MAY 2020

# SCRIPTING FOR SECURITY

## ASSESSMENT

ESZTER KARAJZ

[redacted]

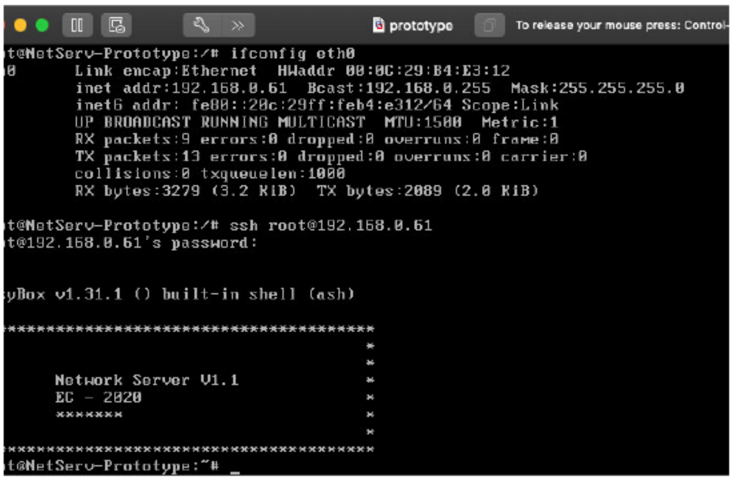
# SCRIPTING FOR SECURITY

## TASK

You have been asked to create a menu system that will perform the following tasks, place the menu script in the home directory of root (/root/) and call it, helpdesk_menu.sh.

1) Display the available interfaces on the router
2) Select an interface and display the IP address
3) Select an interface and display the MAC address
4) Display the uptime of the router
5) Display the disk usage and CPU utilisation of the router
6) Restart the networking service on the router
7) Restart the ssh service on the router
8) Enter a port number and run an nmap scan on this port
9) Return the value of the nmap scan
10) Display the different firewall zones
11) Display the config for each of the firewall zones
12) Reboot the router

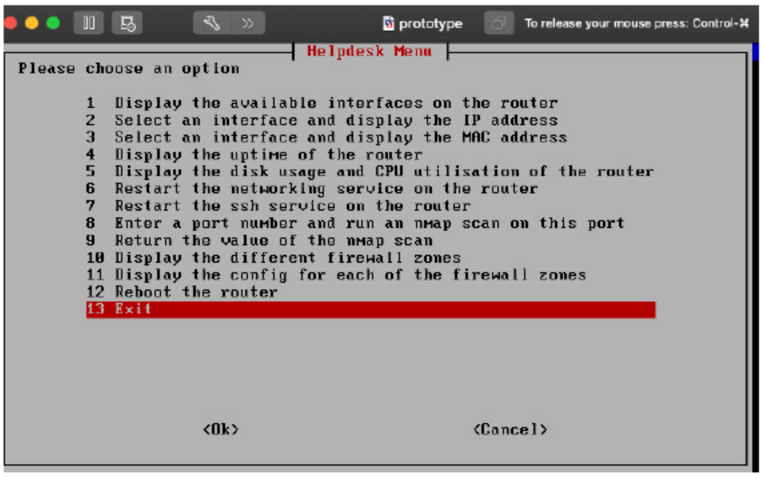
## MAIN MENU



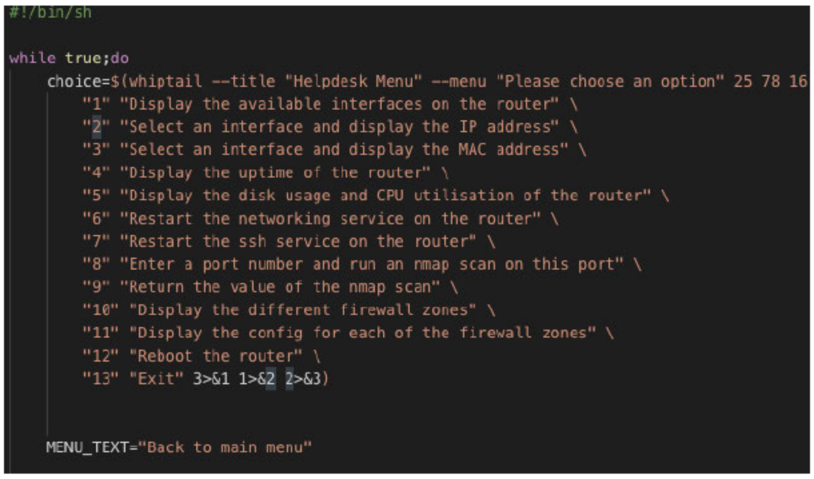
```
t@NetServ-Prototype:/# ifconfig eth0
0         Link encap:Ethernet  HWaddr 00:0C:29:B4:E3:12
          inet addr:192.168.0.61  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feb4:e312/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3279 (3.2 KiB)  TX bytes:2089 (2.0 KiB)

t@NetServ-Prototype:/# ssh root@192.168.0.61
t@192.168.0.61's password:


yBox v1.31.1 () built-in shell (ash)

**************************************
                                     *
                                     *
     Network Server V1.1             *
     EC - 2020                       *
     *******                         *
                                     *
**************************************
t@NetServ-Prototype:~# _
```

After opening the prototype.ova file I was able to connect to the system using the ssh root@ command and logged in with the password (root).

I run the program with the sh helpdesk_menu.sh command and I am presented with the main menu which has 13 options.



```
                        Helpdesk Menu
Please choose an option

        1  Display the available interfaces on the router
        2  Select an interface and display the IP address
        3  Select an interface and display the MAC address
        4  Display the uptime of the router
        5  Display the disk usage and CPU utilisation of the router
        6  Restart the networking service on the router
        7  Restart the ssh service on the router
        8  Enter a port number and run an nmap scan on this port
        9  Return the value of the nmap scan
        10 Display the different firewall zones
        11 Display the config for each of the firewall zones
        12 Reboot the router
        13 Exit


             <Ok>                          <Cancel>
```

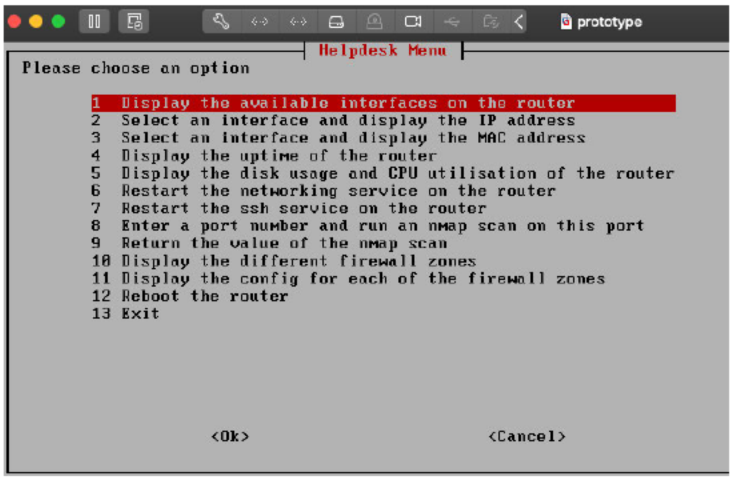


```
#!/bin/sh

while true;do
    choice=$(whiptail --title "Helpdesk Menu" --menu "Please choose an option" 25 78 16
        "1" "Display the available interfaces on the router" \
        "2" "Select an interface and display the IP address" \
        "3" "Select an interface and display the MAC address" \
        "4" "Display the uptime of the router" \
        "5" "Display the disk usage and CPU utilisation of the router" \
        "6" "Restart the networking service on the router" \
        "7" "Restart the ssh service on the router" \
        "8" "Enter a port number and run an nmap scan on this port" \
        "9" "Return the value of the nmap scan" \
        "10" "Display the different firewall zones" \
        "11" "Display the config for each of the firewall zones" \
        "12" "Reboot the router" \
        "13" "Exit" 3>&1 1>&2 2>&3)


    MENU_TEXT="Back to main menu"
```

For the main menu I am using the whiptail --menu command to be able to display all the options needed in this format and a while loop to make sure it exits back to the main menu rather than quitting the script.
Instead of a multilevel if-then-else-fi statement I decided to use the case statement.
The user's choice will be saved in the 'choice' variable.
This screenshot also shows the MENU_TEXT variable which I am using for the "Back to main menu button".

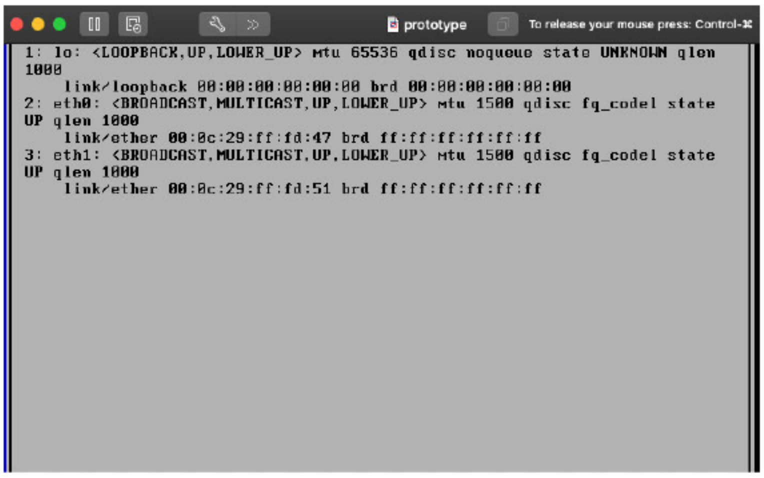
# TASK 1

## 1) DISPLAY THE AVAILABLE INTERFACES ON THE ROUTER



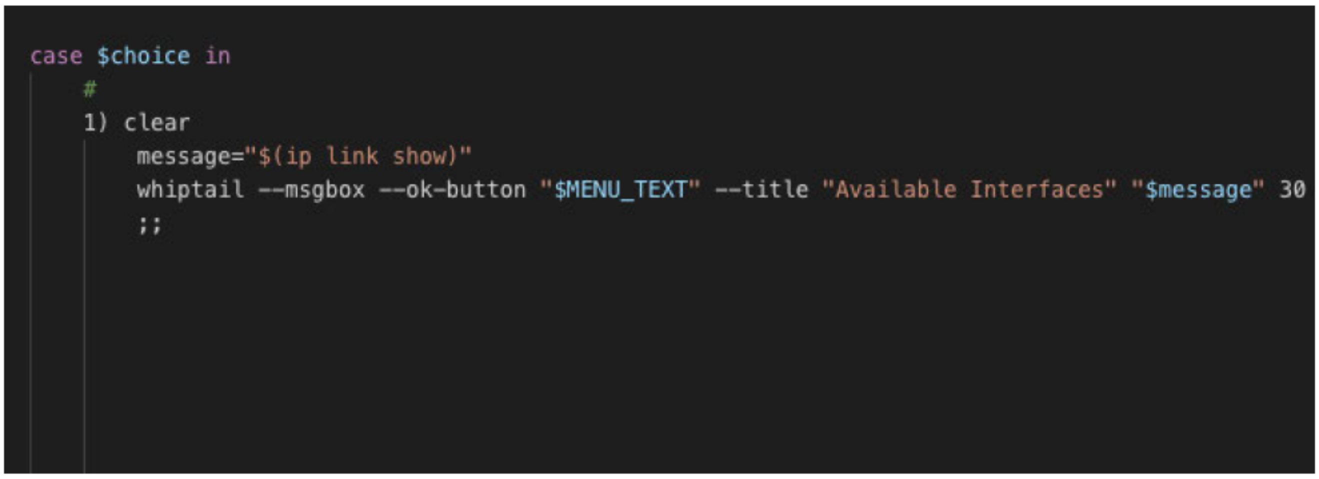
```
Please choose an option

        1  Display the available interfaces on the router
        2  Select an interface and display the IP address
        3  Select an interface and display the MAC address
        4  Display the uptime of the router
        5  Display the disk usage and CPU utilisation of the router
        6  Restart the networking service on the router
        7  Restart the ssh service on the router
        8  Enter a port number and run an nmap scan on this port
        9  Return the value of the nmap scan
        10 Display the different firewall zones
        11 Display the config for each of the firewall zones
        12 Reboot the router
        13 Exit
```

After choosing the first option the program displays the available interfaces on the router.



```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP qlen 1000
    link/ether 00:0c:29:ff:fd:47 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP qlen 1000
    link/ether 00:0c:29:ff:fd:51 brd ff:ff:ff:ff:ff:ff
```

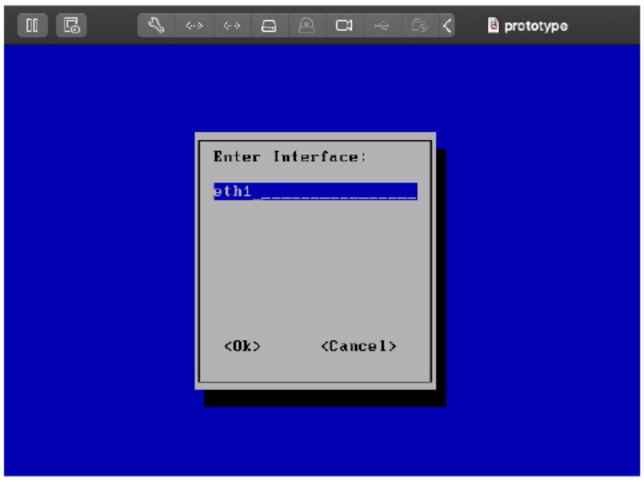
The 'ip link' command is applied to display and modify network interfaces. The result is saved in a variable and used in the whiptail script.



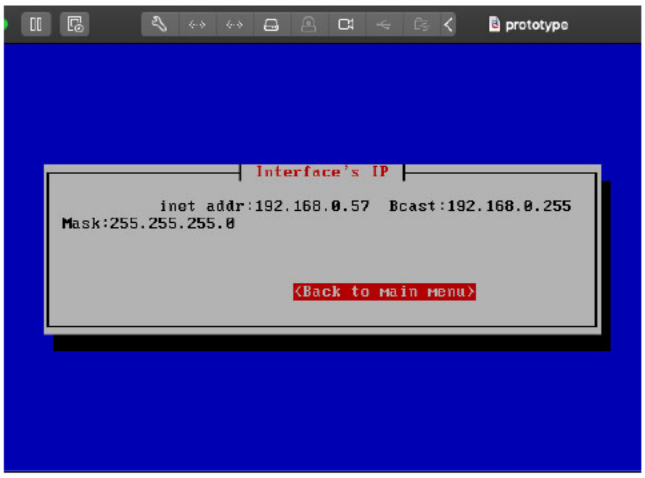
```
case $choice in
    #
    1) clear
        message="$(ip link show)"
        whiptail --msgbox --ok-button "$MENU_TEXT" --title "Available Interfaces" "$message" 30
        ;;
```

# TASK 2

## 2) SELECT AN INTERFACE AND DISPLAY THE IP ADDRESS

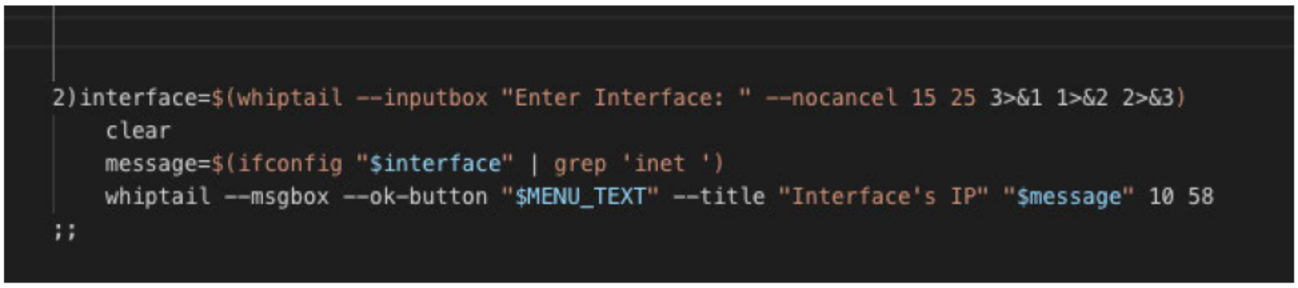


For this task the user is prompted to enter an interface.
When the Ok button is pressed another message box appears showing the IP address.



```
          inet addr:192.168.0.57  Bcast:192.168.0.255
Mask:255.255.255.0
```

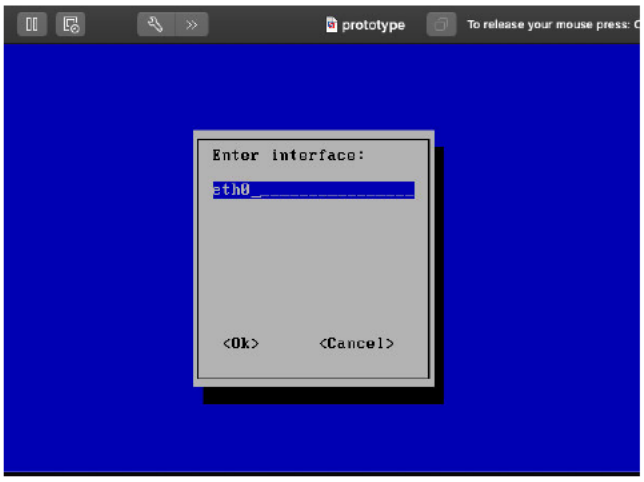
With the whiptail --inputbox command I am able to display the input box part of this task. The ifconfig eth1 | grep 'inet ' command displays the ip address of eth1. I am storing the user input in a variable called 'interface' and that's what I am using to place the user input into the command.



```
2)interface=$(whiptail --inputbox "Enter Interface: " --nocancel 15 25 3>&1 1>&2 2>&3)
    clear
    message=$(ifconfig "$interface" | grep 'inet ')
    whiptail --msgbox --ok-button "$MENU_TEXT" --title "Interface's IP" "$message" 10 58
;;
```

# TASK 3

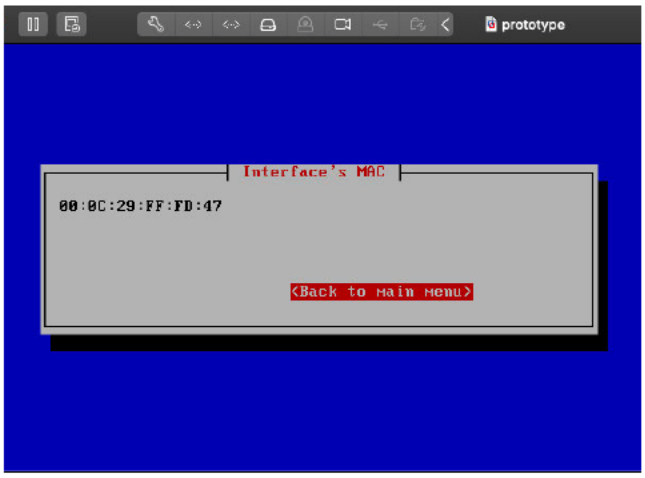
## 3) SELECT AN INTERFACE AND DISPLAY THE MAC ADDRESS



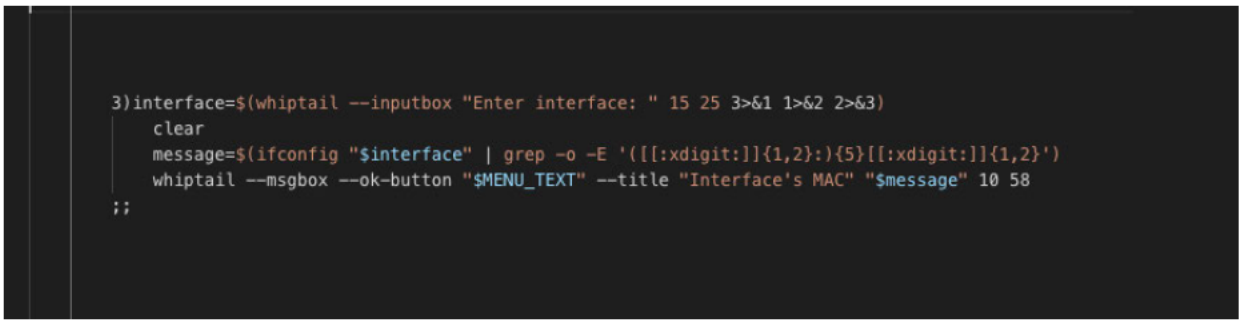
```
Enter interface:
eth0_
<Ok>          <Cancel>
```

The user is asked to enter an interface again but here I am looking for the MAC address of the interface.

The syntax is very similar to the previous task's but here I used a regular expression to filter the results to a certain format which only displays the MAC address.



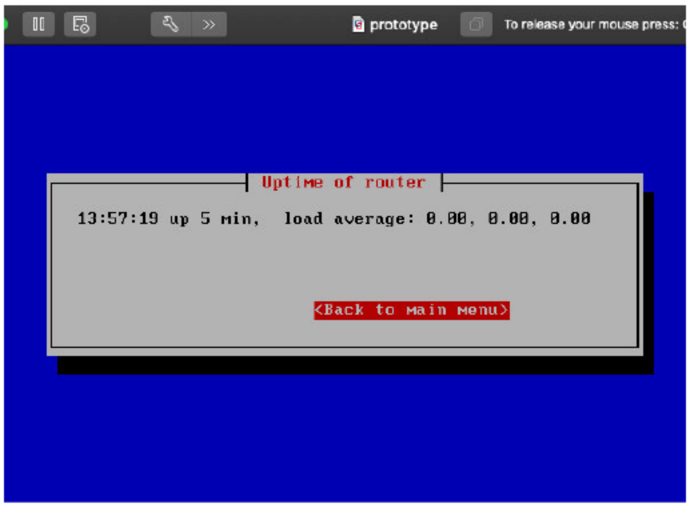
```
Interface's MAC
00:0C:29:FF:FD:47
<Back to Main Menu>
```



```
3)interface=$(whiptail --inputbox "Enter interface: " 15 25 3>&1 1>&2 2>&3)
    clear
    message=$(ifconfig "$interface" | grep -o -E '([[:xdigit:]]{1,2}:){5}[[:xdigit:]]{1,2}')
    whiptail --msgbox --ok-button "$MENU_TEXT" --title "Interface's MAC" "$message" 10 58
;;
```

# TASK 4

## 4) DISPLAY THE UPTIME OF THE ROUTER



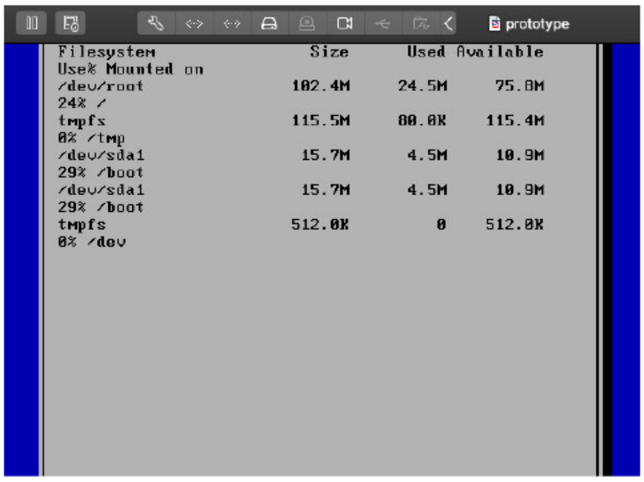
```
Uptime of router

13:57:19 up 5 min,  load average: 0.00, 0.00, 0.00

<Back to Main Menu>
```

The uptime command returns the values of how long the system has been running.

```
4)clear
    message=$(uptime)
    whiptail --msgbox --ok-button "$MENU_TEXT" --title "Uptime of router" "$message" 10 58
;;
```

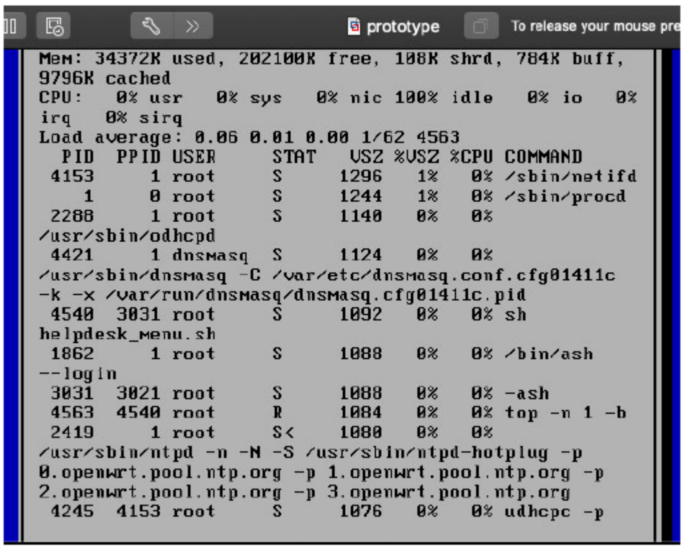
# TASK 5

## 5) DISPLAY THE DISK USAGE AND CPU UTILISATION OF THE ROUTER



```
Filesystem                  Size      Used Available
Use% Mounted on
/dev/root                 102.4M     24.5M     75.8M
24% /
tmpfs                     115.5M     80.0K    115.4M
0% /tmp
/dev/sda1                  15.7M      4.5M     10.9M
29% /boot
/dev/sda1                  15.7M      4.5M     10.9M
29% /boot
tmpfs                     512.0K         0    512.0K
0% /dev
```

When this option is chosen the first message box shows the disk usage and after enter is pressed a different message box appears with the cpu utilisation of the router.



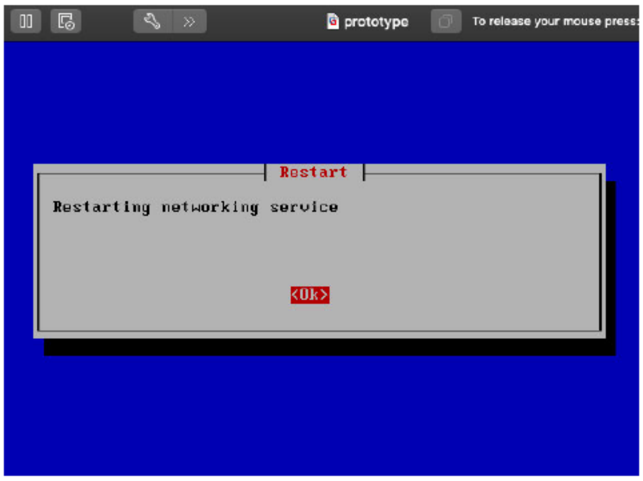
```
Mem: 34372K used, 202100K free, 108K shrd, 784K buff,
9796K cached
CPU:   0% usr   0% sys   0% nic 100% idle   0% io   0%
irq   0% sirq
Load average: 0.06 0.01 0.00 1/62 4563
  PID  PPID USER     STAT   VSZ %VSZ %CPU COMMAND
 4153     1 root     S     1296   1%   0% /sbin/netifd
    1     0 root     S     1244   1%   0% /sbin/procd
 2288     1 root     S     1140   0%   0%
/usr/sbin/odhcpd
 4421     1 dnsmasq  S     1124   0%   0%
/usr/sbin/dnsmasq -C /var/etc/dnsmasq.conf.cfg01411c
-k -x /var/run/dnsmasq/dnsmasq.cfg01411c.pid
 4540  3031 root     S     1092   0%   0% sh
helpdesk_menu.sh
 1862     1 root     S     1088   0%   0% /bin/ash
--login
 3031  3021 root     S     1088   0%   0% -ash
 4563  4540 root     R     1084   0%   0% top -n 1 -b
 2419     1 root     S<    1080   0%   0%
/usr/sbin/ntpd -n -N -S /usr/sbin/ntpd-hotplug -p
0.openwrt.pool.ntp.org -p 1.openwrt.pool.ntp.org -p
2.openwrt.pool.ntp.org -p 3.openwrt.pool.ntp.org
 4245  4153 root     S     1076   0%   0% udhcpc -p
```

The df command displays the amount of disk space available on the file system. -h stands for human readable format.
To determine the CPU utilisation I am using the top command.

```
5)clear
    message=$(df -h)
    whiptail --msgbox --ok-button "See the CPU ussage" --title "Disk Usage" "$message" 30 58

    message=$(top -n 1 -b)
    whiptail --msgbox --ok-button "$MENU_TEXT" --title "Disk Usage" "$message" 30 58
;;
```

# TASK 6

## 6) RESTART THE NETWORKING SERVICE ON THE ROUTER



```
Restart
Restarting networking service
<Ok>
```

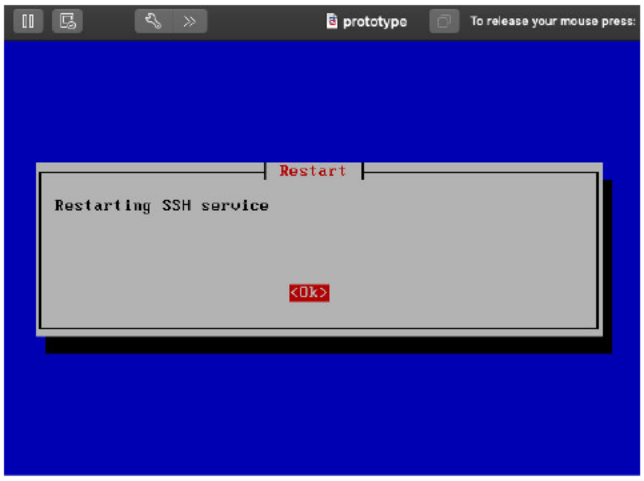
Message box displays that the networking service is about to restart.

The '/etc/init.d/network restart' syntax restarts the networking on Ubuntu or Debian based operating systems.

```
6)/etc/init.d/network restart;
    whiptail --msgbox --title "Restart" "Restarting networking service" 10 58
;;
```

# TASK 7

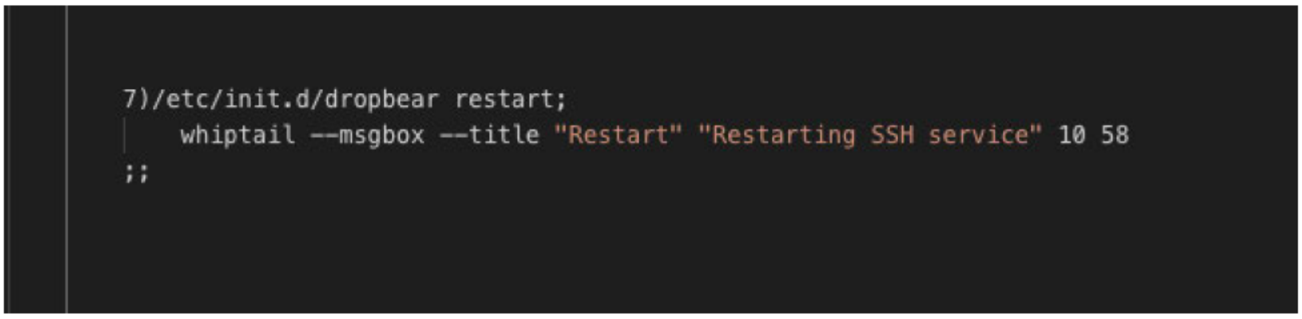
## 7) RESTART THE SSH SERVICE ON THE ROUTER



```
Restart
Restarting SSH service
<Ok>
```

Message box displays that the SSH service is about to restart.

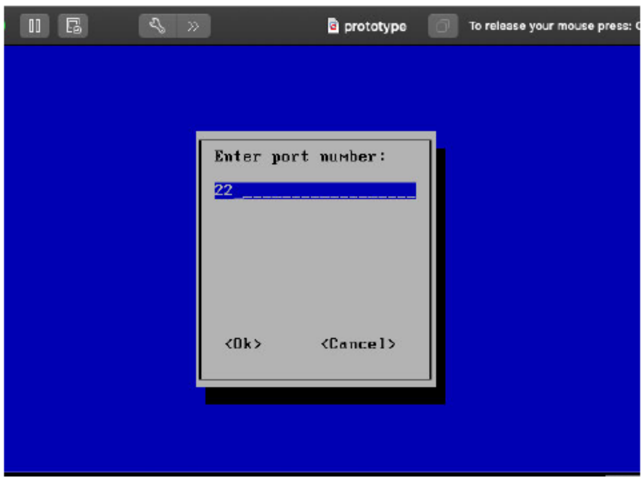
According to the OpenWrt documentation "the ssh configuration is handled by the dropbear subsystem of uci and the configuration file is located in /etc/config/dropbear." so this is what I am using to restart the SSH service.



```
7)/etc/init.d/dropbear restart;
    whiptail --msgbox --title "Restart" "Restarting SSH service" 10 58
;;
```

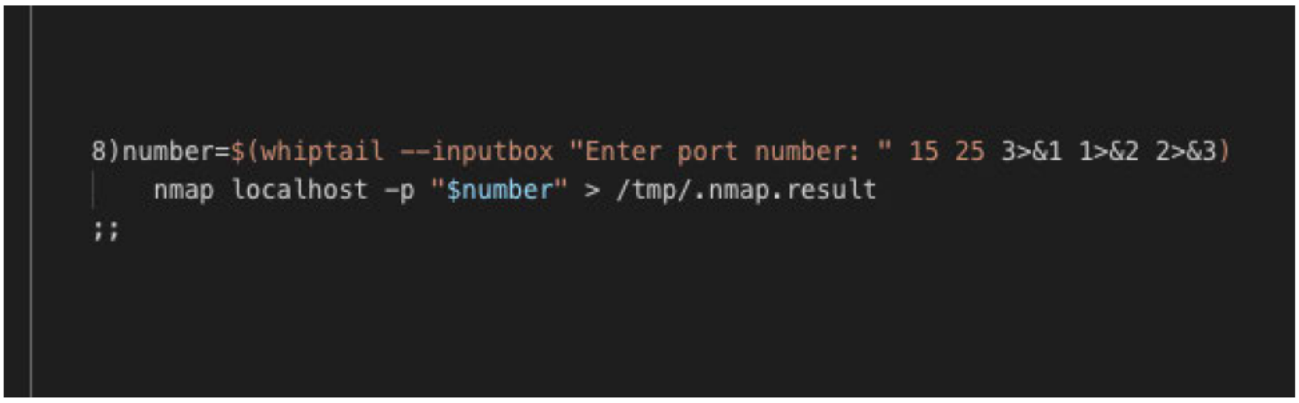
# TASK 8

## 8) ENTER A PORT NUMBER AND RUN AN NMAP SCAN ON THIS PORT



This option allows the user to run an nmap scan after entering a port number.

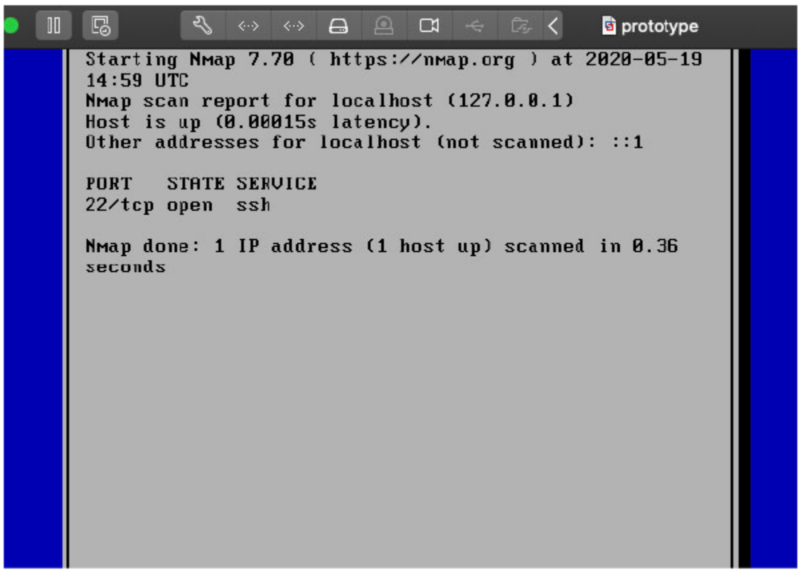
After the script has been executed the results are stored in a file.



```
8)number=$(whiptail --inputbox "Enter port number: " 15 25 3>&1 1>&2 2>&3)
    nmap localhost -p "$number" > /tmp/.nmap.result
;;
```

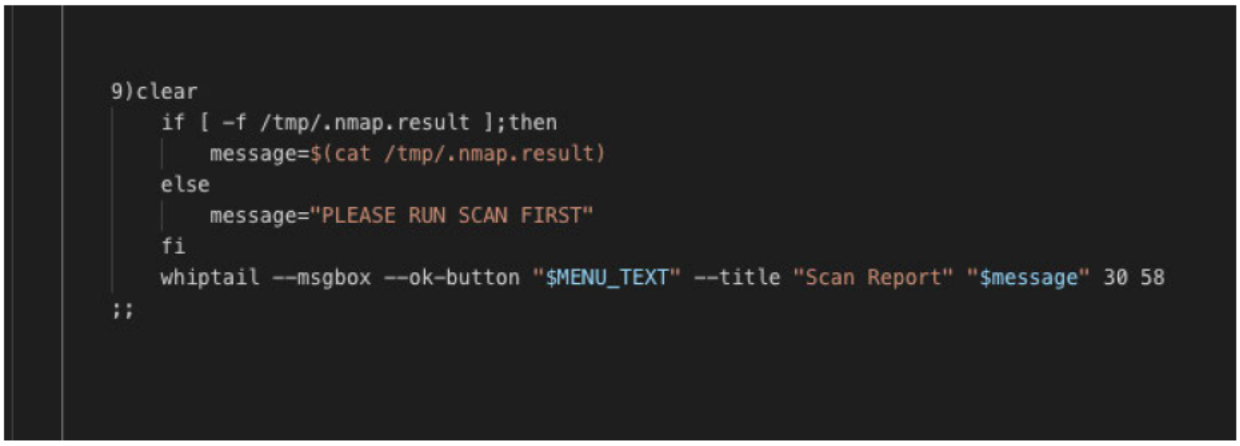
# TASK 9

## 9) RETURN THE VALUE OF THE NMAP SCAN



```
Starting Nmap 7.70 ( https://nmap.org ) at 2020-05-19
14:59 UTC
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00015s latency).
Other addresses for localhost (not scanned): ::1

PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.36
seconds
```

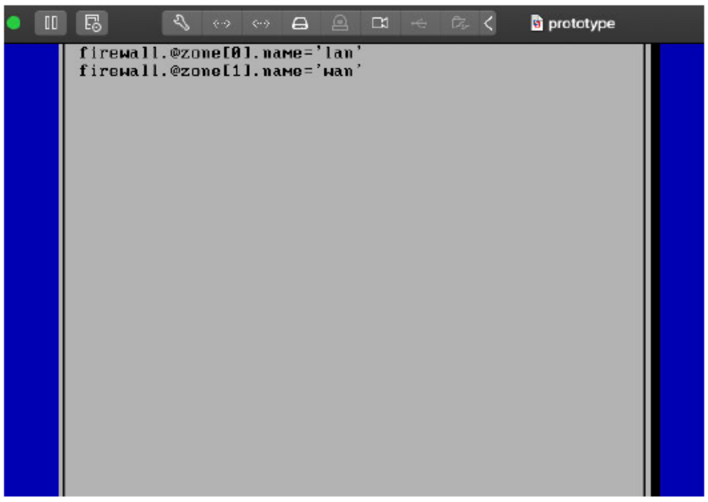
If this option is chosen before the user has run the previous script the program shows the "Please run scan first" message. Otherwise the file is accessed with the cat command and the result is presented in a message box.



```
9)clear
    if [ -f /tmp/.nmap.result ];then
        message=$(cat /tmp/.nmap.result)
    else
        message="PLEASE RUN SCAN FIRST"
    fi
    whiptail --msgbox --ok-button "$MENU_TEXT" --title "Scan Report" "$message" 30 58
;;
```

# TASK 10

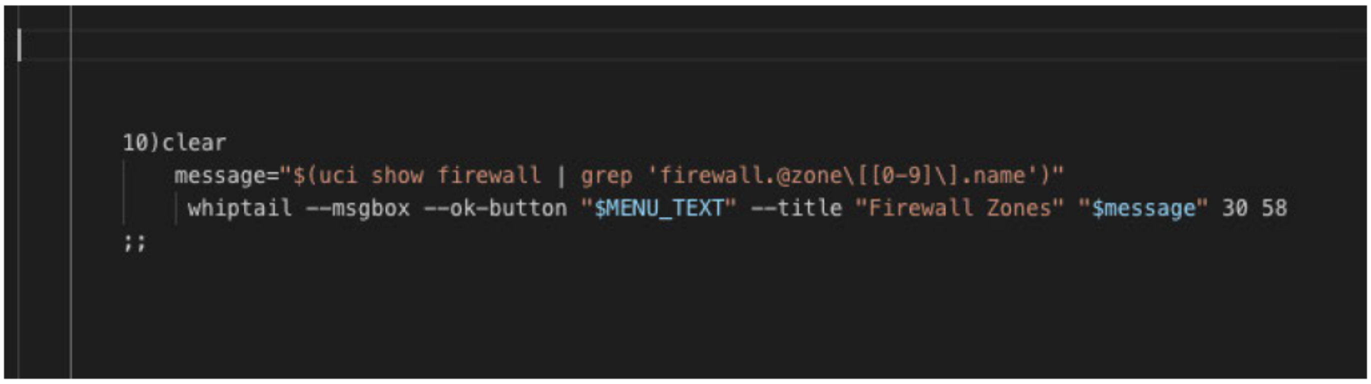
## 10) DISPLAY THE DIFFERENT FIREWALL ZONES



```
firewall.@zone[0].name='lan'
firewall.@zone[1].name='wan'
```

This function displays the firewall zones in a message box.

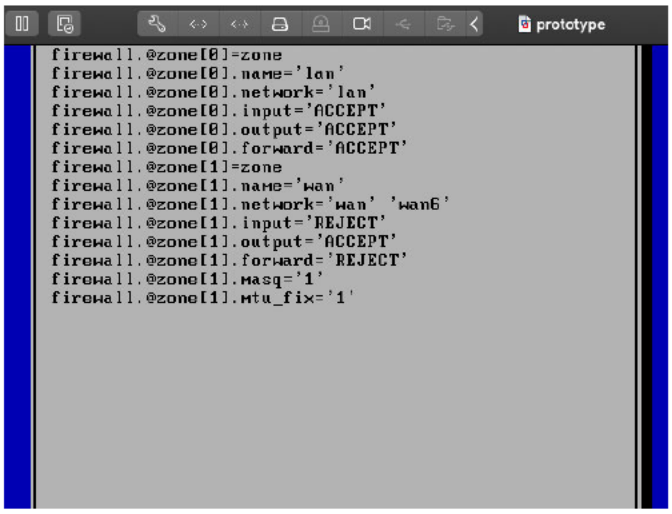
The uci show firewall command is used to access information about the firewall configuration.
| grep 'firewall.@zone\[[0-9]\].name' will filter the results that match a certain format.



```
10)clear
    message="$(uci show firewall | grep 'firewall.@zone\[[0-9]\].name')"
     whiptail --msgbox --ok-button "$MENU_TEXT" --title "Firewall Zones" "$message" 30 58
;;
```

# TASK 11

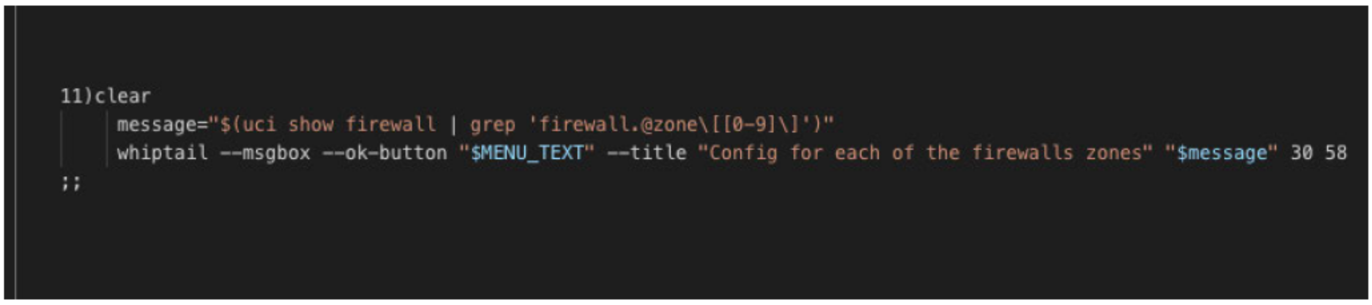
## 11) DISPLAY THE CONFIG FOR EACH OF THE FIREWALL ZONES



```
firewall.@zone[0]=zone
firewall.@zone[0].name='lan'
firewall.@zone[0].network='lan'
firewall.@zone[0].input='ACCEPT'
firewall.@zone[0].output='ACCEPT'
firewall.@zone[0].forward='ACCEPT'
firewall.@zone[1]=zone
firewall.@zone[1].name='wan'
firewall.@zone[1].network='wan' 'wan6'
firewall.@zone[1].input='REJECT'
firewall.@zone[1].output='ACCEPT'
firewall.@zone[1].forward='REJECT'
firewall.@zone[1].masq='1'
firewall.@zone[1].mtu_fix='1'
```

The function displays the firewall configuration in a whiptail message box.

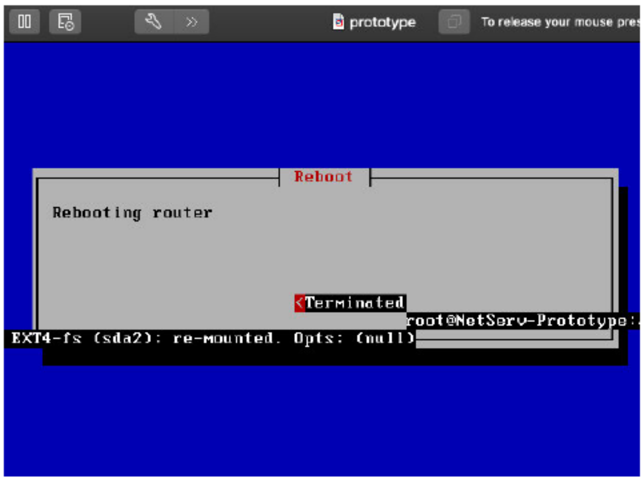
Using the uci show firewall | grep firewall.@zone\[[0-9]\] command to look for the firewall configurations that match a specific format.



```
11)clear
    message="$(uci show firewall | grep 'firewall.@zone\[[0-9]\]')"
    whiptail --msgbox --ok-button "$MENU_TEXT" --title "Config for each of the firewalls zones" "$message" 30 58
;;
```

# TASK 12

## 12) REBOOT THE ROUTER



```
Reboot
Rebooting router
<Terminated
root@NetServ-Prototype:
EXT4-fs (sda2): re-mounted. Opts: (null)
```

Rebooting the rooter with the reboot command.

```
12)whiptail --msgbox --title "Reboot" "Rebooting router" 10 58
    reboot
;;
```