Off-Path TCP Hijacking in Wi-Fi Networks: A Packet-Size Side Channel Attack

Ziqiang Wang
Southeast University
ziqiangwang@seu.edu.cn

Xuewei Feng
Tsinghua University
fengxw06@126.com

Qi Li
Tsinghua University
qli01@tsinghua.edu.cn

Kun Sun
George Mason University
ksun3@gmu.edu

Yuxiang Yang
Tsinghua University
yangyx22@mails.tsinghua.edu.cn

Mengyuan Li
University of Toronto
alyssamengyuanli@gmail.com

Ke Xu
Tsinghua University
xuke@tsinghua.edu.cn

Abstract

In this paper, we unveil a fundamental side channel in Wi-Fi networks, specifically the observable frame size, which can be exploited by attackers to conduct TCP hijacking attacks. Despite the various security mechanisms (*e.g.,* WEP and WPA2/WPA3) implemented to safeguard Wi-Fi networks, our study reveals that an off path attacker can still extract sufficient information from the frame size side channel to hijack the victim's TCP connection. Our side channel attack is based on two significant findings: (i) response packets (*e.g.,* `ACK` and `RST`) generated by TCP receivers vary in size, and (ii) the encrypted frames containing these response packets have consistent and distinguishable sizes. By observing the size of the victim's encrypted frames, the attacker can detect and hijack the victim's TCP connections. We validate the effectiveness of this side channel attack through two case studies, *i.e.,* SSH DoS and web traffic manipulation. Furthermore, we conduct extensive measurements to evaluate the impact of our attack on real-world Wi-Fi networks. We test 30 popular wireless routers from 9 well-known vendors, and none of these routers can protect victims from our attack. Also, we implement our attack in 80 real-world Wi-Fi networks and successfully hijack the victim's TCP connections in 69 (86%) evaluated Wi-Fi networks. We have responsibly disclosed the vulnerability to the Wi-Fi Alliance and proposed several mitigation strategies to address this issue.

1 Introduction

Nowadays, public Wi-Fi networks are widely available in various places, such as airports, coffee shops, hotels, and libraries. To prevent a malicious supplicant (*i.e.,* an attacker) from eavesdropping in wireless channels, security measures (*e.g.,* WEP, WPA2 and WPA3) have been proposed to encrypt Wi-Fi networks at the link layer[1]. As a result, even if an attacker successfully sniffs wireless frames in the shared Wi-Fi channel, it cannot get useful information (*e.g.,* the random sequence and acknowledgment numbers of TCP connections) except for the 802.11 frame header.

However, in this paper, we demonstrate that the encrypted frame size constitutes a reliable side channel that can be exploited by attackers to conduct TCP hijacking attacks. Precisely, we discover that TCP packets can be identified by analyzing the size of the encrypted wireless frames, thus allowing an attacker residing in the same Wi-Fi network to infer the state of a victim TCP connection. By exploiting this side channel (*i.e.,* the encrypted frame size), the attacker can infer the random sequence and acknowledgment numbers of the victim TCP connection. Consequently, the attacker can pretend to be one peer of the victim connection to either terminate the connection or inject malicious data into the connection, *i.e.,* hijacking the connection completely.

Our attack consists of four steps. First, the attacker accesses a public Wi-Fi network and probes alive supplicants in the WLAN. The attacker crafts ARP requests in the WLAN to identify alive supplicants. By collecting the ARP replies, the attacker can obtain the <MAC, IP> address pair of each alive supplicant which is also a potential victim client of our TCP hijacking attack. Then through analyzing the MAC address field of the captured wireless frames in the shared Wi-Fi channels, the attacker can filter the encrypted frames belonging to the victim client. If the Wi-Fi network provides multiple access channels, the attacker can scan all Wi-Fi channels to filter the victim's frames. Once the victim's frames are sniffed, the attacker gains a potent side channel to conduct the TCP hijacking attack.

Armed with this side channel (specifically, the victim's encrypted frame size), the attacker can detect TCP connections issued by the victim supplicant through manipulating the challenge `ACK` mechanism [46]. The attacker impersonates the victim supplicant and sends forged `SYN/ACK` packets to the server. If a TCP connection exists between the victim supplicant and the server, the server will reflect a TCP challenge `ACK` packet to the supplicant. This challenge `ACK` packet (always encrypted as a 68-byte wireless frame at the link layer)

[1] Our study of 80 real-world Wi-Fi networks reveals that 64 of them are encryption enabled.

will be sniffed by the attacker at the shared Wi-Fi channel. By contrast, if no TCP connection exists, the attacker will not capture the 68-byte encrypted frame that carries the challenge ACK packet. Based on this key difference, the attacker can easily detect a target TCP connection between the identified victim supplicant and a remote server. Note that our attack does not directly exploit the vulnerability in the challenge ACK mechanism [14, 15]. Instead, we only use the challenge ACK mechanism as a trigger condition to assist our observations.

Third, the attacker infers the sequence number of the target TCP connection. The attacker pretends to be the victim supplicant and crafts TCP packets to the server. Those crafted TCP packets carry the guessed sequence numbers. If the guessed sequence number is less than the next sequence number to be received, the server will return a ACK packet carrying the SACK[2] option in the TCP header to the supplicant. The SACK option in the TCP header will consume extra bits within the wireless frame. In contrast, if the attacker specifies a sequence number greater than the next sequence number, the return ACK packet from the server will not carry the SACK option. This subtle difference (*i.e.,* the variation in frame size) can be observed by the attacker to infer the correct sequence number.

Fourth, the attacker proceeds to send forged ACK packets to the server, containing guessed acknowledgment numbers. If the specified acknowledgment number in the crafted TCP packet is below the server's accepted window, the server will reflect a challenge ACK packet (68-byte encrypted frame) to the victim supplicant. Otherwise, the server will discard the crafted packet or accepted it silently. By analyzing the size of the victim's encrypted frames, the attacker can easily infer the acknowledgment number of a target TCP connection. At this stage, the attacker has gathered all the necessary elements to hijack a TCP connection.

We conduct a comprehensive measurement study to show that our attack can be performed to cause serious damage in the real world, *e.g.,* terminating a victim SSH connection or poisoning a web traffic within 335 seconds. We test 30 popular wireless routers from 9 well-known vendors, and we discover that none of these routers can protect victims from our attack. Besides, we evaluate our attack in 80 real-world Wi-Fi networks, including most popular Wi-Fi scenarios (*e.g.,* Wi-Fi networks in coffee shops, bookstores, shopping malls, and restaurants). The experimental results show that 69 (86%) out of the 80 evaluated Wi-Fi networks are vulnerable to our TCP hijacking attack.

Finally, we have responsibly reported this vulnerability to the Wi-Fi Alliance. Currently, we are discussing the mitigation measures with the Wi-Fi Alliance. The root cause of this vulnerability lies in the fixed size of Wi-Fi frames at the link layer, which inadvertently creates a reliable side channel and leaks information about TCP connections. As a result, we propose two possible countermeasures: (i) Modifying the 802.11 standards and dynamically padding the encrypted frames to prevent information leakage. (ii) Revisiting the TCP specifications so that the server responds consistently to different conditions.

[2]Selective acknowledgment (SACK) is an option in TCP that allows a receiver to acknowledge non-contiguous blocks of data received from the sender [23].

Contributions. Our main contributions are as follows:

- We uncover a fundamental side channel in Wi-Fi networks, *i.e.,* the observable frame size, which is inherent in all generations of Wi-Fi standards.
- We show that this frame size side channel can be exploited by off-path attackers to infer the random sequence and acknowledgment numbers of TCP connections issued by victim clients in Wi-Fi networks, thus hijacking the target connections completely.
- We conduct an extensive investigation against 30 popular AP routers and 80 real-world Wi-Fi networks. The experimental results show that our attack can cause serious damage in the real world.
- We provide a thorough analysis on the root cause of the identified attack and discuss possible defenses to alleviate this attack.

Ethical Considerations. When we evaluate the impact of our attack in the real world, we carefully design and conduct the following experiments to avoid causing damage or negative impacts on operational Wi-Fi networks. Firstly, we provide a detailed explanation of our experimental procedure to the administrators and obtain their approval prior to conducting any tests. Secondly, our testing does not affect other supplicants or compromise the capabilities of the Wi-Fi network. Precisely, in the SSH DoS attack, we take our laptop as the victim client and utilize our cloud server as the SSH server. In the web manipulation attack, the poisoned client is under our control (*i.e.,* our laptop), and the web server is not affected. Third, we provide feedback to the administrators at the end of our experiments.

2 Background

This section begins with an introduction to the 802.11 frame format and the security mechanisms in Wi-Fi networks. Following that, we briefly review the challenge ACK mechanism and the TCP options that can be used to facilitate our attack.

2.1 Frame Format and Security Mechanisms in Wi-Fi Network

The 802.11 Frame Layout. Figure 1 shows the layout of an 802.11 frame. Firstly, the Frame Control (FC) field contains several flags and defines the type of frame. The Type and Subtype fields together identify the function of the frame.

There are currently three types (*i.e.,* management, control, and data frames) and more than 50 subtypes defined in 802.11 specifications. In our attack, the attacker needs to monitor the victim's TCP packets which will be encapsulated into 802.11 frames with type 2 and subtype 8 in Wi-Fi networks. To identify the victim's encrypted frames, the attacker needs to analyze the addresses of the 802.11 frames. There are four address fields in the MAC frame format. These fields are used to indicate the basic service set identifier (BSSID), source address (SA), destination address (DA), transmitting address (TA), and receiving address (RA). Certain frames might not contain some of the address fields. Certain address field usage is specified by the relative position of the address field (1–4) within the MAC header, independent of the type of address present in that field. Specifically, the Address 1 field always identifies the intended receiver(s) of the frame, and the Address 2 field, where present, always identifies the transmitter of the frame [3]. In our attack, the attacker can identify the victim supplicant's encrypted frames through addresses 1 (RA) and address 2 (TA). After filtering the victim's encrypted frames, the attacker needs to further analyze the payload size of the encrypted frames. The payload (*i.e.,* MSDU in Figure 1) of a normal data frame contains the upper layer data (*e.g.,* TCP packets). The MSDU typically starts with an LLC/SNAP header and is protected by cryptographic encapsulation mechanisms (*i.e.,* TKIP, CCMP, and GCMP). In this paper, we refer to the encrypted frame size as the MSDU size.

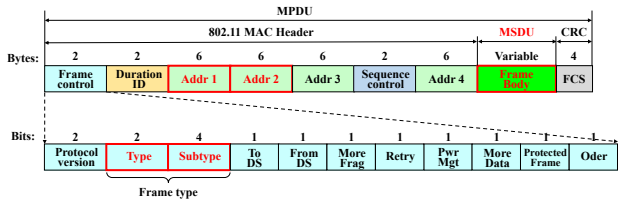


Figure 1: Layout of the 802.11 frame.

Security Mechanisms in Wi-Fi Network. When connecting to a Wi-Fi network, the supplicant initiates a four-step handshake with the access point (AP) to establish a distinctive random session key[3]. Subsequently, both the supplicant and the AP utilize this session key to encrypt Wi-Fi frames and transmit them over the wireless channel [3]. 802.11i [1] outlines the requirements and procedures for ensuring the confidentiality of user information during wireless transmission, as well as the authentication of devices conforming to the IEEE 802.11 standard. For an extended period, the security mechanisms employed by Wi-Fi networks (*e.g.,* WPA2 and WPA3) have primarily emphasized the improvement of confidentiality and data authentication. There has been a general belief that uncracked encrypted frames are secure. However, in this paper we show that the encrypted frame size inadvertently forms a side channel which leaks information about the victim applicant in the Wi-Fi network. It is worth noting that our attack does not sniff the four-step handshake frame to obtain the random session key. Instead, the attacker can directly exploit the size of encrypted frames within the Wi-Fi channel to launch a TCP hijacking attack.

[3]If the AP uses the outdated WEP encryption mechanism, there is no four-step handshake to negotiate the encryption key.

2.2 Challenge ACK Mechanism in TCP

Challenge ACK Mechanism. The challenge ACK mechanism, proposed in RFC 5961 [56], serves as a defense against blind in-window attacks carried out by off-path attackers. In essence, the challenge ACK mechanism introduces more stringent requirements for TCP segment acceptance, where the receiver expects the sender to respond with the precise sequence number instead of falling within the receive window. This effectively thwarts blind injection attacks by off-path attackers. However, we demonstrate that this mechanism can be exploited to infer TCP connection information in the following manner.

Our attack leverages the trigger conditions of the challenge ACK mechanism in two distinct ways. Firstly, when a receiver detects an incoming SYN packet within an established TCP connection, regardless of the sequence number, it responds by sending an ACK (referred to as the challenge ACK) to the remote peer. This ACK serves as a challenge for the remote peer to confirm the loss of the previous connection and the initiation of a new connection. Only the legitimate peer will receive this ACK and respond with a RST segment containing the correct sequence number, derived from the ACK field of the challenge ACK packet, in the event of connection loss. Consequently, a spoofed SYN packet will generate an additional ACK, which will be disregarded by the peer as a duplicate ACK and will have no impact on the established connection. We will demonstrate how this challenging condition can be exploited to detect a victim TCP connection.

Secondly, the receiver employs a verification process for the acknowledgment number of each TCP segment to prevent blind data injection attacks. Acceptance of an acknowledgment number for any data segment is contingent upon its falling within the range of $(SND.UNA - SND.WND, SND.NXT)$, where $SND.UNA$ represents the sequence number of the first unacknowledged octet, $SND.WND$ denotes the maximum window size observed by the receiver from the sender, and $SND.NXT$ is the next sequence number to be sent. If the acknowledgment number of the segment ($SEG.ACK$) is in the range $(SND.UNA - (2^{31} - 1), SND.UNA - SND.WND)$, the receiver responds with a challenge ACK. If the $SEG.ACK$ is greater than $SND.NXT$, the receiver silently discards this TCP segment. That can be exploited by attackers to infer the acceptable acknowledgment number, as described in Sec 4.3.3.

Table 1: TCP packet size analysis with IPv4.

Packet type	TCP options		Packet size (Byte)	Frame size (Byte)
	Time stamp	SACK		
RST	-	-	54	56
ACK	+	-	66	68
SACK-ACK	+	+	78	80

+ represents carrying the option, while - represents not carrying the option.

2.3 TCP Options

TCP options are supplemental fields that can be appended to the TCP header, offering added functionality and control. These options extend beyond the standard 20-byte TCP header and possess a variable size, not exceeding 40 bytes, contingent on the number of options included. Among the various TCP options available, the timestamp and selective acknowledgment options are the most commonly used.

Timestamp Option. The TCP timestamp option is defined in RFC 1323 [12]. It is widely used in modern operating systems and various studies [26, 30, 43]. The timestamp option field spans a size of 10 octets, encompassing the timestamp value and timestamp echo reply fields. In practice, the timestamp option is typically padded with two extra bytes to maintain alignment of the TCP header on a 32-bit boundary. In a TCP connection with timestamp functionality enabled, the `ACK` packet includes a timestamp value indicating its transmission time. This timestamp can be utilized by the sender to calculate round-trip time and estimate the current network state. However, `RST` packets, which are employed for connection termination and lack TCP header options like the timestamp option, possess different sizes compared to `ACK` packets. This disparity in size between `RST` and `ACK` packets is illustrated in Table 1. In this paper, we will show that the different size of the `ACK` packet and the `RST` packet can be used to infer the source port number of a target TCP connection.

Selective Acknowledgment Option. The TCP selective acknowledgment (SACK) is specified in RFC 2018 [23]. It is an optional feature that is typically enabled by default in the majority of TCP implementations. The SACK option is particularly recommended for networks experiencing frequent packet loss or packet reordering. Its utilization can significantly improve the performance and reliability of TCP connections in such environments. When a receiver detects a TCP segment with a sequence number that has already been acknowledged as outdated, it responds by sending a `SACK-ACK` to notify the sender. The SACK option will enable the sender to selectively retransmit lost packets based on the information provided by the receiver. However, if the sequence number of the received TCP segment has not yet been acknowledged, the receiver will reply with an `ACK` packet, which may have a different frame size (as shown in Table 1) or may not respond at all, depending on the acknowledgment number of the segment.

In this paper, we will show that attackers in Wi-Fi networks can differentiate between these situations and thus infer the sequence number of a target TCP connection by analyzing the size of sniffed wireless frames.

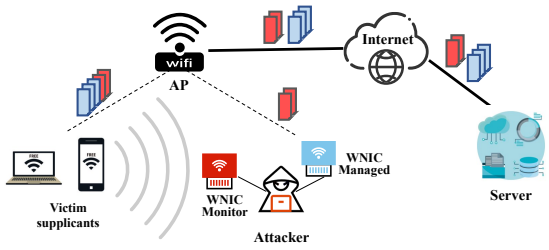


Figure 2: The threat model.

3 Threat Model

Figure 2 illustrates the threat model of our off-path TCP hijacking attack in Wi-Fi networks. The AP encrypts the network traffic of its supplicants via security mechanisms, *e.g.,* WPA2 or WPA3. A victim supplicant, such as a laptop or a smartphone, connects to the AP and establishes TCP connections with remote servers. The attacker, functioning as a regular supplicant without AP management privileges, utilizes multiple wireless network interface cards (WNICs). One (managed model) of these WNICs connects to the AP, while the others (monitor model) are utilized to sniff encrypted frames transmitted over the shared Wi-Fi channels. We make the assumption that the attacker has prior access to the target Wi-Fi network before performing our attack. This is a commonly accepted assumption in Wi-Fi hijacking scenarios, as highlighted in previous studies [22, 60].

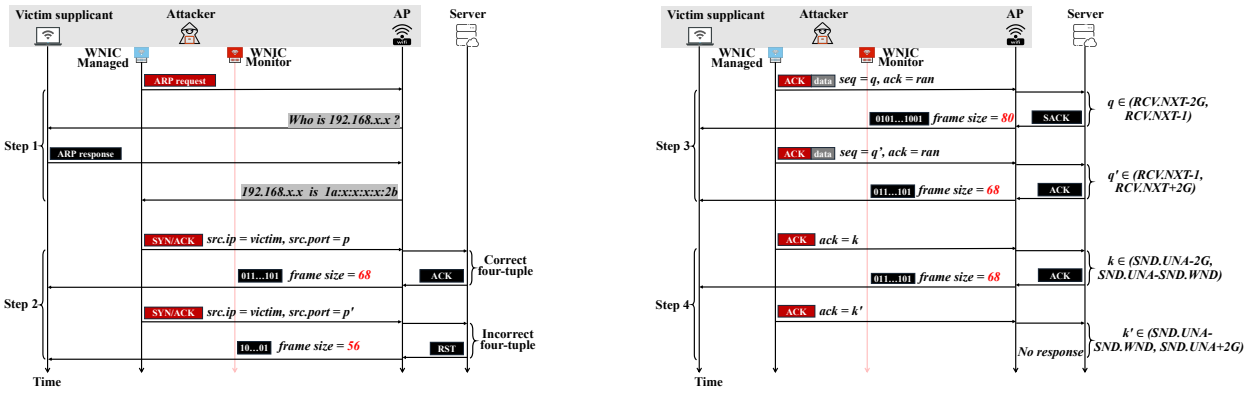
4 TCP Hijacking with Encrypted Frame

Our attack exploits two key aspects. Firstly, the TCP stack exhibits inconsistent responses during packet verification. Depending on the validity of the received packet, the TCP receiver generates four different responses: no response packet, a `RST`, an `ACK`, and a `SACK-ACK`. Due to the presence of TCP options, these responses can be distinguished based on their packet sizes. Secondly, the frames within the Wi-Fi network are observable, and the frame sizes of these responses are consistently fixed (see Table 1). These characteristics create a significant side channel. An attacker can leverage this side channel to detect and hijack the victim's TCP connection.

4.1 Attack Overview

Our TCP hijacking attack consists of four steps.

Step 1: Network Scanning. The attacker accesses a Wi-Fi network and scans the WLAN for potential victim supplicants.



(a) Network scanning and detecting TCP connections.

(b) Inferring sequence and acknowledgment numbers.

Figure 3: Outline of our off-path TCP hijacking attack.

In this step, the attacker identifies the <MAC, IP> pair of the victim to monitor its encrypted frames.

Step 2: Detecting TCP Connections. After detecting potential victims alive in the WLAN, the attacker impersonates the victim supplicant and sends forged SYN/ACK packets to the server. At the same time, the attacker monitors the victim's encrypted frames in the Wi-Fi channel. By analyzing the encrypted frame size, the attacker can determine if a TCP connection exists between the victim and the server.

Step 3: Inferring Sequence Number. After detecting a victim TCP connection, the attacker sends forged TCP packets with guessed sequence numbers to the server. These manipulated TCP packets prompt the server to generate SACK-ACK responses, which will be sniffed by the attacker when they (*i.e.,* 80-byte encrypted frames) are transmitted in the Wi-Fi channel. By monitoring the victim's encrypted frames, the attacker can identify the correct sequence number of the target TCP connection.

Step 4: Inferring Acknowledgment Number. With the inferred acceptable sequence number, the attacker proceeds to send forged ACK packets to the server. These ACK packets will trigger server's challenge ACK, which always appears as a 68-byte encrypted frame in the Wi-Fi network. By exploiting this challenge ACK, the attacker can locate the server's challenge ACK window and subsequently find an acceptable acknowledgment number.

After determining the sequence and acknowledgment numbers of the target TCP connection, the attacker can inject forged TCP packets into the connection with the intent to either terminate the connection or manipulate the data stream.

4.2 Network Scanning and Detecting TCP Connections

Network Scanning. The attacker first prepares the TCP hijacking attack from two aspects, *i.e.,* obtaining the <MAC, IP> address pair of the victim and identifying the Wi-Fi channel used by the victim. The attacker actively sends ARP requests in the WLAN to detect other alive supplicants (*i.e.,* the potential victim clients of our TCP hijacking attack). By observing the ARP responses, the attacker can learn the victim's MAC address and IP address. With the victim's MAC address, the attacker sniffs encrypted frames in the Wi-Fi channel and filters the victim's frames based on address 1 (or address 2) in the 802.11 MAC header (see Figure 1). If the Wi-Fi network supports multiple accessed Wi-Fi channels, the attacker scans all Wi-Fi channels to identify the specific channel used by the victim. Subsequently, the attacker intercepts encrypted frames within the target Wi-Fi channel and filters out the victim's frames.

Detecting TCP Connections. With intercepting and analyzing the victim's encrypted frames, the attacker can identify the victim's TCP connections. A TCP connection is recognized by four elements, *i.e.,* [client IP address, client port number, server IP address, server port number]. Typically, the server IP address and server port number are publicly known to the attacker [14, 58]. The attacker needs to infer the client's IP address and port number. In our attack, the client IP is obtained via ARP response. Thus, the only remaining element to determine is the client port number.

Given that a TCP connection was previously established by the legitimate user on a victim client using a source port p, the attacker impersonates as the client and sends forged SYN/ACK packets to the server. As per the challenge ACK mechanism described in RFC 5961, if the forged SYN/ACK packet contains the same client port number p, the server will respond with a challenge ACK to the client. This challenge ACK packet will be encapsulated into a 68-byte encrypted frame and sniffed by the attacker, during transmission from the AP to the client.

In contrast, when the client port number specified in the forged SYN/ACK packet is not equal to p, the server will reply with a RST packet. This RST packet is encapsulated within a 56-byte encrypted frame. Therefore, by examining the size

of the encrypted frame, as depicted in step 2 of Figure 3(a), the attacker can determine whether the guessed client port number is correct or not.

The attacker iterates through the above procedure by changing the client port number specified in the forged SYN/ACK packet. This procedure continues until the correct port number p is identified. Finally, the attacker identifies a target TCP connection operating on the four-tuple.

4.3 Inferring Sequence and Acknowledgment Numbers

In this section, we begin with a concise overview of the mechanism used to verify the sequence number and acknowledgment number of TCP segments. Next, we introduce a two-step approach to infer the exact sequence number and an acceptable acknowledgment number.

4.3.1 Verifying TCP Segment

According to RFC 9293 [18], upon receiving a TCP segment, the TCP receiver first performs a verification by comparing the sequence number ($SEG.SEQ$) specified in the TCP header with its receive window. In other words, the condition $RCV.NXT \leq SEG.SEQ \leq RCV.NXT + RCV.WND$ must be met, where $RCV.NXT$ denotes the next expected sequence number for an incoming segment, and $RCV.WND$ indicates the size of the receive window. Furthermore, as per the specification, the ACK flag is consistently set to true, except for the initial SYN packet used for connection establishment. If the ACK bit is disabled, the receiver will discard the segment. Therefore, when hijacking the target TCP connection, the attack must also infer an acceptable acknowledgment number, which cannot be evaded by deactivating the ACK flag.

In practice, TCP operates in full duplex mode, thus allowing the attacker to infer the sequence and acknowledgment numbers in either direction. For instance, the client's $RCV.NXT$ (next expected sequence number) and $SND.NXT$ (next sequence number to be sent) are equivalent to the server's $SND.NXT$ and $RCV.NXT$ [18]. In our attack, our main focus is on inferring the sequence and acknowledgment numbers that are deemed acceptable by the server side.

4.3.2 Inferring the Exact Sequence Number

To infer the exact sequence number on the server side, the attacker impersonates the client (*i.e.,* a victim supplicant in WLAN) and sends forged TCP packets containing data to the server. These packets carry the guessed sequence numbers and a random acknowledgment number. The sequence number space is 2^{32} (*i.e.,* 4G), and the server exhibits four distinct responses corresponding to different sequence numbers, as illustrated in Figure 4. (i) If the guessed sequence number falls within the range of $(RCV.NXT - 2G, RCV.NXT - 1)$,

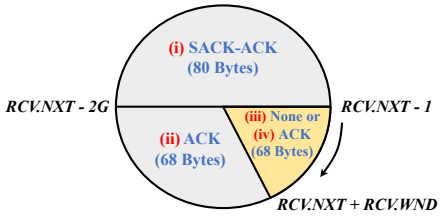


Figure 4: Sequence number window illustration.

the server returns a SACK-ACK response with an encrypted frame size of 80 bytes. (ii) If the guessed sequence number exceeds the upper boundary of the acceptable window (*i.e.,* $RCV.NXT + RCV.WND$), the server sends an ACK response consisting of a 68-byte encrypted frame to the client. (iii) If the guessed sequence number is deemed acceptable but the random acknowledgment number ($SEQ.ACK$) is invalid (*i.e.,* $SEQ.ACK > SND.NXT$), the server silently discards the packet. (iv) If the guessed sequence number is deemed acceptable and the random acknowledgment number falls within the challenge window, the server responds with a challenge ACK in compliance with RFC 5961 [56].

The attacker's goal is to observe the SACK-ACK response, which is contained in an 80-byte encrypted frame. By examining the presence or absence of the SACK-ACK, as illustrated in step 3 of Figure 3(b), the attackers can determine if the guessed sequence number is less than $RCV.NXT - 1$ or greater than $RCV.NXT - 1$ (*i.e.,* identifying the exact sequence number). Employing a binary search strategy, the attacker can progressively refine their guesses and accurately identify the exact sequence number by analyzing the observed SACK-ACK responses.

4.3.3 Inferring an Acceptable Acknowledgment Number

To infer an acceptable acknowledgment number, the attacker firstly leverages the challenge ACK mechanism to locate the lower boundary of the challenge window. Then the attacker can obtain an acceptable acknowledgment number easily, *i.e.,* add 2^{31} to the lower boundary.

The challenge window for TCP segment acknowledgment number is defined in RFC 5961 [56] (see Sec 2.2). As outlined in RFC 5961, the acknowledgment number space can be divided into three distinct cases, as shown in Figure 5. (i) The acknowledgment number falls within the challenge window, defined as $(SND.UNA - 2G, SND.UNA - SND.WND)$. (ii) The acknowledgment number resides within the acceptable ACK window, encompassing $(SND.UNA - SND.WND, SND.NXT)$. (iii) Invalid acknowledgment numbers are those that exceed $SND.NXT$, denoted as $SEG.ACK > SND.NXT$.

In the first case (*i.e.,* falling within the challenge window), the receiver will respond with a challenge ACK to verify the

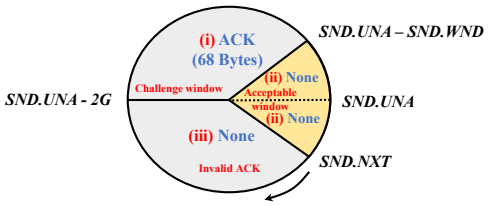


Figure 5: Acknowledgment number window illustration.

legitimacy of the segment. In the second case, the receiver accepts the segment directly for further processing. Otherwise, the receiver will silently discard the TCP segment. For an off-path attacker, the last two cases are indistinguishable. However, the attacker can determine the first case, where a 68-byte encrypted frame is observed.

To locate the server's challenge `ACK` window, the attacker impersonates the client and sends forged `ACK` packets to the server. The forged `ACK` packets carry the guessed acknowledgment numbers, as well as a sequence number in the server's acceptable window inferred in the previous step. If the attacker sniffs a returned 68-byte encrypted frames in the Wi-Fi channel, it indicates that the guessed acknowledgment number falls within the receiver's challenge window (as shown in step 4 of Figure 3(b)). Typically, the window size of challenge `ACK` is between 2^{30} and 2^{31}, *i.e.,* the challenge window is a quarter of the entire acknowledgment number range. Hence, the attacker can divide the acknowledgment range into four blocks and try at most four times to find an acknowledgment number (*ack_challenge*) that is located in the challenge window.

After locating the server's challenge `ACK` window, the attacker can detect the lower boundary of the challenge `ACK` window. In the beginning, the attacker locates the lower boundary of the range ($ack\_challenge - 2G$, $ack\_challenge$). Subsequently, the attacker employs a binary strategy to progressively narrow down the detection range, ultimately determining the lower boundary of the challenge `ACK` window. Once the lower boundary is detected, the attacker can get the server's $SND.UNA$ value by adding $2G$ to the lower boundary. When all previously sent data has been acknowledged, the value of $SND.UNA$ is equal to $SND.NXT$.

5 Case Study Attacks

Similar to the prior works [14, 21], our attack focuses on compromising long-lived TCP connections. In this section, we demonstrate two cases, *i.e.,* SSH DoS and web manipulation, to illustrate how TCP connections can be hijacked by exploiting the encrypted frame size in Wi-Fi networks. In summary, an off-path attacker can reset an SSH service within 270 seconds and inject malicious data into a HTTP web page[4] within 335 seconds.

5.1 TCP DoS Attack

In this case, we demonstrate that an off-path attacker can reset the TCP connection between a victim client and a remote server, resulting in a DoS attack. We specifically conduct the attack under the common scenario of SSH.

Experimental Setup. This case involves three hosts: an SSH server (a rented VPS) running OpenSSH 8.4 and OpenSSL 1.1.1, a victim client (our laptop) running MacOS, and an attacker equipped with Kali 12.04 and multiple wireless network interface cards. The victim client is a supplicant in our Wi-Fi network and connects to the remote SSH server. The client will send commands to the server intermittently. Note that although the attacker and the victim supplicant are in the same Wi-Fi network, the attacker does not know the session key between the victim supplicant and the AP. The attacker attempts to terminate the connection by impersonating the victim supplicant and sending forged `RST` packets to the server. Taking into account the potential impact of packet transmission delays and Linux kernel versions, we strategically position servers with diverse Linux kernel versions in different locations. Detailed configuration information for these servers is provided in Table 2.

Attack Procedure. In this attack, the off-path attacker needs to infer the 4-tuple of [client IP address, client port number, server IP address, server port number] and the exact sequence number of the target TCP connection. The server's IP address and port number are publicly known to the attacker [14, 58], thus it only needs to identify the other three remaining elements to proceed with the attack. The attacker first probe the victim client's IP address and MAC address. Then, the attacker exploits the challenge `ACK` mechanism to determine the client port number of the TCP connection. Third, the attacker infers the exact sequence number by leveraging the TCP header options. Finally, a crafted `RST` packet carrying the inferred value is issued to the server, and the server will be tricked into terminating the current SSH connection with the victim client.

Table 2: Experimental results of SSH connection reset.

Server address	Location	Linux version	Time cost (s)	Bandwidth cost (KB/s)	Success rate
82.x.x.41	BJ, CN	5.4	211.08	2.06	8/10
150.x.x.186	SH, CN	5.15	260.99	2.24	8/10
43.x.x.151	TKY, JP	5.10	283.66	2.14	7/10
43.x.x.84	FRA, DE	4.15	318.01	2.23	9/10
43.x.x.187	SV, USA	3.13	253.17	2.04	9/10

[4] HTTPS can prevent attackers from injecting malicious data. However, reports on HTTPS adoption [64] indicate that there are 17% of websites still based on HTTP as of August 2023.

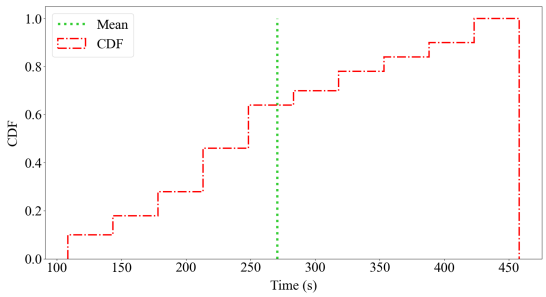


Figure 6: Empirical CDF of time cost of SSH connection reset.

Results Evaluation. Table 2 displays the outcomes of our experiments, revealing that our attack is robust to transmission delays and effective for different Linux versions. In particular, our attack exhibits low bandwidth consumption (*i.e.,* no more than 2.3 KB/s) and maintains an average execution time of 265.9 seconds. The empirical time cost distribution is shown in Figure 6. Our attack achieves a success rate of 82% on average. For the unsuccessful attempts, the primary cause is wireless interference, leading to the attacker missing crucial encrypted frames belonging to the victim. These frames contain the server's responses to the probe packets. We will discuss wireless interference in depth in Sec 7.1.

5.2 TCP Manipulation Attack

TCP connection hijacking poses a substantial threat to higher-layer applications, enabling malicious activities such as injecting harmful data into HTTP websites. In our research, we chose four categories, *i.e.,* government, universities, banks, and financial institutions, which may incur significant impacts, from the Tranco [44] top one million websites based on the keywords of domain names (*e.g.,* gov, edu, bank, and stock, etc). Subsequently, we randomly selected 1000 HTTP sites from these four categories and identified 63 vulnerable websites[5] across 10 countries, encompassing 29 government websites, 17 university websites, 7 banking websites, and 10 financial institutions websites. These websites offer various online services, including government services, customer support, financial data, and more. Attacking them can spread false information and potentially lead to significant financial losses for users. As a case in point, we demonstrate that in a typical financial website scenario, an off-path attacker can hijack the underlying TCP connection, thereby tampering with real-time financial data displayed on the victim's web page.

Experimental Setup. This attack involves three hosts: a web server, a victim client (our laptop), and an off-path attacker. We use a real financial website[6] as the web server. This website employs HTTP to deliver real-time financial data to the client in JSON format at 60-second intervals. Before launching the attack, the attacker can explore this website to familiarize themselves with the JSON data structure. Consequently, the attacker can discern specific data within the packet, as depicted in Figure 8(a), enabling manipulation of the victim's web page. The server maintains a single long-lived TCP connection[7] with the client to transmit real-time financial data. The modern browser may open multiple concurrent TCP connections along with the long TCP connection to speed up the page loading. These concurrent TCP connections are short-lived and have minimal influence on inferring the target long-lived TCP connection. The victim client browses financial information on the web page via Wi-Fi. Both the attacker and the victim client are connected to the same Wi-Fi network. The off-path attacker attempts to detect and hijack the web connection between the victim client and the server.

Attack Procedure. The web connection hijacking attack consists of five steps: (i) The attacker determines the MAC address and IP address of the victim client by leveraging ARP. (ii) By exploiting the encrypted frames, the attacker detects the client's port number to obtain the TCP 4-tuple information. (iii) The attacker infers the exact sequence number and (iv) gets an acceptable acknowledgment number. (v) The attacker impersonates the server and injects forged TCP packets with the inferred values into the victim client. Finally, the client will accept the forged TCP packets, which subsequently update the financial information on the web page.

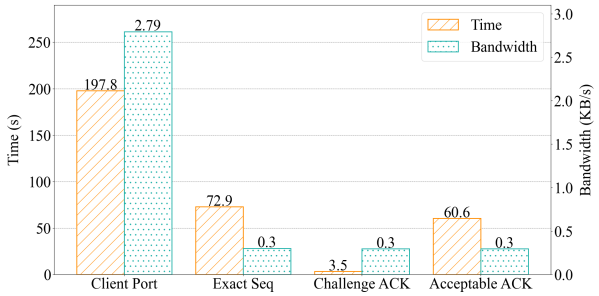


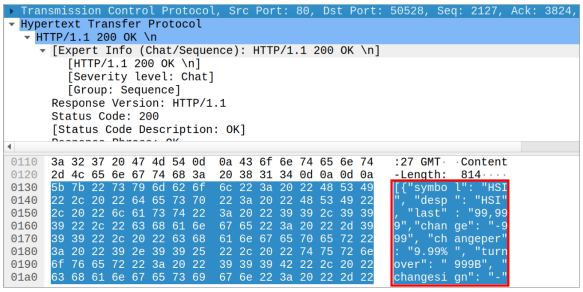
Figure 7: Time/Bandwidth overheads of web manipulation.

Results Evaluation. Figure 7 illustrates the time cost and bandwidth consumption during the attack. It takes an average of 197.8 seconds to identify the client port number and 72.9 seconds to find the exact sequence number. Time cost required to locate the challenge `ACK` window is 3.5 seconds, while

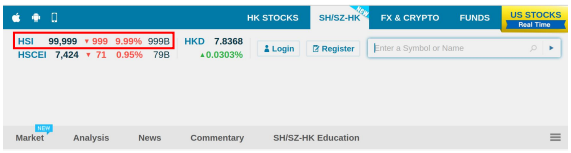
[5]We consider a TCP connection that transmits page data and lasts longer than the empirical average time required for our TCP hijacking attack (335 seconds) to be a valid long-lived TCP connection. A site is considered vulnerable if it maintains no more than two such valid long-lived connections.

[6]For ethical considerations, we anonymize this financial website in the paper. Moreover, our attack do not affect the website, since we only manipulate the web cache of the client side, *i.e.,* our controlled laptop.

[7]As recommended in RFC 2616 [38], the client typically does not maintain multiple long-lived TCP connections with the server simultaneously.



(a) The attacker injects malicious data into the victim's web connection.



(b) The attacker manipulates the HSI data presented on the victim's web page.

Figure 8: Snapshots of web injection.

finding an acceptable acknowledgment number takes 60.6 seconds. Inferring the client port number of the connection consumes approximately 59% of the total time. This is due to the large search space of potential port numbers, which requires the attacker to sequentially probe each possible port. The average duration of the entire attack is 334.8 seconds, with an average bandwidth cost of 1.77 KB/s. In this case, the attacker needs to infer an acceptable acknowledgment number, hence the success rate of this attack is lower than the TCP DoS attack but still exceeds 70%. After obtaining all the necessary information, the attacker sends forged TCP packets to the victim client and manipulates sensitive data on the web page. Figure 8(b) shows a snapshot of the manipulated web page. The attacker alters the HSI data, modifying the values from "19,250 151 0.79% 71B" to "99,999 999 9.99% 999B".

6 Real-world Attacks

To assess the impact of our attack, we conduct an extensive investigation on 80 real-world Wi-Fi networks and 30 popular wireless routers. Precisely, we conduct SSH DoS attack and web hijack attack on the victim (*i.e.,* our device) in the real-world Wi-Fi networks following the experimental setup and procedure in Sec 5. The results reveal that our attack is successful[8] against 69 (86.25%) of the assessed Wi-Fi networks. Besides, we analyze 30 popular wireless routers and find that all the evaluated routers cannot protect the victim from our attack.

[8]We conduct 10 iterations of the SSH DoS attack and the web hijack attack on the victim. In this context, the "successful" means that these two attacks can be successfully executed at least once in the real-world Wi-Fi network.

6.1 Real-world Wi-Fi Networks Evaluation

The Wi-Fi scenarios we tested cover a wide range of public settings, including coffee shops, shopping malls, restaurants, hotels, cinemas, and bookstores. The experimental results illustrate that over 86% (*i.e.,* 69 out of 80) of the evaluated Wi-Fi networks are vulnerable to our attack. Next, we elaborate on the evaluation results.

As shown in Figure 9, out of the 80 Wi-Fi networks we assessed, 74 are found to be IPv4-only networks, while the remaining 6 have IPv6 capabilities[9].

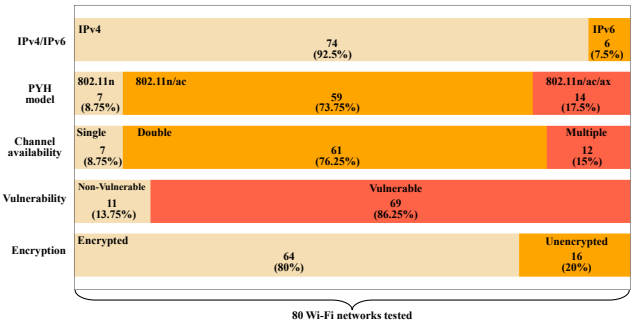


Figure 9: Attack evaluation on 80 real-world Wi-Fi networks.

The 802.11n/ac standards are predominantly utilized (73.75%) in real-world Wi-Fi networks. This indicates that these Wi-Fi networks support two frequency bands (*i.e.,* 2.4 GHz and 5 GHz), with their physical layer models based on the 802.11n and 802.11ac standards, respectively. There is only 17.5% (14 out of 80) of the evaluated Wi-Fi networks support 802.11ax. This is consistent with our expectations, as the 802.11ax standard was defined in 2019 and would require more time for widespread deployment. Furthermore, the simultaneous utilization of two channels in Wi-Fi networks is the most prevalent case (76.25%) due to the widespread support and default configuration of dual-channel capabilities in wireless routers. In certain scenarios, such as office buildings, Wi-Fi networks employ multiple wireless channels to enhance network performance. In our study, we identify 12 Wi-Fi networks that utilize multiple wireless channels. At first glance, the usage of multiple wireless channels might appear as a minor hurdle to our attack, as the attacker needs to perform additional channel scanning to determine the specific channel employed by the victim. Indeed, the attacker can enhance the success rate by employing a channel "eviction" strategy, which will be explained in detail in Section 7.1. Additionally, we encounter several Wi-Fi networks that operate on a single channel. When questioned, network administrators cited security considerations as the rationale behind this

[9]Despite the accelerated deployment of IPv6 networks in recent years [6], the adoption of IPv6 support in Wi-Fi networks remains relatively limited. This can be attributed, in part, to the lack of IPv6 support in legacy wireless routers and the limited incentive for merchants to invest in new wireless routers.

Table 3: Experimental results in 20 real-world Wi-Fi networks.

No.	SSID	AP Vendor	IPv4/IPv6	PHY model	Encryption	Wi-Fi channel	SSH DoS	Web hijack
1	Bookstore 1	ADSLR	◐	802.11n/ac	Yes	6, 161	7/10	6/10
2	Bookstore 2	HUAWEI	◐	802.11n/ac/ax	Yes	11, 44	7/10	7/10
3	Coffee Shop 1	TP-LINK	◐	802.11n/ac	Yes	6, 60	8/10	6/10
4	Coffee Shop 2	HUAWEI	◐	802.11n/ac	Yes	1, 36	8/10	7/10
5	Coffee Shop 3	Tenda	●	802.11n/ac	Yes	4, 153	6/10	5/10
6	Restaurant 1	D-Link	◐	802.11n/ac	Yes	5, 149	7/10	5/10
7	Restaurant 2	TP-LINK	◐	802.11n/ac	Yes	1, 153	6/10	6/10
8	Restaurant 3	iKuai	◐	802.11n/ac	Yes	1, 48	5/10	3/10
9	Office building 1	TP-LINK	◐	802.11n/ac	Yes	11, 36, 40	7/10	6/10
10	Office building 2	H3C	●	802.11n/ac	Yes	1, 48, 153	8/10	7/10
11	Office building 3	ZTE	◐	802.11n/ac	Yes	11, 60	6/10	6/10
12	Experience Store 1	HUAWEI	◐	802.11n/ac	Yes	1, 36	7/10	6/10
13	Experience Store 2	HUAWEI	◐	802.11n/ac	Yes	11, 149	7/10	6/10
14	Experience Store 3	Tenda	◐	802.11n/ac	Yes	4,153	6/10	5/10
15	Fast Food Restaurant 1	HPE	◐	802.11n/ac/ax	Yes	6, 161, 149	6/10	4/10
16	Fast Food Restaurant 2	Ruijie	◐	802.11n/ac	Yes	1,44	6/10	6/10
17	Cinema 1	H3C	◐	802.11n/ac	No	10, 149	7/10	5/10
18	Cinema 2	HUAWEI	◐	802.11n/ac	Yes	1, 157	7/10	6/10
19	Hotel 1	HUAWEI	◐	802.11n/ac	Yes	6, 44	8/10	7/10
20	Hotel 2	Ruijie	◐	802.11n	Yes	1, 11	8/10	6/10

◐ means IPv4 only and ● means both IPv4 and IPv6 are supported.

choice, although they did not provide any further specifics.

Out of all the evaluated networks, 69 (86.25%) of them are vulnerable to our attack. Out of the 69 vulnerable Wi-Fi networks identified, 16 of them are open networks that lack any enabled security mechanisms. This means that an attacker can potentially access and view the contents of all supplicant frames transmitted on these networks. Such a breach of supplicant privacy represents a significant concern.

We elaborate more on the experimental results of 20 Wi-Fi networks in Table 3. We take the first row of Table 3 as an example to analyze the results. In our study, the SSID "Bookstore 1" indicates a Wi-Fi network that is accessible in a bookstore. It is common practice to set the Wi-Fi SSID as the organization name, which may expose the organization's identity. Therefore, to protect anonymity, we have anonymized the Wi-Fi SSID in this paper. This bookstore's Wi-Fi network only supports IPv4 and its AP is produced by ADSLR. This AP provides two access channels (*i.e.,* 6 and 161) and employs the 802.11n and 802.11ac standards. The TCP connection of the victim supplicant can be hijacked using the attack we present in Sec 4. The success rates to conduct SSH DoS and web hijacking on the victim supplicant are 70% and 60%, respectively.

Within the evaluated vulnerable Wi-Fi networks, we have observed a range of success rates for our attack, varying from 30% to 80%. The principal factor influencing this variance is the heterogeneous wireless environments (*e.g.,* different wireless interference and channel contention) encountered in real-world Wi-Fi networks, leading to varying capabilities for attacker to capture the victim's Wi-Fi frames. Factors such as wireless interference (*e.g.,* from microwave ovens and Bluetooth devices) and channel contention can hinder the attacker's ability to capture the victim's Wi-Fi frames, resulting in the failure of attacks. For instance, as illustrated in Table 3, Coffee Shop 1, situated on a university campus, experiences less wireless interference and channel contention compared to Restaurant 3, located within a large shopping mall. Consequently, the success rate of attacks in the latter Wi-Fi network is lower due to the elevated interference and contention in that environment. It is worth noting that out of the 80 Wi-Fi networks assessed, 11 were not impacted by our attack. These Wi-Fi networks enable AP isolation and the router drops ARP requests sent by the attacker, thus thwarting our attack. We will conduct a more in-depth analysis of wireless interference and AP isolation in Section 7.

6.2 Analysis of AP Routers

To protect data transmission in the shared wireless channels, Wi-Fi Alliance has introduced multiple security mechanisms, ranging from WEP to the state-of-the-art WPA3. Although many vendors have released wireless routers that support WPA3, the majority of real-world Wi-Fi networks still utilize the WPA2 security mechanism [35]. In our empirical study, we find that out of the 30 tested wireless routers, 14 support WPA3, while the remaining 16 only support WPA2.

The early WEP used RC4 and MIC algorithms for data encryption, whereas WPA2 replaced them with the AES-CCMP algorithm. In the latest WPA3 standard, AES-GCMP is proposed to be used as the encryption method for WPA3 Enterprise mode. However, none of these security mechanisms can prevent the encrypted frame size from leaking upper layer

Table 4: An examination of routers from 9 AP Vendors.

Router	Generation	WPA	IPv6 Enabled	Vendor	Built-in Firewall	Anti-Flooding	MAC-ADDR Filtering
Mi 4C	Wi-Fi 4	WPA2	No	Xiaomi	●	●	●
TL-WR841N	Wi-Fi 4	WPA2	No	TP-LINK	●	○	●
AX3	Wi-Fi 6	WPA2/WPA3	Yes	HUAWEI	●	●	●
RT-AC66U	Wi-Fi 5	WPA2	Yes	ASUS	●	●	●
AC 6	Wi-Fi 5	WPA2	Yes	Tenda	●	○	○
AX1800	Wi-Fi 6	WPA2/WPA3	Yes	Netgear	●	○	●
E5600	Wi-Fi 5	WPA2	Yes	Linksys	●	●	●
RG-EW1200G PRO	Wi-Fi 5	WPA2	Yes	Ruijie	○	○	●
N21	Wi-Fi 5	WPA2	No	H3C	●	○	●

○ indicates that the security mechanism is not supported by the router, while ● indicates that it is supported.

information. Based on our empirical findings, we confirm that all 30 evaluated wireless routers could not protect the supplicant from our attacks.

Due to space limitations, Table 4 presents a concise overview of routers from the 9 AP vendors included in our investigation. Comprehensive information on all tested routers can be found in Appendix A (see Table 5). Take the first line for example, the evaluated router "Mi 4C" manufactured by Xiaomi belongs to the older Wi-Fi generation (Wi-Fi 4) and lacks support for IPv6 and WPA3. As outlined in the product description, the "Mi 4C" device offers support for various security features. These include a built-in firewall that allows administrators to define packet forwarding rules, flood defense mechanisms that restrict malicious flood traffic to prevent DoS attacks, and MAC address filtering, which enables network access authorization based on hardware addresses. In our investigation, all vulnerable routers claim to support different security mechanisms to prevent various attacks. However, our study demonstrates that the existing security mechanisms are inadequate against our attack.

7 Discussion

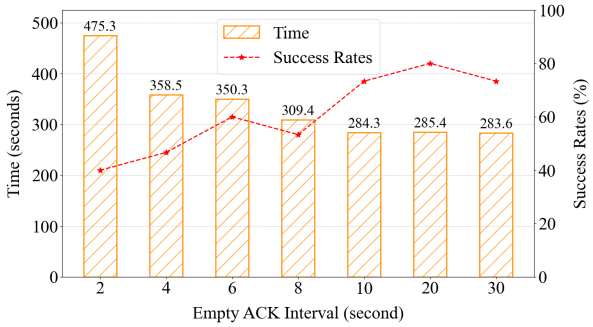
7.1 Success Rate Analysis

Our attack relies on the observation of the victim's encrypted frame size. However, the attacker's ability to monitor the victim's frames may be hindered by wireless interference and Wi-Fi channel contention. These factors can directly affect the success and effectiveness of our attack. Furthermore, background traffic on the victim's device and shifting of the TCP receive window can also influence the attack's success.

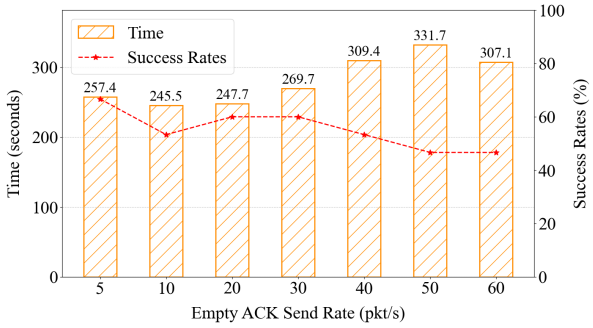
Wireless Interference. The transmission of Wi-Fi frames over the wireless medium is susceptible to losses. These losses are often a result of interference, leading to a diminished signal to interference and noise ratio (SINR) at the receiver [66]. A low SINR decreases the likelihood of successfully decoding all the bits in the frame. Wi-Fi networks face various sources of interference, including microwave ovens, Bluetooth devices, radar signals, and more. Consequently, frame reception failures are frequent occurrences in Wi-Fi networks [33]. Due to wireless interference, the attacker may not be able to sniff all of the victim's encrypted frames. To mitigate wireless interference, we employ a straightforward yet efficient multiple verification strategy. This strategy involves using multiple monitoring wireless network interface cards and performing repeated verifications of the inferred values. By leveraging multiple wireless network interface cards and verifying the inferred values multiple times, we increase the reliability and accuracy of our analysis.

Channel Contention. APs and supplicants based on the 802.11 standards use Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to compete equally for the occupation of the wireless channel. Before transmitting frames, wireless channel listening is conducted to ensure that the channel is not occupied. Frames are transmitted only after verifying the channel's availability. Due to channel contention, there is an uncertain delay in the victim's responses to the probe packets. This uncertain response delay is the main reason for the fluctuating success rate of our attack because the attacker needs to analyze the victim's encrypted frames within a time slice after the probe packets are sent. To eliminate this uncertain response delay, we propose a channel "eviction" strategy. The attacker can evict other supplicants from the channel used by the victim. Specifically, the attacker impersonates the AP and sends decertification frames to the supplicant, causing it to detach from the current channel of the AP. The supplicant will then request to access the Wi-Fi network again, but it will switch to another channel. This strategy requires the Wi-Fi network to support multiple access channels. Fortunately, most Wi-Fi networks provide more than one access channel, as shown in Sec 6.1. Note that the channel switching (*i.e.,* our "eviction" strategy to cause other supplicants detaching from the current channel) is transparent to the users. The only impact is that the user may experience a brief (a few seconds) network jitter during the channel switching.

Background Traffic. The background traffic may degrade the quality of the side channel (*i.e.,* victim's frame size), thereby impacting the effectiveness of the attack. Specifically, other packets received by the victim client have the potential to affect our side channel attack in two ways. (i) The Background traffic may overshadow the target encrypted frames. During our evaluation, the number of responses triggered by the probe packets is typically less than 10% compared to



(a) 40 pkt/s of empty ACK traffic at various time intervals.



(b) Empty ACK traffic with different send rates at 8s time intervals.

Figure 10: Quantitative analysis of empty ACK traffic.

the background traffic. To effectively filter the target frames from the background traffic, the attacker can employ multiple WNICs to monitor the Wi-Fi channel. (ii) The responses triggered by the probe packets may be mixed with other TCP packets. If the TCP packets in the background traffic have the same size (*e.g.,* empty ACK packets) as the response from the victim server, the attacker may mistake them for the actual responses. To quantitatively evaluate the impact of empty ACK packets on our attack, we send empty ACK packets (from other servers) with varying bandwidths (*e.g.,* different time intervals and send rates) to the victim's device. Simultaneously, the attacker infers the 4-tuple, sequence number, and acknowledgment number of the victim's TCP connection by following the attack steps delineated in Section 4. The attacker repeats the attack 15 times under each experimental configuration. Inferring the correct TCP connection information is considered a successful attack. The experimental results are depicted in Figure 10. As depicted in Figure 10(a), when the time interval between empty ACK traffic decreases, the attack requires more time to execute and yields a reduced success rate. This phenomenon occurs because the presence of empty ACK packets confuses the attacker, leading to repeated verification attempts to confirm the inferred value's accuracy. When the send rate of empty ACK packets increases (as shown in Figure 10(b)), the time required for the attack to succeed extends and the success rate diminishes, but not significantly. This phenomenon arises because our attack relies on the presence or absence of frames of specific dimensions, rather than their quantity. Consequently, the attack is more susceptible to the temporal intervals at which empty ACK packets are observed. Fortunately, in most cases (*e.g.,* content servers), the TCP packets sent by the servers contain application data. As a result, TCP packets from background traffic are significantly larger (over 1000 bytes) in size compared to the responses (*e.g.,* challenge ACK) triggered by the probe packets. Additionally, in scenarios with a high volume of empty ACK traffic affecting the victim, the attacker has the option to leverage SACK-ACK to infer the port number and acknowledgment number of the TCP connection, thus avoiding interference from empty ACKs. Specifically, (i) when inferring the port number, the attacker sends two TCP packets containing data to the server, each bearing sequence numbers *seq* and $seq + 2^{31}$ respectively. If the TCP port is accurately inferred, the attacker will encounter an 80-byte encrypted frame, as one of the two packets in question is bound to elicit the server's SACK-ACK response. Conversely, if the inference is incorrect, the attacker will not observe the 80-byte encrypted frame. (ii) Since TCP is full duplex, the attacker can utilize SACK-ACK to infer the sequence number on the client side and consequently obtain the acknowledgment number on the server side.

Shifting Receive Window. An additional factor influencing the attack's success rate is the dynamic adjustment of the receive window. When the victim's TCP connection carries on ongoing traffic, the acceptable sequence and acknowledgment windows will shift during the attack, but it does not impact the attack as long as the inferred sequence number and acknowledgment numbers fall in the sliding windows. The attacker can repeatedly infer the sequence number and acknowledgment number. Even if the receive window slides quickly enough to thwart the attacker's inference, the attacker can opt to target the other end of the TCP connection. In this scenario, the receive window adjusts at a slower pace, as has been demonstrated in prior research [14, 21].

7.2 Special Policies Analysis

In practice, two special network policies (*i.e.,* frame aggregation and AP isolation) may also affect our attack.

Frame Aggregation. The MAC layer frame aggregation technique is proposed in the 802.11n [2] standard to improve the throughput and efficiency of WLANs. There are two methods available to perform frame aggregation, *i.e.,* aggregate MAC protocol service unit (A-MSDU) and aggregate MAC protocol data unit (A-MPDU). The main difference between MSDU and MPDU is that the latter has a MAC header through 802.11 protocol encapsulation while the former becomes MPDU after adding integrity check MIC, encryption, sequence number assignment, CRC checksum, and MAC header. The A-MPDU has no impact on our attack because each MPDU

has a complete MAC header and the attacker can distinguish the encryption payload size of each MPDU. However, if the victim triggers A-MSDU, multiple packets will be encrypted together, preventing the attacker from inferring TCP information based on the encrypted frame size. Fortunately, the server's responses triggered by attackers are rarely aggregated into A-MSDUs. In the following, we analyze the reasons why A-MSDU frames are not triggered.

The A-MSDU completes when the size of the waiting packets reaches the maximum A-MSDU threshold or the maximum delay of the oldest packets reaches a pre-assigned value. Its maximum size can be 3839 or 7935 bytes, depending on the throughput capacity of the station (STA). The size can be found from the High-Throughput (HT) capabilities element of the HT STA release. In case the aggregated frame size does not reach the aggregation threshold, the MSDU buffer queue waits for new MSDUs to reach the MAC layer. But if the maximum delay exceeds the preset maximum, the aggregated frame will be immediately inserted into the channel, even if the aggregated frame size does not reach the aggregation threshold. The maximum delay is typically set to 1 µs [55]. Additionally, only frames with the same receiver and the same TID (traffic identifier) can be aggregated together using A-MSDU. In our attack, few A-MSDU frames are observed. We speculate that this absence may be due to the attacker sending probe packets at a low rate, and the response packets having a different TID than the background traffic (*e.g.,* video packets). Consequently, the TCP responses triggered by the probe packets are not aggregated into A-MSDU.

AP Isolation. When AP isolation is enabled on wireless routers, supplicants are only allowed to access the gateway and the outside network. This means that supplicants in the same Wi-Fi network cannot communicate with each other. Under this circumstance, the probe packets sent by the attacker to the victim supplicant will be discarded at the AP. This particular network policy can throttle our attack. However, AP isolation may impair network availability and impact user experience. For example, the wireless display can no longer be used in Wi-Fi networks when AP isolation is enabled. As a result, we observe that less than 14% (11 out of 80) of the evaluated Wi-Fi networks enforce this policy.

8 Countermeasure

The root cause of our attack can be attributed to the combination of two specific conditions. The first condition is the inconsistent response of the TCP stack under different trigger conditions. The second condition is the leakage of TCP connection information through the frame size side channel. As a result, we propose two countermeasures to mitigate this vulnerability, one derived from the 802.11 standard and the other from the TCP stacks.

Defenses in 802.11 Standard. As Wi-Fi networks rely on shared wireless media, any 802.11-compliant device has the capability to sniff all Wi-Fi frames. To maintain the confidentiality and integrity of Wi-Fi frames, encryption mechanisms are commonly employed in Wi-Fi networks. Although Wi-Fi networks encrypt their frames transmitted in the wireless channel, there exists a strong correlation between the encrypted frame size and the upper layer applications. This correlation allows an off path attacker to analyze the encrypted frame size, infer the victim's TCP information, and subsequently conduct the TCP hijacking attack. Adjusting the security mechanisms of the 802.11 standards so that the AP or supplicant dynamically pads the size of encrypted frames is one possible countermeasure. This countermeasure may require changes and redesign at the Wi-Fi standard level. We are currently in discussions with the Wi-Fi Alliance regarding this countermeasure.

Defenes in TCP Stacks. The packet validation logic in the latest TCP specification handles valid and invalid incoming packets differently depending on whether a response needs to be generated and the type of response required. This difference is reflected in two aspects: (i) The number of response packets is different. For example, during the verification of the acknowledgment number, one challenge `ACK` will be triggered if the packet's acknowledgment number falls within the challenge window. If it falls outside the window, no packet will be sent. (ii) The response packets have different types. The type of TCP packet can be identified by its size, which is influenced by the varying header options. For instance, the size of a `SACK-ACK` packet is 78 bytes, whereas a `RST` packet is only 54 bytes in size. An attacker can infer the state of a TCP connection by observing the size of the response packets, which are encrypted frames in our attack. To resolve this problem, a possible solution is to revise the TCP specification by obfuscating the header sizes for different types of TCP packets (*e.g.,* `RST`, `ACK`, and `SACK-ACK`) and adjusting the trigger conditions for the challenge `ACK`.

9 Related Work

Traffic Analysis. The prior traffic analysis works endeavors aimed to analyze users' encrypted traffic and compromise their privacy by, for instance, tracking the applications [11, 42, 52, 53, 59] and websites [29, 47, 51, 65] they accessed. Ede *et al.* designed a semi-supervised scheme for creating application fingerprints from encrypted network traffic of mobile devices [59]. Shen *et al.* used Graph Neural Networks to identify decentralized applications from encrypted traffic [52]. Hayes *et al.* established that website fingerprinting attacks are a serious threat to online privacy [29]. Rimmer *et al.* harnessed deep learning for web fingerprinting, which de-anonymizes Tor traffic by classifying encrypted web traffic [47]. Furthermore, several academic studies delve into the privacy challenges associated with encrypted DNS [13, 31, 49, 54]. Shulman proposed that encryption alone may not be sufficient to protect users [49], and Siby

et al. demonstrated that classifying encrypted DNS traffic can jeopardize the user privacy [54].

Our attack and prior research on traffic analysis both involve extracting information from encrypted packets. Nonetheless, there are three key distinctions between our attack and traffic analysis work. Firstly, while previous work relies on an on-path attack model, our attack does not require such positioning. Secondly, traffic analysis typically involves the creation of a database as a prerequisite, whereas our attack operates without this necessity. Finally, existing traffic analysis work focuses on upper-layer applications, while our attack interferes with the underlying transport protocol.

Wi-Fi Attacks. While Wi-Fi serves as a widely used access method for end-users to connect to the internet, it presents higher security risks compared to wired LANs, such as Ethernet. Public Wi-Fi networks, in particular, are susceptible to attacks due to their open-access nature. To safeguard wireless users in Wi-Fi networks, numerous security mechanisms have been proposed in recent years, including WEP, WPA, WPA2, and WPA3 [8]. Nevertheless, existing researches [24, 39, 57, 60–63] have revealed implementation vulnerabilities or design flaws in these security mechanisms that can compromise Wi-Fi networks. For example, WPA is vulnerable to key recovery attacks [39, 57] and dictionary attacks [37]. Subsequently, WPA2 and WPA3 were introduced to mitigate these vulnerabilities. However, recent research indicates that WPA2 is susceptible to KRACK attacks [61, 62], and WPA3 can be compromised by downgrade or dictionary attacks [63]. Besides, recent studies [48, 60] have revealed that attackers can leverage the design flaws of Wi-Fi networks to circumvent these security mechanisms. Unlike the aforementioned studies, our attack does not require cracking or circumventing security mechanisms.

In addition to Wi-Fi network cracking, extensive research has been conducted on traffic hijacking within Wi-Fi networks. Attackers can execute an Evil Twins attack by deploying a rogue AP to hijack the traffic of victim supplicants [9, 25, 28, 40]. Additionally, rogue DHCP and ARP poisoning are recognized as common threats in Wi-Fi networks. Notably, these attacks have been subject to extensive research, leading to the development of countermeasures, including rogue AP and rogue DHCP detection [5, 10, 32, 36] and ARP protection [4, 17, 50]. Recently, Feng *et al.* exposed implementation flaws in wireless routers that allow the hijacking of traffic from victim supplicants in Wi-Fi networks using ICMP redirect messages. In contrast to previous studies, our attack is novel and stealthy by revealings a fundamental security vulnerability in the 802.11 protocol that affects all Wi-Fi networks.

Side Channel Attacks. In many cases, off-path attackers rely on a side channel to carry out their attacks, where blind attackers can extract significant information from this channel [19, 20, 27, 41, 45, 67–69]. In one instance, Ensafi *et al.* utilized the side channel of global IPID [20] counters to perform idle port scans and network protocol analyses. They also proposed that these counters could be leveraged to detect intentional packet drops. In another example, Alexander *et al.* inferred the round-trip time (RTT) between two arbitrary hosts by examining the shared `SYN` backlog [67].

In TCP connection hijacking attacks, side channels serve as potent tools for attackers. The IPID, in particular, has been a frequent target for exploitation. For example, Jeffrey *et al.* utilized per-destination IPID counters to estimate the number of packets transmitted between two machines and even detect the presence of a TCP connection [34]. Similarly, Alexander *et al.* used the IPID of triggered RST packets to identify the existence of a victim TCP connection [7]. In a recent instance, Feng *et al.* manipulated the IPID assignment using ICMP to hijack the victim's TCP connection [21]. Besides IPID, the challenge `ACK` mechanism is another side channel exploited by off-path attackers. Cao *et al.*, for example, utilized the global rate limit of challenge `ACK` to infer and hijack TCP connections [14, 15]. Moreover, a timing side channel has been found in half-duplex Wi-Fi technology, which can be exploited by off-path attackers to inject data into a TCP connection [16]. Tolley *et al.* recently proposed a blind in/on-path attack in VPNs, aiming to infer the existence of, interfere with, or inject data into TCP connections forwarded through encrypted VPN tunnels [58]. Different from previous research, our work reveals a new side channel, *i.e.,* information leakage due to the encrypted frame size in Wi-Fi networks. This side channel can be exploited by a pure off-path attacker to hijack victim's TCP connections.

10 Conclusion

In this paper, we present a new off-path TCP hijacking attack that takes advantage of encrypted frame size in Wi-Fi networks to detect and hijack TCP connections belonging to a victim supplicant. This side channel vulnerability is an inherent flaw in Wi-Fi networks, specifically the 802.11 standards. To execute our attack, attackers initially scan the WLAN to identify active victims, then monitor the victim's encrypted frame size to deduce the 4-tuple, exact sequence number, and acceptable acknowledgment number of the victim's TCP connection. We carry out our attack in typical Wi-Fi scenarios, and our evaluation demonstrates that this new off-path TCP hijacking attack can result in significant damage to upper-layer applications, such as SSH DoS and the injection of malicious data into web traffic. Moreover, we conduct comprehensive studies involving 80 real-world Wi-Fi networks and 30 popular wireless routers. The results reveal that a majority of Wi-Fi networks (69 out of 80) and all tested routers are vulnerable to our attack. We have responsibly disclosed this vulnerability. While eliminating the side channel of encrypted frame size in Wi-Fi networks presents challenges, we propose several potential countermeasures to mitigate this vulnerability.

References

[1] 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 6: Medium access control (mac) security enhancements. *IEEE Std 802.11i-2004*, pages 1–190, 2004.

[2] IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac)and physical layer (phy) specifications amendment 5: Enhancements for higher throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages 1–565, 2009.

[3] 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. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2021.

[4] 360-ARP. 360 total security: Free antivirus protection for home and devices. http://www.360totalsecurity.com/en/, Accessed November 2023.

[5] Mayank Agarwal, Santosh Biswas, and Sukumar Nandi. Discrete event system framework for fault diagnosis with measurement inconsistency: Case study of rogue dhcp attack. *IEEE/CAA Journal of Automatica Sinica*, 6(3):789–806, 2017.

[6] Akamai. Ipv6 adoption visualization, 2023. https://www.akamai.com/internet-station/cyber-attacks/state-of-the-internet-report/ipv6-adoption-visualization.

[7] Geoffrey Alexander, Antonio M. Espinoza, and Jedidiah R. Crandall. Detecting TCP/IP connections via IPID hash collisions. *Proc. Priv. Enhancing Technol.*, 2019.

[8] Alliance. Discover wi-fi security, 2022. https://www.wi-fi.org/discover-wi-fi/security.

[9] Ali M Alsahlany, Alhassan R Almusawy, and Zainalabdin H Alfatlawy. Risk analysis of a fake access point attack against wi-fi network. *International Journal of Scientific & Engineering Research*, 2018.

[10] Chip Andrews. Dhcp sentry detection. https://www.sqlsecurity.com/downloads/dhcp-sentry, Accessed November 2023.

[11] Alireza Bahramali, Amir Houmansadr, Ramin Soltani, Dennis Goeckel, and Don Towsley. Practical traffic analysis attacks on secure messaging applications.

[12] David A. Borman, Robert T. Braden, and Van Jacobson. TCP Extensions for High Performance. RFC 1323, May 1992. https://www.rfc-editor.org/info/rfc1323.

[13] Jonas Bushart and Christian Rossow. Padding ain't enough: Assessing the privacy guarantees of encrypted DNS. In *FOCI 2020, August 11, 2020*.

[14] Yue Cao, Zhiyun Qian, Zhongjie Wang, Tuan Dao, Srikanth V. Krishnamurthy, and Lisa M. Marvel. Off-path TCP exploits: Global rate limit considered dangerous. In *USENIX Security 16, Austin, TX, USA, August 10-12, 2016*.

[15] Yue Cao, Zhiyun Qian, Zhongjie Wang, Tuan Dao, Srikanth V. Krishnamurthy, and Lisa M. Marvel. Off-path TCP exploits of the challenge ACK global rate limit. *IEEE/ACM Trans. Netw.*, 2018.

[16] Weiteng Chen and Zhiyun Qian. Off-path TCP exploit: How wireless routers can jeopardize your secrets. In *USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*.

[17] Alexandru Chirila. Arp antispoofer. https://www.softpedia.com/get/Security/Firewall/ARP-AntiSpoofer.shtml, Accessed November 2023.

[18] Wesley Eddy. Transmission Control Protocol (TCP). RFC 9293, August 2022. https://www.rfc-editor.org/info/rfc9293.

[19] Roya Ensafi, Jeffrey Knockel, Geoffrey Alexander, and Jedidiah R. Crandall. Detecting intentional packet drops on the internet via TCP/IP side channels. In *PAM 2014, Los Angeles, CA, USA, March 10-11, 2014, Proceedings*.

[20] Roya Ensafi, Jong Chun Park, Deepak Kapur, and Jedidiah R. Crandall. Idle port scanning and non-interference analysis of network protocol stacks using model checking. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*.

[21] Xuewei Feng, Chuanpu Fu, Qi Li, Kun Sun, and Ke Xu. Off-path TCP exploits of the mixed IPID assignment. In *CCS '20, Virtual Event, USA, November 9-13, 2020*.

[22] Xuewei Feng, Qi Li, Kun Sun, Yuxiang Yang, and Ke Xu. Man-in-the-middle attacks without rogue AP: when

wpas meet ICMP redirects. In *SP 2023, San Francisco, CA, USA, May 21-25, 2023*.

[23] Sally Floyd, Jamshid Mahdavi, Matt Mathis, and Dr. Allyn Romanow. TCP Selective Acknowledgment Options. RFC 2018, October 1996. https://www.rfc-editor.org/info/rfc2018.

[24] Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In *SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*.

[25] Di Gao, Hao Lin, Zhenhua Li, Feng Qian, Qi Alfred Chen, Zhiyun Qian, Wei Liu, Liangyi Gong, and Yunhao Liu. A nationwide census on wifi security threats: prevalence, riskiness, and the economics. In *ACM MobiCom '21, New Orleans, Louisiana, USA, October 25-29, 2021*.

[26] John Giffin, Rachel Greenstadt, Peter Litwack, and Richard Tibbetts. Covert messaging through TCP timestamps. In *PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*.

[27] Yossi Gilad and Amir Herzberg. Spying in the dark: TCP and tor traffic analysis. In *PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings*.

[28] Hao Han, Bo Sheng, Chiu C. Tan, Qun Li, and Sanglu Lu. A timing-based scheme for rogue AP detection. *IEEE Trans. Parallel Distributed Syst.*, 2011.

[29] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *USENIX Security 16, Austin, TX, USA, August 10-12, 2016*.

[30] Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Is it still possible to extend tcp? In *IMC '11, Berlin, Germany, November 2-, 2011*.

[31] Rebekah Houser, Zhou Li, Chase Cotton, and Haining Wang. An investigation on information leakage of DNS over TLS. In *CoNEXT 2019, Orlando, FL, USA, December 09-12, 2019*.

[32] Huawei. Rogue device detection. https://support.huawei.com/enterprise/en/doc/EDOC1100096321/3eb0a62e/example-for-configuring-rogue-device-detection-and-containment, Accessed November 2023.

[33] Muhammad Owais Khan, Lili Qiu, Apurv Bhartia, and Kate Ching-Ju Lin. Smart retransmission and rate adaptation in wifi. In *ICNP 2015, San Francisco, CA, USA, November 10-13, 2015*.

[34] Jeffrey Knockel and Jedidiah R. Crandall. Counting packets sent between arbitrary internet hosts. In *FOCI '14, San Diego, CA, USA, August 18, 2014*.

[35] Saku Lindroos, Antti Hakkala, and Seppo Virtanen. The COVID-19 pandemic and remote working did not improve WLAN security.

[36] Linksys. How to enable rogue ap detection on your linksys wireless-ac access point. https://www.linksys.com/support-article?articleNum=135793, Accessed November 2023.

[37] Robert Moskowitz. Weakness in passphrase choice in wpa interface. *http://wifinetnews.com/archives/2003/11/weakness_in_passphrase_choice_in_wpa_interface.html*, 2003.

[38] Henrik Nielsen, Jeffrey Mogul, Larry M Masinter, Roy T. Fielding, Jim Gettys, Paul J. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999.

[39] Toshihiro Ohigashi and Masakatu Morii. A practical message falsification attack on wpa. *Proc. JWIS*, 2009.

[40] Ryan Orsi. Understanding evil twin ap attacks and how to prevent them, 2019.

[41] Paul Pearce, Roya Ensafi, Frank Li, Nick Feamster, and Vern Paxson. Augur: Internet-wide detection of connectivity disruptions. In *SP 2017, San Jose, CA, USA, May 22-26, 2017*.

[42] Emanuele Petagna, Giuseppe Laurenza, Claudio Ciccotelli, and Leonardo Querzoni. Peel the onion: Recognition of android apps behind the tor network. In *ISPEC 2019, Kuala Lumpur, Malaysia, November 26-28, 2019, Proceedings*.

[43] Benoit Pit-Claudel, Yoann Desmouceaux, Pierre Pfister, Mark Townsley, and Thomas H. Clausen. Stateless load-aware load balancing in P4. In *ICNP 2018, Cambridge, UK, September 25-27, 2018*.

[44] Victor Le Pochat, Tom van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *NDSS 2019, San Diego, California, USA, February 24-27, 2019*.

[45] Zhiyun Qian, Zhuoqing Morley Mao, Yinglian Xie, and Fang Yu. Investigation of triangular spamming: A stealthy and efficient spamming technique. In *SP 2010, 16-19 May 2010, Berleley/Oakland, California, USA*.

[46] Anantha Ramaiah, Randall R. Stewart, and Mitesh Dalal. Improving tcp's robustness to blind in-window attacks. Technical report, 2010.

[47] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning.

[48] Domien Schepers, Aanjhan Ranganathan, and Mathy Vanhoef. Framing frames: Bypassing wi-fi encryption by manipulating transmit queues.

[49] Haya Schulmann. Pretty bad privacy: Pitfalls of DNS encryption. In *WPES 2014, Scottsdale, AZ, USA, November 3, 2014*.

[50] shARP. https://github.com/europa502/shARP, Accessed November 2023.

[51] Meng Shen, Yiting Liu, Liehuang Zhu, Xiaojiang Du, and Jiankun Hu. Fine-grained webpage fingerprinting using only packet length information of encrypted traffic. *IEEE Trans. Inf. Forensics Secur.*, 2021.

[52] Meng Shen, Jinpeng Zhang, Liehuang Zhu, Ke Xu, and Xiaojiang Du. Accurate decentralized application identification via encrypted traffic analysis using graph neural networks. *IEEE Trans. Inf. Forensics Secur.*, 2021.

[53] Meng Shen, Jinpeng Zhang, Liehuang Zhu, Ke Xu, Xiaojiang Du, and Yiting Liu. Encrypted traffic classification of decentralized applications on ethereum using feature fusion. In *IWQoS 2019, Phoenix, AZ, USA, June 24-25, 2019*.

[54] Sandra Siby, Marc Juarez, Claudia Díaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. Encrypted DNS -> privacy? A traffic analysis perspective.

[55] Dionysios Skordoulis, Qiang Ni, Hsiao-Hwa Chen, Adrian P Stephens, Changwen Liu, and Abbas Jamalipour. IEEE 802.11 n mac frame aggregation mechanisms for next-generation high-throughput wlans. *IEEE Wireless Communications*, 15(1):40–47, 2008.

[56] Randall R. Stewart, Mitesh Dalal, and Anantha Ramaiah. Improving TCP's Robustness to Blind In-Window Attacks. RFC 5961, August 2010. https://www.rfc-editor.org/info/rfc5961.

[57] Erik Tews and Martin Beck. Practical attacks against WEP and WPA. In *WISEC 2009, Zurich, Switzerland, March 16-19, 2009*.

[58] William J. Tolley, Beau Kujath, Mohammad Taha Khan, Narseo Vallina-Rodriguez, and Jedidiah R. Crandall. Blind in/on-path attacks and applications to vpns. In *USENIX Security 2021, August 11-13, 2021*.

[59] Thijs van Ede, Riccardo Bortolameotti, Andrea Continella, Jingjing Ren, Daniel J. Dubois, Martina Lindorfer, David R. Choffnes, Maarten van Steen, and Andreas Peter. Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In *NDSS 2020, San Diego, California, USA, February 23-26, 2020*.

[60] Mathy Vanhoef. Fragment and forge: Breaking wi-fi through frame aggregation and fragmentation. In *USENIX Security 2021, August 11-13, 2021*.

[61] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*.

[62] Mathy Vanhoef and Frank Piessens. Release the kraken: New kracks in the 802.11 standard. In *CCS 2018, Toronto, ON, Canada, October 15-19, 2018*.

[63] Mathy Vanhoef and Eyal Ronen. Dragonblood: Analyzing the dragonfly handshake of WPA3 and eap-pwd. In *SP 2020, San Francisco, CA, USA, May 18-21, 2020*.

[64] W3Techs. Usage statistics of default protocol https for websites, 2023. https://w3techs.com/technologies/details/ce-httpsdefault.

[65] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*.

[66] Jiansong Zhang, Haichen Shen, Kun Tan, Ranveer Chandra, Yongguang Zhang, and Qian Zhang. Frame retransmissions considered harmful: improving spectrum efficiency using micro-acks. In *Mobicom'12, Istanbul, Turkey, August 22-26, 2012*.

[67] Xu Zhang, Jeffrey Knockel, and Jedidiah R. Crandall. High fidelity off-path round-trip time measurement via TCP/IP side channels with duplicate syns. In *GLOBECOM 2016, Washington, DC, USA, December 4-8, 2016*.

[68] Xu Zhang, Jeffrey Knockel, and Jedidiah R. Crandall. High fidelity off-path round-trip time measurement via TCP/IP side channels with duplicate syns. In *GLOBECOM 2016, Washington, DC, USA, December 4-8, 2016*.

[69] Xu Zhang, Jeffrey Knockel, and Jedidiah R. Crandall. Original SYN: finding machines hidden behind firewalls. In *INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*.

A Full List of Tested Routers

The detailed information of the 30 tested routers is shown in Table 5.

Table 5: All tested routers.

Router	Generation	WPA	IPv6 Enabled	Vendor	Built-in Firewall	Anti-Flooding	MAC-ADDR Filtering
Mi 4C	Wi-Fi 4	WPA2	No	Xiaomi	●	●	●
Redmi AC2100	Wi-Fi 5	WPA2	Yes	Xiaomi	●	●	●
AX6000	Wi-Fi 6	WPA2/WPA3	Yes	Xiaomi	●	●	●
AX9000	Wi-Fi 6	WPA2/WPA3	Yes	Xiaomi	●	●	●
TL-WR841N	Wi-Fi 4	WPA2	No	TP-LINK	●	○	●
Archer AXE300	Wi-Fi 6	WPA2/WAP3	Yes	TP-LINK	●	●	●
Archer C80	Wi-Fi 5	WPA2/WPA3	Yes	TP-LINK	●	○	●
Archer AX10	Wi-Fi 6	WPA2/WPA3	Yes	TP-LINK	●	●	●
AX3	Wi-Fi 6	WPA2/WPA3	Yes	HUAWEI	●	●	●
WS7200	Wi-Fi 6	WPA2	Yes	HUAWEI	●	●	●
WS7100	Wi-Fi 6	WPA2	Yes	HUAWEI	●	●	●
WS318N	Wi-Fi 4	WPA2	Yes	HUAWEI	●	○	○
RT-AC66U	Wi-Fi 5	WPA2	Yes	ASUS	●	●	●
RT-AC68U	Wi-Fi 5	WPA2	Yes	ASUS	●	●	●
RT-AX86U	Wi-Fi 6	WPA2/WPA3	Yes	ASUS	●	●	●
RT-AX82U	Wi-Fi 6	WPA2/WPA3	Yes	ASUS	●	●	●
AC 6	Wi-Fi 5	WPA2	Yes	Tenda	●	○	○
AC 8	Wi-Fi 5	WPA2	Yes	Tenda	●	○	●
AC 23	Wi-Fi 5	WPA2	Yes	Tenda	●	●	●
F9	Wi-Fi 4	WPA2	No	Tenda	○	○	●
AX1800	Wi-Fi 6	WPA2/WPA3	Yes	Netgear	●	○	●
AX5400	Wi-Fi 6	WPA2/WPA3	Yes	Netgear	●	○	●
E5600	Wi-Fi 5	WPA2	Yes	Linksys	●	●	●
E7350	Wi-Fi 6	WPA2/WPA3	Yes	Linksys	●	●	●
E8450	Wi-Fi 6	WPA2/WPA3	Yes	Linksys	●	○	●
RG-EW1200G PRO	Wi-Fi 5	WPA2	Yes	Ruijie	○	○	●
M32	Wi-Fi 6	WPA2	Yes	Ruijie	○	○	●
N21	Wi-Fi 5	WPA2	No	H3C	●	○	●
NX15	Wi-Fi 6	WPA2/WPA3	Yes	H3C	●	○	●
B6	Wi-Fi 6	WPA2/WPA3	Yes	H3C	●	●	●

○ indicates that the security mechanism is not supported by the router, while ● indicates that it is supported.