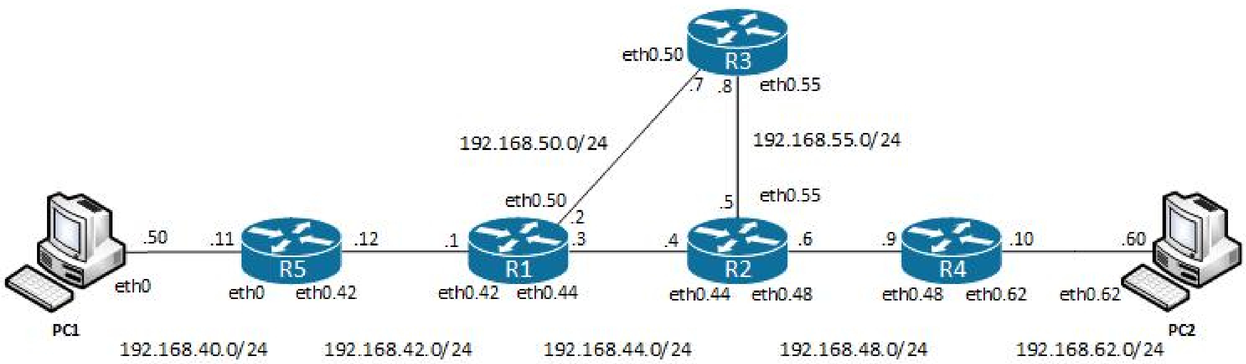
Generation and Analysing Network Attacks using Scapy

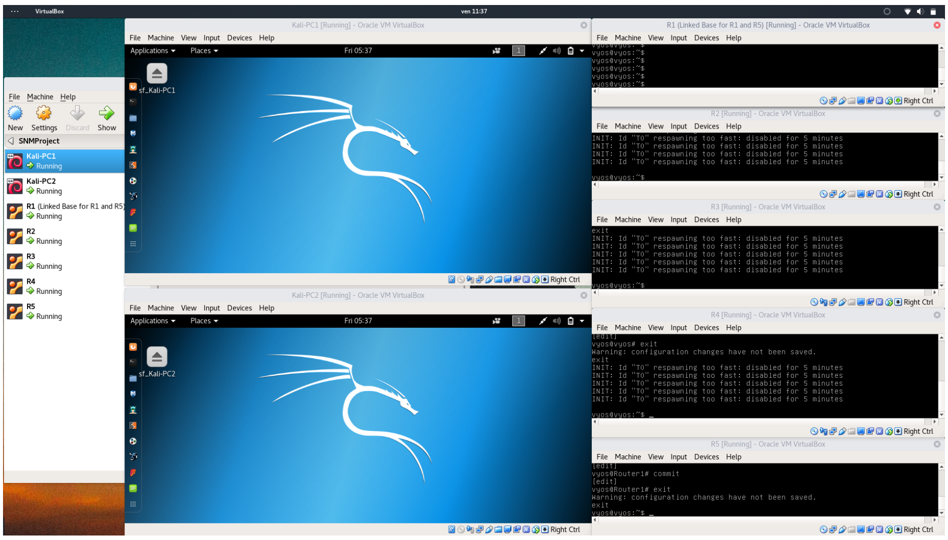
Project of the Secure Network Management course by DECAMP

Pietro Prandini - Stefano Romanello @ UniPD

December 28, 2018

1 The configuration used





1.1 Devices Configuration

R5

The Router5 is a clone of the Router1. The network of this router is composed by two enabled adapters:

- Adapter 1: Internal Network (Name: intnet);
- Adapter 2: NAT Network (Name: NatNetwork).

After the start of the machine it is setted with this commands:

```
# Configuring the router 5 (R5)
## Basic configuration
configure
load /live/image/R1/lab16
commit

## Setting the new ethernet eth0 address
delete interfaces ethernet eth0 address 192.168.40.1/24
set interfaces ethernet eth0 address 192.168.40.11/24
commit

## Setting the new ethernet eth0.42 address
delete interfaces ethernet eth0 vif 44
delete interfaces ethernet eth0 vif 50
set interfaces ethernet eth0 vif 42 address 192.168.42.12/24
commit

## Enablig RIP
set protocols rip interface eth0.42
set protocols rip interface eth0
set protocols rip network 192.168.40.0/24
set protocols rip network 192.168.42.0/24
set protocols rip redistribute connected
set protocols rip timers timeout 35
commit

exit
```

R1

```
# Configuring the router 1 (R1)
## Basic configuration
configure
load /live/image/R1/lab16
commit

## Considering the new router R5
delete interfaces ethernet eth0 address 192.168.40.1/24
commit
set interfaces ethernet eth0 vif 42 address 192.168.42.1/24
commit

## Enablig the RIP protocol
set protocols rip interface eth0.42
set protocols rip interface eth0.44
set protocols rip interface eth0.50
commit
exit
```

R2

```
# Configuring the router 2 (R2)
## Basic configuration
configure
load /live/image/R2/lab16_rip
commit
exit
```

R3

```
# Configuring the router 3 (R3)
## Basic configuration
configure
load /live/image/R3/lab16_rip
commit
exit
```

R4

```
# Configuring the router 4 (R4)
## Basic configuration
configure
load /live/image/R4/lab16_rip
commit
exit
```

Kali-PC1

```
# Contents of /etc/network/interfaces
###########################################################################
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.40.50
        netmask 255.255.255.0
        gateway 192.168.40.11
```

Kali-PC2

```
# Contents of /etc/network/interfaces
###########################################################################
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual
up ifconfig eth0 up

auto eth0.62
iface eth0.62 inet static
        address 192.168.62.60
        netmask 255.255.255.0
```

```
        gateway 192.168.62.10
        vLAN−raw−device eth0
```

1.2 Testing the configuration

Bash version of a test.

```
#!/usr/bin/env bash

# Availability of each device

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.40.11␣(R5)"
ping −c 3 192.168.40.11

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.42.12␣(R5)"
ping −c 3 192.168.42.12

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.42.1␣(R1)"
ping −c 3 192.168.42.1

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.50.2␣(R1)"
ping −c 3 192.168.50.2

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.44.3␣(R1)"
ping −c 3 192.168.44.3

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.44.4␣(R2)"
ping −c 3 192.168.44.4

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.55.5␣(R2)"
ping −c 3 192.168.55.5

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.48.6␣(R2)"
ping −c 3 192.168.48.6

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.50.7␣(R3)"
ping −c 3 192.168.50.7

echo "
----------------------------------------------------------------------------
ping␣to␣192.168.55.8␣(R3)"
ping −c 3 192.168.55.8
```

```
echo "
--------------------------------------------------------------------------------
ping to 192.168.48.9 (R4)"
ping -c 3 192.168.48.9

echo "
--------------------------------------------------------------------------------
ping to 192.168.62.10 (R4)"
ping -c 3 192.168.62.10

echo "
--------------------------------------------------------------------------------
ping to 192.168.40.50 (PC1)"
ping -c 3 192.168.40.50

echo "
--------------------------------------------------------------------------------
ping to 192.168.62.60 (PC2)"
ping -c 3 192.168.62.60
```

Now it’s presented a scapy program used to test if the network was working properly before the attacks.

```python
#! /usr/bin/env python
from scapy.all import *

def check_availability(target, label):
    print("\n--> ping to \"" + target + "\" (" + label + ")")
    ans,unans = sr(IP(dst=target)/ICMP())
    if ans:
        print(target + ' is reachable, summary:')
        ans.summary()
        return ans,unans
    else:
        print(target + ' is not reachable, summary:')
        unans.summary()
        return ans,unans

# Availability of each device
target = "192.168.40.11"
label = "R5"
check_availability(target, label)

target = "192.168.42.12"
label = "R5"
check_availability(target, label)

target = "192.168.42.1"
label = "R1"
check_availability(target, label)

target = "192.168.50.2"
label = "R1"
check_availability(target, label)

target = "192.168.44.3"
label = "R1"
check_availability(target, label)

target = "192.168.44.4"
label = "R2"
check_availability(target, label)
```

```
target = "192.168.55.5"
label = "R2"
check_availability(target, label)

target = "192.168.48.6"
label = "R2"
check_availability(target, label)

target = "192.168.50.7"
label = "R3"
check_availability(target, label)

target = "192.168.55.8"
label = "R3"
check_availability(target, label)

target = "192.168.48.9"
label = "R4"
check_availability(target, label)

target = "192.168.62.10"
label = "R4"
check_availability(target, label)

target = "192.168.40.50"
label = "PC1"
check_availability(target, label)

target = "192.168.62.60"
label = "PC2"
check_availability(target, label)
```

2 Reconnaissance Attacks

2.1 IP Spoofing

2.1.1 Introduction

In order to hide the IP address of a sender machine and at the same time identifying a station active on the network, an attacker could be use a method named *IP spoofing*.

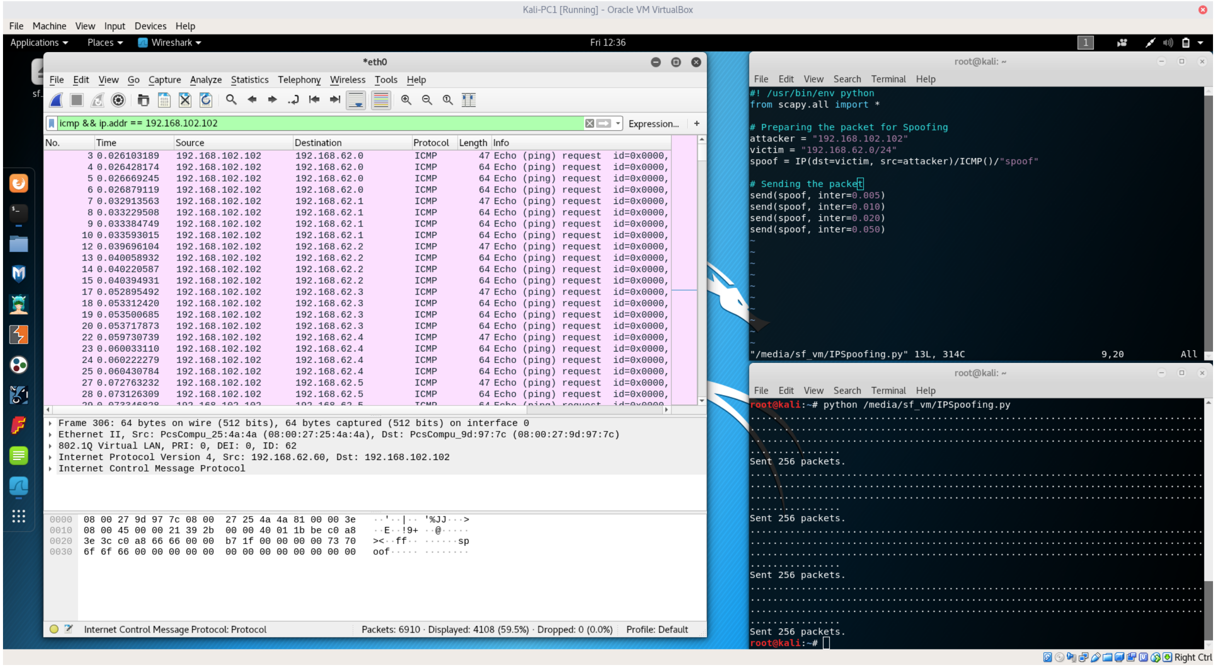
The IP spoofing consists to send ICMP packets with a fake source IP that isn't mapped on the network to recognize the reply of the active host connected to the network.

2.1.2 SCAPY program

In this scapy program the attacker has an IP that is not in the network (192.168.102.102) and it sends ICMP packets to all the host that could be present in the subnetwork 192.168.62.0/24.

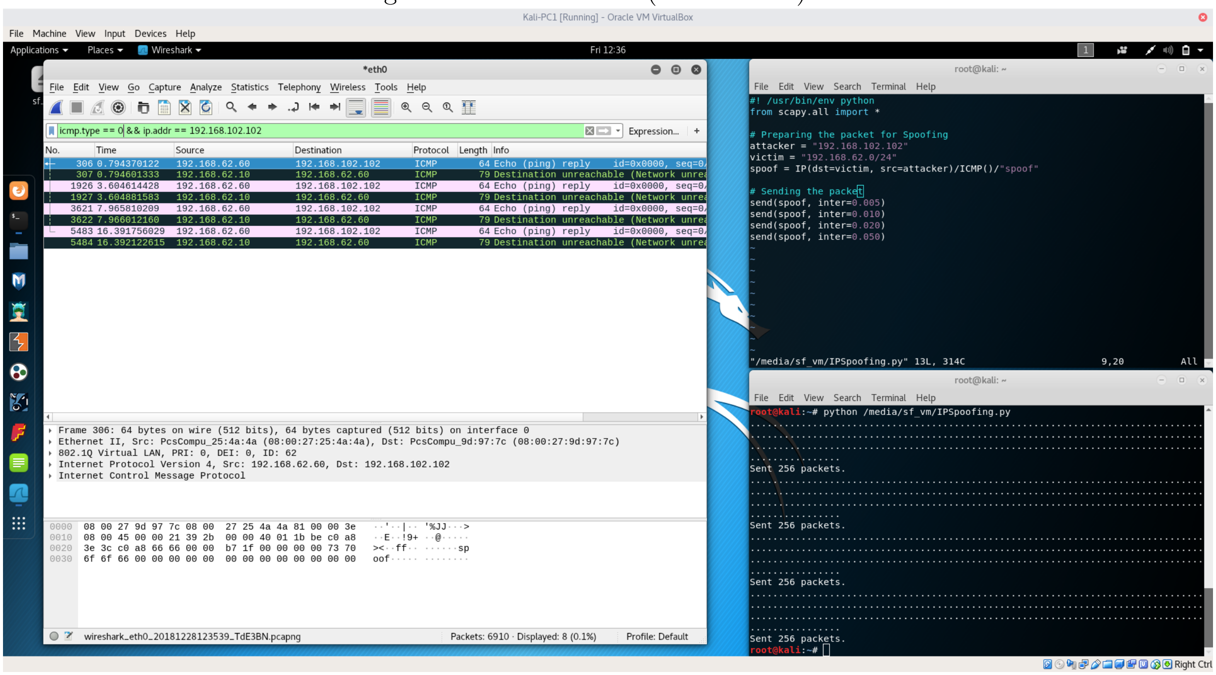
```
#! /usr/bin/env python
from scapy.all import *

# Preparing the packet for Spoofing
attacker = "192.168.102.102"
victim = "192.168.62.0/24"
spoof = IP(dst=victim, src=attacker)/ICMP()/"spoof"

# Sending the packet
send(spoof, inter=0.005)
send(spoof, inter=0.010)
send(spoof, inter=0.020)
send(spoof, inter=0.050)
```

2.1.3 Attacker's messages



2.1.4 Attack's result

Wireshark has received the ICMP reply packets of the host attacked. So the attacker had sent packets to the active host and the host hadn't recognized the real sender (the attacker).



2.1.5 How to protect the network

In order to protect the network a possible solution is to blocks all the incoming ICMP packets from the not known IP.

2.2 No Flags Set

2.3 Introduction

2.3.1 SCAPY program

```
#! /usr/bin/env python
from scapy.all import *
```

2.3.2 Attacker's messages

2.3.3 Attack's result

2.3.4 How to protect the network

3 DoS Attacks

3.1 ICMP Redirect

3.2 Introduction

3.2.1 SCAPY program

```
#! /usr/bin/env python
from scapy.all import *
```

3.2.2 Attacker's messages

3.2.3 Attack's result

3.2.4 How to protect the network

3.3 Ping of Death

3.4 Introduction

3.4.1 SCAPY program

```
#! /usr/bin/env python
from scapy.all import *
```

3.4.2 Attacker's messages

3.4.3 Attack's result

3.4.4 How to protect the network