HADMÉRNÖK

VIII. Évfolyam 2. szám - 2013. június

Dániel Nagy
nagy.daniel@operculum.hu

DETECTING SNIFFERS ON AN ETHERNET NETWORK PART I. – THE THEORY

Abstract

Although wireless networks are spreading quickly nowadays, we find wired Ethernet networks in most institutes, offices and homes. This way most PCs are connected to an Ethernet network. Ethernet networks can be eavesdropped. The act of eavesdropping a network is called sniffing. Sniffing an Ethernet network does not require any special hardware or particularly advanced computer knowledge. Many sensitive data are transmitted unencrypted in practice nowadays. For example, emails, some of the passwords and all of our HTTP requests can be easily read by a sniffer connected to our network. This means sniffing has a serious impact on privacy and secrecy. These simple sniffers can be detected on a network by various methods. This paper presents some of this methods, and discusses the networking fundamentals of these methods.

Bár a vezetéknélküli hálózatok gyors fejlődésben vannak, a legtöbb intézményben, irodában és otthonban vezetékes Ethernet hálózatot találunk. Ilyenformán a legtöbb személyi számítógép Ethernet hálózathoz csatlakozik. Az Ethernet hálózatok lehallgathatók. A hálózat lehallgatását sniffingnek (szimatolás) nevezzük. Egy Ethernet hálózat lehallgatásához nem szükséges speciális hardver vagy különösebb számítástechnikai ismeret. Sok bizalmas információ kódolatlanul kerül továbbításra: az emailek, némely jelszavak és az összes HTTP kérés könnyen lehallgathatók egy sniffer segítségével. Ilyenformán a sniffingnek komoly jelentősége van a személyes és titkos információk tekintetében. Ezek az egyszerű snifferek különféle módokon detektálhatók. Jelen írás bemutat néhányat ezek közül, illetve szükséges hálózati ismereteket tárgyalja.

Keywords: *sniffing detection, Ethernet, OSI Level2, secrecy, privacy ~ hálózati forgalom figyelés, Ethernet, OSI adatkapcsolati retag, biztonság, titkosság*

THE STRUCTURE OF THIS DOCUMENT

This document is divided in two parts.

The *first part* discusses the topic-relevant aspects of Ethernet and Internet Protocol, Ethernet medium, certain header fields, ARP, ICMP, MAC, IP, and how these protocols are encapsulated into each other. Knowledge of this is important to understand sniffing and sniffing detection. Based on the aboves, sniffing methods on an Ethernet network and methods for the detection of these sniffing hosts are discussed.

The *second part* presents the experiments I performed, to put the theory in practice. The experiments were done with a software tool developed by myself for this purpose. The tool is called *mySnifferDetector*, which is a text-mode C++ application, running on Linux. The tool is capable of sending out specially modified Ethernet frames (forged frames) and listens for the replies on the network.

INTRODUCTION

On an Ethernet network a host normally processes only the messages which are addressed to it. On a shared medium all hosts can 'hear' all the traffic, so it is the task and responsibility of every host to process only the messages that are sent to that particular host. In a switched environment this is different, the switch also ensures that only the destination receives a frame, and the other hosts do not.

On the other hand, for several reasons, one might be interested in processing messages intended to others. One of the most common reasons is network debugging. If an administrator of a network experiences some problem with his network, he might want to investigate the reason. In order to do that, he would be interested in all of the traffic on the network. He will start a network analyzer tool which will make that host catch all the traffic on the network. Then he will examine the messages and look for the possible problems he is interested in. Obviously, to be able to undertake such an examination the software tool must make it possible to receive all the traffic on that network segment. This is called sniffing, but a sniffing with a justifiable and legal purpose. However not only network administrators with a good intention are able to set up a sniffer on a network. The purpose of sniffing can be malicious, by eavesdropping network traffic to get access to confidential or private data. The problem is that a sniffing host is normally unnoticed on a network. This is why a special program, a *sniffer detector* shall be used to identify the normally unnoticed, malicious sniffers.

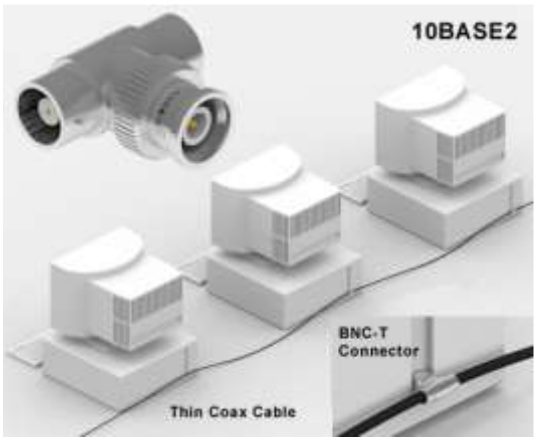
One of the main objectives of this paper is to investigate what methods can be used to detect a sniffer. Although the knowledge of Ethernet and Internet Protocol can be considered as a basic for the intended reader, I found it useful to describe the relevant parts of these protocols to make the understanding of sniffer detecting easier. Sniffing and sniffing detection could grow to be a very broad topic, which is why I used the following restrictions in my research :

- I considered only simple PCs, with ordinary Ethernet cards as a sniffer or a sniffer detector. All special hardwares, and PCs with special hardware were not considered.
- I considered only user space sniffers and sniffer detectors. All modified kernel, and special purpose built kernel modules were not considered.
- Sniffing and sniffing detection is considered only in Ethernet 802.3 (V2) environment.

ETHERNET

Shared medium

10BASE2 was not the first implementation of Ethernet, but it was the most widespread. The medium itself is (was) a coax cable, with a maximum length of 185 meters. Every host's NIC (Network Interface Card) has a BNC connector with a T plug on it. This plug enables the host to connect the cable, while maintaining the electric continuity of the cable. The cable is terminated by 50Ohm resistors at both ends. [1]



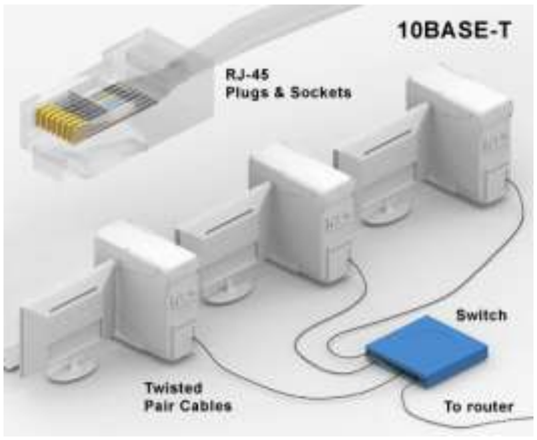
1. Figure. 10BASE2 network [17]

This network type makes it easy to understand the concept of shared medium just by looking at the topology. All network hosts transmit to the same medium, and all hosts receive every transmission. There is no central device which would take care of forwarding the messages to the right host. The hosts themselves need to filter the network traffic to process only those messages which were sent to them.

According to Spurgeon (2000) [2] 'Ethernet uses a broadcast delivery mechanism, in which each frame that is transmitted is heard by every station. While this may seem inefficient, the advantage is that putting the address-matching intelligence in the interface of each station allows the physical medium to be kept as simple as possible. On an Ethernet LAN, all that the physical signaling and media system has to do is see that the bits are accurately transmitted to every station; the Ethernet interface in the station does the rest of the work.'

Since all NICs use the same medium when they want to send data, first they check the medium whether there is another transmission ongoing. If there is, they will back off, if there is not they will start to transmit. Despite this precaution, it may happen that two stations somehow try to transmit at the same time. Should this happen, something that is the mix of the two transmissions will be on the medium, which will make no sense. This event is called a collision, and must be detected. Detection is done by the NICs themselves. While transmitting, the NIC senses the channel, and if it finds that there is another signal on the medium, not only the one that was sent by itself, it will send a jam-signal after which it will back off. The process is somewhat more complicated than this, but for now it is sufficient that Ethernet uses CSMA/CD, which means Carrier Sense Multiple Access with Collision Detection. The CSMA/CD protocol makes it sure that in a specific period of time there is only one message on the medium, and all the NICs can access the medium with fair access. (*Spurgeon, 2000)[2]*

The twisted-pair network topology is different. Every NIC connects to a central device. This central device may or may not examine and process the traffic. Normally the device which just forwards every transmission from every port to every port is called a *HUB*. When a HUB is used in a twisted-pair Ethernet network it is still a shared medium like 10BASE2, and despite the significant differences in the structure of the network, the working principle is the same.



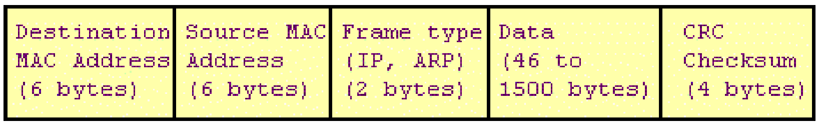
2. Figure. 10BASET network [17]

If this central device has built-in intelligence, it examines the destination address of every Ethernet frame it receives, and then forwards to the port where the destination is connected, in this case this device is called a *switch*.

Sniffing can be easily done in a shared medium environment, but it is also possible in a switched environment.

Ethernet frame

A transmission unit on the Ethernet network is called a frame. The Ethernet header is quite simple. It contains the destination address, the source address, the type of the payload and a CRC. Two possible values in the type field will be crucial in this paper. 0x0806 which indicates that the payload is an IP packet, and 0x0806 which indicates that the payload is an ARP packet. (Infocellar, n.d.) [3]



Destination MAC Address (6 bytes)	Source MAC Address (6 bytes)	Frame type (IP, ARP) (2 bytes)	Data (46 to 1500 bytes)	CRC Checksum (4 bytes)

3. Figure. Header of an Ethernet 802.3 v2 frame (Ethernet II, DIX) [3]

The address in the Ethernet frame is the hardware address of course, which is detailed in the following section.

In case most of the techniques which utilizes forged frames, this destination address field plays a *very* important part.

Addressing

According to Graham (2000)[4] 'The Ethernet MAC address is a 48 bit number. This number is broken down into two halves, the first 24-bits identify the vendor of the Ethernet board, the second 24-bits is a serial number assigned by the vendor. This guarantees that no two Ethernet cards have the same MAC address (unless the vendor fouls up). Duplicate address would cause problems, so uniqueness is very important. This 24-bit number is called the OUI ("Organizationally Unique Identifier").

However, the OUI is really only 22-bits long, two of the bits in that field are used for other purposes. One bit indicates if the address is a "broadcast/multicast" address, the other bit indicates if the adapter has been reassigned a "locally administered address" (where a network administrator reassigns the MAC address to fit some local policy).'

The destination addresses can be organized into types, this is shown in the table below.

MAC Type	Address range
Globally Unique	*0-**-**-**-**-** *4-**-**-**-**-** *8-**-**-**-**-** *C-**-**-**-**-**
Locally Administered	*2-**-**-**-**-** *6-**-**-**-**-** *A-**-**-**-**-** *E-**-**-**-**-**
Multicast	*1-**-**-**-**-** *3-**-**-**-**-** *5-**-**-**-**-** *7-**-**-**-**-** *9-**-**-**-**-** *B-**-**-**-**-** *D-**-**-**-**-** *F-**-**-**-**-** Note: except broadcast address
Broadcast	FF-FF-FF-FF-FF-FF

1. table. MAC address range [5]

When a NIC receives a frame the first thing it does is it checks the target address (the first six bytes of the frame). If the target address is its own MAC address, or the target address is a multicast or a broadcast address it will generate an interrupt. In turn the networking program of the operating system will read the frame from the buffer to process it. If the NIC considers that it is not the destination of that frame, it simply drops is. (Spurgeon, 2000) [2]

As it can be seen from the table, there is an important bit in the MAC address, called the multicast bit. This bit is the last bit of the most significant byte (in host byte order). To put it into other words, if the first byte is odd, then this address is a multicast MAC address. When IP is over Ethernet the MAC addresses 01:00:0E:**:**:** are assigned to IP multicast addresses.

Destination MAC addresses play a very important role in sniffer detection. The basic principle of the implemented sniffer detector methods is to send out an Ethernet frame with a *fake destination MAC address* in it.

INTERNET PROTOCOL

As it was written above, presenting Internet Protocol is beyond the scope of this document, and the intended reader is likely to have this knowledge. Nevertheless, it is necessary here to point out the parts which are needed for understanding sniffing detectors.

IPv4 is the dominant protocol for the Internet and on Ethernet networks as well. Although IPv6 exists, at the time of writing of this paper its use in practice is relatively rare compared to IPv4. Therefore, by the term IP in this document I always refer to IPv4.

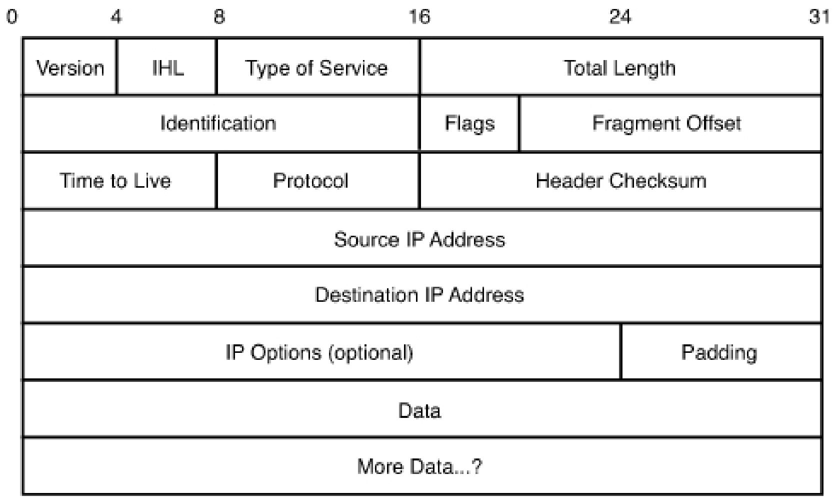
IP's main goal is to establish a logical addressing structure over the link layer, identify hosts on the network by these addresses (IP addresses) and deliver chunks of information called packets between these hosts. (It might be important to note here, that sometimes the expression 'datagram' is used for IP packets as well. It seems that the literature and recommendations are not consistent on this topic. According to tcpipguide.com [6] the term 'packet' will be used for IP throughout this document.)

IP addressing

On an IP network every node has (at least) one IP address. The IP addresses are logical addresses, they are set or assigned at software level. The IPv4 addresses comprise of four bytes. The IP packet may travel through various network nodes, on different physical mediums and different physical addresses. The sender and receiver application do not need to know these details, all they use is the IP address when addressing each other.

IP header

Every IP packet has a header section which contains administrative information about that packet. After the header comes the payload part, which is the actual data the IP packet carries. The figure below and a shortened explanation summarizes the structure of an IP packet by *Casad (2011) [7]*.



2. table. IP header [7]

Encapsulation

After the IP header comes the payload, the actual data that is sent by that IP packet. The payload can theoretically be more than the maximum payload size of an IP packet which is 64 kbytes minus the header size. If the data is bigger than the actual maximum IP data payload size, the data will be divided into chunks, and will be transported in more IP packets.

It is rare for applications to exchange data directly in IP packets. Because of the lowlevelness of IP, there is no port facility (the operating system doesn't know to which application should the data be forwarded), there is no error checking (packets can be simply lost) and packets can arrive not in the order they were sent. To overcome problems like these, higher level protocols are used and this higher level protocol (typically TCP or UDP) is put in the payload part of the IP packet. This is called encapsulation.

For detecting sniffers with the two methods presented in this document we don't need to concentrate on the fact that data might be chopped into more parts and put into numerous IP packets. We deal with ICMP echo request-reply pairs and ARP request-reply pairs. Furthermore, in this paper levels higher than the network level are irrelevant, so TCP will not be an issue here either.

In this document, two kinds of packets are dealt with: ICMP echo and ARP. They are significantly different. ICMP protocol is encapsulated into a normal IP packet. By this characteristic it is similar to the TCP packet, but ICMP is considered as a network layer functionality. While transmitting an ICMP packet the ethertype in an Ethernet frame will be '0x0800' which means IP, since ICMP packet is an IP packet as long as Ethernet is concerned.

The value of the 'protocol' filed in the IP header will indicate that we deal with an ICMP packet.

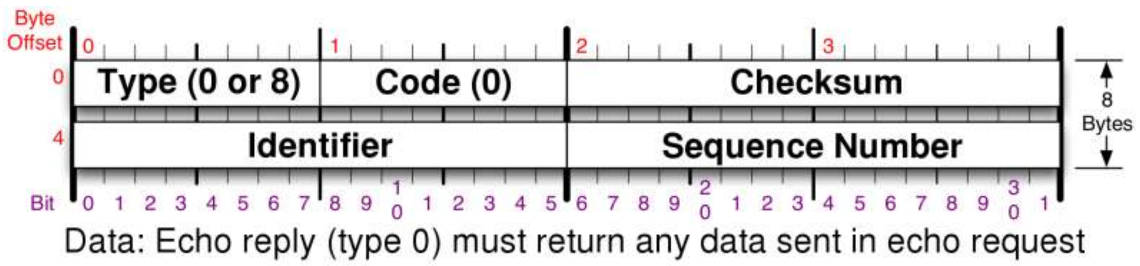
ARP packet on the other hand is different. ARP has its own header, and does not need an IP header. An ARP packet is encapsulated directly into the payload of an Ethernet frame. In accordance with this, when transmitting an ARP packet, the ethertype is set to '0x0806' in the Ethernet header which indicates that the payload is an ARP packet. (Spurgeon, 2000) [2]

Why is this so? *Because ICMP is IP specific, but ARP is not.* ARP must work with networking protocols other than IP as well, so it must not have IP specific solution built into it in order to work.

ICMP PROTOCOL

ICMP stands for Internet Control Message Protocol defined in RFC792. It is not used for data exchange between applications, but rather for hosts to inform each other if there is some problem with the transmission. Although ICMP is encapsulated in IP packets (like TCP or UDP), regarding its functionality it is considered as an Internet layer protocol like IP itself. ICMP header starts right after the IP header (as if it was be the payload of the IP packet).

The ICMP header is rather simple. Its first two bytes determine a type and a code, then comes a two-byte-long checksum. The next four bytes may contain ICMP type specific information. In the case of an echo request, it is an identifier and sequence number. These fields are used by ping application implementations in various ways.



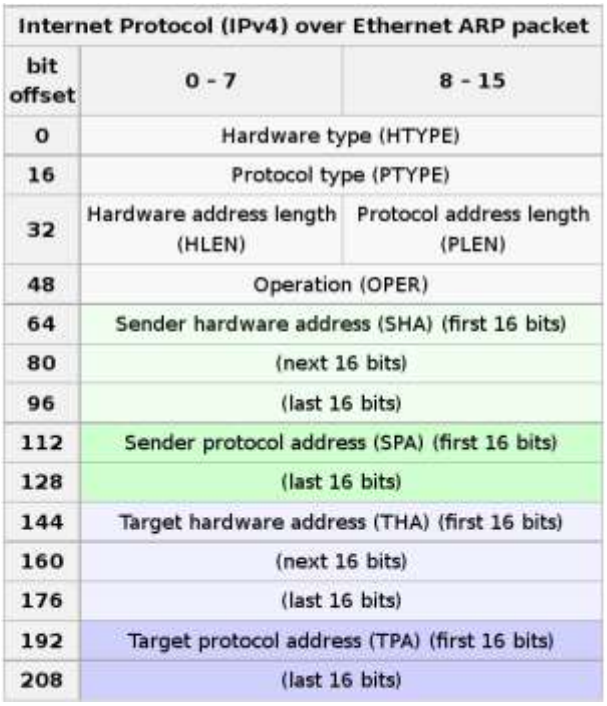
4. Figure. ICMP echo reply packet [8]

When a host receives an echo request ICMP packet it will reply to it with an echo reply ICMP packet. This pair of ICMP packets is the well know *ping*. What will be important as for as sniffing is concerned is that ping is an IP level functionality.

ARP PROTOCOL

The address resolution protocol is used to map IP addresses and MAC addresses together. The principle is quite simple. When the kernel has an IP packet to send, it must construct an Ethernet frame to encapsulate that IP packet in it. The payload part of the Ethernet frame will be the whole IP packet itself. Filling out the source field of the Ethernet frame is easy, since it is the MAC address of the machine from which the packet is being sent. Now the kernel has to find out the MAC address of the computer by the given IP address.

If the kernel does not know the MAC address for an IP address, it constructs an APR packet to ask it on the network. The ARP packet could be verbalized as “who has IP 192.168.1.4” (in case we look for the MAC address of the host identified by IP address 192.168.1.4), and this question is then sent out to the broadcast MAC address. As it was sent to the broadcast address, all computers will receive it, and switches will forward it to all ports. When a computer receives an ARP request packet which contains its own IP address, it will respond to it by the ARP reply message. This message is basically contains an IP address and the corresponding MAC address. (Spurgeon, 2000) [2]



Internet Protocol (IPv4) over Ethernet ARP packet		
bit offset	0 - 7	8 - 15
0	Hardware type (HTYPE)	
16	Protocol type (PTYPE)	
32	Hardware address length (HLEN)	Protocol address length (PLEN)
48	Operation (OPER)	
64	Sender hardware address (SHA) (first 16 bits)	
80	(next 16 bits)	
96	(last 16 bits)	
112	Sender protocol address (SPA) (first 16 bits)	
128	(last 16 bits)	
144	Target hardware address (THA) (first 16 bits)	
160	(next 16 bits)	
176	(last 16 bits)	
192	Target protocol address (TPA) (first 16 bits)	
208	(last 16 bits)	

5. Figure. ARP packet [9]

SNIFFING

Promiscuous mode

As it was written in above, on a shared medium network all hosts receive all the traffic sent to the network, and it is the host's task and responsibility to filter out the messages of its own. What does sniffing mean in such a context?

The NIC itself decides if a frame is addressed to it, and only if it finds this to be the case will the frame be forwarded to the operating system. When the NIC is set into another operating mode, called 'promiscuous' mode by its driver we talk about sniffing. A NIC which is set to promiscuous mode will forward all the received Ethernet frames to the operating system. This way every frame will be processed by the operating system, not only the ones which are meant to be sent to that host, so promiscuous mode is essential for sniffing. Without promiscuous mode, the NIC itself drops the frames with other destination addresses, so the user level application would not have a chance to listen to network traffic between other hosts.

The network program of the kernel also performs filtering, and drops frames with inappropriate addresses. So we have a so-called hardware filtering done by the NIC itself and a higher level filtering, or so called software filtering by the kernel. According to Mumatz & Yasir (2008) [10], software filtering is varied by kernel to kernel so different operating systems and even different kernel versions may act differently regarding this matter. This is an important issue regarding the working principles of sniffer a detector.

Sniffing can be defined as a case when a user space application receives and processes frames sent to other host(s) on the network.

Why can sniffing be malicious? Every data that is sent or received by a host in plain text can be read. Some examples: By reading the HTTP request messages we can track what web pages a host visits. Telnet, POP3 and FTP passwords and logins are in plain text, so if a host logs on to any of these services we can get the login instantly. Normally, emails are not encrypted, so every time a host downloads emails we can simply read them. These are just a few examples, but it is clearly shown that sniffing has a huge impact on privacy and secrecy on the network.

Types of sniffers

Before we step further, it is time to establish categories regarding sniffing.

Mumatz & Yasir (2008) [10] create two categories. We talk about *passive sniffing* when the sniffing host just 'listens'. I found this a somewhat misleading name and explanation. Passive in terms of sniffing does not necessarily mean the host does not send any replies to the network, or its presence is unnoticeable. For example it may transmit, and typically it updates its ARP cache. Passive sniffing means that the sniffer does not alter the networking environment.

Active sniffing on the other hand refers to cases when the sniffing host somehow alters the networking environment to catch messages that would not normally reach it.

As an example for the above, let us imagine that someone sits next to a table in a restaurant to overhear the conversation at another table. He still can be seen, emits heat and sometimes calls for the waiter. This can be an analogue for passive sniffing. In the restaurant example if the sniffer moves tables and chairs to get closer to the conversation that is active sniffing.

The above also means that passive sniffing is normally associated with shared medium Ethernet, and active sniffing is normally associated with switched Ethernet networking environment. In a switched environment traffic from A to B only flows between A and the switch and B and the switch, so the sniffer needs to modify the networking environment to physically receive the packets.

Susid (2004) [11] uses two further categories. A *standalone* sniffer is a device with a purpose of sniffing and nothing else. It might be a PC with special software and/or hardware. Only a standalone sniffer can be made absolutely passive. With modified kernel and/or hardware a standalone sniffer will not generate any traffic on the network.

A *non-standalone* sniffer is a sniffer application that runs on a PC which is normally connected to the network with some other purpose. It runs a user space sniffer program. The operating system, as well as other user space applications run on this PC.

Sniffing on shared medium

So passive sniffing merely means a host with its NIC set into promiscuous mode. In practice, this means simply connecting a PC to the network to be sniffed, and starting a packet analyzer (e.g. Wireshark [16]) on it.

Methods for sniffing on switched network

For sniffing in a switched environment it is not enough just to set the NIC into promiscuous mode, since the switch will only forward Ethernet frames to the destination.

ARP Spoofing/Poisoning

According to Sumit [12] “We have explained earlier how ARP is used to obtain the MAC address of the destination machine with which we wish to communicate. The ARP is stateless, you can send an ARP reply even if one has not been asked for and such a reply will be accepted. Ideally when you want to sniff the traffic originating from machine Venus, you can ARP Spoof the gateway of the network. The ARP cache of Venus will now have a wrong entry for the gateway and is said to be poisoned. This way all the traffic destined for the gateway will pass through your machine. Another trick that can be used is to poison a hosts

ARP cache by setting the gateway's MAC address to FF:FF:FF:FF:FF:FF (also known as the broadcast MAC).

Note: This way you can sniff the traffic from the target machine to the gateway, but not the traffic from the gateway to the target machine. In order to do that, you will need to poison the ARP cache of the gateway too. Given the importance of the gateway machines, quite a few administrators often install programs like arp-watch to detect such malicious activities. Hence trying to poison the gateway might be risky.

MAC flooding

Wireshark Wiki [16] writes “Switches keep a translation table that maps various MAC addresses to the physical ports on the switch. As a result of this it can intelligently route packets from one host to another. The switch has a limited memory for this work. MAC flooding makes use of this limitation to bombard the switch with fake MAC addresses till the switch can't keep up. The switch then enters into what is known as a “failopen mode” wherein it starts acting as a hub by broadcasting packets to all the machines on the network. Once that happens sniffing can be performed easily. ”

SNIFFER DETECTION METHODS

Since we can not tell what a host does with the packets it receives (reads and analyses them or drops them), we qualify a host as a sniffing host, if we found its NIC in promiscuous mode, and/or we found that the host alters the network in some way to route packets to itself. These two circumstances normally do not occur in a network, so if we find this, we have a good reason to suspect the presence of the act of malicious sniffing.

Therefore detecting sniffers can be approached from two directions. One way is to find a NIC in promiscuous mode, another is to find out whether a host has altered the network.

We classified sniffers either as active or passive sniffers. Passive sniffers work on shared mediums and do not work in switched environments. In this paper work the terms 'sniffer' and a 'host with its NIC in promiscuous mode' are equal.

Host based detection

We refer to 'host based detection', when we have access to a PC, and we are able to check the presence of sniffer applications on it. This can be done by checking if the NIC is in promiscuous mode, and by browsing log files. With network based detection methods we do not have access to the PCs, and we perform the sniffer detection through the network. This paper focuses on network-based detection.

For an example, with the following command line parameters, I list the interface attributes of my desktop PC. First the interface is not in promiscuous mode. Then I set it to promiscuous mode, and list again. I marked the change with red.

```
root@bubugep:~# ifconfig
eth1      Link encap:Ethernet  HWaddr 00:26:18:0c:c1:d8
          inet addr:192.168.1.5  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::226:18ff:fe0c:c1d8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6689606 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4542657 errors:30858 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7666707634 (7.1 GiB)  TX bytes:1052889847 (1004.1 MiB)
          Interrupt:41 Base address:0xe000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12390 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12390 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:501687 (489.9 KiB)  TX bytes:501687 (489.9 KiB)

root@bubugep:~# ifconfig eth1 promisc
root@bubugep:~# ifconfig
eth1      Link encap:Ethernet  HWaddr 00:26:18:0c:c1:d8
          inet addr:192.168.1.5  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::226:18ff:fe0c:c1d8/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:6689615 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4542668 errors:30858 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7666709150 (7.1 GiB)  TX bytes:1052891136 (1004.1 MiB)
          Interrupt:41 Base address:0xe000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12390 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12390 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:501687 (489.9 KiB)  TX bytes:501687 (489.9 KiB)
root@bubugep:~#
```

NETWORK BASED DETECTION METHODS

Ping method

Ping is an ICMP echo request/echo reply packet pair. If the networking program of a host operating system receives an echo request, it replies to it with an echo reply packet. When we ping a host with a ping application, we normally just give the destination IP address. The assembly of the ICMP packet and the Ethernet frame will be handled by the operating system. Normally the destination address of the Ethernet frame that encapsulates the ICMP packet determined by the ARP protocol. This ensures that the frame the echo request travels in will have its MAC address and IP address consistent, and ultimately the Ethernet frame will travel to the host that we want to ping.

Promiscuous hosts are not concerned with MAC addresses. The NIC will forward every Ethernet frame to the operating system. The NIC will check neither the destination address field of the frame, nor the content. This means that all echo request ICMP packets will be forwarded to the operating system, regardless of the Ethernet destination field of that request.

The ping method, as sniffer detection method can be summarized as this: Create a ping packet with the suspected host's IP address, and put it into an Ethernet frame, that has some non-existent destination address, and then wait for replies.

In order to do that, we forge an Ethernet frame with mismatching MAC address and IP address. We do not rely on ARP, we forge our own Ethernet frame in our way. If the NIC is in promiscuous mode, it will not filter out this frame and it will pass the payload to the operating system. From the operating system's perspective this is a normal ICMP echo request packet with the its IP address, so as every well-bred operating system, it will send back an echo reply message.

If we receive an echo reply for an echo request which was created with the wrong MAC address on purpose, we can be sure, that the replying host's NIC is in promiscuous mode, and as such likely sniffs the network. (Sumit) [12]

ARP method

The ARP method is very similar to the ping method. The difference is that we use ARP packets, not ICMP packets. We alter Ethernet destination field the same way and with the same purpose.

ARP request messages are normally sent out to broadcast addresses, since the purpose is to find the corresponding MAC address for an IP address. What happens if we send out a (forged) ARP request message that has some random non-existent MAC destination? Every normal host's NIC will drop this packet like they do that with every packet destined to another address. However, the NIC in promiscuous mode will accept this packet and it will forward it to the operating system. From the operating system's point of view it is an ARP request packet, and again as every well-bred operating system it will send back an ARP reply message. If we detect this behavior, we know that the host's NIC is in promiscuous mode.

What is written above is theory. In practice, the operating system has a software level filter built into their networking program, and will not answer blindly every packet from the NIC. This implementation varies from OS to OS. As the experiments of AbdelallahElhadj et.al. (2002) [13] and Mumatz & Yasir (2008) [10] and Sanai (2001) [14] shows, it became very important what MAC address we put into the Ethernet frame. I repeated these experiments with my own sniffer detector application.

ARP poisoning

A sniffer might send out forged ARP reply packets to switches which will cause traffic to be directed to it. This is called APR poisoning, and as such it is an active sniffing technique. However, ARP poisoning can be used to find sniffers, and this is a method for sniffer detection that works in a switched environment as well.

Khcherif [15] described this method. The working principle is to send out forged ARP reply packets with the intention of ARP poisoning. The ARP reply packet contains a fake IP address and our MAC address. We send out this packet in an Ethernet frame that has some forged MAC destination address in order to be received only by hosts in promiscuous mode. This way the sniffer will think that the fake IP address corresponds to our host.

If we send out forged ARP reply packets not to the broadcast address but to some random non-existing address, only the sniffing host will accept that ARP reply packet and so, only the sniffing host ARP cache will be poisoned.

Then we establish a TCP connection and sniff the network to see if the host transmits based on the faked entry.

Latency method

This method relies on the simple assumption that since a sniffer processes all the network traffic, it causes a considerable workload to that machine. According to AbdelallahElhadj et.al. (2002) [13] 'When an NIC is in promiscuous mode, all Ethernet traffic will generate hardware interrupts which will cause the Ethernet driver code to execute. Furthermore, with a sniffer running, captured packets must be passed to the user program running the sniffer. Crossing the kernel boundary is widely known to be relatively expensive. Thus, under heavy traffic, a sniffer will heavily degrade performance on promiscuous host.'

This method works by pinging the suspected machine and measure the average ping times. Then we flood the network with messages that would normally not passed by the suspected NIC, and instantly ping it again. If the NIC is in promiscuous mode, the traffic will cause some workload to the sniffer, since the NIC will not drop any packets.

If the ping after flooding is higher than before, we can assume that the increased ping time is due to the increased workload of the suspected machine, and from that we can conclude that its NIC is in promiscuous mode. Of course this method can cause false positives, since higher ping after flood is not necessarily in a cause and effect relationship with a NIC in promiscuous mode.

Decoy method

The decoy or bait method works somewhat differently, since it is not based on protocol level behavior or characteristics. The bait method involves creating a trap for the sniffer. We generate traffic on the network which is very tempting for the sniffer. This can be FTP, TELNET or POP3 connections (since they are unencrypted) or information sent in non-encrypted mail. The purpose is of course to watch if there is some activity which can be originated from the fact that someone sniffed this information. This is a very environment-specific method, and may not be really considered as an IT based method. AbdelallahElhadj et.al. (2002) [13]

DNS decoy method

The DNS decoy method is a special case of decoy methods.

Graham (2000) [4] describes the DNS decoy method: 'Many sniffing programs do automatic reverse-DNS lookups on the IP addresses they see. Therefore, a promiscuous mode can be detected by watching for the DNS traffic that it generates.

This method can detect dual-homed machines and can work remotely. You need to monitor incoming inverse-DNS lookups on the DNS server in your organization. Simply do a ping sweep throughout the company against machines that are known not to exist. Anybody doing reverse DNS lookups on those addresses are attempting to lookup the IP addresses seen in ARP packets, which only sniffing programs do.

This same technique works locally. Configure the detector in promiscuous mode itself, then send out IP datagrams to bad addresses and watch for the DNS lookups.

One interesting issue with this technique is that hacker-based sniffing programs tend to resolve IP addresses as soon as they are found, whereas commercial programs tend to delay resolution until the point where the sniffer user views the protocol decodes.'

Source-route method

Graham (2000) [4] describes this method too: 'Another technique involves configuring the source-route information inside the IP header. This can be used to detect sniffers on other, nearby segments.

1. Create a ping packet, but put a loose-source route to force it by another machine on the same segment. This machine should have routing disabled, so that it will not in fact forward it to the target.
2. If you get a response, then it is likely the target sniffed the packet off the wire.
3. In the response, doublecheck the TTL field to find out if it' came back due to sniffing (rather than being routed correctly)

Details:

In loose source-routing, an option is added to the IP header. Routers will ignore the destination IP address and instead forward to the next IP address in the source-route option. This means when you send the packet, you can say "please send packet to Bob, but route it through Anne first".

In this scenario, both "Anne" and "Bob" are on the segment. Anne does not route, and therefore will drop the packet when received. Therefore, "Bob" will only respond if he has sniffed the packet from the wire. On the off chance that Anne does indeed route (in which case Bob will respond), then the TTL field can be used to verify that Bob responded from routing through Anne, or answering directly.'

What destination MAC addresses should be used

With both ping and ARP methods, the destination address of the Ethernet frame is critical. As mentioned before, in promiscuous mode the NIC accepts all frames regardless of the destination address. However this is not enough, we need to obtain reply from the networking program of the target host. This depends on the MAC destination address, as well as the OS of the target host.

According to Sanai (2001) [14] the following MAC addresses are worth a try and because of the reasons explained below:

FF-FF-FF-FF-FF-FF broadcast address:

All nodes should receive this kind of packet and respond because it is a broadcast address. A usual ARP request packet uses this address.

FF-FF-FF-FF-FF-FE fake broadcast address:

This address is a fake broadcast address missing the last 1 bit. This is to check whether the software filter examines all bits of the address and whether it will respond.

FF-FF-00-00-00-00 fake broadcast 16 bits:

This address is a fake broadcast address in which only the first 16 bits are the same as the broadcast address. This may be classified as a broadcast address and replied when the filter function only checks the first word of the broadcast address.

FF-00-00-00-00-00 fake broadcast 8 bits:

This address is a fake broadcast address in which only the first 8 bits are the same as the broadcast address. This may be classified as a broadcast address and replied when the filter function only checks the first byte of the broadcast address.

01-00-00-00-00-00 group bit address

This is an address with only the group bit set. This is to check whether this address is considered as a multicast address as Linux does.

01-00-5E-00-00-00 multicast address 0

Multicast address 0 is usually not used. We use this as an example of a multicast address not registered in the multicast list of the NIC. The hardware filter should reject this packet. However, this packet may be misclassified to be a multicast address when the software filter does not completely check all bits. The system kernel thus may reply to such packet when the NIC is set to promiscuous mode.

01-00-5E-00-00-01 multicast address 1

Multicast address 1 is an address that all hosts in the local network should receive. In other words, the hardware filter will pass this kind of packets by default. But it is possible that the NIC does not support multicast mode and does not respond. So this is to check whether the host supports multicast addresses.

CONCLUSION

In this article eavesdropping (sniffing) possibilities on an Ethernet network were discussed. Sniffing is quite straightforward, easy to perform, and has an impact on privacy and secrecy. This leads to the need of sniffer detection. Sniffer detection is possible by various methods. This article focused on methods which utilize the fact that every sniffing host has its NIC in promiscuous mode, which allows them to capture every frame on the medium. By sending out special, custom forged Ethernet frames, these sniffing hosts can be lured to send and answer which in turn will reveal them. With the necessary programing and networking knowledge it does not seem particularly difficult to develop an application which can detect sniffers.

The second part of this article evaluates the statement above. How effective a forged-frame based application can be when it comes to sniffer detection? The second part presents the experiments I performed with my test setup. I used a desktop computer for a sniffer detector (mySnifferDetector) developed by myself to run on, and a laptop computer for sniffing. I sniffed the network with Linux and with WindowsXP and then studied if mySnifferDetector would find the sniffer or not. The conclusion part at the end of the second chapter contrasts the theory and the experiments.

References

[1] Wikipedia (n.d) Ethernet [Online]
Available at: http://en.wikipedia.org/wiki/Ethernet [Accessed: 10.09.2012.]

[2] Spurgeon, C. (2000) Ethernet: The Definitive Guide. O'Reilly and Associates, Inc. 101 Morris Street, Sebastopol, CA 95472

[3] Infocellar (n.d.) Ethernet Frame [Online] Available at:
http://www.infocellar.com/networks/ethernet/frame.htm [Accessed: 10.09.2012.]

[4] Graham, R. (2000) Sniffing (network wiretap, sniffer) FAQ. [Online]
Available at: http://newdata.box.sk/2001/jan/sniffing-faq.htm [Accessed: 10.09.2012.]

[5] mynetwatchman.com (n.d.) Idiot's Guide to Network Analysis [Online]
Available at: http://www.mynetwatchman.com/pckidiot/ [Accessed: 10.09.2012.]

[6] The TCP/IP guide (n.d.) [Online]
Available at: http://tcpipguide.com/
[Accessed: 10.09.2012.]

[7] Casad, J. (2011) Sams Teach Yourself TCP/IP in 24 Hours. 5th ed. Pearson Education Inc. USA

[8] nmap.org (n.d.) TCP/IP Fingerprinting Methods Supported by Nmap
[Online] Available at: http://nmap.org/book/osdetect-methods.html
[Accessed: 10.09.2012.]

[9] Wikipedia (n.d) Address Resolution Protocol [Online] Available at:
http://en.wikipedia.org/wiki/Address_Resolution_Protocol
[Accessed: 10.09.2012.]

[10] Mumtaz AL-Mukhtar, Yasir Ahmed Abdullah (2008) 'Developing a Sniffer Detector for Windows Operating Systems' The 1stRegional Conference of Eng. Sci. NUCEJ Spatial ISSUE 11(1) pp84-90

[11] Susid, D. (2004) An evaluation of network based sniffer detection Sentinel. School of Economics and Commercial Law GÖTEBORG UNIVERSITY Department of Informatics Master of Science Thesis

[12] Sumit D. () Sniffers Basics and Detection Information Security Management Team [Online] Available at:
http://www.just.edu.jo/~tawalbeh/nyit/incs745/presentations/Sniffers.pdf
[Accessed: 10.09.2012.]

[13] H. AbdelallahElhadj , H. M. Khelalfa & H. M. Kortebi (2002) 'An Experimental Sniffer Detector: SnifferWall' SEcurite des Communications sur Internet pp.69-80

[14] Sanai, D. (2001) Detection of Promiscuous Nodes Using ARP Packets. [Online] Available at: www.securityfriday.com/promiscuous_detection_01.pdf [Accessed: 10.09.2012.]

[15] Khcherif, R.(n.d.) ARP cache Poisoning For the Detection of Sniffers in an Ethernet Network. [Online] Available at: www://lyle.smu.edu/~rewini/5-7339/raoudha-9-28-04.ppt [Accessed: 10.09.2012.]

[16] Wireshark Wiki (n.d.) Ethernet capture setup. [Online] Available at: http://wiki.wireshark.org/FrontPage [Accessed: 10.09.2012.]

[17] Computerlanguage (n.d.) [Online] Available at: http://www.computerlanguage.com/ [Accessed: 10.09.2012.]