SSH Tunneling In The Context Of Netwars

By

ace1

So you've made it out of the ISO and are on the airlock. This is where many people ask "How do I use my own tools, there is nothing here?" The answer is, by using SCP or SSH tunnels. Since there is limited space on the airlock, SSH tunnels will be the focus of this write-up.

SSH tunnels can be explained in their most basic form by saying that it is secure port redirection. This means that a port or ports, on a remote network are redirected to a port of your choosing on localhost or vice versa. Once the tunnel is set up, you then access the remote target by pointing your browser or tool to the port you chose to bind to on localhost. If this doesn't make sense, it soon will after a few examples.

Example 1. Syntax and Testing

For this example we will tunnel to the scoreboard and verify that the tunnel is up and working. The scoreboard is on the Netwars network at 192.168.10.110 on port 8888. To set the tunnel up we will begin with the same syntax that we use to login to the network, as a refresher this is what that should look like. This is NOT the syntax for the tunnel yet, simply a refresher

```
ssh -i /path/to/sshkey -p 2020 userXXXX@al[1-7].netwars.info
```

Here you will enter your specific path to your SSH key you have extracted from the ISO and "XXXX" will be replaced by the unique user number you were given, and the al[1-7] in al[1-7].netwars.info will be al plus a number 1-7 (al1.netwars.info or al2.netwars.info etc).

To forward a local port to the remote network we will add -L and the syntax localport:remoteIP:remoteport and we will also use -N so that a command is not executed once the tunnel is set up. -N will make it appear as if the command hangs, but it will show us connection errors if any occur. If -N is not used it will simply connect as normal and you will see a command prompt on the airlock. We now will choose a port we would like to use on localhost, be sure to choose a port that is not already in use for anything, if you have one tunnel up using a specific port and would like to open another tunnel at the same time, the port you choose for your second tunnel must be a new port that is also unused. For this example lets use port 1337 (note: to bind to ports under 1000 you must be root). We can now begin to fill in our syntax, since 1337 is the local port, we will add it to the first part of our syntax, which will now look like this -L 1337:remoteIP:remoteport. we know that the scoreboard is on the IP 192.168.10.110 so we can add that as well, -L 1337:192.168.10.110:remoteport, and finally we know that the scoreboard is on port 8888 so we can finish by adding that -L 1337:192.168.10.110:8888. Now we add this and the -N to the original login command and our final syntax will look like this

ssh -i /path/to/sshkey -p 2020 userXXXX@al[1-7].netwars.info -L 1337:192.168.10.110:8888 -N

At this point you can go ahead and run that command. You will NOT get a connection message or command prompt on the airlock if you used -N, it will appear to hang and the cursor will flash, this is NORMAL. Now open a new terminal window so that we can test connectivity. Since we have bound the remote port to our local port 1337 we will connect to our local port using netcat to verify that we are being redirected to the scoreboard. To do this type

nc 127.0.0.1 1337

or

nc localhost 1337

You should now see the scores. If so, your tunnel is set up successfully, if not verify that your syntax, key name and path, and user name is correct.

Example 2. Metasploit Through a Tunnel

Now that we have seen a basic example of using an SSH tunnel, let's look at how we use a different tool through the tunnel. I will use Metasploit for this example since it is commonly used in Netwars. I will not describe how to use Metasploit as a whole, just the parts that apply to using Metasploit with a tunnel. If you would like to know how to use Metasploit take a look at Metasploit Unleashed as a starting point http://www.offensive-security.com/metasploit-unleashed/

For this example let's say that there is an ftp service (port 21) on the IP address 192.168.10.46 that you would like to try Metasploit on. For this example we will say that we still have port 1337 in use with another tunnel. Since we need a new tunnel on a port that is not in use, we will use port 1338. Our syntax will look like this for this command -L 1338:192.168.10.46:21 and the actual command will be

```
ssh -i /path/to/sshkey -p 2020 userXXXX@al[1-7].netwars.info -L 1338:192.168.10.46:21 -N
```

Once again, remember that the tunnel will appear to hang if you used -N, but needs to stay open. From this point we can begin to use Metasploit as normal in another terminal. The only things that will be different, are the values of RHOST and RPORT. Select everything else as you typically would but when you get to RHOST you will enter 127.0.0.1, and for RPORT you will enter 1338. Remember, even though this is local, the port is being redirected from 127.0.0.1:1338 to 192.168.10.46:21. If you have all your other values set correctly you will have exploited the ftp service through the tunnel. The main thing to remember is that the remote IP(RHOST) will always be 127.0.0.1 (or localhost) when using a tunnel and the remote port(RPORT) will be the local port you chose when setting up the tunnel.

Example 3. Mutiple tunnels, websites, etc.

It is possible to add as many -Ls to one command as you like. Let's use the previous two tunnels and add another and run it as a single command. For this we will add a tunnel to a website and look at how to view that also. The website we will add is the one from 192.168.10.55, and our local port we will bind this to is 1339. Our syntax for this will read -L 1339:192.168.10.55:80. The actual command to setup all three tunnels would read like this, but all on one line

```
ssh -i /path/to/sshkey -p 2020 userXXXX@al[1-7].netwars.info -L 1338:192.168.10.46:21 -L 1337:192.168.10.110:8888 -L 1339:192.168.10.55:80 -N
```

Now to view the website we just created a tunnel for, open a web browser and enter the address 127.0.0.1:1339 and hit enter. You should now be viewing the website on 192.168.10.55. You will also still be able to use the other tunnels that you have created. If you are going to create multiple tunnels I would suggest making a script and hard coding them in so that you don't have to retype everything if your connection is lost for some reason.

Example 4. Dynamic tunnels and SOCKS proxies

At this point it could seem like there is a lot of typing involved if you wish to view multiple things on the network. This is where a dynamic tunnel would be useful. Two things to note. The first is that -D will only be useful for programs that can use a SOCKS proxy. The second is that -D only works for TCP and UDP. For reference you would be able to view a website through the proxy, but you would not be able to ping boxes since it uses ICMP.

The syntax for a dynamic tunnel is basic compared to the syntax that we have been using for the -L tunnels. It is only composed of -D portnumber. Here once again you will need to choose an unused port. for this example we will use 9999. The command to set up a dynamic tunnel will read as follows

```
ssh -i /path/to/sshkey -p 2020 -D 9999 userXXXX@al[1-7].netwars.info
```

Note that in this command I did not add the -N. If you would like to add that so that you can see errors you may do so, remember if -N is used the command will

appear to hang, if it is not used, you will receive a command prompt from the airlock. Once the command has been run we now have a dynamic tunnel and can use it as a SOCKS proxy with programs that support that. The most common program to use this with would be a web browser. This is not the only thing it will work with, but I am using it as an example since everyone should be familiar. To use this in Firefox you will need to open up "Preferences", then go to "Advanced", then click the "Network" tab, and click the "Settings" button. From there click the radio button next to "Manual proxy configuration". Under "SOCKS Host" enter 127.0.0.1 and for "Port" use the local port we chose earlier which was 9999. Click "OK" and then close out the preferences window. Now you can use Firefox to view websites on the network by IP. Note that here we will browse to the actual IP address on the remote network, for example 192.168.10.53. When using this, you won't be able to view sites outside the network (google.com, etc) until you turn off the proxy.

If you would like to view sites on the internet and sites on the network, I recommend using an add-on for Firefox like FoxyProxy. Using FoxyProxy, you can set rules so that the SOCKS proxy is only used when a certain URL pattern is entered, and all other sites are viewed as if the proxy was not in use.

After you have installed FoxyProxy, configuration is fairly simple. First open up FoxyProxy options. Now you will click the button that says "Add New Proxy". Click on the "General" tab if it is not already selected, and name this whatever you would like, and add any notes you choose. Next click on the "Proxy Details" tab. Click on the "Manual Proxy Configuration" radio button and enter 127.0.0.1 as the IP address and enter 9999 (or whatever you setup as the port in the -D tunnel)as the port. Now click on the "URL Patterns" tab. This is where we will tell FoxyProxy to use our SOCKS proxy and under what conditions. Click the button marked "Add New Pattern", name this whatever you like. Now add the URL pattern. Here I typed

192.168.10.

Make sure that the radio button named "Whitelist" is selected and also the one titled "Wildcards". From here you can add another pattern for the .20 network if

you like, just follow the above steps but use *192.168.20.* for the URL. Now just make sure that they are enabled, click "OK" twice and then close out the options. All that is left to do is verify that it works. You should now be able to browse any site on the internet, and also any site on the Netwars network without having to do anything more than typing the address of wherever you wish to visit.

Example 5. ProxyChains

ProxyChains is a program that will allow you run any program through a SOCKS proxy. It can also be used to chain a number of proxies together but that is somewhat outside the scope of this paper since we really only need to use our SOCKS proxy. See the man pages and config file for details on that. ProxyChains is useful when you don't want to type a bunch of -L statements, and the tool which you are using does not have SOCKS proxy support.

ProxyChains is installed by default in BackTrack 4. To use this you will modify the config file which can be found at /etc/proxychains.conf Here you will want to add your proxy at the end of the file. There is currently one default entry at the end of the file that will use Tor. Simply just change the port number of this entry to the one you chose when setting up your dynamic tunnel. If we use the port from example 4 you will change the existing line to read

socks4 127.0.0.1 9999

Once that is done, save the file then exit. Now to use Proxychains all you need to do is preface any command with proxychains, for example

proxychains nc 192.168.10.110 8888

If everything is configured correctly you should once again see the scoreboard. Remember even though ProxyChains will run any program through a SOCKS proxy you are still bound by the limitations of SOCKS that were discussed above.

These examples should cover most of what you will use in Netwars. If anything is not clear feel free to come into the IRC channel and ask a question.