Bypassing Tunnels: Leaking VPN Client Traffic by Abusing Routing Tables

Nian Xue, *New York University;* Yashaswi Malla, Zihang Xia, and Christina Pöpper, *New York University Abu Dhabi;* Mathy Vanhoef, *imec-DistriNet, KU Leuven*

https://www.usenix.org/conference/usenixsecurity23/presentation/xue

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

USENIX'23 Artifact Appendix: Bypassing Tunnels: Leaking VPN Client Traffic by Abusing Routing Tables

Nian Xue
New York University

Yashaswi Malla, Zihang Xia, Christina Pöpper
New York University Abu Dhabi

Mathy Vanhoef
imec-DistriNet, KU Leuven

A Artifact Appendix

A.1 Abstract

In this artifact, we present the instructions for conducting two novel attacks (i.e., called LocalNet Attacks and ServerIP Attacks) that are described in our paper, causing VPN clients to leak traffic outside the protected VPN tunnel. In our paper, we claim that the traffic to the local network and to the VPN server itself can be manipulated by abusing routing tables such that it will be sent in plaintext outside the VPN tunnel. Through extensive experimentation with various VPN clients, we have identified that these attacks pose a general vulnerability across multiple OSs.

A.2 Description & Requirements

Our attacks manipulate the client's routing table such that traffic will be sent outside the VPN tunnel, i.e., without encryption. Normally, when the VPN is not enabled, a client's routing table might look like the following:

```
[tester@zbook ~]$ ip route
default via 192.168.1.1 dev wlp0s20f3
192.168.1.0/24 dev wlp0s20f3 scope link
```

The IP address of the client in this example is 192.168.1.101. The two output lines mean:

- The first line says that by *default* all outgoing IP packets are forwarded via 192.168.1.1. Here 192.168.1.1 is the router. The rule also specifies "dev wlp0s20f3", meaning that the packets are sent over the wlp0s20f3 Wi-Fi network card. All combined, all outgoing IP packets are, by default, sent to the router using the Wi-Fi network card.
- The second line is an exception [2] to the above rule: all IP packets to 192.168.1.0/24, so to IP addresses between 192.168.1.0 and 192.168.1.255, are sent over "dev wlp0s20f3", specifically over the Wi-Fi network card. Moreover, "scope link" means these IP addresses are directly reachable: the packets can directly be sent to their destination instead of first being forwarded to the router.

When a VPN is enabled, a client's routing table might look like this:

```
[tester@zbook ~]$ ip route
default via 10.8.0.1 dev tun0
76.26.140.111 via 192.168.1.1 dev wlp0s20f3
192.168.1.0/24 dev wlp0s20f3 scope link
```

Here, the IP address of the VPN server is 76.26.140.111. The first rule says that by default, all outgoing IP packets are sent over "dev tun0". Here tun0 is a virtual network card representing the encrypted VPN tunnel. In other words, by default, all packets are sent through the VPN tunnel. There are two exceptions:

1. The second rule says that packets with as destination the VPN server must be sent to the router using the Wi-Fi network card. This exception avoids a rooting loop where already-encrypted VPN packets would otherwise get encrypted again.
2. The third rule is the same as when the VPN wasn't enabled: all packets to the local network (notice the "scope link") are directly transmitted over the Wi-Fi network card to the destination (so not through the VPN tunnel). This assures that local devices in the network, such as printers and file servers, remain accessible when using the VPN.

A.2.1 Security, privacy, and ethical concerns

During the experiment, two types of traffic (see two claims) may be sent outside the VPN tunnel. It is best not to enter sensitive information while doing the experiments.

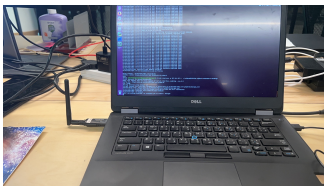
A.2.2 How to access

We have compiled a GitHub repository that provides a readme and additional necessary files.

A.2.3 Hardware dependencies

Table 1 displays the list of required soft- and hardware equipment. In terms of hardware, to conduct the experiment, a malicious AP (Access Point) is necessary to create an AP on any channel. Often the built-in Wi-Fi network card of a laptop can be used. Alternatively, an an external wireless USB adapter can be used, such as the Panda Wireless PAU06 300Mbps Wireless N USB (see Figure 1). The test platform we used is shown in Figure 1.

(a) A Panda PAU06 300Mbps Wireless USB Adapter.



(b) A laptop with an external wireless USB Adapter.

Figure 1: Our example test platform running Ubuntu.

Table 1: Hardware and Software requirements to conduct the experiment.

Class	Item name
hardware	Laptops or cellphones, e.g., iPhones or Androids
hardware	Wireless network card (built-in or USB dongle)
software	Wireshark
software	`create_ap` script
software	VPN clients

A.2.4 Software dependencies

The `create_ap` script to start and configure the AP is required [3]. Wireshark can be used to check leaks outside the VPN tunnel by inspecting traffic. Sections A.4.2 and A.4.3 provide detailed instructions on which commands to run.

The targeted commercial VPN clients are available on Apple App Store and Google Play Store, or on the vendor's website, and can be directly downloaded from these stores. The paid VPN apps require a subscription.

A.2.5 Benchmarks

None.

A.3 Set-up

This section includes all the installation and configuration steps required to prepare the environment to be used for the evaluation of the attacks.

A.3.1 Installation

We used the `create_ap` tool to create a Wi-Fi network for the tests. The generic installation instructions are available here: create_ap. On certain Linux distributions, it can be installed using the package manager. On Ubuntu, it requires to install the following dependencies:

```
sudo apt install hostapd wireshark
```

A standard AP can be created using the command:

```
sudo create_ap wlan1 wlan0 testnetwork abcdefgh
```

Figure 2: An example of creating an AP.

Figure 2 shows an example of how to create a Wi-Fi network called *testnetwork* with password *abcdefgh*. The arguments `wlan1` and `wlan0` depend on the machine used for the test:

- The argument `wlan0` refers to the built-in network card and may be different depending on the machine. Find out this name by executing `ip addr` and picking the interface that is assigned an IP address.
- The argument `wlan1` refers to the Wi-Fi dongle/wireless USB adapter plugged in. Find out its name on the machine by executing `ip addr` before and after plugging in the Wi-Fi dongle and seeing which interface was added.

One should now be able to connect to the created Wi-Fi network. To inspect the traffic of any client connect to the AP start Wireshark and listen for packets on the 'ap0' interface (or on the interface of the Wi-Fi dongle in case it does not support virtual interfaces).

Errors and warnings:

- If the error "ERROR: Failed to initialize lock" occurs, then execute: `sudo rm /tmp/create_ap.all.lock`
- The warning "Your adapter does not fully support AP virtual interface" means the Wi-Fi dongle cannot simultaneously act as a client and AP. If creating the Wi-Fi network fails, then try a different USB dongle.

A.3.2 Basic Test

Install the VPN app and connect to the Wi-Fi hotspot. Open Wireshark on the platform to monitor packets. Then enable the VPN and perform the following tests.

A.4 Evaluation Workflow

This section includes all the operational steps and experiments which must be performed to evaluate our attacks.

A.4.1 Major Claims

(C1 – LocalNet): *Traffic to local IP addresses is not sent through the VPN tunnel.*
(C2 – ServerIP): *Traffic sent to the IP address of the VPN server is not (again) sent through the VPN tunnel.*

A.4.2 Testing LocalNet Attacks

A quick method to test for this vulnerability is to make the router hand out non-RFC1918 IP addresses for the local network, e.g., using 207.241.237.0/24 for the local network. Then enable the VPN and try to visit visit `http://207.241.237.3/`. In Wireshark, this should result in ARP requests for the IP address 207.241.237.3, indicating that the tested VPN is vulnerable to the LocalNet traffic leak attack.

Alternatively, start the `create_ap` script to hand out public IP addresses. For example, if we want to intercept traffic to `web.archive.org`, which has IP address 207.241.237.3 at the time of writing, the hotspot has to hand out IP addresses from a subnet that contains this IP address. This can be done by starting `create_ap` as follows:

```
sudo create_ap wlan1 wlan0 testnetwork abcdefgh
 -g 207.241.237.3
```

Now connect with the created AP and enable the VPN client. Open Wireshark. Then try to visit `http://207.241.237.3` in a browser. If TCP SYN packets can be seen to 207.241.237.3, this means that the VPN app is vulnerable: by using the Wireshark filter `tcp.flags.syn == 1`, it is easy to filter for plaintext TCP SYN packets. One successful example is shown in Figure 3.

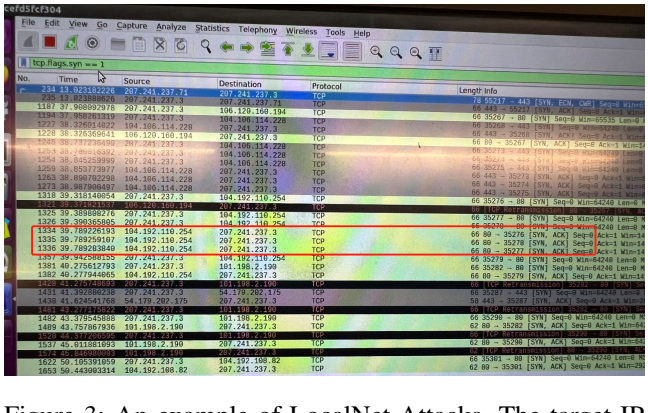


Figure 3: An example of LocalNet Attacks. The target IP address is 207.241.237.3.

A.4.3 Testing ServerIP Attacks

Start the `create_ap` script and then connect with the device being tested:

```
sudo create_ap wlan1 wlan0 testnetwork abcdefgh
```

Now start capturing frames on the test platform. After starting to capture frames, connect to the VPN server, and then identify the IP address of the VPN server based on the transmitted traffic in Wireshark. Then visit "http://$VPN_SERVERIP". If there are no plaintext TCP SYNs in Wireshark, then the VPN client is not vulnerable (we can use the Wireshark filter `tcp.flags.syn == 1` to filter for plaintext TCP SYN packets). If the VPN protocol is using TCP or UDP then you can also try to visit "http://$VPN_SERVERIP:$PORT" where we add the port that is also used by the VPN server. One successful attack is shown in Fig. 4.

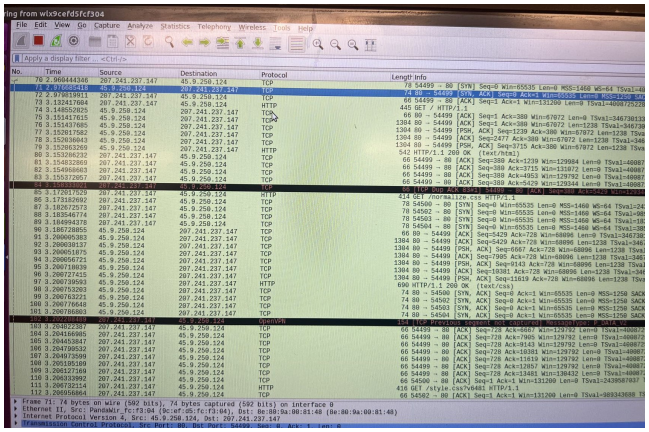


Figure 4: An example of ServerIP Attacks. The IP address of the VPN server is 45.9.250.124.

In case there are plaintext TCP SYN packets, the next step is to test whether the VPN client used plaintext DNS to find the VPN server's IP address. To determine this, we can use the Wireshark filter '`dns.a == $VPN_SERVERIP`'. If there are any results, then the VPN client is highly likely vulnerable.

A.5 Stable URL

We provide a stable URL where the community can find the final copy of our artifact in order to achieve replicability of the experiments [1].

References

[1] `https://github.com/vanhoefm/vpnleaks`, 2023.

[2] Martin A. Brown. Guide to IP Layer Network Administration with Linux. `http://linux-ip.net/html/routing-selection.html`, 2013. Accessed June 12, 2023.

[3] Yiannis M. create_ap. `https://github.com/oblique/create_ap`, 2013. Accessed September 12, 2022.