



NETWORK SECURITY LABORATORY

TOPIC 9 - GROUP 23

Firewall Stateless

Submitted To:
Dottor
Luca Allodi

Submitted By :
Andrea Brun
Matteo Grossele
Matteo Gabburo

Contents

1	Configuration of laboratory	3
1.1	Topology	3
1.2	Virtual Machines	3
1.2.1	VM Green	4
1.2.2	VM Firewall	4
1.2.3	VM Red	5
1.3	Script	6
1.3.1	How to use the script	6
2	Theoretical part	7
2.1	Firewall	7
2.1.1	Stateless	7
2.1.2	Policies	7
2.1.3	Goals	7
2.2	Iptables	8
2.2.1	Table	8
2.2.2	Command	8
2.2.3	Chain	9
2.2.4	Parameter	9
2.2.5	Target	10
3	Task 1 - Default Policies	11
3.1	Example - Block input on Firewall	11
3.2	Exercise - Default deny traffic through Firewall	12
3.2.1	Solution	12
4	Task 2 - Allow single Protocols	13
4.1	Example - Allow only SSH	13
4.2	Exercise - Allow only HTTP	14
4.2.1	Solution	15
5	Task 3 - Block a part of a Protocol	16
5.1	Example - Allow only SSH in one direction	16
5.2	Exercise - Allow only HTTP in one direction	17
5.2.1	Solution	17
6	Task 4 - Modify an existing and well-formed configuration	18
6.1	Example - Allow forward connections	18
6.2	Exercise - Block forward connections	19

6.2.1	Solution	19
7	Task 5 - Modify an existing and bad-formed configuration	20
7.1	Example - Block HTTP	20
7.2	Exercise - Allow only HTTP	21
7.2.1	Solution	21
8	Attachments	22
8.1	Deepening: OSI Layer 3	22
8.2	Deepening: Internet Protocol	22
8.3	Deepening: Transfer Control Protocol	22
8.3.1	TCP: Three-way Handshake	22
8.4	Deepening: Secure Shell	23
8.5	Deepening: HyperText Transmission Protocol	23

1 Configuration of laboratory

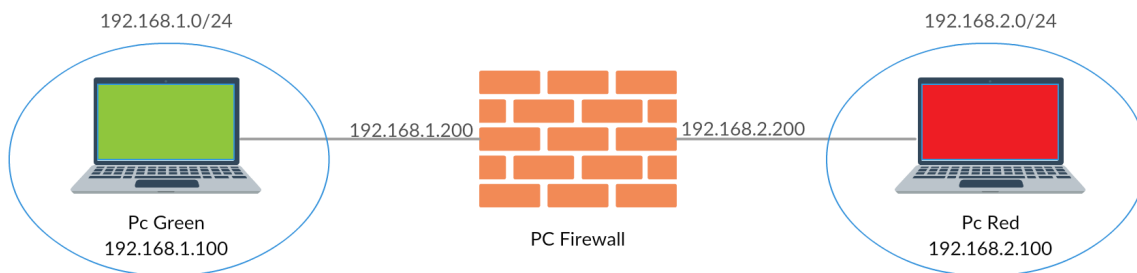
The objective of our laboratory is to demonstrate the use of a stateless firewalls. So we thought to implement a simple structure composed by two machines that communicate with each other through a third that work as router and firewall. To do so we decided to use some virtual machines, instead of a software like Netkit, to allow the use of scripts for performing the exercises in a simpler way.

1.1 Topology

Our network is composed by 2 different subnetworks, with type C addresses, with a third machine connected on both subnetworks. This machines is the Firewall, it is configured to allow packets to flow from a Network to another one, allowing routing through the 2 networks. The 2 subnetworks are :

- Network 1: 192.168.1.0/24
- Network 2: 192.168.2.0/24

On Network 1 there is PC Green, with IP address 192.168.1.100, while PC Red is on Network 2, with IP address 192.168.2.100. PC Firewall is in between them, connected to both the 2 networks with 2 different interfaces. His IP addresses are 192.168.1.200 and 192.168.2.200



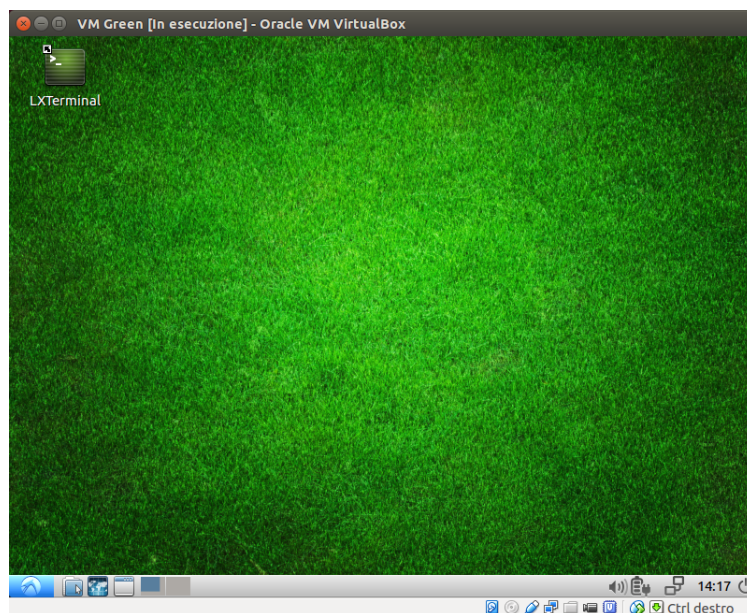
1.2 Virtual Machines

The virtual machines that we use are based on a Linux distribution, Lubuntu 16.04. Before the workshop we have configured the different machines. So for each virtual machine we have created a user with the name of the virtual machine and who has the password "password" and we have set the network features depending on our topology (we can check the configuration with ifconfig command). In particular we

use the host Green and Red to try various type of communication with the others machines and to check the rules of the firewall that we have in Firewall.

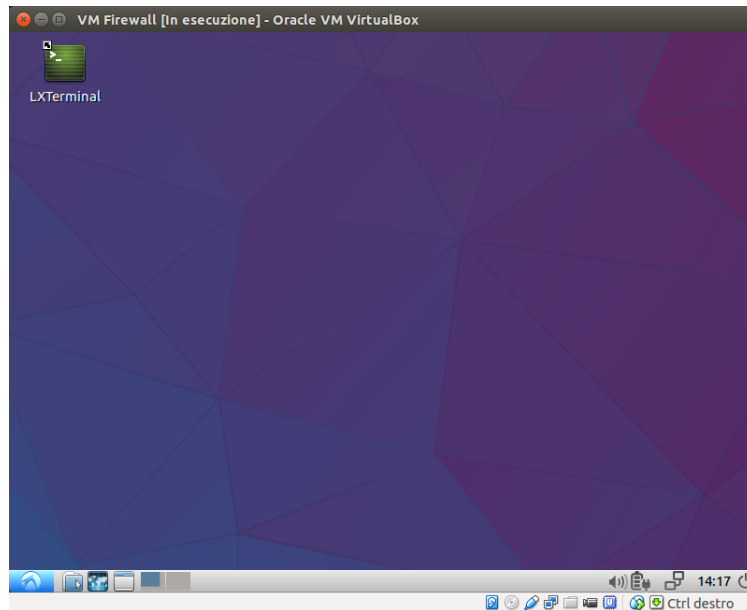
1.2.1 VM Green

The first virtual machine is Pc Green. It is part of the network 1 and its interface has the IP address 192.168.1.100. It has installed only an ssh server.



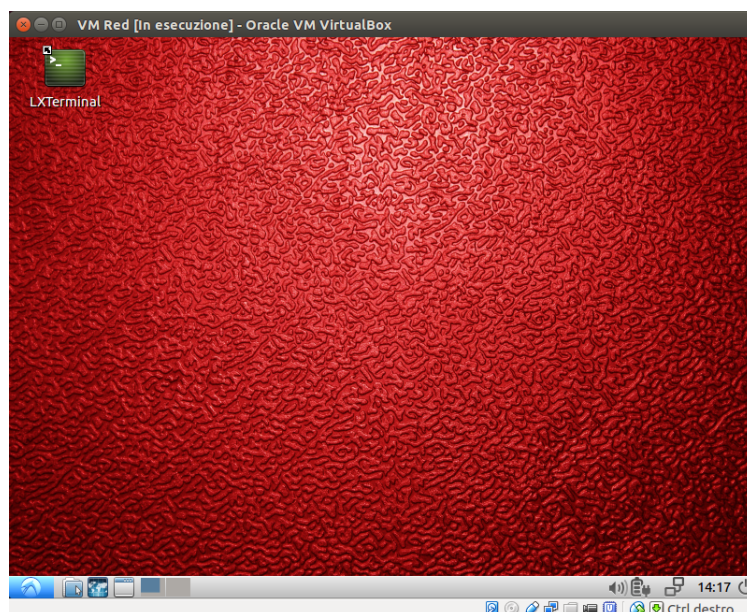
1.2.2 VM Firewall

The Pc firewall is the core of the lab. It is setup has router and do forwarding between the other two machines. So it has two interface, one for subnetwork, that have as IP address 192.168.1.200 and 192.168.2.200. To work with the firewall we have create some scripts that are on its desktop.



1.2.3 VM Red

The last machine is Pc red. It is in the subnet 2 and its IPv4 is 192.168.2.100. In this pc are installed an Apache server and an SSH server.



1.3 Script

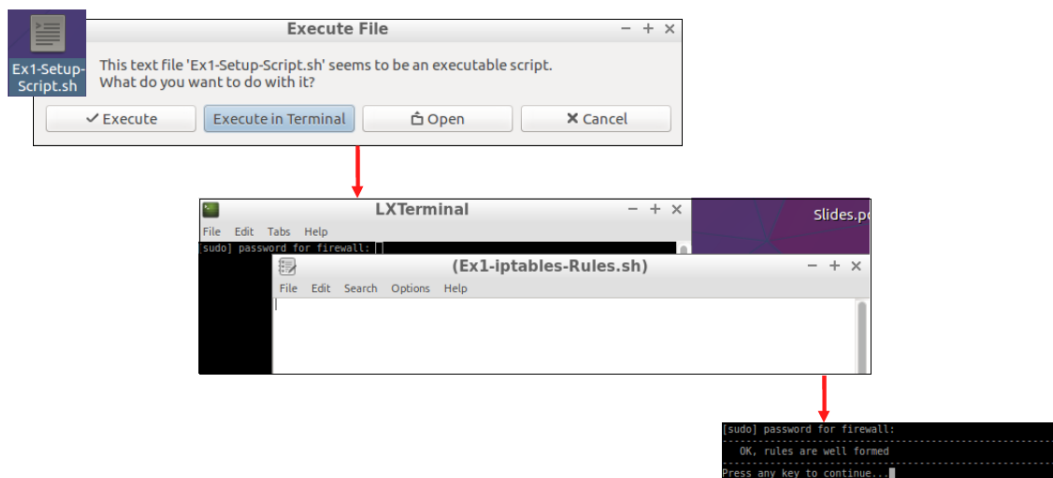
To simplify the task, we created a simple script to setup the files needed. The script do:

- Get Superuser privileges to execute iptables commands
- Initialize iptables with a standard configuration (ALLOW everything)
- Create a file for the user firewall configuration
- Apply the configuration to the firewall
- Report to the user the success or the failure of the rules set

1.3.1 How to use the script

Every exercise has his own script on the Desktop. If you want to do an exercise, double click on the script and select "Execute in Terminal". It will open a terminal and a text editor. Ignore the terminal and write your iptables rules on the text file. Save and close the editor On the terminal you can see if your rules were correct or not. (If it says that the rules are correct, it means that they are well formed, not that they solve the exercise)

If you want to retry the exercise, re-open the script, and this time the text file will contain your previous commands



2 Theoretical part

2.1 Firewall

Firewall is a network security system, physical or software, that monitors and controls the incoming and outgoing network traffic based on predetermined security rules.

2.1.1 Stateless

Stateless firewall treats each IP packet individually so it is examined isolated from what has happened in the past.

Such packet filter operate at the OSI Network Layer(layer 3) and function more efficiently because they only look at the header part of a packet.

Pros:

- Easy to setup,
- Simple behavior of the Firewall,
- No modification on the traffic flow or the characteristics;

Cons:

- Allow a limited set of rules for the firewall,
- Can not prevent application-specific attacks;

2.1.2 Policies

Each firewall can work in two way, it can be set up as:

- Default deny: All what is not explicitly allowed is denied (list what is blocked in a Blacklist),
- Default permit: All what is not explicitly denied is allowed (list what is allowed in a Whitelist);

2.1.3 Goals

The goals of firewall are:

- All traffic from inside or outside must pass through the firewall (physically blocking all access to the local network except via the firewall) so it screen out connection from hackers, viruses, and worm,
- Allow to pass only authorized traffic (defined by the local security policy)
- The firewall itself is immune to penetration (use of a trusted system with a secure operating system)

2.2 Iptables

Iptables is an administration tool for IPv4 packet filtering and NAT. It is used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel.

Each rule specifies what to do with a packet that matches.

2.2.1 Table

Several different tables may be defined and each table contains a number of built-in chains and may also contain user-defined chains.

There are currently five independent tables. These are:

- filter: This is the default table (if no -t option is passed),
- nat: This table is consulted when a packet that creates a new connection is encountered,
- mangle: This table is used for specialized packet alteration,
- raw: This table is used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target,
- security: This table is used for Mandatory Access Control (MAC) networking rules, such as those enabled by the SECMARK and CONNSECMARK targets;

For our purpose we use the default table.

2.2.2 Command

The command specify the desired action to perform. Only one of them can be specified on the command line unless otherwise stated below.

The most important are:

- -A, -append: Append one or more rules to the end of the selected chain,
- -C, -check: Check whether a rule matching the specification does exist in the selected chain,
- -D, -delete: Delete one or more rules from the selected chain,
- -I, -insert: Insert one or more rules in the selected chain as the given rule number,
- -R, -replace: Replace a rule in the selected chain,
- -L, -list (chain): List all rules in the selected chain. If no chain is selected, all chains are selected,
- -F, -flush(chain): Flush the selected,
- -P, -policy: Set the policy for the chain to the given target,
- -h: Give a description of the command syntax;

(chain): chain is not obligatory.

2.2.3 Chain

Each chain is a list of rules which can match a set of packets.

The filter table contains these built-in chains:

- INPUT: for packets destined to local sockets,
- FORWARD: for packets being routed through the box,
- OUTPUT: for locally-generated packets;

2.2.4 Parameter

The parameters make up a rule specification (as used in the add, delete, insert, replace and append commands) or add some options to the call.

The most relevant are:

- (!) -p, -protocol protocol: The protocol of the rule or of the packet to check (tcp, udp, udplite, icmp, esp, ah, sctp, or "all"),
- (!) -s, -src, -source address[/mask]: Source specification (network name, a hostname, or an IP address also with /mask),

- (!) -d, -dst, -destination address[/mask]:Destination specification. (the same case of -s),
- -j, -jump target:This specifies the target of the rule; what to do if the packet matches it,
- (!) -i, -in-interface name:Name of an interface via which a packet was received,
- (!) -o, -out-interface name:Name of an interface via which a packet is going to be sent;

(!): the sense of argument is inverted.

2.2.5 Target

A packet that matches a rule is called a target, which may be a jump to a user-defined chain in the same table. If it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain or one of the special values, that are:

- ACCEPT:means to let the packet through,
- DROP:means to drop the packet on the floor,
- QUEUE:means to pass the packet to userspace,
- RETURN:means stop traversing this chain and resume at the next rule in the previous (calling) chain;

3 Task 1 - Default Policies

3.1 Example - Block input on Firewall

In this example we want to block every packets sent from VM Green to VM Firewall. To achieve this goal we will change the Firewall policy for the input packets.

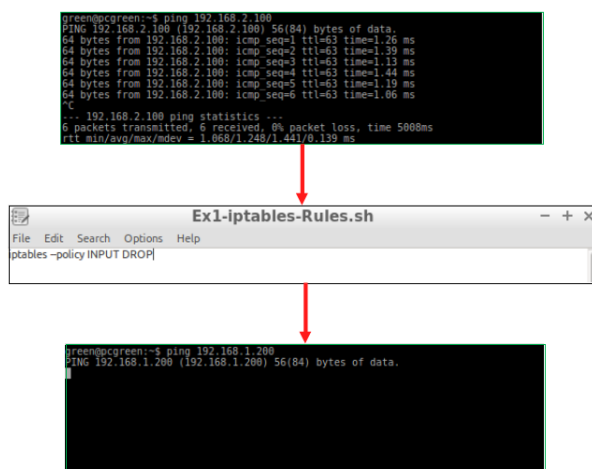
- Launch the script for ex1 on the Firewall, double click on the file and select 'launch on Terminal'. Just leave it open for now, we need it to reset previous rules on the Firewall that may cause problems with the exercise.
- Go on the VM Green and open a terminal.
- Ping the VM Firewall (ping 192.168.1.200). We do this test to verify that the connection between VM Green and Firewall works correctly and to show that Firewall responds to VM Green requests before we apply the rules
- Go back on the VM Firewall and write this iptables rule on the file:


```
# iptables --policy INPUT DENY
```

 Save and close the file to apply the rules. The script will tell you that the rules are malformed, since there is no DENY target in the list of the available targets. Re launch the script, edit the rules so it becomes:


```
# iptables --policy INPUT DROP
```

 Save and close, now the script will tell you that the rules are written correctly.
- Try to ping again VM Firewall from VM Green, and notice that now the Firewall does not respond to VM Green pings.



3.2 Exercise - Default deny traffic through Firewall

First exercise is to set the default policy of iptables to block the traffic from VM Green to VM Red. Re-launch the script we used for the example and change the rule for the default policy. Remember the different chains that the table filter uses to filter the packets.

3.2.1 Solution

The solution for this exercise is achieved changing the default policy for the FORWARD chain. The iptables command you need to use is:

```
# iptables --policy FORWARD DROP
```

4 Task 2 - Allow single Protocols

4.1 Example - Allow only SSH

In this example we want only the ssh connections to be able to travel from VM Green and VM Red. Every other traffic must be denied from the Firewall. For more information on SSH,HTTP,IP Protocol and TCP look on the Attachment section.

The steps you need to follow in order to do the second example are :

- Launch the script for exercise 2 on the Firewall desktop, like for the exercise 1 you need to run it into the terminal
- Go on the VM Green and open a terminal
- Try to communicate with VM Red

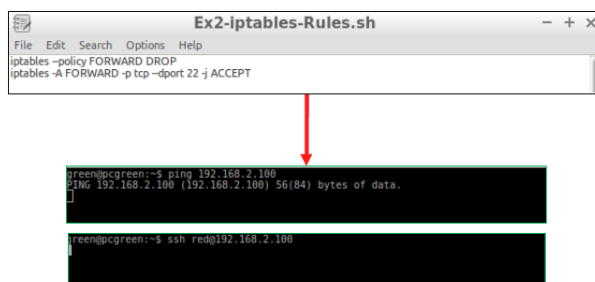
- Ping the VM Red (ping 192.168.2.100) → The ping uses the protocol ICMP
- Connect with SSH to the user red on VM Red (ssh red@192.168.2.100)

- Go back on the Firewall and write the iptables rule on the file :

```
# iptables --policy FORWARD DROP
# iptables -A FORWARD -p tcp --dport 22 -j ACCEPT
```

The first command is the same from the previous exercise and it's there to make the Firewall block every connection not specified on the rules.

The second one is used to allow the SSH connection to pass through the Firewall.



- Test on the VM Green if you can still ping or connect with ssh on MV Red. Now it is impossible to ping the VM Red, but also to connect with ssh. This

is not the expected result for the example, since we want to be able to use ssh without problems.

- The reason is that the firewall allows connections on port 22, but it does not allow replies coming from VM Red to the VM Green.
- Relaunch the script for example 2 on Firewall and add this rule to allow responses :
iptables -A FORWARD -m state --state=ESTABLISHED,RELATED -j ACCEPT
- Go back to VM Green to test if the connection to VM Red works. You will notice that now it's still impossible to ping VM Red but now you will be able to connect with SSH.



The image shows a terminal window titled '*Ex2-iptables-Rules.sh' with a menu bar (File, Edit, Search, Options, Help). The terminal content is:

```
iptables -policy FORWARD DROP
iptables -A FORWARD -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -m state --state=ESTABLISHED,RELATED -j ACCEPT
```

A red arrow points from the third line of the script to a terminal window below. The terminal window shows the following sequence of commands and outputs:

```
green@pcgreen:~$ ping 192.168.2.100
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data.
^C
```

```
green@pcgreen:~$ ssh red@192.168.2.100
red@192.168.2.100's password:
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-21-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
12 packages can be updated.
6 updates are security updates.
Last login: Wed May 11 19:01:01 2016 from 192.168.1.100
red@pcred:~$
```

- Test the same thing for VM Red to VM Green to check that it works in both ways.

4.2 Exercise - Allow only HTTP

In the second exercise we want to allow only specific protocols to pass through the firewall. Now the students need to try on their own to allow only the HTTP protocol through the Firewall.

4.2.1 Solution

The solution for this exercise is achieved changing the iptables rules that allow the SSH traffic to pass through the Firewall. Since HTTP uses the port 80 instead of the port 22 used from SSH, to make the HTTP allowed we need to change that rule. The iptables command you need to use is :

```
# iptables -A FORWARD -p tcp --dport 80 -j ACCEPT
```

The other 2 rules remains the same, since you still want to block every other connection and you still need to be able to make the answers from VM RED come back to VM Green.

5 Task 3 - Block a part of a Protocol

5.1 Example - Allow only SSH in one direction

Our goal in this example is to block all the traffic from the two machines VM Red and VM Green with an exception for the SSH traffic coming from VM Green and directed to VM Red.

- Launch the script for exercise 3 on the Firewall the same way we did for the previous exercises.
- Go on the VM Green and open a terminal
- Connect with SSH to the user red on VM Red and do the same from VM Red, try to connect to user green on VM Green. The two commands to input into the terminals are :

```
# ssh red@192.168.2.100
# ssh green@192.168.1.100
```

- Go back on the Firewall and write the iptables rule on the file :
First, the command to set the default policy to reject everything

```
# iptables --policy FORWARD DROP
```

And the command to allow the SSH connection to pass through the Firewall

```
# iptables -A FORWARD -s 192.168.1.100 -d 192.168.2.100 -p tcp --dport 22 -j ACCEPT
```

This command follows the same idea of the command we used in the exercise 2 but with the addition of a predefined source and destination for the packets.

- Add the rule to allow the responses go through the Firewall

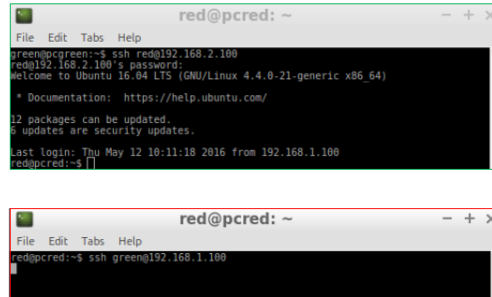
```
# iptables -A FORWARD -m state --state=ESTABLISHED,RELATED -j ACCEPT
```



```
Ex3-iptables-Rules.sh
File Edit Search Options Help
iptables --policy FORWARD DROP
iptables -A FORWARD -d 192.168.2.100 -s 192.168.1.100 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -m state --state=ESTABLISHED,RELATED -j ACCEPT
```

- Test the ssh client from VM Green to VM Red and test it works. Do the same test from VM Red to VM Green and notice that this time this does not work. Use the command :

```
# ssh green@192.168.1.100
```



```
red@pcred: ~  
File Edit Tabs Help  
green@pcred:~$ ssh red@192.168.2.100  
red@192.168.2.100:~$ password:  
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-21-generic x86_64)  
  
* Documentation: https://help.ubuntu.com/  
12 packages can be updated.  
0 updates are security updates.  
Last login: Thu May 12 10:11:18 2016 from 192.168.1.100  
red@pcred:~$  
red@pcred:~$ ssh green@192.168.1.100
```

5.2 Exercise - Allow only HTTP in one direction

In the third exercise we want to allow specific protocols but this time only from one direction of the Firewall. Now try to block everything except the traffic HTTP to VM Red.

5.2.1 Solution

The solution for this exercise is achieved in the same way we solved the previous exercise: changing the iptables rules that allow the SSH traffic to pass through the Firewall. The iptables command you need to use is :

```
# iptables -A FORWARD -s 192.168.1.100 -d 192.168.2.100 -p tcp --dport 80 -j ACCEPT
```

The other 2 rules remains the same, since you still want to block every other connection and you still need to be able to make the answers from VM RED come back to VM Green.

6 Task 4 - Modify an existing and well-formed configuration

Exercises four and five are a bit different from the previous ones, because now we start with an initial configuration of the firewall. This configuration work and it is well formed, but doesn't do what we want. For this reason the fourth and the fifth exercise aims to understand and modify existing and given rules for achieve a goal.

6.1 Example - Allow forward connections

Start by a configuration who not accept connections, modify the existing rules in order to make a communication between host VMGreen and VMRed.

```
firewall@pcfirewall:~/Desktop$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
DROP      all  --  192.168.1.100          anywhere
DROP      all  --  192.168.2.100          anywhere

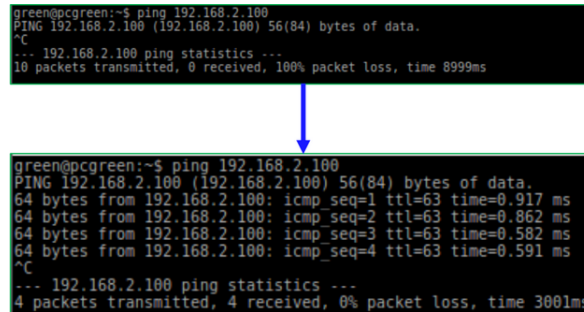
Chain FORWARD (policy ACCEPT)
target     prot opt source                destination
DROP      all  --  192.168.1.100          anywhere
DROP      all  --  192.168.2.100          anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
DROP      all  --  192.168.1.100          anywhere
DROP      all  --  192.168.2.100          anywhere
```

- Launch the script for ex 4 on the Firewall
- Ping VMRed (ping 192.168.2.100), it shouldn't work
- Back to the Firewall and watch the iptables rules which are in the file opened previously
- Try to understand what do you have to modify, and do it, for achieve the goal. In this case what we have to do is change rules in chain FORWARD, to ACCEPT
- Modify these two rules


```
# iptables -I FORWARD -s 192.168.2.100 -j DROP
to
# iptables -I FORWARD -s 192.168.2.100 -j ACCEPT
and
# iptables -I FORWARD -s 192.168.1.100 -j DROP
to
# iptables -I FORWARD -s 192.168.1.100 -j ACCEPT
```
- Try to ping again VM VMRed from VMGreen

At this point we have achieved the goal because VMGreen and VMRed can communicate.



```
green@pcgreen:~$ ping 192.168.2.100
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data.
^C
--- 192.168.2.100 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 8999ms

green@pcgreen:~$ ping 192.168.2.100
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data:
64 bytes from 192.168.2.100: icmp_seq=1 ttl=63 time=0.917 ms
64 bytes from 192.168.2.100: icmp_seq=2 ttl=63 time=0.862 ms
64 bytes from 192.168.2.100: icmp_seq=3 ttl=63 time=0.582 ms
64 bytes from 192.168.2.100: icmp_seq=4 ttl=63 time=0.591 ms
^C
--- 192.168.2.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
```

6.2 Exercise - Block forward connections

The fourth exercise aims to understand the rules behind a firewall and change them. In this case we have a configuration that accepts connections. Modify the existing rules in order to block a communication between host VMGreen and VMRed.

6.2.1 Solution

The solution for this exercise is the same as the exercise explained before (6.1). While in the fourth we had to modify the initial configuration to allow communication between the machines, now we have to block it. For doing it, the better and simplest solution is to change the FORWARD rules, from ACCEPT to DROP. After this modification, VMGreen shouldn't ping VMRed.

7 Task 5 - Modify an existing and bad-formed configuration

The fifth exercise aims to identify and fix possible errors in the rules of an existing configuration of the firewall. All the rules are well formed, but there are semantic error and they can generate same conflicts.

7.1 Example - Block HTTP

Start by a configuration who accept any communication and modify it for block only HTTP communications from VMGreen to VMRed

```
firewall@pcfiredns:~/Desktop$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- 192.168.1.100 anywhere
ACCEPT all -- 192.168.2.100 anywhere

Chain FORWARD (policy ACCEPT)
target prot opt source destination
ACCEPT all -- 192.168.1.100 anywhere
ACCEPT all -- 192.168.2.100 anywhere
DROP tcp -- anywhere anywhere tcp spt:http

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- 192.168.1.100 anywhere
ACCEPT all -- 192.168.2.100 anywhere
```

- Launch the script for ex 5 on the Firewall
- Go on the VMGreen and open a terminal
- Ping VMRed (ping 192.168.2.100), it should works and it is fine
- Try to do an http request to VMRed (wget 192.168.2.100), it should work and it's not good for our purpose
- Back to the Firewall and watch the iptables rules which are in the file opened previously
- Try to understand what do you have to modify for achieve the goal, and do it. In this case what we have to do is simple, we have to cancel two existing rules


```
# iptables -I FORWARD -s 192.168.2.100 -j ACCEPT
# iptables -I FORWARD -s 192.168.1.100 -j ACCEPT
```
- Try to do an http request again from VMGreen



The image shows two terminal windows. The top window shows a successful wget command: `wget 192.168.2.100` which connects to `http://192.168.2.100/`, receives a `200 OK` response, and saves `index.html` (11,06K). A blue arrow points from the bottom of this window to the top of the second window. The second window shows the same `wget` command being executed again, but it is cut off at the point of connecting to `192.168.2.100:80`.

7.2 Exercise - Allow only HTTP

Like the fourth, the fifth aim to understand the rules behind a firewall. In this case the exercise is a bit different, because there are some conflict with the rules and what we have to do is find these conflicts and delete them. In particular the configuration doesn't accept any communication, modify it for allow only HTTP communications from VMGreen to VMRed

7.2.1 Solution

For solve this exercise we have to work to the configuration in order to change it. As we said before every rule are well formed, for this reason iptables doesn't return any error, but the entire scenario has some problem. What we have to do is fix some rules and add one.

- The first thing to do is modify the policy rule,
iptables --policy FORWARD DROP
to
iptables --policy FORWARD ACCEPT
- The Second thing to do is modify the rule that should accept port 80,
iptables -A -p tcp --sport 80 -j DROP
to
iptables -A -p tcp --dport 80 -j ACCEPT
- The third and latest thing to do add the rule for collect the response of a protocol,
iptables -A FORWARD -m state --state=ESTABLISHED,RELATED -j ACCEPT

8 Attachments

8.1 Deepening: OSI Layer 3

The Network Layer is responsible for packet forwarding through the Internet Protocol (IP).

So functions of the network layer include:

- Connectionless communication
- Host addressing
- Message forwarding

8.2 Deepening: Internet Protocol

The Internet Protocol is the principal communications protocol for relaying datagrams across network boundaries.

For this purpose, the IP defines the format of packets and provides an addressing system that has two functions:

- identifying hosts
- providing a logical location service

8.3 Deepening: Transfer Control Protocol

The Transmission Control Protocol is a core protocol of the Internet protocol suite. TCP provides reliable, ordered, and error-checked delivery of a stream of octets between applications running on hosts communicating over an IP network.

8.3.1 TCP: Three-way Handshake

To establish a connection, TCP uses a three-way handshake. After that the server open and listen at a port for connections a client may initiate to establish a connection with the three-way handshake. It is a three-step method that requires both the client and server to exchange SYN and ACK packets before actual data communication begins.

8.4 Deepening: Secure Shell

Secure Shell is a cryptographic network protocol operating at layer 7 of the OSI Model to allow:

- Remote login: gains access to a remote computer system
- Tunneling: allows a network user to access or provide a network service that the underlying network does not support or provide directly
- Forwarding TCP ports: is directing traffic from the outside world to the appropriate server

SSH uses the client-server model and the standard TCP port 22 has been assigned for contacting the server.

When we call the protocol, it does:

1. TCP handshake
2. Algorithms negotiation, key negotiation, server authentication
3. User authentication

8.5 Deepening: HyperText Transmission Protocol

The Hypertext Transfer Protocol is an application protocol(OSI layer7's protocol) for distributed, collaborative, hypermedia information systems.

Hypertext is structured text that uses logical links between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

HTTP functions as a request–response protocol in the client–server computing model and the standard TCP port 80 (occasionally port 8080) has been assigned for contacting the server.