ReDAN: An Empirical Study on Remote DoS Attacks against NAT Networks

Xuewei Feng*, Yuxiang Yang*, Qi Li*†, Xingxiang Zhan†, Kun Sun‡, Ziqiang Wang§,
Ao Wang§, Ganqiu Du¶, Ke Xu*†✉
*Tsinghua University, †Zhongguancun Laboratory, ‡CSIS, George Mason University
§Southeast University, ¶China Software Testing Center
fengxw06@126.com, yangyx22@mails.tsinghua.edu.cn, qli01@tsinghua.edu.cn, zhansingsong@gmail.com,
ksun3@gmu.edu, {ziqiangwang, wangao}@seu.edu.cn, duganqiu@cstc.org.cn, xuke@tsinghua.edu.cn

***Abstract*—In this paper, we conduct an empirical study on remote DoS attacks targeting NAT networks (ReDAN, short for Remote DoS Attacks targeting NAT). We show that Internet attackers operating outside local NAT networks possess the capability to remotely identify a NAT device and subsequently terminate TCP connections initiated from the identified NAT device to external servers. Our attack involves two steps. First, we identify NAT devices on the Internet by exploiting inadequacies in the Path MTU Discovery (PMTUD) mechanism within NAT specifications. This deficiency creates a fundamental side channel that allows Internet attackers to distinguish if a public IPv4 address serves a NAT device or a separate IP host, aiding in the identification of target NAT devices. Second, we launch a remote DoS attack to terminate TCP connections on the identified NAT devices. While recent NAT implementations may include protective measures, such as packet legitimacy validation to prevent malicious manipulations on NAT mappings, we discover that these safeguards are not widely adopted in real world. Consequently, attackers can send crafted packets to deceive NAT devices into erroneously removing innocent TCP connection mappings, thereby disrupting the NATed clients to access remote TCP servers. Our experimental results reveal widespread security vulnerabilities in existing NAT devices. After testing 8 types of router firmware and 30 commercial NAT devices from 14 vendors, we identify vulnerabilities in 6 firmware types and 29 NAT devices that allow off-path removal of TCP connection mappings. Moreover, our measurements reveal a stark reality: 166 out of 180 (over 92%) tested real-world NAT networks, comprising 90 4G LTE/5G networks, 60 public Wi-Fi networks, and 30 cloud VPS networks, are susceptible to exploitation. We responsibly disclosed the vulnerabilities to affected vendors and received a significant number of acknowledgments. Finally, we propose our countermeasures against the identified DoS attack.**

I. Introduction

Network Address Translation (NAT) is a popular technique to map IP addresses between a private realm and the public realm, thus enabling hosts within a private network to transparently access hosts in the external network [46], [45]. Due to space exhaustion of IPv4 addresses, NAT is widely used in various network scenarios, e.g., 4G LTE/5G networks, cloud VPS networks, public Wi-Fi networks, and IoT networks, to condense multiple local private addresses into a public one. According to CAIDA's investigations, more than 23% Autonomous Systems (ASes) use NAT to conserve public IPv4 addresses and the proportion keeps increasing [24]. Moreover, it is widely believed that NAT offers enhanced security [42], [33], [35], [2], since NAT serves as an added security measure for private networks by concealing the actual IP addresses of internal hosts. This prevents direct exposure of the internal hosts to Internet attackers.

In this paper, we undertake a comprehensive empirical study to demonstrate that real-world NAT implementations may exhibit vulnerabilities, which can be exploited by off-path attackers on the Internet to pose a substantial threat to end-to-end communication connectivity. Particularly, by exploiting these vulnerabilities in various NAT devices (e.g., NAT gateways in public Wi-Fi networks or PDN gateways/UPF devices in 4G LTE/5G networks), we demonstrate that off-path attackers operating outside local NAT networks can launch remote DoS attacks against the NAT network (i.e., the network segment linked to the Internet through the NAT device) to cut off TCP connections initiated by the NATed clients to an external server. This identified DoS attack can occur even when the internal NATed clients have a robust TCP/IP implementation and are free from DoS vulnerabilities. Our attack consists of two main steps, namely, i) identifying NAT devices on the Internet and ii) remotely severing TCP connections on the NAT devices.

We reveal that NAT specifications [45], [6] inadequately address the Path MTU Discovery (PMTUD) mechanism [29], [27], thus creating a side channel exploitable by off-path attackers on the Internet. This side channel allows the attackers to distinguish whether a public IPv4 address belongs to a NAT device with multiple clients or a separate IP host, i.e., pinpointing NAT devices on the Internet. As per PMTUD specifications, the path MTU value is maintained at the IP layer of the originator, limiting packet sizes (e.g., TCP, ICMP, UDP) sent to the destination. However, in NAT networks, ICMP packets generated by NAT devices may not align with the path MTU value maintained by Internal clients, despite originating from the same source (i.e., sharing the same public source IP address). This misalignment results in desynchronization

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.230972
www.ndss-symposium.org

and information leakage. By observing disparities in the sizes of the received TCP and ICMP packets, a remote server can discern whether the client is a private host within a NAT network or a separated IP host. Consequently, the server can pinpoint NAT devices with public IP addresses matching the source IP addresses of the received packets.

Upon identifying NAT devices on the Internet, we proceed to initiate our remote DoS attack, thereby terminating TCP connections originating from the NATed clients behind these devices. We discover that, despite the incorporation of specific protective measures into recent NAT implementations, such as validating the legitimacy of received `RST` packets and preventing malicious removal of NAT mappings for corresponding TCP connections (e.g., in the NAT implementation of Netfilter within Linux 5.1 and beyond [49]), these security measures have not been widely adopted in various real-world downstream NAT devices, including NAT gateways in public Wi-Fi networks, 4G LTE/5G networks, and cloud networks. As a result, off-path attackers on the Internet may send crafted `RST` packets without an exact sequence number to the identified NAT device, tricking it into erroneously removing the maintained mappings for TCP connections. This, in turn, effectively disrupts the NATed clients' access to external servers and facilitates a remote DoS attack.

We conduct extensive evaluations on our attack. We first conduct end-to-end evaluations to assess the identified vulnerabilities. The side channel for identifying NAT devices, stemming from inadequate PMTUD considerations in NAT specifications, affects all NAT implementations in our tests, including those within 6 types of native OSes, 8 types of router firmware, and 30 commercial NAT routers. **B**y comparing with prior works [47], [48] in 21 different network configurations, we demonstrate that our identification method excels. Regarding the vulnerability of removing NAT mappings via crafted `RST` packets, we investigate NAT implementation disparities among native OSes, various router firmware, and commercial routers. We show that some NAT implementations within native OSes (e.g., Netfilter in Linux 5.1 and beyond) may have defenses, whereas others (e.g., FreeBSD), particularly the majority of the router firmware types we tested (6 out of 8) and commercial NAT routers (29 out of 30) remain vulnerable. Moreover, we demonstrate that, using a bandwidth of 5.72MBps, an off-path attacker can block all internal clients behind the vulnerable NAT device from connecting to a remote SSH server or downloading files from an FTP server.

In addition to end-to-end evaluations, we conduct real-world assessments to demonstrate that our attack could cause significant damage on the Internet. Over an 11-month period, we identify more than 7,600 public IPv4 addresses used by NAT devices on the Internet by leveraging the identified side channel in PMTUD. These identified NAT devices are distributed across 1,289 Autonomous Systems (AS) in 124 countries around the world. Besides, we conduct evaluations on 180 actual NAT networks, including various NAT scenarios such as public Wi-Fi networks, 4G LTE/5G networks, and cloud networks. The experimental results show that out of the 180 NAT networks, 166 are vulnerable to our attacks, causing a vulnerable proportion of more than 92%.

Finally, we present potential countermeasures. We propose enhancing NAT specifications to fix the side channel issue. This involves requiring NAT devices to both translate path MTU update messages to internal clients and synchronize their own path MTU values based on these messages. This measure ensures the consistency of the path MTU value from the same source IP address, effectively preventing information leakage and thwarting attackers' attempts to identify NAT devices on the Internet. Furthermore, we recommend that NAT devices implement stricter legitimacy checks on received TCP packets. Particularly, verifying the sequence number of the received `RST` packets is one possible countermeasure to foil the attacker's removal on the preserved session mappings for TCP connections, thus throttling the identified DoS attack. Our prototype based on OpenWrt 22.03 confirms the effectiveness of the countermeasure.

Contributions. Our main contributions are as follows:

- We unveil a fundamental side channel in NAT specifications that can be exploited to identify NAT devices on the Internet, causing information leakage of NAT networks.
- We investigate NAT implementation disparities among native OSes, various router firmware, and commercial routers. Our study shows that NAT implementations within downstream router firmware and commercial routers are prone to manipulation of TCP connection mappings, resulting in a DoS attack.
- Our empirical study reveals that over 92% (166 out of 180) of real-world NAT networks, including public Wi-Fi networks, 4G LTE/5G networks, and cloud networks, are vulnerable to the identified DoS attack.
- We analyze the root cause and propose countermeasures to throttle the attack. The experiments confirm the effectiveness of our countermeasures.

Ethical Considerations. In this paper, we perform two types of experiments on the Internet to validate the feasibility and significance of our attacks in the real world. Specifically, we focus on identifying NAT devices on the Internet and discovering real-world NAT networks vulnerable to our DoS attack (see §VII for more details about the two types of experiments). Ethical considerations are given top priority during the experimentation process. First, during identifying NAT devices on the Internet, we deploy 7 vantage points (HTTP servers) around the world. Whenever one of our vantage points receives an HTTP request, we use our method (see §IV) to determine whether the request is from a separate IP host or a NATed host. First of all, we obtained the requester's approval before conducting the identification. We state the purpose and the details of our experiment clearly in the HTTP page. The experiment will only proceed after the requester agrees. Specifically, our identification will not pose any security risks to the requester. **W**e only analyze the requester's packet size and record the requester's source IP address. These data are handled with the utmost care. The only side effect of our

experiment is that we will modify the path MTU value of the requester—this adjustment does not impact the network segment where the requester resides, as the ICMP message used to modify the path MTU value is only received by the requester. Moreover, after the experiment, we mitigate this side effect by sending another ICMP message to restore the requester's path MTU value preserved for our vantage point.

Second, while discovering vulnerable NAT networks in the real world, we only test TCP connections under our control. After the approval of accessing the target NAT network, we set our machine as the TCP client and rent a VPS of ALICLOUD to act as the remote TCP server. We check whether our TCP connection will be terminated due to the attack proposed in §V. If the connection is affected, we infer that the NAT device is vulnerable and thus the NATed network would be impacted. The experiments only impact our own TCP connection and do not affect normal users. We also reported the experimental results to the network administrators. Additionally, we responsibly disclosed the identified vulnerabilities to the affected vendors and Internet Service Providers (ISPs). Our efforts have been met with a great deal of acknowledgments (see §VIII-C for more details).

Availability. The Proof of Concept (PoC) code for identifying NAT scenarios and verifying whether a NAT device is vulnerable to our DoS attack is available at https://github.com/Internet-Architecture-and-Security/Remote-DoS-Attacks-against-NAT-Networks. This repository includes detailed instructions to reproduce our attack.

II. Background

In this section, we first briefly review the NAT technique. Then, we introduce the Path Maximum Transmission Unit Discovery (PMTUD) mechanism, which will be exploited in our attack to identify NAT devices on the Internet.

A. Network Address Translation

Network Address Translation (NAT) is a popular technique allowing hosts (or mobile devices) with private IP addresses to communicate with hosts outside their local network. Through the use of a single publicly routable IPv4 address (e.g., 6.6.6.6), a local network segment behind a NAT gateway (routing devices performing NAT) can enable multiple clients (e.g., $client_2$, $client_3$, and $client_4$) to access Internet servers, thereby conserving the limited availability of public IPv4 addresses. In essence, NAT works by translating the source IP address of packets leaving the local network to the public IP address of the NAT device that connects the local network to the Internet [46], [45], [6]. When the server on the Internet responds, the response packet will be first routed to the public IP address of the NAT device. The NAT device then translates the destination IP address of the response packet to the private IP address of the internal clients that originally sent the request packet, and finally forwards the packet to that client. NAT offers multiple significant benefits, e.g., IP address conservation, improved security by hiding the internal network from the outside world, flexibility in network design and administration.

The key to NAT's normal operation is that the NAT device maintains a session mapping table. When an internal client initiates a session to a remote server, the NAT device will create a session mapping in its cache table to keep track of the session. This mapping typically includes information such as the source IP address and port of the local client, the destination IP address and port of the remote server at the external realm, and the protocol being used for the session (e.g., TCP, UDP). As packets flow between the internal client and the remote server, the NAT device updates the session mapping in its cache table to keep track of the state of the session. In particular, the session mapping for a TCP connection will change as the state of the TCP connection changes. At the beginning, when the internal client issues a TCP `SYN` packet to initiate a connection with the remote server, the NAT device creates a session mapping for that connection and marks the state of the mapping as `SYN_SENT`. As different TCP packets (e.g., the resulting inbound `SYN/ACK` packet, the subsequent outbound `ACK` packet, the TCP `RST` packet, and the TCP `FIN` packet) are received, the state of the mapping will be altered accordingly to translate and forward the packets [6]. In this paper, we show that real-world NAT devices often omit the validation of the received TCP `RST` packets, enabling attackers to manipulate the device's mapping state and construct a remote DoS attack.

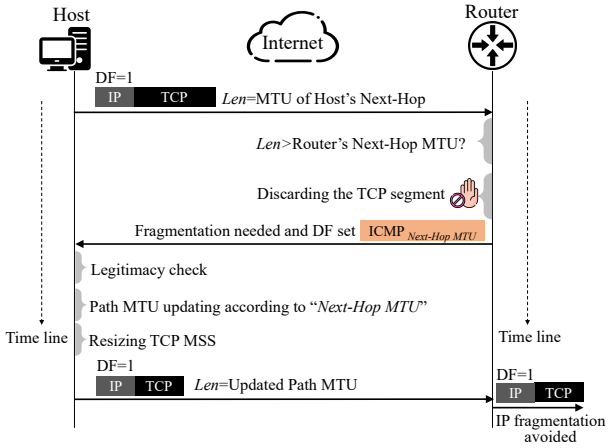


Fig. 1. Workflow of PMTUD to avoid IP fragmentation.

B. Path MTU Discovery

The PMTUD mechanism is a key element of the TCP/IP protocol stack [29], [27], [26]. PMTUD enables the efficient transmission of data over IP networks by determining the maximum packet size that can be transmitted without IP fragmentation over a particular network path, thus reducing network overhead to improve the performance. As shown in Fig. 1, PMTUD works by having the original host send packets with the Don't Fragment (DF) bit set in the IP header. Length of the packets is equal to the default MTU of the host's next-hop (e.g., 1500 octets in Ethernet). If a router along the path encounters the packet that exceeds the router's next-hop MTU,

it discards the packet and issues an ICMP "Fragmentation Needed and DF Set" message (i.e., ICMP error message with `Type`=3 and `Code`=4) back to the host, stating that the packet is too large to be transmitted. The ICMP message carries the MTU of the router's outgoing interface (i.e., next-hop MTU).

Upon receiving the ICMP error message, the host first checks the legitimacy of the message to determine whether the message is truly a reflection of the prior packet sent by itself. For example, if the message carries a TCP packet, the host will check whether the sequence number of the TCP packet is within its own sending window [15]. After that, the host updates its path MTU value preserved for the destination according to the received ICMP error message. Meanwhile, the host reduces the size of the packets for the destination based on the updated path MTU value, e.g., reducing the Maximum Segment Size (MSS) of TCP packets to fit the new path MTU value and thus avoiding IP fragmentation.

According to PMTUD specifications, the path MTU value is maintained at the IP layer of the originator and thus governs all packets sent to the destination. However, in NAT networks, we discover that PMTUD's semantic consistency breaks, leading to desynchronization between internal NATed clients and the NAT device regarding path MTU values. TCP packets from internal clients adhere to the path MTU constraint, but ICMP packets from the NAT device may not, despite originating from the same source IP address. This divergence can be exploited by attackers on the Internet, potentially causing NAT network information leakage.

III. Attack Overview

In this section, we present an overview of our DoS attack. First, we define the threat model of our attack. Then, we describe the steps to construct it.

A. Threat Model

Fig. 2 shows the threat model of our off-path DoS attack. The threat model consists of 5 types of hosts.

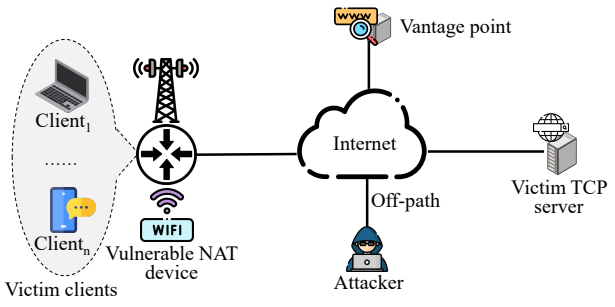


Fig. 2. Threat model of remote DoS attack against NAT networks.

1. A victim TCP server is on the Internet providing various online services (e.g., popular Web services, online banking, and instant messaging) for Internet users.

2. A vulnerable NAT device links a private network segment to the Internet. It translates the source IP address of packets leaving the private network to a publicly routable IP address, enabling access to servers in the external realm. In the context of our attack, the NAT-enabled device may be a gateway in public Wi-Fi networks, a PDN gateway or UPF in 4G LTE/5G networks, or a CPE gateway in IoT networks.

3. Several victim clients are located within the NATed network segment behind the vulnerable NAT device, and they access online services from the victim server.

4. An attacker resides in the external realm, i.e., on the Internet. The attacker is capable of sending IP packets with spoofed source IP address. This capability assumption is practical, since prior studies show that about a quarter of ASes on the Internet do not filter packets with spoofed source addresses leaving their networks [25], [14]. **T**he attacker aims to indiscriminately cut off TCP connections from the victim clients to the specified victim server, thereby performing a DoS attack. This attack does not target one specific client or a few select ones; instead, it affects all clients attached to the vulnerable NAT device.

5. A vantage point is an HTTP server deployed by the attacker on the Internet. Upon deception (e.g., via a URL-based advertisement), the victim client connects to the vantage point. The attacker can then identify whether the client is in a NAT network, potentially pinpointing a target NAT device[1]. **I**t is worth noting that in practice, the vantage point can overlap with the attacker, i.e., using the same host.

B. Attack Steps

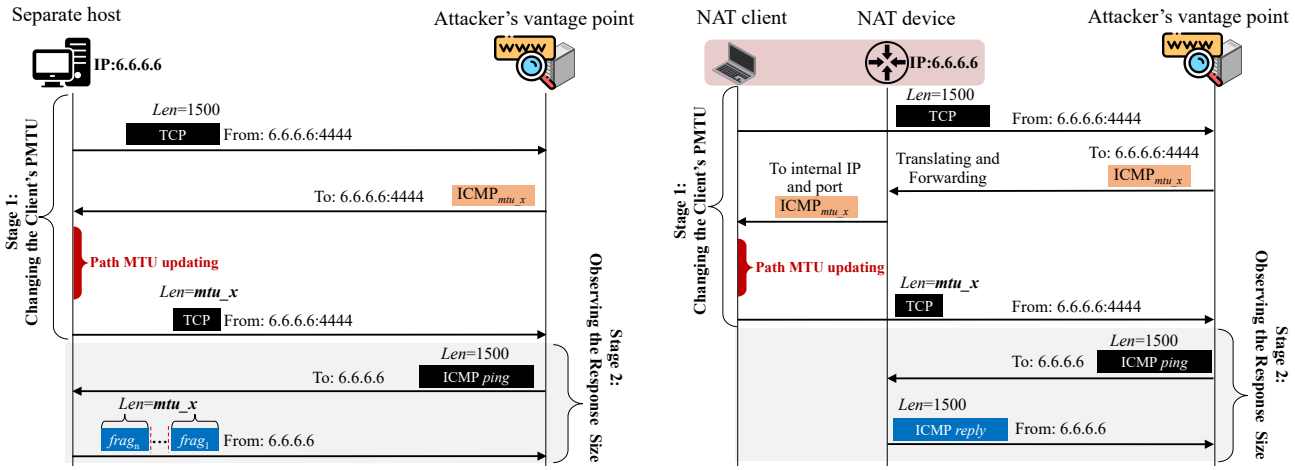
Our DoS attack consists of 2 steps:

Step 1. Identifying NAT Devices. Clients behind NAT devices, tricked by URL-based ads on social platforms or forums, may connect to our vantage point. Leveraging the side channel within the PMTUD mechanism of NAT specifications, our vantage point identifies the client within NAT networks, not as a separate IP host. This leads to the identification of a target NAT device for subsequent attacks, characterized by the client's public source IP address.

Step 2. Conducting DoS Attacks. By crafting `RST` packets without exact sequence numbers and issuing them to the identified NAT device, the attacker deceives the device into mistakenly removing the maintained mappings for TCP connections between the victim clients and the victim server. Subsequently, the attacker injects manipulated TCP packets into the targeted connection, creating an inconsistency between the victim clients and the victim server, effectively executing a DoS attack to terminate the connections. **A**s TCP is a fundamental protocol of the Internet, higher-layer applications built upon it will be affected by our DoS attack, such as SSH, Web, and FTP.

It is worth noting that in the context of the off-path threat model, the victim TCP server and destination port are typically publicly known [8], [9], [12], [13] (e.g., popular HTTP servers on port 80). By crafting TCP `RST` packets, the attacker can remove the identified NAT device's session mappings (or disturb the establishment of session mappings) to such

[1]Our identification bypasses client-side configurations, as explained in §VI-B, without requiring JavaScript installation to access local information, which may be restricted by browser sandboxing mechanisms [47].



(a) The client is a separate host without enabling NAT.

(b) A NAT device with public IP address of 6.6.6.6 is identified.

Fig. 3. Identifying NAT devices by leveraging a side channel presenting in the mechanism of path MTU discovery.

a server. Consequently, the attacker can prevent the NATed clients from accessing the specified victim server. Besides, the attacker is not required to exactly detect a target TCP connection on the NAT device. In other words, the attacker does not rely on side channels [12], [8] to accurately infer the exact source port number (i.e., the random ephemeral port) of a target TCP connection (see §V for the details of our attack). In practice, the source port number of a TCP connection is always within a narrow range, e.g., from 32768 to 61000 in Linux systems and from 49152 to 65535 in Windows systems [12]. As a result, with just the knowledge of the public IP address of the NAT device, the attacker can utilize our attack in parallel to simultaneously cut off multiple existing TCP connections issued from these source ports or prevent the establishment of TCP connections to the specified victim server, thus constructing the DoS attack to disrupt communications between the NAT network and the victim server (refer to §VI-D for detailed insights into the impacts and costs associated with our DoS attack case studies). In the next two sections, we elaborate the two attack steps one by one.

IV. Identifying NAT Devices

According to the NAT specifications [46], [45], [6], servers on the Internet are unable to distinguish whether an access request was issued from a NATed client or a separate IP host. If this were the case, there would be an information leakage of the NAT network, whereby the server would recognize that the source IP address of the received request is a publicly routable IP address of the NAT device, instead of a separate IP host. However, we discover a side channel in the PMTUD mechanism which is inadequately addressed in NAT specifications. This side channel enables the server (e.g., an attacker's vantage point) to stealthily identify NAT devices on the Internet. Fig. 3 shows our method of how to identify a NAT device on the Internet. In a nutshell, our method consists of two stages. First, the attacker's vantage point (i.e., an HTTP server) tricks the connected client into changing its path MTU value preserved for the vantage point. Second, the vantage point sends ICMP requests (i.e., ICMP ping packets) to the client and then observes the size of the replies to identify whether the client is a NATed client or not. Next, we elaborate these two stages.

A. Changing Client's Path MTU

At the beginning, the vantage point acquires the following information from the received TCP packet issued from the client: the client's IP address (e.g., 6.6.6.6), the client's source port number (e.g., 4444), the TCP sequence number of the client (i.e., *seq*), and the packet size (e.g., 1500 octets). Then, the vantage point impersonates an intermediate router and issues a crafted ICMP "Fragmentation Needed and DF Set" message to the client (as shown in Fig. 4), indicating that the prior TCP packet size exceeds the router's next-hop MTU and was discarded by the router. The crafted ICMP message carries the first 28 octets of the prior TCP packet, i.e., IP header of the packet, the source port number of 4444, the destination port number of 80 (i.e., the HTTP server in our example), and the sequence number of *seq*. Besides, the vantage point specifies the Next-Hop MTU value in the crafted ICMP message to *mtu_x*, which is smaller than 1500. Note that according to the ICMP specifications, ICMP "Fragmentation Needed and DF Set" messages may be returned by any router on the path from the client to the vantage point. Therefore, it is difficult to verify the legitimacy of the source of this type of ICMP error messages, which means the source IP address of the message can be arbitrarily specified by the vantage point.

If the client is a separate IP host (as shown in Fig. 3(a)), it will first perform a legitimacy check on the received ICMP "Fragmentation Needed and DF Set" message. The message carries information derived from the prior TCP packet, so it will pass the client's check and the client will respond to the message according to the PMTUD mechanism as mentioned prior in §II-B. The client will adjust the length of its subsequent TCP packets for the vantage point to *mtu_x*. In contrast, if the client is a NATed client (as shown in Fig. 3(b)),

the NAT device will receive the ICMP message first and then translate the message to the internal client. After receiving the ICMP message, the client will respond to the message according to the PMTUD specifications (i.e., updating its path MTU to the attacker's vantage point), and finally adjust the length of the TCP packets to *mtu_x*. As a result, when the client sends subsequent TCP packets to the vantage point again, the vantage point will observe a significant change in the packet length, i.e., decreasing from the previous 1500 octets to *mtu_x*. The vantage point can successfully infer that the client's path MTU value has been deceitfully updated.

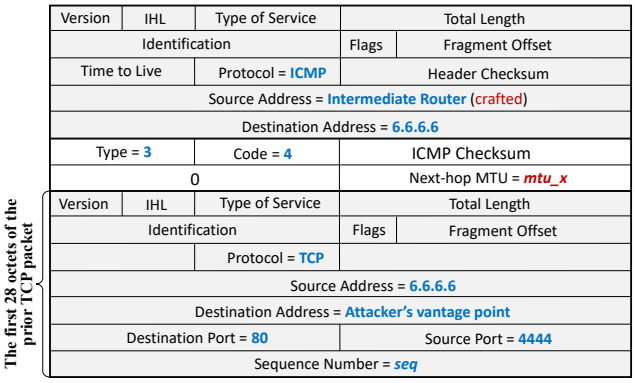


Fig. 4. The crafted ICMP fragmentation needed message.

B. Observing Response Sizes

After changing the client's path MTU, the vantage point initiates an ICMP ping request packet (i.e., ICMP packet with `Type`=8 and `Code`=0) to the client's public routable IP address (i.e., 6.6.6.6). The length of the ping request packet is padded to the previously observed 1500 octets from the regular TCP packets. Distinctively, the generation of the ICMP reply packet (i.e., ICMP packet with `Type`=0 and `Code`=0) significantly differs between the two scenarios of a separate host and a NAT client.

If the remote client is a separate host (as shown in Fig. 3(a)), the ping request packet sent by the vantage point will be received directly by the host. Affected by the previously updated path MTU value (i.e., decreased to *mtu_x*), the returned 1500-octet ICMP reply packet will be fragmented into several IP fragments by the host, each with a length of *mtu_x*. On the contrary, if the client is a NAT client, the ping request packet sent by the vantage point will arrive at the NAT device instead of being received by the client (as shown in Fig. 3(b)). As a result, the length of the returned ICMP reply packet issued by the NAT device will be the default 1500 octets without being fragmented[2]. This is because the previous ICMP "Fragmentation Needed and DF Set" message sent by the vantage point updated the path MTU value of the internal NAT client, while the path MTU value preserved in the NAT device for the vantage point was not affected. Consequently, the path MTU value from the same IP address (i.e., 6.6.6.6 in our example) to the vantage point has been desynchronized, i.e., the length of the TCP packet is *mtu_x* while the length of the ICMP reply packet is 1500 octets, forming a side channel.

In summary, by tricking the client into altering its path MTU value, the attacker can differentiate between separate IP hosts and NAT clients by analyzing the ICMP reply packet lengths, enabling subtle identification of NAT devices on the Internet. If the received ICMP reply packet is not fragmented (as shown in Fig. 3(b)), the client is a NATed host, and the public IP address of the observed NATed client corresponds to that of the NAT device, all achieved without requiring additional assistance. **N**ote that while the attacker can detect the presence of NATed clients and thus conduct a DoS attack against the identified NAT device, it cannot determine the exact number of clients or distinguish between individual ones. Additionally, in cascaded NAT networks (i.e., chained NAT contexts), our method can identify the NATed client accessing the vantage point and enable DoS attacks targeting the outermost NAT device hosting the public IP address. However, it cannot differentiate between the contexts of chained NATs.

V. Conducting DoS Attacks

After identifying NAT devices on the Internet, we proceed to conduct our remote DoS attack against these identified NAT devices to disrupt their TCP connections to a specified victim server. According to the NAT specifications [6], [45], NAT behavior for handling TCP `RST` packets is left unspecified. In practice, NAT implementations in native OSes (e.g., Netfilter in Linux 5.1 and beyond) may validate the legitimacy of received TCP `RST` packets before altering the state of the mappings preserved for associated TCP connections [49]. However, this validation may not be enforced in various real-world downstream router-oriented NAT devices, such as NAT gateways in public Wi-Fi networks, 4G LTE/5G networks, and cloud networks. By exploiting this vulnerability, an off-path attacker can craft TCP `RST` packets to remove the mappings maintained in the NAT device. This disruption will result in a DoS attack against the NAT network. Our DoS attack consists of three stages, i.e., removing NAT mappings, manipulating TCP states, and terminating TCP connections.

A. Removing NAT Mappings

As shown in Fig. 5, at the beginning, the NAT device maintains two session mappings (using the source port x_2 and x_3) for TCP connections issued from two internal clients to the remote victim server. The server preserves two sockets, i.e., `s1` and `s2`, for the two TCP connections. The parameters of the TCP connections (e.g., source port numbers and sequence numbers) cannot be observed by the off-path attacker on the Internet. The attacker impersonates the victim server to craft multiple TCP `RST` (or TCP `RST/ACK`) packets[3] to the

[2]IP fragmentation at intermediate routers will not affect our identification on NAT devices. See §VI-B for our detailed evaluations on this issue.

[3]In practice, some NAT devices, e.g., the carrier-grade NAT (CGNAT) devices within the China Unicom Beijing province network, may also examine the `ACK` flag (not the sequence number) in the received `RST` packet prior to removing TCP connection mappings. Consequently, in our empirical measurement study, we consistently set the `ACK` flag of the crafted `RST` packets to true as well (i.e., a crafted TCP `RST/ACK` packet), ensuring the success of our attack in different NAT networks.

identified public IP address of the NAT device. The source IP address `s_IP` of the crafted `RST` packets is specified as the victim server's IP address. The destination port number `d_port` (i.e., the source port of the client-issued TCP connections) is typically located within a small range, e.g., from 32768 to 61000 in Linux systems and from 49152 to 65535 in Windows systems [12]. The source port number (i.e., the destination port of the TCP connections) is usually known, e.g., 80 in HTTP. The sequence number `seq` of the crafted `RST` packets is arbitrary (i.e., specified by the attacker arbitrarily). The acknowledgment number is not required. The crafted TCP `RST` packets with incorrect `d_port` will be discarded by the NAT device directly. By contrast, the packets with correct `d_port` (i.e., x_2 and x_3) will be translated into the internal clients by the NAT device. Because the sequence number in the packets are incorrect, the internal clients with a robust TCP/IP protocol suite implementation will eventually discard them. However, the NAT device may be tricked into removing the session mappings for the two TCP connections without verifying the legitimacy of the `RST` packets, primarily for performance considerations.

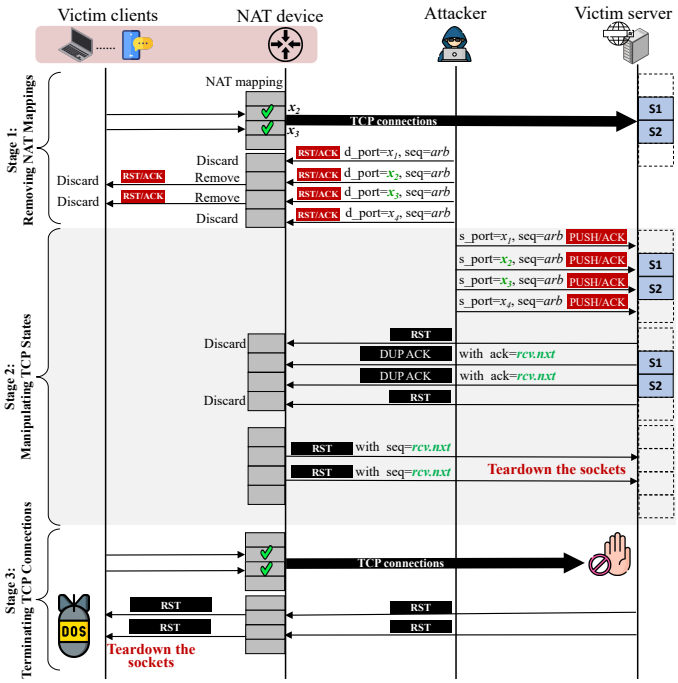


Fig. 5. Design of our remote DoS attack against NAT networks.

B. Manipulating TCP States

After removing the session mappings, the attacker then impersonates the NAT device (i.e., using the identified public IP address of the device) to craft multiple TCP `PUSH/ACK` packets (i.e., with the `PUSH` and `ACK` flags in TCP header enabled) to the server. The source port number `s_port` of the crafted `PUSH/ACK` packets is specified within the range that TCP usually selects, and the destination port number is specified as the known number in the server (e.g., 80 in HTTP). The sequence number and the acknowledgment number are arbitrary. The crafted `PUSH/ACK` packets will trigger the server to respond differently. As shown in Fig. 5, for `PUSH/ACK` packets with source port equal to the previously established TCP connections' source port (i.e., the source port of x_2 and x_3 in our example), the server will issue a duplicate acknowledgment packet to the public IP address of the NAT device according to the fast retransmit mechanism defined in RFC 5681 [1]. Particularly, the duplicate acknowledgment packet carries the client's exact sequence number of *rcv.nxt*, i.e., the lower boundary of the server's receive window[4]. By contrast, for `PUSH/ACK` packets carrying a wrong source port, the server will reflect a TCP `RST` packet to the NAT device, since no TCP sockets with those source ports are preserved on the server for the public IP address of the NAT device. The NAT device will discard the received `RST` packets silently.

Once the duplicate acknowledgment packets reach the NAT device, the NAT device may generate an `RST` packet to the server for each received duplicate acknowledgment packet. This occurs because the mappings for the TCP connections were previously removed due to the attacker's crafted TCP `RST` packets, leading the NAT device to believe that the corresponding TCP sockets have been terminated. According to the TCP specifications [36], [11], a TCP `RST` packet will be sent whenever a non-RST packet arrives at a closed socket. Moreover, the sequence number (i.e., `seq`) of the `RST` packets issued from the NAT device will be specified as the clients' exact sequence number (i.e., *rcv.nxt* of the server), copied from the previously received duplicate acknowledgment packets. In accordance with TCP specifications, the `RST` packets that carry the exact sequence number of the clients will deceive the server into tearing down the corresponding TCP sockets. In practice, the attacker can optimize the attack by interleaving stage 1 (issuing crafted `RST` packets) and stage 2 (issuing crafted `PUSH/ACK` packets), thereby mitigating potential limitations in the victim client's TCP packet transmission between these stages, which could affect the attack's success (refer to §VI-D for experimental details in our case studies).

C. Terminating TCP Connections

Finally, once the clients have subsequent TCP packets to send to the server through the previously established TCP connections, the NAT device will first create new mappings. Once the TCP packets arriving at the server, the server will directly discard the packets. Moreover, the server will issue `RST` packets to the clients according to TCP specifications. These `RST` packets carry the exact sequence number (copied from the `Acknowledgment Number` field of the prior received TCP packets issued by the clients) of the connections, and eventually force the clients to tear down the TCP connections, i.e., leading to a DoS attack.

[4]We observe that servers running Linux, Windows, FreeBSD, and macOS consistently adhere to the fast retransmit mechanism defined in RFC 5681. They generate duplicate acknowledgment packets upon receiving the `PUSH/ACK` packets. In contrast, OpenBSD systems deviate from this behavior, as they do not produce the expected duplicate ACK packets. Consequently, servers equipped with OpenBSD will not be affected.

VI. Evaluations

In this section, we conduct a comprehensive end-to-end evaluation of our attacks. We begin with the experimental setup, followed by the evaluation results. This includes identifying NAT devices via the PMTUD side channel, testing whether NAT mappings in various implementations can be maliciously removed by attackers, and assessing the attack's costs through case studies on DoS attacks over SSH and FTP.

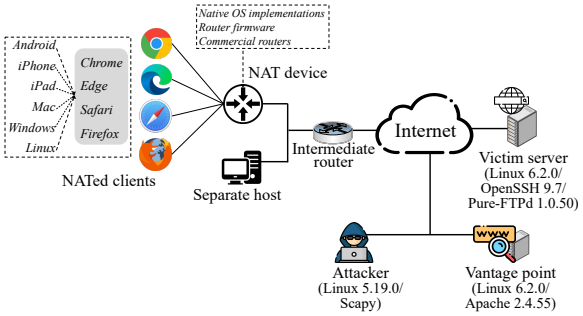


Fig. 6. Experimental setup for end-to-end evaluations.

A. Experimental Setup

Fig. 6 shows the experimental setup for conducting end-to-end evaluations of our attacks. The NAT device setup includes forwarding-enabled hosts with 6 native OSes and 8 types of router firmware implementing NAT, as well as 30 commercial routers with NAT enabled (see Table III for NAT setup details). Connected NATed clients have varied configurations, using different OS and browsers. This diversity facilitates the comparison of our NAT device identification method with malicious JavaScript-based methods [47]. The separate host shares an identical configuration with the NATed clients, the sole difference being its possession of a public IP address, allowing direct access to the vantage point. This setup is designed to evaluate the effectiveness of our NAT device identification method, specifically in its ability to distinguish between a NATed client and a separate host at the vantage point. An intermediate router equipped with OpenBSD 7.4 is utilized to check whether IP fragmentation at intermediate routers affects our identification method. The victim server offers SSH connectivity and FTP file download services to the NATed clients. Our evaluation involves three tests: First, the attacker identifies NAT devices via the identified PMTUD side channel. Second, the attacker examines if NAT mappings in the devices can be manipulated for our DoS attack. Third, we carry out case studies on DoS attacks targeting SSH connections and FTP downloads from the NATed clients to the victim server, assessing the impacts and cost implications of our attack.

B. NAT Devices Identification

Initially, the NATed clients and the separate host separately click on a URL to access the vantage point. Following the procedure in Fig. 3, the vantage point can effectively distinguish requests from the separate host and the NATed clients. In other words, all NAT setups listed in Table III are affected by the side channel in the PMTUD mechanism. The differentiation is based on the alignment between the TCP packet size of the separate host and its ICMP ping reply packet size, both adjusted to a 600-octet path MTU value (i.e., *mtu_x*) specified by the vantage point. In contrast, the NATed clients display a different behavior, with their TCP packet size adhering to the 600-octet path MTU, while the ICMP ping reply packet size (originating from the NAT device) defaults to 1500 octets.

We also conduct experiments to evaluate the potential impact of IP fragmentation at intermediate routers on our identification method. We modify the next-hop MTU value of the intermediate router to different sizes (i.e., 1492 octets used by IEEE 802.3 and 576 octets used by X.25 networks [29]), other than the default 1500 octets, as illustrated in Table I. This allows us to evaluate that even when the reflected ICMP ping reply packets are routed through a path where some intermediate routers may perform IP fragmentation, our identification method remains unaffected, since the ICMP reply packet sizes observed from the separate host and the NATed client are always distinguishable. Another potential disruption to our identification method, arising from intermediate routers, is the possibility of the fragmented ICMP reply packet from the separate host being reassembled by certain routers, thus restoring the IP fragments to the original 1500-octet size. This has the potential to mislead the vantage point, leading to an erroneous identification of the separate host as a NAT client. However, in practice, only the destination host reassembles fragments, and intermediate routers do not enforce fragment reassembly, as fragments do not always take the same routes from source to destination [5], [22]. Therefore, our identification method effectively mitigates the risk of such misidentification.

TABLE I
Affects from intermediate IP fragmentation.

mtu_x	Next-hop MTU of the router	ICMP reply size observed from separate host	ICMP reply size observed from NATed client	Distinguishable
600	1500 (by default)	2*600+300	1500	✔
	1492 (greater than 600)	2*600+300	1492+8	✔
	576 (lower than 600)	2*(576+24)+300	2*576+348	✔

Moreover, we compare our NAT identification method with the JavaScript-based method [47], [48], which requires NATed clients to install a malicious JavaScript to call APIs like WebRTC, enabling access to the local IP address and comparing it with the public IP address. This method has significant limitations due to security policies in modern browsers, which may restrict the exposure of the local IP address to JavaScript [47]. In contrast, our method is configuration-independent, leveraging a fundamental side channel vulnerability, thereby impacting all client configurations in Fig. 6. Table II summarizes our experimental results, showing that our method consistently identifies the NAT scenario and the corresponding public IP address across 21 client configurations[5], while the JavaScript-based method succeeds in only two configurations.

[5]In our tests, we use browsers on various OSes, all of which are recent stable versions, e.g., Chrome 117.0.5938.60, Edge 117.0.2045.33, Safari 16.6, and Firefox 117.3.

TABLE II
NAT IDENTIFICATION WITH DIFFERENT CONFIGURATIONS.

	Chrome	Edge	Safari	Firefox
Android 12	✔—✔	✔—✔	N/A	✘—✔
iOS 16.3	✘—✔	✘—✔	✘—✔	✘—✔
iPadOS 16.61	✘—✔	✘—✔	✘—✔	✘—✔
MacOS 13.0	✘—✔	✘—✔	✘—✔	✘—✔
Windows 10	✘—✔	✘—✔	N/A	✘—✔
Linux 6.2.0	✘—✔	✘—✔	N/A	✘—✔

✔means the JavaScript-based method works.
✘means the JavaScript-based method fails.
✔means our side channel-based method works.

Note that the JavaScript installed on the client may also exploit a timing side channel to identify whether NAT is enforced or not. However, this method also heavily relies on specific network configurations. The JavaScript may connect to common private gateway addresses (such as 10.0.0.1, 192.168.1.1, 192.168.0.1, etc.) through a hidden img HTML tag. Upon successful connection, a JavaScript event is triggered, providing information that the client is a NAT client. In contrast, separate hosts will not achieve success and will raise an error after a timeout. We evaluate this method in two common NAT scenarios: public Wi-Fi networks and 5G cellular networks, using the same experimental setup where a client with malicious JavaScript attempts to connect to the common gateway addresses. In 50 experiments conducted on our campus NAT-enabled Wi-Fi network, the JavaScript successfully connects to one of the 30 common private gateway addresses we listed, with an average time cost of 44 milliseconds (as shown in Fig. 7), effectively identifying the NAT scenario. However, in the 5 tested 5G NAT networks accessible on our campus, all experiments fail due to non-deterministic assignment of gateway addresses in 5G networks, which remains undisclosed to users.

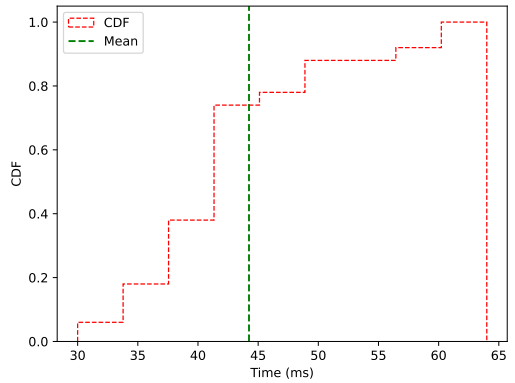


Fig. 7. Time cost of identifying NAT in Wi-Fi networks through connection to common private gateway addresses.

C. NAT Mappings Manipulation

Table III presents the findings of our experiments on off-path removal of NAT mappings in various NAT setups. These setups encompass 6 native OSes, with 2 found to be vulnerable, 8 types of router firmware, 6 of which exhibit vulnerabilities, and 30 commercial routers from 14 vendors, 29 of which display vulnerabilities. The malicious removal against NAT mappings is achieved using crafted TCP `RST` packets with arbitrary sequence numbers. Our experimental results demonstrate the disparities in NAT implementations among native OSes, various router firmware versions, and commercial routers. Particularly noteworthy is the vulnerability exhibited by the majority of downstream real-world NAT devices[6].

TABLE III
TCP SESSION MAPPINGS REMOVAL VIA CRAFTED `RST`.

NAT Setup	OS/Firmware /Router	Version /Vendor	Release Date*	Vulnerable
Native OS	FreeBSD	13.2 and earlier	04/2023	✔
	Linux	5.0 and earlier	05/2019	✔
	Linux	5.1 and beyond	05/2019	✘
	OpenBSD	5.0 and beyond	11/2011	✘
	macOS	13.2.1	02/2023	✘
	Windows	10	07/2015	✘
		11	10/2021	
Router Firmware	OpenWrt	22.03 and earlier	05/2023	✔
	AsusWrt	3.0.0.4.386 and earlier	10/2022	✔
	pfSense	2.7.0 and earlier	06/2023	✔
	OPNsense	23.7 and earlier	07/2023	✔
	iKuai	3.7.6 and earlier	09/2023	✔
	VxWorks	5.5.1	09/2002	✔
	VyOS	1.4 and beyond	11/2020	✘
	RouterOS	6.49 and beyond	08/2021	✘
Commercial Router	RAX20	Netgear	10/2020	✔
	RAX50	Netgear	02/2020	✔
	E5600	Linksys	03/2020	✔
	E9450	Linksys	05/2022	✔
	RT-AX57	ASUS	02/2023	✔
	RT-AX89X	ASUS	10/2020	✔
	AR6140E-9G-2AC	Huawei	05/2023	✔
	AX3 Pro	Huawei	09/2020	✔
	WS5200	Huawei	—	✔
	TC7102	Huawei	04/2020	✔
	TL-R473GP-AC	TP-Link	04/2021	✔
	TL-R4239GP	TP-Link	06/2022	✔
	TL-XDR6020	TP-Link	01/2022	✔
	TL-AC1200	TP-Link	12/2020	✔
	TL-WDR7620	TP-Link	—	✔
	Magic R100	H3C	01/2020	✔
	Magic R365	H3C	09/2022	✔
	EG105G-V2	Ruijie	05/2023	✔
	EG210G-P	Ruijie	01/2023	✔
	X32 Pro	Ruijie	08/2022	✔
	Redmi RA81	Xiaomi	01/2022	✔
	CR6609	Xiaomi	—	✔
	NBR1009GPE	Netcore	—	✔
	MG1200AC	Netcore	—	✔
	Wimaster	Wimaster	—	✔
	Wimaster-mini	Wimaster	—	✔
	Google Wi-Fi	Google	10/2016	✔
	SK-WR6640X	Skyworth	—	✔
	RAX1800Z	China Mobile	11/2021	✔
	Cisco Meraki MX64	Cisco Meraki	02/2015	✘

✔means the NAT implementation is vulnerable.
✘means the NAT implementation is invulnerable.
* the release date information for the commercial routers is sourced from the Internet.

D. Case Study

i) DoS over SSH Connections. As shown in Fig. 6, 4 NATed clients behind the NAT device have established SSH connections with the victim server (our VPS deployed in

[6]After we reported the vulnerability, the FreeBSD and OpenWrt communities promptly acknowledged it. Specifically, TCP sequence number validation on inbound packets is now enabled by default in pf(4) starting from FreeBSD 14.0 and in netfilter starting from OpenWrt 23.05, respectively.

TABLE IV
EXPERIMENTAL RESULTS OF IDENTIFYING NAT DEVICES ON THE INTERNET.

	Frankfurt	Virginia	California	Jakarta	Bangkok	Beijing	São Paulo	Total	
Request	14,487	14,428	15,690	9,102	13,114	10,328	14,312	91,461	
Denial	7,936	8,235	9,617	4,808	8,001	5,832	7,370	51,799	
Approval	6,551	6,193	6,073	4,294	5,113	4,496	6,942	39,662	
Clients	5,616	5,045	3,458	3,536	3,883	3,184	5,432	30,154	
NAT	1,416 (486A 84C)	1,158 (386A 87C)	804 (275A 68C)	927 (316A 66C)	994 (334A 75C)	863 (275A 68C)	1,443 (491A 84C)	7,605 (1,289A 124C)	25.22%
Separate IP	2,449	2,408	1,650	1,636	1,819	1,361	2,425	13,748	45.59%
Unknown	1,751	1,479	1,004	973	1,070	960	1,564	8,801	29.19%

"A" means ASes where the identified NAT devices reside, and "C" means countries that the identified NAT devices belong to.

Tencent Cloud and ALICLOUD, respectively). These clients periodically send messages to the server. Additionally, other NATed clients accessing the NAT network may intend to establish new SSH connections with the server. The attacker on the Internet utilizes the method illustrated in Fig. 5 to launch a DoS attack, aiming to terminate SSH connections already established by the 4 NATed clients and disrupt the establishment of new SSH connections, thereby blocking the NAT network from connecting to the SSH server. The attacker continuously alternates between sending multiple crafted TCP `RST` packets (up to 65,535) to the NAT device, aiming to clear the session mappings, and sending crafted `PUSH/ACK` packets to the server to disrupt the SSH connections. This orchestrated sequence of actions constitutes a DoS attack.

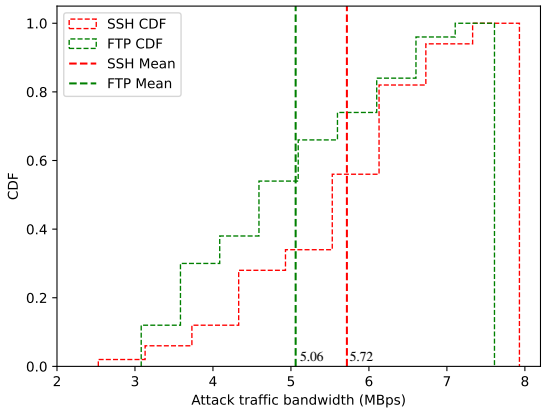


Fig. 8. Attack traffic bandwidth of DoS over SSH and FTP.

Fig. 8 presents the empirical cumulative distribution function (CDF) of the attack traffic bandwidth. We conduct 50 experiments, and our experimental results reveal that using an average bandwidth of 5.72MBps, the off-path attacker can block all NATed clients behind the vulnerable NAT device from connecting to the SSH server (including hindering the establishment of new SSH connections). **T**he crafted TCP `RST` packets and `PUSH/ACK` packets sent to the NAT device and the server, respectively, do not trigger alerts on either the NAT device or the server. Notably, out-of-order `PUSH/ACK` packet transmission is common in practice. We observe that the tested server (running Linux, as well as Windows, FreeBSD, and macOS), secured by Tencent Cloud or ALICLOUD firewalls, does not block the crafted `PUSH/ACK` packets.

ii) DoS over FTP Downloads. Similarly, using our attack method, the attacker can terminate established TCP connections between the NATed clients and the FTP server in clouds or prevent the establishment of new connections. This obstruction hinders communication between the NAT network and the FTP server, leading to disruptions in FTP file downloads. We conduct 50 experiments and the average attack traffic bandwidth is about 5.06MBps, as shown in Fig. 8.

VII. REAL-WORLD ATTACKS

In addition to end-to-end evaluations, we also conduct a comprehensive study to evaluate the impacts of our attacks in the real world. The results of our ethical experiments on the Internet show that our attacks could significantly damage the real world. Firstly, over an 11-month period, we identify more than 7,600 NAT devices that are distributed across 1,289 ASes in 124 countries around the world. Moreover, we conduct evaluations on 180 real-world NAT networks, including various scenarios such as public Wi-Fi networks, 4G LTE/5G networks, and cloud networks. The experimental results demonstrate that more than 92% of the tested NAT networks are vulnerable to our DoS attack. **N**ote that even though our experiments involve human interactions, we fully addressed the ethical issues. We provided detailed explanations of the experiment's purpose and methodology and obtained participants' approval before conducting the experiments. Additionally, our experiments do not cause harm to the participants, and we provide them with reports on the experiment results (see Ethical Considerations in §I for more details).

A. Identifying NAT Devices on the Internet

Experimental Setup. We deploy 7 vantage points in 5 ASes, i.e., HTTP servers equipped with Apache 2.4.55/Linux 6.2.0, located globally in Frankfurt, Virginia, California, Jakarta, Bangkok, Beijing, and São Paulo. We use the method in Fig. 3 to determine whether the client requesting for our vantage point is a NATed client or a separate IP host. We share the URLs for accessing our vantage points in social media platforms (e.g., TikTok and WeChat) and community forums (e.g., Dev community) for seeking voluntary users to participate in our NAT identification. On our vantage points, we open TCP port 80 to listen for incoming requests. When an HTTP request arrives, we first obtain the user's approval before conducting the identification (see §I for our ethical considerations and Fig. 11 in Appendix for the snapshots of

accessing our vantage points[7]). Then, we change the path MTU between the user's machine and our vantage point. After that, we send probing packets and observe the size of the response packets to determine whether the user's machine is a separate IP host or a NATed client. If the user's machine is a NATed client, the source IP address of the request corresponds to a public IP address employed by the NAT device.

Experimental Results. Table IV shows the results of our identification for NAT devices from the 7 vantage points. At different vantage points, the experimental results may vary. For instance, at the vantage point in Frankfurt (the second column of Table IV), we receive a total of 14,487 HTTP requests over a 11-month period, i.e., from May 13, 2023, to April 13, 2024. Out of the 14,487 requests, 7,936 deny our identification, while the remaining 6,551 approve. After eliminating duplicates within the 6,551 approved requests, 5,616 unique clients with different IP address access our HTTP server deployed at this vantage. Among these 5,616 clients, our method reveals that 1,416 are located behind a NAT device, indicating the identification of 1,416 publicly routable IP addresses used by NAT devices. These NAT devices are spread across 486 ASes (i.e., 486A) in 84 countries (i.e., 84C). 2,449 out of the 5,616 clients are operating as separate hosts with unique IP address. The remaining 1,751 clients fall under the "Unknown" category due to undetectable results. This implies that the reflected ICMP reply packets, responding to the vantage point's probing, are either blocked at the client or by middleboxes on the Internet [37]. As a result, it is difficult for the vantage point to determine whether these clients are within NAT networks or functioning as separate IP hosts.

We also encounter scenarios where clients might access our HTTP vantage points via a VPN proxy, making it challenging for the vantage points to distinguish whether the client is behind a NAT device or a separate IP host. This challenge stems from the fact that when the vantage point sends an ICMP "Fragmentation Needed and DF Set" message to the client (in practice, this message is directed to the VPN proxy rather than the real source client) to adjust the client's path MTU (as shown in Fig. 3), our observations indicate that the VPN proxy typically discards the ICMP message instead of forwarding it to the actual source client, unlike a typical NAT device[8]. Consequently, the ICMP "Fragmentation Needed and DF Set" message fails to reduce the client's path MTU value and thus affects the subsequent observations. Under this circumstance, the vantage point cannot distinguish whether the client is a separate IP host or a NATed client, and we classify this circumstance as "Unknown".

By aggregating the results from all 7 vantage points (as shown in the last column of Table IV), we determine that over a 11-month period, we receive a total of 91,461 HTTP requests[9]. Out of the 30,154 clients, 7,605 (25.22%) are identified as residing behind a NAT device. The identified NAT devices with public IP addresses are distributed across 1,289 ASes in 124 countries worldwide. Out of the remaining clients, 13,748 (45.59%) are separate IP hosts, and the status of the remaining 8,801 (29.19%) clients is unknown. Table V provides the details about 21 public IPv4 addresses used by identified NAT devices. For instance, as illustrated in the first row, we identify a NAT device with a public IP address of "*.145.177.*" within a /24 CIDR range from the vantage point in Frankfurt. This NAT device is located in Jakarta, Indonesia. Fig. 9 shows the geographical distribution of all the identified NAT devices.

TABLE V
DETAILS OF 21 PUBLIC IPV4 ADDRESSES USED BY NAT.

Public IP	CIDR	Location	Vantage Point
.145.177.	/24	Jakarta, Indonesia	
.74.192.	/20	London, UK	Frankfurt
.52.158.	/24	Multan, Pakistan	
.147.62.	/24	Kathmandu, Nepal	
.190.220.	/19	Donetsk, Russia	Virginia
.97.49.	/20	Lima, Peru	
.64.76.	/24	Kalemie, Congo	
.104.148.	/19	Frankfurt, Germany	California
.40.66.	/21	Xi'an, China	
.117.5.	/19	Rasht, Iran	
.0.15.	/16	Toledo, Spain	Jakarta
.190.50.	/20	Ramučiai, Lithuania	
.116.1.	/18	Bistagno, Italy	
.64.125.	/16	Banff, Canada	Bangkok
.17.60.	/24	Paris, France	
.113.106.	/24	Palwal, India	
.116.152.	/12	Shenzhen, China	Beijing
.235.251.	/24	Nova Prata, Brazil	
.171.124.	/14	Lomas, Argentina	
.101.186.	/24	Zalesye, Ukraine	São Paulo
.199.82.	/16	Grodzisk, Poland	

B. Identifying Vulnerable NAT Networks

Experimental Setup. Our attack involves four types of devices: 1) *A victim client*, residing within the tested NAT networks, is under our control for ethical reasons. The victim client can be either a Xiaomi 12 cellphone or a VM rented from the tested clouds, equipped with a robust TCP/IP protocol suite implementation. Note that our attack does not impact regular users of the tested NAT networks. 2) *A NAT device* responsible for translating and mapping internal clients' private IP addresses to public ones. This NAT device could be the gateway for public Wi-Fi networks, a PDN Gateway/UPF device in 4G LTE/5G networks, or the gateway for VMs in cloud networks. 3) *An SSH server* located externally to the tested NAT networks. The server is deployed in California and is equipped with Linux 6.2.0 and OpenSSH 9.3. 4) *An attack machine* equipped with Linux 5.19.0 and Scapy, with the capability of IP spoofing. During our experiments, the

[7]In practice, the identification can be carried out without the awareness of the clients.

[8]In our end-to-end experimental setup, we further confirm that a VPN proxy does not forward ICMP "Fragmentation Needed and DF Set" messages, as exemplified by the well-known OpenVPN with version 2.5.9. Note that our setup confirms that VPN proxies obstruct the NAT identification. However, we cannot pinpoint VPN proxies specifically using this method, as other middleboxes might also drop ICMP messages.

[9]If the same source IP address requests our different vantage points, its HTTP requests will be recorded and handled only by the first vantage point requested, to avoid duplication.

victim client initially connects to the target NAT networks. It then establishes a TCP connection to the remote SSH server. The NAT device maintains a session mapping for the TCP connection between the victim client and the remote server. The off-path attacker will disrupt the SSH session between the victim client and the SSH server by exploiting the proposed DoS attack shown in Fig. 5.

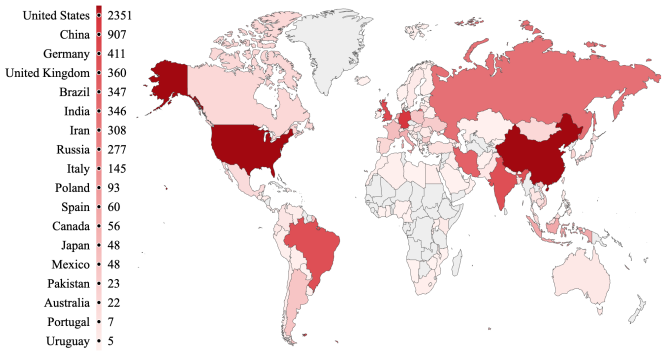


Fig. 9. Distribution of the identified NAT devices.

Experimental Results. Given the widespread use of NAT in the real world, we randomly select 180 NAT networks from three popular network scenarios for a comprehensive evaluation. These networks include 90 4G LTE/5G networks, 60 Wi-Fi networks, and 30 cloud networks, all located in different regions. The overall experimental results are illustrated in Fig. 10. Out of the 90 4G LTE/5G networks, all 52 4G LTE NAT networks and the remaining 38 5G NAT networks are vulnerable. The attacker can successfully terminate TCP connections initiated from cellphones connected to these networks. In regard to the 60 Wi-Fi networks, our attack impacts 16 Wi-Fi 4 NAT networks, as well as 19 Wi-Fi 5 networks and 13 Wi-Fi 6 networks. As a result, the proportion of vulnerable NAT networks for Wi-Fi 4, Wi-Fi 5, and Wi-Fi 6 in our tests are 80%, 76%, and 87%, respectively.

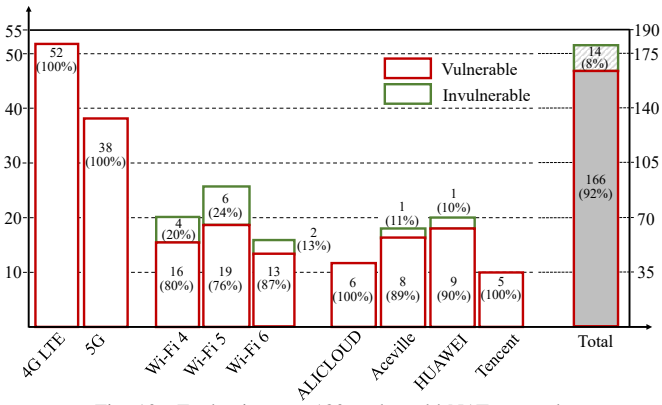


Fig. 10. Evaluations on 180 real-world NAT networks.

Moreover, we conduct tests on the NAT networks from 4 popular cloud providers, i.e., 6 ALICLOUD NAT networks, 9 Aceville NAT networks, 10 HUAWEI CLOUDS NAT networks, and 5 Tencent Cloud NAT networks. Out of these tested NAT networks, all 6 ALICLOUD NAT networks are vulnerable. Besides, our attack affects 8 Aceville NAT networks (i.e., 89% of the tested networks). Similarly, 9 HUAWEI CLOUDS NAT networks experience the impact of our attack (i.e., with a proportion of 90%). Lastly, all 5 Tencent Cloud NAT networks are affected by the attack. In total, out of the 180 real-world NAT networks, 166 are vulnerable to our attack, indicating that the proportion of vulnerable NAT networks are more than 92%. Table VI shows the details of 30 vulnerable real-world NAT networks in our evaluations. For example, as shown in the first row, we find that a 4G LTE NAT network (with a public IP address of "*.216.177.*" and CIDR of /20) located in Virginia, United States is affected by our attacks. This network belongs to Verizon Business. In our testing, out of 10 attacks against this NAT network, 9 of them are successful. The one unsuccessful attempt is due to packet loss of the forged `RST` packet caused by network conditions. It is worth noting that although our attack targets vulnerable NAT implementations and is independent of layer-2 protocols like Wi-Fi and LTE, the comprehensive evaluations show that a wide range of network access scenarios utilizing NAT are susceptible to our attacks.

Reasons for Failures. The failure of our attacks on 14 tested NAT networks can be attributed to two situations. Firstly, the crafted TCP `RST` packets fail to manipulate the preserved mappings in the NAT device of the target NAT networks. During our investigations on 60 public Wi-Fi networks, we identify 12 instances where failures occur due to this reason. In these instances, when the NAT device receives the crafted TCP `RST` packets, it refrains from responding by removing the corresponding mappings. Instead, it directly forwards the `RST` packets to the internal clients (which are eventually discarded by the clients due to the specified incorrect sequence number). We can recognize this situation by noting that the internal client receives the crafted `RST` packets issued by the attacker, but the attack fails. This indicates that the NAT device forwarded the `RST` packets directly and did not remove the preserved mappings. Besides, our attack may also be foiled by specific network configurations, such as middle boxes or firewalls. For instance, during our investigations on cloud networks, we encounter 2 cloud NAT networks where our attack fails. In these cases, the crafted `RST` packets are blocked before they could remove the session mappings. We can identify this situation by observing that the internal client does not receive the crafted `RST` packets issued by the attacker, as well as the failure of our attack.

VIII. Discussion and Countermeasure

In this section, we discuss the impact of our attack on different NAT types and IPv6. We also propose countermeasures.

A. Impacts on Different NAT Types

NAT technology, including various types like CGNAT (Carrier-Grade NAT), SNAT (Source NAT), DNAT (Destination NAT),and NAT-enabled IoT CPE routers, is widely used in real world. Essentially, NAT relies on preserving session state to accurately route traffic between internal and external hosts. These NAT variants maintain a session mapping table to track active sessions, including TCP connections, and log

TABLE VI
EXPERIMENTAL RESULTS OF OUR DOS ATTACK OVER 30 REAL-WORLD VULNERABLE NAT NETWORKS.

No.	Public IP address	CIDR	NAT Scenario	Region	Organization	Success Rate
1	*.216.177.*	/20	4G LTE	Virginia, United States	Verizon Business	9/10
2	*.60.40.*	/16	4G LTE	Canarias, Spain	VODAFONE ESPANA S.A.U.	10/10
3	*.30.41.*	/24	4G LTE	Dhaka, Bangladesh	Grameenphone Limited	10/10
4	*.139.100.*	/19	4G LTE	Guizhou, China	China Telecom	9/10
5	*.144.207.*	/21	4G LTE	Xinjiang, China	China Mobile	10/10
6	*.254.3.*	/24	5G	Beijing, China	China Unicom Beijing	10/10
7	*.108.164.*	/13	5G	Chongqing, China	China Telecom	9/10
8	*.139.124.*	/15	5G	Shannxi, China	China Unicom Shannxi	10/10
9	*.104.41.*	/22	5G	Guangdong, China	China Mobile	10/10
10	*.144.139.*	/23	5G	Sichuan, China	China Mobile	9/10
11	*.88.63.*	/18	VM in cloud	California, United States	ALICLOUD	10/10
12	*.74.95.*	/19	VM in cloud	New South Wales, Australia	ALICLOUD	9/10
13	*.51.98.*	/22	VM in cloud	Ontario, Canada	Aceville	10/10
14	*.130.146.*	/19	VM in cloud	Virginia, United States	Aceville	10/10
15	*.135.216.*	/19	VM in cloud	São Paulo, Brazil	Aceville	10/10
16	*.163.199.*	/19	VM in cloud	Tokyo, Japan	Aceville	10/10
17	*.138.165.*	/20	VM in cloud	Johannesburg, South Africa	HUAWEI CLOUDS	9/10
18	*.44.39.*	/20	VM in cloud	Istanbul, Turkey	HUAWEI CLOUDS	10/10
19	*.46.221.*	/17	VM in cloud	Beijing, China	HUAWEI CLOUDS	10/10
20	*.195.177.*	/18	VM in cloud	Shandong, China	Tencent Cloud	10/10
21	*.36.245.*	/16	Wi-Fi	Virginia, United States	Verizon Business	10/10
22	*.66.18.*	/16	Wi-Fi	Virginia, United States	Verizon Business	10/10
23	*.198.141.*	/22	Wi-Fi	Washington, United States	Cox Communications Inc.	10/10
24	*.223.36.*	/15	Wi-Fi	California, United States	Comcast Cable Communications, LLC	9/10
25	*.58.21.*	/16	Wi-Fi	Burnaby, Canada	Simon Fraser University	9/10
26	*.138.139.*	/10	Wi-Fi	Hesse, Germany	Deutsche Telekom AG	8/10
27	*.92.167.*	/20	Wi-Fi	Kerala, India	Bharat Sanchar Nigam LTD	10/10
28	*.129.63.*	/18	Wi-Fi	Beijing, China	China Unicom Beijing	10/10
29	*.47.33.*	/24	Wi-Fi	Dhaka, Bangladesh	Link3 Technologies Limited	10/10
30	*.114.95.*	/14	Wi-Fi	Yunnan, China	China Telecom	10/10

communication between internal and external hosts. While the differences among various NAT types stem from their use cases, they all fundamentally center on managing the session mappings for routing network traffic. Our attack leverages the TCP session mappings, enabling Internet attackers to craft TCP packets for manipulation. Consequently, the success of our attack hinges on the target NAT device maintaining stateful TCP connection mappings without enforcing legitimacy checks on the received TCP packets. This vulnerability may exist in NAT devices implementing various NAT types, including SNAT, DNAT, CGNAT, and others.

Through our empirical evaluations, we identify that SNAT and CGNAT devices, widely deployed in the real-world, exhibit this vulnerability significantly. A large number of SNAT routers (as shown in Table III) and CGNAT devices from ISPs (as shown in Table VI) are affected by our attack[10]. Regarding DNAT, we establish an experimental environment using the OpenWrt 22.03 router firmware as a NAT device to enable the DNAT functionality. Our experimental results reveal that the DNAT implementation in OpenWrt 22.03 is vulnerable. Consequently, it is highly likely that many NAT devices with the DNAT feature in the real world are also susceptible. This is particularly concerning because OpenWrt serves as the foundational firmware for over 20 derivative projects[11]. Additionally, NAT-enabled IoT CPE routers may be at risk, as the vulnerability persists in many underlying router firmware. For instance, our testing shows that the ZTE 4G CPE 2 PRO router is vulnerable.

B. *Impacts on IPv6*

NAT is also widely used in IPv6 networks, including NAT64, NAT66, and NAT46. Therefore, we discuss the impact of our attack on these IPv6 NAT technologies. Using the widely adopted OpenWrt 22.03 firmware as a NAT device, we construct IPv6 networks for NAT64, NAT66, and NAT46, and test whether the two vulnerabilities revealed in this paper (i.e., the side channel in the PMTUD mechanism for NAT identification and the removal of NAT session mappings via crafted TCP `RST` packets) are still effective in these networks. For the side channel vulnerability, we identify that this fundamental vulnerability consistently exists in three IPv6-related NAT scenarios. First, in our NAT64 network setup, an IPv6-enabled client accesses our server, deployed in Tencent Cloud, through the OpenWrt 22.03-equipped NAT64 device holding a public IPv4 address. The NAT64 device translates the address information from IPv6 space to IPv4 space and

[10]In our tests, certain CGNAT devices, e.g., China Unicom's CGNAT devices in Beijing (Table VI rows 6 and 28), maintain stateful TCP connection mappings without performing legitimacy checks, leaving the mappings vulnerable to malicious removals. Interestingly, these devices do not adhere to the TCP specifications for generating subsequent `RST` packets to tear down the victim server's sockets. Nonetheless, even in this scenario, the DoS attack remains effective because the removal of the NAT device's mappings disrupts synchronization between the server and the NATed clients, resulting in dropped data and exceptions.

[11]OpenWrt followed by more than 20 derivative projects provides firmware for over 2,000 types of NAT devices (https://openwrt.org/toh/start).

also translates the server's response back to IPv6 space. According to our method in Fig. 3, after the server issues the ICMPv4 "Fragmentation Needed and DF Set" message, the NAT64 device translates this message to an ICMPv6 "Packet Too Big" message (ICMPv6 error message with `Type`=2 and `Code`=0) and then forwards this ICMPv6 message to the client. Consequently, the client updates its path MTU value to the server. From the server's perspective, the size of returned TCP packets is then reduced accordingly, while the ping reply packets from the same source remain unaffected. By contrast, path MTU desynchronization does not occur if the requester to the server is a separate IPv4 or IPv6 host. Similarly, in our NAT46 and NAT66 network setups, the ICMPv6 "Packet Too Big" message issued by the server also triggers path MTU desynchronization, regardless of whether the NAT46 device forwards the translated ICMPv4 "Fragmentation Needed and DF Set" message to the IPv4 client or the NAT66 device directly forwards the ICMPv6 message to the IPv6 client.

With regard to the vulnerability of NAT session mapping removal, we also test three IPv6 NAT networks (i.e., networks linked by NAT64, NAT46, and NAT66 devices respectively, with the NAT devices equipped with the popular OpenWrt 22.03 firmware) to determine whether an off-path attacker impersonating the server and forging a TCP `RST` packet with an arbitrary sequence number can deceive the NAT device into removing the corresponding TCP session mappings with the server. The experimental results show that all three NAT setups in our tests are vulnerable because the session mapping maintenance module within the NAT implementation does not enforce TCP sequence number checking. This vulnerability is independent of the specific IP protocol (i.e., regardless of whether it is IPv4 or IPv6).

C. Countermeasures

Responsible Disclosure. We reported the side channel that can be exploited to identify NAT devices to IETF. Currently, we are discussing this issue with the IETF security area directors, and we are told that the issue has been added as an item for an upcoming IETF Security Area meeting. Besides, we reported our attack to the affected OS communities and identified ISPs. Following our disclosure, FreeBSD, China Telecom, China Unicom, China Mobile, ALICLOUD, HUAWEI CLOUDS, and Tencent Cloud confirmed the occurrence of our DoS attack stemming from their NAT implementations. These entities also recognized our efforts in enhancing the security of their services. Furthermore, two prominent NAT firmware platforms, OpenWrt and Asuswrt, verified our DoS attack. The underlying vulnerability within the core NAT firmware extends its impact to a significant number of downstream NAT vendors, including NETGEAR, Linksys, Huawei, TP-Link, H3C, RuiJie, and Xiaomi, among others. We responsibly disclosed this vulnerability to the affected vendors and received acknowledgments from Linksys, Huawei, TP-Link, and Xiaomi. We obtained 5 CVE/CNVD identifiers (CVE-2023-6534, CVE-2023-31635, CNVD-2023-60783, CNVD-2023-30194, CNVD-2023-30193) for the disclosed vulnerabilities. In addition, we recommend our countermeasures to prevent the identified DoS attack.

Fixing the Side Channel in PMTUD. The root cause of the side channel, which allows attackers on the Internet to remotely identify NAT devices, is a design flaw in NAT's PMTUD mechanism. Specifically, this flaw arises from insufficient consideration of the PMTUD mechanism in NAT specifications, leading to information leakage. This lack of synchronization in the path MTU values from the same source IP address results in exploitable information leakage, enabling the identification of NAT devices. NAT technology is extensively employed in diverse network scenarios, such as 4G LTE/5G, Wi-Fi, IoT, ICS, and others. Consequently, it is crucial to address this information leakage effectively and prevent any further attacks. We propose enhancing the design of PMTUD for NAT networks by mandating that NAT devices not only translate ICMP "Fragmentation Needed and DF Set" messages to internal clients but also synchronize and update their own path MTU values for the server. This approach ensures the consistency of path MTU values from the same source IP address, effectively mitigating potential security risks arising from information leakage. Note that NAT devices may face a challenge in verifying the legitimacy of the received ICMP error messages before updating their own path MTU values. Addressing this requires NAT devices to maintain more information, e.g., the window range of TCP connections, enabling the validation for the received ICMP error message's legitimacy. We have reported this issue to the IETF.

Enforcing More Strict Checks on TCP. The vulnerability of NAT mapping removal arises from the absence of essential checks for TCP `RST` packets in NAT implementations across various scenarios. Although some recent OSes and router firmware have implemented security mechanisms to verify the legitimacy of received TCP `RST` packets, the majority of real-world NAT devices—such as commercial NAT routers—often do not perform these security checks. This oversight can potentially lead to DoS attacks. A straightforward solution is to mandate the NAT devices implement more strict checks on the received TCP packets, particularly TCP control packets like `RST`, by verifying if the carried sequence number falls within the acceptable range of the corresponding TCP connection. Note that more strict validations may potentially introduce vulnerabilities, making the NAT device susceptible to attacks. Due to limited resources in NAT devices, attackers could exploit this by flooding the device with a high volume of forged `RST` packets, compelling it to conduct additional checks and depleting excessive resources. Nonetheless, we advocate for the implementation of more strict checks on the received TCP packets by NAT devices to mitigate potential attacks, especially considering the crucial role that NAT plays in the existing Internet infrastructure. A prototype based on OpenWrt 22.03 confirms the effectiveness of our countermeasure by enabling TCP window checks, effectively thwarting remote attackers' manipulation of HTTP sessions issued from NATed clients. **W**e modify the kernel of our NAT device (running

OpenWrt 22.03) to check the sequence number of received TCP packets before responding to them. This prevents off-path attackers from removing TCP session mappings by sending crafted out-of-band TCP `RST` packets with incorrect sequence numbers, thereby thwarting the DoS attack. As a result, HTTP access from our NATed clients to the remote HTTP server remains unaffected in our tests.

IX. Related Work

In this section, we review previous related works from two aspects: NAT issues and DoS attacks.

NAT Issues. NAT plays a crucial role in network connectivity, making it a prominent focus for academic studies. Prior works show that malicious insiders of NAT networks may consume limited resources of the NAT device to construct a DoS attack [6], [32], [50]. For example, a malicious NAT client may establish multiple (as many as 65,535) TCP connections with a target server, thus preventing other clients from being assigned available ephemeral ports to initiate TCP connections to the server. Distinctively, in our attack, the attacker is not limited by network topology, residing remotely on the Internet rather than within the local NAT network. By rewriting NAT session mappings and then intercepting the victim client's TCP packets in Wi-Fi Networks, Yang *et al.* exploited sequence number leakage to hijack TCP connections [51]. They also discussed constructing remote TCP DoS attacks. However, a significant challenge for their DoS attack is the precise identification of a remote NAT-enabled Wi-Fi router. We uncover a fundamental side channel in the PMTUD mechanism of NAT specifications that can be exploited to remotely identify NAT devices on the Internet. Furthermore, we explore NAT implementation disparities across native OSes, various router firmware, and commercial routers, revealing our DoS attack's fundamental impact on a wide range of real-world NAT networks, i.e., beyond Wi-Fi to 4G LTE/5G cellular networks, Cloud VPS networks, and more.

Bellovin proposed a method to identify a NAT network and the associated hosts by examining the distributions of IP IDs originating from a particular IP address [3]. However, this approach may encounter difficulties when dealing with packets that have zero or random IP IDs, leading to potential complications. Other OS features such as the Don't Fragment (DF) flag, the Time-To-Live (TTL) field, the TCP window size, the TCP source port, the initial sequence number (ISN), and the SYN packet size were also utilized to identify NAT networks and determine the number of hosts behind the network [4], [34], [28], [18], [17], [45]. Essentially, the concept behind these methods is that if an IP address is associated with multiple hosts, it could suggest the presence of a NAT device with distinct OS features behind it. However, it is easy to see that these methods may generate substantial false alarms in situations where a computer has multiple OSes installed or when traffic obfuscation techniques are employed.

By exploiting middle-box protocols (e.g., UPnP), an attacker could connect devices behind NAT gateways to identify NAT networks. However, this technique requires the deployment of application-layer middle-box protocols or the enabling of malicious scripts on devices behind NAT gateways [38]. Gokcen *et al.* employed the Naive Bayes learning technique as a classifier to identify NAT devices based on given traffic traces. However, this approach heavily depends on the quality of the collected datasets [16]. NAT was also exploited to execute DNS hijacking. For example, Herzberg *et al.* demonstrated techniques for bypassing source port randomization in NAT networks and hijacked a local DNS resolver [20].

DoS Attacks. DoS attacks and network traffic manipulations have been extensively studied in recent years [10], [39], [31], [44]. Feng *et al.* discovered a side channel in the new mixed IP ID assignment policy that can be exploited by off-path attackers to terminate an SSH session [12], [13]. Cao *et al.* , leveraging a side channel in the challenge ACK mechanism, uncovered a TCP hijacking attack capable of poisoning or resetting victim TCP connections [9], [8]. Fortunately, most of these previous attacks have been addressed by the security community [12], [13], [8], [9]. Additionally, recent research has explored low-rate TCP-targeted DoS attacks [23], [21], [40], [19] and congestion-based attacks on intermediate links [41], [30]. In contrast to these studies, our research focuses on a covert DoS attack specifically targeting NAT networks. Previous works [43], [7] demonstrated that TCP connections can be terminated by crafting `RST` packets, but they require the attacker to be physically present on the host to eavesdrop on the correct sequence number for crafting an acceptable `RST` packet. In contrast, our attack can be executed remotely by off-path attackers on the Internet.

X. Conclusion

In this paper, we conduct an empirical study on remote DoS attacks against NAT networks. We uncover a side channel stemming from insufficient considerations of the PMTUD mechanism in the NAT specifications. This side channel leads to the remote identification of NAT devices on the Internet. Furthermore, we demonstrate that Internet attackers may manipulate TCP connection mappings in various NAT devices using carefully crafted TCP `RST` packets. This manipulation opens the door to remote DoS attacks targeting NAT networks. Through extensively empirical studies conducted on various NAT implementations and real-world NAT networks, we highlight the significance of our DoS attack. Finally, we develop countermeasures against the attack.

Acknowledgment

We thank the anonymous reviewers for their insightful comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3102303, the National Science Foundation for Distinguished Young Scholars of China under No. 62425201, the Science Fund for Creative Research Groups of the National Natural Science Foundation of China under No. 62221003, the Key Program of the National Natural Science Foundation of China under No. 61932016 and No. 62132011. Ke Xu is the corresponding author of this paper.

REFERENCES

[1] M. Allman, V. Paxson, and E. Blanton, "Rfc 5681: Tcp congestion control," Internet Requests for Comments, Internet Engineering Task Force, RFC 5681, September 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5681.txt

[2] F. Audet and C. Jennings, "Network address translation (nat) behavioral requirements for unicast udp," Internet Requests for Comments, Internet Engineering Task Force, RFC 4787, January 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4787.txt

[3] S. M. Bellovin, "A technique for counting natted hosts," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, 2002, pp. 267–272.

[4] R. Beverly, "A robust classifier for passive tcp/ip fingerprinting," in *Passive and Active Network Measurement: 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20, 2004. Proceedings 5*. Springer, 2004, pp. 158–167.

[5] E. S. Biagioni, "Fragmentation and reassembly," https://www2.hawaii.edu/~esb/2006fall.ics651/sep11.html, Accessed December 2023.

[6] K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh, "Nat behavioral requirements for tcp," Internet Requests for Comments, Internet Engineering Task Force, RFC 5382, October 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5382.txt

[7] J. Bruandet, "Killcx: close a tcp connection," https://killcx.sourceforge.net/, Accessed December 2023.

[8] 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 *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 209–225.

[9] ——, "Off-path tcp exploits of the challenge ack global rate limit," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 765–778, 2018.

[10] S. Cho, R. Fontugne, K. Cho, A. Dainotti, and P. Gill, "Bgp hijacking classification," in *2019 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2019, pp. 25–32.

[11] W. Eddy, "Transmission Control Protocol (TCP)," Internet Requests for Comments, Internet Engineering Task Force, RFC 9293, August 2022. [Online]. Available: http://www.rfc-editor.org/rfc/rfc9293.txt

[12] X. Feng, C. Fu, Q. Li, K. Sun, and K. Xu, "Off-path tcp exploits of the mixed ipid assignment," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, p. 1323–1335.

[13] 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 Transactions on Networking*, vol. 30, no. 1, pp. 409–422, 2021.

[14] X. Feng, Q. Li, K. Sun, Z. Qian, G. Zhao, X. Kuang, C. Fu, and K. Xu, "Off-path network traffic manipulation via revitalized icmp redirect attacks," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2619–2636.

[15] X. Feng, Q. Li, K. Sun, K. Xu, B. Liu, X. Zheng, Q. Yang, H. Duan, and Z. Qian, "Pmtud is not panacea: Revisiting ip fragmentation attacks against tcp," in *Network and Distributed System Security Symposium (NDSS)*, 2022.

[16] Y. Gokcen, V. A. Foroushani, and A. N. Z. Heywood, "Can we identify nat behavior by analyzing traffic flows?" in *2014 IEEE Security and Privacy Workshops*. IEEE, 2014, pp. 132–139.

[17] F. Gont and S. Bellovin, "Defending against sequence number attacks," Internet Requests for Comments, Internet Engineering Task Force, RFC 6528, February 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6528.txt

[18] F. Gont and P. Srisuresh, "Security implications of network address translators (nats)," https://www.ietf.org/proceedings/73/slides/behave-12.pdf, Accessed December 2023.

[19] A. Herzberg and H. Shulman, "Stealth dos attacks on secure channels." in *NDSS*, 2010.

[20] ——, "Security of patched dns," in *Computer Security–ESORICS 2012: 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings 17*. Springer, 2012, pp. 271–288.

[21] S. Jero, M. E. Hoque, D. R. Choffnes, A. Mislove, and C. Nita-Rotaru, "Automated attack discovery in tcp congestion control using a model-guided approach," in *NDSS*, 2018.

[22] Keirstenbrager, "Ip fragmentation vs ip reassembly," https://www.keirstenbrager.tech/ip-fragmentation-vs-ip-reassembly/, Accessed December 2023.

[23] A. Kuzmanovic and E. W. Knightly, "Low-rate tcp-targeted denial of service attacks: The shrew vs. the mice and elephants," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 75–86.

[24] I. Livadariu, K. Benson, A. Elmokashfi, A. Dhamdhere, and A. Dainotti, "Inferring carrier-grade nat deployment in the wild," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2249–2257.

[25] M. Luckie, R. Beverly, R. Koga, K. Keys, J. A. Kroll, and k. claffy, "Network hygiene, incentives, and regulation: Deployment of source address validation in the internet," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 465–480.

[26] J. McCann, S. Deering, and J. Mogul, "Path MTU Discovery for IP version 6," Internet Requests for Comments, Internet Engineering Task Force, RFC 8201, July 2017. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8201.txt

[27] ——, "Path mtu discovery for ip version 6," Internet Requests for Comments, Internet Engineering Task Force, RFC 1981, August 1996. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1981.txt

[28] T. Miller, "Passive os fingerprinting: Details and techniques," http://www.ouah.org/incosfingerp.htm, Accessed December 2023.

[29] J. Mogul and S. Deering, "Path mtu discovery," Internet Requests for Comments, Internet Engineering Task Force, RFC 1191, November 1990. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1191.txt

[30] T. Muoi, K. Min, Suk, H. Hsu-Chun, C. Wei-Hsuan, T. Shu-Po, and W. Yu-Su, "On the feasibility of rerouting-based ddos defenses," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 798–813.

[31] G. Nakibly, A. Sosnovich, E. Menahem, A. Waizel, and Y. Elovici, "Ospf vulnerability to persistent poisoning attacks: a systematic analysis," in *Proceedings of the 30th Annual Computer Security Applications Conference*, 2014, pp. 336–345.

[32] S. D. Nguyen, M. Mimura, and H. Tanaka, "Slow-port-exhaustion dos attack on virtual network using port address translation," in *2018 Sixth International Symposium on Computing and Networking (CANDAR)*. IEEE, 2018, pp. 126–132.

[33] A. Pathak, "Network address translation (nat): An introduction," https://geekflare.com/network-address-translation/, Accessed December 2023.

[34] P. Phaal, "Detecting nat devices using sflow," https://sflow.org/detectNAT/, Accessed December 2023.

[35] C. Point, "What is network address translation (nat)?" https://www.checkpoint.com/cyber-hub/network-security/what-is-network-address-translation-nat/, Accessed December 2023.

[36] J. Postel, "Transmission control protocol," Internet Requests for Comments, Internet Engineering Task Force, RFC 793, September 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[37] R. S. Raman, A. Stoll, J. Dalek, R. Ramesh, W. Scott, and R. Ensafi, "Measuring the deployment of network censorship filters at global scale." in *NDSS*, 2020.

[38] T. Rytilahti and T. Holz, "On using application-layer middlebox protocols for peeking behind nat gateways." in *NDSS*, 2020.

[39] P. Sermpezis, V. Kotronis, A. Dainotti, and X. Dimitropoulos, "A survey among network operators on bgp prefix hijacking," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 1, pp. 64–69, 2018.

[40] A. Shevtekar, K. Anantharam, and N. Ansari, "Low rate tcp denial-of-service attack detection at edge routers," *IEEE Communications Letters*, vol. 9, no. 4, pp. 363–365, 2005.

[41] J. M. Smith and M. Schuchard, "Routing around congestion: Defeating ddos attacks and adverse network conditions via reactive bgp routing," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 599–617.

[42] M. Smith and R. Hunt, "Network security using nat and napt," in *Proceedings 10th IEEE International Conference on Networks (ICON 2002). Towards Network Superiority (Cat. No. 02EX588)*. IEEE, 2002, pp. 355–360.

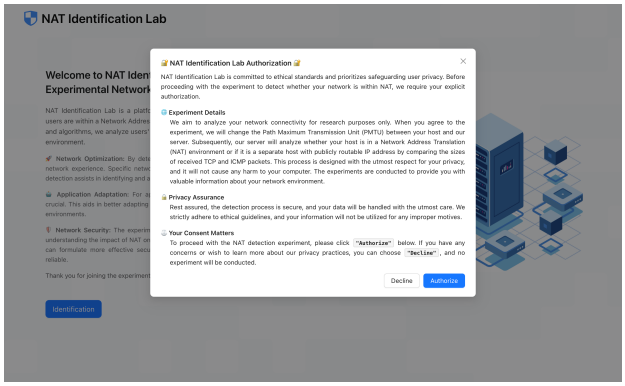
[43] D. Song, "tcpkill," https://www.kali.org/tools/dsniff/, Accessed December 2023.

[44] Y. Song, S. Gao, A. Hu, and B. Xiao, "Novel attacks in ospf networks to poison routing table," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

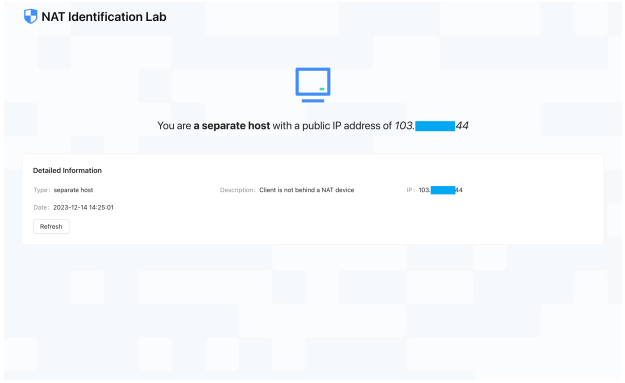
[45] P. Srisuresh and K. Egevang, "Traditional ip network address translator (traditional nat)," Internet Requests for Comments, RFC Editor, RFC 3022, January 2001, http://www.rfc-editor.org/rfc/rfc3022.txt.
[46] P. Srisuresh and M. Holdrege, "Ip network address translator (nat) terminology and considerations," Internet Requests for Comments, RFC Editor, RFC 2663, August 1999, http://www.rfc-editor.org/rfc/rfc2663.txt.
[47] Stream, "Webrtc ip leaks: Should you still be worried?" https://getstream.io/blog/webrtc-ip-leaks/, Accessed December 2023.
[48] Surfshark, "Check for webrtc leaks," https://surfshark.com/webrtc-leak-test/, Accessed December 2023.
[49] L. Torvalds, "netfilter: conntrack: tcp: only close if rst matches exact sequence," https://github.com/torvalds/linux/commit/be0502a3f2e94211a8809a09ecbc3a017189b8fb, Accessed December 2023.
[50] N. Winemiller, *NAT Denial of Service An Analysis of Translation Table Behavior on Multiple Platforms*. Rochester Institute of Technology, 2012.
[51] Y. Yang, X. Feng, Q. Li, K. Sun, Z. Wang, and K. Xu, "Exploiting sequence number leakage: Tcp hijacking in nat-enabled wi-fi networks," in *NDSS*, 2024.

APPENDIX

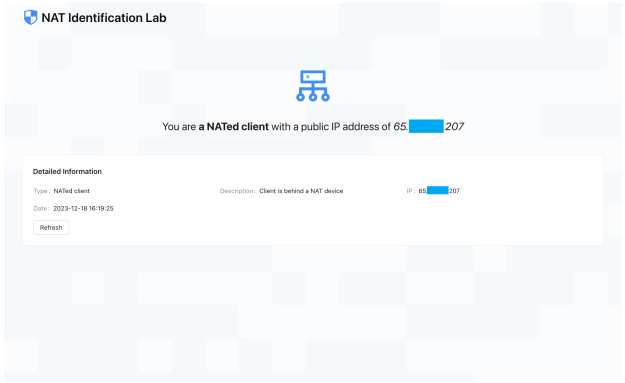
A. Snapshots of NAT Identification on the Internet



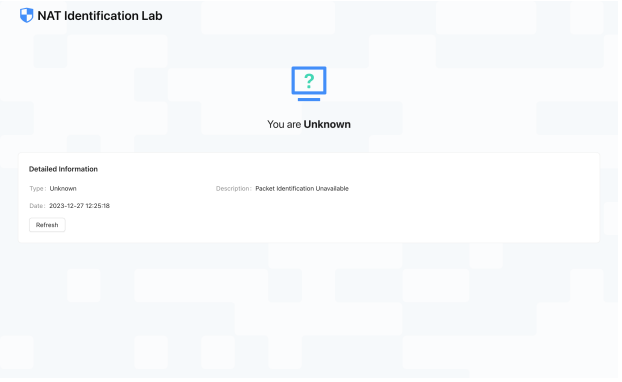
(a) Requiring the requester's approval to proceed with our experiment.



(b) The identification result is sent back to the separate host.



(c) The identification result is sent back to the NATed client.



(d) The identification result is unknown.

Fig. 11. Snapshots of NAT identification results, observed by the NATed client and the separate host in their web browsers, respectively, after clicking the URL to proceed with our experiment.