18/05/2016 - Povo(Trento)

Firewall stateless

Team 23: Andrea Brun, Matteo Gabburo and Matteo Grossele

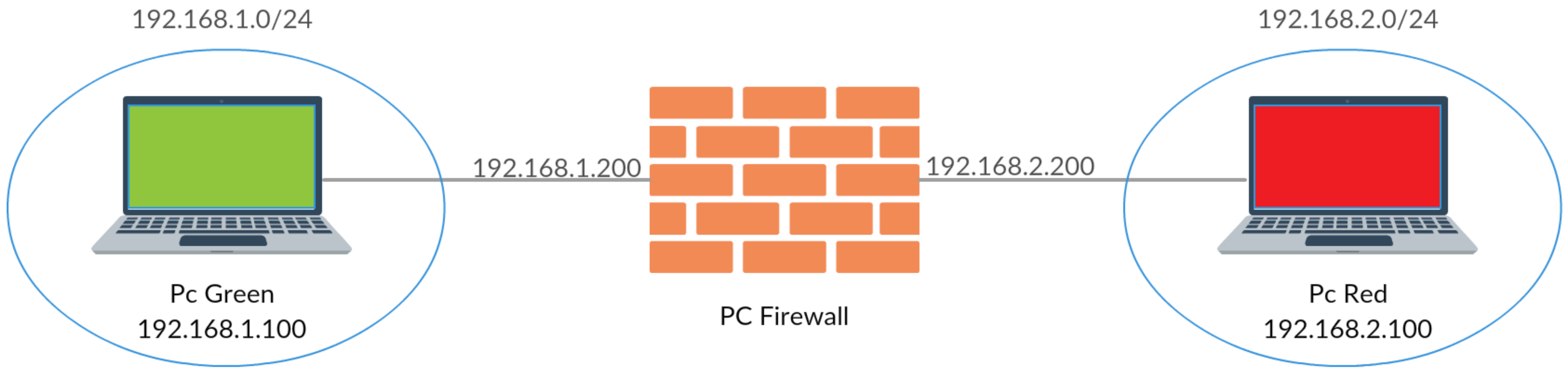
University of Trento

Network Security

Outline

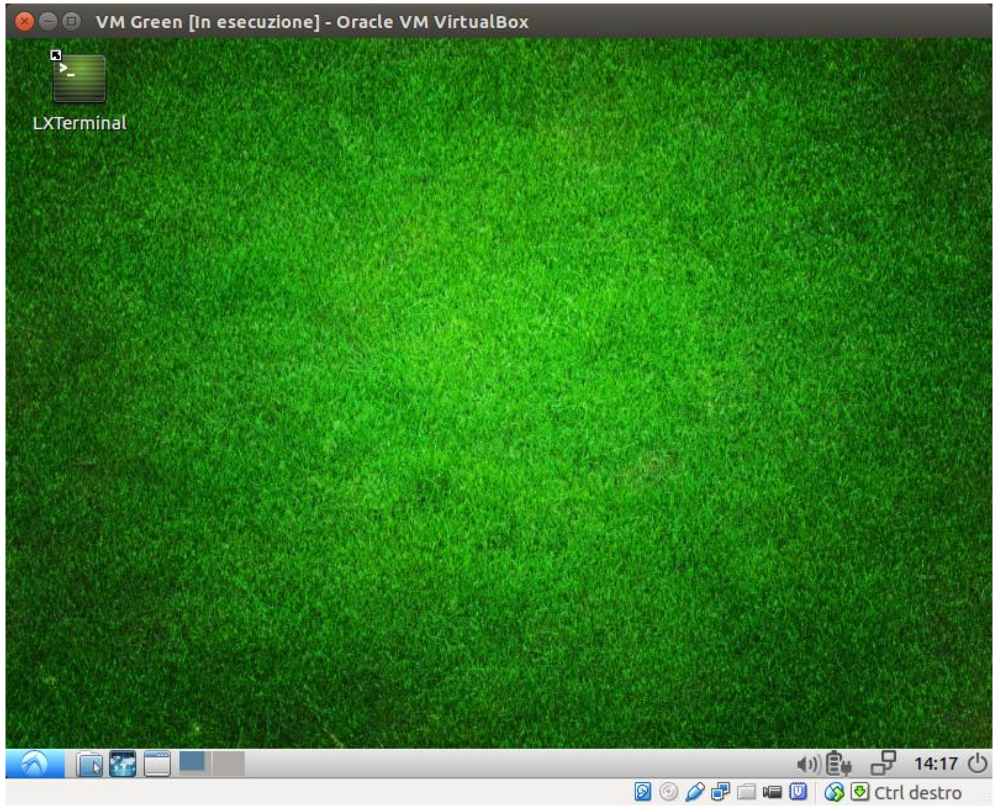
Lab introduction

Our topology



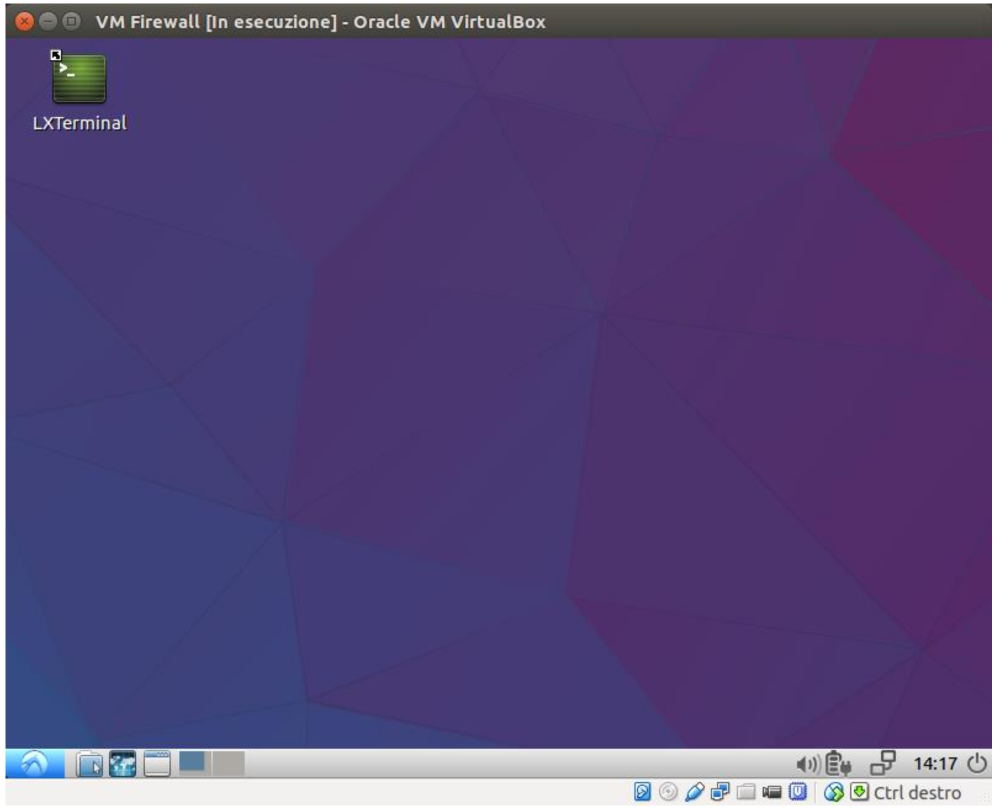
VM Green

- PC Green
  - Lubuntu 16.04
  - Host: green
  - Password: password
- Ifconfig
  - Enp0s8:
    - IPv4: 192.168.1.100
    - Network: 192.168.1.0/24



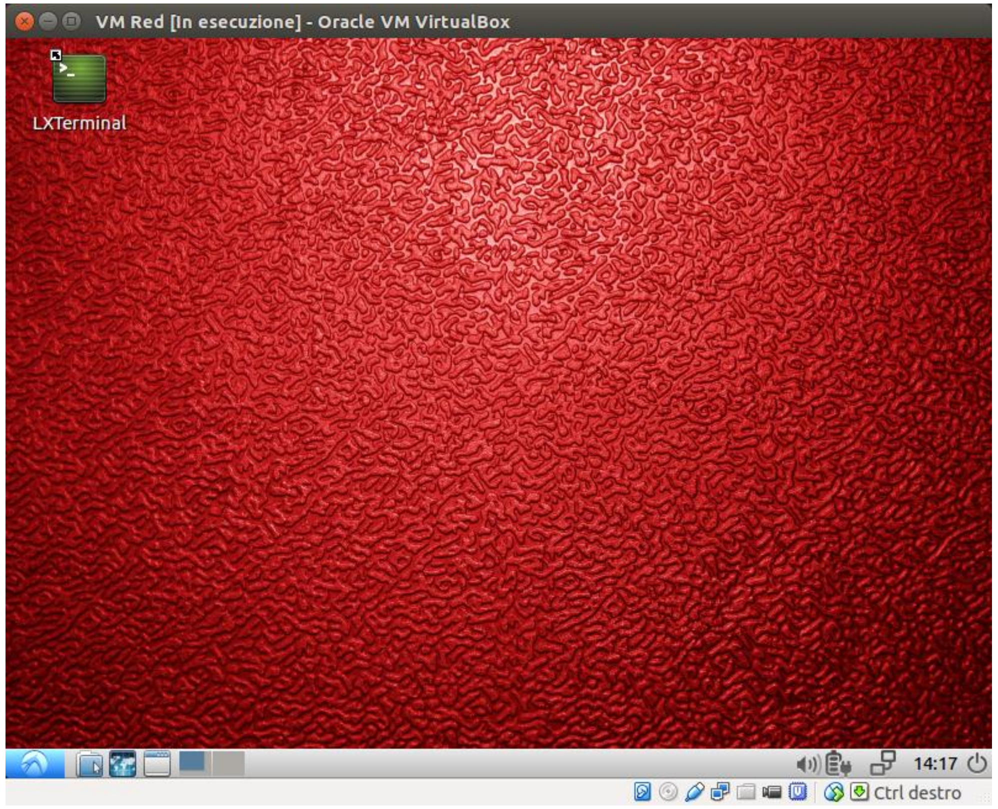
VM Firewall

- PC Firewall
  - Lubuntu 16.04
  - Host: firewall
  - Password: password
- Ifconfig
  - Enp0s8:
    - IPv4: 192.168.1.200
    - Network: 192.168.1.0/24
  - Enp0s9:
    - IPv4: 192.168.2.200
    - Network: 192.168.2.0/24
- Setup
  - As Router (forwarding)



VM Red

- PC Red
  - Lubuntu 16.04
  - Host: red
  - Password: password
- Ifconfig
  - Enp0s8:
    - IPv4: 192.168.2.100
    - Network: 192.168.2.0/24
- Service
  - HTTP
  - SSH



How does it work?

After an explanation on the theoretical concepts and about an instrument, we start with the exercises.

There are two type of exercises:

A. Make your rule (ex. 1, 2, 3): in this type of exercises you have a plain configuration for the firewall and you have to add the rules to make it block the unwanted connections

B. Modify our rule (ex. 4, 5): here you are given iptables with already some rules in it, which blocks the traffic between the two VM. Your goal becomes to find which rules has to be removed or added to make the required connections work.

Every exercise is divided into two parts:

1. Short example done by us
2. A little task for you
   i. Write your rules
   ii. Test it

Setup-script

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

The script do:

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

How to use the script (1)

Every exercise has his own script on the Desktop.

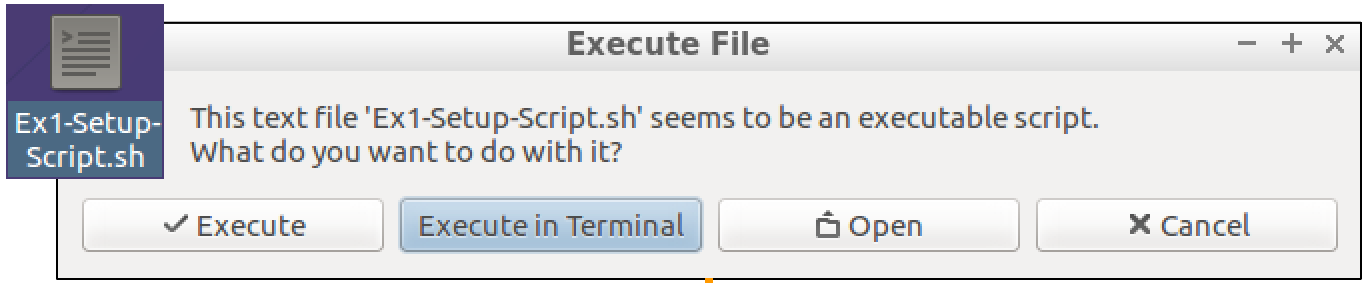
If you want to do an exercise, double click on the script and select ”Execute in Terminal”.

- 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.

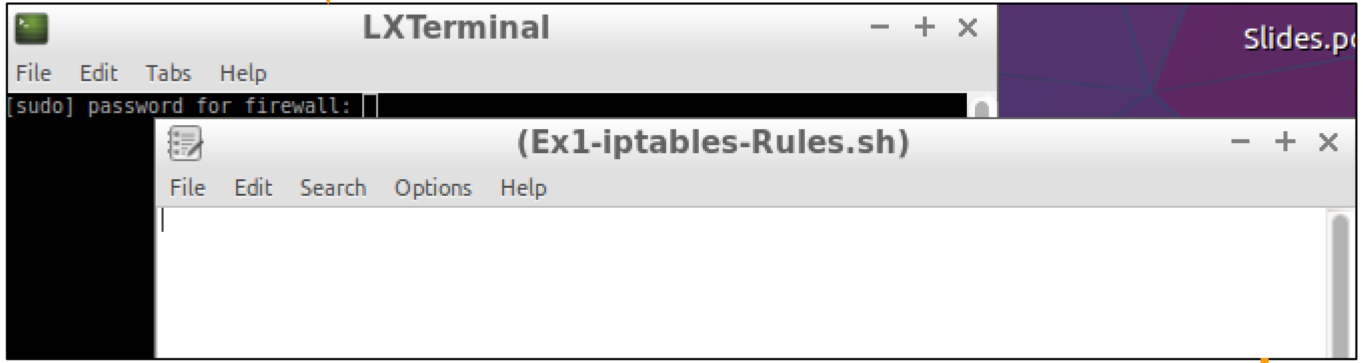
(If it says that the rules are correct, it means that they are well formed, not that they solve the exercise)

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

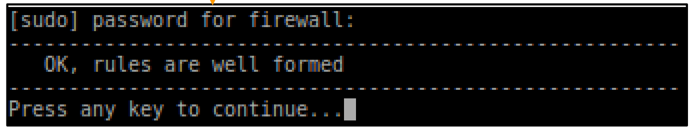
How to use the script (2)



This text file 'Ex1-Setup-Script.sh' seems to be an executable script.
What do you want to do with it?



```
[sudo] password for firewall:
```

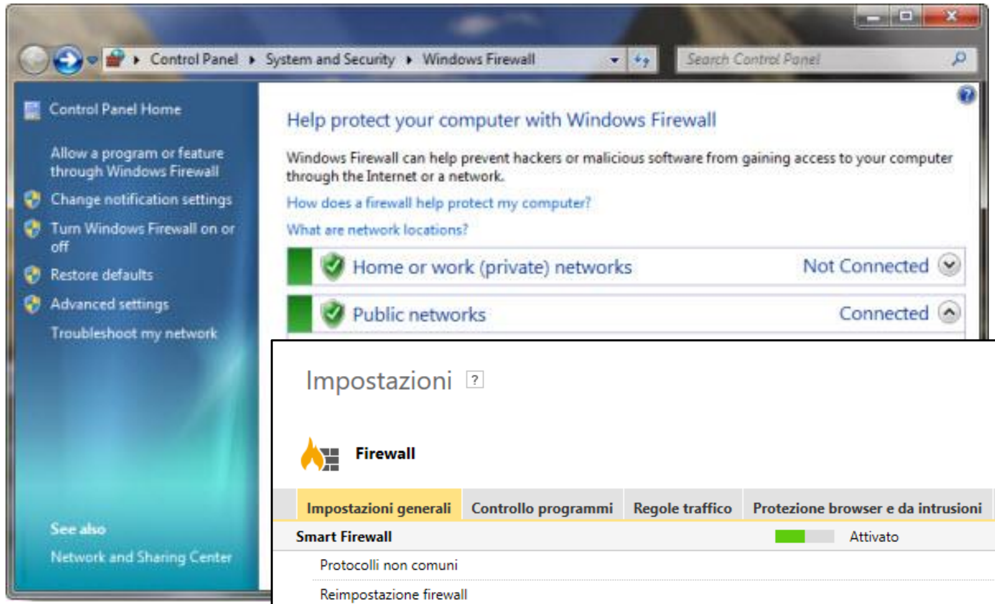


```
[sudo] password for firewall:
------------------------------------------------------------
   OK, rules are well formed
------------------------------------------------------------
Press any key to continue...
```

Firewall

Definition

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



Control Panel ▸ System and Security ▸ Windows Firewall

Search Control Panel

Control Panel Home

Allow a program or feature through Windows Firewall

Change notification settings

Turn Windows Firewall on or off

Restore defaults

Advanced settings

Troubleshoot my network

See also

Network and Sharing Center

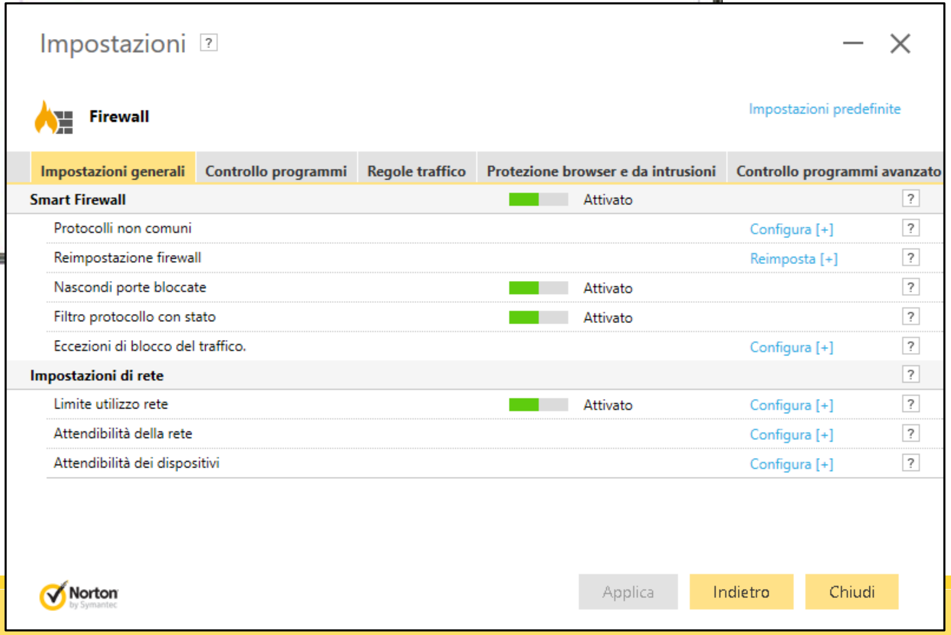
Help protect your computer with Windows Firewall

Windows Firewall can help prevent hackers or malicious software from gaining access to your computer through the Internet or a network.

How does a firewall help protect my computer?

What are network locations?

Home or work (private) networks	Not Connected
Public networks	Connected



Impostazioni

Firewall

Impostazioni predefinite

Impostazioni generali | Controllo programmi | Regole traffico | Protezione browser e da intrusioni | Controllo programmi avanzato

Smart Firewall	Attivato	
Protocolli non comuni		Configura [+]
Reimpostazione firewall		Reimposta [+]
Nascondi porte bloccate	Attivato	
Filtro protocollo con stato	Attivato	
Eccezioni di blocco del traffico.		Configura [+]
Impostazioni di rete		
Limite utilizzo rete	Attivato	Configura [+]
Attendibilità della rete		Configura [+]
Attendibilità dei dispositivi		Configura [+]

Norton

Applica | Indietro | Chiudi

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

Deepening: OSI Network Layer

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

OSI model	
Layer 7	Application
Layer 6	Presentation
Layer 5	Session
Layer 4	Transport
Layer 3	Network
Layer 2	Data Link
Layer 1	Physical

Deepening: IP

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

IP Header							
← Bits →							
0	**4**	**8**	**12**	**16**	**20**	**24**	**28**
Version	IHL	Type of Service		Total length			
Identification				Flags	Fragmentation Offset		
Time to Live		Protocol		Header Checksum			
Sources Address							
Destination Address							
Optional						Padding	

Policies

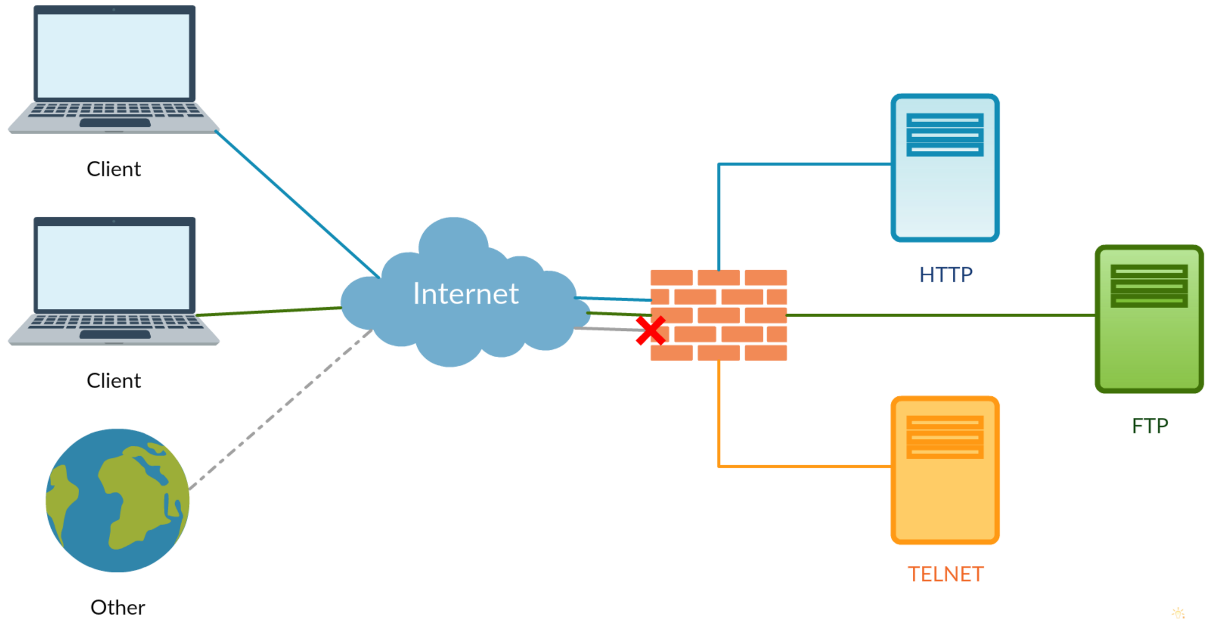
- Default deny
  - All what is not explicitly allowed is denied (list what is blocked in a Blacklist)

VS

- Default permit
  - All what is not explicitly denied is allowed (list what is allowed in a Whitelist)

Goals

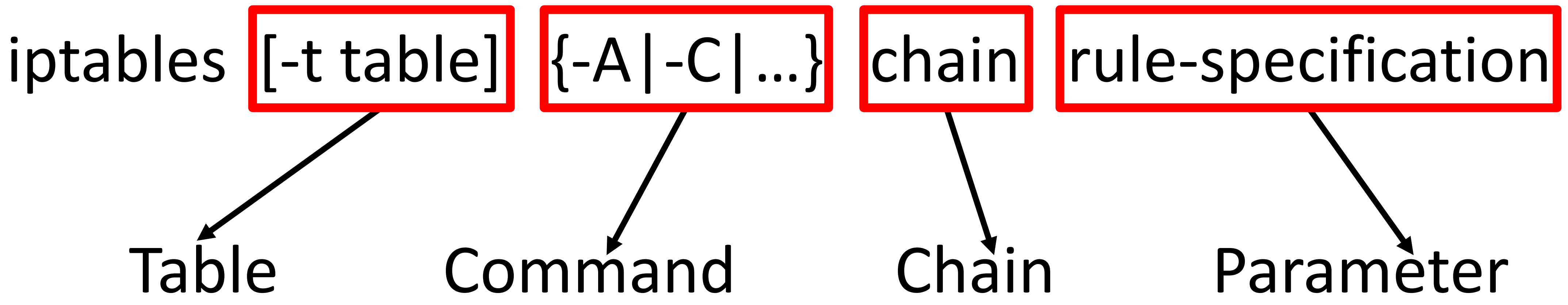
- All traffic from inside or outside must pass through the firewall (physically blocking all access to the local network except via the firewall)
    - Screen out connection from hackers, viruses, and worm
- Allow to pass only authorized traffic (defined by the local security policy)
- The firewall itself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



Iptables

Definition

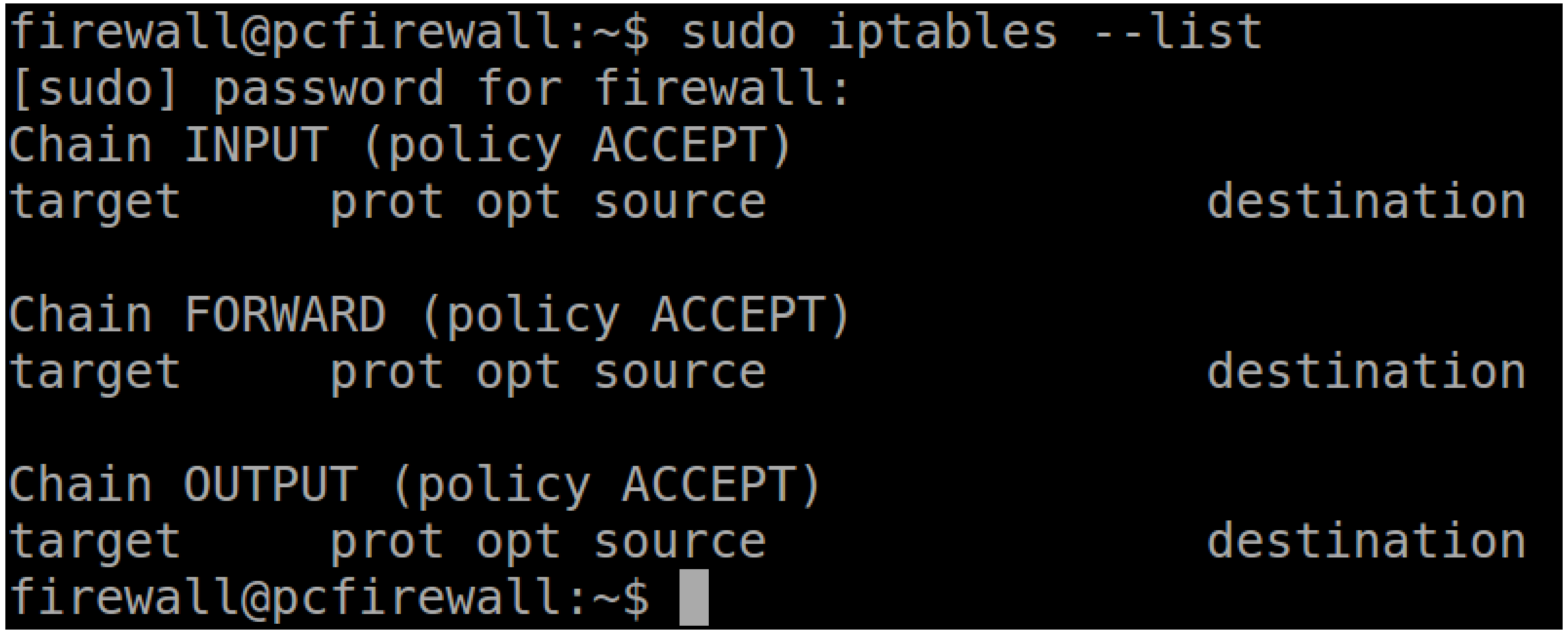
Iptables is an administration tool for IPv4 packet filtering and NAT. It is used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel.



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

Table (1) : definition

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



```
firewall@pcfirewall:~$ sudo iptables --list
[sudo] password for firewall:
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
firewall@pcfirewall:~$
```

(For our purpose we use the default table)

Table (2) : type

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

Commands

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)

Chains

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).

Parameters

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)

Targets

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

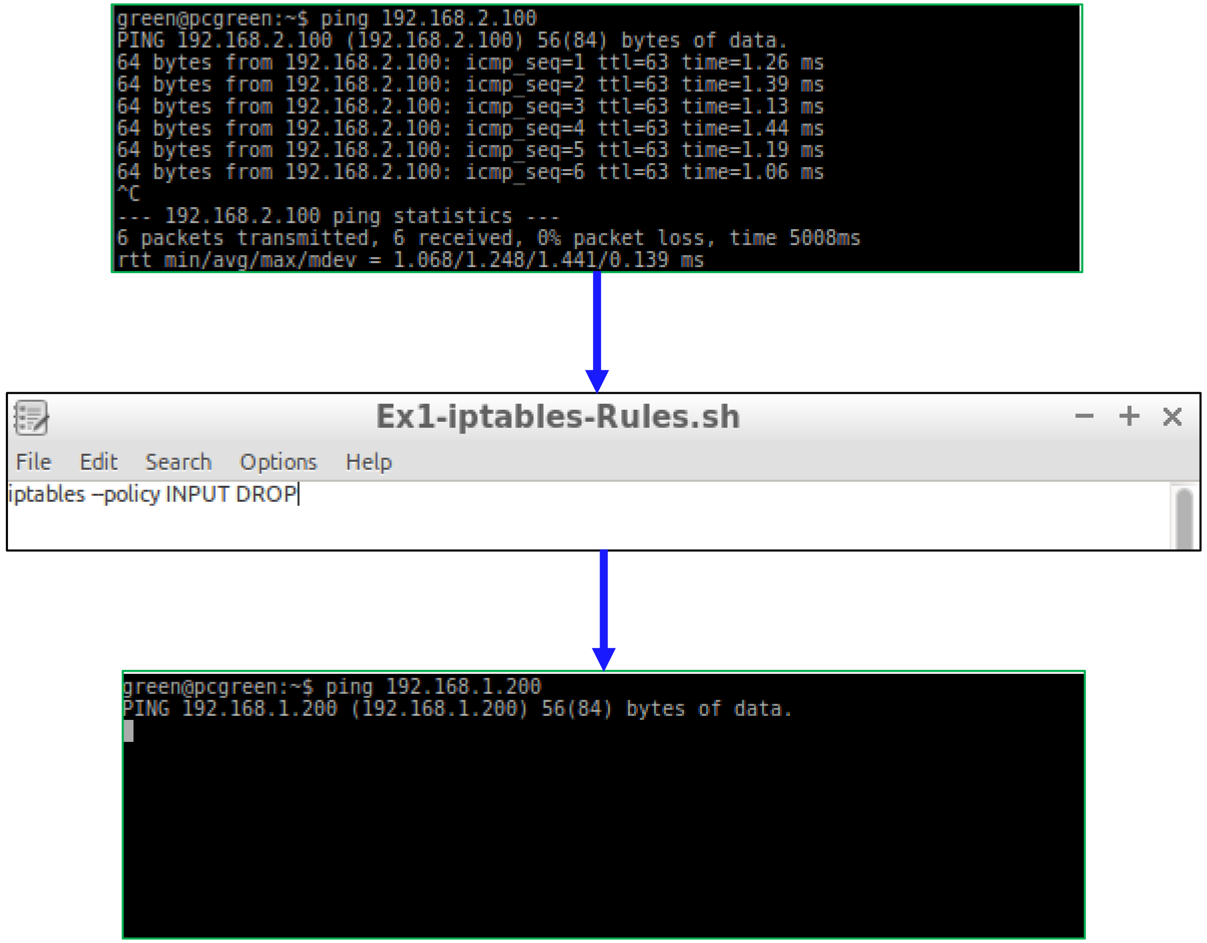
If the end of a built-in chain is reached or a rule in a built-in chain with target **RETURN** is matched, the target specified by the chain policy determines the fate of the packet.

Examples 1 : Block input

How to do

In this example we want to block every packets send from VM Green to VM Firewall.

1. Launch the script for ex1 on the Firewall
2. Go on the VM Green and open a terminal
3. Ping the VM Firewall (*ping 192.168.1.200*)
4. Go back on the VM Firewall and write the iptables rule on the file
   a. *iptables –policy INPUT DENY*
5. Try to ping again VM Firewall from VM Green



```
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=1.26 ms
64 bytes from 192.168.2.100: icmp_seq=2 ttl=63 time=1.39 ms
64 bytes from 192.168.2.100: icmp_seq=3 ttl=63 time=1.13 ms
64 bytes from 192.168.2.100: icmp_seq=4 ttl=63 time=1.44 ms
64 bytes from 192.168.2.100: icmp_seq=5 ttl=63 time=1.19 ms
64 bytes from 192.168.2.100: icmp_seq=6 ttl=63 time=1.06 ms
^C
--- 192.168.2.100 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 1.068/1.248/1.441/0.139 ms
```

```
iptables –policy INPUT DROP
```

```
green@pcgreen:~$ ping 192.168.1.200
PING 192.168.1.200 (192.168.1.200) 56(84) bytes of data.
```

Exercises 1 : Default deny

Delivery

First exercise is to set the default policy of iptables to block the traffic from VM Green to VM Red.

Remember: The order you test the configuration is important

Launch script on Firewall → Test situation before your rules → Write your Rules → Test after your rules have been applied

Examples 2 : Allow only SSH

Deepening: SSH

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.

ssh remote_username@remote_host $\longrightarrow$

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

Deepening: TCP

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.

TCP Header

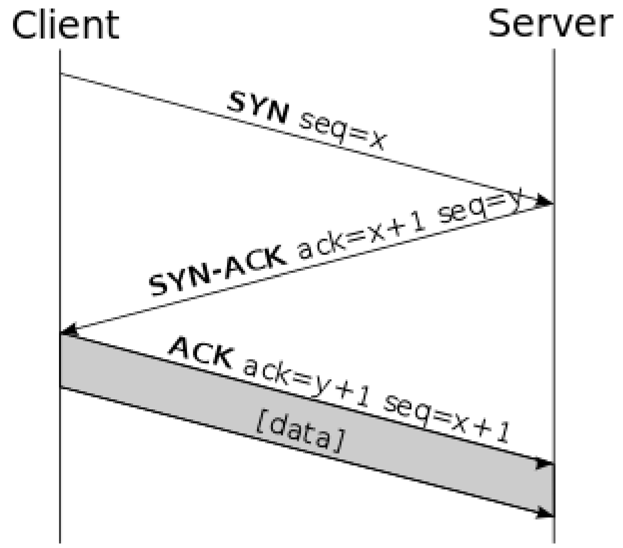
← Bits →

0	4	8	12	16	20	24	28
Source Port				Destination Port			
Sequence Number							
Acknowledgment Number							
Offset	Res.	Flags		Window Size			
Checksum				Urgent Pointer			
Optional						Padding	

Deepening: TCP Handshake

To establish a connection, TCP uses a **three-way handshake**.

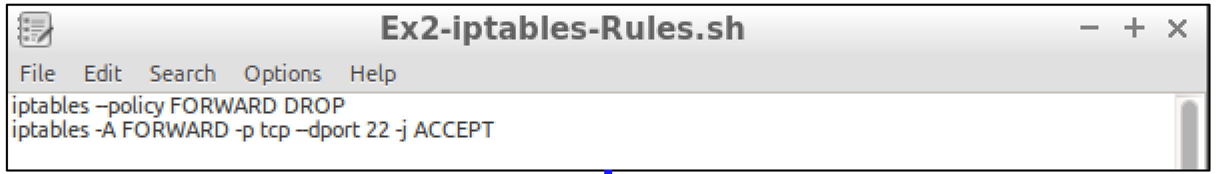
After that the server open and listen at a port for connections a client may initiate to establish a connection with the three-way handshake. It is a three-step method that requires both the client and server to exchange SYN and ACK packets before actual data communication begins.



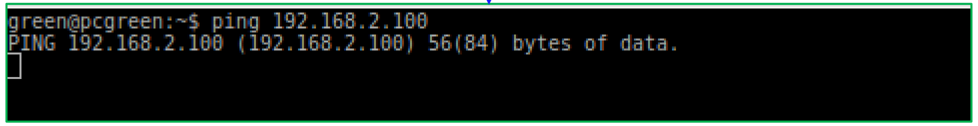
How to do (1)

We want only the ssh connections to be able to travel from VM Green and VM Red. Every other traffic must be denied.

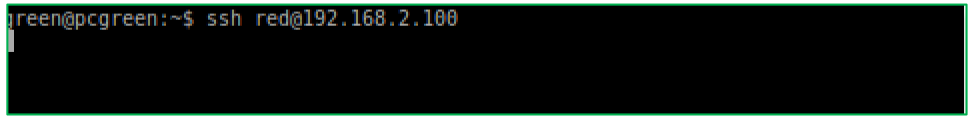
1. Launch the script for ex 2 on the Firewall
2. Go on the VM Green and open a terminal
3. Try to communicate with VM Red
   a) Ping the VM Red (*ping 192.168.2.100*) → The ping uses the protocol ICMP
   b) Connect with SSH to the user red on VM Red (*ssh red@192.168.2.100*)
4. Go back on the Firewall and write the iptables rule on the file
   a. *iptables --policy FORWARD DROP* → To set the default policy to reject everything
   b. *iptables -A FORWARD -p tcp –dport 22 -j ACCEPT*



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



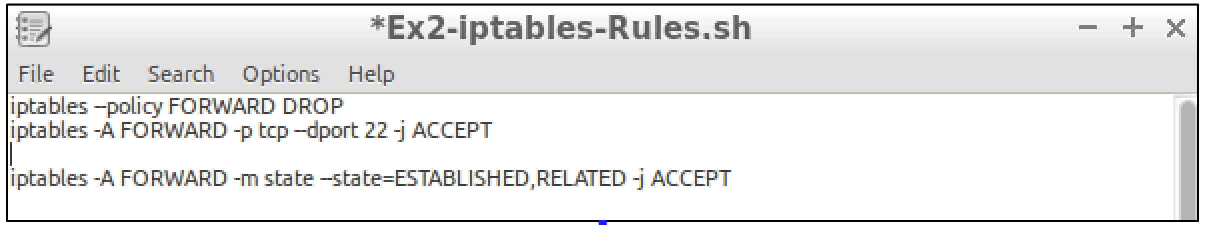
```
green@pcgreen:~$ ping 192.168.2.100
PING 192.168.2.100 (192.168.2.100) 56(84) bytes of data.
```



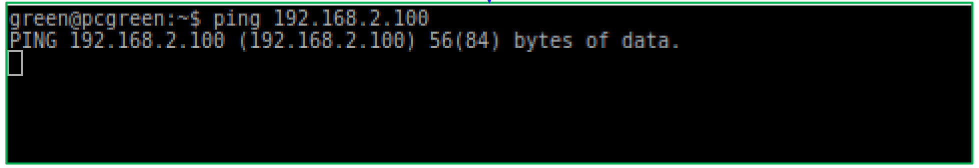
```
green@pcgreen:~$ ssh red@192.168.2.100
```

How to do (2)

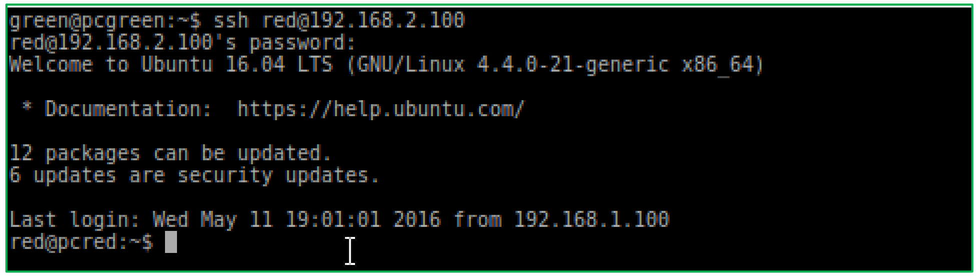
5. Test on the VM Green if you can still ping or connect with ssh on MV Red.
6. Now it is impossible to ping the VM Red, but also to connect with ssh.
7. The reason is that the firewall allows connections on port 22, but it does not allow replies coming from VM Red to the VM Green.
8. Relaunch the script for es 2 on Firewall and add this rule to allow responses
   a) *iptables -A FORWARD -m state --state=ESTABLISHED,RELATED -j ACCEPT*
9. Go back to VM Green to test if the connection to VM Red works
10. Test the same thing for VM Red to VM Green



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

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



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



```
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:~$
```

Exercises 2 : Allow only HTTP

Deepening: HTTP

The **Hypertext Transfer Protocol** is an application protocol (OSI layer 7’s protocol) for distributed, collaborative, hypermedia information systems.

Hypertext is structured text that uses logical links between nodes containing text.

$\longrightarrow$

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.

Delivery

In the second exercise we want to allow only specific protocols to pass through the firewall.

Now try on your own to allow only the HTTP protocol through the Firewall.

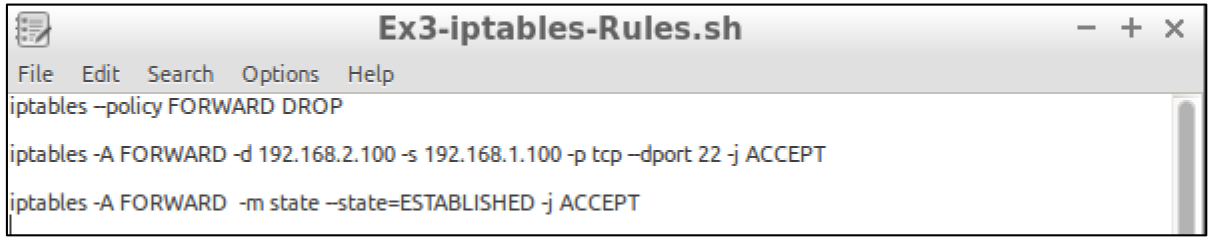
Remember: You need to allow the response packets ,else you will be able to connect, but you won't have any response

Examples 3 : Block a part of SSH

How to do (1)

Our goal is to block all the traffic from the two machines VM Red and VM Green except for the SSH traffic coming from VM Green and directed to VM Red.

1. Launch the script for ex 3 on the Firewall
2. Go on the VM Green and open a terminal
3. Connect with SSH to the user red on VM Red
   a. *ssh red@192.168.2.100*
4. Do the same from VM Red, try to connect to user green on VM Green
   a. *ssh green@192.168.1.100*
5. Go back on the Firewall and write the iptables rule on the file (see next slide)



```
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 -j ACCEPT
```

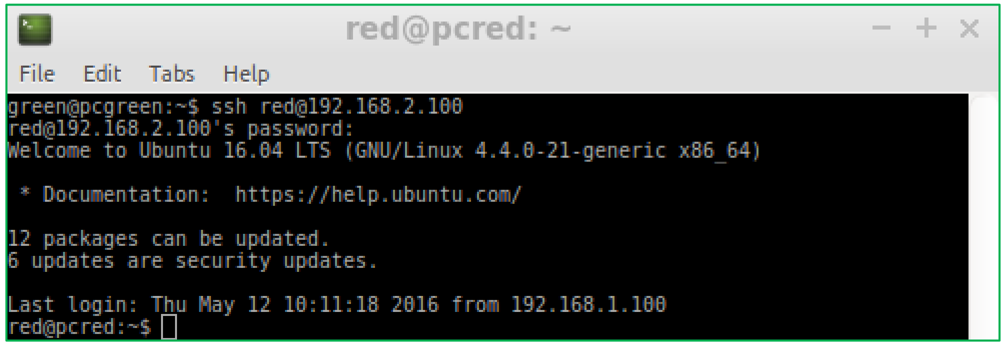
How to do (2)

   a. *iptables --policy FORWARD DROP* ⟶ To set the default policy to reject everything
   b. *iptables -A FORWARD -s 192.168.1.100 -d 192.168.2.100 -p tcp –dport 22 -j ACCEPT*

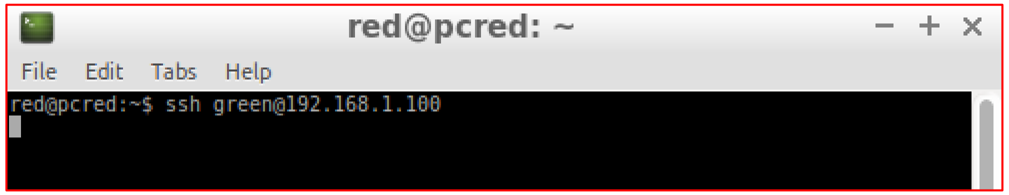
5. Add the rule to allow the responses go through the Firewall
   a. *iptables -A FORWARD -m state --state=ESTABLISHED,RELATED -j ACCEPT*

6. Test the ssh client from VM Green to VM Red and test it works
   a. *ssh red@192.168.2.100*

7. Do the same test from VM Red to VM Green and this time it does not work
   a. *ssh green@192.168.1.100*



```
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: Thu May 12 10:11:18 2016 from 192.168.1.100
red@pcred:~$ 
```



```
red@pcred:~$ ssh green@192.168.1.100
```

Exercises 3 : Block a part of HTTP

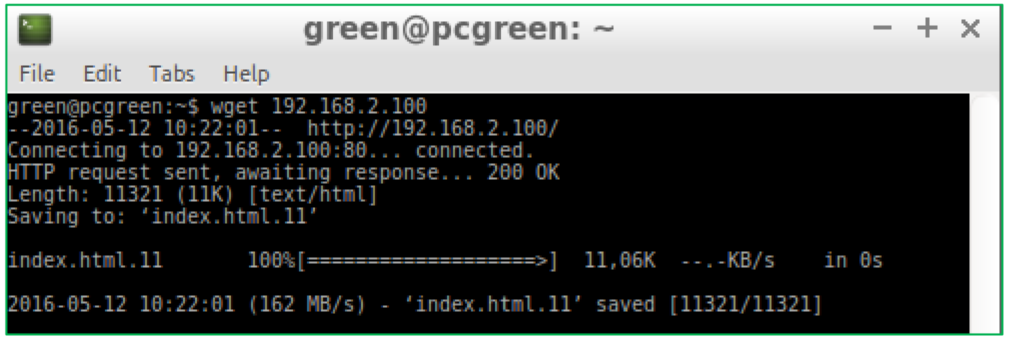
Delivery

In the third exercise we want to allow specific protocols but this time only from one direction of the Firewall.

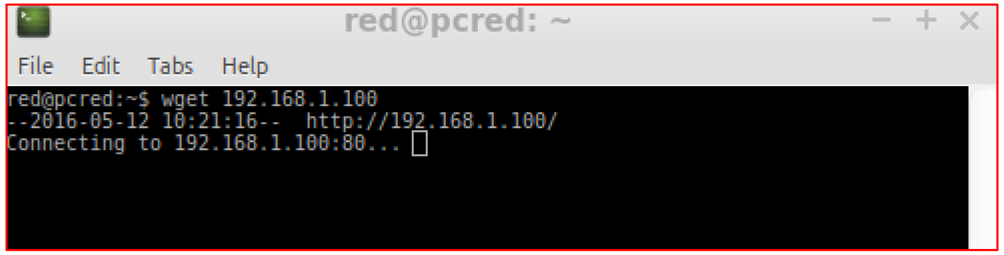
Now try to block everything except the traffic HTTP to VM Red.

Test the correct result by trying to get the web page on VM Red from VM Green (wget 192.168.2.100) from VM Green.

Notice that in the opposite case, we can not even get a response from VM Green if we try to get an HTTP response from VM Green.



```
green@pcgreen:~$ wget 192.168.2.100
--2016-05-12 10:22:01--  http://192.168.2.100/
Connecting to 192.168.2.100:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11321 (11K) [text/html]
Saving to: ‘index.html.11’

index.html.11       100%[===================>]  11,06K  --.-KB/s    in 0s

2016-05-12 10:22:01 (162 MB/s) - ‘index.html.11’ saved [11321/11321]
```



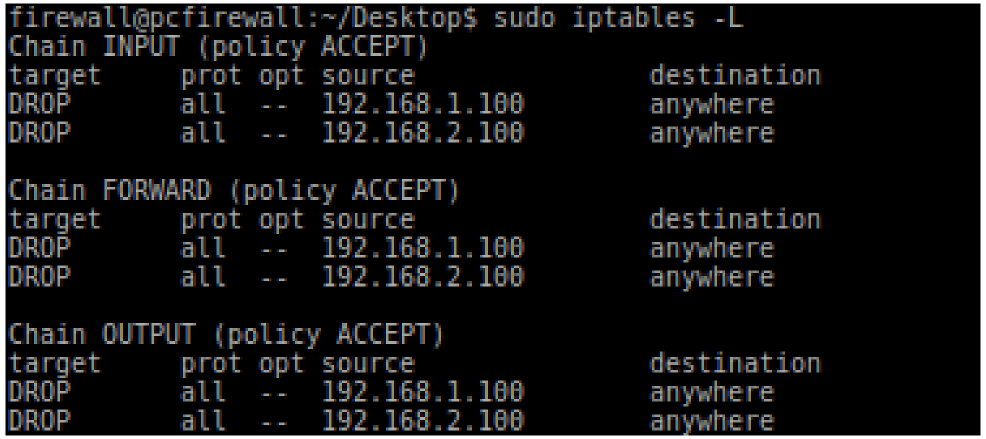
```
red@pcred:~$ wget 192.168.1.100
--2016-05-12 10:21:16--  http://192.168.1.100/
Connecting to 192.168.1.100:80...
```

Examples 4 : Allow forward connections

How to do (1)

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

1. Launch the script for ex 4 on the Firewall
2. Go on the VM Green and open a terminal
3. Ping VM Red (ping 192.168.2.100), it shouldn't work
4. Back to the Firewall and watch the iptables rules which are in the file opened previously

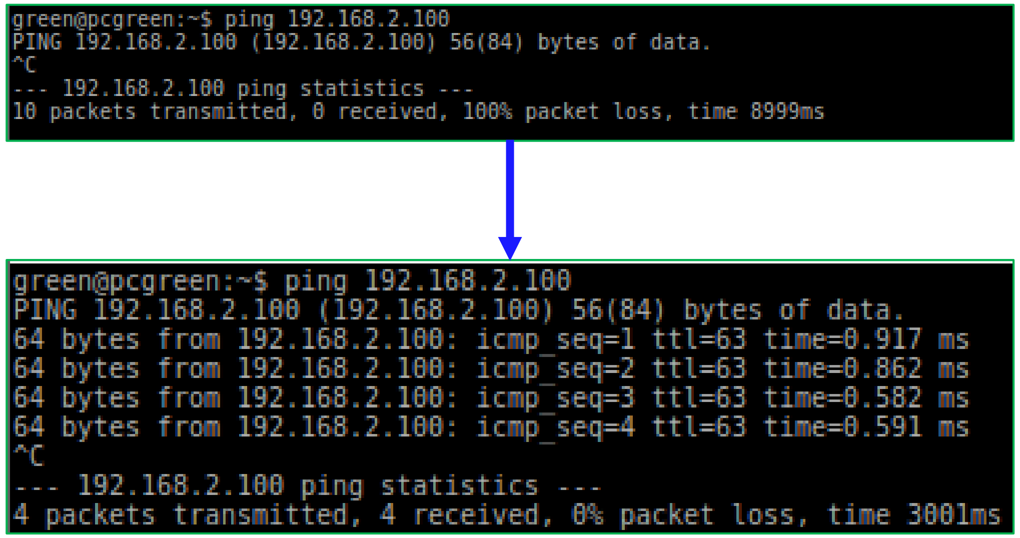


```
firewall@pcfirewall:~/Desktop$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.1.100        anywhere
DROP       all  --  192.168.2.100        anywhere

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
```

How to do (2)

5. 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
   a. iptables -I FORWARD -s 192.168.2.100 -j DROP → iptables -I FORWARD -s 192.168.2.100 -j ACCEPT
   b. iptables -I FORWARD -s 192.168.1.100 -j DROP → iptables -I FORWARD -s 192.168.1.100 -j ACCEPT

6. Try to ping again VM Rer from VM Green



```
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
```

Exercises 4 : Block forward connections

Delivery

The fourth exercise aim to understand the rules behind a firewall and change them. In this case we have a configuration who accept connections. Modify the existing rules in order to block a communication between host Green and Red

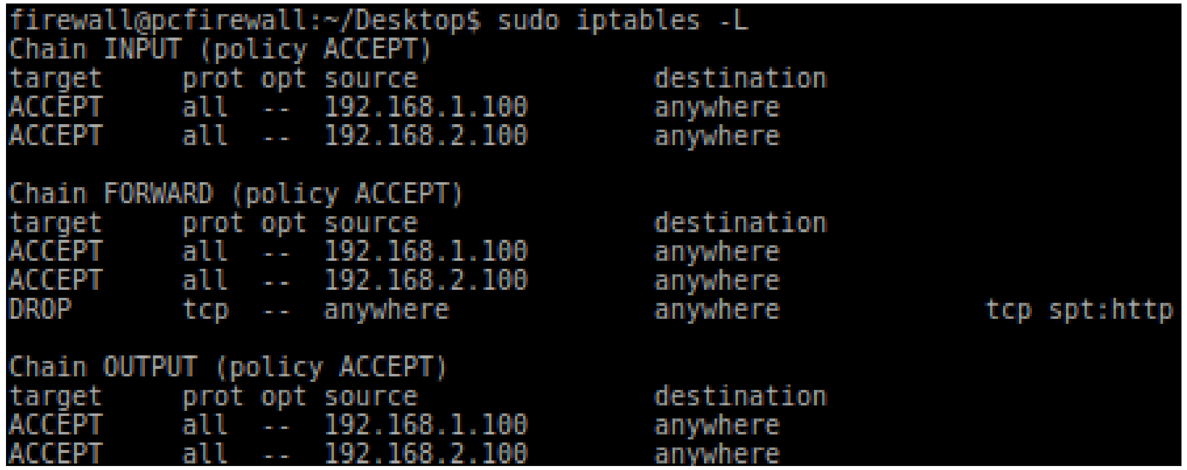
Remember : this exercise is more simple, look the example above for any doubt

Examples 5 : Block only Http

How to do (1)

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

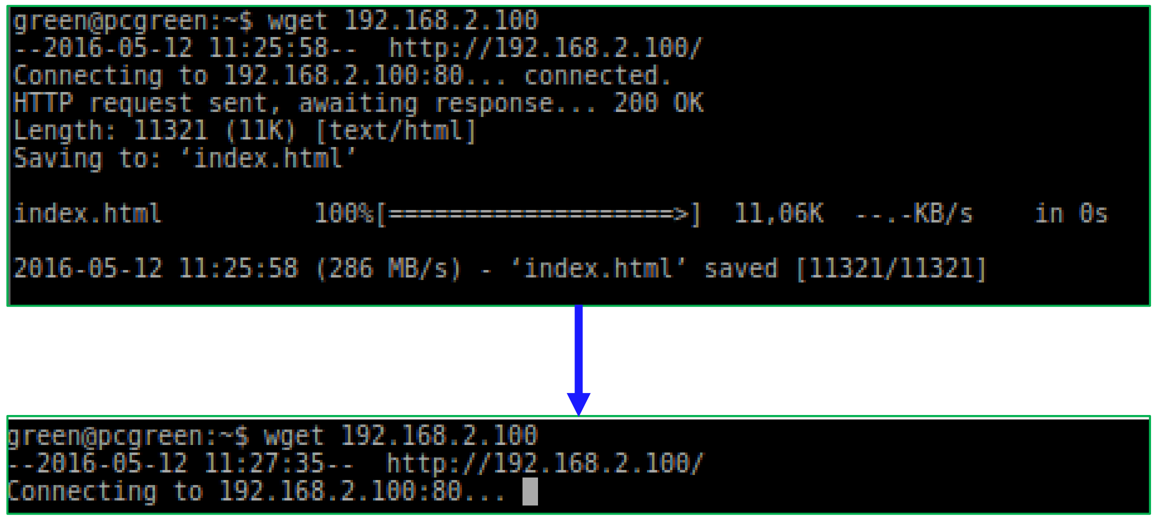
1. Launch the script for ex 5 on the Firewall
2. Go on the VM Green and open a terminal
3. Ping VM Red (ping 192.168.2.100), it should works and it is fine
4. Try to do an http request to VM Red (wget 192.168.2.100), it should work and it's not good for our purpose
5. Back to the Firewall and watch the iptables rules which are in the file opened previously



```
firewall@pcfirewall:~/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
```

How to do (2)

6. 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
   a. *iptables -I FORWARD -s 192.168.2.100 -j ACCEPT*
   b. *iptables -I FORWARD -s 192.168.1.100 -j ACCEPT*
7. Try to do an http request again from VM Green



```
green@pcgreen:~$ wget 192.168.2.100
--2016-05-12 11:25:58--  http://192.168.2.100/
Connecting to 192.168.2.100:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11321 (11K) [text/html]
Saving to: ‘index.html’

index.html          100%[===================>]  11,06K  --.-KB/s    in 0s

2016-05-12 11:25:58 (286 MB/s) - ‘index.html’ saved [11321/11321]
```

```
green@pcgreen:~$ wget 192.168.2.100
--2016-05-12 11:27:35--  http://192.168.2.100/
Connecting to 192.168.2.100:80... 
```

Exercises 5 : Allow only Http

Delivery

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 and modify it for allow only HTTP communications from Green to Red

Remember : look the example above for any doubt