Scripting with Bash

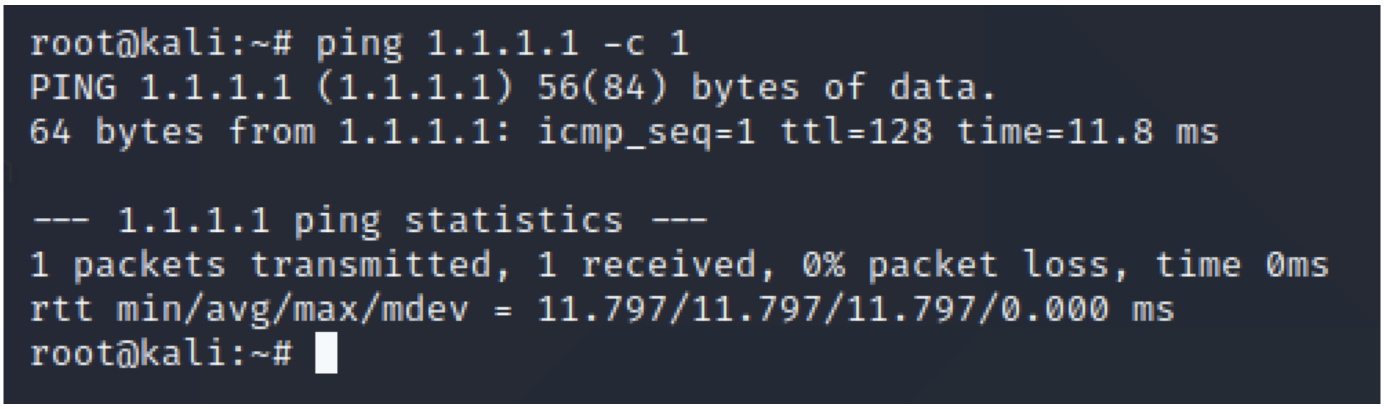
Here are some helpful notes on Bash scripting

Every bash script must start with:

`#!/bin/bash`

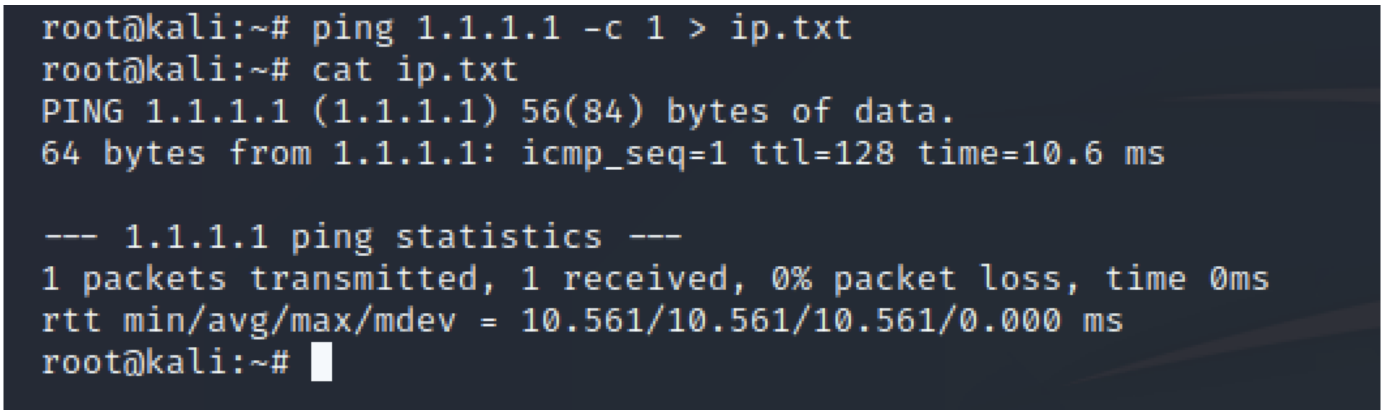
How to narrow down results

In this example, we will be working with a ping test using the "`ping <IP_address> -c 1`" command to send only one packet:



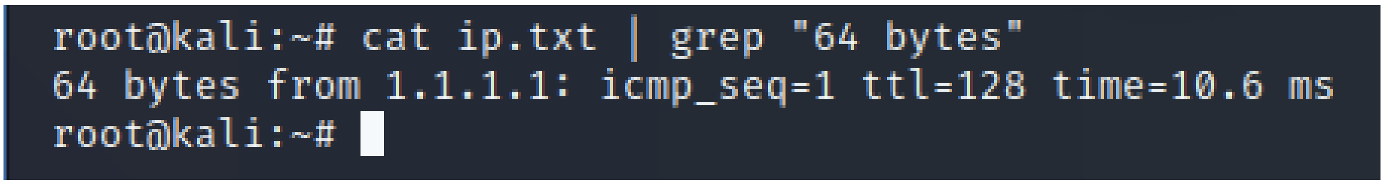
```
root@kali:~# ping 1.1.1.1 -c 1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=128 time=11.8 ms

--- 1.1.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 11.797/11.797/11.797/0.000 ms
root@kali:~# 
```

We can also send the results of the ping test to a txt file using the "`ping <IP_address> -c 1 > ip.txt`" command:



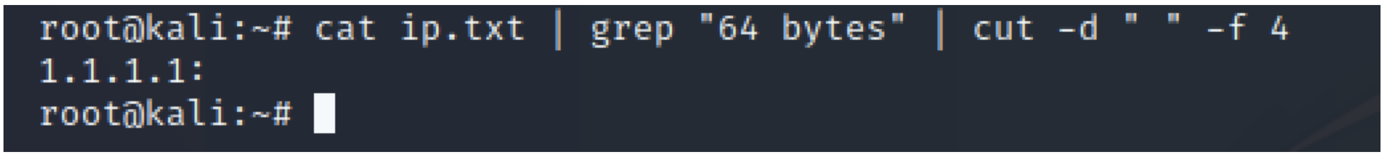
```
root@kali:~# ping 1.1.1.1 -c 1 > ip.txt
root@kali:~# cat ip.txt
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=128 time=10.6 ms

--- 1.1.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.561/10.561/10.561/0.000 ms
root@kali:~# 
```

Let's say we want to find the line(s) where a particular string such as "`64 bytes`" can be found, use the following command "`cat <filename> | grep "string"`":



```
root@kali:~# cat ip.txt | grep "64 bytes"
64 bytes from 1.1.1.1: icmp_seq=1 ttl=128 time=10.6 ms
root@kali:~# 
```

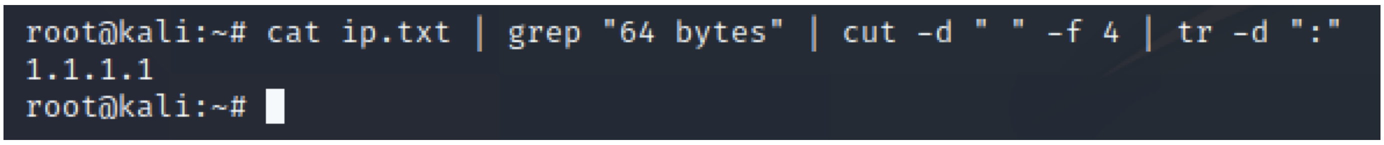
Let's say that we want to find an IP address on the line(s) that have "`64 bytes`", we can use the following command "`cat <filename> | grep "string" | cut -d " " -f 4`":



```
root@kali:~# cat ip.txt | grep "64 bytes" | cut -d " " -f 4
1.1.1.1:
root@kali:~#
```

The "`-d`" in the "`cut`" command refers to the delimiter. The delimiter is the location we are cutting on, and we can see that we have the space (`" "`). So we are cuting on the space. We can also see that we have a "`-f`". The "`-f`" refers to the field. In this case the fourth field (4) from the left. Each "string separated by a space (`" "`) is a field". So the first field would be "`64`" and the sixth field would be "`ttl=128`".

In our previous command we notice the "`:`" at the end of the 4th field (`<IP_address>`) and we do not want that. To fix the issue we will use this command "`cat <filename> | grep "string" | cut -d " " -f 4 | tr -d ":"`":



```
root@kali:~# cat ip.txt | grep "64 bytes" | cut -d " " -f 4 | tr -d ":"
1.1.1.1
root@kali:~#
```

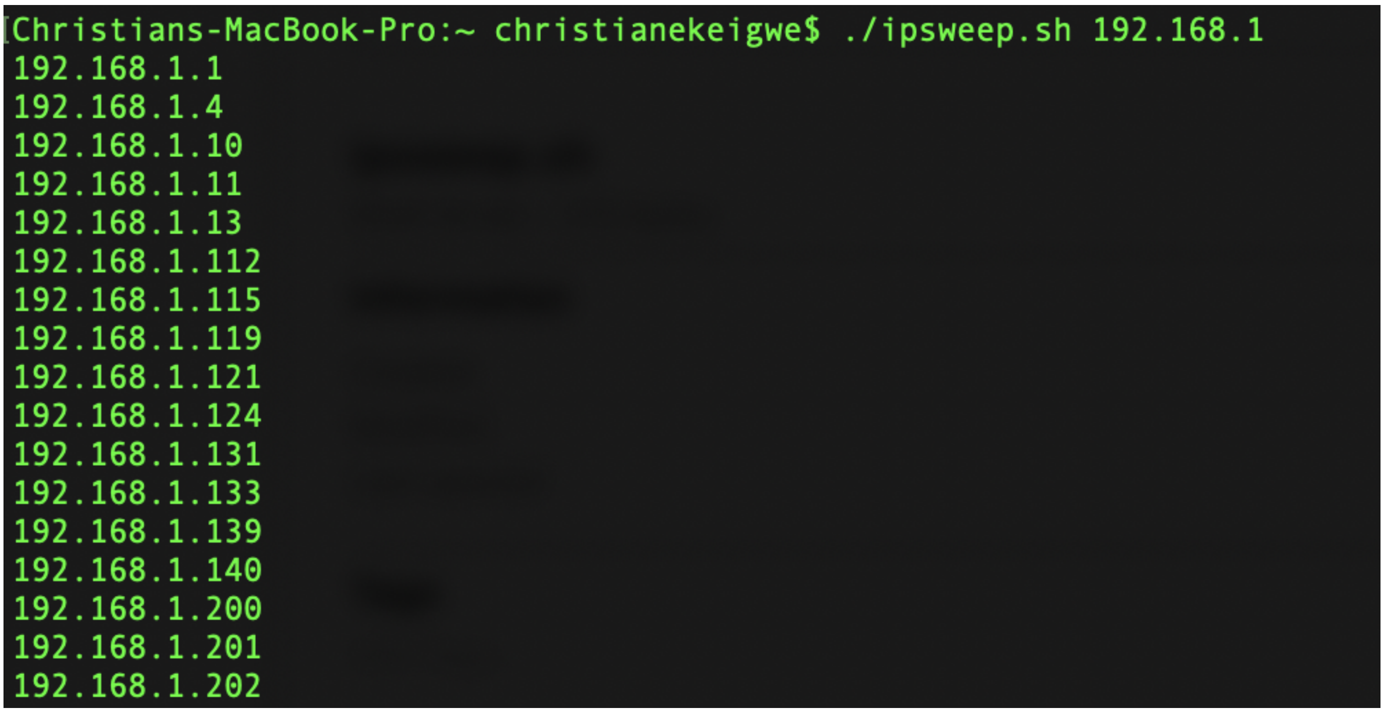
The "`tr`" command refers to "translate". The "`-d`" is also called a delimiter and the character that is being delimited is the "`:`" character.

Let's write a Bash script to scan our network for IP addresses:

1. `gedit` or `nano ipsweep.sh`
2. Enter the following script:

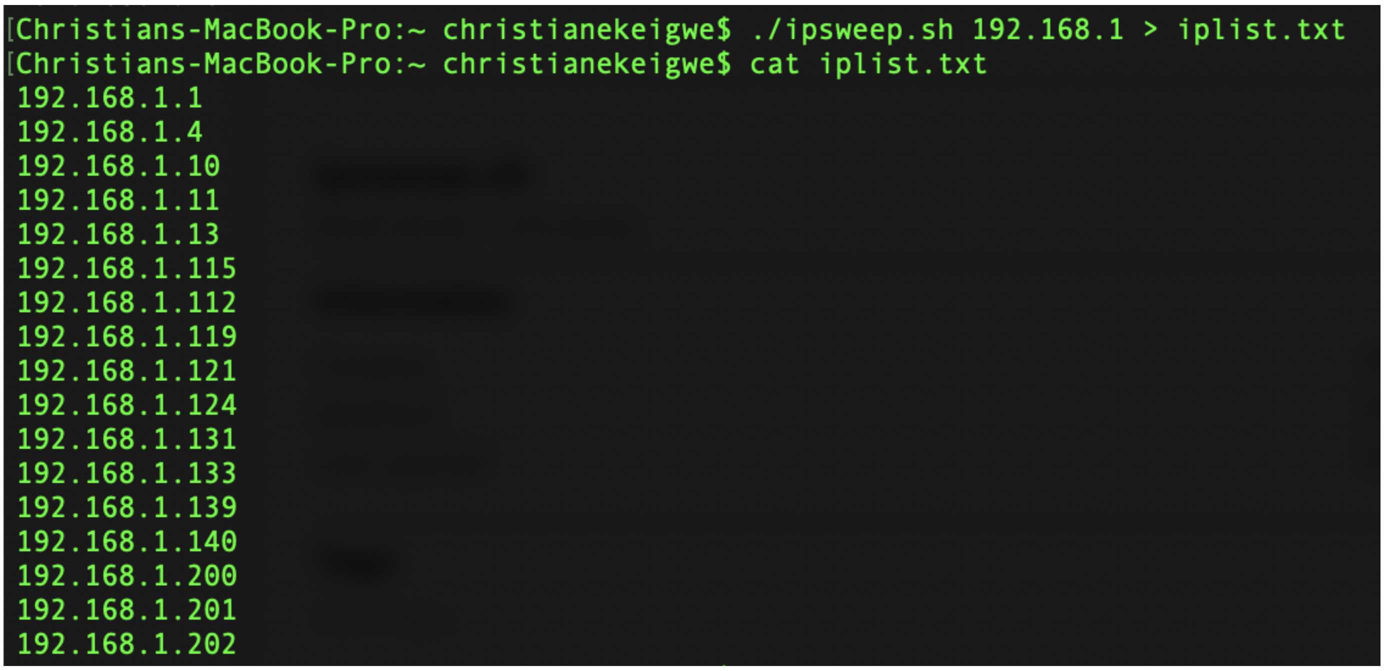
```
#!/bin/bash
if [ "$1" == "" ]
then
echo "You forgot an IP address!"
echo "Syntax: ./ipsweep.sh 192.168.1"

else
for ip in `seq 1 254`; do
ping -c 1 $1.$ip | grep "64 bytes" | cut -d " " -f 4 | tr -d ":" &
done
fi
```

3. Exit the editor, and enter "`chmod +x ipsweep.sh`" to convert the script to an exceutable file
4. Use the "`./ipsweep.sh <first 3 IP_address octets>`" to find the possible IP addresses as shown below:



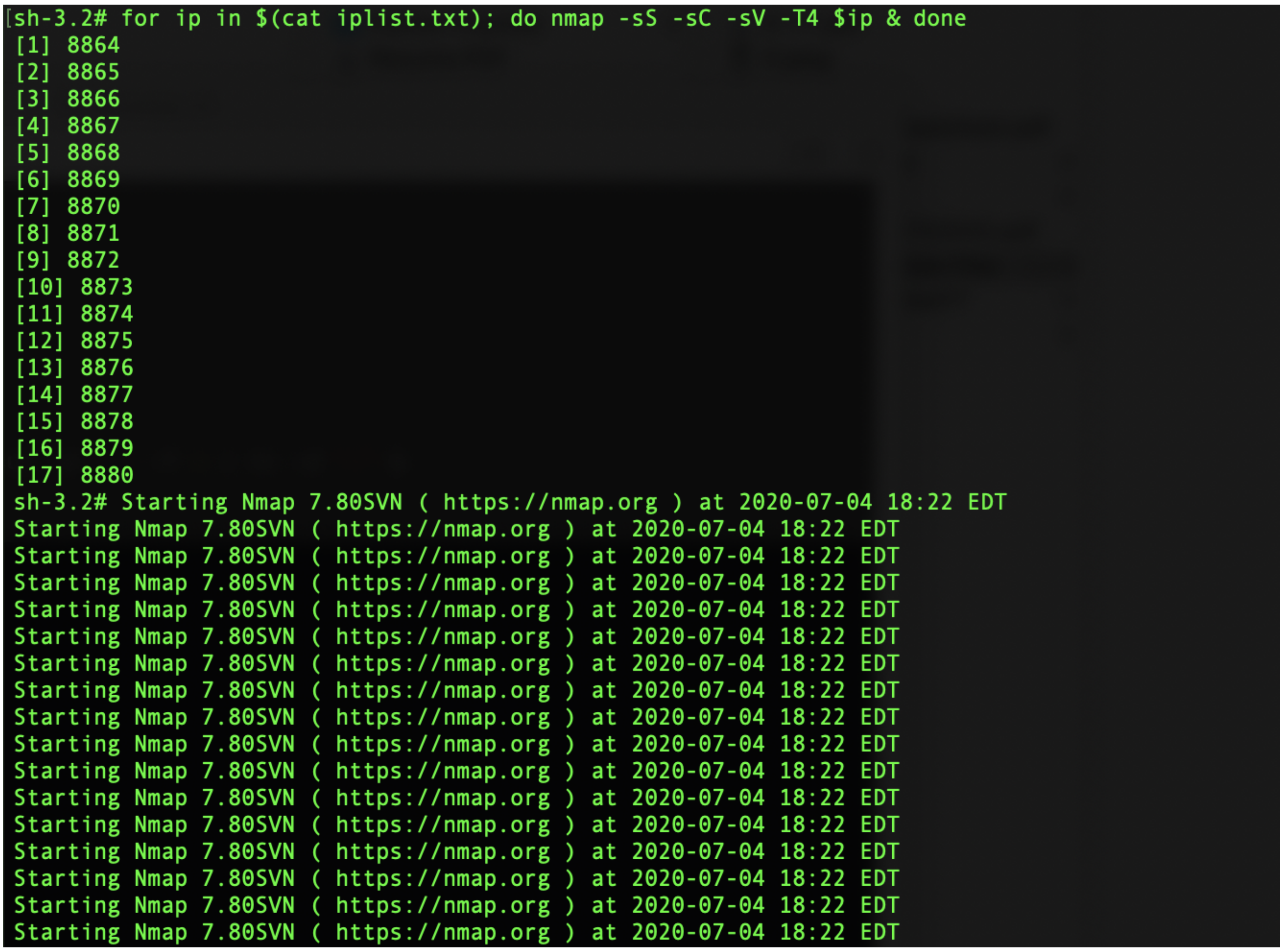
```
[Christians-MacBook-Pro:~ christianekeigwe$ ./ipsweep.sh 192.168.1
 192.168.1.1
 192.168.1.4
 192.168.1.10
 192.168.1.11
 192.168.1.13
 192.168.1.112
 192.168.1.115
 192.168.1.119
 192.168.1.121
 192.168.1.124
 192.168.1.131
 192.168.1.133
 192.168.1.139
 192.168.1.140
 192.168.1.200
 192.168.1.201
 192.168.1.202
```

We can also save the scanned IP adresses to a text file using the "`./ipsweep.sh <first 3 IP_address octets> > iplist.txt`" command:



```
[Christians-MacBook-Pro:~ christianekeigwe$ ./ipsweep.sh 192.168.1 > iplist.txt
[Christians-MacBook-Pro:~ christianekeigwe$ cat iplist.txt
 192.168.1.1
 192.168.1.4
 192.168.1.10
 192.168.1.11
 192.168.1.13
 192.168.1.115
 192.168.1.112
 192.168.1.119
 192.168.1.121
 192.168.1.124
 192.168.1.131
 192.168.1.133
 192.168.1.139
 192.168.1.140
 192.168.1.200
 192.168.1.201
 192.168.1.202
```

Let's say we want to perform an Nmap scan on the iplist.txt file, we can use the following command "`for ip in $(cat iplist.txt); do nmap -sS -sC -sV -T4 $ip & done`" as shown below:



```
[sh-3.2# for ip in $(cat iplist.txt); do nmap -sS -sC -sV -T4 $ip & done
[1] 8864
[2] 8865
[3] 8866
[4] 8867
[5] 8868
[6] 8869
[7] 8870
[8] 8871
[9] 8872
[10] 8873
[11] 8874
[12] 8875
[13] 8876
[14] 8877
[15] 8878
[16] 8879
[17] 8880
sh-3.2# Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
Starting Nmap 7.80SVN ( https://nmap.org ) at 2020-07-04 18:22 EDT
```