INTELIE Challenge

Daniel Carvalho Dehoul

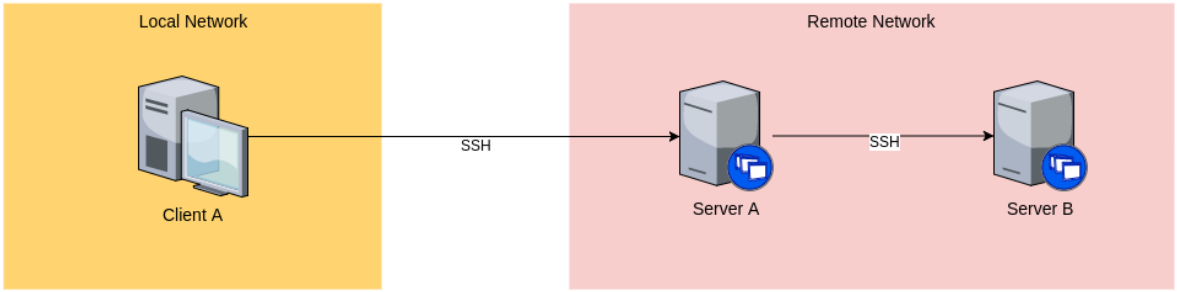
October 30th, 2019

Accessing a remote service

In this challenge, you will propose a way to access a service running on a remotely unreachable port.

Consider the architecture proposed on the figure below:

Example Network Architecture



The only allowed connection between Client A and Server A is via SSH. The only allowed connection between Server A and Server B is via SSH.

We need to access, from Client A and using HTTP, a service running on port 8000 of Server B.

Notes and constraints:

- There is no direct route from Client A to Server B, and no practical way to build one. Imagine the following scenario as example: Server B belongs to a customer internal datacenter, and we were provided with a VPN that allows access to Server A SSH port only.
- There is another service running on port 8000 of Server A, we must not cause impacts on this one

Both client and servers run CentOS 7 without X.

We expect you to:

- Provide the set of commands that allow the service to be accessed
- Provide a document, in English, explaining each of the commands, their possible outputs, including failures, and why you chose them
- You don't need to be concerned about a production-level solution, users will not be accessing the service in a regular basis. You just need to provide an ad-hoc access (as in the typical scenario: support analyst needs to access web interface for some configuration task)
- However it will be welcome to have a short comment on how you would change the architecture/solution to provide permanent access, in the case users in Local Network start to perform heavy usage of application in Server B.

Overview:

Goal: Access service running on server B TCP port 8000. This must be done from Client A, using HTTP protocol.

Restrictions:

- Client A cannot access Server B directly;
- Client A communicate Server A only with ssh;
- Server A communicate Server B only with ssh;
- Server A has a service running on TCP port 8000, this service cannot be impacted.

Set of commands:

Here will be showed the set of commands used to solve the challenge:

- Ifconfig;
- ping;
- service (start, restart, stop and status);
- netstat (with "| grep");
- vi ;
- ssh;
- curl.

Solution:

To solve this challenge i set up a virtual environment on virtual Box, replicating the challenge network architecture. This way i could test my solution and get the commands outputs and their possible failures.

After research about the problem, i found two possible solutions, ssh port forwarding and firewall rules. I decided to use the ssh port forwarding solution.

Before start to implement the solution, i start a test to see if the connections are ok. First, using the command ifconfig on the machines to see the network interfaces of each one (this way i do get the IP address of each network interface of the machines) and then using the ping command (with the flag "–c" to limit the count) to test the connection between those. With that, the expected scenario is:

- Client A should be able to ping the Server A interface that is in same network of it, and, cannot ping Server B network interface;
- Server A should be able to ping server B using the IP address that is in the same network.

The outputs for those commands are showed in the images bellow:

- **Ifconfig (Client A):**

```
[root@client_a ~]# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.25.170  netmask 255.255.255.0  broadcast 192.168.25.255
        inet6 fe80::885a:cb7b:4505:d86f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b7:30:db  txqueuelen 1000  (Ethernet)
        RX packets 644  bytes 74126 (72.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 719  bytes 68388 (66.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- **Ifconfig (Server A):**

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.25.171  netmask 255.255.255.0  broadcast 192.168.25.255
        inet6 fe80::885a:cb7b:4505:d86f  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::22ee:591c:b95c:7c98  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:0f:a0:be  txqueuelen 1000  (Ethernet)
        RX packets 532  bytes 48230 (47.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 389  bytes 50298 (49.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.108  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::35d6:4ea7:ef02:b225  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:82:13:76  txqueuelen 1000  (Ethernet)
        RX packets 38  bytes 4283 (4.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 27  bytes 4539 (4.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- **Ifconfig (Server B):**

```
[root@server_b ~]# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.109  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::885a:cb7b:4505:d86f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4a:f9:bd  txqueuelen 1000  (Ethernet)
        RX packets 61  bytes 9097 (8.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 27  bytes 4539 (4.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- **Ping (Client A):**

```
[root@client_a ~]# ping -c 3 192.168.25.171
PING 192.168.25.171 (192.168.25.171) 56(84) bytes of data.
64 bytes from 192.168.25.171: icmp_seq=1 ttl=64 time=11.9 ms
64 bytes from 192.168.25.171: icmp_seq=2 ttl=64 time=0.466 ms
64 bytes from 192.168.25.171: icmp_seq=3 ttl=64 time=0.510 ms

--- 192.168.25.171 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.466/4.299/11.922/5.390 ms
[root@client_a ~]#
[root@client_a ~]# ping -c 3 192.168.56.108
PING 192.168.56.108 (192.168.56.108) 56(84) bytes of data.

--- 192.168.56.108 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2000ms

[root@client_a ~]#
[root@client_a ~]# ping -c 3 192.168.56.109
PING 192.168.56.109 (192.168.56.109) 56(84) bytes of data.

--- 192.168.56.109 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2001ms
```

As expected, the client A can ping the Server A interface that is in the same network of Client A, but, cannot ping the Server B, or Server A other interface, in those cases i get a 100% packet loss.

- **Ping (Server A):**

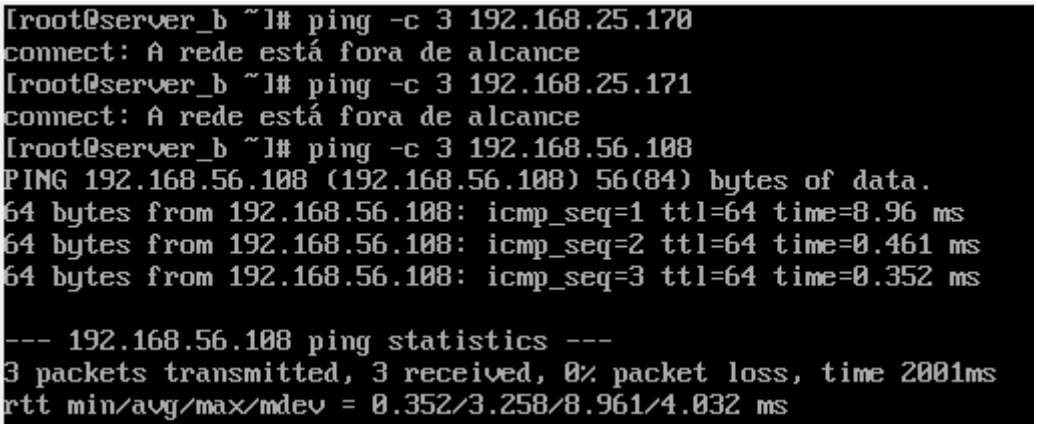
```
[root@server_a ~]# ping -c 3  192.168.25.170
PING 192.168.25.170 (192.168.25.170) 56(84) bytes of data.
64 bytes from 192.168.25.170: icmp_seq=1 ttl=64 time=0.378 ms
64 bytes from 192.168.25.170: icmp_seq=2 ttl=64 time=0.466 ms
64 bytes from 192.168.25.170: icmp_seq=3 ttl=64 time=0.478 ms

--- 192.168.25.170 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.378/0.440/0.478/0.050 ms
[root@server_a ~]# ping -c 3  192.168.56.109
PING 192.168.56.109 (192.168.56.109) 56(84) bytes of data.
64 bytes from 192.168.56.109: icmp_seq=1 ttl=64 time=5.25 ms
64 bytes from 192.168.56.109: icmp_seq=2 ttl=64 time=0.427 ms
64 bytes from 192.168.56.109: icmp_seq=3 ttl=64 time=0.457 ms

--- 192.168.56.109 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.427/2.047/5.258/2.270 ms
```

As expected, the Server A can ping the Client A interface that is in the same network of Server A, and can ping the Server B.

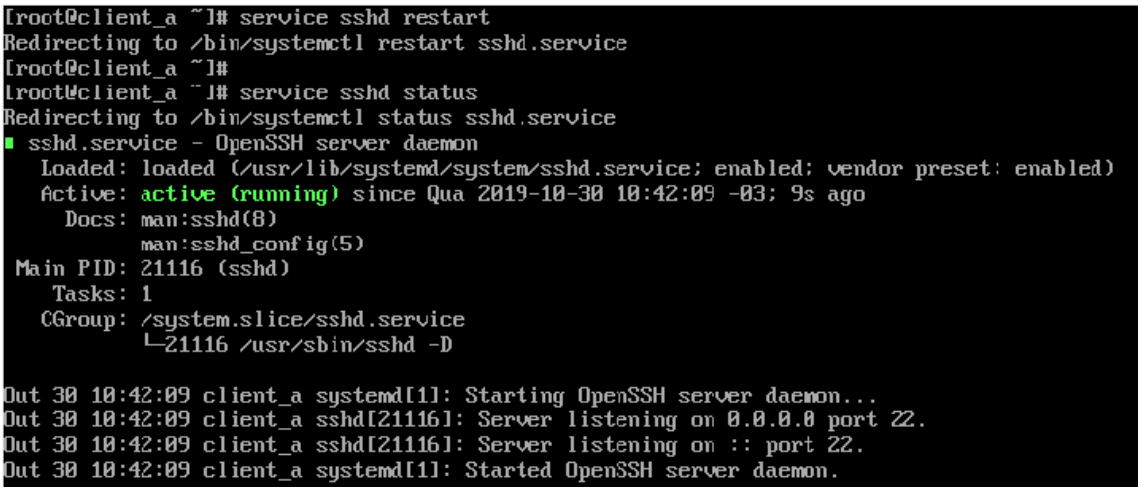
- **Ping (Server B):**



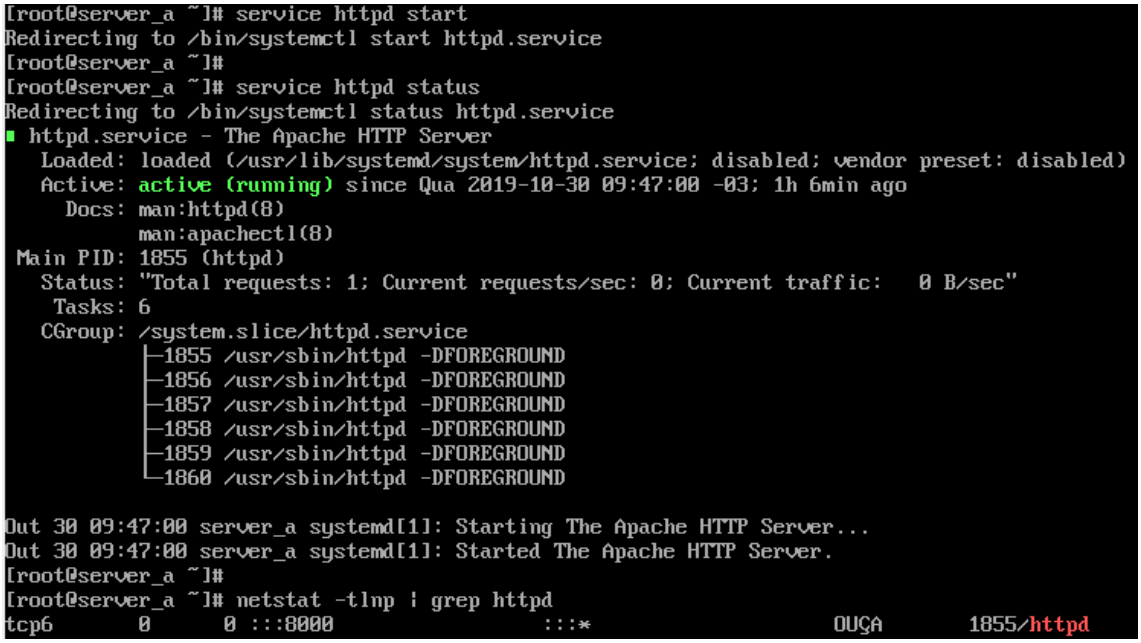
As expected, server B can only ping server A's interface that is on the same network as server B. Any other attempt fails and gives us the error message "network out of range".

To this solution work it is needed to make changes in the sshd config file on “/etc/ssh/sshd_config”. It is required to enable the AllowTcpForwarding and GatewayPorts feature on the config file, to this i used the command “vi /etc/ssh/sshd_config” (this command is used to edit documents, in this case the sshd_config that is in the directory “/etc/ssh/”). Then, i navigate thru the document till find the “#AllowTcpForwarding”, so i could remove the “#” and change the “no” to “yes” (if not already set to yes), same procedure was done to GatewayPorts. This is required to both servers and the client. A warning, when editing the config file, the user must have root privileges, otherwise will result in permission denied error while saving the alterations.

After the changes it is needed the restart the service, so i used the command “service sshd restart”. To check if the service is up again i used the command “service sshd status”, it will show the actual service status. As showed in the image bellow with Client A, but, this procedure must be done in the servers too.



To the scenario proposed, i installed the httpd in the virtual machines, and then change the TCP port from 80 to 8000. Also created a index.html on each server with a simple expression using the vi: “<h1>This is Server (A or B) test page<h1>”. As required in this challenge, Client A should be able to access the service on port 8000 on server B without impact server A service on this port, so after forward the ports i tryed to access both html pages. To this, it is needed to start the httpd service on the servers, I’m using the command "service httpd start", and then use "service httpd status" to verify if worked. In the image bellow is showed the procedure on Server A, but, it must be done on Server B too. I also used the command “netstat –tlnp | grep httpd” to identify the process httpd port.

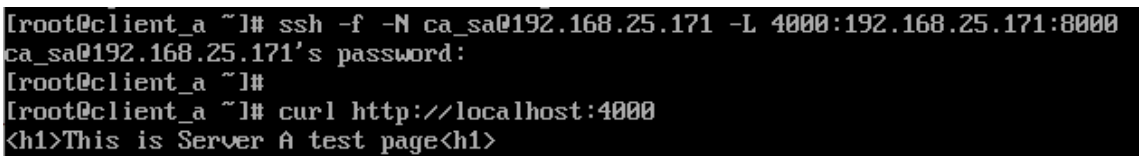


The command “netstat” is used to show network connections, routing tables, interface statistics and masquerade connections. The use of the flags is explained bellow:

- **-t** -> used to show tcp;
- **-l** -> used to show only listen process;
- **-n** -> used to show the IP number and the port number by not resolving IP addresses and port number;
- **-p** -> used to show the process name in the listen port.

The “| grep htppd” is use to filter the output and show only the outputs that has the httpd on it.

Only for a little test, i tried forward the TCP port 4000, on Client A, to the server A TCP port 8000, with this i could reach the index.html from server A. Using the command “ssh –f –N user@ServerA_IP –L 4000:ServerA_IP:8000” (I created a admin user ca_sa, to the client A on Server A), and then i use the command “curl http://localhost:4000” to see if it worked, as showed in the image bellow:



In ssh the flags and its effects are explained bellow:

-f -> Used to run ssh in background

-N -> Used to not get a shell from the server, this way it won't be possible to execute remote commands (the only goal here is to reach the service on the port);

-L -> Used to specify the port forwarding.

The curl is a command to transfer data using a supported protocol, in this case http, with that command we can get data from the index.html file.

There are a few errors that can occur, if the sshd_config file was not correctly modified, the port forwarding won't be possible, when trying the command it will get to an error. Also, if the httpd service was not started, it will cause an error when using the command "curl", to illustrate this scenario i stop the httpd service on server A, and on the client A, I tried the command "curl" once again. The output was the following:

```
[root@client_a ~]# curl http://localhost:4000
channel 2: open failed: connect failed: Connection refused
curl: (52) Empty reply from server
```

Next I repeat the test, but this time is from server A to server B. On Server A, I used the command "ssh –f –N root@ServerB_IP –L 4444:ServerB_IP:8000" then use the command "curl localhost:4444" and the command "curl localhost:8000", this way it supposed to be possible to reach the server b index.html and see if server A index.html still accessible(we cannot cause impact on the service running in the port 8000 on server A). The output for the commands is showed bellow:

```
[root@server_a ~]# ssh -f -N root@192.168.56.109 -L 4444:192.168.56.109:8000
root@192.168.56.109's password:
[root@server_a ~]#
[root@server_a ~]# curl http://localhost:8000
<h1>This is Server A test page<h1>
[root@server_a ~]#
[root@server_a ~]# curl http://localhost:4444
<h1>This is Server B test page<h1>
```

As expected, using port 4444 on server A, it was possible to reach the index.html on server B, and, using the port 8000 it still possible to reach index.html from server A. I used the root user on server B, so, i tried again with an user created for client A on server B (ca_sb), and using the ca_sa user on server A.

```
[ca_sa@server_a ~]$ ssh -f -N ca_sb@192.168.56.109 -L 5555:192.168.56.109:8000
ca_sb@192.168.56.109's password:
[ca_sa@server_a ~]$
[ca_sa@server_a ~]$ curl http://localhost:8000
<h1>This is Server A test page<h1>
[ca_sa@server_a ~]$
[ca_sa@server_a ~]$ curl http://localhost:5555
<h1>This is Server B test page<h1>
```

Now is time to test the connection directly from client A to the server B. This must be done in two steps. First, forward a empty port on server A to port 22 on server B, for this test i used port 2200 on Server A. The second step will be login as ca_sb on localhost to this port 2200 and a forward localhost empty port (I choose the port 5000) to port 8000 also on localhost since we wil be logged in using port 2200. So the commands to each steps are:

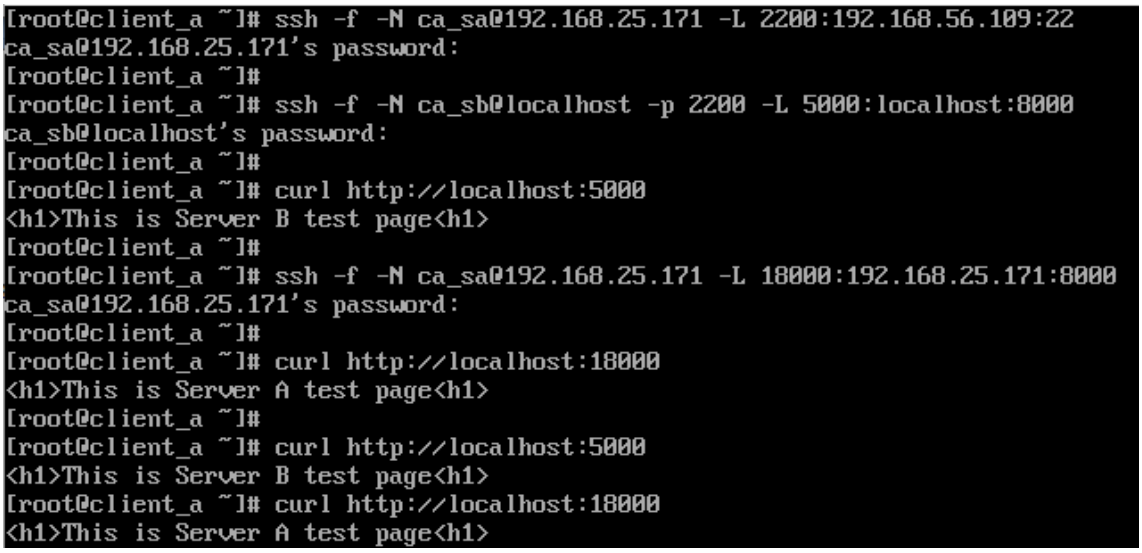
Step 1: ssh –f –N ca_sa@serverA_IP –L 2200:ServerB_IP:22

Step 2: ssh –f –N ca_sb@localhost –p 2200 –L 5000:localhost:8000

Using the command curl "http://localhost:5000" on client A should bring "<h1>This is Server B test page<h1>".

As additional test, i had done a third step, to verify if the service running on Server A port 8000 still accessible. I used the command “ssh –f –N ca_sa@serverA_IP –L 18000:ServerA_IP:8000”. This way, using “curl http://localhost:18000” it should bring “<h1>This is Server A test page<h1>”.

The outputs for those steps are in the image bellow:



Short comment on how you would change the architecture

Maybe a creation of a study to see the possibility of using firewall rules to create a direct route to server B, as it was a restriction to this challenge, it might be something to change in the infrastructure.