Bash Scripting

BY: MIA WEBER

BACKGROUND

The script that I chose to write is a script that utilizes cron in order to automate the execution of the script every minute. When the script executes it checks all the open listening ports on the computer in order to verify that no unnecessary ports are open. It compares the ports that are open with a config file provided by the user that contains the ports that the user approves to be open. If it determines that a port is open that shouldn't be, it will close that port and report everything to the user. Command line flags are used in order to specify whether the script should simply log the port information or if it should kill the process that is running on the incorrectly open ports. This allows for increased customization of the tasks the script will perform.

PURPOSE

The purpose of this bash script is to automate the process of checking for open ports. A listening port is a network port that an application listens on. A user can get a list of all the open listening ports by querying the network stack using tools like **ss, netstat,** or **isolf,** among others**.** This script makes the user's job easier by verifying that only the correct ports are open and can take deliberate action to rectify any errors.

JUSTIFICATION

A system should only have the ports necessary for application functionality open and all others closed. Ensuring that the correct ports are open helps with firewall configuration and network connectivity troubleshooting. In addition, maintaining port control and frequently checking the status of open listening ports is an essential security measure. This script is helpful in maintaining and adhering to good security practices to ensure that there aren't unmonitored ways to access and interfere with a system. By utilizing cron to run this script every minute and log the results, the user can be confident in the state of their machine and can know with certainty how many ways in and out of the system there are.

POTENTIAL USER(S)

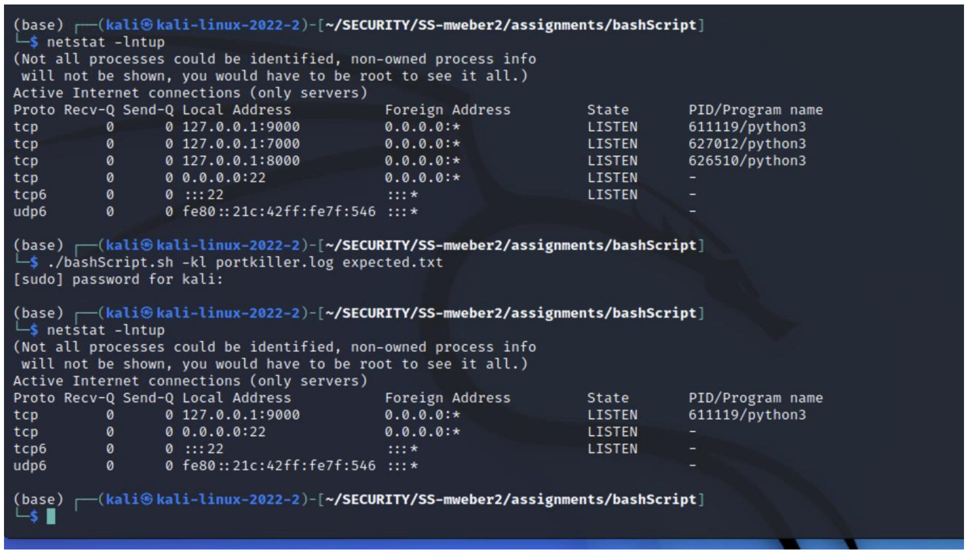
This bash script would be particularly useful for system administrators who need to ensure that a system is closed and all entrances and exits are being monitored and controlled. Automating this process would help people who need to manage large servers or a lot of servers and don't have the time to continuously ensure that applications or potentially malicious parties are doing things they should not be doing in your systems.

HOW TO

The first thing that is important when using this bash script is to ensure that the script's permissions are set as executable. Without the '-x' flag set for the correct user, the computer will only recognize the file as a .txt file and won't know that it contains a script that is meant to be executed. In order to change these permissions, the following command can be run: **chmod +x <NAMEOFSCRIPT>.sh.** After the script has been made executable, the following command can be executed: **./<NAMEOFSCRIPT>.sh.** Because the script is automated with cronjobs it will continue to execute every hour without the need for user intervention. The script will then report its findings and log its actions for the user's reference in accordance with the command line flags that were set by the user. The –**k** flag tells the script that if it identifies incorrectly open ports, it should go ahead and kill the processes that are running on those ports, causing them to close. The **–h** flag outputs the help menu for the script. The –**l** flag tells the script that it should log its findings to the provided log file. The name of the log file must be provided immediately following the flag. After all the flags are provided then the name of the comparison config file can be provided. This is the file that contains the list of approved ports to be open. The script will not run correctly without this file being provided as the last argument from the command line. An example of a correct way to run the script would be: **./bashScript.sh -kl logfile.log comparefile.config**

Manual Testing

In order to manually test the script, I executed the script manually as well as ensured that cron was correctly automating the script execution. In order to ensure that the cronjob was working as intended, I waited for a few "cycles" in order to ensure that everything was executed as expected beyond just the first execution. The other basic script called verify.sh also has a corresponding cron job which checks to make sure that the log file is being updated in correct intervals and therefore that the script's cron job is working as intended. Although it doesn't match the practical applications of this script, for testing purposes I opened python3 http.server processes on ports like 7000, 8000, and 9000. In reality it wouldn't matter that processes were running on ports that high, however running python3 processes on those ports makes manually testing the script easier. The following are screenshots of the log file and the results of the netstat –lntup command after the execution of the script. Below each screenshot it is specified which command line flags were set, which ports were opened prior to the script execution, the contents of the compare file and therefore which ports it should kill.

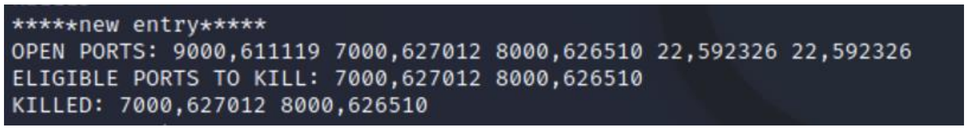


```
(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ netstat -lntup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      611119/python3
tcp        0      0 127.0.0.1:7000          0.0.0.0:*               LISTEN      627012/python3
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN      626510/python3
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp6       0      0 fe80::21c:42ff:fe7f:546 :::*                                -

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ ./bashScript.sh -kl portkiller.log expected.txt
[sudo] password for kali:

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ netstat -lntup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      611119/python3
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp6       0      0 fe80::21c:42ff:fe7f:546 :::*                                -

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ 
```

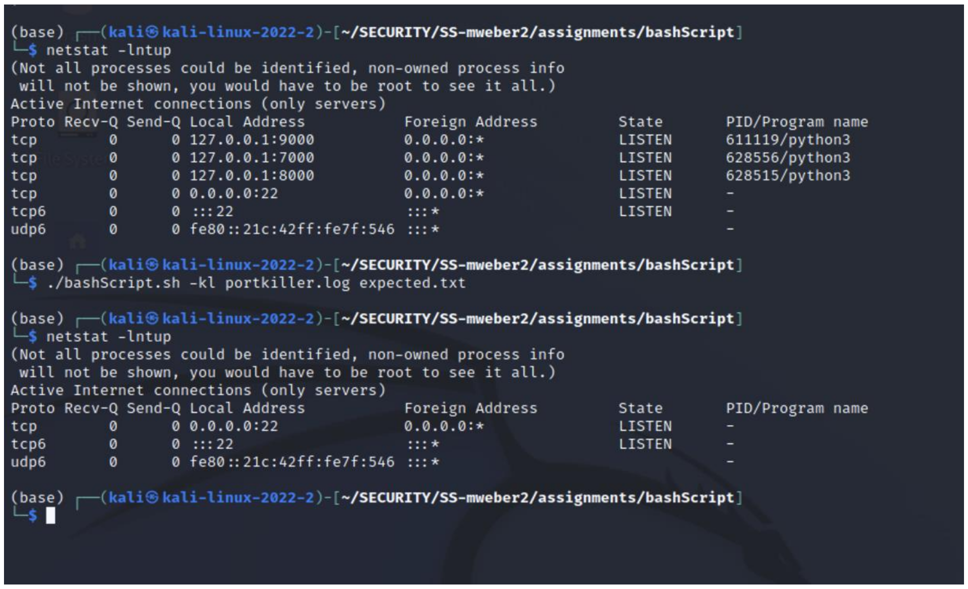


```
*****new entry*****
OPEN PORTS: 9000,611119 7000,627012 8000,626510 22,592326 22,592326
ELIGIBLE PORTS TO KILL: 7000,627012 8000,626510
KILLED: 7000,627012 8000,626510
```

Open ports = 7000, 8000, 9000, 22, 22

Ports listed in compare file = 22, 22, 9000

Ports it should kill = 7000, 8000

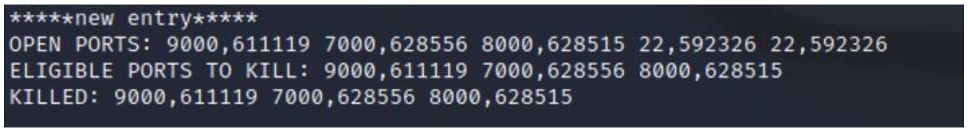


```
(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ netstat -lntup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      611119/python3
tcp        0      0 127.0.0.1:7000          0.0.0.0:*               LISTEN      628556/python3
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN      628515/python3
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp6       0      0 fe80::21c:42ff:fe7f:546 :::*                                -

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ ./bashScript.sh -kl portkiller.log expected.txt

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ netstat -lntup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp6       0      0 fe80::21c:42ff:fe7f:546 :::*                                -

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ █
```

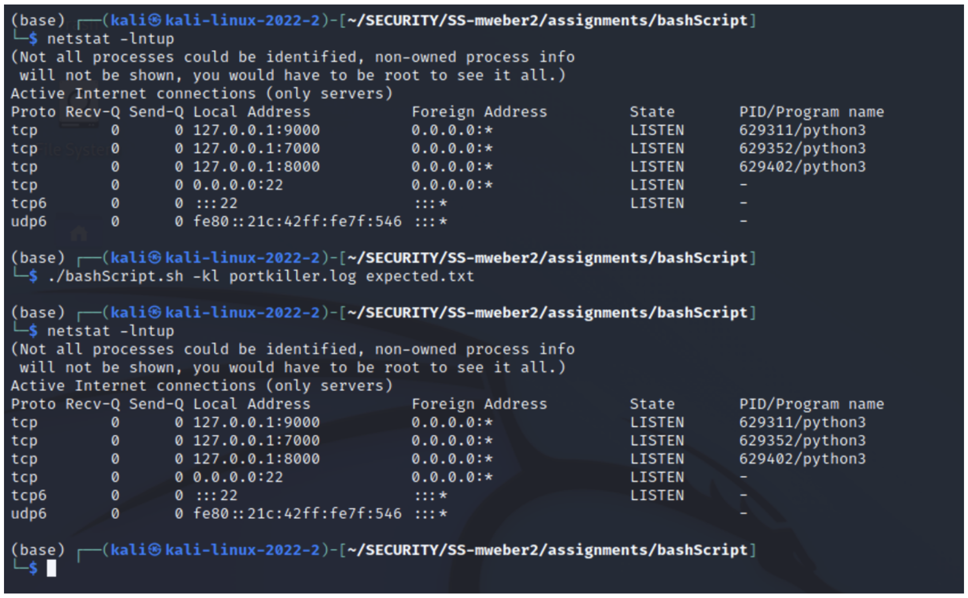


```
*****new entry*****
OPEN PORTS: 9000,611119 7000,628556 8000,628515 22,592326 22,592326
ELIGIBLE PORTS TO KILL: 9000,611119 7000,628556 8000,628515
KILLED: 9000,611119 7000,628556 8000,628515
```

Open ports = 7000, 8000, 9000, 22, 22

Ports listed in compare file = 22, 22

Ports it should kill = 7000, 8000, 9000

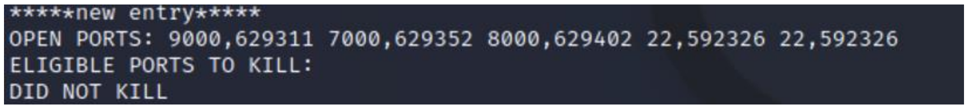


```
(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ netstat -lntup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      629311/python3
tcp        0      0 127.0.0.1:7000          0.0.0.0:*               LISTEN      629352/python3
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN      629402/python3
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp6       0      0 fe80::21c:42ff:fe7f:546 :::*                                -

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ ./bashScript.sh -kl portkiller.log expected.txt

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ netstat -lntup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      629311/python3
tcp        0      0 127.0.0.1:7000          0.0.0.0:*               LISTEN      629352/python3
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN      629402/python3
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp6       0      0 fe80::21c:42ff:fe7f:546 :::*                                -

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ █
```

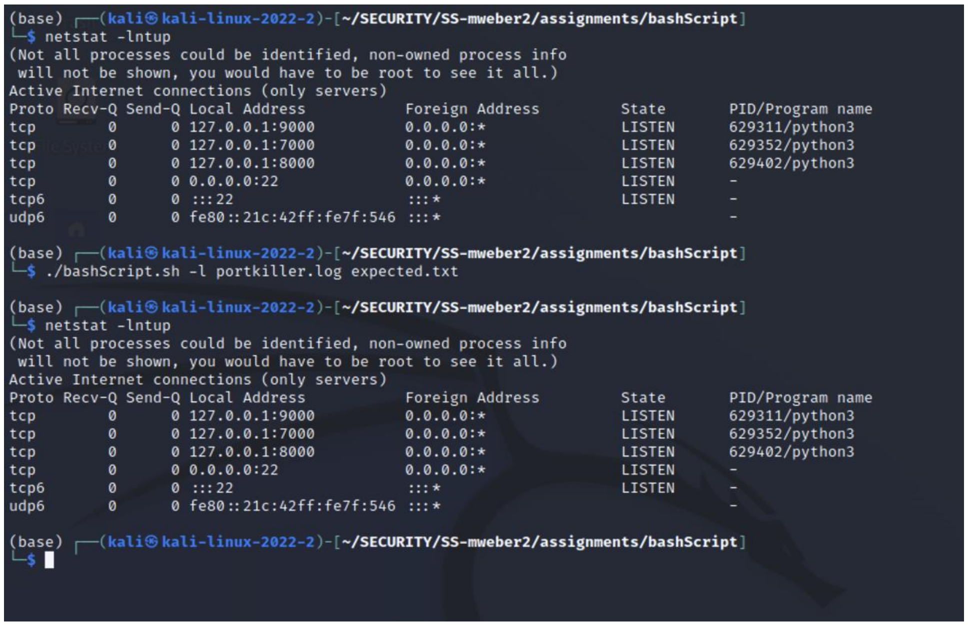


```
*****new entry*****
OPEN PORTS: 9000,629311 7000,629352 8000,629402 22,592326 22,592326
ELIGIBLE PORTS TO KILL:
DID NOT KILL
```

Open ports = 7000, 8000, 9000, 22, 22

Ports listed in compare file = 22, 22 7000 8000 9000

Ports it should kill = N/A

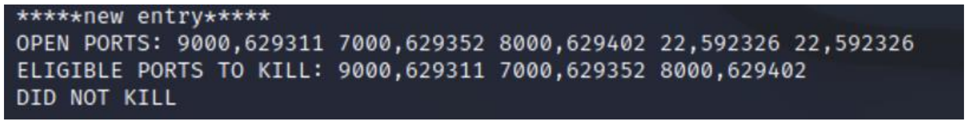


```
(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ netstat -lntup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      629311/python3
tcp        0      0 127.0.0.1:7000          0.0.0.0:*               LISTEN      629352/python3
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN      629402/python3
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp6       0      0 fe80::21c:42ff:fe7f:546 :::*                                -

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ ./bashScript.sh -l portkiller.log expected.txt

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ netstat -lntup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN      629311/python3
tcp        0      0 127.0.0.1:7000          0.0.0.0:*               LISTEN      629352/python3
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN      629402/python3
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp6       0      0 fe80::21c:42ff:fe7f:546 :::*                                -

(base) ┌──(kali㉿kali-linux-2022-2)-[~/SECURITY/SS-mweber2/assignments/bashScript]
└─$ 
```



```
*****new entry*****
OPEN PORTS: 9000,629311 7000,629352 8000,629402 22,592326 22,592326
ELIGIBLE PORTS TO KILL: 9000,629311 7000,629352 8000,629402
DID NOT KILL
```

Open ports = 7000, 8000, 9000, 22, 22

Ports listed in compare file = 22, 22

Ports it should kill = N/A (-k flag is not set. Should not kill ports 7000, 8000, or 9000)

REFERENCES

https://stackoverflow.com/questions/4247932/how-to-parse-netstat-command-in-order-to-get-process-name-and-pid-from-it

https://stackoverflow.com/questions/27002915/parse-netstat-command-to-get-ports

https://regexr.com/

https://www.thegeekdiary.com/cron-script-does-not-execute-as-expected-from-crontab-troubleshoot/