Promiscuous Mode Detection Platform

Zouheir Trabelsi and Hamza Rahmani

College of Telecommunications
The University of Tunisia
Cité Technologique des Communications
Route de Raoued Km 3,5 – 2083 El Ghazala, Ariana, Tunisia

Abstract. Among various types of attacks on an Ethernet network, "sniffing attack" is probably one of the most difficult attacks to handle. Sniffers are programs that allow a host to capture any packets in an Ethernet network, by putting the host's Network Interface Card (NIC) into the promiscuous mode. When a host's NIC is in the normal mode, it captures only the packets sent to the host. Since many basic services, such as FTP, Telnet and SMTP, send passwords and data in clear text in the packets, sniffers can be used by hackers to capture passwords and confidential data.

A number of anti-sniffers have been developed, such as PMD [18], PromiScan [17] and L0pht AntiSniff [19]. An anti-sniffer is a program that tries to detect the hosts running sniffers, in a Local Area Network (LAN). Current anti-sniffers are mainly based on three detection techniques, namely: the ARP detection, the DNS detection, and the RTT (Round Trip Time) detection techniques [13 and 16]. However, sniffers are becoming very advanced so that anti-sniffers are unable to detect them. The main drawback of these detection techniques is that they rely on the ARP, ICMP and/or DNS reply messages generated by the sniffing hosts. Therefore, in order to stay undetectable by anti-sniffers, advanced sniffers do not generate such reply messages while sniffing.

This paper discusses an anti-sniffer based on a new detection technique. The technique uses mainly ARP cache poisoning attack to detect sniffing hosts in an Ethernet network. The technique is implemented in a tool, called SupCom anti-sniffer, which automatically gives system administrator a better helping hand regarding the detection of sniffers. Four anti-sniffers, PMD [18], PromiScan [17], L0pht AntiSniff [19] and SupCom anti-sniffer, are tested and the evaluation results show that SupCom anti-sniffer succeeded to detect more sniffing hosts than the other anti-sniffers.

1 Introduction

Malicious users can easily steal confidential documents and anyone's privacy by sniffing a network. It can be done simply by downloading free sniffer software from the Internet and installing it into a personal computer (PC). Sniffers capture all packets in a network. To achieve this, the sniffer sets the Network Interface Card (NIC) of the computer into a mode called "promiscuous mode". Then the NIC will blindly receive all packets and pass them to the system kernel. Packets that are not

Trabelsi Z. and Rahmani H. (2004).
Promiscuous Mode Detection Platform.
In *Proceedings of the 2nd International Workshop on Security in Information Systems*, pages 279-292
DOI: 10.5220/0002682902790292
Copyright © SciTePress

supposed to arrive to that PC are no longer blocked by the NIC. Those PC's with promiscuous NIC's are running sniffers.

Many basics services, such as FTP, Telnet and SMTP [9], send clear text data in the packets. A sniffer captures all packets and displays their contents on the hacker's computer screen, for examples the passwords used to authenticate during an FTP session, or the message of an email in SMTP packets. Hackers can spy the users of a network, just by reading and analyzing the contents of the packets going to and out of the users' hosts. This type of attack on a network is usually difficult to detect, since it does not interfere with the network traffic at all. System administrators are facing difficulties to detect and deal with this attack.

In this paper, we first explore three different techniques used to detect sniffing hosts in an Ethernet network, and we discuss their limits. The three techniques are: the ARP detection technique, the DNS detection technique and the RTT detection technique. Most anti-sniffers, such as PMD [18], PromiScan [17] and L0pht AntiSniff [19], are based on these detection techniques. However, the techniques present many drawbacks so that advanced sniffers are designed in such a way they can stay undetectable by anti-sniffers.

Then, we discuss an anti-sniffer based on a new detection technique. The technique includes mainly three phases and uses ARP cache poisoning attack to detect sniffing hosts, in an Ethernet network. Based on this technique, a tool, called SupCom anti-sniffer, is implemented. SupCom anti-sniffer gives automatically system administrators a better helping hand regarding the detection of sniffers. Four anti-sniffers, PMD[18], PromiScan[17], L0pht AntiSniff [19] and SupCom anti-sniffer, are tested and the evaluation results show that SupCom anti-sniffer gives a better detection performance than the others.

2 Basic knowledge

2.1 NIC's hardware addresses

All the NIC cards on the Ethernet are represented by a 6-bytes hardware address (MAC address). The manufacturer assigns this address such that each address is unique in the whole world. Theoretically, there are no two NIC's having the same hardware address. All communications on the Ethernet are based on this hardware address. The NIC, however, can set up different filters (called hardware filter) in order to receive different kinds of packets. The following are a list of hardware filters:

- *Broadcast*: Receive all broadcast packets. Broadcast packets have destination address FF:FF:FF:FF:FF:FF. The purpose of this mode is to receive the packets which are supposed to arrive at all nodes existing on the network.
- *Multicast*: Receive all packets which are specifically configured to arrive at some multicast group addresses. Only packets from the hardware multicast addresses registered beforehand in the multicast list can be received by the NIC.

- *All Multicast*: Receive all multicast packets. Since this mode may also correspond to other high level protocols other than IPv4, all Multicast will receive all packets that have their group bit set (01:00:00:00:00:00).
- *Promiscuous*: Receive all packets on the network without checking the destination address at all.

2.2 ARP messages

ARP messages are exchanged when one host knows the IP address of a remote host and wants to discover the remote host's MAC address. For example, if host1 wants host2's MAC address, it sends an ARP request message (Who has?) to the broadcast MAC address (FF:FF:FF:FF:FF:FF) and host2 answers with his addresses (MAC and IP). Basically, an ARP message on an Ethernet/IP network has 8 important parameters:

- Ethernet header:
    - Source MAC address
    - Destination MAC address
    - Ethernet Type (=0x0806 for ARP message)
- ARP message header:
    - Source IP address
    - Source MAC address
    - Destination IP address
    - Destination MAC address
    - Operation code:
        - 1: for ARP request
        - 2: for ARP reply.

It is important to mention that there is nothing specifying that there must be some consistency between the ARP header and the Ethernet header. That means you can provide uncorrelated addresses between these two headers. For example, the source MAC address in the Ethernet header can be different from the source MAC address in the ARP message header.

3 Related Work

3.1 The RTT detection technique

The RTT (Round Trip Time) is the time of the round trip of a packet sent to a host. That is the time that a packet took to reach the destination, plus the time that a response took to reach the source. It is expected that the measurement of the RTT increases considerably when a host is in the promiscuous mode, since all packets are captured.

The idea behind the RTT detection technique ([16] and [13]), is first to send to a host, with a particular OS, a number of request packets, and wait for the responses packets, in order to take the RTT measurements. Then, the host is set to the promiscuous mode. And, the same request packets are sent again to the host, and the corresponding RTT measurements are collected. The RTT averages, the standard deviations, and the percentage of changes of the collected RTT measurements are computed. The RTT averages, standard deviations, percentage of changes are called the training data.

The samples of the collected RTT measurements represent two different populations, called the normal mode population and the promiscuous mode population. To show that the two averages of the samples RTT measurements are statistically different enough and therefore represent two different populations (the normal mode and the promiscuous mode populations), the z-statistics [1] model is used. The z-statistics model allows to make a judgment about whether or not a host's NIC is set to the promiscuous mode.

In the real world, the system administration has to identify first the OS of the suspicious host. This can be done by several available tools, such as Nmap [15]. Then, a number of request packets should be sent to the suspicious host in order to collect the corresponding RTT measurements.

The suspicious host can be either in the normal mode or in the promiscuous mode. Two z-statistics are computed. The first one, called the normal mode z-statistics, uses the training data related to the OS of the suspicious host for the normal mode, as the first population, and the collected data in the real world, as the second population. The second z-statistics, called the promiscuous mode z-statistics, uses the training data related to the OS of the suspicious host for the promiscuous mode, as the first population, and the collected data, as the second population. If the normal mode z-statistics is less than the z value (which is 2.36), then we may conclude that the host's NIC is almost 99% set to the normal mode, else, the host's NIC is set to the promiscuous mode.

***The limits of the RTT detection technique*:** The RTT detection technique is a probabilistic technique. Many known and unknown factors, such as the operating system of the suspicious host, and the LAN traffic, may affect considerably the results generated by any anti-sniffer based on this technique. When the LAN is under heavy traffic, this probabilistic technique may generate false decision regarding whether the suspicious host's NIC card is set to the promiscuous mode or to the normal mode. This is due mainly on the RTT measurements taken which may lead to a false decision. In addition, an advanced sniffer may attempt to put heavy traffic in the network in order to let the anti-sniffer generates misleading results.

The RTT detection technique attempts to send heavy traffic to a suspicious host on a particular open port, usually the FTP port (21). However, it is not common to have always the FTP port (21) open in each host in the network. Finally, to work appropriately, this technique needs to send heavy traffic on the network and then takes the RTT measurements. Such an action may cause some damage to the network's hosts and services, such as denial of service attacks.

3.2 The DNS detection technique

The DNS detection technique [13] works by exploiting a behavior common to many sniffers. Current sniffers are not truly passive. In fact, current sniffers do generate network traffic, although it is usually hard to distinguish whether the generated network was from the sniffer or not. It turns out that many sniffers do reverse DNS lookup (that is looking up a hostname by an IP address) on the traffic that it sniffed. Since this traffic is generated by the sniffer program, the trick is to detect this DNS lookup some how and distinguish it from normal DNS lookup requests.

To do that, we can generate fake traffic to the Ethernet segment with a source address of some unused IP address. Then, since the traffic we generate should normally be ignored by the hosts on the segment, if a DNS lookup request is generated, we know that there is a sniffer on the Ethernet segment. And by sniffing the packets on the Ethernet segment, we can detect which hosts are sending the DNS lookup requests.

The limits of the DNS detection technique: This technique can be quickly side stepped. Sniffers can easily be changed to not perform the reverse DNS lookup. Furthermore, hackers will become more intelligent so as to never perform the reverse DNS lookup either. This will render the technique completely useless.

3.3 The ARP detection technique

The ARP detection technique is described more in detail in our paper [16]. However, we need to describe it again here since; this paper uses some of its results.

The ARP detection technique consist into checking whether or not a suspicious host responds to ARP request packets that are not supposed to be treated by the suspicious host. Since the sniffing host receives all the packets, including those that are not targeting to it, it may make mistakes such as responding to a packet, which originally is supposed to be filtered by the host's NIC. Therefore, the detection is performed by checking the responses of ARP reply packets, when ARP request packets are sent to all hosts on the network.

On an Ethernet linked by IP addresses, packets are in fact sent and received based on hardware addresses (MAC address). Packets cannot be sent by just using an IP address. Therefore, the Ethernet needs a mechanism that converts IP addresses into hardware addresses. At this time, ARP packets are used. ARP packets belong to the link layer, which is the same layer as IP, so ARP packets does not affect the IP layer. Since IP addresses resolving is always available on an IP network, ARP packets become suitable packets for testing the response of the hosts when detecting promiscuous mode.

3.3.1 Promiscuous mode detection

When the NIC is set to promiscuous mode, packets that are supposed to be filtered by the NIC are now passed to the system kernel. Therefore, if we configure an ARP packet such that it does not have broadcast address as the destination address, send it to every host on the network and discover that some hosts respond to it, then those hosts are in promiscuous mode.

In this example, the ARP packet destination hardware address is set to an address that does not exist, for example 00-00-00-00-00-01. When the NIC is in normal mode, this packet is considered to be “to other host” packet, so it is refused by the hardware filter of the NIC. However, when the NIC is in promiscuous mode, the NIC does not perform any filter operation. Then this packet is able to pass to the system kernel. The system kernel assumes that this ARP requests packet arrives because it contains the same IP address as that machine, so it should respond to the packet. However, this is not true. There exists some sort of software filter in the kernel, called the Software Filter, because a packet is actually filtered again by the system kernel. The software filter depends on the operating system kernel.

3.3.2 Software filtering based detection

It is unnecessary to sent ARP packet with MAC addresses that do not exist, since the software filter will block such packets. However, we need to send ARP packets with MAC addresses that may pass the software filter. So that, we can understand the mechanism used by the software filter to filter packets based on their MAC addresses. The following are the list of hardware MAC addresses used to send ARP request packets, when the NIC is in the promiscuous mode (the hardware filter do not filter packets):

- FF:FF:FF:FF:FF:FF broadcast address :
  All nodes should receive this kind of packets 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.
- F0:00:00:00:00:00 fake broadcast 4 bits : This address is a fake broadcast address in which only the first 4 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 4 bits 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. So 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 the other word, 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, but this hypothesis was not available because all the available cards on the market bear multicasting. So this is to check whether the host supports multicast addresses.
- 01:00:5E:00:00:02 multicast address 2 : Multicast address 2 is used to all routers in the local networks. So 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 and also is not accepted by the software filter. The system kernel check the hardware result and one notices while the software filter always comes after the hardware filter, from which for the addresses multicast, if an address was rejected by the hardware filter she is therefore rejected by the software filter.

01:00:5E:00:00:03 multicast address 3 : Multicast address 3 is not assigned. So 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 and also is not accepted by the software filter. The system kernel check the hardware result and one notices while the software filter always comes after the hardware filter, from which for the addresses multicast, if an address was rejected by the hardware filter she is therefore rejected by the software filter.

3.3.3 Experiences and results

The tests are performed against a number of operating systems (Windows 9x, ME, 2000/NT and XP, Linux 2.4x and FreeBSD 5.0). As expected, all kernels respond to the broadcast address and multicast address 1 when the NIC is in normal mode. The test results using the hardware addresses listed in the previous section, are listed in Table 1.

However, when the NIC is set to the promiscuous mode, the results are OS dependent.

Microsoft Windows :

- In the case of Windows 9x and ME, it responds to fake broadcast addresses B47, B16, and B8. Hence, the software filters of Windows 9x and Me determine the broadcast address by checking only the first byte. Because when we test with fake address F0:00:00:00:00:00, it will not response, so the mechanism of check, try to check only FF:??:??:??:??:??. Therefore, the three addresses B47, B16 and B8 can be used to verify whether a NIC card is set to a promiscuous mode or not. If the NIC is in the promiscuous mode, it will responds to an ARP request packet, by an ARP reply packet.
- In the case of Windows 2000/NT, it responds to fake broadcast B47 and B16. Hence, the software filters of Windows 2000/NT determine the broadcast address by checking only the 2 first bytes. Since Windows 2000/NT responds to the fake broadcast B16 in the normal mode also, therefore, only the addresses B47 can be used to verify whether a NIC card is set to a promiscuous mode or not.

- In the case of Windows XP, it responds to fake broadcast addresses B47 and B16. Hence, the software filter of Windows XP determines the broadcast address by checking only the first two byte. Therefore, the two fake broadcast addresses B47 and B16 can be used to verify whether a NIC card is set to a promiscuous mode or not.

Linux and FreeBSD :

In the case of Linux 2.4x and FreeBSD 5.0, it responds to all fake broadcast and to all addresses with the group bit set. Therefore, any fake broadcast addresses can be used to very the promiscuous mode. In addition, any address with the group bit set can be used to verify the promiscuous mode, excluding the multicast address M1. Since, Multicast address M1 is an address that all hosts in the local network should receive.

Hardware Addresses \ Operating Systems		Windows XP		Windows Me/9x		Windows 2k/NT		Linux 2.4.x		FreeBSD 5.0	
		Norm.	Prom.	Norm.	Prom.	Norm.	Prom.	Norm.	Prom.	Norm.	Prom.
FF:FF:FF:FF:FF:FF	Br	O	O	O	O	O	O	O	O	O	O
FF:FF:FF:FF:FF:FE	B47	--	X	--	X	--	X	--	X	--	X
FF:FF:00:00:00:00	B16	--	X	--	X	**X**	**X**	--	X	--	X
FF:00:00:00:00:00	B8	--	--	--	X	--	--	--	X	--	X
01:00:00:00:00:00	Gr	--	--	--	--	--	--	--	X	--	X
01:00:5E:00:00:00	M0	--	--	--	--	--	--	--	X	--	X
01:00:5E:00:00:01	M1	O	O	O	O	O	O	O	O	O	O
01:00:5E:00:00:02	M2	--	--	--	--	--	--	--	X	--	X
01:00:5E:00:00:03	M3	--	--	--	--	--	--	--	X	--	X

O**: Legal response,** X**: Illegal response, --: No response**

3.3.4 The limits of the ARP detection technique

The main limits of this detection technique is that if a host does not generate any ARP reply messages while sniffing, then this technique becomes useless. Because this detection technique relies on the ARP reply messages generated by the sniffing host. Consequently, any anti-sniffer based on this detection technique is unable to detect the sniffing hosts that do not generate ARP reply messages.

4 ARP cache poisoning attack based detection technique

4.1 ARP cache

Each host in a network segment has a table, called ARP cache, which maps IP addresses with their MAC addresses. New entries in the ARP cache can be created or already existing entries can be updated by ARP request or reply messages.

Create a new entry: When an ARP reply message arrives in a host, an entry in the ARP cache should be created. If the entry exists already, then it should be updated. In addition, when a host receives an ARP request message, it believes that a connexion is going to be performed. Hence, to minimize the ARP traffic, it creates a new entry in its ARP cache and puts there the addresses provided in the ARP request message. It is important to mention that sending an ARP request message in unicast is totally RFC compliant. They are authorized to let a system checks the entries of its ARP cache.

Update an entry: When an ARP reply message or an ARP request message arrives in a host, if the entry exists already, then it will be updated by the addresses (the source MAC address and the source IP address) provided in the ARP message.

4.2 ARP cache poisoning attack

ARP cache poisoning attack is the malicious act, by a host in a LAN, of introducing a spurious IP address to MAC address mapping in another host's ARP cache. This can be done by manipulating directly the ARP cache of a target host, independently of the ARP messages sent by the target host. To do that, we can either:

- ➢ add a new fake entry in the target host's ARP cache
- ➢ or, update an already existing entry by fake addresses (IP and/or MAC addresses).

Create a fake new entry: To do that, we send an ARP request message to a target host, with fake source IP and MAC addresses. When the target host receives the ARP request message, it believes that a connexion is going to be performed, and then, creates a new entry in its ARP cache and puts there the fake source addresses (IP and/or MAC) provided in the ARP request message. Consequently, the target host's ARP cache becomes corrupted.

Update an entry with a fake entry: To do that, we just have to send an ARP reply message to a target host with fake IP and MAC addresses. Thus, even if the entry is already present in the target host's ARP cache, it will be updated, with the fake entries.

4.3 Sniffing hosts' detection technique

The proposed detection technique used to detect sniffing hosts is based mainly on the ARP cache poisoning attack. It consists of three different phases:

- In the first phase, we attempt to corrupt the ARP cache of each sniffing host in the LAN, with a fake entry, using ARP cache poisoning attack. We will demonstrate that only the ARP caches of the hosts running sniffers will be corrupted, and this attack on the ARP caches will not make any damage to the attacked hosts.

- In the second phase, we attempt to establish a TCP connexion with each host in the LAN on any port, whether it is an open port or a closed one.
- In the third phase, we sniff the LAN in order to capture any packet containing the fake entry. We will demonstrate that the hosts that sent TCP or ICMP packets containing the fake entry are running sniffers. However, the hosts that sent ARP request packets are not running sniffers.

The following sub-sections describe in detail the three phases. We assume that we use a host in the LAN, called the testing host, to do all the actions needed in the three phases.

4.3.1 Phase 1: ARP cache poisoning

The aim of this phase is to corrupt only the ARP caches of the sniffing hosts in a LAN. First, we send an ARP request message, with fake source IP and MAC addresses (IP-X and MAC-X), to all hosts in the LAN.

The ARP request message sent to the hosts has an Ethernet header and an ARP message header. Hence, we need to choose the values of the fields in each header, in order to let only the sniffing host processes the ARP request message. If we choose the destination MAC address in the Ethernet layer header as a broadcast address (*FF:FF:FF:FF:FF:FF)*, then all the ARP caches of the hosts in the LAN will be corrupted by the ARP cache poisoning attack. Such a destination MAC address is discarded because it does not allow us to detect which hosts are sniffing.

However, if the destination MAC address is the fake broadcast address B47 (*FF:FF:FF:FF:FF:**FE***), then any host, with any OS, set to the promiscuous mode will accept the ARP request message and send it to the ARP layer (refer to section 3.3.3). If the host is set to the normal mode, this ARP request message will be blocked at the Ethernet layer, since the destination MAC address is not an unicast address, a broadcast address nor a multicast address. Consequently, the values of the main fields of the ARP request packet used to corrupt only the ARP caches of the sniffing hosts are:

- **Ethernet header:**
  - Source MAC address = *Any MAC address*
  - Destination MAC address =
    *FF:FF:FF:FF:FF:**FE** (B47)*
  - Ethernet Type = *0x0806 (ARP message)*

- **ARP message header:**
  - Source IP address = *Fake IP address (IP-X)*
  - Source MAC address =
    Fake MAC address (MAC-X)
  - Destination IP address = *IP address of a target host in the LAN*
  - Destination MAC address = *00:00:00:00:00:00*
  - Operation code: *1 (ARP request)*

4.3.2 Phase 2: Establishing TCP connections

Then, for each host in the LAN, we will attempt to establish a TCP connection. To do that, we need to send TCP packets with the bit SYN set, from the testing host, to each host in the LAN. However, the source IP address in the IP header of the TCP packets is not the source IP address of the testing host. But, it is that fake IP address (IP-X). Each host in the LAN will process the received TCP packet.

The values of the some important fields of the TCP packet used to establish a TCP connexion with each host in the LAN are:

- **Ethernet header:**
  - Source MAC address = *The Testing host's MAC address*
  - Destination MAC address = *Target host's MAC address*
- **IP header:**
  - Source IP address = *Fake IP address (IP-X)*
  - Destination IP address = *IP address of a target host*
- **TCP header:**
  - *Destination Port = Any number between 1 and 65535 (for example: 40000)*
  - *Source Port = Any number*
  - *Bit SYN = 1*

4.3.3 Phase 3: Detection of the sniffing hosts

Just following the request for establishing a TCP connexion with each host in the LAN, we expect three types of possible replies packets come from the hosts.

- The first type can be a TCP packet indicating that the connexion can be done (the SYN and ACK bits are set).
- The second type can be an ICMP error message indicating that the connexion cannot be established because the port destination is inaccessible.
- The third type can be an ARP request message sent by a host to look for the MAC address of the fake source IP address IP-X.

The hosts that generate any TCP or ICMP reply packets with the fake addresses IP-X and MAC-X as the destination addresses in the IP header are consequently running sniffers. Because, those host's ARP caches are corrupted with the fake IP and MAC addresses (IP-X and MAC-X) and are able to provided the MAC address MAC-X of the IP address IP-X. It is important to indicate again that during the first phase we used the ARP cache poisoning attack to corrupt only the ARP caches of the sniffing hosts, with the fake entry (IP-X and MAC-X).

We use a sniffer to capture any TCP or ICMP packet on the LAN that has those fake IP and MAC addresses (IP-X and MAC-X) as the destination addresses, and has been sent by a host. All hosts that sent such TCP or ICMP packets are consequently running sniffers, and their IP addresses can be easily identified.

However, any host whose ARP cache is not corrupted would generate an ARP request message in order to get the MAC address of the fake IP address IP-X. This MAC address will be used later to send the reply message which is expected to be a TCP or ICMP packet. Therefore, any host in the LAN that will send ARP request message looking for the MAC address of the IP address IP-X are not running sniffers.

4.4 Discussion

The proposed detection technique does not rely on the DNS, ARP and ICMP messages generated by the sniffing hosts. In a LAN, even an advanced sniffer cannot stay undetectable by an anti-sniffer based on the proposed ARP cache poisoning attack detection technique. Unless the sniffer stops all types of traffic directed to and issued from the sniffing host. In such a situation, the sniffer becomes useless, since no other networking activities can be done while the sniffer is working.

4.5 Implementation

Based on the proposed ARP cache poisoning attack detection technique, an anti-sniffer with a Graphical User Interface (GUI), called SupCom anti-sniffer, has been developed using Visual C++6.0 and WinpCap Library. The anti-sniffer integrates a TCP and ARP packet generator and a sniffer with filtering capabilities. SupCom anti-sniffer allows to generate ARP request packet with fake source IP and MAC addresses. In addition, it is able to sniff the network and capture packets based on filtering rules defined by the users.

SupCom anti-sniffer uses the three phases discussed in the previous sections, to detect the sniffing hosts in a LAN. SupCom anti-sniffer is able to detect hosts running even advanced sniffers. Advanced sniffers are sniffers that do not send any ARP request and reply messages, ICMP and DNS messages, in order to stay undetected by current anti-sniffers.

4.6 Evaluation

Four anti-sniffers, PromiScan [17], PMD [18], L0pht AntiSniff [19], and SupCom anti-sniffer are used to detect sniffing hosts in a LAN, during two tests. In the first test, the sniffing hosts can generate ARP reply messages. The following table, Table 2, shows that all the four anti-sniffers are able to detect all the sniffing hosts. In the second test, the hosts are running an advanced sniffer that does not generate any ARP messages and reverse DNS lookup messages. The following table shows that only SupCom anti-sniffer was able to detect all the sniffing hosts. This experience demonstrates clearly that SupCom anti-sniffer is more efficient than current anti-sniffers, particularly when detecting advanced sniffers.

Table 2: Detection performance of some anti-sniffers

Anti-Sniffers	Test 1: simple sniffer (1)	Test 2: advanced sniffer (2)
PromiScan	All sniffing hosts detected	No sniffing hosts detected
PMD	All sniffing hosts detected	No sniffing hosts detected
L0pht AntiSniff	All sniffing hosts detected	No sniffing hosts detected
SupCom anti-sniffer	All sniffing hosts detected	**All sniffing hosts detected**

(1) ***Simple sniffer***: is a sniffer that allows the sniffing host to generate all type of ARP, ICMP, TCP, UDP, and reverse DNS lookup packets.

(2) ***Advanced sniffer***: is a sniffer that does not allow the sniffing host to generate ARP packets and reverse DNS lookup packets, in order to avoid detection by current anti-sniffers.

5 Conclusion

Today, the need for techniques and anti-sniffers to detect sniffing hosts in a network is unquestionable. Hackers do not need advanced knowledge about TCP/IP protocols or networking to sniff a network. Hackers are just downloading sniffers from the Internet, and using them to spy their target hosts, and steal confidential information.

Current anti-sniffers use many detection techniques, mainly the RTT detection technique, the DNS detection technique, and the ARP detection technique. These techniques have many drawbacks, so that well designed and implemented sniffers can stay undetectable by current anti-sniffers. When the sniffing hosts do not generate any reply ARP and DNS messages, or put heavy traffic on the network, these detection techniques become useless.

In this paper, we discussed a new detection technique based mainly on ARP cache poisoning attack. We demonstrated that an anti-sniffer based on this detection technique is more effective than current anti-sniffers. The experience shows that when the sniffers do not generate any ARP reply and DNS messages, or put continuously heavy traffic on the network, only an anti-sniffer based on the proposed detection technique can detect such sniffers.

Even though sniffers are difficult to detect, the technique can provide system administrator with a consistent decision. However, by combining many detection techniques in a single anti-sniffer, systems administrators will have more results that confirm whether or not a target host is running a sniffer.

References

1. Freedman, Pisani, Purves and Adhikari, "Statistics – Second Edition", W.W. Norton & Company, Inc. 1991.
2. Grundshober, S. "Sniffer Detector Report", Global Security Analysis Lab., Zurich Research Laboratory, IBM Research Division, June 1998.
3. Hornig, C., "A Standard for the Transmission of IP Datagrams over Ethernet Networks", RFC-894, Symbolics Cambridge Research Center, April 1984.
4. Plummer, David C., "An Ethernet Address Resolution Protocol-Converting Network Protocol to 48 bit Ethernet Address for Transmission on Ethernet Hardware", RFC-826, November 1982.
5. Postel, J., "Internet Protocol", RFC-791, USC/Information Science Institute, 1981.
6. Postel, J., "Transmission Control Protocol", RFC-793, USC/Information Science Institute, 1981.
7. Postel, J., "Internet Control Message Protocol", RFC-792, USC/Information Science Institute, 1981.

9. Richard Stevens – "TCP/IP Illustrated : Volume 1", 2001.
10. Security Software Inc., "Antisniff", Technical Report 2000, "http://www.securitysoftwaretech.com",
11. S. Grundschober. "Sniffer Detector Report", Diploma Thesis, IBM Research Division, Zurich Research Laboratory, Global Security Analysis Lab, June 1998.
12. J. Drury., "Sniffers: What are they and how to protect from them", November 11, 2000. http://www.sans.org/.
13. D. Wu and F. Wong., "Remote Sniffer Detection". Computer Science Division, University of California, Berkeley. December 14, 1998.
14. Daiji Sanai, "Detection of Promiscuous Nodes Using ARP Packets", http://www.securityfriday.com/.
15. Nmap Tools, http://securityfocus.com.
16. Zouheir Trabelsi, and all, "Malicious Sniffing Systems Detection Platform", The IEEE/IPSJ 2004 International Symposium on Applications and the Internet (SAINT2004)", Tokyo, Japan, January 26-30, 2004.
17. PromiScan anti-sniffer: "http://www.securityfriday.com".
18. PMD (Promiscuous Mode Detector): "http://webteca.port5.com".
19.L0phtAntiSniff:"http://www.l0pht.com/antisniff/"