Off-Path TCP Hijacking Attack to NAT-Enabled Wi-Fi Networks

Yuxiang Yang, Xuewei Feng, Qi Li, *Senior Member, IEEE*, Kun Sun, *Member, IEEE*, Ziqiang Wang, Ao Wang, *Graduate Student Member, IEEE*, and Ke Xu, *Fellow, IEEE*

***Abstract*—In this paper, we uncover a novel side-channel vulnerability arising from the shared NAT tables of Wi-Fi routers, enabling malicious insiders to hijack TCP connections between other clients and remote servers. First, by creating different NAT mappings within the shared NAT table, an off-path attacker can infer whether a victim client within the same Wi-Fi network is communicating with an external host over TCP, leveraging the widely adopted NAT port preservation strategy and insufficient reverse path validation in Wi-Fi routers. Once an active connection is detected, the attacker can manipulate the victim's NAT mapping in the shared NAT table with spoofed TCP packets, exploiting the lack of TCP window tracking in most routers. In this way, the attacker can intercept TCP packets from the server and obtain the current sequence and acknowledgment numbers, which in turn allows the attacker to forcibly close the connection, poison the traffic in plain text, or reroute the server's incoming packets to the attacker. We test 67 widely used routers from 30 vendors and discover that 52 of them are vulnerable. Also, we conduct an extensive measurement study on 93 real-world Wi-Fi networks and find that 75 of them (81%) are fully affected to our attack. Our case study shows that it takes about 17.5, 19.4, and 54.5 seconds on average to terminate SSH connections, download private files from FTP servers, and inject fake HTTP response packets with success rates of 87.4%, 82.6%, and 76.1%. Moreover, We evaluate the feasibility of the proposed attack in NAT-enabled IPv6 Wi-Fi networks. We responsibly disclose the vulnerability and suggest mitigation strategies to all affected vendors and have received positive feedback, including acknowledgments, CVEs, rewards, and adoption of our suggestions.**

***Index Terms*—TCP hijacking, off-path exploit, Wi-Fi network.**

Received 17 October 2024; revised 11 April 2025; accepted 23 June 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor Y. Liu. Date of publication 10 July 2025; date of current version 18 December 2025. This work was supported in part by the National Science Foundation for Distinguished Young Scholars of China under Grant 62425201, in part by the Science Fund for Creative Research Groups of the National Natural Science Foundation of China under Grant 62221003, and in part by the National Natural Science Foundation of China under Grant 62132011. *(Corresponding author: Ke Xu.)*

Yuxiang Yang and Ke Xu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, and also with Zhongguancun Laboratory, Beijing 100094, China (e-mail: yangyx22@mails.tsinghua.edu.cn; xuke@tsinghua.edu.cn).

Xuewei Feng is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: fengxw06@126.com).

Qi Li is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China, and also with Zhongguancun Laboratory, Beijing 100094, China (e-mail: qli01@tsinghua.edu.cn).

Kun Sun is with CSIS, Department of Information Sciences and Technology, George Mason University, Fairfax, VA 22030 USA (e-mail: ksun3@gmu.edu).

Ziqiang Wang and Ao Wang are with the School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: ziqiangwang@seu.edu.cn; wangao@seu.edu.cn).

Digital Object Identifier 10.1109/TON.2025.3586020

I. INTRODUCTION

Wi-Fi has emerged as one of the most popular technologies to provide Internet access, being widely used in restaurants, offices, and other public places. However, Wi-Fi networks are often exploited by attackers. In addition to breaking encryption [1], [2], [3], prior works have already been conducted on session hijacking in Wi-Fi networks [4], [5], [6], [7], [8], e.g., injecting forged wireless frames via vulnerabilities in WPA2 implementations [4], [5], eavesdropping on wireless channels [9], intercepting packets via side channels in VPN tunnels [6], creating a rogue clone (i.e., Evil-Twin) [7], or just abusing the classic ARP poisoning attack [8]. Fortunately, most of the prior attacks have been mitigated and targeted defense measures have been proposed [3], [5], [10], [11]. Nowadays, with the widespread deployment of wireless security mechanisms (e.g., WPA2/3) and the adoption of protection strategies (e.g., AP isolation, ARP prevention), it is increasingly difficult for an off-path attacker (i.e., with no control over the router) to obtain the communication information of other clients.

In public Wi-Fi networks, network address translation (NAT) is widely used to save IPv4 address space and protect internal clients from external attackers. After attaching to the same Wi-Fi network enabling NAT, clients share the external IP address to access the Internet. When it takes the upper protocols (e.g., TCP) into consideration, the router will create NAT mappings to keep track of the connections, which record the IP addresses, protocol, ports, and timeout, etc. In most cases, the router tries to keep the layer-4 information the same as the originators, such as the source port, which is the so-called *port preservation* [12]. However, cases are that some clients in the LAN may communicate with the same remote server with the same source port at the same time as they have no idea about each other. Although with very little probability, the router has to deal with these cases and it will assign a new TCP source port, change the IP address, and port at the same time when TCP packets pass through it. Besides, due to reasons such as performance considerations, the router will not record all of the session information in the NAT mappings, such as tracking the current TCP window.

In this paper, we uncover a new off-path TCP hijacking attack in Wi-Fi networks that exploits vulnerabilities in the shared NAT table and the NAT mapping strategies of routers. The attack includes three steps. First, the attacker probes the router's external IP address, identifies whether AP isolation is enabled and scans to find potential victims in the same

2998-4157 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

network. Second, the attacker infers the presence of TCP connections between any client and a remote server by sending TCP SYN and SYN/ACK packets. Third, the attacker replaces the NAT mapping of the victim connection with forged RST packets to the router and data packet to the server. After that, it can intercept the ACK packet from the server and obtain the sequence and acknowledgment numbers within it to completely hijack the TCP connection. Compared with prior attacks, the attacker only needs to connect to the same Wi-Fi network as the victim client and does not need any assistance of malicious puppets. Besides, previous TCP hijacking attacks primarily exploit vulnerabilities in endpoint OS implementations and thus depend on specific OS versions [13], [14], [15], [16], [17]. In contrast, our attack leverages weaknesses in NAT systems, making its feasibility largely determined by the router type.

Since NAT tables are shared resources in Wi-Fi routers, an off-path attacker can exploit this shared nature to manipulate NAT mappings and extract sensitive connection details from other clients on the network. In our investigations, most Wi-Fi routers (e.g., Asus, TP-Link) adopt *port preservation* when creating NAT mappings for TCP connections [18], [19]. The attacker can exploit this shared NAT table by intentionally sending a SYN packet, to the target server with a guessed client's port and d observing whether the router changes it. To verify, the attacker can send a spoofed SYN/ACK packet as a response with the server's IP as the source, the router's external IP as the destination, and the guessed port as the destination port. If the router does not enforce strict reverse path validation [20], [21], the forged SYN/ACK packet will not be dropped by the router, which is often the case in most routers. If an active connection exists with the guessed port, the router will choose another source port to initiate the connection, and then the SYN/ACK packet will be forwarded to the victim. Otherwise, the router will keep the port to initiate the connection and forward the SYN/ACK packet back to the attacker. By leveraging this shared NAT behavior, the attacker can infer not only the existence of an active connection but also the precise source port of the client.

Beyond connection inference, the attacker can also manipulate the victim's NAT mapping in the shared NAT table. After identifying a target TCP connection, the attacker can directly get the sequence and acknowledgment numbers of the connection by exploiting the disabled TCP window tracking strategy of Wi-Fi routers. To balance performance and compatibility, routers aim to reduce processing overhead while ensuring seamless operation for applications that rely on non-standard retransmissions or out-of-order packet acceptance. As a result, most routers will not strictly validate sequence and acknowledgment numbers of TCP packets. So the attacker can send forged TCP reset packets to clean the NAT mapping of the victim. After the NAT entry times out (typically 1–10 seconds), the attacker can send a TCP data packet using its private IP address and the same source port to the server with arbitrary sequence and acknowledgment numbers. The router will only translate the IP address of the packet except for the source port, as there is no port collision anymore. And the packet will match the victim connection from the perspective of the server, which will return a TCP ACK packet carrying the exact sequence and acknowledgment numbers of the victim connection upon seeing the packet with wrong numbers [22]. When arriving at the router, this ACK packet will be routed to the attacker as the NAT mapping has been falsified, and thus it steals the *SEQ* and *ACK* of the victim connection easily.

Once the sequence and acknowledgment numbers are obtained, the attacker can choose to launch three types of attacks: (i) TCP Denial-of-Service (DoS) attack to terminate victim TCP connections directly by sending RST packets. (ii) TCP hijacking attack to take over the NAT mapping and wait for the router to forward packets (intended for the victim client) to itself. (iii) TCP injection attack to poison the victim TCP traffic by sending crafted data packets after restoring the victim's mapping. Note that traffic encryption (e.g., HTTPS) may disturb the attacker's poisoning. However, about 15% of websites still transmit traffic in plaintext according to the reports on HTTPS adoption [23]. Moreover, various other applications, such as email clients using unencrypted protocols (e.g., POP3, IMAP, SMTP), legacy remote access protocols (e.g., Telnet) and certain messaging services, also operate in plaintext, making them susceptible to similar attacks.

We conduct a large-scale empirical study to assess the real-world impact of the attack. First, we investigate the default settings of routers on the market and have tested 67 router models from 30 vendors and find that 52 models from 24 vendors are vulnerable. Next, we evaluate 93 real-world Wi-Fi networks in six months, including many popular Wi-Fi scenarios (e.g., coffee shops, hotels, bookstores, and enterprises). The experimental results show that 75 (81%) of them are fully vulnerable to our attacks. We implement a PoC and perform case studies on applications like SSH, FTP, and HTTP to validate the effectiveness of the attack. In our experiments, an off-path attacker can detect and terminate an SSH connection in 17.5 seconds with a success rate of 87.4%, download private files from an FTP server within 19.4 seconds with a success rate of 82.6%, and manipulate web traffic within 54.5 seconds with a success rate of 76.1%, on average.

Additionally, we evaluate the feasibility of conducting TCP hijacking attacks in NAT-enabled IPv6 Wi-Fi networks, focusing on two primary NAT strategies: NAT66 and NAT64. We investigate whether the necessary attack conditions are met under these configurations and find that the vulnerability is also present in IPv6 scenarios. Using the OpenWrt firmware through experimental validation, an off-path attacker is able to successfully hijack TCP connections in both NAT66 and NAT64 configurations with a success rate of approximately 94% and 92%, respectively. The average time required to complete the attack is 23.24 seconds for NAT66 and 27.39 seconds for NAT64, demonstrating that the attack is feasible and efficient in IPv6 settings.

Besides, we identify the root cause and suggest mitigation to void this attack. We have responsibly disclosed the vulnerability to OpenWrt and the affected router vendors. Till now, developers from OpenWrt and 7 of these vendors have confirmed the vulnerability and are repairing it in their products according to our suggestions. In addition, 10 CVE numbers (CVE-2023-30305 to CVE-2023-30314) have been

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

assigned for products from different vendors. Other vendors are still investigating the issue.

Contributions. Our main contributions are the following:

- We uncover new side channel vulnerabilities of the shared NAT tables in Wi-Fi routers that can be exploited to attack TCP connections by off-path malicious insiders.
- We perform a large-scale measurement and reveal a number of routers and Wi-Fi networks vulnerable to the attack. Our extensive case studies on SSH, FTP, and HTTP show that our attacks can cause potential damage in the real world.
- We study the feasibility of the attack in NAT-enabled IPv6 Wi-Fi networks extendedly. Our comprehensive analysis of both NAT64 and NAT66 implementations demonstrates that the attack remains effective in IPv6-based Wi-Fi networks.
- We suggest three countermeasures by eliminating the conditions to fight back the attack, and some of them have been adopted by the affected manufacturers.
- We have made the attack code open-source to facilitate reproduction and encourage further research.

II. Background

A. NAT and Port Allocation Strategies

Network Address Translation (NAT) is a technology developed to solve the shortage of IPv4 addresses and hide the network topology from an external entity [24], which is widely used by routers in Wi-Fi networks. When packets traverse through the router, it has to translate the IP addresses of the packets between internal and external addresses[1] and record the other necessary information of the related connection. The router maintains a NAT mapping table to keep track of the internal IP addresses and ports associated with each corresponding external IP address and port, which allows incoming packets to be directed to the correct host on the private network. Since our work focuses on the TCP protocol, we will illustrate the NAT behavior of TCP mappings henceforth.

When an internal host initiates a connection to an external server, i.e., sending a `SYN` packet, the router will create a new mapping in the table, which is called a binding in NAT terminology [25]. Besides, we find that not only `SYN` packets but also packets with `PUSH`, or `ACK` flags can incur new NAT mappings at the router. The mapping will record the source IP addresses and ports translated before and after, the destination IP address and port, protocol, session state, and timeout. After receiving the replies, the router will forward the packets to the internal host according to the mapping and updates its state simultaneously. As for the translation of source ports, there are three main strategies for NAT devices [19]: (1) *port preservation*, where the NAT device attempts to preserve the source port if possible. When a collision happens, i.e., different internal hosts choose the same source port, it should resolve the collision by selecting another unused port. (2) *random selection*, where the NAT device translates the source port to another random port. (3) *sequential selection*, where the NAT device selects a random port for the first connection to each destination and translates the ports of subsequent packets to that destination consecutively based on the first port.

As with any stateful middle device, routers have to manage the state of mappings and track active flows. Generally, the routers often rely on both the states of connections and timeouts of mappings to prune unnecessary NAT mappings. RFC 5382 recommends that the minimum timeout for the `ESTABLISHED` state is 2 hours and 4 minutes, which is faithfully implemented in most routers [12], and the routers also set timeouts for other states (e.g., 1 second or 10 seconds for the `CLOSE` state which the mapping will turn into upon seeing corresponding `RST` packets).

B. TCP Window Tracking in Routers

As a middle device between the client and server, the router has to record the connection information of the related hosts. However, as the TCP protocol was originally designed for end-to-end communication, the router cannot and will not record all connection details for various reasons, including performance optimization and compatibility considerations. For instance, the router will choose not to track the current TCP window of the connection, and thus it will not check the sequence and acknowledgment numbers of TCP packets strictly. By adopting this approach, the router reduces processing overhead while ensuring compatibility with applications that rely on non-standard retransmissions. The open-sourced router firmware, i.e., OpenWrt and AsusWrt, both have related options to reduce CPU overhead, i.e., the `nf_conntrack_tcp_no_window_check` option in OpenWrt and the `ip_conntrack_tcp_be_liberal` option in AsusWrt. These options are set to true by default. The difference between the two systems is that OpenWrt does not check the sequence number of the packet at all, while AsusWrt only checks if the sequence number is beyond the current sequence number in a 2G space. Besides, we found most of the routers in the market also disable the TCP window tracking strategy by default and have similar behaviors to the two systems above.

C. Reverse Path Validation

To prevent IP spoofing attacks, RFC 2827 and RFC 3704 propose the concept of reverse path validation, which verifies the authenticity of inbound traffic by checking whether the source IP address can be routed back via the interface on which packets are received against the routing table, to ensure they come from an authorized sender [20], [21]. With this strategy enabled, only if the packets can be routable back from the incoming interface will they be processed by the kernel and routed to their destinations. Otherwise, they will be dropped. Most Linux-based systems control the strategy through the *rp_filter* kernel variable, which offers three options [26]:

- **0:** The source address validation is disabled.
- **1:** Strict Mode. The device should compare the source address of incoming packets to the Forwarding Information Base (FIB). If the incoming interface is not the best reverse path, packets will be dropped.

[1]Since our work considers multiple levels of NAT, the router's external IP address may not be a public IP address.

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

- **2:** Loose Mode. The device compares the source address of packets against the FIB, and only if the packets are not reachable via any interface will they be dropped.

RFC 3704 recommends using strict mode for reverse path validation to prevent IP spoofing attacks. However, many NAT implementations default to a less strict or completely disabled reverse path filtering mode. This is because strict enforcement can break applications that rely on asymmetric routing, such as devices with multiple network interfaces (e.g., smartphones using both Wi-Fi and cellular data). In such cases, legitimate packets may arrive on unexpected interfaces, causing strict reverse path filtering to incorrectly discard them. To balance security with functionality, routers often use the loose mode (or disable filtering entirely) to maintain application compatibility, even though this increases vulnerability to spoofing attacks. Previous research [6] has shown that in VPN scenarios, the lack of reverse path validation on client devices allows a blind in-path attacker (e.g., a malicious AP) to spoof packets to learn the virtual IP used by the tun0 interface for the VPN connection. By contrast, we find that most routers also do not obey the recommendation, and they will not drop packets with spoofed source addresses matching a connection in the NAT mappings and will accept them on any interface.

III. Threat Model

Figure 1 illustrates the threat model of our off-path TCP attacks in Wi-Fi networks. The model consists of three hosts and one router, namely, a remote server, a victim client, an off-path attacker, and a vulnerable router. The router acts as the gateway to provide Internet services for the Wi-Fi network. The remote server may be a web application, an SSH or FTP server in different attack scenarios. The victim client (e.g., a mobile phone) is connected to the router to communicate with the remote server, i.e., visiting web pages, downloading files through FTP, or using the SSH service to control remote hosts. The off-path attacker is a malicious client who can connect to the same Wi-Fi network as the victim client.

Existing studies [7], [27] demonstrate that a malicious insider can create an evil twin of the network and trick the victims into connecting to it by broadcasting the same SSID in open (with no encryption) or home mode (accessed through pre-shared key) Wi-Fi networks, thus hijacking their traffic. Besides, ARP poisoning also allows an attacker in the same LAN to manipulate local network routing. However, these attacks can be throttled by existing defenses, e.g., Rogue AP detection, network isolation [28], [29], [30], [31], [32]. It is widely believed that only AP isolation enabled enterprise mode Wi-Fi networks can effectively protect clients from each other, whereas open and home mode Wi-Fi networks face challenges in preventing insider threats. In this work, we propose a novel attack that can evade all defenses above in Wi-Fi networks, even in enterprise mode Wi-Fi networks. Our attack operates at a different layer of the network stack by targeting the NAT table at the router, enabling off-path TCP hijacking without requiring direct control over packet forwarding. These attacks do not subsume each other but instead pose complementary threats to wireless network security. We will further discuss these attacks in Section VII-B.

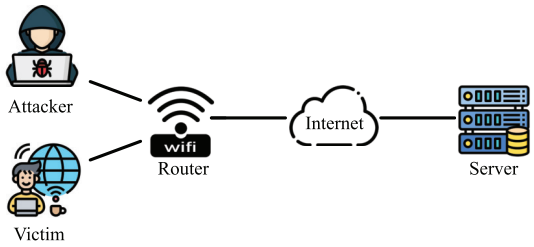


Fig. 1. Threat model of TCP hijacking attacks in Wi-Fi networks.

There are some requirements to successfully launch the attacks. First, the attacker should be able to probe the external IP address of the router. Then the attacker should be able to get the status of AP isolation in the network [33]. The attacker can successfully carry out the TCP DoS and TCP hijacking attacks regardless of whether AP isolation is enabled, while the TCP injection attack will be thwarted when enabled. Besides, the attacker has to target the remote server that the client is communicating with or will connect to, which can be set as those providing popular services as previous works [16], [17], e.g., famous servers, web search engines, or social sites.

Second, the router should adopt *port preservation*. Our investigations show that most routers adopt it except for two from Huawei and pfSense [34]. Also, the router should disable reverse path validation which is the default behavior in 24 out of 30 router vendors we tested. Moreover, the router should disables TCP window tracking which is also disabled by most routers by default, with the exception of Cisco Meraki.

Third, the victim client does not communicate with the server continuously. The NAT mapping will be removed completely after its timeout (1 second or 10 seconds in our test) after receiving TCP `RST` packets. If the client's communication continues during this period, it may interfere with the attack as the mapping will be refreshed. As there are many long-lived TCP connections that clients periodically retrieve new data from the server in minutes and 42% of the tested routers set the timeout to only 1 second, the attacker has been provided with enough time to finish its attack.

IV. Attack Procedure

A. Attack Overview

To perform our attacks, the attacker has to carry out the following three steps:

1. Probe the router's external IP address and identify the status of AP isolation.
2. Make inferences about whether there is any active connection from the LAN to the server.
3. Evict and construct NAT mappings at the router and then intercept the sequence and acknowledgment numbers from the replies to unsolicited packets from the server.

After the above steps, the attacker can terminate the connection directly or hijack the connection by replacing the victim client. Besides, when AP isolation is disabled, the attacker can restore the original NAT mapping of the victim client at the router and send fake response packets to the client.

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

B. Phase 1: Probing the Network

In this step, the attacker prepares the attack in two aspects, namely, identifying the status of AP isolation in the network and probing the external IP address of the router. Firstly, the attacker detects whether AP isolation is enabled via network scanning tools (e.g., Nmap [35], MacStealer [33]). If it is disabled, the attacker records the scanning results of potential victim clients for the futural TCP injection attack. Note that the attacker does not need to know the specific private IP address of the victim client (i.e., which IP is the victim), as we will show that it only needs to send related packets to the router, the server, or all of the clients in the subsequent attack phases, which is different from previous works that they will choose a target victim client beforehand, i.e., identifying whether a given client is communicating with the server [16], [17] or redirecting the victim's traffic to the attacker [36].

Secondly, the attacker probes the router's external IP address. With the widely deployed carrier-grade NAT [37], the Wi-Fi networks in the real world may consist of multiple levels of NAT [38], which means that the router's external IP address is not always a public IP address that can be obtained easily by querying its own public IP. We adopt the following methods to deal with this problem. (i) First, the attacker gets the gateways along the way to any outside host (e.g., 8.8.8.8) through Traceroute [39]. Second, the attacker issues the ping command to the second gateway with the *RECORD_ROUTE* option, which will record the passed routes [40], and then all the IP addresses of the passed interfaces will be returned (see Figure 8 in Appendix A, the Supplementary Material). (ii) In certain scenarios, the aforementioned method may encounter failure. In such cases, the attacker can opt to scan the subnet of the second gateway to identify live hosts' IP addresses. Subsequently, it can proceed to ping these IPs using the *RECORD_ROUTE* option. When the ping reaches the external IP of the router, the previously passed routes will be returned. However, when pinging other IPs, the routes will not be returned. Besides, the attacker can access these IPs via a web browser. When accessing the external IP of the router, the router's setting page (Web GUI) will be displayed, whereas accessing the other IPs will lead to different pages.

C. Phase 2: Making Inferences About Active Connections

Assuming that the attacker has connected to a Wi-Fi network, in which one of the normal users has established a TCP connection with a remote server from source port `m`. The router has a corresponding NAT mapping to keep track of the connection. The attacker intends to infer which source port is used by the victim client of the connection.

Figure 2 illustrates the side-channel vulnerability present in the shared NAT table and demonstrates how an attacker leverages the shared resource by exploiting the NAT *port preservation* strategy and insufficient reverse path validation of the router to infer a client's source port. First, the attacker sends a `SYN` packet targeting the server with its own IP address and a guessed port number as the source. If the source port number (e.g., `n`) specified in the `SYN` packet does not equal `m`, the router will create a new NAT mapping with source port `n` that records this new TCP connection. Note that with the wide

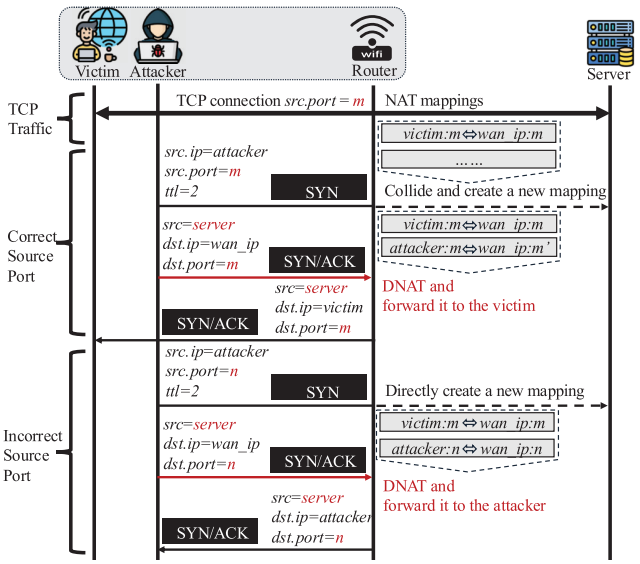


Fig. 2. Inferring the source port of the victim TCP connection.

deployment of NIDS at the server side [41], a large amount of `SYN` packets arriving at the server may be detected, and the server may find it attacked and take corresponding fightback. The attacker can set the `TTL` of the `SYN` packet to a small number (e.g., 2 in our test), and thus the packet will be dropped quickly at the intermediate routers. In most cases, the routers do not deploy detection systems typically.

Then, the attacker impersonates the server and sends a forged `SYN/ACK` packet whose destination is the router's external IP address and whose destination port is the guessed source port `n`. According to RFC 3704, the reverse path of packets received should be strictly checked to prevent IP spoofing attacks. In this case, as the `SYN/ACK` packet is received from the router's internal interface while its source address is a public IP address and it actually cannot be routed from this incoming interface for responding to the packet, then it should be dropped. However, many routers in the real world do not adopt the RFC recommendation and will forward it according to NAT mappings. Since there is a NAT mapping that translates the router's external IP to the attacker's private IP when the source is the remote server and the destination port is `n`, the forged packet will match this mapping and be forwarded to the attacker. In this way, the attacker can receive the forged packet that is sent from itself again if the guessed source port `n` is not equal to the victim client's source port `m`.

If the attacker guesses the right source port, i.e., `m`, when the `SYN` packet arrives at the router, it will translate the source port of the new mapping to another port due to the collision. Let us say that the changed source port is `m'`. In the second step, when the forged `SYN/ACK` arrives at the router, however, it will be forwarded to the victim according to the client's NAT mapping as the port specified in it is `m` instead of `m'`. Thus, the attacker cannot receive the forged `SYN/ACK` packet again if the port it guesses is right, i.e., previously occupied. In this way, the attacker can infer that there is a connection from some local host to the target server with the source port `m`.

The attacker repeats the above procedure, i.e., changing the guessed source port number specified in the forged `SYN`

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

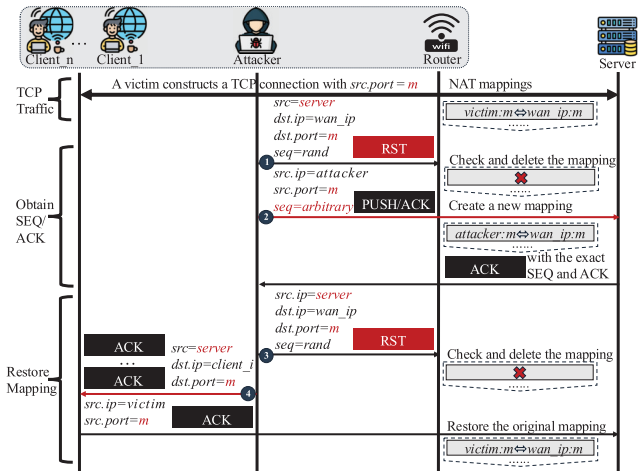


Fig. 3. Hijacking active connections.

and SYN/ACK packets and then observing if it can receive SYN/ACK back until the correct port m is identified, which will be used for the subsequent attacks.

D. Phase 3: Hijacking Active Connections

Once the attacker has determined an active TCP connection to a remote server with source port m, it will attempt to obtain the current sequence number *SEQ* and acknowledgment number *ACK* of the server. Previous works mostly infer these values by exploring the entire possible 4G space via some side channels [16], [17]), which are rather time-consuming. However, we demonstrate a new method to obtain these two values directly. We assume natural communication intervals between the client and server, where the client periodically initiates requests and waits for responses, or the server proactively pushes notification messages, which are often the cases in real-world services.

Figure 3 illustrates our method, where the attacker manipulates NAT mappings in the shared NAT table to hijack the TCP connection between the victim client and the server. Firstly, the attacker cleans the router's NAT mapping of the victim connection by sending spoofed TCP RST packets whose source is the server and destination IP is the router's external IP, and destination port is the previously inferred port m. The sequence numbers specified in these packets are crafted for various brands and types of routers due to their different behaviors of disabling the TCP window tracking strategy. Generally, there are two popular behaviors as stated in Section II-B that the first type of router does not check the sequence number at all, and the second type of router will check if the sequence number is beyond the exact sequence number in a 2G space. For the first type of router, the attacker can specify an arbitrary sequence number in one crafted RST packet to clean the NAT mapping. And for the second type of router the attacker can send two RST packets, one with sequence number x and the other with $(x + 2G)\ \%\ 4G$, which ensures that one of them will fall within the required range.

To simplify the description, we will only elaborate on the details of the attack on the first type of router. After receiving the RST packet, the router will falsely process the packet as it does not perform reverse path validation, and the state of the recorded TCP mapping will transfer from ESTABLISHED to CLOSE. The CLOSE state will last for a timeout according to a kernel variable (e.g., *nf_conntrack_tcp_timeout_close* in OpenWrt-based systems). In our empirical investigation, 41% of the tested routers set this value to as short as 1 second, and the other vulnerable routers set it to 10 seconds. After the countdown of the timeout, the original TCP NAT mapping at the router will be completely cleared.

Secondly, after the original NAT mapping has been fully evicted, the attacker replaces the victim client by constructing a new mapping at the router via sending a data packet with PUSH/ACK flags to the remote server, using m as the source port and its private IP as the source IP address. The sequence and acknowledgment numbers specified in the packet can be arbitrary as the router does not verify them. From the perspective of the remote server, this packet is from the same public IP address of the client with source port m and it will match the victim connection. Since the sequence and acknowledgment numbers specified in the packet are wrong, an ACK packet with the server's exact sequence and acknowledgment numbers will be triggered back [22]. When the ACK packet arrives at the router, it will be routed to the attacker according to the new NAT mapping created just now. Then it can obtain the *SEQ* and *ACK* of the victim connection directly.

After the above two steps, the attacker can decide on the follow-up procedures according to its attack purpose and we illustrate three types of possible attacks. (i) TCP DoS attack. To forcibly close an encrypted tunnel (e.g., SSH or HTTPS), the attacker sends forged TCP RST packets to the server using the obtained connection details. This terminates the session on the server side, preventing the client from receiving responses and resulting in a denial-of-service attack. (ii) TCP hijacking attack. To hijack the traffic from the server to the victim client, the attacker can take over the NAT mapping and impersonate the client again with the exact sequence and acknowledgment numbers to launch requests to the server and wait for responses from the server. For instance, at the beginning, the victim client logins into the FTP server. After the attack, the attacker can bypass the initial verification stage by replacing the victim client to send requests to the server, which may lead to permission bypass and privacy leakage. (iii) TCP injection attack. To send forged responses by impersonating the server when the victim client initiates a new request later, the attacker needs to restore the victim's original NAT mapping to avoid disrupting its normal communication. We will elaborate on this case in the last two steps in Figure 3.

The attacker repeats the first step to clean its mapping and waits for another NAT mapping timeout of CLOSE state. To restore the victim's original mapping without knowing its private IP, the attacker can send forged ACK packets to all probed local hosts (see Section IV-B) when AP isolation is disabled. The source of the ACK packets is the server, and the destination port is m. After arriving at the irrelevant hosts with no corresponding connections, the packets will be dropped. Yet the victim will send an ACK packet back to the server, which restores its NAT mapping at the router. Then the attacker can inject forged responses to the client by sending them to the router's external IP via abusing the disabled reverse

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PARTIAL TESTED ROUTERS FROM 30 VENDORS

No.	Router Model	Vendor	OS	Generation	Port Preservation	Reverse-path Validation Disabled	TCP Window Tracking Disabled	TCP Close Timeout (second)	Vulnerable
1	TL-XDR6020	TP-Link	Linux-based	Wi-Fi 6	✔	✔	✔	1	✔
2	TL-WDR7620	TP-Link	Vxworks-based	Wi-Fi 5	✔	✘	✔	1	✘
3	AX3 Pro	Huawei	EMUI (Linux-based)	Wi-Fi 6	✔	✔	✔	10	✔
4	AR6140E-9G-2AC*	Huawei	VRP (Linux-based)	-	✘	✘	✔	10	✘
5	V6G	360	360OS(Linux-based)	Wi-Fi 6	✔	✔	✔	1	✔
6	Magic R365	H3C	Comware(Linux-based)	Wi-Fi 5	✔	✔	✔	10	✔
7	W30E	Tenda	Linux-based	Wi-Fi 6	✔	✔	✔	1	✔
8	RAX1800Z	China Mobile	AOS(Linux-based)	Wi-Fi 6	✔	✔	✔	10	✔
9	X32 Pro	Ruijie	RGOS(Linux-based)	Wi-Fi 6	✔	✔	✔	1	✔
10	Redmi RA81	Xiaomi	MiWiFi(Linux-based)	Wi-Fi 6	✔	✔	✔	1	✔
11	MW300R	Mercury	Vxworks-based	Wi-Fi 4	✔	✘	✔	1	✘
12	X30G	Mercury	Linux-based	Wi-Fi 6	✔	✔	✔	1	✔
13	RAX50	Netgear	DumaOS(Linux-based)	Wi-Fi 6	✔	✘	✔	10	✘
14	RT-AX89X	ASUS	AsusWrt(Linux-based)	Wi-Fi 6	✔	✘	✔	10	✘
15	E9450	Linksys	Linux-based	Wi-Fi 6	✔	✔	✔	10	✔
16	QUANTUM D2G	Wavlink	Linux-based	Wi-Fi 5	✔	✔	✔	10	✔
17	CF-616AC	Comfast	OrangeOS(Linux-based)	Wi-Fi 5	✔	✔	✔	10	✔
18	DI-7003GV2*	D-Link	Linux-based	-	✔	✔	✔	1	✔
19	AX3000	ZTE	ZXR10ROS(Linux-based)	Wi-Fi 6	✔	✘	✔	10	✘
20	M80*	IP-COM	Linux-based	-	✔	✔	✔	1	✔
21	SK-WR6640X	Skyworth	Linux-based	Wi-Fi 6	✔	✔	✔	10	✔
22	VE5200G*	Volans	Linux-based	-	✔	✔	✔	1	✔
23	NBR1009GPE	Netcore	NOS(Linux-based)	-	✔	✔	✔	1	✔
24	Wimaster*	Wimaster	Linux-based	-	✔	✔	✔	10	✔
25	IK-Enterprise*	iKuai	iKuaiOS(Linux-based)	-	✔	✔	✔	10	✔
26	Instant On AP22	Aruba	ArubaOS(Linux-based)	Wi-Fi 6	✔	✘	✔	10	✘
27	EdgeRouter X*	Ubiquiti	Linux-based	-	✔	✔	✔	10	✔
28	AX1800	JdCloud	Linux-based	Wi-Fi 6	✔	✔	✔	10	✔
29	Cisco Meraki 64*	Cisco Meraki	Linux-based	-	✔	✘	✘	-	✘
30	eero pro	Amazon	Linux-based	Wi-Fi 5	✔	✔	✔	10	✔
31	Google Wi-Fi	Google	ChromeOS(Linux-based)	Wi-Fi 5	✔	✔	✔	10	✔
32	GL-MT3000	GL.iNet	Linux-based	Wi-Fi 6	✔	✔	✔	10	✔
33	pfSense 2.7.0*	pfSense	FreeBSD-based	-	✘	✘	✔	90	✘

✔means that the router is satisfied with the condition, and ✘means that the router is dissatisfied with the condition.
✔means that the router is vulnerable to our attack, and ✘means that the router is immune to our attack.
* means that the model is an enterprise router which does not support Wi-Fi by itself and needs to work together with wireless access points.

path validation and NAT mappings. However, if AP isolation is enabled, this step fails as the attacker cannot reach other clients, and thus this attack is thwarted.

Note that the countdown of timeouts when the NAT mappings are in `CLOSE` state will be refreshed if related packets are traveling through the router, which may lead to interference with the attack. We will discuss it in Section VII-A.

V. EMPIRICAL STUDY

In this section, we first investigate the vulnerable routers on the market. Next, we conduct case studies to evaluate the attack in real-world Wi-Fi networks.

Ethical Considerations. We addressed the ethical issues from the following perspectives. First, we provided the Wi-Fi network administrators with detailed explanations of our experimental plans and obtained their approval before conducting the experiments. Second, with the help of the administrator, we ensured that no other users were accessing the Wi-Fi network during our experiments, thus avoiding potential risks or side effects for other users. We then deployed our machine (a laptop or a cellphone) as the victim client in the Wi-Fi network, thus ensuring that all the machines involved in our experiments were under our control and would not affect other machines. Finally, after completing the experiments, we provided feedback on the results to the administrator. Moreover, we recommended that they restart the Wi-Fi router and clear the cache to restore the network to a safe state.

A. Analysis of Routers

The attack leverages the strategies adopted by the router, and only if all of the conditions are fulfilled can our attack succeed. In order to explore the coverage of vulnerable routers, we perform tests on real router models from lots of vendors, including 360, Aruba, ASUS, Amazon, Cisco Meraki, China Mobile, Comfast, D-Link, GL.iNet, Google, H3C, Huawei, IP-COM, iKuai, JdCloud, Linksys, Mercury, Netgear, Netcore, Ruijie, Skyworth, Tenda, TP-Link, Ubiquiti, Volans, Wavlink, WiMaster, Xiaomi, and ZTE. To our best knowledge, the operating systems of most routers we tested are based on Linux with custom modifications, except for some router models from TP-Link and Mercury, which are based on VxWorks. We also build a soft routing environment with a FreeBSD-based firmware, i.e., pfSense 2.7.0 [34]. In total, we perform tests on 67 router models from 30 vendors. For each router model, we test if it fits all attack conditions proposed in Section III. We list the test results of 33 representative routers from these 30 vendors in Table I. The detailed information of the 67 tested routers is listed in Table IV of Appendix B in the Supplementary Material. In conclusion, 52 of the 67 tested routers are vulnerable, and 15 models are immune to the attack as they do not fulfill all of the conditions.

First, the router has to take the *port preservation* strategy. In our test, most of the routers adopt this strategy by default. Only the enterprise wired router model "AR6140E-9G-2AC" produced by Huawei and the soft routing machine with pfSense take the *random selection* strategy that

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

prevents the attacker from inferring the source port of clients ' connections.

Second, we investigate the deployment of the reverse path validation strategy. Among these routers, Netgear and Asus set the kernel variable *rp_filter* to 1 by default and will check the packet received strictly. Some old-styled models of TP-Link and Mercury (i.e., VxWorks-based) will also validate the received packets. However, their newest models (e.g., designed for Wi-Fi 6) and some enterprise routers will not validate them anymore. In addition, ZTE routers, the router model "AR6140E-9G-2AC" from Huawei, and pfSense will also not forward the forged packets. The routers from the other vendors all disable reverse path validation.

Third, the router has to disable the TCP window tracking strategy. In our test, only one enterprise wired router model "Meraki 64" produced by Cisco Meraki will check the sequence number strictly and all of the other routers disable TCP window tracking while the processing logic is slightly different. Asus, Google, Netgear, Tenda, Wimaster, and ZTE routers will check if the sequence number is in the 2G space beyond the exact sequence number. And the other routers do not check the sequence number at all.

Fourth, the timeout of NAT mappings in TCP `CLOSE` state may influence our attack. The shorter it is, the easier the attack can succeed. Among the 66 router models without TCP window tracking, 28 of them will clean the NAT mapping in only 1 second, 37 of them will be tricked to clean the mapping in 10 seconds, and the default setting of pfSense is 90 seconds.

We take the first row as an instance to analyze the results in Table I. The Linux-based router model "TL-XDR6020" produced by TP-Link, provides the latest generation of Wi-Fi 6 for network services. As for the four metrics mentioned above, this model takes the *port preservation* strategy, does not validate the reverse path of received packets, disables the TCP window tracking, and sets the TCP `CLOSE` timeout to 10 seconds by default. In this way, it is vulnerable to our attacks.

B. Attack Evaluation

To evaluate the impacts of the attack in the real world, we also conduct thorough experiments of the attack in 93 various Wi-Fi networks. We investigate whether the conditions of the attack are fulfilled in each network by taking three case studies of attacks on SSH, FTP, and HTTP applications and measuring the time cost and success rate of each attack.

Experimental Setup. Our experiments consist of four devices:

- *Router.* The router in Wi-Fi networks works as the gateway to provide Internet access and forward packets between local clients and outside servers.
- *Remote Server.* For the DoS attack, we set up an SSH server equipped with Ubuntu 22.04 (kernel version 5.15.0), OpenSSH 8.9, and OpenSSL 3.0.2. For the hijacking attack, we set an FTP server equipped with Ubuntu 22.04 (kernel version 5.15.0) and vsftpd version 3.0.3. And for the injection attack, we pick a well-known finance website *www.ANONYMOUS.com* (anonymized ethically) in which the client initiates a long-lived TCP connection that periodically retrieves data updates every minute.
- *Victim Client.* Though the OS type or version of the client is unrestricted in our attack, we still deploy victim clients equipped with five typical OSes (i.e., Windows, Linux, Mac, iOS, and Android). Each victim client has connected to the Wi-Fi network. In the case of DoS and hijacking attacks, the victim client will communicate with the remote server through SSH and FTP. And in the case of the injection attack, the victim client will access the website above to get the newest future index of stocks (e.g., `HSI`, `HSCEI`).
- *Attacker.* An attack machine is equipped with Linux 5.15.0, which is capable of crafting packets. The attacker aims to forcibly close the client's SSH connection with the remote server, steal private files from the FTP server, or inject fake HTTP responses to the client.

Attack Procedure. The attacker first tries to get the router's external IP address and test whether AP isolation is enabled. Next, the attack can be constructed in the following steps: (1) inferring the source port of the TCP connection to the given server, (2) evicting the router's original NAT mapping and constructing a new one by sending a TCP data packet to the server, which in turn incurs an `ACK` packet from it, (3) terminating the SSH connection or requesting an FTP file download, (4) restoring the original NAT mapping and answering the client's HTTP requests with forged segments. For the SSH DoS attack, we define the result that the client or the server cannot receive messages from each other as successful. The FTP hijacking attack can be deemed successful when the attacker manages to download files from the FTP server that belong to the victim client. And for the HTTP injection attack, we define the result that the client receives forged packets, and the falsified data is displayed on the web page as a successful attack. Compared with SSH DoS and FTP hijacking attacks, the conditions are more difficult to meet as the attacker only knows that the request interval is 60 seconds while it does not know when the client will request an update.

We repeat the experiments 20 times in each tested Wi-Fi network. Each experiment is conducted independently with a renewed connection between the client and server. To limit the time of experiments, we take an experiment as a failure if the attacker cannot achieve its goal in 5 minutes. As mentioned before, it takes time (mostly 1 second or 10 seconds) for the mapping to disappear completely. The countdown will be refreshed if the client sends packets during this period, which may interfere with the time cost and success rate of our attack. To simulate real-world situations, we require the tested client to send requests to the server for random times, and we set the interval between two requests as a random number from 5 to 30 seconds during the 5 minutes of an experiment.

Experimental Results. We evaluate our attack against 93 real-world Wi-Fi networks across common scenarios like coffee shops, hotels, and offices. As the attacker can sniff the unencrypted packets in open networks, we mainly launch our experiments under networks protected by WPAs, e.g., 45 with WPA2-Personal enabled (home mode), 22 with WPA2-Enterprise enabled (enterprise mode), and 26 with

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

TABLE II
EXPERIMENTAL RESULTS IN OUR TESTS (ON AVERAGE)

Attack Type	Inferring Port(s)	Hijacking Connection(s)	Finishing Attacking(s)	Total Time(s)	BW (pkts)	Success Rate
SSH DoS	8.1	8.4	1.0	17.5	4000	87.4%
FTP Hijacking	9.1	9.2	1.1	19.4	4000	82.6%
HTTP Injection	9.4	15.2	29.9	54.5	4000	76.1%

WPA3-Personal enabled (home mode) and we do not find any network with WPA3-Enterprise enabled. The experimental results illustrate that more than 81% Wi-Fi networks (i.e., 75/93) are fully vulnerable that they satisfy all attack conditions. For the other 18 networks, 9 of them have AP isolation enabled, which thwarts the HTTP injection attack. However, the SSH DoS and FTP hijacking attacks remain unaffected. Our attack fails in 7 networks as they do not use the vulnerable routers, and we cannot get the router's external IP as described in Section IV-B in the rest two networks. We can acquire the external IP addresses of routers using the route-recording method in 80 networks. Additionally, we employ the scanning method in 11 networks, utilizing router models from 6 vendors. In the remaining two networks using routers from Cisco Meraki and Aruba, we fail to get the external IP address unless we log in to the control page, and we take them as failures.

Next, we elaborate on our experimental results in the Wi-Fi networks as shown in Table II, which presents the experimental results for the attacks, including the averaget time to infer the port with the steps in Figure 2, the averaget time to hijack the connection with the steps in Figure 3, the time to finish the attack (e.g, inject the malicious packets), total attack duration, bandwidth usage, and success rate. In the case of the SSH DoS attack, the average time cost of identifying the client's source port is 8.1 seconds with a bandwidth of 4000 packets per second, which is much shorter than previous methods [16], [17], [42] as we only need to transfer packets in the same LAN and we are not restricted by rate limits. And the average time cost of hijacking the connection, i.e., obtaining the exact sequence and acknowledgment numbers is 8.4 seconds, as this step mainly relies on the default settings of the timeout of `CLOSE` state in NAT mappings. Besides, the communication between the client and the server may also influence the time cost. Finally, the average time cost of totally terminating an SSH connection is 17.5 seconds, and the average success rate is 87.4%. The failure cases in the tests are due to continuous communications between the client and the server (e.g., the client requests a file download and related packets will always refresh the NAT mapping). After the attack succeeds, the client's SSH terminal will be stuck for a period of time, which greatly affects the user experience.

For the FTP hijacking attack, the average time costs of identifying the source port and getting *SEQ* and *ACK* are 9.1 and 9.2 seconds, respectively, which results in a time cost of 19.4 seconds for the entire attack to get a private file from the server with a success rate of 82.6% on average. The failure cases are due to two reasons. The first is the continuous communication as that in the SSH DoS attack. The second is that the attacker happens to begin the attack when the victim connection has been constructed while the victim has not logged in.

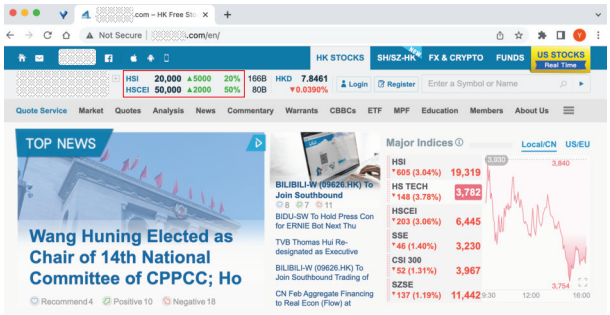


Fig. 4. Snapshots of web poisoning.

As for the HTTP injection attack, the average time costs of identifying source port, hijacking the connection, and the whole attack to inject forged HTTP responses are 9.4, 15.2, and 54.5 seconds, respectively, with a success rate of 76.1%. Compared to SSH DoS and FTP hijacking, HTTP injection is more time-consuming. First, during the connection hijacking phase, the attacker must complete not only Steps 1 and 2 (obtaining SEQ and ACK) but also Steps 3 and 4, which involve restoring the victim's original NAT mapping. Second, the attacker must wait for the appropriate opportunity to inject fake responses, often having to delay until the next request from the victim. Moreover, the attack has a lower success rate. First, after the attacker has replaced the NAT mapping, if the client sends a packet, the source port will be translated to a random port, which will incur a `RST` packet from the server and the client's connection will be closed. Second, the attacker fails to win the condition race of returning responses with the server, i.e., the client accepts the data from the server. Figure 4 shows the attack result against *www.ANONYMOUS.com*. The `HSI` index is manipulated to 20,000 with an increase of 5,000 and a growth rate of 20%. The same is true for the data of `HSCEI`. The attack may lead to wrong stock purchase or sale, affecting the financial status of the victim.

The experimental results of 30 Wi-Fi networks in our investigations are listed in Table III. As shown in the first row, an enterprise mode Wi-Fi network with the SSID of "Campus 1" (anonymized ethically) is located in a campus. The router is produced by the vendor of Huawei, and it supports the generation of Wi-Fi 6. We take the experiment of SSH DoS attack in this network, and it takes the attacker 15.43 seconds to terminate an SSH connection with a success rate of 90%.

VI. EXTENDED WORK

In this section, we verify and evaluate the feasibility of attacks in NAT-enabled IPv6 Wi-Fi networks.

A. Analysis of NAT66 and NAT64

With the continuous development of networking technologies, IPv6 is gradually replacing IPv4, particularly in modern home and enterprise networks. While IPv6 alleviates the issue of IPv4 address exhaustion by providing a larger address space, NAT is still widely utilized in IPv6 networks [43], [44], [45]. Considering significant differences between IPv6

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

TABLE III
EXPERIMENTAL RESULTS OF TCP ATTACKS IN 30 WI-FI NETWORKS

No.	Network Mode	SSID	Router Vendor	Wi-Fi Generation	WPA2/3 Enterprise/Personal	Attack Result	Time Cost (s)	Success Rate
1	Enterprise mode	Campus 1	Huawei	Wi-Fi 6	WPA2-Enterprise	SSH DoS	15.43	18/20
2	Enterprise mode	Campus 2	TP-Link	Wi-Fi 4	WPA2-Enterprise	FTP Hijacking	10.32	18/20
3	Enterprise mode	Campus 3	H3C	Wi-Fi 6	WPA2-Enterprise	HTTP Injection	48.87	15/20
4	Enterprise mode	Enterprise 1	TP-Link	Wi-Fi 6	WPA2-Enterprise	SSH DoS	11.56	16/20
5	Enterprise mode	Enterprise 2	TP-Link	Wi-Fi 5	WPA2-Enterprise	FTP Hijacking	11.43	18/20
6	Enterprise mode	Enterprise 3	Netcore	Wi-Fi 6	WPA2-Enterprise	HTTP Injection	87.20	15/20
7	Enterprise mode	Office building 1	TP-Link	Wi-Fi 5	WPA2-Enterprise	SSH DoS	9.56	18/20
8	Enterprise mode	Office building 2	iKuai	Wi-Fi 6	WPA2-Enterprise	FTP Hijacking	21.46	17/20
9	Enterprise mode	Office building 3	Mercury	Wi-Fi 6	WPA2-Enterprise	HTTP Injection	31.14	15/20
10	Enterprise mode	Hotel 1	Netcore	Wi-Fi 5	WPA2-Enterprise	SSH DoS	15.75	18/20
11	Enterprise mode	Hotel 2	D-Link	Wi-Fi 6	WPA2-Enterprise	FTP Hijacking	9.45	19/20
12	Enterprise mode	Hotel 2	iKuai	Wi-Fi 6	WPA2-Enterprise	HTTP Injection	71.32	16/20
13	Home mode	Restaurant 1	TP-Link	Wi-Fi 5	WPA2-Personal	SSH DoS	8.95	17/20
14	Home mode	Restaurant 2	Comfast	Wi-Fi 5	WPA2-Personal	FTP Hijacking	21.56	18/20
15	Home mode	Restaurant 3	Skyworth	Wi-Fi 6	WPA2-Personal	HTTP Injection	62.35	13/20
16	Home mode	Coffee shop 1	Mercury	Wi-Fi 4	WPA2-Personal	SSH DoS	8.98	17/20
17	Home mode	Coffee shop 2	TP-Link	Wi-Fi 4	WPA2-Personal	FTP Hijacking	9.29	18/20
18	Home mode	Coffee shop 3	Wavlink	Wi-Fi 5	WPA2-Personal	HTTP Injection	45.22	13/20
19	Home mode	Shopping mall 1	Tenda	Wi-Fi 6	WPA3-Personal	SSH DoS	24.23	18/20
20	Home mode	Shopping mall 2	TP-Link	Wi-Fi 4	WPA2-Personal	FTP Hijacking	11.44	19/20
21	Home mode	Shopping mall 3	Huawei	Wi-Fi 6	WPA3-Personal	HTTP Injection	78.44	15/20
22	Home mode	Bookstore 1	360	Wi-Fi 5	WPA2-Personal	SSH DoS	19.45	18/20
23	Home mode	Bookstore 2	Xiaomi	Wi-Fi 6	WPA3-Personal	FTP Hijacking	10.61	18/20
24	Home mode	Bookstore 3	H3C	Wi-Fi 6	WPA3-Personal	HTTP Injection	56.12	14/20
25	Home mode	Experience store 1	Xiaomi	Wi-Fi 6	WPA3-Personal	SSH DoS	16.97	17/20
26	Home mode	Experience store 2	Huawei	Wi-Fi 6	WPA3-Personal	FTP Hijacking	23.98	18/20
27	Home mode	Experience store 3	Xiaomi	Wi-Fi 5	WPA2-Personal	HTTP Injection	52.14	16/20
28	Home mode	Cinema 1	Ruijie	Wi-Fi 5	WPA2-Personal	SSH DoS	8.89	19/20
29	Home mode	Cinema 2	Mercury	Wi-Fi 6	WPA3-Personal	FTP Hijacking	11.31	18/20
30	Home mode	Cinema 2	Huawei	Wi-Fi 6	WPA3-Personal	HTTP Injection	54.26	16/20

and IPv4, it is crucial to evaluate the feasibility of attacks within NAT-enabled IPv6 Wi-Fi networks. In this work, we construct IPv6 networks of NAT66 [46] and NAT64 [47] and test whether the three requirements for the attack are satisfied, using the widely adopted OpenWrt 22.03.3 firmware as a NAT device.

- Port Allocation Strategy: We conducted tests to ascertain whether the router employs *port preservation* when configured with NAT66 and NAT64. This was achieved by transmitting IPv6-based TCP packets to a remote server and observing the packets received at the server. Notably, under both configurations, the remote server received TCP packets with the identical source port as the initial IPv6-based packets, thereby confirming that both NAT implementations adopt *port preservation* as their default setting.
- Reverse Path Validation: To verify the status of reverse path validation in NAT66 and NAT64, the attacker dispatches packets with a spoofed source address as the remote server to the router and observes whether it can receive it back. It is pertinent to mention that Linux does not provide the *rp_filter* option for IPv6, as elaborated in Section II-C. However, during our tests, the router still forwards the forged packets to the attacker without discarding them under both NAT configurations. This suggests that neither NAT64 nor NAT66 enables reverse path validation by default.
- TCP Window Tracking: To ascertain whether the router enforces TCP window tracking, the attacker sends IPv6-based RST packets with arbitrary sequence numbers to a TCP connection, aiming to clear the existing NAT mapping of the router. Under both NAT64 and NAT66 implementations, subsequent packets from the server fail to traverse the router after a 10-second timeout, which indicates that the NAT mapping has been cleared and that the router does not rigorously enforce TCP window tracking.

The experimental results demonstrate that NAT66 and NAT64 setups in our tests are vulnerable and satisfy all three conditions required for the TCP hijacking attack.

B. Implementation Details

We implement the attack using approximately 1200 lines of C++ code and 100 lines of Python code, leveraging the *Libtins* [48] and *Scapy* [49] library for crafting and sending forged packets. Both libraries support IPv4 and IPv6, enabling us to generate the required packet types for our experiments. To ensure the attack remains effective in IPv6 networks, we made several modifications to our implementation to accommodate key differences between IPv4 and IPv6:

- Adjustments for IPv6 Header Structure. IPv6 removes the *ttl* and *protocol* fields from its header, replacing them with the *hop_limit* field to indicate the maximum number of router hops and the *next_header* field to specify the transport protocol or an extension header. While this does not directly impact the attack methodology, we adjusted our packet crafting and parsing logic to correctly handle these changes in our implementation.
- Support for NAT64 and NAT66. The code has to be flexibly structured to support testing different NAT configurations of the routers. During the execution of an actual attack, there are distinctions in the packets sent in NAT64 and NAT66 environments. For NAT64, all packets

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

sent from the client to the server are IPv6 packets, while those from the server to the client are IPv4 packets. In contrast, for NAT66, all packets are expected to be IPv6.

- Attack Strategy Adaptation. In IPv4, the attacker can forge a `PUSH/ACK` packet to elicit an `ACK` packet containing the exact *SEQ* and *ACK* numbers (refer to Fig. 3). This method is equally viable in NAT66 scenarios. However, we have observed that TCP implementations associated with NAT64 exhibit slight differences. Specifically, this approach fails in NAT64 environments because when the attacker sends an IPv6-based `PUSH/ACK` packet, the server responds with an IPv4-based `RST` packet that contains the same incorrect sequence number as the forged `PUSH/ACK` packet. To address this limitation, we have revised our strategy by substituting the `PUSH/ACK` packet with an IPv6-based `SYN` packet, which prompts the server to generate a `Challenge ACK` containing the exact *SEQ* and *ACK* numbers. This modification enables us to successfully execute the attack within NAT64-enabled Wi-Fi networks.

To facilitate further research, we have made the complete attack codes openly available at https://github.com/Internet-Architecture-and-Security/TCP-Hijacking-in-NAT-Enabled-Wi-Fi-Networks/ for different NAT-enabled Wi-Fi networks.

C. Attack Evaluation

To evaluate the vulnerability within NAT-enabled IPv6 Wi-Fi networks and assess the practicality of TCP hijacking attacks, we conducted three sets of comparative experiments on the SSH DoS attack detailed in Section V-B targeting NAT4, NAT66, and NAT64. We set up the environment with our own devices, i.e., an OpenWrt-based Wi-Fi router, a laptop with Ubuntu 22.04 as the attacker and a laptop with Windows 11 as the victim client both connected to the router, and a remote server with Ubuntu 22.04. We repeat the tests 50 times to record the time costs and success rates.

Our findings indicate that, for inferring the source port, the average time for NAT4 and NAT66 is comparable, at approximately 8.66 seconds and 8.7 seconds, respectively. In contrast, NAT64 requires significantly more time, at 11.49 seconds. When obtaining sequence and acknowledgment numbers, NAT4 and NAT66 configurations necessitate 13.98 and 13.46 seconds, respectively, whereas NAT64 demands 14.92 seconds. The execution time for an SSH DoS attack remains relatively consistent across all three configurations, with NAT4, NAT66, and NAT64 taking 1.04, 1.07, and 0.97 seconds, respectively. When considering the total time, the NAT64 configuration exhibits a higher total time of 27.38 seconds, compared to 23.69 seconds for NAT4 and 23.24 seconds for NAT66. As for the success rate, the attack demonstrates a high success rate across all three NAT settings, achieving 96%, 94%, and 92%, respectively. These high success rates are attributed to the attacker's ability to continuously attempt the attack until a successful SSH DoS is achieved.

The higher time consumption in NAT64 can be attributed to its operational mechanism. NAT64 frequently leverages middleware solutions, such as *TAYGA* [50], for translating between IPv4 and IPv6 addresses. This translation process introduces additional overhead due to the requirement of multiple NAT procedures. Consequently, the efficiency of the attack diminishes in NAT64 compared to NAT4 and NAT66, where these additional translation steps are unnecessary.

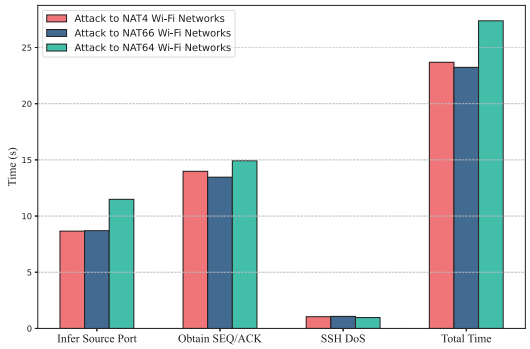


Fig. 5. Time costs of the attacks in different NAT environments.

These results show the efficiency of the attack on NAT4 and NAT66 networks, while revealing a slightly higher time consumption on NAT64 networks. Despite these differences, the attack remains feasible across all configurations, highlighting its potential impact on various NAT configurations.

VII. Discussion

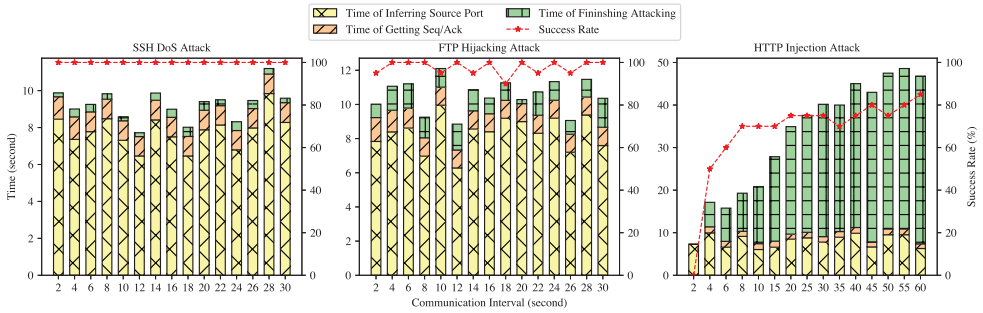
In this section, we discuss the factors that affect the attack's effectiveness and compare it with existing attacks in WLANs.

A. Factors Impacting the Attack

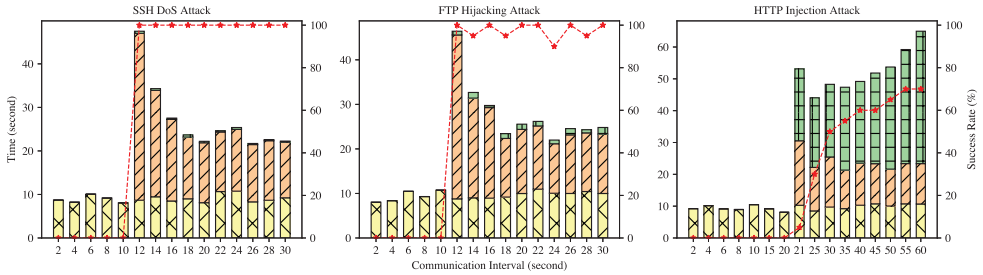
Impacts of Traffic Load. We analyze the possible impact from two aspects: 1) the bandwidth between the client and the server, and 2) the communication interval between them. First, we extend the experiments of the FTP hijacking attack with varied bandwidths (i.e., 10KBps, 100KBps, 1000KBps) and set the communication interval to 16 seconds with a NAT mapping timeout of 10 seconds. We repeat the experiment 50 times for each bandwidth. The experimental results show that the average time costs (i.e., 29.46 seconds, 28.78 seconds, 29.21 seconds) and success rates (i.e., 96%, 94%, 94%) remain largely unaffected since our attack mainly relies on the time interval left for the attacker to clean the NAT mappings. Second, we evaluate the three attacks under various communication intervals (e.g., 2 seconds, 4 seconds, 6 seconds, etc.) between the client and server, with a bandwidth of 100KBps. We repeat the experiments 20 times for each communication interval. The experimental results are shown in Figure 6.

Here, we take the FTP hijacking attack as an example. As shown in Figure 6(a), when the NAT mapping timeout is set to 1 second, if the communication interval is below 1 second, the attack will fail due to the continuously refreshed NAT mappings. When the communication interval is above 1 second, the attack can succeed with a high success rate (97.67%), where the small partial failures are due to the attacks being launched during the login phase of the FTP application. The average time cost is less affected by the communication interval, and it shows a fluctuating trend, which mainly depends on the time to infer the client's source port. Similarly, when the NAT

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.



(a) The time costs and success rates in different communication intervals when the timeout of the NAT mapping is 1 second.



(b) The time costs and success rates in different communication intervals when the timeout of the NAT mapping is 10 seconds.

Fig. 6. The time costs and success rates in different communication intervals and different NAT mapping timeouts.

mapping timeout is 10 seconds (as shown in Figure 6(b)), if the communication interval is below 10 seconds, the attack will fail due to the same reason. When the interval is above 10 seconds, the attack can succeed with a high success rate (97.5%). The average time cost shows a downward trend with the increment of communication interval, as the attacker can try fewer times and wait less time for the NAT mappings to be cleaned when the communication interval is longer.

Distribution of Time Cost and Failures. Besides the FTP hijacking attack, we also measure the time costs and failure reasons of the other two attacks, i.e., the SSH DoS attack and the HTTP injection attack. For the SSH DoS attack, as shown in Figure 6(a) of 1-second timeout, if the communication interval is below 1 second, the attack will fail. However, when the communication interval is larger than 1 second, the attack can always succeed. The average time cost is not very impacted with an average value of 9.24 seconds. The main cost is incurred by inferring the client's source ports. Figure 6(b) shows similar results when the NAT mapping timeout is 10 seconds. As for the HTTP injection attack, as shown in Figure 6(a) of 1-second timeout, if the communication interval is below 2 seconds, the attack will fail due to the continuously refreshed NAT mappings, or the connection will be terminated if the client sends a request when the attacker occupies the mapping. However, when it is larger than 2 seconds, with the increase of the communication interval, the time cost and success rate tend to increase. The time cost mainly depends on the waiting period until the client sends a request, as injections before that will not be accepted. In this way, the longer the communication interval is, the more waiting time will cost on average. The failures mainly come from two aspects. First, the original connection may also be terminated. Second, the attacker should inject data before the response of the server. Similar results can be found in Figure 6(b) of the 10-second timeout. The difference is that the time to get *SEQ* and *ACK* occupies a larger proportion compared with that of 1 second.

Impacts of NAT Mapping Timeout. We analyze the impact of NAT mapping timeout by comparing the results of the HTTP injection attack where we set the communication interval to 60 seconds, reflecting an actual client-server scenario in the real world. With a 1-second timeout, we have a time cost of 46.80 seconds and a success rate of 85%. However, we observe a time cost of 64.96 seconds and a success rate of 70% with a 10-second timeout. The reason is that a larger timeout value incurs longer probing overhead. For example, when the timeout is 10 seconds, the time to obtain *SEQ* and *ACK* takes up a larger proportion as the attacker must wait for the mappings to be completely cleared (i.e., at least 10 seconds). Under a long timeout, the connection is more likely to be unintentionally terminated, and thus the success rate is reduced. In summary, the increase of NAT mapping timeouts will incur increased time costs and decreased success rates.

Impacts of AP Isolation. AP isolation may influence some phases of our attack. With the policy enabled, the attacker cannot probe potential victim clients and reconstruct the original NAT mapping at the router by sending spoofed `ACK` packets to all of the potential clients when launching the HTTP injection attack. However, the SSH DoS and FTP hijacking attacks are not affected. Besides, only less than 10% (9 out of 93) of real-world Wi-Fi networks enforce AP isolation. We also enabled AP isolation on three routers (i.e., TP-Link TL-XDR6020, Linksys E5600, and Xiaomi RA81) in our laboratory and performed the two attacks. The experiment results show that the time cost and success rate are not affected by AP isolation.

Impacts of the Number of Clients. NAT mappings rely on the five-tuple (source IP, source port, destination IP, destination port, and protocol). Therefore, even in environments with

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

multiple clients (e.g., coffee shops or airports), port collisions among clients depends on their communication targets: (1) Different external servers. Since each client establishes a unique five-tuple, there is no source port conflict, and the attack's feasibility remain unaffected. (2) The same external server. The source port is randomly assigned by the client's operating system from a 65,535-port range, making the probability of two clients selecting the same source port relatively low. If clients select different source ports, the router will preserve each client's unique mapping. Consequently, the attack can still succeed. The only impact is an increase in attack time, as the attacker must probe the full source port space rather than stopping upon identifying a single active port. If clients select the same source port, the router will preserve only the first client's source port and assigns a new random port to other clients due to collision. As a result, our attack can only infer the first client's source port, but fails for subsequent clients whose ports were reassigned.

We also conducted experiments in a Wi-Fi network with single and multiple clients (10 clients), repeating each test multiple times: When clients connected to different external servers, our method successfully inferred the source port, and the attack time remained consistent. When multiple clients connected to the same server using different source ports, the attacker successfully inferred all source ports, but the total attack time increased. The average time to infer a single client's source port was 8.1 seconds, while scanning the entire space took 16.7 seconds. As expected, in the rare case of clients using the same source port, the attacker could only infer the first client's source port, confirming the analysis.

B. Comparison With Prior Attacks in Wi-Fi Networks

ARP Poisoning Attack. ARP poisoning attack in wired or wireless LANs has been well-researched, which can intercept traffic in both directions, i.e., from the victim client and the router, while our attack can only intercept TCP traffic from the router. However, Users can install some open-sourced tools [30], [31], [32] to prevent the attack. Besides, some routers (e.g., TP-Link) offer built-in ARP protections. Moreover, AP isolation can also defend against the ARP poisoning effectively by preventing communication between clients, while our attack is only partially affected as stated in Section IV-D and VII-A. We make a further empirical study in 10 real-world Wi-Fi networks. Three of them have enabled AP isolation and the ARP poisoning attack can only succeed in the other networks. However, we observe that it fails when we enable protections on routers and client devices (e.g., using the tool developed in [32]). In contrast, our attack can still succeed in these networks, even when these protections are in place.

Eavesdropping Attack. Against WPA2-Personal mode Wi-Fi networks, a malicious attacker who knows the pre-shared key can sniff frames in the air. If it wants to decrypt the frames, it needs to capture the 4-way handshake frames when other clients are connecting to the network. Though the attacker can force the victim client to be detached from the current AP [51] and wait for its re-connection, the attack is perceivable by the victim that its device will lose Internet access. However, our attack is stealthier as the attacker only needs to connect to the same network and does not need to make the client disconnect from the existing network. Besides, it's much harder to decrypt frames encrypted with WPA2-Enterprise or WPA3-Personal, while our attack can also influence these networks.

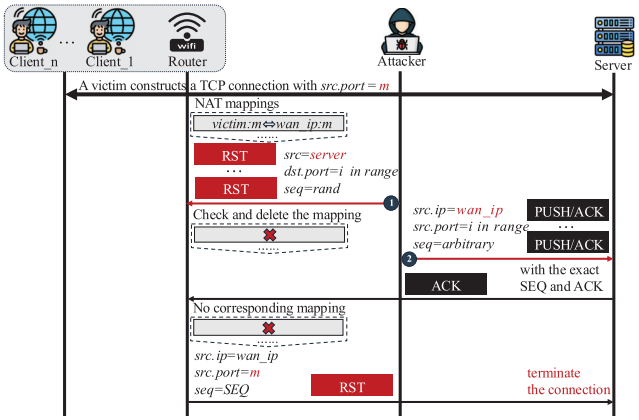


Fig. 7. Remote TCP DoS attack.

Rogue AP Attack. The malicious insider who knows the pre-shared key can also create a rogue clone of the network and intercept the traffic of innocent users connected to it [7], [27]. This attack requires broadcasting the same SSID, which can be detected by the network administrator, and some routers provide Rogue AP detection [28], [29]. In contrast, our attack is stealthier as there is no specific strategy to detect it. In addition, a lightweight device compromised by the attacker remotely may not have enough resources to provide the services as a rogue AP. The attacker who is physically in the LAN can set its own device as a rogue AP, but it has to provide a stronger signal than the original AP, and thus the influence is limited to clients in close proximity. Conversely, our attack does not face the signal race that any device in the same Wi-Fi network can launch the attack and potentially influence all clients. Besides, enterprise mode Wi-Fi networks can protect clients from the Rogue AP attack. However, our empirical measurements have revealed that our attack can compromise the traffic within 22 different enterprise mode Wi-Fi networks.

C. Extending the Attack Model

In our extended model, we eliminate the requirement that the attacker and the victim client have to be located in the same Wi-Fi network. Instead, we demonstrate that a remote attacker from the Internet can launch a DoS attack on TCP connections between victim clients behind a vulnerable router and an external server. We require that the attacker can send packets with spoofed source IP addresses, which is a practical assumption considering that approximately a quarter of autonomous systems still do not employ source address validation (SAV), as reported by the Spoofer project [52].

Assuming that there is a live connection between an outside server and the victim client who resides behind a vulnerable router. The outside attacker cannot infer the source port of the victim client anymore using the method before. However, as we show in Figure 7, the attacker can send forged TCP `RST` packets covering the entire source port space to the vulnerable router. As the routers do not check the sequence number

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

strictly, these RST packets can easily bypass routers' checks to clean the NAT mapping of the victim connection. After the NAT mapping disappears, the attacker can send forged TCP data packets to the server with a spoofed source IP address of the router, covering all possible source ports, too. The server will respond to the matched one with an ACK packet specified with the current sequence and acknowledgment numbers to the router. However, as there is no corresponding NAT mapping anymore, the router will just send a RST packet back to the server with the sequence number received just now. Then the connection will be terminated from the server side. If the client continues communicating with the server, it will receive a TCP RST packet back from the server afterward. In this way, the attacker can interfere with TCP communications between the victim and server, causing a DoS attack.

The attacker needs to detect such victim clients who access the Internet through these vulnerable routers. We find that it is of great convenience for the attacker to identify these vulnerable routers through open search engines [53], [54], which contain a large amount of publicly accessible devices (e.g., routers, web servers, and webcams). For example, millions of TP-Link routers with public IP addresses can be found through FOFA [54]. We estimate that there are tens of millions of vulnerable routers existing in the world which may be influenced by the attack, and we believe this attack is promising and practical and may affect many more users. We leave it as future work to validate the real-world impact in practice owing to reasons such as ethical considerations.

VIII. Countermeasures

Responsible Vulnerability Disclosure. We have reported the issue to the affected manufacturers by submitting vulnerability reports and contacting them via email. At the time of writing, we have received positive responses from the OpenWrt community that confirms our findings and has released patches to fix the vulnerability, and seven router vendors (i.e., TP-Link, Huawei, Xiaomi, 360, Mercury, Ubiquiti, and Linksys) that have all acknowledged our reports and are trying to repair their products. In addition, we have been assigned 10 CVE numbers for the vulnerability in different vendors (i.e., TP-Link, Linksys, Mercury, Ruijie, D-Link, Comfast, H3C, OpenWrt, Wavlink, and 360). The other vendors are still investigating the vulnerability. We also provide them with countermeasure suggestions to mitigate the identified attack. As mentioned in Section III, we outline several conditions that characterize a vulnerable router implementation. Intuitively, any breach of these conditions will render the attack ineffective.

Random Port Allocation. The first solution is for the router to use the *random selection* strategy when creating new NAT mappings. With this strategy, the attacker cannot identify whether the port has been used by other internal hosts, and the attack will be foiled. It should be noted that some TCP punch-through schemes (e.g., TCP simultaneous open) may be influenced by *random selection* as they rely on port prediction [55]. Alternatively, clients can utilize some other common-used schemes (e.g., TURN relaying) for NAT punch-through, which will not be affected [56].

Reverse Path Validation. Another effective measure to prevent the attack is to adopt the RFC 3704 recommendation, which suggests using the strict mode to filter out forged packets. In our test, some routers have taken this recommendation by default and can defend against the attack. However, this strategy may introduce additional performance overhead and impact the reliability of networking for certain applications (e.g., OpenVPN running on the router may be affected as the reverse path validation may affect packet delivery [57]).

TCP Window Tracking. As a middle device between the internal clients and outside servers, the router has to keep the necessary information about connections. It is essential for routers to record the current the sequence and acknowledgment numbers and strictly check them for received packets. The OpenWrt community has implemented this mitigation as they believe the performance impact should not matter anymore on any currently supported hardware.

IX. Related Work

Traffic hijacking has been widely studied, and lots of attacks have been proposed. Vulnerabilities that lead to traffic hijacking may exist in protocols at all levels of the TCP/IP protocol stack. For instance, in the same LAN, an attacker can exploit the vulnerability of the ARP protocol to hijack network traffic by sending fake ARP packets and compromising the victim device's ARP cache, which allows the attacker to intercept, modify, or even discard the traffic of victims, thus hijacking the victims' traffic completely [8].

At the IP layer, attackers may leverage the ICMP redirect mechanism to hijack victims' traffic by placing themselves in the man-in-the-middle position [58], [59]. Recently, Feng et al. developed a new method to circumvent the ICMP redirect legitimacy checks in Wi-Fi networks and presented an attack to evade the security mechanisms of WPAs [36]. However, the attack targets out-of-date systems (e.g., iOS 1-8, Android before 10.0) except for the latest versions of Linux and FreeBSD. Besides, they also showed that off-path attackers from the Internet could trick public servers into redirecting their traffic to neighboring hosts with forged ICMP redirect messages, thus causing a DoS attack [60].

DNS cache poisoning attacks can also be abused to hijack traffic. In the same LAN, Herzberg and Shulman proposed three methods to circumvent source port randomization, which leverages the port allocation strategies used by NAT devices [19]. Zheng et al. developed an attack targeting DNS forwarders (e.g., home routers) by forcing fragmentation using attacker-owned authoritative name servers [61]. Man et al. proposed that a purely off-path attacker from the Internet can exploit the side channel in ICMP rate limit or the limited space for storing the next hop exception cache to infer the source ports of DNS requests and poison DNS caches maliciously [62], [63].

To hijack TCP connections so as to inject forged TCP segments into the target connection or terminate it, attackers mainly rely on various side-channel vulnerabilities. Cao et al. demonstrated that a global shared variable used in the challenge ACK mechanism could be abused for an off-path attacker to manipulate the victim TCP traffic [16]. Moreover,

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

Cao et al. further analyzed TCP side channels with model checking systematically to uncover new TCP off-path vulnerabilities [64]. Chen and Qian showed that a timing side channel that exists in half-duplex IEEE 802.11 or Wi-Fi technology [42] and Feng et al. discovered a side channel in the mixed IPID assignment [17], [65], which can also be exploited to manipulate TCP traffic by off-path attackers. Tolley et al. demonstrated that blind in/on-path attackers could learn the virtual IP of a host behind a VPN and hijack TCP connections supposedly protected by the tunnel [6]. Besides, Schepers et al. discovered that modern operating systems fail to manage the security context of their transmit queues securely, thereby allowing a malicious attacker to intercept frames in Wi-Fi networks, thus hijacking TCP connections [4].

Recently, Mixon-Baca et al. also exploited the shared nature of VPN connection tracking frameworks to detect or manipulate encrypted traffic from other VPN clients [66]. They leveraged conflicts between a VPN server's listening port and an attacker's source port to infer whether a client has connected to the VPN server. In contrast, our attack leverages source port conflicts among clients to determine whether any client is communicating with an external server. Besides, our attack exploits the vulnerability in router's lack of TCP window tracking to hijack TCP connections. Both works emphasize the need for stronger NAT isolation mechanisms to prevent session manipulation attacks.

Fortunately, most of the prior vulnerabilities have already been addressed [11], [16], [65], and the security community has developed corresponding defense measures [67], [68], [69]. However, we present a new type of TCP traffic hijacking attack leveraging the vulnerabilities in routers, which can circumvent traditional defenses against TCP traffic hijacking attacks and lead to new challenges for the security communities.

X. Conclusion

In this paper, we uncover a new off-path TCP hijacking attack in the Wi-Fi networks that leverages vulnerable routers. We find that a malicious insider can abuse the NAT port preservation strategy and insufficient reverse path validation strategy of the router to infer the existence of TCP connections from the LAN to a remote server and then obtain the sequence and acknowledgment numbers by manipulating the state of NAT mappings with forged reset packets due to the vulnerable routers disabling TCP window tracking strategy. We confirm the vulnerability in a wide range of routers from different manufacturers and evaluate the new attack in different scenarios, such as SSH DoS, FTP hijacking, and HTTP injection in various Wi-Fi networks. We have also verified and evaluated the attack in NAT-enabled IPv6 Wi-Fi networks. Finally, we suggest countermeasures, report the vulnerabilities to the affected manufacturers, and have received positive acknowledgments.

References

[1] 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.

[2] M. Vanhoef and F. Piessens, "Release the Kraken: New KRACKs in the 802.11 standard," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 3–299.

[3] M. Vanhoef and E. Ronen, "Dragonblood: Analyzing the dragonfly handshake of WPA3 and EAP-pwd," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 517–533.

[4] D. Schepers, A. Ranganathan, and M. Vanhoef, "Framing frames: Bypassing Wi-Fi encryption by manipulating transmit queues," in *Proc. USENIX Secur. Symp.*, 2023, pp. 53–68.

[5] M. Vanhoef, "Fragment and forge: Breaking Wi-Fi through frame aggregation and fragmentation," in *Proc. USENIX Secur. Symp.*, Jan. 2021, pp. 161–178.

[6] W. J. Tolley, B. Kujath, M. T. Khan, N. Vallina-Rodríguez, and J. R. Crandall, "Blind in/on-path attacks and applications to VPNs," in *Proc. USENIX Secur. Symp.*, Aug. 2021, pp. 3129–3146.

[7] A. M. Alsahlany, A. R. Almusawy, and Z. H. Alfatlawy, "Risk analysis of a fake access point attack against Wi-Fi network," *Int. J. Sci. Eng. Res.*, vol. 9, pp. 322–326, May 2018.

[8] S. Hijazi and M. S. Obaidat, "Address resolution protocol spoofing attacks and security approaches: A survey," *Secur. Privacy*, vol. 2, no. 1, Jan. 2019, Art. no. e49.

[9] T. Ohigashi and M. Morii, "A practical message falsification attack on WPA," in *Proc. JWIS*, 2009, pp. 1–66.

[10] M. Vanhoef, P. Adhikari, and C. Pöpper, "Protecting Wi-Fi beacons from outsider forgeries," in *Proc. 13th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, Jul. 2020, pp. 155–160.

[11] S. Y. Nam, S. Jurayev, S.-S. Kim, K. Choi, and G. S. Choi, "Mitigating ARP poisoning-based man-in-the-middle attacks in wired or wireless LAN," *EURASIP J. Wireless Commun. Netw.*, vol. 2012, no. 1, pp. 1–17, Dec. 2012.

[12] B. Ford, S. Guha, K. Biswas, S. Sivakumar, and P. Srisuresh, *NAT Behavioral Requirements for TCP*, document RFC 5382, Oct. 2008.

[13] Y. Gilad and A. Herzberg, "Off-path TCP injection attacks," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 4, pp. 1–32, Apr. 2014.

[14] Z. Qian and Z. M. Mao, "Off-path TCP sequence number inference attack—How firewall middleboxes reduce security," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 347–361.

[15] Z. Qian, Z. M. Mao, and Y. Xie, "Collaborative TCP sequence number inference attack: How to crack sequence number under a second," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2012, pp. 593–604.

[16] Y. Cao, Z. Qian, Z. Wang, T. Dao, S. V. Krishnamurthy, and L. M. Marvel, "Off-path TCP exploits: Global rate limit considered dangerous," in *Proc. USENIX Secur. Symp.*, Jan. 2016, pp. 209–225.

[17] X. Feng, C. Fu, Q. Li, K. Sun, and K. Xu, "Off-path TCP exploits of the mixed IPID assignment," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1323–1335.

[18] M. Kol, A. Klein, and Y. Gilad, "Device tracking via Linux's new TCP source port selection algorithm," in *Proc. USENIX Secur. Symp.*, 2023, pp. 6167–6183.

[19] A. Herzberg and H. Shulman, "Security of patched DNS," in *Computer Security—ESORICS 2012*. Berlin, Germany: Springer, 2012.

[20] D. Senie and P. Ferguson, *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing*, document RFC 2827, May 2000.

[21] F. Baker and P. Savola, *Ingress Filtering for Multihomed Networks*, document RFC 3704, Mar. 2004.

[22] J. Postel, *Transmission Control Protocol*, document RFC 793, Sep. 1981.

[23] W3Techs. *Usage Statistics of Default Protocol HTTPS for Websites*. Accessed: Mar. 2024. [Online]. Available: https://w3techs.com/technologies/details/ce-httpsdefault

[24] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig, "NATBLASTER: Establishing TCP connections between hosts behind NATs," in *Proc. ACM SIGCOMM ASIA Workshop*, 2005, pp. 1–3.

[25] M. Holdrege and P. Srisuresh, *IP Network Address Translator (NAT) Terminology and Considerations*, document RFC 2663, Aug. 1999.

[26] Linux. *Rp_filter*. Accessed: Mar. 2023. [Online]. Available: https://sysctl-explorer.net/net/ipv4/rp_filter/

[27] R. Orsi. *Understanding Evil Twin AP Attacks and How to Prevent Them*. Accessed: Mar. 2023. [Online]. Available: https://www.darkreading.com/attacks-breaches/understanding-evil-twin-ap-attacks-and-how-to-prevent-them

[28] Huawei. *Rogue Device Detection*. Accessed: Jul. 2023. [Online]. Available: https://support.huawei.com/enterprise/en/doc/EDOC1100096321/3eb0a62e/example-for-configuring-rogue-device-detection-and-containment

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.

[29] Linksys. *How to Enable Rogue AP Detection on Your LinkSys Wireless-AC Access Point*. Accessed: Jul. 2023. [Online]. Available: https://www.linksys.com/support-article?articleNum=135793
[30] A. Chirila. *ARP Antispoofer*. Accessed: Jul. 2023. [Online]. Available: https://www.softpedia.com/get/Security/Firewall/ARP-AntiSpoofer.shtml
[31] *ShARP*. Accessed: Jul. 2023. [Online]. Available: https://github.com/europa502/shARP
[32] 360-ARP. *360 Total Security: Free Antivirus Protection for Home and Devices*. Accessed: Jul. 2023. [Online]. Available: http://www.360totalsecurity.com/en/
[33] D. S. M. Vanhoef. *Macstealer: Wi-Fi Client Isolation Bypass*. Accessed: Jul. 2023. [Online]. Available: https://github.com/vanhoefm/macstealer#id-test-isolation
[34] pfSense. *Pfsense—World's Most Trusted Open Source Firewall*. Accessed: Jul. 2023. [Online]. Available: https://www.pfsense.org/
[35] Nmap. *The Network Mapper*. Accessed: Mar. 2023. [Online]. Available: https://nmap.org/
[36] X. Feng, Q. Li, K. Sun, Y. Yang, and K. Xu, "Man-in-the-middle attacks without rogue AP: When WPAs meet ICMP redirects," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 3162–3177.
[37] P. Richter et al., "A multi-perspective analysis of carrier-grade NAT deployment," in *Proc. Internet Meas. Conf.*, Nov. 2016, pp. 215–229.
[38] B. Ford and P. Srisuresh, *Unintended Consequences of NAT Deployments With Overlapping Address Space*, document RFC 5684, Feb. 2010.
[39] Traceroute. *Linux Manual Page*. Accessed: Mar. 2023. [Online]. Available: https://man7.org/linux/man-pages/man8/traceroute.8.html
[40] Ping. *Linux Manual Page*. Accessed: Mar. 2023. [Online]. Available: https://man7.org/linux/man-pages/man8/ping.8.html
[41] V. Mahajan and S. K. Peddoju, "Deployment of intrusion detection system in cloud: A performance-based study," in *Proc. IEEE Trustcom/BigDataSE/ICESS*, Aug. 2017, pp. 1103–1108.
[42] W. Chen and Z. Qian, "Off-path TCP exploit: How wireless routers can jeopardize your secrets," in *Proc. USENIX Secur. Symp.*, Jan. 2018, pp. 1581–1598.
[43] E. Boswell, S. McQuistin, C. Perkins, and S. Strowes, "Measuring NAT64 usage in the wild," 2024, *arXiv:2402.14632*.
[44] H. Scott. *You Thought There Was No*. Accessed: Jul. 2024. [Online]. Available: https://blogs.infoblox.com/ipv6-coe/you-thought-there-was-no-nat-for-ipv6-but-nat-still-exists/
[45] A. Alejandro and A. Alejandro. *Detecting and Measuring IPV4 and IPV6 NAT*. Accessed: Jul. 2024. [Online]. Available: https://www.lacnic.net/innovaportal/file/2675/1/measuring-nat.pdf
[46] F. Baker and M. Cullen, *IPv6-to-IPv6 Network Prefix Translation*, document RFC 6296, Jun. 2011.
[47] P. Matthews, I. van Beijnum, and M. Bagnulo, *Stateful NAT64: Network Address and Protocol Translation From IPv6 Clients to IPv4 Servers*, document RFC 6146, Apr. 2011.
[48] Libtins. (2023). *Libtins: Packet Crafting and Sniffing Library*. [Online]. Available: https://libtins.github.io/
[49] Scapy. *Packet Crafting for Python2 and Python3*. Accessed: Mar. 2023. [Online]. Available: https://scapy.net/
[50] N. Lutchansky. (2024). *Tayga: Simple, No-Fuss Nat64 for Linux*. [Online]. Available: http://www.litech.org/tayga/
[51] D. Schepers, A. Ranganathan, and M. Vanhoef, "On the robustness of Wi-Fi deauthentication countermeasures," in *Proc. WiSec*, May 2022, pp. 245–256.
[52] Q. Lone, A. Frik, M. Luckie, M. Korczynski, M. van Eeten, and C. Gañán, "Deployment of source address validation by network operators: A randomized control trial," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 2361–2378.
[53] SHODAN. *Search Engine for the Internet of Everything*. Accessed: Mar. 2023. [Online]. Available: https://www.shodan.io/
[54] FOFA. *FOFA, Search Engine*. Accessed: Mar. 2023. [Online]. Available: https://en.fofa.info/
[55] Z. Yongjun and Z. Shiquan, "NAT hole punching based on simultaneous TCP open," in *Proc. 5th Int. Conf. Multimedia Inf. Netw. Secur.*, 2013, pp. 235–238.
[56] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators," in *Proc. ATC*, Apr. 2005, pp. 1–13.
[57] OpenVPN. *Concepts-Policyrouting-Linux*. Accessed: Jul. 2023. [Online]. Available: https://community.openvpn.net/openvpn/wiki/Concepts-PolicyRouting-Linux
[58] C. Low. *ICMP Attacks Illustrated*. Accessed: Mar. 2023. [Online]. Available: https://www.sans.org/reading-room/whitepapers/threats/paper/477
[59] A. Ayer. *ICMP Redirect Attacks in the Wild*. Accessed: Mar. 2023. [Online]. Available: https://www.agwa.name/blog/post/icmp_redirect_attacks_in_the_wild
[60] X. Feng et al., "Off-path network traffic manipulation via revitalized ICMP redirect attacks," in *Proc. USENIX Secur. Symp.*, 2022, pp. 2619–2636.
[61] X. Zheng et al., "Poison over troubled forwarders: A cache poisoning attack targeting DNS forwarding devices," in *Proc. USENIX Secur. Symp.*, Jan. 2020, pp. 577–593.
[62] K. Man, Z. Qian, Z. Wang, X. Zheng, Y. Huang, and H. Duan, "DNS cache poisoning attack reloaded: Revolutions with side channels," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 1337–1350.
[63] K. Man, X. Zhou, and Z. Qian, "DNS cache poisoning attack: Resurrections with side channels," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 3400–3414.
[64] Y. Cao, Z. Wang, Z. Qian, C. Song, S. V. Krishnamurthy, and P. Yu, "Principled unearthing of TCP side channel vulnerabilities," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 211–224.
[65] X. Feng, Q. Li, K. Sun, C. Fu, and K. Xu, "Off-path TCP hijacking attacks via the side channel of downgraded IPID," *IEEE/ACM Trans. Netw.*, vol. 30, no. 1, pp. 409–422, Feb. 2022.
[66] B. Mixon-Baca et al., "Attacking connection tracking frameworks as used by virtual private networks," in *Proc. PETS*, vol. 2024, Jun. 2024, pp. 109–126.
[67] J. McCann, S. Deering, and J. Mogul, *Path MTU Discovery for IP Version 6*, document RFC 1981, Aug. 1996.
[68] J. Mogul and S. Deering, *Path MTU Discovery*, document RFC 1191, Nov. 1990.
[69] M. Lepinski and K. Sriram, *BGPsec Protocol Specification*, document RFC 8205, IETF, Sep. 2017.

Yuxiang Yang is currently pursuing the Ph.D. degree with Tsinghua University. His research interests include network security.

Xuewei Feng received the Ph.D. degree from Tsinghua University in 2022. His research interests include network security and software vulnerability detection.

Qi Li (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University. He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include Internet and cloud security. He is an Editorial Board Member of IEEE TDSC and ACM TOPS.

Kun Sun (Member, IEEE) received the Ph.D. degree in computer science from North Carolina State University. He has more than 15 years working experience in both industry and academia, and serves as the Director of the Sun Security Laboratory (SunLab) and the Associate Director of the Center for Secure Information Systems (CSIS). He has published more than 100 peer-reviewed conference and journal articles. His research focuses on systems and network security.

Ziqiang Wang is currently pursuing the Ph.D. degree with Southeast University. His research interests include network architecture and security.

Ao Wang (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with Southeast University. His research interests include network security.

Ke Xu (Fellow, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China. He is currently a Full Professor with the Department of Computer Science and Technology, Tsinghua University. He has published more than 200 technical articles in the research areas of next generation internet, blockchain systems, and network security. He won the IWQoS 24 Best Paper Award and the Distinguished Paper Award at USENIX Security 23, USENIX Security 24, and NDSS 25.

Authorized licensed use limited to: Tsinghua University. Downloaded on June 30,2026 at 07:01:44 UTC from IEEE Xplore. Restrictions apply.