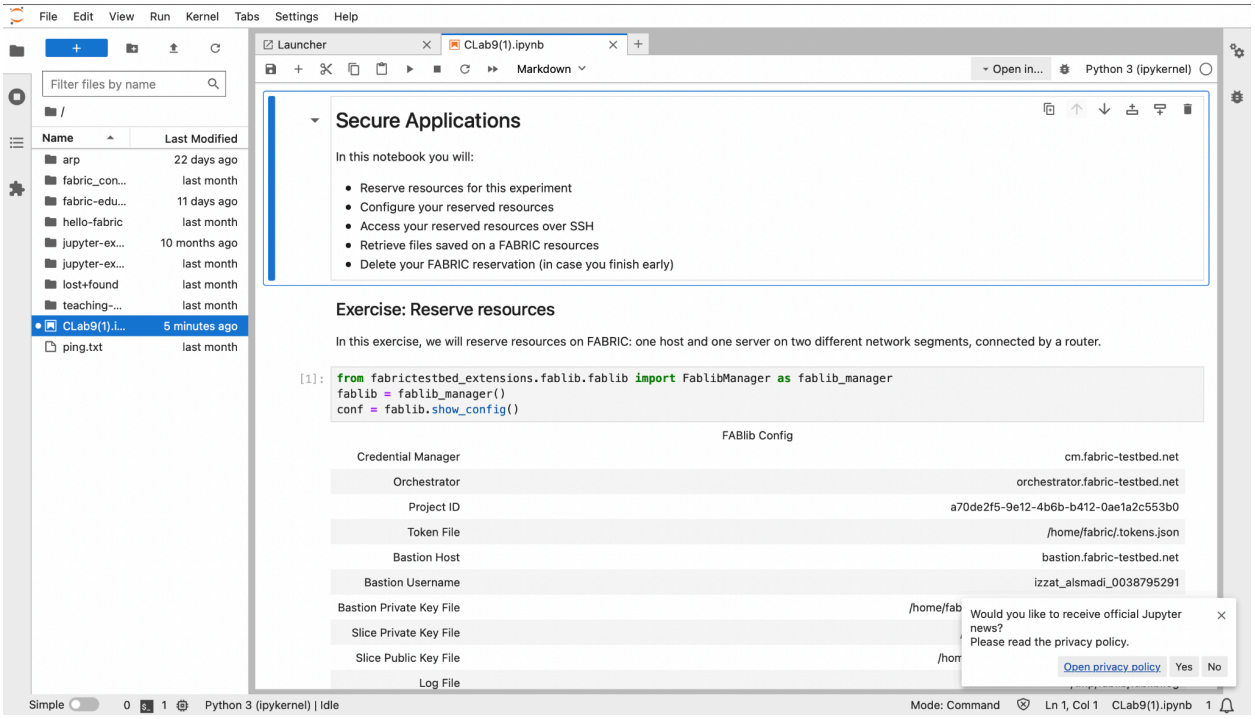
Samantha Jackson
CSCI4406
11-20-2024

CLab 9 - Secure Networked Applications Pt.1

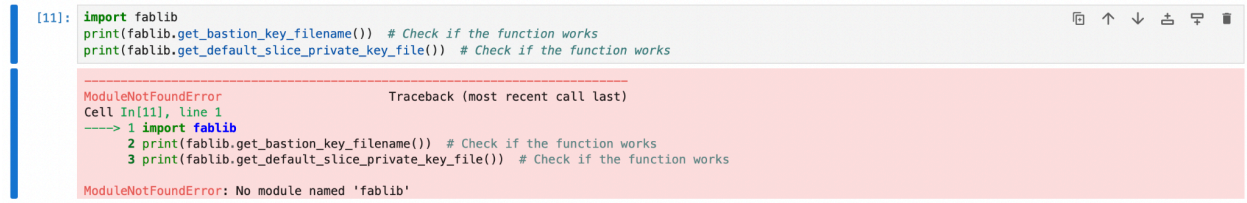
Opening CLab9.ipynb

Observations/Issues: No issues opening the file in Jupyter Testbed.



Setting Up the Exercise

Observations/Issues: The chmod commands are throwing unexpected token errors. I believe the program doesn't recognize the functions in the script for whatever reason. It seems it is because the fablib isn't correctly imported or JupyterHub is incorrectly configured. Jupyter doesn't recognize the fablib module in this test. Attempting to pip install Jupyter doesn't have a distro for fablib? We're just going to attempt to continue.



```
[12]: !pip install fablib

ERROR: Could not find a version that satisfies the requirement fablib (from versions: none)
ERROR: No matching distribution found for fablib
```

Observations/Issues: Continuing on. We have the EDUKY print. We then set up the slice and slivers. After around 240 seconds we have a setup.

Site

Name	EDUKY
State	Active
Address	301 Hilltop Avenue,Lexington, KY 40506
Location	(38.0325, -84.502801)
PTP Capable	True
Hosts	18
CPUs	36
Cores Available	73344
Cores Capacity	73728
Cores Allocated	384
RAM Available	7204
RAM Capacity	8604
RAM Allocated	1400
Disk Available	126872
Disk Capacity	134082
Disk Allocated	7210
Basic NIC Available	4360
Basic NIC Capacity	4445
Basic NIC Allocated	85
ConnectX-6 Available	0
ConnectX-6 Capacity	0
ConnectX-6 Allocated	0

Slice

ID	ca0cbe79-e43b-4c6d-bb23-48a79ba2aaea
Name	lzzat_Alsmadisecure-applications_sjack012
Lease Expiration (UTC)	2024-11-21 22:14:23 +0000
Lease Start (UTC)	2024-11-20 22:14:23 +0000
Project ID	a70de2f5-9e12-4b6b-b412-0ae1a2c553b0
State	StableOK

Nodes

ID	Name	Cores	RAM	Disk	Image	Image Type	Host	Site	Username	Management IP	State	Error	SSH Command
d367925d-1179-4130-80c1-1a2de412fd95	romeo	2	4	10	default_ubuntu_20	qcow2	eduky-w15.fabric-testbed.net	EDUKY	ubuntu	2610:1e0:1700:206:f816:3eff:fee9:cec	Active		ssh -i /home/fabric/work/fabric_c -F /home/fabric/work/fabric_conf ubuntu@2610:1e0:1700:206:f816
a63d9a51-45bd-46db-a0e6-268bb7e78777	router	2	4	10	default_ubuntu_20	qcow2	eduky-w11.fabric-testbed.net	EDUKY	ubuntu	2610:1e0:1700:206:f816:3eff:fee5:d883	Active		ssh -i /home/fabric/work/fabric_c -F /home/fabric/work/fabric_conf ubuntu@2610:1e0:1700:206:f816
9d4d7c40-99d5-4df4-b7c6-6ea99a3f071a	server	2	4	10	default_ubuntu_20	qcow2	eduky-w12.fabric-testbed.net	EDUKY	ubuntu	2610:1e0:1700:206:f816:3eff:feb4:58fd	Active		ssh -i /home/fabric/work/fabric_c -F /home/fabric/work/fabric_conf ubuntu@2610:1e0:1700:206:f816

Networks

ID	Name	Layer	Type	Site	Subnet	Gateway	State	Error
b5334b91-4659-491e-ab41-f6dbea9f0602	net0	L2	L2Bridge	EDUKY	net0.subnet	net0.gateway	Active	
7182d25b-2577-4da7-97d9-16442c57c941	net1	L2	L2Bridge	EDUKY	net1.subnet	net1.gateway	Active	

Configuring Resources

Observations/Issues: No issues observed.

```
}

for iface in slice.get_interfaces():
    if_name = iface.get_name()
    hostname = if_conf[if_name]['hostname']
    iface.ip_addr_add(addr=if_conf[if_name]['addr'], subnet=IPv4Network(if_conf[if_name]['subnet']))


slice.get_node(name='romeo').execute(f"echo '{if_conf['server-net1-p1']['addr']}\t{if_conf['server-net1-p1']['hostname']}' | sudo tee -a /etc/hosts > /dev/null"
slice.get_node(name='server').execute(f"echo '{if_conf['romeo-net0-p1']['addr']}\t{if_conf['romeo-net0-p1']['hostname']}' | sudo tee -a /etc/hosts > /dev/null")
slice.get_node(name='router').execute(f"echo '{if_conf['romeo-net0-p1']['addr']}\t{if_conf['romeo-net0-p1']['hostname']}' | sudo tee -a /etc/hosts > /dev/null")
slice.get_node(name='router').execute(f"echo '{if_conf['server-net1-p1']['addr']}\t{if_conf['server-net1-p1']['hostname']}' | sudo tee -a /etc/hosts > /dev/null
```

[22]:
```
('', '')
```

Let's make sure that all of the network interfaces are brought up:

[23]:
```
for iface in slice.get_interfaces():
    iface.ip_link_up()
```

Then, we'll add routes so that romeo knows how to reach server, and vice versa.

[24]:
```
rt_conf = [
    {"name": "romeo",   "addr": "10.10.2.0/24", "gw": "10.10.1.1"},
    {"name": "server",  "addr": "10.10.1.0/24", "gw": "10.10.2.1"},

]
for rt in rt_conf:
    slice.get_node(name=rt['name']).ip_route_add(subnet=IPv4Network(rt['addr']), gateway=rt['gw'])
```

And, we'll enable IP forwarding on the router:

[25]:
```
for n in ['router']:
    slice.get_node(name=n).execute("sudo sysctl -w net.ipv4.ip_forward=1")
```
```
net.ipv4.ip_forward = 1
```

Logging Into Resources

Observations/Issues: This has the same issue as the previously stated errors with the Fablib package. It seems that the package may have gotten corrupted or is just not properly installed.

Exercise: Log in to resources

Now, we are finally ready to log in to our resources over SSH! Run the following cells, and observe the table output - you will see an SSH command for each of the nodes in your topology.

[26]:
```
import pandas as pd
pd.set_option('display.max_colwidth', None)
ssh_str = 'ssh -i ' + slice.get_slice_private_key_file() + \
    ' -J ' + fablib.get_bastion_username() + '@' + fablib.get_bastion_public_addr() + \
    ' -F /home/fabric/work/fabric_config/ssh_config '
slice_info = [{'Name': n.get_name(), 'SSH command': ssh_str + n.get_username() + '@' + str(n.get_management_ip())} for n in slice.get_nodes()]
pd.DataFrame(slice_info).set_index('Name')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[26], line 4
      1 import pandas as pd
      2 pd.set_option('display.max_colwidth', None)
      3 ssh_str = 'ssh -i ' + slice.get_slice_private_key_file() + \
----> 4     ' -J ' + fablib.get_bastion_username() + '@' + fablib.get_bastion_public_addr() + \
      5     ' -F /home/fabric/work/fabric_config/ssh_config '
      6 slice_info = [{'Name': n.get_name(), 'SSH command': ssh_str + n.get_username() + '@' + str(n.get_management_ip())} for n in slice.get_nodes()]
      7 pd.DataFrame(slice_info).set_index('Name')

AttributeError: 'FablibManager' object has no attribute 'get_bastion_public_addr'
```

Now, you can open an SSH session on any of the nodes as follows:

- In Jupyter, from the menu bar, use File > New > Terminal to open a new terminal.
- Copy an SSH command from the table, and paste it into the terminal. (Note that each SSH command is a single line, even if the display wraps the text to a second line! When you copy and paste it, paste it all together.)

You can repeat this process (open several terminals) to start a session on each host and the router. Each terminal session will have a tab in the Jupyter environment, so that you can easily switch between them.

Now you can continue to perform the Secure applications experiment on these host sessions.

PCAP from Fabric Host

Observations/Issues: This seems to also be broken because of previous issues not being able to log into the resources. We get file not found errors for every attempt to create a PCAP file. However for whatever reason these files are still created by these scripts. These could be directly downloaded so we'll see what these files contain if anything.

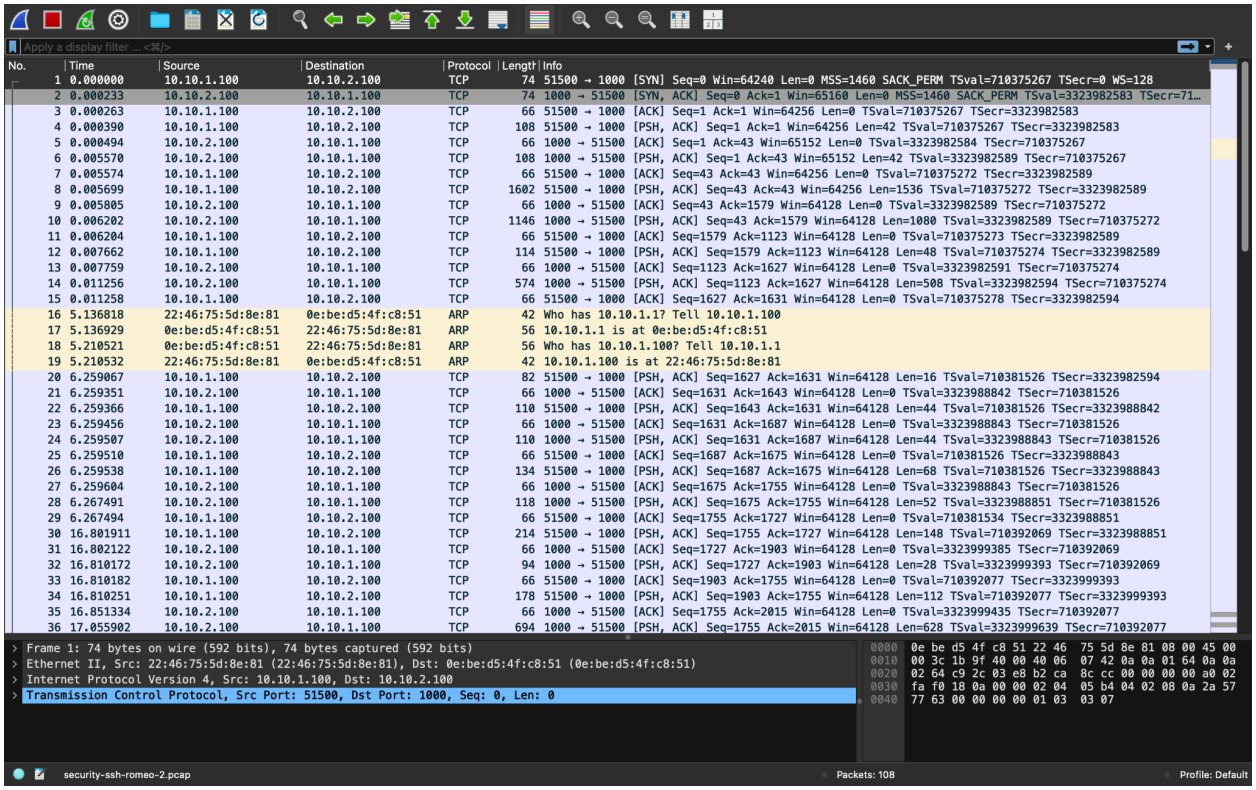
security-ftp-romeo.pcap	2 minutes ago
security-http-romeo.pcap	47 seconds ago
security-https-romeo.pcap	14 seconds ago
security-sftp-romeo.pcap	1 minute ago
security-ssh-romeo.pcap	3 minutes ago
security-telnet-romeo.pcap	4 minutes ago

SSH and Telnet Traffic PCAP Files

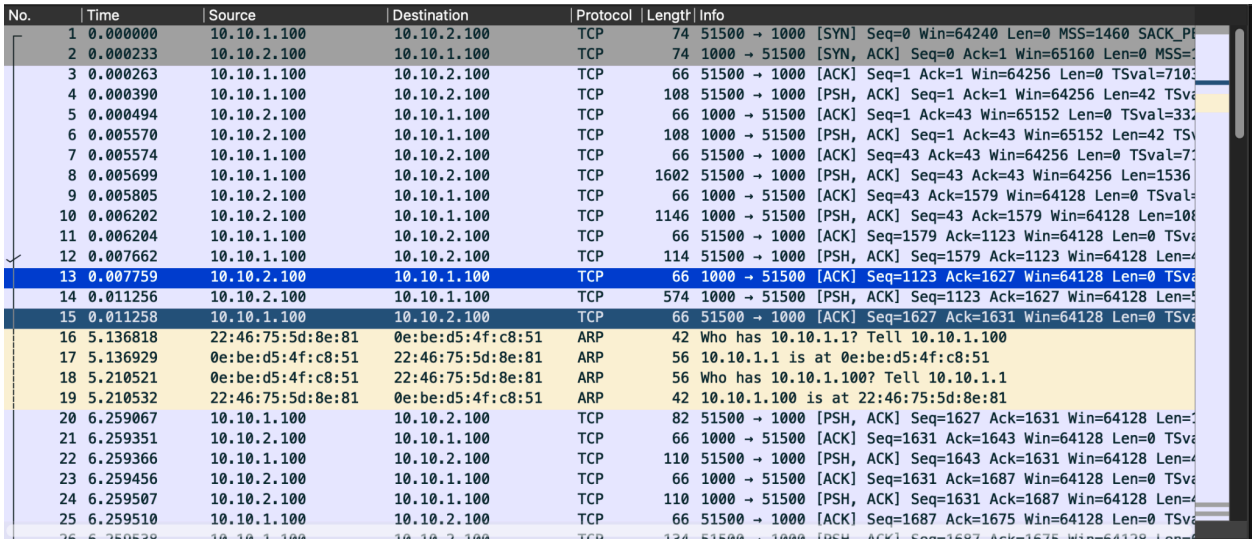
Observations/Issues: As suspected when attempting to download these files they are both empty. However I have working versions of these files so we'll analyze those.

SSH Traffic

Observations/Issues: No issues opening these working files so we should be good to start the analysis.



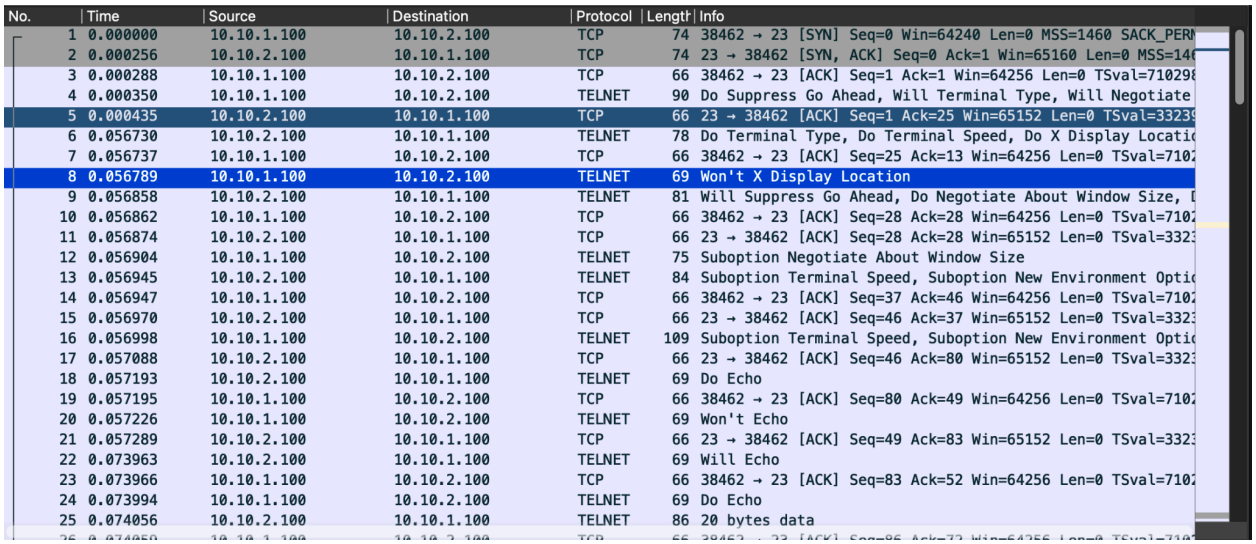
This file contains 108 frames from security-ssh-romeo. Interestingly despite port 22 being the most common port for SSH there is no TCP port 22 in this file. SSH primarily uses TCP as its transport protocol. You can see because of the overwhelming amount of TCP frames. The few ARP protocols in this script are used to discover the mapping between protocols and hardware.



SSH is a replacement for older remote shell programs like Telnet. Unlike Telnet it uses encryption which is significantly more secure. It requires the use of a key log file in order to decrypt SSH programs.

Telnet Traffic

Observations/Issues: No issues opening these working files so we should be good to start the analysis.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.1.100	10.10.2.100	TCP	74	38462 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
2	0.000256	10.10.2.100	10.10.1.100	TCP	74	23 → 38462 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=146
3	0.000288	10.10.1.100	10.10.2.100	TCP	66	38462 → 23 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=710298
4	0.000350	10.10.1.100	10.10.2.100	TELNET	90	Do Suppress Go Ahead, Will Terminal Type, Will Negotiate
5	0.000435	10.10.2.100	10.10.1.100	TCP	66	23 → 38462 [ACK] Seq=1 Ack=25 Win=65152 Len=0 TSval=33239
6	0.056730	10.10.2.100	10.10.1.100	TELNET	78	Do Terminal Type, Do Terminal Speed, Do X Display Locatio
7	0.056737	10.10.1.100	10.10.2.100	TCP	66	38462 → 23 [ACK] Seq=25 Ack=13 Win=64256 Len=0 TSval=7102
8	0.056789	10.10.1.100	10.10.2.100	TELNET	69	Won't X Display Location
9	0.056858	10.10.2.100	10.10.1.100	TELNET	81	Will Suppress Go Ahead, Do Negotiate About Window Size, D
10	0.056862	10.10.1.100	10.10.2.100	TCP	66	38462 → 23 [ACK] Seq=28 Ack=28 Win=64256 Len=0 TSval=7102
11	0.056874	10.10.2.100	10.10.1.100	TCP	66	23 → 38462 [ACK] Seq=28 Ack=28 Win=65152 Len=0 TSval=3323
12	0.056904	10.10.1.100	10.10.2.100	TELNET	75	Suboption Negotiate About Window Size
13	0.056945	10.10.2.100	10.10.1.100	TELNET	84	Suboption Terminal Speed, Suboption New Environment Optio
14	0.056947	10.10.1.100	10.10.2.100	TCP	66	38462 → 23 [ACK] Seq=37 Ack=46 Win=64256 Len=0 TSval=7102
15	0.056970	10.10.2.100	10.10.1.100	TCP	66	23 → 38462 [ACK] Seq=46 Ack=37 Win=65152 Len=0 TSval=3323
16	0.056998	10.10.1.100	10.10.2.100	TELNET	109	Suboption Terminal Speed, Suboption New Environment Optio
17	0.057088	10.10.2.100	10.10.1.100	TCP	66	23 → 38462 [ACK] Seq=46 Ack=80 Win=65152 Len=0 TSval=3323
18	0.057193	10.10.2.100	10.10.1.100	TELNET	69	Do Echo
19	0.057195	10.10.1.100	10.10.2.100	TCP	66	38462 → 23 [ACK] Seq=80 Ack=49 Win=64256 Len=0 TSval=7102
20	0.057226	10.10.1.100	10.10.2.100	TELNET	69	Won't Echo
21	0.057289	10.10.2.100	10.10.1.100	TCP	66	23 → 38462 [ACK] Seq=49 Ack=83 Win=65152 Len=0 TSval=3323
22	0.073963	10.10.2.100	10.10.1.100	TELNET	69	Will Echo
23	0.073966	10.10.1.100	10.10.2.100	TCP	66	38462 → 23 [ACK] Seq=83 Ack=52 Win=64256 Len=0 TSval=7102
24	0.073994	10.10.1.100	10.10.2.100	TELNET	69	Do Echo
25	0.074056	10.10.2.100	10.10.1.100	TELNET	86	20 bytes data

Telnet is inherently more insecure than SSH, because of its lack of encryption. They use both Telnet and TCP protocols as seen here. This instance contains 157 frames. Telnet holds its usernames and passwords in plain text. This makes the use of Telnet particularly outdated compared to SSH and should not be used in any real world applications. Though it could be used for debugging other network services.

You'll see with these Telnet protocols you can see exactly how many bytes of data are being transferred and what commands shell commands are being run like Do Echo. This is very insecure compared to SSH.

I learned the differences between SSH and Telnet. I understand why Telnet is no longer used for client server protocols because of its inherent lack of security. It makes sense why Telnet was phased out in favor of SSH in an evolving network environment. These innovations ensure SSH cannot be easily fingerprinted or packet sniffed comparatively to Telnet. Though SSH is not completely invulnerable to cyberattacks it is significantly more secure.