Template Final Exam

|  |  |  |
| --- | --- | --- |
| # | Full Name | Group |
|  | Bosyn Akylbek | Cs-2121 |
|  | Adil Sarsembek | Cs-2121 |
|  |  |  |
|  |  |  |

|  |
| --- |
| Link to the repository: |

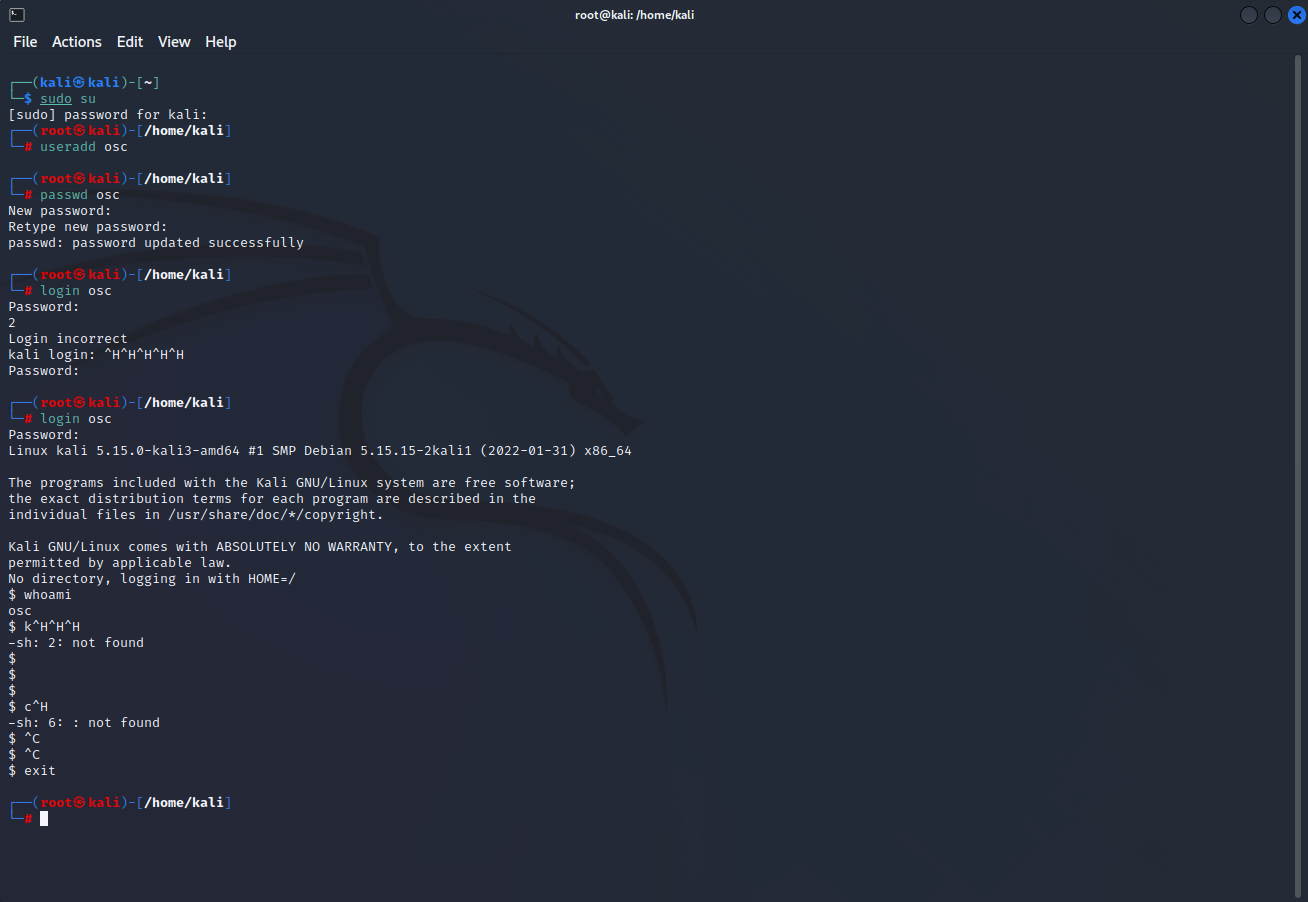
Step-by-step task completion:

Task 1:

First of all I added new user, password and login.

Then I checked with a help of “whoami”

Last command “logout” and ‘’exit”

Screenshots of the code compilation result: 

Task 2:

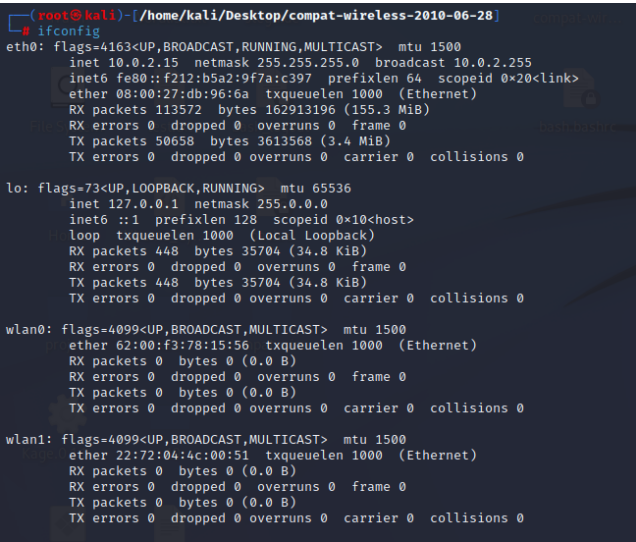
Screenshots of the code compilation result:

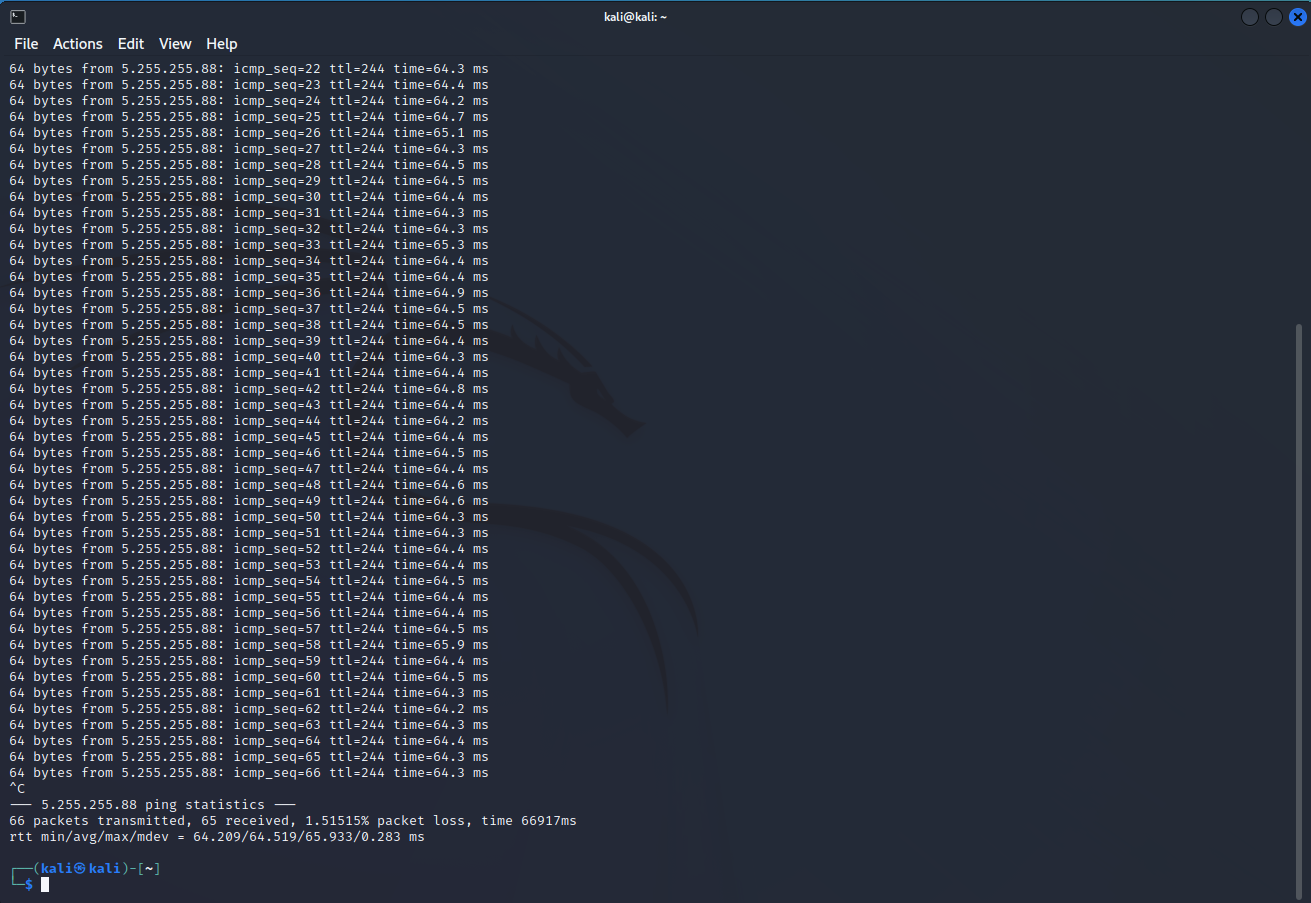
Direct IP Connection

1. For start, Provide the virtual computer a direct public IP address.

For illustration, I'll use the ip domain of yandex.ru which, ip address is - 142.250.74.100. I find it by pinging.

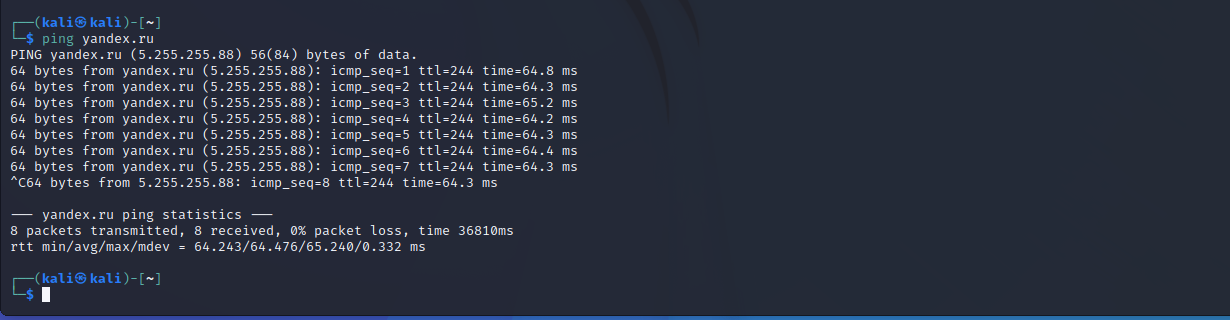
1. Because of public IP addresses are a limited resource, this method requires the virtual machine to have its own publicly available IP address, which may not always be possible.
2. The virtual machine can now have direct connection to the Internet.

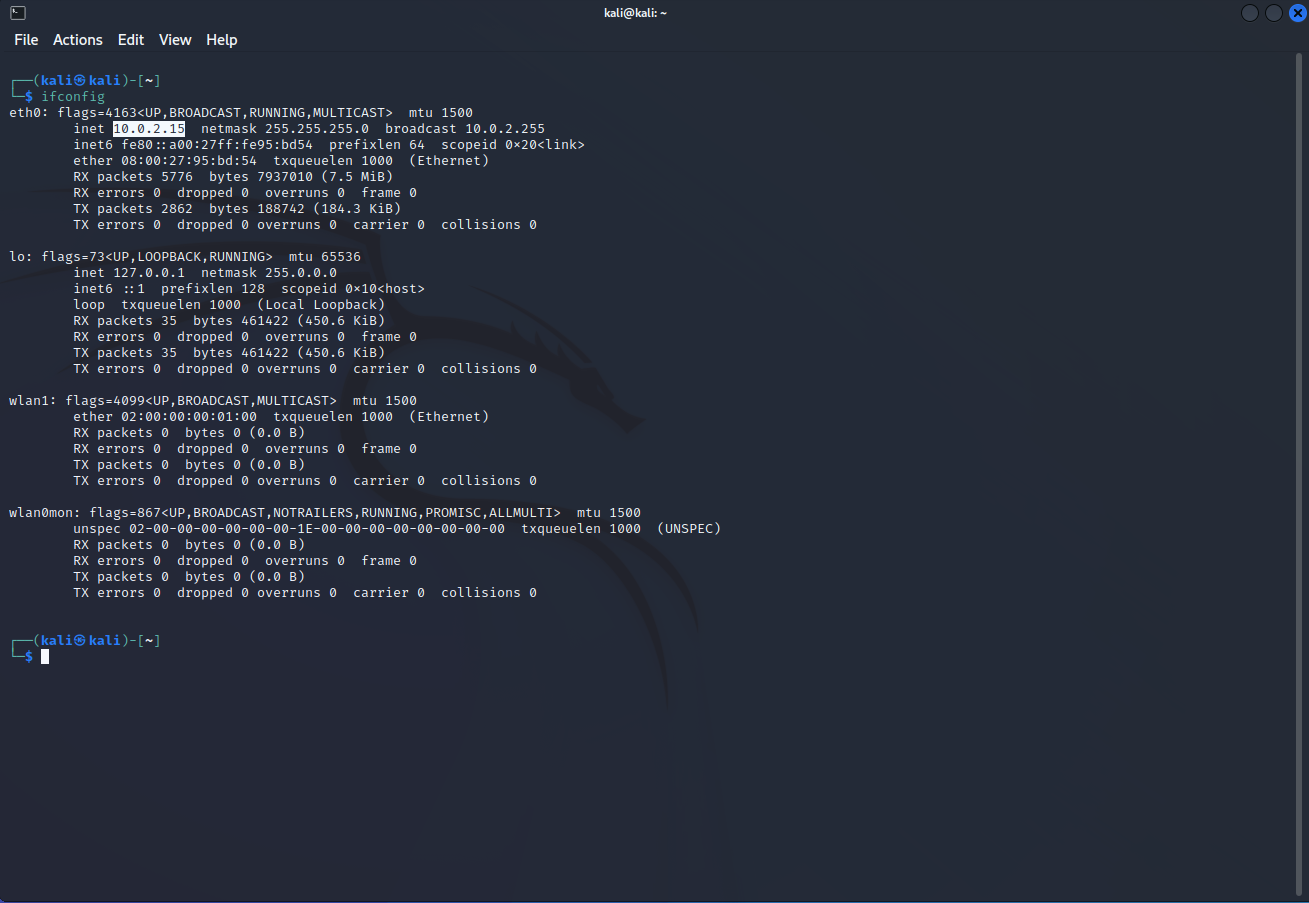




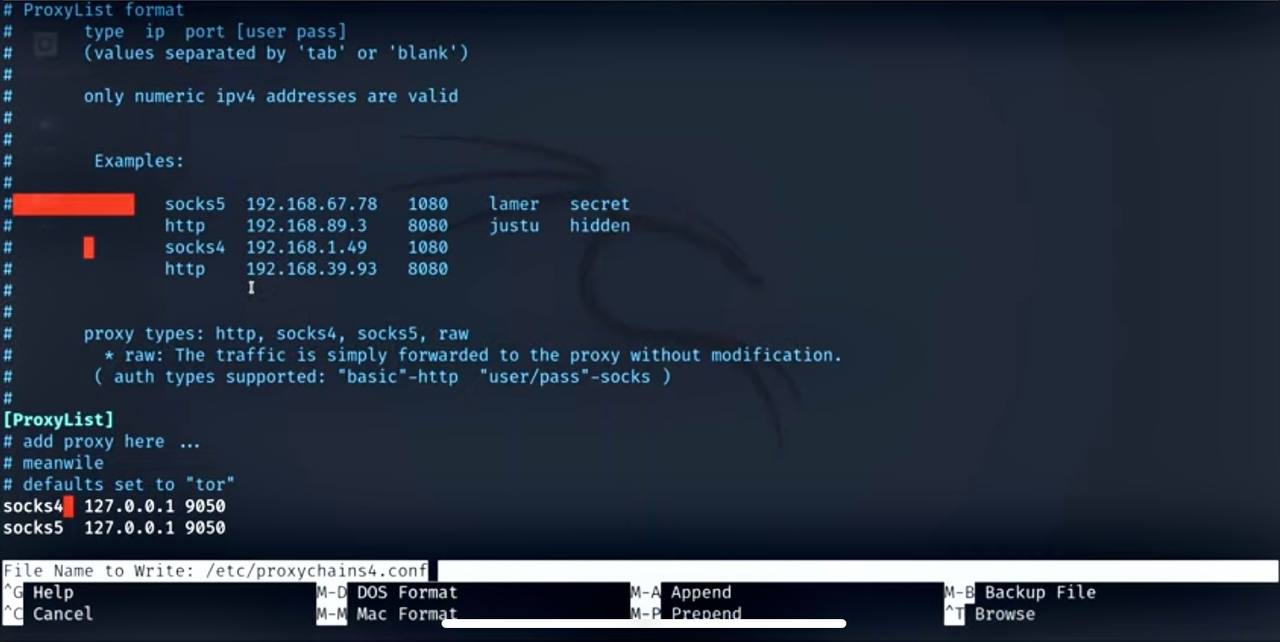
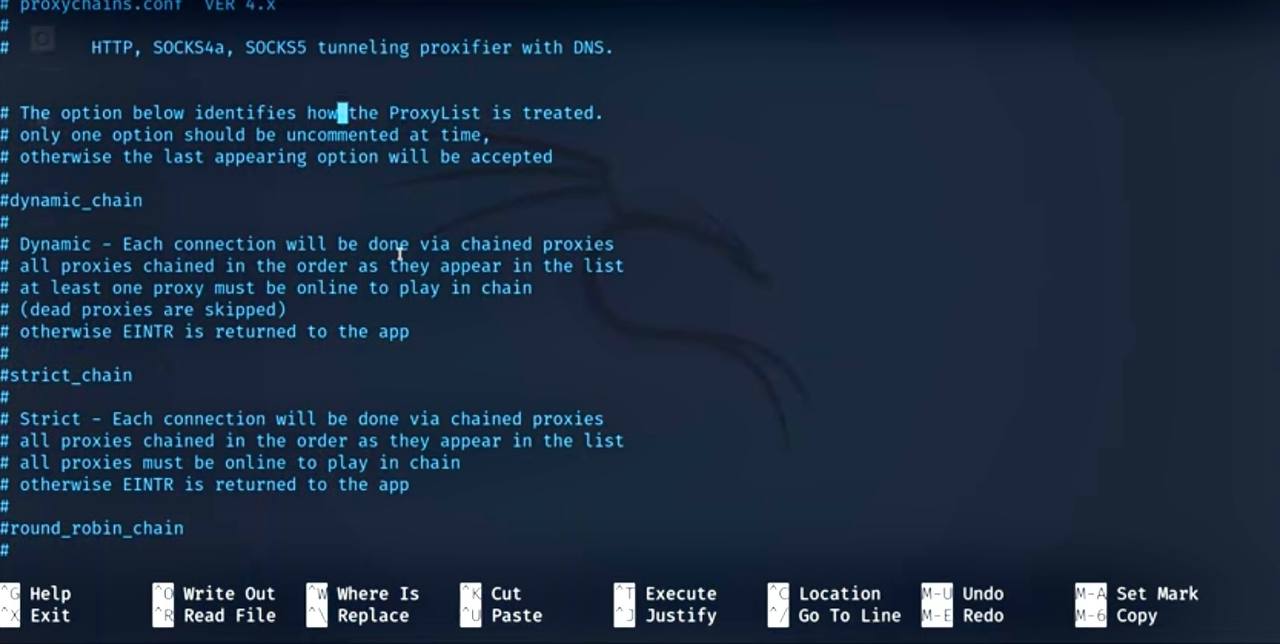
NAT

1. A private IP address is given to the virtual machine, and a NAT gateway is utilized to convert the private IP address to a public IP address.
2. This keeps the finite amount of public IP addresses while enabling the virtual machine to access the Internet.
3. Virtual private clouds and residential networks both frequently employ a NAT.





via Proxy Server

Connection via Proxy server is the easiest way to away direct connection in server. To start we should edit our “proxychains.conf” in our linux adding active proxy server IP. By proxychain we can check our connection – is it active or not. 

Task 3:

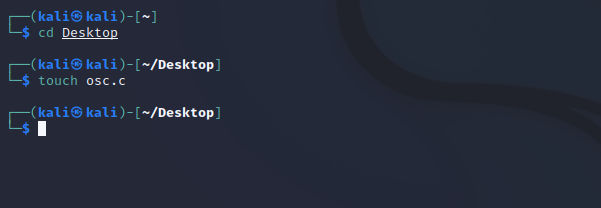
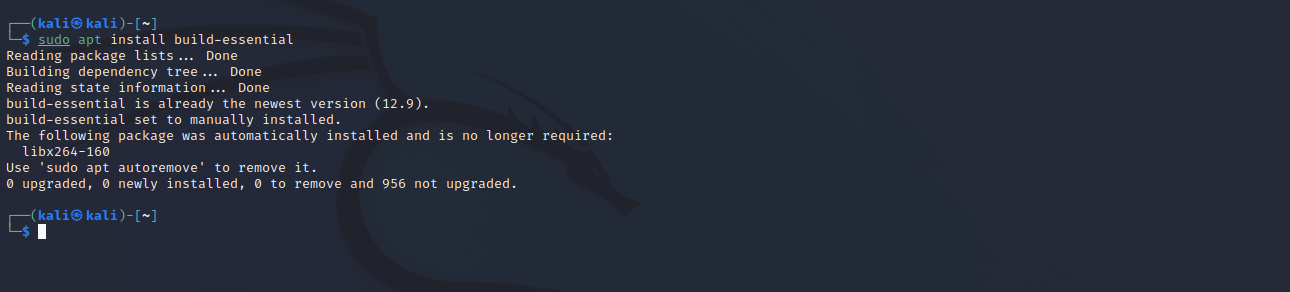
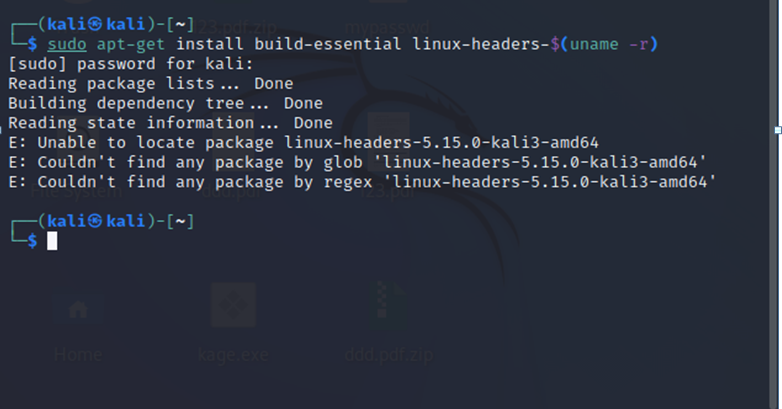
Some common steps involved in developing a Linux kernel module include:

1. Setting up a development environment: This involves installing the necessary tools and libraries, such as the GNU Compiler Collection (GCC) and kernel headers, to build and test the module.
2. Writing the module code: This involves using a programming language such as C or C++ to implement the desired functionality, including any necessary data structures and algorithms.

Compiling the module: This involves using the development environment to compile the module code into a binary file that can be loaded into the kernel

Screenshots of the code compilation result:

Firstly I installed necessary packages for KMD





Module testing: this involves the use of various testing methods to ensure that the module is working properly and does not cause instability or system failures.

Packaging and distribution: once the module is completed and tested, it can be packaged and distributed as part of a larger software project, such as a Linux distribution or a dedicated software package.