IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 24, 2020, accepted January 17, 2021, date of publication January 25, 2021, date of current version January 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054130

Malicious Data Frame Injection Attack Without Seizing Association in IEEE 802.11 Wireless LANs

WOOCHEOL KIM[1], SOYEON KIM[2], AND HYUK LIM[1,3], (Member, IEEE)

[1]School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Republic of Korea
[2]Second Research and Development Institute, Agency for Defense Development (ADD), Daejeon 34186, Republic of Korea
[3]AI Graduate School, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Republic of Korea

Corresponding author: Hyuk Lim (hlim@gist.ac.kr)

This work was supported in part by the Electronic Warfare Research Center, Gwangju Institute of Science and Technology (GIST), in part by the Defense Acquisition Program Administration (DAPA), and in part by the Agency for Defense Development (ADD), Republic of Korea.

ABSTRACT In the infrastructure mode of IEEE 802.11 wireless local area networks (LANs), an access point (AP) provides wireless networking to multiple wireless client nodes in the shared medium. Client nodes that have established an association with an AP fully trust the AP, and exchange data packets with other nodes on the Internet through the associated AP. If the AP has malicious intent or an attacker deprives the AP, the associated nodes are exposed to the potential danger of security attacks such as a denial of service, data traffic sniffing and spoofing, and malware attacks. These association extortion based attacks can be detected by monitoring frames at the medium access control (MAC) layer. In this article, we propose a malicious-frame-injection based attack without seizing the association between an AP and client nodes. The proposed attack performs wireless jamming, MAC frame sniffing, and spoofing successively in the shared wireless medium. We have implemented the proposed attack using software-defined radio (SDR) on a real-world experimental testbed with off-the-shelf nodes and performed the attack on hypertext transfer protocol (HTTP) communication using transmission control protocol (TCP) transport protocol to demonstrate its wireless LAN security risk.

INDEX TERMS Wireless jamming, malicious frame injection, man-in-the-middle attack, IEEE 802.11.

I. INTRODUCTION

Wi-Fi is one of the most popular wireless technologies massively deployed in public places to offer high-speed networking services at a low cost. In the IEEE 802.11 infrastructure mode, wireless clients need to be associated with an access point (AP) for wireless connectivity. To be associated with an AP, clients have to exchange a series of management frames with the AP. This procedure is performed in the medium access control (MAC) layer in three steps; exchanging probe request/response frames, exchanging authentication request/response frames, and exchanging association request/response frames. Once the association between an AP and client is established, data frame exchanges are performed using a distributed coordinated function (DCF) protocol, which is a fundamental MAC technique of the IEEE 802.11 for reliable communication. The DCF is based on carrier sense multiple access/collision avoidance (CSMA/CA) with binary exponential backoff algorithm, in which a transmitter may transmit a data frame only when the wireless channel is idle [1]. Once the channel becomes idle for DCF inter-frame space (DIFS) duration, a transmitter node generates a random backoff interval for an additional delayed time before transmitting. After the backoff time, the transmitter node is allowed to transmit a data frame to a receiver. If the receiver node receives the data frame without error, it transmits an acknowledgment (ACK) frame to the transmitter node in a short inter-frame space (SIFS) interval. If the transmitter receives the ACK frame from the receiver within the SIFS, it considers the data frame transmission is successful. In addition to the above data/ACK exchange of the DCF, virtual carrier sensing with request-to-send (RTS)/clear-to-send (CTS) mechanism can be also used for reserving the wireless channel for mitigating a frame collision problem in a dense wireless network, as shown in Fig. 1. If the AP receives the RTS frame from the client, the AP transmits the CTS frame

The associate editor coordinating the review of this manuscript and approving it for publication was Ghufran Ahmed.

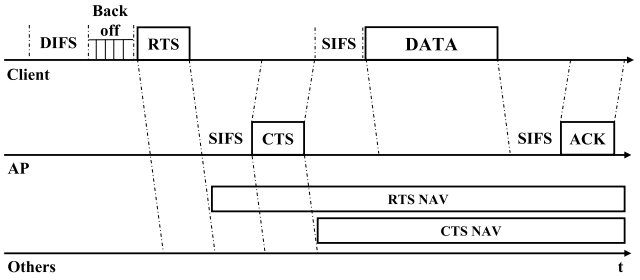


FIGURE 1. IEEE 802.11 dcf with RTS/CTS handshaking.

to the client after the SIFS interval. If other nodes receive the RTS/CTS frames, they read the duration field on the RTS/CTS frames and update the network allocation vector (NAV). Other nodes consider that the medium is busy and defer their transmissions during the NAV duration from first SIFS to the ending of ACK.

In wireless networks where the wireless frames are transmitted over a shared channel, attackers can easily perform wireless attacks by exploiting the broadcast nature of wireless communication, e.g., jamming, sniffing, and spoofing.

- **Jamming:** Jamming disrupts communication by exploiting the fact that communication is affected by signals transmitted from neighbor nodes in a wireless environment. The most naive jamming method is continuous jamming, which generates an interfering signal constantly to disrupt the channel.
- **Sniffing:** Sniffing implies that a node that does not participate in communication eavesdrops on frames between victim nodes. By sniffing the broadcast frames in an IEEE 802.11 environment, an attacker can collect information about the victim nodes.
- **Spoofing:** Spoofing confuses communication by generating the same frames that normal users would generate. For example, in IEEE 802.11 LANs, an attacker can generate fake control frames, data frames, and management frames, as if victim nodes have sent.

Because an attacker shares the wireless medium with clients, it is easy to perform jamming for service disruption, sniff frames from victim nodes, and generate fake frames for spoofing. When clients are associated with an AP and use an Internet application through IEEE 802.11 wireless networking, an attacker may attempt to perform more advanced attacks to inject malicious data contents into the clients or AP during their communication. Previously studied malicious data injection attacks first attempt to steal the association between the victim node and AP, using evil twin attack, address resolution protocol (ARP) spoofing, and so on [2]. Then, the attacker becomes a malicious AP after the extortion of association attack. Eventually, the victim nodes are exposed to inject malicious frames because the attacker can freely attempt jamming, sniffing and spoofing attacks as an intermediary of communication in wireless LANs. However, these attacks based on the association extortion technique can be detected and prevented by a lot of algorithms that have been studied extensively [3]–[7].

Unlike the above attacks, the Airpwn attacker attempts to inject spoofed malicious data frames to a victim client without seizing association, when the victim clients use hypertext transfer protocol (HTTP) application through TCP connection in IEEE 802.11 wireless network [8]. Here, when the victim client transmits a request packet to a server through an AP, the Airpwn attacker sniffs this request packet on the MAC layer, and it transmits a malicious response packet to the victim client faster than the server in the local network. The malicious response packet appears to have come from the server through the AP. Because the malicious data frame is received before the legitimate response packet from the server arrives at the victim client, the legitimate packet is ignored by the client's application. However, if the client finds that a duplicated response packet arrives via packet monitoring, the victim can suspect and detect a data frame spoofing attack because this attacker cannot prevent the legitimate response packet from being sent by the server.

We propose a malicious-frame-injection based attack using software-defined radio (SDR) technology without directly seizing the association between an AP and victim nodes. Instead of stealing the associations, the malicious frame injection attack is achieved by successively performing that data frame sniffing, data frame jamming, and fake data frame spoofing on the victim network. We construct an experimental environment using off-the-shelf devices and SDRs, and demonstrate the attacking scenarios using HTTP applications in IEEE 802.11 wireless LANs. The experimental results indicate that the proposed attack can successfully inject a malicious image file into an HTTP client node and that the client and server cannot detect the attack at the link, transport, and application layers. Our primary contributions are summarized as follows.

- We propose a new method for malicious data frame injection that does not require to seize the association between a client and an AP on an IEEE 802.11 environment. Unlike existing data frame injection attacks, the proposed attack does not pretend to be either of a victim client or AP to inject a malicious data frame. Because the attacker does not have any MAC/IP addresses that are related to the victim network, its existence is hardly detectable at the link/network layer level inspection.
- The proposed attack can perform malware injection at the application-layer by exploiting IEEE 802.11 frames at the link-layer level. Specifically, the proposed attack replaces an IEEE 802.11 wireless frame sent by a TCP client node with a malicious frame encapsulating a spoofed payload that can trigger a malware download on the air. It deceives the transmitter on the TCP client into believing that the data frame transmission was successful using fake ACK spoofing and makes the TCP client keep waiting for a TCP response packet from the TCP server. Eventually, the TCP client is infected by a

compromised packet that the client does not intend to receive.

- The proposed attack exploits SDR technology to generate and transmit jamming frames and fake ACK frames at the MAC layer. Unlike off-the-self nodes using the IEEE 802.11 DCF protocol, the SDR enable us to react to the victims' frames in real-time without the CSMA/CA backoff mechanism because the SDR can override the normal IEEE 802.11 physical/MAC layer operations. It directly generates an IEEE 802.11 radio-frequency waveform at the physical layer level and transmits the waveform on the wireless channel.
- We have implemented the proposed attack in a real-world experimental environment. We demonstrate how the proposed attack can be applied to an HTTP client and server scenario based on TCP communication. We show that both client and server are not aware of the existence of the attacker, and the client that requests a normal image unintendedly downloads a malicious image object.

The remainder of this article is organized as follows. Section II provides an overview of wireless attacks, and the security rules in IEEE 802.11, and Section III describes a wireless attack scenario with TCP connection in IEEE 802.11. Section IV discusses the flow of the proposed malicious frame injection attack, and the analysis of attack success probability. Section V demonstrates the performance of the proposed attack in a real-world environment. Section VI instantiates countermeasures of the proposed attack in few network layers. Section VII concludes the paper.

II. RELATED WORK AND BACKGROUND

A. JAMMING ATTACKS IN IEEE 802.11

Continuous jamming generates a continuous signal to interfere with victim's communication. It consumes a significant amount of energy and also can be detected easily because of its continuous signal. How to reduce energy consumption and mitigate the detection of jamming attack is one of the most important issues, especially, in the military domain. To the end, there has been a lot of research on intelligent attacks for cyber electronic warfare [9]–[11]. If a protocol type of the target network is known in advance, intelligent jamming attacks exploit the vulnerabilities of the network protocol [12], [13]. For example, an IEEE 802.11 WLAN attacker can generate a jamming signal within SIFS interval after receiving a victim node's data frame in order to selectively attack ACK frames. Because of the jamming signal, the ACK frame is disrupted at the data frame transmitter. Similarly, when RTS/CTS mechanism is enabled in IEEE 802.11 WLANs, CTS or data frames can be disrupted selectively after receiving an RTS frame to improve energy efficiency and reduce the probability of detection.

During jamming attacks, victim nodes would experience many data frame transmission failures and retransmissions in the MAC layer. If a series of frame retransmissions happen frequently on the MAC layer, victim nodes may suspect the presence of jamming attack. To avoid such a suspicion on the data frame transmitter a covert jamming attack has been proposed in [14]. The authors proposed a link layer attack mechanism to make the data frame transmitter not detect the transmission failure in the MAC layer. In [14], the covert jammer transmits a fake ACK frame after a data frame jamming. The covert jammer deceives that the transmitter succeeds in delivering the data frame, because the covert jammer transmits the fake ACK frame to the transmitter instead of its intended receiver that is supposed to receive but has failed to receive the data frame. Recently, Yin *et al.* showed that a fake ACK frame attack can reduce victim nodes' throughput to nearly zero in [15].

B. MAN-IN-THE-MIDDLE ATTACK IN IEEE 802.11

Man-in-the-middle (MITM) attack invades a communication channel between a transmitter and a receiver as an intermediary of the communication. An MITM attacker eavesdrops and manipulates the information delivered from the transmitter, and the attacker delivers it to the receiver as if it were sent by the transmitter. When there is an MITM attack, the transmitter and receiver believe that they communicate with each other directly without an intermediary, but actually they are connected via an attacker. IEEE 802.11 WLANs are easily exposed to MITM attacks, especially in the infrastructure mode where the wireless networking service is provided by an AP. In most cases, MITM attacker attempts to seize the association between a client and an AP. Once it succeeds in seizing the association, it pretends to be the AP that the client was associated with. The rogue AP attack is to install an unauthorized AP in a network with a malicious intent, and it tries to associate with normal clients. The evil twin attack is to install a malicious AP that has the same service set identifier (SSID) and MAC address with a target AP in the shared medium. The evil twin AP has a stronger signal than the victim AP. The evil twin attacker transmits de-authentication management frames to a client to make the existing association between the client and the victim AP be disconnected After this disconnection, the client will be associated with the evil twin AP because the AP has the same SSID, MAC address, and even a stronger signal than the victim AP.

Another naive MITM attack is ARP spoofing attack. In the shared channel, nodes bind the IP and MAC addresses of each node by exchanging ARP packets, and they construct an ARP table. By spoofing an ARP packet, attackers attempt to modulate the ARP tables of clients and APs [16]. To seize an association between a client and an AP, the APR spoofing attacker transmits a spoofed ARP packet to the client. Then, the IP address to MAC address mapping for the associated AP is changed to and MAC address of the attacker in the ARP map of the client. Similarly, if the ARP spoofing attacker transmits a spoofed ARP packet to the associated AP, the mapping for the IP address of the client is changed to MAC address of the attacker. By seizing associations between

clients and APs with the ARP spoofing attacks, attackers are able to insert malicious packets or interfere with the original communication; however, many techniques have been proposed to prevent or detect to such steal associations [3]–[7].

C. WIRELESS ATTACKS USING SDR IN IEEE 802.11

The SDR is a software communication system that supersedes tasks mainly performed by hardware such as filters, amplifiers, and modulators in wireless communications [17], [18]. SDRs are widely used to implement client/AP nodes that communicate through physical and MAC layers by supporting a real-time reaction. These days, there has been lots of research on wireless attacks implemented by SDRs because SDRs support software codes for easy reconfiguration of physical and MAC layer parameters. In this article, we implement a data frame jammer and a fake ACK frame injector using SDR wireless open access research platform (WARP) boards to create an attack node that responds in real-time exploiting information from the IEEE 802.11 MAC layer. In [19], Bayraktaroglu *et al.* analyzed the saturation throughput of IEEE 802.11 under diverse jamming attacks, and they conducted simulation and real-world experimentation of jamming attacks, using universal software radio peripheral (USRP) SDRs in user datagram protocol (UDP) communication on IEEE 802.11. In [20], Patwardhan and Thuente performed jamming attacks in IEEE 802.11ac standardizing beamforming environment, the jammer is implemented with USRP boards. In [21], Vo-Huu *et al.* proposed and implemented a reactive interleaving jammer using USRP boards. The reactive interleaving jammer performs a fast response time of 30μs making it practical for high rates small packets.

D. SECURITY FUNCTIONS FOR WIRELESS ATTACKS IN IEEE 802.11

The IEEE 802.11 standards provide the wired equivalent privacy (WEP), Wi-Fi protected access (WPA), and WPA2 protocols for encryption and integrity [1]. The WEP supports an RC4 encryption algorithm with a key that combines a pre-shared key and a randomly generated initial vector (IV). The WEP is not recommended for use due to vulnerabilities in the RC4 encryption algorithm, and the WEP shared key used for authentication and encryption can be easily extracted [22], [23]. The WPA supports the RC4-temporal key integrity protocol (TKIP) encryption algorithm without replacing the WEP hardware by extending the WEP in software, but the WPA has the vulnerability of encryption key by wireless frame collection due to using the RC4 encryption algorithm [24], [25]. The WPA2 supports TKIP like WPA and additionally provides AES-CCM mode protocol (CCMP) with improved encryption function [26]. The WPA2 used in our experiment nodes in Section V are as follows. After a four-way handshake for the association between a client and an associated AP, they generate a pairwise master key (PMK) using the pre-shared key and SSID information in a wireless personal network. The client and AP negotiate session keys, the pairwise transient key (PTK) for encrypted unicast communication and the group temporal key (GTK) for encrypted multicast/broadcast communication, through the four-way handshake with encapsulation over LAN (EAPoL) frames. The PTK is determined by a parameter set of a pre-shared key, two nonce values from the client/AP, and the MAC addresses of the client/AP. In the 4-way handshake, the first and second EAPoL frames contain nonce information for the PTK, and the third and fourth EAPoL frames contain GTK information.

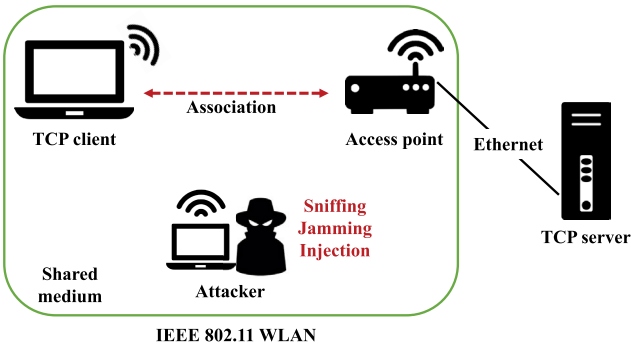


FIGURE 2. Attack scenario for TCP client-server communication in a wireless network.

However, if an attacker is present in a shared IEEE 802.11 medium, the attacker can sniff the EAPoL four-way handshake between the client and AP, and it can secure the nonce values. If the attacker acquires the pre-shared key and the MAC addresses of the client/AP, it can generate the PTK for WPA2 communication of the client and AP. Even if the attacker does not know the pre-shared key, it can find the pre-shared key via a brute force attack using a dictionary attack [27]. Recently, attacks exploiting new vulnerabilities of WPA2 have been proposed [28]–[30].

III. WIRELESS ATTACK SCENARIO WITH TCP CONNECTIONS

While IEEE 802.11 standards define the physical and link layer for wireless LANs, the most popular transport layer protocol is the TCP for reliable data transmission and flow control of Internet applications and services [31]. We consider a TCP client-server communication on a wireless network where a TCP client is associated with an AP, and an attacker is present in the IEEE 802.11 shared medium as shown in Fig. 2. In the scenario shown in Fig. 2, when a TCP client transmits TCP packets to an AP, the TCP packets are transmitted to MAC data frames through the shared wireless medium as shown in Fig. 3. Figure 3 shows the primary fields of a data frame that contain TCP contents. If the AP receives the MAC frames destined for the TCP server, the AP transmits the frames to the destination (i.e., TCP servers) for Ethernet packets. In this shared medium, the attacker node sniffs all wireless frames between the TCP client and AP, and the attacker node can perform jamming and inject malicious frames based on information obtained from the sniffed frames.

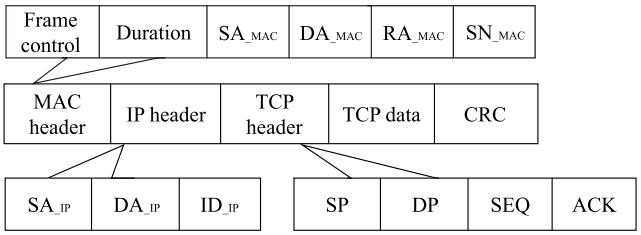


FIGURE 3. Data frame structure containing TCP packet.

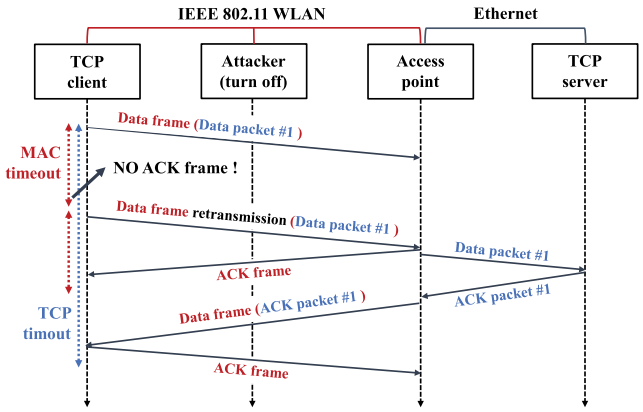


FIGURE 4. TCP client-server communication flow without attacks on IEEE 802.11 WLAN.

In TCP client-server communication, a TCP client establishes a session with a TCP server for reliable communication using a three-way handshake with data frames containing IP/TCP information as shown in Fig. 3, but the frames do not contain the TCP data. During the three-way handshake, the client and server exchange randomly set *SEQ* and *ACK* numbers in the TCP header field. After the three-way handshake, the TCP client transmits the data packet containing the TCP data to the TCP server with the data frame in the MAC layer. Here, the data packet means a data packet that contains the TCP data. Figure 4 shows the data exchange between TCP client and server in a network where the client is associated with IEEE 802.11 WLAN AP and the server is connected to Ethernet. If the TCP client fails to receive an ACK frame within the MAC timeout period (SIFS + ACK transmission duration + SlotTime) as shown in Fig. 4, the client retransmits the same data frame to the AP after executing the backoff algorithm [1]. When the AP receives the data frame without errors, the AP transmits the ACK frame to the TCP client. Then, the AP transmits the data packet to the TCP server over Ethernet. The TCP server then transmits the ACK packet to the TCP client after receiving the data packet if there is no error in the data packet. Here, the TCP ACK packet's *SEQ* number is equal to the TCP data packet's *ACK* number, and the TCP ACK packet's *ACK* number is added by the length of the TCP data packet bytes successfully received. The AP delivers the ACK packet to the TCP client with a data frame, and then the client announces it has received the data frame (ACK packet) with an ACK frame. If the TCP client receives the TCP ACK packet within the TCP timeout

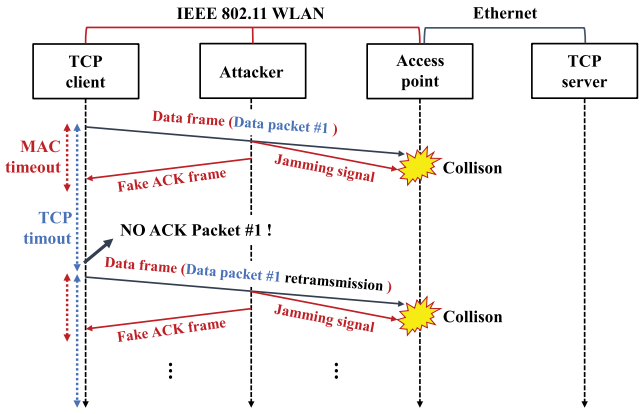


FIGURE 5. TCP client-server communication flow under data frame jamming with fake ACK frame injection on IEEE 802.11 WLAN.

in the transport layer, the TCP client determines that it has successfully transmitted the TCP data packet. However, if the TCP client does not receive a TCP ACK packet within the TCP timeout or receives a TCP ACK packet with incorrect *SEQ*/*ACK* values, the TCP client retransmits the TCP data packet from the transport layer. After the TCP client receives the final TCP packet, TCP client and TCP server disconnect the session via four-way handshake.

IV. MALICIOUS FRAME INJECTION ATTACK

We propose a malicious data frame injection attack without seizing association using jamming, sniffing, and spoofing in TCP communication. The proposed attack consists of a) data frame jamming with fake ACK frame injection and b) malicious data frame injection. First, we describe the data frame jamming with fake ACK frame injection between a TCP client and its associated AP in the situation TCP packets are encapsulated in IEEE 802.11 MAC frames. Second, we propose a malicious data frame injection that manipulates IEEE 802.11 MAC frames to replace an original TCP content with a malicious payload in time.

A. DATA FRAME JAMMING WITH FAKE ACK FRAME INJECTION

During the TCP three-way handshake between the TCP client and the TCP server, the attacker obtains the MAC address values (i.e., SA_{MAC}, DA_{MAC}, and RA_{MAC} in Fig. 3) of the frames that the TCP client sends to the TCP server. As shown in Fig. 5, the next step is that the attacker disrupts a data frame of the target TCP client by transmitting a jamming signal when the attacker receives a frame with the MAC address values that the TCP client uses. This data frame jamming makes the AP fail to receive correctly the data frame due to a cyclic redundancy check (CRC) error, and eventually the data frame is not delivered to the TCP server. Here, the attacker should know the exact time when the data frame transmission is completed to stop transmitting the jamming signal when the TCP client transmission is completed. The attacker can know the time when the transmission ends using the information of the duration field on the frame. If the RTS/CTS mechanism

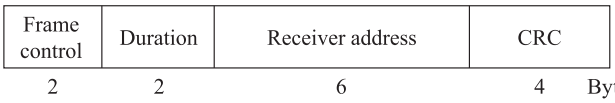


FIGURE 6. ACK frame structure.

is enabled, the attacker can use the duration field in the RTS frame sent by the client to find the transmission duration.

After the data frame jamming, the attacker generates a fake ACK frame and transmits the fake ACK frame to the TCP client node within the SIFS. Figure 6 shows that the fake ACK frame comprises 2-bytes frame control, 2-bytes duration, 6-bytes receiver address, and 4-bytes CRC fields. The attacker can extract the receiver address of the TCP client from the previous frames transmitted by the TCP client.

If the attacker succeeds in jamming the data frame and injecting fake ACK frame, the transmitter waits for a TCP ACK packet about the data packet#1. The TCP client does not retransmit the data frame in the MAC layer because the TCP client mistakes that the data frame transmission was successful by the fake ACK frame reception [14]. The TCP client starts to wait the TCP ACK packet until the TCP timeout in the transport layer. After the TCP timeout, the TCP client retransmits the data packet in the transport layer because the TCP client fails to receive the TCP ACK packet within the TCP timeout. When there is a data frame jamming with fake ACK frame spoofing in the MAC layer, the TCP client repeatedly transmits the same data packet#1 in the transport layer until the TCP retransmission limit set by the operating system has been reached. If the client repeats the transmission of the same packet until the TCP retransmission limit is reached, the transmitter will drop the packet in the transport layer. Due to the repeated transmissions with packet drops, the TCP session will be terminated abnormally and the TCP client is able to suspect that an attack exists during communication.

B. MALICIOUS DATA FRAME INJECTION

The attacker sniffs the previous MAC frames of TCP client node and collects information from the frames, which are exploited to generate spoofed frames. Furthermore, the attacker can selectively catch the jamming and injection timing by MAC frame sniffing. As shown in Fig. 7, after the data frame jamming with a fake ACK frame injection, the attacker transmits the malicious data frame including a malicious data packet to the AP. Here, the malicious data frame deceives the AP as if it were sent by a TCP client. If the AP receives the malicious data frame from the attacker, the AP transmits the ACK frame to the TCP client; however, the TCP client ignores the ACK frame because the client already received the fake ACK frame. After receiving the malicious data frame, the AP delivers it to the TCP server. As shown in Fig. 7, the TCP server receives the malicious data packet delayed by d compared to when there is no attack. Here, the attack delay d is the sum of the time for manipulating the malicious data frame and the time for

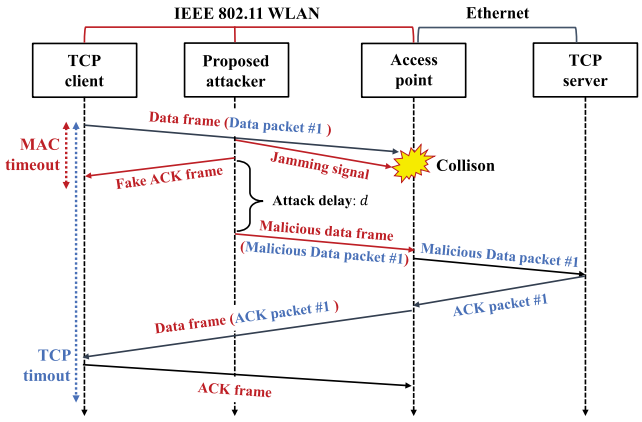


FIGURE 7. TCP client-server communication flow under malicious data frame injection on IEEE 802.11 WLAN.

transmission of the malicious data frame in the attacker node. When the TCP server receives the malicious data packet, the server transmits the TCP ACK packet containing an TCP *ACK* number increased by the length value of the malicious packet. If the AP receives the TCP ACK packet from the TCP server, the AP delivers the TCP ACK packet to the TCP client. If the TCP client receives the data packet within the TCP timeout period and the *ACK* number is correct, the TCP client determines that the data packet transmission was successful.

As shown in Fig. 3, to generate a MAC frame that contains malicious TCP contents, the attacker has to extract MAC information from the MAC header of the previous frames between the TCP client and AP, i.e., SA_{MAC}, DA_{MAC}, RA_{MAC}, and sequence number (SN_{MAC}) in the sequence control field. Whenever a client transmits a data frame, SN_{MAC} of the frame is incremented by one except in the case of frame retransmissions. If the client retransmits a data frame, SN_{MAC} does not change. The control frames such as the ACK frame and RTS/CTS frame do not contain the sequence control field. In addition, the attacker extracts information from the IP header, i.e., source IP address (SA_{IP}), destination IP address (DA_{IP}), and identification (ID_{IP}), and the TCP header, i.e., source port (SP), destination port(DP), and *SEQ*/*ACK* numbers. ID_{IP} is a unique identifier of a packet. Whenever a client transmits a packet, ID_{IP} of the packet is incremented by one if the packet is not fragmented. If a packet is fragmented, the fragmented packets have the same ID_{IP}s. After collecting the information shown in Fig. 3, the attacker implements a TCP data packet that is the same as the TCP client's data packet header; however, this data packet will contain a malicious payload that the attacker wants to inject. Here, the attacker should configure the length of the malicious TCP data to be equal to the length of TCP data transmitted by the TCP client in order to escape from errors in the transport layer of the TCP client about incorrect TCP *SEQ*/*ACK* numbers. If the TCP client receives the ACK packet that contains incorrect TCP *SEQ*/*ACK* numbers from the TCP server, the TCP client retransmits the TCP data packet transmitted previously. Note that the attack node can utilize the full-duplex technique [32]

Algorithm 1 Malicious Frame Injector in TCP Communication

1: // Set *m_frame* MAC header address fields
2: SA_{MAC}, DA_{MAC}, RA_{MAC} ← target's MAC addresses
3: // Set *m_frame* IP header address fields
4: SA_{IP}, DA_{IP} ← target's IP addresses
5: *count* ← 0
6: **while** *count* < 3 **do**
7: **if** There is a new data frame from the target **then**
8: *s_frame* ← new data frame
9: *count* + = 1
10: **end if**
11: **if** *count* == 2 **then**
12: *m_frame.SN* ← *s_frame.SN* + 1
13: *m_frame.ID* ← *s_frame.ID* + 1
14: $m\_frame.header_{TCP}$ ← $s\_frame.header_{TCP}$
15: $m\_frame.data_{TCP}$ ← malicious TCP data
16: Set IP, TCP checksum, and CRC
17: Transmission *m_frame*
18: *count* + = 1
19: **end if**
20: **end while**

to obtain (i.e., eavesdrop) the length information of the data packet transmitted by the client by jamming the data packet simultaneously.

The procedure of the malicious frame injector is described in Algorithm 1. The malicious data frame *m_frame* and the recent data frame from the target client *s_frame* have the data frame structure shown in Fig. 3. In Algorithm 1, the addresses of *m_frame* in the MAC header and IP header are set to MAC and IP addresses of target on line 2 and 4. To inject *m_frame* after the TCP three-way handshake between the TCP client and TCP server, *count* is used to check the number of received frames from the victim client within the while loop. If there is a newly received data frame from the victim client, the new frame is saved in *s_frame*, and *count* is increased by one (lines 7 to 9). When the malicious frame injector receives the second data frames from the TCP client for the three-way handshake, *count* becomes two. When *count* becomes two, SN_{MAC} of *m_frame* is set to a value increased by one from SN_{MAC} of *s_frame*, and similarly, ID_{IP} of *m_frame* is set to a value increased by one from ID_{IP} of *s_frame* (lines 11 to 13). Here, we assume that the packet from the target client is not fragmented. On line 14, *m_frame* sets *SP*, *DP*, *SEQ*, and *ACK* to those in the TCP header from *s_frame*. After filling the TCP data in *m_frame* payload with malicious TCP data (line 15), the checksums of the IP and TCP headers, as well as the CRC of the frame, are calculated (line 16). Finally, the malicious data frame is transmitted (line 17), and *count* becomes three (line 18).

C. PERFORMANCE ANALYSIS

When there exists the proposed malicious data frame injection attack, if the TCP client fails to receive the ACK packet from the TCP server within the TCP timeout, the TCP client retransmits the data packet to the TCP server. In this case, the proposed attacker fails to inject a malicious frame. To analyze the probability of the proposed attack failure/success in a TCP communication, we deal with round trip time (RTT) between the TCP server/client and retransmission timeout (RTO) value about the TCP timeout. For example, when a TCP client transmits a data packet to a TCP server, the RTT of TCP client is the duration between the time of data packet transmission and the time of the ACK packet reception from the TCP server. The RTO value is calculated by the past RTT values. We consider Jacobson's RTO algorithm mostly used in current TCP implementations [33]. The RTO for packet is given by the following equation [34]:

$$RTO = MRTT + \alpha VRTT, \tag{1}$$

where *MRTT* is the mean RTT, *VRTT* is the RTT variation, and α is a positive constant, typically $\alpha = 4$.

When there are not attacks, we assume that the normal RTT is a random variable X_1. Usually, the RTTs are modeled by a shifted Gamma distribution based on experimental results [35], [36]. When the random variable $X_1 \sim Gamma(\lambda, \beta)$, the shifted Gamma distribution is given by

$$f_{X_1}(x) = \begin{cases} \dfrac{1}{\Gamma(\lambda)\beta^{\lambda}} x^{\lambda-1} e^{-\frac{x}{\beta}}, x \geq \gamma \text{ if } \lambda > 0 \\ \delta(x-\gamma) \text{ if } \lambda = 0 \end{cases} \tag{2}$$

where $\lambda > 0$ is the scale parameter, $\beta > 0$ is the shape parameter, $\Gamma(\cdot)$ is the Gamma function, $\Gamma(\lambda) = \int_0^{+\infty} z^{\lambda-1} e^{-z} dz$ for $\lambda > 0$, γ is the shifted value, and $\delta(\cdot)$ is the unit impulse function. The X_1 has mean $\gamma + \beta\lambda$ and variance $\beta^2\lambda$. Denote that the delay of the proposed attack d shown in Fig. 7 is a random variable X_2. Here, we assume that the X_1 and X_2 are independent. When there is the proposed attack, the RTT has the random variable $Y = X_1 + X_2$. In a normal situation without attack, *RTT* is RTT_n and *RTO* is RTO_n. Here, the probability of the proposed attack failure is defined by $P(Y > RTO_n) = 1 - P(Y \leq RTO_n) = 1 - \int_{-\infty}^{RTO_n} f_Y(y)dy$.

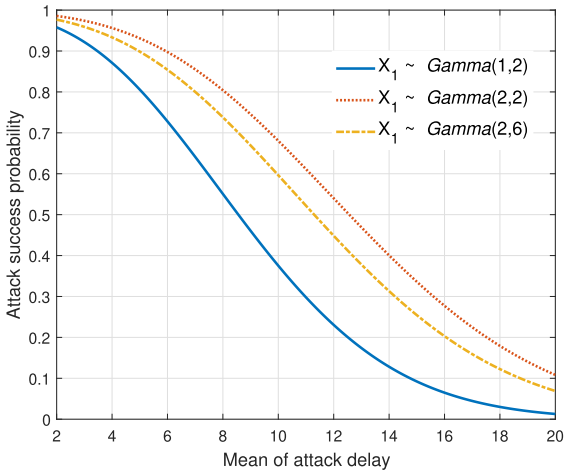


FIGURE 8. The proposed attack success probability within TCP timeout.

FIGURE 9. Malicious frame injection test-bed.

If we assume that the delay of the proposed attack random variable X_2 follows the shifted Gamma distribution $X_2 \sim Gamma(\lambda_2, \beta)$, and the $X_1 \sim Gamma(\lambda_1, \beta)$, the RTTs with the proposed attack Y follow the shifted Gamma distribution $Y \sim Gamma(\lambda_1 + \lambda_2, \beta)$. To simplify RTO_n in this simulation, we set RTO_n to sum of the mean of X_1 and the standard deviation of X_1 multiplied by α. In this simulation, we set the shape parameter of X_2 is the same with the shape parameter of X_1, α is 4, and the shifted value γ is 2. Figure 8 shows the proposed attack success probability decreases when the mean of proposed attack delay increases. When the mean and deviation of X_1 are increased, the attack success probability is increased at the same attack delay, as shown in cases of $X_1 \sim Gamma(1, 2)$, $X_1 \sim Gamma(2, 2)$, and $X_1 \sim Gamma(2, 6)$. The proposed attacker lowers the delay of the RTT, injecting the fake ACK frame after the data frame jamming. When there is the data frame jamming with fake ACK frame injection, the probability of data frame retransmission in the TCP client's MAC layer is lower than the data frame jamming without fake ACK frame injection [14].

V. IMPLEMENTATION AND EVALUATION

In this section, we describe the experimental setup of test-bed, implementation of attacker node, experimental scenario and verification of the proposed attack. For a realistic attack scenario, we setup an HTTP client-server communication on an IEEE 802.11 wireless LAN where the data exchange between the AP and off-the-shelf clients is protected by WPA2. Through the experimental evaluation, we demonstrate how the proposed attack can deceive a victim client that sends an HTTP request to a server through its associated AP with WPA2 security capability, and confirm that this attack is hard to be detected by packet monitoring on the TCP client-server connection.

A. EXPERIMENTAL SETUP AND IMPLEMENTATION

As shown in Fig. 9, we have implemented a test-bed using two laptops for HTTP client and HTTP server, an off-the-self AP, and an SDR based attacker. For the HTTP client and server, we use laptops with an Intel PRO/Wireless 3945ABG WLAN card supporting IEEE 802.11a/b/g modulation. We use an ipTIME A1004 AP supporting IEEE 802.11 a/b/g modulation and WPA2-AES security mode. Here, the HTTP client associates with the AP in the IEEE 802.11g environment and knows the IP address of the HTTP server connected by Ethernet link. For data encryption and integrity,

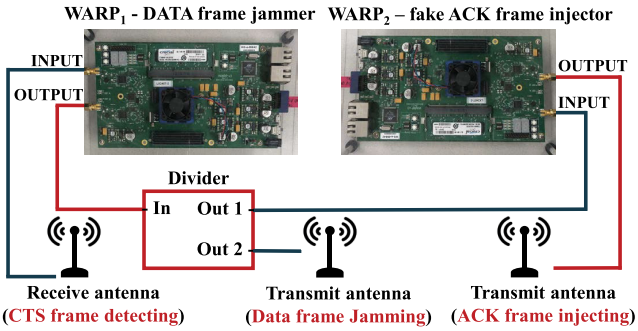


FIGURE 10. Data frame jammer and fake ACK frame injector.

the AP and the HTTP client use the WPA2 protocol in IEEE 802.11 communication. We set the IP address of the HTTP client to 192.168.0.4 and that of HTTP server to 192.168.0.2.

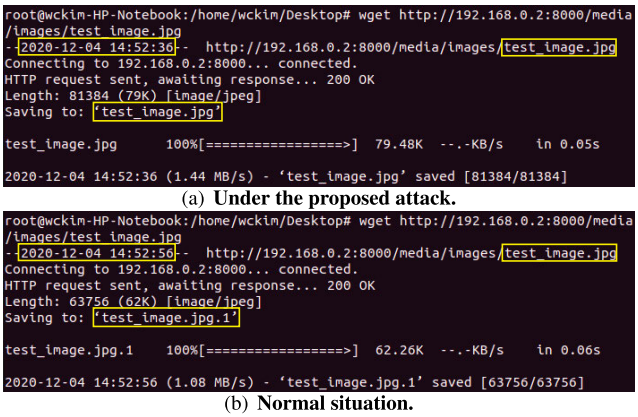
The proposed attacker comprises two components: i.e., a data frame jammer with fake ACK frame injector, and a malicious frame injector. We use SDRs to implement a data frame jammer with a fake ACK frame injector to perform the MAC layer attack that requires a real-time reaction, and use a laptop as the malicious frame injector dealing with packets in the upper layer. As shown in Fig. 10, we have implemented the data frame jamming and fake ACK frame injecting node using two WARPv3 SDRs and the WARPNet software framework [18]. The WARPv3 is the newest generation of WARP hardware, integrating a Virtex-6 field-programmable gate array (FPGA), two programmable radio frequency (RF) interfaces and diverse peripheral devices. We publish our source code for the data frame jamming and fake ACK frame injector node over Github [37]. The two SDRs for the data frame jammer and the ACK frame injector are directly connected with wired cables. To deliver jamming frame signal generated by the data frame jammer to the fake ACK frame injector, we use a two-way power divider, which passes the signals received at *in* to both *out*1 and *out*2. The data frame jammer selectively disrupts the data frames that contain the target's MAC address by referring to MAC addresses of the RTS/CTS frames that are transmitted before the transmission of the data frame. When the data frame jammer receives the CTS frame transmitted from the target AP to the client node via the receive antenna, after the SIFS interval, the data frame jammer transmits a jamming frame to the shared channel through the transmit antenna for *out*2 on the divider. Here, the shape of the jamming frame is the same as that of the ACK frame. When the AP receives the data frame from the

victim node, the AP discards the data frame due to the CRC error caused by collision with the jamming frame that has the shape of the ACK frame. When the ACK frame injector receives a jamming frame through the wired cable for *out*1 on the divider without any collision, after the SIFS interval, the injector transmits a fake ACK frame to the victim node. At this point, the destination MAC address of the fake ACK frame is the same as the MAC address of the victim node. Note that we have placed the CTS frame detecting antenna close to the AP to quickly respond to the CTS generated by the AP, and the data frame jamming antenna is also located close to the AP such that it can interfere with the AP's data frame reception at high probability. In addition, we have placed the ACK frame injecting antenna close to the data frame transmission node that should receive the ACK frame.

The malicious frame injector is comprised of a laptop with a TL-WN722NC WLAN card supporting IEEE 802.11 monitor mode, and we have implemented the malicious frame injecting tool using the Python Scapy library [38] on Ubuntu 14.04. After performing the data frame jamming with a fake ACK frame injection on the SDRs, the malicious frame injector transmits the malicious data frame to the target AP. Because the TCP client is not able to detect the data frame loss by the fake ACK frame injection, it continues to wait TCP ACK packet from the HTTP server. Meanwhile, the attacker generates a malicious data frame with the spoofed MAC and IP addresses using the injection Python program, and injects the malicious data frame to the AP through the laptop Wi-Fi interface card. Note that the interval for TCP ACK retransmission timeout (RTO) is long enough for the attacker to generate and inject the malicious data frame by the user-level application program. This injection can be performed because the link-level retransmission is disabled by the fake ACK frame injection. Note that IEEE 802.11 DCF MAC protocol attempts to retransmit a dropped MAC frame in a short random backoff.

B. EXPERIMENTAL SCENARIO

In our scenario, an attacker is to force an HTTP client to unwittingly download a malicious image file when the HTTP client sends a HTTP request to download a normal image. We set up an HTTP server to provide a bulletin board service to upload and download images. Firstly, the attacker uploads a malicious file on the server bulletin board. We assume that the normal image is *test_image.jpg* and the malicious image is *test_lmage.jpg* and that *test_image.jpg* is one of the highly accessed images such as a bulletin board log. The length of the file names should be the same in our implementation. Secondly, the attacker chooses a target wireless node that runs an HTTP client to download the *test_image.jpg* from the HTTP server, and identifies the AP associated with the target node. Thirdly, the attacker generates a Wi-Fi WPA2 master session key of the target node and the AP. In order to generate the WPA2 master session key, the attacker needs information of target client and AP's MAC addresses, two nonce values of EAPoL frames, and pre-shared key [39]. The attacker



(a) **Under the proposed attack.**

(b) **Normal situation.**

FIGURE 11. **HTTP request for downloading the *test_image* on HTTP client.**

sniffs the exchanges of EAPoL frames between the HTTP client and the AP to get information of nonce values. Note that the HTTP client node and the AP perform an EAPoL four-way handshake, after a four-way handshake to establish the association between the HTTP client and the AP. The attacker uses a brute force attack tool based on a dictionary attack [27] to find the pre-shared key. Using EAPoL frame information (nonce values) with targets' MAC addresses and the pre-shared key, the attacker generates the WPA2 master session key.

Fourthly, if the attacker sniffs encrypted frames between the target node and the AP using the WPA2 master session key, and finds a TCP three-way handshake between the HTTP client and HTTP server, after the three-way handshake, the attacker transmits jamming signal and fake ACK frame. Then, the attacker generates malicious data frames encrypted by the WPA2 master session key, and transmits the encrypted malicious HTTP request message to the AP. Here, the malicious HTTP request packet contains a payload message to download *test_lmage.jpg* rather than *test_image.jpg*. If the HTTP server receives the malicious HTTP request to download *test_lmage.jpg* from the attacker, the HTTP server transmits HTTP response packets to the victim node running the HTTP client because the source address of the malicious HTTP request packet is filled with that of the victim node. Finally, the HTTP client receives packets containing *test_lmage.jpg*, not *test_image.jpg*.

C. VERIFICATION

The HTTP client generates an HTTP GET request including uniform resource locator (URL) for downloading *test_image.jpg* to the HTTP server. To generate the HTTP request, the HTTP client uses Wget command built in the Ubuntu 20.04.1 LTS environment. We compare the results of image download when an attack is performed and is not performed. As shown in Fig. 11(a), in the proposed attack, the HTTP client sends a request for *test_image.jpg* to the HTTP server. Figure 11(a) shows that the HTTP client saves *test_image.jpg* (file size 81,384 bytes) with the same name (*test_image.jpg*) in the local file system.

(a) Injected image under the proposed attack.

(b) Normal image without the proposed attack.

FIGURE 12. Injected image (the image file name is *test_image* but the image is *test_lmage*) and normal image (*test_image*).

FIGURE 13. HTTP server request log.

The downloaded image of *test_image.jpg* is the injected one as shown in Fig. 12(a). Then, we turn off the proposed attack. The HTTP client uses Wget to download the same image of *test_image.jpg*, as shown in Fig. 11(b). At this time, the Wget command downloads the file named *test_image.jpg*.1 automatically because *test_image.jpg* exists in the same local file system. The downloaded image of *test_image.jpg*.1 is the normal one as shown in Fig. 12(b). However, the image size is 63,756 bytes, and the content of *test_image.jpg*.1 is also different from *test_image.jpg*, as shown in Fig. 12. Under the proposed attack, the client believes that the normal *test_image.jpg* file is received and saved. However, the image file (*test_image.jpg*) is the malicious image file of *test_lamge.jpg* in the HTTP server because a *test_lamge.jpg* request was delivered to the HTTP server from the attacker instead of the *test_image.jpg* request as shown in Fig. 13.

To validate the procedures of the proposed attack, we have captured the packets on both the HTTP server and HTTP client using Wireshark [40] during the HTTP *test_iamge.jpg* download. Fig. 14(a) shows the packets captured on the HTTP client, and Fig. 14(b)

No	Time	Source	Destination	Proto	Len	Info
1	0.00…	192.168.0.4	192.168.0.2	TCP	74	59996 → 8000 [SYN] Seq=0
2	0.00…	192.168.0.2	192.168.0.4	TCP	74	8000 → 59996 [SYN, ACK] Seq=0 Ack=1
3	0.00…	192.168.0.4	192.168.0.2	TCP	66	59996 → 8000 [ACK] Seq=1 Ack=1
4	0.00…	192.168.0.4	192.168.0.2	HTTP	236	GET /media/images/test_image.jpg HTTP/1.1
5	0.02…	192.168.0.2	192.168.0.4	TCP	66	8000 → 59996 [ACK] Seq=1 Ack=171
6	0.02…	192.168.0.2	192.168.0.4	TCP	83	8000 → 59996 [PSH, ACK] Seq=1 Ack=171
7	0.02…	192.168.0.4	192.168.0.2	TCP	66	59996 → 8000 [ACK] Seq=171 Ack=18
8	0.02…	192.168.0.2	192.168.0.4	TCP	5858	8000 → 59996 [ACK] Seq=18 Ack=171
9	0.02…	192.168.0.4	192.168.0.2	TCP	66	59996 → 8000 [ACK] Seq=171 Ack=5810
10	0.02…	192.168.0.2	192.168.0.4	TCP	1514	8000 → 59996 [ACK] Seq=5810 Ack=171
11	0.02…	192.168.0.4	192.168.0.2	TCP	66	59996 → 8000 [ACK] Seq=171 Ack=7258

(a) HTTP client Wireshark capture.

No	Time	Source	Destination	Proto	Len	Info
14	20.8…	192.168.0.4	192.168.0.2	TCP	74	59996 → 8000 [SYN] Seq=0
15	20.8…	192.168.0.2	192.168.0.4	TCP	74	8000 → 59996 [SYN, ACK] Seq=0 Ack=1
16	20.8…	192.168.0.4	192.168.0.2	TCP	66	59996 → 8000 [ACK] Seq=1 Ack=1
17	20.9…	192.168.0.4	192.168.0.2	HTTP	236	GET /media/images/test_lmage.jpg HTTP/1.1
18	20.9…	192.168.0.2	192.168.0.4	TCP	66	8000 → 59996 [ACK] Seq=1 Ack=171
19	20.9…	192.168.0.2	192.168.0.4	TCP	83	8000 → 59996 [PSH, ACK] Seq=1 Ack=171
20	20.9…	192.168.0.2	192.168.0.4	TCP	1514	8000 → 59996 [ACK] Seq=18 Ack=171
21	20.9…	192.168.0.2	192.168.0.4	TCP	1514	8000 → 59996 [ACK] Seq=1466 Ack=171
22	20.9…	192.168.0.2	192.168.0.4	TCP	1514	8000 → 59996 [ACK] Seq=2914 Ack=171
23	20.9…	192.168.0.2	192.168.0.4	TCP	1514	8000 → 59996 [ACK] Seq=4362 Ack=171
24	20.9…	192.168.0.2	192.168.0.4	TCP	1514	8000 → 59996 [ACK] Seq=5810 Ack=171

(b) HTTP server Wireshark capture.

FIGURE 14. Wireshark captures on TCP packets HTTP client and HTTP server under the proposed attack.

shows the packets on the HTTP server. Here, after the TCP three-way handshake, the HTTP client transmitted an HTTP packet with *'/media/images/test_image.jpg'*; however, the HTTP server received an HTTP packet with *'/media/images/test_lmage.jpg'*. After this request, the HTTP server transmitted *test_lmage.jpg* file packets (Fig. 14(b)). Then, the HTTP client received the response packets (Fig. 14(a)). Under the proposed attack, there is no way that the HTTP client figures out the downloaded content is not the same that it requests to download. Consequently, the HTTP client cannot avoid receiving the malicious content without being aware of the existence of the attack. In Fig. 14, we cannot find any packet retransmissions during the attack, and it is nearly not possible that the HTTP client and server detect the attack by monitoring packet flows on the transport layer. Additionally, when there is the proposed attack, the average RTT of the HTTP request packet is about 0.03 seconds. When there is no attack, the average RTT of the HTTP request packet is about 0.0007 seconds. The reported values are the average of 10 runs.

VI. COUNTERMEASURES

The proposed attack consists of data frame jamming with fake ACK frame spoofing that exploits the broadcast properties of radio frames for the physical and MAC layers, and HTTP data packet spoofing technique through the transport and application layers. We have shown experiment results that the proposed attack is fatal and difficult to be detected in the existing IEEE 802.11 environment despite using WPA2. In this section, we instantiate methods to detect and prevent the proposed attack.

Firstly, in the MAC layer, the TCP client can suspect the fake ACK frame injection, if it counts the number of received ACK frames. In Fig. 7, after the reception of the malicious frame in the AP, the AP transmits an ACK frame to the TCP client. However, the TCP client discards the ACK frame in the existing IEEE 802.11 system because the TCP client

already received the fake ACK frame before the generation of the malicious data frame. The TCP client receives two ACK frames per one data frame in case that there exists the fake ACK frame injection attack and therefore can suspect the fake ACK frame injection attack in the MAC layer.

Secondly, the TCP client can suspect the malicious data frame attack by monitoring the addresses of data frames on the shared channel. In normal IEEE 802.11 environments, nodes only receive frames containing their address or broadcast/multicast address in the destination address field of the frame. If nodes receive and monitor frames containing their address in the source address field, the nodes can detect the malicious data frame attack.

Thirdly, in the application layer, the TCP client/server can prevent the malicious data packet injection using a secure application-layer protocol based on the secure sockets layer (SSL)/transport layer security (TLS). If the HTTP client/server nodes communicate using the SSL/TLS such as the HTTPs, our attacker cannot eavesdrop on encrypted HTTP packets between HTTP client/server nodes and manipulate HTTP packets containing malicious contents. However, if the attacker cracks SSL/TLS based applications by exploiting vulnerabilities of the SSL/TLS, the attacker can eavesdrop and manipulate encrypted HTTP packets in HTTPs communication [41], [42].

VII. CONCLUSION

In this article, we have proposed a malicious data frame injection attack that does not require to seize an association between an AP and a client in IEEE 802.11 wireless LANs. We have implemented the proposed attacker that transmits a jamming signal and the manipulated Wi-Fi MAC frames using SDRs and injects spoofed TCP packets with malicious HTTP requests using Scapy library tool. Through real-world HTTP service experiments on a constructed test-bed, we validated that the proposed attack is successfully applicable to TCP-based HTTP application communication on IEEE 802.11 networks with WPA2-AES enabled. Our proposed attack can be applied to various scenarios such as domain name system (DNS), file transfer protocol (FTP) and so on. As future work, we will investigate how to make IEEE 802.11 WLANs more secure against malicious data frame injection attacks using a fake ACK frame injection.

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*, IEEE Standard 802.11, 2016, pp. 1–3534.

[2] C. Kohlios and T. Hayajneh, "A comprehensive attack flow model and security analysis for Wi-Fi and WPA3," *Electronics*, vol. 7, no. 11, p. 284, Oct. 2018.

[3] V. Roth, W. Polak, E. Rieffel, and T. Turner, "Simple and effective defense against evil twin access points," in *Proc. 1st ACM Conf. Wireless Netw. Secur. (WiSec)*, 2008, pp. 220–235.

[4] Y. Song, C. Yang, and G. Gu, "Active user-side evil twin access point detection using statistical techniques," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 5, pp. 1638–1651, Oct. 2012.

[5] V. Ramachandran and S. Nandi, "Detecting ARP spoofing: An active technique," in *Proc. Int. Conf. Inf. Syst. Secur.* Springer, 2005, pp. 239–250.

[6] X. Hou, Z. Jiang, and X. Tian, "The detection and prevention for ARP spoofing based on snort," in *Proc. Int. Conf. Comput. Appl. Syst. Modeling (ICCASM)*, Oct. 2010, vol. 171, no. 5,pp. 137–139.

[7] P. Satam and S. Hariri, "WIDS: An anomaly based intrusion detection system for Wi-Fi (IEEE 802.11) protocol," *IEEE Trans. Netw. Service Manage.*, early access, Nov. 5, 2020, doi: 10.1109/TNSM.2020.3036138.

[8] *Airpwn Sourceforge Website*. [Online]. Available: http://airpwn.sourceforge.net/Airpwn.html

[9] M. Lichtman, J. D. Poston, S. Amuru, C. Shahriar, T. C. Clancy, R. M. Buehrer, and J. H. Reed, "A communications jamming taxonomy," *IEEE Secur. Privacy*, vol. 14, no. 1, pp. 47–54, Jan. 2016.

[10] N. Yasar, F. M. Yasar, and Y. Topcu, "Operational advantages of using cyber electronic warfare (CEW) in the battlefield," *Proc. SPIE*, vol. 8408, May 2012, Art. no. 84080G.

[11] O. Askin, R. Irmak, and M. Avsever, "Cyber warfare and electronic warfare integration in the operational environment of the future: Cyber electronic warfare," *Proc. SPIE*, vol. 9458, May 2015, Art. no. 94580H.

[12] T. David and A. Mithun, "Intelligent jamming in wireless networks with applications to 802.11b and other networks," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, vol. 6, Dec. 2006, p. 100.

[13] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proc. IEEE*, vol. 104, no. 9, pp. 1727–1765, Sep. 2016.

[14] W. Kim, J. Park, J. Jo, and H. Lim, "Covert jamming using fake ACK frame injection on IEEE 802.11 wireless LANs," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1502–1505, Oct. 2019.

[15] W. Yin, P. Hu, J. Wen, and H. Zhou, "ACK spoofing on MAC-layer rate control: Attacks and defenses," *Comput. Netw.*, vol. 171, Apr. 2020, Art. no. 107133.

[16] S. Whalen. (2001). *An Introduction to ARP Spoofing*. [Online]. Available: http://packetstormsecurity.nl/papers/protocols/intro to arp spoofing.pdf

[17] *Universal Software Radio Peripheral*. [Online]. Available: http://www.ettus.com

[18] *Warp Project*. [Online]. Available: http://warpproject.org

[19] E. Bayraktaroglu, C. King, X. Liu, G. Noubir, R. Rajaraman, and B. Thapa, "Performance of IEEE 802.11 under jamming," *Mobile Netw. Appl.*, vol. 18, no. 5, pp. 678–696, 2013.

[20] G. Patwardhan and D. Thuente, "Jamming beamforming: A new attack vector in jamming IEEE 802.11ac networks," in *Proc. IEEE Mil. Commun. Conf.*, Oct. 2014, pp. 1534–1541.

[21] T. D. Vo-Huu, T. D. Vo-Huu, and G. Noubir, "Interleaving jamming in Wi-Fi networks," in *Proc. 9th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, Jul. 2016, pp. 31–42.

[22] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *Proc. Int. Workshop Sel. Areas Cryptogr. (SAC)*. Springer, 2001, pp. 1–24.

[23] A. Stubblefield, J. Ioannidis, and A. D. Rubin, "Using the Fluhrer, Mantin, and Shamir attack to break WEP," in *Proc. Int. Workshop Fast Softw. Encryption*, 2002.

[24] K. G. Paterson, B. Poettering, and J. C. N. Schuldt, "Plaintext recovery attacks against WPA/TKIP," in *Proc. Int. Workshop Fast Softw. Encryption*. Springer, 2014, pp. 325–349.

[25] E. Tews and M. Beck, "Practical attacks against WEP and WPA," in *Proc. 2nd ACM Conf. Wireless Netw. Secur. (WiSec)*, 2009, pp. 79–86.

[26] A. Sari and M. Karay, "Performance of IEEE 802.11 under jamming," *Int. J. Commun., Netw. Syst. Sci.*, vol. 8, no. 12, pp. 483–491, 2015.

[27] *Aircrack*. [Online]. Available: https://www.aircrack-ng.org/doku.php?id=cracking_wpa

[28] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1313–1328.

[29] M. Vanhoef and F. Piessens, "Release the kraken: New KRACKs in the 802.11 standard," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 299–314.

[30] J. Steube. *New Attack on WPA/WPA2 Using PMKID*. [Online]. Available: https://hashcat.net/forum/thread-7717.html

[31] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K.-J. Grinnemo, P. Hurtig, N. Khademi, M. Tuxen, M. Welzl, D. Damjanovic, and S. Mangiante, "De-ossifying the Internet transport layer: A survey and future perspectives," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 619–639, 1st Quart., 2017.

[32] J. Moon, H. Lee, C. Song, S. Lee, and I. Lee, "Proactive eavesdropping with full-duplex relay and cooperative jamming," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6707–6719, Oct. 2018.
[33] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer," *Computer*, 2011.
[34] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, 1988.
[35] A. Mukherjee, "On the dynamics and significance of low frequency components of Internet load," *Internetworking, Res. Exper.*, vol. 5, pp. 163–204, 1994.
[36] L. Ma, K. E. Barner, and G. R. Arce, "Statistical analysis of TCP's retransmission timeout algorithm," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 383–396, Apr. 2006.
[37] *Data Frame Jammer and ACK Frame Injector Source Code*. [Online]. Available: https://github.com/woogotcha/Datajam-Ackinject
[38] *Scapy Framework*. [Online]. Available: https://scapy.readthedocs.io/en/latest/
[39] V. Kumkar, A. Tiwari, P. Tiwari, A. Gupta, and S. Shrawne, "Vulnerabilities of wireless security protocols (WEP and WPA2)," *Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET)*, vol. 1, no. 2, pp. 34–38, 2012.
[40] *Wireshark*. [Online]. Available: https://www.wireshark.org/
[41] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the HTTPS protocol," *IEEE Secur. Privacy*, vol. 7, no. 1, pp. 78–81, Jan./Feb. 2009.
[42] S. W. Han, H. Kwon, C. Hahn, D. Koo, and J. Hur, "A survey on MITM and its countermeasures in the TLS handshake protocol," in *Proc. Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2016, pp. 724–739.

WOOCHEOL KIM received the B.S. degree from the School of Convergence Security and Computer Science (double major), Kyonggi University, Suwon, South Korea, in 2017, and the M.S. degree from the School of Electrical Engineering and Computer Science (EECS), Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea, in 2019, where he is currently pursuing the Ph.D. degree. His research interests include cybersecurity for various network domains and the technologies on electronic warfare and cyber electronic warfare.

SOYEON KIM received the B.S. degree in computer engineering from Chonnam National University, Gwangju, South Korea, in 1998, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 1999 and 2016, respectively. She has been a Senior Researcher with the Agency for Defense Development (ADD), Daejeon, South Korea, since 2001. Her research interests include signal processing, pattern recognition, jamming, and the technologies on electronic warfare and cyber electronic warfare.

HYUK LIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, South Korea, in 1996, 1998, and 2003, respectively. From 2003 to 2006, he was a Postdoctoral Research Associate with the Department of Computer Science, University of Illinois at Urbana–Champaign, Champaign, IL, USA. He is currently a Full Professor with the AI Graduate School and jointly with the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea. His research interests include wired and wireless networks, cyber-security, and artificial intelligence.