An Experimental Sniffer Detector: SnifferWall

H. AbdelallahElhadj[1] , H. M. Khelalfa[2] & H. M. Kortebi[3]

1,2 & 3: Basic Software Laboratory, CERIST,
3, rue des frres Aissiou, Ben Aknoun, Algiers, Algeria.
{habdelallahElhadj,mkortebi}@mail.cerist.dz, khelalfa@wissal.dz

Abstract

An experimental sniffer detector, SnifferWall, is presented. Based on an open architecture, it combines between detecting hosts in promiscuous mode and hosts replaying information sniffed from honey pots. Two majors methods of detection are used:

- A detection based on MAC addresses.
- A decoy-based detection method.

Keywords: Sniffer, IDS, MAC, Honeypot

1. Introduction

The explosive growth of the Internet has been bringing about revolutionary changes in the ways we conduct daily activities such as government, business, health care, education, entertainment, etc. It has made possible e-commerce, e-banking and even e-government. The Internet phenomenon has paved the way for a new era of humanity: the information society. Such a technological development has had also its downside; Internet has experienced an unprecedented growth in security incidents and attacks on computers systems, networks, etc [20]. In addition, attacks have grown in sophistication as well, using a very large set of tools and techniques built upon the vulnerabilities of Internet in general and TCP/IP in particular. We count *sniffers* amongst these tools. Originally, sniffers were used as network administration tool. Now, they are used as a malicious means to violate a system security. In fact, currently, a sniffer designates a passive attack tool that can gather valuable information over a network without the knowledge of the networks legitimate owner. Using sniffers can have extremely grave consequences. Indeed, information gathered by a sniffer can include passwords, e-mail messages, encryption keys, sequence number, etc. [3][16]. In this article we present *SnifferWall*, a tool based on an integrated approach to remotely detect sniffers in a Local Area Network. It is based on two methods:

- Detection based on MAC addresses.
- Detection through deception or decoys (honeypot).

Our aim is to build an in-house sniffer detector that can be also used as a teaching aid in computer security classes.

An overview of Intrusion Detection Systems is presented in the following section. The third one introduces sniffers. In the fourth section, current approaches to sniffer detection are reviewed. In the fifth one, we describe the design of SNIFFERWALL as well as its current implementation.

2. Intrusion Detection Systems (IDS)

Intrusion detection (ID) is an important security mechanism. It is usually used in combination with other security mechanisms, to implement a site security policy [21]. By providing information to site administration, ID allows not only for the detection of attacks explicitly addressed by other security components (such as firewalls and service wrappers), but also attempts to provide notification of new attacks that may be bypassed by other mechanisms. Intrusion detection systems (IDS) can also be used in computer forensics. Thus, IDSs can make attackers more accountable for their actions; henceforth, they may act as a deterrent to attacks[2].

There are two main models of Intrusion detection: [5].

a) A Misuse Detection model: Detection is performed by looking for the exploitation of known weak points in the system, which can be described by a specific pattern or sequence of events or data, called signature.

b) An Anomaly detection model: Detection is performed by detecting changes in the patterns of utilization or behavior of the system, and users [6].

IDSs are either host-based or network-based [1]. Host-based systems make their decisions on information obtained from a single host (usually audit trails), while network-based systems obtain data by monitoring the traffic of information in the network to which the hosts are connected. IDS are commonly used to detect active as well as passive attacks; *sniffers* are included amongst the tools used in conducting passive attacks.

3. Sniffers

Because Ethernet networks are shared communication channels, the network interface of a computer on such a network sees all the packets transmitted on the segment it resides on. Each packet has a header indicating the recipient of the packet. Under normal operating procedures, only the machine with that proper address is supposed to accept the packet. However, the Network Interface Card (NIC) can be set to a special mode called *promiscuous mode* to receive all packets on the wire. A *sniffer* is a special program or piece of code that put the NIC in the promiscuous mode. A user may operate the sniffer remotely.

As stated in the introduction, sniffers are one of the first *legitimate* tools that allowed system administrators to analyze their network and troubleshoot network problems. Unfortunately, crackers have used sniffers maliciously to spy on networks and steal various kinds of data. Crackers install sniffers to obtain user names, passwords, credit card numbers, personal information, and other information that could be damaging to a person, a corporation or even a nation. When they obtain this information, crackers will use the passwords to attack other Internet sites and they can even turn a profit from selling credit card numbers.

Sniffer software can be downloaded from the Internet. The most popular sniffers are: Tcpdump [7] (the most famous), Sniffit [8], Ethereal [9], under Unix-like platform and Analyzer [10], Windump [11]. At `SecurityFocus.com`, there are 10 pages worth of sniffer tools. Here we should note that the belief that one can be protected from sniffers by using switches is no longer true [22].

4. Overview of Existing Sniffer Detectors

Sniffer detection can be divided into:

- Detection at local host level (host-based).
- Detection at Local Network Segment level (Network-based).

4.1. Host-based detection

According to the literature, most of the published work in the domain refers primarily to UNIX based tools.

For most versions of UNIX, use `ifconfig`. This command will tell the user whether the network interface card is in promiscuous mode or not. However, it is not a reliable means of detection, since an attacker may have trojanized the `ifconfig` command prior to installing the sniffer. In such a case, the output of `ifconfig` is compromised. There are other key utilities that an administrator can use to detect the presence of a sniffer. They include the following commands: `ls`, `df`, `du`, `ps`, `find`, and `netstat`, However, they can be trojanized as well [12].

BSD UNIX offers a tool called `cpm` [13] ("check promiscuous mode") developed by CERT/CC. `cpm` uses `socket(2), ioctl(2)` to read whether the network card (or cards if multihomed) have been set in promiscuous mode; and then reports the results to the console. Only those devices found in promiscuous mode will be listed.

For SunOS 5.5 and 5.6 use `ifstatus` [14]. It reports to the console the flags of network interface cards, indicating which cards are in debug or promiscuous mode.

4.2. Network based detection

The network approach allows the checking of an entire network from a single point of entry. It allows for the remote detection of a sniffer on a local network segment. The administrator runs the sniffer detector from a specified host and can perform various tests against other hosts to detect the presence of NICs in promiscuous mode in the network.

Detecting sniffers is a difficult task since they are passive tools of attack. This is reflected in the state of research in the domain. Indeed, few remotely based sniffer detector have been developed the two most famous tools are the two most famous tools for sniffer detection are: AntiSniff [15] (Windows platform) developed by l0pht Heavy Industries, and SNIFFER DETECTOR (Linux platform) developed by IBM-Zurich [3].

a) AntiSniff: Is the only commercially available production level tool for detecting sniffers at the network level. Currently version 1.x of AntiSniff performs three classes of tests:

- Operating System specific tests (ARP and Ether ping tests).
- DNS tests.
- Network latency tests.

The last test is the most powerful [15]. The latency technique (also named "load technique") uses the fact that NIC interrupts to the operating system degrades system performance. 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. The load technique works by remotely measuring the traffic on the segment. Response time is measured twice: one measurement is taken when there is no heavy network traffic. Another measurement is taken to determine the response time of the machine under heavy traffic, after the network has been flooded with huge quantities of fictitious traffic. Comparing the two measurements determine whether a sniffer is running on that host or not. At this point, AntiSniff version 1.x builds a baseline for the machine(s) being probed by issuing ICMP echo requests with microsecond timers.

The disadvantage [16] of this technique is that it can significantly degrade network performance. Furthermore, packets can be delayed simply because of the load on the wire, which may cause timeouts and therefore false positives. On the other hand, many sniffing programs are user mode programs whereas pings are responded to in kernel mode", and are therefore independent of CPU load on a machine, thereby causing false negatives.

b)Sniffer Detector(IBM-Zurich): It was developed by Stéphane Grundschober [3] at the Global Security Analysis Lab. The main idea of this tool is to spread bait that is presumably especially attractive to the sniffer's owner. First, fictitious `ftp` and `telnet` connections are created; then the tool waits for the intruder to use the information bait—i.e., reconnect to the `ftp` or `telnet` server.

In conclusion, we remark that *no integrated approach to sniffer detection exists yet*. Current tools resolve part of the problem. We feel that an integrated tool is necessary so that different complementary methods can be provided to the network administrator so that chances of sniffers eluding a network protection are minimized. In the next section we present SnifferWall, a sniffer detection tool based on an integrated approach.

5. SnifferWall

In this section, we present the design and implementation of a tool called SnifferWall, which help the system administrator detect remotely if a sniffer is running on an Ethernet network. SnifferWall offers an integrated approach to sniffer detection. The current types of methods that SnifferWall utilizes include:

- MAC-based detection methods.
- Decoy or deception-based methods.

5.1. MAC Based Detection

In normal mode, a received packet is filtered by the NIC; if the MAC address is correct the packet is given the operating system. When in promiscuous mode, all incoming packet are passed to the OS, as shown in the flowcharts of figure 1 and figure 2. Due to a TCP/IP stack vulnerability an Ethernet driver may not properly check the target MAC address of the Ethernet header when the NIC is in promiscuous mode. Hence, normally rejected packet might be accepted. Our idea is to exploit this vulnerability by building packets with fake MAC address, then sending them out to the suspect machine. If the latter is in promiscuous mode it will issue a response to our packets.

Two techniques are used:

- The Etherping test which use ICMP echo packets.
- The ARP test which use ARP packets.

5.1.1. Etherping Test

ICMP echo packets are sent to a target with the correct destination IP address, but with a bogus destination hardware address. Normally such packets are discarded. But when in promiscuous mode, some old Linux kernel and NetBSD will grab these packets as legitimate packets since their IP information is correct, and respond accordingly. If the target in question replies to our request, we know it is in promiscuous mode. This test has been included in SnifferWall mostly for pedagogical reasons. Students would be able to experiment with SnifferWall on old versions of Linux and BSD.

5.1.2. ARP test

An ARP request is sent to the target with all information valid except a bogus destination hardware address. Normally, if a machine is not in promiscuous mode, it will never see the packet, since the packet is not directed at it. If a machine is in promiscuous mode, the ARP request would be seen and the kernel would process it and reply. If a machine replies, we know that it is in promiscuous mode.

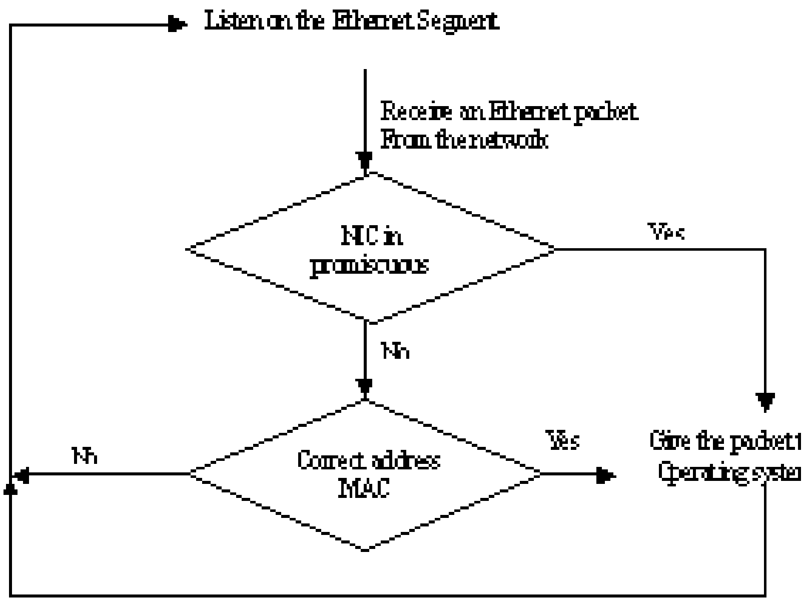

Figure 1: The logic control performed by the NIC

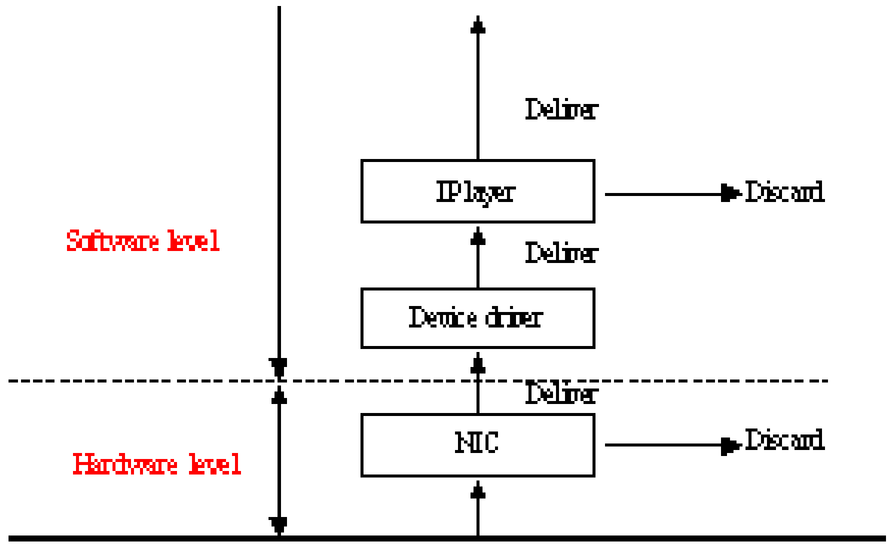

Figure 2: The different steps of filtering through the protocols stack

We are exploiting a vulnerability present in both Linux and Windows operating systems. This vulnerability makes the kernel of a machine in promiscuous mode respond to an ARP request with a bogus destination address.

This bogus MAC destination address is selected via an analysis of the behavior of Linux and Windows against ARP packets.

a) Case of LINUX: Since Linux is an open source operating system, we were able to examine its source code to know how ARP packets are processed.

The received ARP packets, after they bypass the NIC, are first received by the Ethernet module and then passed to the ARP module.

In the Ethernet module, the first thing to check is the group bit (the bit with the heaviest weight of the first byte of the destination MAC address), if it is set to 1 the address is classified as broadcast if it matches the broadcast address (`FF:FF:FF:FF:FF:FF`); otherwise it is classified as multicast. (The vulnerability is at this level; in fact, the Ethernet module checks only the first bit regardless of the other bits of the destination MAC address.) If the group bit is not set to 1, the address is classified either other host if it does not matches the local Mac address or `to\_us`, broadcast or multicast by the Ethernet module.

This is the reason why our test uses a fake destination MAC address with the group bit set; this address should not be set to an already existing Ethernet address (broadcast or multicast). The flowchart of figure 3 explains how the two modules work.

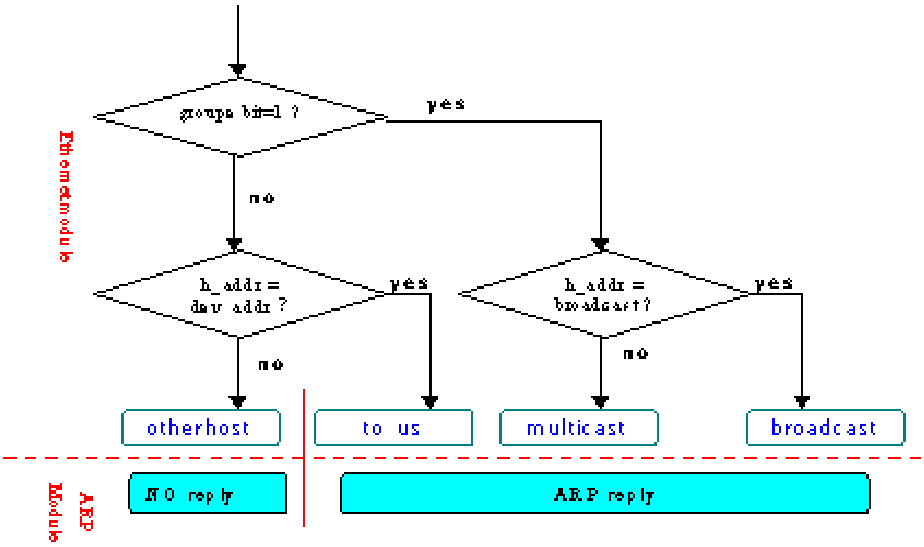

Figure 3: The Ethernet module working in Linux kernel

b) Case of Windows: Since Windows is not open source, finding out how packets are processed is more difficult. We decided to experimentally verify the test suggested by Promiscan [19].

MAC address	Why?
FF-FF-FF-FF-FF-FE	To see if the OS check all bits of the MAC address and whether it will respond
FF-00-00-00-00-00	To see if the OS check the first byte only
FF-FF-00-00-00-00	To see if the OS check the first word only
01-00-00-00-00-00	To see if the OS consider this address as multicast
01-00-5E-00-00-00	To see if the OS consider this address as IP multicast

Table 1: destination MAC addresses choice in the ARP test.

c) Summary of the ARP based detection: After having analyzed the two systems, Table 2 summarizes the results of the various tests performed:

Ethernet Address	Win 9x/Me/XP(Home)	Win NT4.0/2000/XP(Pro)	Linux 2.2/2.4
FF:FF:FF:FF:FF:FE	X	X	X
FF:FF:00:00:00:00	X	X	X
FF:00:00:00:00:00	X		X
FF:00:00:00:00:00	X		X
01:00:5E:00:00:00			X
01:00:00:00:00:00			X

Table 2: Summary of the ARP based detection.

5.2. Decoy Method

The decoy method is based on the concept of deceit or honeypot. The idea is to design a tool which allows spreading out bait (false passwords, false user names), which are supposed to be especially attractive for the sniffer owner and await him to launch an attack by reusing the fake information spread out (knowing that nobody except sniffer owner knows these false passwords).

Currently, our decoy method includes:

- Using Telnet, FTP, and POP3 on one hand.
- And using DNS on the other hand.

5.2.1. Decoy Based on Telnet, FTP, and POP3

For example, an FTP decoy is set by:

- First, adding one or more accounts to an FTP sever, with enticing user name (root, administrator, CIO, . . .). The account should contain files whose names can attract an intruder. As a measure of security, this type of account will be granted the minimum level of access right.
- Second, once the bait is set, connections to the appropriate server (in this case FTP) are established by the detector. The sniffing part of the detector is launched at this stage. When the detector detects an attempt of reusing of the pair (password, user name) of a bait connection, it triggers an alarm and transmits the IP address of the suspect machine. The same technique is used in the case of Telnet and POP3.

We have privileged using real connections with real servers instead of using fictive connection and fictive servers for practical purposes [3]. Indeed, using fictive servers and connections may not stand in court of law.

5.2.2. Decoy based on DNS

This technique works by exploiting a behavior common to almost all password sniffers [16]. Numerous fake TCP connections are created on a network segment, expecting that a sniffer pick up on those connections and resolve the IP addresses of the nonexistent hosts (do a reverse DNS lookup). When this occurs, SnifferWall sniffs the DNS request to check if the target is the one requesting a resolution of that nonexistent host.

6. Architecture design

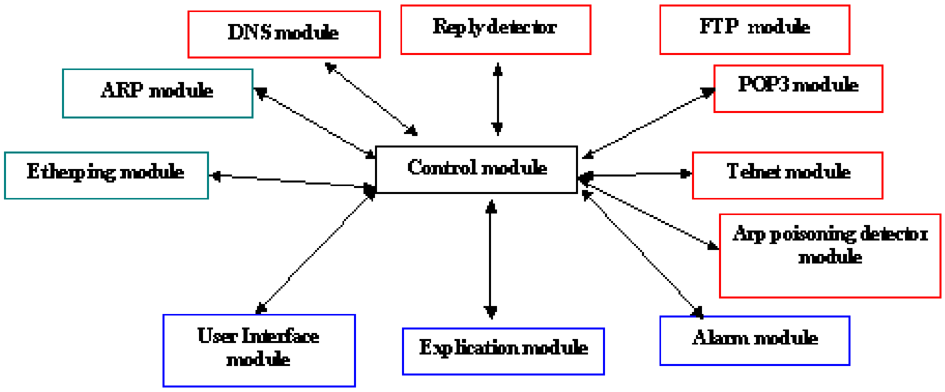

Figure 4: Architecture of SnifferWall

SnifferWall has an open architecture; it is divided into eleven modules (Figure 4).

The control module: The control module coordinates the activity of all the other modules. It answers their requests by starting the suitable modules. This type of reaction (module activation) can occur when a module announces its state to the control module. For example, when the user interface module sends a request of test beginning, the control module starts the module responsible for this test, suppose that is the ARP module. Once the test is accomplished, the ARP module announces the end of its task to the control module, when communicating the result of the test. If the result is positive the control module calls the alarm module to trigger alarm. Thus, the control module serves also as an intermediate if two modules would communicate between them.

The alarm module: It is responsible for setting off the alarm if a sniffer is detected after a test.

The explication module: It provides explanations to the user concerning the tests and the methods used for detection as well as operation of the detector.

The ARP module: It carries out the ARP test. It is composed of two sub-modules: ARPSend which is charged of sending ARP requests and ARPSniff which captures and filters packets for detecting a possible ARP reply.

The Etherping module: It is composed of two sub modules: *EtherSend* which is in charge of sending ICMP echo requests and *EtherSniff* which captures and filters the ICMP traffic to detect a possible ICMP echo reply.

The DNS module: It is comprised of two sub-modules: the first one generates false `ftp` and `telnet` sessions while the second one detects reverse DNS lookups.

The ftp, Telnet and POP3 modules: are respectively an `ftp`, `telnet` and `pop3` clients used to establish connections with the server.

The reply detector module: This module analyzes the network traffic, it collects the network packets, analyzes them, and announces possible attempts of reply, in real time, when running the decoy method. This module is equivalent to the A-Boxes and the E-Boxes of the CIDF (Common Intrusion Detection Framework) [17] model.

The detection process requires communication by the control module of a rule base containing:

- The type of the generated session (`pop3`, `ftp`, `telnet`).
- The IP address or the server name of the server (`ftp`/`telnet`/`pop3`) used.
- The (user name, password) pair.

Each time a new session is generated, a new input is added to the rule base by the control module. The flowchart of figure 5 illustrates how the reply detector operates.

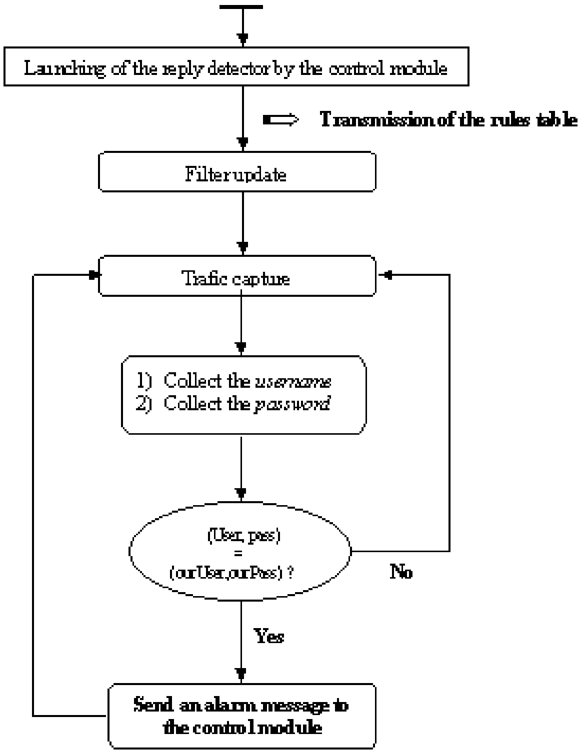

Figure 5: Reply detector state diagram

The user interface module: it allows a user to require that a set of tests be performed, as well as providing the necessary information (configuration of the machine, configuration of the scheduler, IP addresses to be tested) and it also shows the results of tests. Results of the tests are currently sent via email messages to the administrator.

7. Implementation

Currently, SnifferWall is implemented under Microsoft Windows. It is written in Visual C++ 6.0. Winpcap [11] and LibnetNT [18] are used to respectively capture the network traffic and to build packets.

8. Evaluation and Conclusion

Today, insidious attacks using sophisticated tools, and exploiting software vulnerabilities make Intrusion Detection Systems an essential component of a sound protection of information assets. Access control and authentication are no longer self sufficient. Sniffer detection is a crucial part of such protection mechanisms.

An integrated approach to sniffer detection has been proposed. Our tool, SnifferWall, reached its main objective of detecting the presence of a sniffer on an Ethernet segment. It combines searching for machine in promiscuous mode and using honeypot to detect potential use of sniffed information. Hence, SnifferWall covers online detection as well as after the fact or *information replay* detection regardless of the platform.

In addition, the detection based on MAC addressing makes it possible to detect any machine of the network which is in promiscuous mode for all Windows platform (9x/ME, NT/2000) or on Linux platforms (kernel / 2.0 to 2.4). According to the literature, our POP decoy is the very first implementation.

These results have never been reached, in the past, by a detector of sniffer, at least by a single sniffer detector.

Currently, we are planning the following extensions:

- Using mobile code to extend detection to a network made of several segments, so that sniffer detection can be distributed over an entire WAN, and the administrator can selectively scan part of the networks he/she administers.
- Extending the methods of detection based on MAC addressing on different platforms than Windows and Linux.

References

[1] B. Mukherjee, T.L. Heberlein, and K.N. Levitt. Network intrusion detection. *IEEE Network*, vol. 8, no. 3, pages 26–41, May/June 1994.

[2] S. Northcutt, J. Noval. *Network Intrusion Detection: An Analysts' Handbook*. New Riders Publishing, Second Edition, 2001.

[3] S. Grundschober. Sniffer Detector report. Diploma Thesis, IBM Research Division, Zurich Research Laboratory, Global Security Analysis Lab, June 1998. `http://packetstormsecurity.nl/UNIX/IDS/grundschober_1998.letter.ps.gz`.

[4] J. Drury. Sniffers: What are they and how to protect from them. November 11, 2000. `http://www.sans.org/infosecFAQ/switchednet/sniffers.htm`.

[5] J.S. Balasubramaniyan, J.O. Garcia-Fernandez, D. Isacoff, E. Spafford, D. Zamboni. An Architecture for Intrusion Detection using Autonomous Agents. CERIAS Technical Report 98/05, June 11, 1998.

David Isacoff, Eugene Spafford, Diego Zamboni

[6] D.E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, vol. 13, no. 2, pages 222–232, February 1987.

[7] http://www.tcpdump.org/

[8] http://reptile.rug.ac.be/˜coder/sniffit/sniffit.html

[9] http://www.ethereal.com/

[10] http://netgroup-serv.polito.it/analyzer/

[11] http://netgroup-serv.polito.it/windump/

[12] J. Downey. Sniffer detection tools and countermeasures. October 19, 2000. http://rr.sans.org/covertchannels/sniffer.php

[13] ftp://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/cpm/

[14] ftp://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/ifstatus/

[15] http://www.securitysoftwaretech.com/antisniff/

[16] D. Wu and F. Wong. Remote Sniffer Detection. Computer Science Division, University of California, Berkeley. December 14, 1998. http://spisa.act.uji.es/spi/docs/redes_doc/fredwong-davidwu.ps.

[17] S. Staniford-Chen. Common Intrusion Detection Framework (CIDF). http://seclab.cs.ucdavis.edu/cidf/.

[18] http://www.eeye.com/

[19] http://www.securityfriday.com/

[20] Riptech Internet Security Report, vol. I, January 2002, http://www.riptech.com/securityresources/form10.html. Vol. II, July 2002, http://www.riptech.com/securityresources/form_istr2.html.

[21] International Standards Organization. ISO 7498-2, Information Processing systems—Open System Interconnection—Basic Reference Model. Part 2: security architecture. Geneva, Switzerland, 1984.

[22] S. McClure and J. Scambray. Switched networks lose their security advantage due to packet capturing tool. Infoworld, July 22, 2002. http://www.infoworld.com/articles/op/xml/00/05/29/000529opswatch.xml.