# SSH Tunnels

OpenSSH allows you to create a connection with a remote host, and then run a program on the local host so that it acts as if it is running on the remote host. This is handy for penetration testers (and attackers.) Once the tester has taken over a machine inside the target organization’s firewall, they can use that machine to run scans and attacks inside the organization without worrying about the firewall. It is also handy for creating a homemade Virtual Private Network (VPN).

SSH tunnels come in three general flavors, local, dynamic, and remote. A local tunnel uses -L in the SSH command and allows one local port to be tunneled to a single remote host. For example, this command  
ssh -L 8000:www.somesite.com:80 svgs@centos  
takes any traffic received on local port 8000, puts it in an SSH tunnel to centos, and tells centos to forward the traffic to [www.somesite.com](http://www.somesite.com), port 80. You send the traffic you want to be tunneled to localhost:8000 or 127.0.0.1:8000. You can pick any port number for the local port (8000 in this case) if the port is not in use, and you remember to send your data to that port. The port number at the end (80 in this case) must be the port that [www.somesite.com](http://www.somesite.com) listens on. Unfortunately, web sites have become complicated enough that local tunnels no longer work well for web browsing.

Local SSH tunnels do work well for other protocols, however. Windows file sharing over Server Message Block (SMB) protocol works well through SSH tunnels. There is an exercise later in the course that makes use of this.

## Dynamic SSH tunnels

A dynamic SSH tunnel uses the SOCKSv5 Proxy protocol. SOCKSv5 allows the tunnel to send traffic to any host. It is not limited to a single host listed in the command line as in the SSH Local tunnel. It also has procedures for passing authentication traffic so that browsing works just as it would in a direct connection.

### Aside--web proxy servers

Wise organizations force all user web traffic to go through a proxy server, where attacks can be filtered, malware outbound Command and Control (CnC) traffic may be blocked, and all traffic can be logged. Nowadays, this function is often incorporated into the organization’s Next Generation Firewall (NGFW). In the past, the proxy was a separate appliance and the browser had to be told to send its traffic to the proxy server instead of the default gateway. Most browsers include settings that will allow us to direct the browser traffic to the proxy, or SSH tunnel in our case, instead of to the default gateway.

# A Homemade VPN—Scenario

You are at Starbucks, or worse yet, the Blackhat convention and you want to surf without worrying about Man in the Middle (MitM) attacks. If you have a trusted remote server that allows SSH and you already have the server’s public key on your laptop so that you know that you are connecting to your server and not some imposter, you can surf through an encrypted SSH tunnel. But where can you find an SSH server you trust, and where you already have its public key? Hmmm, remember the Cloud VM lab?

Diagram

Description automatically generated

The connection between your host machine and the SSH Server is encrypted. Both sides of the connection are authenticated with private/public keys. If the host machine has previously connected to the SSH server, it has a copy of the server’s public key so no attacker will be able to perform a Man in the Middle (MitM) attack against you. Also, the Internet will see any traffic you send through the tunnel as coming from the SSH server’s IP address. This provides you with some degree of anonymity. (If you run your SSH server on AWS, the FBI will be able to subpoena AWS for your information so do not use this for criminal activity.)

## Lab Setup

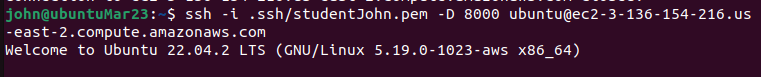
This lab requires a host machine (can be a hardware machine or a virtual machine) and an SSH server (can be a local VM or a cloud VM). The host machine must be able to connect to the SSH server via SSH, and the SSH server must have an Internet connection. This example will use an Ubuntu VM as the host machine, and an AWS VM as the SSH server. The host machine can also be Windows, as the commands are identical to Linux since both use OpenSSH clients.

### Create the tunnel

On the host machine, open an SSH connection to the SSH server using the dynamic tunnel option  
-D [port number]. The port number can be anything we choose if the port is not already in use. It is also good to choose a port number above 1024, since Linux requires root privileges to use ports below 1024.

Here’s the SSH connection from Windows, using local port 8000 (our choice, can be anything) and using an AWS instance as the SSH server. The only thing different from our previous connections is the   
-D 8000  
ssh -i .\.ssh\studentJohn.pem -D 8000 ubuntu@ec2-3-136-154-216.us-east-2.compute.amazonaws.com  
Text

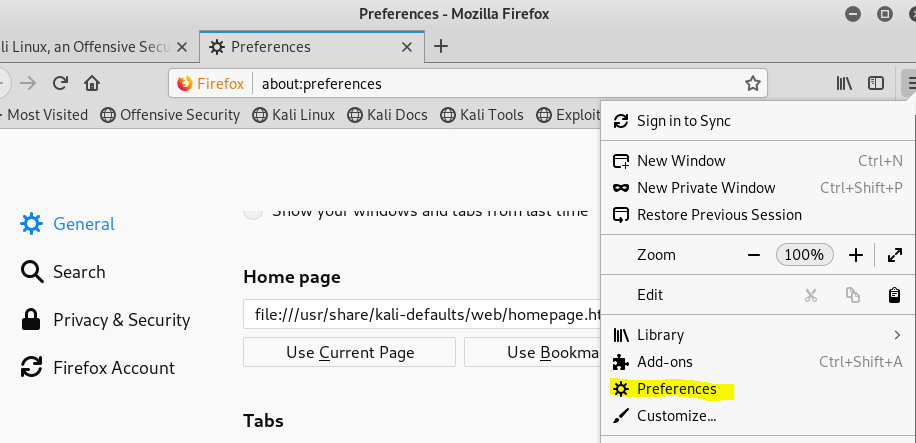
Description automatically generated

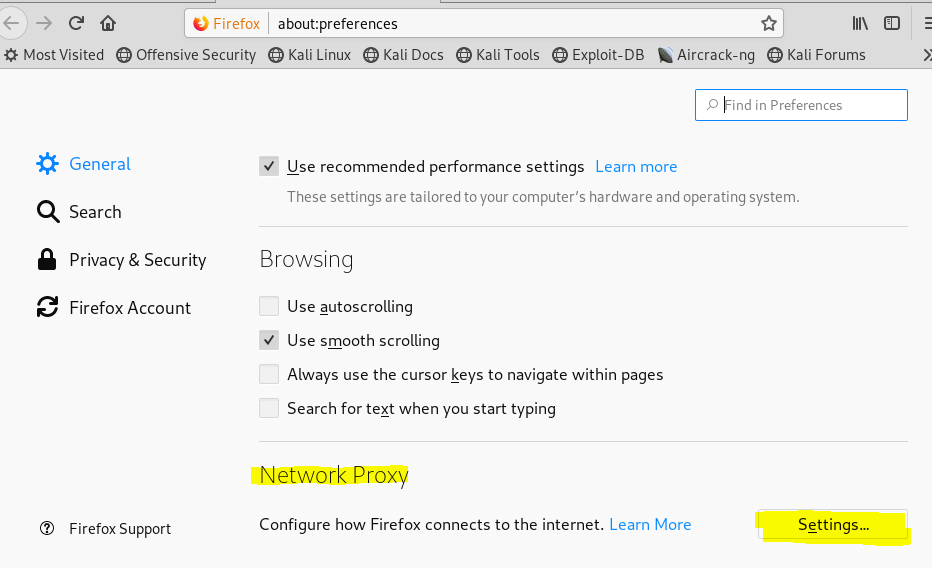
Here's the SSH connection from an Ubuntu VM on my local host.  


Leave the SSH connection open, so your browser can use it.

### Configure Proxy Settings on the Firefox browser

This works best in Firefox. I have had problems getting it to work in Chrome.

On Firefox in your host machine, select Preferences and then General.  


Scroll to the very bottom, and select Network Proxy Settings.  


Manually configure the proxy to use SOCKSv5. The address in the SOCKS Host field can be either localhost or 127.0.0.1. The port number should match whatever we picked when we created the tunnel.

Leave the HTTP Proxy and HTTPS Proxy fields blank.

Text

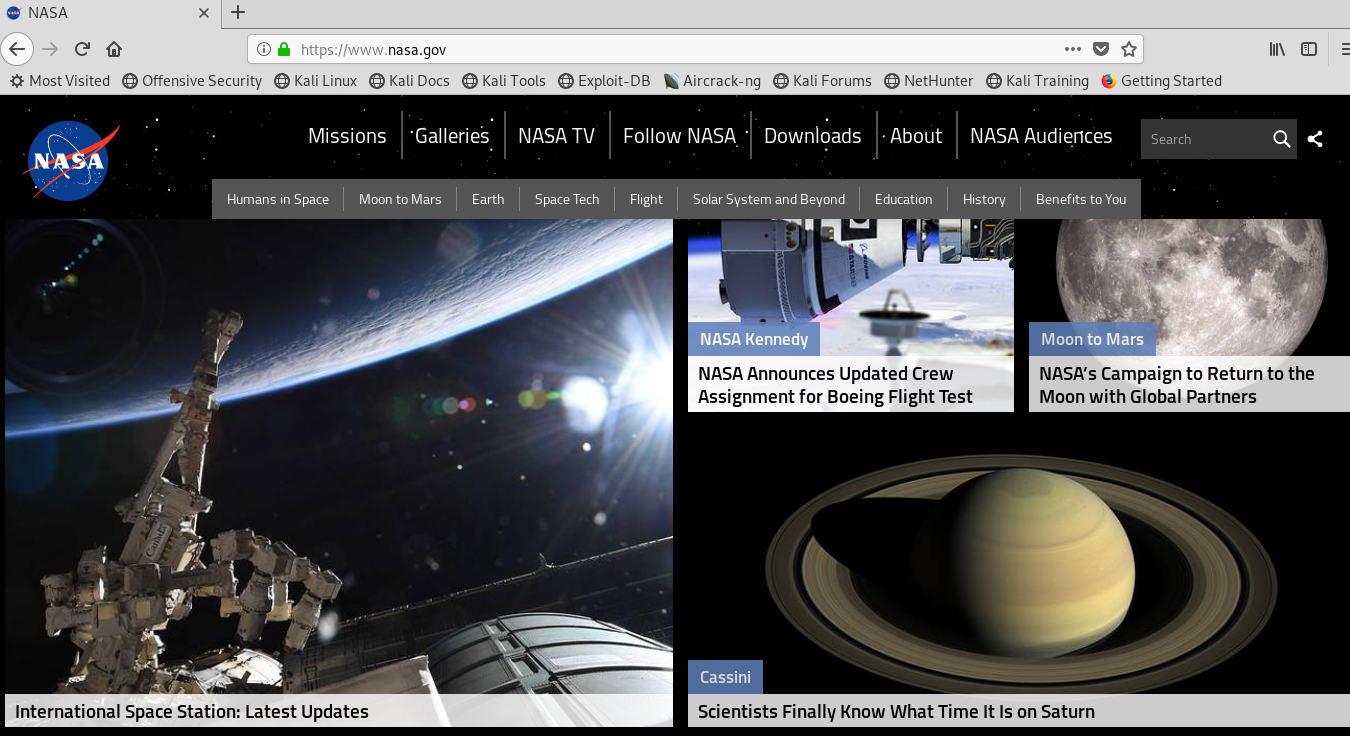
Description automatically generated with medium confidence

Scroll down and select Proxy DNS when using SOCKS v5. You want your DNS requests to go through the encrypted tunnel. Otherwise someone who was monitoring your Internet connection could use your DNS requests to determine which sites you are browing.

Graphical user interface, text, application, email

Description automatically generated

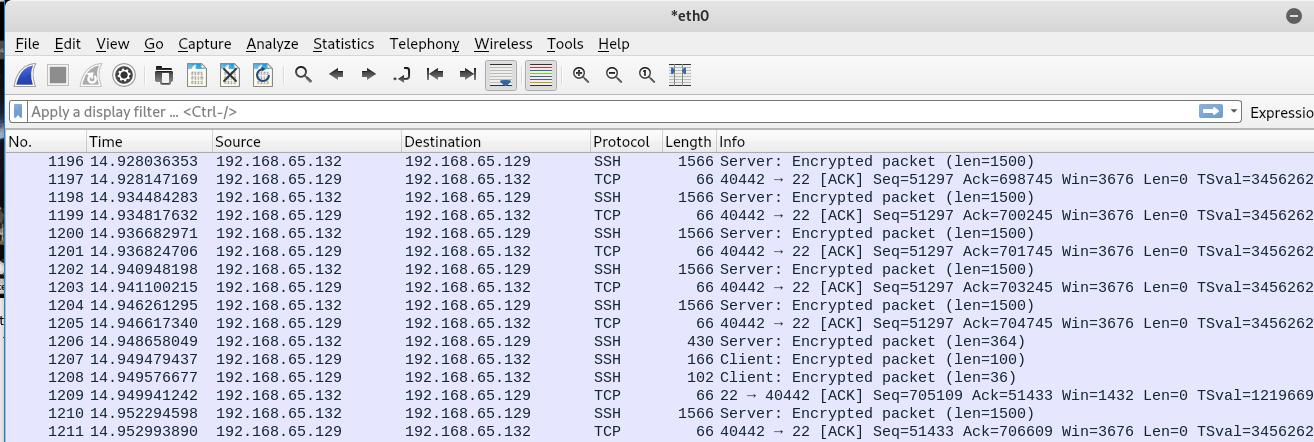
Note: People doing web application penetration tests change proxies so often they install browser plugins to allow them to change proxy information from the main browser window.

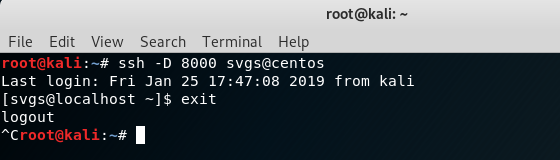
With the dynamic tunnel configured we can browse normally. The browser will send its requests through the encrypted SSH tunnel to the CentOS machine. The CentOS VM will forward them to the Internet, receive the answers, and send them back to the Firefox browser through the tunnel. 

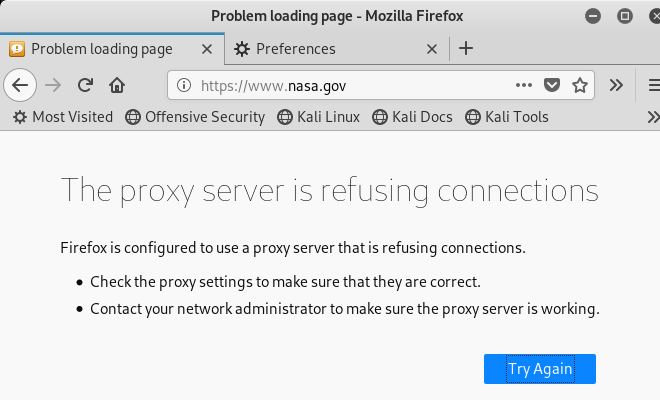
Browse to the site, <https://whatismyipaddress.com/>. You should see that your browsing traffic is coming from the IP address of the SSH server. In this case, I am using an AWS instance located in Ohio.  
Graphical user interface

Description automatically generated  
(The site is trying to sell you a VPN, so they will tell you that your information is exposed. Ha, we already have a VPN!)

Another way to verify that our browsing is really going through the tunnel is to run Wireshark or tcpdump to examine the traffic. All the traffic is SSH traffic to the SSH server. There is no HTTP or HTTPS traffic visible since it is inside the encrypted SSH tunnel.

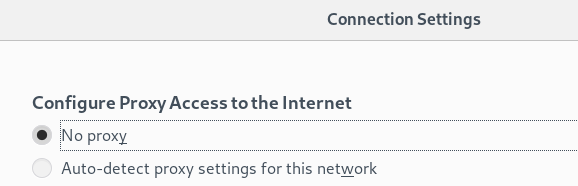


A third way to verify that the traffic is using the SSH tunnel is to disconnect the SSH session.  


The proxy server (SSH dynamic tunnel using SOCKSv5) is no longer running, so the browser can no longer connect to the Internet.  


## Return the VMs to their normal configuration

On the CentOS VM, delete the second network adapter (Host-only) we added.

On the Kali VM, change the network adapter back to the NAT network, and set the Firefox browser back to “No proxy”.  


## Note

Network connections in VMware Workstation Player work like hubs instead of switches; a VM will see all traffic on its network, not just traffic that is addressed to the VM. If we used our original configuration with the CentOS and Kali VMs both on the NAT network, and just one network adapter on the CentOS VM, the Wireshark capture would have been confusing. We would see the SSH connection, like the one above, but we would also see the traffic from the CentOS VM to and from the Internet. It would not have been so obvious that the browser traffic was using only the SSH tunnel.