Stealthy Off-Target Coupled-Control-Plane Jamming

Shreya Gupta, Chia-Yi Yeh, Edward W. Knightly
Department of Electrical and Computer Engineering, Rice University, Houston, Texas, USA
{sg106, cy20, knightly}@rice.edu

***Abstract*—Multi- and single-user beamforming is a key feature for realizing a high data rate in next-generation Wi-Fi such as IEEE 802.11ax. In this paper, we study for the first time, a jammer that strategically attacks layer two (L2) control frames associated with beamforming to realize a denial-of-service (DoS) attack on transport (L4) and application layer (L7) control planes. By coupling the attack across control plane layers, the attacker targets throughput or availability DoS while also maintaining stealth via low air time jamming and "off-target jamming" in which the targeted higher layer message is never directly jammed. With end-to-end application layer experiments, we show that such a jammer can reduce TCP throughput to 1% with less than 0.1% of jamming air-time. Moreover, the attack can yield seconds-to-minute scale outages by targeting the L4 or L7 setup messages while leaving a minimal footprint.**

***Index Terms*—coupled control plane mechanisms, control frame jamming, stealthy jamming**

I. Introduction

Wireless jamming has been the subject of intense prior research [1]–[6]. For example, half-duplex Wi-Fi jammers have been shown to be capable of selectively jamming frame types, by decoding the header and preamble, and then jamming the remaining portion of the frame [7]. Likewise, data frame jamming coupled with the generation of fake ACKs has been demonstrated to give a false impression of successful data transmission [8].

In this paper, we study for the first time, a jammer that targets coupled control plane functions in order to simultaneously realize DoS and stealth objectives. In other words, the jammer seeks to radically reduce network throughput and deny access to network resources while restricting jamming air time to extremely small values and never directly jamming the higher-layer target. We consider a strong adversary that can reverse engineer control plane functions spanning layers two to seven and exploit their interactions in order to realize an effective attack. In particular, we make the following contributions.

We begin by describing the system and threat model. We consider that a server Alice is transmitting to clients, Bobs, who are connected to a WLAN that employs multi- and single-user beamforming, as in the IEEE 802.11ax Wi-Fi standard [9]. The jammer (Mallory) is in range of Bob and can identify and selectively jam Wi-Fi control frames such as beamforming setup messages. Yet, Mallory's target is not the layer two control itself but is rather the control planes of layers four and seven. For example, if Mallory targets a TCP SYN message, she will not jam the message itself, but will rather jam the layer two control messages surrounding the target in order to cause a failed or aborted transmission. Hence, we term Mallory as an *off-target jammer*, since she never jams her ultimate target directly. We consider that Mallory has three possible targets: TCP congestion control, TCP connection establishment, and HTTP session establishment. In all three cases, she attempts to disrupt layer four or seven (L4 or L7) control by jamming L2 control messages.

We implement a testbed that enables end-to-end experiments using commercial web servers and Linux protocol implementations for Alice and Bob. For the 802.11ax capable WLAN and jammer, we use the PERFORM real-time emulator [10], [11]. We perform an extensive set of experiments that yield the following findings: First, we establish an experimental baseline in which Mallory solely attacks L2 control *without* coupling to higher layers. Namely, we consider uncontrolled fully backlogged UDP traffic between Alice and Bob. We find that jamming different control frame types can radically transform Mallory's throughput vs. jamming air-time trade-off with group messages being the most vulnerable, as they can thwart an entire multi-user transmission. Nonetheless, the threat is quite mild, as Mallory only reduces throughput to 48% when she jams alternate control messages yielding a 0.7% jamming air-time.

Next, we consider a stronger adversary that targets to couple her attack on L2 and L4 control. In particular, we define the *Multi-User Control to Congestion Control* (*M2C*) attack as one in which Mallory jams multi-user L2 control messages (Null Data Packet or NDP messages) in order to send false severe congestion indication information to the L4 sender. We show that Mallory can effectively disable TCP's fast retransmit mechanism and force slow retransmission timeout (RTO) recoveries. The attack yields a surprising non-monotonicity in which Mallory can simultaneously reduce her jamming airtime and reduce network throughput. The result yields a severe DoS attack in which Mallory can reduce throughput to below 1% with less than 0.1% air time, i.e., Mallory retains stealth by jamming less than $1/1000^{\text{th}}$ of the time.

Third, we study an L4 connection availability attack in which Mallory targets the TCP 3-way handshake (SYN, SYN-ACK, ACK) by jamming the associated L2 beamforming control frames. We find that Mallory can force a timeout of the TCP connection establishment process with only 0.0157% jamming air time, corresponding to milliseconds of jamming over a minute-scale attempt procedure. In this case, Mallory's small air-time footprint is aided by Alice's slow recovery due to Alice's high initial RTO value of one second (Alice has no round-trip-time measurements to set RTO to twice the measured round-trip-time) and Alice's subsequent doubling of

RTO increments upon failure.

Finally, we consider an attack in which Mallory allows the TCP connection to be established and instead targets the subsequent layer 7 control messages, namely, HTTP session establishment messages. We show that Mallory can force the session establishment procedure to time out for the disrupted HTTP response. Moreover, if Mallory eventually allows the session to be established, she delays the video playback by an additional 30 seconds using only 0.01% air-time jamming. We find that while Alice's first HTTP response retransmission is relatively rapid (e.g., 0.28 seconds), her subsequent response arrives at the AP 2.35 seconds later, enabling Mallory to maintain a very low air-time footprint.

The remainder of the paper is organized as follows. Section II-B describes the threat model and Mallory's jamming scheme. The testbed and experimental setup are described in Section III. Experimental results for the baseline of non-coupled control planes and solely L2 jamming are presented in Section IV. Experimental results for coupled controlled planes are presented in Sections V, VI, and VII for respective targets of TCP congestion control, TCP connection establishment, and HTTP session establishment. Finally, Section VIII presents related work and Section IX concludes.

II. Scenario and threat model

In this section, we describe the network scenario, threat model, and performance metrics.

A. End-to-end network scenario with 802.11ax

We consider the network topology as shown in Figure 2, where M Wi-Fi client stations (Bob$_1$ to Bob$_M$) access the Internet through a single Wi-Fi access point (AP). Each client has a full protocol stack and utilizes an end-to-end networked application, such as a web browser making an HTTP(s) connection from a server (Alice). The AP and the stations employ features from the latest 802.11ax standard, also known as the Wi-Fi 6 [12]–[14], in which the AP employs multi-user (MU) beamforming for the downlink data transmissions. The AP can likewise employ single-user beamforming when there is only backlog for one station.

B. Threat model

1) Overview: We consider an adversary, Mallory, who can overhear all Wi-Fi transmissions. Moreover, she can identify and selectively jam any L2 frame according to the frame control message type. In particular, since the MAC header is not encrypted, Mallory can overhear the beginning of the MAC header and choose whether to jam the rest of the frame, a capability that has been experimentally demonstrated in prior work [1], [7], [15], [16]. In addition to the ability of identifying L2 frame types, we consider that the strong adversary can infer which type of L4 or L7 information will be transmitted along with the associated L2 control message. For example, if the L4 target is a TCP SYN-ACK message, we consider that Mallory can identify the L2 control frames associated with this message. Because this payload is typically encrypted, Mallory's identification will be imperfect in practice. For example, Mallory might estimate that an appropriately sized and temporally isolated uplink frame corresponds to a TCP SYN message so that the subsequent downlink transmission will correspond to her targeted TCP SYN-ACK. Nonetheless, here we consider that Mallory's identification process is perfect in order to assess the worst-case damage that she can do.

2) L2 Beamforming Control: Here, we first review the Wi-Fi MU downlink transmission timeline with control and data frames. Figure 1 illustrates the L2 timeline for Wi-Fi MU downlink transmissions from the AP to 4 stations. The Wi-Fi MU beamforming downlink transmission procedure begins with a null data packet announcement (NDPA) frame (green) to gain channel control and identify the selected stations. Next, the AP sends a null data packet (NDP) frame (blue) for the stations to estimate their channel state information, control information that is used by the AP for beamforming. The stations then feedback their channel state information to the AP using the compressed beamforming report (CBR) frame (purple). The beamforming feedback poll report (BF-POLL) frame (peach) coordinates the timing and order of the channel feedback from the different stations participating in the multi-user transmission. The data transmission comprises parallel transmissions to multiple stations, with interference among the parallel transmission enabled by the AP's use of channel state information together with mechanisms such as zero-forcing beam-forming [17]. After data transmission, the AP sends an ACK request (or Block ACK request) frame (grey) to coordinate the acknowledgment and the stations reply with ACK (or Block ACK) frames (yellow) to indicate successful data delivery.

While Figure 1 shows a successful MU downlink transmission, unsuccessful transmission due to collision will trigger Wi-Fi contention window doubling and increase the retransmission count. With consecutive failures, once the retransmission limit is reached, the AP will discard the data frame from its queue. Compared to single user transmissions, MU transmission failure impacts more data streams and thus becomes the preferred target for Mallory [5]. Nonetheless, if only a single user is backlogged, the AP uses the same procedure for single-user beamforming and Mallory can jam this setup as well.

3) Off-target jamming: Next, we describe Mallory's strategy of jamming only L2 control messages for MU downlink transmissions, since the transmission duration of control frames is typically shorter than data frames by two orders of magnitude considering frame aggregation [9], [18]. To disrupt the L2 data frame transmission which contains messages from higher layers, Mallory exploits the Wi-Fi retransmission mechanism which discards the data frame after the maximum retrial limit [15]. To this end, Mallory must *(i)* jam the beamforming setup frame preceding the data transmission, and *(ii)* jam consecutively until the maximum retrial limit so that the data frame is discarded without transmission. We consider that Mallory is aware of the AP's maximum retrial limit for control frame retransmission, which is seven by default according to the 802.11 standard.

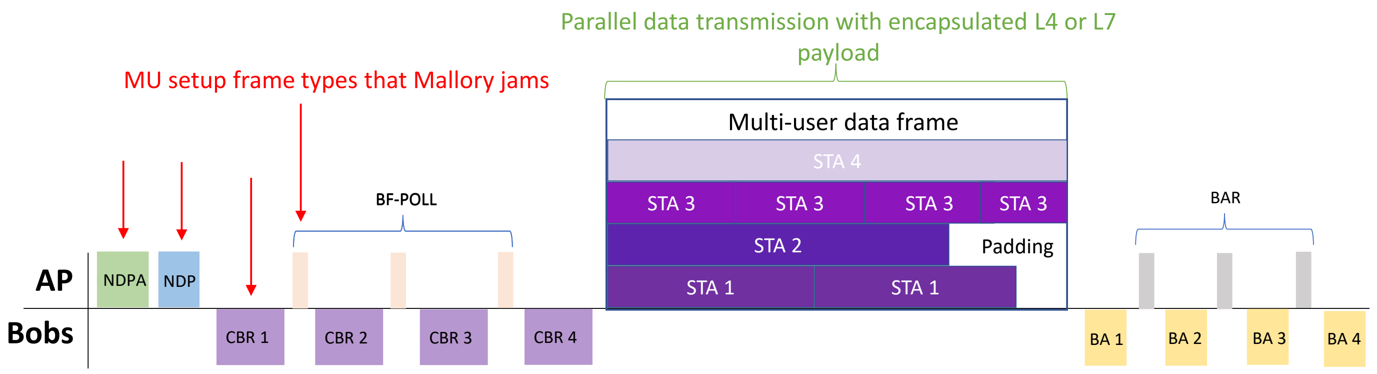

Fig. 1: Communication timeline at the link layer for multi-user downlink transmissions that Mallory exploits to jam the beamforming setup control frames.

Mallory can target any control frame type in the beamforming setup prior to the data transmission, yet, when she targets the individual control frames (CBR or POLL), she must jam all CBR or POLL frames corresponding to each user in the multi-user transmission, as opposed to jamming a single frame for the group control frames (NDPA or NDP). For instance, when Mallory targets the CBR frames in the example in Figure 1, she needs to jam all 4 CBR frames to stop the MU transmission. If she jams only a subset of CBR frames, the multi-user transmission will proceed without the jammed stations. However, jamming all POLL frames is not a successful attack for Mallory as the AP receives the CBR from the first user. Hence, the network switches to single user mode, with successful transmission for the first user.

Mallory controls her air-time exposure by using a wait cycle between jamming epochs. We define the wait cycle w as the number of uninterrupted non-jammed MU downlink transmissions that Mallory allows before she resumes jamming. In this way, a longer wait cycle w corresponds to less intense jamming and in the extreme case that $w = 0$, Mallory jams all control frames of the selected type.

Thus, Mallory's jamming strategy can be summarized as follows:

1) Mallory selects a control frame type from NDPA, NDP, or CBR that she will jam.
2) Mallory jams the selected control frame for 7 consecutive MU downlink transmissions so that the corresponding data frames in the AP's queue are discarded.
3) Mallory pauses jamming and allows w MU downlink transmissions before she resumes jamming.
4) Mallory repeats steps (2) and (3) to alternate between jamming and waiting for the entire attack duration.

C. Coupled control plane jamming with three targets

Using the off-target jamming strategy defined in the previous subsection, Mallory aims to launch DoS with three representative targets: *(i)* TCP data for throughput DoS, *(ii)* TCP setup for L4 availability DoS, and *(iii)* HTTP setup for L7 availability DoS. For each of the three DoS attacks, Mallory targets a control mechanism in L4 or L7.

1) TCP congestion control: Since TCP interprets segment loss as a congestion indicator, Mallory targets to trigger false severe congestion indication by purposely jamming L2 control frames before the multi-user data frames encapsulating TCP payload to force the AP to discard those data frames. Missing TCP segments trigger TCP retransmission, fast recovery, congestion window reduction, and retransmission timeout (RTO). With consistent off-target jamming, Mallory can launch a TCP throughput DoS attack.

2) TCP connection establishment: Mallory can also target the TCP handshake process, delaying or even preventing the TCP connection from being established. TCP responds slowly to a missing TCP handshake message and only retransmits after the retransmission time out (RTO) timer expires. Moreover, the missing TCP handshake message further triggers an increase in RTO. Thus, when Mallory targets TCP handshake message consecutively, Mallory can delay TCP connection establishment or prohibit entirely with repeated jamming.

3) L7 establishment: Similar to L4 availability DoS, Mallory can target HTTP control messages to launch a L7 service availability DoS attack.

Since HTTP operates over TCP, Mallory first allows the TCP connection to be established in this attack. After waiting for TCP connection establishment to succeed, she subsequently delays or prevents HTTP session establishment by jamming the L2 control frames associated with the HTTP control messages. Consequently, HTTP fails to deliver application layer content (text, video, image) to Bob, effectively causing an application layer service outage.

D. Metrics for attacks and exposure

To quantify Mallory's detriment to the Alice-Bob connections, we use two metrics: network throughput and connection or session establishment delay. In particular, we use the network throughput metric when Mallory launches a TCP throughput DoS attack, whereas the connection delay metric is used when Mallory prevents L4 or L7 availability. From Mallory's perspective, for the attack to be successful, Mallory either causes a large throughput degradation so that the connection is unusable for Alice and Bob, or Mallory causes a long delay or

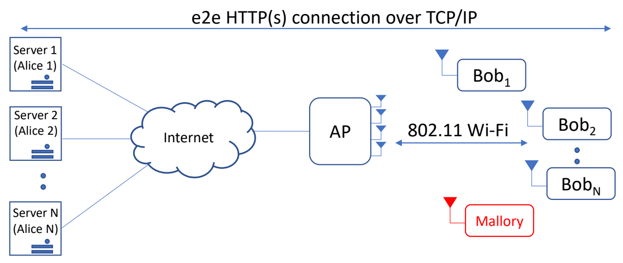

Fig. 2: Threat model: End-to-end network scenario with servers, an AP, and Bobs. Mallory jams Wi-Fi control messages to disrupt higher layer control plane functions.

timeout for TCP or application connection establishment and thus she creates a significant availability outage.

Since Mallory also aims to remain stealthy while launching the DoS attack, we use the jamming air-time percentage to quantify Mallory's exposure. We consider the entire transmission time of the jammed L2 control frames as jamming time such that the percentage of air-time jammed is the total air time of jammed L2 control frames divided by the total air time, expressed as a percentage.

III. Implementation and Experimental Platform

Here, we describe our implementation of Mallory's capabilities in the PERFORM WLAN testbed, along with the experimental setup for end-to-end experiments using real elements spanning from commercial web servers to commercial client web browsers.

A. *PERFORM WLAN emulation*

To experimentally study the impact of jamming across networking layers, we conduct the experiments using the PERFORM testbed [10], [11], which allows real-world Internet traffic experiments by emulating the link layer frame exchanges.

Unfortunately, to realize an over-the-air Mallory encounters the obstacle that commercial Wi-Fi modules are preprogrammed and thus cannot be easily modified unless with laborious reverse engineering [19]. Although some APs provide access to PHY [20] parameters such as the signal-to-noise ratio, received signal strength, and modulation, they are not sufficiently accessible and programmable for Mallory's purposes.

In contrast, PERFORM enables full control and observably of Wi-Fi, including Wi-Fi features such as MIMO transmissions, channel contention with binary exponential backoff, while simultaneously being embeddable with real clients and the Internet to yield end-to-end real-time network traffic integration. Namely, PERFORM is capable of routing traffic from any of the networking layers. For example, if a client wants to open Netflix on a web browser, PERFORM can receive the traffic from the Netflix server and forward it to the client, emulating the station to AP Wi-Fi link.

The basic idea behind PERFORM is to emulate WLAN in infrastructure mode using a high-speed wired LAN and precise timing control. The platform is implemented in C code on a Linux kernel based on in-built libraries (IPTABLES/NFQUEUE) for traffic filtration. To implement the complete Wi-Fi module, PERFORM consists of four major parts:

1. **Commodity devices:** PERFORM allows physical or virtual 802.11 devices to act as Wi-Fi stations.
2. **Physical connection:** Ethernet cables are used to connect the AP and the stations for traffic exchange.
3. **Queue manipulation:** The platform deploys 4 queues, 2 at the AP and 2 at the stations, to store uplink and downlink Wi-Fi traffic. A real-time scheduler releases the Wi-Fi frames in these 4 queues according to the 802.11 standard timing.
4. **MAC protocol implementation:** PERFORM provides programmable user-space that allows implementations of MAC protocols for a given 802.11 standard using C programming.

B. *Coupled control plane jamming implementation*

We customize the C code to implement Mallory's jamming functionality on PERFORM. We implement the MAC protocols from the latest 802.11ax with MU-MIMO. For simplicity, we use MU-MIMO in the downlink only while the uplink is single user.

The platform abstracts the PHY model such that the true physical channel is the LAN cable between the AP and the stations. We note that even though the Ethernet cable between the two computers can provide Gbps rates, the emulator code runs a scheduler to match the Wi-Fi timing. Unlike wireless connections, the Ethernet cable ensures minimal losses due to the physical environment. Thus, any network degradation comes solely from the impact of emulated jamming.

C. *Experimental setup*

We conduct all experiments on PERFORM for the scenario where all stations download the same TCP network application from a local or distant web server and PERFORM allows the traffic flow between AP and stations using the Wi-Fi timings. We use virtual machines (VMs) as the stations. PERFORM sends all data emulating Wi-Fi 6 MAC features such as multi-user beamforming and MIMO in the downlink, all traffic being best-effort.

The PHY parameters for the MIMO channel are full-rank, and thus the aggregate rate is proportional to the number of users in the network. For simplicity, we use MCS zero for all the Wi-Fi frames, including the data frame. The data sent at MCS zero emulates the transmissions of aggregated data frames with longer duration. Note that tuning the control and data frames to different MCS values will result in different frame airtime and system throughput, yet, once normalized, the throughput degradation caused by jamming remains similar.

We implement selective jamming for the three control frame types preceding the MU data transmission: Null data packet announcement (NDPA), null data packet (NDP), and compressed beamforming report (CBR), as described in Section

II-B. We configure channel sounding to precede every (multi-user/single-user) L2 downlink data transmission, and every jamming attempt by Mallory is successful.

Each user establishes an independent TCP/UDP flow with a server connected to the AP, either using Iperf3 or accessing a website. For TCP, the AP sends the TCP traffic from the Iperf3 server to the users using downlink multi-user beamforming after connection establishment. As UDP does not have a 3-way handshake, the downlink data flows start immediately. We record the transmissions under jamming for 60 seconds. For throughput experiments, we use the download throughput reported by Iperf3 as the network throughput. For the web server DoS experiments, we use commercial web browsers and servers. We obtain the delay measurements for the TCP and HTTP availability DoS attacks from the Wireshark traces.

IV. Mallory's Attack efficiency at layer 2

We begin with experiments in which Mallory attacks only the layer two control plane, *without* coupling the attack to other network control functions. These experiments serve as a baseline for studying more sophisticated attacks by quantifying layer two vulnerabilities in isolation. Thus, here, we consider fully backlogged UDP downlink traffic in which Mallory jams the beamforming control frames preceding the multi-user data transmissions, as their failure can prevent transmission to all stations included in the multi-user frame. We study the jamming of different types of control frames and different jamming frequencies and measure the corresponding throughput degradation and air-time exposure.

Experimental design and setup: We configure a scenario with 1 AP and 4 stations, all receiving UDP data, as shown in Figure 1. UDP traffic allows studying the L2 jamming effects without the additional L4 congestion control effects as in closed-loop TCP traffic, which is Mallory's ultimate target in Section V. We conduct the experiments on PERFORM with one computer acting as the AP while the other computer has four virtual machines to emulate Wi-Fi stations, as described in Section III-C. We use Iperf3 to generate one UDP flow per station, resulting in downlink traffic from the AP to all stations in the WLAN. To study jamming multi-user WLAN scenarios, we configure the AP to employ multi-user data transmissions to the stations.

The UDP transmissions last for 60 seconds, and Mallory begins jamming after 5 seconds when the network reached a steady state. Mallory selects one of the three MU setup frames: NDPA, NDP, and CBR. As described in Section II-B, and jams the selected control frames 7 times consecutively: As per the 802.11 standards, the AP will drop the multi-user data frames after a maximum retransmission limit, 7 in this case, and hence Mallory's choice. After the 7 consecutive jams, Mallory lets w subsequent messages pass without jamming so that a larger w indicates a less aggressive attack. That is, after jamming, Mallory has a wait cycle, w which, for this set of experiments, varies from 1 to 10. If $w = 0$, Mallory jams all of the selected MU setup frames, which in this configuration, would yield zero throughput. To quantify network performance degradation and Mallory's exposure, we use throughput, reported by Iperf3, and jamming air time, calculated from the number of jammed frames.

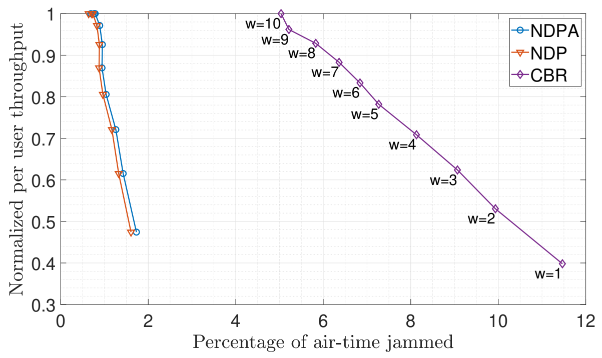

Fig. 3: Jamming beamforming setup control frames for quantifying throughput and air time on UDP traffic in a multi-user WLAN

Results: Figure 3 depicts average normalized throughput as a function of the percentage of air time jammed. The blue, orange, and purple curves show results when Mallory targets NDPA, NDP, and CBR, respectively. Throughput is normalized to the case of no jamming as described in Section II-D. The wait cycle values from $w = 1$ to $w = 10$ are displayed on the CBR (purple) plot only, and they follow the same order for the other three curves.

First, observe that the NDP and NDPA curves are well separated from the CBR curve and that Mallory's attack is far more effective for NDP/NDPA than CBR. This is because they can be categorized as a group versus individual control messages. In particular, if Mallory jams an NDP or NDPA frame, the entire MU transmission is thwarted. In contrast, if Mallory jams a CBR frame, then only the particular user reporting CSI will be thwarted.

Next, consider the NDPA curve and note the trade-off between throughput degradation and air time: With aggressive jamming and a short wait cycle of 1 control frame ($w = 1$), the throughput reduces by more than 50% of the no jamming value while consuming 1.7% of the air time. Yet, even with aggressive NDPA jamming and $w = 1$, Mallory's attack is largely ineffective and Mallory would need to jam all NDPA frames ($w = 0$) to reduce the throughput to zero.

When Mallory jams the other beamforming setup frame types (CBR), the same throughput vs. air time trade-off exists but with different characteristics. For example, for a particular throughput degradation, jamming NDP and NDPA frames result in a similar air time, whereas jamming CBR requires over four times more air time.

In addition, the slope of the CBR curve is less steep indicating that for a similar change in throughput, Mallory must invest more air time. For example, throughput difference for NDPA and CBR from $w = 10$ to $w = 1$ is similar, 48% and 40% respectively. However, the air time difference for NDPA is only 1% (0.7% vs 1.7%), while it is significantly higher for

CBR, 6.5% (5% vs 11.5%).

The performance impact from jamming different frame types arises for two reasons: First, each control frame has a different length, from shortest to longest being NDP, NDPA, and CBR. More importantly, frame types such as NDPA and NDP occur only once in the beamforming setup, whereas CBR occurs multiple times, requiring Mallory to jam more of them to thwart the entire downlink transmission.

Findings: *Mallory can realize the largest throughput degradation for a particular air-time exposure by jamming NDPA/NDP frames instead of CBR, as NDP/NDPA controls the entire MU transmission whereas CBR frames control an individual station. This difference will therefore amplify with increasing group size. Most critically, when Mallory attacks only a single control plane at layer two, she fails to realize an effective attack, since even aggressive jamming with wait cycle* $w = 1$ *only decreases network throughput to approximately* 40%.

V. DoS Coupling Multi-User Control and Congestion Control

Here, we study Mallory's ability to attack TCP's congestion control algorithm via jamming L2 control frames. Her objective is to reduce TCP throughput with limited jamming air time exposure. We refer to her attack as Multi-User Control to Congestion Control DoS *M2C*.

Experimental design and setup: We use the same experimental setup as previously, with two exceptions: First, all traffic is TCP download with TCP data on the downlink and TCP ACKs on the uplink. Second, because we found in Section IV that the strongest threat is when Mallory jams NDP frames, we henceforth consider only NDP jamming.

Because TCP uses loss as a congestion indicator and consecutive losses as a severe congestion indicator that can impede fast recovery and cause timeouts [21], Mallory jams NDPs corresponding to two consecutive segments. Thus, we define Mallory's *M2C* attack as targeting consecutive L4 losses by jamming 14 consecutive NDPs such that Alice is forced to drop two consecutive multi-user frames. Mallory exploits that the jammed NDPs correspond to TCP payload so that Mallory triggers a false severe congestion indicator. To maintain stealth, Mallory waits until w NDPs have been successfully transmitted before repeating the attack. As a baseline, we also implement and study *single loss jamming* in which Mallory jams 7 consecutive NDPs corresponding to a single TCP segment loss (per stream) before waiting for w.

Results: Figure 4 depicts average throughput vs. percentage of air-time jammed. The orange curve depicts the case for Mallory's *M2C* strategy of jamming NDPs corresponding to two consecutive segments before waiting, whereas the blue curve depicts the baseline strategy in which Mallory jams NDPs corresponding to only a single TCP segment before waiting.

First, observe that compared to targeting only layer two control with UDP traffic (Figure 3), *M2C* is not only a more severe DoS threat (throughput is significantly lower), but is also significantly more stealthy as jamming air time is decreased by

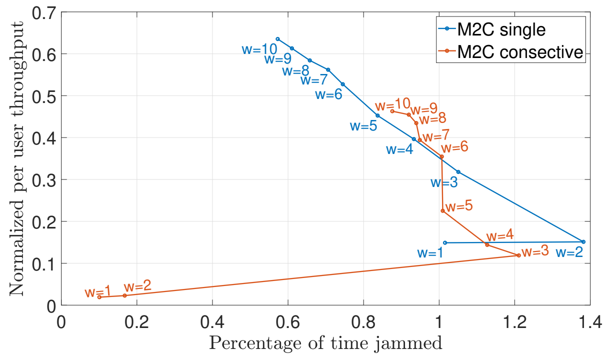

Fig. 4: Jamming NDP frames for congestion control DoS

an order of magnitude. For example, with $w = 1$ wait cycle, Mallory can only reduce UDP throughput to 48%, whereas for the *M2C* attack coupling to congestion-controlled traffic, she reduces throughput to 2%. Moreover, for $w = 1$ and both variants of *M2C*, Mallory requires only 0.1% air-time which is less than $10\times$ the air-time used for jamming UDP with the same w. The dissimilarity stems from TCP's recovery mechanism from the false severe congestion indicator that causes the TCP congestion window to be cut in half twice, requires two TCP retransmissions, and potentially results in a TCP time out. Yet, despite the severity of the attack, by the design of *M2C*, Mallory never jams TCP data or TCP ACKs directly, but only jams associated L2 control frames. That is, she jams "off target."

Second, recall that in the experiments solely targeting L2 control with UDP traffic, jamming air time always *decreases* with wait cycle, as Mallory's increased pauses in jamming reduce her total time spent jamming. In contrast, with *M2C*, jamming air-time is not monotonic with w. In particular, once Mallory decreases the wait cycle to below $w = 3$, the air time *decreases* as does TCP throughput, yielding an increasingly effective attack by both metrics, even as Mallory becomes more aggressive. Wireshark analysis reveals that when $w \leq 3$, Alice cannot utilize fast transmissions. Indeed, the number of fast retransmissions reduces, as w decreases from 10 to 3 and is 0 for $w \leq 3$. Thus, for $w > 3$, even after two consecutive discarded TCP segments, the wait cycle allows the TCP sender to utilize fast retransmission and rapidly recover.

When $w \leq 3$, most retransmissions occur after the TCP ACK timer (retransmission timeout or RTO) expires. RTO is set to an initial value of 0.02 seconds in our TCP implementation. Moreover, after a timeout, TCP infers severe congestion, sets its congestion window to one, and enters a slow start. Thus, because the sender is now transmitting data much slower, Mallory needs very little jamming air time to subsequently keep throughput low and the congestion window small.

Finally, we study the importance of jamming NDPs corresponding to consecutive segments vs. individual segments. The blue curve in the figure depicts this baseline in which Mallory jams 7 consecutive NDPs (corresponding to a single segment) before waiting. First, observe that even with single segment jamming, Mallory can still trigger TCP's severe congestion indicators and realize the same non-monotonicity described above, albeit less effectively and with an inflection point at when $w = 2$. For example, with $w = 1$ and single-segment jamming, Mallory reduces throughput to 14% with jamming air time 1.01%. In contrast, using *M2C* targeting two consecutive segments, Mallory further disrupts TCP's congestion control algorithm exploiting its vulnerability to consecutive losses and reduces throughput to 1.0% with a jamming air time of 0.1%.

Findings: *Mallory's attack successfully couples the L2 control plane and L4 control plane to reduce throughput to below* 1.0% *with a minimal footprint of less than* 0.1% *jamming air time. She realizes this regime by triggering false indicators of severe congestion, effectively disabling fast retransmit, and forcing Alice to rely on RTOs to retransmit segments. Moreover, despite Mallory's target of TCP segments, she maintains stealth and low air time by never directly jamming segments, but rather jamming L2 NDP control messages. Surprisingly, the attack effectiveness is non-monotonic with Mallory's pauses between attacks and she no longer needs to have a high wait time to keep her total jamming time small.*

VI. TCP 3-Way Handshake DoS

With the *M2C* attack, while Mallory radically reduces network throughput, Alice and Bob are still able to communicate, albeit at a very low rate. Here, we study a DoS attack on availability in which Mallory uses L2 control frame jamming to prevent TCP's connection establishment. In particular, as with the *M2C* attack, Mallory never directly jams the targeted messages, which in this case, is TCP's 3-way handshake used to establish the connection. Instead, Mallory repeatedly jams the beamforming control frames that precede the handshake messages that are to be transmitted on the AP's downlink.

Experimental design and setup: As the TCP handshake is a three message exchange process, Mallory can target any one of those messages to prevent TCP connection establishment. Here, we show experiments only with Mallory targeting the first message, i.e., the SYN, as it is transmitted from the AP to the stations. We use the same experimental setup as previously, except that Mallory only jams NDPs preceding SYN messages. In this setup (without other background traffic) SYNs arrive at the AP asynchronously and are unlikely to be grouped into a multi-user transmission. In such cases, the AP employs single-user beamforming which employs the same setup procedure as multi-user, and hence Mallory retains her strategy of jamming NDP control frames.

To prevent a successful TCP handshake, Mallory must have no wait cycle for this target, i.e., $w = 0$. Thus, Mallory prevents connection establishment for as long as she jams NDPs associated with SYNs. Hence, we use average connection establishment latency as a key performance metric for the damage done by Mallory in this attack. We define the connection establishment delay as the difference between the time when Iperf3 starts and the time when the actual data transmission starts. Mallory controls the jamming air time by selecting the

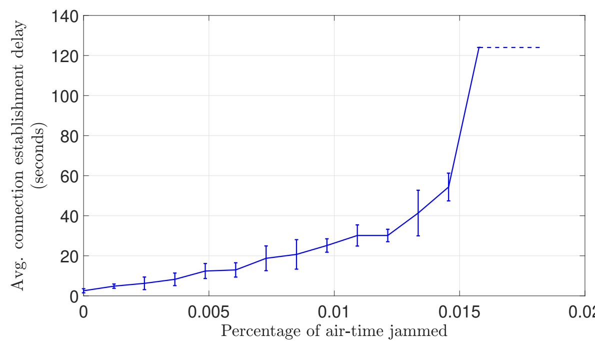

Fig. 5: Impact of jamming NDPs corresponding to TCP SYN handshake message for TCP connection establishment delay

number of NDP frames to be jammed, always in multiples of 7 to ensure layer two loss.

Results: Figure 5 shows the average TCP connection establishment delay in seconds vs. the fraction of air time jammed. Each experiment is repeated 5 times and the standard deviation is also depicted.

First, observe that Mallory can realize second- to minute-scale outages with only a small fraction of air-time exposure. For example, in the lowest jamming point on the curve, Mallory can delay connection establishment by over 5 seconds with only 0.001% of air time corresponding to 0.6 ms of total jamming time. She realizes this attack by jamming 7 NDPs corresponding to the SYN requests before she stops jamming and Mallory is successful.

Second, the curve gradually increases until approximately 60 seconds and has a sharp upward trend thereafter: After transmitting each SYN, Alice expects a SYN-ACK reply. When it is not received after RTO, Alice retransmits the SYN and increases RTO starting from the initial value of 1 second. With every SYN retransmission, the RTO increases from 1 to 3, 7 seconds, etc., until approximately 64 seconds. The default TCP implementation in Linux [22] allows 6 SYN retries that correspond to an RTO value of 64 seconds. If Mallory continues to jam Alice's sixth retransmitted SYN, then Alice times out the connection. This corresponds to the flattening of the curve at 124 seconds, in which the connection is never established. Although 6 SYN retries correspond to approximately 64 seconds, the maximum timeout in our results is 124 seconds.

Findings: *By attacking multi-user control frames associated with TCP connection establishment, Mallory can deny service indefinitely (timing out the establishment process) with only 0.0157% jamming air time. Likewise, she can delay connection establishment by seconds to minutes with even less air time due to Alice's default one second initial RTO value and her subsequent increments of doubling values upon failure.*

VII. Application Layer Control-Plane DoS

Analogous to the L4 availability attack, here we study an attack in which Mallory uses the L2 control plane to target application layer control messages. In particular, Mallory targets HTTP session setup messages by jamming the preceding NDP frames.

Experimental design and setup: The experimental setup differs from previous experiments in two ways: First, Mallory jams the NDPs preceding HTTP response messages rather than TCP SYNs. Second, unlike traffic generated via Iperf3 in previous experiments, Mallory attacks traffic generated by a web server. In particular, the client (Bob) attempts to access the web page *cluthcitycf.com* via HTTP. When Bob enters a website URL into the browser, a DNS probe first converts the site name in the URL to the server's (Alice's) IP address and Alice and Bob establish a TCP connection. Mallory does not attempt to disrupt these initial procedures, and she instead waits until after they have been completed to target the subsequent HTTP response to the HTTP GET message. Mallory controls the jamming air time by selecting the number of NDP frames to be jammed, always in multiples of 7 to ensure layer two loss.

Results: Figure 6 shows the average and standard deviation of web page loading time in seconds vs. percentage of air time jammed.

First, observe that the maximum delay for establishing the HTTP session is approximately 142 seconds, corresponding to 9 GET response transmissions and re-transmissions. As HTTP operates over TCP, when an HTTP control message is dropped (due to NDP jamming), TCP retransmits it as a TCP segment. In our experiments, when the HTTP response message is first sent, the webserver can have an RTO value below the 1 second default, as it can set it to twice the round-trip-time that the server measured during the (non-attacked) TCP session establishment handshake. Upon attack, the server will rapidly increase RTO. In one experiment, we observe the retransmitted responses arriving at the AP with increases by nearly an order of magnitude from the initial value. For example, the retransmission delays observed by repeated HTTP responses at the AP are 0.28, 2.35, 6.75, and up to 90 seconds. After nine attempts, the server aborts the HTTP response.

Finally, Mallory also disrupts the playback delay of the short video that is part of the clutchcity.com home page. In particular, we measure the playback delay at Bob's Google Chrome browser and find that in each jamming experiment, the video on the home page loads approximately 30 seconds after the arrival of the GET response. With no jamming, this time difference is 5 seconds on average. The HTML script of a website has multiple GET requests for different elements (text, images, videos, etc.) on the website, and each has a different order of execution in the JSON file designing script. If an HTTP response is delayed, the order and the timing of the other GET requests and responses are also affected. Thus, even when the HTTP response is successful, the video loading takes increased time. However, the text on the web page appears approximately at the same time as the GET response arrival observed in Wireshark.

Findings: *Mallory can delay access to a website by several seconds by jamming only the NDPs corresponding to the targeted HTTP control messages and she can delay video playback for more than one minute with only 0.01% air-time of jamming. In this attack, Mallory allows the TCP connection to be established before jamming. Consequently, Alice can set her RTO corresponding to twice the estimated round trip time and rapidly respond to the first jam in only 0.28 seconds. However, after timing out, Alice's second retransmission arrives 2.35 seconds later. This enables Mallory to remain inactive for long periods of time, keeping her total jamming time minimal, while also denying service to Bob.*

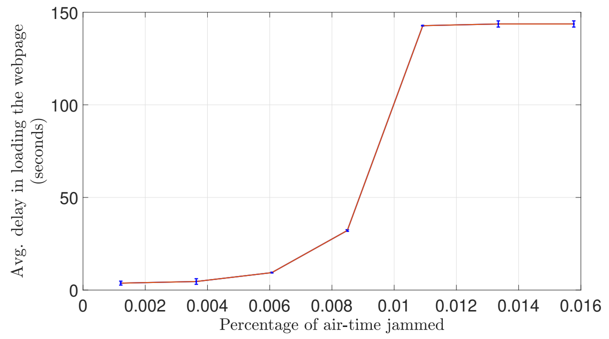

Fig. 6: The delay in application content delivery to the end-user when Mallory jams NDPs corresponding to the HTTP payload.

VIII. Related Work

L2 jamming. Many prior works have focused on jamming vulnerability in 802.11 networks, including analysis of jamming transmissions of data or control information [6], [8], [23]–[25]. Prior work showed that similar throughput degradation can be achieved by targeting control frames preceding data transmissions [1], [3]–[5]. Moreover, if Mallory jams symbols from the channel setup frames [2], [26]–[28], she can reduce the achievable rates for the beamforming transmissions or even thwart the communication leading to denial-of-service. In addition, to control frame jamming, prior work also demonstrated MAC DoS via short jamming pulses sent periodically to mislead legitimate nodes into assuming the channel is busy or occupied [29]–[31]. Such jamming schemes can significantly reduce the packet delivery ratio, thus reducing the throughput. Strategies that combine jamming and fake L2 frame injection to deceive the clients into believing a false network state were also studied [8]. In contrast to prior work that studies only link-layer impacts, we for the first time demonstrate the coupling of layer 2 control with control up to layer 7.

Cross-layer attacks and analysis. Another class of prior work focused on attacking the TCP protocol by jamming the TCP control packets like TCP ACKs (uplink or downlink or TCP SYN-ACKs that can severely reduce the throughput or even terminate the TCP connection [32]. Moreover, the effect of TCP throughput has been shown to reduce by jamming the link layer control frames like RTS, CTS, MAC ACK [15], [33], [34]. In comparison, in our study, Mallory never jams TCP control packets. Instead, she jams a layer two control message that will be used to transmit such TCP messages. In this way, we focus on coupled control algorithms.

IX. Conclusion

We present the first study of coupled control plane off-target jamming in which Mallory can disrupt L2, L4, and L7 control mechanisms by jamming only the L2 beamforming setup frames. We demonstrate jamming-based experimental results on PERFORM, with real-time Internet traffic. We study three types of off-target DoS attacks under a low air-time regime and show that the adversary can successfully launch a DoS attack that reduces performance by orders of magnitude at L4 and L7 with air time less than 0.1%.

Acknowledgements

This research was supported by Cisco, Intel, NSF grants CNS- 1955075, CNS-1923782, CNS-1824529, CNS-2148132 and DOD: Army Research Laboratory grant W911NF-19-2-0269.

References

[1] F. Klingler and F. Dressler, "Jamming wlan data frames and acknowledgments using commodity hardware," in *Proceedings of IEEE Workshop on Conference on Computer Communications*, Paris, France, 2019.

[2] X. Zhang and E. W. Knightly, "Pilot distortion attack and zero-startup-cost detection in massive mimo network: From analysis to experiments," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 12, pp. 3094–3107, 2018.

[3] A. Moussa and I. Jabri, "Impact of RTS/CTS jamming attacks in IEEE 802.11ah dense networks," in *Proceedings of IEEE IWCMC*, 2021, pp. 1551–1556.

[4] D. Thuente and M. Acharya, "Intelligent jamming in wireless networks with applications to 802.11b and other networks," in *Proceedings of MILCOM*, vol. 6, 2006, p. 100.

[5] G. Patwardhan and D. Thuente, "Jamming beamforming: A new attack vector in jamming IEEE 802.11ac networks," in *Proceedings of IEEE MILCOM*, 2014, pp. 1534–1541.

[6] L. Zhang, F. Restuccia, T. Melodia, and S. M. Pudlewski, "Jam sessions: Analysis and experimental evaluation of advanced jamming attacks in mimo networks," in *Proceedings of ACM Mobihoc*, Catania, Italy, 2019, p. 61–70.

[7] H. Rahbari, M. Krunz, and L. Lazos, "Security vulnerability and countermeasures of frequency offset correction in 802.11a systems," in *Proceedings of IEEE INFOCOM*, Totonto,Canada, 2014, pp. 1015–1023.

[8] W. Kim, S. Kim, and H. Lim, "Malicious data frame injection attack without seizing association in IEEE 802.11 wireless LANs," *IEEE Access*, vol. 9, pp. 16 649–16 660, 2021.

[9] "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (MAC) and physical layer (phy) specifications amendment 1: Enhancements for high-efficiency wlan," *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*, pp. 1–767, 2021.

[10] V. D. S. Goncalves and E. W. Knightly, "An experimental study of triggered multi-user uplink access with real application traffic," in *proceedings of IEEE/ACM International Symposium on Quality of Service (IWQoS)*, 2022.

[11] V. Da Silva Goncalves, "Perform: A platform for experimental research in wlan with focus on real network traffic and multi-user channel access," Master's thesis, Rice University, 2020.

[12] E. Khorov, I. Levitsky, and I. F. Akyildiz, "Current status and directions of IEEE 802.11be, the future Wi-Fi 7," *IEEE Access*, vol. 8, pp. 88 664–88 688, 2020.

[13] J. Oh, H.-J. Hong, and H.-D. Choi, "Performance analysis for channel sounding in IEEE 802.11ac network," in *Proceedings of IEEE ICTC*, 2015, pp. 1240–1242.

[14] T. T. Thao Nguyen, L. Lanante, Y. Nagao, M. Kurosaki, and H. Ochi, "MU-MIMO channel emulator with automatic channel sounding feedback for IEEE 802.11ac," in *Proceedings of IEEE WCNC*, 2016, pp. 1–6.

[15] A. Proano and L. Lazos, "Packet-hiding methods for preventing selective jamming attacks," *IEEE Transactions on dependable and secure computing*, vol. 9, no. 1, pp. 101–114, 2011.

[16] R. Miller and W. Trappe, "On the vulnerabilities of CSI in MIMO wireless communication systems," *IEEE Transactions on mobile Computing*, vol. 11, no. 8, pp. 1386–1398, 2011.

[17] E. Aryafar, N. Anand, T. Salonidis, and E. W. Knightly, "Design and experimental evaluation of multi-user beamforming in wireless LANs," in *Proceedings of ACM MobiCom*, Chicago, Illinois, 2010.

[18] I. . W. Group *et al.*, "Part 11: wireless lan medium access control (MAC) and physical layer (phy) specifications: higher-speed physical layer extension in the 2.4 ghz band," *ANSI/IEEE Std 802.11*, 1999.

[19] M. Vanhoef and F. Piessens, "Advanced Wi-Fi attacks using commodity hardware," in *Proceedings of the 30th Annual Computer Security Applications Conference*, 2014, pp. 256–265.

[20] S. Balakrishnan, S. Gupta, A. Bhuyan, P. Wang, D. Koutsonikolas, and Z. Sun, "Physical layer identification based on spatial–temporal beam features for millimeter-wave wireless networks," *IEEE Transactions on Information Forensics and Security*, 2019.

[21] M. Allman, V. Paxson, and E. Blanton, "Tcp congestion control," Tech. Rep., 2009.

[22] "Linux networking documentation." [Online]. Available: Available: https://docs.kernel.org/networking/ip-sysctl.html

[23] D. Nguyen, C. Sahin, B. Shishkin, N. Kandasamy, and K. R. Dandekar, "A real-time and protocol-aware reactive jamming framework built on software-defined radios," in *Proceedings of ACM Workshop on Software Radio Implementation Forum*, 2014.

[24] H. Pirzadeh, S. M. Razavizadeh, and E. Björnson, "Subverting massive mimo by smart jamming," *IEEE Wireless Communications Letters*, vol. 5, no. 1, pp. 20–23, 2016.

[25] R. Chinta, T. F. Wong, and J. M. Shea, "Energy-efficient jamming attack in IEEE 802.11 MAC," in *Proceedings of IEEE MILCOM*, 2009, pp. 1–7.

[26] X. Zhou, B. Maham, and A. Hjorungnes, "Pilot contamination for active eavesdropping," *IEEE Transactions on Wireless Communications*, vol. 11, no. 3, pp. 903–907, 2012.

[27] T. T. Do, H. Q. Ngo, T. Q. Duong, T. J. Oechtering, and M. Skoglund, "Massive MIMO pilot retransmission strategies for robustification against jamming," *IEEE Wireless Communications Letters*, vol. 6, no. 1, pp. 58–61, 2016.

[28] S. Gvozdenovic, J. K. Becker, J. Mikulskis, and D. Starobinski, "Truncate after preamble: PHY-based starvation attacks on IoT networks," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 89–98.

[29] A. Benslimane, M. Bouhorma *et al.*, "Analysis of jamming effects on IEEE 802.11 wireless networks," in *Proceedings of IEEE ICC*, 2011, pp. 1–5.

[30] A. Hussain, N. A. Saqib, U. Qamar, M. Zia, and H. Mahmood, "Protocol-aware radio frequency jamming in Wi-Fi and commercial wireless networks," *Journal of Communications and Networks*, vol. 16, no. 4, pp. 397–406, 2014.

[31] A. Ahmed, U. Ashraf, F. Tunio, K. Abu Bakar, and M. S. AL-Zahrani, "Stealth jamming attack in WSNs: Effects and countermeasure," *IEEE Sensors Journal*, vol. 18, no. 17, pp. 7106–7113, 2018.

[32] T. X. Brown, J. E. James, and A. Sethi, "Jamming and sensing of encrypted wireless ad hoc networks," in *Proceedings of ACM MobiHoc*, 2006, pp. 120–130.

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

[34] A. Proaño and L. Lazos, "Selective jamming attacks in wireless networks," in *Proceedings of IEEE ICC*, 2010, pp. 1–6.