Global Information Assurance Certification Paper

Security and Vulnerability Analysis of an Ethernet-based attack on Cisco IOS

Abstract

We note the recent attack on Cisco routers, publicized in July 2003, and analyze this work and expand upon it. This exploit used crafted packets to overflow the input buffer on Cisco devices and caused a Denial of Service, making the device unavailable for legitimate users, leading to loss of network connectivity. Cisco has provided a patch for this vulnerability.

A test regimen was developed by the author, utilizing two Cisco routers (at once) of the 2500-series class, IP-based network traffic devices, a hardware-based Ethernet traffic generator, Ethernet packet capturing devices, and support equipment. The goal of the exercise is to develop additional information about the type and behavior of the vulnerability, assuming that many Cisco routers today remain un-patched. We provide additional information about the behavior of the Cisco vulnerability. Another goal is development of a test system, applying this problem scenario to it, and learning what issues need to be generally addressed when doing such testing. This is a valid security issue. Finally, a third goal was to do this at low cost, using legacy equipment as possible. This work is a narrative; details and other reference and supporting information are found in appendixes or notes, at the rear of the paper, to avoid undue distraction from the flow of information.

This work is in submission for the Practical for the GSEC, version 1.4b option 1.

1.0 Description of the problem.

Software vulnerabilities are a “hot topic” in contemporary computer security circles, however they are not frequently linked to a router, and not all are publicized widely. There were 42 incidents reported in 2001. [1] The case in point is one of the exceptions which tend to prove the rule. Generally it is operating systems that attract such notice. Often, vulnerabilities are password-related. [2]

This one well-publicized router vulnerability, publicized in mid-2003, affected the Cisco IOS software which is the software component of virtually all enterprise routers in the Cisco subset of enterprise networking. This is about 70% to 80% of ALL installed routers, per industry estimates. The scale and breadth of this vulnerability becomes apparent when viewed in context of these numbers.

The vulnerability was mitigated by patches to the Cisco image software [3] that were available just days after the news broke. This information is included here for historical reasons and is not presented here as being any new information.

However it is reasonable to assume that, in any such event where a vulnerability is announced, some of the affected users will be proactive in applying the remedy, but many will not. This strongly parallels the world of operating systems, where we constantly learn of attacks to (mainly) Microsoft systems. These attacks are successful despite the existence of remediation which is already available. Followers of online news sources such as Slashdot [4] or Techweb [5], and security mailing lists such as Bugtraq [6] are well acquainted with this paradigm.

Here we make a distinction of some importance. Many O/S vulnerabilities exist at a user level, one example is the Port 135 DCOM RPC attacks using the Windows Messenger Service to send "pop-up" ads. (This may be thought of as an annoyance, at best, by some). However more serious issues exist at the enterprise, or IT-department, level. One example was the vulnerability to Slammer caused by the SQL service listening on an open port 1434. This would not be thought of as a home-user issue. By extension, one would assume that departmental servers with 1434 exposed would have been patched, but the success of Slammer showed that many were not. Since the SQL listener needs to be available on these servers, blocking this port at a firewall is not an option.

The corollary to this assumes that there are many Cisco devices that have been installed a year or greater in the past, that are vulnerable, since these devices tend to be placed in racks or closets, away from daily attention, and not needing much care. This point is addressed in Phrack #60 [7] in a similar context. (see section 8.0 for discussion)

1.1 Justification for doing this testing

This test scenario was created and done solely for the purpose of the SANS practical assignment, and for no other purpose or beneficiary, and this information has not been given to any person or group, other than the SANS review committee to which it is being now submitted. (March 2004)

Existing advisories for this issue refer to known exploits but provide varying detail. We seek to provide new information by (1) attempting to determine the level and severity of attack needed to replicate the failure, (2) determine if any new failure modes can be found and (3) gather and report quantitative details on such findings as we may obtain. In other words, how (at what rate) does the input buffer fill with 'legal' traffic, does it fill at a different rate with experimental traffic added to the legal traffic, or in the absence of any legal traffic, and is there any non-linear "trip point" at which, if a certain input rate is gradually reached, the buffer rapidly fills and collapses, causing the Denial of Service.

The justification for this work includes the following:

1. Provide ongoing awareness of the problem (a security issue).
2. Add to the knowledge of specific behavior of the Cisco device when subjected to specific IP traffic.
3, Attempt to experimentally discover different or unknown attack vectors.

Selection of Cisco as a subject for this kind of testing was helped by our interest in Cisco technology and services, and by the availability of some hardware acquired for our home testing lab. Lab preparation encompassed three areas: Information, hardware, and software, which are addressed in the next section.

1.2 Disclaimer

It will become apparent to the reader that a lot of the material in the lab is old, "obsolete" (but still functional) software, used machines bought at computer shows or on-line, or loan items. The Red Hat version 9 software is contemporary (about a year old) and the Cisco IOS in 2 routers for testing is about a year old. However everything works as intended and gives useful information and since we do not have access to any corporate patrons, the cost factor, being self-supplied, is an important consideration.

No corporate resources or networks, or network connections were used in conducting this test. All IP address we use are of the RFC-1918 variety (private, non-routable). The network used is not connected to any outside routers or gateways during the test scenario. However it is possible to connect the "10" router to the home LAN and set the Ethernet port to act as a default route to anywhere through my gateway [8] and run the NTP client to secure clock synchronization; this however was not essential for the test results and was not done here. The intent in other tests is to make the one router a NTP server to the other routers after synchronizing it to a public reference server. [9]

All research was done by the author and all sources from which material was excerpted or developed are cited in the end of this paper.

2.0 History of the Original Attack

The case history begins on or near 17 July 2003 with posting to forums such as Bugtraq. It was an active news story for just a few days and is now faded away.

The detail of the Cisco vulnerability and patch announcement is referenced at [10].

3.0 Laboratory Equipment

3.1 Routers (devices)

The hardware used in this test includes several Cisco 2500-series routers [11] with support for Ethernet and serial interfaces. We have also some token-ring devices but these were not used here. These devices were obtained through private sale, or via eBay auction. Only 2 routers were used at any one time with a third (2511) for console access.

3.2 PC based devices

We are using three Dell Optiplex GX-100 466 MHz systems running a full ("everything") install of RedHat linux 9.0 (kernel version 2.4.20) [12] for traffic generation and monitoring. The Dells have an onboard 3Com LAN chip for which linux detects and loads the 3c90x "tornado" driver. The Dells are good performers for units of their size (a "pizzabox" with dimensions of 31.5 cm W x 39 cm D x 9 cm H). They were acquired for creating portable test traffic generator units that could be taken to a site and set up easily; being more rugged than many laptops. These can also boot with any of the several available "run-from-CD" linux variants that are now available as download ISO files. [13] as well as running from hard-disk installs. The availability of toolkit and forensic ISO boot-from-CD images allow implementation of a compact system with a range of testing capabilities now available in linux.

We also have in service two DOS-based low profile systems, also in a "pizzabox" case configuration, perhaps some of the earlier ones made, dating from 1996 using Pentium 133 MHz CPU's. One runs a terminal emulation program called pc-plot [14] allowing serial line communication with the console ports of the routers. [15] This program allows flawless connection over a serial line at 9600 baud to the router console port for administration and configuration of the routers. [16] This little program handles the escape display characters in the unix editor vi flawlessly and does not tie up a valuable linux machine just to run minicom or some other terminal program.

3.3 Traffic analyzer (device)

In addition to tcpdump on linux (machine 53), as an added reference, the following is available: Another similar low-profile "pizzabox" system running MS-DOS 6.22 supports a 16-bit SMC 8416-BT ISA NIC by loading either a packet driver or an ODI driver. At boot the ODI driver must load with high memory options disabled due to DPMI issues. This is done by selecting the F8 key and parsing the autoexec.bat and config.sys entries manually. We can run either of two decent Ethernet capture programs, called PacketView [17] or Landecoder/e [18]. These machines have a low profile and power consumption, boot rapidly,

work with old cheap VGA monitors, give accurate results in this application and appear indestructible, since they are working with 80 MB Sandisk ® flashdisk drives.

These DOS machines allow captures of screen text information to a Panasonic dot matrix printer using the old DOS “PrintScreen” functionality, which works well enough to provide useful results.

The laboratory supports a 10 Mbit/sec infrastructure which makes sense as the port speed of the 25xx-series router is fixed at this speed. Cat-5 cabling and Netgear EN-108 hubs (10 Mbit) connect these devices. Hubs are necessary to permit monitoring of the traffic flows with analyzers, as switches with port replication are not available here due to cost. Transceivers of type Cisco BL50R (802.3 spec.) are used to interface each router AUI port to Ethernet Cat5 cabling.

This equipment is powered through two APC Back-UPS 650 uninterruptible power supply units to ensure saving state on all devices in case of momentary power glitches. Such momentary glitches occur several times per week at this site as the local power grid adjusts load.

3.4 Hardware traffic generator

A Shomiti Surveyor [19] hardware based packet generator with a Century Media Module hardware card, and a software based “Packet Blaster” plug-in is used to generate flooding traffic [20]. The use of the “Blaster” and the hardware card allows the user to generate packets of arbitrary length (including less than, or greater than the Ethernet legal values of 64 through 1518 bytes) and with various ethertypes (other values in addition to 0x0800, DOD internet Protocol), and with any protocol value (if using IP) from 0 to 255.

This is outside the scope of our testing, but is a great tool to test decodes of various analyzers we tried. It is running on a Dell XPS 266 MHz system. The hardware module handles traffic generation at up to full wire rate and is CPU-independent, and uses a media-independent interface which can auto-select the correct speed. The GUI/user-level control runs on a Windows 98 base. This is 7 to 8 year old technology but it works well here. This device supports capture (with full errored frame decodes), replay of captures, and transmitting user-defined patterns, and was used for this project solely in the transmit mode.

We considered trying to obtain some sort of loan access to a SmartBits generator [21] but decided that this gear worked well enough when gathering data. The ability to generate arbitrary traffic at high, stable rates is useful, but since devices such as this don’t generate a true connection-oriented TCP stream, (no incrementing sequence numbers; no ACK’s, and an invalid IP header checksum), the results have to be considered in this light.

3.5 Lab facilities

Two folding tables are available, one for the linux computers and another for the analyzers and the generator. The routers and hubs are installed in a standard 19-inch open frame Chatsworth 84-inch freestanding relay rack. This is on a 3x3 foot ¾-inch plywood base which makes it tip-proof. A total of 6 CRT's are involved (3 linux, terminal, capture, generator) which at this point is still easier than trying to implement a good KVM solution. Besides the usefulness of seeing multiple screens at once soon became evident. Use of a KVM would have actually slowed down runs a lot, as during a test run, the command line was changed on linux, verification of the change on traffic monitored with tcpdump on a second machine was done, and testing the result on a third machine connected to the router, running commands to view the interface statistics,

4.0 Laboratory Configuration.

The configuration of the test system is shown on this diagram. See [4.0]; p.30.

4.1 Network Configuration.

Two networks are created. Each is created "in" one of the two Netgear EN-108 hubs, which may be thought of as a "collapsed backbone". One of the networks is 10.73.73.0 /24 and the other is 192.168.30.0 /24. One each of the Dell linux machines is set up on each of these networks. The IP addresses of these machines are 10.73.73.52 and 192.168.30.51. They are set up with "class C" type addressing (a /24 mask) and the default gateways are the IP addresses of the Ethernet ports on the routers, which are 192.168.30.1 and 10.73.73.110 [This was chosen for the router being a node on the home net for NTP purposes – see 9] [see 4.6 for explanation of 51, 52, 53 for the machine numbering convention].

4.2 Router Configuration.

Two routers are used in the test. One is considered the "DUT" (device under test) and is the one on which the Ethernet port is numbered as, and connects to, the 10.73.73.0 network. This router was originally the one called "10" and runs IOS 12.2 enterprise software. This router was tested for the vulnerability and then was replaced with an older router of similar hardware, but running IOS 12.0. That code was compiled earlier, and that second router, which is called "2" was also tested, and did show the fault. This will be addressed in detail later. [22] gives a detailed spec for each router.

The other router, facing the 192.168 network, is called “3” and was used to provide a symmetrical “mirror image” device to translate the serial data back to Ethernet on the 192.168 network so the data could be handled with Ethernet at both ends of the link. This is a consequence of each router having a single Ethernet port and one usable serial port (Serial0). The second, unneeded Serial1 port was administratively shutdown on each router. This router “3” was not fault tested.

The serial0 ports of the routers are set up as 172.16.3.1 and 172.16.3.2 with a /24 mask which works, as addresses are plentiful here (!) although we could have used a /30 mask if desired, as is the custom for single-ended point-to-point WAN links where addresses ARE often scarce. It makes no difference here. This addressing becomes helpful to have different subnets with connecting several routers in series and studying propagation delay through the whole chain with tools such as netperf [23] or iperf [24] when the serial interface speeds are varied. This is one of the purposes of acquiring these devices before assuming this test project for SANS.

4.3 Serial Devices

For the purpose of this particular test, the two serial ports were connected through a pair of Adtran TSU devices [25] (channel service units/data service units, sold in one physical case) which simulate the routers being interconnected, at two separate sites, through a T-1 line. This was considered to be a good approximation of a scenario where two sites were actually connected through a telco or similar circuit at T-1 speed.

Each TSU is a full T-1 rate device and they are set up so that one provides internal clock and supplies clock to the other, so the router serial interfaces need not be run as DCE with a speed statement in the interface configuration. The TSUs are set up so that all 24 DS-0’s are active, which gives a connect speed of 1544 kbit/sec. The TSUs are connected with a 2-foot length of Cat5 cable so that they talk to each other [26] in the same manner as when a “null modem” cable connects two DTE serial devices to each other.

Testing with iperf, which is installed on both linux machines, verifies use of this data rate [27]. This becomes the ‘bottleneck’ in the ultimate maximum transfer rate between the linux machines, since the Ethernet interfaces are several times faster.

4.4 Diagnostic issues

We considered use of SNMP [28] to try to get the stats of the router. To allow this, the command "snmp-server community public RO" is issued while in global config mode to allow reading the values. Also, "snmp-server xx" commands can be issued. The 'xx' are options, such as contact name, location, chassis-id etc. The default is for services such as this, the http web server, and the finger service, now is to be "off". Then snmpwalk [29] or a "snmp Get" can be issued against the router if the proper MIB value is known. We don't know if this value we wish to test for even has a MIB and if so, it would be accessible in a real-time fashion. We decided that old fashioned polling of the interface stats at the command line was good enough to get the queue data since that data varied only on change of offered load, and not instantly during application of load. We used a FreeBSD machine running 4.9-STABLE for the SNMP testing. Snmpwalk was run against router 10 and a few hundred MIB values were returned, most of which were of no relevance here. This machine was needed for another project involving Patty and the FreeBSD project was shelved. The idea would have been to poll from the 192.168 side to make the Get query enter the router under test from the non-flooded interface, to minimize risk of losing the UDP data on which it rode. We plan to revisit the SNMP diagnostic questions again.

MIB references are available from the Cisco site and other web sites [30].

Another consideration would be running mrtg (multi-router traffic grapher) against the DUT router, but this too was seen as labor intensive for the value returned as the data probably would not be nearly granular enough to analyze in the desired manner.

4.5 Routing protocol

There are a number of protocols available for use here. They work equally well. The choices are RIP, IGRP, EIGRP, OSPF and for simplicity, RIP [31] was selected. This allows the 10 network linux machine to talk to the 192.168 net linux machine. RIP needs to run on each router in the network with route entries to all other networks, including the loopback addresses. Because RIP v.1 understands only classful addressing, RIP is started on each router ("router rip" while in global config mode) with network statements of 10.0.0.0, 172.16.0.0, 192.168.30.0 and 192.168.31.0 (the loopback net on device 3, and symmetric networks on the other router). RIP announcements are propagated to all interfaces every 30 seconds and the updates are seen on the analyzer. This does not interfere with the testing. [32] These announcements can be disabled if needed on any interface with the "passive-interface <interface>" command when in "config-if" mode. .

4.6 Use of Linux-based PC Devices

The PC devices with IP addresses encode a unit identifier in the last octet of the assigned address. See [33] This is done for unit identification. Physical descriptions are in 3.2.

4.6.1 Ping (ICMP Echo request/reply)

The linux systems support easily running multiple instances of ICMP [34] echo testing, with sub-second intervals and varying packet lengths. This achieves reasonable traffic density. The purpose is to force traffic, at a known, steady rate, through the test interface and monitoring that rate, to determine what kind of failure mode of the interface develops, and to see it in real time.

Five terminal windows are opened on a single workspace (unlike Windows, linux offers four workspaces to choose from, and each can support multiple tasks at once). The 5 terminal windows are stacked vertically in one workspace, each with a command prompt. Running the command "`ping 10.73.73.52 –i 0.1 –s 1420`" creates close-to-legally-maximum size packets with a repetition rate of 10 per second.. Then, four more instances of ping, with the same command, are run in each of the 4 remaining terminal windows, however, the data lengths are set to 1430, 1440, 1450, 1460 and 1470 bytes. The packets are decoded with the LD/E analyzer, and the IP and MAC headers, along with the 4-byte frame-check sequence at the end, increase the frame length, as decoded, to 1476, 1486, 1496, 1506 and 1516 bytes. The frame length was shortened slightly to ensure there would never be any fragmentation issues, though none are expected, even with a 1518-byte frame. The reason for this is solely to determine - on the analyzer - that all 5 windows are active, although looking at the terminal windows directly on the linux ping generator box is also confirmation that they are running. However it is instructive to determine whether the sequencing of each ping instance is invariant in the time domain with respect to the others. Apparently some sort of context switching ensures at some times that the five streams occurs in a nearly random fashion, but slightly change sequence, which seems to show that the nominal 100 ms. interval between any two adjacent ping requests can vary slightly.

Running these 5 ping instances simultaneously creates a load of almost exactly 10% network utilization which is a constant, uniform data flow. This then transits (crosses) the Ethernet interface which is to be tested later (see section 5 and 6). We felt that 10% utilization for the data flow through the router was a good number to start with to test failure of the interface processing rate when subjected to attacking flooded traffic. This value was a subjective judgment call.

See [35] for information on ping timings through the test system.

4.6.2 tcpdump

The third RedHat box which is called 53 (IP 10.73.73.53) is an identical machine to 51 and 52 which are used for performance testing. However here we are running tcpdump [36] for real-time monitoring, but not for data archiving. It is not truly part of the test setup as none of its results are recorded for analysis. It gives however a good quick confirmation that our commands are working as intended when generating attacks.

.

4.6.3 hping

Hping2 [37] is a program by Salvatore Sanfilippo which is available on the web as a gzipped tar archive of source, which installs cleanly (except for a missing manpage path error) on linux as well as on other u*ix-workalikes.. A separate HTML-formatted manpage (user 'manual' guide to command syntax and options) is included, which lists the many options that may be used. This program was central to being able to recreate the test scenario. We will discuss this in section 6.0 later in this paper. We used version 2.00 release-candidate 2.

4.7 Traffic analyzers (application)

We have mentioned tcpdump on linux. It is run at the command prompt in a terminal window as "`tcpdump –e –n –v –x –i eth0`" while as root (so the interface can run in promiscuous mode). This gives a useful amount of detail, and does NOT force attempted DNS resolution of IP's (the "n" switch). This is usually a good idea in security work, so as to minimize your own intrusion onto the network being looked at. [38] It is a 'rough and ready' test of ongoing activity. Some of the packets may be dropped by the kernel at high data rates, a risk with any software-based analyzer.

While we could capture with PV, LD/E was found to be the more versatile program. This 1996 vintage program has similarity in look-and-feel to the old Network General *Sniffer* ™ [39] that was available when Net.Gen.was a standalone firm and their product ran on a heavy Toshiba laptop using DOS with an orange gas-plasma display and a special purpose 16-bit ISA NIC with an AUI and coax-jack. One useful function is the skyline display which can be set to show traffic levels on a per-second increment with an accuracy of a couple of percent. As mentioned before, it was easy to do screen prints to a dot matrix Panasonic printer.

Summary and detail decodes are available, error stats can be seen and packet/byte counts can be seen in real-time during captures, and capture sessions can be saved to disk for later analysis. PV's one real advantage is the ability to examine a packet detail while still capturing. Fortunately Ethereal does this also. LD/E requires the capture to be stopped before doing a detail. However

LD/E allows viewing error statistics (due to use of an ODI driver) and offers somewhat more comprehensive 7-layer decodes.

5.0 Interface Behavior Analysis with "non-attack" traffic

There exists a conceptual difference between applying a heavy offered load of traffic TO an interface, rather than attempting to force the traffic THROUGH the interface (where it would have to be routed through and appear on a different router interface). The test model we use considers it as sufficient to apply the offered load TO the interface, which of course has to deal with it as it would any other frame-level traffic, as long as we know the interface is accepting the traffic.

Having said this, we note that we first attempted to use the Shomiti to generate "attack" traffic, i.e. we attempted to "crash" the interface with this device. We were unable to do so. However we did get some interesting behavioral information, believed to be new material. Perhaps, however, it is "new" only outside of a corporate developmental lab.

Loading of the interface is accomplished by using the Shomiti to generate the experimental traffic, which does not apparently have to be offered at extremely high rates. Background ICMP traffic is provided by the Dell linux systems.

We find that offering a traffic load to the Ethernet interface will cause the input queue value to rise at a somewhat (almost) linear rate, the queue utilization value being dependent on the network utilization value.

Frames were created with the following characteristics: Destination MAC set to the router interface MAC. Source MAC is arbitrary (using 00b0.bf00.abcd), ethertype 0800, protocol decimal 53 ("swipe", one of the identified protocols of interest). The destination IP is the same as the router interface IP (10.73.73.110), the source IP is arbitrary (10.73.73.170). These values mean that the router interface handles, but does not answer any of these frames. Recall also that this generator does not send ARPs. Without setting the destination MAC and IP, the router did not react in any way to this traffic.

We hoped that testing we did with different protocol types would recreate the exploit but this did not happen.

The data fields are constructed with arbitrary but realistic values for fields such at TTL, ToS, fragment/offset and ID. One problem was that an IP checksum value has to be entered in the correct offset, and is certain to be incorrect; in addition every time any other field is changed, the (real) checksum changes as well. All the analyzers here complain that the IP header checksum is invalid. Fixing this is like the dog chasing his tail. However the FCS (Layer 2) is correct, though the user has the option of making this errored, we never had reason to do so.

The pertinent variables then become adjustment of the frame length (64 to 1518 bytes) and the utilization percentage rate. Once the adjustable data field length is exceeded, the balance of the frame length is filled with 0x00 null chars.

The value of the input queue is seen by running “show interface ethernet0” or simply, “sh int eth0” at the console terminal. The router input queue length in IOS 12.2 reports, as data, in this format: 2/75/0/0, where these numbers show 2 (queue size), 75 (queue max), 0 (drops) and 0 (flushes). However in IOS 12.0 it is shown just as 2/75. When the queue is filled, 76/75 is seen.

When generating load, the input queue can be made to fill to the 75/75 or even 76/75 point, however the interface simultaneously continues to handle the ping traffic at that point, and when the load is removed, the queue value drops back to zero immediately. This behavior was quite unexpected. Data was taken on the IOS 12.2 router with load offered at fixed rates sufficient to fill the queue, which is of some academic interest.

It is unclear how the queue can appear filled at 76/75 yet continue handling the ping traffic. Later testing with hping (Section 6) gave very different results.

When doing this testing we have determined that apparently the 12.2 IOS device came from a source (via eBay) who proactively patched the system. The 12.0 has been in existence since before the patch was announced. This appears to validate our measurements.

Observations we can make are as follows: (a) Short frames (64 byte) fill the queue much faster, as a percentage of network utilization, than do larger frames. (b) A test was run on the 12.2 IOS to see whether the input queue filled at the same rate as the number of frames/second, for any frame length. At low rates such as at 200 frames/second, the input queue (equal to 10) was consistent for all frame sizes. A significant divergence in the input queue number, in the curves for different sizes began appearing at 300 to 400 frames/second. The larger size frame was increasing the queue value as a function of length somewhat faster than the smaller, as rates went to 700 frames/second. (c) The IOS 12.2 (presumably fixed) version we tested was much slower to fill the queue, as utilization increased, than the unfixed 12.0 {see graph 13 of 128 byte curves). The vulnerable software filled the input queue to 76/75 at half the utilization, somewhat over 4%, as did the non-vulnerable software (10% utilization), when the frame length was fixed at 128 bytes. We did not do a detail list of the other length values, believing the single 128-byue case is illustrative..

For the 12.2 IOS, the input queue filled, but not ‘destructively’ at these rates: 64 bytes at 5.5% utilization. 128 byte at 10%. 256 bytes at 17%. 512 bytes at 32%. 1024 bytes at 58%. and 1518 bytes at 75% utilization. Again when the load was removed the value returned to zero and processing continued. Each plotted curve of the rate exhibits a similar bowed aspect of a slight upwards-facing cusp

in an otherwise straight line. We consider this as being of academic interest. These graphs are available in the appendix, starting on page 40.

A number of open source traffic generation tools are available, including netperf and iperf as well as ftp transfer. These can create TCP connected streams between the two devices, in which maximum legal frame sizes (1518 bytes) are sent from one machine to the other, and small ACKs are returned. These tests tend to run at the maximum data rate supported by the physical media and the equipment. We believe that, as long as the connection is valid TCP and there are no frame errors (FCS, alignment, babbles, etc.) then the actual data content of the frame is unimportant. See [40].

A word about the terminal program: during the testing, we found it advantageous to frequently refresh the display for the Ethernet interface often ("sh int eth0") to see the status of the input queue. A simple < up-arrow + enter > keyboard sequence did this. As in section 3.5, the use of a KVM switch on these CRTs is not really helpful here.

6.0 Interface behavior analysis with "attack" traffic

We next implemented hping2 to create traffic that in fact did "crash" the interface, as the exploit information we used advised it would. Again we have what we believe is new behavioral material to report. Hping is discussed in [38]. We run it against the router using 12.0 IOS.

Hping is run from the command prompt in a terminal window, as are most of all of the really useful tools. The directory is at /usr/local/perftools/hping/hping-rc2 and the command is run as:

```
hping –c 100 –i u500 –0 –H 53 –t 1 –rand-source –d 500
10.73.73.110.
```

This command sends a count of 100 packets, at an interval of 500 usec (2000 pkts/second) using raw IP mode (not ICMP, UDP or TCP) with a protocol number of 53 (decimal), a time-to-live value of 1, random source IPs and a data length of 500 bytes. The destination address is that of the router. Material was located on the web, written by Pat Donahue and Martin Kluge, (posted to Bugtraq on 21 July), that outline some details on implementing the exploits and creating the framework for this testing. We are indebted to them for their contribution.

We saw that sending small groups of packets will increment the input counter by the same or nearly the same count (count – 1) e.g. sending 10 packets in a row rapidly will step the queue counter by 9 or 10, and this value stays there, not dropping back to zero. The continuous ping traffic remains at the normal rate. The " –c " (count) argument sets this value, and if set to 1 sends a single packet as desired.

An attacker can therefore use either of two strategies. (a) He or she can send a burst of 80 or more frames at once and fill the queue immediately. However an IDS (intrusion detection system) may well alert on this. If this is an inside attack, it may be injected inside of the point that edge router ACL's are effective, or where an IDS see it.

A second scenario is possible where the attacker could send single frames at widely spaced intervals. This may well escape IDS detection thresholds. In so doing, the queue could be slowly "stepped up" on an unpatched router to just below the "trip point", seeing as how the device apparently appears to work normally at that point. Then literally a single packet could be sent later to "finish the job" and it is quite possible this single packet would not be logged if administrators did not set trip points on their IDS to log data in such detail.

The behavior we have observed is that, once the queue reaches the 76/75 value, the interface does not immediately fail, but rather goes into a mode where, about 10 to 15 seconds after the final attack packet is sent, the throughput rate of the ping traffic, a steady, invariant value, starts showing some raggedness and "holes" in the throughput, where in this case, occasional frames (echo replies) start disappearing. The percentage of replies that "disappear" during this interval when the interface is "collapsing" never exceeds 10 to 15%, until about one minute after the final attack packet is received, when the replies totally stop. The end is quite abrupt, throughput then is zero, and the hard power reset is then called for.

Note that hping reports, as does icmp, the packets sent and received, and here, the absence of replies, e.g. 100 packets sent, zero replies, 100% loss.

7.0 Packet analysis information

The art of decoding packet captures is of special interest to this writer. We cite some worthwhile references [41] that we hope are of value and use to others in this area of interest.

The reader who is not familiar with this is recommended to run ethereal and/or tcpdump on a linux installation, and capture and baseline known traffic to gain familiarity with the contents of different types of traffic, and learn how to do decodes and look for faults that may be present.

Baselining a known good network is always worth doing. It is suggested that a complete list of local MAC addresses be kept on hand so that problem stations have a better chance of being identified.

8.0 Related Issues

Phrack magazine, issue 60 has some comment by FX of Phenoelit about Cisco that are of interest, in a piece about Cisco exploits [7]. It may be a bit dated by now. To quote,

> "According to Cisco Systems, about 85% of all software issues in IOS are the direct or indirect result of memory corruptions. By the time of this writing, yours truly is not aware of one single case, where overflowing something in Cisco IOS led to a direct overwrite of a return address ... Overflow bugs in Cisco IOS are fairly rare and not easily compared to one another ... If you find or know an overflow vulnerability for IOS 11.x and you think it is not worth all the trouble to exploit since everyone should run 12.x by now, let me challange this. Nobody with some experience on Cisco IOS will run the latest version. It just doesn't work correctly. Additionally, many people don't update their routers anyway."

This last sentence is some of the reasoning for our having done this paper.

9.0 Conclusion. What is new here?

Some of the topics and issues that were brought together here include Vulnerability Analysis, router technology and behavior, packet generation with standard test tools and with specialty tools, packet analysis using varying types of tools (linux and windows) and related router-centric security issues.

We strongly believe it is essential to have experience in generating and analyzing network traffic to permit studying and recreating newly found attack vectors, and hope that the description of this lab setup will provide guidance as to how such a system could be initially developed. Obviously this is a "bare-bones" system and has room for expansion and growth. As vulnerabilities seem to constantly be discovered, a means and ability of testing them is a valuable asset.

In conclusion, we hope that this project demonstrates that useful work can be done with limited resources (limited here primarily meaning self-funded)

Notes and Appendices

[1] There are about 10 pages of detail about the Cisco IOS password vulnerability in (Cole 660-670),

[2] "IOS Version Security", (Akin 6)

[3] The Cisco IOS is a binary image file from the standpoint of the user. There is very little information available about the internal architecture of IOS. The size can range from the 1 to 2 megabyte range up to nearly 15 megabytes depending

on the recentness of the version, host platform requirements, and the capabilities that are compiled into it, e.g. some images support IP only, some support IPX, Appletalk etc. as well, some support ISDN and firewalling etc..

What is available are descriptions of the command interface (hundreds of commands exist), and descriptions of the naming convention for the image, which describe the hardware platform, capability set, and software version, in the image file. This file is normally stored in flash memory and loaded into RAM memory on booting the router. A config file is also loaded from NVRAM into RAM on booting. This file is normally backed up or loaded from a TFTP server on the local network.

[4] Slashdot is an on-line news resource and comment board of a somewhat free-form nature, which posts 15 or 20 news stories each day and allows comments from readers which appear in threaded form, They style themselves "news for nerds, stuff that matters". http://slashdot.org.

[5] Techweb is a somewhat more "corporate" news resource that has more of an industry orientation but which has good links to current news item. It is a bit slower to be updated and does not cover the more esoteric stories. http://www.techweb.com

[6] The bugtraq and nt-bugtraq mailing lists are a vital resource for tech and security matters where contributors post messages for advice or discussion. See www.securityfocus.com and www.ntbugtraq.com for starters.

[7] Phrack issue #60 (===Phrack, Inc.=== Volume 0x0b, Issue 0x3c, Phile #0x07 of 0x10 |+----------={Burning the Bridge: Cisco IOS exploits by FX of Phenoelit ,fx@phoenoelit.de>) see http://www.phrack.org. (emphasis original)

[8] One may add an outbound default route when configuring RIP (etc.) as a way to reach an ISP or other outbound link from their site. In global config mode; here, we use

```
Router(config)# ip route 0.0.0.0 0.0.0.0 10.73.73.1
```

Allowing any network and any netmask not more specifically matched in route statements to reach our gateway. This is described on page 691, Network Consultants Handbook by Matthew Castelli, Cisco Press. (2002)

[9] Public NTP servers can be found by looking at lists available at www.ntp.org, or at the Mills site www.eecis.udel.edu/~mills/ntp/clock2a.html.
Look for a Stratum 2 server at the highest, (don't peer with a Stratum 1) and one with open access that is on a short routing path to your network.

[10] The original Cisco vulnerability announcement was posted to Bugtraq on 17 July 2003, titled Cisco Security Advisory: Cisco IOS Interface Blocked by IPv4 Packet (rev. 1.3) and said,

"Cisco routers and switches running Cisco IOS® software and configured to process Internet Protocol version 4 (IPv4) packets are vulnerable to a Denial of Service (DoS) attack. A rare sequence of crafted IPv4 packets with specific protocol fields sent directly to the device may cause the input interface to stop processing traffic once the input queue is full. No authentication is required to process the inbound packet. Processing of IPv4 packets is enabled by default. Devices running only IPversion 6 (IPv6) are not affected. A workaround is available

Cisco has made software available, free of charge, to correct the problem.

This advisory is available at
http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml

Affected Products

This issue affects all Cisco devices running Cisco IOS software and configured to process Internet Protocol version 4 (IPv4) packets. Cisco devices which do not run Cisco IOS software are not affected. Devices which run only Internet Protocol version 6 (IPv6) are not affected.

Details

Cisco routers are configured to process and intercept Internet Protocol version 4 (IPv4) packets by default. A rare, specially crafted sequence of IPv4 packets with protocol type 53 (SWIPE), 55 (IP Mobility), 77 (Sun ND [network disk –ed.]), or 103 (Protocol Independent Multicast – PIM) which is handled by the processor on a Cisco IOS device may force the device to incorrectly flag the input queue on an interface as full, which will cause the router to stop processing inbound traffic on that interface. This can cause routing protocols to drop due to dead timers. Interfaces which are explicitly configured to run PIM will not be affected by traffic with protocol type 103. An interface with PIM enabled will have one of the following three commands in the interface configuration: ip pim dense-mode, ip pim sparse-mode, or ip pim sparse-dense mode.

On Ethernet interfaces, Address Resolution Protocol (ARP) times out after a default time of four hours, and no traffic can be processed. The device must be rebooted to clear the input queue on the interface, and will not reload without user intervention. The attack may be repeated on all interfaces causing the router to be remotely inaccessible. A workaround is available, and is documented in the Workarounds section."

See the advisory for more details, including affected IOS versions..

[11] The Cisco 25xx (2500-series) is a low-end access device with a fixed configuration which is currently end-of-life. It is replaced by the 2600-series which offers new hardware options including 10/100 MBit Ethernet. The 2500 can support IOS through version 12.3 and can accept 16 megabytes of flash memory and 16 megabytes of RAM, which hold the software image, and the running configuration and data respectively. The device fits into a 1U rack space (using optional brackets) and can be found widely on the used equipment market. Options for Ethernet, token-ring, serial sync and async, ISDN BRI are all available by model number. Devices used here employed only Ethernet AUI ports and serial ports.

[12] Red hat linux 9 is the last of the “boxed set” distributions offered for sale at a retail level. Red Hat is moving to an Enterprise version aimed at the corporate market. Possibilities for the user include finding a copy of the distribution, obtaining the Fedora distribution which is a replacement with developer support, using another distribution such as SuSE, Mandrake or Debian, or running a distribution such as FreeBSD which is a good desktop but is more suitable for the server market and is optimized for that role.

[13] In addition to Knoppix-3.3 it is easy to load the Knoppix STD (Security Tools Distribution), as well as Phlak, FIRE, TPM-SecurityServer, or Local Area Security. Some of these use dhcp for address granting, at least one of them requires the user to add network address and route information manually. These provide added functionality to allow the user to experiment with other diagnostic tools. We recommend trying STD.

[14] The DOS program “PC-Plot” was a VT-100 type terminal emulator, and a Tektronix 4014 graphics display device emulator. Such a terminal capability today would be achieved by using a program such as minicom in Linux, or hyperterminal (offered with Windows) or TeraTerm (a shareware running under Windows that does a good VT-100 emulation.

[15] The CON port (Console port) on 250x routers is available on a RJ-48 compatible 8-pin connector on the router. A “roll cable” (rollover) allows connection to a 8-pin to DB-9 adapter which permits connection to a DB-9 serial port. The roll cable is crimped using 8-conductor flat ribbon cable with pin 1 flipped to pin 8 at the other end and vice versa. This is Cisco part number 72-0876-01 and the adapter is p/n 74-0495-01. These are commonly available on eBay for a few dollars. The aux port is a similar connection device on the router intended for a modem connection. Use of dialable modem connections is a risk.

[16] The 2511 router allows single serial line access and offers 16 asynchronous ports and one of two special “octal” cables called “ASYNC BREAKOUT 72-0845-01” which allows connection to 8 of those ports (ports 1 through 8 or 9 through 16) by a fan-out to eight RJ-48 connectors which connect to the CON ports of eight other routers. A procedure called “reverse telnet” can be configured on the 2511 to allow a single console on the 2511 CON port to telnet to the other eight routers, in turn (one at a time). The telnet command on the 2511 uses up to 16 port numbers above 2000 embedded in the command, to access one of the eight other routers.

By far the best description we have seen of configuring reverse telnet is in (Heap and Maynes 58-67). See also (Harrington 792).

[17] PacketView is offered by Klos Technologies. See www.klos.com. This program, designed to run with MS-DOS and a packet driver, may not have been updated for some time; the most recent known version is 1.23.

[18] Landecoder/e is an offering of Triticom Systems, see www.triticom.com. The version used here was 3.0 and worked best with a patch from the manufacturer, use of an Open Datalink Interface driver, and a 10-MBit ISA card that used I/O interrupts in the 0x60 range. We were unable to make a 10/100 PCI card work with this software due to IRQ handling.

[19] Shomiti Surveyor www.shomiti.com Shomiti has been absorbed by Finisar in 2001. The current product line supports gigabit Ethernet, VoIP, QoS, H.323 decodes, RTP streams etc. as well as the standard protocols.

[20] Flooding traffic can have a specific meaning such as in BGP routing updates when routers see a topology change or in VTP pruning, which are of no significance here, or it can generically mean any kind of junk traffic that is flooded to an interface at a high rate, to try to deny the interface from receiving wanted traffic. Here we flooded the eth0 interface on the router with data to see how the input queue behaved as the offered load increased.

[21] The Smartbits hardware based traffic generator is widely respected as a source of load-based traffic testing. See: www.netcomsystems.com

[22] Here is info for the IOS version for each router, for “sh ver” and “sh flash:”

number 2	number 3	number 10	number 2511
IOS C2500-I-L	**IOS** C2500-JS-L	**IOS** C2500-JS-L	**IOS** C2500-JS-L
Ver 12.0(17a)	Ver 12.2(10a)	Ver 12.2(19)	Ver 12.2(12a)
RELEASE (fc1)	RELEASE (fc1)	RELEASE (fc1)	RELEASE (fc1)
11-Feb-02 kellythw	21-May-02 pwade	12-Aug-03 kellmill	24-Sep-02 pwade
ROM BOOTSTRAP			
Ver 5.2(8a) REL	Ver 5.2(8a) REL	Ver 11.0(10c) SW	Ver 11.0(10c) SW
PROCESSOR			
Rev F 16384/2048	Rev D 16384/2048	Rev L 14336/2048	Rev M 14336/2048
FLASH			
5,743,000 b.	15,694,664 b.	15,965,420 b.	15,778,868 b.

(The 2511 information is for reference and comparison purposes only)

[23] Netperf and iperf are open source throughput testing tools available to anyone by downloading the source and compiling and installing. We have had good luck installing these packages onto linux systems. Probably the most thorough treatment of these programs is found in (Blum 61-77, 99-115).

[24] iperf description: see netperf , above.

[25] The Adtran TSU is made by Adtran, Inc. Huntsville AL 35806. These devices are increasingly common on eBay as users move away from T-1 installations to fiber or microwave solutions.

[26] A crosswired connection is made when pin 1 and 2, the receive data ring and tip, are crossed over to pins 4 and 5, the transmit data ring and tip. The connector uses the USOC RJ-48C standard, (Adtran 2-21)

[27] Iperf runs in client and server mode on 2 machines and creates a true TCP connection, which runs through both of the routers in the lab. The slowest part of the path defines the throughput and reliability of the connection.. The package is installed here in /usr/local/perftools/iperf_170/ and is run as root. At 192.168.30.51 we run the command

```
./iperf –s –f k –c 10.73.73.52
```

and the program returns the following information: (run this server command first)

Server listening on port 5001
TCP window size 85.3 Kbytes (ed.: this is default)
[4] local 192.168.30.51 port 5001 connected with 10.73.73.52 port 3804 (varies).

[ID] Interval	Transfer	Bandwidth
[4] 0.0 – 10.6 sec	1856 Kbytes	1441 Kbits/sec

On the 10.73.73.52 machine, run the client as

```
./iperf –c 192.168.30.51
```

and the program returns the following information:

Client connecting to 192.168.30.51 TCP port 5001
TCP window size 16.0 Kbyte (ed.: this is the default)
[3] Local 10.73.73.52 port 3819 connected with 192.168.30.51 port 5001

[ID] Interval	Transfer	Bandwidth
[3] 0.0 – 10.5 sec.	1.82 Mbytes	1.46 Mbit/sec

Notice the absence of “-f k” in the command switch, reporting results in MB and not KB

The reported bandwidth, which varies by perhaps 1 % on successive 10-second runs, is consistent with that of a T-1 line. Inclusion of frame header data (not payload) adds this small amount of overhead, which iperf does not count as data. 1441/1544 = .933, or 6.7% overhead.

[28] SNMP is the Simple Network Management Protocol. This allows polling of a device from a remote location over a network connection according to well-known values of certain system properties. The Management Information Base contains the descriptors of these properties. The protocol runs via UDP (connectionless) on ports 161 and 162.

Cisco is one example of a manufacturer having their own registered MIB values. Common types of values that can be returned by SNMP polling include system uptime, interface statistics, other operation parameters. Traps exist, which is a method of sending unsolicited alerts or warnings to the remote management console.

RFC 1157, 1187 and 1213 are original RFCs. Today version 3 exists with increased security attributes and is described by RFC 2271.

[29] Snmpwalk is a program to use SNMP to "walk" down the "tree" of MIB values and return those values methodically. We can run this against the 12.2 router called 10:

```
#Snmpwalk –v 2c –c public 10.73.73.110 > snmp10.txt
```

and then parse the output file snmp10.txt. A few returned values chosen at random are:

SNMPv2-MIB::sysUpTime.0 = Timeticks: (511091378) 59 days, 3:41:53.78
RFC1213-MIB::atNetAddress.1.1.10.73.73.53 = Network Address: 0A:49:49:35
SNMPv2-SMI::transmission.7.2.1.5.1 = counter32: 2677
SNMPv2-SMI::mib-2.16.19.6.0 = STRING: "flash:c2500-js-l.122-19.bin"

These are selected from 740 items returned. The command "`cat snmp10.txt | wc –l`" (counts the number of lines in a text file) returns 743 lines and the SysDescr(iption) single item filled four lines.

[30] Cisco MIBs are searchable through http://www.cisco.com/public/mibs. Some trees to pursue are at 1.3.6.1.4.1.9.2.1.X for general router buffer stats (the basic device MIB) (iso.org.dod.internet.private.enterprises.cisco.local.lsystem.X) and another is 1.3.6.1.2.1.4 which is the general MIP for IP.

A nice graphical site is at http://carsten.familie-doh.de/mibtree/cisco.html which shows graphically the MIB Enterprises Cisco 1.3.6.1.4.1.9 x where x is 5 (workgroup) – 1 (stack) – then system, chassis, module, port, monitor.grp (1,2,3,4,8).

At the same site, /mibtree/cisco-mgmt.html the Cisco MGMT MIB Tree 1.3.6.1.4.1.9 9.X leads to 20 (channel), 37 (queue), 48 (MemoryPool), 84 (ciscoIpStat), 131 (ciscoSystem – clock etc.) and 151 (ciscoL2L3IfConfig).

If there is a usable MIB for the input queue it is possibly in this area but it hasn't yet been found. We include this level of detail to encourage further discovery.

[31] RIP v1 is the first version of the Routing Information Protocol. This works by sending lists of all routes the router believes it can reach, every 30 seconds, via a UDP broadcast to the local subnet (MAC destination FF.FF.FF.FF.FF.FF and IP destination 255.255.255.255 on port 520). A typical RIP frame will include data which decodes as:

Family ID: 2 IP Address: 10.73.74.0 Distance: 1
Family ID: 2 IP Address: 192.168.31.0 Distance 2
Family ID: 2 IP Address 192.168.30.0 Distance: 2
Family ID: 2 IP Address: 172.16.0.0 Distance: 1

RIP does not scale well for large networks but works well here. Today an enterprise site would likely use EIGRP, the Enhanced Interior Gateway Routing Protocol. RIP was described in RFC 1058 and the version 2 in RFC 1723. See (n.a. Cisco Press 675 et seq.)

[32] Other traffic of no consequence are the loopback broadcasts (ethertype 9000) and the CDP neighbor discovery broadcasts (Layer 2, 802.3 type) every 30 seconds. These can be of use to ensure the Adtran serial link is functioning if ICMP was not connecting for some reason.

[33] It was decided early on that each device here with a unique MAC address would be given a unique number identifier, allowing use of as many as 254 devices. Thus any given PC with an Intel NIC, or a 3Com NIC, etc. could run any kind of software, but each instance of software on that machine would be given the same IP address. With as many as 8 to 10 machines in service at some point, this helps with recordkeeping. Following this scheme, if our Dell Optiplex we call 51 was on the home LAN, it would be assigned 10.73.73.51 so that it would have a permanent presence there. Upon moving it to this test setup, where it was placed on the 192.168 net, it is then given 192.168.xx.51 (the third octet was chosen arbitrarily here). Depending on swapping of hard disks, any of Solaris, FreeBSD, Linux, might be run in turn on that same machine, and each install would have an IP matching xx.xx.xx.51 on that machine. A Windows 2000 install on "51" then would also have a NetBIOS name of "FIFTYONE".

[34] ICMP is the Internetwork Control Message Protocol which provides error reporting for IP which is itself connectionless, and cannot report back error status. Included in ICMP is what is its probably most widely known feature, echo request and reply, designed to test if a remote host is reachable, since IP itself cannot provide this information. This is commonly known as a "ping". Lore has it that the "description" of "ping" was reverse-engineered to mean "packet internetwork groper"

[35] ping: For reference, the rtt (round trip time for any one ping to be answered) of 64 byte pings varies from about 4.6 ms to about 6 ms. The rtt of the 1450-1500 byte ping range runs 21.3 to 21.4 ms. The delay is due to the serial link speed. As a comparison, when 192.168.30.51 pings 192.168.30.1 the rtt is typically 2.4 to 2.8 ms at 64 bytes, and is typically 4.96 ms when using a length of 1420 bytes. These values of course are true once the ARP resolution is done; the first packet is typically somewhat longer to respond when ARP resolution is needed.

We were able to change parameters on the ping flows to crank up the utilization to about 27%, at which point the rtt of the pings had increased by about 10 times, to a few hundred ms and the sequencing became quite scrambled due to congestion. We stayed with 10% utilization for the testing.

[36] tcpdump (see 4.2b). This comes with virtually all linux and other unix-workalikes. This is a very well-known workhorse and needs no introduction from us. See http://www.tcpdump.org. The latest version is 3.8.1.

[37] hping information is at www.hping.org. It is available in compiled form on some of the recently available boot-from-CD distributions. It is considered a security auditing tool and a TCP stack analysis tool. This quote from the Phlak website www.phlak.org which is host of the Hackers Linux Assault Kit says,

"hping2 - hping is a software to do TCP/IP stack auditing, to uncover firewall policy, to scan TCP port in a lot of different modes, to transfer files across a firewall, test network performance, test of TOS is handled, etc."

[38] Use of Windows vs linux machines: Windows machines are inherently "chatty" as they are always broadcasting NetBIOS name registration requests etc. Running a default configuration with a windows-based analyzer such as Observer will give your identity away to any other device monitoring the same network you are on. A linux machine, if it is not ARP'ing for a gateway, or a NTP server, or syslog server etc. will be much more quiet. The optimal configuration here is to use one of the DOS-based analyzers referred to. These machines never transmit anything onto the network unless the user forces it to do so. The machines don't even have an IP address; programs that test for promiscuous interfaces on Unix machines should not see them. For information, see *cpm* ("check promiscuous mode") reference (Sloan 131) or search on CERT/CC, the program's origin.

Another DOS based program is "Dr. Watson, the network detective's assistant" (DWTNDA) (Not the NT error diagnostic Dr. Watson) which can be found at www.cavebear.com. This program, which works with a packet driver, will let the user transmit some diagnostic data, including SNMP.and do limited capturing. This little program has its own TCP stack and is light enough to fit on a micro DOS box using a legacy NIC.

[39] The Network General "sniffer" ™ product referred to here was described in Operation Manual PA-302 dating to around 1989 or 1990 and is of historical interest.

[40] Data content in some tests default to a payload of 0x00 (null character), while others allow an arbitrary payload, in which case a useful value is 0x55 (ASCII character "U", binary 01010101). The use of "U" is good in that it causes the maximum number of state changes at the PHY level, per character, which causes the greatest "exercise" of the transmission media at a layer 1 level, i.e voltage changes in the hardware and processor ICs. Tests using 0x00 (all null's are useful when one's insertion needs to be done to provide clock synch. (refer to "zero's density rule" etc.) The hping tests use 0x58, or, "X" for data field but this can be changed by the user. The unix ping uses incrementing chars from 0x01 to 0x7e or 0xff, while many windows versions of ping use the chars from "a" through "w" (0x61 – 0x77, decimal 97 through 119), all in repeating loops.

[41] Packet analysis information is available in two excellent books, (Haugdahl), and (Wilson). Wilson also includes a CD-ROM with files designed to work with Microsoft's *Network Monitor* program. This has a heavy emphasis on SMB and other Microsoft-specific troubleshooting. Haugdahl also includes a CD-ROM and is somewhat more generic, and even has a section on Token-Ring decodes. Coincidentally they both have almost the same number of pages (357 vs. 359). There is a Sniffer book available also which is occasionally seen in bigger bookstores although not lately by the author.
.

Works Cited

n. a. Adtran Corp. 61200.060L1-1A user's manual for Adtran T1-FT1 Data/Channel Service Unit, Adtran: Huntsville, AL, March 1994.

n. a. Cisco Press: Internetworking Technologies Handbook, 3rd edition, chapter 47. Indianapolis: Cisco Press, 2002 rev.

Akin, Thomas. Hardening Cisco Routers. Sebastopol, CA., O'Reilly, 2002

Blum, Richard. Network Performance Open Source Toolkit. Indianapolis: Wiley, 2003.

Cole, Eric. Hackers Beware (defending your network from the wily hacker). Indianapolis: New Riders, 2003.

Harrington, Donna. CCNP Practical Studies: Troubleshooting. Indianapolis: Cisco Press, 2003.

Haugdahl, J. Scott. Network Analysis and Troubleshooting. Reading, MA: Addison-Wesley, 2000.

Heap, Gary; Maynes, Lynn. CCNA Practical Studies. Indianapolis: Cisco Press, 2002

Sloan, Joseph D. Network Troubleshooting Tools, Indianapolis: O'Reilly, 2001

Wilson, Ed. Network Monitoring and Analysis – A Protocol Approach to Troubleshooting, Upper Saddle River, N. J. Prentice-Hall PTR, 2000

Network diagrams, packet decodes, and illustrations are given on the following pages:

Illus. 1. Logical layout of test facility, showing two hubs B and C (A not used)

Illus. 2. PV decode (summary view) of Shomiti test output, protocol type 99, inter-frame gap is about 7 ms, 150 pkts/sec, a low rate

Illus. 3. Example of detail view of PV decode of test packet, MAC (Ethernet decode). Generated to prove out the capture and decode of the test gear.

Illus. 4. Same as 3 but showing protocol-level decode

Illus. 5. Example of Ethereal decode of the same packet, useful to confirm decoding accuracy and consistency of all three tools

Illus. 6. Example of LD/E decode in detail window of same test packet, detail window at top.

Illus. 7. Same as 6 but viewing the bottom of the decode window.

Illus. 8. LD/E decode in skyline view of ping traffic through test interface on router, continuously running at 10% network utilization. The middle of the graph (viewed horizontally) shows flood of traffic from the Shomiti set for 98% network utilization with frame size = 1518, while ping traffic continued. Note that ping traffic is still present at 10% showing that throughput continued through the interface in both directions. This shows that the interface stayed up even though showing 76/75 on the console when polling the "show interface ethernet0" command on the router terminal. The module traffic cannot replace the pings. The true rate from the module is pulled down to 88% utilization, even though

having been set to supply 98%. Stopping the ping traffic allowed it to again reach 98% reported utilization. The frame error rate was 40 to 50 frames/sec with the pings running, and dropped to zero when the pings stopped, indicating collision activity.

Illus. 9. LD/E summary view (capture mode) showing five distinct traffic flows (1) Flood from Shomiti averaging 407 pkts/sec. This shows the arbitrary self-assigned source MAC, the destination MAC being that of the router interface, 27K frames of length 128, set for 5% utilization. (2) Ping traffic running at 10% utilization. Note the equal counts for flows A to B and for B to A with average frame size 1485 bytes. These did not affect the test: (3) Loopback broadcasts from router, identifiable as the source and destination MACs are the same; length 64 bytes. (4) RIP broadcasts from the router which are identifiable as the destination MAC is Broadcast (all ff) and the frame size is 130, which depends on the routing table size. (5) Cisco Discovery Protocol broadcasts from the router. This is a layer-2 protocol for neighbor discovery using a multicast address of 01:00:0c:cc:cc:cc. Multicast uses an odd byte in the first character which is typically 01, 03 (NetBIOS raw), 09 or ab (DecNET). This capture ran for 1 min 40 seconds, or, 100 seconds, which makes getting per-second rates easy. At that time the network was disconnected and the counts halted. The elapsed time on the panel keeps running until the screen print is saved somewhat after the test ended. The column view is distorted due to formatting characters embedded in the screen display which confuses the printer. The correct values are Frames A to B, Frames B to A, Average length of A, Average length of B, Errors A and Errors B.

Illus. 10. Router stats for Ethernet interface during the time the test in 9 was running. Note that the input queue was showing full, but was processing traffic. See also [28].

Illus. 11. LD/E detail of flood from Shomiti at 1518 bytes, 99% utilization. This demonstrates the LD/E was handling the packet count load (if not the decode load) at this rate. 797 frames/sec X 1518 bytes/frame = 1,209,846 bytes/sec X 8 bits/byte = 9,678,768 bits/sec, which is 96.78 percent utilization at 10 Mbit/sec.

Illus. 12. Graphs are shown (12 through 18) of input queue behavior on the router 10 (newer, non-vulnerable 12.2 IOS) when flooded with Shomiti traffic, protocol type 57, but not causing any interface to fail. Ethertype is 0800. We feel it is useful to provide this information for the newer version of software available. Packet length = 64 and the queue reached 76% at 5.6 % utilization.

Illus. 13. Packet length = 128, queue filled at 10 % utilization (on router 10 with newer IOS), queue was at 60 at only 4% utilization with older software i.e. the queue filled twice as fast.

Illus. 14. Packet length 256, queue fuilled at 18 %.

Illus. 15. Packet length 512, queue filled at 32%.

Illus. 16. Packet length 1024, queue filled at 58%.

Illus. 17. Packet length 1518, queue filled at 75%.

Illus. 18. Graph curve family showing the rate at which the input queue fills, as a function of the number of packets/second, for six different packet lengths (see section 5).

Illus. 19. Attack traffic Packetview decode (in summary view) of attack traffic generated by hping, as described in section 6. Note the use of random source IPs as suggested last July in advisories posted to Bugtraq. It is reasonable to assume that an attacker on the local LAN would use this decoy method to try and hide his origin. If this traffic was originating outside the local LAN, it should be stopped by router ACLs. If it is on the same local Ethernet, it may be identified by the source MAC if that has not also been disguised. For this to be effective, the local network admin needs to keep track of all of his/her local MAC addresses. Note that with the –u500 switch, the IPGs (inter-packet gap, the inverse of utilization) were typically 7 to 10 ms, but the gap is not at all consistent. Illustration 2, taking during the spec'ing process, demonstrates that the analyzer decode is quite uniform and that the variation in timing is in the app.

Illus. 20. PacketView detail decode; IP header checksum is good.

Illus. 21. LanDecoder/E detail window, compare style with 20 above.

Illus. 22. PV decode of the final single "attack" packet sent by hping at 46178, no reply at 46214. Clock was locked up but the rate was the same as that in 19. The stopped clock, noticed later, is apparently a software bug when the analyzer capture buffer fills.

Illus. 23. Decode showing "ragged" throughput in which some reply traffic is being lost at random, in the half minute after the attack packet was sent. Note there are 47 echos and 23 replies, which would show roughly as 7 to 8 % utilization.

Illus. 24. Throughput of the ping traffic stops totally at 48428. The machine 52 sending the echo requests ARPs for its now "missing" gateway connection at 48429.

Illus. 25. Skyline view on LD/E showing gaps in throughput during another run of the attack scenario, showing an approximately 45 second interval as the interface becomes progressively more unreliable and then dies completely. The placement of the attack packet on the skyline is off the left edge of the window at

about 2 cm (original scale) or about 10 seconds previously. The point at which the interface dies is approximately 2.5 cm from the right hand edge of the window (original scale) At this latest point, the analyzer is seeing ping requests only and the "rate" is 5 %.

Illus. 26. MAC level decode on LD/E of ping traffic transiting the ethernet0 interface after sending the final attack packet. This shows that some of the ping traffic is apparently becoming corrupted. Refer to the frames showing as 500 to 600 bytes with a CRC error (right hand margin). These corrupt frames would be invisible to higher layers as the NIC ordinarily would discard them. This would appear as reduced throughput to the user. This print is a detail view at later one of the traffic visible in [25] in skyline mode. Viewing this same data at later 2 shows the valid frames as ethertype 0800 (IP).

Illus. 27. IP level view of attack traffic and ping traffic running simultaneously. This gives a good indication that each traffic flow is roughly balanced, in a 3/2 ratio, in terms of the per/second rate. This particular window shows 28 IP "attack" packets and 17 ICMP packets.

Illus. 28. Screen print of VT-100 terminal on PC (same as [10]) showing the input queue as 70/75 after 72 attack packets had been sent, in order to fill the queue to this point. When left at that level for several minutes, the throughput of the ping traffic was not affected.

Illus. 29. Photograph of three linux Dell Optiplex GX-100 units on tabletop underneath keyboards, left to right are units 51, 52 and 53. Unit 51 generates pings in five terminal windows, 52 generates attack traffic and 53 runs tcpdump to check and verify real-time traffic flow parameters. The machine at right (Windows 98) is not part of the test but was configured with a flatbed scanner.

Illus. 30. CRT on left is VT-100 ttype terminal used as the router console. The second CRT is showing PV in "continuous capture" mode. The DOS based "pizzaboxes" are underneath the CRT shelf.

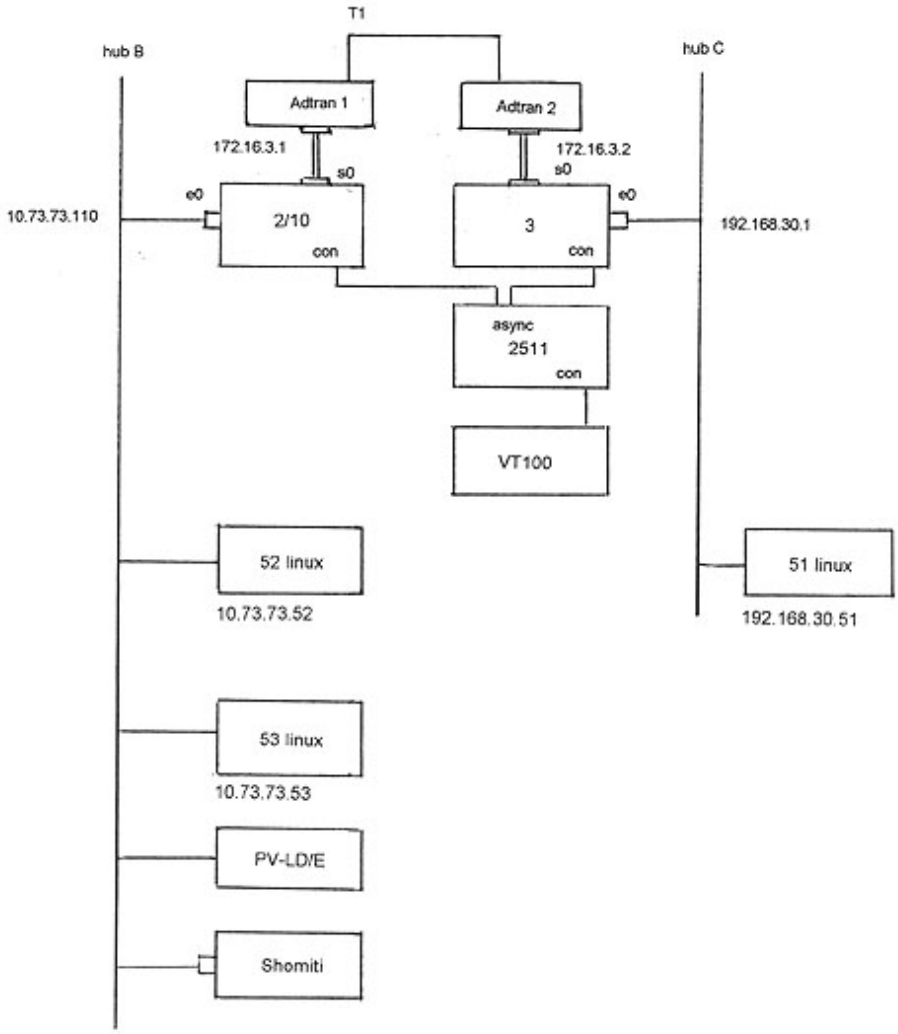
Illus. 31. Close up view of the VT-100 terminal to left, and LD/E in throughput "skyline" mode. The two Adtrans connecting the routers are visible at upper left on top of the CRT.

Illus. 32. The CRTs visible are for the machines for PV, for Windows XP (not part of the test) and the Shomiti. The last 2 machine cases are on the tabletop.

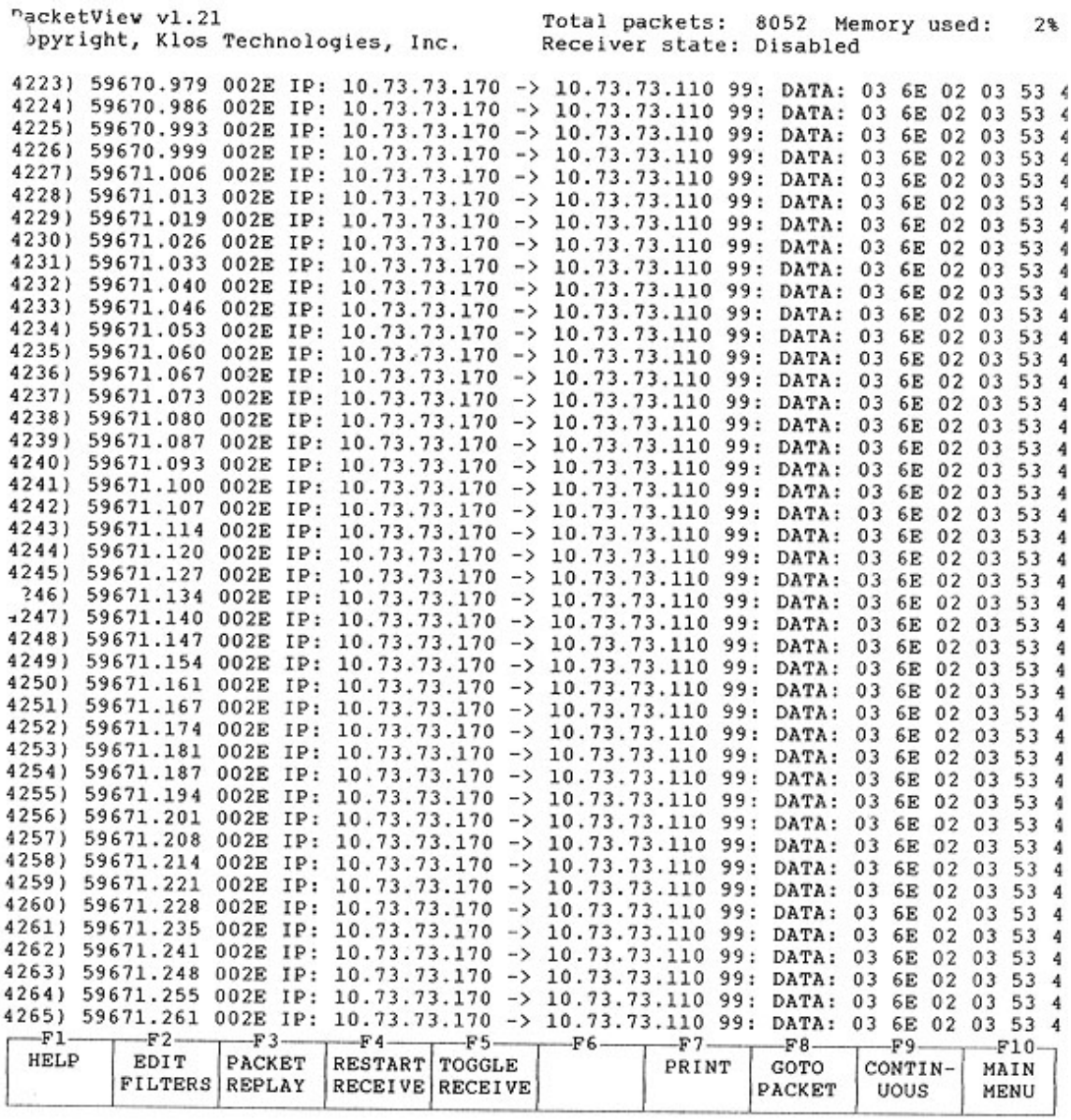
Illus. 33. The Shomiti screen (at right) showing the continuous traffic generation mode, and in foreground is the window for the module configuration mode in which the parameters such as length, rate, ethertype and data fields are set.

Illus. 34. Router stack in relay rack. Those routers that are part of the test are the type 2501 called “10” (underneath the 2364 CSU’s, Ethernet disconnected) and the 2511. Below the 2511, which shows the async octal cable, are the two EN108 hubs. The two routers at top, and the 2364 are not part of the test.

All graphic images and photographs following, were created by the author



1



2

```
PacketView v1.21                          Total packets:  8052  Memory used:   2%
Copyright, Klos Technologies, Inc.        Receiver state: Disabled

    ETHERNET MAC:
                                    Size: 003C  Number:   4254
                                                Time: 59671.187

    MAC DATA:
    0000   00 B0 BF 01 02 03 00 B0-BF 00 AB CD 08 00 45 00   .0?....0?.+M..E.
    0010   00 40 AB CD 00 00 40 63-27 3F 0A 49 49 AA 0A 49   .@+M..@c'?.II*.I
    0020   49 6E 03 6E 02 03 53 41-4E 53 47 49 41 43 00 00   In.n..SANSGIAC..
    0030   00 00 00 01 00 00 00 00-00 00 00 00               ............
```

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
HELP	EDIT FILTERS	PACKET REPLAY	RESTART RECEIVE	TOGGLE RECEIVE		PRINT	GOTO PACKET	CONTIN-UOUS	MAIN MENU

3

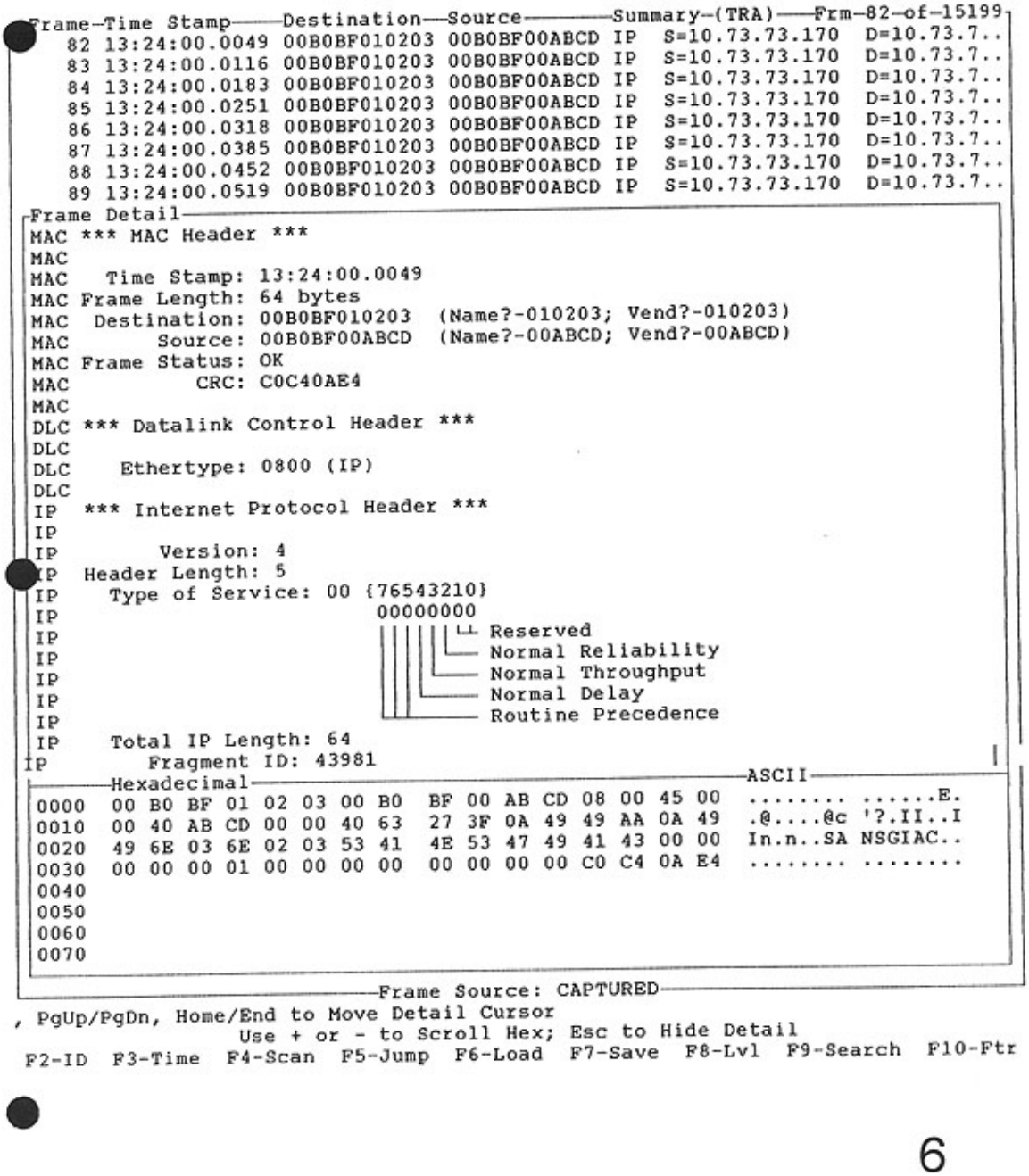
```
PacketView v1.21                          Total packets:  8052  Memory used:   2%
Copyright, Klos Technologies, Inc.        Receiver state: Disabled

    DIX:
        Destination: 00B0BF010203  Size: 002E  Number:   4254
        Source:      00B0BF00ABCD  Type: 0800  Time: 59671.187

    IP:
    IP version: 4  IP header length: 5 (32-bit words)
    Type of service: 00
        Precedence = Routine
        Delay = Normal
        Throughput = Normal
        Reliability = Normal
    Packet length: 0040  Packet ID: ABCD
    More fragments: NO  Fragment offset: 0000
    Time-to-live: 64  Protocol: 99  Header checksum: 273F (BAD! [26E4])
    Source host id:      10.73.73.170
    Destination host id: 10.73.73.110

    IP Data:
    0000  03 6E 02 03 53 41 4E 53-47 49 41 43 00 00 00 00   .n..SANSGIAC....
    0010  00 01 00 00 00 00 00 00-00 00                     ..........
```

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
HELP	EDIT FILTERS	PACKET REPLAY	RESTART RECEIVE	TOGGLE RECEIVE		PRINT	GOTO PACKET	CONTIN-UOUS	MAIN MENU

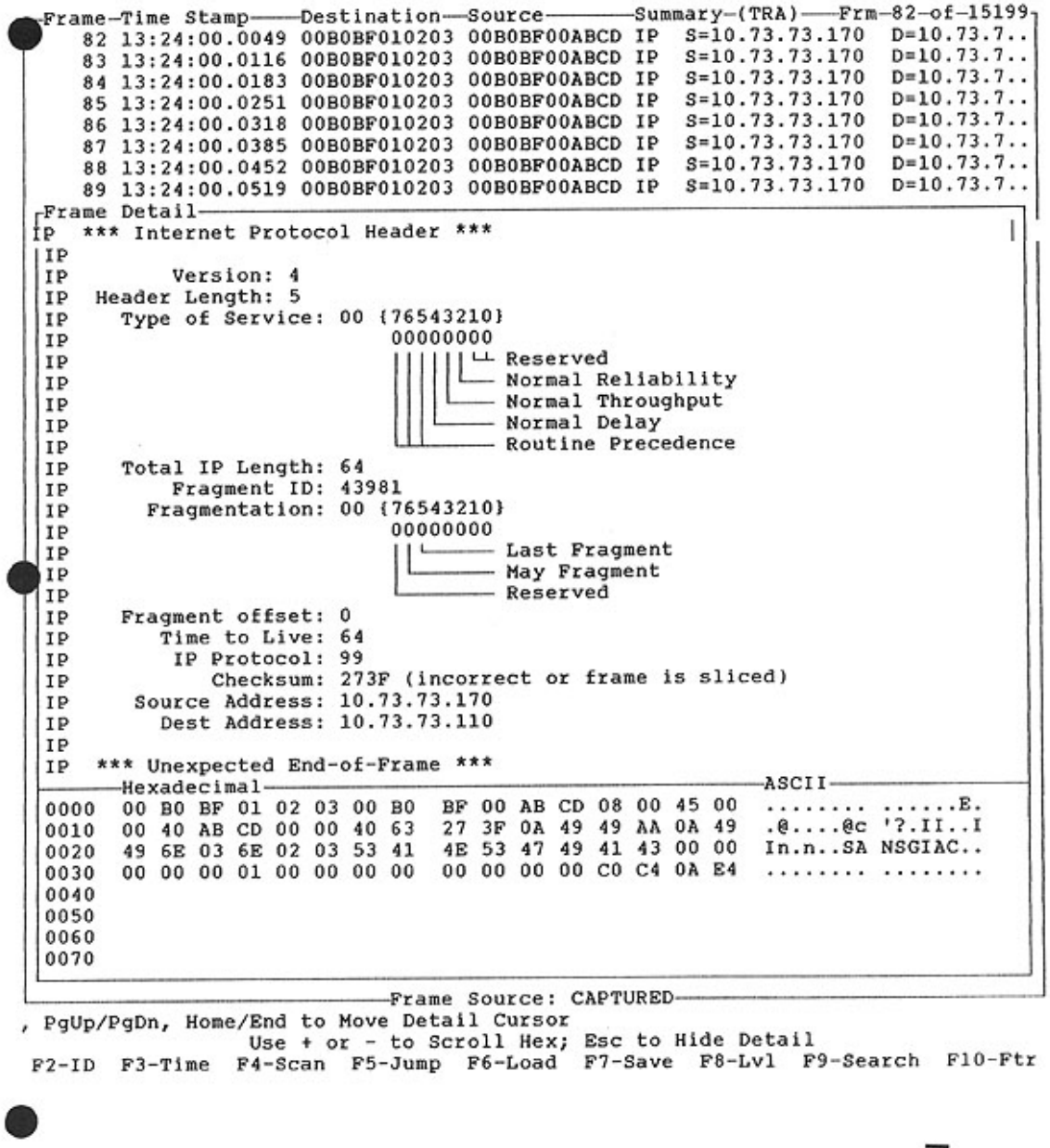
4

```
Frame 9176 (60 bytes on wire, 60 bytes captured)
    Arrival Time: Dec 10, 2003 08:48:57.926797000
    Time delta from previous packet: 0.006713000 seconds
    Time relative to first packet: 35745.171729000 seconds
    Frame Number: 9176
    Packet Length: 60 bytes
    Capture Length: 60 bytes
Ethernet II, Src: 00:b0:bf:00:ab:cd, Dst: 00:b0:bf:01:02:03
    Destination: 00:b0:bf:01:02:03 (00:b0:bf:01:02:03)
    Source: 00:b0:bf:00:ab:cd (00:b0:bf:00:ab:cd)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.73.73.170 (10.73.73.170), Dst Addr: 10.73.73.110 (10.73.73
.110)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 64
    Identification: 0xabcd
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: Unknown (0x63)
    Header checksum: 0x273f (incorrect, should be 0x26e4)
    Source: 10.73.73.170 (10.73.73.170)
    Destination: 10.73.73.110 (10.73.73.110)
Data (26 bytes)

0000  03 6e 02 03 53 41 4e 53 47 49 41 43 00 00 00 00   .n..SANSGIAC....
0010  00 01 00 00 00 00 00 00 00 00                     ..........
```

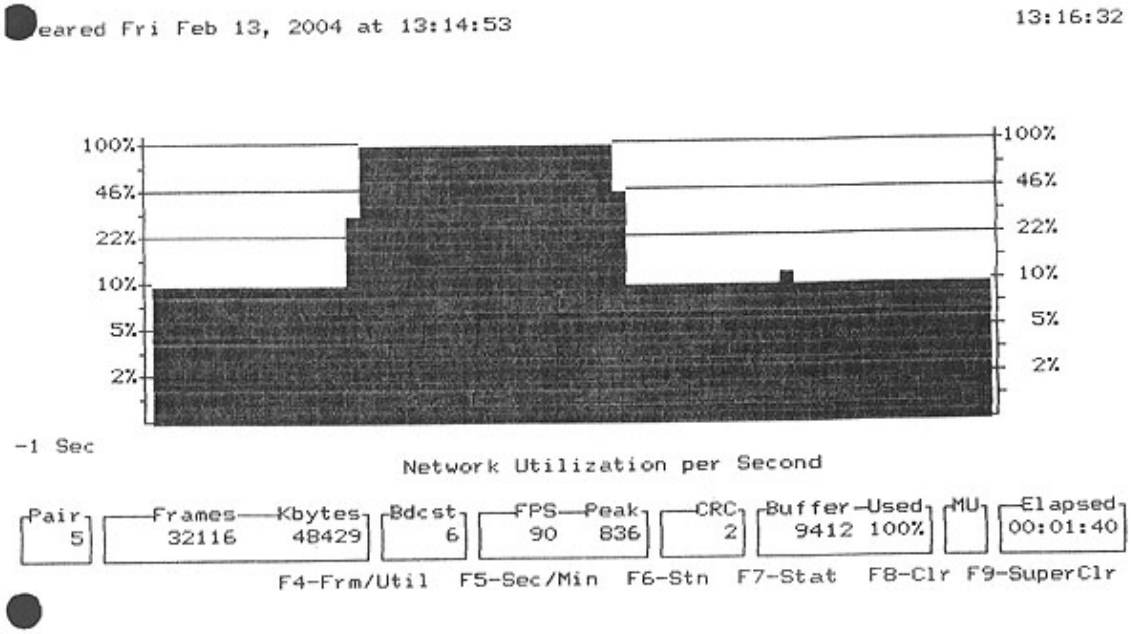
5



6



7



8

```
Cleared Fri Feb 13, 2004 at 12:42:27                                    12:46:40
Network StnANetwork StnB  Frames AB Frames AAve A Ave B  Err AB Err A
00605CF4941800B0BF00ABCD          0       40766      0   128         0       0
00605CF4941800C04F0B915B       4492        4491   1485  1485         1       0
00605CF4941800605CF49418         10           0     64     0         0       0
00605CF49418FFFFFFFFFFFF          4           0    130     0         0       0
00605CF4941801000CCCCCCC          1           0    307     0         0       0
```

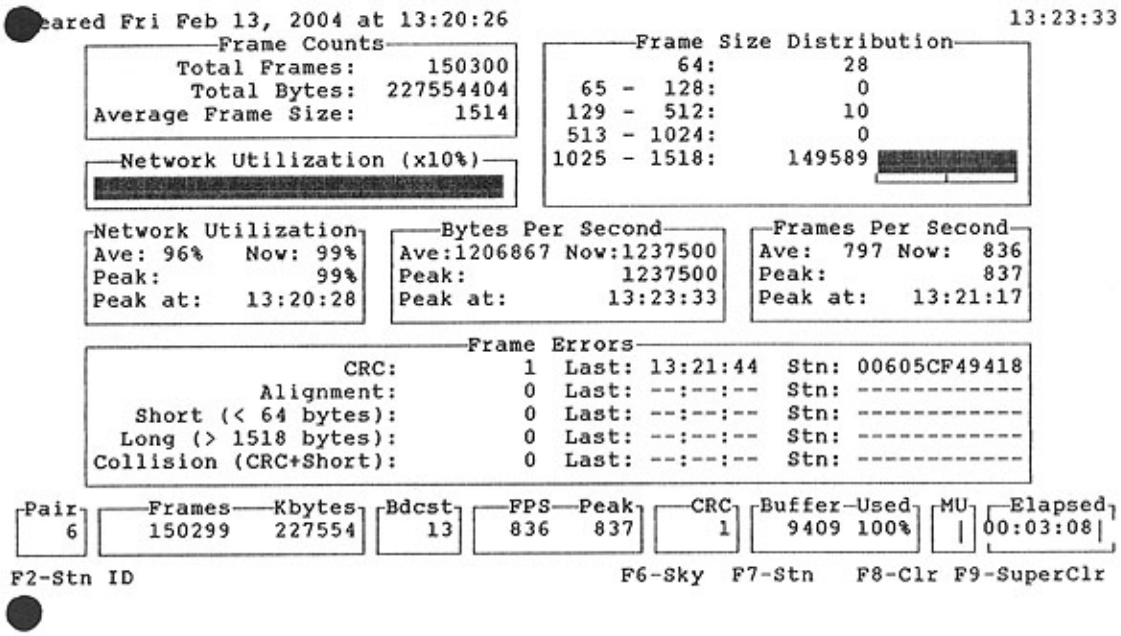
Pair	Frames	Kbytes	Bdcst	FPS	Peak	CRC	Buffer Used	MU	Elapsed
5	49764	18561	5	0	524	1	36593 100%		00:04:13

Press <F9> again to SuperClear, any other key to resume

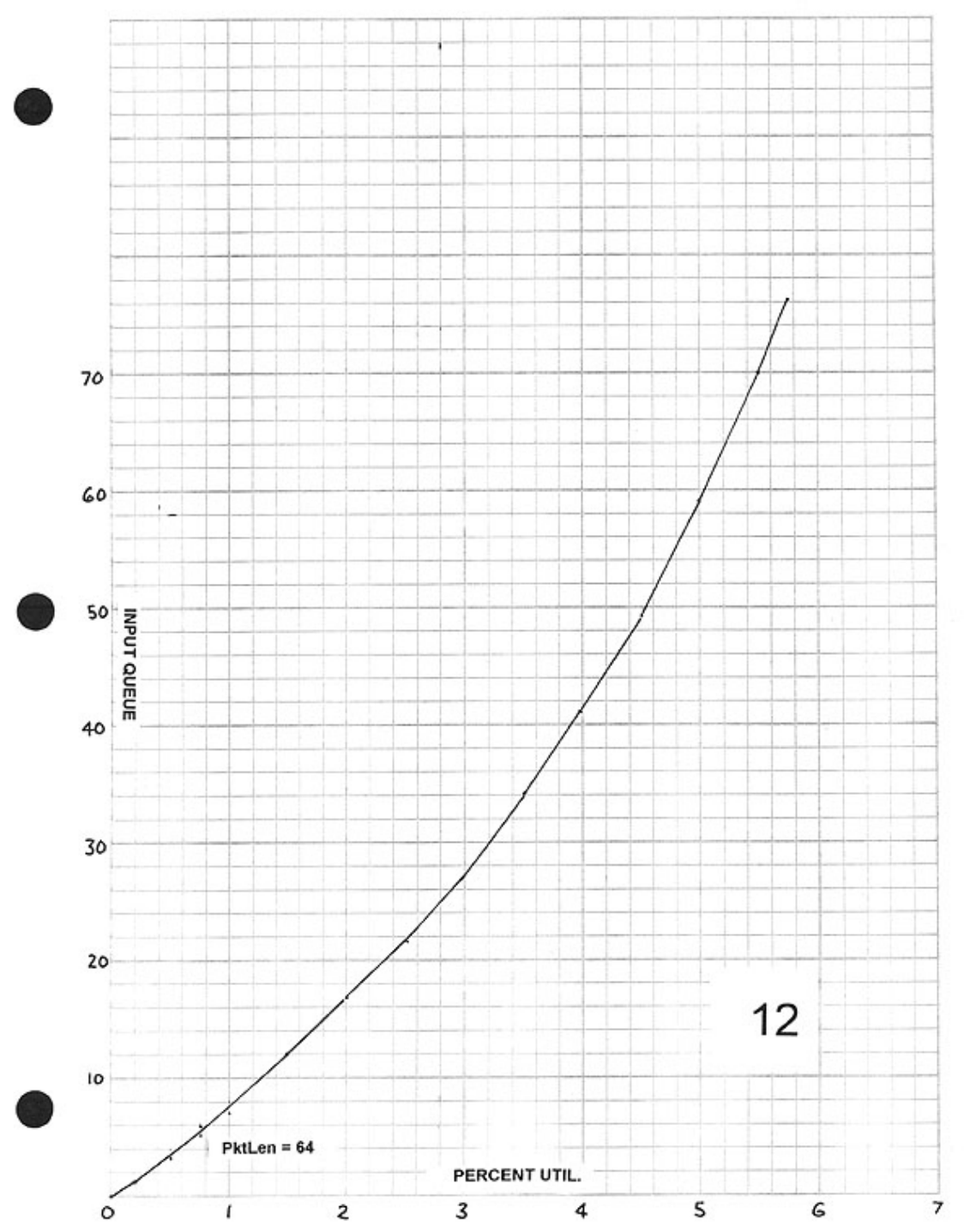
9

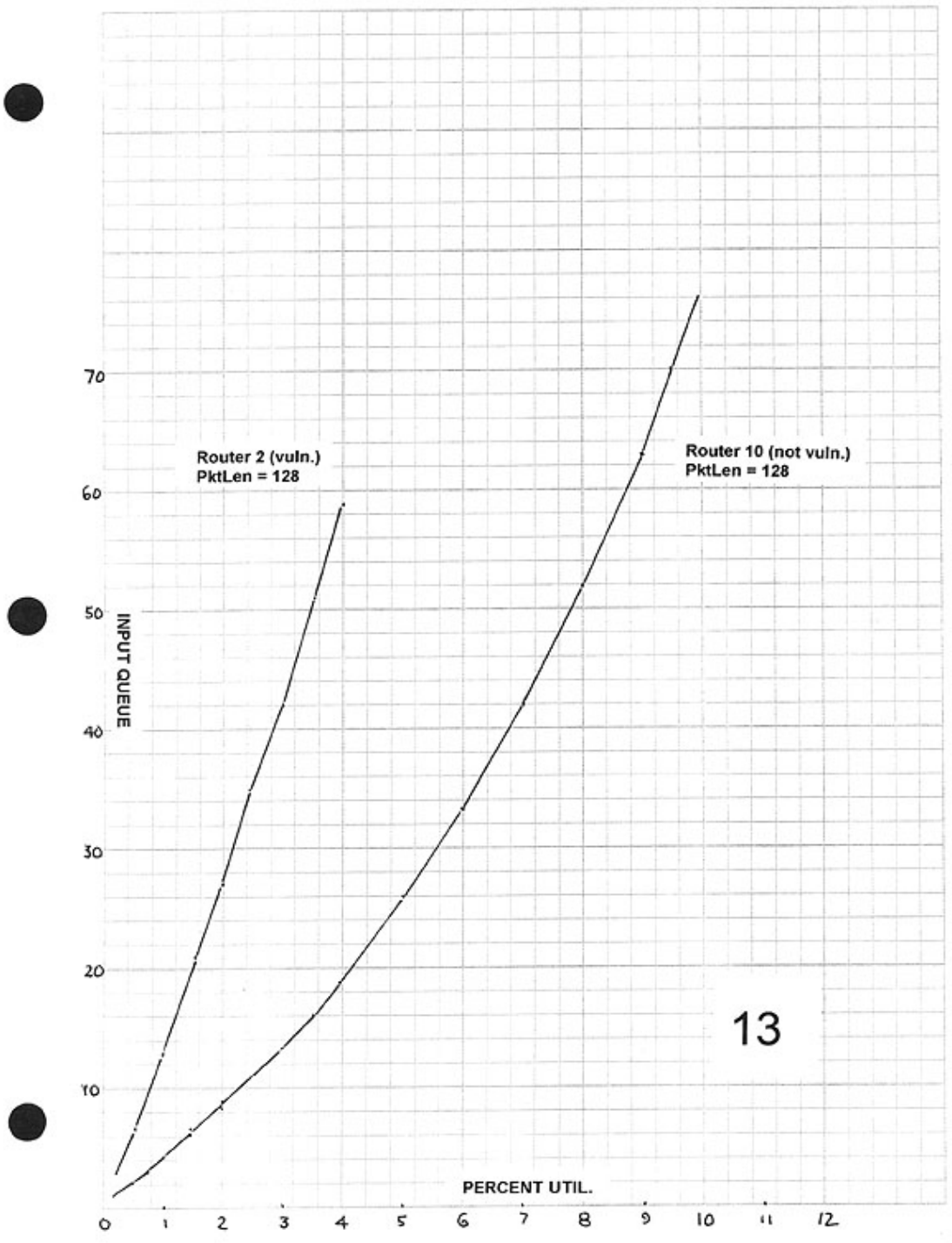
```
xy_2#sh int eth0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0060.5cf4.9418 (bia 0060.5cf4.9418)
  Description: net 201-5-5-0
  Internet address is 10.73.73.110/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 5/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 76/75, 43 drops
  5 minute input rate 296000 bits/sec, 110 packets/sec
  5 minute output rate 204000 bits/sec, 23 packets/sec
     323140 packets input, 61597648 bytes, 3 no buffer
     Received 1068 broadcasts, 0 runts, 0 giants, 44 throttles
     1 input errors, 1 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     26942 packets output, 24757520 bytes, 0 underruns
     0 output errors, 15 collisions, 94 interface resets
     0 babbles, 0 late collision, 73 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
foxy_2#
| Terminal # 5 vt100_9600_8n1   |PC-PLOT-IV  4.10e|  Now Online | ALT-H=HELP |
```

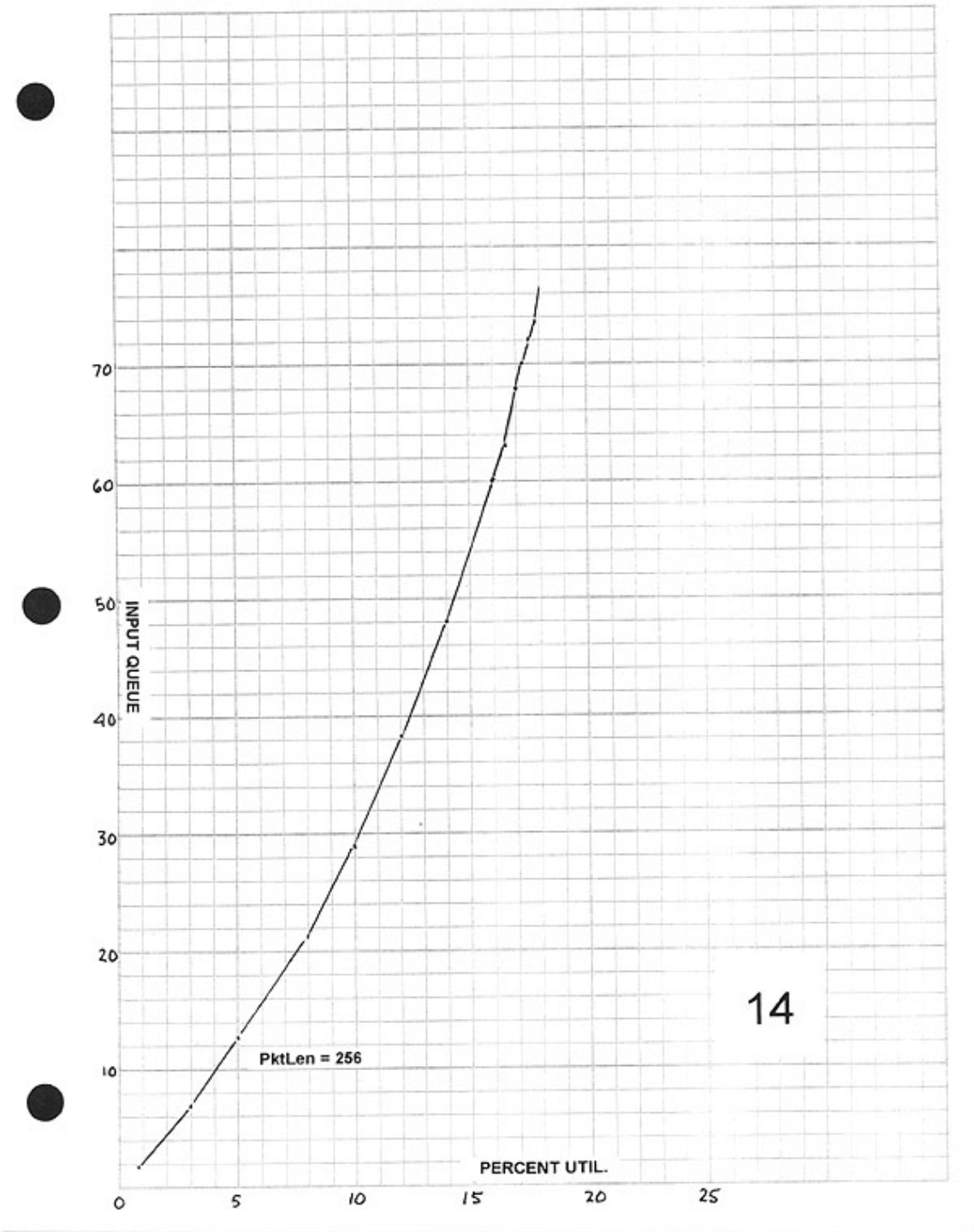
10

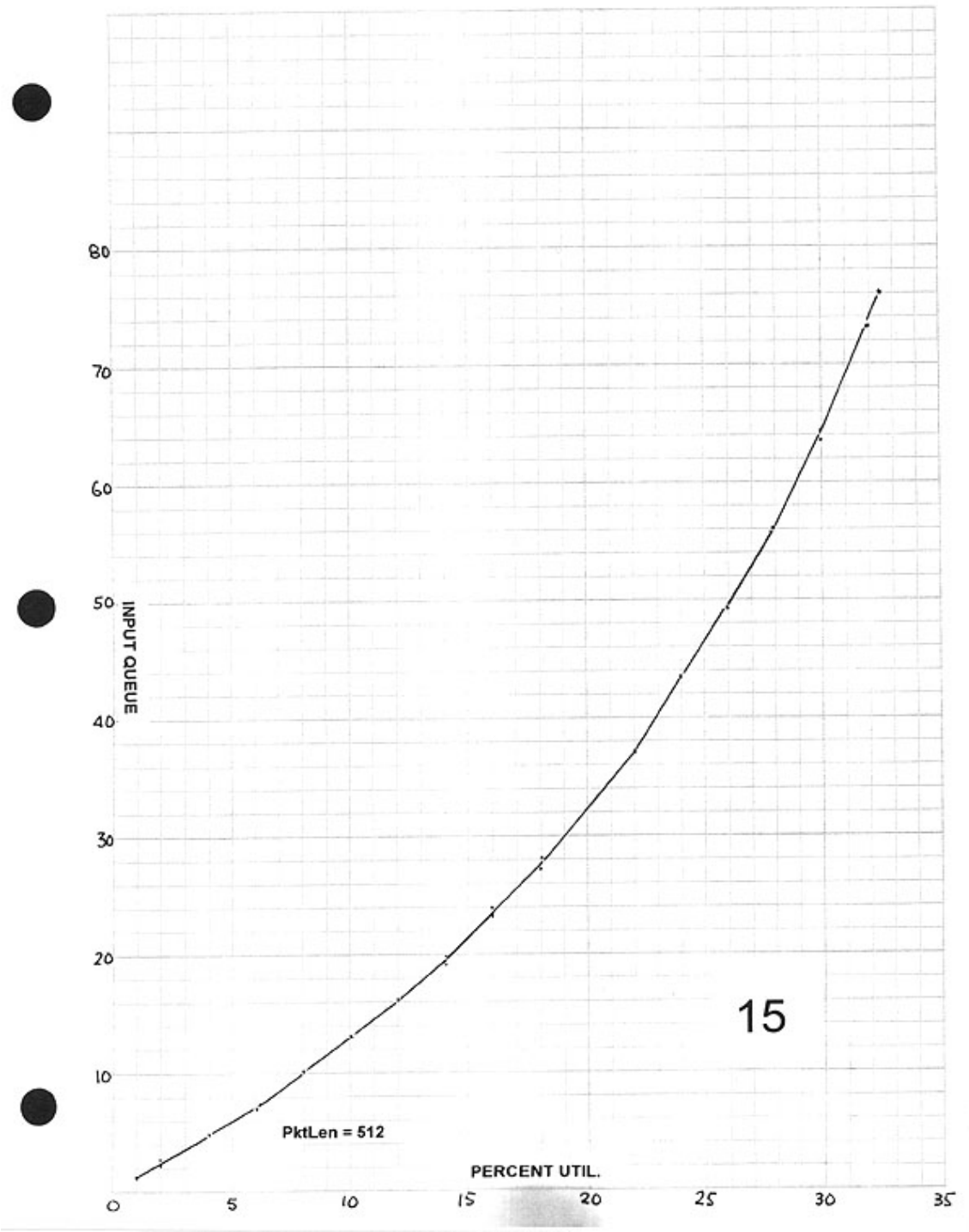


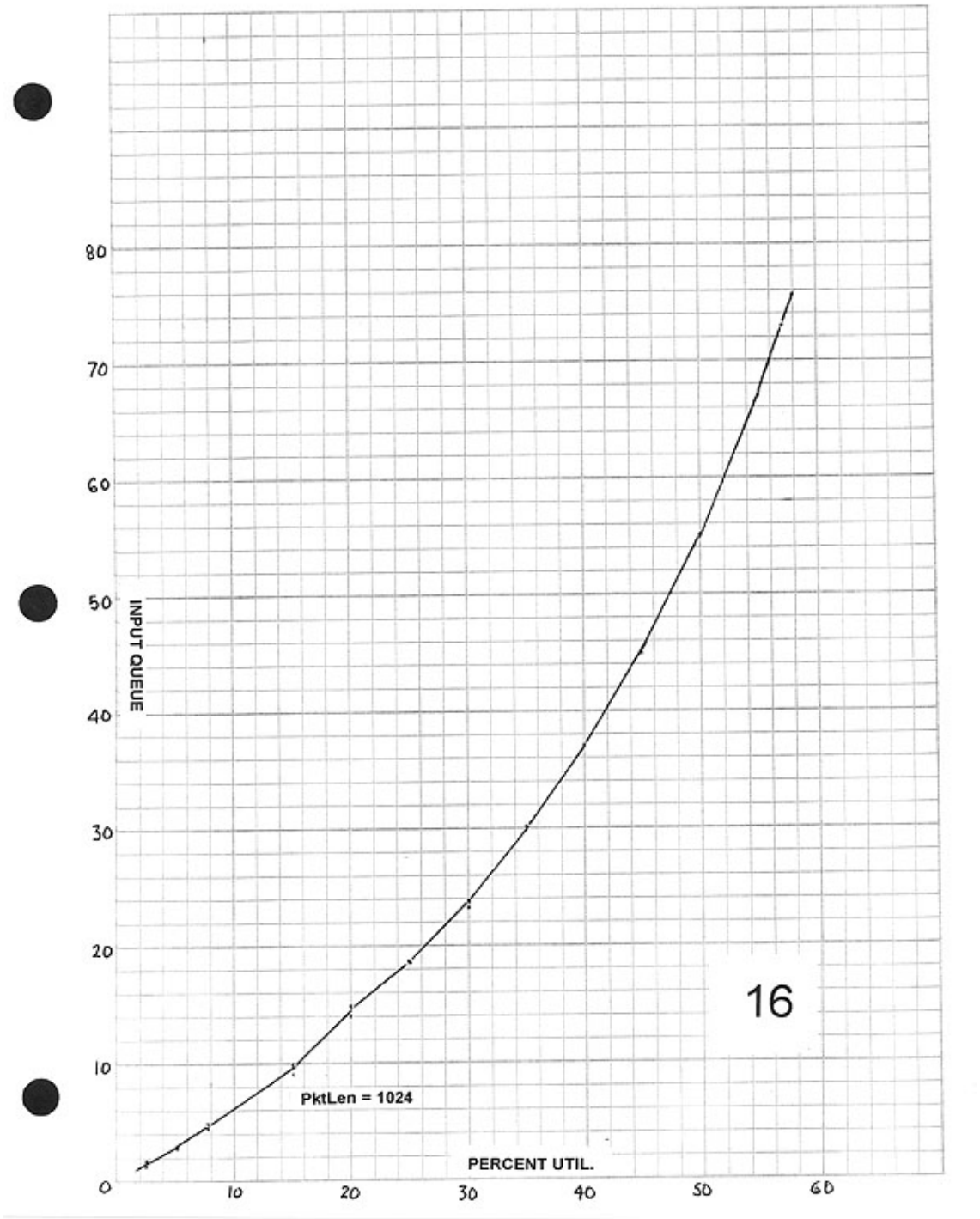
11

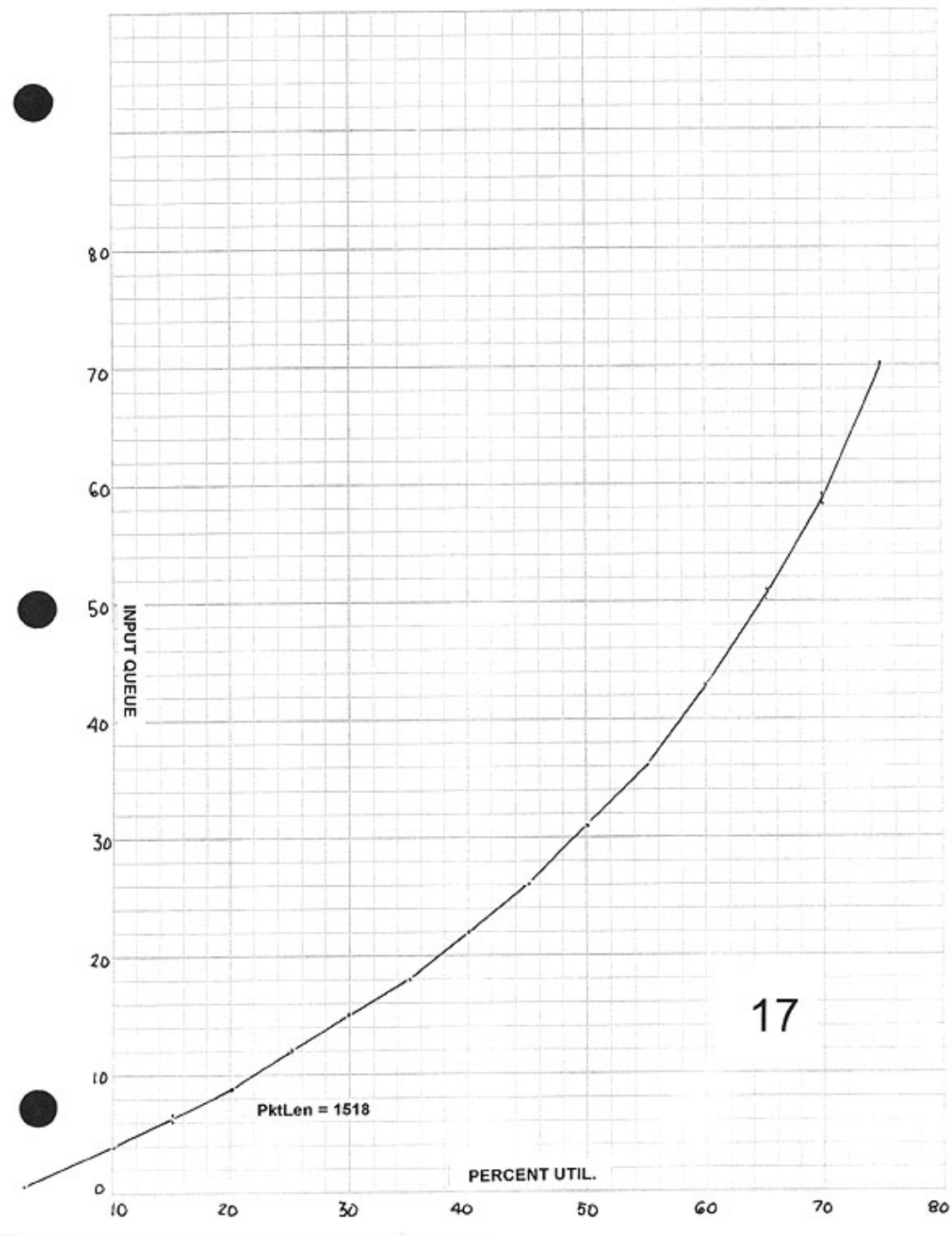


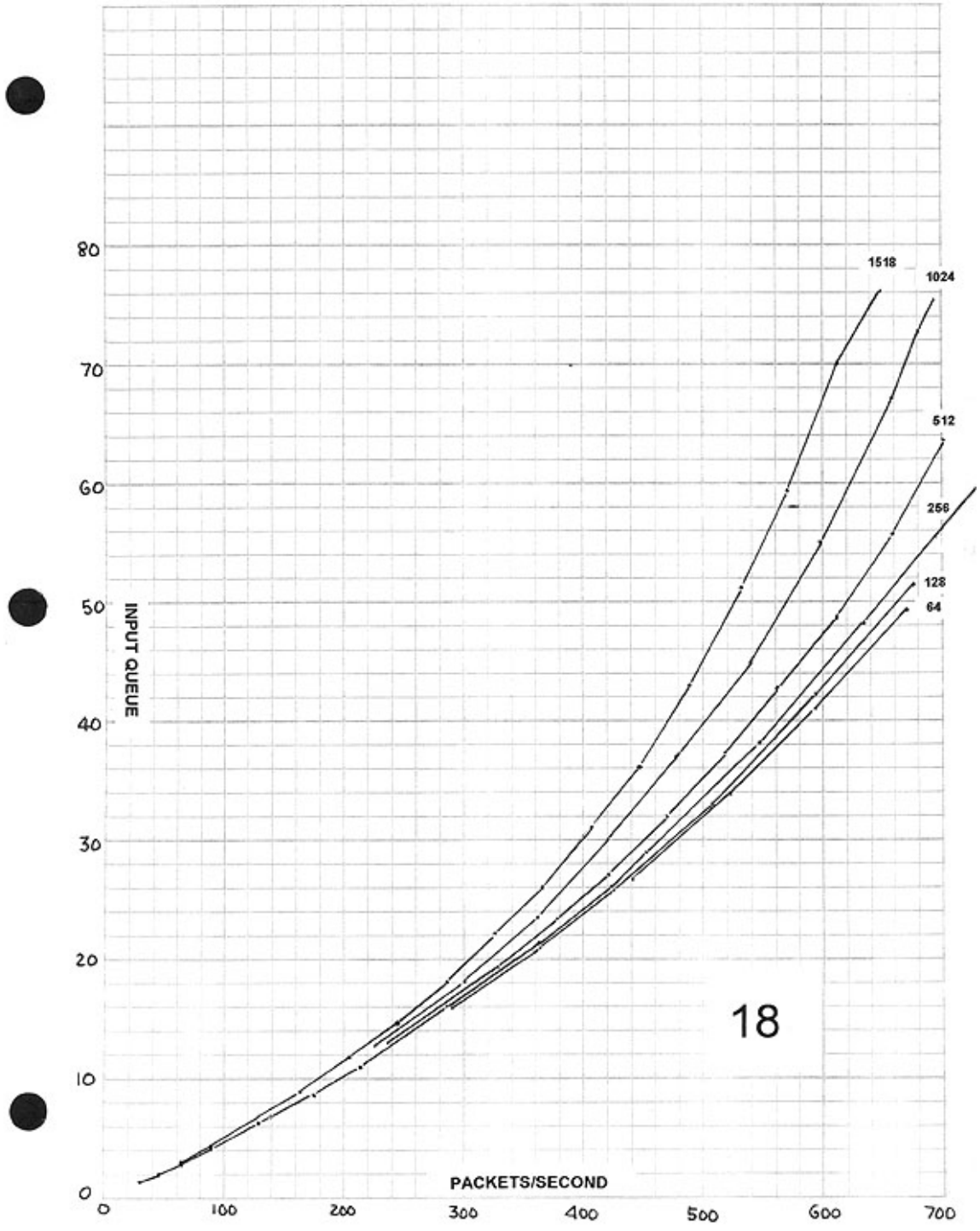












2.0 10211

```
cketView v1.21                          Total packets:  1376  Memory used:   2%
yright, Klos Technologies, Inc.         Receiver state: Enabled

442)  943.465 002E ARP: (0800) REQUEST from 10.73.73.95 for gw__________
443)  943.647 0208 IP: 23.27.53.181 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXXX
444)  943.657 0208 IP: 41.12.113.251 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXXX
445)  943.667 0208 IP: 171.207.13.113 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
446)  943.677 0208 IP: 118.90.88.50 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXXX
447)  943.687 0208 IP: 248.45.107.215 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
448)  943.708 0208 IP: 228.235.42.59 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
449)  943.717 0208 IP: 192.82.19.245 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
450)  943.732 0208 IP: 152.30.20.9 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXXX
451)  943.737 0208 IP: 214.180.232.227 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXX
452)  943.747 0208 IP: 147.89.127.235 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
453)  943.760 0208 IP: 90.228.185.197 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
454)  943.767 0208 IP: 6.170.167.48 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
455)  943.777 0208 IP: 158.240.56.177 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
456)  943.787 0208 IP: 64.73.4.84 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXXXX
457)  943.797 0208 IP: 30.42.6.6 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXXXX
458)  943.807 0208 IP: 43.154.103.170 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
459)  943.823 0208 IP: 242.5.105.171 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
460)  943.827 0208 IP: 37.177.116.204 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
461)  943.837 0208 IP: 90.106.210.146 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
462)  943.847 0208 IP: 184.210.101.188 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXX
463)  943.857 0208 IP: 184.218.81.190 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
464)  943.867 0208 IP: 94.234.123.197 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
465)  943.880 0208 IP: 0.184.153.106 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
466)  943.887 0208 IP: 100.143.21.216 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
467)  943.897 0208 IP: 246.50.197.200 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
468)  943.910 0208 IP: 225.128.150.71 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
469)  943.917 0208 IP: 189.255.23.14 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
470)  943.927 0208 IP: 248.108.167.115 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
471)  943.940 0208 IP: 60.116.234.213 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
472)  943.947 0208 IP: 77.57.109.98 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
473)  943.957 0208 IP: 200.89.68.141 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
474)  943.967 0208 IP: 9.111.162.159 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
475)  943.977 0208 IP: 223.92.181.246 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
476)  943.987 0208 IP: 114.239.215.26 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
477)  944.003 0208 IP: 9.86.214.243 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXXX
478)  944.007 0208 IP: 180.64.101.33 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXXX
479)  944.017 0208 IP: 119.233.252.187 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
480)  944.027 0208 IP: 29.196.230.191 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
481)  944.037 0208 IP: 156.158.192.81 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXXXX
482)  944.461 002E ARP: (0800) REQUEST from 10.73.73.95 for gw__________
483)  945.461 002E ARP: (0800) REQUEST from 10.73.73.95 for gw__________
484)  945.785 0070 IP: 10.73.73.110 -> _all_Bcst UDP: rip(520)->rip(520) RIP:
```

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
HELP	EDIT FILTERS	PACKET REPLAY	RESTART RECEIVE	TOGGLE RECEIVE		PRINT	GOTO PACKET	CONTIN-UOUS	MAIN MENU

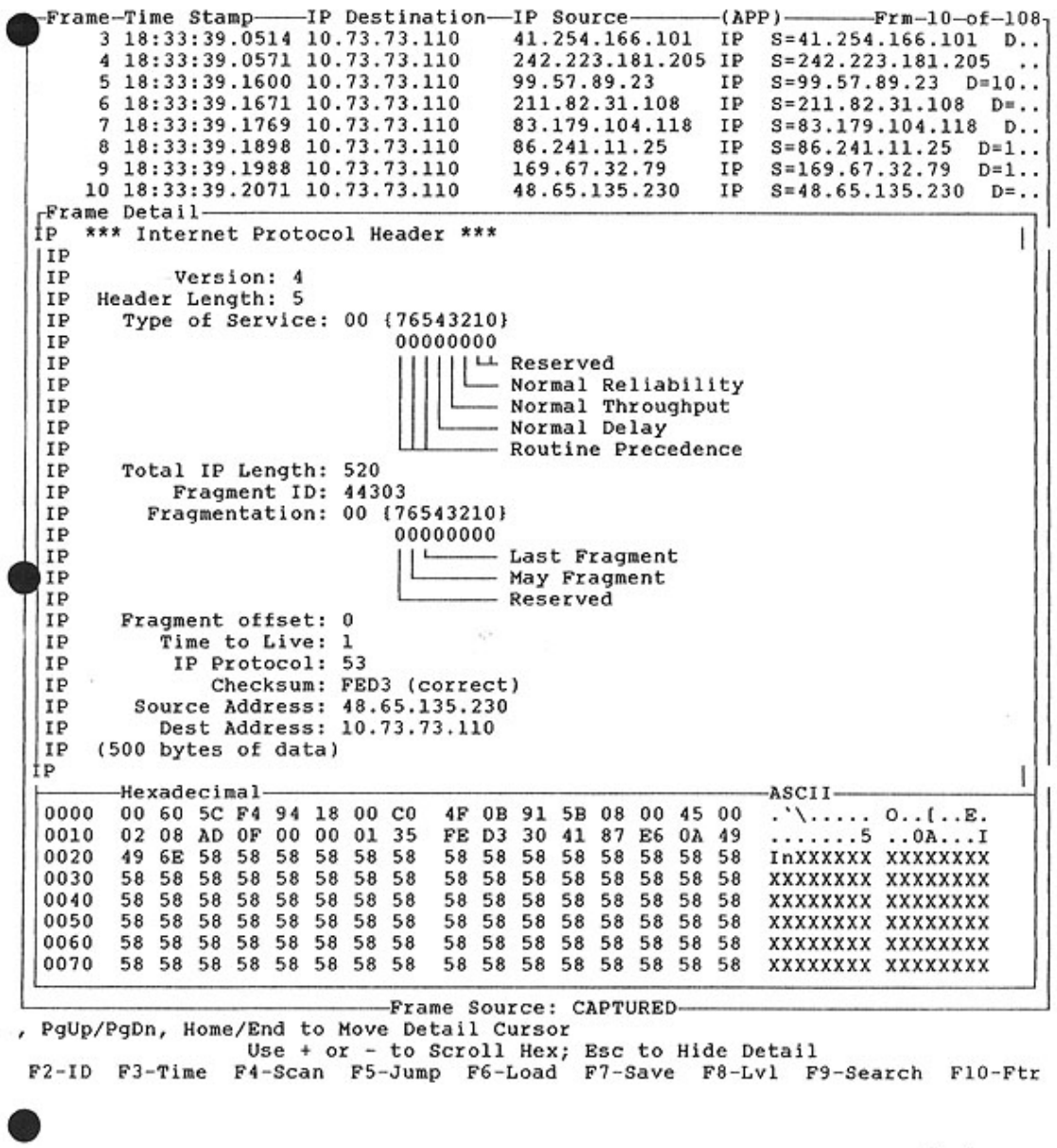
19

2604624

```
PacketView v1.21                            Total packets:  1337  Memory used:    2%
Copyright, Klos Technologies, Inc.          Receiver state: Enabled

DIX:
    Destination: Cisco-4A7E3F  Size: 0208  Number:     460
    Source:      00C04F0B915B  Type: 0800  Time:   943.827

IP:
IP version: 4  IP header length: 5 (32-bit words)
Type of service: 00
    Precedence = Routine
    Delay = Normal
    Throughput = Normal
    Reliability = Normal
Packet length: 0208  Packet ID: 59E1
More fragments: NO  Fragment offset: 0000
Time-to-live: 1  Protocol: SWIPE(53)  Header checksum: 6FAC (GOOD)
Source host id:      37.177.116.204
Destination host id: 10.73.73.110

SWIPE Data:
============================ Start of Text ==============================
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
============================= End of Text ===============================
```

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
HELP	EDIT FILTERS	PACKET REPLAY	RESTART RECEIVE	TOGGLE RECEIVE		PRINT	GOTO PACKET	CONTIN-UOUS	MAIN MENU

20



21

```
PacketView v1.21                          Total packets: 64000* Memory used: 1
Copyright, Klos Technologies, Inc.        Receiver state: Buffers full

46173) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46174) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46175) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46176) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46177) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46178) 65033.965 0208 IP: 98.32.41.95 -> 10.73.73.110 SWIPE: 'XXXXXXXXXXXXXX
46179) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46180) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46181) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46182) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46183) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46184) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46185) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46186) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46187) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46188) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46189) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46190) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46191) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46192) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46193) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46194) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46195) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46196) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46197) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46198) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46199) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46200) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46201) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46202) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46203) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46204) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46205) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46206) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46207) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46208) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46209) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46210) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46211) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46212) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46213) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46214) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46215) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46216) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46217) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46218) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46219) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
```

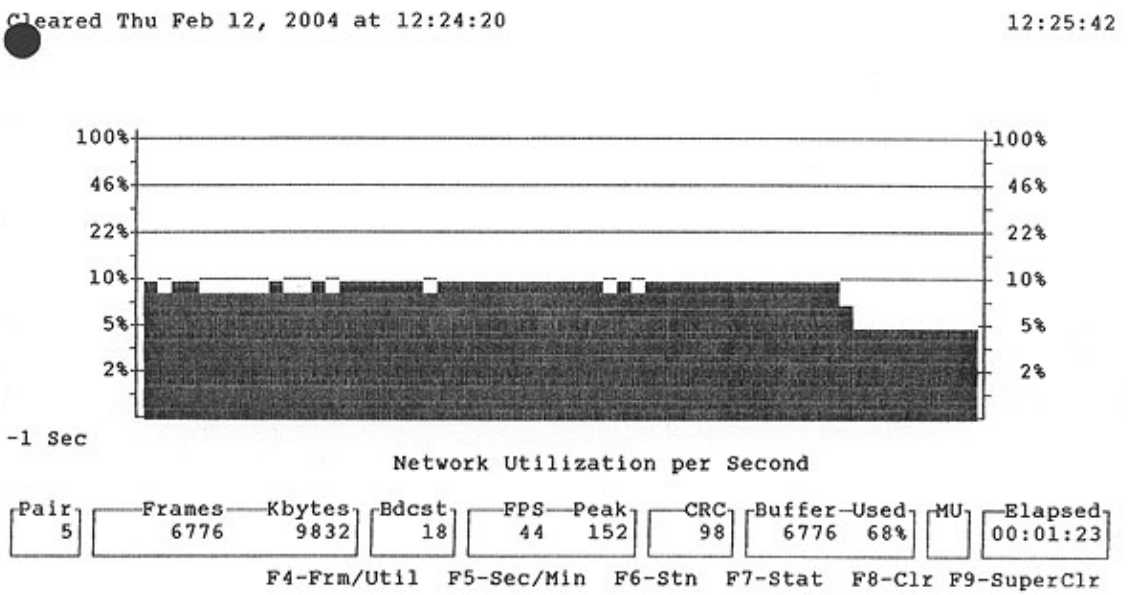
22

```
PacketView v1.21                                Total packets: 64000* Memory used: 11
Copyright, Klos Technologies, Inc.              Receiver state: Buffers full

46219) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46220) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46221) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46222) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46223) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46224) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46225) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46226) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46227) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46228) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46229) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46230) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46231) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46232) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46233) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46234) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46235) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46236) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46237) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46238) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46239) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46240) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46241) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46242) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46243) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46244) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46245) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46246) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46247) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46248) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46249) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46250) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46251) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46252) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46253) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46254) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46255) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46256) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46257) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46258) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46259) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46260) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46261) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46262) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
46263) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46264) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
46265) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
```

23

```
PacketView v1.21                          Total packets: 64000* Memory used: 118
Copyright, Klos Technologies, Inc.        Receiver state: Buffers full

48407) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48408) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48409) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48410) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48411) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48412) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48413) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48414) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48415) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48416) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48417) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48418) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48419) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48420) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48421) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48422) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48423) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48424) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48425) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48426) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48427) 65033.965 00E8 IP: 10.73.73.52 -> 192.168.30.51 ICMP: Echo Reply
48428) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48429) 65033.965 002E ARP: (0800) REQUEST from 10.73.73.52 for 10.73.73.110
48430) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48431) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48432) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48433) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48434) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48435) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48436) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48437) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48438) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48439) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48440) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48441) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48442) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48443) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48444) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48445) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48446) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48447) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48448) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48449) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48450) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48451) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48452) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
48453) 65033.965 00E8 IP: 192.168.30.51 -> 10.73.73.52 ICMP: Echo
```

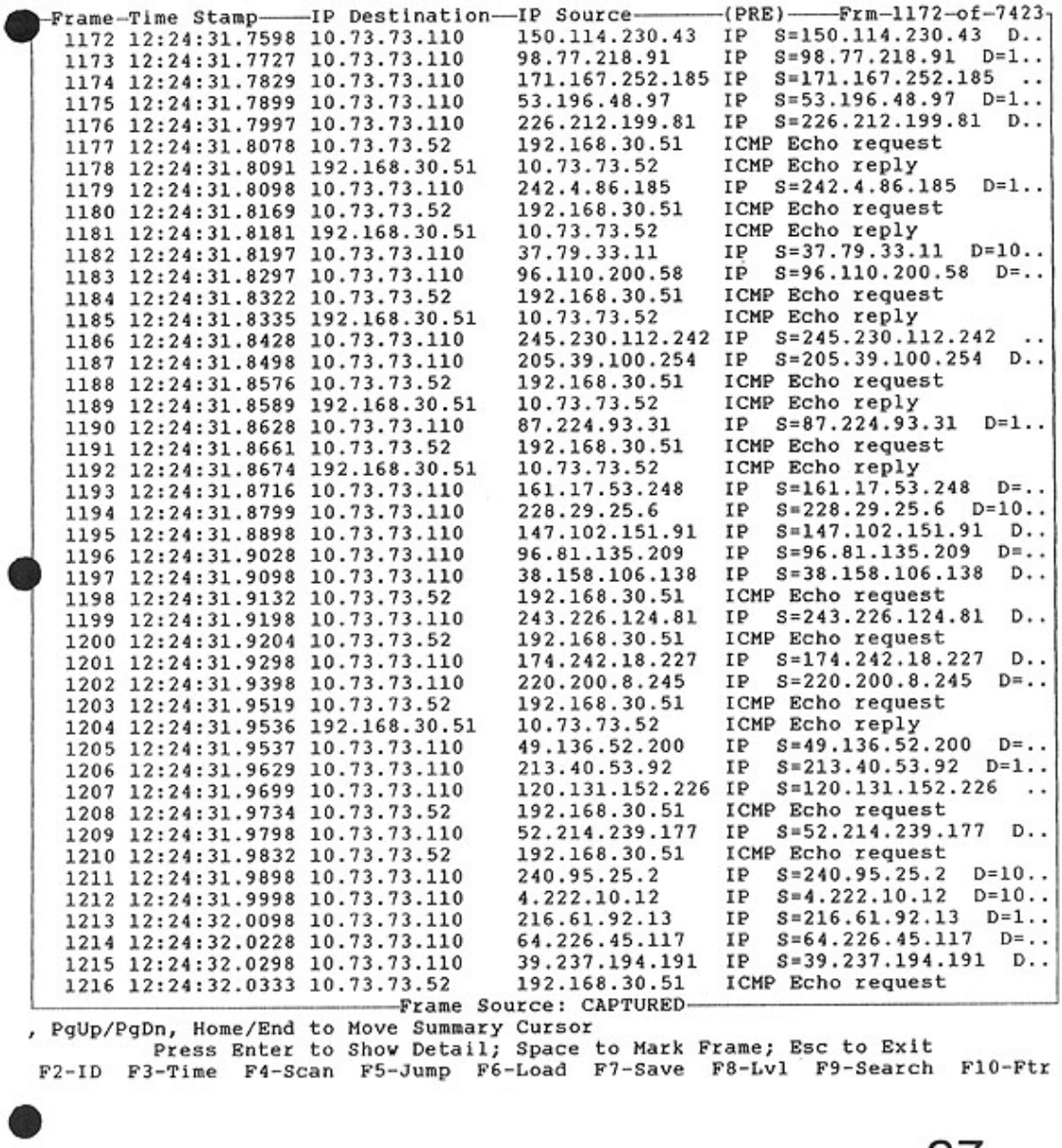
24



25

```
Frame-Time Stamp-----IP Destination--IP Source--------(MAC)-----Frm-2704-of-7423
  2660 12:24:49.2451 192.168.30.51   10.73.73.52      MAC  1486 Bytes  OK  CR..
  2661 12:24:49.2526 10.73.73.52     192.168.30.51    MAC  1506 Bytes  OK  CR..
  2662 12:24:49.2539 192.168.30.51   10.73.73.52      MAC  1506 Bytes  OK  CR..
  2663 12:24:49.2569 10.73.73.52     192.168.30.51    MAC   539 Bytes  CRC Er..
  2664 12:24:49.3464 10.73.73.52     192.168.30.51    MAC  1466 Bytes  OK  CR..
  2665 12:24:49.3477 192.168.30.51   10.73.73.52      MAC  1466 Bytes  OK  CR..
  2666 12:24:49.3552 10.73.73.52     192.168.30.51    MAC  1476 Bytes  OK  CR..
  2667 12:24:49.3565 192.168.30.51   10.73.73.52      MAC  1476 Bytes  OK  CR..
  2668 12:24:49.3640 10.73.73.52     192.168.30.51    MAC  1506 Bytes  OK  CR..
  2669 12:24:49.3654 192.168.30.51   10.73.73.52      MAC  1506 Bytes  OK  CR..
  2670 12:24:49.3728 10.73.73.52     192.168.30.51    MAC  1486 Bytes  OK  CR..
  2671 12:24:49.3741 192.168.30.51   10.73.73.52      MAC  1486 Bytes  OK  CR..
  2672 12:24:49.3773 10.73.73.52     192.168.30.51    MAC   633 Bytes  CRC Er..
  2673 12:24:49.4663 10.73.73.52     192.168.30.51    MAC  1466 Bytes  OK  CR..
  2674 12:24:49.4676 192.168.30.51   10.73.73.52      MAC  1466 Bytes  OK  CR..
  2675 12:24:49.4750 10.73.73.52     192.168.30.51    MAC  1496 Bytes  OK  CR..
  2676 12:24:49.4763 192.168.30.51   10.73.73.52      MAC  1496 Bytes  OK  CR..
  2677 12:24:49.4838 10.73.73.52     192.168.30.51    MAC  1486 Bytes  OK  CR..
  2678 12:24:49.4851 192.168.30.51   10.73.73.52      MAC  1486 Bytes  OK  CR..
  2679 12:24:49.4927 10.73.73.52     192.168.30.51    MAC  1506 Bytes  OK  CR..
  2680 12:24:49.4939 192.168.30.51   10.73.73.52      MAC  1506 Bytes  OK  CR..
  2681 12:24:49.4971 10.73.73.52     192.168.30.51    MAC   618 Bytes  CRC Er..
  2682 12:24:49.5865 10.73.73.52     192.168.30.51    MAC  1466 Bytes  OK  CR..
  2683 12:24:49.5878 192.168.30.51   10.73.73.52      MAC  1466 Bytes  OK  CR..
  2684 12:24:49.5954 10.73.73.52     192.168.30.51    MAC  1476 Bytes  OK  CR..
  2685 12:24:49.5966 192.168.30.51   10.73.73.52      MAC  1476 Bytes  OK  CR..
  2686 12:24:49.6042 10.73.73.52     192.168.30.51    MAC  1506 Bytes  OK  CR..
  2687 12:24:49.6055 192.168.30.51   10.73.73.52      MAC  1506 Bytes  OK  CR..
  2688 12:24:49.6129 10.73.73.52     192.168.30.51    MAC  1486 Bytes  OK  CR..
  2689 12:24:49.6142 192.168.30.51   10.73.73.52      MAC  1486 Bytes  OK  CR..
  2690 12:24:49.6173 10.73.73.52     192.168.30.51    MAC   555 Bytes  CRC Er..
  2691 12:24:49.7064 10.73.73.52     192.168.30.51    MAC  1466 Bytes  OK  CR..
  2692 12:24:49.7077 192.168.30.51   10.73.73.52      MAC  1466 Bytes  OK  CR..
  2693 12:24:49.7143 10.73.73.52     192.168.30.51    MAC  1496 Bytes  OK  CR..
  2694 12:24:49.7156 192.168.30.51   10.73.73.52      MAC  1496 Bytes  OK  CR..
  2695 12:24:49.7225 10.73.73.52     192.168.30.51    MAC  1486 Bytes  OK  CR..
  2696 12:24:49.7238 192.168.30.51   10.73.73.52      MAC  1486 Bytes  OK  CR..
  2697 12:24:49.7314 10.73.73.52     192.168.30.51    MAC  1506 Bytes  OK  CR..
  2698 12:24:49.7327 192.168.30.51   10.73.73.52      MAC  1506 Bytes  OK  CR..
  2699 12:24:49.7402 10.73.73.52     192.168.30.51    MAC  1476 Bytes  OK  CR..
  2700 12:24:49.7414 192.168.30.51   10.73.73.52      MAC  1476 Bytes  OK  CR..
  2701 12:24:49.8263 10.73.73.52     192.168.30.51    MAC  1466 Bytes  OK  CR..
  2702 12:24:49.8276 192.168.30.51   10.73.73.52      MAC  1466 Bytes  OK  CR..
  2703 12:24:49.8341 10.73.73.52     192.168.30.51    MAC  1476 Bytes  OK  CR..
  2704 12:24:49.8354 192.168.30.51   10.73.73.52      MAC  1476 Bytes  OK  CR..
-------------------------Frame Source: CAPTURED-------------------------------
, PgUp/PgDn, Home/End to Move Summary Cursor
          Press Enter to Show Detail; Space to Mark Frame; Esc to Exit
 F2-ID  F3-Time  F4-Scan  F5-Jump  F6-Load  F7-Save  F8-Lvl  F9-Search  F10-Ftr
```

26

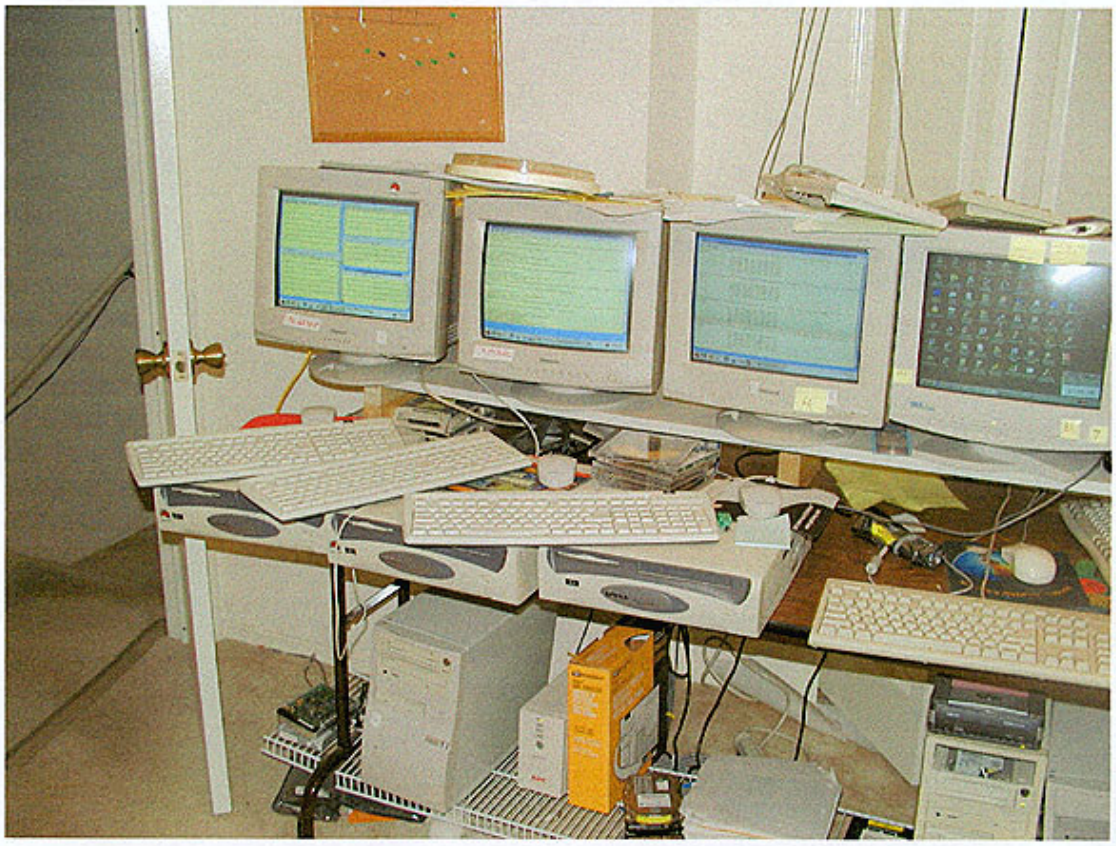


```
Frame-Time Stamp-----IP Destination--IP Source--------(PRE)-----Frm-1172-of-7423
  1172 12:24:31.7598 10.73.73.110     150.114.230.43  IP  S=150.114.230.43  D..
  1173 12:24:31.7727 10.73.73.110     98.77.218.91    IP  S=98.77.218.91  D=1..
  1174 12:24:31.7829 10.73.73.110     171.167.252.185 IP  S=171.167.252.185  ..
  1175 12:24:31.7899 10.73.73.110     53.196.48.97    IP  S=53.196.48.97  D=1..
  1176 12:24:31.7997 10.73.73.110     226.212.199.81  IP  S=226.212.199.81  D..
  1177 12:24:31.8078 10.73.73.52      192.168.30.51   ICMP Echo request
  1178 12:24:31.8091 192.168.30.51    10.73.73.52     ICMP Echo reply
  1179 12:24:31.8098 10.73.73.110     242.4.86.185    IP  S=242.4.86.185  D=1..
  1180 12:24:31.8169 10.73.73.52      192.168.30.51   ICMP Echo request
  1181 12:24:31.8181 192.168.30.51    10.73.73.52     ICMP Echo reply
  1182 12:24:31.8197 10.73.73.110     37.79.33.11     IP  S=37.79.33.11  D=10..
  1183 12:24:31.8297 10.73.73.110     96.110.200.58   IP  S=96.110.200.58  D=..
  1184 12:24:31.8322 10.73.73.52      192.168.30.51   ICMP Echo request
  1185 12:24:31.8335 192.168.30.51    10.73.73.52     ICMP Echo reply
  1186 12:24:31.8428 10.73.73.110     245.230.112.242 IP  S=245.230.112.242  ..
  1187 12:24:31.8498 10.73.73.110     205.39.100.254  IP  S=205.39.100.254  D..
  1188 12:24:31.8576 10.73.73.52      192.168.30.51   ICMP Echo request
  1189 12:24:31.8589 192.168.30.51    10.73.73.52     ICMP Echo reply
  1190 12:24:31.8628 10.73.73.110     87.224.93.31    IP  S=87.224.93.31  D=1..
  1191 12:24:31.8661 10.73.73.52      192.168.30.51   ICMP Echo request
  1192 12:24:31.8674 192.168.30.51    10.73.73.52     ICMP Echo reply
  1193 12:24:31.8716 10.73.73.110     161.17.53.248   IP  S=161.17.53.248  D=..
  1194 12:24:31.8799 10.73.73.110     228.29.25.6     IP  S=228.29.25.6  D=10..
  1195 12:24:31.8898 10.73.73.110     147.102.151.91  IP  S=147.102.151.91  D..
  1196 12:24:31.9028 10.73.73.110     96.81.135.209   IP  S=96.81.135.209  D=..
  1197 12:24:31.9098 10.73.73.110     38.158.106.138  IP  S=38.158.106.138  D..
  1198 12:24:31.9132 10.73.73.52      192.168.30.51   ICMP Echo request
  1199 12:24:31.9198 10.73.73.110     243.226.124.81  IP  S=243.226.124.81  D..
  1200 12:24:31.9204 10.73.73.52      192.168.30.51   ICMP Echo request
  1201 12:24:31.9298 10.73.73.110     174.242.18.227  IP  S=174.242.18.227  D..
  1202 12:24:31.9398 10.73.73.110     220.200.8.245   IP  S=220.200.8.245  D=..
  1203 12:24:31.9519 10.73.73.52      192.168.30.51   ICMP Echo request
  1204 12:24:31.9536 192.168.30.51    10.73.73.52     ICMP Echo reply
  1205 12:24:31.9537 10.73.73.110     49.136.52.200   IP  S=49.136.52.200  D=..
  1206 12:24:31.9629 10.73.73.110     213.40.53.92    IP  S=213.40.53.92  D=1..
  1207 12:24:31.9699 10.73.73.110     120.131.152.226 IP  S=120.131.152.226  ..
  1208 12:24:31.9734 10.73.73.52      192.168.30.51   ICMP Echo request
  1209 12:24:31.9798 10.73.73.110     52.214.239.177  IP  S=52.214.239.177  D..
  1210 12:24:31.9832 10.73.73.52      192.168.30.51   ICMP Echo request
  1211 12:24:31.9898 10.73.73.110     240.95.25.2     IP  S=240.95.25.2  D=10..
  1212 12:24:31.9998 10.73.73.110     4.222.10.12     IP  S=4.222.10.12  D=10..
  1213 12:24:32.0098 10.73.73.110     216.61.92.13    IP  S=216.61.92.13  D=1..
  1214 12:24:32.0228 10.73.73.110     64.226.45.117   IP  S=64.226.45.117  D=..
  1215 12:24:32.0298 10.73.73.110     39.237.194.191  IP  S=39.237.194.191  D..
  1216 12:24:32.0333 10.73.73.52      192.168.30.51   ICMP Echo request
                                  Frame Source: CAPTURED
, PgUp/PgDn, Home/End to Move Summary Cursor
          Press Enter to Show Detail; Space to Mark Frame; Esc to Exit
 F2-ID  F3-Time  F4-Scan  F5-Jump  F6-Load  F7-Save  F8-Lvl  F9-Search  F10-Ftr
```

27

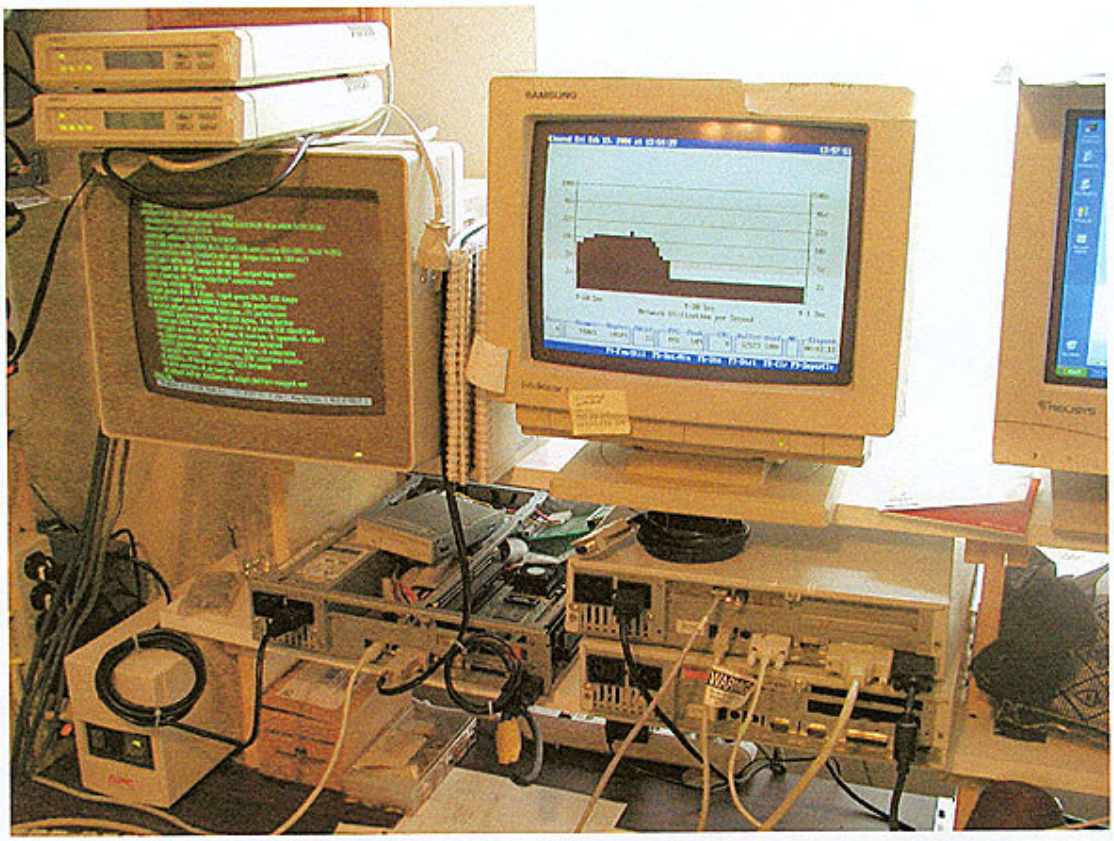
```
foxy_2#sh int ethernet0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0060.5cf4.9418 (bia 0060.5cf4.9418)
  Description: net 201-5-5-0
  Internet address is 10.73.73.110/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 2/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:18, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:27:15
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 70/75, 0 drops
  5 minute input rate 88000 bits/sec, 45 packets/sec
  5 minute output rate 88000 bits/sec, 45 packets/sec
     66748 packets input, 16433196 bytes, 0 no buffer
     Received 3 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 input packets with dribble condition detected
     66921 packets output, 16418915 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
foxy_2#
| Terminal # 5 vt100_9600_8n1   |PC-PLOT-IV  4.10e|  Now Online | ALT-H=HELP
```

28



29

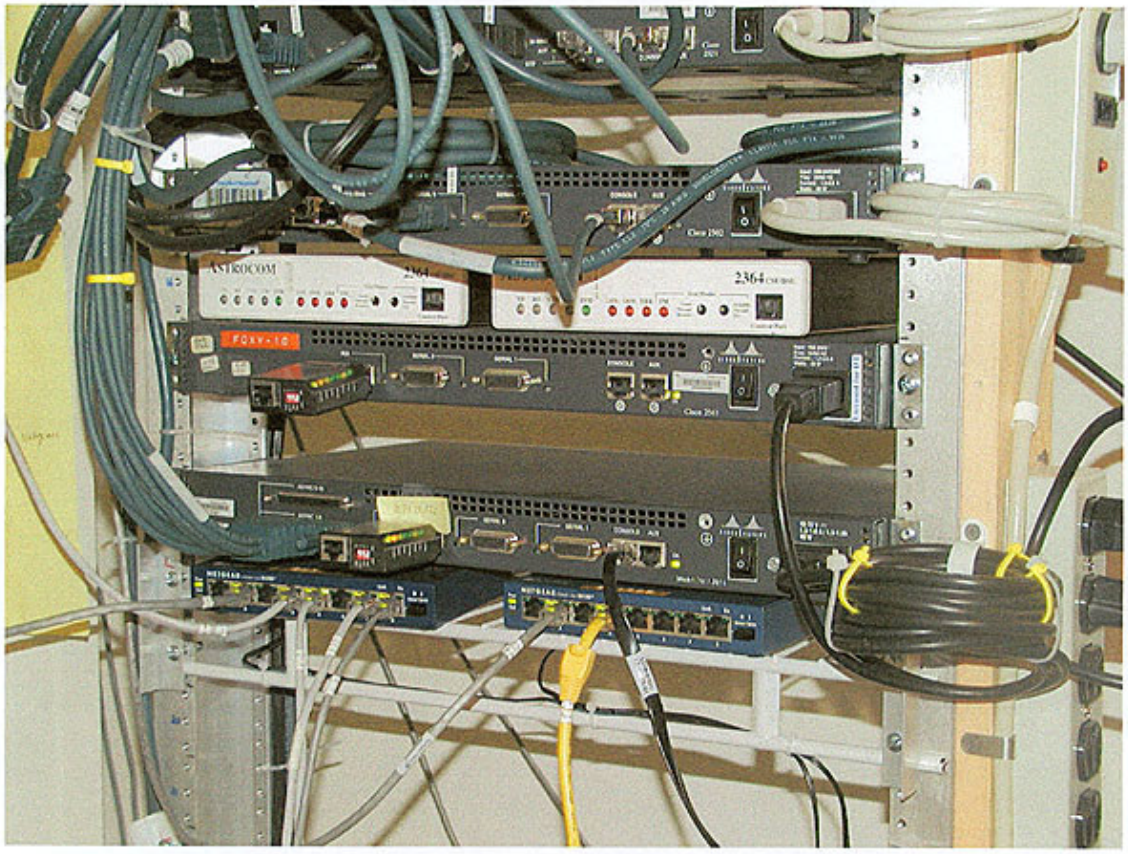
30



31

32

33



34

Submitted for GIAC Security Essentials Certification (GSEC) Practical version 1.4b amended August 2002 Option 1.

The editorial “we” is used herein for all personal pronoun references to the author.
