

# **DVG001**

## **Introduction to Linux and Small Networks Laboration 6**

### **Servers & Firewalls**

av

Jonatan Rassekhnia (001000)

**Akademin för teknik och miljö  
Avdelningen för industriell utveckling, IT och samhällsbyggnad**

Högskolan i Gävle  
S-801 76 Gävle, Sweden

Datorpost: johnny.rass@outlook.com

## Contents

1. Introduction.....	1
1.1. Background .....	1
1.2. Purpose .....	1
2. Planning and implementation .....	1
2.1. Planning.....	2
2.2. Implemenation.....	<b>Error! Bookmark not defined.</b>
3. Description of result .....	5
4. Discussion.....	5
5. Conclusion .....	5
6. Reference .....	6
I. Annex(es).....	7
II. Next Attachment.....	11

# 1. Introduction

In laboratory 6, I will explore the implementation of network security measures using an NFS server, NFS client, and the *ufw* firewall in a Linux environment. My objective is to establish secure network access and control for various services, such as SSH and HTTP, while ensuring restricted access to NFS services within the local LAN. The instructions are provided in the assignment, that I will configure the NFS server and client to allow reading access to the */nfs/public* directory for the entire local network while restricting the write access to a single machine. The lab also involves in setting up a firewall to safeguard the server, utilizing its complicated interface to define rules for the incoming connections. The firewall will be configured to permit SSH and HTTP access from any machine, but will only allow NFS access within the local LAN. Furthermore, I will explore the use of *nmap* to test the effectiveness of the firewall by examining the allowed and blocked connections. The tasks given will give me hands-on experience in securing the network services and maintain the control over the network access in a Linux environment.

## 1.1. Background

Lab 6 is based Network Security; network security is maintaining the integrity and confidentiality of data in today's world. Network systems are to store and access critical information, it becomes essential to implement a robust security measure to protect against unauthorized access with potential threats. In this assignment, I delve into the configuration and implementation of an *NFS* server and *NFS* client, as well as the *ufw* firewall, in a Linux environment.

*NFS* (Network File System) facilitates remote files sharing and access amount multiple machines, while the *ufw* firewall provides a user-friendly interface for managing network traffic and enforcing access control policies. By understand these technologies, I can establish secure network connections, control access to share resources, and enhance the overall security posture of our networked system.

## 1.2. Purpose

In order to address this question; I need to consider the fundamental objectives behind these tasks. Firstly, what is the purpose of configuring an NFS server and client? Configuring an NFS server and client allows for seamless file sharing and access across a network, enabling efficient collaboration and data exchange among multiple machines. This raises the question of how to ensure secure and controlled access to the shared resources. Secondly, why do I need to implement a firewall in this environment? The firewall servers as a protective barrier that filters network traffic, allowing me to regulate incoming and outgoing connections based on predefined rules. Combining the functionalities of NFS and a firewall, I can create a secure network environment that enables authorized access to shared data while avoiding potential security threats.

# 2. Planning and implementation

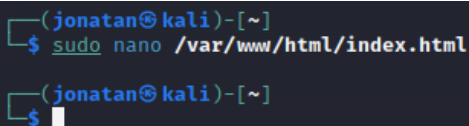
The planning and implementation that will take place in lab 6, is to configure an NFS server, NFS client, and firewall requiring me to make a careful and systematic approach. The planning phase involves identifying the specific requirements of a network environment, such as the need for file sharing, access control, and security measures. It is essential to determine which machine or network should have access to the shared resources and which should be restricted. The role is to select appropriate firewall settings and rules that play a role in ensuring network protections. Once the planning phase is complete, the implementation process involves executing the necessary steps to install and configure the NFS server, NFS client and firewall based on the predefined requirements.

## 2.1. Planning

The planning in Lab 6 is critical for the assigned tasks, my objective was to establish file sharing and access across a network while ensuring network security. I began by installing and customizing an Apache web server, followed by configuring an NFS server and client for seamless data sharing. To protect the network, I implemented a firewall using *ufw*. Throughout the planning process, I considered dependencies and potential challenges to ensure a structured and organized approach to achieve my goals. This planning begins with the necessary steps for installation and setting up an Apache web server. In this case the package is called Apache2, the next process is to modify the `index.html` file to verify the changes by accessing the appropriate URL. Moving on to the NFS configuration between the server and client and to establish a network file sharing system. The final stage will be focusing on implementing a firewall to safeguard the network environment. Understanding the importance of network security. This planning enhances my understandings of the tasks and provides a solid foundation for successful implementation.

## 2.2. Implementation

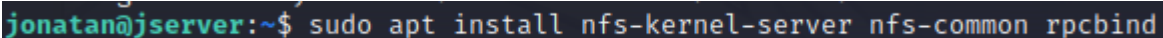
During the implementation phrase of the assigned tasks, require a careful planning to ensure a structured and organized approach. I analyzed the objectives of the tasks which involved configuring an NFS server and client, as well as implementing a firewall, and the approach with Apache2 and the modification that took place. To being my implementation is setting up a Apache web server, from reading the article “How to Install Linux, Apache, MySQL, PHP (LAMP) Stack on Ubuntu 22.04” [1]. Show me how to install, and create a virtual host for my website.



```
(jonatan@kali)~$ sudo nano /var/www/html/index.html
(jonatan@kali)~$
```

If you look at the image above this shows you the default document, and modify it which you can look at the implementation from annex section in this document called illustration 1 & 2. You can assign ownership to the directory with `$USER` variable by using *chown* and what rules you want to add to it. You can also create a virtual host for your own website, in the article on step 4 it shows you how. The implementation on this process was for apache2 webserver and modifying the `index.html` file.

Moving on to the next implementation NFS configuration, started out by installing Ubuntu server, to gain experience on the server side. The need for both NFS server and client is to establish a network file sharing system. I choose my Kali Linux machine as the client for this process, while using SSH to access my server or client. After the installation process I started out by installing these packages for my Ubuntu Server. If you look at the image line with the terminal command for the package that were needed.



```
jonatan@jserver:~$ sudo apt install nfs-kernel-server nfs-common rpcbind
```

After installing the packages, I made a directory with a folder called NFS and in that folder it contained another folder called public. Gave the permission rule with 777 then I started editing the exports for the server. To follow on with the installation process of this implement I had to read tutorial article called “How to Set Up an NFS Mount on Ubuntu 20.04” [2]. This helped me with every single process as I am explaining my steps given.

```

jonatan@jserver:~$ sudo mkdir /var/nfs
jonatan@jserver:~$ sudo mkdir /var/nfs/public
jonatan@jserver:~$ sudo chmod 777 /var/nfs/public/
jonatan@jserver:~$ sudo nano /etc/exports
jonatan@jserver:~$ sudo ufw allow nfs
Rules updated
Rules updated (v6)
jonatan@jserver:~$ sudo reboot
Connection to [REDACTED] closed by remote host.
Connection to [REDACTED] closed.

```

To check the export file, look at illustration 2, you can see what is written in the export file. Moving back to my Kali Linux client I had to install the following packages *rpcbind* with *nfs-common* being the to continue this process.

```

(jonatan@kali)-[~]
$ sudo apt install rpcbind nfs-common

```

After the installation, I wanted to make a directory for the *nfs* to be shared and mounted from. Based on the article “In order to make the remote shares available on the client, we need to mount the directories on the host that we want to share to the empty directories on the client.” [2] after the mount was done with no errors, I then moved forward with checking with the command *showmount -e*.

```

(jonatan@kali)-[~]
$ sudo mkdir /mnt/nfs
(jonatan@kali)-[~]
$ sudo mount [REDACTED]srv/data /mnt/nfs
(jonatan@kali)-[~]
$ ls /mnt/nfs
(jonatan@kali)-[~]
$ sudo nano /etc/fstab
(jonatan@kali)-[~]
$ showmount -e
Export list for kali:
/srv/data [REDACTED]
(jonatan@kali)-[~]
$

```

You can check illustration 4 to check how the *fstab* was edited in the annex section of this report. After the implementation of this process. The next implement took place by changing the directory to *nfs/public*. Created two text files in my client called *Linux.txt* & *Lab6.txt*. The Linux text file is blank while the Lab6 text file will be edited in various of locations. To check the locations and what is edited in that document you can take a look at annex section in this report for illustration 5. In the illustration takes place both on my PowerShell and client, using SSH and testing the implementation of other terminals.

```

jonatan@jserver:~$ showmount -e
Export list for jserver:
/var/nfs/public [REDACTED]
jonatan@jserver:~$ cd /var/nfs/public
jonatan@jserver:/var/nfs/public$ ls
Linux.txt
jonatan@jserver:/var/nfs/public$ ls
Lab6.txt  Linux.txt
jonatan@jserver:/var/nfs/public$ sudo nano Lab6.txt
[sudo] password for jonatan:
jonatan@jserver:/var/nfs/public$ cd
jonatan@jserver:~$ sudo nano /etc/exports
[sudo] password for jonatan:
jonatan@jserver:~$ sudo nano /etc/exports
jonatan@jserver:~$ ping [REDACTED]
PING [REDACTED] 56(84) bytes of data.
64 bytes from [REDACTED]: icmp_seq=1 ttl=64 time=0.306 ms
64 bytes from [REDACTED]: icmp_seq=2 ttl=64 time=0.460 ms
64 bytes from [REDACTED]: icmp_seq=3 ttl=64 time=0.998 ms
64 bytes from [REDACTED]: icmp_seq=4 ttl=64 time=0.532 ms
^C
--- [REDACTED] ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3047ms
rtt min/avg/max/mdev = 0.306/0.574/0.998/0.258 ms

```

Changing directories and looking at the text files created in my Linux environment, was the final process to set up and NFS server and client, while trying to logon to them from SSH. By editing the Lab6.txt document gave me confirmation that I was able to share files and that my server and client is working including the SSH tool.

In the final stage is dealing with firewalls, I added my own rules, and each port and action has a meaning.

To	Action	From
--		
OpenSSH	LIMIT	Anywhere
22/tcp	ALLOW	Anywhere
22/udp	ALLOW	Anywhere
80/tcp	ALLOW	Anywhere
80/udp	ALLOW	Anywhere
2049/tcp	ALLOW	Anywhere
2049/udp	ALLOW	Anywhere
111/tcp	ALLOW	Anywhere
111/udp	ALLOW	Anywhere
8080/tcp	ALLOW	Anywhere
8080/udp	ALLOW	Anywhere
2049	ALLOW	[REDACTED]
111	ALLOW	[REDACTED]
32767:32769/udp	ALLOW	[REDACTED]
OpenSSH (v6)	LIMIT	Anywhere (v6)
22/tcp (v6)	ALLOW	Anywhere (v6)
22/udp (v6)	ALLOW	Anywhere (v6)
80/tcp (v6)	ALLOW	Anywhere (v6)
80/udp (v6)	ALLOW	Anywhere (v6)
2049/tcp (v6)	ALLOW	Anywhere (v6)
2049/udp (v6)	ALLOW	Anywhere (v6)
111/tcp (v6)	ALLOW	Anywhere (v6)
111/udp (v6)	ALLOW	Anywhere (v6)
8080/tcp (v6)	ALLOW	Anywhere (v6)
8080/udp (v6)	ALLOW	Anywhere (v6)

jonatan@jserver:~\$

My focus is on implementing a firewall to safeguard the network environment. I needed to know

the understanding that importance of network security, really how ports work. Because I decided to work with *ufw* it made it simple and easily compatible with IPv6. By reading the article “UFW Essentials: Common Firewall Rules and Commands” [3] I’ve learned how to allow incoming connections to a network interface, or to my private local network which I have done. Commands such as *sudo ufw allow “client ip” to any port 2049 and 111*. This allows me to restrict the NFS access to a specific client IP address. I also wanted to limit my OpenSSH by using *sudo ufw limit OpenSSH from “IP address”* or no ip address. In the end it all depends on each rule you want to add on the machine and set a default firewall rule like “REJECT or DROP” to restrict incoming TCP/UDP connections. Allowing each specific connection based on the required services, it gives me the insurance to exposure minimal security threats. By typing *cat etc/services* I can see the list of ports in my network, which is considered a tool, and I learned these various of tools from reading this article “How to Find Out List of All Open Ports in Linux” [4] this guided me through transmission control protocol (TCP) and User Datagram protocol (UDP) which is the most two popular transport protocols on the internet today. The implementation enhanced my understand of the tasks given to me throughout the lab 6 and will be further discussed within my results and conclusion.

### 3. Description of result

The implementation of the tasks resulted in a concrete outcome that can be observed and evaluated in lab 6. Started out by installing the Apache2 web server, by changing the content of the default webpage “index.html”, I personalized the web page to reflect the specific requirements and preferences. This allowed me to showcase the content and the design of the webpage. It accessible from the URL of directory, localhost, or the server IP address. The result was shown on a successful customization of the Apache servers index.html file through the demonstration. The NFS server & client configuration allowed for the successful sharing and access of files across the network I implemented on. This was the evident that gave the confidence on the ability to mount the directories from the server on the client machines, enabling seamless data exchange and collaboration. The last result that I received with my experiment was the firewall implementation to enhance the network security by filtering and controlling incoming and outgoing connections. The ensured that only authorized access to share resources. Protecting against potential security threats. The implementation gave me the results to demonstrate the establishment of the secure environment created that gave a concrete end result of the tasks implemented in this assignment.

### 4. Discussion

The tasks involved configuring an NFS server and client, as well as setting up a firewall for network security, and dealing with apache2 and editing a HTML file. The objective was to establish a secure environment for file sharing and network communication. Dealing with servers and clients, including webserver. The implementation of the NFS server and client configuration, along with the deployment of *ufw* firewall served the fundamental objectives of facilitating a seamless file sharing and accessing it over the network while ensuring network security. These tasks are collaboration and data exchange across multiple machines, while firewall acted as a protective barrier, regulating network traffic based on predefined rules. I gained more knowledge about servers due to the fact I never had a practical experience in the network configuration, server administration and security practices this empowered me to create a sure and collaborative network environment under the Linux environment.

### 5. Conclusion

In conclusion, by addressing the question of purpose, I recognized the importance of balancing network and security in a Linux environment. The purpose of configuring an NFS server, NFS client, and implementing a firewall is to establish a robust and secure network infrastructure. This infrastructure aims to facilitate seamless fire sharing and collaboration while ensuring controlled

access to shared resources. This approach to network security safeguards sensitive data and it enhances the overall security posture.

## 6. Reference

### Bibliography

- [1]: K. Yang and E. Heidi, "How to Install Linux, Apache, MySQL, PHP (LAMP) Stack on Ubuntu 22.04," DigitalOcean, Apr. 26, 2022. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-22-04>. [Accessed on May 20, 2023].
- [2]: B. Boucheron, "How To Set Up an NFS Mount on Ubuntu 20.04," DigitalOcean, May 14, 2020. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-set-up-an-nfs-mount-on-ubuntu-20-04#step-5-creating-mount-points-and-mounting-directories-on-the-client>. [Accessed on: May 29, 2023.]
- [3]: M. Anicas and E. Heidi, "UFW Essentials: Common Firewall Rules and Commands," DigitalOcean, Aug. 20, 2015. [Online]. Available: <https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands>. [Accessed on: May 29, 2023.]
- [4]: A. Kili, "How to Find Out List of All Open Ports in Linux," Tecmint, Nov. 4, 2016. [Online]. Available: <https://www.tecmint.com/find-open-ports-in-linux/>. [Accessed on: May 29, 2023.]



## I. Annex(es)

```

jonatan@kali: /var/www/html
File Actions Edit View Help
GNU nano 7.2 index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1"
<html xmlns="http://www.w3.org/1999/xhtml">
<title>Apache2</title>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Apache2 Debian Default Page: It works</title>
<h1>Jonatan Rassekhnia</h1>

<style type="text/css" media="screen">
* {
margin: 0px 0px 0px 0px;
padding: 0px 0px 0px 0px;
}

body, html {
padding: 3px 3px 3px 3px;

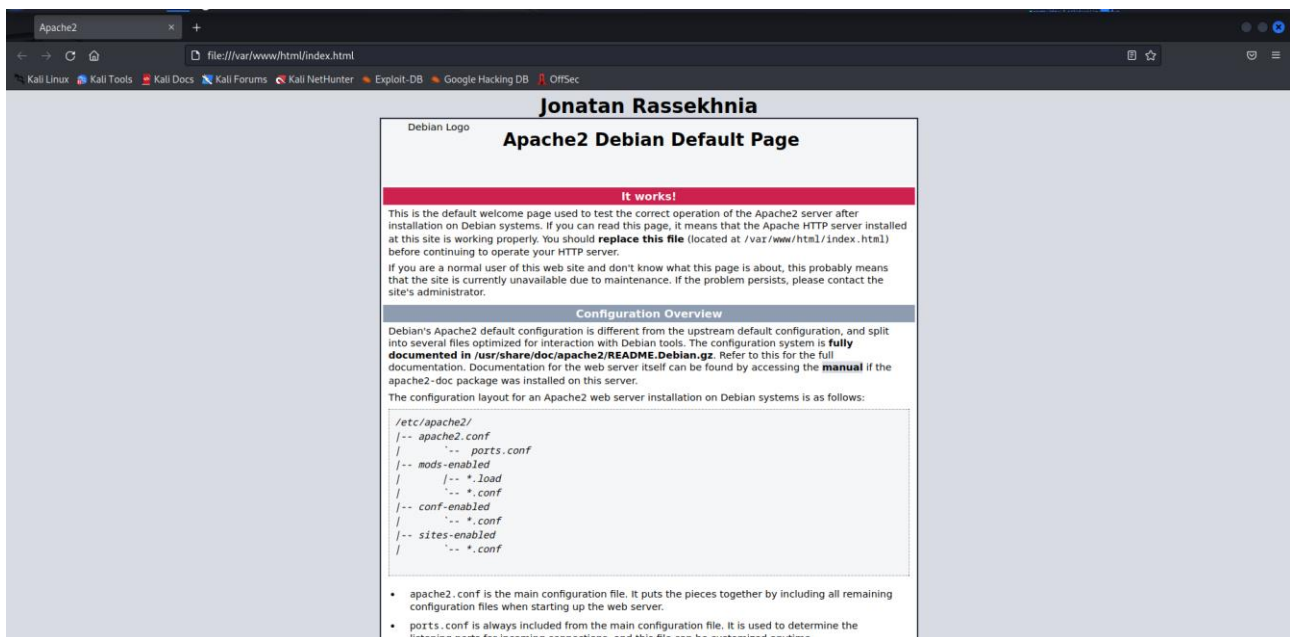
background-color: #D8DBE2;
}

```

[ Read 371 lines ]

Help	Write Out	Where Is	Cut	Execute	Location
Exit	Read File	Replace	Paste	Justify	Go To Line

*Illustration 1: index.html*



*Illustration 2: index-html edited*

```
jonatan@jsserver: ~
File Actions Edit View Help
GNU nano 6.2 /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/var/nfs/public [REDACTED](rw,sync,no_subtree_check)
/var/nfs/public [REDACTED](rw,sync,no_subtree_check)

[ Read 12 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut
^X Exit      ^R Read File ^\ Replace   ^U Paste
^T Execute
^J Justify
```

Illustration 3: Exports

```
jonatan@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/fstab  
# /etc/fstab: static file system information.  
#  
# Use 'blkid' to print the universally unique identifier for a  
# device; this may be used with UUID= as a more robust way to name devices  
# that works even if disks are added and removed. See fstab(5).  
#  
# systemd generates mount units based on this file, see systemd.mount(5).  
# Please run 'systemctl daemon-reload' after making changes here.  
#  
# <file system> <mount point> <type> <options> <dump> <pass>  
# / was on /dev/sda1 during installation  
UUID=b4216606-8b4f-486d-be6a-4ee070519486 / ext4 errors=r>  
# swap was on /dev/sda5 during installation  
UUID=526cdf4a-c3d2-46da-87aa-4ab0bb3eec33 none swap sw >  
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0  
192.168.1.28:/var/nfs/public /mnt/nfs-pubic nfs rw 0 0  
[ Read 17 lines ]  
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute  
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify
```

*Illustration 4: fstab*

```
jonatan@jsrver: /var/nfs/public
File Actions Edit View Help
GNU nano 6.2 Lab6.txt *
Lab 6, Testing Document.

Write Back here from Kali Linux via SSH
This is a confirmation from Terminal Kali Linux

Write Back from Windows Terminal via SSH
This is a confirmation from Terminal on Windows PowerShell

Jonatan Rassekhnia Lab 6
Test File Purposes

you become, the more you are able to hear

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

*Illustration 5: Lab6.txt Test Document*



## II. Next Attachment

```
jonatan@kali: ~  
File Actions Edit View Help  
  
(jonatan@kali)-[~]  
$ sudo apt-get update  
[sudo] password for jonatan:  
Get:1 http://ftp.acc.umu.se/mirror/kali.org/kali kali-rolling InRelease [41.2 kB]  
Get:2 http://ftp.acc.umu.se/mirror/kali.org/kali kali-rolling/main amd64 Packages [19.3 MB]  
Get:3 http://ftp.acc.umu.se/mirror/kali.org/kali kali-rolling/main amd64 Contents (deb) [44.7 MB]  
Get:4 http://ftp.acc.umu.se/mirror/kali.org/kali kali-rolling/contrib amd64 Packages [115 kB]  
Get:5 http://ftp.acc.umu.se/mirror/kali.org/kali kali-rolling/contrib amd64 Contents (deb) [172 kB]  
Get:6 http://ftp.acc.umu.se/mirror/kali.org/kali kali-rolling/non-free amd64 Packages [217 kB]  
Get:7 http://ftp.acc.umu.se/mirror/kali.org/kali kali-rolling/non-free amd64 Contents (deb) [928 kB]  
Fetched 65.4 MB in 12s (5,629 kB/s)  
Reading package lists... Done  
  
(jonatan@kali)-[~]  
$ sudo apt-get install apache2  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
apache2 is already the newest version (2.4.57-2).  
apache2 set to manually installed.  
The following packages were automatically installed and are no longer required:  
bluez-firmware catfish dh-elpa-helper firmware-ath9k-htc firmware-atheros  
firmware-brcm80211 firmware-intel-sound firmware-iwlwifi firmware-libertas  
firmware-realtek firmware-sof-signed firmware-ti-connectivity firmware-zd1211  
gir1.2-xfconf-0 kali-linux-firmware libcfitsio9 libgdal31 libmpdec3  
libnginx-mod-http-geoip libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter  
libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip libpoppler123  
libprotobuf23 libpython3.10 libpython3.10-dev libpython3.10-minimal libpython3.10-stdlib  
libtiff5 libzxingcore1 nginx-core python-pastedeploy-tpl python3-commonmark  
python3-speaklater python3.10 python3.10-dev python3.10-minimal ruby3.0 ruby3.0-dev  
ruby3.0-doc  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 125 not upgraded.  
  
(jonatan@kali)-[~]  
$ sudo nano /var/www/html/index.html  
  
(jonatan@kali)-[~]  
$
```

*Attachment 1: Task 1 Process*

```
jonatan@jserver: /var/nfs/public
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 26 15:01:29 2023 from [REDACTED]
jonatan@kali: ~
$ exit
Connection to [REDACTED] closed.
PS C:\Users\Johnny> ssh jonatan@[REDACTED]
jonatan@[REDACTED]: password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-72-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Sat May 27 01:33:05 AM UTC 2023

System load: 0.24658203125   Processes:           136
Usage of /: 29.3% of 24.44GB   Users logged in:     1
Memory usage: 3%             IPv4 address for enp0s3: [REDACTED]
Swap usage: 0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

 * Introducing Expanded Security Maintenance for Applications.
Receive updates to over 25,000 software packages with your
Ubuntu Pro subscription. Free for personal use.

https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

50 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sat May 27 01:28:25 2023 from [REDACTED]
jonatan@jserver:~$ cd /var/nfs/public
jonatan@jserver:/var/nfs/public$ ls
Lab6.txt  Linux.txt
jonatan@jserver:/var/nfs/public$ sudo nano Lab6.txt
[sudo] password for jonatan:
jonatan@jserver:/var/nfs/public$ jonatan@jserver:/var/nfs/public$
```

*Attachment 2: Login from PowerShell to Test the text file*