Universidad
Carlos III de Madrid
Security Group *– Dept. of CS*

Universidad Carlos III de Madrid

Network analysis

Security Engineering
Fall 2011

Important note:
*During this practical you must **not** introduce any actual username and password unless otherwise stated. In such a case, the instructor will provide you with appropriate credentials.*

Introduction

A **network protocol analyser** (a.k.a. packet inspectors or simply *sniffers*) is a tool to capture and inspect network traffic over a network link. Actually, most sniffers allow the user to inspect traffic other than that generated at (or destined to) the host where the tool is running. This is possible by setting the network interface in the so-called *promiscuous* mode, which instructs the card to capture all traffic rather than just those packets with destination the host. In order to set the card in promiscuous mode, the user must have superuser privileges.

In this practical the student will learn the basic functioning of two network analysers: Tcpdump and Wireshark. Whilst the former works in the command line, the latter offers a graphical interface. However, both contain essentially the same functionalities.

As many other security tools, network analysers play an important role both for an attacker and for the security administrator. From a defence perspective, a network analyzer helps to inspect traffic traversing the network. This could be extremely useful to, for example, **detect the presence of "strange" packets** or analyse the correct functioning of existing network applications. However, it could also be a valuable tool for an attacker interested in capuring the traffic exchanged between two or more parties. Regarding this, it is important to remark that this constitutes a **passive attack** since the attacker only listens to the communication but does not participate actively. Nevertheless, analysing network activities is often a starting point for a more sophisticate attack.

Depending on the network topology and the probe location, a sniffer will have access to different network flows:

- In a wired LAN using a HUB, the sniffer will have access to *all* traffic traversing the link where the sniffer is located.
- In a wired LAN using a SWITCH, the sniffer will only access the traffic originated and received at the host where the sniffer is running.

Next we give a brief introduction to Tcpdump and Wireshark. The student must check the manuals too.

Wireshark

Open a terminal and execute:

`$ xhost +` (this have to be done only the first time we execute Wireshark)

`$ sudo /usr/bin/ethereal`

Once the program is opened, we can start to capture packets through the menu: Capture → Start.

Keep in mind that the option "capture in promiscuous mode" must be activated. It could also be beneficial to activate the option "update packets in real time," as otherwise you could not visualise captured traffic until the process is finished.

Capturing packets in promiscuous mode will likely produce a huge amount of data. In fact, in a large network this could eventually collapse the machine unless a purposely designed storage solution is in place. In order to avoid this, Wireshark allows to define *filters* for two different puposes. *Capture filters* specify what traffic is cpatured, while *visualisation filters* help the analyst by specifying what packets are shown. The syntax for both filters is different. Next we provide a brief overview:

Capture filters (these are used with Tcpdump as well):

They are formed by a number of primitive expressions connected by logical operators:

`[not] primitive [and|or][not] primitive,` `where primitives are:`
`[src|dst] host <host>`
`[tcp|udp] [src|dst] port <port>`
`[less|greater] <length>`
`[ip|ether] proto <proto>`
`[ether|ip] [broadcast|multicast]`
For instance: `tcp port 23 and not host 10.0.0.5`
Note: These filters must be defined before the capture starts (Edit -> Capture filters)

Visualisation filters

They are formed by a number of comparisons (see Table 3) between expressions connected by logical operators (see Table 2).

Fields	Contents	Expression
tcp	Source port	tcp.srcport
tcp	Receiver port	tcp.dstport
IP	IP address (src or dst)	ip.addr
IP	IP destination	ip.dst
IP	IP source	ip.src
IP	Time-To-Live	ip.ttl
IP	**Protocol**	**Ip.proto**

Table 2. Comparison operators:

Operator	Equivalent in C
eq	**==**
ne	**!=**
gt	**>**
lt	**<**
ge	**>=**
le	**<=**

Table 3: Logical operators

Operator	Equivalent in C
And	**&&**
Or	**ll**
Not	**!**
Xor	**^^**

Tcpdump

Open a terminal and execute:

```
$ sudo /usr/sbin/tcpdump -w /tmp/capture
```

Note: Option –w sets the file where captured traffic will be written.

As in the case of Wireshark, Tcpdump admits capture filters. The syntax is identical. Once traffic is captured into a file, we have two options to read the packets:

1. Show the contents on the command line:

```
$ sudo /usr/sbin/tcpdump -X -r /tmp/capture
```

Note: Option -X visualises each packet's contents. Option -r reads the capture file specified.

2. Read the capture through Wireshark. Find out how to do this, as it will be useful for the remaining of the practical. (Hint: Check Tcpdump manual!)

Exercise 1. Analysis of a LAN

1.1. Discovering the topology of the Lab network.

- To get an idea of how the topology looks like, we run Wireshark and capture as much traffic as we can (promiscuous mode).
- Now repeat the process but with a capture filter such that we only capture packets originated at or destined to our host. Can you spot any difference? What's the topology of the computers connected in the lab?

Capturing passwords

One of the most used applications of sniffers is capturing passwords. We will check the security of several email servers offering a web interface and also some application layer protocols, such as ftp, ssh, etc.

Webmail services

- Run Wireshark and start a traffic capture. Use an appropriate filter to facilitate the process. Connect to http://correo.orange.es/ and insert some fictitious username and password. Stop the capture and try to locate the password.

  Hint: Use option "Follow TCP stream" to observe the exchange of commands between client and server.

- Repeat the experiment using http://www.gmail.com. Is is the same as above?

- Now repeat the process sending an email from a hotmail account. Can you spot any piece of cleartext information?

Application layer confidentiality

- Repeat the process using an FTP server (e.g., ftp://ftp.rediris.es) and an SSH server (e.g., guernika.lab.inf.uc3m.es). See what happens in both cases.

Exercise 2. Remote network analysis

In this exercise the student will connect to some other student's computer to run a sniffer (tcpdump) from there. The captured traffic will be sent to his/her machine to be analysed with Wireshark.

- Connect to the other machine using ssh:

```
$ ssh -p 22 <IP_address>
```

- Run Tcpdump and store the capture in a file. To facilitate the process, ask the user in the target machine to connect to a vulnerable webmail server. Stop the capture and send the capture file back with:

```
scp –P 22 <user>@<IP_address>:<capture_file>
<local_capture_file>)
```

- Open the capture with Wireshark and inspect its contents. Can you spot anything useful? Pay attention to the IP addresses: packets are destined to the machine where the sniffer was running.

Part II: Remote network analysis with ARP Poisoning.

IMPORTANT NOTE: This part of the practical requires 2 computers, so you must work in pairs. No more than 2 simultaneous groups should do it. The exercise will ALWAYS be done under the supervision of the instructor.

One possibility to inspect traffic from a remote machine in a switched network is to use a technique known as ARP poisoning, which creates a man-in-the-middle attack. **Before starting the exercise, proceed first to the appendix at the end of this script and read carefully what ARP is and how it works the exercise.**

- Execute in both computers (A and B) the following command to have a look at their ARP caches:

```
$ /usr/sbin/arp –a
```

- We will poison B's cache from A. Thus, B we will believe that A's MAC address is the router's MAC address:

```
$ sudo /usr/sbin/arpspoof -t IP_B IP_router
```

- Run Wireshark from A and observe the "ARP reply storm". Analyse how the attack works.
- Check again the ARP caches in both computers:

```
$ /usr/sbin/arp –a
```

- From B, connect to http://correo.orange.es/ and repeat exercise 1. Can you capture traffic originated at (or destined to) B from A? Can you obtain the username/password?
- Now we will carry out a Denial-of-Service attack against B and the URL http://www.uc3m.es/

```
$ sudo /usr/sbin/tcpkill host www.uc3m.es
```

- Try to connect to http://www.uc3m.es/ from B.

Appendix: ARP SPOOFING – Spying on switched networks

At first sight, using a switch might look like a good solution to avoid the threat posed by sniffers. However, a major vulnerability in ARP can be exploited to bypass this. Next we discuss one technique used to sniff traffic on a switched network: ARPSPOOFING.

Ethernet networks

Ethernet was designed around a very simple idea: all the hosts in a local area network share the same transmission medium (the wire). As a consequence, all network traffic is accessible to every host. In order to avoid this, Ethernet cards contain a basic filtering mechanism that ignores frames not destined to it. This is implemented by comparing each frame's destination MAC address with the card's. Such a filtering is ignored when the card is put in **promiscuous mode**, so all the traffic becomes visible.

To an extent, switched networks avoid this problem by splitting the network into a number of different segments. The switch will route each frame to the appropriate segment. Thus, the only traffic that will circulate through the wire is that originated at or destined to the host (plus broadcast).

What's a MAC address?

Each network interface (card) contains a unique network identifier known as its MAC address. Such an address is used to identify who is sending a frame and to whom it is destined. In the case of Ethernet networks, the MAC address is a 48-bit number generally written down in hexadecimal (12 hex symbols). The first 24 bits identify the manufacturer, while the remaining 24 bits are a unique serial number assigned to the card. Consequently, there are not two Ethernet cards with the same MAC address. Having two interfaces with the same MAC address will cause some troubles.

If we want to know the MAC address(es) of the network interfaces in our machine, run `ifconfig –a` in Linux. The output will look like this:

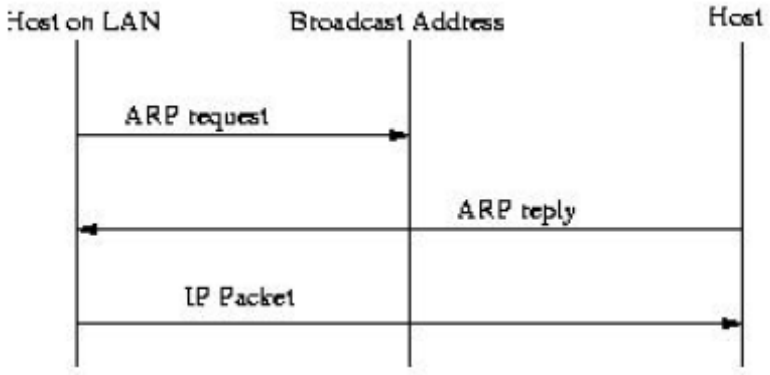
```
eth0 Link encap:Ethernet HWaddr 00:C0:4F:68:BA:50 inet
addr:192.168.0.1 Bcast:192.168.0.255 Mask:255.255.255.0 UP BROADCAST
RUNNING MULTICAST MTU:1500 Metric:1 RX packets:31658 errors:0
dropped:0 overruns:0 frame:0 TX packets:20940 errors:0 dropped:0
overruns:0 carrier:0 collisions:0 txqueuelen:100 Interrupt:19 Base
address:0xdc00
```

here you can see the MAC address is 00:C0:4F:68:BA:50.

If we are interested in knowing the MAC address of any other machine in the network, we will query our local ARP cache (`arp –a`). The output shows the IP and MAC addresses currently stored in the cache. (If you want to know the MAC address of some other machine you have not communicated with before, ping it first to force ARP to learn its MAC). Everytime we transmit data destined to some other machine, we need to know its MAC address. If it is not in the cache, the host sends an "ARP Request" command to the broadcast address, requesting the MAC address of the host whose IP is specified. The target host will reply with an "ARP Reply" message containing its MAC address. The address will be stored, generally for a few minutes, in the ARP cache. Thus, any future attempt to communicate with the host will check first the cache and retrieve the MAC address before launching another ARP request.

ARP

The basic function of ARP (Address Resolution Protocol) is to translate IP addresses to MAC addresses. The inverse translation is done by RARP (Reverse ARP). When a machine A needs to know the MAC address of another machine B, it sends an ARP Request message containing B's IP address to the broadcast address (FF:FF:FF:FF:FF:FF) of the network segment. The process is illustrated by the next figure:



In order to reduce the amount of traffic generated by ARP requests and replies, every ARP reply message is captured by all the hosts in the network segment, who store (or update) their ARP caches. This function plays a central role in the ARP spoofing attack.

ARP spoofing and poisoning
This method allows us to capture traffic between two machines A and B. This is done by "poisoning" the ARP caches of the two machines, making A (resp. B) believe that B's (resp. A's) MAC address is our MAC address. Thus, all the traffic sent by A and B will come to us. We will subsequently reroute it to the appropriate destination, so the entire process is transparent to them. Note that this process does not require us to put the interface in promiscuous mode. A potential way to discover if a man-in-the-middle attack is under way is to check our local ARP cache and see if it contains two machines with the same MAC addres..

The attack is simple: From our machine, we send **fake ARP reply** packets to both hosts. These replies inform host A that B's MAC address is our MAC address. Thus all subsequent traffic sent by A to B will actually be received by our machine (the switch will do this for us).

Suppose the following situation:

We send a constant flow of ARP reply packets (this is to avoid caches being refreshed with true information) to hosts A and B with the following contents:
HOST A: ARP reply informing that 192.168.0.1 has MAC address 03:03:03:03:03:03
HOST B: ARP reply informing that 192.168.0.2 has MAC address 03:03:03:03:03:03
From now on, all packets coming from A and B will be routed to us. Once captured, we have to reroute them to the intended recipient, depending on who the sender is:
Packets coming from Host A → forward to 02:02:02:02:02:02
Packets coming from Host B → forward to 01:01:01:01:01:01

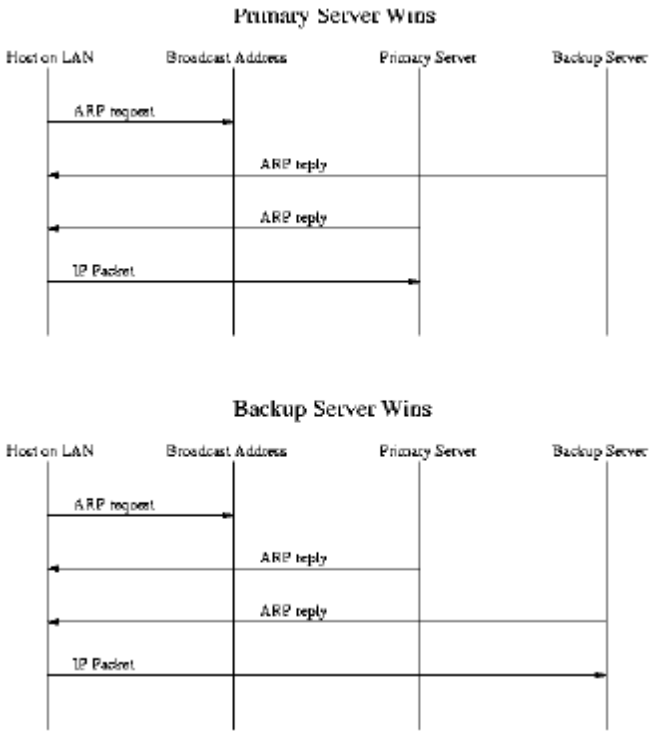
There are several freely available programs to carry out an ARP spoofing attack, such as Arptool, Arp Fun, ettercap, etc. Ettercap can be executed either in command mode or through a GUI. The latter shows a listing with all the hosts found in the LAN. This is done by sending ARP requests to every possible IP, which are obtained from our IP and the netmask. These should be done carefully as, for example, if we operate in a network with a class B mask (255.255.0.0), the program will attempt 255*255=65025 ARP Request messages. Considering that each one will take approximately 1ms, this will take some time.

ARP vulnerabilities can be exploited for uses other than ARP poisoning. For example, if we impersonate the network gateway, every communication with the ouside world will pass through us. Thus, if we choose to drop certain flows rather than forwarding them to the actual gateway, the affected host(s) will no be able to get out of the network. Furthermore, some switches can be actively manipulated through ARP packets to set them from the usual "bridging" mode into "repetition" mode (i.e., to forward every incoming frame through all ports). This can be achieved by flooding the switch's ARP table with a huge amount of fake MAC addresses. Thus, when the switch receives a packet destined to a MAC address that is not contained in its cache, it forwards it through all the ports

ARP Spoofing and redundant servers

ARP spoofing can be useful in usual network operations too. For example, if a server stops working and we need an auxiliary machine to take over its identity, this can be easily done by setting up an IP alias and using ARP spoofing. This is by no means an orthodox solution, but could prove handy if our budget is very limited but nonetheless we must ensure the continuity of various services (HTTP, FTP, SMTP, POP, …)

Assume a main server with two network interfaces (the second one will give us access to the server when the backup is working) and a backup server with two network interfaces too (or else just one and an IP alias to give it two IP addresses). We must configure the backup server in such a way that when it starts working (e.g., because the main server is down) it will use the IP address of the main server (either configuring the second network interface, or setting up an alias on the first one). Next, the backup server will use ARP spoofing to ensure that all packets destined to the main server will be sent to it. The sending of ARP replies must be carried out continuously until the main server is back, thus avoiding that the ARP caches will refresh with the main server's true MAC address. Otherwise, if the ARP cache expires and gets updated with the main server's true MAC address, we will have a race condition such as that illustrated below:



Thus, if the backup server replies to the ARP Request message with its ARP Reply before the main server does it, the corresponding entry in the table will be updated with the information contained in the last message, and vice versa.

Appendix II: Countermeasures

In this practical we have learnt some of the threats posed by network analysers and ARP vulnerabilities. The impact of some attacks based on these techniques can be reduced by taking appropriate countermeasures.

An obvious mechanism to prevent an attacker from sniffing traffic is to always use protocols that guarantee the confidentiality of our communications by encrypting traffic (e.g., SSL). Furthermore, there are a number of tools generally known as *antisniffers* that help to detect if a sniffer is running in some network host.

Regarding ARP spoofing, there are also some tools that detect whether an attack is launched (e.g. ARPWatch). Additional countermeasures include:

- Use the opción arp –s, indicating a permanent IP-MAC correspondence for some trusted hosts. This is not always possible and could be quite inflexible, but nonetheless could be useful in some environments.
- Use protocols such as DHCP (Dynamic Host Configuration Protocol). Thus, every time a host connects to the network a correct configuration is obtained from an authorised machine (the DHCP server). Keep in mind, however, that the security of the DHCP server should be paramount.