Sniffers

1)Simple tcpdump capture of ping

Our machine's IP: 192.168.49.5
DNS IP: 192.168.49.4

Execute:
```
tcpdump -n host 192.168.49.5
ping -c 1 192.168.49.5
```

Firstly we can see how our machine is trying to find the MAC Address of the computer that owns 192.168.49.4:

```
10:59:50.693933 arp who-has 192.168.49.4 tell 192.168.49.5
```

And the answer:

```
10:59:50.694097 arp reply 192.168.49.4 is-at 00:a0:c5:7c:74:62
```

* Notice that ARP answers are cached.

Finally the ping is done:

```
10:59:50.805452 IP 192.168.49.5 > 192.168.49.4: icmp 64: echo request seq 1
10:59:50.805474 IP 192.168.49.4 > 192.168.49.5: icmp 64: echo reply seq 1
```

2)Pinging a non-existent IP

Execute:
```
tcpdump -n host 192.168.49.5
ping -c 2 -b 192.168.49.5
```

We are trying to find the MAC Address of 192.168.49.2 but no answer is received because there is not any computer with this IP:

```
11:10:49.768794 arp who-has 192.168.49.2 tell 192.168.49.5
11:10:50.767932 arp who-has 192.168.49.2 tell 192.168.49.5
11:10:51.767778 arp who-has 192.168.49.2 tell 192.168.49.5
```

3)Ethereal

Capture a ping with ethereal:

```
No.     Time        Source                Destination           Protocol Info
      1 0.000000    192.168.49.5          192.168.49.4          ICMP     Echo (ping) request

Frame 1 (98 bytes on wire, 98 bytes captured)
Ethernet II, Src: 00:90:f5:0e:95:53, Dst: 00:a0:c5:7c:74:62
Internet Protocol, Src Addr: 192.168.49.5 (192.168.49.5), Dst Addr: 192.168.49.4 (192.168.49.4)
Internet Control Message Protocol

No.     Time        Source                Destination           Protocol Info
      2 0.000201    192.168.49.4          192.168.49.5          ICMP     Echo (ping) reply

Frame 2 (98 bytes on wire, 98 bytes captured)
Ethernet II, Src: 00:a0:c5:7c:74:62, Dst: 00:90:f5:0e:95:53
Internet Protocol, Src Addr: 192.168.49.4 (192.168.49.4), Dst Addr: 192.168.49.5 (192.168.49.5)
Internet Control Message Protocol
```

* See detailed file attached for more information.

With ethereal we can get a more detailed view of the traffic. As we can see no arp packets has been sent thanks to the arp cache of each host.

4)ARP request to an real address not in cache

* Manipulation of ARP table:
    - List table:
    ```
    arp -n
    ```

    - Delete entry:
    ```
    arp -d 192.168.49.4
    ```

    - Adding a static entry that will not be replaced:
    ```
    arp -s 192.168.49.4 00:A0:C5:7C:74:62
    ```

– What is the destination MAC address of an ARP Request packet? What does this mean?

```
Target MAC address: 00:00:00:00:00:00 (00:00:00_00:00:00)
```

This is broadcast MAC address, every computer will see the request, but only the one which has 192.168.49.4 as IP will answer.

– What are the different values of the Type field in the Ethernet headers that you observed?

Type determines the content of the ethernet frame:

```
Type: ARP (0x0806)
Type: IP (0x0800)
```

– Use the captured data to discuss the process in which ARP acquires the MAC address for IP address <existing target ip address>

When a machine wants to connect to a local IP which is not in ARP cache, it is sent a broadcast packet to all the computers' LAN (ethernet level, not IP level) requesting for the MAC Address of that IP. Only the computer with that IP address will answer indicating its MAC address. At this point our computer has enough information to send packets directly to this machine.

5)ARP request for a non-real address

– Using the saved output, describe the time interval between each ARP Request packet issued by your PC. Describe the method used by ARP to determine the time between retransmissions of an unsuccessful ARP Request. Include relevant data to support your answer.

Every 1 second the ARP request is resent:

```
No.     Time        Source                Destination           Protocol Info
      1 0.000000    Clevo_0e:95:53        Broadcast             ARP      Who has 192.168.49.2?  Tell
192.168.49.5
      2 0.999049    Clevo_0e:95:53        Broadcast             ARP      Who has 192.168.49.2?  Tell
192.168.49.5
      3 1.156138    192.168.49.5          192.168.49.4          DNS      Standard query PTR
2.49.168.192.in-addr.arpa
      4 1.235141    192.168.49.4          192.168.49.5          DNS      Standard query response, No
such name
      5 1.998895    Clevo_0e:95:53        Broadcast             ARP      Who has 192.168.49.2?  Tell
```

```
192.168.49.5
```

As we can see, it also tries to get the domain name for the IP when no answer is received for its ARP requests.

– Why are ARP Request packets not transmitted (i.e. not encapsulated) as IP packets? Explain your answer.

ARP request packets are designed for a specific hardware technology: Ethernet.

6)Netstat

a. What are the network interfaces of your PC?

I have two interfaces:

eth0: Ethernet interface
lo: Loopback interface (virtual)

Output:

```
# netstat -i
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500 0     28464      0      0      0   22022      0      0      0 BMRU
lo    16436 0    294363      0      0      0  294363      0      0      0 LRU
```

b. How many IP datagrams, ICMP messages, UDP datagrams, and TCP segments has your machine transmitted and received since it was last rebooted?

```
# netstat -s eth0
Ip:
    323756 total packets received
    0 forwarded
    0 incoming packets discarded
    323309 incoming packets delivered
    317223 requests sent out
Icmp:
    19 ICMP messages received
    1 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 8
        echo requests: 3
        echo replies: 8
    18 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 15
        echo replies: 3
Tcp:
    4994 active connections openings
    4342 passive connection openings
    1 failed connection attempts
    11 connection resets received
    11 connections established
    322292 segments received
    316146 segments send out
    20 segments retransmited
    0 bad segments received.
    150 resets sent
Udp:
    991 packets received
    7 packets to unknown port received.
    0 packet receive errors
    1045 packets sent
```

c. Show your machine's routing table. What do the columns in this table mean. Explain, based on this table, how your machine determines routing behavior.

```
# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.49.0    *               255.255.255.0   U         0 0          0 eth0
default         192.168.49.4            0.0.0.0         UG        0 0          0 eth0
```

Most important fields:
- Destination is the IP that must match the packet to be sent through that route.
- Gateway is the IP where it will be routed.
- Genmask is the mask that determines the subnet.
- Iface determines which interface will be used to send the packet.

The first routing line determines how should be routed the packets targeting an IP within 192.168.49.0/24 subnet. The second line stablish a default gateway for the rest of packets (e.g. Internet Server connection).

d. Explain the role of interface lo, the loopback interface.

The loopback interface is used to give us the possibility of connecting to our own computer without going through network. 127.0.0.1 is assigned to that interface.

e. In the port table produced by "netstat -a", pick one of the connections and explain it completely. What local program is using the port? What do "listening" and "established" mean

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:swat                  *:*                     LISTEN
```

This line explains that we are listening on port 901 (swat) TCP and accept connection targeted to any of our IPs.

To see what local program is using the port we can execute “netstat -ap”:

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 *:swat                  *:*                     LISTEN     4113/inetd
```

Inetd is a daemon that manage several services in UNIX.

LISTENING means that we are ready to receive connections.
ESTABLISHED means that a connection has been done.

7)Understanding how DNS works

a) What sequence of packets are required when the network layer must go to the DNS to resolve a name?

```
      1 0.000000    192.168.49.5          192.168.49.4          DNS      Standard query A google.es
      2 0.000175    192.168.49.4          192.168.49.5          DNS      Standard query response A
216.239.39.104 A 216.239.57.104 A 216.239.59.104
```

The machine asks the DNS server for the IP of a domain and the servers sends the answer.

b) How does your machine know what IP address should be used to ask for that name?

We must indicate the DNS IP address in /etc/resolv.conf.

* Notice that GNU/Linux does not cache DNS resquest unless you have had installed the named (bind) service in cache mode.

8)Snoop passwords from an FTP session

1. Using the saved output, identify the port numbers of the FTP client and the FTP server.

```
Transmission Control Protocol, Src Port: 39127 (39127), Dst Port: ftp (21), Seq: 1, Ack: 1, Len: 0
```

FTP client uses 39127 and FTP server uses 21.

2. Identify the login name and the password, shown in plain text in the payload of the packets that you captured.

```
USER anonymous
PASS anonymous@
```

9)Snoop passwords from a Telnet session – Part 1

a) Does Telnet have the same security flaws as FTP? Support your answer using the saved output.

Yes, we can see the user/password in plain text:

```
login: mmaarrbblleemmaa .
Password:secret
```

Every letter of login name is duplicated because of remote echo.

b) Attach the saved output to your report. Justify why three packets are sent in a telnet session for each character typed on the terminal.

If we see the output we can find two packets per each character and not three, the first one corresponds to the character sent and the second one to the remote echo. When we are doing a telnet, every key received through stdin is sent and no one is showed to stdout, that is the reason why telnet server echoes our characters so we can see what we are writing.

10) Snoop passwords from a Telnet session – Part 2

a) Show the ethereal output that demonstrates that your telnet is sending the required data to the webserver.

```
GET / HTTP/1.0

HTTP/1.0 200 OK
Date: Sun, 17 Oct 2004 12:33:21 GMT
Server: Apache/1.3.20 (Unix) PHP/4.3.2-RC1
Content-Type: text/html
Age: 0
X-Cache: MISS from saturno.marblestation.homeip.net
X-Cache-Lookup: MISS from saturno.marblestation.homeip.net:3128
Connection: close

<html>
...
```

b) Describe the packets that are sent by the URV webserver back to your telnet. You don't

need to show the details of the packets, but describe the number and type of packets are sent. How many packets are text, how many pictures, etc.

The web server sends 5 packets containing the HTML of the main page. No pictures are received, Internet browsers first request the main page and after interpretation they request the pictures. We have just requested the main page so we only get HTML (text).

11) Snoop passwords from a SSH session

a) Describe what you see with respect to username/password encryption. Explain the sequence of packets that accomplish this login.

Everything is encrypted so it is impossible to get a clear username or password. Asymmetric encryption with public/private keys is used for communication.

b) Describe what you see with respect to data transfer.

Only encrypted data.

c) Given enough data and knowing the data being sent, could you crack this encryption?

Yes but it would take a very very long time.

12) Sniffing in switched environments

We can use two methods:

- Arp poisining: sending false ARP answers to ARP requests. With this method we can make other hosts believe we are for example the gateway, everybody will send us traffic and we can forward it transparently to the real gateway so sniffing everything is possible (man-in-the-middle attack). It is possible to detect just by watching if two computers answer an ARP request.
- CAM poisining: make switch think you have a MAC that you don't have. The CAM table is where switches save the MAC of the PC connected to a physical port. If we can change the CAM table, we would receive the traffic destined to other computer. Difficult to detect.

13) Detecting sniffers

We can detect sniffers using several techniques:

- Checking if the interface is in promiscuous mode. We need access to every computer.
- Suspicious DNS queries. Some sniffers try to get the domain name of every IP of each packet processed.
- Latency. We can generate a lot of traffic for hosts different than X, and then check response time of host X.
- Depending on the operating system, putting a interface in promiscuous mode makes it behaves differently. For example, accepting packets with wrong Ethernet address but right IP address.
- ARP answers from different hosts for the same ARP request.

Author: Sergio Blanco Cuaresma