December 23, 2025

SSH TUNEL[1]

Abusing OpenSSH, TUN/TAP devices
and other shenanigans

What's reality? I don't know. When my bird was looking at my computer monitor I thought, "That bird has no idea what he's looking at."

Terry A. Davis — Wikiquote

Contents

[1] Yes that is misspelt correctly.

First a Warning. These can be used for good and for bad, so be careful on whose network you do this on. I could be interpreted in a bad light. Because of its encrypted nature, it becomes difficult to detect a malicious connection purely by the connection data.

This has been used, specifically reverse SSH tunneling, by known threat actors, including state sponsored threat actors.[2]

1 What

In this write-up I will decribe how to do some more advanced network tricks on linux. Namely 1) how to connect two linux computers directly over Ethernet, and 2) how to tunnel over SSH using two TUN/TAP interfaces.

There are easier ways to accomplish a SSH tunnel for the purposes of proxying data from one computer to another.[3] However this write-up does not cover those.

1.1 What an SSH tunnel?

A SSH (Secure Shell[4]) tunnel is pretty much exactly as it sounds. It is a tunnel, like all those infografics and advertisments like to tell you. It, like a vpn, is "encrypted", but operates differently, mainly being that of their uses. While VPNs are used for full packet encryption of all outgoing/incomming connections, SSH is usd to securely access remote devices to execute commands.

So what SSH tunneling is, is using SSH as if it was a VPN. This allows for more flexability when it comes to directing these tunnels, and what applications are put through it.

[2]Here are a few examples:

The Hacker News	—	site	archive	*APT CL-STA-0969 (Liminal Panda)*
GBHackers News	—	site	archive	*FIN7 (Savage Ladybug)*
Department of War	—	site	archive	*APT28 (Fancy Bear)*

[3]You can use SSH to create tunnels besides the one discribed in this write-up. You can find information on this pretty easy, but here are some sources.

robotmoon	—	site	archive	*A visual guide to SSH tunnels*
ssh.com	—	site	archive	*Explains what SSH tunneling is*
linuxmind.dev	—	site	archive	*Explains how to tunnel using SSH*

[4]If you do not know what the Secure Shell is, then here are some sources to read about it.

Cloudflare	—	site	archive	*What is SSH?*
wikipedia	—	site	archive	*Secure Shell*
Adarsh GS (Medium)	—	site		*Beginner's Guide to SSH*
IETF	—	site	archive	*RFC 4251 – The SSH Protocol*

It used to be that SSH tunnelling used to mean using TUN/TAP interfaces over SSH, but that has since changed. So for a brief moment, SSH tunneling and VPNs had one more commonality, they both used TUN/TAP interfaces.[5]

1.2 What is a TUN/TAP interfaces?

TUN/TAP interfaces are software networking devices used for networking tunneling. The combined name, *TUN/TAP*, is in refrence to the pair of interfaces used for tunnelling, both being able to, for the most part, be interchanged with the other. The one difference between these two are on what level of the OSI model the operate at.

TUN devices (can be thought of as a network TUNnel) operate at Layer 3, or the Networking Layer, of the OSI Model. This means it used and carries IP packets.

TAP devices (can be thought of as a network TAP) operate at Layer 2, or the Data Link Layer, of the OSI Model. This means that it carries Ethernet frames, instead of IP packets.[6]

1.3 How do these fit together?

These two things, SSH and TUN devices (the device used in this write-up), do not seem particularly related, and in some sense they are not. But when they are put together, with some other tricks, one can allow a guest computer, that is connected to a host computer, but not to the internet, the ability to access another network that the host computer is connected to.

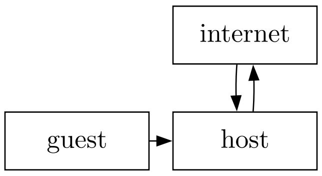

Figure 1: Visualization

[5]Not all VPNs may use TUN/TAP interfaces, for example when using a dedicated VPN device/concentrator, or if a vpn uses IPsec (it may or may not with ipsec).

robotmoon	—	site	archive	*A visual guide to SSH tunnels*
comparitech	—	site		*What are TUN/TAP Adapters*
The Linux Channel	—	site	archive	*VPN Stack Architecture*

[6]Further reading on TUN/TAP devices:

wikipedia	—	site	archive	*TUN/TAP wiki article*
comparitech	—	site		*What are TUN/TAP Adapters*

This can be done using SSH, specifically using OpenSSH[7] for now, with the *-w* option. This argument takes two numbers, or any (for the next available tun device), which specify the tun device ID to use (for example tun5 would be 5).

2 Why

You may be wondering why go through these lengths to do this. There are a couple of reasons. For example 1) giving internet access to a restricted, normally air gaped computer briefly for updates (for those who dont have enterprise grade offline update solutions), 2) to login to your parents Netflix account, 3) using a secondary computer as a gateway device to reduce your attack surfice, and more.

3 How

Generally this can be done in a couple of steps; 1) Creating a connection to the guest device, 2) creating the TUN devices, 3) establish the tunnel, and then 4) configure both the guest and host network.

Also note, that most of these commands require root/sudo access to run.

Here is some further reading, and potential help:

SSH-TUN				
Archlinux Wiki	—	site	archive	*Arch Linux – VPN over SSH*
Unix Stackexchange	—	site	archive	*Network Interface from SSH Tunnel*
Marc Fargas	—	site	archive	*IP tunnel over SSH with TUN*
Jeffry Walsh	—	site	archive	*Tunneling under SSH using TUN*
Johnnie J. Alan	—	site	archive	*TUN interfaces advance use*
TUN/TAP				
Hachao’s Blog	—	site	archive	*TUN/TAP Interfaces What/Using*
Linux Command Library	—	site		*ip-tuntap commands*
The Linux Kernel	—	site	archive	*Linux Kernel Doc – TUN/TAP*
Direct C2C Networking				
Superuser	—	site	archive	*Directly connecting computers*

3.1 P2P Ethernet

For this project, I decided to connect two computers together over ethernet to accomplish the desired isolation of the Guest Computer (since I did not have a spare router or two ethernet ports). This is a sort of trick within itself, so

[7]There are multiple implementations of the SSH client and server. The most popular Client/Sever SSH implementation is OpenSSH.

OpenSSH	—	site	archive	*OpenSSH Server/Client*
Matt	—	site	archive	*DropBear SSH Server/Client*
wikipedia	—	site	archive	*Comparison of SSH Servers*
wikipedia	—	site	archive	*Comparison of SSH Servers*

I have deticated a section to this. Normally you would just ensure that both computers have an IP address and can talk to each other for the *“conniction”* phase.

You first need to connect the computers to each other, in this case, via an ethernet cable (note: a crossover cable is no longer needed for this kind of thing). After which you will need to assign an IP to both the Host and Guest ethernet interfaces.

This can be done using the following commands:

```
# Can be any address.
# Make sure to change the 'eth0' to your ethernet cards id,
# which you can find using the 'ip a' command.
ip address add 192.168.75.1/24 dev eth0

# You can delete this address as follows.
#  ip address del 192.168.75.1/24 dev eth0
```

Once done for each device you should test the connection, you can use ping on the ip address assigned, or even use SSH, which is neccessary for this.

3.2 Create TUN devices

Finding information on how to create a TUN device is supprisingly difficult, since it dives into the programming side, which is not difficult, but teadious. Once you know what you are looking for, it becomes much easier, however.

First thing, make sure you have 'iproute2'[8] installed. Some distributions, like Alpine Linux, uses minimal 'ip' utilities.[9]

Next you just have to use the following commands:

```
# Create the device 'tun5' as a TUN device.
# The name 'tun5' is only a convention.
# To allow a user to access and write to this device
#  add the 'user <user>' at the end.
ip tuntap add dev tun5 mode tun user <user>

# To delete just change add to del
#  ip tuntap del dev tun5 mode tun
```

[8]You also may need to run 'modprobe tun' if your setup does not have the iproute2 package. This enables the TUN module for linux, but not permanently. To do this, most commonly, you just need to add 'tun' to the '/etc/modules' or in a new file in '/etc/modprobe.d'.

[9]This problem caused me a good amount of headache. It is also possible to create a TUN/TAP device without the 'ip' utility.3

Do this for each device, making sure to note the name of the newly created device (note: OpenSSH can create these interfaces if the user has the correct permissions).

Next you will need to give your interfaces IPs and put them in the UP state, this can be done with the following two commands:

```
# Same as the one above for the Ethernet Connection.
# Change the IP, CIDR and interface as needed.
ip address add 10.0.0.1/24 dev tun5

# Replace 'tun5' with your tun interface.
ip link set tun5 up
```

3.3 Create the tunnel

The next part is to create the tunnel. All you need to do is create an SSH session using a specific flag and arguments. You can view the manpage for this option for more information. The *-w* option, for OpenSSH, allows you to accomplish the creation of a tunnel that forwards the TUN device of your choosing to another.

The command looks similar to this, if the guest is connecting to the host, and local tun device is 'tun2' and remote is 'tun1':

```
# Can use the argument any for either of the numbers
#  to specify the use of the first TUN interface.
ssh user@192.168.75.1 -w 2:1
```

You now have the tunnel created, but not everything is done yet.

3.4 Configuring the network

Next, you will need to configure the network to use the TUN interface, and for the host computer to forward IPv4 traffic.

At this point, you should also check if your endpoint firewall will block this sort of thing, if you have deployed one.

To use the Guest Computer to use the TUN device as the default interface you can use the following command:

```
# The 10.0.0.1 is the IP of the gateway,
#  or the host computers ip for the tun interface
ip route replace default via 10.0.0.1 dev tun5
```

Next you will have to enable the IPv4 forwarding on the host. This has the host act as a router, and forward the IPv4 packets it receives on to the rest of the network. Without this, you can only access the host. This can be done a couple of ways, but generally, you just need to change the '/proc/sys/net/ipv4/ip_forward' file from '0' to '1'. That can be done as follows:[10]

```
echo 1 > /proc/sys/net/ipv4/ip\_forward
```

You should be able to now access the internet from your guest computer. If it is not working, check your router settings, since it is weird to have a computer send data by another IP than its own. You could also, optionally, set up NAT (Masquerading) on the host.

3.4.1 (Optional) Nat Masquerading on Host

This has the host use Network Address Translation (NAT) to masquerade the traffic from the guest computer as comming from the host computer. This can be done by changing the iptable. Using 'iptables' you can use the following command:/footnote If you use nftables, then the command will be different.

```
# Change the interface to the one connected to the internet.
# 10.0.0.0/24 specifies the network source this applies two,
#  in this case all 10.0.0.x ips.
iptables -t nat -A POSTROUTING -o <interface> -s 10.0.0.0/24 -j
    MASQUERADE
```

3.5 Teardown

To teardown on the guest, you will have to delete the TUN interfaces (shown next to the creation commands), and change the default route. To change back the default rout you can use the following command:

```
ip route replace default via <ORIGINAL GATEWAY> dev <ORIGINAL INTERFACE>
```

On the host, you have to do the same thing and more. You will have to change the '/proc/sys/net/ipv4/ip_forward' file back to its original value and remove the iptables entry. To remove the iptables entry, all you have to do is run the command used to add it to the table, and change the *-A* to a *-D*.

You could also just reboot.

There are 3482 words in this document.

[10]Note: this is not permanent. You can generally make it so by adding the 'net.ipv4.ip_forward = 1' to '/etc/sysctl.conf':