6-2019

BUILDING A SECURE NETWORK TEST ENVIRONMENT USING VIRTUAL MACHINES

Byungjin Lee

Building a secure network test environment using virtual machines

A Project

Presented to the

Faculty of

JHB College of Business and Public Administration,

California State University

San Bernardino

In Partial Fulfilment

of the Requirements for the Degree

Master of Information Systems and Technology

by

Byungjin Lee

June 2019

Building a secure network test environment using virtual machines

A Project

Presented to the

Faculty of

JHB College of Business and Public Administration,

California State University

San Bernardino

by

Byungjin Lee

June 2019

Approved by:

Joon Son, PhD., Advisor Date

Conrad Shayo, PhD, Reader

Jay Varzandeh, PhD., Chair, Information & Decision Sciences Department

Dedication

This project is wholeheartedly dedicated to my beloved parents, Minyeong Lee and Heejeong Min. They have been my source of inspiration and strength when I was riddled with the thoughts of giving up. My parents continually provide their moral, spiritual, emotional, and financial support. In addition, my beloved brother Byunggo Lee, sister in law Yuji Hyun, and niece Nawon Lee have been encouraging during my studies. To Annie, whose words of advice and encouragement led me to finish this project. Special thanks to Professor Conrad Shayo, for your instruction and support throughout the project. Lastly, I dedicated this project to Professor Joon Son, thank you for the guidance, strength, power of the mind, protection, and skills and for giving me a healthy life. All of these, I offer to you. Now I stand at another starting point. I will not forget the support and encouragement of many people and must do my best in my life.

Abstract

The objective of this project is to provide an overview of how to create a secure network test environment using virtual machines with Red Hat CentOS 7. Using virtual machines to create a secure network test environment simplify the workflow of testing several servers including network segmentation, network path redundancy, and traffic control using a firewall. This study suggests a set of guidelines for building a secure network test environment that includes a Domain Name Server (DNS), Web Server, File Transfer Protocol (FTP) Server, and a firewall. The documentation provided in this project is primarily useful for IT students looking to recreate a similar environment of their own and to practice special skills needed within their field of study.

Keywords: Secure network test environment, Virtual machine, Red Hat CentOS 7, Windows, DNS, Web Server, FTP, Firewall, Iptables

Table of Contents

1. Introduction

With the rapid development of electronic devices and the Internet of Things, there are numerous network security threats in our daily lives. For decades, network security exploits happen frequently. These threats include malicious programs, worm, Trojan horse, spyware, rootkit, and ransomware. For this reason, most organizations require secured servers within their network to protect their data.

Students are met with several major challenges when it comes to building servers, one being budget constraints. Nevertheless, virtual machines make it possible to create a diversified virtual network lab at a minimal fee. Tsihouridis et al. (2014) noted that "students were able to use both real and virtual lab according to their educational needs." "A virtual machine (VM) is a logical process (most often an operating system) that interfaces with emulated hardware and is managed by an underlying control program. Originally, virtual machines were run on mainframes to provide resource multiplexing and isolation" (Gum, 1983). "Most modern virtual machine systems use the virtual machine monitor (VMM) model for managing and controlling individual virtual machines" (Zhao, Borders & Prakash, 2009).

Although Windows is a very popular system, this virtual environment also focuses on Linux systems. Linux is a completely open-source operating system that is more secure in comparison to other operating systems, even Windows. "The Linux OS has had about 60 to 100 viruses listed until date. But now-a-days unfortunately none of these are actively working. On the other hand, there are more than 60K kwon viruses in Windows." (Abhilash & Abhinay, 2015). For example, Linux does not require the installation of an anti-virus program because the virus cannot execute without the administrator's password. This system also offers Iptables, an application that allows a system administrator to configure the specific access control rules provided by the Linux kernel's Netfilter framework. The objective of this project is to build a secure network test environment utilizing virtual machines consisting of the Linux system as well as Windows system for IT students looking to expand their skills at minimal costs.

2. Problem Statement

The objective of this project is to provide IT students with suggested guidelines for building their own secure network test environments using virtual machines. The documentation includes how many machines are required, how the servers should be configured, which open-source tools are needed and how to ensure the secure network works. This allows student users to practice their desired network security skills at the minimal costs of $918 compared to using physical machines that may cost well over $8,000 depending on the types and number of servers used. Detailed costs can be reviewed in the Project Materials section below.

3. Project Materials

The design of this project requires one physical computer that has multiple virtual machines running on it. The hardware specifications are as follows:

< Physical Machine > - $ 768.00

- CPU – Intel ® Core ™ i7-7500U @ 2.70GHz 2.90
- RAM – 12 GB
- Operating System – Windows 10 64-bit
- Hardware – Solid State Drive 1TB

<VMware Workstation Pro 15> - $150.00

- Memory - 2 GM

- Processors - 1
- Hard Disk (SCSI) –80 GB
- CD/DVD (IDE) – CentOS 7 and Auto Detect
- Network Adapter –NAT and Bridged

As previously mentioned, it is more cost effective to build a secure network environment using virtual machines. The total cost for building the physical test environment using the physical machines shown below is $ 9,291.00. Here is an example price comparison of how much building a similar environment using physical machines would be.

< Server > requires 2 machines

- PowerEdge R640 Rack Server - $1,969.00

< Firewall > requires 1 machine

- Cisco Meraki MX100 Cloud Managed Security Appliance - $3,447.00

< Switch > requires 2 machines

- Netgear JGS524 ProSafe 24 Port Gigabit Ethernet Desktop Switch - $185.00

< Physical computer > requires 2 machines

- HP Pavilion 15.6-inch FHD 1080P Laptop PC, Intel Core i7 Processor, 12GB Memory, 1TB Hard Drive, Backlit Keyboard, Webcam, Bluetooth, USB 3.1, Windows 10 - $ 768.00

The thus, building a test environment using virtual machines is significantly more cost-effective than a test environment using physical machines. There are selected the price based on Amazon website.

4. Project Structure

The structure of this project is for building a secure network test environment using virtual machines, as shown in Figure 1.

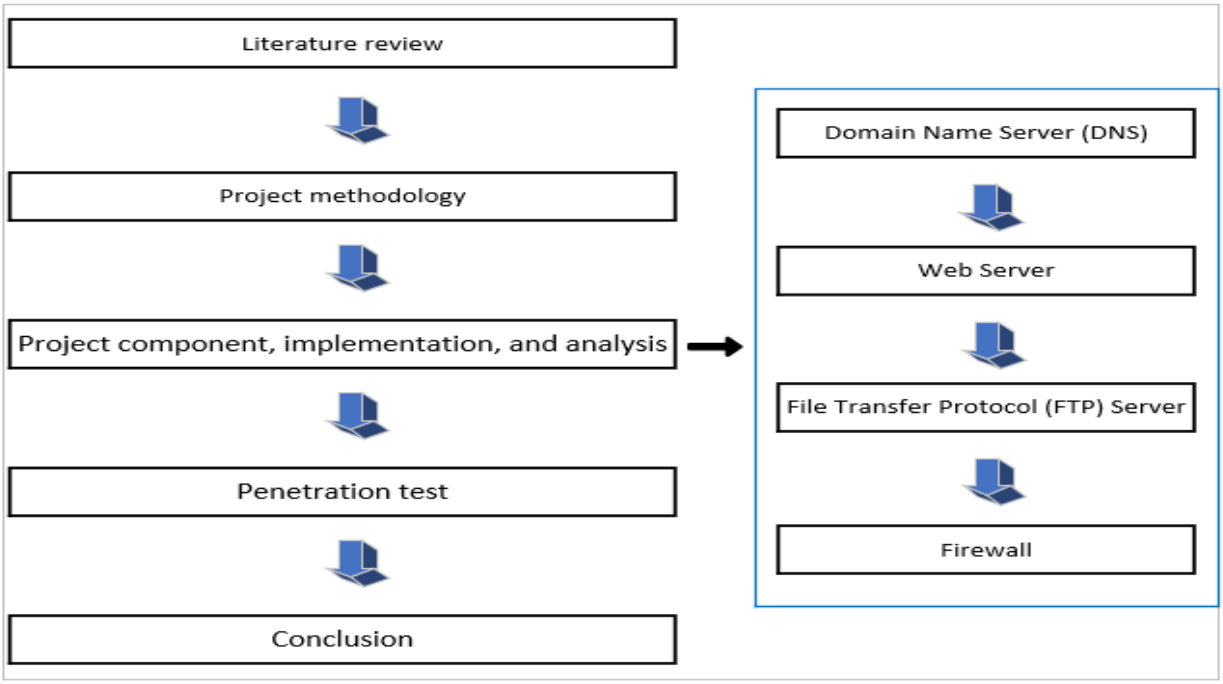


Figure 1. The flow of the entire project.

5. Literature Review

This project presents the experiences from using virtual machines to test a secure network environment on Linux or Windows operating systems. “Virtual Machine technology has increased in popularity over the last few years because production environments have had success using Virtual Machines to reduce many types of operational costs” (Baker Hart, 2002). In addition, “with limited educational resources and dramatically increasing number of students, efficient and timely access to physical computer labs becomes more difficult [for students]” (Alharbi, 2018). Therefore,

it is possible for a virtual machine to support self-contained and potable security experiences to improve student technical skills because it is essentially another computer inside a physical computer. “The resources of lab computer systems can be utilized more effectively, multiple environments can be configured quickly and easily, and access to external resources can be provided without permitting attacks to those resources” (Bulbrook, 2006).

A virtual machine requires an operating system such as Linux or Window and several programs may operate separately on the same physical computer. Although one may think guidelines for building a secure network test environment are readily available, it has proven to be difficult to come across one. As a result, building a secure network test environment using virtual machines and the Linux operating system, as presented in this project, provides a unique, cost-effective and easy guide for IT students to improve their technical skills.

6. Project Methodology

A configuration of network servers run based on access control policy specification implemented using virtual machines. To accomplish the final network architecture, the 3 test environments such as Domain Name Server (DNS), Web Server, and File Transfer Protocol (FTP) Server should be prerequisites to execute a secure network test environment. The goal of this architecture is to implement a test environment through several virtual machines within the private network and one virtual machine with the public network. For example, Server A has a private network of 10.0.0.0/24 with bridged and a public network of 192.168.111.0/24 with Network

Address Translate (NAT). Server B and Linux client have a private network of 10.0.0.0/24. On the other hand, Window client has a public network of 192.168.111.0/24.

7. DNS, Web server, and FTP Components and Analysis

The final network architecture shown in Figure 2, building the 3 virtual environments such as Domain Name Server (DNS), Web Server, and File Transfer Protocol (FTP) Server is necessary for a secure network test environment, which I will build in a later chapter.

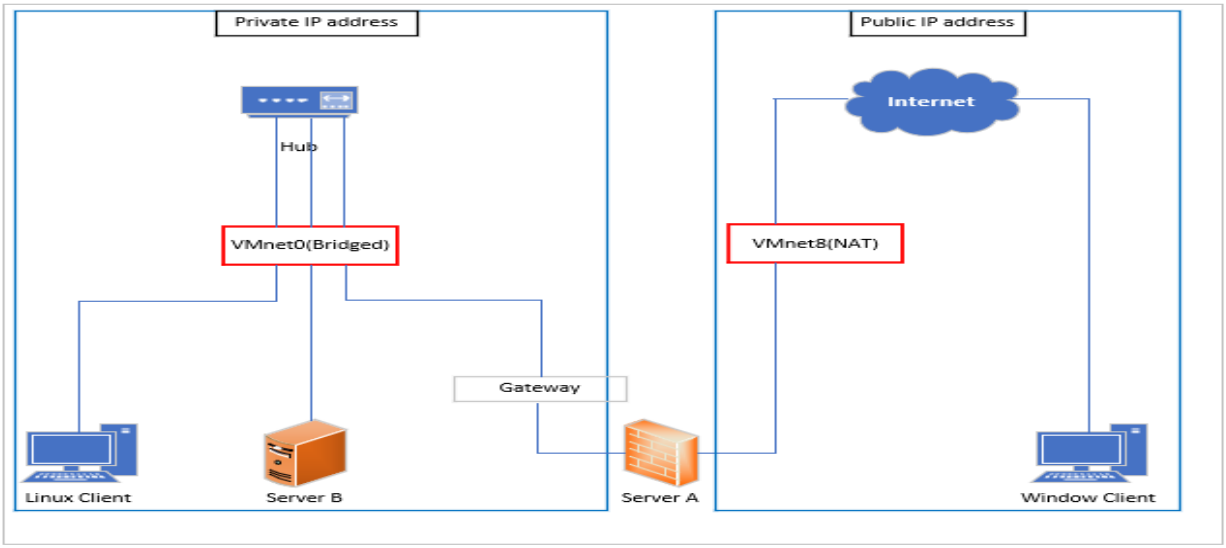


Figure 2. Secure network test environment using virtual machines

7.1 Test case for Domain Name Server (DNS)

Domain Name Server (DNS) serves a specific purpose in this project and is necessary for a secure network test environment. A DNS is a computer server that contains a database of public IP addresses and associated hostnames. Mainly, a DNS is used to resolve or translate the requested

IP addresses (Margaret, 2019). In order to build the DNS server, two things are required: a Caching-only name Server and a Master Server. A Caching-only name Server, in a secured environment with a strict firewall implementation, allows local clients to obtain name service without having to pierce the firewall. A Master Server is authoritative and contains a complete copy of all information for all hosts in the DNS domain. Lastly, a Web server processes incoming network requests over HTTP (Port 80).

7.1.1 Domain Name Server (DNS) – Components

As shown in Figure 3, Server A (10.1.1.1) has the Caching-only Name Server and Master Server. Server B (10.1.1.20) has the Web Server. Both server A (10.1.1.1) and server B (10.1.1.20) support for a highly configurable secure network environment. Detailed configurations can be reviewed in the Appendix (Domain Name Server (DNS) – Configurations).

- Server A – Caching-only Name Server and Master Server
- Server B – Web Server

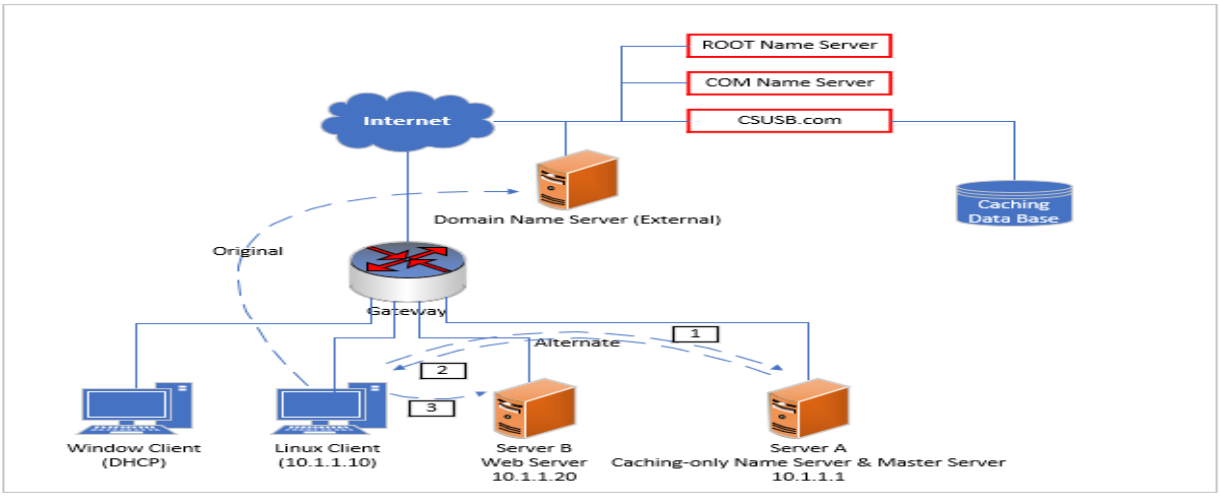


Figure 3. Domain Name Server (DNS)

7.1.2 Domain Name Server (DNS) – Implementation and Analysis

In order to implement the Domain Name Server (DNS), the Linux client requests not only the IP address to a Caching-only name Server which translates from A Uniform Resource Locator (URL) to IP address but also a Master Server which manages all information for hosts in the DNS domain then the Linux client can access the Web Server.

I tested and reviewed the proposed configuration and components to ensure they work as expected. As a result, I was able to verify the Domain Name Server (DNS) through Linux Client (10.1.1.10) as shown in Figure 4. Both Server A (10.1.1.1) and Server B (10.1.1.20) were successful through the Linux client (10.1.1.10). Here are the steps to verify if the system is correctly configured:

Step 1. A user in the Linux client (10.1.1.10) types www.CSUSB.com

Step 2. DNS into Server A (10.1.1.1) responds to Linux client (10.1.1.10)

Step 3. Linux client with IP (10.1.1.10) accesses Web Server into Server B (10.1.1.20)

Here is the analysis of Linux Client as shown in Figure 4

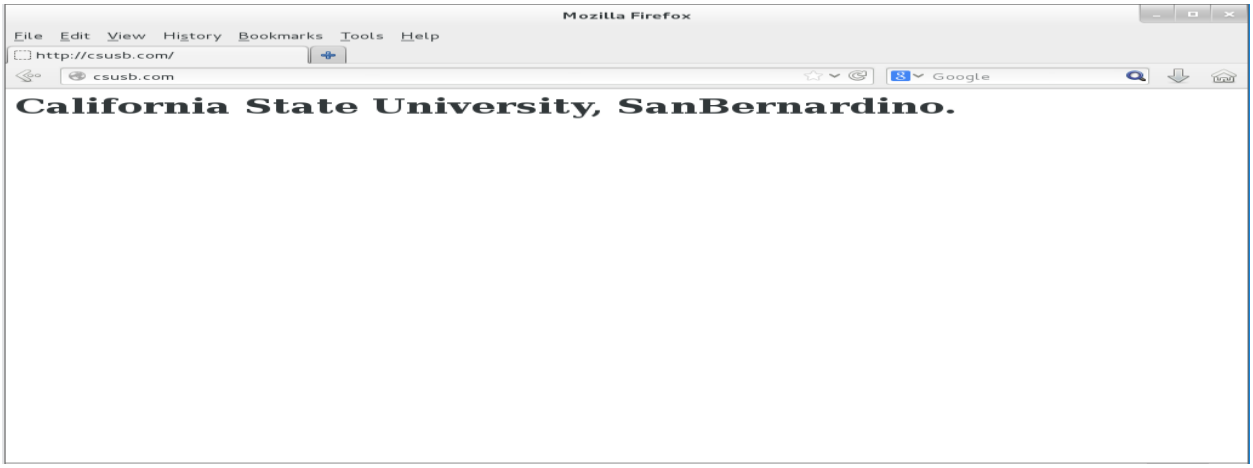


Figure 4. Linux Client can access the Sever B

7.2 Test case for Web Server

Web Servers serve a specific purpose in this project and is necessary for a secure network test environment. A Web Server is a computer server that stores and transmits data via the Internet (Art, 2010). They are mainly used to ultimately dictate the data upon the request of guest's browser. In order to build Web server, there are two things required. The first is a set of software subsystems or components needed to create a complete web hard (Pydio). The software needed include Linux, Apache, PHP, and MariaDB (LAPM). The second is a Web hard (Pydio). A web hard is an open-source file sharing for the practice of providing access to an online storage service and synchronization software that runs on the own server. (Kanghyun & Jongmoon. 2013).

7.2.1 Web Server – Components

The main goal of setting up this environment is for IT students to gain experience building a secure web server. More specifically, these students will learn how to configure a web hard. As shown in Figure 5, Server B (10.1.1.20) has the Linux, Apache, PHP, and MariaDB (LAPM), as well as the Web hard. Server B (10.1.1.20) supports for a highly configurable environment. Detailed configurations can be reviewed in the Appendix (Web Server – Configurations).

- Server B –Linux, Apache, PHP, and MariaDB (LAPM), and Web hard

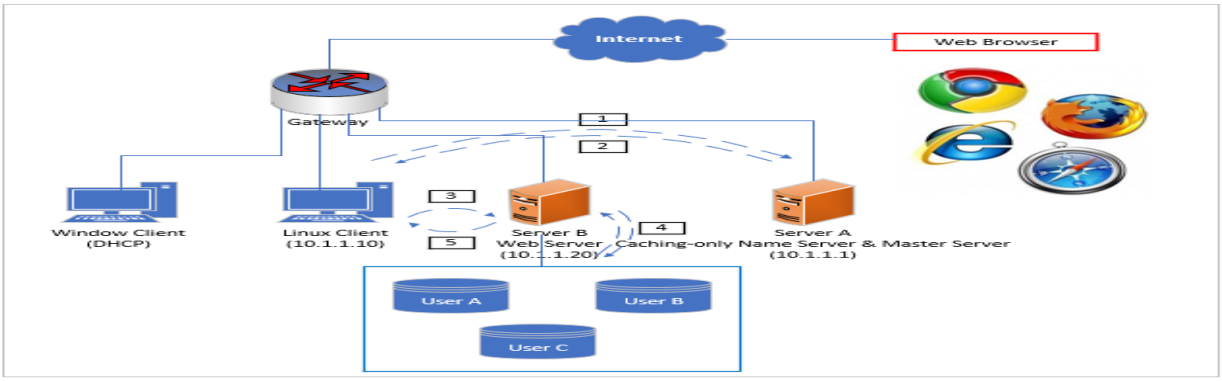


Figure 5. Web Server

7.2.2 Web Server – Implementation and Analysis

In order to implement the Web Server, the Linux client requests the IP address to a Domain Name Server (DNS) then the Linux client can access a Web hard (Pydio) to share an open source file to an online storage service and synchronization software.

I tested and reviewed the proposed configurations and components to ensure they work as expected. As a result, I was able to verify if the Web server correctly functions through the Linux client. Here are the steps to verify if the system is correctly configured:

Step 1. A user in the Linux client (10.1.1.10) types 10.1.1.20/webhard/

Step 2. DNS into Server A (10.1.1.1) responds to Linux client (10.1.1.10)

Step 3. Log into Pydio (10.1.1.20) through identification (CSUSB) and password (1234)

Step 4. Admin access - admin in admin identification (admin) password (12345678)

Global options - English in default language

Configurations storage – user identification (xeUser) and password (1234)

Add some users – identification CSUSB, email (admin@csusb.com), and password (1234)

Step 5. Linux client (10.1.1.1) accesses Pydio (10.1.1.20)

Here is the analysis of Linux Client as shown in Figure 6

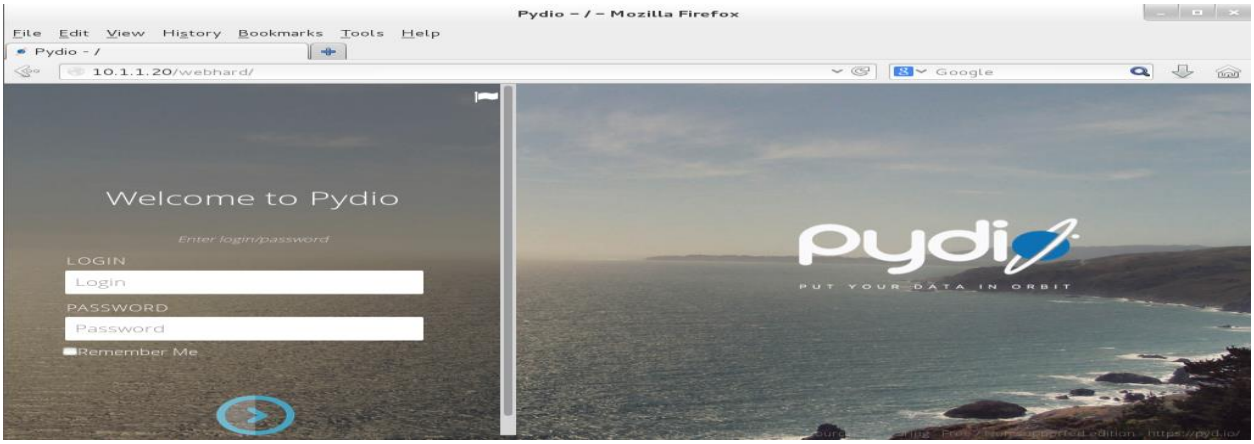


Figure 6. Linux Client can access the Sever B

7.3 Test case for File Transfer Protocol (FTP) Server

File Transfer Protocol (FTP) Server serves a specific purpose in this project and is necessary for a secure network test environment. An FTP Server is a computer server that communicates between the client and the server to control data transmission. Mainly, FTP uses credentials in the form of username and password in order to allow or deny authorization. In order to secure an FTP server, Very Secure FTPD (VSFTPD) is necessary because it is designed to secure systems against most common attacks, providing for more security, scalability and simplicity (Chris, n.d.).

7.3.1 File Transfer Protocol (FTP) – Components

As shown in Figure 7, Server B (10.1.1.20) has Very Secure FTPD (VSFTPD). Server B (10.1.1.20) supports for a highly configurable security environment. Detailed configurations can be reviewed in the Appendix (File Transfer Protocol (FTP) – Configurations).

- Server B – Very Secure FTPD (VSFTPD) for server

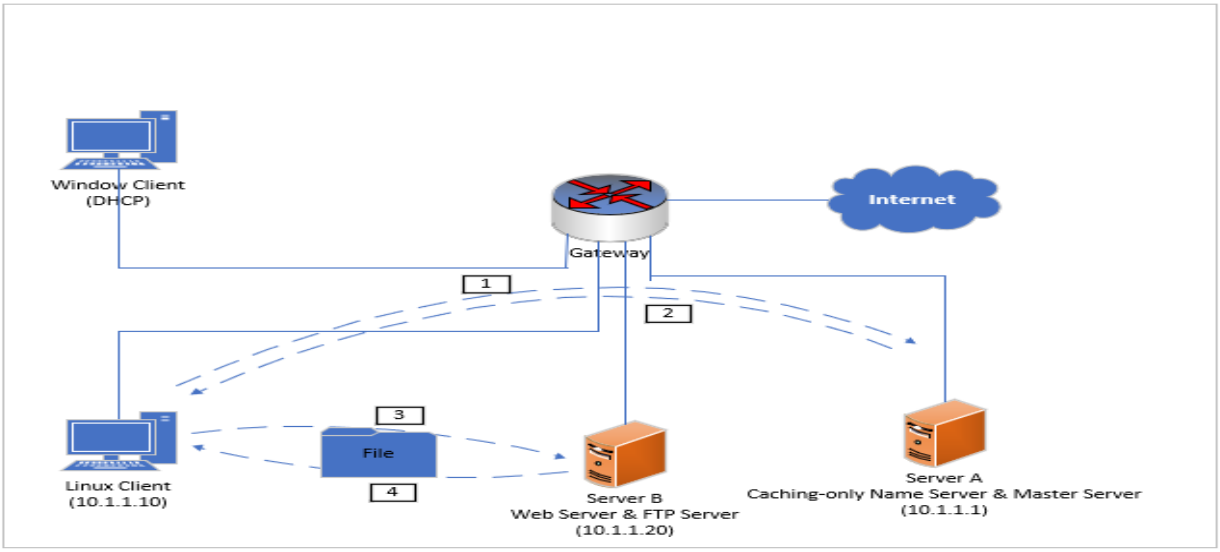


Figure 7. File Transfer Protocol (FTP)

7.3.2 File Transfer Protocol (FTP) - Implementation and Analysis

In order to implement the File Transfer Protocol (FTP), the Linux client requests the IP address to a Domain Name Server (DNS) then the Linux client can access the File Transfer Protocol (FTP) to control data transmission to allow or deny authorization.

I tested and reviewed the proposed configuration and components to ensure they work as expected. As a result, I was able to verify the File Transfer Protocol (FTP) through Linux Client as shown in Figure 8. Server B was successful through Linux Client. Here are the steps to verify if the system is correctly configured:

Step 1. A user in the Linux client (10.1.1.10) machine types ncftp 10.1.1.20

Step 2. DNS into Server A (10.1.1.1) responds to Linux client (10.1.1.10)

Step 3. Logging in File Transfer Protocol (FTP) server (10.1.1.20)

Step 4. Linux client (10.1.1.10) accesses successfully

Here is the analysis of Linux Client as shown in Figure 8

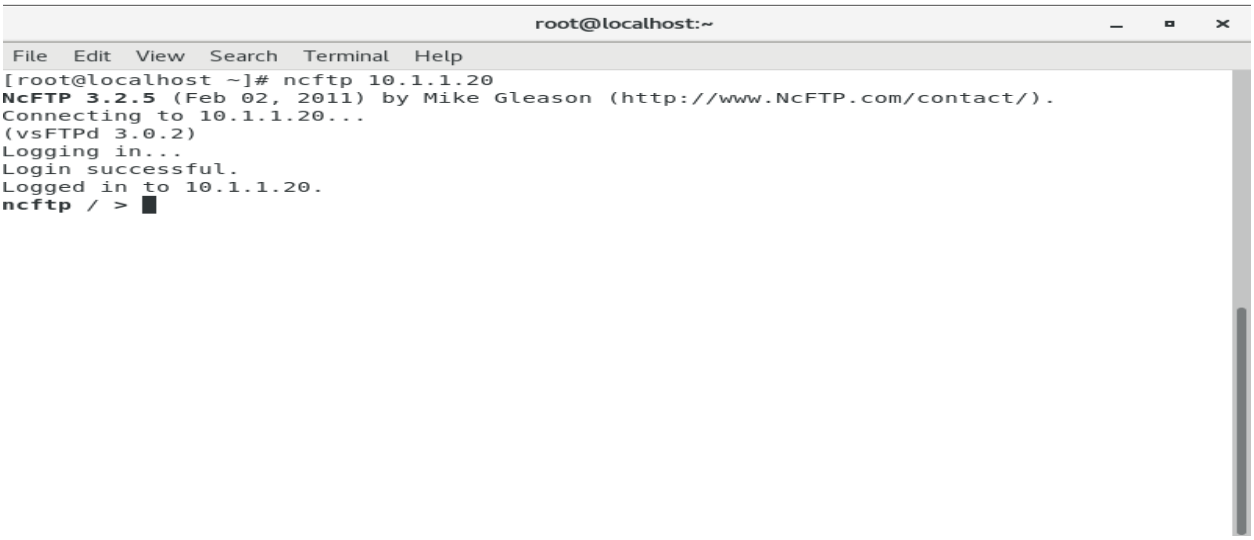


Figure 8. Linux Client can access the Sever B

8. Firewall Components, Implementation, and Analysis

Now, students are ready to build a secure network test environment and should successfully reach these objectives:

1. Translate security policy specification to actual implementation via Iptables.
2. Test if the firewall correctly enforces the security policy specifications and explore the vulnerabilities of the firewall.

8.1 Test case for Firewall

Firewalls serve a specific purpose in this project and is necessary for a secure network test environment. A Firewall is as security setting that is intended to act as a barrier or shield from unauthorized internet users from accessing a private network. Mainly, a Firewall is used to monitor the packets of data between servers and clients to accept or reject request based on access control policy specification such as Iptables. Iptables is a generic table structure for access control specification rules as part of the netfilter framework that facilitates Network Address Translation (NAT), packet filtering, and packet mangling in the Linux operating systems. (Margaret, 2005)

8.1.1 Access control policy specifications

I configured three Iptables in the Firewall which can enforce the access control specification. If I do not configure correctly both letters and spaces as well, Iptables will not operate. Detailed configurations can be reviewed in the Appendix (Firewall – Configurations). Here is a list of access control rules to enforce:

Rule 1. The users (any machine) within a private network can access the public network.

Rule 2. If the Window Client within a public network attempts to access the Web Server or the FTP Server, which is installed on Server B, the Firewall, that is installed on Server A, allows access into Server B within the private network.

Rule 3. The users (any machine) within a public network cannot access the private network other than the exception shared in Rule 2.

8.1.2 Firewall - Components

As shown in Figure 9, Server A has two IP addresses: one for private network address of 10.1.1.0/24 and another one for public network address of 192.168.111.0/24. It also contains the Iptables. Server B (10.1.1.20) runs the Web Server and FTP Server, which was installed in the previous chapter. Both Server A and Server B provide for a highly configurable security environment as shown in Figure 9. Detailed configurations can be reviewed in the Appendix (Firewall – Configurations).

- Server A – Two IP addresses: one for private network address of 10.1.1.0/24 and another one for public network address of 192.168.111.0/24, and iptables
- Server B – Web Server (Port 80) and FTP Server (Port 21) run two services
- Window Client – Nmap

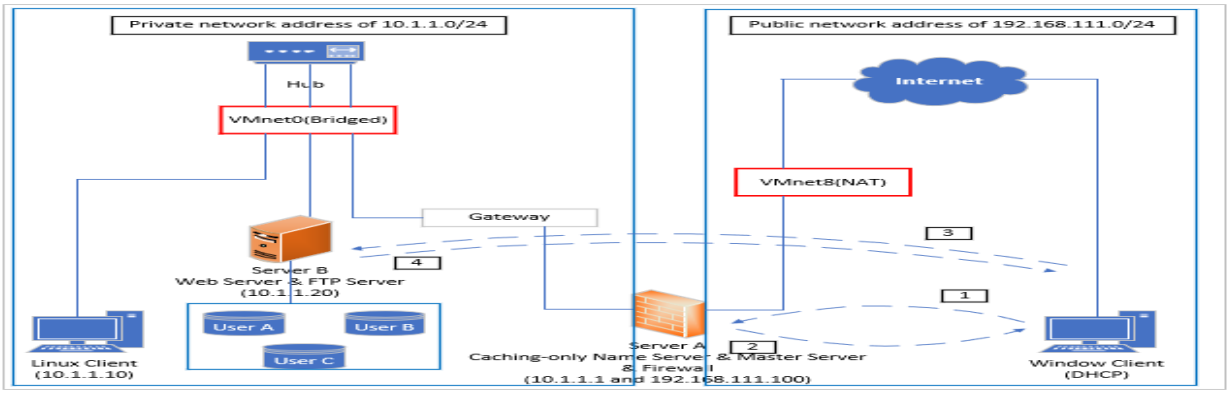


Figure 9. Firewall

8.1.3 Firewall – Analysis

I tested and reviewed the proposed configuration and components to ensure they work as expected. The following three subsections will explain how I tested and verified if the firewall (Iptables) is able to correctly enforce the access control rules. Here is the test result of Rule 1 for access control specifications as shown in Figure 10:

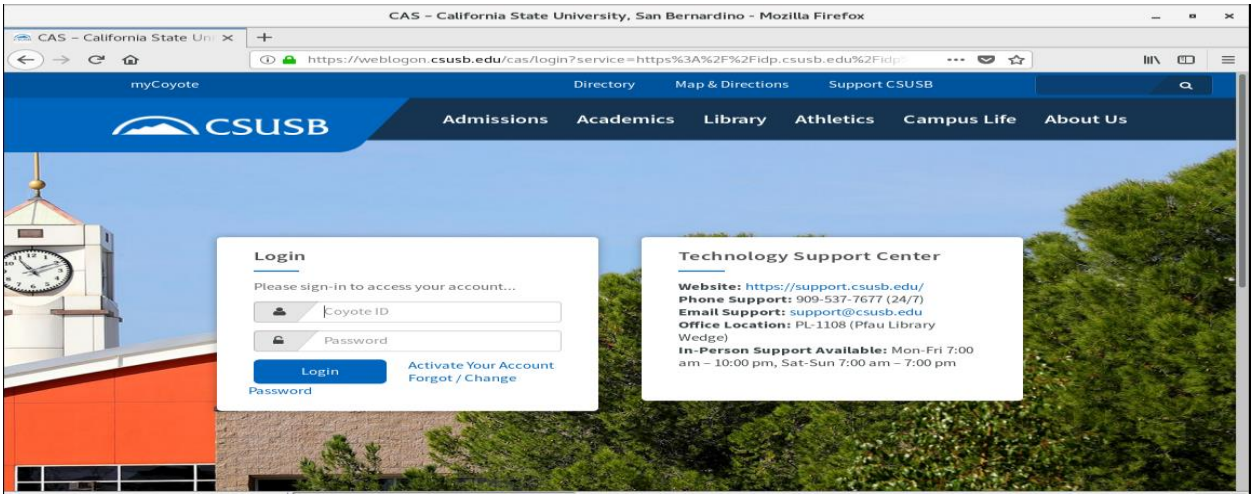


Figure 10. Linux client with private network address accesses the public network address

Here is the test result of Rule 2 for access control specifications as shown in Figure 11 and Figure 12, Figure 13, and Figure 14:

< Web Server >

Step 1. A user in the window client (DHCP) types 10.1.1.20

Step 2. Firewall in Server A checks Iptables then responds to window client (DHCP)

Step 3. Window client (DHCP) tries to access Web Server (10.1.1.20)

Step 4. Window client (DHCP) can access successfully

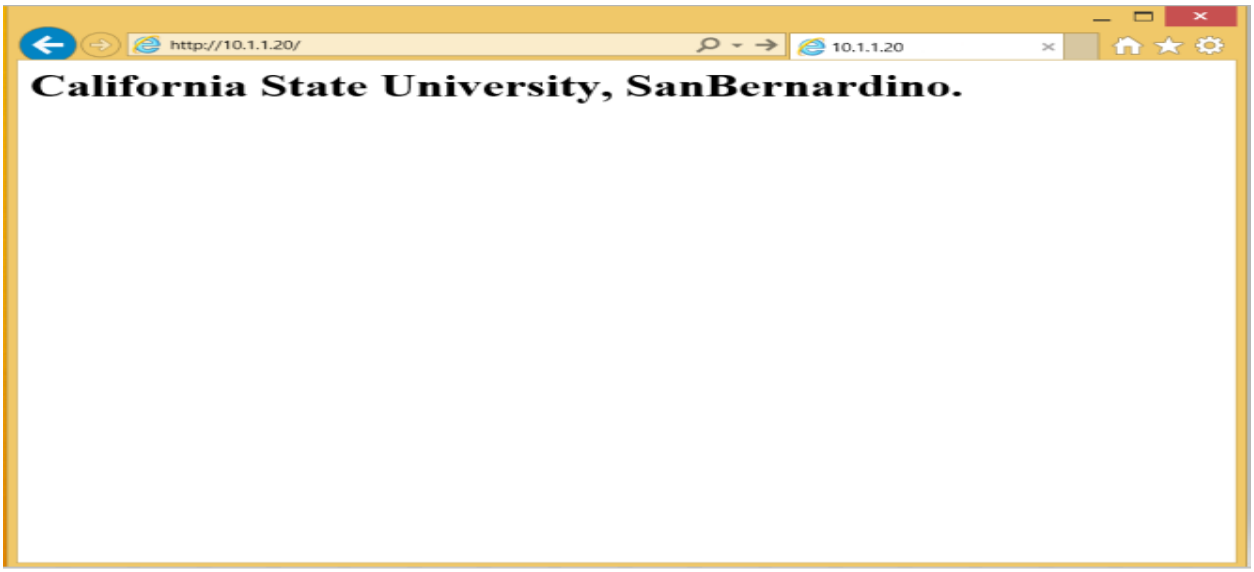


Figure 11. Window client with public network address accesses the Web server with private network address

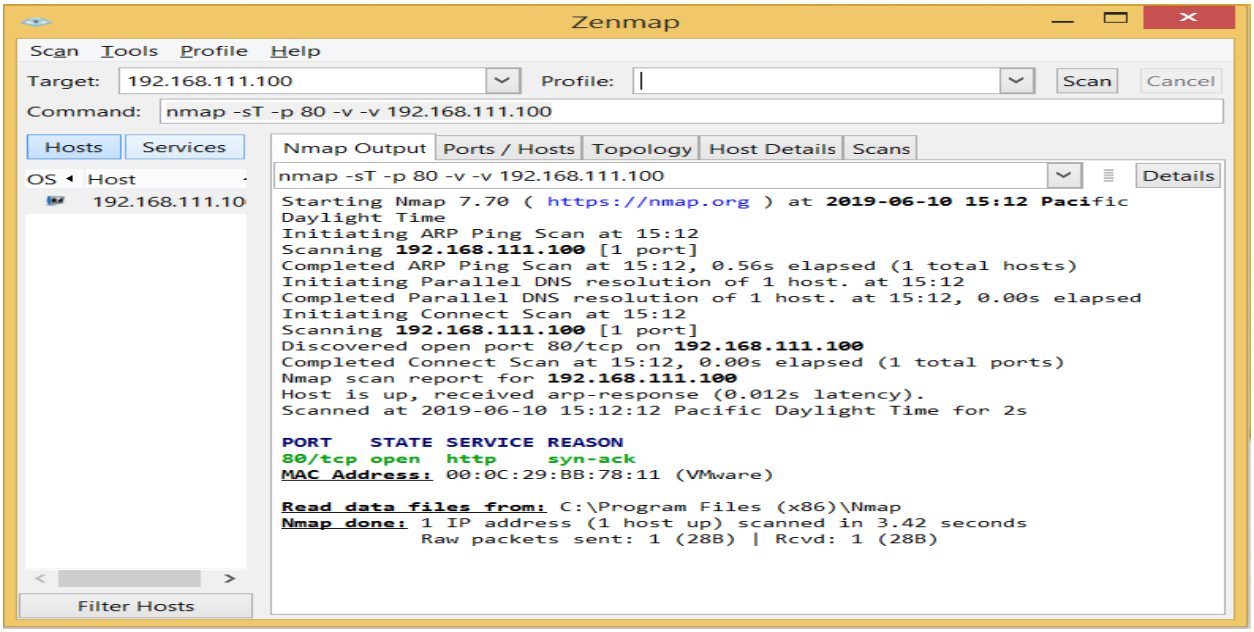


Figure 12. Nmap was used to scan HTTP (Port 80) state

< File Transfer Protocol (FTP) >

Step 1. A user in the window client (DHCP) types host (10.1.1.20), username (anonymous), and password (1234)

Step 2. Firewall in Server A checks Iptables then responds to window client (DHCP)

Step 3. Window client (DHCP) tries to access File Transfer Protocol (FTP) server (10.1.1.20)

Step 4. Window client (DHCP) can access successfully

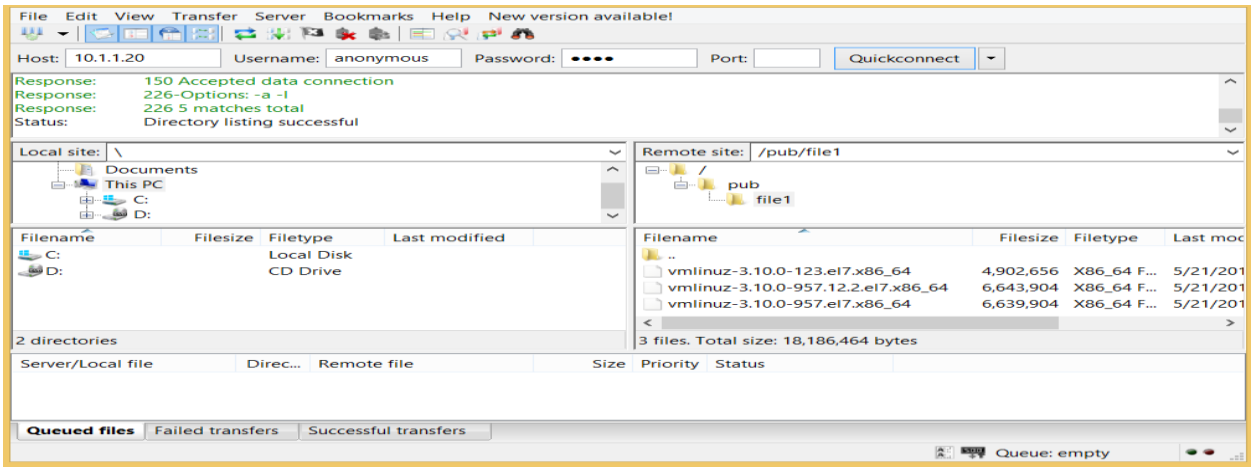


Figure 13. Window client with public network address accesses the File Transfer Protocol (FTP) server with private network address

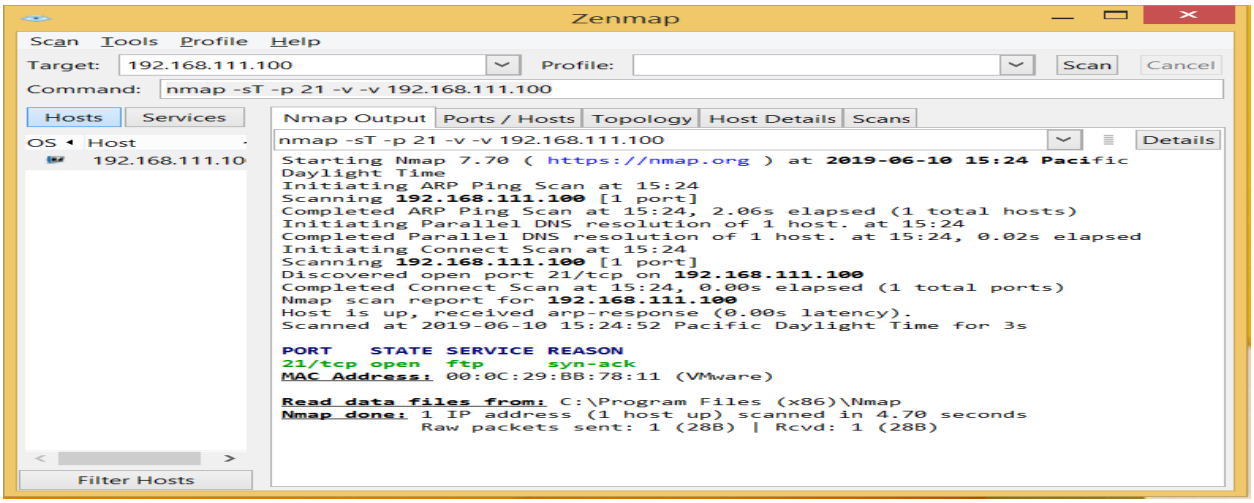


Figure 14. Nmap was used to scan FTP (Port 21) state

9. Penetration Test

I wanted to see if my security test environment can be efficiently used for educating IT students. In order to do so, I had two Cyber Security Graduate Students from California State University, San Bernardino conduct penetration tests on my secure network test environment to see if they could find any vulnerabilities.

9.1 Vulnerability testing done by Cyber Graduate Students

I first asked them to recreate the steps and commands I previously used on Nmap, as shown in the Firewall-Analysis section of this project. Then, both of them used other Nmap commands of their choice to further try and scan for vulnerabilities of the firewall. This process is needed to see if the firewall is at least faithfully following the access control rules and so that the Graduate Students get familiarized with the secure network test environment, as shown in Figure 15, and Figure 16.

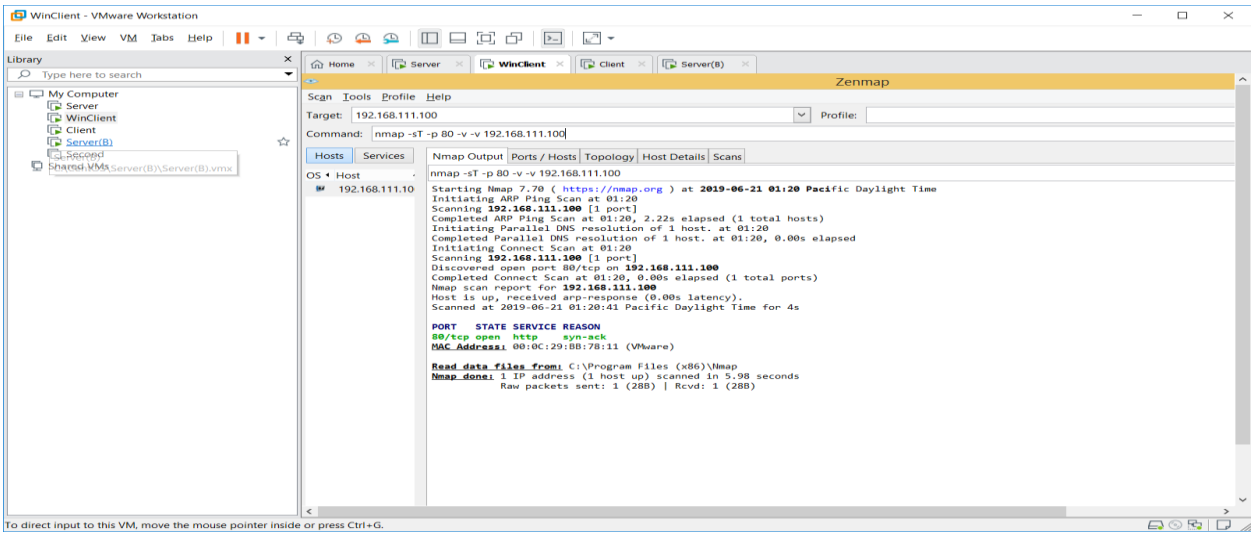


Figure 15. Penetration test- Nmap

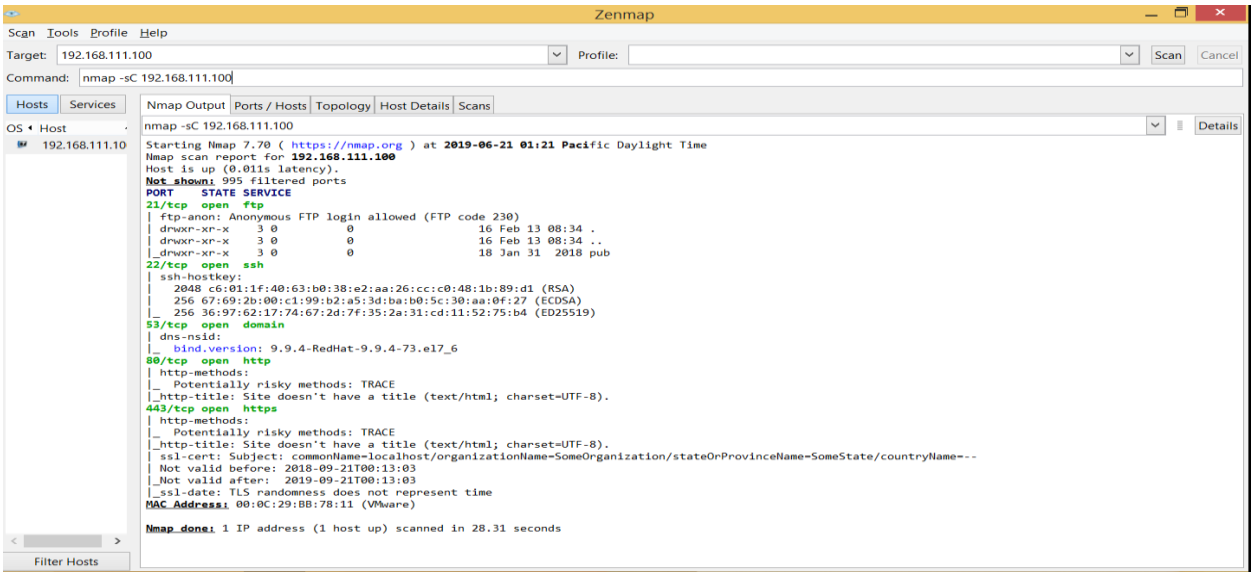


Figure 16. Scan for vulnerabilities of the firewall

9.2 SYN flood

I gave them Iptables rules and had them scan for vulnerabilities of the firewall again. They were able to find two vulnerabilities of Iptables. One of them is that the Firewall cannot protect against SYN flood attacks.

9.2.1 SYN flood attacks – Test

The Graduate Students used Hping3 and Wireshark security tools to analyze its vulnerability of Iptables. Finally, they tested SYN flood attacks from the Linux Client with a public IP address to the Web Server with a private IP address. They then verified the Firewall does not fully protect HTTP from SYN flood attacks (200,471 packets transmitted, 186,438 packets received, 7% packet loss) as shown in Figure 17.

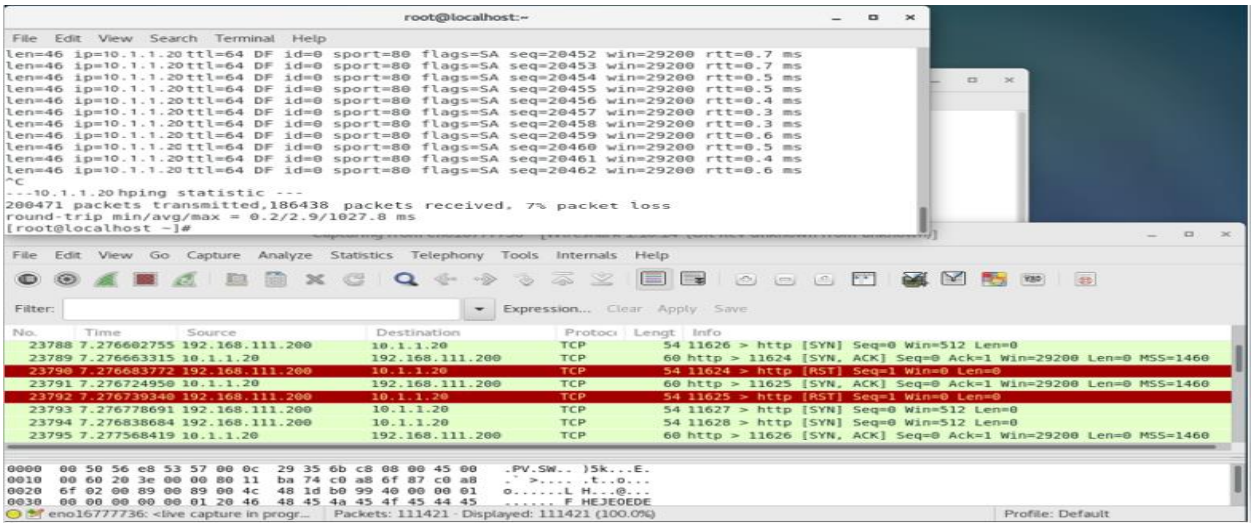


Figure 17. SYN flood attacks to the Firewall were successful

9.2.2 SYN flood attacks – Solution

They made new Iptables to protect against SYN flood attacks. The Graduate Students conducted another test using these new Iptables to verify if it works. As a result, the firewall now protected HTTP against SYN flood attacks (30,926 packets transmitted, 3624 packets received, 88% packet loss) as shown in Figure 18.

- iptables -I INPUT -p tcp --dport 80 -i eth0 -m state --state NEW -m recent --update --seconds 1 --hitcount 10 -j DROP

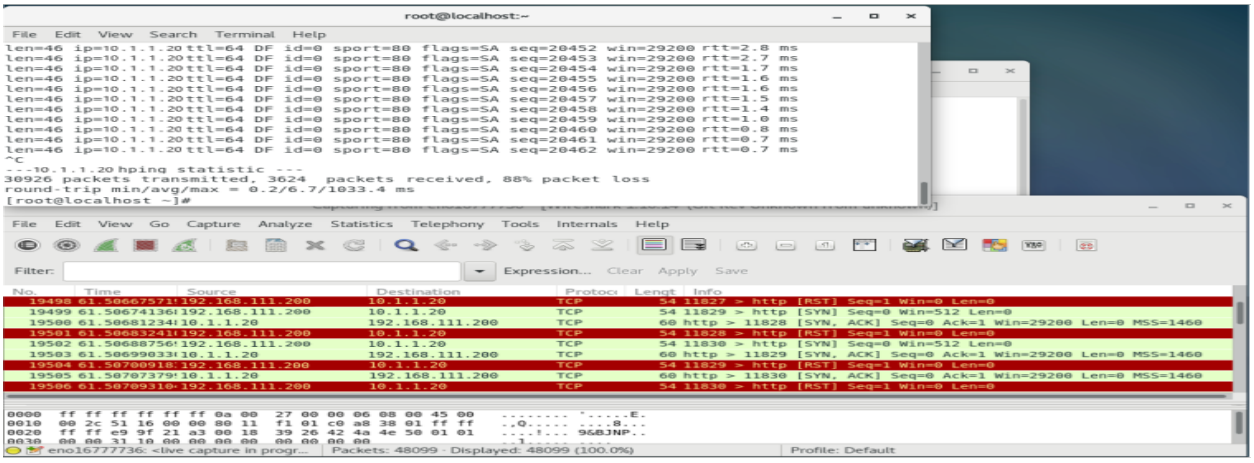


Figure 18. SYN flood attacks failure – Blocked by the Firewall

9.3 SYN-ACK

They tested against SYN-ACK attacks from the Linux Client with a private IP address to the Firewall with a public IP address to determine if it is stateful or stateless.

9.3.1 SYN-ACK attacks - Test

The Graduate Students used Hping3 and Wireshark security tools to analyze its vulnerability of Iptables. Finally, they sent several SYN-ACK packets from a private IP address to a public IP address. All SYN-ACK attacks (sent 242,335 Tx packets, received 242,335 Rx packets) passed the Firewall without first sending SYN of a Three-way handshake to the Firewall with a public IP address, as shown Figure 19.

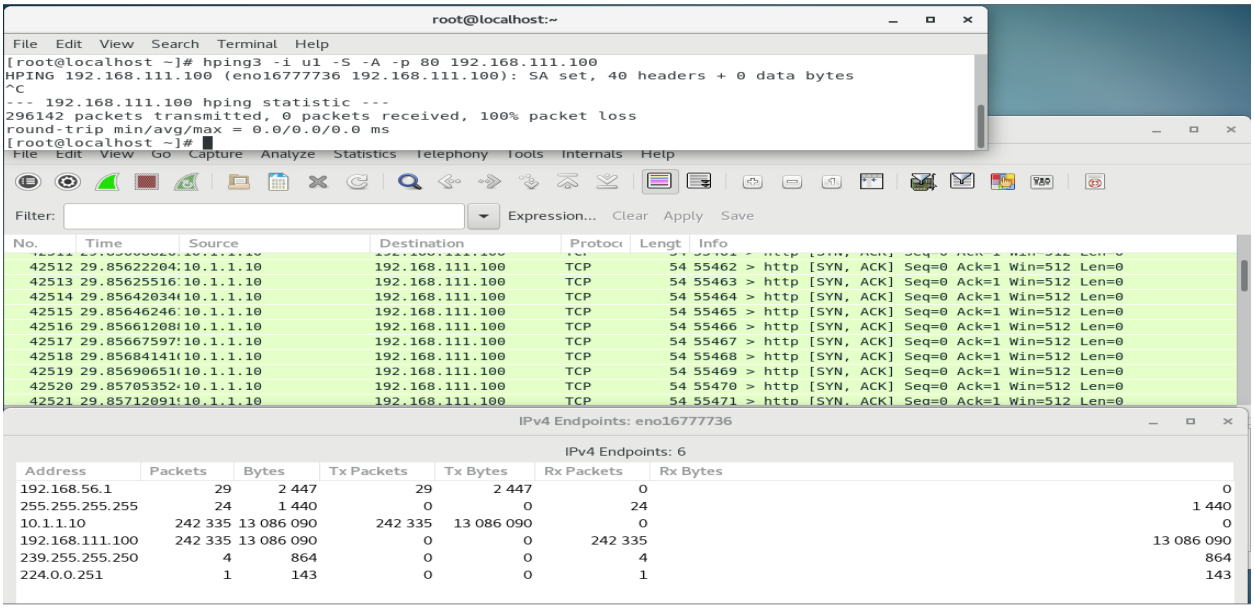


Figure 19. SYN-ACK attacks to the Firewall were successful

9.3.2 SYN-ACK attacks – Solution

They made new Iptables to protect against SYN-ACK flood attacks. The Graduate Students conducted another test using these new Iptables to verify if it also works. As a result, the firewall now blocked SYN-ACK flood attacks (sent 113,933 Tx packets, received 0 Rx packets) as shown Figure 20.

- iptables -A OUTPUT -p tcp --tcp-flags SYN.ACK SYN,ACK -m state --sate NEW -j DROP

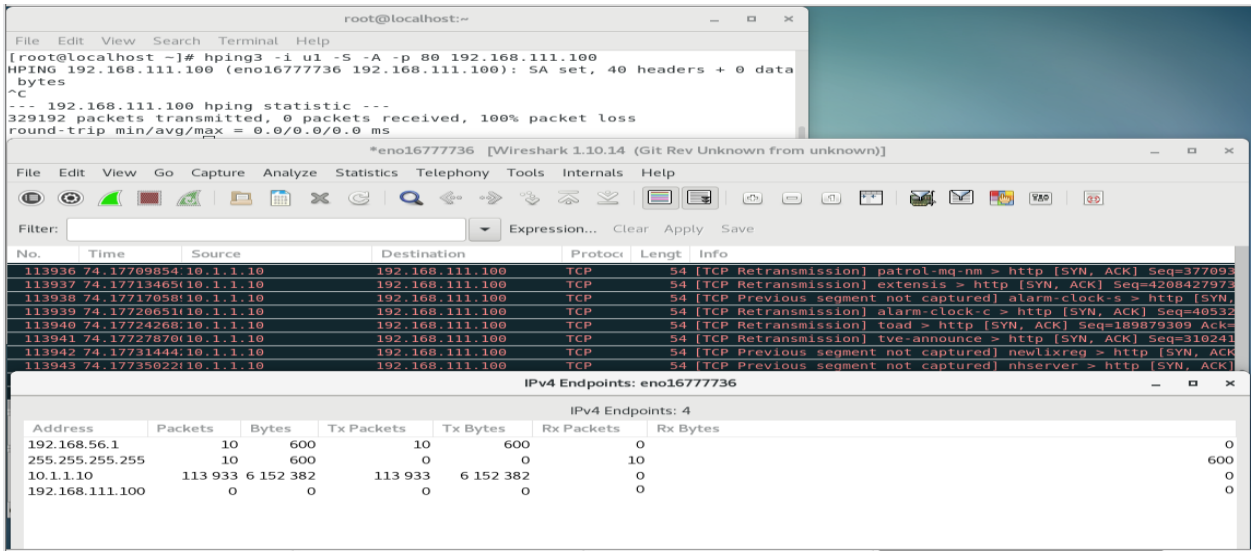


Figure 20. SYN-ACK attacks failure – Blocked by the Firewall

As a result, both Cyber Security Graduate Students were successful in penetrating the system. Although the proposed configurations enforce security, it is impossible to create a 100% vulnerability-free system. Nonetheless, security can always be improved upon, which was evident through the Graduate Students' penetration tests. This clearly shows that the test environment can be an effective tool for IT students to learn from. The Cyber Graduate Students were able to secure the Firewall by adding new Iptables rules to prevent SYN flood attacks and SYN-ACK attacks.

Conclusion

This set of guidelines to build a secure network test environment using virtual machines provide guidance for IT students. This study provides the workflow of testing several servers including network segmentation, network path redundancy, and traffic control. In order accomplish this, a Domain Name Server (DNS), Web Server, File Transfer Protocol (FTP) Server, Caching-only Name Server, Master Server, and Firewall were installed within virtual machines. Graduate Students were provided with a secure network test environment to verify all services, to test the access control rules, to find vulnerabilities of the Firewall, and to create solutions to protect the Firewall vulnerabilities they found. The suggested architecture provided within these guidelines are invaluable for IT students to help them recreate a similar environment of their own and to improve their technical skills.

References

Abhilash, P., Abhinay sri vasthav. V. (2015). Comparison of Windows and Linux Operating Systems in Advanced Features. *Journal of Engineering Research and Applications, 2,* pp.81-83

Alharbi, A. H. (2018). A Portable Virtual LAB for Informatics Education using Open Source software MILAB. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, *9*(2), 142-147.

Art. (2010, August). What Are Web Servers and Why Are They Needed? Retrieved from https://webhostinggeeks.com/blog/what-are-web-servers-and-why-are-they-needed/

Baker Hart (2002). Building a Security Lab with Virtual Machines. *Global Information Assurance Certification Paper. SANS Institute*

Bulbrook, H. (2006). Using virtual machines to provide a secure teaching lab environment. *White paper. Durham Technical Community College, Durham*.

Chris. (n.d.) Probably the most secure and fastest FTP server for UNIX-like systems. Retrieved from https://security.appspot.com/vsftpd.html

Gum, P. H. (1983). System/370 extended architecture: facilities for virtual machines. *IBM Journal of Research and Development*, *27*(6), 530-544.

Kanghyun, C., Jongmoon, J. (2013, July). Webhards make illegal sharing of files too easy. Retrieved from http://koreajoongangdaily.joins.com/news/article/article.aspx?aid=2974881

Margaret, R. (2019, May). Domain name system (DNS). Retrieved from https://searchnetworking.techtarget.com/definition/domain-name-system

Margaret, R. (2005, September). Iptables. Retrieved from https://whatis.techtarget.com/definition/iptables

Tsihouridis, C., Vavougios, D., Ioannidis, G. S., Alexias, A., Argyropoulos, C., & Poulios, S. (2014, December). Using sensors and data-loggers in an integrated mobile school-lab setting to teach Light and Optics. In *2014 International Conference on Interactive Collaborative Learning (ICL)* (pp. 439-445). IEEE.

Zhao, X., Borders, K., & Prakash, A. (2009). Virtual machine security systems. *Advances in Computer Science and Engineering*, *1*, 339-365.

Appendix

1. Domain Name Server (DNS) - Configurations

Each machine requires a different process in order to be built. Here are the necessary components for each server machine and client machine.

Configurations of Server A:

1) Caching-only name server

1.1) Start the Linux operating systems (Server A), by clicking Applications / Terminal.

1.2) Type yum -y install bind bind-chroot

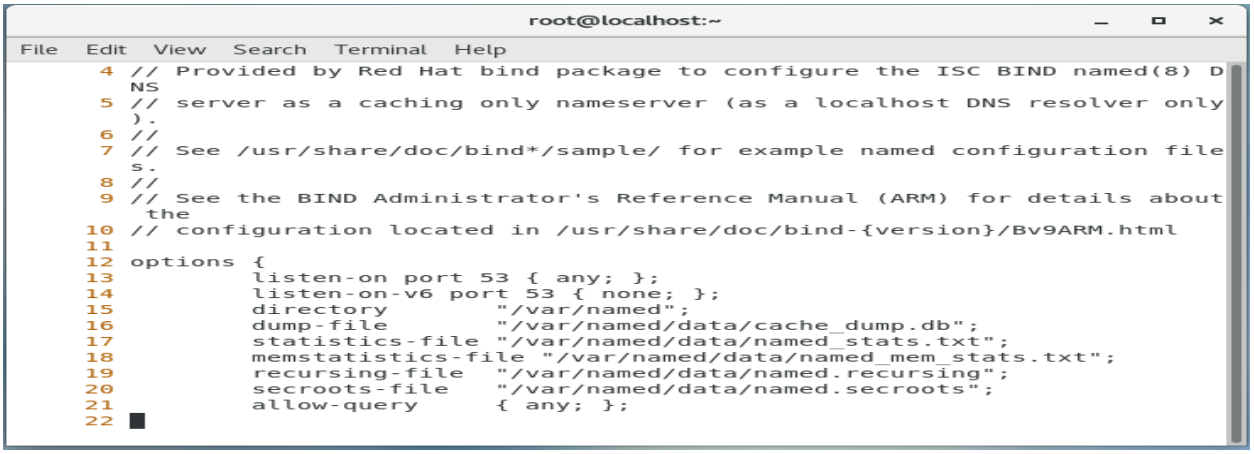
1.3) Type vi /etc/named.conf

1.3.1) Type :set number

1.3.1) Change IP address from 127.0.0.1 to any - around 13 columns

1.3.2) Change IP address from ::1 to none - around 14 columns

1.3.3) Change IP address from localhost to any - around 21 columns

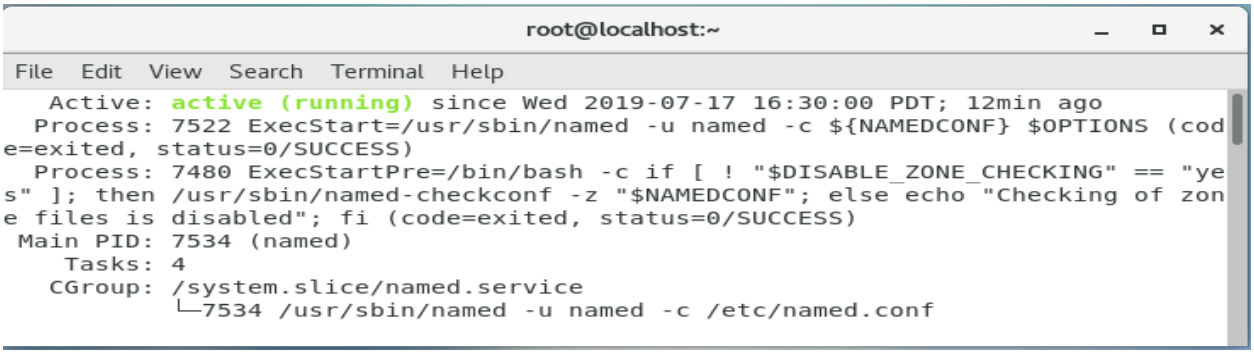


1.4) Change named for active

1.4.1) Type systemctl status named

1.4.2) Type systemctl restart named

1.4.3) Type systemctl enable named



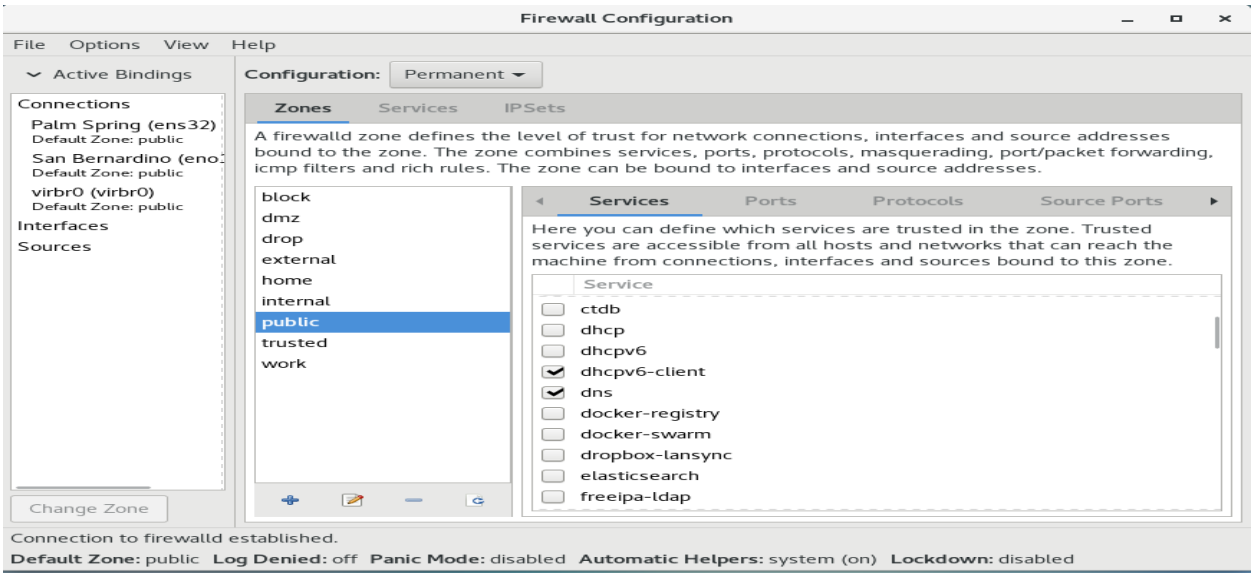
1.5) Change firewall

1.5.1) Type firewall-config

1.5.2) Change configuration from runtime to permanent

1.5.3) Click dns in public zone

1.5.4) Click reload firewalld in options



2) Master name server

2.1) Start the Linux operating systems (Server A), by clicking Applications / Terminal.

2.2) Type vi /etc/named.conf

2.2.1) Type :set number

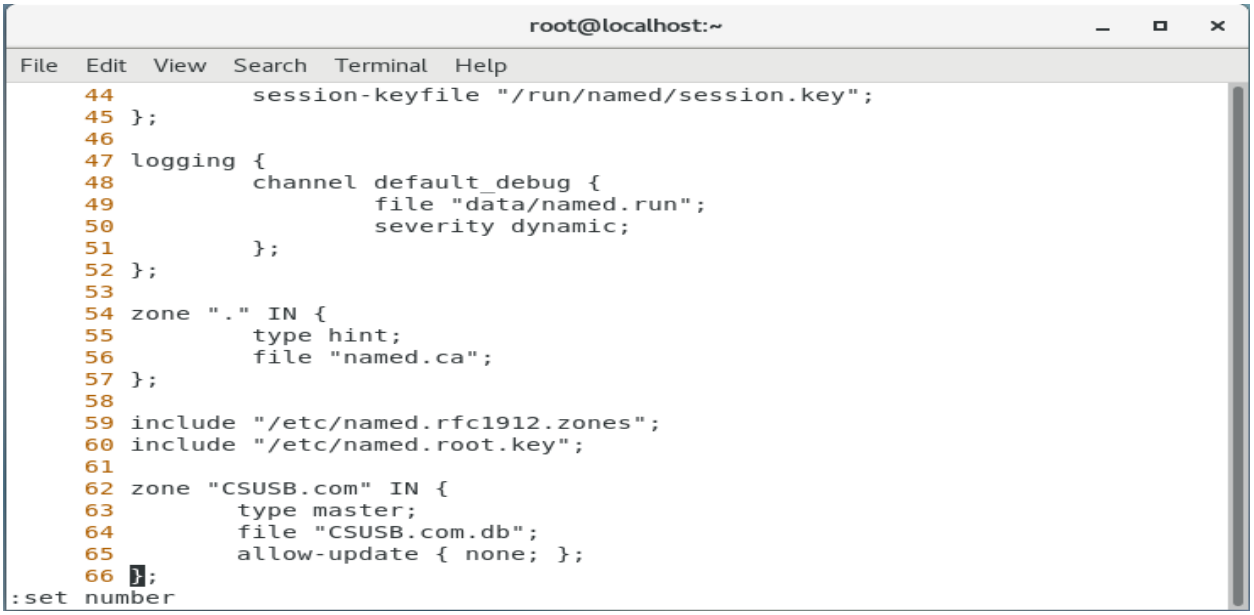
2.2.2) zone “CSUSB.com” IN { - around 62 columns

type master;

file “CSUSB.com.db”;

allow-update { none; };

}; ”



2.2) Type cd /var/named

2.3) Type touch CSUSB.com.db

2.4) Type vi CSUSB.com.db

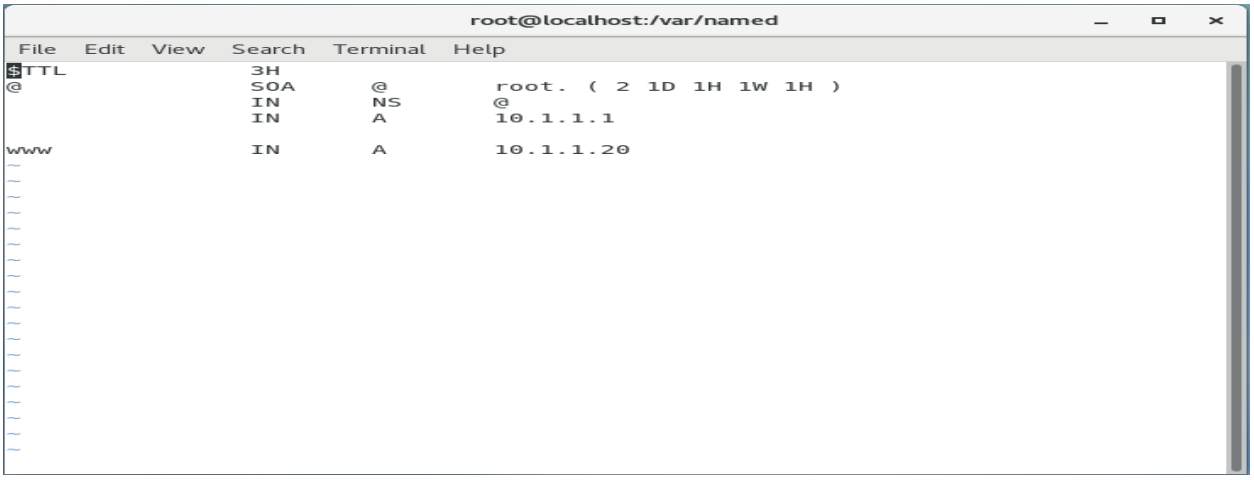
2.4.1) $TTL 3H

@ SOA @ root. (2 1D 1H 1W 1H)

IN NS @

IN A 10.1.1.1

www IN A 10.1.1.20

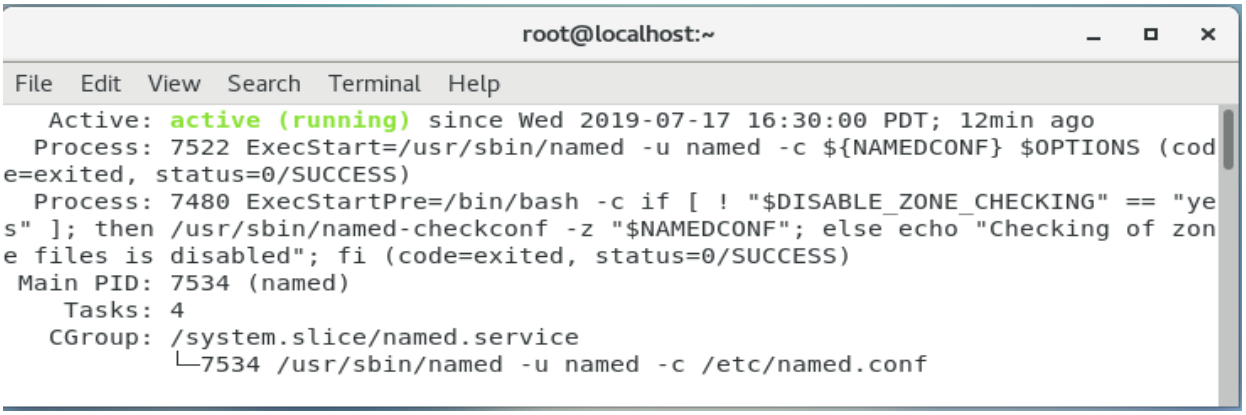


2.5) Change named for active

2.5.1) Type systemctl status named

2.5.2) Type systemctl restart named

2.5.3) Type systemctl enable named



Configurations of Server B:

3) Web Server

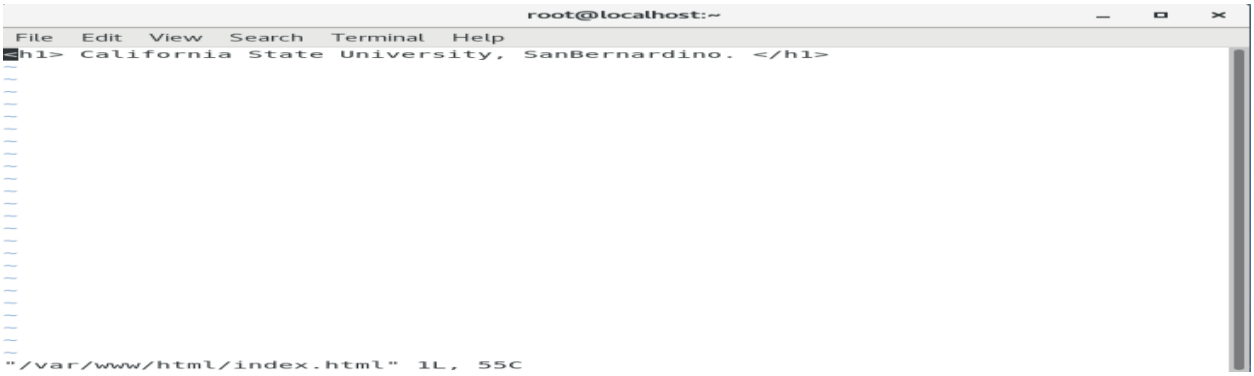
3.1) Start the Linux operating systems (Server B), by clicking Applications / Terminal.

3.2) Type cd /var/www/html

3.3) Type touch index.html

3.4) Type vi index.html

3.4.1) California State University, San Bernardino



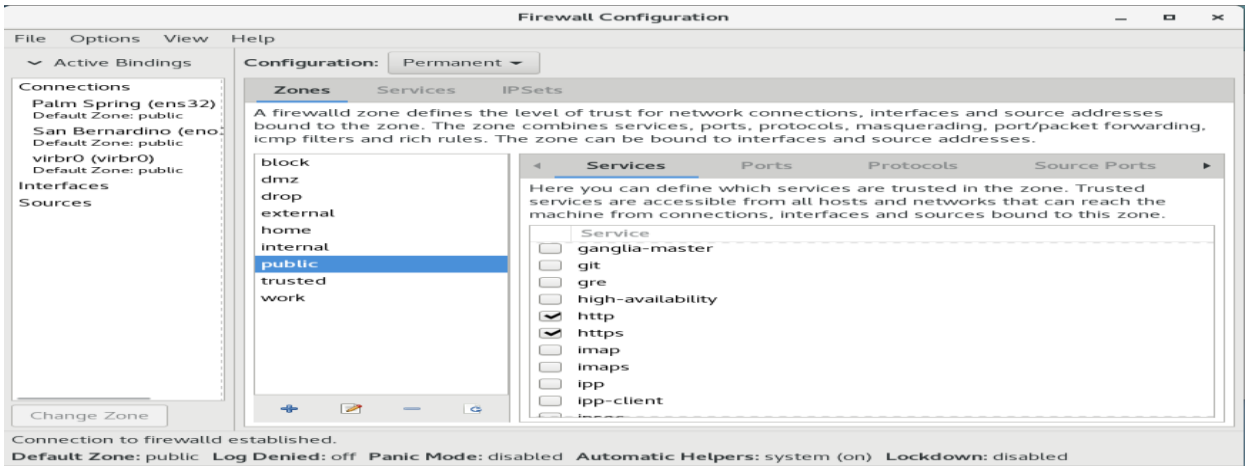
3.5) Change firewall

3.5.1) Type firewall-config

3.5.2) Change configuration from runtime to permanent

3.5.3) Click http in public zone

3.5.4) Click reload firewalld in options

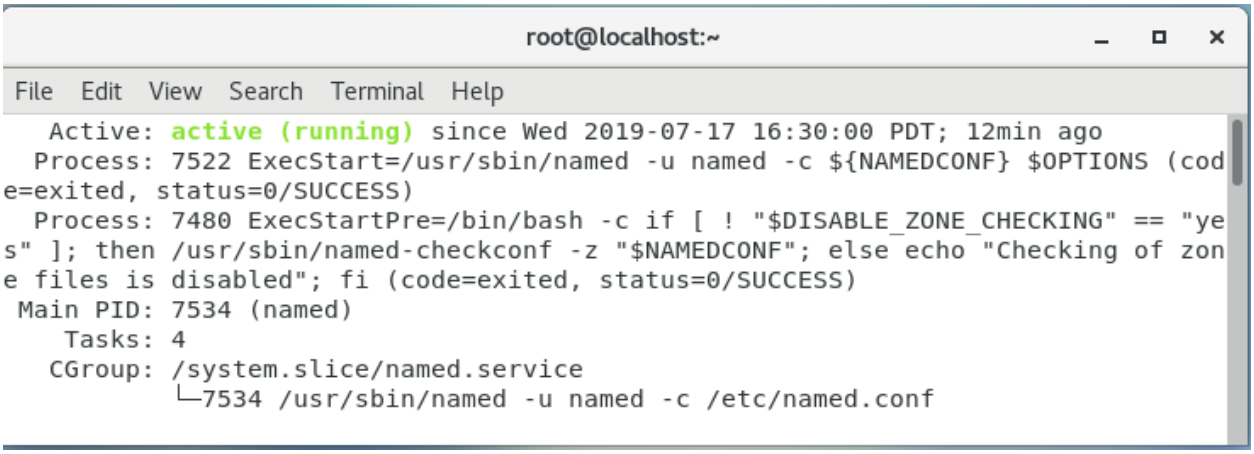


3.6) Change named for active

3.6.1) Type systemctl status named

3.6.2) Type systemctl restart named

3.6.3) Type systemctl enable named



2. Web Server – Configurations

Each machine requires a different process in order to be built. Here are the necessary components for each server machine and client machine.

Configurations of Server B:

1) MariaDB

1.1) Start the Linux operating systems (Server B), by clicking Applications / Firefox web browser.

1.2) Type http://www.mariadb.org/

1.2.1) Click download mariadb now! then download 10.4.6

1.2.2) Click Red Hat, Fedora, and CentOS Packages

1.3) Click Applications / Terminal

1.4) Type yum -y localinstall maria*

1.5) Type yum update

1.6) Type yum -y install mariadb-server mariadb php php-mysqlnd

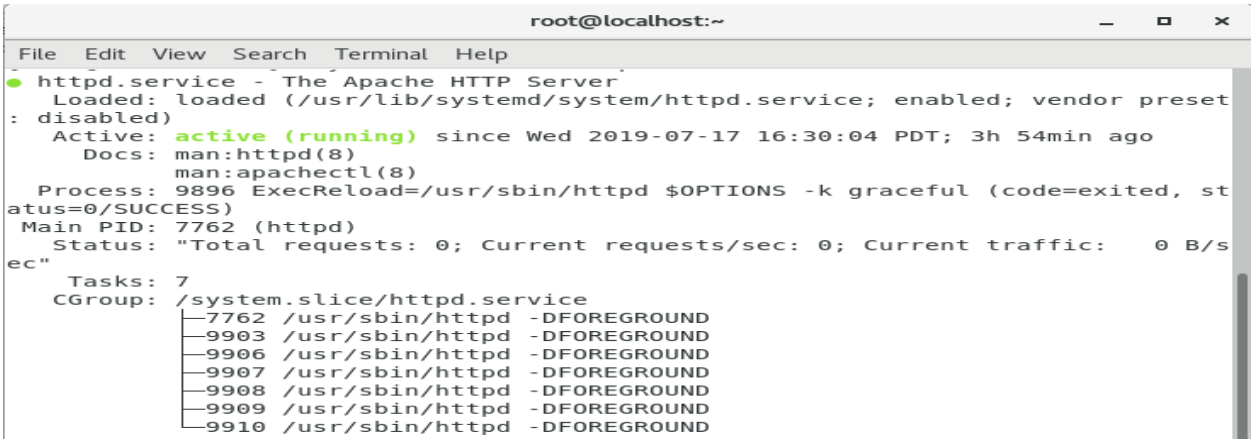
1.7 Type "rpm -qa httpd php mariadb-server

1.8) Change httpd for active

1.8.1) Type systemctl status httpd

1.8.2) Type systemctl restart httpd

1.8.3) Type systemctl enable httpd

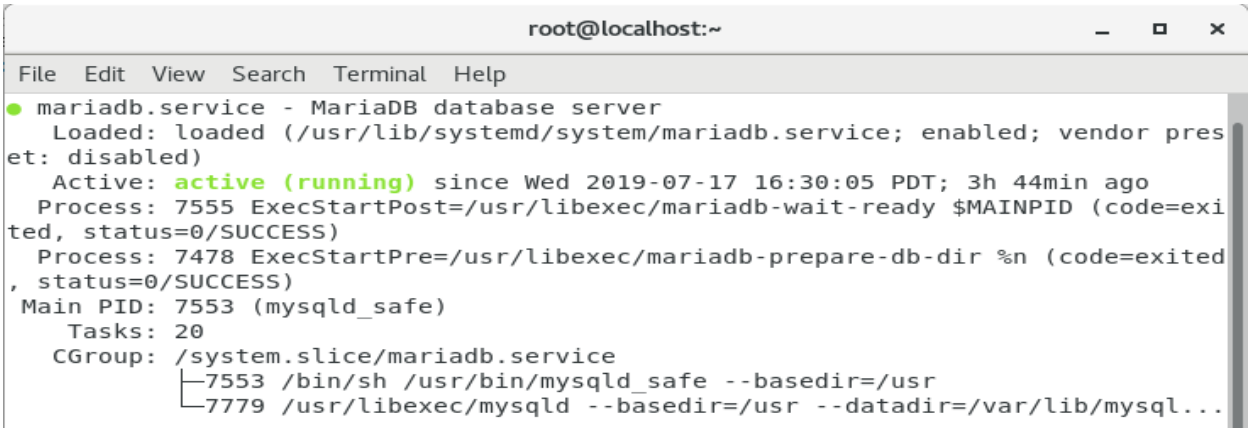


1.9) Change mariadb for active

1.9.1) Type systemctl status mariadb

1.9.2) Type systemctl enable mariadb

1.9.3) Type systemctl restart mariadb



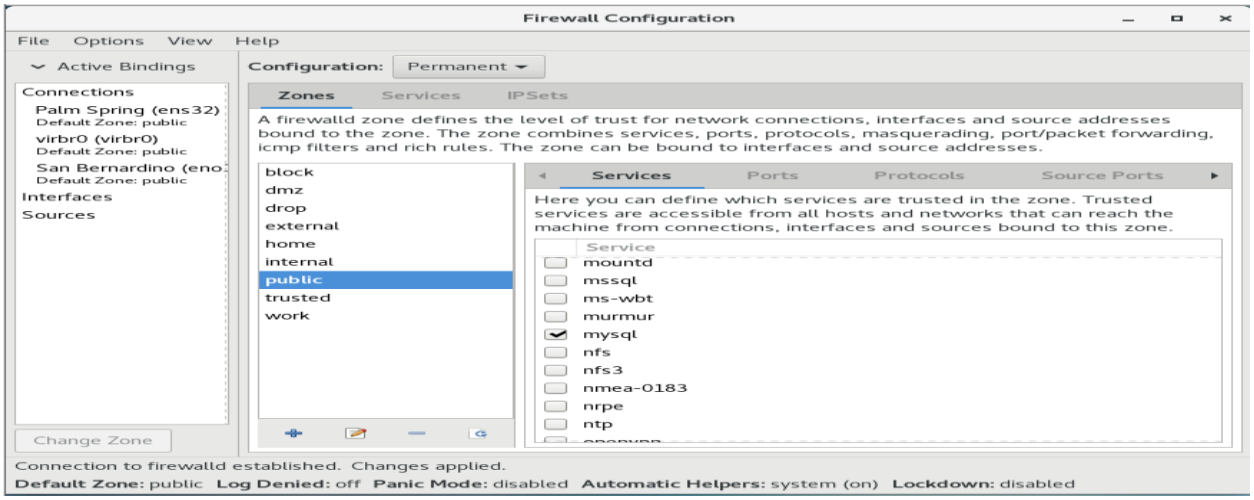
1.10) Change firewall

1.10.1) Type firewall-config

1.10.2) Change configuration from runtime to permanent

1.10.3) Click mysql in public zone

1.10.4) Click reload firewalld in options



1.8) Type cd /var/www/html/

1.9) Type touch phpinfo.php

1.10) Type vi phpinfo.php

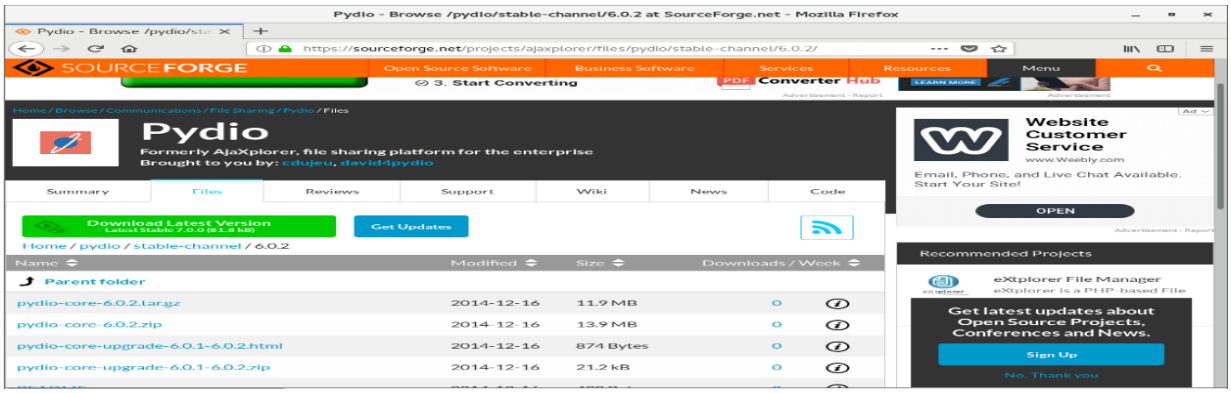
1.22) Type <?php phpinfo(); ?>

2) Web hard

2.1) Start the Linux operating systems (Server B), by clicking Applications / Firefox web browser.

2.2) Type https://sourceforge.net/projects/ajaxplorer/files/pydio/stable-channel/6.0.2/

2.2.1) Click to download pydio-core-6.0.2.tar.gz



2.3) Click Applications / Terminal

2.4) Type cd /var/www/html/

2.5) Type mv /root/Downloads/pydio-core-6.0.2.tar.gz .

2.6) Type tar xfz pydio-core-6.0.2.tar.gz

2.7) Type mv /pydio-core-6.0.2 webhard

2.8) Type chmod 707 webhard

2.9) Type chown -R apache.apache webhard

2.10) Type yum -y --skip-broken install php-*

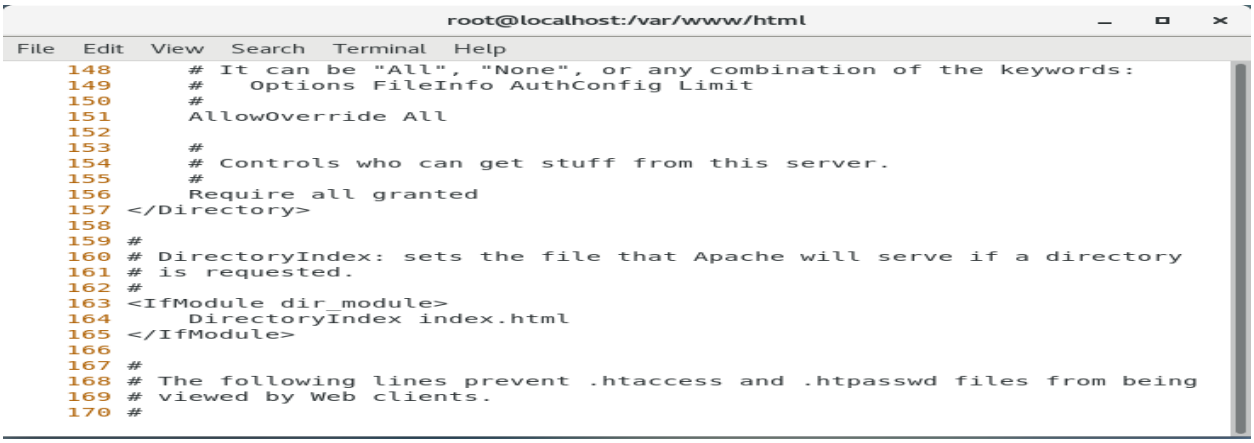
2.11) Type yum -y install epel-release

2.12) Type yum -y install php-mcrypt

2.13) Type vi /etc/httpd/conf/httpd.conf

2.13.1) Type :set number

2.13.2) Change from AllowOverride None to AllowOverride All - around 151 columns



2.14) Type vi /etc/php.ini

2.14.1) Type :set number

2.14.2) Change from max_execution_time = 30 to max_execution_time = 300 - around 384 columns

2.14.3) Change from post_max_size = 8M to post_max_size = 100M - around 672 columns

2.14.4) Change from upload_max_filesize = 2M to upload_max_filesize = 100M - around 800 columns

```
root@localhost:/var/www/html
File  Edit  View  Search  Terminal  Help
787 ;;;;;;;;;;;;;;;;;
788
789 ; Whether to allow HTTP file uploads.
790 ; http://php.net/file-uploads
791 file_uploads = On
792
793 ; Temporary directory for HTTP uploaded files (will use system default i
    f not
794 ; specified).
795 ; http://php.net/upload-tmp-dir
796 ;upload_tmp_dir =
797
798 ; Maximum allowed size for uploaded files.
799 ; http://php.net/upload-max-filesize
800 upload_max_filesize = 100M
801
802 ; Maximum number of files that can be uploaded via a single request
803 max_file_uploads = 20
804
805 ;;;;;;;;;;;;;;;;;;;
806 ; Fopen wrappers ;
807 ;;;;;;;;;;;;;;;;;;;
808 ■
```

2.15) Type cd /var/www/html/webhard/data/cache/

2.16) rm -f plugin*

2.17) Change httpd for active

2.17.1) systemctl status httpd

2.17.2) systemctl restart httpd

2.17.3) systemctl enable httpd

```
root@localhost:~
File  Edit  View  Search  Terminal  Help
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset
: disabled)
   Active: active (running) since Wed 2019-07-17 16:30:04 PDT; 3h 54min ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Process: 9896 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, st
atus=0/SUCCESS)
 Main PID: 7762 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/s
ec"
    Tasks: 7
   CGroup: /system.slice/httpd.service
           ├─7762 /usr/sbin/httpd -DFOREGROUND
           ├─9903 /usr/sbin/httpd -DFOREGROUND
           ├─9906 /usr/sbin/httpd -DFOREGROUND
           ├─9907 /usr/sbin/httpd -DFOREGROUND
           ├─9908 /usr/sbin/httpd -DFOREGROUND
           ├─9909 /usr/sbin/httpd -DFOREGROUND
           └─9910 /usr/sbin/httpd -DFOREGROUND
```

Configurations of Linux Client:

3) Install Web hard

3.1) Start the Linux operating systems (Linux Client), by clicking Applications / Firefox web browser.

3.2) Type 10.1.1.20/webhard

3.2.1) Click pydio, English, Start wizard

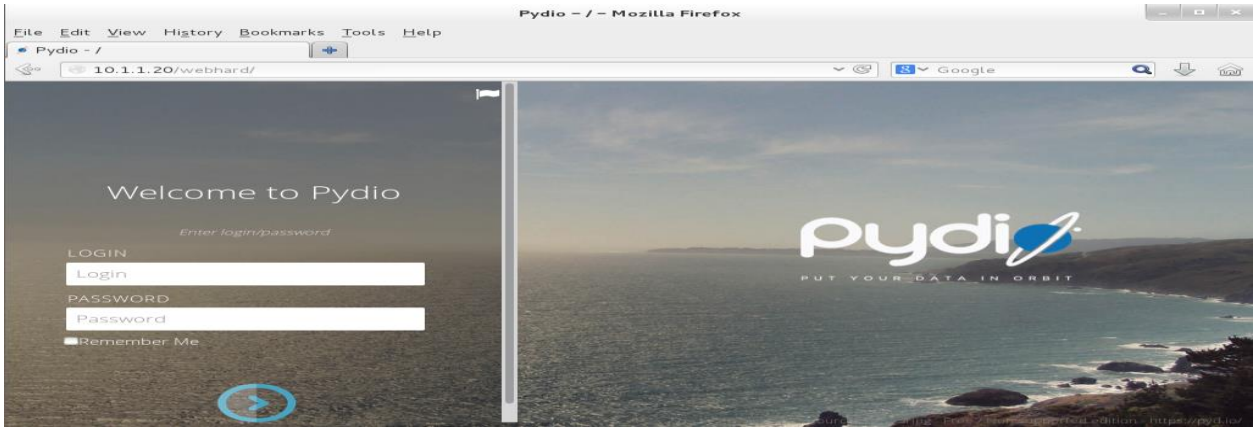
3.2.2) CSUSB in id, 1234 in password

3.2.3) Admin access - admin in admin login, admin in admin display name, 12345678 in admin password

3.2.4) Global options - English in default language

3.2.5) Configurations storage - Click Database system in stroage type, xeDB in database, xeUser in user, 1234 in password click try connecting to the database

3.2.6) Add some users - Type "CSUSB in login, admin@csusb.com in user email, User in display name, 1234 in password" then click install pydio now



3. File Transfer Protocol (FTP) Server – Configurations

Each machine requires a different process in order to be built. Here are the necessary components for each server machine and client machine.

Configurations of Server B:

1) FTP Server

1.1) Start the Linux operating systems (Server B), by clicking Applications / Terminal.

1.2) Type yum -y install vsftpd

1.3) Type cd /var/ftp/

1.4) Type cd /pub/

1.5) Type cp /boot/vmlinuz-3* file1

1.6) Type vi /etc/vsftpd/vsftpd.conf/

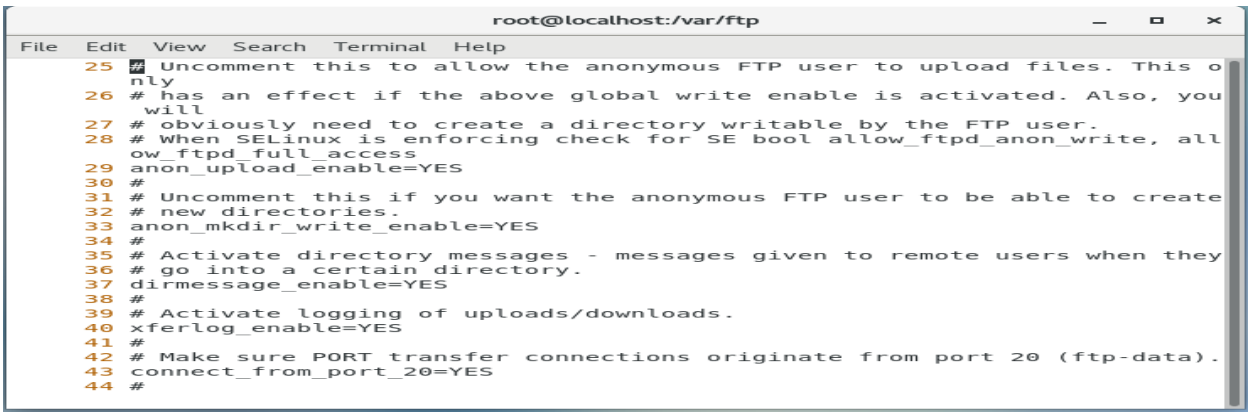
1.6.1) Type :set number

1.6.1) Change from #write_enable=YES to write_enable=YES - around 19 columns

1.6.2) Change from #anon_upload_enable=YES to anon_upload_enable= YES

- around 29 columns

1.6.3) Change #anon_mkdir_write_enable=Yes to anon_mkdir_write_enable=Yes - around 33 columns



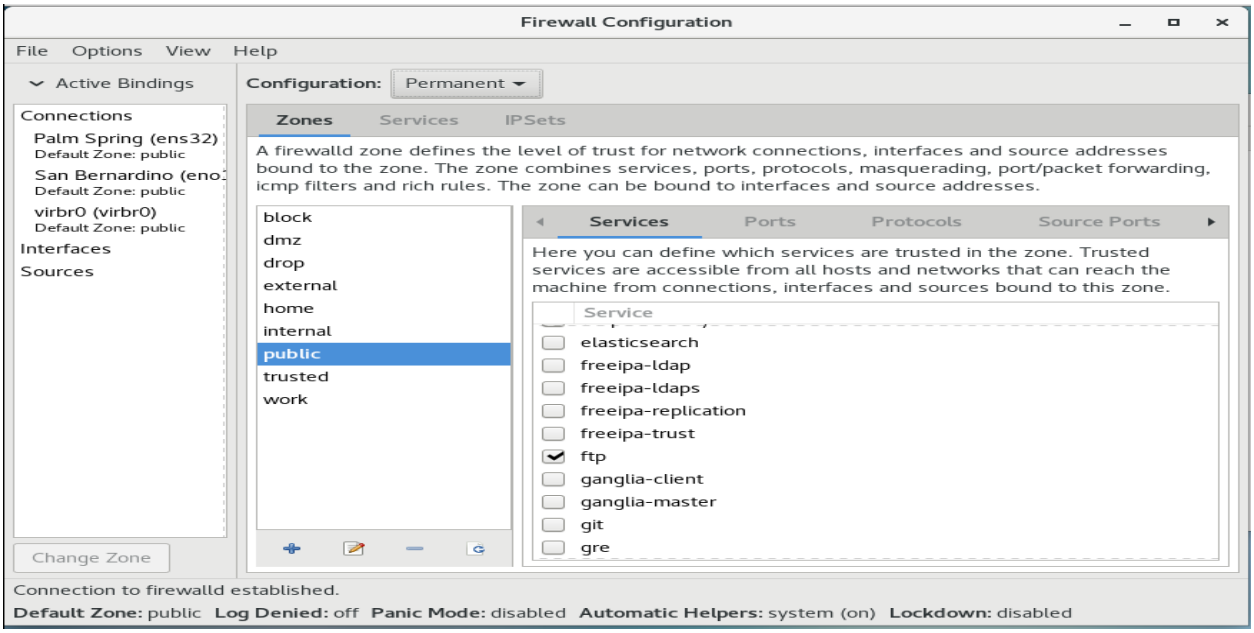
1.7) Type chown ftp.ftp /var/ftp/pub/

1.8) Change firewall

1.8.1) Type firewall-config

1.8.2) Change configuration from runtime to permanent

1.8.3) Click ftp in public zone

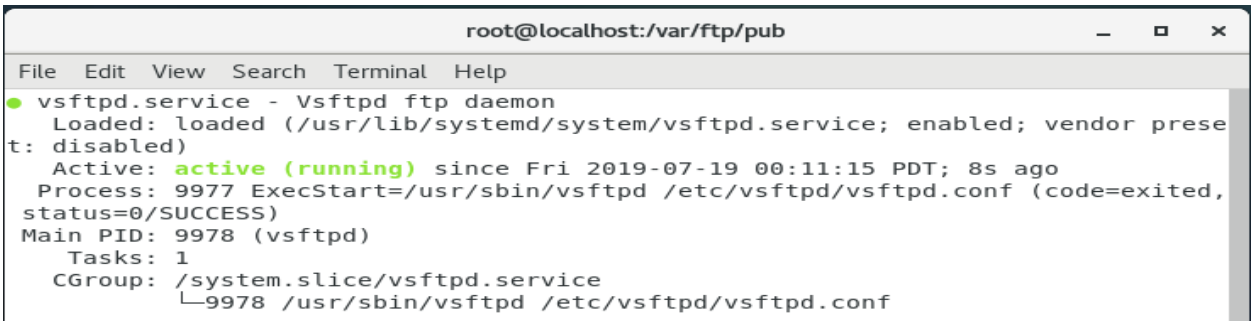


1.9) Change vsftpd for active

1.9.1) Type systemctl status vsftpd

1.9.2) Type systemctl restart vsftpd

1.9.3) Type systemctl enable vsftpd



Configurations of Linux Client.

2) FTP Linux Client

2.1) Start the Linux operating systems (Linux Client), by clicking Applications / Terminal.

2.2) Type yum -y install epel-release

2.3) Type yum -y install filezilla

2.4) Type filezilla

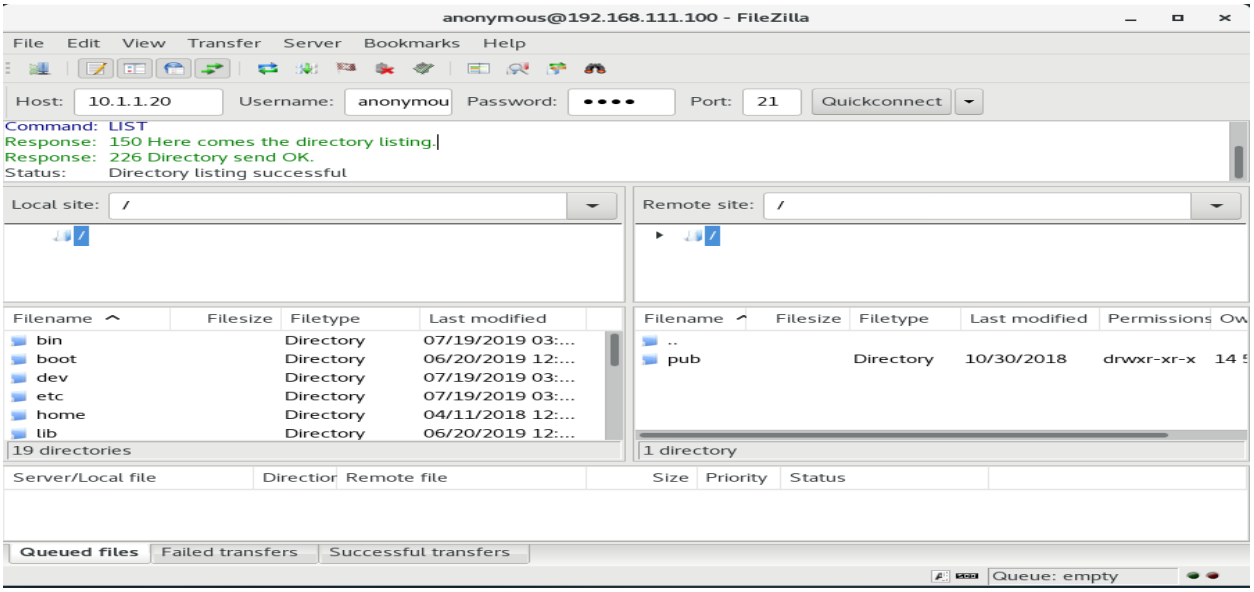
2.4.1) Type 10.1.1.20 in host

2.4.2) Type anonymous in username

2.4.3) Type 1234 in password

2.4.4) Type 21 in port

2.4.5) Click quickconnect



4. Firewall - Configurations

Each machine requires a different process in order to be built. Here are the necessary components for each server machine and client machine.

Configurations of Server A:

1) Network setting

1.1) Start the Linux operating systems (Server A), by clicking Applications / Terminal.

1.2) Type cd /etc/sysconfig/network-scripts/

1.3) Type vi ifcfg-eno16777736

1.3.1) Type :set number

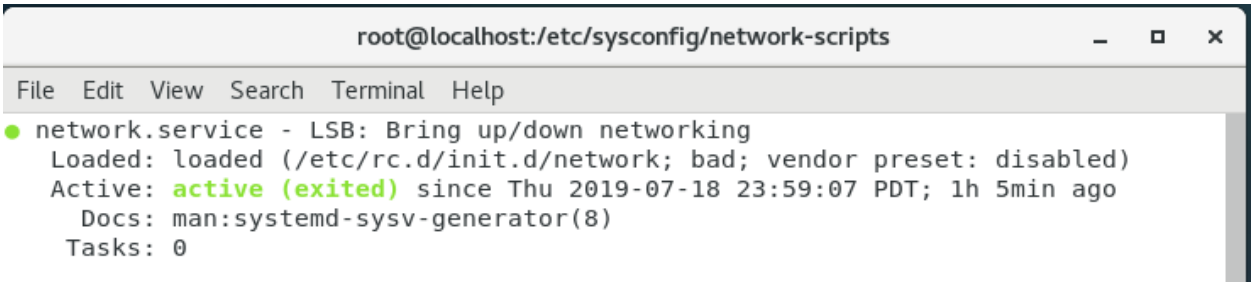
1.3.2) Change from BOOTPROTO="dhcp" to BOOTPROTO=none – around 3 columns

1.4) Change httpd for active

1.4.1) Type systemctl status httpd

1.4.2) Type systemctl restart network

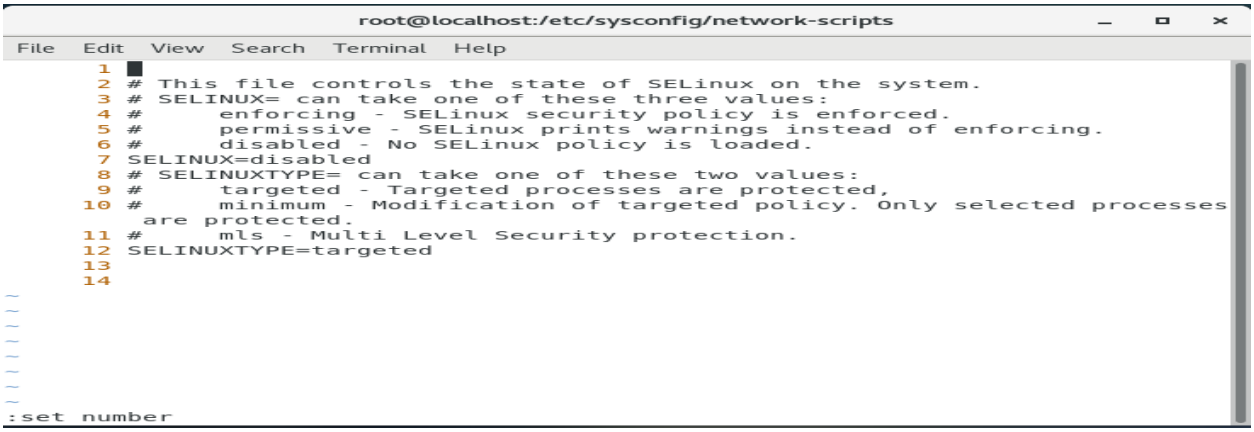
1.4.3) Type systemctl enable network



1.5) Type vi /etc/sysconfig/selinux

1.5.1) Type :set number

1.5.2) Change from SELINUX=enforcing to SELINUX=disabled – around 7 columns

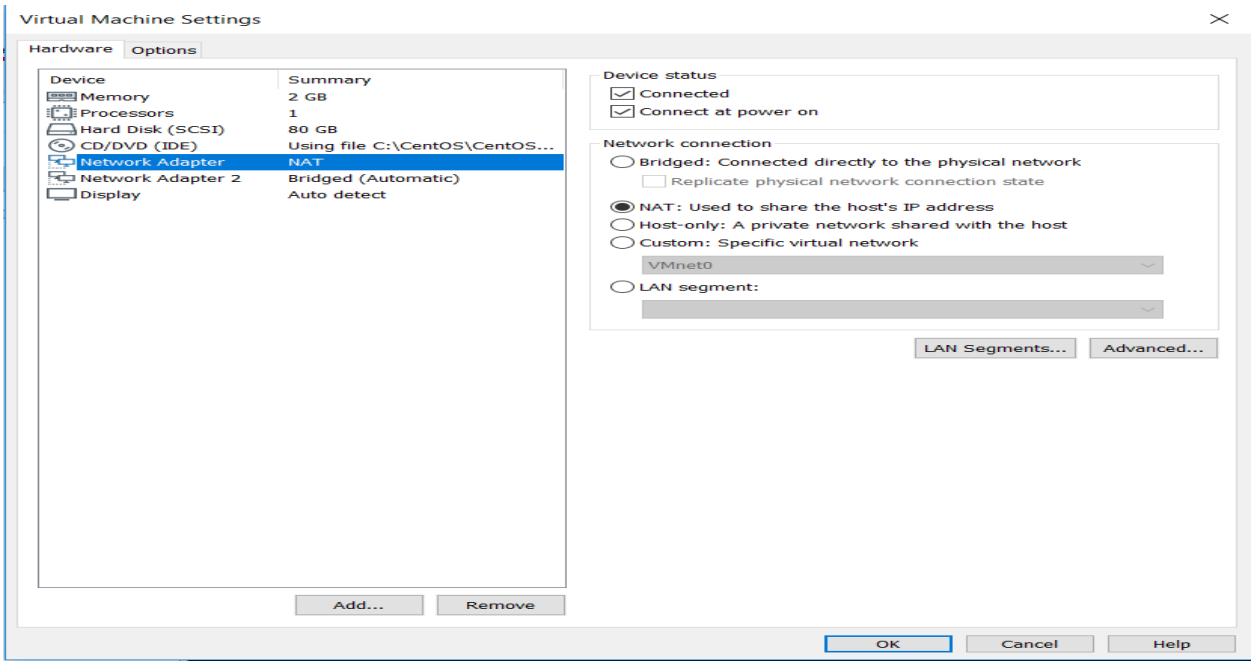


2) First IP address

2.1) Virtual machine settings

2.1.1) Click network adapter

2.1.2) Click nat: used to share the host's IP address



2.2) Click Applications / Terminal.

2.3) Type nmtui edit Wired connection 1

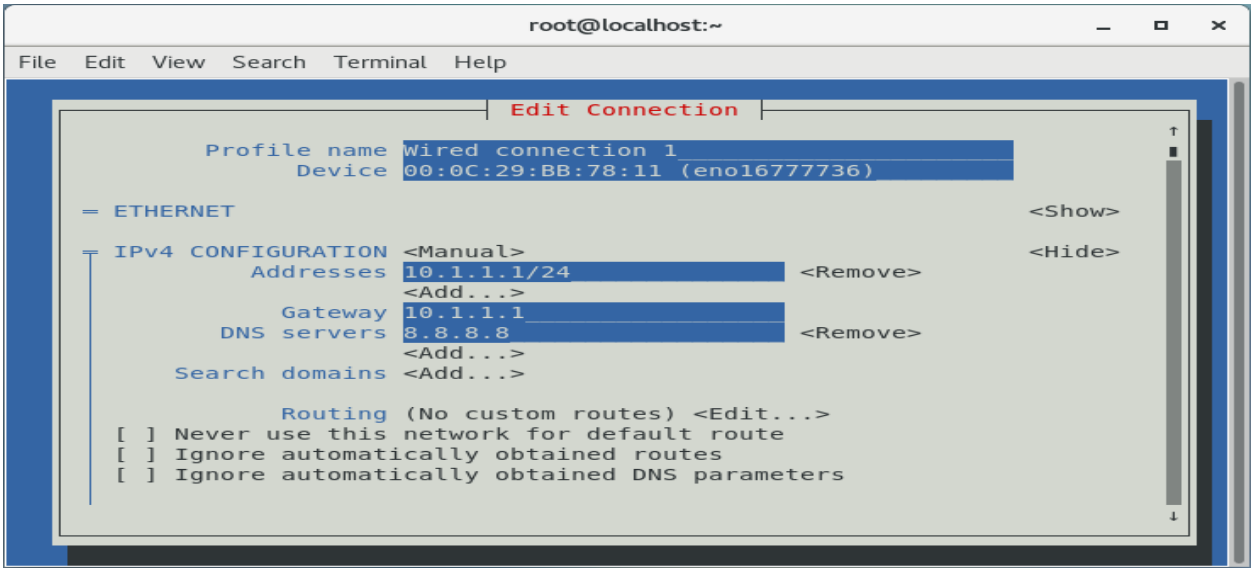
2.3.1) 10.1.1.1/24 in addresses

2.3.2) 10.1.1.1 in gateway

2.3.3) 8.8.8.8 in DNS servers

2.3.4) Click ok

2.3.5) reboot



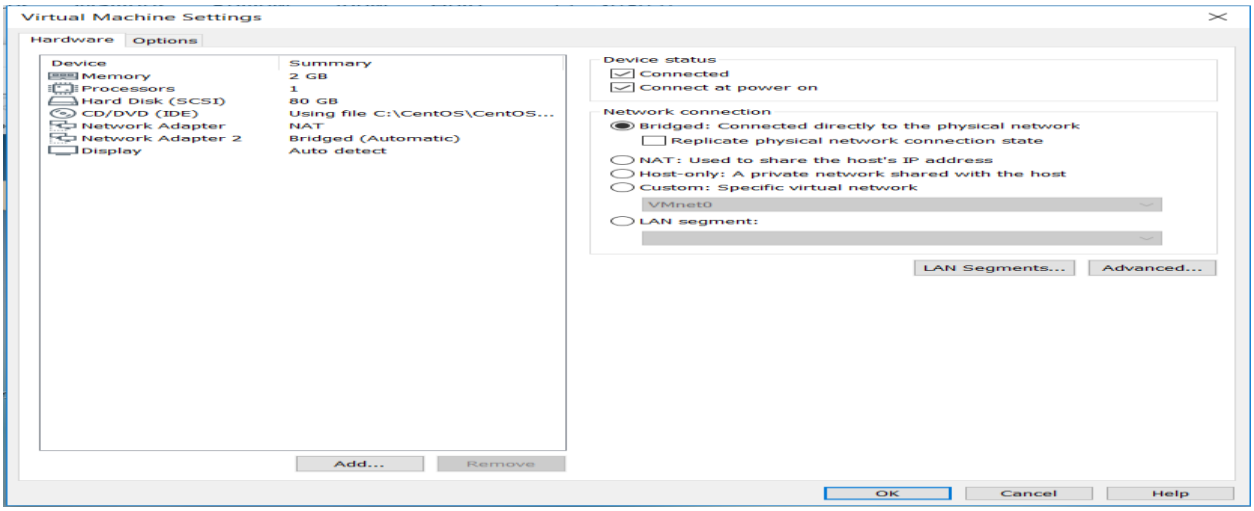
3) Second IP address

3.2) Virtual machine settings

3.2.1) Click add

3.2.2) Click network adapter

3.2.3) Click bridged: connected directly to the physical network



3.3) Click Applications / Terminal.

3.4) Type nmtui edit Wired connection 2

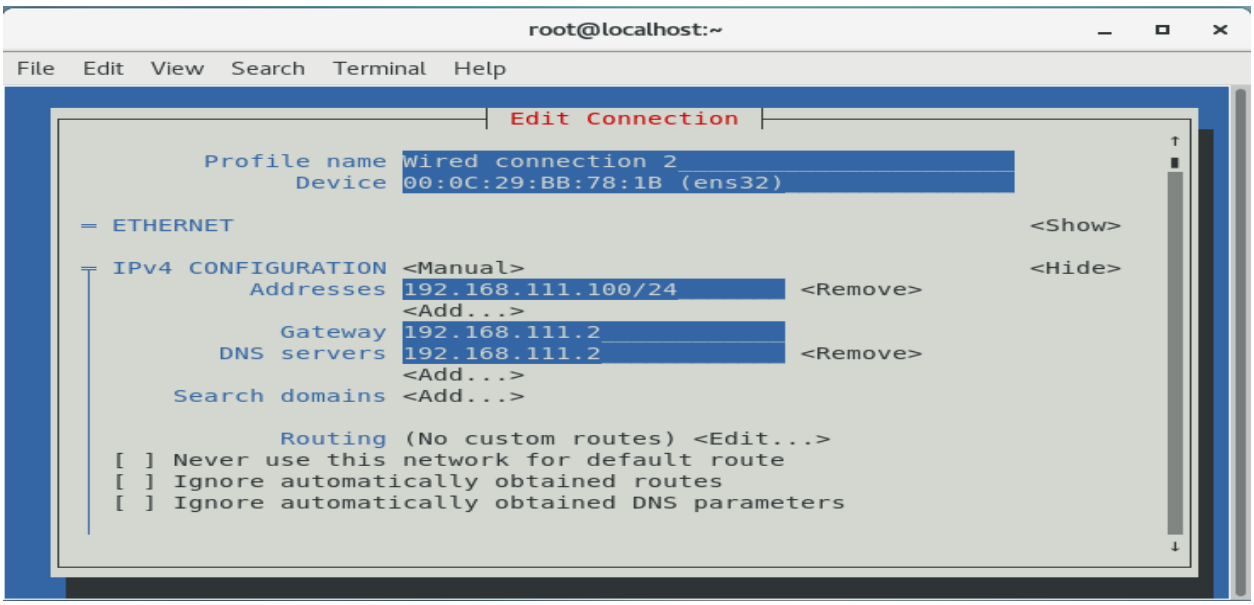
2.3.1) 192.168.111.100/24 in addresses

2.3.2) 192.168.111.2 in gateway

2.3.3) 192.168.111.2 in DNS servers

3.11) Click ok

3.12) reboot



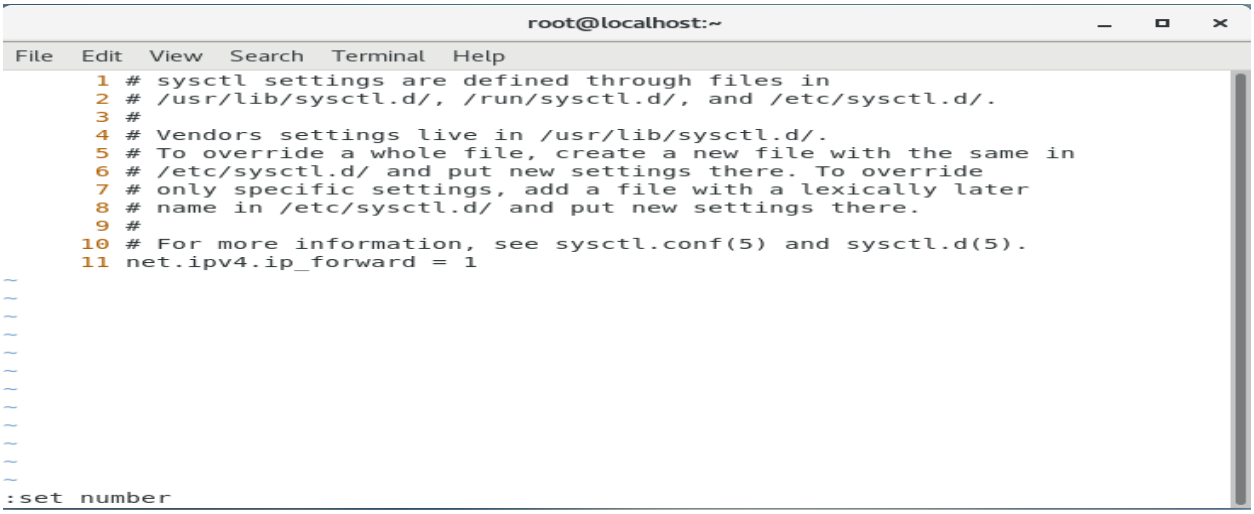
4) iptables

4.1) Click Applications / Terminal.

4.2) Type vi /etc/sysctl.conf

4.2.1) Type :set number

4.2.2) Type net.ipv4.ip_forward = 1 - around 11 columns



4.3) Type echo 1 > /proc/sys/net/ipv4/ip_forward

4.4) Change iptables policy

4.4.1) Type iptables --policy FORWARD DROP

4.4.2) Type iptables --policy INPUT DROP

4.4.3) Type iptables --policy OUTPUT DROP

4.5) Iptable policy specifications

4.5.1) iptables --append INPUT --in-interface ens32 --source 10.1.1.0/24 --match state --state NEW,ESTABLISHED --jump ACCEPT

4.5.2) iptables --append OUTPUT --out-interface ens32 --destination 10.1.1.0/24 --match state --state NEW,ESTABLISHED --jump ACCEPT

4.5.3) iptables --append FORWARD --in-interface ens32 --source 10.1.1.0/24 --destination 0.0.0.0/0 --match state --state NEW,ESTABLISHED --jump ACCEPT

4.5.4) iptables --append FORWARD --in-interface eno16777736 destination 10.1.1.0/24 --match state --state NEW,ESTABLISHED --jump ACCEPT

4.5.5) iptables --table nat --append POSTROUTING --out-interface eno16777736 --jump MASQUERADE

4.5.6) service iptable save

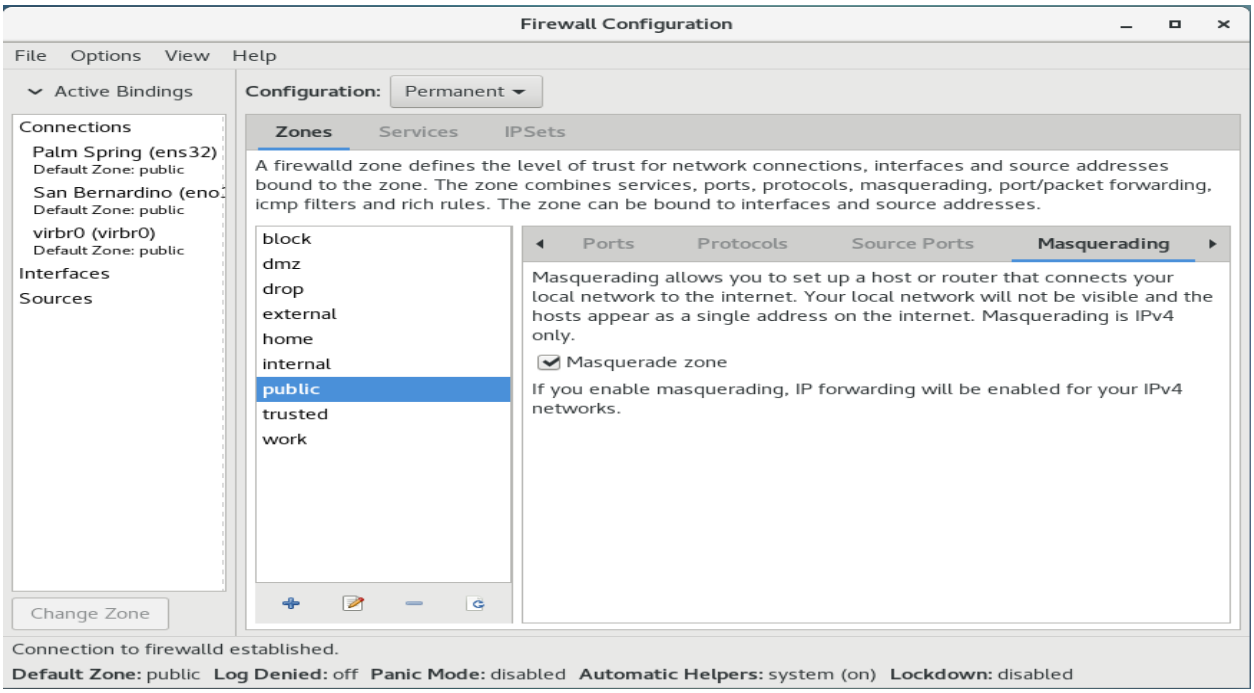
4.6) Change firewall

4.6.1) Type firewall-config

4.6.2) Change configuration from runtime to permanent

4.6.3) Click masquerading then masquerade zone

4.6.4) Click reload firewalld in options



5) iptables for http

5.1) Click Applications / Terminal.

5.2) Type iptables --table nat --append PERROUTING --proto tcp --in-interface eno16777736 --dport 80 --jump DNAT --to-destination 10.1.1.20

5.4) Type service iptable save

6) iptables for ftp

6.1) Click Applications / Terminal.

6.2) iptables --table nat --append PERROUTING --proto tcp --in-interface eno16777736

--dport 21 --jump DNAT --to-destination 10.1.1.20

6.3) service iptable save

Configurations of Server B:

7) Network setting

7.1) Start the Linux operating systems (Server B), by clicking Applications / Terminal.

7.3) Type cd /etc/sysconfig/network-scripts/

7.4) Type vi ifcfg-eno16777728

7.4.1) Type :set number

7.4.1) Change from BOOTPROTO="dhcp" to BOOTPROTO=none

7.5) Change network for active

7.5.1) systemctl status network

7.5.2) systemctl restart network

7.5.3) systemctl enable network

```
root@localhost:~
File  Edit  View  Search  Terminal  Help
● network.service - LSB: Bring up/down networking
   Loaded: loaded (/etc/rc.d/init.d/network; bad; vendor preset: disabled)
   Active: active (exited) since Fri 2019-07-19 16:18:36 PDT; 1h 15min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 7129 ExecStart=/etc/rc.d/init.d/network start (code=exited, status=0/
SUCCESS)
    Tasks: 0
```

7.6) Type vi /etc/sysconfig/selinux

7.6.1) Type :set number

7.6.2) Change from SELINUX=enforcing to SELINUX=disabled – around 7 columns

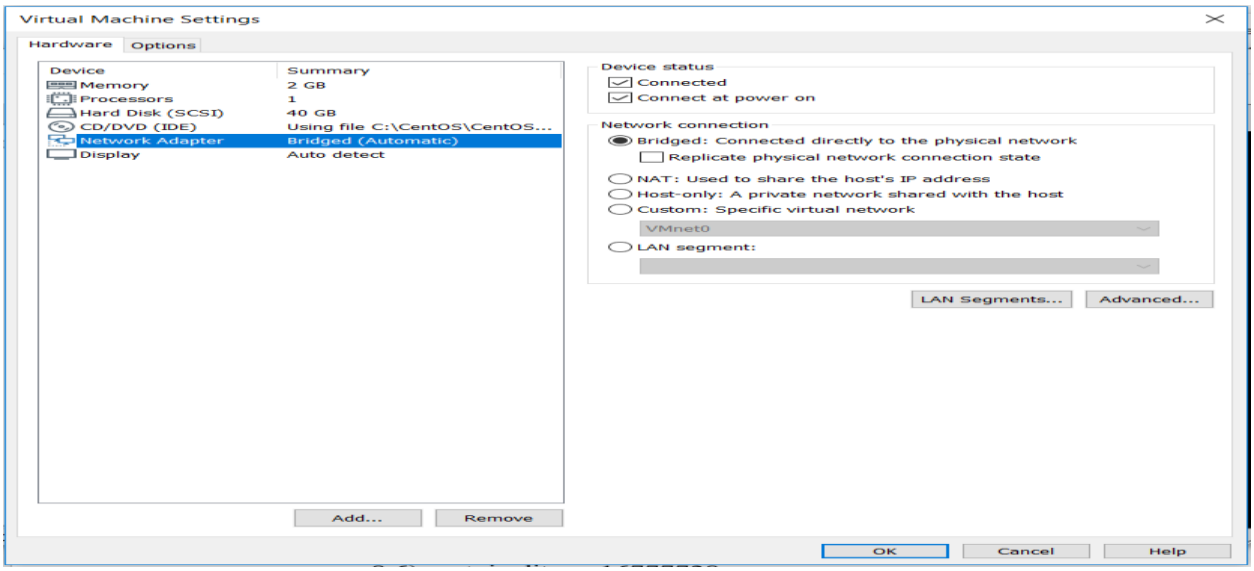
```
root@localhost:/etc/sysconfig/network-scripts
File  Edit  View  Search  Terminal  Help
  1
  2 # This file controls the state of SELinux on the system.
  3 # SELINUX= can take one of these three values:
  4 #     enforcing - SELinux security policy is enforced.
  5 #     permissive - SELinux prints warnings instead of enforcing.
  6 #     disabled - No SELinux policy is loaded.
  7 SELINUX=disabled
  8 # SELINUXTYPE= can take one of these two values:
  9 #     targeted - Targeted processes are protected,
 10 #     minimum - Modification of targeted policy. Only selected processes
    are protected.
 11 #     mls - Multi Level Security protection.
 12 SELINUXTYPE=targeted
 13
 14
~
~
~
~
~
~
~
:set number
```

8) IP address

8.1) Virtual machine settings

8.2) Click network adapter

8.3) Click bridged: connected directly to the physical network



8.4) Click Applications / Terminal.

8.5) Type nmtui edit eno16777728

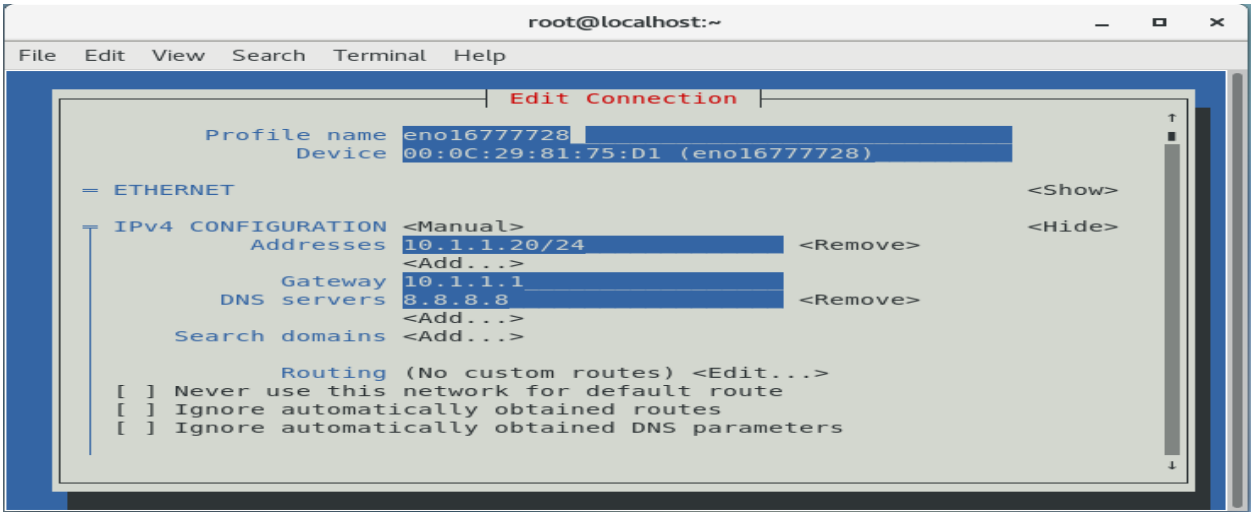
8.5.1) 10.1.1.20/24 in addresses

8.5.2) 10.1.1.1 in gateway

8.5.3) 8.8.8.8 in DNS servers

8.5.4) Click ok

8.5.5) reboot



Configurations of Linux Client:

9) Network setting

9.1) Start the Linux operating systems (Linux Client), by clicking Applications / Terminal.

9.2) Type cd /etc/sysconfig/network-scripts/

9.3) Type vi ifcfg-eno16777736

9.3.1) Type :set number

9.3.1) Change from BOOTPROTO="dhcp" to BOOTPROTO=none – around 2 columns

```
root@localhost:/etc/sysconfig/network-scripts
File  Edit  View  Search  Terminal  Help
  1 TYPE=Ethernet
  2 BOOTPROTO=none
  3 DEFROUTE=yes
  4 IPV4_FAILURE_FATAL=no
  5 IPV6INIT=yes
  6 IPV6_AUTOCONF=yes
  7 IPV6_DEFROUTE=yes
  8 IPV6_FAILURE_FATAL=no
  9 NAME=Information
 10 UUID=bf8c4053-77d1-4683-bd71-79ed91766459
 11 ONBOOT=yes
 12 GATEWAY0=10.1.1.1
 13 DNS1=8.8.8.8
 14 HWADDR=00:0C:29:B8:34:12
 15 PROXY_METHOD=none
 16 BROWSER_ONLY=no
 17 IPADDR=10.1.1.10
 18 PREFIX=24
 19 GATEWAY=10.1.1.1
~
~
~
~
:set number
```

9.4) Change network for active

9.4.1) systemctl status network

9.4.2) systemctl restart network

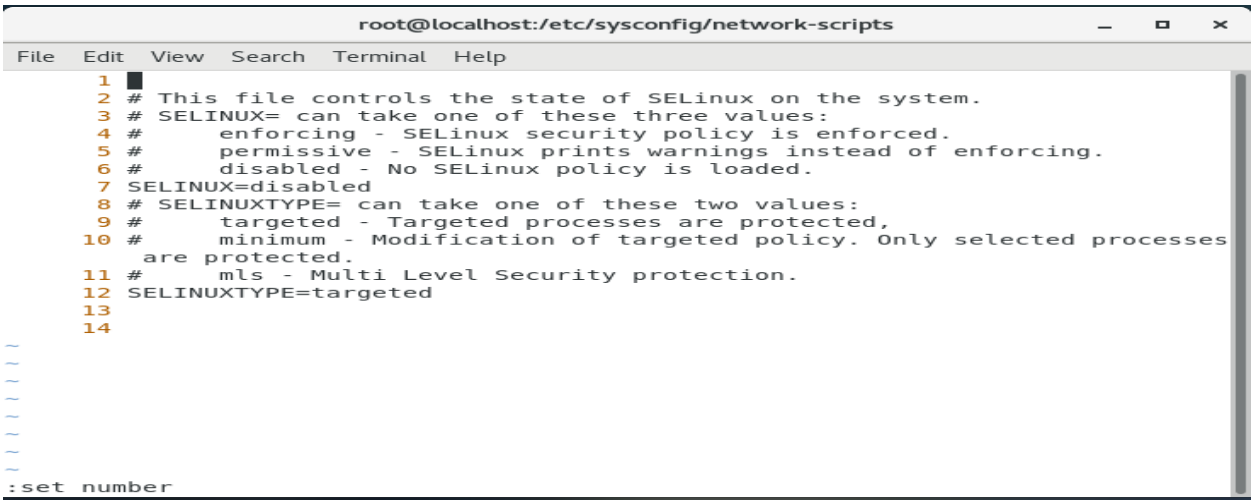
9.4.3) systemctl enable network

```
root@localhost:~
File  Edit  View  Search  Terminal  Help
[root@localhost ~]# systemctl status network
● network.service - LSB: Bring up/down networking
   Loaded: loaded (/etc/rc.d/init.d/network; bad; vendor preset: disabled)
   Active: active (exited) since Sat 2019-07-20 03:04:14 EDT; 2min 22s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 10888 ExecStop=/etc/rc.d/init.d/network stop (code=exited, status=0/S
UCCESS)
  Process: 11087 ExecStart=/etc/rc.d/init.d/network start (code=exited, status=0
/SUCCESS)
```

9.5) Type vi /etc/sysconfig/selinux

9.5.1) Type :set number

9.5.2) Change from SELINUX=enforcing to SELINUX=disabled

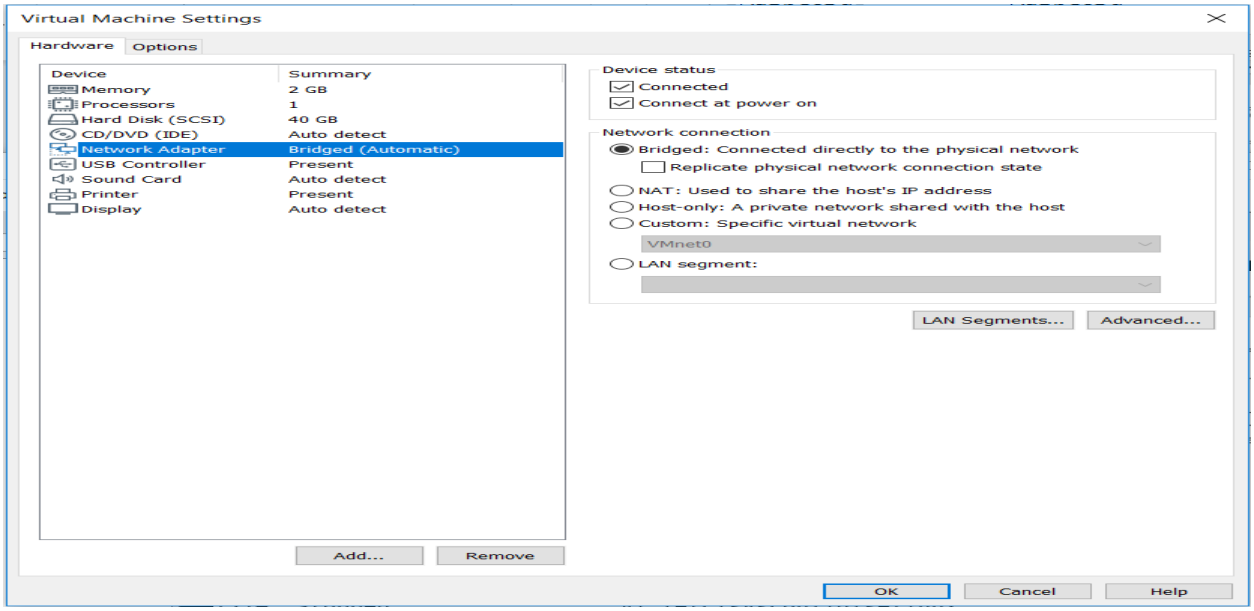


10) IP address

10.1) Virtual machine settings

10.2) Click network adapter

10.3) Click bridged: connected directly to the physical network



10.2) Click Applications / Terminal

10.3) Type nmtui edit eno16777736

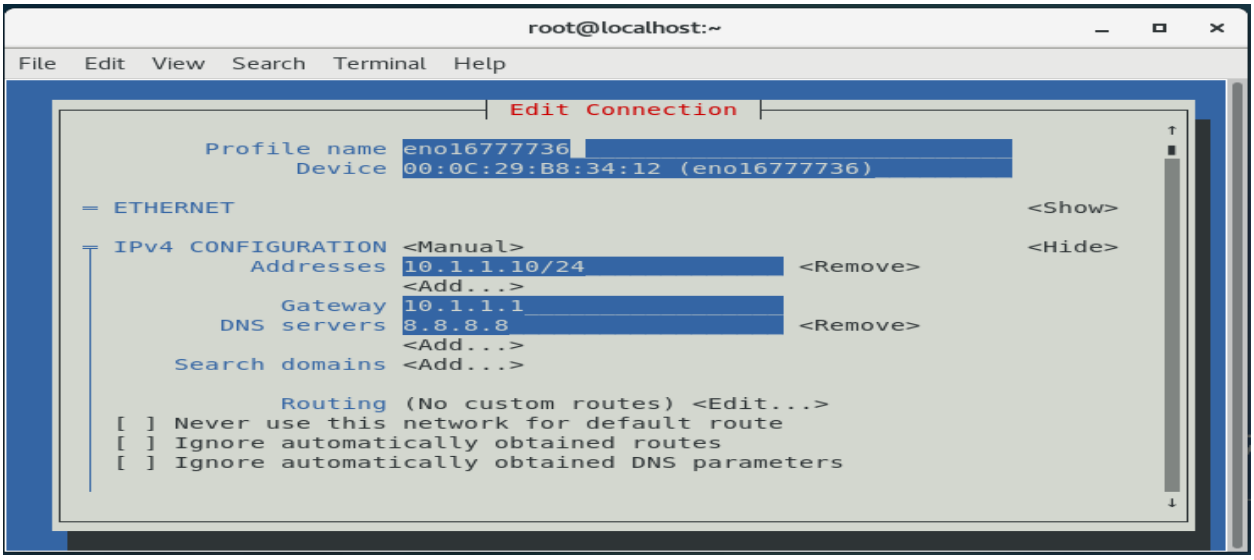
10.3.1) 10.1.1.10/24 in addresses

10.3.2) 10.1.1.1 in gateway

10.3.3) 8.8.8.8 in DNS servers

10.3.4) Click ok

10.3.5) reboot



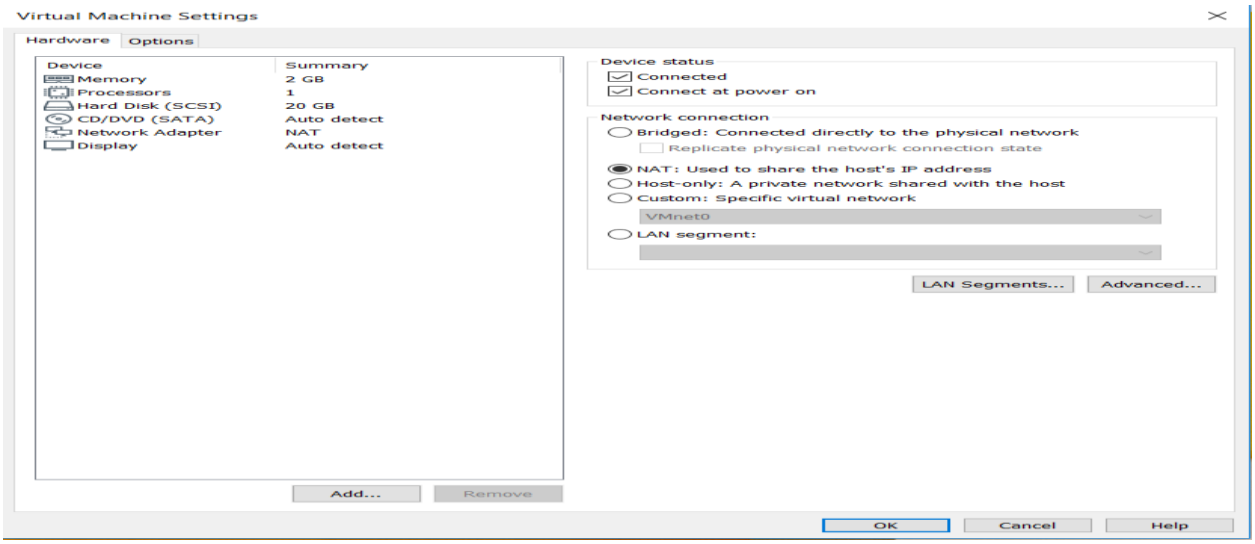
Configurations of Window Client:

11) IP address

11.1) Virtual machine settings

11.1.1) Click network adapter

11.1.2) Click nat: used to share the host's IP address



11.2) Network connections

11.2.1) Click Ethernet 0 / properties

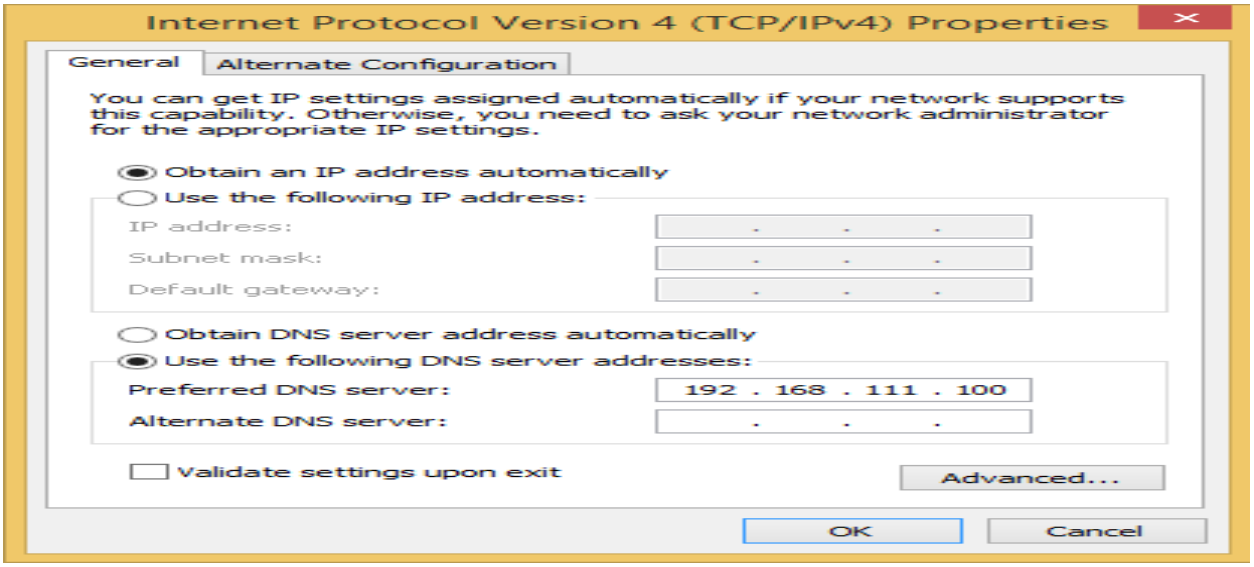
11.2.2) Click Internet Protocol Version 4 (TCP/IPv4)

11.2.3) Click obtain an IP address automatically

11.2.4) Click use the following DNS server address automatically

11.2.5) Type 192.168.111.100

11.2.6) Click ok



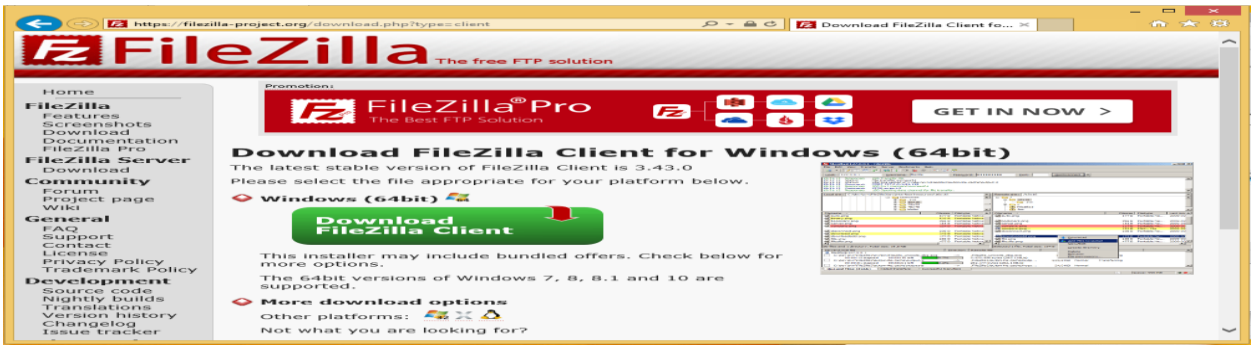
12) Install FTP Client

12.1) Click Internet explorer

12.2) Type http://www.filezilla-project.org

12.3) Click download

12.4) Click download Filezilla client



13) Nmap

13.1) Click Internet explorer

13.2) Type https://nmap.org/download.html

13.3) Click download

13.4) Click nmap-7.70-setup.exe