FragJam: DoS Attacks Using IP Reassembly Congestion

Yepeng Pan *Christian Rossow*
CISPA Helmholtz Center for Information Security

Abstract

UDP, one of the major transport protocols for multiple popular services such as DNS and video conferencing, is an important component of today's network. Its reliance on IP fragmentation is known to cause both security and reliability issues. To avoid fragmentation, UDP-based applications hence usually limit the payload size. Currently, Linux's IP fragment reassembling algorithm relies on a per-network-namespace buffer size limit. That is, a classic resource-exhaustion DoS attack against UDP services relying on IP fragmentation is possible. However, the fragility of present real-world services has not yet been thoroughly researched.

In this paper, we examine the practicability of such an IP-fragmentation-based DoS attack against real-world providers of popular UDP-based services, including VPN, video conferencing, and RADIUS. We assess the attack from both the client and service provider perspectives. On the server side, we observe that many real-world service providers fragment the server-to-client traffic with respect to artificially small path MTUs when receiving attacker forged ICMP *Fragmentation Needed* messages. On the client side, we show the chance of dropping server-to-client fragmented traffic by flooding Linux-based NAT gateways with bogus fragments. Through simulated attacks, we demonstrate that the fragmentation-based DoS attack is realistic, affecting various providers such as Zoom and ExpressVPN. We conduct a comprehensive disclosure to affected parties and suggest possible mitigations.

1 Introduction

IP fragmentation is an important mechanism that ensures network availability. When a packet's size exceeds the maximum transmission unit (MTU) of a link, it can be fragmented at the IP layer to match MTU constraints. When receiving fragmented packets, the recipient needs to reassemble the fragments to obtain the original packet.

However, IP reassembly is inefficient and prone to a series of vulnerabilities [20, 26, 34, 68–73]. In particular, back in 2003, Kaufman and Perlman reported that NetBSD and Solaris OS are vulnerable to a resource exhaustion DoS attack [44]. An attacker can congest the reassembly buffer with IP fragments and block the reassembling of future legitimate fragmented packets, thereby preventing the IKE setup, as it requires sending fragmented large packets. Since a patch in 2018, also Linux' reassembling algorithm is prone to a similar resource-exhaustion DoS attack [12, 51]. Acknowledging the threat, the kernel developers conclude that the problem of reassembly buffer congestion is non-fixable: "*IP defrag units can easily be flooded by attackers, no strategy can guarantee that non malicious traffic can make it*" [51].

While the general attack vector is publicly documented, little is known about its potential impact. Since only fragmented packets would be blocked by the DoS attack, at first sight, the risk appears mild, as the community has been actively addressing IP fragmentation issues. For example, RFC 8900 [4] suggests reducing the reliance on IP fragmentation. This was particularly conducted in DNS, which by now employs multiple methods to avoid fragmentation [14, 24]. In general, the occurrence of IP fragmentation also decreases over years, from 0.68% in 2001 to 0.046% in 2012 [41, 56, 61, 80, 81].

In this paper, we show that the threat of reassembly buffer congestion is widely underestimated – likely as developers ignore the ease in which attackers can provoke fragmentation. In the FragJam attack, an off-path attacker blocks UDP-based services by causing all fragments to be dropped. Figure 1 sketches the general attack workflow. An attacker first sends many fragments to congest the global IP reassembly buffer of a middlebox (e.g., a NAT gateway) on path between the target client and server ①. Once the buffer no longer accepts new fragments, the attacker forces the sender to downgrade the MTU ② such that UDP packets sent to the target client will be fragmented ③. Consequently, any larger (now fragmented) response will be dropped ④, causing a DoS attack.

Focusing on real-world feasibility and impact, we evaluate the FragJam attack from both the client and server perspectives and make the following contributions. On the client side, we examine whether home routers use an affected Linux ker-

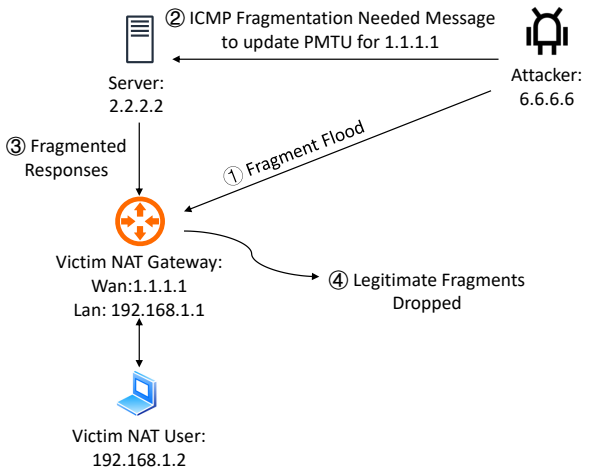

Figure 1: Off-Path FRAGJAM Attack.

nel and thus can be congested to drop legitimate fragmented packets. Such routers, when used as a NAT gateway (which requires reassembly to support address translation [4]), expose the risk that legitimate fragmented packets of all its users can be dropped by the FRAGJAM attack. On the server side, targeting popular UDP-based services including video conferencing, RADIUS authentication, and VPNs, we examine which senders can be forced to send fragmented packets. To this end, we show how an attacker can use forged ICMP *Fragmentation Needed* messages notifying the server an artificially small server-to-client path MTU to trigger fragmentation. Though such ICMP errors are known to be exploited in attacks such as DNS cache poison [34], and are now ignored by multiple popular DNS server implementations [24], all evaluated services would still send fragments, including widely used providers such as Zoom and Webex (video conferencing), Jumpcloud and VendorA[1] (RADIUS authentication), ExpressVPN and Proton VPN (VPN).

Finally, we demonstrate the FRAGJAM attack end-to-end against both real-world service providers and open-source implementations of said services, leveraging a local NAT setup to adhere to ethics. We find that the tested services can indeed be blocked by FRAGJAM, and most implementations also cannot recover from the packet loss incurred by the attack. Shown the threat and practicality of the FRAGJAM attack, we notify both the router vendors and service providers regarding the vulnerability and discuss possible mitigations.

The rest of the paper is organized as follows. In Section 2, we provide background information on IP fragmentation and the current Linux implementation. In Section 3, we present the general threat model and attack methodology. In Section 4, for each assessed service, we provide the detailed attack setups and conduct evaluations and measurements to confirm the practicality of attacks against these services. In Section 5, we discuss mitigations, attack variants, and the disclosure status of the vulnerabilities. Finally, in Sections 6 and 7, we provide the related work and the conclusion.

[1]Anonymized under request of the vendor.

2 Background

In this section, we introduce the basics of IP fragmentation, path MTU discovery, and fragmentation reassembling in Linux. This work focuses on IPv4; for IPv6 see Section 5.2.

2.1 IP Fragmentation and ICMP PTB

Each Internet path is constrained by the number of bytes it can convey in a single IP packet, known as the Path MTU (PMTU) [4]. For any given path, the PMTU is equal to the smallest of its link MTUs. When an IP packet cannot be transmitted due to the PMTU constraint, IP fragmentation can split it into fragments that fit the PMTU. In IPv4, packet fragmentation can be performed by on-path routers unless the packet has "*Don't Fragment*" (DF) flag set. If a packet with the DF flag set is too big to be transmitted, an on-path router drops the packet and notifies the sender about the small MTU using an ICMP "*Fragmentation Needed*" message, which we abbreviate as ICMP PTB ("packet too big") in the rest of the paper. Upon receiving an ICMP PTB, the sender would fragment an overly-large UDP packet to fit the new PMTU, unless non-default socket options or configurations are applied (see Section 3.4).

Operating systems usually restrict the minimum PMTU learned from such ICMP messages. In Linux, the `sysctl` configuration *net.ipv4.route.min_pmtu* sets the minimum acceptable PMTU to 552 bytes by default. We will elaborate further on Linux ICMP PTB handling and packets framgnetaiton behavior by applications in Section 3.3 and Section 3.4.

In practice, TCP packets rarely fragment, as the maximum segment size (MSS) can be adjusted according to the PMTU. Even if fragmentation is triggered by malicious ICMP PTBs, TCP fragmentation is temporary [20]. We thus focus on UDP-based applications that are more prone to fragmentation.

2.2 IP Reassembly in Linux

Recipients need to reassemble IP fragments based on the IP addresses, the protocol (e.g., TCP or UDP), and the IPID before handing the IP packets to upper layers. In this work, we focus on the reassembly implementation in Linux. Since a patch in 2018 [49], Linux has utilized a tail-drop behavior for fragments reassembling. In Appendix Listing 1, we provide a simplified version of the Linux reassembly implementation. The reassembly buffer space usage is tracked by a per-network-namespace counter that accumulates the *truesize* of all buffered *sk_buffs* (socket buffer, data structure to represent network packets) and the fragment queue struct sizes. The actual cost for buffering an IP fragment depends on multiple factors, e.g., paging, driver. When a new

IP fragment arrives, the kernel first checks whether the reassembly memory usage exceeds the limit.[2] If so, the new fragments are dropped directly without further processing. By default, the memory limit for reassembling is 4 MB (controlled by *net.ipv4.ipfrag_high_thresh*) and the default timeout of buffered fragments is 30 seconds (controlled by *net.ipv4.ipfrag_time*). Unfortunately, attackers can easily exhaust this limit by congesting the reassembly buffer with long-lasting fragments, effectively hindering any benign fragmented traffic from arriving at the targeted host.

Worse, the end recipients are not the only affected party. Linux-based on-path middleboxes using connection tracking also utilize the same reassembly implementation at the *PREROUTING* phase (see Appendix Listing 2). For instance, a NAT gateway requires the TCP/UDP header to perform address translation, while this header is only available in the first fragment. Thus, by flooding a NAT gateway with fragments that can never be reassembled, an attacker can block the legitimate fragmented traffic for all NAT users.

The Linux reassembling algorithm's limitation was already reported by Man [51] and Corallo [12] in 2021 – without a planned patch. As also acknowledged by the Linux developers [51], any reassembling algorithm is likely to be subject to DoS attacks. This underlines the need for a systematic vulnerability analysis showing the impact of this issue.

3 Methodology

In this section, we first propose the general threat model. Next, we introduce the strategies for sending fragment floods and forging ICMP PTBs to trigger fragmentation. Finally, we discuss the fragmentation behavior of applications.

3.1 Threat Model

In FRAGJAM, an off-path attacker aims to congest the IP reassembly buffer of a target client's gateway such that future fragmented packets sent by a target service (e.g., video conferencing service) will no longer arrive at the client. Illustrated in Figure 1, we assume that the victim client is a user of a home or enterprise NAT network. The NAT gateway uses an affected Linux kernel, and its public IP address is known to the off-path attacker. As the first step, the attacker sends a flood of fragments to the NAT gateway to congest its reassembly buffer. To ensure that these fragments can never be reassembled, the attacker never completes the IP packet (i.e., omits at least one fragment). In the second step, the attacker forces the target service to fragment its server-to-client traffic, which will then be dropped on the NAT gateway due to congestion. To achieve this, the attacker forges an ICMP PTB, informing the server about a small PMTU of the server-to-client path, such that the server starts fragmenting packets.

[2]Linux used to only check the memory limit when a new fragment queue needs to be created [48], but this is fixed since LTS v4.19.

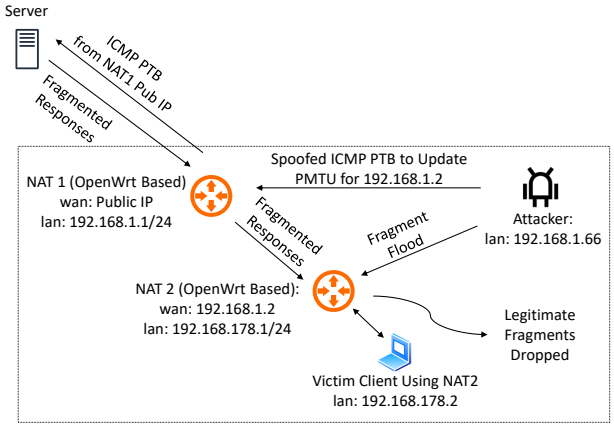

Figure 2: Evaluation Setup.

The threat model requires the attacker to have the following capabilities: (1) The attacker can send IP-spoofed or non-spoofed fragments to congest the reassembly buffer of the NAT gateway (or, to generalize, of a connection-tracking middlebox using an affected Linux kernel on the server-to-client path). (2) The attacker needs the IP spoofing capability to send forged ICMP PTBs to update the server's PMTU cache for the victim client and thus trigger fragmentation.[3]

Through the evaluation, for ethical reasons, we simulate the off-path attack in a controlled environment, as shown in Figure 2. In our setup, the attacker resides in the same network as the victim's NAT gateway. This setup thus mimics the scenario where an off-path attacker can flood the victim's NAT gateway and send spoofed ICMP PTBs from its public IP. Both of the NAT1 and NAT2 routers in our setup use OpenWrt-based routers, with a v5.15 Linux kernel.

3.2 Fragmentation Flood Strategies

As discussed earlier, the attacker needs to send a flood of fragments to the target NAT gateway or endpoint to congest its fragments reassembling buffer. We propose two strategies for performing flooding: the restless flood and the lossy flood.

Restless Flood: Since incomplete fragments are only buffered for 30 seconds by default, the attacker needs to refill the buffer again once the old fragments time out. By increasing the traffic rate, the attacker can minimize the time gap between the old fragments' timeout and the next refill. We implement an attacker that sends ten 572-byte fragments every 1ms, i.e., 10k pps (approximately 45Mbps). To estimate the reassembly success rate under the restless flood strategy, we perform the flooding against an OpenWrt router under our

[3]Sending forged ICMP PTB itself does not require IP spoofing. As any on-path routers could send ICMP PTBs, attackers can also simply use their own IP address to send ICMP PTBs. However, as shown in Section 3.3.1 and Section 4, many real-world servers do not accept a plain ICMP PTB without a matching flow. Consequently, an attacker would need IP spoofing capability to send acceptable ICMP PTBs.

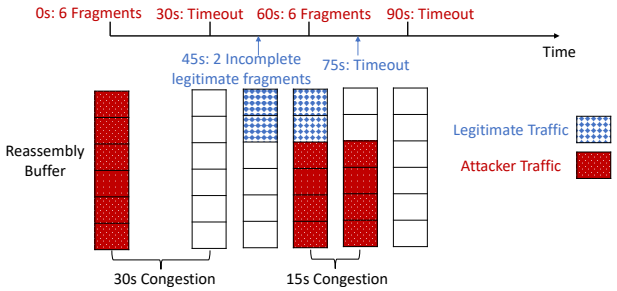

Figure 3: Noise Induced by Legitimate Fragments.

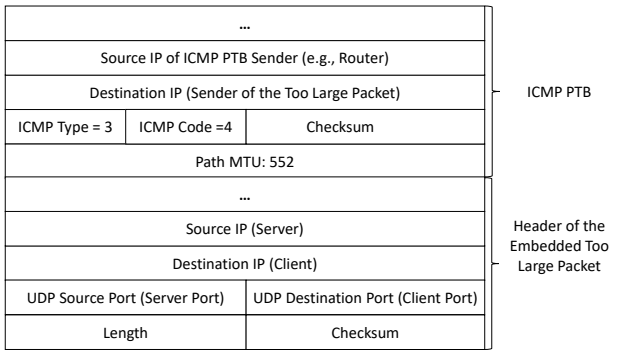

Figure 4: Example ICMP PTB.

control. At the same time, we use a client in the NAT network to receive fragmented UDP packets from a server outside of the NAT network every 0.001s and examine the number of successfully received responses after 10 minutes. Indeed, only less than 1% of the responses can be successfully reassembled during the attack (all fragments in the non-attack case).

Lossy Flood: In a lossy flood, the attacker occasionally allows legitimate fragments to be reassembled. While counterintuitive at first, a lossy flood strategy can help avoid some services from falling back to TCP. To implement the lossy flood, the attacker can continue the 10kpps flood for 29 seconds, then pause for 31 seconds. This way, the attacker's fragments in the reassembly buffer are guaranteed to timeout at the 59th second, and thus allow legitimate fragments to be reassembled. The attacker may also implement the lossy flood using a lower packet rate. For example, the attacker may send 10 fragments every 20ms (500pps). Given a router that can buffer, e.g., 1921 fragments, the attacker would need approximately 4 seconds to congest the buffer, and thus allow legitimate fragments to be reassembled. Similar to the restless flood, we also use a UDP session to confirm that we can indeed allow and block legitimate fragments as expected.

Implication of Incomplete Legitimate Fragments: In our evaluation setup depicted in Figure 2, fragments from servers are first virtually reassembled at NAT1 and only then forwarded to NAT2 (still being fragmented). Due to this, NAT2's reassembly buffer rarely ever contains benign fragments, and if so, only for a short time. In networks with minimal packet loss, such a setup does not introduce bias, as all fragments received on NAT1 can be virtually reassembled and forwarded to NAT2. In our evaluation, we also observe that captures on NAT1 show minimal loss of fragmented packets. However, empty reassembly buffers may not be guaranteed in practice due to reasons such as fragment loss. We thus discuss the impact of a non-empty reassembly buffer.

Consider the simplified example in Figure 3, in which a target router can buffer 6 fragments, and the attacker only sends 6 fragments every 60 seconds. In between the 30-second gap among timeouts and refilling, two legitimate fragments that cannot be reassembled are received (after 45 s). In this case, the next refill by the attacker (after 60 s) can only control four entries, and the legitimate fragments will time out before the attacker fragments, rendering the congestion ineffective for much longer than desired – on top of the non-congested window between 30 s and 60 s, the buffer now also has two free slots between 75 s and 90 s.

Nevertheless, as long as the attacker continuously sends the fragments during the attack, e.g., using the aforementioned 10kpps flooding, the impact of such fragments that time out at unexpected times can be alleviated. That is, the buffer would be congested again shortly when the timeout occurs. We evaluate the strategy's performance against a scenario with consistent incomplete fragments not controlled by the attacker. We start a script that sends 5 incomplete fragments every second to prepare a noisy reassembly buffer 30 s before and continue the fragment injection throughout the simulation. Next, we start the flooding and use the same UDP program to observe the number of fragments that can be reassembled after 10 minutes. Indeed, the attacker can still block over 98% of the fragmented traffic using the 10kpps flood.

3.3 Forcing Senders to Fragment Traffic

UDP-based applications usually avoid sending packets that would fragment in typical network scenarios. However, an attacker can send forged ICMP messages to the server to pretend a small server-to-client PMTU. When accepted, the server updates the PMTU and fragments UDP payloads that exceed the reduced maximum size. There are two ways to forge ICMP PTB messages acceptable to the server, which mainly differ in what the PTB embeds: (i) a UDP packet with the server IP and port as source and the correct client IP with a dummy port as destination (ii) a standard ICMP "*Echo Reply*" with the server IP as source and client IP as destination.

We first describe the PTB message that embeds a UDP packet. This is the common message an on-path router would send when the packet to be forwarded is too large and has the DF flag set. To allow the receiving server to correlate the ICMP message, the ICMP message includes at least the triggering packet's IP header and 8-byte payload [66]. Figure 4 provides an example of an ICMP PTB embedding a server UDP response packet. The ICMP message suggests a MTU

of 552 bytes and embeds a UDP packet from the server to the client, where the embedded UDP packet has the server IP and port as source and the client IP and port as destination. When the server receives the ICMP PTB, it extracts and examines the embedded packet and then reduces the PMTU accordingly if acceptable. In Figure 6 (see appendix), we provide the simplified processing procedure of an incoming ICMP PTB in Linux. If the PTB message embeds a UDP packet, the kernel examines whether the quoted IP and UDP header match an existing UDP socket. For a UDP server, the implementation usually uses an unconnected UDP socket bound to a specific server port. In this case, the Linux kernel only checks whether the quoted UDP source port matches the local listening port and will then update the MTU to the destination client IP in the quoted IP header. Thus, for an off-path attacker, even though the true client port is unknown, the attacker can still use a dummy client port to forge acceptable ICMP PTBs. For a "connected" UDP socket, the kernel further examines whether the client IP and port match those of the connected peer. In this case, the attacker also has to guess the client port.

The second technique does not embed UDP packets but a (forged) "*Echo Reply*" instead, i.e., the standard `ping` response. When facing such an ICMP message, the Linux kernel will update the MTU to the client IP directly – even if no prior "*Echo Request*" was seen. The second embedding technique thus represents an equally strong alternative to UDP embedding, especially for the case of connected UDP sockets.

By default, the destination-specific PMTU learned from an ICMP PTB will be cached globally for 10 min.

3.3.1 Bypassing Server Firewalls

ICMP PTBs built with the earlier methodology can update the PMTU for servers that are not protected by firewalls. However, in practice, servers can be behind firewalls that drop unrelated or invalid ICMP errors. In this case, a plain ICMP PTB without relevant flows would be dropped. In our evaluation, we observed that many real-world servers do not accept plain ICMP PTBs, which we suspect to be such firewall behaviors.

To tackle this issue, attackers can forge an empty UDP packet from the client to the server with a dummy client port before sending the following ICMP PTB embedding the UDP packet with the same dummy client port (see Figure 5). In Linux's connection tracking, such a forged UDP packet can cause the following ICMP error message to be considered as related. Note that the forged UDP packet does not need to contain a valid request nor trigger a server response, which is also usually challenging in practice. Similarly, the attacker can forge an ICMP "*Echo Request*" from the client to the server before sending an ICMP PTB embedding an Echo reply, in case the server does not filter incoming Echo requests.[4]

[4]The setup in Figure 2 requires the attacker to send ICMP PTB with the preceding UDP/Echo-Request. Otherwise, the ICMP PTB will not be translated or forwarded by the NAT1 router. To evaluate whether servers accept plain ICMP PTBs, we send them directly from the NAT1 router.

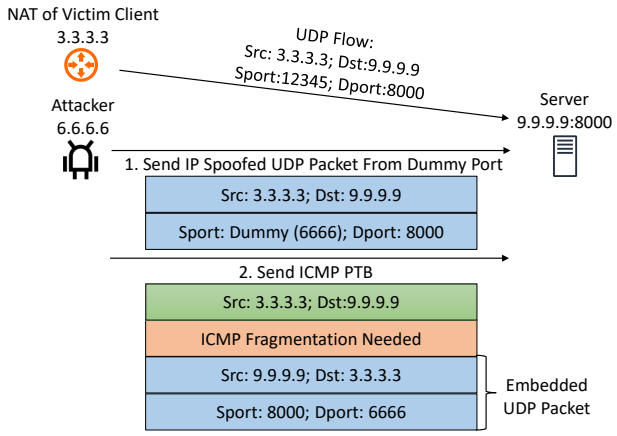

Figure 5: Forge Related ICMP PTB.

In principle, not only a firewall but also a connected UDP socket can filter ICMP PTBs embedding a UDP packet with a dummy client port. Feng et al. [19] proposed a similar method to use a preceding UDP request to trick the server into establishing a UDP socket and thus accept a spoofed ICMP redirect message. However, as UDP servers usually use a wildcard UDP socket, and we never send a valid UDP request or trigger a server response, a connected socket triggered by the spoofed UDP request is less likely in our case.

The described technique requires IP spoofing, as the preceding UDP or ICMP Echo request needs to spoof the victim client's source IP. We provide a more detailed discussion of the IP-spoofing capability in Section 5.3.

3.4 Application Implementation Choices

Linux provides several socket options and system configurations to alter the server's behavior when (i) receiving an ICMP PTB and (ii) wanting to send fragmented packets. In the default setting, servers would accept the aforementioned PTB messages to reduce the PMTU to 552 bytes, and also fragment packets exceeding this MTU. In this section, we investigate how servers or applications that deviate from this default could change the behavior.

Socket Options: The Linux kernel provides the *IP_MTU_DISCOVER* socket option to control the socket behavior when an ICMP PTB is received. When not set explicitly, the default value is *IP_PMTUDISC_WANT*. In Table 1, we summarized the UDP socket (not-connected) behavior when receiving an ICMP PTB embedding a UDP packet under different *IP_MTU_DISCOVER* configurations. Overall, sockets configured with *IP_PMTUDISC_WANT / DONT* are ideal targets and allow fragmentation based on the discovered PMTU.

Note that an ICMP PTB embedding an Echo response operates at the IP layer and will update the PMTU cache regardless

Table 1: *IP_MTU_DISCOVER* Behavior Summary.

*IP_PMTUDISC_**	Update PMTU	Fragment Packet > PMTU
WANT (default)	✓	✓
DONT	✓	✓
DO	✓	✗
PROBE	✓	✗
INTERFACE	✗	✗
OMIT	✗	✗

of the socket option applied to a UDP socket. However, even when the PMTU cache is updated, UDP packets sent from a socket configured with *IP_PMTUDISC_DO / PROBE / INTERFACE / OMIT* do not fragment at the source.

Except for the above socket options to control the fragmentation and ICMP PTB acceptance behavior, applications may also actively fetch the cached PMTU (e.g., using a rtnetlink socket [65] or the *IP_MTU* socket option) and adjust the application-layer payload size to avoid IP fragmentation.

Sysctl Configurations: Several sysctl configurations can also change the server's reaction to ICMP PTBs. *net.ipv4.ip_no_pmtu_disc* can be used to disable or restrict PMTU discovery. For example, it is possible to only accept ICMP PTBs for a TCP/SCTP/DCCP socket, which can verify additional information, such as the sequence number. This configuration is disabled by default. *net.ipv4.min_pmtu* controls the minimum discovered PMTU, which is 552 bytes by default. The timeout of the discovered PMTU is controlled by *net.ipv4.route.mtu_expires*, which defaults to 600s.

Reliable PMTU Discovery: Since the ICMP PTBs can be filtered, a server sending large packets may never be informed of the small PMTU, causing consistent packet loss, which is known as the PMTU blackhole [4]. *Packetization-Layer Path MTU Discovery* (PLPMTUD) [55] is a more robust approach to discover the PMTU without relying on ICMP PTBs. The Linux kernel implements PLPMTUD for TCP, which is controlled by *net.ipv4.tcp_mtu_probing* (disabled by default). When enabled, a server sends TCP segments of varying sizes and infers the PMTU according to the received ACKs. However, PLPMTUD is not directly applicable for UDP. Instead, implementing PLPMTUD for UDP requires support from the application layer.

4 Evaluation

In the previous section, we introduced the general FragJam attack methodology. In this section, we first provide an overview of operating systems (OSes) and router devices that are subject to the reassembling buffer exhaustion vulnerability. Next, we introduce three services that are suspicious to IP fragmentation. Finally, for each service, we evaluate the FragJam attack impact and feasibility against real-world providers and open-source implementations (local deployments over VPSes).

4.1 OSes & Router Models Evaluation

As a necessary ingredient of the FragJam attack, an attacker congests the reassembly buffer of a NAT gateway. We now study whether popular Linux distributions and router devices that function as NAT gateways are affected by the attack. To extend our evaluation beyond devices available to us, we also examine the vendor-provided GPL code for the routers.

Both routers used in our evaluation setup in Section 3.1 are using OpenWrt with an affected Linux kernel (v5.15). They use the default reassembly buffer size (4 MB) and timeout (30 s). Empirically, to congest the reassembly buffer (using 572-byte fragments) for the NAT1 router (a GL-MT6000 router flashed with OpenWrt), the attacker needs 922 fragments, and for the NAT2 router (a Turris Ominia Wi-Fi 6 router running TurrisOS, an OpenWrt-based OS), 1921 fragments. We also tested two routers available to us, ASUS RT-AX57 and Netgear RAX30, both of which appear vulnerable and use a 30s fragments timeout, requiring 2497 fragments to congest the buffer. In addition, we examined the GPL source code provided by popular home router vendors for their router products, including TP-Link, Netgear, Linksys, and Gl.iNet. It turns out that over 20 models (see Appendix C, many released since 2023) appear to use an affected Linux kernel.

In terms of popular Linux distributions, both Ubuntu 22.04 (kernel 5.15) and Debian 12 (kernel 6.1) use the default configuration for reassembly buffer and timeout. The required minimum number of fragments to congest the buffer slightly varies per driver. For example, 2081 and 1766 fragments are required using the *virtio_net* and *e1000e* drivers, respectively. Having said this, although the minimum number of fragments slightly varies, given the default maximum buffer size of 4 MB, an attacker can resort to sending 4096×1024-byte fragments to *guarantee* congestion.

4.2 Suspicious Applications & Services

As the FragJam attack only denies fragmented packets, and the Linux kernel by default may only fragment packets down to 552 bytes, we examine three services that are susceptible to the attack by sending packets larger than 552 bytes.

Video-Conferencing: Widely used services provided by video conferencing vendors, such as Zoom and Google Meet, enable real-time video and audio communication among multiple participants. In terms of media transmission, implementations typically use RTP, over UDP [78]. Based on both empirical measurements and official documents [87], real-world video conferencing service providers typically send UDP packets within 1300 bytes, which is larger than the default minimum path MTU in Linux.

RADIUS Authentication: RADIUS, which typically uses UDP as the transport layer protocol, is widely used for network access control and VPN authentication. In this work, we focus on the typical scenario of using RADIUS for network access control. When a client joins a network with WPA-Enterprise (WPA-EAP) enabled, the access point acts as the authenticator and sends RADIUS authentication requests encapsulating EAP messages to the RADIUS authentication server and receives corresponding responses. In WPA-EAP, the common authentication methods include PEAP, EAP-TLS, and EAP-TTLS. All three methods involve certificate exchanges and thus can require the server to send large UDP packets that are possible to be fragmented.

VPN/Tunneling: VPN and UDP-based tunneling services are widely used by individuals and enterprises to ensure privacy and security. In this work, we focus on popular VPN protocols used by real-world providers, OpenVPN (UDP) and WireGuard. When using these protocols, the original IP packets are encapsulated into UDP packets and transmitted. The size of the encapsulated UDP packets depends on the original IP packet size. Thus, given common MSS settings, e.g., 1380 bytes in WireGuard, the encapsulated UDP packets sent from the VPN server can be fragmented for smaller path MTUs.

In the next section, we provide the evaluation of the FRAGJAM attack against the three services. For evaluated services, we confirm that when *not* flooding the NAT gateway (or when flooding with repeat fragments using the same IPID, which would not congest the reassembly buffer), the service operates properly when the server-to-client traffic is fragmented due to forged ICMP PTBs. This ensures that the cause of the DoS is dropped server-to-client fragments due to the congested reassembly buffer. Nevertheless, due to the limited visibility, we can only inspect the traffic received on our NAT gateways, clients, and the servers when using local deployments on remote VPSes. Thus, we are not aware of possible link congestion or packet loss between our NAT gateway and the servers of service providers. During the FRAGJAM attack, dropped fragmented packets may incur different server sending behaviors, such as aggressive retransmission. It is thus possible that packet loss also occurs before arriving at our NAT gateway due to link congestion, which may further contribute to the attack effectiveness.

4.3 Video Conferencing Service

To evaluate the practicability of the FRAGJAM attack against the video conferencing service, we study one open-source implementation, Jitsi Meet [57], and four popular real-world service providers – Zoom, Microsoft Teams, Webex, and Google Meet. Figure 7 in the Appendix illustrates the evaluation setup for the video conferencing service. The setup includes, first, a video conferencing server from one of the service providers, and second, two clients joining one online meeting session. One client operates as the presenter of the meeting, who initiates the meeting and starts video and screen sharing. The second client is the victim who tries to join the same meeting. As our attack targets the UDP traffic from the server to the victim, we try to avoid direct P2P communication between the two clients, which is common when there are only two attendees of a meeting. For example, as reported by Michel et al. [59], Zoom would try to use a P2P connection when there are only two clients in a meeting. Similarly, the open-source implementation Jitsi meet also attempts P2P communication when there are only two clients and switches back to the server if a third client joins [58]. For the local deployment of Jitsi Meet, we disable the P2P configuration on the server. For the real-world service providers, we configure the presenter to join the meeting through a VPN that uses symmetric NAT to avoid P2P communication. For all service providers, we use their proprietary client software (Windows) on both clients. The attacker follows the general threat model in Section 3.1 and sends an ICMP PTB suggesting a 552-byte PMTU to the server. For now, we assume that the attacker knows the server IP and port and will elaborate on how an attacker can obtain this information in Section 4.3.1.

In Table 2, we summarize the evaluation results. Overall, Microsoft Teams and Google Meet are not vulnerable to the attack, as they do not accept the forged ICMP PTB. Not only when receiving a forged ICMP PTB using a dummy client port, even if the attacker can guess the exact client port used by the victim, but the forged ICMP PTB is still not accepted. Instead, Zoom, Webex, and Jitsi Meet all accept an ICMP PTB embedding a UDP packet with a dummy client port (a different port than the one used by the client-to-server flow) and perform fragmentation. The attacker can block or delay new content of the victim's downstream video, screen sharing, and whiteboard for Zoom, and downstream video and screen sharing for Webex and Jitsi Meet.

We continued the attack for each evaluated service to observe whether the application can recover from packet loss caused by the attack. Neither Jitsi Meet nor Webex can recover from the attack. In contrast, Zoom's screen share service recovers after approximately 1 minute by switching to TCP. However, the fallback behavior of Zoom only occurs when the attacker floods the NAT gateway with the restless flood strategy. When using the lossy flood strategy described in Section 3.2, the attacker can still degrade the screen share service without triggering TCP fallback.

4.3.1 Attack Scalability and Feasibility

To perform the attack in practice, the attacker needs to know the server IP address and port used by the victim to forge ICMP PTBs. We now discuss how an attacker could obtain this information for Zoom and Webex. Both Zoom [95] and Webex [88] provide documentation to assist with firewall configuration, which includes server IP ranges and port information. We observe that both Zoom and Webex servers also

Table 2: Video Conferencing Service Evaluation Summary. Legend: ✓= affected by FRAGJAM, ✗= not affected.

Provider	PTB Type❖	Audio	Chat	Video	Screen Share	Whiteboard	Recover
Zoom (Desktop)	UDP	✗	✗	Up ✗| Down ✓	Up ✗| Down ✓	Up ✗| Down ✓	Yes
Webex (Desktop/Browser)	UDP	✗	✗	Up ✗| Down ✓	Up ✗| Down ✓	✗	No
Jitsi Meet (Desktop)❁	UDP/Echo	✗	✗	Up ✗| Down ✓	Up ✗| Down ✓	-	No
Microsoft Teams	None	-	-	-	-	-	-
Google Meet	None	-	-	-	-	-	-

❖ All the real-world service providers require an attacker to use an ICMP PTB with a preceding UDP packet, as discussed in Section 3.3.1.

❁ For the screen share service of Jitsi Meet, possibly due to retransmission, when legitimate fragmented traffic is dropped by the attack, the traffic rate on the server also increases significantly, causing more packet loss between the VPS and our gateway, which may also contribute to the attack effectiveness.

listen on TCP port 443; thus, an attacker can perform a port 443 SYN scan over the provided IP ranges to obtain the list of server IP addresses. We summarize the server IP and port information in Table 3. Overall, the number of active server IPs is much smaller than the full CIDR ranges provided. Given Webex's and Zoom's MTU timeout of 10 minutes, an attacker has to send only 83–153 PTB messages per second overall to reduce the PMTU cache for the client at *every* active server. When considering the region of the victim, the attacker may further reduce the number of potential server IPs.

To validate the completeness of the obtained IP list, we repetitively initiate a new meeting 15 times. Across the experiments, both the victim and the presenter connect to a server IP within the obtained IP list. Furthermore, to confirm whether servers use a consistent configuration allowing forged ICMP PTBs, we send a forged ICMP PTB embedding a UDP packet with a dummy client port to the server in each run. We observe that all attempted Zoom servers accept the ICMP PTB. However, for Webex, 11 out of 15 servers accept the ICMP PTB, while the remaining 4 servers do not accept the ICMP PTB using a UDP packet with a dummy port. For these failed servers, we attempt to send 9 more forged ICMP PTBs with different dummy ports and finally send an ICMP PTB using the correct client port. However, these servers do not accept the ICMP message even if the correct client port is used. Given that the four failure cases are all caused by servers from two /24 prefixes, we consider it is possible that there is an inconsistent configuration or region-specific network restrictions.

4.4 RADIUS Authentication

To evaluate the real-world relevance of the FRAGJAM attack against the RADIUS authentication service, we assess the attack against one open-source implementation, FreeRADIUS, and three real-world service providers: VendorA, Jumpcloud, and Portnox. In this evaluation, we focus on the setup of using RADIUS for enterprise wireless network access. Figure 8 in the Appendix illustrates the evaluation setup. For the access point, we use a Turris Omnia (OpenWrt-based) router, running hostapd [92]. We configure the router's wireless interface to use WPA2-EAP and the remote RADIUS authentication server, as specified by each provider. Our evaluation covers three common EAP methods, EAP-PEAP, EAP-TLS, and EAP-TTLS whenever possible. To simulate a client accessing the WiFi network, we use an Android 16 phone (with *wpa_supplicant*) to repeat the WiFi access attempt 5 times for each EAP method. The attacker forges ICMP PTBs with a path MTU of 552 bytes, and uses the restless flood strategy. As of now, we assume the attacker knows the server IP and port to assess the effectiveness of the attack.

In Table 4, we summarize the evaluation results. Both Jumpcloud and VendorA accept forged ICMP PTBs and fragment the server response. For them, the attacker can effectively block the client's access attempt under all EAP methods tested. That said, as discussed in Section 3.2, even if the attacker uses the restless flood strategy, legitimate fragments may still be reassembled with a low success rate. Thus, the victim client may still access the network with a low success rate. PortNox does not accept the forged ICMP PTB and is not affected.

For Jumpcloud, we observe that the forged ICMP PTB failed to trigger server fragmentation or showed a MTU timeout of less than 10 minutes in a few attempts. We believe this is related to the deployment of Jumpcloud and will provide more discussion in the next section.

4.4.1 Attack Scalability and Feasibility

To perform the attack in practice, the attacker needs to know the server IP and port to forge the ICMP PTBs. Both Jumpcloud and VendorA provide globally accessible RADIUS authentication servers through anycast IPs from AWS, where the information can be retrieved from documentation [42]. All these servers use the standard RADIUS port 1812. Recall the failed ICMP PTB attempts we observed for Jumpcloud; we consider these issues could be caused by an unstable upstream route that forwards our request to different anycast sites. In practice, such an anycast deployment may require an off-path attacker to share the anycast site with the victim. Note that, given the globally distributed RADIUS servers hosted via anycast, we cannot confirm whether all the servers accept the forged ICMP PTB.

Except for the globally accessible authentication servers, VendorA also provides additional RADIUS servers with dedicated IPs or per-user registerable RADIUS servers. It appears

Table 3: Video Conferencing Service Provider IP Ranges.

Provider	CIDR IP Num	UDP Port	Active IP Num	Median/Max IP Num Per Country	Hit Rate	PTB Success
Zoom	546,288	8801–8810	50,006	1,609/29,499	15/15	15/15
Webex	405,504	5004	91,898	1,058/35,382	15/15	11/15

Table 4: RADIUS Authentication Service Evaluation Summary.

Provider	PTB type❖	MTU Timeout	Max Pkt Size	Access Attempt	Framed-MTU
Jumpcloud (PEAP/TTLS/EAP-TLS)	UDP	600s	1108 Bytes	Access Blocked	Support
VendorA (EAP-TLS)	UDP	600s	1104 Bytes	Access Blocked	Support
FreeRadius (PEAP/TTLS/EAP-TLS)	UDP/Echo	600s	1096 Bytes	Access Blocked	Support
PortNox	None	-	-	Access Succeeds	-

❖ All the real-world service providers require an attacker to use an ICMP PTB with a preceding UDP packet, as discussed in Section 3.3.1.

that these RADIUS servers are not affected, as they do not accept forged ICMP PTBs.

4.5 VPN Services

VPNs can be used to access various services securely over the Internet. In the evaluation, we target two popular VPN protocols, OpenVPN and WireGuard, both of which support UDP transport, and we deploy them locally. For real-world service providers, we evaluate the attack against Proton VPN and ExpressVPN. Both of them provide OpenVPN and WireGuard, where ExpressVPN also provides another UDP-based protocol, Lightway UDP [18].

Figure 9 illustrates the evaluation setup. Similar to the previous evaluations, we again assume that the attacker knows the server IP and port. The attacker first floods the NAT gateway with fragments and then sends a forged ICMP PTB to the remote VPN server. Next, we use a Windows client to initialize the tunnel with the remote VPN server, and then use a Selenium script to simulate web browsing. For proof-of-concept, we choose a set of five popular websites (including Google, YouTube, BBC, Wikipedia, and Amazon), which cover HTTP3 (QUIC), HTTP2, and HTTP1.1, and perform the access five times (with a 10s timeout per attempt).

We summarize the evaluation results in Table 5. Our own deployments of OpenVPN and WireGuard servers accept both forged ICMP PTB embedding either the UDP packet or the Echo response. During the flooding attack, the attacker can block users' access to remote web servers through the VPN tunnel effectively. For real-world providers, ExpressVPN also accepts both types of forged ICMP PTBs. Proton VPN, on the contrary, only accepts the ICMP message embedding a ping response. Note that Proton VPN does accept an ICMP message embedding a UDP packet with the correct server and client port, but forging such an ICMP message would require the attacker to guess the client port correctly. In the ExpressVPN case, all three of the protocols provided, OpenVPN, WireGuard, and Lightway UDP, are vulnerable, and an attacker can block the web access effectively. In contrast, for Proton VPN, both OpenVPN and WireGuard appear to be able to recover from the FRAGJAM attack. Concretely, the first access attempt to a website through Proton VPN would time out or use significantly more time, while subsequent access attempts would be successful with a normal load time (close to the baseline without the attack). We observe that the recovery is performed by adjusting the TCP packet size encapsulated in the UDP packet to fit the available MTU and avoid IP fragmentation. That is, Proton VPN implemented a behavior similar to the TCP PLPMTUD as discussed in Section 3.4. However, the initial access attempt would still be delayed by the FRAGJAM attack.

Finally, we also evaluate if our observations are specific to certain VPN regions only. For both real-world VPN providers, we successfully repeat our experiment with VPN servers from three regions (Germany, France, and the UK) and consistently confirm the universality of the attack impact.

4.5.1 Attack Feasibility and Scalability

In this section, we discuss the practical aspects of attacks against VPN services, including the discussion of the default behavior of ExpressVPN and Proton VPN, and how an attacker could obtain the server IP information.

Both ExpressVPN and Proton VPN provide TCP-based protocols in addition to UDP-based protocols. By default, both providers' client software use the "smart" or "automatic mode" to automatically select a protocol to initiate the tunnel, where ExpressVPN prefers Lightway UDP and Proton VPN prefers WireGuard UDP. As shown in Table 5, for the "auto" mode of ExpressVPN and Proton VPN, the attacker can use a path MTU of 1400 bytes and 1100 bytes, respectively, to allow the UDP-based tunnel initiation and block future legitimate fragmented traffic. In the auto mode, choosing an overly small path MTU would render the attack ineffective, as the client would fall back to TCP-based protocols if the UDP tunnel initiation fails. For example, the Lightway UDP protocol requires large UDP packets of approximately 1350 bytes for the initial tunnel setup. When a small path MTU is

Table 5: VPN Service Evaluation Summary.

Provider	PTB Type❖	PTB MTU	MTU Timeout	Attack Impact
WireGuard (Local Deployment)	UDP/Echo	552	600s	Access Blocked
OpenVPN UDP (Local Deployment)	UDP/Echo	552☆	600s	Access Blocked
ExpressVPN (Auto/Lightway UDP)	UDP/Echo	1400	600s	Access Blocked
ExpressVPN (OpenVPN UDP)	UDP/Echo	1100☆	600s	Access Blocked
ExpressVPN (WireGuard)	UDP/Echo	1400	600s	Access Blocked
Proton VPN (Smart/WireGuard)	Echo	1100	600s	Delay First Access
Proton VPN (OpenVPN UDP)	Echo	1100☆	600s	Delay First Access

❖ All the real-world service providers require an attacker to use an ICMP PTB with a preceding UDP/Echo Request packet, as discussed in Section 3.3.1.
☆ When using the corresponding PMTU values, we use the lossy flood strategy to allow tunnel initiation.
❁ For all the providers, TCP-based tunnels are not affected by the attack.

Table 6: VPN Service Provider IP Ranges.

Provider	IP Num	Median IP Num per Country
Proton VPN	1,337	2
ExpressVPN❖	3,060	11

❖ For ExpressVPN, the IP number does not include WireGuard servers.

suggested, the initialization of the tunnel would fail because of dropped fragments, and the TCP-based protocol is used. A similar problem exists for OpenVPN UDP. For example, ExpressVPN's OpenVPN UDP requires approximately 1300 bytes UDP packet for the initial tunnel setup, while a path MTU of 1300 is too large to fragment future encapsulated application traffic. In our evaluation, we observe that the client software does not have TCP fallback behavior when not using the automatic mode. Thus, an attacker may target clients who explicitly choose to use, e.g., OpenVPN over UDP, by blocking the tunnel setup. Alternatively, an attacker can use the lossy flood strategy as discussed in Section 3.2 to probabilistically allow tunnel setup while still blocking future legitimate fragmented traffic. Through our evaluation, we observe that a lossy flood can indeed allow the UDP tunnel to be initialized successfully, while allowing a few web access attempts.

In terms of obtaining the server IP information, both ExpressVPN and Proton VPN would cache the server IP address in a JSON file on the client. For Proton VPN, the server IP list is directly available from the JSON file. For ExpressVPN, the JSON file only provides a partial list of the server IPs.[5] To address this, an attacker can repeatedly restart the ExpressVPN daemon to gather the full server IP list. In Table 6, we list the number of server IPs for each provider. For ExpressVPN, we also repeatedly initialized the tunnel and confirmed the completeness of the retrieved server IPs. Overall, both Proton VPN and ExpressVPN use a moderate number of server IPs, which makes the attack scalable for an off-path attacker without knowledge of which server IP is used by the client.

[5]The ExpressVPN's Linux client does not support WireGuard, and thus the JSON file does not contain WireGuard server IPs. In the worst case, the attacker can repetitively connect to the server to enumerate server IPs.

Given an MTU cache timeout of 10 minutes, even if the attacker needs to send forged ICMP messages to each server, the attacker can still perform the attack at a low rate. In practice, an attacker may even reduce the number of possible IPs, as a victim may prefer to choose servers with low RTTs. In terms of the server port information, both Proton VPN and ExpressVPN do not use the standard OpenVPN port 1194 and WireGuard port 51820. However, learning the server port is not necessary, as the attacker can use ICMP PTB embedding an Echo response.

5 Discussion

In the previous sections, we showed that FRAGJAM affects real-world services. We now discuss mitigations, attack variants, and the status of vulnerability disclosure.

5.1 Mitigations

In this section, we discuss both general mitigations and service-specific mitigations against the FRAGJAM attack.

Strict PTB Validation: A fundamental necessity for FRAGJAM is that attackers can cause fragmentation using ICMP PTBs. A strict PTB validation thus presents a strong angle of defense. Though real-world servers are already protected by firewalls that drop irrelevant ICMP errors, most servers in our evaluation allow a forged ICMP message following an empty UDP request. Such loose ICMP validation eases the attack, as the attacker does not need to know the client port or forge a valid UDP request to trigger server responses. A counterexample is Proton VPN (see Section 4.5), which shows a stricter ICMP PTB validation that only accepts PTBs quoting a UDP packet matching the correct client port. From this, we can derive concrete countermeasures.

First, ICMP PTBs embedding an Echo response shall be ignored. Second, ICMP PTBs embedding a UDP packet corresponding to an unvalidated client port shall be ignored. In Linux, a server can use iptables to drop an ICMP PTB when it does not match a *SEEN_REPLY* flow. That is, an

ICMP PTB shall not be received from the client if the server never sends a response to the client. Alternatively, the UDP application could use a "wildcard" socket configured with *IP_PMTUDISC_OMIT* to accept the user request, and create a "connected" UDP socket with *IP_PMTUDISC_WANT* for a validated client. In this case, a forged ICMP PTB embedding an empty UDP packet with a dummy client port would match the wildcard UDP socket, and the PMTU is ignored. In contrast, a real user's ICMP PTB embedding a UDP packet matching the client port would match a connected UDP socket and update the PMTU cache.

Either way, an attacker would need to guess the correct client port to forge acceptable ICMP PTBs. Considering the number of potential client ports (up to 2^{16}), the cost of the attack would increase significantly. Having said this, an attacker may indeed spoof a UDP packet from a dummy client port with a correct request payload to trigger a server response or obtain a connected UDP socket and thus still produce an acceptable ICMP PTB using the same port. Only when this is not possible, hiding the client source port has value. For instance, the RADIUS server requires the access point to have the correct shared secret to generate an acceptable request.

Finally, it is also possible in some circumstances to ignore the received ICMP PTBs directly. We also observe such behavior from real-world vendors, e.g., Google Meet, Microsoft Teams, and Portnox. A server may still allow fragmentation by on-path routers by not setting the DF flag for sent packets.

TCP Fallback: Applications may implement more smart fallback behavior when a potential MTU or connectivity issue is detected. For example, ExpressVPN, Proton VPN, and Zoom all implemented certain TCP fallback behavior. However, as discussed in Section 4.3 and Section 4.5.1, an attacker may bypass such fallbacks by changing the path MTU used in the ICMP PTB or by using the lossy flood strategy. It is thus possible to implement a more aggressive fallback when a low quality-of-service is detected on the UDP channel.

Video Conferencing Mitigation: For the Video Conferencing Service, P2P direct communication is a possible mitigation. In this case, the attacker cannot easily obtain P2P peers' IP and port information to forge ICMP PTBs. However, P2P communication is not always possible, especially when there are more clients for a meeting.

RADIUS Authentication Mitigation: In fact, the RADIUS design is already aware of the security and reliability risks of IP fragmentation, and several methods are provided to avoid IP fragmentation on both the RADIUS server and the access point. As summarized in Table 4, both Jumpcloud and VendorA's servers never send a UDP packet that exceeds approximately 1100 bytes. This is performed by splitting large EAP messages into multiple RADIUS packets [82] to avoid large packets needing IP fragmentation, which is also implemented in FreeRADIUS as *fragment_size* (1024 by default) [23]. However, given that the default acceptable minimum path MTU is 552 bytes, an attacker can still fragment the response with forged ICMP PTB in common configurations. Thus, a possible mitigation is to further decrease the *fragment_size* configured on the RADIUS server to bring the UDP packet size to less than 552 bytes.

As an alternative defense, the Framed-MTU attribute [10, 75], when used by the access point in an access request, can suggest a preferred MTU to the server and thus reduce the server packet size. However, the server does not necessarily honor the hint [11, 75]. To evaluate if the servers respect the hint, we configure our access point to send requests with Framed-MTU=400 (1400 by default). Both the open-source implementation FreeRADIUS and the real-world provider Jumpcloud and VendorA honor the Framed-MTU and reduce the server response size to a maximum of approximately 500 bytes, which thus avoids fragmentation given the smallest possible path MTU suggested by an attacker. Thus, an access point may employ this configuration to avoid fragmentation.

One more radical mitigation would be to switch to TCP-based RADIUS authentication like RadSec [90]. As TCP rarely fragments, it is not vulnerable to the FRAGJAM attack. One example of RadSec adoption is *eduroam* [91]. The real-world providers also offer the option to use RadSec [43]. However, the traditional UDP-based RADIUS remains supported by the real-world vendors, and thus users relying on the UDP RADIUS service are still targets of the attack.

VPN Mitigation: The open-source implementations of both WireGuard and OpenVPN provide options to adjust the MTU and MSS. For OpenVPN, the server could configure a small *mss-fix* value to advertise a small MSS to the TCP server and thus reduce the encapsulated packet size to avoid fragmentation. Similarly, for WireGuard, a small interface MTU can be used together with MSS clamping to achieve the same effect. We confirmed through evaluation that this approach indeed allows the user's web access under the FRAGJAM attack. However, using a small MSS would also reduce the performance of TCP.

Recall that Proton VPN appears to implement a TCP PLPMTUD-like behavior to adjust the encapsulated TCP packet size to fit the available path MTU. In principle, when accessing a web server enabling TCP PLPMTUD through VPN, the service can recover from the packet loss caused by the FRAGJAM attack. However, the number of web servers that enable TCP PLPMTUD appears to be limited. According to a simulated experiment where we access the Alexa top-1k domains [1] with an `iptables` rule that drops packets larger than 1000 bytes (Generic Receive Offload is disabled to avoid dropping artificially large merged packets), only 21 websites can be loaded in 20 seconds. Note that *wikipedia.org* appears to use TCP PLPMTUD, while in our evaluation, which accesses the website under the FRAGJAM attack, the access is not successful, possibly because 10 seconds is too short for the page to be loaded under the attack.

Finally, as discussed in RFC 8900 [4] and RFC 7588 [5], IP-in-IP encapsulation and GRE encapsulation have imple-

mented a mitigation that allows to encapsualte fragmented packets instead of fragmenting the encapsulated packet to avoid IP fragmentation. A similar option is available in OpenVPN. Using the *–fragment max* configuration [63], internal datagram fragmentation can be performed so that no UDP datagrams larger than the *max* length are sent. However, this configuration is not encouraged due to its complexity [64].

5.2 Attack Variants

Server-Side Attack: In this paper, we focus on a setup where the attacker floods the NAT gateway to drop legitimate server-to-NAT-client fragments. In principle, any middlebox using an affected kernel that buffers fragments could be targeted by the attack. For example, servers protected by on-path stateful firewalls could also be flooded to drop fragments *of all their clients*. For ethical considerations, we cannot perform such real-world experiment to validate this, though.

IPv6 Attack: In this work, we focus on IPv4. However, fragmentation is also possible in IPv6. Linux uses a similar logic for IPv6 reassembly. Thus, an attack could use an ICMPv6 PTB message to trigger fragmentation and then flood the endpoint (an affected Linux host) with fragments. However, IPv6 is less vulnerable to the FRAGJAM attack due to a larger minimum MTU. As specified in RFC 8200 [15], the minimum MTU for IPv6 is 1280 bytes, which is also implemented in Linux (*net.ipv6.conf.all.mtu=1280*). Recall that service deployments usually limit the UDP payload size to avoid fragmentation in common network environments. For example, as discussed in Section 4.2 and Table 4, the video services typically send packets within 1300 bytes, whereas the RADIUS service typically sends packets within 1100 bytes. Thus, it is less likely to trigger fragmented responses in IPv6.

Our evaluation setup does not support IPv6 and thus cannot confirm IPv6-related behavior of services. Nevertheless, IPv6 is not possible for clients without IPv6 support or services that are only available through IPv4.

CGNATs: In today's network, home routers may not receive a public IP from the ISP, but instead use CGNAT, which includes an additional layer of NAT before the Customer Premises Equipment (CPE).[6] In this case, the attacker fragments would be buffered at the CGNAT deployed by the ISP. If the CGNAT uses a relevant Linux kernel, it could likewise be a target of the attack. Unfortunately, we cannot obtain the CGNAT devices used by ISPs, and it is not ethical to assess the fragility of real-world CGNAT deployments.

Insider Attacker: It is also possible to perform the attack with an insider attacker that shares the same NAT gateway as the victim. An insider attacker has several advantages: 1. The attacker can easily learn the public IP of the NAT; 2. The attacker does not require IP spoofing capability to send the UDP or Ping request before the forged ICMP PTBs; 3. The attacker may flood a Linux endpoint instead of the NAT gateway if the NAT gateway forwards fragmented packets instead of the reassembled packets [47]; 4. In the anycast scenario discussed in Section 4.4.1, an insider attacker has a higher chance to share the same anycast site as the victim.

5.3 IP Spoofing Capability

We observed that many real-world service providers only accept a spoofed ICMP PTB if they succeeded an IP-spoofed UDP/Echo Request, showing the need to leverage IP spoofing. In this section, we thus discuss the feasibility of IP spoofing.

We first discuss countermeasures against IP spoofing, most importantly Source Address Validation (SAV). SAV can be performed on both outbound and inbound. For outbound, as discussed in RFC 2827 and RFC 3704 [3, 79], an AS can drop outbound traffic with source IPs that are not owned by it. If all ASes deploy such filtering, IP spoofing can be largely mitigated. However, in practice, the filtering is not deployed by many ASes for several reasons, e.g., concern of false positives, equipment limitations. Recent CAIDA reports show that 20% of tested ASes perform insufficient outbound filtering [9]. Compared to outbound filtering, the identification of inbound spoofed traffic is, in general, harder. A recipient AS can deploy filtering that drops inbound traffic with source IP addresses that originate from itself or those with bogon addresses. However, as our off-path attacker spoofs the client IP against servers of a service provider, such filtering is ineffective. As noted by CAIDA, there is currently "no way to ensure that inbound traffic is legitimate as long as there exist entry points for spoofed traffic" [8].

To demonstrate that sending IP-spoofed ICMP PTBs and the preceding UDP or ICMP Echo request from an off-path attacker to trigger server-to-client traffic fragmentation is feasible, we obtained a spoofing-capable VPS from one AS listed by CAIDA. For services evaluated in this work, we first use the NAT client to communicate with the server and observe non-fragmented traffic. Next, we use the VPS to send the IP-spoofed preceding ICMP Echo Request or UDP packet and the ICMP PTB to the server, spoofing our own NAT's public IP. When spoofing UDP packets, we always use a different client port than the actual server-to-client flow to simulate an attacker without knowledge of the source port used by the client. After sending the spoofed ICMP PTB, we observe whether the server fragments its response to the victim client.[7]

For the RADIUS service, we tested against VendorA, and the off-path attacker successfully triggers fragmentation, despite the anycast setup. Jumpcloud shares a similar anycast setup, but is not retested as our trial expired.

For the Video conferencing service, we repeated the test for both Zoom and Webex with 15 servers. The spoofed PTBs

[6] The prevalence of CGNAT can be found in a recent cloudflare report [27] and several past research [53, 74], which is less common in several regions, e.g., US and Australia, or in non-cellular networks.

[7] This evaluation is performed after disclosure, so the ICMP PTB acceptance behavior of certain services may diverge from those in Section 4.

always trigger fragmentation for Zoom. For Webex, only 3/15 servers accept PTBs and thus perform fragmentation, countering the result before disclosure, where we succeeded for 11/15 attempts. We confirmed that the failure are not caused by IP-spoof filtering, as non-spoofed PTBs are also ignored.

For the VPN service, we experimented with services in 15 countries (one server per region) from both Proton VPN and ExpressVPN. Most servers accept the IP-spoofed ICMP PTBs and fragment the responses. We observe that one server for ExpressVPN and two servers for Proton VPN do not accept any ICMP PTBs (even non-spoofed), which could be caused by either inconsistent network setups that drop ICMP PTBs or changes incurred by our disclosure. We only observe one server of Proton VPN accepting the non-spoofed ICMP PTB while ignoring the spoofed ICMP PTB sent from the off-path attacker, which could be caused by a different route where the upstream of our VPS filters the spoofed packets.

Overall, spoofing ICMP PTBs and the preceding UDP or Echo requests is largely possible. However, we acknowledge that our evaluation is scoped in scale, and we cannot confirm whether the spoofed or non-spoofed ICMP PTB would be accepted by *all* servers of a provider to trigger fragmentation.

5.4 Disclosure Status (as of May 2026)

After confirming which services the FRAGJAM attack was effective against, we contacted the corresponding vendors, including Zoom, Webex, Jumpcloud, VendorA, ProtonVPN, and ExpressVPN, as well as the router vendors that may use an affected kernel. We disclosed the attack to them and explained potential countermeasures. For Video Conferencing service providers, we received feedback from Webex. The Webex team is investigating the cause of inconsistent ICMP PTB acceptance behavior. While the Webex team also acknowledges that whether a client can be affected depends on whether the path between Webex and the client can be flooded, and a server-side attack (see Section 5.2) is unlikely because of the DoS protection deployed. However, our disclosure efforts to Zoom were unsuccessful. We did not receive a response to our initial email report, and a subsequent disclosure through a third-party vulnerability disclosure platform was rejected during the platform's triage. For the VPN service provider, we received feedback from both Proton VPN and ExpressVPN. Proton VPN acknowledged the issue but did not provide further details on the patch plan. The ExpressVPN team has patched the software to use the *IP_PMTUDISC_OMIT* socket option to avoid PMTU spoofing [17]. For RADIUS service providers, we received feedback from VendorA, which decide to stick with the current server behavior of accepting ICMP PTBs and may apply changes in future infrastructure hardening. The Jumpcloud team is still investigating the issue. For router vendors, we received feedback from TP-Link, which acknowledges our findings and suggests that routers that use the same chipset vendor are likely affected as well. The TP-Link team also acknowledges that they use default sysctl configurations. However, as there is currently no broadly accepted fix or official guidance, they do not plan to take further action.

6 Related Work

For the related work, we focus on the security risks of IP fragmentation, abuse of ICMP PTBs, and the current status of IP fragmentation.

6.1 Security Risk of IP Fragmentation

IP fragmentation exposes multiple security risks, including DoS, traffic manipulation, and firewall evasion.

DoS: Due to the complexity of fragmentation reassembly and the requirement to buffer fragments, DoS is a natural risk. The general idea to send many fragments to exhaust the victim's resources is widely known [4, 31]. Possibly closest to our work, earlier works also showed that certain protocols with large application-layer payloads are susceptible to reassembly congestion. In 2003, Kaufman et al. showed that the reassembly buffer tail-drop scheme of NetBSD and Solaris OS affects the IKE handshake, which relies on large UDP packets [44, 83]. Similarly, rose fragmentation attack against Windows 2000, as proposed in 2004, sends small fragments to block legitimate fragmented packets [2]. What these earlier works did not explore, though, was that services can be tricked into sending fragmented packets *even if* the true PMTU is sufficiently large. Indeed, seeing that the studied services do not send payloads that exceed typical PMTUs, it seems intuitive to believe they are not affected. However, as show in the FRAGJAM attack, the threat of fragment losses can be generalized to application-layer protocols sending payloads that do not fit into the minimum PMTU (552 B). We showed that several popular services can be forced to fragment their packets, countering the assumption that reassembly congestion only affects "overly large" payloads.

Next to reassembly buffer congestion, other types of DoS vectors were found, many of which are fixed by now. Back in Linux kernel before v2.2.3, an attacker could use a series of 0-length IP fragments to exhaust the dst route cache quota and disable IP communication [68].

Also CPU exhaustion has been target of fragmentation-based DoS attacks. In early Linux kernel v2.4, an attacker could use crafted fragments to trigger a large number of hash table collisions and cause excessive CPU consumption upon reassembly [69]. More recently, the FragmentSmack [73] attack likewise triggered expensive reassembly operations.

Finally, prior work proposed DoS attacks that exploit IPID misassociations of benign fragments. That is, IPID, which is used to reassemble IP fragments, can wrap around or be reused, especially at high data rates [54]. When two fragments from different packets with the same source and destination IP and protocol collide in an IPID they are reassembled into

corrupted packets. Gilad et al. turn this observation into a DoS attack [26]. Targeting an IP-in-IP tunnel or NAT network setup, the attacker can cooperate with a puppet to expose the victim's IPID. With the knowledge of the IPID, the attacker can achieve the misassociation DoS attack more effectively by injecting malicious fragments accordingly. The work also discusses using ICMP PTBs to trigger fragmentation. In general, the attack requires more knowledge, in particular the IPID. The FRAGJAM attack is available for a pure off-path attacker. Furthermore, given that the attacker only needs to forge a single ICMP PTB using a dummy client source port for each possible server IP, FRAGJAM is more practical and scalable.

Memory Corruption Attacks Using Fragments: IP reassembly has also been repeated target of memory corruption attacks. In the Ping of Death attack [36], the attacker sends a series of legitimate fragments, but when reassembled larger than 65,535 bytes (largest possible IP packet) to crash the the operating systems. The Teardrop attack exploits overlapping IP fragments to trigger an overly large memory copy and can result in reboot or halt [37]. More recently, [72] exploits a race condition to cause use-after-free in Linux with fragmented ICMP Echo requests.

Traffic Manipulation: Recall the fragment misassociation attack discussed earlier: an attacker can use a similar strategy to reassemble a spoofed fragment with malicious content with a legitimate fragment with benign content. This attack allows the attacker to perform both data injection and traffic interception. In [50], Malhotra et al. reported a fragmentation-based NTP clock-altering attack. Similarly, a several works show that IP fragmentation can be exploited to conduct DNS cache poisoning [7,32–35,39,77,94]. Though TCP rarely fragments, previous works show that there are servers that fragment TCP packets and allow a TCP injection attack [13,20]. With a similar idea, in a NAT network, the attacker may exploit spoofed fragments to replace the UDP header of a legitimate fragmented packet, thereby altering the NAT decision and intercepting the packet with an attacker-controlled endpoint [26].

These works share a different motivation from DoS and typically require the knowledge of IPID. To allow fragment mis-association, ICMP PTBs are also discussed as one method to trigger fragmentation. Our work instead focuses on DoS caused by fragmented traffic.

Firewall Evasion: IP fragmentation can be used to evade traffic filtering [25]. For example, a stateless firewall that examines TCP or UDP ports cannot determine the legitimacy of the second fragment of a fragmented packet, as the transport header is only available in the first fragment [4]. RFC 1858 [67] and RFC 3128 [60] introduce firewall evasion techniques exploiting tiny fragments and overlapping fragments.

6.2 Other Forms of ICMP PTB Abuse

While our work leverages ICMP PTB to trigger fragmentation, the literature contains several other attack goals achievable by abusing PTBs. Jacquin et al. introduced a DoS attack that exploits the difference in the minimum acceptable PMTU on the end-user device and the IPsec gateway (the end-user device cannot reduce the packet to the small PMTU required by the gateway) to perform a DoS attack [38]. Forged ICMP PTBs can also be used to downgrade TCP performance [29, 30]. RFC 6269 introduced a potential attack where a malicious user of a shared IP (e.g., NAT or CGNAT scenario) may forge ICMP PTBs notifying a small MTU (e.g., 68 bytes) to incur more fragmented or small packets to increase the process load of both the server and the NAT device [6]. Finally, ICMP PTBs can be used in side channels to infer connection or network information. For example, Feng et al. introduced a technique where ICMP PTBs can be used to infer the sequence number of an ongoing TCP connection of a NAT user with the help of an in-network puppet [21]. Given a NAT client connecting to the attacker's vantage point, an attacker may also use ICMP PTB to induce different PMTU caches on the NAT gateway and NAT client to infer the existence of NAT [22]. In [52], Man et al. showed that ICMP PTBs can be abused to infer the ephemeral port used by DNS resolvers.

Our work presents an orthogonal malicious use case enabled by forged ICMP PTBs.

6.3 Mitigations for IP Fragmentation Issues

RFC 8900 [4] provides best current practices related to IP fragmentation. Except for the security risks exposed by IP fragmentation and path MTU spoofing, IP fragmentation can cause reliability and performance issues even without an attacker [4, 45, 86]. For instance, faulty implementation may drop fragments directly. Overall, IP fragmentation exposed various security and reliability issues and is not encouraged. RFC 8900 suggests several alternatives to IP fragmentation: 1. rely on transport protocols that can avoid fragmentation, e.g., TCP; 2. implement PMTUD or PLPMTUD to learn PMTU and avoid fragmentation; 3. use a conservative estimate of PMTU, e.g., 576 bytes for IPv4, to avoid fragmentation. Overall, it is suggested to avoid developing new protocols or applications that rely on IP fragmentation, and protocols that rely on IP fragmentation shall be updated to break the dependency.

Real-world services that were known to be prone to IP fragmentation-related issues, such as DNS [14, 24] and GRE [5], have already adopted countermeasures. For services evaluated in this work, the standards community [10,16,40,76,82, 84, 89] has discussed the drawbacks of fragmentation and corresponding mitigations, and implementations have provided measures to reduce the reliance on IP fragmentation (see Section 5.1). However, we are not aware of works that systematically evaluate the robustness of real-world services and

deployments against a fragmentation-based DoS attack in the present network. Our evaluation reveals that the FRAGJAM attack is a practical threat to several real-world services.

7 Conclusion

In this paper, we explore the practicality of the FRAGJAM attack against today's common UDP-based services. Our evaluation shows both the recent trend of router vendors to adopt a newer Linux kernel with a fixed fragmentation buffer space and the chance of triggering fragmentated responses from real-world service providers. The two components suggest that an attacker can use forged ICMP messages and floods of fragments to block real-world services, including video conferencing, RADIUS authentication, and VPNs. Though the risks of IP fragmentation seem to be well explored, reassembly congestion attacks such as FRAGJAM are easier to realize and more generically applicable than assumed before.

Acknowledgments

We thank Jonathan Flueren and Bastian Korte for bringing to our attention the Linux kernel's IP fragment reassembly implementation, which motivates this work. We thank Prof. Roland van Rijswijk-Deij for suggesting the RADIUS authentication service as a potential target.

Ethical Considerations

Stakeholders and benefits: Primary stakeholders of the work are: (1) the provider of the tested services; (2) end users of the tested services; (3) individual deployment or users who use a similar service or open-source software. The work demonstrates the DoS risk posed by the spoofed ICMP fragmentation needed message (PTB) and the reassembly implementation in Linux. Service providers and individuals can benefit from the per-service and general mitigations proposed in the work to alleviate the risk of a fragmentation-based DoS attack. Future developers may also benefit from the work by understanding the cause of the attack.

Impact on stakeholders: During the evaluation, we interact with the stakeholders' servers in the following manner: (1) We use the service provided by the stakeholders. During our simulation, the victim client (controlled by us) behaves as a client experiencing significant packet loss, which may induce corresponding server actions, e.g., retransmission. However, the same operations are performed for benign clients experiencing packet losses as well. (2) We sent the stakeholders' servers ICMP PTBs indicating a small path MTU. (3) For Zoom and Webex, we perform a port 443 scan to obtain the list of servers. After publication, users of the studied services and services sharing a similar behavior may be potential targets of the attack described in the work.

Mitigation taken for negative impacts: Our evaluation takes several operations to avoid negative impacts: (1) Our evaluation setup avoids spoofing any IPs that are not under our control. (2) The fragment flooding in our evaluation setup also only targets a router under our own control, and would not affect the service provider or other benign users. (3) The sent ICMP PTBs only update PMTU for our own IP, which is not shared with other users, and would thus not affect other benign users. (4) When performing the port scan for Webex and Zoom, we perform the scan in accordance with best practice and apply a strict rate limit of 2000 packets per second. Necessary information is provided for a stakeholder to opt out of the research as well. We try to tackle potential post-publication negative impacts: (1) We discuss service-specific and general mitigation options to defend against the attack, where service providers that are not involved in our evaluation and individual users of open-source software can also benefit. (2) As discussed in Section 5.4, we disclosed to the potentially vulnerable service providers before submission.

How the decision to both do and publish the research was reached, respectively: (1) We raise awareness of the fragmentation reassembly buffer congestion issue in the current Linux kernel implementation. Given that the kernel developer stated the problem cannot be patched and the prevalence of Linux-based routers, we consider this a potential risk for services that still rely on fragmentation or that fragment after receiving spoofed ICMP PTBs. We thus decided to evaluate the potential risk. (2) We decide to publish the research to warn of the security risk rising from the reliance on fragmentation and the loose validation for spoofed ICMP PTBs. By understanding the risks, the community can develop services that are robust against this type of attack.

Open Science

We provide one open and one restricted repository. The open repository (`https://doi.org/10.5281/zenodo.20407155`) provides (1) The script implementing the flooding strategies. (2) The script for sending spoofed ICMP PTBs. (3) The script for emulating web access when evluating the VPN service. The restricted repository (`https://doi.org/10.5281/zenodo.20408165`) provides sample pcaps we collected when evaluating the attack against different services.

References

[1] Alexa. Alexa Top 1 Million Domains Backup. `https://github.com/PeterDaveHello/top-1m-domains`, Accessed at Feburary 3, 2026.

[2] Internet Archive. Rose and New Dawn Fragmentation Attack Explained. `https://web.archive.org/web/20120224113108/http:/www.digital.net/~gan`

dalf/Rose_Frag_Attack_Explained.htm#item2c, Accessed at Feburary 3, 2026.

[3] Fred Baker and Pekka Savola. Ingress Filtering for Multihomed Networks. RFC 3704, March 2004.

[4] Ron Bonica, Fred Baker, Geoff Huston, Bob Hinden, Ole Trøan, and Fernando Gont. IP Fragmentation Considered Fragile. RFC 8900, September 2020.

[5] Ron Bonica, Carlos Pignataro, and Dr. Joseph D. Touch. A Widely Deployed Solution to the Generic Routing Encapsulation (GRE) Fragmentation Problem. RFC 7588, July 2015.

[6] Mohamed Boucadair, Mat Ford, Phil Roberts, Alain Durand, and Pierre Levis. Issues with IP Address Sharing. RFC 6269, June 2011.

[7] Markus Brandt, Tianxiang Dai, Amit Klein, Haya Schulmann, and Michael Waidner. Domain Validation++ For MitM-Resilient PKI. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 2060–2076, 2018.

[8] CAIDA. CAIDA FAQ: Is Spoofing Still Really a Significant Attack Vector? https://www.caida.org/projects/spoofer/faq/, Accessed at May 15, 2026.

[9] CAIDA. State of IP Spoofing. https://spoofer.caida.org/summary.php, Accessed at Feburary 3, 2026.

[10] Pat R. Calhoun and Dr. Bernard D. Aboba. RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP). RFC 3579, September 2003.

[11] Cisco. Troubleshoot EAP Fragmentation Implementations and Behavior. https://www.cisco.com/c/en/us/support/docs/lan-switching/8021x/220576-eap-fragmentation-implementations-and-be.html#anc15, Accessed at Feburary 3, 2026.

[12] Matt Corallo. Report for Fragmentation-Based DoS Issue in Linux Kernel. https://lore.kernel.org/netdev/CANn89i+L2DuD2+EMHzwZ=qYYKo1A9gw=nTTmh20GV_o9ADxe2Q@mail.gmail.com/T/#u, Accessed at Feburary 3, 2026.

[13] Tianxiang Dai, Haya Schulmann, and Michael Waidner. DNS-over-TCP considered vulnerable. In *ANRW '21: Applied Networking Research Workshop, Virtual Event, USA, July 24-30, 2021*, pages 76–81, 2021.

[14] DNS Flag Day. DNS Flag Day 2020. https://www.dnsflagday.net/2020/, Accessed at Feburary 3, 2026.

[15] Dr. Steve E. Deering and Bob Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, July 2017.

[16] Lars Eggert, Gorry Fairhurst, and Greg Shepherd. UDP Usage Guidelines. RFC 8085, March 2017.

[17] ExpressVPN. ExpressVPN Patch. https://github.com/expressvpn/lightway/commit/1f9c57c8e63eb540058d4426b818e105d42883d8, Accessed at May 15, 2026.

[18] ExpressVPN. The Lightway VPN Protocol. https://github.com/expressvpn/lightway-core, Accessed at Feburary 3, 2026.

[19] Xuewei Feng, Qi Li, Kun Sun, Zhiyun Qian, Gang Zhao, Xiaohui Kuang, Chuanpu Fu, and Ke Xu. Off-path network traffic manipulation via revitalized ICMP redirect attacks. In Kevin R. B. Butler and Kurt Thomas, editors, *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 2619–2636. USENIX Association, 2022.

[20] Xuewei Feng, Qi Li, Kun Sun, Ke Xu, Baojun Liu, Xiaofeng Zheng, Qiushi Yang, Haixin Duan, and Zhiyun Qian. PMTUD is not Panacea: Revisiting IP Fragmentation Attacks against TCP. In *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*, 2022.

[21] Xuewei Feng, Zhaoxi Li, Qi Li, Ziqiang Wang, Kun Sun, and Ke Xu. Off-Path TCP Exploits: PMTUD Breaks TCP Connection Isolation in IP Address Sharing Scenarios. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*, pages 4574–4587, 2025.

[22] Xuewei Feng, Yuxiang Yang, Qi Li, Xingxiang Zhan, Kun Sun, Ziqiang Wang, Ao Wang, Ganqiu Du, and Ke Xu. ReDAN: An Empirical Study on Remote DoS Attacks against NAT Networks. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*, 2025.

[23] FreeRADIUS. FreeRADIUS EAP Module Configuration. https://www.freeradius.org/documentation/freeradius-server/4.0.0/reference/raddb/mods-available/eap.html, Accessed at May 24, 2026.

[24] Kazunori Fujiwara and Paul A. Vixie. IP Fragmentation Avoidance in DNS over UDP. RFC 9715, January 2025.

[25] Pierce Gibbs. Intrusion Detection Evasion Techniques and Case Studies. https://www.sans.org/white-papers/37527, Accessed at Feburary 3, 2026.

[26] Yossi Gilad and Amir Herzberg. Fragmentation Considered Vulnerable. *ACM Trans. Inf. Syst. Secur.*, 15(4):16:1–16:31, 2013.

[27] Vasilis Giotsas and Marwan Fayed. One IP address, many users: detecting CGNAT to reduce collateral effects. https://blog.cloudflare.com/detecting-cgn-to-reduce-collateral-damage/, Accessed at Feburary 3, 2026.

[28] GL.iNet. GL.iNet Firmware & OpenWrt Version Status Updates. https://www.gl-inet.com/support/firmware-versions/, Accessed at Feburary 3, 2026.

[29] Matthias Gohring, Haya Schulmann, and Michael Waidner. Path MTU Discovery Considered Harmful. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, pages 866–874, 2018.

[30] Fernando Gont. ICMP Attacks against TCP. RFC 5927, July 2010.

[31] Fernando Gont. Security Assessment of the Internet Protocol Version 4. RFC 6274, July 2011.

[32] Elias Heftrig, Haya Schulmann, and Michael Waidner. Poster: Off-Path DNSSEC Downgrade Attacks. In *Proceedings of the ACM SIGCOMM 2023 Conference, ACM SIGCOMM 2023, New York, NY, USA, 10-14 September 2023*, pages 1120–1122, 2023.

[33] Amir Herzberg and Haya Schulmann. Security of Patched DNS. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, volume 7459 of *Lecture Notes in Computer Science*, pages 271–288, 2012.

[34] Amir Herzberg and Haya Schulmann. Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org. In *IEEE Conference on Communications and Network Security, CNS 2013, National Harbor, MD, USA, October 14-16, 2013*, pages 224–232, 2013.

[35] Amir Herzberg and Haya Schulmann. Vulnerable Delegation of DNS Resolution. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, volume 8134 of *Lecture Notes in Computer Science*, pages 219–236, 2013.

[36] INSECURE.ORG. Ping of Death. https://insecure.org/sploits/ping-o-death.html, Accessed at Feburary 3, 2026.

[37] INSECURE.ORG. Teardrop Attack. https://insecure.org/sploits/linux.fragmentation.teardrop.html, Accessed at Feburary 3, 2026.

[38] Ludovic Jacquin, Vincent Roca, and Jean-Louis Roch. Too big or too small? The PTB-PTS ICMP-based attack against IPsec gateways. In *IEEE Global Communications Conference, GLOBECOM 2014, Austin, TX, USA, December 8-12, 2014*, pages 530–536, 2014.

[39] Philipp Jeitner, Haya Schulmann, and Michael Waidner. The Impact of DNS Insecurity on Time. In *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020*, pages 266–277, 2020.

[40] Randell Jesup, Tom Kristensen, Yekui Wang, and Roni Even. RTP Payload Format for H.264 Video. RFC 6184, May 2011.

[41] Wolfgang John and Sven Tafvelin. Analysis of internet backbone traffic and header anomalies observed. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC 2007, San Diego, California, USA, October 24-26, 2007*, pages 111–116, 2007.

[42] Jumpcloud. Jumpcloud RADIUS Configuration Guide. https://jumpcloud.com/support/configure-a-wap-vpn-or-router-for-radius, Accessed at Feburary 3, 2026.

[43] Jumpcloud. Jumpcloud RadSec Configuration Guide. https://jumpcloud.com/support/configure-radsec-for-radius, Accessed at Feburary 3, 2026.

[44] Charlie Kaufman, Radia J. Perlman, and Bill Sommerfeld. DoS protection for UDP-based protocols. In *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, October 27-30, 2003*, pages 2–7. ACM, 2003.

[45] Christopher A. Kent and Jeffrey C. Mogul. Fragmentation considered harmful. *SIGCOMM Comput. Commun. Rev.*, 25(1):75–87, January 1995.

[46] Linksys. Linksys GPL Code Center. https://support.linksys.com/kb/article/316-en/, Accessed at Feburary 3, 2026.

[47] Linux. Linux Kernel Patch – Always Refragment IP Defragmented Packets. https://github.com/torvalds/linux/commit/bb4cc1a18856a73f0ff5137df0c2a31f4c50f6cf, Accessed at Feburary 3, 2026.

[48] Linux. Linux Kernel Patch – Enforce Memory Limits Earlier. https://github.com/torvalds/linux/commit/56e2c94f055d328f5f6b0a5c1721cca2f2d4e0a1, Accessed at Feburary 3, 2026.

[49] Linux. Linux Kernel Patch – Use rhashtables for Reassembly Units. https://github.com/torvalds/linux/commit/648700f76b03b7e8149d13cc2bdb3355035258a9, Accessed at Feburary 3, 2026.

[50] Aanchal Malhotra, Isaac E. Cohen, Erik Brakke, and Sharon Goldberg. Attacking the Network Time Protocol. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.

[51] Keyu Man. Report for Fragmentation-Based DoS Issue in Linux Kernel. https://www.mail-archive.com/netdev@vger.kernel.org/msg401638.html, Accessed at Feburary 3, 2026.

[52] Keyu Man, Xin'an Zhou, and Zhiyun Qian. DNS cache poisoning attack: Resurrections with side channels. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 3400–3414. ACM, 2021.

[53] Anna Maria Mandalari, Andra Lutu, Amogh Dhamdhere, Marcelo Bagnulo, and Kimberly C. Claffy. Tracking the Big NAT across Europe and the U.S. *CoRR*, abs/1704.01296, 2017.

[54] Matt Mathis, Ben Chandler, and John Heffner. IPv4 Reassembly Errors at High Data Rates. RFC 4963, July 2007.

[55] Matt Mathis and John Heffner. Packetization Layer Path MTU Discovery. RFC 4821, March 2007.

[56] S Mccreary and k claffy. Trends in wide area IP traffic patterns - A view from Ames Internet Exchange. In *ITC Specialist Seminar*, September 2000.

[57] Jitsi Meet. Jitsi Meet. https://jitsi.github.io/handbook/docs/intro/, Accessed at Feburary 3, 2026.

[58] Jitsi Meet. Jitsi Meet P2P Configuration. https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-configuration/#p2p, Accessed at Feburary 3, 2026.

[59] Oliver Michel, Satadal Sengupta, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. Enabling passive measurement of zoom performance in production networks. In *Proceedings of the 22nd ACM Internet Measurement Conference, IMC 2022, Nice, France, October 25-27, 2022*, pages 244–260. ACM, 2022.

[60] Ian Miller. Protection Against a Variant of the Tiny Fragment Attack (RFC 1858). RFC 3128, June 2001.

[61] David Murray and Terry Koziniec. The state of enterprise network traffic in 2012. In *18th Asia-Pacific Conference on Communications, APCC 2012, Jeju, Korea (South), October 15-17, 2012*, pages 179–184, 2012.

[62] Netgear. Netgear Open Source Code for Programmers (GPL). https://kb.netgear.com/2649/NETGEAR-Open-Source-Code-for-Programmers-GPL, Accessed at Feburary 3, 2026.

[63] OpenVPN. OpenVPN Configuration Manual. https://openvpn.net/community-docs/community-articles/openvpn-2-5-manual.html, Accessed at Feburary 3, 2026.

[64] OpenVPN. OpenVPN Directives not Supported by OpenVPN Connect. https://openvpn.net/connect-docs/directives-not-supported.html, Accessed at Feburary 3, 2026.

[65] Linux Manual Page. rtnetlink – Linux Manual Page. https://man7.org/linux/man-pages/man7/rtnetlink.7.html, Accessed at Feburary 3, 2026.

[66] Jon Postel. Internet Control Message Protocol. RFC 792, September 1981.

[67] Darren Reed, Paul S. Traina, and Paul Ziemba. Security Considerations for IP Fragment Filtering. RFC 1858, October 1995.

[68] CVE Report. CVE-1999-0431. https://nvd.nist.gov/vuln/detail/CVE-1999-0431, Accessed at Feburary 3, 2026.

[69] CVE Report. CVE-2003-0364. https://www.cve.org/CVERecord?id=CVE-2003-0364, Accessed at Feburary 3, 2026.

[70] CVE Report. CVE-2011-3188. https://www.cve.org/CVERecord?id=CVE-2011-3188, Accessed at Feburary 3, 2026.

[71] CVE Report. CVE-2012-2744. https://www.cve.org/CVERecord?id=CVE-2012-2744, Accessed at Feburary 3, 2026.

[72] CVE Report. CVE-2014-0100. https://www.cve.org/CVERecord?id=CVE-2014-0100, Accessed at Feburary 3, 2026.

[73] CVE Report. CVE-2018-5391. https://www.cve.org/CVERecord?id=CVE-2018-5391, Accessed at Feburary 3, 2026.

[74] Philipp Richter, Florian Wohlfart, Narseo Vallina-Rodriguez, Mark Allman, Randy Bush, Anja Feldmann, Christian Kreibich, Nicholas Weaver, and Vern Paxson. A Multi-perspective Analysis of Carrier-Grade NAT Deployment. In *Proceedings of the 2016 ACM on Internet Measurement Conference, IMC 2016, Santa Monica, CA, USA, November 14-16, 2016*, pages 215–229, 2016.

[75] Allan Rubens, Carl Rigney, Steve Willens, and William A. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000.

[76] Pekka Savola. MTU and Fragmentation Issues with In-the-Network Tunneling. RFC 4459, April 2006.

[77] Haya Schulmann and Michael Waidner. Fragmentation Considered Leaking: Port Inference for DNS Poisoning. In *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, volume 8479 of *Lecture Notes in Computer Science*, pages 531–548, 2014.

[78] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, July 2003.

[79] Daniel Senie and Paul Ferguson. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, May 2000.

[80] Colleen Shannon, David Moore, and Kimberly C. Claffy. Characteristics of fragmented IP traffic on internet links. In *Proceedings of the 1st ACM SIGCOMM Internet Measurement Workshop, IMW 2001, San Francisco, California, USA, November 1-2, 2001*, pages 83–97, 2001.

[81] Colleen Shannon, David Moore, and Kimberly C. Claffy. Beyond folklore: observations on fragmented traffic. *IEEE/ACM Trans. Netw.*, 10(6):709–720, 2002.

[82] Daniel Simon, Ryan Hurst, and Dr. Bernard D. Aboba. The EAP-TLS Authentication Protocol. RFC 5216, March 2008.

[83] Valery Smyslov. Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation. RFC 7383, November 2014.

[84] Dr. Joseph D. Touch and Mark Townsley. IP Tunnels in the Internet Architecture. Internet-Draft draft-ietf-intarea-tunnels-15, Internet Engineering Task Force, May 2025. Work in Progress.

[85] TP-Link. TP-Link GPL Code Center. `https://www.tp-link.com/us/support/gpl-code/`, Accessed at Feburary 3, 2026.

[86] Gijs Van Den Broek, Roland Van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. DNSSEC meets real world: dealing with unreachability caused by fragmentation. *IEEE Communications Magazine*, 52(4):154–160, 2014.

[87] Webex. Network Requirements for Webex Services. `https://help.webex.com/en-us/article/WBX000028782/Network-Requirements-for-Webex-Services#id_134142`, Accessed at Feburary 3, 2026.

[88] Webex. Webex Firewall Configuration Guide. `https://help.webex.com/en-us/article/WBX264/How-Do-I-Allow-Webex-Meetings-Traffic-on-My-Network?#id_135011`, Accessed at Feburary 3, 2026.

[89] Magnus Westerlund. How to Write an RTP Payload Format. RFC 8088, May 2017.

[90] Klaas Wierenga, Mike McCauley, Stefan Winter, and Stig Venaas. Transport Layer Security (TLS) Encryption for RADIUS. RFC 6614, May 2012.

[91] Klaas Wierenga, Stefan Winter, and Tomasz Wolniewicz. The eduroam Architecture for Network Roaming. RFC 7593, September 2015.

[92] Wikipedia. Hostapd. `https://en.wikipedia.org/wiki/Hostapd`, Accessed at Feburary 3, 2026.

[93] Wikipedia. OpenWrt. `https://en.wikipedia.org/wiki/OpenWrt`, Accessed at Feburary 3, 2026.

[94] Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan, and Zhiyun Qian. Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 577–593, 2020.

[95] Zoom. Zoom Firewall Configuration Guide. `https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0060548`, Accessed at Feburary 3, 2026.

A Linux IP Fragment Reassembly

```
int ip_defrag(struct net *net, struct sk_buff *skb, u32 user){
    ......
    qp = ip_find(net, ip_hdr(skb), user, vif);
    if (qp) { ret = ip_frag_queue(qp, skb); }
    ......
    return -ENOMEM;
}

static struct ipq *ip_find(struct net *net, struct iphdr *iph,
              u32 user, int vif){
    ......
    q = inet_frag_find(net->ipv4.fqdir, &key);
    if (!q) return NULL;
    return container_of(q, struct ipq, q);
}

struct inet_frag_queue *inet_frag_find(struct fqdir *fqdir, void *key){
    struct inet_frag_queue *fq = NULL, *prev;
    if (!fqdir->high_thresh || frag_mem_limit(fqdir) > fqdir->high_thresh)
        return NULL;
    prev = rhashtable_lookup(&fqdir->rhashtable, key, fqdir->f->rhash_params);
    if (!prev) fq = inet_frag_create(fqdir, key, &prev);
    return fq;
}

static int ip_frag_queue(struct ipq *qp, struct sk_buff *skb){
    ......
    if (!(IPCB(skb)->flags & IPSKB_FRAG_COMPLETE) &&
        unlikely(ip_frag_too_far(qp)) &&
        unlikely(err = ip_frag_reinit(qp))) {
            ipq_kill(qp);
            goto err;
```

```
    }
    err = inet_frag_queue_insert(&qp->q, skb, offset, end);
    add_frag_mem_limit(qp->q.fqdir, skb->truesize);
    ......
}
```

Listing 1: Linux IP Fragment Reassembly

```
static const struct nf_hook_ops ipv4_defrag_ops[] = {
    {
        .hook           = ipv4_conntrack_defrag,
        .pf             = NFPROTO_IPV4,
        .hooknum        = NF_INET_PRE_ROUTING,
        .priority       = NF_IP_PRI_CONNTRACK_DEFRAG,
    },
    ......
};


static unsigned int ipv4_conntrack_defrag(void *priv, struct sk_buff *skb,
    const struct nf_hook_state *state){
    ......
    if (ip_is_fragment(ip_hdr(skb))) {
        enum ip_defrag_users user =
            nf_ct_defrag_user(state->hook, skb);

        if (nf_ct_ipv4_gather_frags(state->net, skb, user))
            return NF_STOLEN;
    }
    return NF_ACCEPT;
}

static int nf_ct_ipv4_gather_frags(struct net *net, struct sk_buff *skb,
                   u_int32_t user){
    err = ip_defrag(net, skb, user);
}
```

Listing 2: Linux IP Fragment Reassembly (conntrack)

B ICMP Process Flow in Linux Kernel

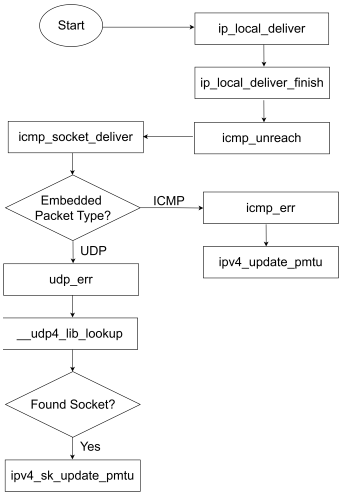

Figure 6: Simplified ICMP PTB Processing Flow.

C Potentially Affected Router Models:

TP-Link [85]:
Deco BE25 / BE65 / BE85; TL-WR3602BE, TL-WR3002X; Archer BE230 / BE400 / BE600 / BE700 / BE800 / BE900 / GE650 / GE800 / GXE75;
Netgear [62]:
RS70 / RS90 / RS100;
RAXE300 / RAXE290 / RBE370 / RBE770;
Linksys [46]:
LN1100 / LN1200 / LN1400 / LN1600 / MBE70 / MBE7000 / MX6200;
GL.iNet [28, 93]:
GL-BE6500 / GL-BE9300 / GL-MT6000 / GL-AX1800 / GL-B1300;

D Service-Specific Evaluation Setup

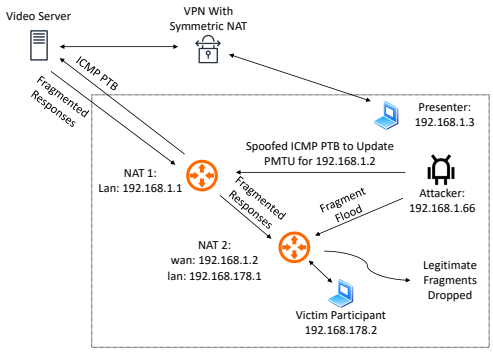

Figure 7: Video Conferencing Application Evaluation Setup

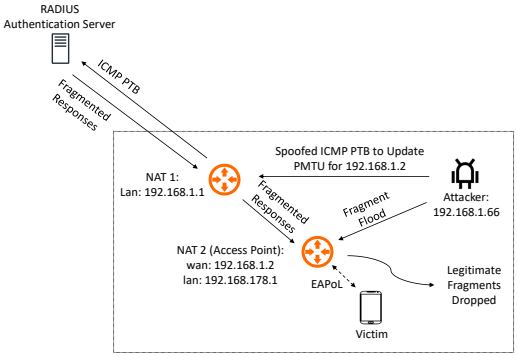

Figure 8: Radius Authentication Evaluation Setup

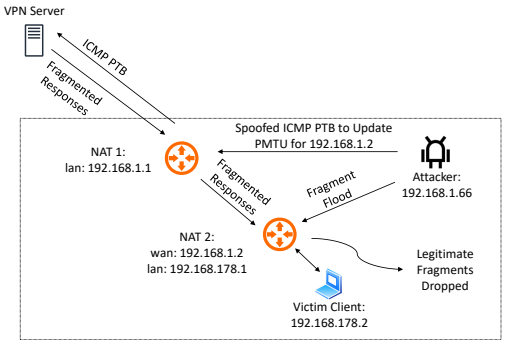

Figure 9: VPN Evaluation Setup