

Lab 6 – Scripting

Name: Paul

Course/Section: IS-1003-01

Date: 4/15/2025

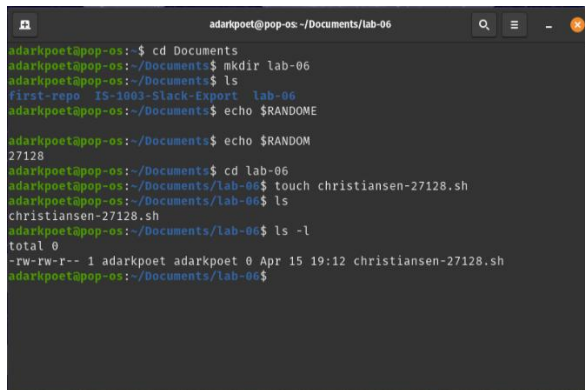
INTRODUCTION

In cybersecurity, professionals gotta be able to script efficiently for both offensive and defensive operations. In this lab we explore how to create and implement a modular bash script with three functions that demonstrate various reconnaissance techniques. These functions include displaying date information for logging, scanning IP addresses and ports, and implementing a simple Caesar cipher for encryption and decryption.

BREAKPOINT 1

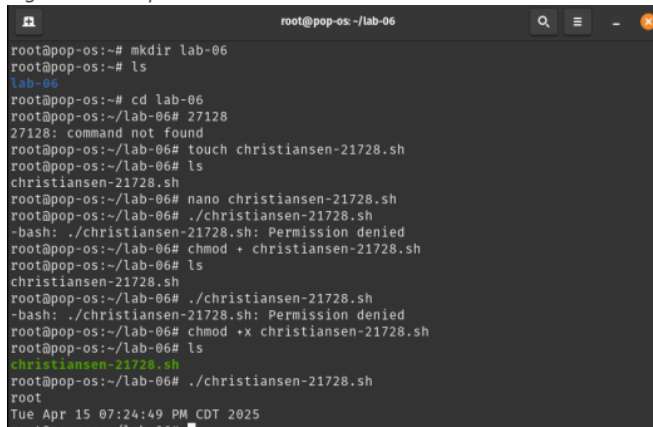
This break point was extremely easy. In this breakpoint we explored the \$RANDOM command and then used the number to create a file with that number. In this same breakpoint we set up our lab folder by creating the directory for it.

When comparing scripting via typing in the terminal or scripting via the nano/file. The biggest advantage is the ability to run the executable more frequently. We can run the script itself then typing the entire command we want to do but also we can run more commands in a single executable then typing in the terminal.



```
adarkpoet@pop-os: ~/Documents/lab-06
adarkpoet@pop-os:~$ cd Documents
adarkpoet@pop-os:~/Documents$ mkdir lab-06
adarkpoet@pop-os:~/Documents$ ls
first-repo  IS-1003-Slack-Export  lab-06
adarkpoet@pop-os:~/Documents$ echo $RANDOM
27128
adarkpoet@pop-os:~/Documents$ cd lab-06
adarkpoet@pop-os:~/Documents/lab-06$ touch christiansen-27128.sh
adarkpoet@pop-os:~/Documents/lab-06$ ls
christiansen-27128.sh
adarkpoet@pop-os:~/Documents/lab-06$ ls -l
total 0
-rw-rw-r-- 1 adarkpoet adarkpoet 0 Apr 15 19:12 christiansen-27128.sh
adarkpoet@pop-os:~/Documents/lab-06$
```

Figure 1 Breakpoint 1 Execution



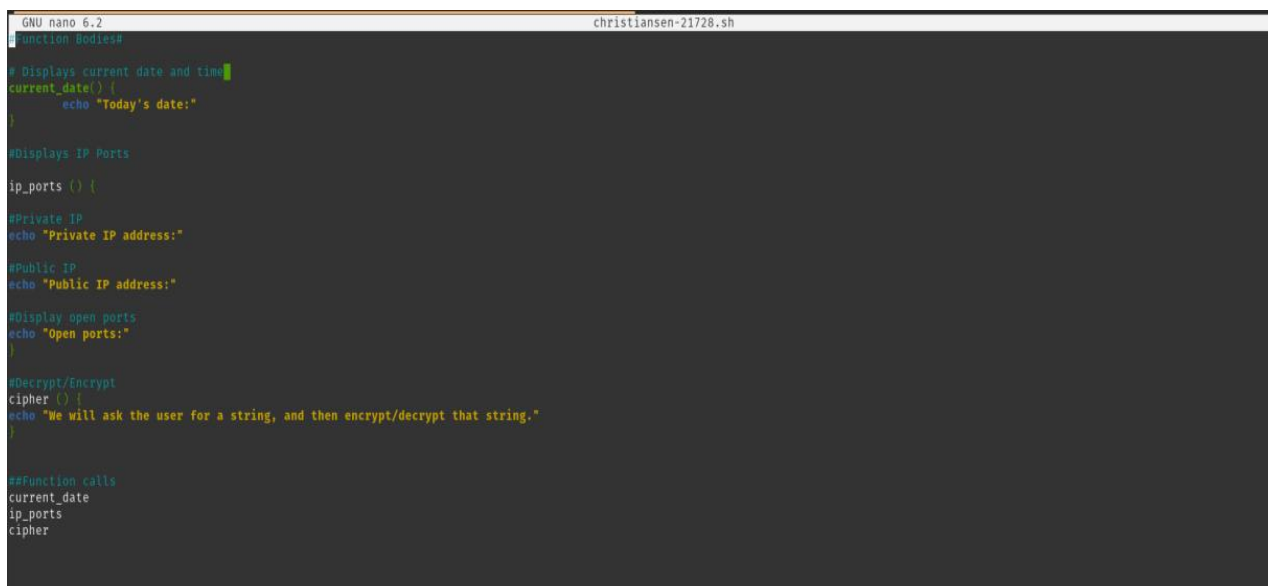
```
root@pop-os: ~/lab-06
root@pop-os:~# mkdir lab-06
root@pop-os:~# ls
lab-06
root@pop-os:~# cd lab-06
root@pop-os:~/lab-06# 27128
27128: command not found
root@pop-os:~/lab-06# touch christiansen-27128.sh
root@pop-os:~/lab-06# ls
christiansen-27128.sh
root@pop-os:~/lab-06# nano christiansen-27128.sh
root@pop-os:~/lab-06# ./christiansen-27128.sh
-bash: ./christiansen-27128.sh: Permission denied
root@pop-os:~/lab-06# chmod +x christiansen-27128.sh
root@pop-os:~/lab-06# ls
christiansen-27128.sh
root@pop-os:~/lab-06# ./christiansen-27128.sh
-bash: ./christiansen-27128.sh: Permission denied
root@pop-os:~/lab-06# chmod +x christiansen-27128.sh
root@pop-os:~/lab-06# ls
christiansen-27128.sh
root@pop-os:~/lab-06# ./christiansen-27128.sh
root
Tue Apr 15 07:24:49 PM CDT 2025
root@pop-os:~/lab-06#
```

Figure 2 Breakpoint 1 Execution

BREAKPOINT 2

This breakpoint, much like breakpoint one, was more just setup for the next breakpoints. We set up different sections for the designated tasks we were gonna do-show date/time, show ip ports, and cipher display. A lot of what we were doing was putting echo code in the designated areas to make sure that the areas actually execute properly.

Design is important when coding as having a consistent structure and COMMENTS helps make code readable and allows others to be able to come in and maintain the code as well. Overall both concepts contribute to maintenance and efficiency.



```
GNU nano 6.2 christiansen-21728.sh
function Bodiless

# Displays current date and time
current_date() {
    echo "Today's date:"
}

#Displays IP Ports
ip_ports () {
    #Private IP
    echo "Private IP address:"

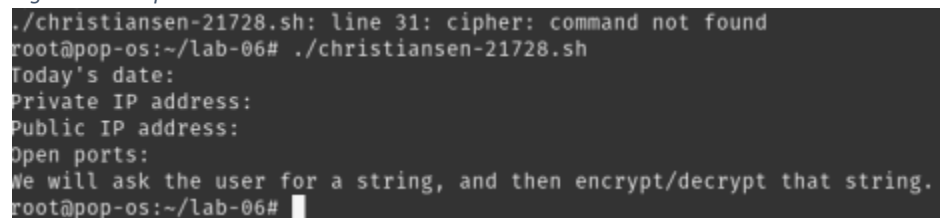
    #Public IP
    echo "Public IP address:"

    #Display open ports
    echo "Open ports:"
}

#Decrypt/Encrypt
cipher () {
    echo "We will ask the user for a string, and then encrypt/decrypt that string."
}

##Function calls
current_date
ip_ports
cipher
```

Figure 3 Breakpoint 2 Code



```
./christiansen-21728.sh: line 31: cipher: command not found
root@pop-os:~/lab-06# ./christiansen-21728.sh
Today's date:
Private IP address:
Public IP address:
Open ports:
We will ask the user for a string, and then encrypt/decrypt that string.
root@pop-os:~/lab-06#
```

Figure 4 Breakpoint 2 code execution

BREAKPOINT 3

In this breakpoint, we filled in the `current_date` function of this exam. The first line we set the name of the command, today, then we assigned the month/date/and year format. Then we followed that up with the time as well. Defining the function as `current_time` and the formatting in the hours, minutes, seconds and then if its am or pm.

In the screenshots you provided in this lab, you followed consistent indentation and descriptive function names for every function written.

```
GNU nano 6.2
#function_bodies#

# Displays current date and time
current_date() {
    today=$(date +%m-%d-%y)
    echo -e "\nToday's date: $today "
    #time
    current_time=$(date +%H:%M:%S %p)
    echo -e "The current time: $current_time\n"
}

#Displays IP Ports
ip_ports() {
    #Private IP
    echo "Private IP address:"
    #Public IP
    echo "Public IP address:"
    #Display open ports
    echo "Open ports:"
}

#Decrypt/Encrypt
cipher() {
    echo "We will ask the user for a string, and then encrypt/decrypt that string."
}

##Function calls
current_date
ip_ports
cipher
```

Figure 5 Breakpoint 3 Code

```
root@pop-os:~/lab-06# ./christiansen-21728.sh

Today's date: 04-15-25
The current time: 19:52:45 PM

Private IP address:
Public IP address:
Open ports:
We will ask the user for a string, and then encrypt/decrypt that string.
```

Figure 6 Breakpoint 3 Execution

BREAKPOINT 4

So for this breakpoint we filled in the ports module of this executable. I wrote the first IP code and calls the hosts IP address. The second one pulls my public IP address through the curl command. The curl command is used to transfer either to or from a server from various protocols like HTTP, HTTPS, etc. With curl, we connected to ifconfig.me and that is how we were able to pull our public IP. The Nmap command, aka network mapper, is used to scan networks and find what ports or various other services are available, open and or are running. I have used this command personally as my dad taught me how to use when I first started learning network security when I was little.

```
root@pop-os:~/lab-06# ./christiansen-21728.sh
Today's date: 04-15-25
The current time: 20:02:07 PM

Private IP address:
10.0.2.15 fd00::1af7:d1fb:6d4:d218 fd00::26a4:20bc:50c0:5d40
Public IP address:
24.26.232.129
:
Open ports:
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-15 20:02 CDT
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.016s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01:f03c:91ff:fe18:bb2f
Not shown: 997 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
31337/tcp open  Elite

Nmap done: 1 IP address (1 host up) scanned in 6.12 seconds

We will ask the user for a string, and then encrypt/decrypt that string.
root@pop-os:~/lab-06#
```

Figure 8 Breakpoint 4 Execution

```
[1/1]
function Bodies#
#Displays current date and time
current_date() {
    today=$(date +"%m-%d-%y")
    echo -e "\nToday's date: $today "
    #time
    current_time=$(date +"%H:%M:%S %p")
    echo -e "The current time: $current_time\n"
}

#Displays IP Ports
ip_ports () {
    #Private IP
    echo "Private IP address:"
    hostname -i
    #Public IP
    echo -e "Public IP address:"
    curl ifconfig.me
    echo -e "\n:"
    #Display open ports
    echo -e "Open ports:"
    nmap scanme.nmap.org
    echo -e "\n"
}

#Decrypt/Encrypt
cipher () {
    echo "We will ask the user for a string, and then encrypt/decrypt that string."
}

#Function calls
```

Figure 7 Breakpoint 4 Code

BREAKPOINT 5

This final breakpoint we are filling out the cipher command at the end of the nano file. What I did, in figure 9, was write the beginning of the file stating that this file starts running on true which allows the loop to execute. Continuing, I wrote a while loop that will allow it to run until it is no longer true. Following that, I set the false case for the program where if the user enters 0, then it sets the run equal to false and as such ends the program. The else case of the while loop is the code that encrypts and decrypts the code. In figure 9, it uses the 5 figures question asked but the last figure is the cipher that was originally displayed in the lab since I forgot to show that code but still wanted to show that I did get it to work. In terms of format, the only difference between the 5 backwards and 3 forwards is the cypher letters in the " for each encrypted and decrypted.

```
#Decrypt/Encrypt
cipher () {
run-true #control the loop

while :run; do
read -p "Enter string (or '0' to quit): " input #Gets user input
if [[ "$input" == 0 ]]; then
echo "Thank you! Bye!!" #closing message
run=false #ends loop
else
encrypted=$(echo "$input" | tr 'a-zA-Z' 'f-ze-eu-zd-wf-ZA-EU-ZD-W') #Caesar Cipher
decrypted=$(echo "$encrypted" | tr 'a-zA-Z' 'v-za-uf-ze-wV-ZA-UF-ZE-W') # Decrypt Cipher
echo "Encrypted: $encrypted" #shows encryption
echo "Decrypted: $decrypted" #shows decryption
fi
done
}
```

Figure 10 Breakpoint 5 Code

```
Enter string (or '0' to quit): test
Encrypted: yjxy
Decrypted: test
Enter string (or '0' to quit):
```

Figure 9 5 backwards cipher

```
Enter string (or '0' to quit): Yourmomisgay
Encrypted: Brxuprplvjdb
Decrypted: Yourmomisgay
Enter string (or '0' to quit): 0
Thank you! Bye!!
```

Figure 11 3 backwards cipher

CONCLUSION

This lab was overall very easy. The only problem I really ran into was small typo errors when in the nano. Thankfully this was very easy to correct when comparing what was in the lab and in my own vm. This lab kinda opened my eyes on how to capture IP ports and gonna go test it on devices that I have permission to do so as if I use the nmap command on a device that I am not allowed to do that on...that can cause me some problems. Overall, this was dipping my toes into scripting and this lab reinforces concepts my dad taught me as a kid.

REFERENCES

R. Mitra, "Lab 6: Scripting," The University of Texas at San Antonio (2024). Last accessed: April 15, 2025.

COLLABORATION
I worked alone.