

# Creating A TCP Port Scanner With Bash

Every penetration tester needs to know how to write code in order to automate a task or to develop a tool that will perform a specific activity that it might be needed in a penetration test. So in this tutorial we will see how we can create simple tcp port scanner in bash. Of course the best tool for this job is Nmap but the scope of this post is to familiarize with bash scripting and to inspire the readers to develop their own tools.

This port scanner is going to be created in bash so in the first line we will have to write the following:

```
#!/bin/bash
```

The shebang (#!/) is used because it will instruct the operating system that what comes next will be the interpreter of the script which in this case is /bin/bash.

Next we have to assign the values from the argument to the suitable variables. We can also use the # sign when we will want to put comments in the code. So the next lines of our code will be like the following:

```
#Defining the variables
```

```
IP=$1
```

```
firstport=$2
```

```
lastport=$3
```

As an input our port scanner will take an IP (which it will be the IP of the target that we want to scan), the port that we want to scan first which has defined as firstport and the port that we want to scan last (lastport).

The next thing that we have to do is to use a function. Functions in general allow us to take piece of code and to use it again and again without the need of rewriting. We will call our function portscan because it will test the ports that will specify to see if they are open. Then we can set a for loop in order to loop from the first port to the last port. You can see the code below:

```
function portscan
```

```
{
```

```
for ((counter=$firstport; counter<=$lastport; counter++))
```

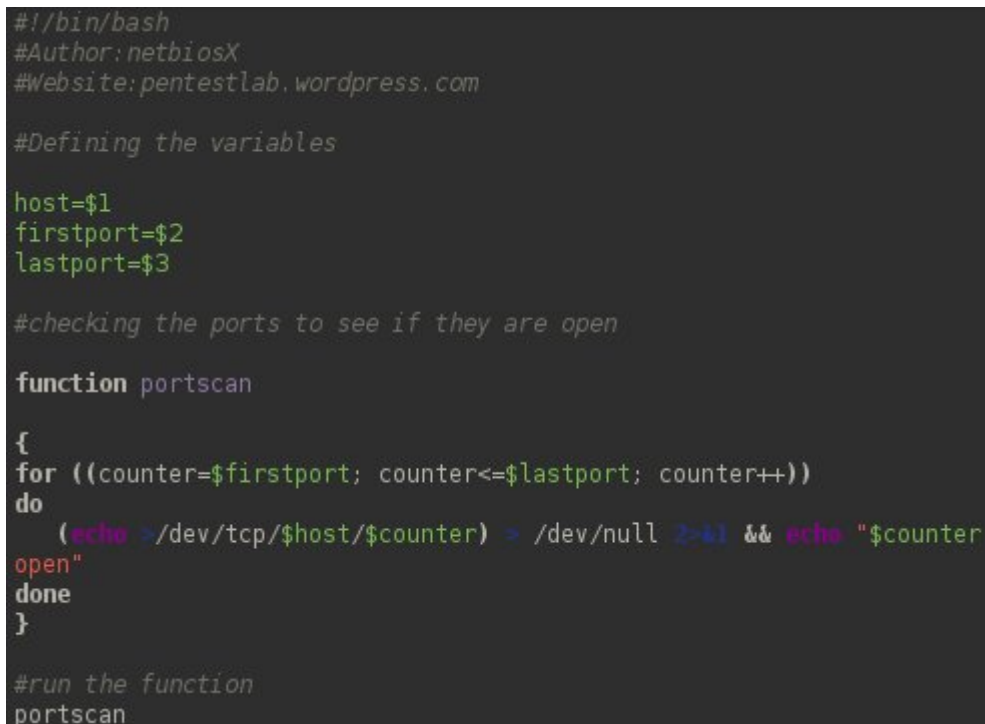
Then we will use a do loop and we will try to echo each port of the IP that we are scanning. We will send the output of this echo to the /dev/null. The > sign is used in order to make this redirection. Then we will redirect it again to &1 and we will use the &&

operator (and) in order to echo the string <port number> open to the console. Below is the piece of the code that we have to write:

```
do
(echo >/dev/tcp/$IP/$counter) > /dev/null 2>&1 && echo "$counter
open"
done
}
```

**portscan**

You can see the complete code of the tcp port scanner in the image below:



```
#!/bin/bash
#Author: netbiosX
#Website: pentestlab.wordpress.com

#Defining the variables

host=$1
firstport=$2
lastport=$3

#checking the ports to see if they are open

function portscan
{
for ((counter=$firstport; counter<=$lastport; counter++))
do
    (echo >/dev/tcp/$host/$counter) > /dev/null 2>&1 && echo "$counter
open"
done
}

#run the function
portscan
```

Source Code

Before we attempt to run the script we need to make it executable. In order to do this we need to be in the directory that contains the file and to type the command in terminal

**chmod u+x pentestlab\_scanner**

This command will add execute permissions for the user that is the owner of the file. Now that the file is executable we can run it with the command **./pentestlab\_scanner IP**

**Firstport Lastport**

```
root@encode: ~/Desktop# chmod u+x pentestlab_scanner
root@encode: ~/Desktop# ./pentestlab_scanner 127.0.0.1 1 1000
80
open
443
open
631
open
902
open
root@encode: ~/Desktop#
```

Bash TCP Scanner Demonstration

## Conclusion

As we saw creating a port scanner in bash is very easy if we know the basics of this scripting language. The main function of this port scanner is to check only if the tcp ports of a host are open. The user can select from which port the scan will start and in which port it will end. Of course the port scanner can be improved in many ways. For example implementing additional functions like scanning and the UDP ports as well and allowing the user to scan multiple hosts.