How to tunnel Internet traffic over SSH in Windows

using free software

This is a basic guide to SSH dynamic port forwarding. It is intended as an introduction to this technology for intermediate to advanced computer users in the hopes that it will be useful. It is not intended to be the best nor most comprehensive guide on the subject. I found a similar document here.

SSH is a protocol for secure (encrypted) communications, most commonly used for remote login sessions to the command line on various Unix-like environments (Linux, Solaris, BSDs, Darwin, etc.). Many academic and other institutions offer accounts on Unix clusters or other machines with a Unix-like operating system. Often these accounts allow login using SSH. If you do not already have one of these accounts, you may be able to get one at one of the sites listed here. [Note: I do not endorse any of the services.]

Most other Internet traffic can also be transmitted through this secure channel through several options called "tunneling" or "port forwarding". Here I will introduce one of these methods, called "dynamic port forwarding", which I find particularly useful. It emulates a SOCKS proxy on the local computer, which Internet applications can then use to tunnel their traffic. [Note: If you are using a corporate computer, restrictions may prohibit this from being done.]

Note that this specific method only works for *outgoing TCP connections*. UDP connections and incoming connections cannot take advantage of this method. If you need to listen to incoming connections from specific ports (and those ports are not already reserved on the SSH server computer), you can use remote port forwarding; it is pretty straightforward, but outside the scope of this tutorial.

A similar but more versatile method that is often used to solve many of the same problems is a secure virtual private network (VPN). However, VPN services may not always be available in many institutions, or may cost additional money.

Why?

Why would one want to tunnel Internet traffic through SSH? Here are some of the reasons:

- You are on a local network where people might intercept your traffic (like an unsecured wireless network). Tunneling traffic through a secure channel protects your data from being readable once intercepted. Also, someone watching your connections will only see one connection (the SSH connection to the SSH server) and not any of the possibly many Internet connections that may be tunneled through it. This hides information about which sites you visit.
    - Note: Your connections are only protected from your computer to the SSH server. The traffic will still be exposed from the SSH server to its eventual destination on the Internet. Usually this is not an issue as the SSH server runs on backbone wired connections that are not as susceptible to snooping as, for example, unsecured wireless. Nevertheless always use secured protocols (HTTPS, IMAPS, etc.) for your actual Internet communications whenever possible, for additional security.
    - Note 2: Only use SSH servers that you trust. If a malicious person has control over the computer running the SSH server, they will be able to intercept all Internet traffic tunneled over the connection.
- You need to access a site or port on a site which is not accessible to your computer, but which is accessible to the computer running the SSH server. This works because the connection will appear to come from the SSH server computer rather than from your own local computer. The need arises, for example, in the following situations:
    - There are sites which are restricted to certain IPs that belong to their institution, and the SSH server is running inside the institution, whereas you are away

  - The SSH server is running from inside an internal network, and you want to visit other sites on the internal network from the outside
  - The computer you are using is behind a firewall that prohibits you from contacting certain IPs or ports on the outside (e.g. censorship), and the SSH server is outside this firewall

Part 1: setting up the SSH connection

- You need an SSH client. For Windows I recommend the free (libre) GUI client PuTTY with lots of features, including the ones we will need. PuTTY will be used for the rest of this section.
- Run PuTTY. It starts in the "Session" screen; fill in the settings for your SSH connection. The fields "Host Name" and "Port" are pretty self-explanatory. You can enter the username too by filling the "Host Name" field in the "user@host" format. Make sure "SSH" is selected in "Connection type:".
- Go to the "Connection" -> "SSH" -> "Tunnels" screen to configure our tunnel.
  - Under "Add new forwarded port:", enter some big integer of your choice to enter for the "Source port" field. (The first thousand or so ports are sometimes reserved by the operating system; so pick something bigger.) Here I will use arbitrarily choose 1080 (the SOCKS port).
  - Leave the "Destination" field blank.
  - Select the "Dynamic" radio button.
  - Click the "Add" button. You should see a line in the text box that reads "D1080" (or whatever number you chose).
  - (For those interested, this is the "-D" option in OpenSSH.)
- (Optional:) By default the a login session is opened in the terminal, which usually runs a "shell", allowing you to run commands on the command line on the remote computer. If you absolutely do not wish to use this, you may be able to disable it via the following:
  - Go to the "Connection" -> "SSH" screen.
  - Check the "Don't start a shell or command at all" box.
  - (For those interested, this is the "-N" option in OpenSSH.)
- (Optional:) At this point, it is a good idea to create a saved session, so you do not have to go through this process every time. If you wish to do so, go back to the "Session" screen; enter a name for the session and click "Save".
- Now you can open the connection. Click the "Open" button at the bottom.
- The session window will open. If this is your first time connecting, it will ask you to add the key; "yes" is recommended. Enter the password when prompted. (You may also set it up to authenticate using public key instead of password, but that is beyond the scope of this tutorial.)
- The login session is now connected. As long as the session is open, you will now have a SOCKS proxy running on on the local computer (localhost) at port 1080 (or whatever port you chose).

Part 2: using the SOCKS proxy

Method 1: SOCKS-supporting applications

Many applications support using SOCKS proxies to connect.

Warning: Many SOCKS-supporting applications "leak" DNS requests; i.e. even though the data is transmitted through the proxy, they look up domain names through the regular outside connection. If this occurs, it is bad for many reasons:

- Any eavesdropper will be able to tell which sites you visit (even though they do not know exactly what data is being transferred).
- Sometimes the local DNS server refuses to look up certain domains (e.g. censorship); resulting in not being able to find certain sites.
- On a network such as unsecured wireless, it is possible for a malicious user to pretend to be the DNS server and "hijack" the request. They return a fake IP to an imitation of the real site (you would not notice because the URL looks correct), and phish your private information.

If you use an application which uses hostnames (rather than just IPs), such as a browser, and you care about DNS request leaks (and you probably should), you should either use an application which specifically supports remote DNS lookups through the proxy (SOCKS 4a protocol); or use Method 2 below.

Example: Mozilla Firefox browser

- Go to "Tools" menu -> "Options"
- Go to "Advanced" screen -> "Network" tab
- In the "Connection" section, click the "Settings..." button
- Select the "Manual proxy configuration" radio button
- Make sure "Use this proxy server for all protocols" is unchecked
- Make sure the "HTTP Proxy", "SSL Proxy", "FTP Proxy", "Gopher Proxy" fields are cleared
- For "SOCKS Host", enter "127.0.0.1", and for "Port" enter 1080 (or whatever port you chose)
- Select the "SOCKS v5" radio button
- Click OK. Click OK.
- Preventing DNS leaks is supported in Firefox 1.5.0.2 and above. Do the following:
    - Go to the URL "about:config"
    - Find the setting "network.proxy.socks_remote_dns" and set it to "true"

Example: Internet Explorer browser

- Go to "Tools" menu -> "Internet Options"
- Go to "Connections" tab
- Click the "LAN Settings" button
- In the "Proxy server" section, make sure the "Use a proxy server for your LAN..." box is checked
- Click the "Advanced" button
- Make sure "Use the same proxy server for all protocols" is unchecked
- Make sure the "HTTP", "Secure", "FTP" fields are cleared
- For "Socks", enter "127.0.0.1" as the address, and for "Port" enter 1080 (or whatever port you chose)
- Click OK. Click OK. Click OK.
- I don't know of any built-in support for preventing DNS leaks

Method 2: SOCKSify any application

- Get a program which can "socksify". For Windows there are several free programs to do this, including:
    - SocksCap - freeware; website no longer available but downloads can still be found around the Internet
    - FreeCap - open-source free software; development stopped a few years ago
    - WideCap - freeware; automatically socksifies all applications on the system without doing it individually like FreeCap
    - FreeCap will be used for the rest of the section.
- Run FreeCap. Go to "File" menu -> "Settings"
- It will show the "Proxy settings" screen.
    - For "Server" enter "127.0.0.1"
    - For "Port" enter 1080 (or whatever port you chose)
    - Under "Protocol" select "SOCKS v5"; make sure "Authentication required" is cleared
    - Click "Apply"
- Now go to the "Program" screen
    - Under "DNS name resolving", make sure it is set to "Remote"
    - Click "OK"
- You return to the main FreeCap window
- To add an application, either
    - Drag the icon of your application from the desktop or Windows Explorer into the FreeCap window, or
    - Click the "New application" button and "Browse" to the location of the program
- Double click on the icon in the FreeCap window to run the application

- Now its connections should be tunneled over SSH

Last updated: 2011
Questions and suggestions are welcome.
xuanluo at ucla dot edu