# Lab 1: Vyatta Firewalls - Overview

**Aim:**

The aim of this lab is to introduce the vSoC virtualisation teaching platform and vSphere client access to your own virtual machines and to understand how to configure a Vyatta firewall for NAT and firewall rules, demonstrating some fundamentals around network security and device configuration.

**Time to Complete:**

4 hours (two supervised hours in the lab, and two additional unsupervised hours).

**Activities:**

* **Complete Lab 1:** Vyatta Firewall

**Learning activities:**

At the end of this lab, you should understand:

* How to access the vSoC Cloud, working with your own folder within CSN09112.
* How to launch virtual machines, such as your Ubuntu, Windows Server, and Vyatta ones.
* How to configure the network settings of the Ubuntu and Windows Server machines, as well as using some basic Linux and Windows commands.
* How to configure the Vyatta firewall, for basic routing, NAT, and filtering - to grant or block access to certain types of packets and protocols.
* How to use Wireshark to capture network packets for deep analysis.

**Reflective statements (end-of-exercise):**

* What is the most important things when setting up a host, in order that it can connect with other networks?
* With the Vyatta firewall, how does the firewall protect against threats?

# Lab 1: Vyatta Firewalls

Part 1 Demo: https://youtu.be/LBTRGbuSUDg

## A Setting up the network

Figure 1 outlines the setup of a lab which uses the Vyatta firewall, and has three main networks: public, private and DMZ. Log into vSphere (**vsoc2.napier.ac.uk**) and locate the **CSN09112** folder. You should then be able to find your virtual machines. The public network uses DHCP to assign IP address, but the other two networks are assigned as:

10.10.*x*.0/24 10.10.*y*.0/24 [You will be allocated an IP address range in the lab]

Timeline

Description automatically generated

**Figure 1:** Lab setup (eth10 – Public, eth9 – Private, eth11 – DMZ)

Graphical user interface, application

Description automatically generated  
**Figure 2:** vSoC 2 setup

Use the network diagram in Figure 3, filling in the boxes with your addressing, the allocated networks, subnets, and IP addresses, and use as reference, as you complete the lab.

**User Passwords for VMs**

Vyatta: User: vyos, Password: vyos

Windows 7: Password: napier123

Ubuntu: User: root, password: napier123

Kali: User: root, password: toor

Diagram

Description automatically generated

**Figure 2:** Your network setup

Now conduct the following steps. The underlined option defines the answer that you should get.

☞ **Select your Ubuntu host** (User: napier, Password: napier123) and configure for 10.10.x.7 with a default gateway of 10.10.x.254 and a subnet mask of 255.255.255.0.

Can you ping the 10.10.*x*.7 port from the Ubuntu host? Yes/No

Commands used:

sudo ip addr add 10.10.**x**.7/24 dev ens32

sudo ip route add default via 10.10.**x**.254 dev ens32

☞ **Select your Windows server** (User: Administrator, Password: napier123) and configure it at 10.10.y.7 with a default gateway of 10.10.*y*.254 and a subnet mask of 255.255.255.0.

Can you ping the 10.10.y.7 port from the Window’s host? Yes/No

☞ From each of your hosts, can you ping the other host? Yes/No

Why can’t you ping the other host?

## B Routing between connected networks

Now, we will enable the firewall, and setup the IP addresses of the interfacts. Start up your Vyatta firewall.

☞ Login to the firewall, with: (User: vyos, Password: vyos)

View the current configuration with the command:

show configuration

Initially erase the configuration in the firewall, and reboot it, with:

cp /opt/vyatta/etc/config.boot.default /opt/vyatta/etc/config/config.boot

reboot

Next perform the following:

☞ Setup a few simple things, such as the hostname, a username and password, and so on:

$ configure

# set system host-name *yourname*

# set system login user *yourname* authentication plaintext-password *yourpass*

☞ Configure the firewall using the following commands (changing the *x* and *y* for your net):

$ configure

# set interfaces ethernet eth10 address dhcp

# set interfaces ethernet eth9 address 10.10.**x**.254/24

# set interfaces ethernet eth11 address 10.10.**y**.254/24

# set protocols static route 0.0.0.0/0 next-hop **10.221.3.254**

Before you commit the configuration, can you ping the 10.10.y.7 port from Ubuntu, and 10.10.x.7 from the Windows host? Yes/No

Now go ahead and commit the configuration with:

# commit

And exit the current configuration mode with:

# exit

Can you ping the 10.10.y.7 port from the Ubuntu host? Yes/No

Can you ping the 10.10.x.7 port from the Windows host? Yes/No

Now run Wireshark on your hosts, and repeat the ping command. Examine you network trace, and determine the successful ping request, and ping reply. Which ICMP **type codes** are used for the request and the successful reply:

What do the following commands do:

show configuration

show interface

Note: On Ubuntu you run Wireshark with: sudo wireshark

Now delete the IP address on the eth1 interface on the firewall, and reassess:

Can you ping the 10.10.y.7 port from Ubuntu? Yes/No

Can you ping the 10.10.x.7 port from Windows? Yes/No

Note:

$ configure

# delete interfaces ethernet eth1 address 10.10.x.254/24

# commit

Now, reapply the IP address, and using the arp –a command, determine the MAC addresses of the gateway interface on Vyatta, and check this against the configuration of the firewall.

What are the MAC addresses of the firewall:

Now with a Web browser on each host, access the Web server on the other network.

Can you access the Web server on the 10.10.y.7 from Ubuntu? Yes/No

Can you access the Web server on the 10.10.x.7 from Windows? Yes/No

As before, delete the IP address on the eth1 port, and reapply (make sure you refresh the cache on the browser):

Can you access the Web server on the 10.10.y.7 from 10.10.x.7? Yes/No

Can you access the Web server on the 10.10.x.7 from 10.10.y.7? Yes/No

**Reapply** everything as before, and test that it still works.

Startup Wireshark on each of your hosts, and capture traffic.

☞ Run an nmap scan from the Windows host to 10.10.*x*.7. What ports are open on the Linux host:

☞ Run an nmap scan from the Linux host to 10.10.*y*.7. What ports are open on the Windows host:

Commands:

nmap 10.10.x.0

## C Setting up NAT

Now we need to setup NAT to map the addresses on the DMZ and the private network to an address taken from the public network. We are using NAT overloading (or NAT masquerade) and which will map the private addresses to a public address (taken from eth0). This will allow hosts on the private and DMZ to connect to the Internet (through the main gateway – 10.221.3.254):

To map the addresses from the private to the public network:

# set nat source rule 1 outbound-interface eth10

# set nat source rule 1 source address 10.10.x.0/24

# set nat source rule 1 translation address masquerade

# commit

# save

To map the addresses from the DMZ to the public network:

# set nat source rule 2 outbound-interface eth10

# set nat source rule 2 source address 10.10.y.0/24

# set nat source rule 2 translation address masquerade

# commit

# save

You should now have a network connection from the private and DMZ networks to the public network.

On your Ubuntu host change the name server to 10.221.3.254 with:

sudo nano /etc/resolv.conf

And change the nameserver to:

nameserver 10.221.3.254

Now can you ping 10.221.3.254 from Ubuntu? Yes/No

Now can you ping 10.221.3.254 from Windows 7? Yes/No

Now can you ping 8.8.8.8 from Ubuntu? Yes/No

Now can you ping 8.8.8.8 from Windows 7? Yes/No

Now can you access Google.com from Ubuntu? Yes/No

Now can you access Google.com from Windows 7? Yes/No

If you answer No to any of these questions, your network is not working correctly. Ask you tutor to help find the problem.

## D Setting up services on firewall

Now save your configuration in edit mode with:

# save

# exit

$ reboot

Now restart Wireshark on the Linux install. Next enable the SSH server on the Vyatta firewall with:

# set service ssh

# commit

Now ssh into the Vyatta firewall from the Linux host using:

ssh 10.10.y.254 –l *username*

Was the login successful? Yes/No

Check to see if you have a **public Kali** instance in your group folder. If so, complete the following:

From your Kali instance, can you ping each of the interfaces on the firewall: Yes/No

From your Kali instance, can you ping each of the interfaces on the hosts: Yes/No

Now setup the default gateway of your public Kali host to be the IP address of your public IP address port on your firewall. Are you now able to ping your Ubuntu and Windows machines? Yes/No?

If so, why can you now ping them?

Note: To setup a default gateway on Kali (and where w.x.y.z is the IP address of your public interface):  
  
sudo ip route add default via *w.x.****y****.z* dev eth0

## E Firewalling

The Vyatta firewall uses zones to define security regions. In this case, we can setup **public**, **dmz** and **private**. Then we apply firewall rules to define how the traffic between the zones is filtered. In this case we will only setup the traffic between the dmz and private, with two rules: **dmz2private** and **private2dmz**. Possible filtering is to allow connections on certain ports from private to DMZ (80 and 433), but allow all the connections from DMZ to private. Figure 4 outlines the setup.

Diagram

Description automatically generated

**Figure 4:** Zone and firewall rule setup

Now we will block all the traffic from Private to DMZ (apart from Port 80 and 433), and allow all the traffic from DMZ to the Private network. To enable firewalling, we first define some zones (private, public, and dmz):

set  zone-policy  zone  private  description  "Inside”

set  zone-­policy  zone  public  description  "Outside”

set  zone-­policy  zone  dmz description  "DMZ”

These zones are then applied onto the interfaces:

set  zone-­policy  zone  public  interface  eth0

set  zone-­policy  zone  private interface  eth1

set  zone-­policy  zone  dmz  interface  eth2

Now try to access services from the Linux instance to the Windows one:

Run nmap, and see if you can access these services:

HTTP Yes/No

HTTPs Yes/No

FTP Yes/No

Note:   
  
nmap 10.10.y.7

Now try to access services from the Windows instance to the Linux one:

Run nmap, and see if you can access these services:

HTTP Yes/No

HTTPs Yes/No

FTP Yes/No

Note:   
  
nmap 10.10.x.7

Now we will allow only established connections from the DMZ to the private network:

set  firewall  name  dmz2private description  "DMZ to private"

set  firewall  name  dmz2private rule  1  action  accept

set  firewall  name  dmz2private rule  1  state  established  enable

set  firewall  name  dmz2private rule  1  state  related enable

Then we will accept connections on port 80 and 443 from the private network to the DMZ:

set  firewall  name  private2dmz description  "private to DMZ"

set  firewall  name  private2dmz rule  1  action  accept

set  firewall  name  private2dmz rule  1  state  established  enable

set  firewall  name  private2dmz rule  1  state  related enable

set  firewall  name  private2dmz rule  10  action  accept

set  firewall  name  private2dmz rule  10  destination  port  80,443

set  firewall  name  private2dmz rule  10  protocol tcp

Now we have zones of public, dmz and private, and rules of dmz2private and private2dmz. To apply the rules to zones we complete with:

set  zone-­policy  zone  private from  dmz firewall  name  dmz2private

set  zone-­policy  zone  dmz from  private firewall  name  private2dmz

Commit this, and try and connect from each of the networks to the other:

From the Linux host on the private network access the following services on the Windows server in the DMZ:

HTTP Yes/No

HTTPs Yes/No

SSH Yes/No

FTP Yes/No

From the Windows machine on the public network access the following services on the Linux server in the private network:

HTTP Yes/No

HTTPs Yes/No

SSH Yes/No

FTP Yes/No

Are the results as expected?

Now enable FTP (Port 21) access from the private network to the DMZ.

Enable Wireshark on the Windows host, and observe the trace when you nmap from the Linux host. What can you observe:

Enable Wireshark on the Linux host, and observe the trace when you nmap from the Windows host. What can you observe:

Note. You can test whether the port is open by using telnet on the given port number:

Test FTP: telnet 10.10.x.7 21

Test SSH: telnet 10.10.x.7 22

Test HTTP: telnet 10.10.x.7 80

Test SMTP: telnet 10.10.x.7 25

If you receive a response, the port is open, if not it is closed.

## E DoS Protection

A particularly difficult area to protect against is Denial of Service (DoS). The Vyatta firewall has protection for this, where it limits the number of connections over a given amount of time. Now let’s limit the number of Web connections to 5 in 10 seconds:

set  firewall  name  private2dmz rule 5 action  drop

set  firewall  name  private2dmz rule 5 protocol tcp

set  firewall  name  private2dmz rule 5 destination  port 21,23,25,80,443

set  firewall  name  private2dmz rule 5 recent  count  5

set  firewall  name  private2dmz rule 5 recent time 10

Commit this.

Run Wireshark on the Windows host. From the Linux host on the private network, now try and hping from the Linux host to the Windows host. What do you observe from the Wireshark trace and also from the return from hping:

How many connections where accepted before it stopped?

Note. To perform an hping on 10.1.1.7 on port 80:

hping 10.1.1.7 -S -V -p 80

## F Appendix

Now restart Wireshark on the Linux install. Next enable the DHCP server for the Linux host on the Vyatta firewall with:

# set service dhcp-server shared-network-name ETH1 subnet 10.10.x.0/24 start 10.10.x.9 stop 10.10.x.100

# set service dhcp-server shared-network-name ETH1 subnet 10.10.x.0/24 default-router 10.10.x.254

# commit

Now renew the IP address on the Linux host with:

sudo dhclient –r

sudo dhclient

What is the IP address that was allocated to the Linux instance from the DHCP server:

What is the data packet that is sent to release the IP address from the interface:

IP addresses used:

UDP ports used:

Bootstrap Message:

What is the handshake that is used to gain the IP address from the DHCP server:

## Configuration Mode Commands

|  |  |
| --- | --- |
| **Cisco** | **Vyatta** |
| **SAVE** |  |
| copy run tftp N/A | save tftp://*ip*/*name*  save /mnt/floppy/config/config.boot |
| **SHOW** |  |
| show | show running-config |
| **SET SERVICE** |  |
| ip server http line vty 0 4   password | set service http set service telnet   set service ssh |
|  |  |
| ip dhcp pool   network   default-router     ip dhcp excluded-address | set service dhcp-server name  edit service dhcp-server name    set start ... stop    set default-router   set network-mask   set interface   set exclude |
| **SET SYSTEM** |  |
| ip domain-name ip default-gateway hostname username … password … ntp server ip name-server terminal monitor clock timezone | set system domain-name set system gateway-address set system host-name set system login set system ntp-server set system name-server set system syslog console set system time-zone |
| **INTERFACES** interface   set description    ip address    duplex   speed | edit interfaces    set description    set address ... prefix-length   set duplex   set speed |
| **STATIC** ip route | set protocols static route ...next-hop ... |

## Operational Mode Commands

|  |  |
| --- | --- |
| **Cisco command**  ping  traceroute  show arp  show ip ospf neighbor  show ip ospf database  show ip route  show ip route | include  show ip interfaces    show clock  show ntp associations  show ip dhcp binding  show ip dhcp server statistics  show ip nat translations  show ip nat statistics | **Vyatta command**  ping  raceroute  show arp  show ospf4 neighbor  show ospf4 database  show route  show route | match ...  show interfaces ethernet ... physical  show interfaces ethernet ... statistics  show host date  show ntp associations  show dhcp lease  show dhcp statistics  show nat rules  show nat statistics |