The Danger of Packet Length Leakage: Off-path TCP/IP Hijacking Attacks Against Wireless and Mobile Networks

Guancheng Li*§, Minghao Zhang†§, Jianjun Chen†✉, Ge Dai*,
Pinji Chen†, Huiming Liu*, Yang Yu*, Haixin Duan†, Zhiyun Qian‡
*Tencent Security Xuanwu Lab
Email: {atumli,huimingliu,Tkyu}@tencent.com, dg.icepng@gmail.com
†Tsinghua University
Email: {zhangmh24, cpj24}@mails.tsinghua.edu.cn, {jianjun,duanhx}@tsinghua.edu.cn
‡University of California, Riverside
Email: zhiyunq@cs.ucr.edu

***Abstract*—To combat eavesdropping and injection attacks, wireless networks widely adopt encryption to provide confidentiality and integrity guarantees. In this paper, we present a novel and generic attack, termed LenOracle, which can hijack the TCP/UDP connections over encrypted wireless networks (e.g., 5G/4G/3G and Wi-Fi) via packet injections from the Internet. Due to the design nature of wireless networks and stream ciphers they used, the length of IP packets being transmitted can be acquired by radio sniffing. It thus provides a side channel for adversaries. We found that adversaries could utilize this side channel with TCP features to infer the presence of a connection, infer the protocol state (sequence number, acknowledge number) of the connection, and finally hijack TCP/IP connections over wireless networks. Through real-world experiments in commercial LTE networks and real Wi-Fi networks, we demonstrated that the LenOracle attack is practical and severe against both TCP and UDP connections. For the former, we successfully injected a fake short message into a victim TCP connection; For the latter, we were able to inject a fake DNS response into a UDP connection and poisoned the DNS cache of the victim device. Following the responsible disclosure policy, we have reported our findings and mitigation recommendations to GSMA and Wi-Fi Alliance. The GSMA acknowledged that the issue affects 5G/4G/3G, notified all its members (operators and vendors worldwide) of this issue, and highlighted the mitigation we proposed.**

***Index Terms*—Network Security, Mobile Security, TCP/IP Hijacking Attack, Side Channel**

1. Introduction

Wireless networks have become a pivotal avenue for Internet access. Compared with wired networks, wireless networks are inherently more susceptible to eavesdropping and injection attacks. In response, the community has developed various security protocols that ensure confidentiality and integrity. For instance, the Wi-Fi Alliance presents Wi-Fi Protected Access (WPA) protocols to protect Wi-Fi networks. The 3GPP group introduces encryption mechanisms to protect 5G/4G/3G mobile networks. Those encryption mechanisms provide a strong security guarantee against packet leakage or injection attacks.

In this paper, we propose a novel attack, named LenOracle Attack. This attack enables an off-path attacker to infer the presence and hijack arbitrary TCP/IP connections, over the victim wireless network protected by encryption mechanisms. Compared with previous attacks [38], [46], this attack is general and affects multiple major wireless protocols, including but not limited to 5G/4G/3G and Wi-Fi.

The root cause of the vulnerability lies in the inherent characteristics of wireless networks. Wireless networks widely adopt stream ciphers to protect their transmission. By the nature of stream ciphers, the length of ciphertext is equal to the length of the corresponding plaintext. Therefore, it is feasible to build a side channel that observes the length of IP packets being transmitted over wireless networks protected by stream ciphers (e.g., 5G/4G/3G and Wi-Fi). This widely available side channel seems to be trivial as it only tells the length. However, we found that this trivial side channel, when combined with the functionalities of NAT and TCP, can be exploited to infer the presence and protocol state of TCP or UDP connections. By employing a series of "guess-then-check" strategies, this method enables the execution of sophisticated TCP/IP hijacking attacks.

The attack follows a three-phase approach to infer critical connection details and manipulate TCP/IP connections. First, the attacker identifies active connections by sending spoofed data packets for different four-tuples (including the client's IP address and port, the server's IP address and port) and observing which packets pass through the NAT using a radio sniffer. This creates an opportunity for the attacker to probe and infer the four-tuple by exploiting the NAT device positioned before the target client. Second, the attacker exploits challenge ACK features to locate the sequence window by sending spoofed reset packets to provoke challenge ACK responses from the server, segmenting the sequence space to efficiently find numbers within the server's receive window.

§. Both authors contributed equally to this work
✉. Corresponding author: jianjun@tsinghua.edu.cn

Finally, the attacker infers acknowledge number and exact sequence number by sending PSH-ACK packets, fine-tuning the acknowledge and sequence numbers within recognized windows through systematic probing and binary search methods, and ultimately pinpointing the exact sequence and acknowledge numbers required to hijack the TCP/IP connections.

We evaluated our attack in various metrics and demonstrated its practicality and severity via real-world attack scenarios. Specifically, we deployed the attack in two real attack scenarios. The first attack scenario is a TCP hijacking attack over a commercial LTE network. In this scenario, the victim connection is the keep-alive TCP connection of Rich Communication Service (RCS), which allows short messages transmitted over TCP. We successfully launched a hijacking attack and injected a fake short message into the victim user's phone over the hijacked connection. The second attack scenario is a UDP hijacking attack over some real Wi-Fi networks like the coffee shop. In this scenario, we targeted the victim's DNS query communication and hijacked the result of the query. These experiments show that the LenOracle attack is practical and easy to launch, while the consequence is damaging.

We have reported this issue and mitigation recommendations to the standards organizations GSMA and Wi-Fi Alliance, following the responsible disclosure policy. GSMA acknowledged the issue does affect 5G/4G/3G and appreciated our submission. They have notified all its members (operators and vendors worldwide) of our findings and highlighted the mitigation we proposed.

Contributions. In this paper, we make the following contributions:

- *A Novel Attack.* We present LenOracle Attack, a novel and significant threat to encrypted wireless networks, which affects major wireless network protocols, including 5G/4G/3G and Wi-Fi.
- *Real World Evaluations.* We deployed the attack in two real attack scenarios, such as hijacking a TCP connection of RCS over the LTE network and hijacking a UDP communication of DNS over the Wi-Fi network.
- *Responsible Disclosure and Mitigation.* We have submitted our findings to the standards organizations GSMA and Wi-Fi Alliance and received acknowledgments of GSMA. We also propose several mitigation approaches for this attack.

2. Background

2.1. Wireless Network Architecture

Figure 1 illustrates a typical architecture of an IP-based wireless network, which consists of three components: User Equipment (UE), wireless access point (eNodeB or AP), and back-end IP network. The back-end IP network (e.g., the Internet) interconnects devices with the IP protocol. Wireless access point enables the connected UE to access the back-end IP network by radio [1], [15].

We emphasize three characteristics of wireless networks related to our research. Firstly, as the radio frame is broadcast over the air, users are required to have a unique

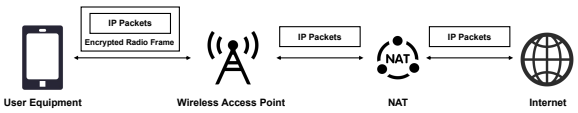

Figure 1. Typical architecture of IP based wireless network

radio ID, such as a Radio Network Temporary Identifier (RNTI) in 5G/4G/3G and MAC address in Wi-Fi. Secondly, encryption and integrity protection are widely deployed to prevent eavesdropping and frame injection. Finally, due to the scarcity of the radio frequency spectrum, stream ciphers are the most popular choice of frame encryption scheme in wireless networks, which does not require additional paddings to packets being transmitted. In practice, both Wi-Fi and 5G/4G/3G networks take *stream ciphers* as the encryption mechanism [2]–[4], [27], [39], [49].

Based on these three characteristics, attackers seeking to obtain the length of the IP data packet merely need to imitate the victim (using its radio ID) to capture the encrypted packets from the air and then calculate the plain-text length by stream-cipher design.

2.2. Network Address Translation

Network Address Translation (NAT) is an indispensable technique in contemporary networking, primarily utilized to alleviate the scarcity of public IPv4 addresses. By translating the private IP addresses of hosts within a local network to a single public IP address, NAT enables multiple devices to access internet resources efficiently. This process not only conserves valuable IPv4 space but also obfuscates the internal network structure from the external internet, thereby enhancing security. Furthermore, the inherent design of NAT allows for significant flexibility in network administration and architecture, facilitating seamless integration and changes within the network without necessitating modifications on the broader internet.

Most people believe that NAT can effectively protect the security of internal nodes, but this is not the case. We found that NAT may act as a filter that accelerates the four-tuple inference. Specifically, the attacker can encode a unique length for each candidate four-tuple and send it. Only the correct four-tuple can be sniffed from the air, as it can be distinguished by its unique length. This is critical for inferring UDP connection parameters. For TCP, since NAT standards [16], [19], [24] do not require verification of sequence numbers in TCP connections, the attacker can also use the same trick to speed up the inference process. We verified this in many Wi-Fi environments. Even for NAT that has enabled sequence number verification, the challenge ACK mechanism can be utilized to infer TCP four-tuple.

2.3. Challenge ACK Mechanism

In order to defend blind in-window attacks, the IETF community proposed a challenge ACK mechanism in RFC 5961 [41]. In short, the challenge ACK mechanism forces the TCP peer to issue a challenge ACK in response to the receipt of a packet that triggers challenge conditions. An

TABLE 1. PREVIOUS HIJACKING ATTACKS OVER THE PAST DECADE VERSUS OUR WORK

Method	Side Channel or Vulnerability	Attack Requirements		Affected Networks		Affected Protocols	
		No Client Auxiliary	No MITM	LTE	Wi-Fi	TCP	UDP
Cao *et al.* [9]	Global challenge ACK rate limit	✓	✓	✓	✓	✓	✗
Feng *et al.* [18]	Shared IPID counter	✓	✓	✓	✓	✓	✗
Chen *et al.* [14]	Wi-Fi timing channel	✗	✓	✗	✓	✓	✗
Vanhoef *et al.* [46]	Key reinstallation vulnerabilities	✓	✗	✗	✓	✓	✓
Rupprecht *et al.* [38]	Lack of integrity protection	✓	✗	✓	✗	✓	✓
Our work	packet length leakage	✓	✓	✓	✓	✓	✓

example of challenge conditions is where the SEQ number falls in the window but is not exactly matched.

Theoretically, this mechanism could prevent blind injection attacks unless the off-path attacker is able to guess the exact same sequence number (with a small probability of $1/2^{32}$). Unfortunately, the challenge ACK mechanism can be used as an auxiliary for protocol state inference. It identifies the incoming packets with the correct state (e.g., SEQ Num, ACK Num) and responds with a challenge ACK. As previous studies described [9], with a side channel that could observe the existence of challenge ACK, adversaries could infer the state of a TCP connection of the victim through the "guess-then-check" method.

We highlight only the necessary details related to our study here. First, the modifications in processing the SYN packets allow off-path adversaries to predict *if a specific TCP connection exists*. As required in RFC 5961, if a receiver sees an incoming SYN packet with four-tuple belonging to an existing connection, regardless of the sequence number, it must send back a challenge ACK to the sender to confirm the loss of the previous connection. Consequently, leveraging a side channel that can observe the challenge ACK, an attacker is able to detect the four-tuple of a victim's TCP connection, such as the source port.

Secondly, the modifications in processing the RST packets allow off-path adversaries to predict *if a guessed sequence number is correct*. The RFC 5961 requires the receiver to simply drop the incoming RST packet with a sequence number outside the valid receive window. However, if the sequence number is in-window but does not exactly match the expected next sequence number (RCV.NXT), the receiver must send back a challenge ACK to the remote peer.

Third, a much smaller valid ACK number range is suggested by RFC 5961 as illustrated in Figure 2, and it allows off-path adversaries to predict *if a guessed acknowledge number is correct*. If a receiver sees an incoming ACK segment, it regards the packet as acceptable only if the ACK number is not too old and not too new, i.e., [SND.UNA - MAX.SND.WND, SDN.NXT]. The remaining ACK values will be in the range of the challenge ACK window, as the specification requires the receiver to perform a challenge ACK to the remote peer.

As these challenge conditions illustrate, the challenge ACK mechanism introduces a valuable tool for adversaries to infer the sequence number and acknowledge number of the connection of victims.

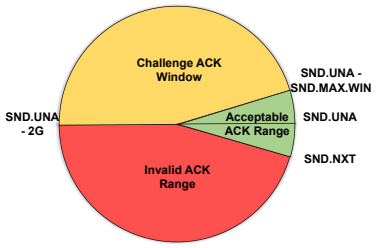

Figure 2. ACK window illustration

2.4. Previous Hijacking Attack

Hijacking attacks, particularly in the realms of TCP/UDP connections, involve the unauthorized interception and modification of data flows between two entities over a network. Previous research found such attacks can occur in both wired and wireless network scenarios, often exploiting vulnerabilities or side channels in certain protocol designs or implementations. We comprehensively compared our work with previous hijacking attacks in Table 1. Notably, our attack does not rely on additional requirements, such as client auxiliary or man-in-the-middle (MITM) model, and is general to typical wireless networks as well as TCP/IP protocols owing to the characteristics of packet length leakage at a ubiquitous low-layer level.

Specifically, many previous works typically have to deploy non-privileged malicious scripts at the victim's device to observe side channels [14], [21], [36] or rely on MITM model to lure the victim to connect to a fake base station beforehand [38], [46]. Only a few work [9], [18] are able to launch completely off-path hijacking attack, however, these attacks usually depend on the implementation flaws in certain protocol, thereby limiting the real-world attack scenarios and easy to be fixed.

3. Overview

3.1. Threat Model

As shown in Figure 3, our threat model concerns six entities: Victim Client, Victim Server, Wireless Access Point, NAT, Radio Sniffer, and Spoofed Server. The victim client (e.g., a smartphone or personal computer) communicates with the victim server via a wireless access point (e.g., eNodeB or AP) and NAT.

We assume an off-path adversary with two capabilities: First, *the attacker can sniff wireless frames with a*

TABLE 2. THREAT MODEL AND ATTACK CONSTRAINT

Side	Constraint	Optional	Description	Workaround
Attacker side	Geographical proximity	✗	Attacker must can sniffer packets from victim.	High-gain antenna allows attackers to sniff from several miles away [17]
	Spoofed server	✗	Attacker must can send spoofed packets from Internet	No workaround, but it is feasible in practice [8]
Victim side	Plain-text communication	✗	Hijacking attacks require plaintext protocol, like RCS [51]	Attackers can launch DoS attacks by resetting connections for encrypted protocols
	Sufficient bandwidth	✗	Bandwidth should more than 1Mbps for TCP, 5Mbps for UDP	No workaround, but it is common in practice(§5.2)
	Specific NAT design	✓	NAT does not check sequence number, common in practice(§5.2)	Attackers can search the NAT's IP and port by challenge ACK mechanism (§4.2)

passive radio sniffer. This radio sniffer can capture the radio frames between the victim client and the access point. This is a basic requirement in wireless network attacks (e.g., [38], [42], [44]–[46]), and a simple way to achieve this assumption is to place the sniffer near the geographical location of the victim's client or access point. *Second, the attacker can send spoofed packets at the IP layer with a spoofed server.* This assumption is still practical today, even though some mitigation against address spoofing has been deployed. According to the latest study [8], at least one-fifth of autonomous systems (ASes) on the Internet do not filter packets with spoofed source addresses.

For victims, those who use plaintext protocols for communication, such as HTTP, SIP and DNS, are susceptible to hijacking attacks. We selected SIP and DNS as targets and demonstrated that hijacking attacks can indeed be successfully carried out on these protocols as mentioned in §5.3. Those who use encrypted protocols, such as SSH, are susceptible to DoS attacks by resetting the connection. Previous work demonstrated that 12% website are still using HTTP [10], [47], and many real-world RCS services based on the SIP protocol lack encryption protection [51]. In addition, the effectiveness of the attack is constrained by the victim's bandwidth. For example, the default DNS timeout for both Windows and Linux is 10 seconds, so networks under 1Mbps UDP bandwidth may experience failures due to timeout issues. We also summarize other optional constraints in Table 2.

The goal of our attack is to *infer the presence of a given TCP/IP connection and execute hijacking attacks over encrypted wireless networks*. This is particularly challenging because TCP/IP involves strict packet validation on incoming packets, including checks for the *connection's four-tuple* (source IP, destination IP, source port, destination port), *sequence (SEQ) number*, and *acknowledge (ACK) number*. Moreover, due to the encryption protection, it's impractical to extract such critical information from the network traffic. Therefore, we decomposed the attack implementation, first using some techniques to probe the client's public IP and then using side-channel attacks to iteratively guess the port, SEQ, and ACK numbers.

3.2. Attack Overview

To achieve this goal, we come up with our approach. The core concept of our approach is to *send spoofed packets and observe whether the spoofed packets have triggered responses or not over radio*. We found while the TCP/IP stack performs strict checks on the incoming packets, in each check the TCP/IP receiver could possibly generate responses depending on the validity of the incoming packet. This enables us to probe and infer

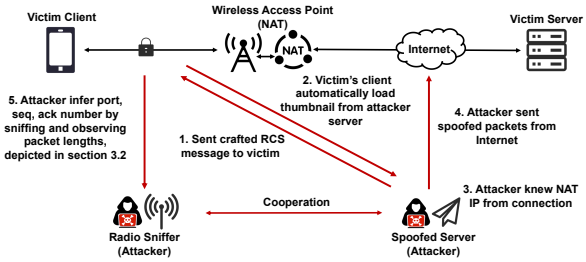

Figure 3. Threat model

the critical information of the targeted TCP/IP connection subtly without directly decrypting the traffic.

Wireless networks widely adopt stream ciphers to protect their transmission. By the nature of stream ciphers, the length of ciphertext is equal to the length of the corresponding plaintext. Therefore, it is feasible to build a side channel that observes the length of IP packets being transmitted over wireless networks protected by stream ciphers (e.g., 5G/4G/3G and Wi-Fi).

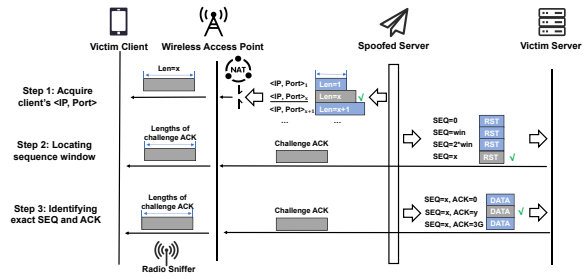

Figure 4. Overview of LenOracle attack

TABLE 3. PROTOCOL WEAKNESS EXPLOITED FOR EACH STEP

Step	Prior knowledge	Protocol weakness exploited	Section
1	Server IP, Port	Packet with correct four-tuple pass through NAT	§4.2
2	-	RST packet with SEQ in sequence window trigger Challenge ACK	§4.3
3	-	DATA packet with ACK in challenge ack window trigger Challenge ACK	§4.4

The attacker can exploit this trivial side channel to infer the presence of a TCP/IP connection and launch a hijacking attack in three steps. Figure 4 and Table 3 illustrate the application of the length-based side channel, and a comprehensive discussion of these steps follows.

Step 1: Detecting connection four-tuple. In this step, the attacker determines if a given connection is established between the client and the server. The key idea is to send spoofed data packets with distinct payload sizes for different four-tuples and observe which packets pass through the NAT using radio sniffers. Of these packets, only those with a four-tuple matching an established connection can traverse the NAT device and subsequently become observable on the radio channel. Consequently, the attacker can

infer the presence of a connection by observing the packet length via radio sniffers. For a given TCP connection, the server IP address and the server port are typically known as they define the target we choose to attack. As the victim client is behind a NAT device, the client source IP is the public IP address of the NAT device. We present four solutions to acquire the NAT's IP address and verified their effectiveness in §4.2. Another challenge is the source port that the NAT uses. The maximum possible port range is [1024, 65535]. We also developed techniques to optimize the search efficacy in §4.2.

Step 2: Locating sequence window. With an active connection's four-tuple identified, the attacker now needs to infer a valid sequence number acceptable by the server. This step takes advantage of two TCP/IP features: (1) per RFC 5961, the TCP receiver generates a challenge ACK packet in response to an RST packet that contains an in-window sequence number that does not match exactly the expected value; (2) the challenge ACK packet has a fixed specific length. Combining these two features, an attacker can send spoofed packets with guessed sequence numbers to trigger a challenge ACK. To improve search efficiency, the attacker can segment the sequence number space into blocks matching the receive window size, probing each block with a hypothesized sequence number to identify those within the receive window, as detailed in §4.3.

Step 3: Identifying acknowledge number and exact sequence number. Once an acceptable sequence number of an active connection has been determined, the attacker must subsequently estimate a valid ACK number deemed acceptable by the server. This step utilizes PSH-ACK packets to provoke the challenge ACK: If a PSH-ACK packet's sequence number is an in-window sequence number and the ACK number falls within the challenge ACK window, the receiver triggers a challenge ACK. Following a similar methodology as with sequence number inference, the attacker can infer whether the guessed ACK number is within the challenge ACK window.

Next, utilizing the characteristics of the PSH-ACK packets, the attacker continues to infer the acceptable ACK number (SND.UNA) and the exact SEQ number (RCV.NXT). Firstly, the attacker fixes the sequence number as an in-window sequence number, then the attacker progressively decreases the ACK number until it reaches the left boundary of the challenge ACK window (SND.UNA - 2G). Secondly, with the ACK number held within the challenge ACK window, the attacker reduces the sequence number gradually until it reaches the left boundary of the receiver's window (RCV.NXT). This process can be optimized using binary search as detailed in §4.4.

4. End-to-End Attacks

In this section, we elaborate on the methodology for executing end-to-end attacks.

Initially, the attacker must observe the encrypted wireless frames transmitted from the wireless access point to the targeted victim client. Further discussion will take place in §4.1.

Subsequently, the attacker will complete the attack by following three steps:

Step 1: the attacker determines whether a given connection has been established between the client and the server by employing connection four-tuple inference, as detailed in §4.2.

Step 2: the attacker infers a sequence number that is located in the server receiver window, discussed further in §4.3.

Step 3: the attacker identifies both the acknowledge number (SND.UNA) and the exact sequence number (RCV.NXT) deemed acceptable by the server, as described in §4.4.

Throughout the execution of the aforementioned attack process, the attacker needs to consider several practical issues, including noise and packet loss, which are addressed in §4.5.

4.1. Preliminary Preparations

To execute the attack, the attacker must first capture encrypted frames transmitted from the wireless access point to the victim client. As described in §2.1, this requires the attacker to obtain the RNTI (for LTE) or the MAC address (for Wi-Fi).

To obtain RNTI, the attacker first maps the phone number to M-TMSI by parsing the paging message [40], and then maps the M-TMSI into RNTI by parsing Random Access Response, as well as Contention Resolution Identity, sniffed during the Random Access Procedure [38]. After this, the attacker could use an open-source LTE project termed srsLTE [23] to obtain encrypted frames.

To obtain the MAC address, the attacker could use the aircrack-ng tool. For example, they can use airodump-ng to identify the Wi-Fi channel, then set the network interface card (NIC) to monitor mode and align it with the detected channel. Subsequently, the attacker collects Wi-Fi frames and analyzes them, using the device's OUI and other information to determine the victim's MAC address, which can then be verified through further steps.

4.2. Detecting connection four-tuple

In this phase, the attacker determines if a given connection is established between the client and the server by probing whether a specific four-tuple (client IP, client port, server IP, server port) is currently active. For a given connection, the IP address and port of the victim server are typically known. For example, Google's DNS server IP and port is '8.8.8.8' and '53'. As the client is behind NAT devices, the victim client's IP is the public IP of NAT, which can be obtained in advance via various techniques.

Acquiring the client public IP address. We discuss four ways to obtain the IP address of the NAT gateway: (1) The attacker can obtain the public IP of the NAT server through the protocol features. For example, by exploiting a feature of the RCS protocol, the attacker can acquire the IP address of the victim's client without interaction [51]. Specifically, the attacker sends an RCS media message containing a crafted media preview link pointing to the attacker's server. Listing 1 shows an example of such an RCS message, where the attacker replaces the `attacker's IP` field with their own in the `url` attribute within the `data` block. Upon receiving the message, the victim's client automatically loads the

thumbnail from the attacker's server. This action causes the victim's client to send a request to the attacker's server, thereby exposing its public IP address. (2) The attacker can also infer the public IP of the NAT server via side channels. In our threat model, the attacker knew the victim's geo-location and the base station the victim connects to, thus, the possible public NAT IP can be enumerated in advance. As the search space of the IP pool under a given base station is limited, inferring the accurate IP address of the victim client by the side channel becomes feasible. The attacker can limit the NAT IP addresses to a manageable list of candidate IPs, denoted as $\{\text{IP-Guess}_m \mid 1 \leq m \leq M\}$. In the subsequent port inference process, packets are sent to various ports of these IP addresses. By observing the channel and determining for which specific IPs the packets can be detected in the channel, the true IP can be identified. (3) The attacker can also lure the victim to visit a website to collect the public IP from NAT in advance. (4) If the attacker and the victim are on the same Wi-Fi network, the attacker can easily obtain the public IP using ICMP messages with the record route field. We also conducted experiments to verify the effectiveness of the four approaches. For the first approach, we tested popular RCS-enabled phones (Xiaomi, Redmi, ZTE, Samsung) in real-world operators and confirmed that the NAT IP can be revealed without victim interaction. For the second, third, and fourth approaches, we verified their effectiveness across 22 tested wireless routers in Table 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<file xmlns="urn:gsma:params:xml:ns:rcs:rcs:fthttp">
  <file-info type="thumbnail">
    <file-size>9397</file-size>
    <content-type>image/jpeg</content-type>
    <data url="http://<attacker'IP>:<Port>/Thumbnail
        /b0446a26be40434cb3b70a77364447d6DT.jpeg"
        until="2025-02-22T00:09:09.000Z"/>
  </file-info>
  <file-info type="file" file-disposition="
      attachment">
    <file-size>79074</file-size>
    <file-name>IMG_6863.JPG</file-name>
    <content-type>image/jpg</content-type>
    <data url="http://<attacker'IP>:<Port>/File/
        b0446a26be40434cb3b70a77364447d6DF.JPG"
        until="2025-02-22T00:09:09.000Z"/>
 </file-info>
</file>
```

Listing 1. RCS messages payload example

Detecting client public port. The major challenge of connection four-tuple inference is the source port the NAT uses. The maximum possible port range is 2^{16} = 65536. However, in different OS implementations, the port range is often narrower. For example, the port range in Linux ranges from 32,768 to 61,000. Furthermore, since NAT often adopts the port reservation strategy [52], the search range remains consistent. For port inference, we assume that the entire port scanning range is defined as $\{\text{Port-Guess}_n \mid 1 \leq n \leq N\}$. The attacker impersonates the victim server and sends packets to the NAT IP, with the destination port set to Port-Guess_n, each carrying a payload of length n. Due to the presence of NAT, only packets with the correct port will be forwarded to the victim client and can be sniffed over the wireless channel. The attacker can then use the length-based reverse lookup to identify which specific port is open. Figure 5 show

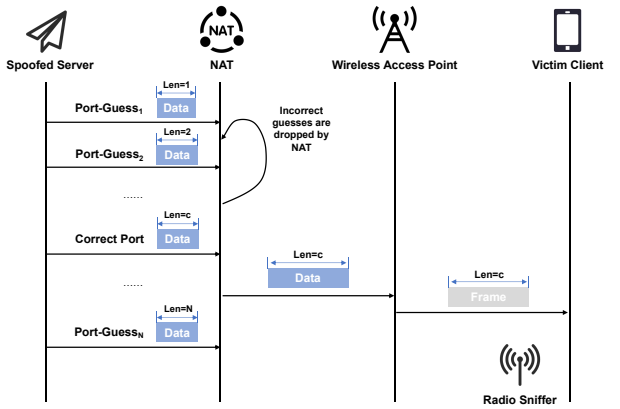

Figure 5. Detecting client public port without considering MTU size. The correct port is Port-Guess_c

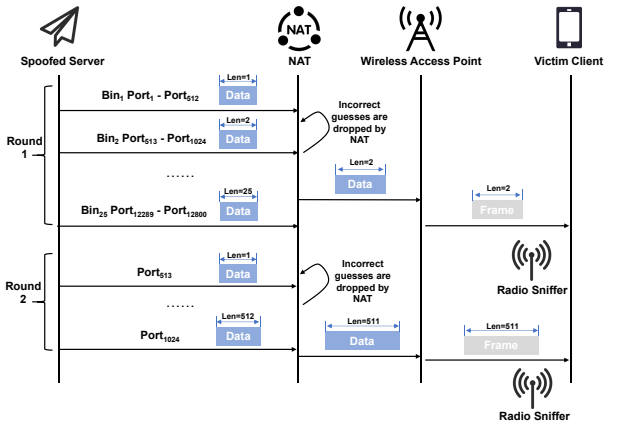

Figure 6. Detecting client public port with considering MTU size and bandwidth about 5Mbps. The correct port is Port_{1023}

about this procedure, and Port-Guess_c is correct port.

However, in real-world attacks, the length of a single IP packet is constrained by the MTU (Maximum Transmission Unit) and bandwidth, so the attacker cannot assign a unique length for each guessing attempt. To overcome this issue, as shown in Figure 6, the attacker can uniformly divide different guessing attempts into a number of BINs and assign a different length payload to each BIN. The attacker can first infer which BIN the port is in and then determine which port in the BIN it is.

The time consumed by the attacker's inference is closely related to the bandwidth. Assuming the bandwidth is X (Byte per second), and the probing time for each round is t_{round} (Second). This time includes the time to transmit the packets and the time to wait for the sniffer to respond. Assuming B Bins are divided, and b Bins are probed per round, we could know the time t_N of inferring entire port searching space N is calculated as:

$$t_N = \left\lceil \frac{B}{b} \right\rceil \cdot t_{\text{round}} + t_{\text{N/B}} \tag{1}$$

In this formula, the preceding $\lceil \frac{B}{b} \rceil$ represents the time spent determining which BIN the Port is in, while $t_{\text{N/B}}$ represents the time spent locating which element in the BIN it is. It can be seen that when b is as large as possible

and B is as small as possible, the total time will reach its minimum.

It should be noted that if B is chosen too small, $t_{\text{N/B}}$ may be too long as shown in 2. Since we don't want to search the N/B port range in Bins, we try to complete this phase within one round, which means that each port can be length-encoded and sent in one round.

$$t_{\text{N/B}} = \sum_{i=1}^{N/B} (i+H)/X \propto (\frac{1}{B}) \tag{2}$$

Moreover, due to the influence of bandwidth and MTU, B and b have the following limitations. This means that each round of packet sending needs to be less than the bandwidth limit as formula 3, and the number of ports in each group needs to be less than the MTU as formula 4. Only in this way can a port be bound to a specific length. These two formulas show that B cannot be too small and b cannot be too large.

$$\sum_{i=1}^{b} (i+H) \cdot \frac{N}{B} \leq X \cdot t_{\text{round}} \tag{3}$$

$$\frac{N}{B} \leq MTU - H \tag{4}$$

In these formulas, we use the following notations: The header of the IP datagram is denoted as H. For TCP, H equals 40 (Byte), and for UDP, H equals 28 (Byte). Additionally, we have standardized the units of measurement.

Therefore, the attacker could first get the range of values for B according to formulas 4, then substitute it into formulas 3 to find the maximum b. For example, under 5Mbps bandwidth, the attacker can adopt the following scheme: dividing this 2^{15} port range into 64 BINs, each containing 512 ports. The attacker probes 25 sets per second (sum(range(40, 64)) * 512 * 8 = 4.8 Mbps). Therefore, it would only take at most 3s to identify the BIN they are in and then 1s to determine the exact port, just like Figure 6.

However, this rapid inference strategy will fail against NAT that verifies sequence numbers, such as some in LTE networks. Therefore, we propose a more covert method of inference that leverages the SYN-ACK message in the challenge ACK mechanism to determine if the four-tuple exists. The specific process is illustrated in Figure 7. The attacker, impersonating a possible four-tuple, transmits a SYN-ACK packet to the server. A response from the server in the form of a challenge ACK confirms the correctness of the four-tuple inference; conversely, if the response is an RST packet, which differs in length from the challenge ACK, the attacker would ascertain the incorrectness of this attempt. By systematically examining the entire set of candidate tuples, the attacker is ultimately able to deduce the accurate four-tuple inference.

4.3. Locating sequence window

As described in §3.2, the attacker should first pinpoint a Sequence Number (SEQ Num) that falls within the receiver's window. By leveraging the challenge ACK mechanism, only if RST packet's SEQ Num falls within

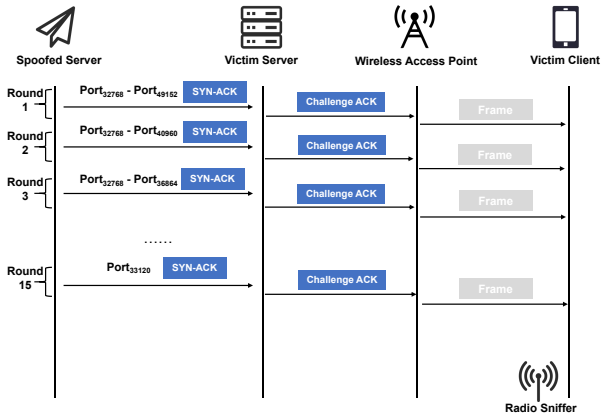

Figure 7. Detecting client public port if NAT verifies sequence number. The correct port is Port_{33120}

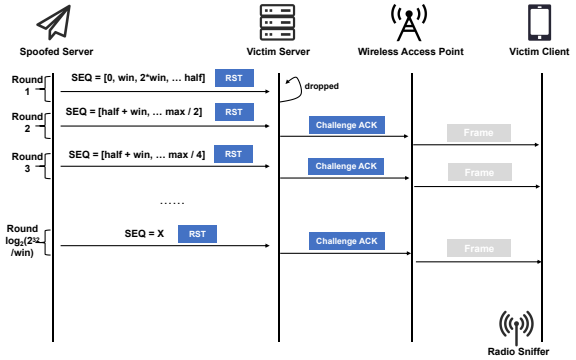

Figure 8. Locating sequence window. SEQ_{inw} is X

the receiver's window, the receiver will send the ACK packet to the sender. Therefore, the attacker can divide the entire space into blocks according to the window size.

So the initial search space for SEQ Num is [0, win, win*2, ...], where 'win' is the size of the TCP receiver window. As shown in Figure 8, the attacker sends TCP RST packets with SEQ numbers from the initial search space. If the server receives a TCP RST packet containing a SEQ number within the window, it will respond with a challenge ACK packet to the client. The attacker sniffs for challenge ACKs through the length side channel and identifies the search space for the next round. The attacker can find the SEQ number within the window after $log(2^{32}/win)$ rounds of binary search. For the convenience of subsequent explanation, this SEQ Num records as SEQ_{inw}.

4.4. Identifying acknowledge number and exact sequence number

After obtaining SEQ_{inw}, the next step is to locate an acknowledge Number (ACK Num) that falls within the challenge ACK window. Finally, the attacker needs to conjecture the left-side boundaries of the receiver's window and challenge ACK window to determine the exact SEQ Num (RCV.NXT) and acceptable ACK Num (SND.UNA) that the server side will acknowledge. It should be noted that the attacker uses the PSH-ACK packets, that its SEQ Num falls within the receiver's window and its ACK

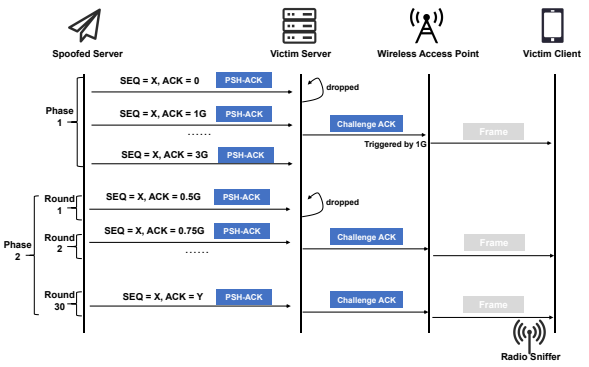

Figure 9. Identifying acknowledge number. SND.UNA - 2G is Y

Num falls within the challenge ack window, to trigger the challenge ACK packet.

Locating challenge ACK window. According to RFC 1323 [7], the maximum receiver's window size can be expanded from 2^{16} to a maximum of 2^{30} = 1G using the window scaling option. Therefore, the MAX.SND.WND cannot exceed 1G. Consequently, the challenge ACK window size is between 1G - 2G, which is a quarter of the entire ACK space. Therefore, the attacker divides the entire ACK space into four blocks and probes each block to determine in which block the challenge ACK window lies. In our implementation, the attacker sequentially probes the first value of each bucket, namely 0, 1G, 2G, and 3G, until the attacker detects at least one that triggers a challenge ACK packet. This process is mentioned in Figure 9 Phase 1, and this ACK Num records as ACK_{inw}.

Identifying acceptable acknowledge number. Attacker sets the ACK_{inw} as the right boundary, and the left boundary is chosen as ACK_{inw} - G. Specifically if 0 triggers a challenge ACK, we then need to probe 3G to see if it triggers a challenge ACK. If it doesn't, then 3G is chosen as the left boundary; otherwise, the attacker probes between 2G and 3G. Next, utilizing the characteristic that a challenge ACK is only triggered within the challenge ACK Window, the attacker conducts a binary search within the set boundaries to find the boundary value SND.UNA - 2G. Specifically, the attacker sends ACK = $\frac{left+right}{2}$ every round and determines the next boundary situation based on whether there is a challenge ack triggered until the left boundary is equal to the right boundary, as mentioned in Figure 9. This value + 2G is the SND.UNA attacker needs to solve for.

Identifying exact sequence number. This process is very similar to acceptable ACK Num inference, with the key difference being that this time we fix the ACK Num and perform a binary search for the Seq Num by PSH-ACK packets. We select SEQ_{inw} as the right boundary and SEQ_{inw} - win as the left boundary. The ACK Number is set as ACK_{inw} and fixed. Through a similar binary search, we can determine the left boundary of the receiver's window, which corresponds to the next SEQ number the server accepted (RCV.NXT).

4.5. Practical Attack Consideration

The length-based side channel exhibits limited robustness, being susceptible to two primary factors: noise and packet loss. The former refers to the routine communication data of the victim device, where its length may influence the adversary's assessment. The latter is associated with the spoofed server and the radio sniffer. We provide some effective measures for these two points.

4.5.1. Countermeasures to Noise. To confront the noise of the side channel, we propose two simple yet effective strategies that are suitable in different cases. The first strategy is repeating, the idea of which is to repeat packets of a single round during the guess phase. In the checking phase, the side channel will report several packets with the same length in the case of the correct guess. This strategy is robust against noise as it is unlikely that noise packets of the same length will occur simultaneously many times.

The drawback of this strategy is that it does not work for challenge ACK based TCP hijacking on the state-of-the-art Linux kernel, as the kernel limits the rate of challenge ACK to one packet per 500ms. Resend and confirm is designed to complement this strategy. To make sure a guess is correct, the attacker resends the payload several times. This approach works because the probability of noise packets with the same length occurring continuously is very low.

4.5.2. Countermeasures to Packet Loss. To reliably launch the attack, we should consider two sources of packet loss: (1) packet loss of the spoofed server; and (2) packet loss of the radio sniffer.

Due to the turbulence of the Internet, packet loss of spoofed servers is sometimes unavoidable. The loss of the guessed packet causes the side channel to miss the correct guess, which will finally fail the attack. A simple way to deal with this issue is to repeat the packets from different locations. It is unlikely that all repeated packets will be lost simultaneously.

On the other hand, packet loss of the radio sniffer is usually caused by implementation issues. It can be subverted by optimizing the implementation of the sniffer, which is beyond the scope of this paper. Therefore, we choose to apply some tricks against such a kind of packet loss in our evaluation. For TCP hijacking, we send spoofed packets with both left and right sections of binary search to detect the packet loss. If both left and right sections trigger no challenge ACK, it can be inferred that a packet loss has happened to the sniffer. In such a case, we will retry the current round. For Wi-Fi scenarios, we further decrease the packet loss rate by increasing the number of network interface cards.

5. Evaluation

In this section, we conduct an evaluation. Initially, we delineate the experimental setup, encompassing the devices and network utilized (§5.1). Subsequently, a local experiment is carried out to assess the practicability of the attack, with particular emphasis on the influence of bandwidth and the type of network equipment on the attack (§5.2). Finally, the attack is deployed in real-world scenarios: executing the fake short message injection (TCP) within a commercial LTE environment, and performing the DNS cache poisoning attack (UDP) over public coffee shop Wi-Fi. The discussion further extends

Figure 10. Devices for evaluation

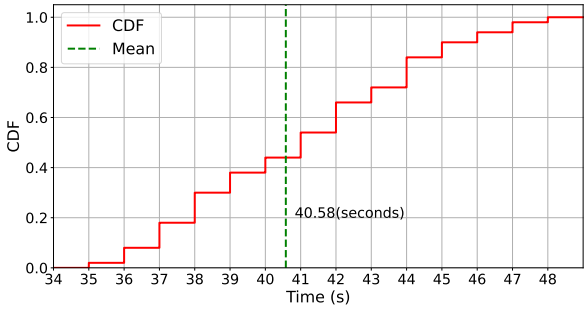

Figure 11. Time cost of TCP parameters inference

to the implications of noise and packet loss on the efficacy of the attacks (§5.3).

5.1. Experiments Setup

This section will present the device and network utilized in our experiment. Figure 10 shows several devices employed during our evaluation. We chose a Xiaomi 10 as the victim client. To closely simulate real-world conditions, we installed 132 applications on the victim device, including the top 30 free applications from Google Play. We procured a remote server on Vultr, operating on Ubuntu 22.04, to fulfill the role of the victim server. This server, configured with OpenSSH 8.2 and OpenSSL 1.1.1, served as a remote SSH service node to deliver TCP services for our local experiments. It was also configured with BIND 9.18.28, which set the response rate limit as 1 per second to function as a DNS service node, facilitating DNS services for local and real-world experimental settings.

We need the radio sniffer and the spoof server to serve as the attacker. For the former, we used the srsLTE framework [23] along with a USRP X310 to monitor LTE environments, and used three Wi-Fi adapters for the Wi-Fi scenario. For the latter, we rent several servers equipped with spoofing abilities to send spoofed packets in real-world environments.

For the network used in the experiment, we selected multiple routers for the local experiment and selected a commercial LTE network and a coffee shop Wi-Fi network for the real experiment. We carefully consider the ethical issues when deploying the attack in real-world scenarios and discuss them in §6.2.

5.2. Local Experiments

In this section, we conduct our experiments within the local network. Specifically, we evaluate the following three aspects: (1) The *time* required for the attack; (2) How *bandwidth* impacts the efficiency of the attack; (3) How router models impact on our attacks.

5.2.1. Time cost. We evaluate using the devices described in §5.1. For TCP connections, the victim establishes an SSH connection with the remote server, and the attacker needs to infer the exposed port, the sequence, and the acknowledge numbers of this SSH connection followed by three steps as mentioned in §4. For UDP connections, the victim sends a DNS query to the remote server and the attacker needs to infer the exposed port before this query times out. For each scenario, we conducted fifty rounds for both TCP and UDP experiments and the bandwidth of the attacker was under 5Mbps.

The initial step involves port inference, wherein the multi-bin search methodology delineated in §4.2 is employed. Regarding TCP connections, the port space can be partitioned into 147 bins, with 49 bins searched per round. It requires at most three iterations to determine the bin containing the target port, followed by a single iteration to identify the specific port. For UDP connections, the port space can be divided into 62 bins, allowing for 31 bins in each iteration. Identifying the precise port requires no more than three iterations. To mitigate the risk of packet loss resulting from channel congestion, the timeout for each round is regulated to 2 seconds.

For Step 2 and Step 3, we use the scheme mentioned in §4.3 and §4.4 to identify the acceptable acknowledge number and the exact sequence number. We calculated the average bandwidth of 50 experiments and found that only 0.329Mbps is needed.

Cumulative distribution function (CDF) graphs were plotted for both TCP and UDP experiments, as mentioned in Figure 11 and Figure 12. Analysis of these graphs indicates that the mean parameter inference time cost is 40.58 seconds for TCP and 3.55 seconds for UDP. It's important to highlight that the timeout of a TCP long-lived connection is much longer than these 40.58 seconds, as mentioned in previous work [34]. Regarding UDP, particularly DNS, our observations revealed that a DNS query generally undergoes four to five retries, all originating from the same port. In our experiments, the attacker could execute the inference within the constraints of a query timeout using a bandwidth of 5Mbps.

Summary. Our attack can be executed within a realistic time cost. Under the 5Mbps bandwidth, TCP parameters inference was completed 40.58 seconds on average, while UDP parameters inference was achieved in 3.55 seconds on average.

5.2.2. Bandwidth impact on time cost. According to §4.2, the multi-bin methods depend on the bandwidth. Increasing bandwidth greatly reduces inference time. So we analyze various bandwidth levels—0.5Mbps, 1 Mbps, 2 Mbps, 3 Mbps, 5 Mbps, and 8 Mbps—to assess the time cost of port inference. For each bandwidth, we conducted ten tests and calculated the mean.

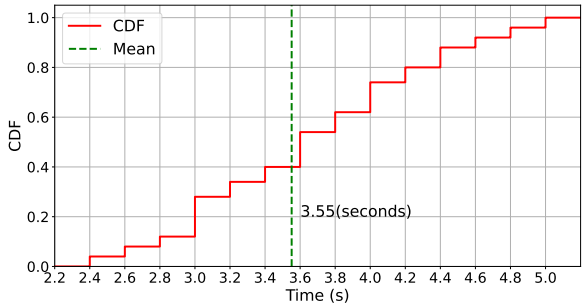

Figure 12. Time cost of UDP parameters inference

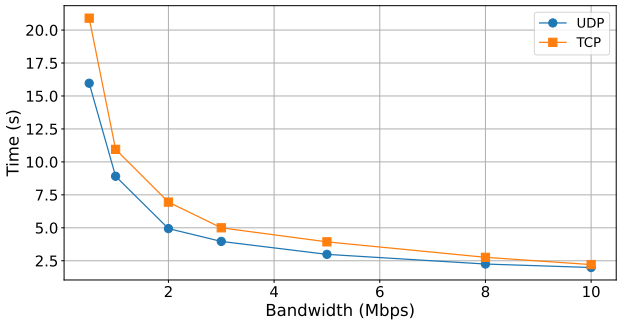

Figure 13. Impact of bandwidth on port inference time cost

We plotted the experimental results in Figure 13. It is worth noting that for bandwidths of 0.5Mbps and 1Mbps, the attacker can only barely infer the port number (because the NAT mapping table for this item has not been deleted) and has almost no time to complete the injection of the DNS message. However, when the bandwidth is greater than 2Mbps, the attacker can infer and complete the message injection in time. And with the increase in bandwidth, port inference can be completed within a few seconds.

The experimental results are illustrated in Figure 13. It is noteworthy that at bandwidths of 0.5Mbps and 1Mbps in UDP scenario, the adversary is only marginally known of the port number, as the NAT mapping table for this particular item remains. Consequently, the adversary has insufficient time to finalize the injection of the DNS message. In contrast, when the bandwidth exceeds 2Mbps, the adversary can infer and execute the message injection on time. Moreover, with increasing bandwidth, port inference can be accomplished within a matter of seconds. Compared to previous works [14], [18] that rely on the challenge ACK mechanism of SYN packets for port inference, our attack method utilizes bandwidth more efficiently and accomplishes port inference by less time.

Summary. Our findings show that enhancements in bandwidth yield a considerable reduction in time expenditure, indicating our attack has better performance in modern networks with large bandwidth, e.g., 100Mbps.

5.2.3. Wi-Fi Router Evaluation. To encompass a comprehensive spectrum of router models, we conducted tests on an assortment of vendors using the default factory settings within a controlled local environment. The vendors assessed include Xiaomi, TP-Link, Huawei, Mercury, Asus, Tenda, Ruijie, H3C, 360, and ZTE. The data obtained from these evaluations is systematically presented

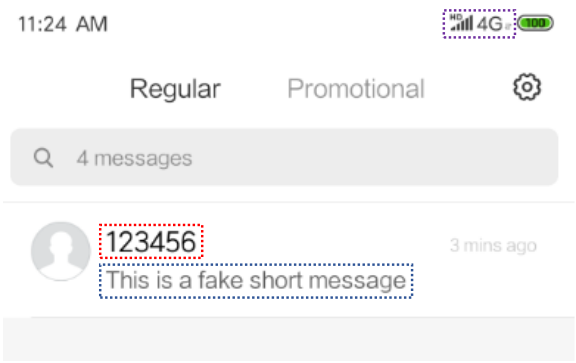

Figure 14. Demonstration of fake short message injection. The field with red border is faked sender "123456", the field with blue border is faked SMS content "This is a fake short message". The whole attack is accomplished over an LTE network.

in Table 4. Our primary concern lies in whether these routers have enabled reverse path validation and TCP window tracking. The activation of the former necessitates that an attacker introduce an external spoofed server, whereas the latter requires the employment of SYN-ACK to deduce the TCP connection port. Notably, among the 22 routers examined across 10 distinct vendors, merely 5 devices had reverse path lookup enabled, and none provided support for window checking. We used a similar approach to conduct TCP and UDP parameters inference on these routers and found that they are all vulnerable. This phenomenon is easy to explain because the core reason for our attack is due to the selection of the stream cipher, rather than the specific router model.

Summary. Our findings indicate that our attack is independent of Wi-Fi routers. The principal issue lies in the wireless network's decision to employ stream cipher encryption, rather than in any other design factors.

5.3. Real-World Experiments

In our testing of real-world scenarios, we investigated two cases: LTE and Wi-Fi. For the LTE scenario, a commercial LTE network was selected, within which we attempted to transmit a fake RCS message, an application-layer protocol operating over TCP, to the mobile device. With regard to the Wi-Fi scenario, our focus was on typical environments, including a coffee shop, a hotel, and a campus network. In these contexts, we endeavored to inject a DNS response into the device. Considering ethics, the victim device was restricted to our personal devices, and prior notifications were provided to the network managers of the coffee shop, hotel, and campus.

5.3.1. Fake Short Message Injection. In this scenario, we aimed to hijack the connection of Rich Communication Services (RCS) to inject a spoofed short message into the victim client. With this technique, the attacker can send a fake message that looks like one sent by the authority and requires the victim to install a malicious application. If the victim user trusts the spoofed message and installs the application, the attacker could possibly compromise the phone and steal secrets. We implemented this scenario over a commercial LTE network. In the scenario, both

TABLE 4. TESTED WIRELESS ROUTERS LIST

Router Model	Vendor	Generation	WPA	Wi-Fi Channel	Reverse Path Validation Disabled	TCP Window Tracking Disabled	Vulnerable
Redmi RA81	Xiaomi	Wi-Fi 6	WPA2/WPA3	6, 40	✓	✓	✓
Redmi RM2100	Xiaomi	Wi-Fi 5	WPA2	3, 161	✓	✓	✓
Mi 4C	Xiaomi	Wi-Fi 4	WPA2	6	✓	✓	✓
TL-XDR6020	TP-Link	Wi-Fi 6	WPA2/WPA3	153	✓	✓	✓
TL-WAR1200L	TP-Link	Wi-Fi 5	WPA2	52	✓	✓	✓
TL-WDR7620	TP-Link	Wi-Fi 5	WPA2	44	✗	✓	✓
TL-WR886N	TP-Link	Wi-Fi 4	WPA2	11	✗	✓	✓
AX3 Pro	Huawei	Wi-Fi 6	WPA2/WPA3	149	✓	✓	✓
TC7001	Huawei	Wi-Fi 6	WPA2/WPA3	149	✓	✓	✓
TC7102	Huawei	Wi-Fi 5	WPA2	44	✓	✓	✓
WS5200	Huawei	Wi-Fi 5	WPA2	40	✓	✓	✓
D196G	Mercury	Wi-Fi 5	WPA2	11, 153	✓	✓	✓
MW300R	Mercury	Wi-Fi 4	WPA2	11	✗	✓	✓
RT-AX57	Asus	Wi-Fi 6	WPA2/WPA3	161	✗	✓	✓
W30E	Tenda	Wi-Fi 6	WPA2/WPA3	44, 161	✓	✓	✓
EM12	Tenda	Wi-Fi 6	WPA2/WPA3	149	✓	✓	✓
X32pro	Ruijie	Wi-Fi 6	WPA2	44, 153	✓	✓	✓
R100	H3C	Wi-Fi 5	WPA2	40	✓	✓	✓
R300G	H3C	Wi-Fi 5	WPA2	44	✓	✓	✓
T7	360	Wi-Fi 6	WPA2/WPA3	40	✓	✓	✓
C301	360	Wi-Fi 5	WPA2	149	✓	✓	✓
E2631	ZTE	Wi-Fi 6	WPA2/WPA3	153	✗	✓	✓

the RCS server and the spoofing server are located on the Internet. We used iptables to redirect all traffic to our server for ethical considerations. We craft an SIP message that encapsulates a fake short message. Targeted to the keep-alive TCP connection of RCS, we infer the source port, SEQ number, and ACK number by the attack we propose. With inferred information, we send the crafted SIP message into the hijacked connection. As shown in Figure 14, the device accepts the packet and notifies the victim user that a short message has been received in the default SMS application.

We repeated the experiments ten times, successfully executing nine within 5 minutes at a bandwidth of 0.2 Mbps using the covert strategy mentioned in §4.2. The time CDF distribution is shown in Figure 15. It is worth mentioning that background noise in our experiments can arise from two sources: (1) traffic from other clients connected to the same network and (2) traffic generated by other apps on the victim's device. For the former, the attacker can isolate and filter the victim's traffic in multi-client environments because each device on the network is assigned a unique Radio ID. For the latter, during our experiments, the victim device, with popular apps installed and logged in, was kept in a screen-off state. We performed continuous packet capture on the client and observed that the channel was mostly silent.

Based on our experimental log, we uncovered factors that affect the time cost of the attack. The time consumption of an attack increases with the noise rate, as the noise triggers our strategy to perform extra inference rounds. We also analyze the reason for the failed attempt. By inspecting the log, we discover that the reason is packet loss of the radio sniffer (maybe caused by excessive background traffic, which causes the real message to be

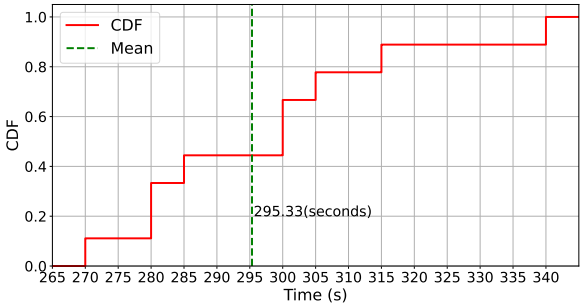

Figure 15. CDF of time cost for fake short message injection over LTE network

overwhelmed). The strategy against noise works perfectly during the evaluation, but the countermeasure against packet loss sometimes does not work well. Fortunately, we can suppress the packet loss by implementation optimization, like improving LTE radio sniffer capabilities.

The real-world deployment of RCS messages often lack integrity or confidentiality protection, rendering them vulnerable to manipulation. In our study, we measured three major mobile operators, each with hundreds of millions of subscribers. We also tested six popular 5G messaging-enabled mobile phones. Our findings show that all RCS traffic from these devices, when connected to the aforementioned operators via cellular networks, is transmitted in plain text. These devices directly establish SIP channels with the operators' servers without mechanisms for ensuring integrity or confidentiality. The design rely on encrypted wireless networks for protection, which can prevent traditional man-in-the-middle (MITM) attacks, but it still leaves room for potential off-path hijacking attacks.

Summary. The attacker can leverage the length side channel to infer the TCP connection four-tuple, sequence number, and acknowledge number, thereby enabling the injection of fake messages into commercial mobile networks within 5 minutes.

5.3.2. DNS Poisoning Attack. In this attack scenario, we aimed to conduct a DNS poisoning attack to demonstrate UDP connection hijacking. Accordingly, we conducted our tests in three typical Wi-Fi networks: a coffee shop, a hotel, and a campus network.

To launch the attack, we first trigger the victim device to send a DNS query by a zero-click method described in [51]: The attacker sends an RCS message with a crafted media preview link to the victim device like Listing 1 which change code IP to what attacker want to DNS poisoning. Once the victim device accepts that message, a DNS query will be automatically sent out. After the victim device sends out the DNS query, we extend the attack window through the method (by RRL) described in [31] and perform connection inference. We inject a fake DNS response to the victim with all possible transaction IDs. The injected response with the correct transaction ID is accepted and cached by the client.

We conducted ten trials at each location. In the case of the campus network and the coffee shop, we achieved success in seven and five trials respectively, due to packet loss caused by the radio sniffer. This phenomenon can be ascribed to the packets sent by the attacker being obscured within the background traffic generated by the substantial number of users connected to the Wi-Fi network. Importantly, throughout the attack, the mobile phone remained silent, while in the screen-off state, thus not interfering with the execution of the attack. At the hotel, all trials were conducted successfully.

Summary. The attacker can leverage the length side channel to infer the source port of the DNS query, thus enabling the injection of DNS responses with all transaction id to the victim device before the query timeout.

6. Mitigation and Discussion

6.1. Mitigation

Mitigation in wireless protocol. One key element of the attack is the leakage of the IP packet. Therefore, hiding the packet length is one of the approaches to mitigate the vulnerability. A straightforward idea to implement this approach is to apply random padding to stream cipher or use block cipher. However, padding is fundamentally equivalent to adding noise to the side channel. The idea won't prevent the attack if the attackers apply the "wait for idle" strategy we have proposed.

Alternatively, We propose a new approach termed onetime pad radio ID, which stops attackers from filtering out the radio frame of the victim. Specifically, a one-time pad radio ID is maintained between UE and the access point. The one-time pad radio ID rolls ahead for each user plane communication. With this mitigation, attackers cannot track the communication of the specific user. Consequently, all the traffic for this access point is potentially to be the noise of the side channel, in which case launching LenOracle Attack will be almost impossible.

During the attack, the NAT acts as a filter for attackers by discarding incorrect packets. Ideally, the NAT should verify TCP sequence and acknowledge numbers, allowing only completely correct packets to pass through. Although this would increase the NAT's workload, considering the advancements in computing capabilities, such additional load should not be a significant concern.

Mitigation in network deployment. A server that is capable of sending spoofed IP packets is one of the requirements of the attack. For this, blocking spoofed IP packets from being sent or received is another aspect of mitigation. Blocking spoofed IP packets from being sent out has been popularized and applied to ASes for several years. However, as described in [30], about a quarter of ASes still did not block packets with spoofed source IP addresses by 2019. What's worse, the attack is still alive even in the case where only one AS can send spoofed IP packets.

On the other hand, the operator can deploy policies to prevent spoofed IP packets from being received by the victim. For example, A operator can deploy a policy that prevent all packets with a source address that belongs to an AS from entering to that AS.

Mitigation in application layer. Encryption at the application layer is yet another effective measure to prevent data injection attacks. For example, the RCS service provides an option of encrypting SIP payloads using TLS. As for DNS, there are projects like DNSSEC [37] and DNSoverTLS [25]. Attackers cannot achieve the RCS injecting attack and the DNS poisoning attack if these options are enabled. However, this mitigation won't alleviate the Transport layer attacks such as connection presence inference. On the other hand, this mitigation is not always practical in real world scenarios. There are plenty of plaintext applications and protocols in cellular networks, primarily due to the cost of encryption.

6.2. Discussion

Ethical consideration. We referenced existing papers [33], [54] on security testing of cellular networks, as well as authoritative guidance such as the Menlo Report [6], to design an ethically compliant experiment. We avoided interfering in the commercial network during our implementation and evaluation. For RCS hijacking, we use lower bandwidth of 0.2 Mbps. For DNS hijacking, we use a self-deployed DNS server as the target to avoid hindering public DNS servers.

Responsible disclosure. We follow the responsible disclosure policy and have reported our findings to GSMA and Wi-Fi Alliance. GSMA appreciated our submission and confirmed the issue does affect 5G/4G/3G. GSMA notified all its members (operators and vendors worldwide) of the issue and highlighted our recommendations on IP spoofing protection and application layer encryption in the notification. We have yet to receive a response from Wi-Fi Alliance.

LenOracle attack for block cipher. Unlike stream cipher, block cipher applies padding to the plaintext before the encryption. As a result, we can only obtain a padded length of plaintext for a given ciphertext. The consequence

of the inaccurate length is the increased noise of the side channel, as attackers cannot distinguish packets with the same padded length. To overcome this, attackers has to apply strategies that subvert the consequence caused by the noise. In addition, since attackers has to send packets only with MTU/block size different sizes per round, the efficiency of NAT-based connection inference will significantly decrease.

In summary, the padding operations in block cipher increase the time consumption of a successful exploit and affect the attack that requires a short exploit time (e.g., DNS hijacking). We regard the evaluation of LenOracle Attack in such a situation as our focus of work in the future.

7. Related Work

Wireless network security. The field of wireless network security has been a popular research topic. The research is mainly divided into two levels.

On the one hand, attackers are focusing on passive attacks to violate user privacy. As indicated in references [5], [28], deep learning methods are used to analyze the type of video represented in encrypted traffic, allowing attackers to determine the content being watched by the victim. Moreover, attackers have been able to pinpoint the victim's location by sniffing wireless frame traffic, as noted in [26].

On the other hand, active attacks are often characterized by the utilization of counterfeit base stations, or malicious relays, or by the overshadowing of signals to take advantage of the inherent vulnerabilities in the broadcast nature of communications. For example, as referenced in [42]–[46], attackers have exploited design flaws in protocols and the malicious use of relays to break the encryption of WPA, WPA2, and WPA3. Relating to LTE networks, active attacks have been executed using fake base stations [38] and signal overshadowing [50].

In contrast, our attack performs packet injection by exploiting the inherent features existing in IP-based wireless networks, which are more prevalent. In addition, our attack only relies on a passive radio sniffer, without fake base stations or signal inferences.

Off-path TCP hijacking attacks. For the past decade, significant efforts have been devoted to uncovering security vulnerabilities in TCP connections. Researchers have proposed various approaches to infer the TCP SEQ and ACK numbers thus hijacking connections to inject crafted malicious content. However, previous works on off-path TCP sequence number inference rely heavily on exploiting the side channel of the operating system [9], [18], [20], [22], or executing an unprivileged malicious script on the client side [14], [35], [36].

As for the roadmaps of these attacks, they all fall under the same scheme of "guess-then-check". These side-channel attacks utilize some critical information observable by the off-path attackers or the malicious script hosted on the client side. The most common exploits of side channels include the rate limit of global challenge ACK [9], the system shared packet counter [35], [36] and IPID counter [18], [20], [22]. In general, these vulnerabilities are mainly caused by incorrect implementations of specific operating systems and can be fixed by software updates. In recent work [14], by exploiting the timing side channel introduced by the half-duplex nature of IEEE 802.11, an attacker can inject malicious data into an HTTP session. Due to the protocol design, timing side channel attacks are extremely hard to mitigate, but it requires the cooperation of a client-side sandboxed malicious script. Prior studies have shown that even off-path adversaries can mount powerful attacks by exploiting vulnerabilities at the application layer [11]–[13], [29], [32], [48], [53], while our work operates independently of application logic and targets fundamental transport-layer behavior — making it more widely applicable.

Our attack exploits inherent features of wireless networks and cryptosystems, it is hard by nature to mitigate through software updates. Moreover, our attack only requires an additional sniffer device, which is a more lenient prerequisite compared to unprivileged malware or sandboxed script execution clients. Additionally, compared to previous works, our attack can more rapidly complete the guessing of four-tuples. Lastly, our attack affects the security of not only TCP connections but also UDP.

8. Conclusion

In this paper, we presented LenOracle Attack, which leverages the length of IP packets as a side channel to hijack TCP/IP communication across wireless networks (e.g., 5G/4G/3G and Wi-Fi). We conducted a comprehensive evaluation of this attack using various metrics and implemented a TCP hijacking attack on the LTE network and a UDP hijacking attack on the Wi-Fi network. The hijacking attack consequently caused a faked short message to be injected to the victim under LTE, and DNS hijacking for the victim using Wi-Fi. Our research demonstrated that this novel hijacking technique poses a threat to a wide array of IP-based wireless networks, including 5G, 4G, 3G, and Wi-Fi. Lastly, we proposed potential defense mechanisms against this attack.

Data Availability

The implementation of LenOracle is open source at https://github.com/zmh02/LenOracle.

Acknowledgement

We sincerely thank all anonymous reviewers and our shepherd for their insightful and constructive feedback to improve the paper. This work is supported by the National Natural Science Foundation of China (grant #62272265).

References

[1] 3rd Generation Partnership Project (3GPP). 5g; nr; overall description; stage-2 (3gpp ts 38.300 version 15.3.1 release 15), 2018.

[2] 3rd Generation Partnership Project (3GPP). 5g; nr; packet data convergence protocol (pdcp) specification (3gpp ts 38.323 version 15.2.0 release 15), 2018.

[3] 3rd Generation Partnership Project (3GPP). Lte; evolved universal terrestrial radio access (e-utra); packet data convergence protocol (pdcp) specification (3gpp ts 36.323 version 15.3.0 release 15), 2019.

[4] Wi-Fi Alliance. Wi-fi protected access: Strong, standards-based, interoperable security for today's wi-fi networks. *White paper, University of Cape Town*, pages 492–495, 2003.

[5] Sangwook Bae, Mincheol Son, Dongkwan Kim, CheolJun Park, Jiho Lee, Sooel Son, and Yongdae Kim. Watching the watchers: Practical video identification attack in LTE networks. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 1307–1324. USENIX Association, 2022.

[6] Michael D. Bailey, David Dittrich, Erin Kenneally, and Douglas Maughan. The menlo report. *IEEE Secur. Priv.*, 10(2):71–75, 2012.

[7] David A. Borman, Robert T. Braden, and Van Jacobson. TCP Extensions for High Performance. RFC 1323, May 1992.

[8] CAIDA. Caida spoofer project. https://spoofer.caida.org/summary.php, 2025. Accessed: 2025-03-06.

[9] 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 *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 209–225. USENIX Association, 2016.

[10] Jianjun Chen, Jian Jiang, Haixin Duan, Nicholas Weaver, Tao Wan, and Vern Paxson. Host of Troubles: Multiple Host Ambiguities in HTTP Implementations. In *23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[11] Jianjun Chen, Jian Jiang, Xiaofeng Zheng, Haixin Duan, Jinjin Liang, Kang Li, Tao Wan, and Vern Paxson. Forwarding Loop Attacks in Content Delivery Networks. In *Proceedings 2016 Network and Distributed System Security Symposium*, 2016.

[12] Jianjun Chen, Vern Paxson, and Jian Jiang. Composition Kills: A Case Study of Email Sender Authentication. In *29th USENIX Conference on Security Symposium*, 2020.

[13] Pinji Chen, Jianjun Chen, Mingming Zhang, Qi Wang, Yiming Zhang, Mingwei Xu, and Haixin Duan. Cross-Origin Web Attacks via HTTP/2 Server Push and Signed HTTP Exchange. In *Proceedings 2025 Network and Distributed System Security Symposium*, 2025.

[14] Weiteng Chen and Zhiyun Qian. Off-path TCP exploit: How wireless routers can jeopardize your secrets. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 1581–1598. USENIX Association, 2018.

[15] Brian P. Crow, Indra Widjaja, Jeong Geun Kim, and Prescott Sakai. IEEE 802.11 wireless local area networks. *IEEE Commun. Mag.*, 35(9):116–126, 1997.

[16] Kjeld Borch Egevang and Pyda Srisuresh. Traditional IP Network Address Translator (Traditional NAT). RFC 3022, January 2001.

[17] Simon Erni, Martin Kotuliak, Patrick Leu, Marc Roeschlin, and Srdjan Capkun. Adaptover: adaptive overshadowing attacks in cellular networks. In *ACM MobiCom '22: The 28th Annual International Conference on Mobile Computing and Networking, Sydney, NSW, Australia, October 17 - 21, 2022*, pages 743–755. ACM, 2022.

[18] Xuewei Feng, Chuanpu Fu, Qi Li, Kun Sun, and Ke Xu. Off-path TCP exploits of the mixed IPID assignment. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1323–1335. ACM, 2020.

[19] Bryan Ford, Saikat Guha, Kaushik Biswas, Senthil Sivakumar, and Pyda Srisuresh. NAT Behavioral Requirements for TCP. RFC 5382, October 2008.

[20] Yossi Gilad and Amir Herzberg. Off-path attacking the web. In *6th USENIX Workshop on Offensive Technologies, WOOT'12, August 6-7, 2012, Bellevue, WA, USA, Proceedings*, pages 41–52. USENIX Association, 2012.

[21] Yossi Gilad and Amir Herzberg. When tolerance causes weakness: the case of injection-friendly browsers. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 435–446. International World Wide Web Conferences Steering Committee / ACM, 2013.

[22] Yossi Gilad, Amir Herzberg, and Haya Schulmann. Off-path hacking: The illusion of challenge-response authentication. *IEEE Secur. Priv.*, 12(5):68–77, 2014.

[23] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D. Sutton, Pablo Serrano, Cristina Cano, and Douglas J. Leith. srslte: an open-source platform for LTE evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization, WiNTECH@MobiCom 2016, New York City, New York, USA, October 3-7, 2016*, pages 25–32. ACM, 2016.

[24] Matt Holdrege and Pyda Srisuresh. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, August 1999.

[25] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, May 2016.

[26] Martin Kotuliak, Simon Erni, Patrick Leu, Marc Röschlin, and Srdjan Capkun. Ltrack: Stealthy tracking of mobile phones in LTE. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 1291–1306. USENIX Association, 2022.

[27] Arash Habibi Lashkari, Farnaz Towhidi, and Raheleh Sadat Hosseini. Wired equivalent privacy (wep). In *2009 International Conference on Future Computer and Communication*, pages 492–495. IEEE, 2009.

[28] Ying Li, Yi Huang, Richard Yi Da Xu, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Darren Webb, and Guillaume Jourjon. Deep content: Unveiling video streaming content from encrypted wifi traffic. In *17th IEEE International Symposium on Network Computing and Applications, NCA 2018, Cambridge, MA, USA, November 1-3, 2018*, pages 1–8. IEEE, 2018.

[29] Yuejia Liang, Jianjun Chen, Run Guo, Kaiwen Shen, Hui Jiang, Man Hou, Yue Yu, and Haixin Duan. Internet's Invisible Enemy: Detecting and Measuring Web Cache Poisoning in the Wild. In *31th ACM Conference on Computer and Communications Security*, 2024.

[30] Matthew J. Luckie, Robert Beverly, Ryan Koga, Ken Keys, Joshua A. Kroll, and kc claffy. Network hygiene, incentives, and regulation: Deployment of source address validation in the internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 465–480. ACM, 2019.

[31] Keyu Man, Zhiyun Qian, Zhongjie Wang, Xiaofeng Zheng, Youjun Huang, and Haixin Duan. DNS cache poisoning attack reloaded: Revolutions with side channels. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1337–1350. ACM, 2020.

[32] Keran Mu, Jianjun Chen, Jianwei Zhuge, Qi Li, Haixin Duan, and Nick Feamster. The Silent Danger in HTTP: Identifying HTTP Desync Vulnerabilities with Gray-box Testing. In *34th USENIX Conference on Security Symposium*, 2025.

[33] Shiyue Nie, Yiming Zhang, Tao Wan, Haixin Duan, and Song Li. Measuring the deployment of 5g security enhancement. In *WiSec '22: 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks, San Antonio, TX, USA, May 16 - 19, 2022*, pages 169–174. ACM, 2022.

[34] Vern Paxson. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM Trans. Netw.*, 2(4):316–336, 1994.

[35] Zhiyun Qian and Zhuoqing Morley Mao. Off-path TCP sequence number inference attack - how firewall middleboxes reduce security. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 347–361. IEEE Computer Society, 2012.

[36] Zhiyun Qian, Zhuoqing Morley Mao, and Yinglian Xie. Collaborative TCP sequence number inference attack: how to crack sequence number under a second. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 593–604. ACM, 2012.

[37] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. DNS Security Introduction and Requirements. RFC 4033, March 2005.

[38] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. Breaking LTE on layer two. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1121–1136. IEEE, 2019.

[39] Nazmus Sakib, Shamim Ahmed, Samiur Rahman, Ishtiaque Mahmud, and Md Habibullah Belali. Wpa 2 (wi-fi protected access 2) security enhancement: Analysis & improvement. *Global Journal of Computer Science and Technology*, 12(6), 2012.

[40] Altaf Shaik, Jean-Pierre Seifert, Ravishankar Borgaonkar, N. Asokan, and Valtteri Niemi. Practical attacks against privacy and availability in 4g/lte mobile communication systems. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[41] Randall R. Stewart, Mitesh Dalal, and Anantha Ramaiah. Improving TCP's Robustness to Blind In-Window Attacks. RFC 5961, August 2010.

[42] Erik Tews and Martin Beck. Practical attacks against WEP and WPA. In *Proceedings of the Second ACM Conference on Wireless Network Security, WISEC 2009, Zurich, Switzerland, March 16-19, 2009*, pages 79–86. ACM, 2009.

[43] Mathy Vanhoef. Fragment and forge: Breaking wi-fi through frame aggregation and fragmentation. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 161–178. USENIX Association, 2021.

[44] Mathy Vanhoef and Frank Piessens. Practical verification of WPA-TKIP vulnerabilities. In *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, pages 427–436. ACM, 2013.

[45] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1313–1328. ACM, 2017.

[46] Mathy Vanhoef and Frank Piessens. Release the kraken: New kracks in the 802.11 standard. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 299–314. ACM, 2018.

[47] W3Techs. Default protocol https usage statistics, 2025. Accessed: 2025-03-09.

[48] Qi Wang, Jianjun Chen, Zheyu Jiang, Run Guo, Ximeng Liu, Chao Zhang, and Haixin Duan. Break the Wall from Bottom: Automated Discovery of Protocol-Level Evasion Vulnerabilities in Web Application Firewalls. In *2024 IEEE Symposium on Security and Privacy*, 2024.

[49] Wi-Fi Alliance. WPA3 Specification. https://www.wi-fi.org/file/wpa3-specification, 2023.

[50] Hojoon Yang, Sangwook Bae, Mincheol Son, Hongil Kim, Song Min Kim, and Yongdae Kim. Hiding in plain signal: Physical signal overshadowing attack on LTE. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 55–72. USENIX Association, 2019.

[51] Yaru Yang, Yiming Zhang, Tao Wan, Chuhan Wang, Haixin Duan, Jianjun Chen, and Yishen Li. Uncovering security vulnerabilities in real-world implementation and deployment of 5g messaging services. In Yongdae Kim, Jong Kim, Farinaz Koushanfar, and Kasper Rasmussen, editors, *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2024, Seoul, Republic of Korea, May 27-29, 2024*, pages 265–276. ACM, 2024.

[52] Yuxiang Yang, Xuewei Feng, Qi Li, Kun Sun, Ziqiang Wang, and Ke Xu. Exploiting sequence number leakage: TCP hijacking in nat-enabled wi-fi networks. In *31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024*. The Internet Society, 2024.

[53] Jiahe Zhang, Jianjun Chen, Qi Wang, Hangyu Zhang, Chuhan Wang, Jianwei Zhuge, and Haixin Duan. Inbox Invasion: Exploiting MIME Ambiguities to Evade Email Attachment Detectors. In *31th ACM Conference on Computer and Communications Security*, 2024.

[54] Jinghao Zhao, Qianru Li, Zengwen Yuan, Zhehui Zhang, and Songwu Lu. 5g messaging: System insecurity and defenses. In *10th IEEE Conference on Communications and Network Security, CNS 2022, Austin, TX, USA, October 3-5, 2022*, pages 37–45. IEEE, 2022.