# Tunneling and Pivoting:

The goal here is to send traffic through a compromised host (which I’ll refer to as beachhead) to other target hosts the beachhead can talk to. There’s a lot you’ll be able to do from the beachhead itself. But there will be times that you want to use tools on your workstation to communicate with hosts through the beachhead. How I do that will depend on what kind of access I have to the beachhead host. The best case is if I can ssh into that host, because it allows me to port forward, and better yet, opens the door for a really nice tool, shuttle. But, more often than not, I’ll find myself with only a nc reverse shell, and I’ll show some options here as well.

Live Off the Land

Why?

Before going to a ton of effort to figure out how to get your workstation talking to target hosts through the beachhead, consider what you can do from the beachhead itself, since you can already run commands there. Linux workstations may have nmap already installed. They will likely have python and perl, and potentially gcc for compiling things. Bash scripting will take you a long way even if it’s just doing a ping sweep in parallel (putting the command in () with a & at the end will start them all in parallel, so this runs in a second):

root@host:~# for i in $(seq 1 254); do (ping -c 1 10.2.2.${i} | grep "bytes from" &); done;

64 bytes from 10.2.2.10: icmp\_seq=1 ttl=64 time=0.013 ms

Scanning / nmap

Regardless of what kind of access I have to my beachhead, I’m going to want to scan the new network for host and port discovery. While it is possible to set up tunnels to scan, it’s very difficult to do, and flaky at best. If nmap isn’t already on the beachhead, my preferred method is to bring a copy of nmap that’s statically compiled to beachhead (typically via wget or curl on linux, or smb on windows).

You can [compile the source yourself](https://blog.zsec.uk/staticnmap/), or there’s a few GitHub repos out there with statically compiled tools for various oses / architectures:

* https://github.com/andrew-d/static-binaries
* https://github.com/static-linux/static-binaries-i386
* https://github.com/yunchih/static-binaries

For nmap, if you’re in a very stripped down container, you may get an error Unable to open /etc/services for reading service information. Just grab a copy of that file from your local box, upload it to the beachhead and drop it in /etc. You won’t have access to all the nmap scripts, but you can get feel for what exists.

SSH Into Beachhead Target

SSH Tunneling

The easiest tunneling case is when you have ssh access to the beachhead machine. I wrote a post earlier about [SSH Tunneling](https://0xdf.gitlab.io/2018/06/10/intro-to-ssh-tunneling.html). I won’t repeat that here, but the summary is this:

* To tunnel a single port through an SSH tunnel, connect with -L [local listen port]:[target ip]:[target port]. Then send traffic to 127.0.0.1:[port], and it will go through the tunnel to the [target ip]:[port].
* To set up a proxy, use -D [port], and then set your proxy to 127.0.0.1:[port].

When you’re using a proxy, you can do that with a browser (either in the browser settings, or I use [FoxyProxy](https://getfoxyproxy.org/) for quick changing), or you can use a tool called proxychains.

To use proxychains, first edit /etc/proxychains.conf by adding your proxy under [ProxyList] at the bottom of the file (and commenting others out). Mine looks like this when working with a -D 1080:

60 [ProxyList]

61 # add proxy here ...

62 socks4 127.0.0.1 1080

Then you can run proxychains [tool], and it will run that tool proxied through the tunnel. Some tools behave better than others. Also, if you are sending some kind of exploit to the target host, consider what payload you use. If you use a reverse tcp shell, can the new target talk back to your listener on localhost? Your exploit is likely kicking off a new process that will not be aware of this proxied traffic. You can solve this by listening on the beachhead if nc is there.

sshuttle

During PWK is discovered a tool called sshuttle. It’s so awesome. Install with apt install sshuttle or pip install sshuttle.

So if I have a beachhead device at 10.1.1.1, and it also has an interface on 10.2.2.0/24 with other hosts behind it, I can run:

# sshuttle -r root@10.1.1.1 10.2.2.0/24

root@10.1.1.1's password:

client: Connected.

This creates a VPN-like connection, allowing me to visit 10.2.2.10 in a browser or with curl, and see the result.

Some milage may vary. I’ve never had success running nmap through sshuttle, and there are a lot of people out there posting similar complaints. But it is a very nice way to interact with a host over a tunnel.

Without SSH Access to Beachhead

Metasploit Meterpreter

portfwd

I tend to try to avoid using Meterpreter, but the port forwarding ability is one place where it really outshines other options. If you can get a shell on a box, you can likely get a meterpreter shell as well. From there, you can run something like:

meterpreter > portfwd add -l 80 -r 172.19.0.4 -p 80

Now, you can point your browser at http://127.0.0.1, and it will forward traffic through your meterpreter session, and from there to a remote host, in this case 172.19.0.4 port 80.

The biggest drawback is that you’ll need to add this for each port you want to tunnel.

Autoroute

If you are working in Metasploit, you can also background the session, and then use post/multi/manage/autoroute. The options are relatively straight forward:

msf post(multi/manage/autoroute) > options

Module options (post/multi/manage/autoroute):

Name Current Setting Required Description

---- --------------- -------- -----------

CMD autoadd yes Specify the autoroute command (Accepted: add, autoadd, print, delete, default)

NETMASK 255.255.255.0 no Netmask (IPv4 as "255.255.255.0" or CIDR as "/24"

SESSION yes The session to run this module on.

SUBNET no Subnet (IPv4, for example, 10.10.10.0)

Give it the subnet you want to target, and the session you want to forward over, and run it, and then you can work against the subnet from within Metasploit as if you can talk directly to it.

Metasploit socks proxy

You can also use auxiliary/server/socks4a. This will allow you to route things through Metasploits routes as a proxy. So after settign up autoroute, you can create a socks proxy that will listen, route traffic to the appropriate session, and then send it from there. I don’t have too much experience here, but it’s something that would work if you work within Metasploit.

Reverse SSH

Most linux hosts will have an ssh client. And while it is less common on Windows, you can upload one (plink.exe is a stand-alone exe that is at /usr/share/windows-binaries/plink.exe on Kali). From there, you can ssh back to your attacker box with a -Rflag, which will open up listening ports on your attacker box, that are forwarded through the tunnel and out the other side. This will also require you to create a tunnel for each target/port combination you want to talk to.

SSH Support Escape Sequences

If you’re going to be creating tunnels over SSH, you’re almost certainly going to need to change the tunnels or create new ones. That’s really annoying, if it means disconnecting and reconnecting with new flags. SSH Control Sequences to the rescue. There’s a [post from Jeff McJunkin](https://pen-testing.sans.org/blog/2015/11/10/protected-using-the-ssh-konami-code-ssh-control-sequences) which describes this well. The sort version is, hit enter, then ~ (the tilde, top left of the US keyboard), then one of the characters to interact with the SSH session. The most useful is C, which opens the command prompt, and allows you to add in something like -D 9001, and then resumes the session.

So, for example, to add a port forward port 8080 from your local host to a target 10.3.3.3 on port 80, you’d do the following:

1. enter
2. ~C
3. At the ssh> prompt, -L 8080:10.3.3.3:80

It looks like this:

root@host:~$

ssh> -L 8080:10.3.3.3:80

Forwarding port.

root@host:~$

Summary

Pivoting into a network can be intimidating, but there are tools that will help. Consider what you can do directly from the beachhead host. Bring tools there to work from there. Try to get SSH access if you can. Use meterpreter where you can’t.