Netman Version 1.2 Project Report

A Network Manager’s Workbench Application Set

Gary Ratterree • Udo Pooch, Ph.D., P.E. • Willis Marti
(garyr@cs.tamu.edu, pooch@cs.tamu.edu, willis@cs.tamu.edu)
Version 1.1a by Curtin University

May 1995

1.0 General Disclaimer

This program bears a copyright notice to prevent rights from being claimed by any other party. Texas A&M University intends that the executable version of this program be placed into the public domain and grants permission for its restricted use and distribution, provided that:

1) the source code, when provided, is distributed without changes (source code distribution is limited under agreement from Curtin University and is not in the public domain),
2) this copyright notice is retained in all copies of the source code, and
3) the program is distributed free of charge, and is not sold without written approval from Texas A&M University -- Department of Computer Science and Curtin University.

The program is distributed "as is." TEXAS A&M UNIVERSITY DOES NOT WARRANT THAT THE OPERATION OF THE PROGRAM WILL BE UNINTERRUPTED OR ERROR-FREE, AND MAKES NO REPRESENTATIONS OR OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. WE ARE NOT RESPONSIBLE AND CANNOT BE HELD LIABLE FOR DAMAGE CAUSED TO NETWORK SERVERS OR TO NETWORKS BY THIS PRODUCT. No support service will be provided unless special arrangements have been made to do so. Certain manufacturers and trade names are mentioned in this code for the purpose of describing their communications protocol. Such reference does not constitute an endorsement or recommendation of such equipment, but is provided for informational purposes only.

Earlier versions of Netman are Copyright 1993 by Curtin University. The source code for any version of Netman (including this version) may not be distributed without written permission from Curtin University.

2.0 Table Of Contents

3.0 List of Figures

4.0 Abstract

The Network Manager's Workbench project is composed of a set of applications for the management and analysis of computer networks. Part of the Network Manager's Workbench project is concerned with the monitoring of computer networks to provide network managers with real time information about the performance of their network. The Netman tool set comprises part of the monitoring component of the Network Manager's Workbench project.

The Netman tool set is currently composed of three application programs that run in the X-Windows graphical environment on several Unix platforms. The first program, Etherman, is used to monitor all traffic on a local Ethernet network. Etherman shows the network traffic in a graphical display where the network is represented as a circle with the hosts clustered around it. Network traffic is shown as colored lines connecting the hosts. The color of the lines represents the dominant protocol in use between the hosts. The lines increase in size as the amount of network traffic between the hosts increases. The second program, Interman, is used to monitor TCP/IP conversations from the local network to the rest of the Internet. Like Etherman, Interman displays the network in a graphical format. Each TCP/IP network is represented by a circle with lines connecting hosts that are communicating with each other. Interman also uses colored lines to represent different protocols on the network. The last application in the Netman tool set, Packetman, is used to capture and decode packets on an Ethernet network. Packetman uses a graphic user interface (GUI) to allow the user to easily capture, select and decode packets. Packetman includes a large number of filters to aid in gathering only desired network packets. Together, these three applications comprise a powerful real time network monitoring toolkit.

This paper describes the three Netman applications in detail and discusses the modifications that were made to them at Texas A&M University. Their use in network monitoring and management as part of the Network Manager's Workbench project is also discussed. The history of these applications and security concerns regarding their use is also covered in this report.

5.0 Introduction

Computer networks are a difficult technology to manage because of their complexity and large scale. Network managers must find and diagnose problems occurring across miles of wire and hundreds of computing devices. Because of this management challenge, it is desirable to have a set of tools that can reduce the complexity of managing a computer network. The Network Manager's Workbench project provides the network manager with a set of applications that can assist with the management of a large computer network. The Network Manager's Workbench has several types of tools to assist the network manager:

- *Network Monitoring Tools*: The Network Manager's Workbench provides facilities for the real time capture and analysis of network traffic. This allows the manager to watch problems as they occur. These tools can also be used to locate and isolate trouble areas on the network before they turn into major problems.
- *Network Data Analysis Tools*: Although the real time network monitoring tools provide some analysis of the data that they gather, it is also desirable to have long term, in depth analysis of the data. Network Manager's Workbench provides tools to analyze data for use in network simulation and long term performance studies.
- *Network Topology Builder:* The Network Topology Builder uses active methods such as ping and SNMP queries to automatically build a map of the local network topology. The map that is generated shows the logical topology of the network and can be a useful tool for keeping track of very large networks.
- *Network Simulation Tool (NetOracle)*: Often, changes are made to computer networks without a real understanding of the impact they will have on network operations. The Network Manager's Workbench project provides a set of simulation tools to allow network changes to be examined in detail before they are made to the actual network. The *NetOracle* simulation tools use data gathered by the network monitoring tools and data analyzed by the network data analysis tools.
- *Queuing Model Tool:* The Queuing Model Tool provides an analytical representation of the network to complement the network simulation tool. The Queuing Model Tool also helps the network manager analyze changes before they are made to the actual network. Because this

tool uses a mathematical model of the network, it avoids the lengthy run times needed to produce useful results from the simulation tools.

- *Reliability Analysis Tool:* This tool helps the network manager identify single points of failure in the network. Risk factors are also analyzed with this tool, so the manager can identify critical locations where redundant equipment needs to be placed.
- *Traffic Generation Tool:* The Traffic Generation Tool allows the network manager to place controlled loads on parts of the network to test the reliability and performance of network components under stress. This tool can also provide traffic data as an input to the Network Simulation Tool.

Together, these applications combine to form a powerful platform for network management and analysis.

The Netman tool set is an integral part of the Network Manager's Workbench project. The Netman tool set contains a set of three tools that provide the network monitoring services for the Network Manager's Workbench. Local Ethernet network monitoring, TCP/IP network monitoring, and packet capture and analysis can all be accomplished with the Netman tool set. The Netman tool set is composed of three tools that run in X-Windows on several different Unix platforms. The first tool, Etherman, provides real time local network monitoring. This tool can display any type of traffic seen on an Ethernet network [see 15, 33 and 35 for more information on Ethernet networks]. Because Etherman uses X-Windows, it can run on remote computers to display information on remote networks. The second tool, Interman, displays real time TCP/IP network connections from the local network to all other TCP/IP computers on the Internet. This tool can also be run remotely to allow monitoring of remote networks. The last tool, Packetman, provides real time packet capturing and analysis for the local network. Like the other tools, it can also be run remotely. Together, these three tools meet most of the network monitoring requirements of the Network Manager's Workbench project.

6.0 History of Netman and Network Manager's Workbench

The Netman application tool set was originally developed at Curtin University in Western Australia. Version 1.2 of the Netman tools was developed under license at Texas A&M

University in College Station, Texas. The Network Manager's Workbench project has been completely developed at Texas A&M University.

6.1 Netman Version 1.1 and 1.1a

Etherman Version 1.1 was developed in 1991 at Curtin University by George Benko, Gregory Barron, Mike Schulze, and Craig Farrell as part of the *Netman* project [2, 16, 32]. This version runs on *Irix 4.x*, *Ultrix 4.x* and *SunOS* 4.x, and *Solaris 2.x*.

6.2 Netman Version 1.2

Version 1.2 was developed at Texas A&M University by Gary Ratterree, Udo Pooch, Ph.D., P.E., and Willis Marti. This version includes several modifications to the user interface to make the program easier to use. Command line parameters were added to the program to allow it to be used from shell scripts and other automated programs. A secure mode of operation has also been added, allowing the program to be used for classroom instruction without compromising the security of the network. Version 1.2 runs under *Irix 4.x*, *SunOS* 4.x, *Ultrix 4.x*, and *Solaris* 2.x. A launch application for Etherman written in Tcl/Tk is also provided with the version 1.2 distribution [21].

6.3 History of the Network Manager's Workbench Monitoring Efforts

The Network Manager's Workbench project initially sought to use network data collection only as a method to provide input to the simulation tools that were being developed as part of the project. It was quickly realized that the data collection effort was as useful to the network manager as the simulation results. It was at this point that the effort began to develop the Netman tools into part of the Network Manager's Workbench project. The initial efforts to capture data with the Computer Science Department's NAT Ethermeter proved to be an useful experience when modifying Netman. A detailed description of this effort is provided to show the other network monitoring methods that were tried by the Network Manager's Workbench group before settling on Netman [see 4, 13, 18, and 22 for descriptions of other network monitoring attempts].

6.3.1 Network Monitoring with a NAT Ethermeter

Initially a NAT Ethermeter model 250 was used to monitor the overall utilization of the Computer Science Departmental network at Texas A&M University. This device is a "smart box" that is attached to Ethernet networks. The Ethermeter has an extremely sensitive Ethernet port, on board microprocessor, and on board memory. The device passively listens and records traffic found on the Ethernet network. Because this is a physical layer device, the Ethermeter can only monitor one subnet of the network at a time. The information gathered by the Ethermeter is stored in its volatile random access memory for later retrieval. Data is stored in the Ethermeter in the Internet standard Remote Monitoring (RMON) Management Information Base (MIB) format. The data is collected through the use of Simple Network Management Protocol (SNMP) commands such as get and set [23, 30]. The use of the standard RMON MIB interface with SNMP commands means that the Ethermeter can be accessed by a wide variety of standard SNMP RMON compliant programs.

The NAT Ethermeter was used to monitor the utilization of several Ethernet networks in the Computer Science Department. A set of SNMP programs from Carnegie Mellon University was used to collect the utilization data from the Ethermeter. Because of storage limitations, the network utilization was averaged over a 30 second period by the Ethermeter. One data point was created every 30 seconds representing the average utilization during that period. The data collected from the Ethermeter was analyzed with a series of C programs, AWK scripts and SAS scripts to produce utilization graphs of the network. The network data collection process is show in Figure 6.3.1.1 and is described in detail in Hamilton *et al.* [8].

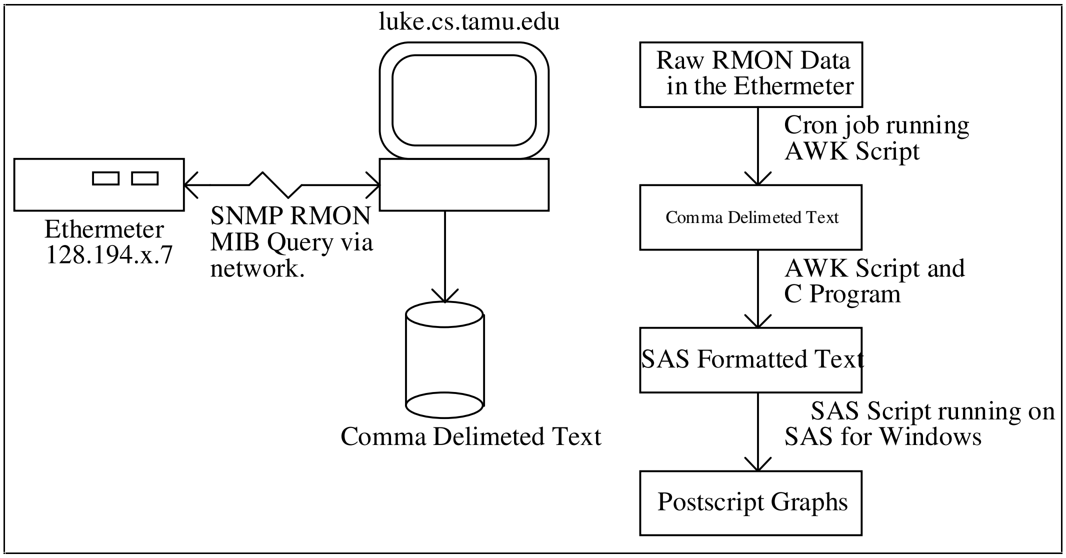


Figure 6.3.1.1. Ethermeter Data Collection Process.

This method of network data collection is quite accurate. Since it is a dedicated hardware device, the NAT Ethermeter is capable of accurately monitoring very heavy loads on the network. The disadvantage of this method is its extreme complexity. A single network utilization graph as shown in Figure 6.3.1.2 takes over an hour to produce using these techniques. Busy network managers often do not have the time or the resources to undertake a similar monitoring effort. A goal of the Netman project was to reduce the complexity and time required to produce similar graphs. The Etherman program meets this goal. Etherman has a strip chart at the top of its display that shows an utilization graph like that in Figure 6.3.1.2 in real time. The data collection and analysis from the Ethermeter often took several weeks to complete. Network problems had been fixed or had become much worse by the time the utilization information was available to the network management. The network manager can run Etherman and quickly get an idea of the performance of the network in real time. The flexibility of the Netman tools makes them much more useful for network management than the Ethermeter data collection process.

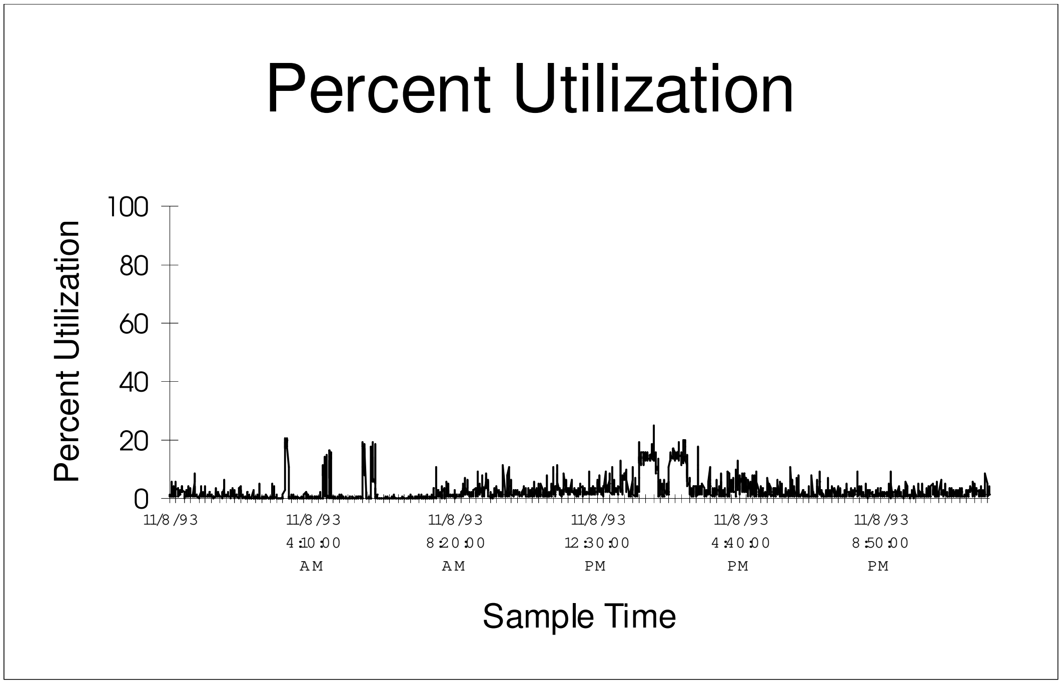


Figure 6.3.1.2. Sample Network Utilization Graph.

Figure 6.3.1.3 summarizes the TCP networks that were monitored during the initial phases of the Network Manager’s Workbench monitoring project. All of these subnets were graphed for use by the network administrators at the Department of Computer Science using the procedures described above [see 8].

Subnet	Start Time	End Time
132	10/29/93 at 5:35 PM	11/5/93 at 5:30 PM
133	11/5/93 at 6:21 PM	11/13/93 at 5:30 PM
134	11/13/93 at 6:11 PM	11/20/93 at 5:30 PM
135	11/22/93 at 11:09:25 AM	12/11/93 at 9:30 PM

Figure 6.3.1.3. Networks Monitored with NAT Ethermeter.

6.3.2 Data Collection of Traffic Between Individual Machines

Another monitoring effort was also conducted as part of the Network Manager’s Workbench project to get an idea of the types of communications that were taking place between machines on an Ethernet network. A NAT Ethermeter was placed into matrix mode to collect information about transmissions between pairs of machines. This information was collected from the Ethermeter with the Carnegie Mellon SNMP software. It was analyzed with a set of scripts similar to those used in the initial effort. Figure 6.3.2.1 illustrates the process that was used to collect and analyze this data.

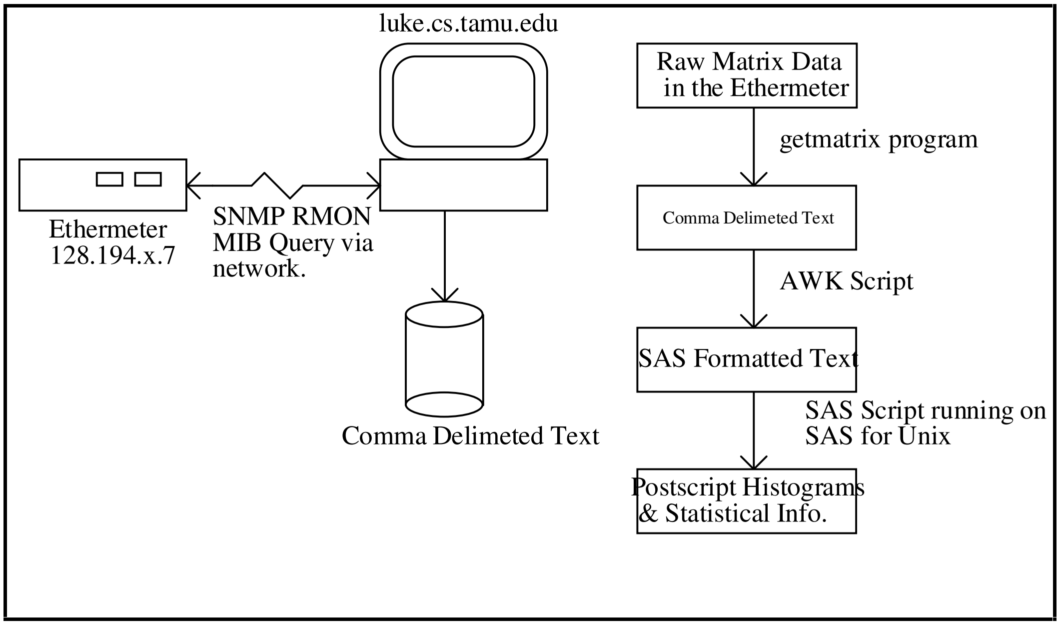


Figure 6.3.2.1. Ethermeter Individual Machine Data Collection Process.

This analysis effort was also very costly in terms of resources and effort required. Several days and many hours of computer time were required to produce graphs showing the traffic flowing between pairs of machines. This amount of effort to product a single graph was clearly unacceptable for a non-academic environment. The long delay between monitoring and production of results made this method unusable for network problem solving. The network administration wanted monitoring results for troubleshooting in minutes instead of days and hours. Figure 6.3.2.2 shows what a typical graph of a hour of traffic between a source/destination pair looks like.

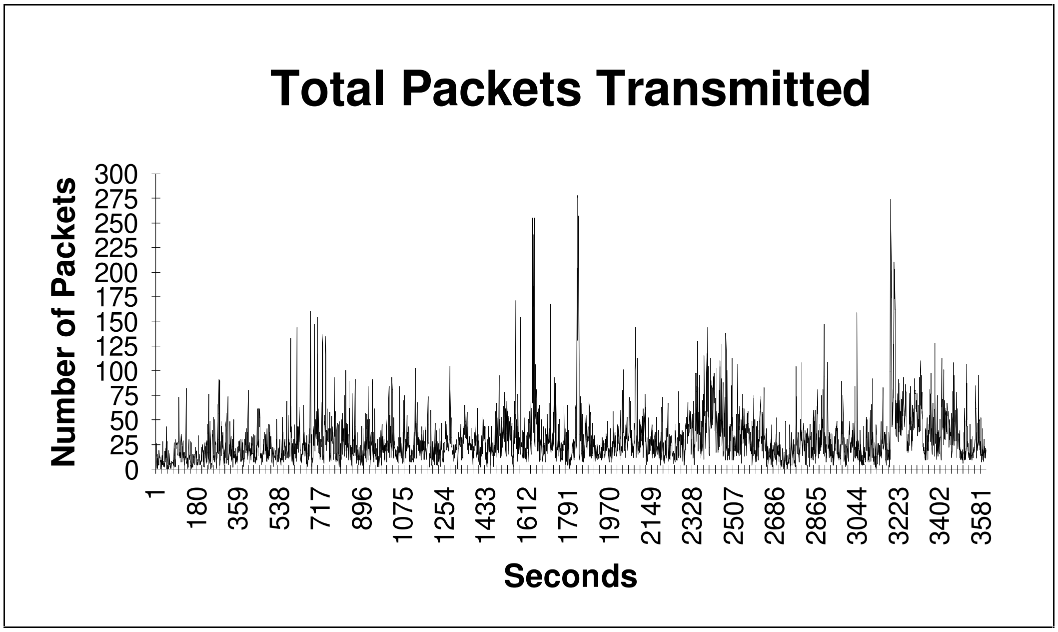


Figure 6.3.2.2. Source / Destination Graph Example.

With Etherman and Interman, the network manager can see the hosts that are communicating with each other in real time. Etherman varies the size of the links and hosts to represent the amount of traffic that is being transmitted between each pair of hosts. All of the connections are color coded to represent the dominant protocol that is being used. A network manager using the Netman tools can quickly find and identify network problems at a glance without an excessive amount of effort.

7.0 Netman Overview

7.1 The Netman Programs

The Netman network monitoring toolkit is composed of three programs. Each of these programs can be used either independently or as part of the toolkit. Figure 7.1 summarizes the primary uses of the three Netman programs.

Application Name	Primary Use
Etherman	Monitoring all transmissions on a local Ethernet network.
Interman	Monitoring all TCP/IP transmissions into and out of a local Ethernet network.

Packetman	Capturing and analyzing packets from a local Ethernet network.

Figure 7.1.1. The Netman Applications.

7.2 Netman Launch Utility

The Netman toolkit is integrated by means of a Tcl/Tk script [21]. This script is started by typing `netman` at the command prompt. Once started, a window will appear with buttons that can be clicked to start each of the Netman utilities. This window is shown in Figure 7.2.1. Each of the Netman applications has its own Tcl/Tk script to allow the user to set the available options before starting the program. The Netman launch script allows the user to start any of the Netman application's launch scripts simply by clicking the left mouse button on the appropriate application's button. Clicking on the Quit button will cause the Netman launch script to stop running. The Netman launch script does not have to be running to use the Netman applications; however, it provides a convenient way to start the programs.

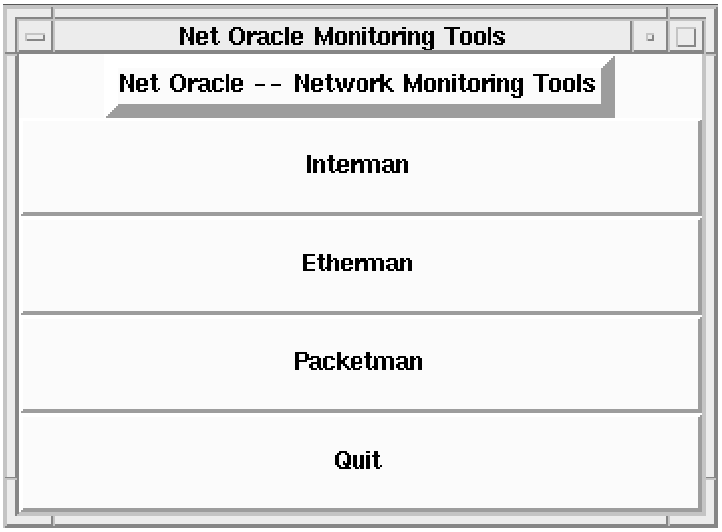
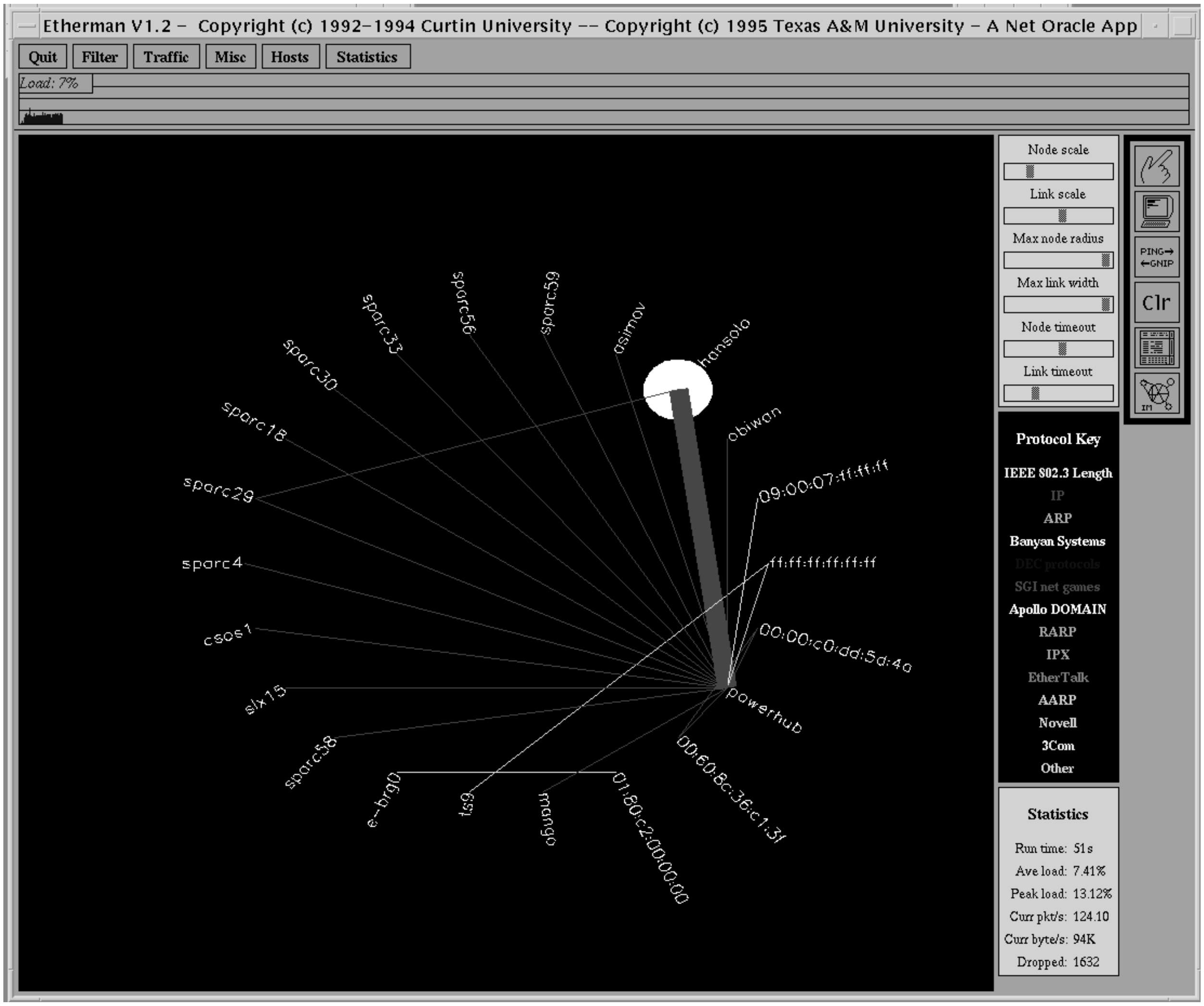


Figure 7.2.1. Netman Launch Utility.

Each of the Netman program's launch scripts can also be started from the command line, bypassing the Netman Tcl/Tk launch utility. The commands `etherman.tcl`, `interman.tcl`, and `packetman.tcl` will start the launch scripts for Etherman, Interman, and Packetman respectively. The Netman programs can also be started directly from the command line. The commands `etherman`, `interman`, and `packetman` will start the Netman tools directly without the use of a launch script. Each of the Netman applications has a large number of command line switches that can be used to set various program options on startup. These switches and their use are documented in the respective program's User's Guide.

8.0 Etherman

8.1 Etherman Overview



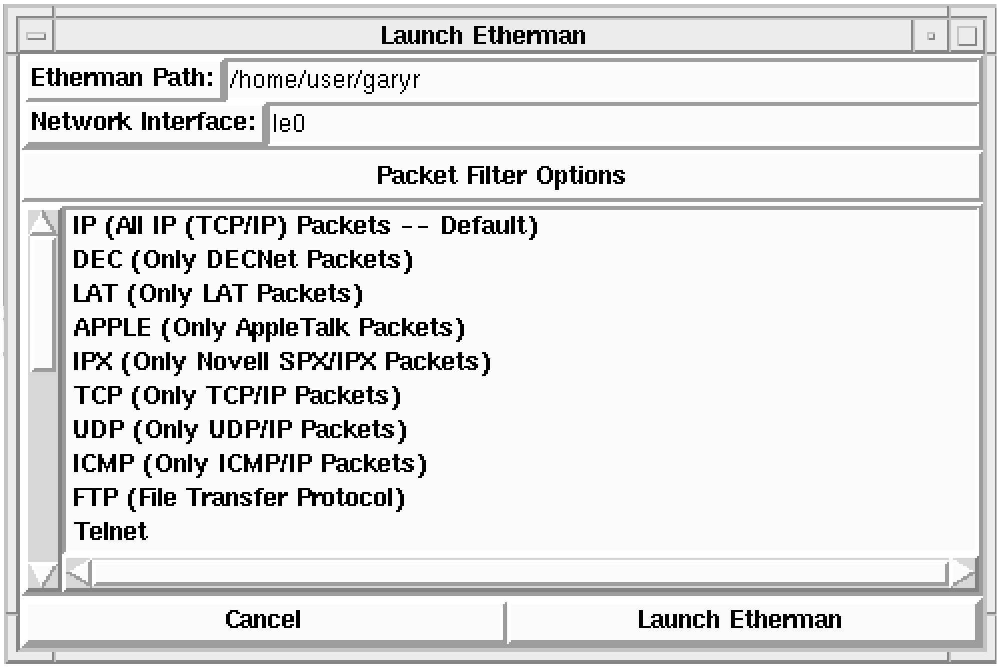
Statistics

Run time: 51s
Ave load: 7.41%
Peak load: 13.12%
Curr pkt/s: 124.10
Curr byte/s: 94K
Dropped: 1632

Figure 8.1.1. Etherman Display.

The Etherman application is designed to monitor traffic on a local Ethernet network in real time and display the information in an easy to understand format. Etherman runs on Unix workstations under the X-Windows system. The program places the workstation's network interface in promiscuous mode and uses the interface to monitor traffic on the local network. The network is represented as a large circle with hosts placed around it. Figure 8.1.1 shows a typical Etherman display. Each host is connected by a colored line to the other hosts that it is communicating with. These lines are colored to represent the primary or dominant protocol in use between the two hosts. As the traffic between two hosts increases, the line connecting them will increase in size. Each host will also expand in size to show the total amount of traffic that it is transmitting on the network. The Etherman display is constantly updated in real time to allow the network manager to monitor the traffic on the network as it occurs.

Etherman also monitors the overall utilization of the network that it is monitoring. A strip chart is located along the top of the Etherman display to show the network utilization history. This history chart is updated once per second to show the amount of network bandwidth that is being consumed. The program also displays information about the current network utilization in a text box at the bottom right side of the display. This box displays the average utilization of the network and the peak utilization that has been seen on the network since the program was started. The strip chart and utilization counters allow the network manager to watch for networks that are being over utilized.



Etherman Path: /home/user/garyr
Network Interface: le0

Figure 8.1.2. Etherman Launch Utility.

The Etherman program currently runs under the X-Windows systems and provides a convenient Graphical User Interface (GUI) for the user. There is a set of menu buttons located along the top of the Etherman display that can be used to control the various options of the program. There is also a toolbar located along the right side of the Etherman display. These tool buttons launch several Unix commands for the convenience of the network administrator. The scroll bars located above the protocol key allow the user to set the host and link display width and time-out settings. The use of the menu buttons, toolbar buttons, and scroll bars is documented in the *Etherman User's Guide*.

Like all of the Netman applications, Etherman has a Tcl/Tk script that can be used to set many of its options at startup. This script can be started from the Netman launch utility or directly from the command line. The Etherman launch script's display is shown in Figure 8.1.2. This GUI based launch application allows the user to set filter options that will be used when Etherman is started. The use of this launch script is documented completely in the *Etherman User's Guide*.

8.2 Etherman Technical Overview

Etherman is designed to monitor Ethernet networks for all types of traffic. In order to accomplish this, Etherman places the network interface of the workstation that it is running on in promiscuous capture mode. In this mode, the workstation passes Etherman all of the packets that are transmitted on the network. On an Ethernet network, all packets will be transmitted to the station that Etherman is running on. Since Etherman relies on being able to monitor all network traffic from a single station, it will not work on other kinds of network media.

Once in promiscuous mode, Etherman requests the workstation to buffer the packets that it receives in a small RAM buffer. When this buffer fills, Etherman reads it from the network interface and scans each packet to determine its type. Etherman uses the type/length field that is present in every Ethernet packet to determine the network protocol contained in the packet. Figure 8.2.1 shows the structure of an Ethernet frame. The length of each packet section in octets is shown in parenthesis. In 802.3 packets, the type/length field contains the length of the Ethernet packet [9, 14]. In Ethernet II packets, the type/length field contains the type of the packet [9, 14]. All values of this field less than 1518 are recognized as 802.3 length packets.

These packets are all displayed as the same protocol, since Etherman has no easy way of determining their origin. The Ethernet II packets rely on the fact that Ethernet packets cannot be longer that 1518 bytes to store the packet type in the length field. All values in the type/length field greater than 1518 identify the type of the packet. These type codes from Ethernet II or DIX Ethernet packets are decoded by Etherman to determine the protocol that the packet contains.

Preamble	SFD	Destination	Source	Type /	Data / Pad	FCS
(7)	(1)	(6)	(6)	Length (2)	(46 - 1500)	(4)

Figure 8.2.1. Ethernet Frame Format [9, 14].

In order for Etherman to determine the type of the packet, it uses a table of known values from RFC 1700 [29] to increment its protocol counters and update its display. Figure 8.2.2 shows the table that Etherman uses to decode the type values. Each type has a range of values that are used to match the type field. The decoding table also contains an X-Windows color constant value that is used to display the protocol on Etherman's window. The entry Other is used to match any packet that cannot be explicitly decoded.

```
0x0000 0x05DC   \"IEEE 802.3 Length\"   Pink \
0x0800 0x0800   \"IP\"                  Red \
0x0806 0x0806   \"ARP\"                 Green \
0x0bad 0x0baf   \"Banyan Systems\"      Yellow \
0x6000 0x6009   \"DEC protocols\"       Blue \
0x8014 0x8014   \"SGI netapps\"         brown \
0x8019 0x8019   \"Apollo DOMAIN\"       PeachPuff \
0x8035 0x8035   \"RARP\"                Cornflowerblue \
0x8037 0x8037   \"IPX\"                 LimeGreen \
0x809b 0x809b   \"EtherTalk\"           Magenta \
0x80F3 0x80F3   \"AARP\"                orange \
0x8137 0x8138   \"Novell\"              Cyan \
0x9001 0x9003   \"3Com\"                aquamarine \
0x0000 0xffff   \"Other\"               Grey70",
```

Figure 8.2.2. Etherman Protocol Decoding Table.

Etherman maintains a count of the number of packets that are traveling between each host in an internal data structure. It also maintains a count of the different protocols that are being transmitted between each host pair. A line is drawn on the Etherman display to connect hosts that are communicating with each other. Etherman uses the color values in Figure 8.2.2 to color the lines to represent the dominant protocol that is in use between the hosts. Etherman also changes the size of the line connecting the hosts and the size of the hosts to represent the amount of traffic that is being transmitted. A larger line or host size indicates an increased amount of traffic.

Because Etherman is designed to decode packets only from Ethernet networks, the program does no further decoding of the packet once its Ethernet type has been determined. The packets are discarded from memory, and Etherman waits until the packet receive buffer fills up again. Because Etherman does not completely decode the packets that it receives, it is much faster than Interman or Packetman. These programs must decode more of the packets they receive, and are slightly less efficient.

8.3 Modifications Made In Version 1.2

8.3.1 Etherman Secure Mode

Etherman requires promiscuous mode access to the Ethernet device in order to run correctly. On most Unix systems, only the supervisor account (root on most systems) can open the network adapter in promiscuous mode. Etherman can be run by ordinary users if the file permissions are set to allow it to run with privileges equivalent to those of the supervisory user. There are security risks created anytime a program is run by an ordinary user with supervisor privileges. Because of these risks, Version 1.2 of Etherman has been modified to run in a secure mode when it is started by an ordinary user (i.e. an user other than the supervisor or root account). The secure operating mode disables certain functions to prevent them from being used to circumvent the system and network security.

There were two major modifications made to Etherman to allow it to run in secure mode. The first modification involved changing the way the all of the functions that saved files to disk worked. These functions were changed to run with the permissions of the user that is using Etherman and not the supervisor account. These modifications prevent the user from overwriting or changing system files. The user can only overwrite files that would be ordinarily accessible while not running Etherman. The second modification involved restricting access to external programs while in secure mode. The user cannot use the telnet, finger, or ping utilities from the toolbar while in secure mode. This modification prevents the user from starting an external program with supervisor privileges.

With these secure mode modifications, Etherman can be run by ordinary users without allowing them to compromise the network's security. Because Etherman is a powerful tool, system administrators should still carefully consider the requests of users to run it. Individual users should be trusted by the system administration before they are allowed to use any network monitoring tool. A clear policy should be in place that covers the release of information gathered by network monitoring tools. With careful supervision, Etherman can be a valuable tool that both ordinary users and network administrators can use.

8.3.2 Command Line Switches

Etherman is primarily controlled though the use of a graphical user interface (GUI). Because it is often convenient to set program options from the command line before starting the program, Version 1.2 of Etherman was modified to allow control with command line switches. These switches can be used to set various program settings and packet filter options before starting the program. The command line switches can also be used to start and control Etherman from other programs. The Etherman launch utility makes extensive use of the command line switches to control how Etherman starts. The syntax for using Etherman's command line switches and their functionality is covered extensively in the *Etherman User's Guide*.

8.3.3 Miscellaneous Modifications

Version 1.2 of Etherman also contains several miscellaneous modifications to the program. The first miscellaneous modification that was made fixed the reporting features of Etherman. Etherman can print three different types of network traffic summaries to disk. The first two types of summaries were included with version 1.1 of Etherman. The format of these summary reports was modified to make them easier to read and understand. Figure 8.3.3.1 shows a part of the long format network traffic summary. This report shows the destination and type of all traffic seen on the local Ethernet network. A shorter format of this report is also available.

```
Node Name visiris2 ( Silicon-Graphics )
     Packets to Node powerhub
          TCP_LOGIN pkts = 22 (1.29 K)

 Node Name hansolo ( Sun )
     Packets to Node sparc29
          Other_UDP pkts = 17 (2.14 K)
          UDP_DOMAIN pkts = 10     (0.83 K)
```

```
     Packets to Node powerhub
          Other_TCP pkts = 264      (166.71 K)
          TCP_TELNET pkts = 1 (0.13 K)
          UDP_nfs pkts = 1    (0.08 K)
          UDP_SUNRPC pkts = 1 (0.12 K)
```

Figure 8.3.3.1. Etherman Network Traffic Summary.

A new network traffic report was created in Etherman Version 1.2 to show the utilization of the local Ethernet network. The utilization numbers that are reported are the same as those used to create the strip chart at the top of the display. Figure 8.3.3.2 shows an example of this new report. The first column of numbers is the number of seconds that have elapsed since Etherman was started. The second column of numbers reports the percent utilization of the bandwidth of the Ethernet network. This report contains up to the last hour of Ethernet traffic (a maximum of 3600 values).

```
0    1.529
1    41.320
2    1.001
3    2.001
4    7.648
5    4.706
6    4.234
7    4.424
8    4.119
9    5.786
10   12.070
```

Figure 8.3.3.2. Etherman Network Load Summary.

A button bar was also added to Etherman Version 1.2. This tool bar contains buttons that allow the network administrator to run the Unix ping, telnet, and finger commands on hosts on the network. There are also buttons to clear the Etherman display and start the other Netman applications (Interman and Packetman). This button bar is consistent with the button bar found on Interman. Several of the buttons are restricted in secure mode to prevent their misuse by ordinary users. The use of the Etherman button bar is covered in detail in the *Etherman User's Guide*.

Several other minor modifications were also made to Etherman Version 1.2. The packet filter menu was rearranged and several new packet filters were added to make it consistent with the rest of the Netman applications. This menu was modified to make check marks appear next to the filters that are currently selected. In previous versions of Etherman, the user was unable to determine which filters were currently active. With the check marks, it is now easy to see which filters are active. The ability to enter custom Ethernet, TCP/IP, and UDP/IP packet filters was also added. The user can enter a custom frame ID or a custom TCP or UDP port number to filter on [see 6, 28 and 36 for information about how TCP and UDP ports are used in application programs]. This extension allows Etherman to be used to filter for any network protocol that it does not currently support as a native filter selection.

These miscellaneous modifications all help to make Etherman Version 1.2 easier to use than the previous versions. With the addition of Secure Mode and the command line switches, Version 1.2 is much more flexible than earlier versions. This new version provides a very powerful and easy to use way for network administrators to monitor the activity of their Ethernet networks.

9.0 Interman

9.1 Interman Overview

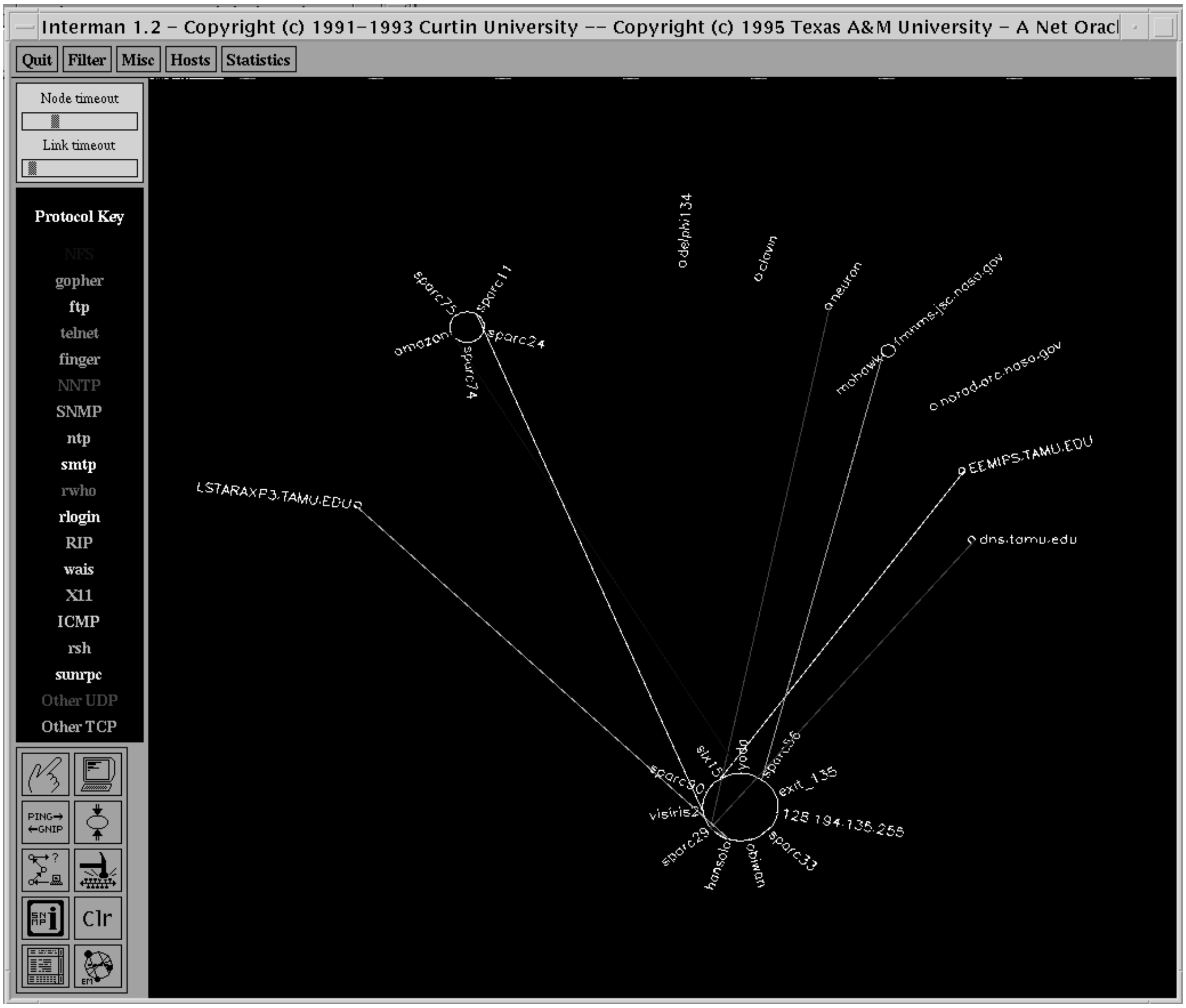
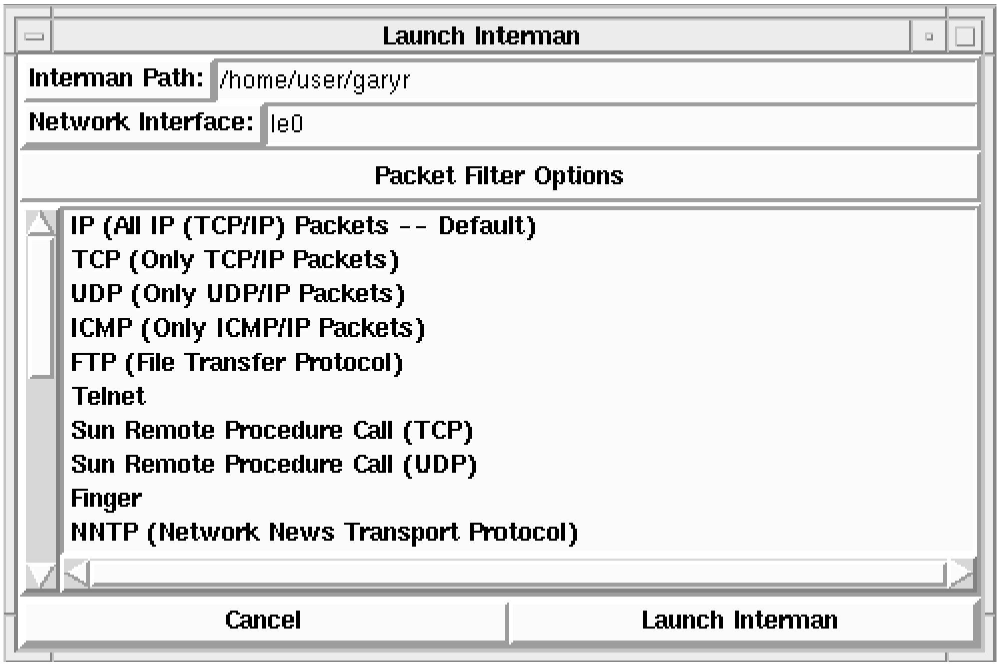


Figure 9.1.1. Interman Display.

Interman is the second application in the Netman toolkit. Interman is used to monitor TCP/IP traffic that is traveling into and out of the local Ethernet network. Like Etherman, Interman runs on Unix workstations under the X-Windows system. Interman also places the workstation into promiscuous capture mode, and it uses the workstation’s network interface to capture packets. The TCP/IP connections are displayed on a convenient graphical display in real time.

The Interman display represents each TCP/IP network as a circle. The hosts in each network are placed around the outside of the network circle. TCP, UDP, and ICMP connections between hosts are represented with colored lines. The color of the line represents the dominant protocol in use between the hosts. Interman does not increase the line or host width to indicate network traffic like Etherman. Hosts on the display can be shown either by name or by IP address.

Interman uses a convenient graphical user interface (GUI) that allows the user to change the program's settings with the mouse. A set of menu buttons is located along the top of the Interman window. These menu buttons allow the user to change the program's options. A set of tool buttons is located at the bottom left side of the display. These buttons allow the user to perform common operations and to start several Unix commands. A set of scroll bars is provided above the protocol key to allow the user to change the host and link time-out settings. The operation of all of the Interman controls is covered in detail in the *Interman User's Guide*.


Launch Interman

Interman Path: /home/user/garyr

Network Interface: le0

Packet Filter Options

IP (All IP (TCP/IP) Packets -- Default)
TCP (Only TCP/IP Packets)
UDP (Only UDP/IP Packets)
ICMP (Only ICMP/IP Packets)
FTP (File Transfer Protocol)
Telnet
Sun Remote Procedure Call (TCP)
Sun Remote Procedure Call (UDP)
Finger
NNTP (Network News Transport Protocol)

Figure 9.1.2. Interman Launch Utility.

Interman is also provided with a Tcl/Tk launch script. The script allows the user to set Interman's filter options before the program is started. This script can be started from the

Netman launch script or from the command line. The Interman launch script display is shown in Figure 9.1.2. The operation of this script is covered in detail in the *Interman User's Guide*.

9.1 Interman Technical Overview

Interman is designed to decode all Internet Protocol (IP) traffic on a local Ethernet. It currently decodes Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) packets to display IP based traffic in a similar manner to Etherman. Interman also uses a simple packet decoding scheme for maximum speed and efficiency. Interman first places the workstation's network adapter into promiscuous mode. Promiscuous mode monitoring allows Interman to capture all traffic that is present on the local Ethernet network. Since Interman is only interested in IP based traffic, it rejects all Ethernet packets that do not have a type of 2048 [see 29]. Ethernet packets that have a type of 2048 are collected in a buffer to be decoded.

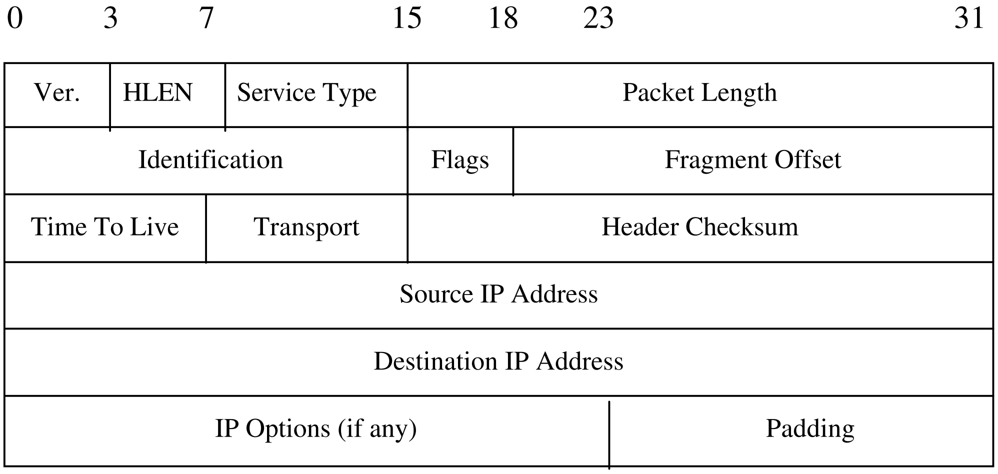


Figure 9.2.1. IP Protocol Header [5, 9, 25, 31].

Each IP packet has a Service Type field that determines the protocol that the packet is carrying. Figure 9.2.1 shows the format of an IP packet header, including the Service Type field. Interman decodes the Service Type field and calls the appropriate decoding function. For TCP and UDP packets, Interman decodes the UDP or TCP source and destination port numbers. Usually only one of these port numbers will match a distinguished port number. ICMP packets are not decoded any further, so they are simply counted. The UDP and TCP packets are compared to Interman's packet decoding table to determine their type. Packets that do not match an entry in the table are grouped into the "Other" category.

```
17          2049    2049      \"NFS\"         Blue \
 6            70      70      \"gopher\"      MistyRose4 \
 6            20      21      \"ftp\"         Cyan \
 6            23      23      \"telnet\"      Magenta \
 6            79      79      \"finger\"      LimeGreen \
 6           119     119      \"NNTP\"        brown \
```

Figure 9.2.2. Interman Protocol Decoding Table.

Figure 9.2.2 shows a portion of the Interman protocol decoding table. This table has five columns. The first column is the IP service number of the packet. Service 17 is UDP and service 6 is TCP. ICMP (service 1) is also decoded by Interman. The next two columns represent the range of port addresses that match a particular service. Most services only use one port, so both numbers are the same. The next column is the name of the protocol, and the last column is the color code used to represent the protocol. Like Etherman, Interman colors the links between hosts in the color representing the dominant protocol in use.

Interman maintains structures in memory that track the number of packets and their protocols that pass between each host on the local TCP/IP network. These structures are used to construct the Interman display. Interman does not show the amount of traffic that is passing between hosts. Since the IP traffic does not represent all of the traffic on the network, it was felt that this functionality would not be useful. Also, the Etherman application can already be used to measure the amount of all traffic on the local network (including IP packets).

Network Class	Range of First Octet	Possible Addresses
A	1 - 126	16,777,214
B	128 - 191	65,534
C	192 - 223	254

Figure 9.2.3. IP Network Address Classes [10, 11, 14].

Interman displays each host in its own IP network. Each IP network that has been seen by the program is represented as a circle on the display. To do this, Interman must first determine which class of network each host resides in. As figure 9.2.3 shows, Interman can determine the network class by checking the range of the first octet of the IP address. Interman groups each host into its respective class A, B, or C network. For networks in the same local domain as the workstation, Interman uses the local netmask to determine how the local domain is subdivided. Interman will show the local domain as it is subnetted with the local netmask. Since Interman is a passive tool, it cannot determine how remote domains are subnetted. It also has problems with variable length networks. Section 13.3 discusses this problem in more detail. The network display features of Interman makes it a very useful tool for tracking connections to different locations on the Internet.

Since Interman also uses a simple packet decoding scheme to capture its packets, it is also more efficient than Packetman. It is less efficient than Etherman, since Etherman decodes the packets only at the Ethernet level. Interman must decode the packets up to the UDP and TCP port level. Since Interman runs on a wide variety of hardware, it captures all of the packets that are present on the network and discards those that are not based on the IP protocol. This is less efficient than relying on hardware packet filters, but it is more flexible. Because of this flexibility, it would be easy to extend Interman to decode other internetwork protocols.

9.3 Modifications Made In Version 1.2

9.3.1 Interman Secure Mode

Interman has also been modified to run in a secure mode when it is used by a user that is not the supervisor account (usually root). Interman's secure mode functions in a similar fashion to Etherman's. Interman restricts the use of most of the Unix networking tools from the button bar to prevent them from violating system security. Interman also only creates files with the permissions of the actual user. These measures help to prevent Interman from being used to violate system or network security.

Like the other Netman tools, Interman's use should be closely monitored even when used in secure mode. Any network analysis tool can provide information that could potentially be used for unauthorized activity. Network administrators should establish and maintain a clear policy regarding the information that is gathered with these tools.

9.3.2 Command Line Switches

Interman Version 1.2 includes a set of command line switches that can be used to control the operation of the program from the command line. These switches are extensively used by the Netman launch utility to control how Interman is started. The command line switches provide flexibility for the user by allowing Interman to be controlled from other programs. Complete instructions for the use and syntax of the command line switches can be found in the *Interman User's Guide*.

9.3.3 Miscellaneous Modifications

Interman has also received several miscellaneous modifications in Version 1.2. The button bar has been extended with the addition of buttons to start Etherman and Packetman. These buttons help to integrate the Netman applications into an unified toolkit. Also, the packet filter menu has had several new filters added to it. The filter menus have also been rearranged to make them consistent with the other applications in the Netman tool set. Each filter item now has a check mark next to it when it is active to indicate that it has been selected. These check marks makes it much easier for the user to see which filters have been selected. A mechanism has been added to allow the user to filter on custom TCP and UDP port numbers. These custom filters allow Interman to be used with any new protocol that the network administration needs to monitor. The cumulative and dynamic statistics display functions were also repaired to work correctly.

A new short protocol summary function was also added to Interman. Figure 9.3.3.1 shows an example of part of this new report. This report shows traffic between all of the hosts that have been seen on the local IP network. The number of bytes and number of packets transmitted between the hosts are shown in columns three and four. Hosts are reported by name and domain unless Interman cannot resolve their addresses. IP addresses are shown for those hosts that cannot be resolved. A long protocol summary report is also available.

```
aos.arl.army.mil                    sparc29   271  1
mx.nsi.nasa.gov                     sparc29   110  1
terp.umd.edu                        sparc29   154  1
HAMGATE.CC.WAYNE.EDU                hamgate   230  2
sparc64                             sparc33   82   1
ftcl01                              yoda      198  3
aos.arl.army.mil                    sparc29   238  1
```

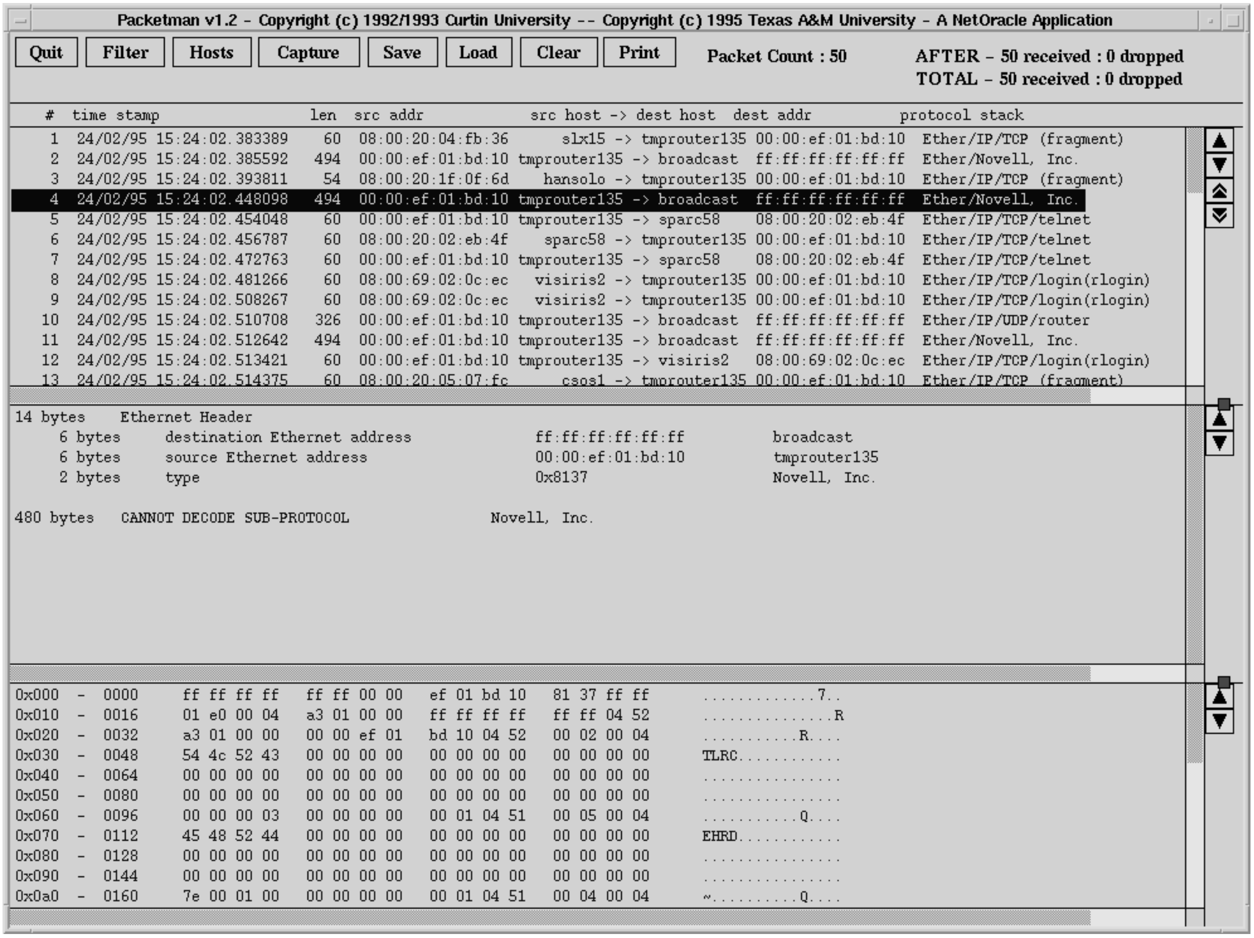
```
HUTIN2.SUGAR-LAND.ANADRILL.SLB.COM hamgate   386  5
colossus.apple.com                 sparc29   98   1
ns2.sprintlink.net                 sparc29   183  1
```

Figure 9.3.3.1. Interman Short Network Traffic Summary.

The Interman Version 1.2 enhancements help make the program a better management tool for network administrators.

10.0 Packetman

10.1 Packetman Overview



Packetman v1.2 - Copyright (c) 1992/1993 Curtin University -- Copyright (c) 1995 Texas A&M University - A NetOracle Application

Packet Count : 50

AFTER - 50 received : 0 dropped
TOTAL - 50 received : 0 dropped

```
 #  time stamp                len  src addr          src host -> dest host  dest addr          protocol stack
  1  24/02/95 15:24:02.383389   60  08:00:20:04:fb:36      slx15 -> tmprouter135 00:00:ef:01:bd:10  Ether/IP/TCP (fragment)
  2  24/02/95 15:24:02.385592  494  00:00:ef:01:bd:10 tmprouter135 -> broadcast  ff:ff:ff:ff:ff:ff  Ether/Novell, Inc.
  3  24/02/95 15:24:02.393811   54  08:00:20:1f:0f:6d    hansolo -> tmprouter135 00:00:ef:01:bd:10  Ether/IP/TCP (fragment)
  4  24/02/95 15:24:02.448098  494  00:00:ef:01:bd:10 tmprouter135 -> broadcast  ff:ff:ff:ff:ff:ff  Ether/Novell, Inc.
  5  24/02/95 15:24:02.454048   60  00:00:ef:01:bd:10 tmprouter135 -> sparc58    08:00:20:02:eb:4f  Ether/IP/TCP/telnet
  6  24/02/95 15:24:02.456787   60  08:00:20:02:eb:4f     sparc58 -> tmprouter135 00:00:ef:01:bd:10  Ether/IP/TCP/telnet
  7  24/02/95 15:24:02.472763   60  00:00:ef:01:bd:10 tmprouter135 -> sparc58    08:00:20:02:eb:4f  Ether/IP/TCP/telnet
  8  24/02/95 15:24:02.481266   60  08:00:69:02:0c:ec   visiris2 -> tmprouter135 00:00:ef:01:bd:10  Ether/IP/TCP/login(rlogin)
  9  24/02/95 15:24:02.508267   60  08:00:69:02:0c:ec   visiris2 -> tmprouter135 00:00:ef:01:bd:10  Ether/IP/TCP/login(rlogin)
 10  24/02/95 15:24:02.510708  326  00:00:ef:01:bd:10 tmprouter135 -> broadcast  ff:ff:ff:ff:ff:ff  Ether/IP/UDP/router
 11  24/02/95 15:24:02.512642  494  00:00:ef:01:bd:10 tmprouter135 -> broadcast  ff:ff:ff:ff:ff:ff  Ether/Novell, Inc.
 12  24/02/95 15:24:02.513421   60  00:00:ef:01:bd:10 tmprouter135 -> visiris2   08:00:69:02:0c:ec  Ether/IP/TCP/login(rlogin)
 13  24/02/95 15:24:02.514375   60  08:00:20:05:07:fc      csos1 -> tmprouter135 00:00:ef:01:bd:10  Ether/IP/TCP (fragment)
```

```
14 bytes    Ethernet Header
    6 bytes     destination Ethernet address           ff:ff:ff:ff:ff:ff         broadcast
    6 bytes     source Ethernet address                00:00:ef:01:bd:10         tmprouter135
    2 bytes     type                                   0x8137                    Novell, Inc.

480 bytes   CANNOT DECODE SUB-PROTOCOL               Novell, Inc.
```

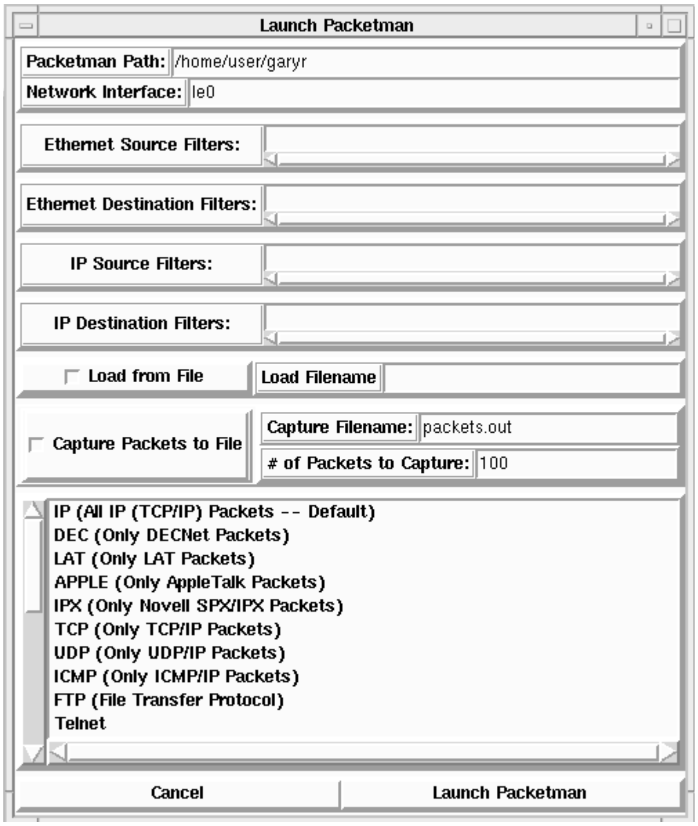
```
0x000  -  0000    ff ff ff ff   ff ff 00 00   ef 01 bd 10   81 37 ff ff     .............7..
0x010  -  0016    01 e0 00 04   a3 01 00 00   ff ff ff ff   ff ff 04 52     ...............R
0x020  -  0032    a3 01 00 00   00 00 ef 01   bd 10 04 52   00 02 00 04     ...........R....
0x030  -  0048    54 4c 52 43   00 00 00 00   00 00 00 00   00 00 00 00     TLRC............
0x040  -  0064    00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00     ................
0x050  -  0080    00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00     ................
0x060  -  0096    00 00 00 03   00 00 00 00   00 01 04 51   00 05 00 04     ...........Q....
0x070  -  0112    45 48 52 44   00 00 00 00   00 00 00 00   00 00 00 00     EHRD............
0x080  -  0128    00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00     ................
0x090  -  0144    00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00     ................
0x0a0  -  0160    7e 00 01 00   00 00 00 00   00 01 04 51   00 04 00 04     ~..........Q....
```

Figure 10.1.1. Packetman Display.

Packetman is the third Netman application. Packetman captures and displays protocol information about individual packets. The Packetman display is divided into four sections. The top section of the display contains the menu buttons. These buttons are used to control the various functions that Packetman can perform. Please see the *Packetman User's Guide* for detailed information about how to use the program's menu and other features. The next section displays all of the packets that are currently in Packetman's buffer. These packets are shown in groups of 200. The double arrow buttons on the right side of the display switch between sets of 200 packets. The scroll bars can be used to move inside the set of 200 packets. The user can select a packet to analyze from this list by clicking on it with the left mouse button. The packet will be highlighted in blue to indicate that it has been selected. The packet's decoded information will be shown in the third and fourth sections of the display.

The third section of the display shows the decoded protocol information from the currently selected packet. This information is as detailed as the protocol decoding routine permits. This display can be scrolled by using the scroll bars at the right side of the display. Currently most of the major TCP/IP protocols such as Telnet, Sun RPC, DNS, NFS, and NIS are decoded by Packetman. The decoding routines can be extended as needed by the user by writing new functions and linking them into the program.

The last section of the Packetman display shows an ASCII and hexadecimal dump of the currently selected packet. This dump is censored in secure mode and shows only zeros. The dump information is shown even if the packet cannot be decoded. This information can be useful for creating customized protocol decoding routines.



Packetman Path: /home/user/garyr
Network Interface: le0
Ethernet Source Filters:
Ethernet Destination Filters:
IP Source Filters:
IP Destination Filters:
☐ Load from File
Load Filename
☐ Capture Packets to File
Capture Filename: packets.out
\# of Packets to Capture: 100

IP (All IP (TCP/IP) Packets -- Default)
DEC (Only DECNet Packets)
LAT (Only LAT Packets)
APPLE (Only AppleTalk Packets)
IPX (Only Novell SPX/IPX Packets)
TCP (Only TCP/IP Packets)
UDP (Only UDP/IP Packets)
ICMP (Only ICMP/IP Packets)
FTP (File Transfer Protocol)
Telnet

Figure 10.1.2. Packetman Launch Utility.

Packetman Version 1.2 also includes a separate utility program that can be used to start or launch the program. This program allows the user to set many of Packetman's options from a convenient graphical user interface before starting Packetman. This utility is a Tcl/Tk script that builds a set of command line options and starts Packetman for the user. Figure 10.1.2 shows the Packetman launch utility.

The operation and use of Packetman is covered in much more detail in the *Packetman User's Guide*. This report gives just a brief overview of Packetman and some technical details regarding its operation. It is not intended to be used as an operator's manual for the program. Prospective users should also carefully read section 11.0 about security issues associated with this program.

10.2 Packetman Technical Overview

The Packetman program operates by capturing packets from an Ethernet network through a workstation's network interface. The workstation's network interface is placed in promiscuous mode by the Packetman program. While in promiscuous mode, the workstation captures every packet that is being transmitted across the network. The promiscuous capture mode allows Packetman to examine every packet on the network instead of just the packets addressed to the workstation. Because each hardware vendor uses a different type of network interface, Packetman must provide a custom network driver for each type of workstation that it runs on. Section 10.3.1.1 discusses the network interface drivers on Sun workstations in more detail.

The custom network driver simply captures every packet that it sees on the network from the workstation's network interface and places it in network byte order in a memory buffer. A count of the number of packets that have been captured is also maintained by the network interface procedure. Once the specified number of packets have been captured, Packetman reads the memory buffer and updates the display. The use of a memory buffer with a specific byte ordering allows Packetman to be used on a wide variety of Unix workstations. Each workstation has a completely different network interface procedure, but all of the procedures operate by filling the buffer in the same format. The packet decoding procedures are completely independent of the type of hardware that is being used.

After Packetman has captured the number of packets that the user has specified, it updates the display to show a list of the captured packets. The decoding routines are called with the global `FullyDecode` flag set to false. This flag instructs the decoding routines to append the name of the protocol to the end of a global string variable. This string is then shown with the size and address of the packet in the top pane. When a packet is selected by the user for decoding, the `FullyDecode` flag is set to true and the decoding routines print all of the information that they can determine about the packet in the middle pane of the display.

The packet decoding procedures follow a standard format, so they can be used with each other without any special knowledge of which decoding routing is being called. Each procedure is passed a pointer to the start of the packet and an integer that can be used to pass special information to subordinate decoding routines. The procedure decodes all of the information that it can from the packet and then calls the next decoding procedure. The next procedure is also passed a pointer to the packet and an integer. The pointer to the packet is always set to point at the point where the last decoding routine has stopped decoding. This is typically in the data area of the protocol header. Each decoding routine is responsible for determining and calling the next packet decoding routine that needs to be called. After the last decoding function returns, the ASCII and hexadecimal dump is printed and the packet decoding routine exits.

The packet decoding control flow can be illustrated by a simple example. In a IP packet, the Ethernet header is decoded by the first function. The `Decode_Ether` function is always the first function that is called by Packetman to decode a new packet. This function is passed a pointer pointing to the beginning of the packet. The Ethernet function decodes the Ethernet information based on the bits that are in various positions. Figure 10.2.1 shows the format of an Ethernet packet. The information about the Destination Address, Source Address, Type, and Frame Check Sequence (FCS) is printed in the middle frame on the Packetman display. After the packet's Ethernet header has been completely decoded and printed to the display, the `Decode_Ether` function must call the next decoding procedure.

Preamble	SFD	Destination	Source	Type / Length	Data / Pad	FCS

Figure 10.2.1. Ethernet Frame Format [9, 14].

Each Ethernet packet header contains a field that describes the type of packet. This field is labeled Type/Length in Figure 10.2.1. The Ethernet decoding function uses a table as shown in Figure 10.2.2 to look up the type of the packet and determine the next decoding routine to call (this figure is abbreviated for space).

```
{0x0800, 0x0800, Decode_IP,     "IP",             },
{0x0801, 0x0801, NULF,          "X.75 Internet", },
{0x0802, 0x0802, NULF,          "NBS Internet",  },
{0x0803, 0x0803, NULF,          "ECMA Internet", },
{0x0804, 0x0804, NULF,          "Chaosnet",       },
{0x0805, 0x0805, NULF,          "X.25 level 3",  },
{0x0806, 0x0806, Decode_ARP,    "ARP",            },
```

Figure 10.2.2. Packetman Ethernet Packet Lookup Table [29].

The Ethernet type lookup table returns a pointer to the next decoding function and a string identifying the type of the packet. In the case of an IP packet, the packet type field contains the value 0800 (this is a hexadecimal number). The entry in the table for 0800 returns a pointer to the `Decode_IP` function and the string "IP" identifying the packet's type. The `Decode_Ether` function will then call the `Decode_IP` function and pass it a pointer pointing to the start of the Data area of the Ethernet packet. The `Decode_IP` function does not receive any information about the Ethernet header. The table contains two numeric values for each type to accommodate network protocols that require a range of numbers to identify internal protocols.

0 3 7 15 18 23 31

<table>
<tr><td>Ver.</td><td>HLEN</td><td>Service Type</td><td colspan="3">Packet Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragment Offset</td></tr>
<tr><td colspan="2">Time To Live</td><td>Transport</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
<tr><td colspan="6">Destination IP Address</td></tr>
<tr><td colspan="5">IP Options (if any)</td><td>Padding</td></tr>
</table>

Figure 10.2.3. IP Protocol Header [5, 9, 25, 31].

Once the `Decode_IP` decoding procedure has been called, the packet decoding proceeds in a similar fashion to the `Decode_Ether` procedure. The IP header as shown in Figure 10.2.3 contains the service type of the IP packet. This service type is then compared to a table as shown in Figure 10.2.4 (this figure is abbreviated for space). If the type is 1, the `Decode_ICMP` function will be called, if it is 6, the `Decode_TCP` function will be called and so on. If no procedure is available for a protocol an entry with a null pointer is placed in the table. If the `Decode_IP` function encounters a null pointer, it simply returns and the protocol decoding is completed. The set of protocol decoding routines can be expanded simply by creating a new decoding function and adding its name to the appropriate decoding table.

```
{0, 0, NULF,            "reserved-0",   },
{1, 1, Decode_ICMP,     "ICMP",         },
{2, 2, NULF,            "IGMP",         },
{3, 3, NULF,            "GGP",          },
{4, 4, NULF,            "IP",           },
{5, 5, NULF,            "ST",           },
{6, 6, Decode_TCP,      "TCP",          },
```

Figure 10.2.4. Packetman IP Packet Lookup Table [29].

0 3 9 15 23 31

<table>
<tr><td colspan="3">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="5">Sequence Number</td></tr>
<tr><td colspan="5">Acknowledgement Number</td></tr>
<tr><td>HLEN</td><td>Reserved</td><td>Flags</td><td colspan="2">Window</td></tr>
<tr><td colspan="3">Checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="4">Options</td><td>Padding</td></tr>
</table>

Figure 10.2.5. TCP Protocol Header [5, 9, 27, 31].

Transmission Control Protocol (TCP) packets can also be decoded with the use of a table. These packets contain a source and destination port that corresponds to a specific protocol. One of the ports is always fixed based on the protocol and will match an entry in the table. The other port is random and will usually not match an entry in the table. These port numbers are available in RFC 1700 and several other references [6, 28, 29, 36].

```
{20, 20, NULF,                  "ftp-data"              },
{21, 21, NULF,                  "ftp"                   },
{22, 22, NULF,                  "Unassigned"            },
{23, 23, Decode_telnet,         "telnet"                },
{24, 24, NULF,                  "any private mail system" },
{25, 25, NULF,                  "smtp"                  },
```

Figure 10.2.6. Packetman TCP Packet Lookup Table [29].

As figure 10.2.6 shows, most of the TCP port numbers do not have decoding routines assigned to them. This table is just a small portion of the complete table in the Packetman program. The large number protocols in any networking system is a challenge for any protocol decoder to cover completely. By using a table driven system, Packetman allows new decoding routines to be easily added to the program as they are needed.

<table>
<tr><td>0</td><td>15</td><td>31</td></tr>
</table>

Source Port Number	Destination Port Number
Length	Checksum

Figure 10.2.7. UDP Protocol Header [5, 24, 31].

The User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) packet decoding functions are similar to the TCP protocol decoding function. These protocols also have Port and Type fields that can be used to create function lookup tables as shown in figures 10.2.8 and 10.2.10.

```
{2046, 2046, NULF,              "sdfunc"     },
{2047, 2047, NULF,              "dls"        },
{2048, 2048, NULF,              "dls-monitor"},
{2049, 2049, Decode_NFS,        "nfs"        },
```

Figure 10.2.8. Packetman UDP Packet Lookup Table [29].

<table>
<tr><td>0</td><td>7</td><td>15</td><td>31</td></tr>
</table>

<table>
<tr><td>Type</td><td>Code</td><td>Checksum</td></tr>
<tr><td colspan="3">Miscellaneous</td></tr>
<tr><td colspan="3">IP Protocol Header and 8 further bytes or test data</td></tr>
</table>

Figure 10.2.9. ICMP Protocol Header [5, 26, 31].

The ICMP packets are used for network control messages on TCP/IP networks. The ICMP lookup table in figure 10.2.10 shows several of the types that Packetman can decode. The ICMP type numbers have been replaced with standard Unix constant identifiers in this table. The ICMP decoding routines have no subordinate protocols, so no pointers to other decoding routines are present in the table.

```
{ICMP_UNREACH_NET,        "Network Unreachable"             },
{ICMP_UNREACH_HOST,       "Host Unreachable"                },
{ICMP_UNREACH_PROTOCOL,   "Protocol Unreachable"            },
{ICMP_UNREACH_PORT,       "Port Unreachable"                },
{ICMP_UNREACH_NEEDFRAG,   "Fragmentation Needed and DF set"},
```

Figure 10.2.10. Packetman ICMP Packet Lookup Table.

Several protocols that are based on TCP/IP and UDP/IP have large numbers of decoding functions of their own. The Network File System is an example of this type of protocol. This protocol is based on UDP/IP packets. It also requires a SUN Remote Procedure Call (RPC) decoding procedure to be present. The RPC decoding function determines the type of NFS packet and calls the appropriate NFS decoding function. Packetman can decode all of the basic NFS procedure types. Figure 10.2.11 shows the types of NFS procedures that Packetman can decode.

NFS Procedure Number	NFS Procedure Name
0	Null Procedure
1	Fetch File Attributes
2	Set File Attributes
3	Read File System Root (obsolete)
4	Look Up File Name
5	Read the Contents of a Symbolic Link
6	Read a File
7	Write to Cache
8	Write to File
9	Create File
10	Delete File
11	Rename File
12	Generate Link
13	Generate Symbolic Link
14	Create Directory

15	Delete Directory
16	Read Directory
17	Read File System Attributes

Figure 10.2.11. NFS Protocol Specification [31].

Packetman uses lookup tables with pointers to decoding routines in order to decode packets. With this method, Packetman can decode a large number of protocols in an easy to understand manner. Expansion of the Packetman protocol decoding routines is easily accomplished by writing a new procedure based on the protocol decoding procedure standard and add a pointer to the procedure to the proper table. Since no protocol decoding program can keep up with the large number of protocols on the Internet, it is essential to have a mechanism to add new decoding routines.

10.3 Modifications Made In Version 1.2

Packetman received extensive modifications for Version 2.1. The primary modification to the program was porting it to run under the *Solaris 2.x* operating system on Sun workstations. A secure operating mode was also added to the program, allowing it to be used without compromising the security of the host workstation or the network. Several other modifications were made to the program. These include several new packet decoding routines and a packet printing routine. The ability to control most program functions from command line switches was also added.

10.3.1 Modifications Made For Solaris 2.x

The primary objective of Packetman Version 1.2 was to produce a package that would compile and run under both *SunOS 4.x* and *Solaris 2.x* on Sun workstations. Packetman version 1.1 compiles and runs correctly under *SunOS 4.x*, but will not run under *Solaris 2.x*. After the modifications were made, version 1.2 will correctly compile with the GNU GCC C compiler and run under both of the Sun operating systems. Since Solaris 2.x is System V Release 4 compliant, the new Packetman version should now be portable to a much wider range of Unix systems [see 17 for Solaris portability and porting issues]. Many of the modifications that were made will help improve the portability of the program code.

The majority of the changes made to Packetman were made in the networking code and the packet decoding code. The X-Windows display code received only minor modifications to run under Solaris 2.x. This code is basically unchanged from Version 1.1. However, several minor user interface improvements were made to the program. The user interface changes are described in section 10.3.4.

10.3.1.1 Network Interface Modifications

Each operating system that Packetman runs under requires a different method for accessing the network interface on the workstation. The Unix operating system does not provide a standard mechanism for accessing raw traffic from the network device. Because of the existence of programs like Packetman that require this capability, each vendor has provided a proprietary capability for directly accessing the network device. Since each vendor provides a different mechanism for accessing the network device, Packetman requires a different module for each type of workstation that it is going to run on. The majority of the effort in porting Packetman to a new operating system or hardware platform involves creating a driver to access the network hardware directly.

In *SunOS 4.x*, Packetman uses the NIT or Network Interface Tap pseudo driver to access the network device directly. The NIT driver is a proprietary extension to the *SunOS 4.x* operating system that allows user programs to directly access the network hardware [19]. Normally, user programs cannot access the network device directly and should rely on the TLI, socket, or RPC interfaces instead [19]. The NIT pseudo driver allows user level programs managed access to the network device low level drivers. Because NIT places a layer of software between the user program and the network device, it has a significant performance penalty. Programs that use the NIT interface can quickly degrade the performance of all network operations taking place on the workstation. Since NIT is the only way to access the network hardware directly in *SunOS 4.x*, Packetman is forced to use it regardless of the performance problems. Packetman has a procedure that uses NIT to gather packets from the network device and places them in a memory buffer in a common byte ordered format. A memory packet buffer is used with a common format, so the rest of the Packetman packet analysis and packet decoding code can be independent of the operating system and hardware that is being used. The *Solaris 2.x* operating system does not support NIT, so a new interface procedure had to be written for Version 1.2 of Packetman.

The *Solaris 2.x* operating system allows user programs to directly access the low level network hardware though the STREAMS Date Link Provider Interface (DLPI) [19]. The NIT interface is no longer supported, so *SunOS 4.x* programs that use it will not be able to run under *Solaris 2.x*. The DLPI interface is hardware independent and will work on Ethernet, FDDI, Token Ring, Token Bus, X.25, and many other datalink-level protocols [19]. The Packetman program currently only works with Ethernet, but could be extended to work with other DLPI compliant devices. A new driver procedure was created for Packetman Version 1.2 that uses DLPI to place packets in the memory packet buffer in a machine independent common format. The DLPI interface appears to provide slightly better performance than the NIT interface; however, it still produces an extremely high CPU load on the workstation that the software is running on.

Since each Unix hardware vendor provides a different method of directly accessing the network hardware, Packetman must provide several different network driver procedures. The correct driver procedure must be compiled and linked into the Packetman program using conditional compilation switches. Because there are separate network interface driver routines for each operating system, the *Solaris 2.x* version of Packetman will not run under *SunOS 4.x* and vice versa. The creation of a raw network interface driver procedure for Packetman is a major step in porting it to a new operating system.

10.3.1.2 Packet Decoding Routine Modifications

There were several modifications that had to be made to Packetman's packet decoding routines to account for differences between *SunOS 4.x* and *Solaris 2.x*. *SunOS 4.x* is a BSD Unix based operating system and *Solaris 2.x* is a System V Release 4 Unix based operating system. Because they use different underlying architectures, many of the network library functions in *SunOS 4.x* work slightly differently under *Solaris 2.x*. Modifications had to be made to many of the Packetman packet decoding functions and procedures to account for these differences.

Most of the modifications that were made occurred in the Network File System decoding routines. These routines are the most complex of all the packet decoders in Packetman. *Sun OS*

4.x uses the C structure `direct` for NFS file access. *Solaris 2.x* uses the C structure `dirent` for NFS file access. All references to NFS file access structures had to be located in Packetman and modified with conditional compiler flags to work under both operating systems. There were also several NFS structures that retained the same naming convention, but that took on different meanings. Several structures in *Solaris 2.x* have variables for a buffer containing NFS information and a pointer to the buffer. The structures in *SunOS 4.x* have a variable for the buffer only. The *SunOS 4.x* variable name for NFS buffers often refers to the variable containing the pointer to the buffer in *Solaris 2.x.* Attempting to directly access this pointer caused many crashes in early *Solaris 2.x* versions of Packetman. Figure 10.3.1.2.1 shows an example of the types of structures that had to be identified and modified.

```
struct nfsdiropargs {
/*** THIS IS THE SOLARIS 2.x VERSION ***/
  fhandle_t  *da_fhandle; /* pointer to directory file handle */
  char       *da_name;    /* name (up to NFS_MAXNAMLEN bytes) */
  fhandle_t  da_fhandle_buf; /* directory file handle */
  int        da_flags;       /* flags, see below */
};


struct nfsdiropargs {
/*** THIS IS THE SUNOS 4.x VERSION ***/
  fhandle_t  da_fhandle;     /* directory file handle */
  char       *da_name;       /* name (up to NFS_MAXNAMLEN bytes) */
};
```

Figure 10.3.1.2.1. NFS Directory Structure Under *SunOS 5.x* and *Solaris 2.x*.

In the above example, the buffer `da_fhandle` has the same name as the pointer to the buffer in the *Solaris 2.x* NFS header file. These references would compile correctly, but would cause the program to crash when it incorrectly accessed the data from the pointer. These types of changes between *Solaris 2.1* and *SunOS 4.x* were not well documented, and they required many hours of exhaustive debugging to locate. Because of this effort, the NFS packet decoding routines and other packet decoding routines in Packetman are now very stable in both the *SunOS 4.x* and *Solaris 2.x* versions.

10.3.2 Packetman Secure Mode

The Netman tools all incorporate a secure mode that allows them to be used by persons that would not normally be able to use network monitoring tools. Most network monitoring tools require root or supervisor access to the machine to be able to run. The Netman tools also require this level of access, but they can be set to allow ordinary users to run them with a limited amount of supervisory privileges. When Packetman and the rest of the Netman tools are run by ordinary users, they enter a secure operating mode. This mode censors incoming packet data and restricts certain operations to prevent the compromise of the network security. Section 11.0 deals with some of the security problems and issues associated with the Netman tools. This section describes how the secure mode was created and implemented.

Packetman Version 1.1 did not incorporate a secure operating mode, so all of the code for this functionality has been added for Version 1.2. The secure operating mode is initiated when the program detects that it is being run at the effective supervisor (or root) privilege level by a user other than the supervisor (or root) account. This mode has two main objectives:

- Censor the incoming packet data on the display and only show common data fields that do not reveal privileged or confidential information.
- Restrict access to certain operations that could be abused by users or used by users to gain privileges on the machine that they have not be granted access to.

These objectives allow the Packetman program to be useful, but protect the machine and other users from the violation of their security and privacy.

The first objective is met in the packet decoding routines. Each of the decoding routines was modified to print zeros or blanks in the place of sensitive data while in secure mode. Since each packet decoding function controls what data is shown, the function writer can choose which items to censor and what items to display. All of the packet decoders that are currently part of Packetman have been modified to operate in secure mode. Several of the decoding routines print informative messages in the place of data such as “Secure Mode -- Password Not Shown”. This allows the person using Packetman as a learning tool to see what would have been displayed without compromising the security of the system. This method of data censoring is extremely flexible since each packet decoding routine can be modified to show different amounts of information based on how sensitive the protocol is. Network control protocols might be decoded fully while data carrying protocols such as telnet and ftp might be heavily censored.

Packetman also modifies and restricts access to certain operations to prevent them from being used to abuse the network or to circumvent system security. All functions and procedures that save files in secure mode were modified to save files at the permission level of the user and not the supervisor account. This prevents the accidental or malicious overwriting of system files by the user. Packetman prevents the user from saving raw packet traces to a file in secure mode. This is done to prevent the user from using a different program (without a secure mode) to look at the packet data. The user can print the packet to a file in ASCII format, but the same censored information on the screen is saved to the file. The user may load packet files that have been saved by users that have permission to do so. These packets are censored like packets that are captured from the network. This allows the use of Packetman in an instructional setting where the instructor may wish to provide packet files that have been previously screened to remove sensitive data. Packetman does not invoke external commands like Etherman and Interman, so there are no restrictions on external commands as in those programs. These modifications help prevent Packetman from being used to violate system or network security.

The secure operating mode is a valuable addition to the Packetman program. This mode allows the program to be used in instructional settings where packet analyzers could not previously be used. Each packet decoding routine censors its own data, so each routine can be customized to the level of security that is desired. Several other operations in Packetman are modified or restricted in secure mode to prevent the compromise of network and system security. When used carefully, Packetman can be used as a valuable tool for research and education as well as for network administrators.

10.3.3 Command Line Switches

Version 1.1 of Packetman provided a convenient, easy to use graphical user interface (GUI) for the user under the X-Windows system. While this interface is easy to use, it is often convenient to set commonly used settings from the command line before starting the program. It is also desirable to be able to control the program from other programs such as system scripts. Version 1.2 now allows the user to set most of Packetman’s options from the command line. The command line switches are described in detail in the Packetman Users Guide. These command line switches make the program much more convenient to use from the command line and from automated programs.

Packetman was also modified in Version 1.2 to allow it to be used to capture packets without starting the user interface. The user can now ask Packetman to capture a specified number of packets to a file while running in the background. This mode can be used with automated scripts and cron jobs to collect packets while the user is not present. All of Packetman's packet filters can be used while in this mode. The packet files that are saved can be read into Packetman at a later time for analysis with the graphical user interface. This mode makes Packetman flexible enough to be used both as a real time packet capture tool and as an unattended packet capture tool.

10.3.4 Miscellaneous Modifications

Several miscellaneous modifications were also made to Packetman Version 1.2 to make it easier to use. A menu button was added to the program to allow the packet information on the screen to be printed to a file in ASCII format. The ASCII file can be printed or viewed by the user as needed A filter selection was also added to the program so the user can filter for hosts based on both the IP address and the Ethernet address. Version 1.1 of Packetman could only filter on the Ethernet address. The filter entry screen was also extended to allow the IP and Ethernet addresses to be entered numerically as well as by giving the host name of the machine. The packet filter menus were re-arranged and filters were added to make them consistent with Etherman and Interman. Tick marks now appear next to each filter when they are selected. This makes it much easier for the user to identify which filters are active. There were also two new packet decoding routines added in Version 1.2. A routine to decode IP packets encapsulated inside other IP (IP/IP protocol) packets was added. A routine to decode Amateur Radio AX.25 based packets was also added [1]. These miscellaneous modifications help to make Packetman easier to user and more flexible.

11.0 Security Issues

11.1 Introduction

Almost every major commercial computer system is now connected with some type of network. These networks require tools to manage them; however, the management tools can also often be used to circumvent the security of the network that they are managing. Therefore, computer security is an important topic for any network management tool to consider. Netman is no exception to this problem. If Netman is misused, it can cause compromises in the security of the network and systems that it is used on. This section describes how Netman can endanger system security and steps that can be taken to prevent the misuse of Netman.

There are two types of security problems to consider in a distributed or networked computer system: host compromise and network compromise [40]. Since the Netman toolkit must be run as the root user, it can be used to compromise the host computer as well as the network. The Netman toolkit can also be used to aggressively attack the network. Woo and Lam have identified three different types of attacks that can take place on a computer network [40]:

- Eavesdropping of messages transmitted over network links to extract information on private conversations (A passive attack).
- Arbitrary modification, insertion, and deletion of messages transmitted over network links to confound a receiver into accepting fabricated messages (An active attack).
- Replay of old messages (An active attack that is a combination of the first two attacks).

Netman can be used to assist with all three of these types of network compromise attacks. Since Netman is a passive toolkit, it is most dangerous as a passive attack agent. The following sections discuss each of these attacks in detail.

11.2 Host Compromise

Host compromise refers to the act of taking control of a host computer in a networked or distributed system [40]. Depending on the type of host that is compromised and the other hosts that trust the compromised host, this type of attack can also lead to subversion of other hosts on the network. For this reason, it is important to prevent any of the hosts on the network from being compromised. Since all of the Netman tools (except the launch utilities) must be run as the root or supervisory user or at an equivalent permission level, they can be used to compromise the host computer that they are running on by allowing ordinary users to gain supervisory privileges. This type of compromise is covered in this section. These tools can also be used to compromise the host by capturing passwords and other sensitive items from the network. This type of compromise is covered in the next section.

When the Netman tools are run as the root or supervisor user, they run in a non-secure operating mode. In this mode, all of the tool's functions are available to the user. Any files that are created by the Netman tools are saved with the permission of the supervisor user (usually root). It is possible to overwrite system files and other files necessary for the operation of the computer system. Netman will provide a warning before overwriting an existing file, but the user can choose to ignore this warning. Netman does not provide many safeguards against abuse and host compromise while in the non-secure operating mode. Since the user already is the supervisor of the system, Netman is no more dangerous than any of the other commands that the supervisor user can use. Once this level of privilege has been gained by an intruder, the host has already been compromised. There are many other safeguards that can (and should) be taken to prevent users from gaining supervisory privileges. These are beyond the scope of this report and are not covered here.

The Netman tools also have a secure operating mode that allows them to be used by normal (i.e. non-supervisor) users. This mode of operation must be made much more secure to prevent the user from gaining full supervisory privileges. The Netman programs do not allow any external programs to be used while in the secure operating mode. This is to prevent the user from compromising the host by launching another program with supervisor permissions set. The Netman tools will start each other, but they switch out of supervisory mode before doing so. The tools also switch out of supervisory mode before saving any files to disk. This is to prevent the user from compromising the host by overwriting or deleting essential system files. The user will only be able to save files in locations that would be normally accessible to the user. These safeguards help prevent the compromise of the host with the Netman tools.

Although the Netman tools provide many safeguards that help prevent host compromise, they should still be used with caution. It is almost always possible to subvert any tool that operates with partial supervisory privileges into giving the user full supervisory privileges. Users that are not trusted should not be allowed to execute the Netman tools, even in the secure operating mode. Close monitoring of hosts should be used to detect any possible violation of system security. Also, tools such as Cops and Tiger that are available on the Internet should be used to regularly check system security on machines that are running Netman.

11.3 Eavesdropping of Messages

Eavesdropping of Messages involves listening to traffic on a computer network to gain information that would not normally be available to the user [40]. Encryption, the basic protection against almost all types of network compromise, is the only defense against this type of attack [38]. Network managers should be aware that a person that is monitoring the network with network management tools will be able to see all un-encrypted traffic on the network. This data may be sensitive and should not be seen by ordinary users. Passwords can be captured on the network and used to compromise host computers. For this reason, any tool that can be used to eavesdrop on network messages should be closely monitored. Etherman, Interman, and Packetman can all be used for several different types of eavesdropping attacks.

Etherman and Interman present the least serious threat of network compromise through eavesdropping on network messages. These tools simply show connections between hosts on the network and the dominant protocol that is being used. On systems where most users know the locations of the hosts and the types of activity that is taking place, this type of monitoring is relatively benign. On systems where users should not know the names of hosts or their communications patterns, the use of these tools should be severely restricted. These tools can also be used to determine which computers on a network are important. These types of computer might include management stations, file servers, and gateways. These computers might then be singled out for host compromise attacks. This is another important reason to only allow the use of these tools by trusted individuals.

Other forms of network compromise attacks against individual users are also possible with these tools. Companies with policies against accessing Usenet News or other Internet services such as the World Wide Web during working hours may want to restrict the use of these tools to network analysis and debugging only. A clear policy should be in place to determine what should be done with information that is gathered that indicates that an employee is violating company policy from their workstation. Managers could use these tools to monitor employee work patterns and work rates during the day. Such use should be done only if it is consistent with a clearly defined policy that has been explained to the users of the network. Use of these tools for spying on users that have not been warned that such activity is taking place is at best unethical and may be illegal.

Packetman represents the most serious threat of all the Netman tools for network compromise. The Netman tool was intentionally designed to eavesdrop on network communications. It provides a powerful set of decoding routines to capture and analyze network traffic. When Packetman is used by a supervisor user (usually only root) it will display the contents of all packets that it captures. It can also save these packets to a file for later analysis. Packet files that are saved to disk should be closely guarded since they can be read by a wide variety of other network analysis tools. ASCII data in the files might be visible by looking at them with a text editor. This tool is intended to be used only as a diagnostic and management aid for network managers, but it should be clear that it can also be used to spy on network traffic. Access to the Packetman tool by supervisory users should be closely regulated, and the types of acceptable use should be specified clearly in a written policy.

Packetman also has a secure operating mode that it enters when it is used by a non-supervisory user. In this mode, it censors the packet data that it shows to the user. Because the tool would be useless if it censored everything, a great deal of information is still shown. The network manager or other responsible person should extensively test Packetman in the secure operating mode to determine if what it displays is acceptable. Users that are not trusted should never be allowed to run Packetman even in secure mode.

The Netman tools are designed to capture and analyze information about traffic on the network. They can also be used to eavesdrop on network traffic without modification. Knowledge gained through such eavesdropping can lead to the release of sensitive information and the compromise of host computers. The use of the Netman tools (even in their secure operating mode) should be closely guarded and monitored. Users should be forced to abide by a clear written policy regarding the acceptable use of these tools. Since the Netman tools are not the only tools that can provide this type of information, any user with information that cannot be compromised should consider encrypting the information before sending it on the network. Encryption is the only sure defense against these types of tools.

11.4 Modification of Messages

Modification of messages involves removing a legitimate message from the computer network and replacing it with another message with a modified content [40]. The Netman tools do not currently support the modification of any of the information that they capture. Packetman can save packets to a disk file. These files could then be modified by a skilled user and used to compromise the network. Packetman does not allow non-supervisory users to save packets to a file. Only the root or supervisor user can save packets. These files should be safeguarded to ensure that they are not accessible by individuals that are not trusted.

The Netman tools do not support the transmission of network information. They operate as passive tools only. Therefore, they cannot be used to directly make a message modification attack. They can only be used to gather information about the network to facilitate such an attack. The users of these tools should be trusted individuals, and the network manager should watch for problems with their use in combination with any other type of network tool. The use of encryption for sensitive data will also help prevent these types of attacks.

11.5 Replay of Messages

A replay of messages attack is a combination of the eavesdropping of messages attack and the modification of messages attack [40]. The replay of messages attack attempts to circumvent security systems by replaying legitimate messages at a later time. The messages are legitimate messages, so the hosts on the network will often respond to them. By playing the messages out of sequence, the hosts can be made to perform actions other than those done by the original user. The messages might also be modified before transmission to change their meaning.

Since the Packetman tool is a capable network eavesdropping tool, it can make up half of the necessary components for a replay of messages attack. The Packetman packet trace files could be used as a source of packets for this type of attack. Packetman should only be used by trusted users and the files it produces should be safeguarded. Encryption and time stamping techniques are useful defenses against this type of attack [40].

11.6 Secure Use of Netman

The Netman tools provide the potential to be used to compromise the security of host computers and the network. For this reason, their use should be limited to trusted users. Their secure mode of operation should also be closely monitored to prevent abuse. Network managers should provide a written policy covering the acceptable use of these tools. Users should be clearly instructed on the types of traffic that they should and should not monitor. Users and managers should guard their packet data file and watch for unauthorized use of the Netman tools.

If proper policy and security safeguards are in place, the Netman tools can be used as a valuable component of a network management effort. Users and managers should be aware of the risks involved and take the proper precautions. System security through other methods should be utilized to prevent against compromise not related to the Netman tools. Users with sensitive information should consider the use of encryption to prevent their data from being intercepted by parties that have compromised the network or the host system.

12.0 Comparison With Other Network Monitoring Tools

12.1 Solaris Snoop

Solaris *Snoop* is a packet capture and analysis utility program that is included in the *Solaris 2.x Operating System* distribution [34]. This program provides many packet capture and analysis functions that are similar to those in Packetman. It does not provide any features that are comparable to those found in Interman or Etherman. *Snoop* can capture packets from a network device in both promiscuous and non-promiscuous mode. The packets may be saved to a file for later analysis or they can be decoded as they arrive. Figure 12.1.1 shows a single packet that has been captured by *Snoop*.

```
# snoop -v -c 1
Using device le0 (promiscuous mode)
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 9:54:29.95
ETHER:  Packet size = 158 bytes
ETHER:  Destination = 0:0:ef:1:da:e0, Alantec
ETHER:  Source      = 8:0:20:7:f1:d6, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
```

```
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 144 bytes
IP:   Identification = 49924
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = eb50
IP:   Source address = 128.194.135.112, sparc12
IP:   Destination address = 128.194.132.18, stellar
IP:   No options
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 1017
UDP:  Destination port = 2049 (Sun RPC)
UDP:  Length = 124
UDP:  Checksum = 0000 (no checksum)
UDP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Transaction id = 1002664
RPC:  Type = 0 (Call)
RPC:  RPC version = 2
RPC:  Program = 100003 (NFS), version = 2, procedure = 4
RPC:  Credentials: Flavor = 1 (Unix), len = 32 bytes
RPC:     Time = 04-Apr-95 15:54:30
RPC:     Hostname = sparc12
RPC:     Uid = 931, Gid = 200
RPC:     Groups = 200
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:
NFS:  ----- Sun NFS -----
NFS:
NFS:  Proc = 4 (Look up file name)
NFS:  File handle = 0080009000000002000A000000009503
NFS:                2A4ED8BE000A00000000000240A1B9FC
NFS:  File name = EXTERN.h
NFS:
```

Figure 12.1.1. Packet Captured by Snoop.

While *Snoop* is quite flexible in the types of packet capture and analysis functions it can perform, it does not provide all of the functionality of Packetman. Packetman can be operated in a command line mode like *Snoop*, and it can be used in X-Windows with a graphical user interface (GUI). The GUI interface makes Packetman much easier to use. The specification of packet filters is one area where *Snoop's* command line interface makes it much harder to use. It is much easier to create and set filters in Packetman with the GUI than in *Snoop*. To capture NFS packets in Packetman, the user selects the NFS choice from the menu. *Snoop* requires that packet filters be entered in a specific filter expression format on the command line [34]. The command `snoop` `nfs` is required to capture NFS packets in *Snoop*. Since Packetman also has a command line, it can be used in most situations where *Snoop* would be used. The GUI and command line modes of Packetman make it much easier to use than *Snoop*.

Since both *Snoop* and Packetman both use the DLPI interface when running under *Solaris 2.x*, they have similar accuracy. Packetman has more overhead than *Snoop*, so is more likely to drop packets. Both programs will drop packets if the machine is not able to keep up with the amount of traffic on the network. Packetman is more flexible than *Snoop* and can be run on a wide variety of Unix platforms. *Snoop* will only function under *Solaris 2.x*. Neither program currently supports remote packet capturing from other machines or SNMP RMON agents.

Snoop currently decodes only TCP/IP packets completely. Since this program is provided by Sun, it is not possible to extend it for new protocols. Packetman currently fully decodes the same types of packets as *Snoop*. Packetman can also be extended to decode any Ethernet based protocol by creating a decoding function for the new protocol. Packetman can also be used to provide more detailed explanations of items that are decoded. Many of the Packetman decoding functions provide more information about what is being decoded than *Snoop*. These advantages also make Packetman easier to use and more flexible than *Snoop*.

12.2 NNStat

The *NNStat* packet is a set of software designed to monitor the traffic on the Internet [3]. This software can gather a wide variety of statistics about TCP/IP and UDP/IP traffic from Ethernet segments that are connected to the Internet. *NNStat* is primarily designed to be used on Ethernet backbones where there are large amounts of traffic. *NNStat* uses its own configuration language to provide the maximum amount of flexibility to the user. *NNStat* runs in promiscuous mode on a Unix host attached to an Ethernet network segment.

Because *NNStat* is designed for use by networking professionals to monitor large volumes of traffic, it is very difficult to use. The configuration language allows an expert to do almost any type of packet analysis with the *NNStat* tools. This same language makes the package very hard for beginners to use. It is a stated position of the *NNStat* project that fancy display and analysis programs are not supported and will never be created [3]. The user is left to provide the tools for analysis and display. Interman and Etherman provide can similar statistics as *NNStat* in a convenient, easy to use format. Since Interman and Etherman manipulate graphical displays, they are not as accurate as *NNStat*. The *NNStat* package makes no attempt to capture or analyze individual packets like Packetman. The *NNStat* toolkit is primarily of use to internetworking professionals that need accuracy and flexibility at the expense of ease of use.

12.3 Beholder

Beholder is a Simple Network Management Protocol (SNMP) compliant Remote Monitoring (RMON) agent [20]. *Beholder* monitors network traffic on Ethernet networks and stores this information in the Internet standard SNMP RMON Management Information Base (MIB) format. *Beholder* runs on Sun machines running *Solaris 2.x* and on IBM PCs running *OS/2*. The *Beholder* program itself provides no user interface to view the collected information with. The information must be collected by a client that supports the SNMP protocol. Because *Beholder* supports the SNMP standard, it is possible to use *Beholder* with a wide variety of SNMP RMON compatible programs. The lack of a user interface helps *Beholder* maintain a more accurate summary of the network by avoiding spending processing time on user interface elements. The source code for *Beholder* is available on the Internet, so it may be modified to meet local requirements. The *Beholder* program provides services that are similar to Packetman. With the correct set of software, it can also be used to provide displays similar to those found on Interman and Etherman.

Unlike *Beholder*, Packetman spends a great deal of effort to maintain a good user interface. This makes Packetman less accurate than *Beholder*, but it is much easier to use. The *Beholder* program must always be used in conjunction with another display program. The SNMP display program can be run either locally or over the network from a remote computer. Since *Beholder* supports the SNMP standard, it can be used with a wide variety of commercial and public domain software. The ability to query *Beholder* over the network is also an advantage over Packetman. Packetman does not currently contain a SNMP module, so it cannot gather information from programs like *Beholder*. Since it uses SNMP, Beholder requires a significant amount of setup before it can be accessed. Packetman can be used without any configuration or setup. Packetman is a better solution for users needing a self contained solution that does not require any additional software. The graphical user interface for Packetman also makes it easier to use than *Beholder*. *Beholder* is a better solution for sites with an investment in SNMP software or sites that need to remotely monitor several networks.

12.4 Ethload

Ethload is a public domain Ethernet traffic analyzer that runs on IBM PCs and compatibles [39]. This program places the PC's network card into promiscuous mode and monitors the traffic on the local Ethernet segment. The program displays traffic statistics for several major network protocols including TCP/IP, Novell Netware, AppleTalk, and DECNet. It also has some basic packet decoding features.

While *Ethload* has some basic decoding features, it is more similar to Etherman and Interman. The majority of its functions are intended to show the amount of traffic on the network and display statistics for traffic between hosts. It operates in real time like Etherman and Packetman, but its display is completely text based. Figure 12.4.1 shows a typical *Ethload* display. Ethload does not attempt to use a graphical user interface because of the processing limitations of the PC host. The PC host should not be used for any other task while running *Ethload*. Since most PCs are less powerful than the majority of Unix workstations, they are even more prone to dropping packets. This makes Ethload less accurate than the Netman tools. Etherman and Interman provide an easier to understand display of the network, while *Ethload* provides a text based display of several detailed statistics.

```
Bytes received:   0000000989 (   0.02% of Ethernet capacity)
Frames received:  0000000002 (mean inter-frame time    2500.00 msec)


Statistics for the busiest 5 second(s) Sun Apr 09 16:13:01 1995 0.00% lost
Bytes received:   0000010116 (   0.16% of Ethernet capacity)
Frames received:  0000000022 (mean inter-frame time     227.27 msec)


Cumulative statistics since Sun Apr 09 16:12:56 1995 0.00% lost
Bytes received:   0000031128 (   0.10% of Ethernet capacity)
Frames received:  0000000066 (mean inter-frame time     378.79 msec)
```