# Lost in Encapsulation: Exploiting Open Tunnelling Hosts and Attacking Private Networks

Angelos Beitis
*DistriNet, KU Leuven*
*angelos.beitis@kuleuven.be*

Mathy Vanhoef
*DistriNet, KU Leuven*
*Mathy.Vanhoef@kuleuven.be*

## Abstract

Tunnelling protocols form an essential backbone of today's Internet, where they primarily interconnect remote networks. However, many tunnelling hosts do not implement authentication, making them behave as open one-way proxies and, in turn, introducing denial-of-service (DoS) risks.

In this paper, we show that the security risk of open tunnelling hosts extends beyond DoS attacks. First, we study NAT-enabled tunnelling hosts and demonstrate a novel attack that manipulates the NAT's TCP state, enabling an attacker to fingerprint and inject traffic into the co-located private network. We also study tunnelling hosts that translate global IP addresses, and turn them into *two-way proxies*. We demonstrate how these two-way proxies enable an attacker to: (1) spoof TCP connections; and (2) launch a novel loop DoS attack. Additionally, we show that Layer 2 (L2) tunnelling hosts, specifically GRETAP, can be abused to attack the LAN, and we demonstrate the impact with 5 attacks: leaking the host's global and private MAC addresses, DHCP starvation, DNS poisoning, ARP spoofing, and an infinite loop DoS attack.

To evaluate the prevalence of vulnerable hosts, we conduct 9 Internet-wide scans and identify that 438,280 hosts are potentially vulnerable. Overall, we show that open tunnelling protocols enable new attacks on co-located private networks, and we hope our work encourages the adoption of new defences.

## 1 Introduction

Packet transport through intermediary hosts is a crucial feature of modern Internet architecture. Both users and networks rely on tunnels to connect otherwise disconnected networks, preserve anonymity, create Virtual Private Networks (VPNs), or access resources that would otherwise be inaccessible. Furthermore, tunnelling protocols such as IPsec can be used to secure these mechanisms, ensuring traffic authentication and encryption [21]. However, many hosts implement tunnelling protocols without incorporating any form of authentication. By decapsulating and forwarding traffic from arbitrary sources, these open tunnelling hosts can be abused to launch denial-of-service (DoS) attacks while concealing the attacker's identity, ultimately becoming a *one-way proxy* for the attacker [6]. Despite these risks, scans by the Shadowserver Foundation show that, at the time of writing, only around 15% of open tunnelling hosts have been patched [45]. This motivates a more detailed analysis of the security risks of open tunnelling hosts, especially with a focus on abusing them to access co-located private networks behind Network Address Translation (NAT).

In the first part of our paper, we examine the impact of IP-in-IP (IPIP) [41] and Generic Routing Encapsulation (GRE) [18] open tunnelling hosts that implement NAT. We illustrate the issue with a concrete attack that allows a remote attacker to fingerprint hosts, establish TCP connections, and receive replies from hosts behind a private network. To identify the prevalence of susceptible hosts, we scan the entire IPv4 address space and identify that 283,378 hosts translate private IP addresses. We also demonstrate that many open tunnelling hosts perform *public* address translation, enabling attackers to abuse them as *two-way proxies*. Unlike one-way proxies, which only forward the attacker's packets [6], *two-way* proxies enable an attacker to establish connections and receive replies while appearing to the target as the proxy's IP address. We demonstrate two novel attack vectors: 1) An attacker can abuse such hosts to establish TCP connections behind a spoofed address, and 2) an attacker can create a bidirectional mapping between two-way proxies, forcing packets to loop until they expire. To identify two-way proxies in the wild, we developed a new scanning methodology that discovered 27,286 hosts are vulnerable.

In the second part of our paper, we explore how Layer 2 (L2) tunnelling protocols, such as Generic Routing Encapsulation Terminal Access Point (GRETAP), can enable attackers to access the LAN. This enables a remote attacker to leverage an almost *adjacent* position and directly attack a private network. We demonstrate this issue through 5 attacks, showcasing how an attacker can tamper with a LAN's configuration

or launch DoS attacks. Furthermore, we scanned the Internet and identified 147,167 hosts that may be susceptible.

Overall, our results indicate that the security risk posed by open tunnelling hosts has been underestimated. For instance, in addition to posing a DoS risk, many of them can act as two-way proxies or be exploited to access internal private networks. We hope our results will accelerate the patching of open tunnelling hosts, and we outline several mitigations for our new attacks.

To summarise, our main contributions are:

- We explore a new attack on NAT-enabled open tunnelling hosts and 2 novel two-way proxy attacks in Section 4.
- We analyse the impact of open GRETAP hosts through 5 attacks in Section 5.
- We conduct 9 Internet-wide scans to identify open tunnelling hosts susceptible to our attacks in Section 6 and present our findings in Section 7.

We finally discuss defences in Section 8, compare with related work in Section 9, and conclude in Section 10.

# 2 Background

## 2.1 Tunnelling & Encapsulation

Tunnelling is a technique that relies on encapsulation to connect or traverse networks. Tunnelling can be utilised for various use cases and topologies, such as connecting two or more remote networks, accessing private resources remotely, or browsing the web anonymously via VPNs. A standard use case of encapsulation and decapsulation in an established tunnel between two hosts (encapsulator and decapsulator) is as follows: 1) The encapsulator embeds a packet within an outer packet; the specifics of encapsulation depend on the tunnelling protocol in use. This outer IP header then has the decapsulator as the destination and is sent over the Internet. 2) Upon receipt, the decapsulator strips the added header(s) attached to reveal the original packet, and finally, 3) the packet is served or forwarded according to the topology and use case.

Furthermore, when secure tunnelling protocols are used, tunnelling can provide properties such as authentication and encryption [21]. In this paper, we focus on two tunnelling protocols which do not incorporate authentication or encryption by default [18,41]: IPIP and GRE.

**IPIP.** The IPIP protocol was standardised in RFC 2003 [41] in 1996. The IPIP protocol encapsulates an IP header inside an IP packet and does not provide authentication or encryption by default. One of the significant benefits of IPIP is its simple setup and minimal overhead (20 + 20 bytes).

**GRE & GRETAP.** GRE is a more general implementation of IPIP, introduced in RFC 2784, which defines its own header, allowing for the encapsulation of any EtherType protocol, such as IPv4, IPv6 and Ethernet frames [18]. Similar to IPIP, GRE does not provide encryption or authentication by default. However, the GRE header includes a 4-byte plain-text key field, which can be *optionally* used for authentication. In the case of L2 frame encapsulation, the protocol is also known as GRETAP. In this paper, we use the two terms to distinguish between Layer 3 (L3) and L2 encapsulation in GRE.

## 2.2 Open Tunnelling hosts

We define an *open tunnelling host* as a host on the Internet which implements a tunnelling interface without: 1) implementing authentication or encryption, and 2) specifying a remote tunnelling endpoint [6]. When creating a tunnelling interface, the "remote address" field denotes the remote tunnelling endpoint, and leaving this field empty enables the host to decapsulate packets from any source [6, 31]. More concretely, when an open tunnelling host receives an encapsulated packet, it decapsulates it without verifying the sender and serves it or forwards it accordingly. Assume an attacker with address $a$, an open tunnelling host with address $t$ and a victim with address $v$. An attacker can exploit open tunnelling hosts by sending an encapsulated packet with the inner IP source address set to $t$, and the inner destination set to $v$. The packet in the case of IPIP will have the following structure: IP($a$,$t$)/IP($t$,$v$)/Payload. The open tunnelling host, upon receiving this packet, will decapsulate it and forward it to $v$, thereby serving as a *one-way* proxy for the attacker. This one-way proxy can now serve as an attacker's DoS vessel, enabling them to remain anonymous from the victim's perspective. When an open tunnelling host resides in an unfiltered network, the inner source IP address of the packet may be spoofed, enabling the attacker to spoof packets remotely [6, 33].

To identify such hosts, an attacker can scan the Internet using various methodologies [6,33] and, based on the responses, classify the host as an open tunnelling host. In general, there are two different methodologies relevant for this paper: 1) Encapsulating a packet with the inner destination set to the scanner; 2) Encapsulating a packet with the inner destination set to the host being scanned and followed by an ICMP Echo request. With the former methodology, the scanner anticipates a packet that exactly matches the encapsulated packet in the probe, since the host will receive the probe, decapsulate it, and forward the inner packet back. In the latter methodology, the scanner listens for an ICMP Echo reply; ultimately, entailing that the host decapsulated the packet, served the ICMP Echo request, and produced an ICMP Echo reply.

## 2.3 Network Address Translation (NAT)

NAT is a mechanism introduced to address the limited number of available IPv4 addresses and was standardised by RFC 1631 in 1994 [16]. The idea behind NAT is to map private IPv4 addresses to a single global IPv4 address. For example, in a private network behind a NAT, when a host attempts to establish a connection to a remote server, the NAT translates the source IP address into the globally routable network address and forwards the packet to the Internet. The NAT also maintains the connection state in a NAT table, enabling it to translate and forward incoming packets destined for the private address of the corresponding host.

We use the term *connection tracking* to describe the ability of a NAT to keep track of connection or session flows [26]. A NAT must maintain the state of connections or flows to forward and translate traffic to the correct destination. This typically involves tracking addresses, ports, protocols, and the current state (in the case of stateful protocols such as TCP). For example, assume a private network with two hosts, 192.168.0.2 and 192.168.0.3, and a NAT with private address 192.168.0.1 and public address 1.1.1.1. Suppose host 192.168.0.2 wants to initiate a TCP connection with host 2.2.2.2. In that case, the NAT will translate the IP address to 1.1.1.1 and send the packet to the Internet, while keeping track of the TCP connection state in its table in the following way: `<192.168.0.2:sport, 2.2.2.2:80, TCP, SYN_SENT>`. If *port randomisation* is enabled, i.e., a mechanism that translates the source port to another available random port [49], the translation between the source port chosen by the client and the random source port chosen by the NAT is also recorded. When host 2.2.2.2 replies with a TCP `SYN/ACK`, the NAT now knows that it corresponds to host 192.168.0.2 and can thus translate the destination address and forward the packet.

At first glance, NAT may also be regarded as a security feature, as it prevents a host behind it from being accessible to the Internet unless it initiates a connection with a remote host first. Unfortunately, it is non-trivial for an intermediary NAT machine to completely model all possible states reliably, as well as to completely reflect the actual state of a connection, leading to several bypasses and exploitations being possible [19, 37].

# 3 Methodology

**Motivation.** In 2025, Beitis & Vanhoef conducted Internet-wide scans to identify open tunnelling hosts. During disclosure, they note that some affected hosts also served as home routers [6, §3.3]. This motivated us to analyse the possible impact of open tunnelling hosts that act as gateways to a private network, and by extension, hosts that implement either NAT or L2-encapsulating tunnels. In this threat model, an attacker could theoretically abuse open tunnelling hosts to access or tamper with resources behind a LAN. More specifically, our research questions are as follows: **RQ1:** How can a remote attacker target the LAN by manipulating connection-tracking states on open tunnelling hosts? **RQ2:** How can an attacker exploit the L2 encapsulation capabilities of a host to attack the LAN? **RQ3:** How can we identify NAT-enabled tunnelling hosts and tunnelling hosts implementing L2 encapsulation protocols?

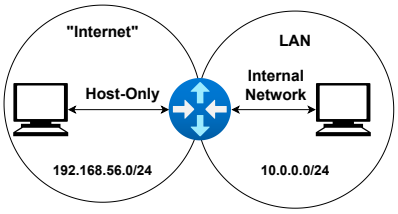


Figure 1: Topology used for preliminary experiments. The host has two adapters, one connects to the "Internet" (the attacker) while the other is connected to a private network.

## 3.1 Experiments

For our preliminary experiments, we set up Linux VMs in the topology shown in Figure 1. The open tunnelling host has two adapters: a host-only adapter, which serves as the interface to the "Internet", and an internal network adapter, which connects to the private network. The client has only an internal network adapter, which allows it to communicate with the tunnelling host. The objective is to simulate a private network in which clients lack direct Internet connectivity and rely on the gateway's NAT to access the Internet. Furthermore, to create an *open* tunnelling host, we configured the tunnelling interface using the `ip link` [28] and `ip tunnel` [31] commands without specifying a remote endpoint and enabled the Linux flags `accept_local` and `forwarding` [6].

**RQ1.** To address our first research question, we set up a Linux gateway that implements a tunnelling protocol and configured NAT on it using `iptables`. We specifically tested IPv4 L3 tunnelling protocols that we know are still widely vulnerable [6]. More specifically, we tested our attacks against IPIP and GRE open tunnelling hosts. Furthermore, the client behind this gateway was running an HTTP service on TCP port 80, to which the attacker would attempt to send a GET request and receive a response. We send carefully crafted packets and demonstrate that an attacker can exploit open tunnelling hosts to fingerprint private networks and establish connections with clients behind a NAT, as explained in detail in Section 4. The main contributor to the issue is that an attacker can encapsulate packets that appear to originate from within the network. Since the connection-tracking framework updates its state based on these packets, an attacker can abuse

them to manipulate a TCP state and access hosts on the LAN.

**RQ2.** For our second research question, we set up an open GRETAP Linux host[1] in a virtual environment. Further details on the network topology are provided in Section 5. Alternative L2 encapsulation protocols include GENEVE [24] and VXLAN [35], but, since these protocols rely on UDP encapsulation, scanning the Internet for them is 1) more intrusive as it will most likely trigger Intrusion Detection Systems (IDSs) to flag it as potentially malicious and 2) less reliable (increased false negatives) since we also need to predict the UDP port these protocols are served on. As a result, we only tested our attacks using GRETAP. We tested several LAN attacks, such as ARP spoofing and DHCP starvation, and found that a GRETAP open tunnelling host can enable a remote attacker to execute them. It is evident from our experiments that traditional LAN attacks are possible in this threat model if the attacker 1) does not need to predict MAC addresses of other hosts on the LAN, and 2) if the attack does not require the attacker to receive a reply. The primary reason L2 tunnelling protocols, such as GRETAP, can be exploited for LAN attacks is that L2 hosts can forward Ethernet frames. This is not the case with L3 encapsulation since forwarding broadcast IP packets is not possible [38, §7].

**RQ3.** Once we determined that Linux hosts could be vulnerable to our experimental attacks, we conducted Internet-wide scans to identify and quantify the impact. We extended ZMap, a modular Internet-wide scanner, to facilitate our novel scanning methodologies [14]. Before conducting a full Internet-wide scan, we first scan 1% of the Internet. If this initial 1% scan yielded any results, we conducted a full Internet-wide scan. We also scanned for open GRETAPv6 hosts; i.e., we sent an encapsulated ICMPv6 Echo request to the "all-nodes" IPv6 multicast domain, but since we received no replies in our 1% scan, we did not conduct a full Internet-wide scan.

## 4 NAT-Tunnel Attacks

This section demonstrates two types of attacks against NAT-enabled open tunnelling hosts. The first is against open tunnelling hosts that translate *private* IP addresses, and the second is against open tunnelling hosts that translate *global* IP addresses.

### 4.1 Open Tunnel Fingerprinting

We demonstrate how an attacker can exploit a NAT-enabled open tunnelling host to inject traffic and fingerprint hosts behind private networks. The attacker can send carefully crafted TCP packets that deceive the NAT into allowing bidirectional traffic across the network. There are two steps to this attack: *identifying private hosts* and *traffic injection*.

[1] Ubuntu 24.04 LTS 6.8.0-88-generic

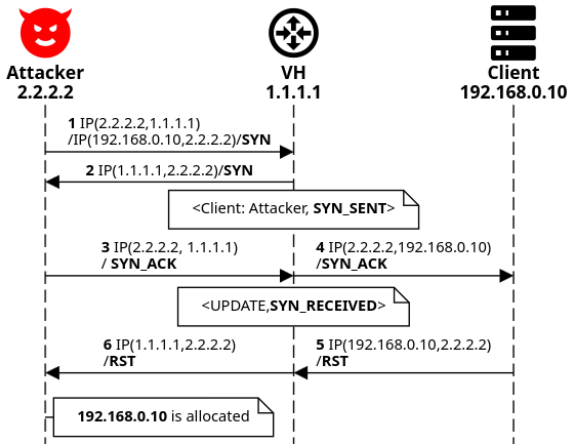


Figure 2: Detecting private hosts behind a NAT-enabled tunnelling host.

#### 4.1.1 Threat Model

We assume that the host implementing NAT can be a VPN server or a home router, ultimately masquerading IP addresses behind its (virtual) private network and blocking external connections to private hosts. Furthermore, the host decapsulates and forwards traffic from any source and can serve as an IPIP or GRE open tunnelling host. In this threat model, the attacker is entirely remote and off-path, having identified that the host implements an unauthenticated tunnelling protocol via an Internet-wide scan, as shown by [6]. Finally, the attacker's goal is to fingerprint the victim's network and access unauthenticated private resources.

#### 4.1.2 Identifying Private Hosts

In the first step of the attack, the attacker needs to identify which addresses are allocated/active behind a NAT. An attacker can identify all possible allocated addresses behind a private network and attempt to fingerprint them or inject traffic into them. An example of identifying an active private address behind a NAT is shown in Figure 2, where the attacker has the global IP address 2.2.2.2, the Vulnerable Host (VH) has the global IP address 1.1.1.1, and the client behind the VH has the private address 192.168.0.10. In step 1, the attacker sends an encapsulated TCP `SYN` packet to VH. The inner source IP address of the packet is set to the private IP address being tested (e.g., 192.168.0.10), and the inner destination is set to the attacker's global IP address. Once VH receives this packet, it will decapsulate it and forward the inner packet back to the attacker, as shown in step 2. The connection-tracking framework will record the egress packet and create a new `SYN_SENT` TCP state from the private IP to the attacker, while

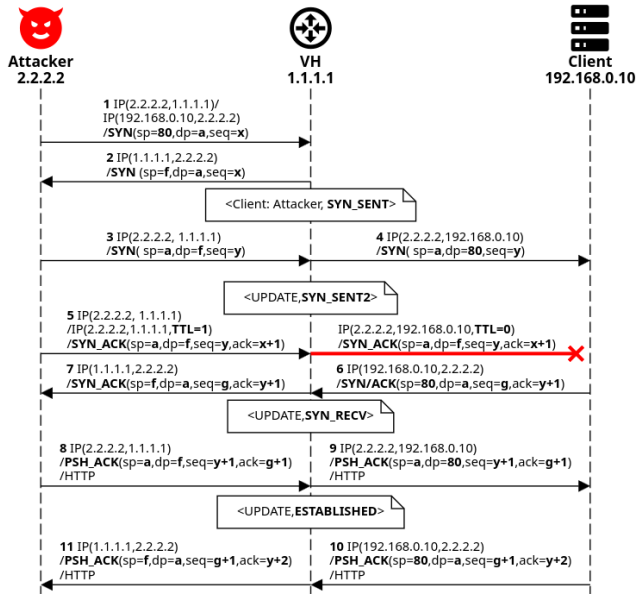


Figure 3: Traffic injection attack. A remote off-path attacker injects packets into a private network.

also translating the private address to the global. In step 3, the attacker sends a TCP `SYN/ACK` packet to VH. Since the connection-tracking framework assumes that the private host initiated the TCP connection via the previous `SYN` packet, the packet's destination address will be translated and forwarded to the private client (192.168.0.10). When *port randomisation* is enabled, the packet in step 2 will have a different source port than the packet in step 1. The attacker can use the source port from step 2 as the destination port in step 3, which will be translated back to the source port initially used in step 1. In steps 5 and 6, if the attacker receives a `RST` packet, it indicates that the private client has received the packet; however, because the private client has not initiated a TCP handshake, it will reply with a TCP `RST` packet. Alternatively, if the private IP address is not allocated, the packet will be dropped, and the attacker will not receive a reply.

### 4.1.3 Traffic Injection

Once the attacker has identified active IP addresses, they can inject traffic into the private network, establish TCP connections with private hosts, and receive replies from the injected traffic. For an attacker to inject and receive packets to and from a private host, they need to complete a TCP handshake from the perspective of the private host *and* the NAT. This becomes more challenging since the NAT will drop incoming `SYN` requests from the Internet [20].

A way to circumvent that is for the attacker to use the `SYN_SENT2` TCP state, also known as *Simultaneous Connection Synchronisation* [15, §3.5]. The attack is illustrated in Figure 3. Assume an attacker wants to determine whether port 80 is open on a selected client behind a NAT and inject HTTP traffic. In step 1, the attacker sends an encapsulated `SYN` packet with a random sequence number $x$. The inner packet is sourced from the private client to the attacker. The source port of this packet should be the port the attacker intends to access, while the destination port can be any random port (e.g., sport=80, dport=2000). In step 2, VH receives this packet, decapsulates it, and forwards it back to the attacker while setting the connection-tracking state to `SYN_SENT`. If port randomisation is enabled, then the forwarded packet will have a different source port (e.g., 3000). In step 3, the attacker sends to VH a second `SYN` packet with a random sequence number $y$ and the same port numbers as step 2, but in reverse (e.g., sport=2000, dport=80 or 3000). This packet updates the connection state to `SYN_SENT2`; thus, in step 4, the packet will reach the client. In step 5, the attacker sends an encapsulated `SYN/ACK` reply corresponding to the initial packet received in step 2 with the inner packet's TTL set to 1. The ack number of this packet will be $x+1$. When VH receives this packet and decapsulates it, the state will be recorded, and the destination IP will be translated, but the packet will not reach the client because its TTL will be decremented and it will expire in transit. More specifically, when VH receives and decapsulates the packet, it first updates the TCP state and then decrements the inner packet's TTL, which is set to 1. This way, although the inner packet has influenced the TCP connection, its TTL will reach 0 *at VH* and be discarded, ultimately never reaching the client. This will work regardless of the hops between VH and the Client; for example, it will also work if there is no hop between VH and the Client (if they are directly connected). Alternatively, if the client received this packet (with the TTL set to a value greater than 1), it would reset the connection. Simultaneously, in steps 6 and 7, the client responds to the `SYN` packet sent by the attacker in steps 3 and 4 with a `SYN/ACK` packet, using a new sequence number, thereby updating the connection state to `SYN_RECEIVED`. Since the attacker has both sequence and ack numbers of the connection, the final step is to reply with an `ACK` to complete the three-way handshake. This will enable the attacker to send HTTP requests and receive responses. Note that if the attacker-selected port is not in use by the client, the client will send a `RST` packet after step 4.

**Limitations.** A limitation of the attack is that the NAT must have `rp_filter` set to 0 (disabled) or 2 (loose mode) at the *tunnelling* interface. More specifically, the packet sent by the attacker in step 1 of Figures 2 and 3, when decapsulated, will then be dropped if the Reverse Path Filtering (RPF) is set to 1 (strict mode) at the tunnelling interface. This is because the inner IP source address is typically reached by the interface with a private address range, e.g., 192.168.0.0/24 and not the tunnelling interface. The kernel thus assumes this packet is spoofed and drops it. Note that on Linux, the default value of RPF is 0 for all interfaces, meaning it is disabled, although

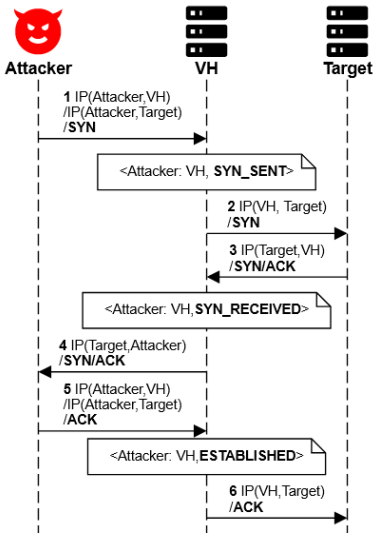


Figure 4: Two-way proxy attack, allowing an attacker to complete a TCP handshake behind VH's address.

some distributions enable it in startup scripts [29]. Furthermore, setting the RPF to 1 by default is not recommended, since it would break asymmetric routing [5]. For example, if a multihomed host has two interfaces, *eth*1 and *eth*2, and receives traffic on *eth*1, then if *eth*2 would be used to reply to the packet's source, the packet is dropped if strict RPF is enabled.

## 4.2 Two-way Proxy Attacks

In some cases, a NAT-enabled tunnelling host can be tricked into translating one global IP to another, ultimately making it a *two-way proxy*. A two-way proxy in the context of open tunnelling hosts is a host that can forward received replies to the attacker. We present two novel attacks against two-way proxies: 1) An attacker can establish a connection with a remote host behind the open tunnel's address, and 2) An attacker can conduct a DoS attack against two-way proxies by creating a NAT mapping between them, causing packets to loop until their TTL expires.

### 4.2.1 Two-way Proxy Handshake

In this attack, we show how an attacker can complete a TCP handshake with a victim while remaining anonymous behind a two-way proxy. The attack is illustrated in Figure 4, where VH is a two-way proxy that translates global IP addresses, and Target is the victim of the attack. In step 1, the attacker sends an encapsulated packet with the inner source address set to the attacker's address and the inner destination address set to a selected target. When VH receives this packet, it will decapsulate it, translate the source IP address to its own IP address, and forward the packet to the victim. In this case, the connection state will be set to `SYN_SENT`. In step 2, the target responds to VH with a `SYN/ACK`. VH will then change the state to `SYN_RECEIVED`, translate the destination address to the attacker's address, and forward the packet. In step 5, the attacker can finally send an encapsulated `ACK`, which is decapsulated, translated and forwarded to the target in step 6, completing the handshake. This can allow an attacker to deliver malicious payloads or conduct application-layer DoS attacks while remaining completely anonymous to the target.

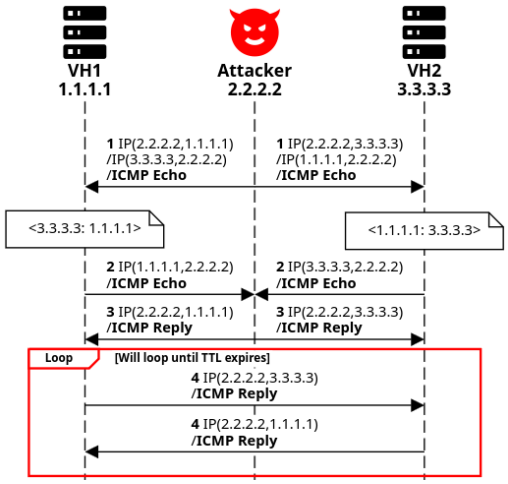


Figure 5: Two-way proxy loop attack. An attacker forces two hosts into forwarding traffic between them.

### 4.2.2 Two-way proxy loop DoS Attack

An attacker can abuse hosts that translate global IP addresses to facilitate a novel loop DoS attack. The attacker creates a source NAT mapping between two two-way proxies for a connection originating entirely from the attacker. Any subsequent traffic from the attacker will be forwarded between the two two-way proxies, until each packet expires in transit (TTL=0).

The attack is illustrated in Figure 5. In step 1, the attacker sends two different encapsulated packets to two different two-way proxies. Each of these packets has its inner source address set to the other's IP, and the inner destination address is set to the attacker's IP. For simplicity, we assume the header following the inner IP header is an ICMP Echo request, although other protocols may also be used. The two hosts will then decapsulate their respective outer headers, translate the IP source addresses to their global addresses, and forward the packet to the attacker in step 2. This will create a source NAT mapping between the two corresponding two-way proxies. In step 3, the attacker can respond to the two identical requests. The hosts, upon receiving these packets, will translate the destination address to the other two-way proxy and forward them, creating the loop shown in the red rectangle in the Figure. Finally, the packet will ultimately be dropped when its TTL expires; the attacker can repeat step 3 indefinitely.

**Limitations.** One limitation of our handshake attack is that the attacker cannot establish a connection using any spoofed address; only the address of the two-way proxy itself is possible. This only prevents the attacker from impersonating, as their identity remains anonymous in any case. An additional limitation is that the two-way proxy must be able to spoof, since the `SYN/ACK` packet received in step 4 will have its source address set to the Target.

The main limitation of our two-way proxy DoS attack is that the amplification factor depends on the number of nodes between the two proxies. For example, if the two hosts were adjacent, a single packet would be sent and received a total of 255 times by a single host (1 received from the attacker, 127 sent to $V_2$ and 127 received from $V_2$), resulting in an amplification factor of 255. We can estimate the amplification factor $f$ from the number of nodes between the two hosts (assuming the same path in both directions). If we assume that a packet travels from $V_1$ to $V_2$ and passes from $n$ hosts (including $V_1$ and $V_2$) in the manner: $V_1 \rightarrow h_1 .. \rightarrow h_{n-2} \rightarrow V_2$, then the packet passes a total of $n$ hosts, $n-1$ times in total, where $n \geq 2$. For the packet to be returned, it passes through the same number of hosts. We can approximately estimate the times host $V_1$ will receive a packet to be equal to $a \approx \frac{255}{2 \cdot (n-1)}$. Since we count both ingress and egress traffic, the amplification factor for $V_1$ is then $f \approx \frac{255}{n-1}$.

**Comparison with Ping-Pong** We can compare our attack with the Ping-Pong attack [6, §5.3], since both attacks attempt to amplify an attacker's traffic by abusing open tunnelling hosts. In Ping-Pong, an attacker targets two open tunnelling hosts by encapsulating multiple IP headers into a single packet, each header destined to one of the two victims interchangeably. For example, if a header is sourced from one host to another, then the following header will have the source and destination addresses in the opposite direction. When this packet is sent to a victim, the outermost header is decapsulated, and the remainder of the packet is then forwarded to the second victim. The process is then repeated until no more headers can be decapsulated, ultimately forcing two or more hosts to forward traffic between them. The idea is that the attacker will construct a very large packet with multiple encapsulated headers (e.g., GRE or IPIP), to cause a DoS against two open tunnelling hosts.

The Ping-Pong attack can leverage a higher amplification factor than the two-way proxy loop, since a single packet can be amplified by a maximum of 1638, whereas if we assume the two hosts in the two-way proxy attack to be directly connected, the maximum possible amplification factor is 255. An additional benefit of Ping-Pong is that the attacker's IP address can remain completely hidden from both victims, whereas in the two-way proxy loop attack, the attacker must reveal their IP address to receive a reply in step 2. For example, the TCP version of the attack shown in Figure 5 would require the attacker to send an `ACK` packet in step 3, where the ack number is determined by the seq number set in step 2 by the victims. Alternatively, the attacker may use an intermediary or a spoofed source to transport the packet when the reply does not require the attacker to predict a parameter (e.g., ICMP Echo Request/Reply). Finally, in Ping-Pong, the two hosts selected by the attacker do not need to spoof, whereas in the two-way proxy loop attack, both hosts must be able to spoof, since the packets sent in step 4 both have a spoofed source address (that of the attacker).

A benefit of the two-way proxy attack over Ping-Pong is that the attacker does not need to construct large packets to cause the amplification attack, allowing them to send more packets towards the two victims. An additional benefit over Ping-Pong is that, if packets with multiple IP headers are rejected as a defence mechanism introduced by [6], the attack packets will be dropped, while in the two-way proxy loop attack, the attacker only needs to have one depth of encapsulation, i.e., one encapsulated IP header. Finally, the two-way proxy attack can provide traffic variance, ultimately making the attacker avoid detection. The attacker can utilise multiple different protocols to create different NAT mappings between the victims. For example, an attacker can create multiple NAT mappings between two hosts, using different protocols (such as TCP and UDP) and ports, so the looped packets in step 4 of the attack will differ for each mapping. In Ping-Pong, the attacker must rely on sending tunnelled packets to the victims, which can make it easier for a victim to identify malicious intent when observing identical traffic.

## 5 GRETAP Attacks

As mentioned in Section 2, GRETAP is the L2 encapsulation variant of GRE. Although GRE attacks have been previously explored [6], it remains unknown whether an attacker can exploit GRETAP to attack the LAN. In this section, we explore how the encapsulation of L2 frames can enable traditional LAN attacks when forwarded by an open GRETAP tunnelling host. Furthermore, we also explore two novel attacks: 1) An attacker can leak a victim's global and private interface MAC addresses, and 2) an attacker can cause an infinite DoS loop.

**Topology.** The topology is illustrated in Figure 6. In this case, the open tunnelling host acts as a gateway, with a global and a private interface connected to the Internet and the LAN, respectively. For a remote host to communicate with the LAN, the tunnelling host will set up a bridge interface. This bridge interface connects the GRETAP interface and the host's *private* interface. The process typically operates as follows: the global interface receives the GRETAP packet, decapsulates the outer IP and GRE headers, and forwards the remaining payload to the GRETAP interface. This GRETAP interface forwards the frame to the bridge interface, which, in turn, forwards it to the private network via the private interface.

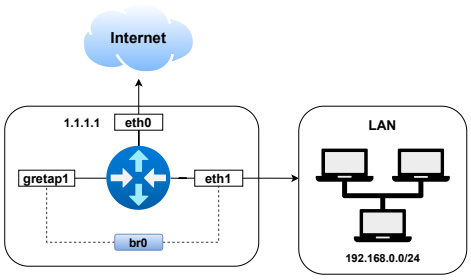


Figure 6: GRETAP Topology. `eth0` is the global interface and `eth1` is the local interface. The `gretap` and `eth1` interfaces are connected through the bridge interface `br0`.

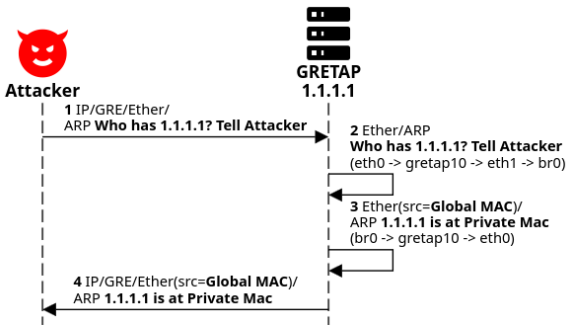


Figure 7: GRETAP MAC address leak. The attacker leaks the MAC addresses of the global and private interfaces on Linux.

## 5.1 GRETAP MAC Address Leak

An attacker can force an open GRETAP host to leak its private and public MAC addresses. The attack is shown in Figure 7. In step 1, the attacker sends a packet of the following structure: 1) An IPv4 packet from the attacker to the victim, followed by 2) a GRE header, 3) an Ethernet frame with random source and broadcast destination MAC addresses, and 4) an ARP request, requesting the victim's address (the source does not matter). Overall, the structure can be visualised as follows: IP/GRE/Ether/ARP. The victim will decapsulate the outer IP and GRE headers and forward the packet to the GRETAP interface in step 2. In step 3, when the packet reaches the bridge interface, the ARP request is processed, and an ARP reply is generated, with the MAC address set to the bridge interface address. The host will then forward the packet to the tunnelling interface, which will encapsulate it in a GRE header and forward it to the source specified in step 1. More specifically, since the tunnelling interface lacks a remote destination, the destination is chosen dynamically based on metadata collected from the initial ingress packet sent by the attacker [23]. The attacker will receive a packet of the following structure: IP/GRE/Ether/ARP. Although the ARP reply provides the victim's *private* interface MAC address, the preceding Ethernet frame uses the victim's *public* interface MAC address as its source. It is important to note that this attack is specific to Linux open tunnelling hosts. Furthermore, to the best of our knowledge, only ARP replies will be re-encapsulated [3].

**Limitations.** The Linux kernel provides two flags that can partially mitigate MAC address leakage on open GRETAP hosts. These flags are `arp_filter` and `arp_ignore` [29]. If `arp_filter` is set to 1 (default is 0), the host will not reply since the sender IP in the ARP request does not belong to the same subnet as the victim's private interface. To circumvent this, the attacker can use an address in the same private subnet as the victim in the ARP request's sender IP address. Although this appears to be a trivial bypass, an attacker would need to predict the victim's private subnet space. If `arp_ignore` is set to greater than 0 (default is 0), the attacker would need to ensure that the destination IP address of the ARP request is equal to the victim's private IP address. If the flag is set to 2, the sender IP must be on the same subnet, and the destination IP must match the victim's private address. Although predicting the victim's private IP address is non-trivial, in a targeted attack, receiving a reply can also reveal whether the predicted address belongs to the victim, ultimately leaking more information.

## 5.2 GRETAP ARP Spoofing

An attacker can spoof ARP responses to cause a DoS attack against hosts behind the victim's private network. The attacker can send an ARP request for a specific IP address in the same private subnet, as described in Section 5.1. When the victim replies, the attacker can follow up with an ARP response of the following structure: IP/GRE/Ether/ARP. The inner Ethernet frame has the broadcast address as the destination, the ARP sender IP is sourced from a private client's IP, and the target destination is the victim's private IP. The hardware address of this ARP response can be any random MAC address. The victim will update its ARP table to a random MAC address, causing packets destined for the private client's IP address to be dropped until the victim generates a new ARP request for this broadcast domain. An attacker can repeat the process to map all addresses of the subnet to random MAC addresses, ultimately causing traffic towards private clients to be dropped.

## 5.3 GRETAP Infinite DoS

When an attacker receives the host's MAC address assigned to the private interface, as shown previously in Section 5.1, they can launch a powerful infinite DoS attack against the victim. This attack is based on the previous ARP Spoofing attack, but instead of mapping an address to a random MAC address, the attacker maps an address in the same subnet to the MAC address of the GRETAP host. Subsequently, the attacker must predict the victim's private IP subnet, e.g., 192.168.0.0/24.

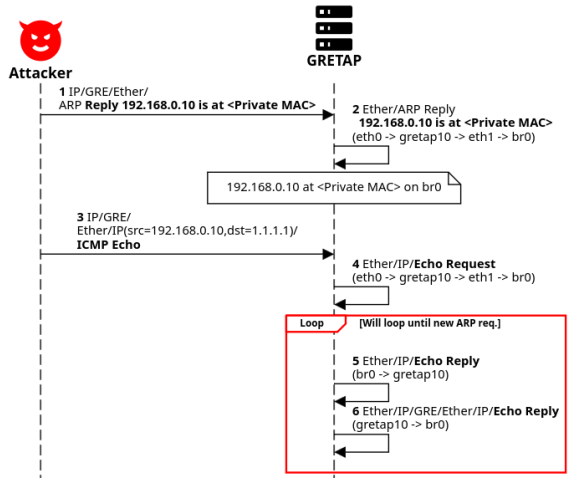


Figure 8: GRETAP Infinite DoS attack. The attacker creates an ARP entry that maps the host's MAC address to another address on the same subnet. Packets with that source address will be looped and re-encapsulated continuously.

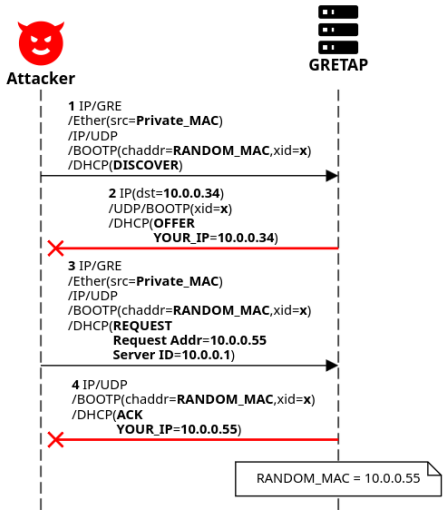


Figure 9: DHCP Starvation attack against a GRETAP open tunnelling host.

The attack is shown in Figure 8. For this example, we assume the host has a global address 1.1.1.1 and a private address 192.168.0.1. In step 1, the attacker can encapsulate an ARP reply with the target IP set to either the victim's global or private IP address, and the sender IP set to an address in the same private subnet as the victim, e.g., 192.168.0.10. The Ethernet frame preceding the IP header has the global MAC address as the destination and the private MAC as source. In step 2, once the packet is received and decapsulated, it is forwarded across interfaces until it reaches the bridge interface (`br0`). Furthermore, the host updates its ARP table and maps the address 192.168.0.10 to the MAC address of its private interface. In step 3, the attacker sends an encapsulated ICMP Echo request, sourced from address 192.168.0.10 and destined for the victim's global address (1.1.1.1). This packet will eventually reach the bridge interface, which will generate a reply and forward it to the GRETAP interface. Once the GRETAP interface receives this reply, it will encapsulate it in a GRE header and an Ethernet frame and forward it back to the bridge interface. This process will continue indefinitely until the host generates a new ARP request. Furthermore, each time the host forwards a packet to the GRETAP interface, it will be encapsulated in GRE.

## 5.4 GRETAPv6 DNS Poisoning

An attacker can leverage an open GRETAP host to send an IPv6 router advertisement and change the network's default DNS server. More specifically, an attacker can encapsulate an IPv6 router advertisement with the Recursive DNS Server (RDNSS) option extension. The RDNSS option allows a network to dynamically acquire a DNS configuration, such as the default DNS server [32]. In the RDNSS header, the attacker can include the IPv6 address of a DNS resolver under their control, along with the entry's lifetime. Clients behind that network will update their default DNS entries to the new address under the attacker's control. The structure of the packet sent by the attacker is as follows: IP/GRE/Ether/IPv6/ICMPv6 (router advertisement)/RDNSS. Furthermore, the attacker does not need to predict the router's link-local address, since the source IP address in the inner IPv6 header can be any link-local address. Additionally, the attacker can remove prior DNS entries by sending the same packet with a lifetime set to 0. If the attacker knows the previous DNS resolver's IPv6 address (e.g., the ISP's default DNS address), they can flush it from the DNS cache.

## 5.5 GRETAP DHCP Starvation Attack

An additional protocol that can be abused in this threat model is DHCP. DHCP is a network protocol used by hosts to automatically obtain configuration information, such as the network's default DNS server or an IP address [10]. In the presented threat model, an attacker cannot receive DHCP replies when using an open GRETAP interface on Linux; consequently, many attacks that rely on receiving DHCP server replies are not possible. That said, a DHCP starvation attack remains possible. DHCP starvation aims to exhaust all available IP addresses in the subnet, preventing new clients from obtaining an IP address when joining the network [2]. The attack is shown in Figure 9. After the attacker has acquired the MAC address of the private interface, as explained in Section 5.1, they can begin the DORA process. DORA is the process of acquiring an IP address on a network and consists of four message exchanges: DISCOVER, OFFER, REQUEST,

and ACK. For the remainder of this attack, all DHCP packets are encapsulated in UDP headers with source port 68 and destination port 67. The attacker begins in step 1 by sending an encapsulated DISCOVER packet within a broadcast Ethernet frame whose source MAC address is the router's private address. This DISCOVER packet has the source set to the "any" IPv4 address (0.0.0.0) and a broadcast destination. The BOOTP Transaction ID (`xid`) and client hardware address (`chaddr`) fields contain random values. The server will send a unicast OFFER in step 2, with an available IP address. This OFFER message will not be received by the attacker and will be dropped by the router since the MAC address used in the DISCOVER message of step 1 is spoofed. In step 3, the attacker can reply with an encapsulated REQUEST broadcast packet, using the same `xid` and `chaddr` as the initial DISCOVER packet, and request an address, using the DHCP server's address as the Server ID. If the attacker requests *any* address that is not already reserved (even if different from the address offered by the server in step 2), the DHCP server will reply with an ACK and lease that address to the random MAC address included in the `chaddr` field of the two packets. Furthermore, the attacker can repeat the process using random MAC addresses, exhausting all IP addresses on the network.

**Limitations.** A limitation of this attack is that the attacker must predict the DHCP server's IP address for the server ID field of the DHCPREQUEST packet, as shown in step 3 of Figure 9. This becomes increasingly difficult when the DHCP server is not provided by the router. If that address is incorrect, the server will reply with a DHCPNAK.

## 6 Scanning Methodology

To understand the potential impact of our attacks, we conducted 9 Internet-wide scans to identify hosts that may be susceptible. In total, we conducted 6 NAT scans, 1 broadcast scan and 2 two-way proxy scans, shown in Table 1.

### 6.1 Private Address Scans

The general goal of private address scans is to identify hosts that, when receiving encapsulated packets with a private inner IP source address (e.g., 10.0.0.1), will decapsulate them and translate the source IP address before forwarding the traffic to the destination. A non-intrusive way of identifying NAT-enabled open tunnelling hosts, is to repeat the first step of the attacks shown in Figures 2 and 3, but instead of encapsulating a `SYN` packet, we encapsulate an ICMP Echo request. A more concrete way to determine whether a host can be exploited by our traffic-injection attack is to follow the complete methodology presented in Section 4.1.2. We did not perform such scans, as this would involve sending multiple intrusive packets to hosts and accessing their private networks. The probe structure is as follows: 1) The outermost IP header is sourced from the scanner to the probe address. 2) The inner IP header is sourced from the private IP address and destined to the scanner, and 3) the innermost header is an ICMP Echo request. We scanned for both IPIP and GRE hosts, where for GRE hosts, we insert a GRE header between the two IP headers. The underlying idea of the scan is that the probed host receives our packet, decapsulates it, translates the private IP source address into its corresponding global IP address, and forwards the ICMP Echo Request to the scanner. In this scanning method, the header following the inner IP header is essential. If an ICMP Echo *reply* were to be encapsulated instead, the packet might be dropped since the connection tracking framework has not previously recorded an incoming ICMP Echo request. This observation was confirmed when we first conducted a 1% scan, in which an ICMP Echo *reply* was encapsulated and received drastically fewer replies.

To identify differences in the number of hosts translating specific private addresses, we conducted three scans for GRE and IPIP, respectively, each targeting an inner private address from a different private subnet. The following three addresses were selected: 192.168.0.1, 10.0.0.1, and 172.16.0.1. Furthermore, the same scanning methodology can be used to identify hosts that can spoof Martian packets, i.e., packets on the Internet with a private/reserved source. Since our scanner resides in a filtered network, ingress packets with private source addresses will be filtered and dropped.

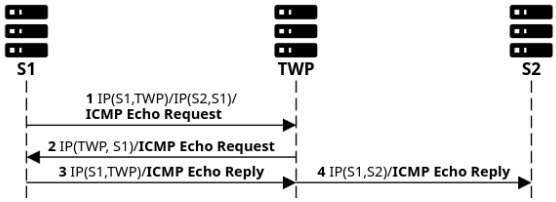


Figure 10: Two-way proxy scan example. Our S1 sends an encapsulated packet with the inner destination set to itself. When TWP receives and replies to this packet, we expect the reply to be forwarded to S2 by TWP.

### 6.2 Two-way Proxy Scans

We conducted scans to determine whether NATs could be manipulated to create *two-way proxies*. We scanned for GRE and IPIP open tunnelling hosts using a novel scanning methodology illustrated in Figure 10. This methodology requires two scanners under our control (S1, S2): one scanner will be sending the probes, while the second will listen for any potential responses. In step 1, S1 encapsulates an IP header with the source and destination IPs set to S2 and S1, respectively, followed by an ICMP Echo request. When the Two-Way Proxy (TWP) receives this probe, it decapsulates it, translates the source IP address (S2) to its own address, and, in step 2,

sends the packet back to S1. In step 3, after S1 receives the ICMP Echo request, it replies to TWP with an ICMP Echo reply. Upon receiving this ICMP Echo reply, TWP will translate the destination address to S2's address and forward the reply. S2 will then listen for packets with S1's source IP address and record the address encapsulated in the ICMP Echo request as vulnerable.

The actual address of TWP is split in half and populates the ICMP ID and SEQ fields of the probe (16 bits each). When the second scanner receives the response in step 4, it can extract these two values and reconstruct the complete IPv4 address. Furthermore, to reduce false positives, we established a firewall rule on S1 that drops ICMP Echo responses to S2. This is because, in the case where the open-tunnelling host does not implement NAT, the packet in step 2 will have S2 as its source, ultimately causing our first scanner to reply directly to our second scanner, creating a false positive.

## 6.3 GRETAP Broadcast Scan

A trivial way to identify hosts vulnerable to our GRETAP attacks would be to send a probe identical to that shown in Figure 7. Because this packet would leak the host's MAC addresses, we did not conduct such a scan for ethical reasons and therefore opted for an alternative methodology. To identify hosts susceptible to the attacks introduced in Section 5, we scan the Internet for GRETAP hosts that can forward broadcast ICMP Echo requests. We opted for a broadcast scan since 1) our attacks mainly rely on the forwarding of broadcast Ethernet frames and 2) we cannot predict the MAC address of a host to use as the destination of the Ethernet frame. The probe structure is as follows: 1) IP header, 2) GRE header, 3) Ethernet frame with a broadcast MAC destination address and a random source, 4) IP header sourced from the scanner to the broadcast IPv4 address, 5) ICMP Echo request. In the ICMP Echo request data section, we include the probed IP address to compare with the responding addresses. The main idea behind the scan is that an open tunnelling host will decapsulate the outer IP and GRE headers and forward the inner Ethernet frame and IP header to every host on their broadcast domain, which will reply with an ICMP Echo reply to the source of the packet, i.e., our scanner. Depending on the configuration, some hosts may forward broadcast packets, but hosts on the private network may not respond to broadcast ICMP Echo requests. For example, on Linux, the flag `icmp_echo_ignore_broadcasts` is set to 1 by default, ultimately causing broadcast ICMP Echo requests to be ignored [29]. In that case, only the GRETAP host will reply. In any case, although this scanning methodology does not guarantee that hosts are vulnerable to our attacks, it nevertheless indicates that a host is a GRETAP host that decapsulates packets from any source and forwards broadcast Ethernet frames.

**GRETAPv6 Multicast Scan.** As mentioned in Section 3.1, we additionally conducted a GRETAPv6 scan for 1% of the IPv4 address space, which did not yield any results. In this scan, the sent probe encapsulates an ICMPv6 Echo Request packet inside a GRETAP packet. More specifically, the packet structure is the following: IPv4/GRE/Ether/IPv6/ICMPv6. The destination MAC address of the Ethernet frame is the multicast `33:33:00:00:00:01`, while the destination IPv6 address is the "all-nodes" address (`ff02::1`) and the source is the IPv6 address of the scanner. The idea of this scan is that the receiving host decapsulates the outer IPv4 and GRE headers and forwards the inner multicast Ethernet ICMPv6 Echo requests to all hosts in that domain, which will ultimately reply to the source of the IPv6 header with an ICMPv6 Echo Reply. Since a 1% scan did not reveal any hosts, we did not complete a full scan.

### 6.3.1 Internet-wide Scan Limitations

Internet-wide scans have the inherent limitation of false-negative results. A factor that can lead to false-negatives is being blocklisted. Because some organisations may not want to receive scanning probes, they might block the scanning IP address or IP ID used by ZMap, causing some hosts to drop our probes. This can also impair scans performed at a later time. For example, our IP address may be blocked during initial scans, resulting in fewer replies in subsequent scans. A second factor is packet loss. Since there is no retransmission, our probes or host replies may be lost in transit. Furthermore, to limit the number of probes each host receives, we ran each scan only once, further reducing our ability to eliminate false-negatives.

Furthermore, since our GRETAP Broadcast scan relies on a non-intrusive scanning methodology to avoid leaking hosts' MAC addresses, there is a possibility of false-positive results. It is possible that, with our current scanning methodology, we detect hosts that implement open GRETAP interfaces and forward broadcast packets, but might not re-encapsulate ARP responses. This means that although the attacks presented in Section 5 might still be possible, they become more difficult because the attacker would need to predict the correct MAC addresses of the interfaces.

# 7 Scanning Results

In this section, we discuss and analyse the results of our 9 scans. We conducted these scans from March until September of 2025, with each scan taking approximately 4 days to complete. We use the term *replying addresses* to denote addresses that responded to our scans. More specifically, in a received reply, the *replying address* is the source IP address of the packet. Furthermore, we use the term *probing address* to denote the address to which our traffic is directed, i.e., the destination address of our probe. We make this distinction

Table 1: Scan results. The table shows the scan type, tunnelling protocol and inner source/destination (S/D) IP where applicable, along with the number of probed addresses identified.

| # | Scan Type | Proto. | Inner Src/Dst | # Vuln. |
|---|---|---|---|---|
| 1 | NAT | GRE | S=192.168.0.1 | 142,235 |
| 2 | | GRE | S=10.0.0.1 | 143,811 |
| 3 | | GRE | S=172.16.0.1 | 138,029 |
| 4 | | IPIP | S=192.168.0.1 | 20,172 |
| 5 | | IPIP | S=10.0.0.1 | 20,540 |
| 6 | | IPIP | S=172.16.0.1 | 21,078 |
| 7 | Broad. | GRETAP | D=255.255.255.255 | 147,167 |
| 8 | Two-way Proxy | GRE | S=S2 (Scanner 2) | 26,585 |
| 9 | | IPIP | S=S2 (Scanner 2) | 6974 |

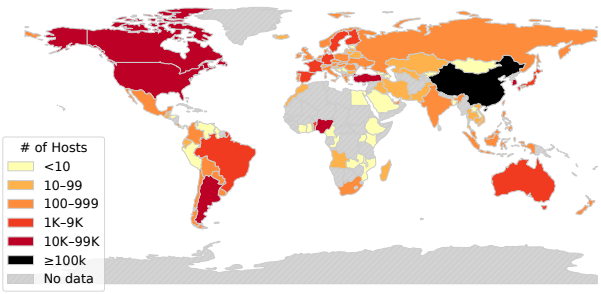


Figure 11: World map of identified hosts based on country. The colours represent buckets from $< 10$ to $\geq 100,000$.

since, as shown in this section, on some occasions the *replying address* and *probing address* differ.

Moreover, we used *IPtoASN* [30] to identify the country and Autonomous System (AS) for each collected address. Using the *Censys* Universal Internet Dataset [12], we collected additional host information when available. Censys conducts daily Internet-wide scans and makes the collected data available to researchers. It is important to note that we collect information on both probed and responding addresses when they differ. Through Censys, we gather OS, open ports, associated domains, labels (given to hosts by Censys) and software vendors and products for each IP address collected from our scans. We did not conduct any port or application layer scans ourselves and relied on Censys for additional data. This additional information helps us identify potential outliers/prevalences, thereby enabling us to better understand why the identified hosts respond to our scans and potentially the underlying properties of those hosts. Furthermore, the additional information assists us during the disclosure process in identifying hosts that appear prevalent, e.g., those belonging to a specific network/organisation with a common misconfiguration. We identified a total of 438,280 probed addresses across all our scans, while the total number of addresses collected, including both probed and replying addresses, was 446,277. We present our results in Tables 1 and 2.

## 7.1 NAT Scan Results

### 7.1.1 Private Address Scans

**GRE Scans.** A total of three GRE scans were conducted, each using a different private source IP, resulting in the identification of 265,780 hosts. The most prevalent territories identified were China, followed by Argentina and South Korea, with China accounting for over 70% of hosts across all three scans. The most vulnerable AS was AS4134 with more than 27% of hosts. Regarding open ports, we identified that the three most prominent were Session Initiation Protocol (SIP), Simple Network Management Protocol (SNMP), and Internet Key Exchange (IKE) across the three scans. SIP is an application-layer protocol used to establish and manage multimedia communication sessions [43]. Given the high number of hosts with the SIP port open, we conjecture that these hosts may be SIP servers, possibly handling multimedia communications such as Voice over IP (VoIP) [34]. This is also in line with the label assigned by Censys to these hosts, with the most common label for GRE scans being "voip", although it is unclear whether these hosts utilise GRE tunnels to facilitate VoIP services. The top domain associated with these hosts, with over 27% of hosts, is `ipnxtelecoms`. OS vendors were available for 3153 of probed addresses, with the top 4 being Hikvision, Synology, Microsoft, and Ubuntu, with 22%, 16%, 16%, and 11% of hosts, respectively.

**IPIP Scans.** For IPIP scans, we identified a total of 34,384 hosts. The three most prevalent countries identified in these scans were South Korea, Australia and the US, with over 34%, 11% and 9% of hosts, respectively, across all three subnet addresses. The most vulnerable AS was AS4766 with over 29% across the three scans. The three most prevalent ports were those associated with SNMP, IKE, and Network Time Protocol (NTP). This differs from the GRE, in which SIP was the most prevalent protocol. The most prominent label assigned to these hosts is "remote-access" (presumably Secure Shell (SSH)), with other labels, such as "network.device.vpn", also prevalent. We conjecture that the IPIP hosts identified in our scans may act as VPN servers, since "remote-access" indicates a host is running an SSH service, while the VPN label indicates a host is running a VPN service such as IKE. The most vulnerable domain was `myvzw`. Similar to GRE, the 4 main OS vendors identified among the 1446 probed addresses were Hikvision (31%), AXIS (15%), Microsoft (11%), and Ubuntu (10%).

**Duplicate Responses.** Interestingly, we notice that for various probes, we receive multiple replies from the same address. More specifically, some hosts respond to multiple probes, meaning the reply address differs from the probing address.

Table 2: Most prevalent Countries, AS, Ports, Domain and Label for each of the 9 scans. The first column refers to the scans in Table 1. The label is an identifier assigned by Censys for a specific host. The "ne-ad" label refers to "network-administration" and "re-ac" refers to "remote-access".

| # | Top 3 Countries | Top AS | Total with ports | Top 3 Ports | Total with domain | Top Domain | Total with Labels | Top Label |
|---|---|---|---|---|---|---|---|---|
| 1 | CN (73%),AR (6%),KR (5%) | 4134 (29%) | 55,205 | SIP (43%),SNMP (33%),IKE (18%) | 3208 | ipnxtelecoms (32%) | 33,046 | voip (44%) |
| 2 | CN (72%),AR (8%),KR (5%) | 4134 (29%) | 55,961 | SIP (43%),SNMP (32%),IKE (17%) | 3628 | ipnxtelecoms (33%) | 34,057 | voip (43%) |
| 3 | CN (71%),AR (9%),KR (5%) | 4134 (27%) | 52,428 | SIP (43%),SNMP (30%),IKE (16%) | 3559 | ipnxtelecoms (27%) | 35,114 | voip (44%) |
| 4 | KR (37%),AU (11%),US (10%) | 4766 (32%) | 16,435 | SNMP (49%),IKE (48%),NTP (45%) | 2952 | myvzw (26%) | 10,298 | re-ac (68%) |
| 5 | KR (34%),AU (11%),US (9%) | 4766 (30%) | 16,340 | SNMP (46%),IKE (46%),NTP (43%) | 3139 | myvzw (25%) | 10,572 | re-ac (68%) |
| 6 | KR (34%),AU (13%),US (9%) | 4766 (29%) | 17,074 | IKE (44%),SNMP (44%),NTP (42%) | 3170 | myvzw (24%) | 11,475 | re-ac (66%) |
| 7 | CN (49%),CA (23%),US (10%) | 56046 (18%) | 51,558 | L2TP (92%),SNMP (7%),NTP (2%) | 2394 | telus (68%) | 3245 | ne-ad (90%) |
| 8 | US (28%),NG (22%),BJ (16%) | 29091 (22%) | 12,265 | SIP (61%),DNS (16%),NTP (7%) | 1870 | ipnxtelecoms (63%) | 8042 | voip (44%) |
| 9 | US (42%),SE (14%),AU (14%) | 7474 (11%) | 2250 | NTP (36%),SNMP (34%),L2TP (30%) | 386 | bredband2 (8%) | 1430 | ne-ad (66%) |

The maximum number of times a host replied on behalf of others occurred during the GRE-192.168.0.1 scan, where a total of 822 replies were recorded. In the GRE-172.16.0.1 scan, a specific host replied on behalf of 2904 different probes. Using Censys search, we identified that although the specific host had no information available, the second-most prevalent replying host which replied on behalf of 1734 probes, exposes a login page on a non-standard port, and the title lists "Firewall" in Chinese. Therefore, we conjecture that this host acts as a firewall for the specific subset of hosts. We also identify that some probes will generate multiple replies. In the GRE-172.16.0.1 scan, 883 probes generated more than one reply, whereas in the same scan, 7 probes generated a total of 41 replies.

**Address Subnet impact.** We compare the addresses collected in each scan to determine whether the choice of subnet impacts the results. For GRE, the largest percentage of intersected addresses is between 192.168.0.1 and 10.0.0.1, with 93,840 (48%) of hosts being in both sets. The lowest intersection percentage was between 192.168.0.1 and 172.16.0.1, with 57,148 (25%) addresses appearing in both sets. Overall, only 19% of hosts appear in all three scans. For IPIP, the greatest intersection is between 192.168.0.1 and 10.0.0.1 (46%), whereas the lowest is between 192.168.0.1 and 172.16.0.1 (42%). A total of 31% of hosts appear in all three scans. We conjecture that although many hosts are common across the three subnets, most hosts remain unique to a specific scan. This could also indicate that private address scans may leak a host's private subnet.

#### 7.1.2 Two-way Proxy Scan Results

We identified a total of 26,585 and 6974 GRE and IPIP two-way proxy hosts, respectively. The total number of accumulated addresses is 27,286, which entails an overlap of 6273 addresses identified in the two scans. In both GRE and IPIP two-way proxy scans, we find that the most vulnerable country is the US with 28% and 42% of hosts, respectively. For GRE, the most prevalent open port is SIP, while the most prevalent label is "voip". This observation is similar to the private address GRE scans. For IPIP, three ports have similar prevalence: NTP, SNMP, and Layer 2 Tunneling Protocol (L2TP) are open in 36%, 34%, and 30% of hosts, respectively. The most prevalent label for IPIP hosts is "network-administrator", which we assume indicates that these hosts run network-administration protocols, such as SNMP or NTP. The most vulnerable AS in the GRE and IPIP scans were AS29091 (22%) and AS7474 (11%), respectively. Furthermore, only 179 had OS vendors associated with them, where Synology (32%), MikroTik (11%), Digi (9%) and Microsoft (8%) were the most prominent.

### 7.2 GRETAP Broadcast Results

Our GRETAP broadcast scan identified a total of 147,167 addresses. A total of 138,431 hosts responded with an address different from the one probed. Furthermore, 195 hosts generated more than one reply. The greatest number of replies generated from a single probe was 12 by one host, 9 by a total of seven hosts, 8 by one host and 2 by the remaining 186 hosts. Furthermore, of 195 hosts, only 4 replied with the same address as the one probed. A total of 8546 hosts replied on behalf of others, with one host responding for up to 1396 distinct probed addresses. More interestingly, 92% of hosts with information obtained via Censys had the L2TP port open. Further manual inspection of GRETAP hosts on the Censys search engine did not reveal additional information. Since the majority of these hosts only expose the L2TP UDP port, an alternative protocol to GRETAP but at the transport layer [47], we conjecture that these hosts are gateways to private networks. Additionally, 68% of hosts with domain information were associated with `telus`. Only 12 probed addresses had information regarding OS vendors. The most prominent were Cisco, Mikrotik, Ubuntu, and Microsoft, with 4, 4, 3, and 2 hosts, respectively. The three most prominent territories identified in the GRETAP scan were China, Canada and the US, with 49%, 23% and 10% of hosts, respectively, while the most prominent ASs was AS56046 with 18% of hosts.

**DoS Amplification.** Since some probes generate up to 12 replies, this can enable a trivial DoS attack. An attacker can

send a single probe with a spoofed inner-source address set to the victim's address. The GRETAP host will forward the inner packet to all hosts in its broadcast domain, which will all reply to the victim. In our examples, an attacker could amplify their traffic by a factor of 12 for each probe sent.

## 7.3 General Results

The general geolocation results across all scans are shown in Figure 11. We identified 446,277 unique addresses (probed and replied), across 127 different territories and 1861 ASs. The GRETAP broadcast scan identified the most hosts (147,167), whereas the IPIP two-way proxy scan identified the fewest (6974). The countries with the most vulnerable hosts were China, Canada and the US, with 280,294 (63%), 37,973 (9%) and 26,973 (6%) of hosts, respectively. The most vulnerable ASs were AS4134, AS4812, and AS9808, with 115,323 (26%), 49,515 (11%) and 28,327 (6%) of hosts, respectively. Furthermore, we compare the intersection of addresses for each *category* of scans. Private NAT and two-way proxy scans have the greatest number of intersecting addresses, with 16,826 addresses in common. This entails that 6% of the addresses collected in both sets (62% of two-way proxy addresses) translate *both* private and public IP addresses. Interestingly, the GRETAP broadcast and two-way proxy scans had *zero* intersecting addresses. Overall, our results showcase the prevalence of both open NAT-enabled and GRETAP tunnelling hosts in the wild, making a wide-range of networks susceptible to our identified attacks. Furthermore, it is apparent that vulnerable hosts are present across many territories and ASs, but they are highly prevalent in China.

### 7.3.1 Software Vendors & Products

Censys provides information about software running on the identified hosts' ports. An example of such information is the vendor which made the software, as well as the product itself. This means that one host might run multiple software applications on multiple ports. We gather this additional information to help us better categorise the identified hosts. A total of 222 hosts identified through our GRETAP broadcast scan run OpenSSH. Another interesting finding is that both IPIP and GRE two-way proxy scans have "DahuaSecurity" as the most prominent vendor, with 191 for IPIP and 179 for GRE. Dahua security provides several network cameras [8]. Upon further inspection, the service provided by these hosts is an HTTP login page for a network security camera.

For IPIP-NAT scans, the most prominent vendor is EmbedThis, a device manager for smart devices [17], with over 500 hosts across all scans. We also notice Digi routers use EmbedThis software to provide their login pages (more details about Digi are discussed in Section 7.3.2). In contrast, the most prominent product for GRE-NAT scans is Dropbear SSH, an SSH client/server for Unix-based and Linux OSs [11], with over 400 hosts across all three scans using both. For the GRE-192.168.0.1, the most prominent vendor was Microsoft, with multiple products such as Windows, Windows Embedded Compact (CE), and Windows Server.

### 7.3.2 Case Studies

Using data collected by Censys [12], we discovered that many of the identified hosts expose HTTP login pages. These login pages are often served on non-standard ports (not 80 or 443) and provide additional information about the identified hosts, such as the OS and device type, which might not be available on Censys. Using the labels assigned by Censys, we explicitly collect a subset with the "login-page" label and manually explore it further through the Censys search engine. We present some interesting case studies:

*Digi Cellular Routers.* IPIP scans, including NAT and TWP, reveal hosts running Digi login pages, along with the Digi router model on the HTTP title. We identified pages of the WR21 and WR44 cellular router models. These devices indeed support IPIP, as indicated in the vendor's statement regarding CVE-2020-10136 [9, 27]. These cellular routers also provide NAT functionality and could, in theory, be susceptible to our identified attacks.

*NexG VForce.* A prevalent recurring login page is that of NexG VForce. Although it is not clear which specific product is vulnerable, NexG VForce is a VPN router with multiple capabilities, including NAT, VoIP support, and tunnelling [40]. These hosts all have the following ports open: NTP (123), SNMP (161), IKE (500), TELNET (2650) and HTTP (17877). Furthermore, all of these hosts are shown in Korea. We conjecture that a misconfiguration of these devices might be causing hosts to respond to our scans and, by extension, making them susceptible to the identified attacks. We find that, for all scans except the GRETAP broadcast and two-way proxy scans (two-way proxy scans had 7 and 6 hosts respectively), there are more than 5000 probed hosts with the exact same port list, for a total of 7584 unique probed hosts across all scans.

*Milesight.* An additional observation is that some hosts appear to host a Milesight [36] login page. We observe that the HTML title lists the device as an Industrial router, an Industrial Cellular router, or a LoRaWAN Gateway, depending on the host. A common characteristic of LoRaWAN routers is that they run Message Queuing Telemetry Transport (MQTT), a publisher-subscriber protocol widely used in IoT devices [39]. We identify 958 probed hosts with port 1883 open across all scans (except for GRETAP), which could indicate the use of similar IoT gateway devices.

## 8 Defences

**GRETAP Defences.** Ideally, hosts should use more secure tunnelling protocols, such as the IPsec suite, which incorporates encryption and authentication [21]. IPsec can be used in

conjunction with GRETAP when Ethernet frame encapsulation is required [6]. For L2 encapsulated attacks, hosts should ideally drop encapsulated broadcast packets when unnecessary. More specifically, the attacks demonstrated in Section 5 rely on encapsulated broadcast frames. An additional recurring issue is the implementation of "*open*" tunnelling hosts, which decapsulate packets from arbitrary sources. We conjecture that the prevalence of open tunnelling hosts may be due to Linux not supporting the allowlisting of multiple remote tunnel addresses, rendering multipoint tunnels infeasible unless no remote address is specified. A defence would be to allow users to specify multiple allowed remote addresses when creating a GRETAP tunnel. For example, instead of a single remote address, a host could specify a subnet or multiple remote addresses from which packets could be decapsulated. This would mean that tunnelling hosts could be configured as multipoint without relying on firewalls. In this case, multipoint tunnels could dynamically re-encapsulate packets only if the source is one of their trusted remote hosts.

**NAT Attack Defences.** If RPF is enabled at the tunnelling interface, then the packets in step 1 of the attacks presented in Section 4.1.3 will be dropped. Furthermore, we recommend permitting simultaneous open connections only on specific ports. This will limit attackers to accessing only ports where clients expect `SYN` packets, such as in peer-to-peer protocols. Furthermore, a more concrete defence is to disallow encapsulated packets from altering TCP states. If the tunnelling host is also the NAT, all encapsulated TCP packets can be forwarded with metadata indicating that the packet should not alter the TCP state. This would effectively prevent an attacker from manipulating the NAT state and thus mitigate the attacks demonstrated in Section 4.

**DoS Loop Defences.** For the DoS loop attacks presented in Section 5.3, we propose a loop detection algorithm. In general, loops are mitigated by the TTL value in a packet; however, in our case, because the packet is re-encapsulated each time and a new IP header is added, the TTL effectively does not decrement. A loop detection algorithm could track packets passing between interfaces through metadata, ultimately dropping them after a certain number of ingress and egress flows.

## 9 Related Work

**Open Tunnelling Hosts.** Beitis & Vanhoef showed that over 4 million hosts will decapsulate packets without verifying the sender and showcase how attackers can use such hosts to cause a DoS, with attacks such as *Ping-Pong* and *TuTL*, with amplification factors of 16 and 75, respectively [6]. Furthermore, they showed how vulnerable tunnelling hosts can be abused as one-way proxies that decapsulate and forward traffic to selected victims, ultimately allowing an attacker to remain anonymous while conducting attacks. Their spoofing scans identified one-way proxies capable of spoofing arbitrary traffic, making them ideal for volumetric network-layer DoS attacks. In this paper, we demonstrated how a subset of those tunnelling hosts can be abused as two-way proxies, enabling attackers to not only forward traffic through them but also receive replies, ultimately increasing the attack surface. Unlike one-way proxies, which can be abused to spoof traffic and conduct DoS attacks, two-way proxies can enable attackers to establish TCP connections behind tunnelling hosts, enabling a broader range of more sophisticated attacks. We also demonstrate a novel two-way proxy loop attack that forces two two-way proxies to forward traffic between them.

Although previous research has conducted scans to identify open tunnelling hosts [1,4,6,7,25,33], we are the first to scan for NAT-enabled and GRETAP tunnelling hosts. Beitis & Vanhoef conducted 26 scans using 7 methodologies to identify IPv4 and IPv6 hosts that implement 6 tunnelling protocols without authentication [6]. While they conducted scans to identify vulnerable IPIP and GRE tunnelling hosts, this work studied in more detail how they can be abused. In particular, our scans reveal tunnelling hosts that are NAT-enabled, and through our traffic-injection attack, we demonstrate how benign clients behind such hosts can be targeted by remote off-path attackers. We are also the first to conduct novel Internet-wide scans for open GRETAP tunnelling hosts. Our GRETAP scans are the first to consider hosts that will decapsulate and forward Ethernet frames, allowing us to identify tunnelling hosts that could enable attackers to access private networks, as we demonstrated through 5 attacks.

**NAT Attacks.** Feng et al. [19] demonstrate that a remote attacker can identify NATs via a PMTUD side channel and destroy TCP connections, causing a DoS for victims behind a NAT. Through their vantage points, they identified a total of 7605 NATs and evaluated 180, out of which 92% were vulnerable to their DoS attacks. Furthermore, Mixon-Baca et al. [37], among other attacks, demonstrate that an attacker can conduct a port scan of hosts connected to the same VPN server. The attacker establishes a connection with a remote server under their control and then disconnects from the VPN server. A victim who later connects to the VPN server and is assigned the same IP address is thereby exposed to a simultaneous open connection, enabling an attacker to fingerprint them. Our traffic-injection attack assumes a completely remote attacker and does not require the victim to connect to the same VPN server. Furthermore, our attack allows fingerprinting and traffic injection for *all* hosts behind a NAT, without requiring a concurrent connection from a victim. Tolley et al. [46] can infer private VPN client addresses, similar to our fingerprinting attack. In our attack, the attacker is remote and off-path, and does not require the victim to connect to an access point under the attacker's control. Although others have

demonstrated various attacks against NATs [22, 42, 48, 49], we are the first to consider open tunnelling hosts as the means to enable remote NAT attacks.

## 10 Conclusion

In this paper, we examined the security implications of NAT-enabled and GRETAP open tunnelling hosts. Through our NAT attack, we demonstrated that NAT-enabled tunnelling hosts can expose a private network to remote fingerprinting and traffic injection. Furthermore, we identified that hosts that translate *global* IP addresses can be exploited by attackers to function as two-way proxies, facilitating a novel attack vector. We also show that GRETAP hosts are also susceptible to abuse. Through our 5 attacks, we demonstrated how an attacker can leak the gateway's MAC address, alter the LAN configuration, and cause DoS attacks. Our findings are especially concerning, as our 9 Internet-wide scans identified that more than 430K hosts are potentially vulnerable.

Overall, our paper highlights that the attack surface and security impact of open tunnelling hosts have been underestimated. These hosts can facilitate more than DoS attacks, and we are the first to show that some can be turned into two-way proxies or can enable various attacks against co-located private networks. We hope our findings will motivate and facilitate better protection of tunnelling hosts.

## Acknowledgments

This research is partially funded by the Research Fund KU Leuven and the Cybersecurity Research Programme Flanders.

## Ethical Considerations

**Stakeholders.** There are two groups of stakeholders impacted by our research: 1) stakeholders affected by our research methodology (Internet-wide scans & attack experiments) and 2) stakeholders affected by the publication of our paper. In our research, we identify stakeholders affected by our methodology as primarily hosts and, by extension, organisations that have received our Internet-wide scan traffic, as explained in Section 6. These stakeholders include network administrators, Internet Service Providers (ISPs) and ASs, cloud providers, personnel responsible for publicly facing hosts and end-users. Stakeholders affected by the publication of our research include developers, vendors such as router OS developers and Linux kernel developers, network administrators, ASs and ISPs identified as susceptible to the attacks presented in the paper, and end-users.

**Research Impact.** We base our risk assessment on Menlo Reports' following 4 principles:

*Respect for Persons.* Our research and Internet-wide scans generally have the inherent disadvantage of collecting data from research subjects who are not directly consenting. Subsequently, we take measures to ensure that the traffic being sent minimises the following side effects: 1) Traffic can be misinterpreted as malicious, causing alarm and additional work-hours for network administrators, and 2) Our traffic is received by non-consenting recipients. 1) To reduce the chances of our traffic being misinterpreted as malicious, all encapsulated packets have benign payloads, such as ICMP headers, and no packets do port scans. This reduces the chances that our traffic is misclassified as malicious or induces firewall warnings. 2) To ensure network administrators and organisations have a way to opt out, we host an information page on our scanning instance, listing details about our scans, including our contact information, allowing administrators to email us to be added to the blocklist, thereby excluding them from subsequent scans. Furthermore, to ensure that no traffic reaches organisations that may have blocklisted our address, we conducted our scans from a single IP address, thereby allowing system administrators to block that IP address. Furthermore, ZMap uses a standard IP ID of 54321, enabling system administrators to identify and block our scans.

*Beneficence.* During scanning, we make sure to limit the impact of our traffic, causing the following side effects: 1) Our traffic causes harm, such as Internet outages to networks or end-users, and 2) Our probes lead to private information leakage. 1) We ensure that our Internet-wide scans do not cause harm by limiting the scanning rate to 10 Mbps. This means that a single host will receive our traffic approximately once every four days. The traffic being sent does not include any malicious payloads. Furthermore, ZMap randomises the addresses to which our traffic is sent, reducing the likelihood that a specific ISP or AS will receive consecutive packets, which could cause problems [14]. 2) To ensure that hosts do not leak private information such as MAC addresses, we did not conduct a scan based on the methodology explained in Section 5.1. Our methodology for identifying potentially susceptible hosts relies on identifying hosts that may forward broadcast packets. All attack experiments were tested in a controlled environment with hosts under our control, meaning no actual tunnelling hosts received intrusive or potentially harmful traffic. Furthermore, all DoS attacks were tested only on VMs running on our own machines.

*Justice.* During disclosure, we take measures to maximise the number of affected hosts that have the information required to mitigate the issue, and to provide them with a testing tool to evaluate whether any of their hosts may be vulnerable. We disclose our findings to CERT/CC so that possibly affected vendors can mitigate the issue before publication. Furthermore, we collaborated with the Shadowserver Foundation to identify and notify affected organisations through their scans. Finally, given the sheer number of affected hosts, it is non-trivial to notify them individually; therefore, we implement

the aforementioned measures and notify the most affected parties directly to mitigate potential negative impacts.

**Publication Impact.** We identify the main possible harms from the publication of this paper: 1) Attackers conduct Internet-wide scans, identify vulnerable hosts and attempt to abuse them. 1) Although we acknowledge that it is infeasible for all affected parties to promptly mitigate the issue, we begin the disclosure procedure in advance to ensure affected organisations have enough time to mitigate the issue. Furthermore, we will not publicly release the IP addresses identified in our scans to reduce the potential harm of them being targeted.

**Decision & Justification.** Although our research methodology and the publication of this paper are subject to harm, we have taken rigorous measures to mitigate these risks to the best of our ability. More specifically, through our scans, we identified and notified potentially vulnerable hosts that would otherwise remain vulnerable and misconfigured. We argue that, through our scanning, we have provided more benefits to hosts identified as vulnerable than drawbacks. Furthermore, although publishing the paper could introduce the previously discussed harms, it will also raise awareness, ultimately leading to broader adoption of the defences described in Section 8. Furthermore, we introduce a range of solutions, from trivial, immediate defences to more fundamental ones, that will reduce misconfigurations over the long term. Overall, we believe the ethical benefits of openly releasing our results outweigh the harms.

**Internet Citizenship.** We place our scanning methodology in the context of Durumeric et al. six scanning best practices for measurement studies [13]:

*Minimise Internet Impact*. To reduce Internet impact, we conducted only novel scans not previously performed by other researchers; we only ran 1 scan of each type, and we set the scanning rate to 10 Mbps to ensure that hosts are not overwhelmed by our traffic. We follow the advice on ZMap's GitHub page [44], which recommends conducting scans at speeds under 1 Gbps. Although no specific recommendation is given for Internet-wide scans, we keep our scans at a very low rate to ensure no accidental DoS is caused (to remote networks or our provider's network).

*Signal Intent.* Since the IP address conducting the scan belongs to our cloud provider, there exists an rDNS record linking to a unique domain of our instance, where at port 80, we host an extensive information page listing scan dates, scan types, scanning address IP, protocol numbers to expect for received packets (protocol field on IP header, e.g., protocol 47 for GRE), information about our research and finally explains that by sending an email to the listed email addresses we can expand our blocklist so that those parties will no longer receive our packets. Finally, the IP WHOIS record and the top-level domain of our instance are registered with our cloud provider, which has been notified of our scans and allows them for research purposes (we made it clear to them that we do not perform port scans). If our cloud provider receives an abuse report, it will be forwarded to us so we can take appropriate measures. During scanning, we received an email requesting more information about the scans and expressing interest in the research conducted (without asking to be added to a blocklist). Although we did not explicitly set up the rDNS and WHOIS records ourselves, we believe that our goal of signalling intent has been sufficiently achieved.

To reduce the likelihood that our scans will be continuously performed by third parties, we present our findings in this paper. Our further analysis of data collected through Censys, including OSs, domains, open ports, and ASs, aims to provide a clear picture of the identified hosts without prompting others to rescan them.

*Provide An Opt-Out Mechanism.* On our static page, we list our email addresses and explain how admins can send us their IP address ranges, which we add to our blocklist so they are excluded from all future scans.

*Proactively Investigate Effects.* Before conducting Internet-wide scans, we always first configure a vulnerable host under our control and test our scans against it. We then scan that singular host under our control and ensure that 1) the sent probe is correctly constructed and 2) the reply is correctly catalogued as vulnerable. Furthermore, before conducting scans, we first scan 1% of the Internet. If the 1% scan returns results, we continue the scan until it is complete; otherwise, we stop the scan.

*Coordinate Locally.* As mentioned, our cloud provider has been notified in advance and will forward any inquiries and complaints to us.

*Disclose Results.* We first contacted major organisations identified in our scans individually and contacted CERT/CC to further reduce the vulnerability/misconfiguration. We are also working with ShadowServer, which is helping by notifying more vulnerable hosts. We also emailed companies/organisations identified in Section 7 (domains and case study organisations).

## Open Science

We provide our code at the following URL: `https://doi.org/10.5281/zenodo.20401534`, along with a `README.md` file that provides instructions for running the experiments and `.sh` scripts that set up the appropriate interfaces needed to run the experiments. We provide the following: 1) The attack against NAT-enabled tunnelling hosts presented in Section 4 split into two scripts (one for fingerprinting and one for traffic injection), 2) the 2 Two-way proxy attacks presented in Section 4.2, 3) the 5 attacks against GRETAP hosts in Section 5, and finally, 4) The 6 scanning probe modules used to

conduct the 9 full Internet-wide scans and the additional 1% GRETAPv6 scan described in Section 6. Our ZMap extension, along with our 6 scanning modules, is also provided at the following URL: https://github.com/AngelosBeitis/zmapv6_tunnelling_protocols.

We do not provide the identified vulnerable IP addresses to reduce exposure.

## References


[1] Shubair A Abdulla. Survey of security issues in IPv4 to IPv6 tunnel transition mechanisms. *International Journal of Security and Networks*, 12(2):83–102, 2017.

[2] Salah A. Jaro Alabady. Design and implementation of a network security model using static vlan and aaa server. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6, 2008.

[3] ipv4: send arp replies to the correct tunnel. https://github.com/torvalds/linux/commit/63d008a4e9ee86614ca5671b7f3ba447df007190. Accessed on May 20, 2026.

[4] Zeeshan Ashraf, Adnan Sohail, Sohaib A Latif, Abdul Hameed Pitafi, and Muhammad Yousaf Malik. Challenges and mitigation strategies for transition from IPv4 network to virtualized next-generation IPv6 network. *Int. Arab J. Inf. Technol.*, 20(1):78–91, 2023.

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

[6] Angelos Beitis and Mathy Vanhoef. Haunted by Legacy: Discovering and Exploiting Vulnerable Tunnelling Hosts. In *34th USENIX Security Symposium (USENIX Security 25)*, Seattle, WA, 2025. USENIX Association.

[7] Lorenzo Colitti, Giuseppe Di Battista, and Maurizio Patrignani. Discovering IPv6-in-IPv4 tunnels in the internet. In *IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2004.

[8] Dahua Security. https://www.dahuasecurity.com/ph/products. Accessed on May 21, 2026.

[9] Reflection attack wr11,wr21,wr31,wr41,wr44 series routers - vu#636397 - cve-2020-10136. https://www.digi.com/resources/security. Accessed on May 21, 2026.

[10] Ralph Droms. Dynamic Host Configuration Protocol. RFC 2131, March 1997.

[11] Dropbear SSH. https://matt.ucc.asn.au/dropbear/dropbear.html. Accessed on May 21, 2026.

[12] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. A search engine backed by Internet-wide scanning. In *22nd ACM Conference on Computer and Communications Security*, October 2015.

[13] Zakir Durumeric, David Adrian, Phillip Stephens, Eric Wustrow, and J. Alex Halderman. Ten years of ZMap. In *Proceedings of the 2024 ACM on Internet Measurement Conference*, IMC '24, page 139–148, New York, NY, USA, 2024. Association for Computing Machinery.

[14] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. ZMap: fast internet-wide scanning and its security applications. In *Proceedings of the 22nd USENIX Conference on Security*, SEC'13, page 605–620, USA, 2013. USENIX Association.

[15] Wesley Eddy. Transmission Control Protocol (TCP). RFC 9293, August 2022.

[16] Kjeld Borch Egevang and Paul Francis. The IP Network Address Translator (NAT). RFC 1631, May 1994.

[17] Embedthis. https://www.embedthis.com/. Accessed on May 21, 2026.

[18] Dino Farinacci, Tony Li, Stanley P. Hanks, David Meyer, and Paul S. Traina. Generic Routing Encapsulation (GRE). RFC 2784, March 2000.

[19] 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 *Proceedings of the 32nd Annual Network and Distributed System Security Symposium (NDSS 2025)*, Feb 2025.

[20] Bryan Ford, Dan Kegel, and Pyda Srisuresh. State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs). RFC 5128, March 2008.

[21] Sheila Frankel and Suresh Krishnan. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, February 2011.

[22] Yossi Gilad and Amir Herzberg. Fragmentation considered vulnerable. *ACM Trans. Inf. Syst. Secur.*, 15(4), April 2013.

[23] ip_gre: Make none-tun-dst gre tunnel store tunnel info as metadat_dst in recv. https://github.com/torvalds/linux/commit/c0d59da79534e85eb550d863e35eccc8c3fd8ceb. Accessed on January 31, 2026.

[24] Jesse Gross, Ilango Ganga, and T. Sridhar. Geneve: Generic Network Virtualization Encapsulation. RFC 8926, November 2020.

[25] Kejun Gu, Liancheng Zhang, Zhenxing Wang, and Yazhou Kong. Comparative studies of IPv6 tunnel security. In *ICNC-FSKD*. IEEE, 2017.

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

[27] IP-in-IP protocol routes arbitrary traffic by default. https://kb.cert.org/vuls/id/636397. Accessed on May 21, 2026.

[28] ip-link(8) — linux manual page. https://man7.org/linux/man-pages/man8/ip-link.8.html, December 2012. Accessed on February 4, 2026.

[29] ip_sysctl. https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt. Accessed on October 13, 2025.

[30] IP to ASN. https://iptoasn.com/, 2024. Accessed on February 6, 2024.

[31] ip-tunnel(8) linux user's manual. https://man7.org/linux/man-pages/man8/ip-tunnel.8.html, December 2011. Accessed on February 4, 2026.

[32] Jaehoon Paul Jeong, Soohong Daniel Park, Luc Beloeil, and Syam Madanapalli. IPv6 Router Advertisement Options for DNS Configuration. RFC 8106, March 2017.

[33] John Kristoff, Mohammad Ghasemisharif, Chris Kanich, and Jason Polakis. Plight at the end of the tunnel: Legacy IPv6 transition mechanisms in the wild. In *PAM 2021*, Berlin, Heidelberg, 2021. Springer-Verlag.

[34] Yiu Lee and Adam Uzelac. Voice over IP (VoIP) SIP Peering Use Cases. RFC 6405, November 2011.

[35] Mallik Mahalingam, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Larry Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348, August 2014.

[36] Milesight. https://www.milesight.com/. Accessed on May 21, 2026.

[37] Benjamin Mixon-Baca, Jeffrey Knockel, Diwen Xue, Tarun Ayyagari, Deepak Kapur, Roya Ensafi, and Jedidiah R. Crandall. Attacking connection tracking frameworks as used by virtual private networks. *Proceedings on Privacy Enhancing Technologies*, 2024(3):109–126, July 2024.

[38] Jeffrey Mogul. Broadcasting Internet Datagrams. RFC 919, October 1984.

[39] MQTT Version 5.0. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. Accessed on May 21, 2026.

[40] NexG VForce Products & Services. https://www.nexg.net/eng/product/prdt0105. Accessed on May 21, 2026.

[41] Charles E. Perkins. IP Encapsulation within IP. RFC 2003, October 1996.

[42] Zhiyun Qian and Z. Morley Mao. Off-path TCP Sequence Number Inference Attack - How Firewall Middleboxes Reduce Security. In *2012 IEEE Symposium on Security and Privacy*, pages 347–361, 2012.

[43] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark J. Handley, and Eve Schooler. SIP: Session Initiation Protocol. RFC 3261, June 2002.

[44] Phillip Stephens. ZMap: Getting started guide. https://github.com/zmap/zmap/wiki/Getting-Started-Guide. Accessed on May 21, 2026.

[45] The Shadowserver Foundation. Shadowserver Dashboard: Time series (general statistics) — ip tunnel and ip tunnel6 (unique ips). https://dashboard.shadowserver.org/statistics/combined/time-series/?date_range=365&source=ip_tunnel&source=ip_tunnel6&dataset=unique_ips&stacking=stacked, 2026. Accessed on February 6, 2026.

[46] William J. Tolley, Beau Kujath, Mohammad Taha Khan, Narseo Vallina-Rodriguez, and Jedidiah R. Crandall. Blind In/On-Path Attacks and Applications to VPNs. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3129–3146. USENIX Association, 2021.

[47] Andrew J. Valencia, Glen Zorn, William Palter, Gurdeep-Singh Pall, Mark Townsley, and Allan Rubens. Layer Two Tunneling Protocol "L2TP". RFC 2661, August 1999.

[48] Yuxiang Yang, Xuewei Feng, Qi Li, Kun Sun, Ziqiang Wang, Ao Wang, and Ke Xu. Off-Path TCP Hijacking Attack to NAT-Enabled Wi-Fi Networks. *IEEE Transactions on Networking*, 33(6):3270–3285, 2025.

[49] Yuxiang Yang, Xuewei Feng, Qi Li, Kun Sun, Ziqiang Wang, and Ke Xu. Exploiting Sequence Number Leakage: TCP Hijacking in NAT-Enabled Wi-Fi Networks. In *Proceedings of the 31st Network and Distributed System Security (NDSS) Symposium*. Internet Society, 2024.