Faculty of Natural and Applied Sciences

CSC316
Computer Security

Term Paper

Network Sniffers

Presented to Dr. M. Khair

As part of the requirements for CSC316 Spring 2002

By
Maria Ghosn Chelala
Pedro Maroun Eid
Assaad Sakha

Submitted: June 11, 2002

Abstract

Network Sniffers are tools used to watch over networks as well as collect all kinds of information including diagnostic information. In this paper we will introduce sniffers and discuss how they work, their advantages, their disadvantages and how to stop network sniffers being used by intruders. Code samples and examples are included.

Outline

Introduction

Network sniffers are tools used to monitor LANs, WANs and web services as well as to collect all kinds of data about what is going on over the network.

The sniffer was first introduced by Network General Corporation in 1988. Old sniffers were only able to give network administrators certain details about the packets being sent e.g. addresses and file sizes. This low-level information was and still is used to help administrators identify problems within the network hardware and applications. These days however, sniffers are able to provide high-level information and access every bit being sent. They can even tell who is logged in as well as which applications are being used. Sniffers can also give possible solutions to problems within the network. Sniffers are generally categorised as being one of the two following types:

- Stand alone products (usually in portable PCs) which can be plugged in to gather diagnostic data.
- Large hardware and software packages used to monitor networks at a high level.

The latter will be discussed in depth in this paper.

How Sniffers Work

During normal tasks such as Web surfing and messaging, computers are constantly communicating with other machines. A user should be able to see all the traffic traveling to or from their machine. Most PCs, however, are on a Local Area Network (LAN), meaning they share their connection with several other computers. Sharing means that computers can receive information that was intended for other machines. If the network is not switched, the traffic destined for any machine on a segment is broadcasted to every machine on that segment. This means that a computer actually receives the data traveling to and from each of its neighbors, but ignores it, unless otherwise instructed.

The sniffer program tells a computer's Network Interface Card (NIC), to stop ignoring all the traffic headed to other computers and pay attention to it. It does this by placing the NIC in a state known as *promiscuous* mode. Once a NIC is in promiscuous mode, a status that requires administrative or root privileges, all the data transmitted on its segment can be seen by it. The program then begins a constant read of all information entering the PC via the network card.

The Inner Workings of a Basic Sniffer

A basic Ethernet sniffer works at the Ethernet level by filtering out all the packets to different sources. The packet header contains the proper address of the destination machine. Only the machine with the matching address is supposed to accept the packet. When the NIC in promiscuous mode it accepts all packets no matter what the packet header says. Then, the sniffer filters out all unreadable, unusable, and uninteresting data, and logs the packets that are interesting to disk. But, since most basic sniffers log to disk, you can very easily discover the existence of a sniffer on your computer by watching file access. Further techniques will be outlined later on in this paper.

What Does Sniffed Data Look Like?

It is easy to grasp the concepts discussed above by watching a sniffer in action. The information in the following example was derived using "tcpdump", a program that has been around for quite sometime and is available for many platforms. This particular snippet is an abbreviated exchange between a machine and the SecurityFocus Web server:

```
21:06:30.786814 0:1:3:e5:46:6b 0:4:5a:d1:46:ad 0800 650: 192.168.1.3.32946 > 66.38.151.10.80: P
[tcp sum ok] 1:585(584) ack 336 win 64080 <nop,nop,timestamp 608776 899338> (DF) (ttl 64, id
7468, len 636)
0x0000     4500 027c 1d2c 4000 4006 8074 c0a8 0103  E..|.,@.@..t....
0x0010     4226 970a 80b2 0050 54ac b070 78ef d6c3  B&.....PT..px...
0x0020     8018 fa50 c663 0000 0101 080a 0009 4a08  ...P.c........J.
0x0030     000d b90a 4745 5420 2f63 6f72 706f 7261  ....GET./corpora
0x0040     7465 2f69 6d61 6765 732f 6275 696c 642f  te/images/build/
0x0050     626c 6c74 5f72 645f 312e 6769 6620 4854  bllt_rd_1.gif.HT
0x0060     5450 2f31 2e31 0d0a 486f 7374 3a20 7777  TP/1.1..Host:.ww
0x0070     772e 7365 6375 7269 7479 666f 6375 732e  w.securityfocus.
0x0080     636f 6d0d 0a55 7365 722d 4167 656e 743a  com..User-Agent:
0x0090     204d 6f7a 696c 6c61 2f35 2e30 2028 5831  .Mozilla/5.0.(X1
0x00a0     313b 2055 3b20 4c69 6e75 7820 6936 3836  1;.U;.Linux.i686

21:06:30.886814 0:4:5a:d1:46:ad 0:1:3:e5:46:6b 0800 402: 66.38.151.10.80 > 192.168.1.3.32949: P
[tcp sum ok] 2363393025:2363393361(336) ack 1437810754 win 8616 <nop, nop, timestamp 899338
608766> (ttl 61, id 10825, len 388)
0x0000     4500 0184 2a49 0000 3d06 b74f 4226 970a  E...*I..=..OB&..
0x0010     c0a8 0103 0050 80b5 8cde 8401 55b3 4042  .....P......U.@B
0x0020     8018 21a8 0543 0000 0101 080a 000d b90a  ..!..C..........
0x0030     0009 49fe 4854 5450 2f31 2e31 2032 3030  ..I.HTTP/1.1.200
0x0040     204f 4b0d 0a41 6765 3a20 320d 0a41 6363  .OK..Age:.2..Acc
0x0050     6570 742d 5261 6e67 6573 3a20 6279 7465  ept-Ranges:.byte
0x0060     730d 0a44 6174 653a 2054 7565 2c20 3132  s..Date:.Tue,.12
0x0070     2046 6562 2032 3030 3220 3033 3a30 343a  .Feb.2002.03:04:
0x0080     3538 2047 4d54 0d0a 436f 6e74 656e 742d  58.GMT..Content-
0x0090     4c65 6e67 7468 3a20 3433 0d0a 436f 6e74  Length:.43..Cont
0x00a0     656e 742d 5479 7065 3a20 696d 6167 652f  ent-Type:.image/
0x00b0     6769 660d 0a53 6572 7665 723a 2041 7061  gif..Server:.Apa
```

```
0x00c0     6368 652f 312e 332e 3232 2028 556e 6978   che/1.3.22.(Unix
0x00d0     2920 6d6f 645f 7065 726c 2f31 2e32 360d   ).mod_perl/1.26.
```

The above excerpt shows two packets: an HTTP request by the client and the server's response. Note that the first few lines of each sniffed packet provide a summary of the transaction: timestamps, source and destination MAC addresses as well as source and destination IP addresses and several other bits of information. The numbered lines (0x00##) show the data transmitted by each packet in hexadecimal format. Additionally, an ASCII decode of the payload is located off to the right - a convenient feature for crackers and nosey neighbors watching you on the network (http://online. securityfocus. com/ infocus/ 1549).

The sniffers read the data packets on the network providing details about the addresses of the senders and receivers, file sizes, messages in the packets… it depends on what the person using the sniffer wants. Using graphs and text-based descriptions, sniffers help network managers evaluate and diagnose performance problems with servers, the network wire, hubs and applications. Now, sniffers can even decode data from all seven layers of the Open System Interconnection network stack and can often recommend solutions to problems. If a solution is still not found sniffers can drill into low-level activities.

Not all sniffers work based on the promiscuous mode, network administrators and engineers can hook up a sniffing device to monitor and analyze hardware faults like CRC errors, voltage problems, cable programs, "dribbles", "jitter", negotiation errors, and so forth. The device they use is called a capture device, which is used to capture and filter traffic then store them in a buffer. There are a couple of capture modes: capture until the buffer fills up, or, use the buffer as a "round robin" where the newest data replaces the old.

Detecting Sniffers

Detecting a sniffing device that only collects data but does not respond to any of it requires a physical check of all of the user's Ethernet connections by walking around and checking each Ethernet connection individually. Some sniffers require a Trojan, so it has to be located or the sniffer's log file has to be found. However, there are other ways to detect sniffing:

1) Ping Methods - There are two ping methods -

1- A message sent trough a TCP/IP with only an IP address but no Ethernet address will be discarded, unless the machine is running in promiscuous mode. So an echo request (ping) can be sent with an IP but with no Ethernet addresses. If an answer is received then a sniffer is present on the network

2- Bad IP header values might be used to generate an ICMP error. If a "local broadcast" like 255.255.255.255 or a "directed broadcast" like 10.0.0.255 is sent to bypass IP address filtering, and put an error in the headers this will probably cause an ICMP error. If no error is obtained, a sniffer might be present on the network.

2) ARP Methods

The ARP method is similar to the ping method, but an ARP packet is used instead. The simplest ARP method transmits an ARP to a non-broadcast address. If a machine responds to such an ARP of its IP address, then it must be in promiscuous mode.

3) DNS Method

Many sniffing programs do automatic reverse-DNS lookups on the IP addresses they see. Therefore, a machine in 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.

Advantages of Sniffers

Because network sniffers are able to monitor all traffic passing through a connection, they are very useful for monitoring and analysis of a specific network. Networks are becoming more and more complicated as they expand, and it's a very time consuming and tiresome task to pin point a problem. New technology for network sniffers now allows network administrators to capture, decode, and analyze packets in real time. With this technology, a system captures packets off the network, decodes them into human-readable format, runs the packet through an expert system for analysis, and finally displays the information to the administrator. Today a network administrator might be alerted to a network issue before users experience any significant problems. In EtherPeek NX, for example, packets can be grouped together by source address, destination address, port, conversation, and protocol tokens. With this feature, analyzing specific network communications no longer requires poring over logs and having hard time searching in a log file, but is as easy as a click of the mouse (http://www.infoworld.com/articles/tc/xml/01/12/03/011203tcpackets.xml).

Packet analysis tools also provide graphical representations and statistics. The peer map is an impressive feature that graphically shows which systems are communicating with one another and the volume of traffic they are passing, providing a quick, high-level overview of all traffic traversing the network. More detailed statistics are also available, such as the percentage of network traffic attributed to a specific protocol (including Routing Information Protocol, HTTP, NetBios), detailed statistics for a specific node, statistics for a specific protocol, summary statistics for the entire network, and historical statistics to compare present and past performance.

These tools can also act as rudimentary network intrusion-detection systems. Many include analysis modules for basic Internet attacks, such as Jolt IP attacks, Land TCP attacks, RipTrace attacks, Teardrop IP attacks, and WinNuke TCP attacks. Additional modules could include detailed analysis of HTTP, SMTP, POP3, FTP, and Telnet (http://www.infoworld.com/articles/tc/xml/01/12/03/011203tcpackets.xml).

Modern sniffers typically incorporate remote monitoring standards (Rmon and Rmon 2), which define a standard way for systems to automatically collect key performance data points such as resource utilization. Rmon-savvy sniffers can take constant readings on the health of network components and compare those readings against historical trends. If necessary, they can trigger alarms when traffic loads or performance delays surpass limits set by network administrators (intro&use.doc).

Among today's wide-ranging network analysis products is the Sniffer Total Network Visibility Suite from the Sniffer Technologies unit of Santa Clara, California-based Network Associates. Intended for companies that conduct business over the Internet, the application can generate reports about protocol and bandwidth usage when traffic over the public network stalls (intro&use.doc).

Disadvantages

As seen in the last paragraph, sniffers are a good tool for detecting and monitoring a Network. They are good tools to help network administrator test, analyze and predict network crashes before they occur. However, there's always a bad side to everything. When Sniffers or sniffer programs are available to hackers, crackers or even team mates, they can be of real help to them and a big threat to you. To start, a program cannot be installed unless you have access to the root of a certain operating system, which means you should have the required rights to install a program. A sniffer is a normal program that is installed as any other. The difference is that, in most countries, it is illegal to sniff "unowned" information, which therefore makes the use of a sniffer illegal. The people who are using such programs are accessing private data just to know what's going on or to get free access to different personal accounts. A sniffer can only sniff around where the victim's machine is linked. A team mate or co-worker or even your boss may want to sniff your data just to know and be prepared for any competition between you and him. He just needs to install the sniffer on his PC and since you and him are on the same network, the sniffer gathers all info with your IP address as the criteria. A hacker or cracker uses a sniffer for the possible causes of stealing server access accounts and snooping on the network activity of that specific network. A hacker usually installs a Trojan on the machine that the sniffer is aimed at and that Trojan then sends the collected data via the sniffer to the attacker.

Usually sniffers look for plaintext only. On a normal LAN there are thousands of packets exchanged by multiple machines every minute; a nice and interesting supply for any attacker. Anything transmitted in plaintext over that LAN is vulnerable: passwords, web pages, database queries and messaging are some. A sniffer can easily be customized to capture specific traffic like telnet sessions or e-mail. Once traffic has been captured, crackers can quickly extract the information they need: logins, passwords and the text (body) of messages. The users will likely never know that their packets have been sniffed since sniffers cause no damage or disturbance to a network environment. The sniffer taps and records the raw data flowing between the victim and other machines. This data is then filtered and scanned for specific constraints and keywords for example “username”, “password”, etc…

Only a few Internet protocols use encryption. E-mail is most often sent and retrieved as plain text, and the password needed to break into someone's electronic mailbox is very rarely encrypted. If encryption is used, a key logger can often be used to discover the password that unlocks the data. According to the Hacker’s Encyclopaedia written by practitioner LogikBomb, the first hundred bits of any connection to a remote site usually include a username and password in plain text (http:// www.sisna.com/ users/ Ryder/ hack.html). This is of great help to the attackers since they only have a small part to search for in the beginning of a connection to a remote machine. Sniffers are very hard to detect. Attackers don’t want to be discovered, so sniffers are good and useful tools for them. Sniffers do not pose a direct threat to your data in the common sense of the word i.e. viruses or malicious code. The threat lies in the fact that sniffers are network analyzers designed to monitor network traffic. An attacker or a hostile user can gather information that travels through the network. Usually, the presence of a sniffer on the network can indicate future, more serious attacks against the network. Information gathered through the usage of sniffers can be used for upcoming attacks. Further network compromises can lead to a complete data disclosure. [Network

Sniffers, Aleksandar Stancin for Help Net Security, Posted By: Benjamin D. Thomas, 7/25/2001 14:14]

Civilian tools that can sniff LAN traffic, even on networks supposedly protected from monitoring by network switches, are widely available for free via the Internet. Even more, free source code is available and once you got what it takes, you could write it yourself; the protocol (TCP/IP) is widely known and its architecture is published. An example of an attack over a Distributed Network is shown in NetTrack Whitepaper written by Underground Research Labs January 2000. The study describes the whole process. First, Distributed Network Sniffing and Attacks are also known as "Echelon." The recent developments in network sniffing and strong encryption allow adept programmers to implement remote sniffers that are able communicate to logging daemons on outside servers. Most of these sniffers are detectable, but a few use strong encryption, secure protocols, and/or promiscuous mode packet filtering which make them very hard to detect. Such "echelon" sniffers are a real threat to user who does not use encryption to secure transfers of information of the Internet. Since packet sniffing is, in essence, spying, many states have laws against sniffing; but these laws do not seem to stop the few who are sniffing today. After the sniffer captures a packet, it chops off all data preceding the IP header. After that, it checks the IP header for the protocol type. If the type matches a list of types to filter, it records what protocol was used, and chops the IP header off. Once all this is done, the sniffer records the packet and sends it to the logging daemon, if it passes an information filter. The most advanced of these sniffers use strong encryption. "Echelon -4-Dummies", a beginner's sniffer program, uses 256-bit CAST encryption and base 64 encoding before sending data off to the logging daemon. After this, the sniffer lets the TCP/IP stack handle the data, making the sniffer close to unperceivable. When the captured and encoded packet reaches the logging daemon, it is base 64 decoded, unencrypted, and logged to a disk file. The logging daemon is usually very basic. However, more sophisticated programmers implement a random protocol daemon. What this means is that the daemon listens on multiple ports, known by the sniffers, with different protocols. In addition, the sniffer chooses a random protocol and sends the packet to the logging daemon. This keeps things very unpredictable for the administrator trying to prevent sniffing on a network. The distributed attack is just like the above, except that the sniffers become Trojan Horses, and the logging daemon becomes the administrator program. From this administrator program, the user can talk to all of the Trojan Horses at the same time. With the Trojan Horses, from the administrator program, the user can start remote DoS (Denial of Service) attacks on a target that can be specified at run-time. This posses a real threat to businesses since a user can start a 10-system DoS attack on a web-storefront. The "Tribe Flood Network 2000", a good example of the distributed attack, uses all of these techniques, and the ones mentioned above: secure encryption, random protocols, and base 64 encoding. From the administrator program, the user can start remote exploit floods. Some of these floods are the TCP SYN flood, a UDP flood, a basic ping flood, and a flood called TARGA 3 [NetTrack Whitepaper, 2000]. Unfortunately, one of the most commonly networked OS's, Unix, is vulnerable to sniffer attacks since it sends login names and passwords in the clear both upon logging in and in ftp and telnet sessions. Finally, sniffers are a good tool for people who know how and when to use it. It can uncover important and sensitive information which often satisfy the attacker's curiosity.

Code Sample

In this section, a sample code written in C language will be described. The code represents a basic sniffer for TCP/IP protocol; a packet sniffer. The code is also updated to search for the “password” constraint in the raw data. It collects all the packets that only contain the keyword; all others packets are discarded.

The program starts by a generic definition of all classes and definitions that will be used. Most relevant are the time, network device controls, system input output and so on. Then the definitions and main (global) variables are defined which will be used throughout the program. An important definition should be the stack, and the protocol elements that need to manage the captured data. “TCPflags” is used to initialize the buffer that stores the parts of the TCP packet. “Servp” specifies the used port and application that should be observed by the sniffer. “ADD_NODE” is a function that adds an item to the list of items that have passed the constraint specified after the packet is captured from the network link. “GET_NODE” returns a node specified by the source IP and Port number and the destination IP and Port number. “ADDDATA_NODE” adds the filtered (useful) data to a text buffer that the attacker looks at in the end of the run cycle. “PR_DATA” formats the output text to look as nice as required or as they have been received. “END_NODE” writes the finished, formatted, and filtered node to a specified log file for future reference. “IDLE_NODE” detects and logs idle packets using “END_NODE”. “filter” is the function that is used to filter all incoming packets. First, it checks if it is of an IP type packet. If it is not then it is discarded. Then the layout is formatted and the packet is checked if it is ok. If the packet passes the filter, it is saved to text buffer that stores all useful information. “death” handles all the signals that are either logged or result in some failure. “do_it” is the function that opens the network interface and upon the receipt of a packet calls “filter” to take charge of it. “getauth” performs optional authorization for the user using the sniffer. This is used to prevent anyone but the installer to access the information caught by the sniffer. “main” is the driver program that is used to combine all the above functions and make a reasonable run sequence for the whole program. The sniffer code is appended at the end of this paper; in the appendix section.

Conclusion

As has been seen throughout this paper, sniffers can be useful tools but can also present great disadvantages if in the wrong hands. Therefore this paper will be concluded with some protection possibilities against unwanted sniffers.

- Active Hubs

  Active hubs will only send a packet to its specified destination system which means a sniffer cannot receive packets that were not originally intended for its address.

- Encryption

  In order to help keep the integrity and secrecy of data when sending it over a network, data can be encrypted. There are packages that will perform this task. Therefore if an intruder intercepts the data it will be encrypted and if the encryption is "good" it is highly unlikely that the intruder will be able to decrypt it. Some packages that do this are: "deslogin", "swIPe" and "Netlock".

- Kerberos

  The Kerberos system, which was developed by MIT, allows a user to gain access to files without exposing his or her password on the network, even encrypted. The user, after sending his identification, must obtain a ticket (like a token) and session key from the Kerberos server if he is authenticated. Basically the Kerberos server will provide the user with a session key so that the workstation can communicate with the ticket granting server (which also receives this session key from the Kerberos server). This is all encrypted under the user's password which is stored on the Kerberos server (the password is not sent on the network). The user then decrypts with his password and obtains the session key and the ticket. Now the user can communicate with the ticket granter and ask for tickets to access any services he needs. The user identification is verified. Tickets are only valid for a certain amount of time. They are unforgeable and they contain time-stamps (Pfleeger, 2000).

- One Time Password Technology

  The method of one time password can be used for protection against a sniffer that has intercepted a certain password. Passwords should generally be changed periodically. However on time password changes a password every time it is used. Instead of using an English word as the password, functions are used. The user knows the specific function. Then at login time, the system suggests an argument for the function, the user will calculate the outcome and enter it. If his answer matches the system calculations he will be admitted. This can however be tiresome.

- Non-promiscuous Interfaces

  When NIC cards are in promiscuous mode that is when they are sniffing. Some cards however do not support promiscuous mode. If all the cards on the network are of this type, then they cannot sniff. A tool called Gobbler can be used to check if a NIC card is able to go into promiscuous mode. Lists of non-promiscuous cards are available.

References

- Security In Computing, Charles P. Pfleeger, second edition, Reprinted 2000.
- NetTrack White Paper, UnderGround Research Labs, January 2000.
- Hacker's Encyclopaedia, LogikBomb, First Edition, 1996 (hackersdictionary.txt).
- Network Sniffers**,** *Aleksandar Stancin for Help Net Security,* 2001 (somerisks.doc).
- Sniffer FAQ, Christopher William Klaus, Internet Security Systems, Inc.(sniffers.txt), 2000.
- Sniffer Sample Code, "what is Ethernet sniffing.txt", unknown author, 1996.
- Network Sniffers, Alan Joch, 2001(Intro&Use.doc).
- http://www.infoworld.com/articles/tc/xml/01/12/03/011203tcpackets.xml, Mandy Andress, 2001 (get to know your network.htm).
- http://online.securityfocus.com/infocus/1549, Mathiew Tanase, 2002 (SecurityFocus Home infocus Sniffers what they are and how to protect yourself.htm).
- http://www.robertgraham.com/pubs/sniffing-faq.html, Robert Graham, 2000 (Sniffing (Network wiretap, sniffer) FAQ.htm).