting	Jammer (Mbps)	Jammer gain	Gain Jammer-side nodes	Targets average (Mbps)
1/1	16.64	58%	N/A	2.03
1/2	13.43	287%	N/A	2.20
1/3	6.40	234%	N/A	2.84
1/4	4.00	153%	N/A	2.43
1/5	1.58	33%	N/A	4.02
2/2	6.55	23%	55%	2.14
2/3	2.07	-30%	155%	2.57
3/3	1.98	-49%	52%	2.01

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

Setting	1/1	1/2	1/3	1/4	1/5	2/2	2/3	3/3
A	16.9	19.1	41.6	46.6	64.4	38.5	92.8	61.9
B	8.21	8.05	8.02	8.01	8.01	8.23	8.11	8.18

Estimated backoff window: Table III shows the estimated backoff window of the jammer and the nominal backoff window in slots, (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 [3], 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. Further careful

TABLE IV: Average RTT: simulation results

Setting	1/1	1/2	1/3	1/4	1/5	2/2	2/3	3/3
Jammer (ms)	57	65	83	95	104	65	85	77
Target (ms)	145	158	164	172	181	128	135	127

TABLE V: Number of collisions α: simulation results

Setting	1/1	1/2	1/3	1/4	1/5	2/2	2/3	3/3
Jam.	0.4	0.7	1.6	2.1	2.7	0.7	1.1	1.0
No jam.	0.2	0.3	0.3	0.3	0.2	0.4	0.6	0.6

analysis based on packet inspection, transmissions and collisions is needed to achieve a valid detection scheme. Another possible misbehaviour, "Shorter than DIFS" in the DOMINO mechanism is also not valid for the attack due to the jamming algorithm mentioned above.

RTT: Table IV shows the average RTT in millisecond for both sides in the simulation, displaying a large gap between them. However, as the round-trip delay for both streams is 12ms, the jammer's TCP traffic also experiences a large delay. Thus using the difference in RTT as an indication of jamming is not effective. 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.

Number of collisions: Finally, Table V compares the number of collisions experienced under jamming and no jamming, where α is the number of collisions divided by the number of frames correctly received by the AP (rounded to one decimal). From the comparison we can say that α seems to be a good indication 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.

IV. Conclusion

Jamming forms an important part of WLAN security issues, 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, in this paper we introduced a new jamming attack on hidden targets against downlink TCP traffic. 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 increasing the targets' RTT. 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. We are currently focused on implementing and deploying this attack in a real wireless network node to test it in real networks.

Appendix

Notice that the countdown must take into account all possible sample paths that lead to a given estimated number of slots. For example, if the timer of the target up till a given slot is 20 slots of backoff, the target could have experienced in its first transmission 10 backoff slots drawn from CW_{min} yet failed, then is two-thirds through counting down 15 backoff slots in the first retry drawn from $2CW_{min}$; or, it could have tried 5 backoff slots and failed in the first transmission, then tried 9 backoff slots and failed again in the first retry, and is half-way through trying 6 slots in the second retry drawn from $4CW_{min}$. Obviously these two possible sample paths occur with different probabilities since the retransmission process has no steady state, and all are taken into account in the model.

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

$$\begin{aligned} p(x) &= \sum_{i=1}^{m+n-1} \sum_{j=0}^{c} \Pr[A(x-n+i,j), \\ &\quad B(x-n+1, x-n+i-1) | C(x-n+1)] \end{aligned} \quad (1)$$

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 in the sum 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 (1) by conditional probability we have

$$p(x) = \sum_{i=1}^{m+n-1} \sum_{j=0}^{c} \frac{\Pr[D(x-n,i,j)]}{\Pr[C(x-n+1)]}. \quad (2)$$

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

$$\begin{aligned} \Pr[D(x,i,0)] &= \Pr[(x+i) \text{ chosen as backoff} \\ &\quad \text{from the minimum contention window}] \\ &= \begin{cases} 1/(CW_{min}+1) & \text{if } x+i \le CW_{min} \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \quad (3)$$

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 $E(x,j)$,

$$\begin{aligned} \Pr[D(x,i,j)] &= \sum_{k=1}^{\min\{x-1, x+i-n-s\}} \Pr[E(k,j-1)] \\ &\quad \times \Pr[\text{collision at } k] \\ &\quad \times b(x+i-k-n-s, j-1), \end{aligned} \quad (4)$$

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

$$b(i,j) = \begin{cases} 1/(CW_j+1) & \text{if } i \in [0, CW_j] \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

with CW_j being the size of the jth contention window, i.e. $CW_j = 2^j(CW_{min}+1)-1$.

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

$$\begin{aligned} \Pr[\text{collision at } k] &= 1-(1-q\,(d(k-1))) \\ &\quad \times (1-\mathbb{B}(\tau+1))\,, \end{aligned} \quad (6)$$

where $\mathbb{B}(n)$ is the probability of collision for n nodes in Bianchi's model [17], τ is the number of stations on the targets' side and $d(x)$ is the slot where the last jamming decision before the xth slot is made,

$$d(x) = \left\lfloor \frac{x}{m+n-1} \right\rfloor \times (m+n-1)+n-1. \quad (7)$$

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

$$\begin{aligned} \Pr[E(x,j)] &= \sum_{k=1}^{x-n-s} \Pr[E(k,j-1)] \times \Pr[\text{collision at } k] \\ &\quad \times b(x-k-n-s, j-1), \end{aligned} \quad (8)$$

with the base case being

$$\Pr[E(i,0)] = \begin{cases} 1/(CW_{min}+1) & \text{if } i \le CW_{min} \\ 0 & \text{otherwise;} \end{cases} \quad (9)$$

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

By substituting (5), (6) and (8) into (4) we have a recursive form on $\Pr[D(x,i,j)]$. The denominator in (2), $\Pr[C(x-n+1)]$ can also be represented by $\Pr[E(x,j)]$:

$$\begin{aligned} \Pr[C(x)] &= \sum_{k=1}^{x-1} \sum_{j=0}^{c} \Pr[E(k,j)] \times \Pr[\text{collision at } k] \\ &\quad \times a(x-k-n-s, j) + t(x), \end{aligned} \quad (10)$$

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

$$a(i,j) = \begin{cases} \frac{(CW_j-i+1)}{(CW_j+1)} & \text{if } CW_j \ge i \text{ and } i > 0 \\ 1 & \text{if } i \le 0 \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

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

$$t(x) = \begin{cases} \frac{(CW_{min}+2-x)}{(CW_{min}+1)} & \text{if } x \le CW_{min}+1 \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

By putting (3), (4) and (10) into (2) we obtain the value of $p(x)$. Note that in the calculation of $p(x)$, previously calculated values of $q(x)$ with smaller x 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.

REFERENCES

[1] Z. Lu, W. Wang, and C. Wang, "On order gain of backoff misbehaving nodes in csma/ca-based wireless networks," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[2] D. Thuente and M. Acharya, "Intelligent jamming in wireless networks with applications to 802.11 b and other networks," in *Proc. of IEEE MILCOM*, vol. 6, 2006.

[3] M. Raya, I. Aad, J.-P. Hubaux, and A. El Fawal, "Domino: Detecting mac layer greedy behavior in ieee 802.11 hotspots," *Mobile Computing, IEEE Transactions on*, vol. 5, no. 12, pp. 1691–1705, 2006.

[4] K. Pelechrinis, M. Iliofotou, and S. V. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *Communications Surveys & Tutorials, IEEE*, vol. 13, no. 2, pp. 245–257, 2011.

[5] P. Kyasanur and N. Vaidya, "Selfish mac layer misbehavior in wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 4, no. 5, pp. 502–516, 2005.

[6] S. Radosavac, J. S. Baras, and I. Koutsopoulos, "A framework for mac protocol misbehavior detection in wireless networks," in *Proceedings of the 4th ACM workshop on Wireless security*, pp. 33–42, ACM, 2005.

[7] A. L. Toledo and X. Wang, "Robust detection of selfish misbehavior in wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 6, pp. 1124–1134, 2007.

[8] Y. Rong, S. K. Lee, and H.-A. Choi, "Detecting stations cheating on backoff rules in 802.11 networks using sequential analysis.," in *INFOCOM*, Citeseer, 2006.

[9] J. Choi, A. W. Min, and K. G. Shin, "A lightweight passive online detection method for pinpointing misbehavior in wlans," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 12, pp. 1681–1693, 2011.

[10] D. P. Blinn, T. Henderson, and D. Kotz, "Analysis of a Wi-Fi hotspot network," in *Workshop Wireless Traffic Meas. Model.*, pp. 1 – 6, Jun 2005.

[11] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren, "Usage patterns in an urban WiFi network," *IEEE/ACM Trans. Netw.*, vol. 18, pp. 1359 – 1372, October 2010.

[12] A. Proano and L. Lazos, "Selective jamming attacks in wireless networks," in *Communications (ICC), 2010 IEEE International Conference on*, pp. 1–6, IEEE, 2010.

[13] A. Kuzmanovic and E. W. Knightly, "Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 75–86, ACM, 2003.

[14] K.-L. Hung and B. Bensaou, "Throughput analysis and bandwidth allocation for ieee 802.11 wlan with hidden terminals," *Journal of Parallel and Distributed Computing*, vol. 71, no. 9, pp. 1201–1214, 2011.

[15] P. Sessini and A. Mahanti, "Observations on round-trip times of tcp connections," *SIMULATION SERIES*, vol. 38, no. 3, p. 347, 2006.

[16] K. Fall and K. Varadhan, "The ns manual (formerly ns notes and documentation)," *The VINT project*, vol. 47, 2005.

[17] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 3, pp. 535–547, 2000.