2008

# Performance Analysis of a Light Weight Packet Scanner

Paras Gandhi
*Virginia Commonwealth University*

**Performance Analysis of a Light Weight Packet Scanner**

A thesis submitted in partial fulfillment of the requirements for the degree Master of Science at Virginia Commonwealth University

By
Paras Gandhi

Director: Dr. Ju Wang
Assistant Professor, Department of Computer Science

Virginia Commonwealth University
Richmond, Virginia

December 2008

# Acknowledgements

First and foremost, I would like to thank Dr. Wang for being such a great advisor and a great friend. I would also like to thank Dr. Choi for always answering any questions I had and helping me understand the theoretical aspects of networking. I would like to give my special thanks to Dr. Sarkozi for providing me with all the resources I asked for and always giving me the right advice as a professor and most importantly for the precious fatherly advice which I needed most.

In addition, I would like to thank the faculty of the Computer Science Department for facilitating the excellent academic experience that I have had here at Virginia Commonwealth University.

Finally, I would like to thank Vrushali, Ashwin and Ravi and all my friends and parents for the great support that they have provided me through the course of my studies.

# Table of Contents

# List of Figures

**Abstract**

**Performance Analysis of a Light Weight Packet Scanner**

By Paras Gandhi

A thesis submitted in partial fulfillment of the requirements for the degree Master of Science at Virginia Commonwealth University

Virginia Commonwealth University, 2008

Director: Dr. Ju Wang, Assistant Professor, Department of Computer Science

The growth of networks around the world has also given rise to threats like viruses and Trojans. This rise in threats has resulted in counter measures for these threats. These counter measures are in the form of applications called firewalls or IDS. The incorporation of these applications in the network results in some delay in communications. The aim of the experiment in this thesis is to measure the delay introduced by such a firewall in the best case and compare it with the communication done on a network without such an application. These experiments are done using a special miniature computer called the net4801 with an embedded operating system and the packet scanning application (firewall or IDS) executing on it.

# 1. Introduction

In today's world of technology and internet, network security is a very critical issue. For Instance, there are hundreds of new viruses, Trojans and other types of attacks. Viruses are small programs that are usually intentional and harmful but sometimes they are not intentional. All viruses however small they may be, have the capability of affecting day-to-day life. There are other types of attacks made by someone. These computer attacks only corrupt a system's security in very specific ways. For example, an attacker might be able to read some files from a remote computer but not change them. Some types of attacks might enable the attacker to shutdown the remote computer but not access the files on it. All these types of attacks result in violation of availability, confidentiality, integrity, and control. These violations are described below. (Rebecca Bace 2001)

- *Confidentiality*: Information is available to a user who is not authorized to access it.

- *Integrity*: An attack which allows the attacker to change the state of the remote computer system or affect the data passing through the system.
- *Availability*: An attack causes the remote system or the resources on that system to be unavailable to the authorized user whenever he/she needs it.
- *Control*: An attack which enables the attacker to grant himself/herself, unauthorized control over the remote system. This privilege enables a subsequent confidentiality, integrity, or availability violation. (Rebecca Bace 2001)

To combat these computer violations, there are many different tools and programs available in the market today. We call these tools or programs Intrusion Detection Systems (IDS) or Firewalls. Some of them are free and others are not. When a computer is connected to a network, the Network Interface Card (NIC) receives and sends packets of information from and to other devices on the network. Each of these packets could potentially contain viruses or malicious scripts, therefore these packets need to be scanned for such entities. When a packet is scanned for

viruses, it is held up for a while before it is used in any way. This introduces a delay in the packet use. (Lau 2000) For example when we upload a file to a web server, before the server stores the file on the disk, its firewall scans the file packets for viruses. If the file contains a virus, then it is blocked thus saving the server from being infected. Just to see how much delay, on an average is introduced by such a packet scanning software, we conducted an experiment. Multiple files are uploaded to a web server using multiple threads but without any scanner. The time it took to upload each of the files was recorded. Next, the same size files were uploaded, but with a packet scanner in the network.

For our experiment, we use a device built by Soekris Engineering (Soekris 2004) as a special computer in the network, which scans all the packets sent out on the network and blocks ones with a virus on them.

**Figure 1 Soekris net4801**

The actual packet scanning software on net4801 was made available by Dr. Ju Wang. The operating system that was installed on this computer was a stripped down version of FreeBSD called NanoBSD (BSD 1995). This operating system was installed on the net4801 (Soekris 2004) using a FreeBSD Virtual Machine installed on a regular PC. An image of NanoBSD was made first on the Virtual Machine (VMware 2008) and then this image was transferred on a 512 MB CF card. The kernel of this image was compiled with only the minimum required modules and "user-land" utilities and programs. This was all done using a Linux shell script (Gerzo 2006). Once the net4801 computer was up and running, we prepare another regular PC with Linux running on it. The Linux computer had a web server running on it which in turn had a php script which allowed clients to upload files to it.

We then prepared a third regular PC with a Linux Virtual Machine running on it. We installed a special program called Curl on this machine. Curl is a program which allows the user to transfer files from the command line. (Stenberg 2004) It actually simulates a users actions on a web form such that the server does not know that the file is coming from a command line argument and not the actual web form. The user directly types in the hostname with the script name and the file he/she wants to upload.

# 2. Major Entities

This chapter contains all the major entities of the experiment, the experiment setup and the results.

## 2.1 *Soekris net4801:*

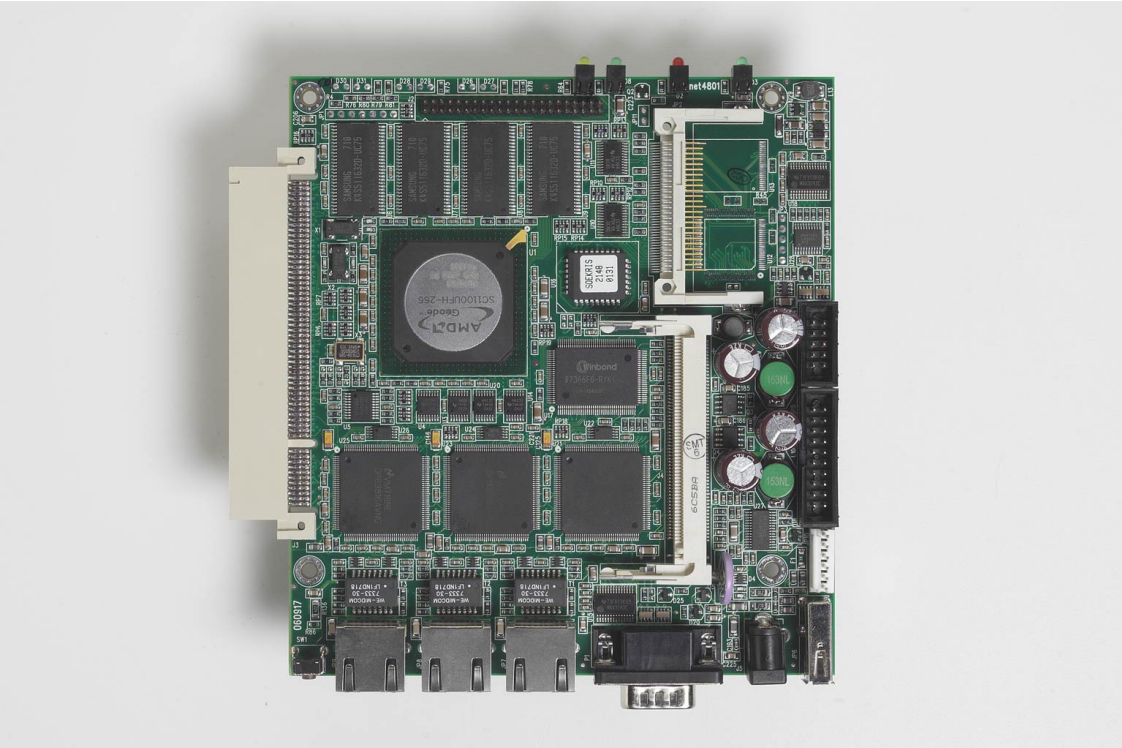



**Figure 2 net4801**

Figure 2 represents a small size, cost effective and a very advanced computing device which has a 266 MHz

586 class processor from AMD. It has three 10/100 Mbit ethernet ports, 128 Mbyte SDRAM and we put in a 512Mb CF card for programs and data. This computer is best used for a firewall or a VPN Router or even an Internet Gateway. It can also be used for other purposes as a communication appliance. The following are the hardware specifications for the experimental computer :

- 233 to 266 MHz NSC SC1100 single chip processor
- 128 Mbyte SDRAM, soldered on board
- 4 Mbit BIOS/BOOT Flash
- Compact FLASH Type I/II socket
- UltraDMA-33 interface with 44 pins connector for 2.5" Hard Drive
- 3 10/100 Mbit Ethernet ports, RJ-45
- 2 Serial ports, DB9 and 10 pins header
- USB 1.1 interface
- Power LED, Activity LED, Error LED
- Mini-PCI type III socket. (t.ex for optional hardware encryption.)

- PCI Slot, right angle 3.3V signaling only, dual PCI slot option planned.
- 12 bit general purpose I/O, 20 pins header
- Temperature and voltage monitor
- Hardware watchdog
- Board size 5.2" x 5.7"
- Power using external power supply is 6-20V DC, max 15 Watt
- Option for 5V supply using internal connector
- Operating temperature 0-60 °C

The following are the software specifications :

- comBIOS for full headless operation over serial port
- PXE boot rom for diskless booting
- Designed for FreeBSD, NetBSD, OpenBSD and Linux
- Runs most real-time operating systems (Soekris 2004)

As previously noted, a stripped down version of FreeBSD called NanoBSD running on the net4801 as the operating system. It also has our firewall software running as a daemon program on it.

## *2.2 Installing an Operating System on Net4801:*

Installing an operating system on this computer could be done in two ways. We followed both.

- Install a version of FreeBSD called MiniBSD (Courtney 2005).
- Install a version of FreeBSD called NanoBSD.

### 2.2.1 Installing MiniBSD:

This is a manual process and it has a lot of room for error. It is also very difficult to reproduce. In this method we created a FreeBSD “jail” which was actually another installation of the same operating system within itself. We then configured the remaining operating system for compatibility with our net4801 by following the steps shown below (Courtney 2005).

- Create Directory Structure
- Rebuild the boot loader
- Building Dynamic Executables

- Copying the binaries over
- Configuring boot files
- kernel compile
- Copying the libraries
- Populating /etc
- Building the binary image
- Writing the binary image to the media
- First Boot
- Add the desired shell(s)

The generic kernel could also work on net4801 but we used a very specific kernel to save space.

### 2.2.2 Installing NanoBSD:

The second method for installing a FreeBSD based operating system on net4801 is installing NanoBSD. The biggest advantage of NanoBSD over other operating systems is that it can be installed using only one shell script. Although it is not very easy for first timers, it is simpler than most other methods. NanoBSD is suitable for

embedded applications and can be installed on a CF card. The features of NanoBSD include the following :

- Ports and packages are the same in FreeBSD and NanoBSD i.e. all the applications that can be installed on FreeBSD can also be installed on NanoBSD.
- No missing functionality, i.e. Everything that you do on FreeBSD can also be done in NanoBSD unless the user specifically removes a part of the functionality from the NanoBSD image while creating it.
- Everything is read-only at run-time i.e. no data can be changed when the operating system is running. Therefore when there is a sudden power outage or any such failure which can potentially hamper the file system, it won't affect our NanoBSD file system.
- Easy to build and customize i.e. Making use of just one shell script and one configuration file it is possible to build reduced and customized images satisfying any arbitrary set of requirements.

Two directories on the root partition are md (malloc disks). They are in the main memory rather than the CF

card. The /etc directory and the /var directory. The /etc directory contains all the setting files which are actually scripts for various programs. If one wants to make any changes to any of these files, he/she has to mount a special directory called /cfg and copy the changed files from /etc to /cfg and then unmount /cfg. What actually happens is that all the files in the /etc and /var directories are lost when the machine reboots because they are in the main memory so there has to be some place on the CF card that has to have the same information. On every startup, these files are fetched from the /cfg directory and put in the memory (Gerzo 2006).

### *2.3 FreeBSD Virtual Machine:*

A NanoBSD image is built from a FreeBSD Virtual Machine. Virtual Machines are complete operating systems running on top of another operating system but as a guest (VMware 2008). There are Virtual Machine applications which are available in the market. Two such popular applications are MS Virtual Server and VMWare Server. The way it works is this, in a regular PC running Windows, we install a

Virtual Machine server. In our case we used the freely available VMware server. We then install a FreeBSD guest in that application. The server is, infact, a complete computer emulated in software. This emulated computer excites programs just like a real computer. Today, there are processors which are available which support virtual machines. These processors have specialized architectures which are conducive for virtual machines. These Virtual Machines have many advantages, the biggest being cost. At the price of a simple PC that people use in day-to-day life, a Virtual Machine enterprise software can be bought. This software can be used to install multiple guest operating systems which act as different kind of servers thus saving the cost of different hardware for each one of those servers. A sample schematic of such a configuration would look something like this:


**Figure 3 Virtual Machine System**

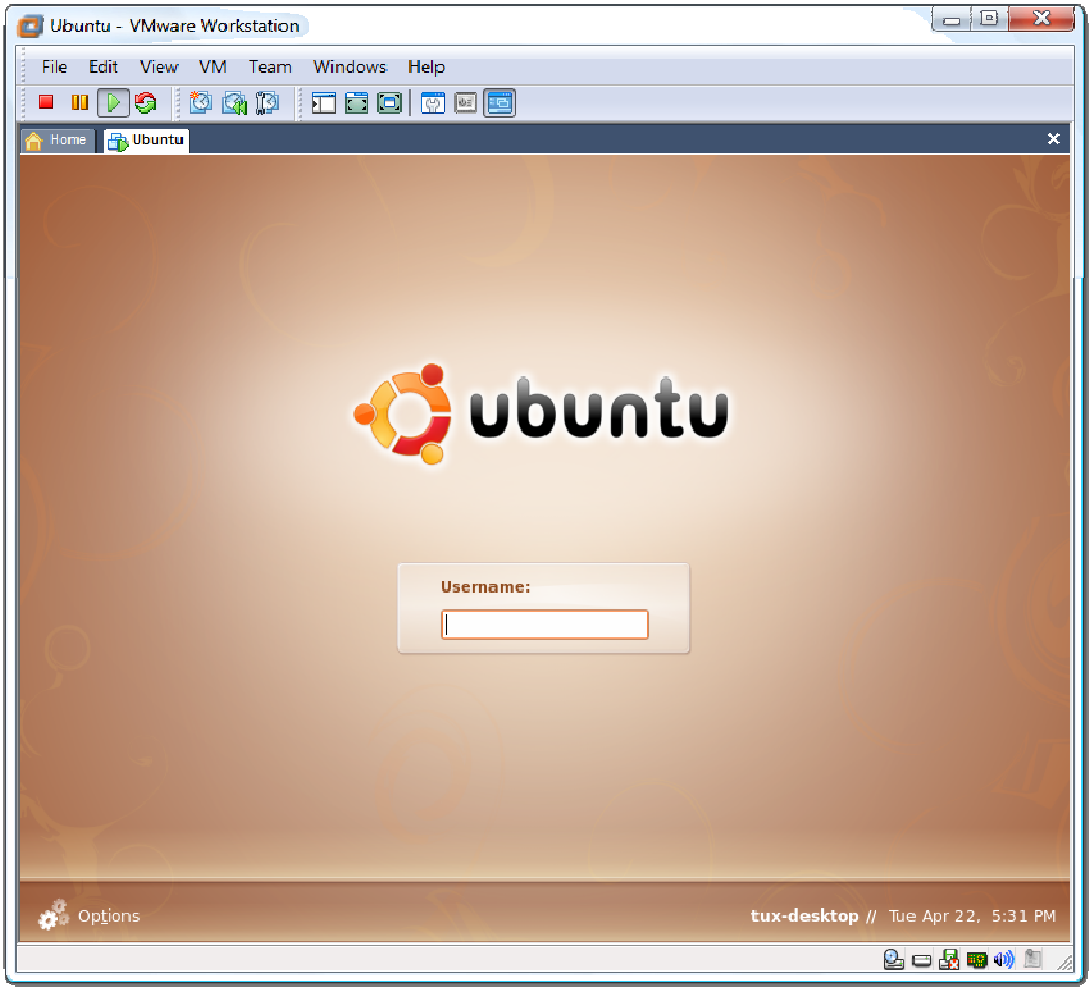



**Figure 4 VMware Workstation running Ubuntu as a guest OS**

The above picture shows a guest operating system (Ubuntu) installed in VMware Workstation which is an enterprise virtual machine software. This is a picture of one guest running in a tab as seen in the picture. Just like this, multiple guest operating systems can be running on the same physical computer. In our case we install a FreeBSD Virtual Machine instead of other guests. The following image shows multiple FreeBSD virtual machines running on VMware Server.

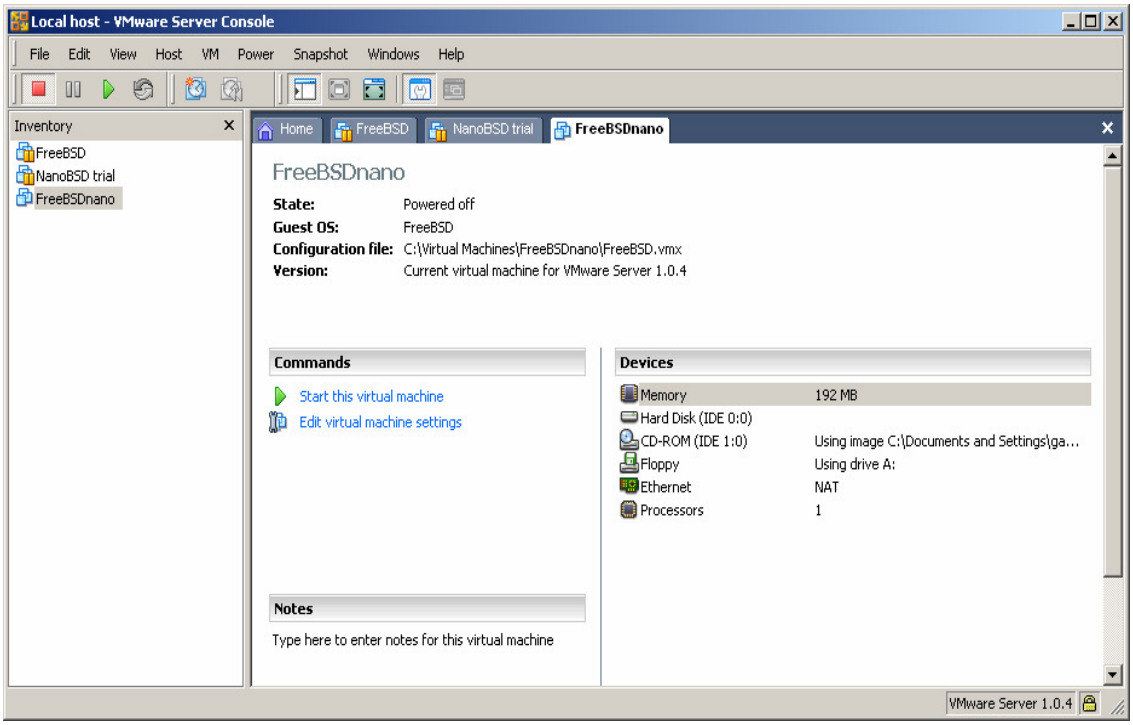



**Figure 5 Multiple FreeBSD Virtual Machines**

### *2.4 NanoBSD Image:*

Once we had the FreeBSD virtual machine up and running we had to create a NanoBSD image. It was created using a simple script which could be found in /usr/src/tools/tools/nanobsd directory. Executing this script create a NanoBSD image in the /usr/obj/nanobsd.full directory. In our case we had certain modifications done to the source code of FreeBSD to accommodate the packet scanning software. We also used a customized kernel configuration file to strip off the modules which we did not need and we also added certain packages in the image which were not a part of the standard installation. We made the following changes to the FreeBSD installation and the NanoBSD image which were out of the ordinary:

- Change a small Part of the source code
- Use a custom kernel configuration file
- Add extra packages which are needed for later use

### 2.4.1 Source Code Patch:

There were some parts of the FreeBSD source code that our packet scanner would use and we had to modify that for the packet scanner to work correctly. Therefore we applied a patch to the FreeBSD source code. This patch was applied in the /usr/src directory and the patch file would take care of the rest of the file paths with respect to the current directory.

### 2.4.2 Custom Kernel Configuration File:

The generic kernel for FreeBSD had many modules which were not useful when you installed the operating system on net4801 for example, the display is not available on the net4801, but the FreeBSD kernel had modules for display. The USB 2.0 module was not useful because the net4801 had only a USB 1.1 available. Hence all these unwanted modules were removed and a custom kernel configuration file was created.

### 2.4.3 Adding extra packages:

The NanoBSD image was customized to accommodate extra packages and the kernel configuration file. This customization could be done using two ways:

- Configuration options
- Custom functions

### 2.4.4 Configuration Options:

With the configuration options it is possible to configure the options which are passed to the buildworld and installworld stages of the image formation. In the buildworld stage, the FreeBSD source which was now modified using the patch, was compiled along with other userland programs which were necessary. In the installworld stage the compiled binaries are injected into the image. These options can enable the user to cut down the size of the NanoBSD image to as low as 64MB and even further down until the image consists of only the kernel and a few more files which are absolutely necessary for the user to interact with the OS. A separate configuration file has to have the

various configuration options, the most important of those being :

- NANO_NAME -- Name of build (used to construct the workdir names).
- NANO_SRC -- Path to the source tree used to build the image.
- NANO_KERNEL -- Name of kernel configuration file used to build kernel.
- CONF_BUILD -- Options passed to the buildworld stage of the build.
- CONF_INSTALL -- Options passed to the installworld stage of the build.
- CONF_WORLD -- Options passed to both the buildworld and the installworld stage of the build.
- FlashDevice -- Defines what type of media to use. Check the FlashDevice.sub file for more details.

These options are also there in the nanobsd.sh file but they have the defaults assigned. The options in the custom configuration file are used to override the default values.

### 2.4.5 Custom Functions:

It is possible to apply the finishing touches to the NanoBSD image using customized shell functions. One can do almost anything desired to the image right from adding packages to adding users and assigning passwords which is a bit of a security loop hole in the FreeBSD architecture. A few examples of such functions are:

- ```
  cust_foo () (
  echo "bar=topless" > \
     ${NANO_WORLDDIR}/etc/foo
  )
  customize_cmd cust_foo
  ```

- ```
  cust_etc_size () (
      cd ${NANO_WORLDDIR}/conf
      echo 30000 > default/etc/md_size
  )
  customize_cmd cust_etc_size
  ```

- ```
  install_packages () (
  mkdir -p ${NANO_WORLDDIR}/packages
  cp /usr/src/tools/tools/nanobsd/packages/*
  ${NANO_WORLDDIR}/packages
  chroot ${NANO_WORLDDIR} sh -c 'cd packages; pkg_add -v
  *;cd ..;'
  rm -rf ${NANO_WORLDDIR}/packages
  )
  customize_cmd install_packages
  ```

We also made one very important change to the nanobsd.sh file instead of keeping the partitions read only, we changed it to read write for our convenience. This can always be changed back to the usual read only format. Once all this is done, we get the net4801 up and running and we install the packet scanning software on it.

### *2.5 Intrusion Detection System:*

Intrusion Detection Systems are softwares which look for attacks signatures. These signatures are specific
  ```

patterns which are usually indications of malicious intent. There is usually a promiscuous node in the network, which looks out for these signatures. There are basically three types of IDSs (Liang 2000) (Marinova-Boncheva 2007). These are:

- Network Based
- Host Based
- Stack Based

### 2.5.1 Network Based IDS:

Network Based IDS use the packets on a network as their data source. They work on the network at real time as packets travel on the network. All the network traffic goes through a first level of filter. This filters out all the unsuspicious traffic. This helps in increasing the performance of the network because all the known un-malicious traffic is filtered out. An example of this would be the following scenario. Consider a node on a network which is suppose to receive a hundred packets, which together are an executable file. Five packets out of these

contain malicious scripts. The first level of filters recognizes these packets and let the remaining ninety-five packets through. The IDS can be configured so that all future packets arriving from the location of the previous hundred packets can be filtered out before the source is repaired and cleaned according to requirement. It is very easy to mis-configure the IDS so that it blocks more traffic than what is required. Once all the above is accomplished, an attack recognition module is brought into picture which recognizes the actual attack packets and finally a third module which is the action module is invoked which takes actions based upon the IDS configurations, for example, it can create a log of the packet or show an alert on a terminal. Hence in our scenario, by the time the ninety-five non-malicious packets are available on the destination, the remaining five packets are acted upon and the user knows what happened to the remaining five packets.

### 2.5.2 Host Based IDS:

This type of IDS came into existence long before networks were prevalent. They work on one simple

fundamental principle, that is, audit logs. The operating system creates logs of all the activities on the host. These logs are reviewed from time to time for any suspicious activity. These days, all the operating systems have a facility. The only difference between now and a few years back is that these systems have become much more automated and sophisticated with the detection and responses. The IDS monitors these logs continuously and responds to new activity or suspicious activity in near real-time. Some host based IDS can also monitor specific ports and alert when some specific port is accessed any time.

### 2.5.3 Stack Based IDS:

This type of IDS is the newest. This type works very closely with the TCP/IP stack. The packets are monitored closely as they traverse the different layers of the OSI reference model. This allows the IDS to recognize malicious packets before they the Operating System or the application has a chance to process them.

Our packet scanning software is primarily a network based IDS. We now are ready to conduct the test to measure performance of this IDS. We check how much time it takes to upload a file from a client to a server without any such IDS and then we do the same with our packet scanner in the network. The whole experiment setup and results are explained in the next section.

## 3. Experiment Setup and Results:

As stated earlier, the main goal of this experiment was to check the performance of a lightweight packet scanner, which scanned packets in real time. To do this we setup two different experiment ideas. In the first experiment, we tried uploading some files based on an interlaid connection from the client to the server via the NET4801 computer and hence via the packet scanner. Where as, in our next experiment, the file transfer was done by eliminating the NET4801 computer and instead directly connecting the client to the server. The client and the server were connected to a DHCP switch in the case where there was no NET4801 in the network. In the case where we had the network packets go through the NET4801 computer on the network, we connected the server to the switch, we connect the NET4801 to the switch and we connect the client to the NET4801. This way all traffic had to go through the NET4801 computer and hence the packet scanner. Both the setups are explained in the diagrams below:

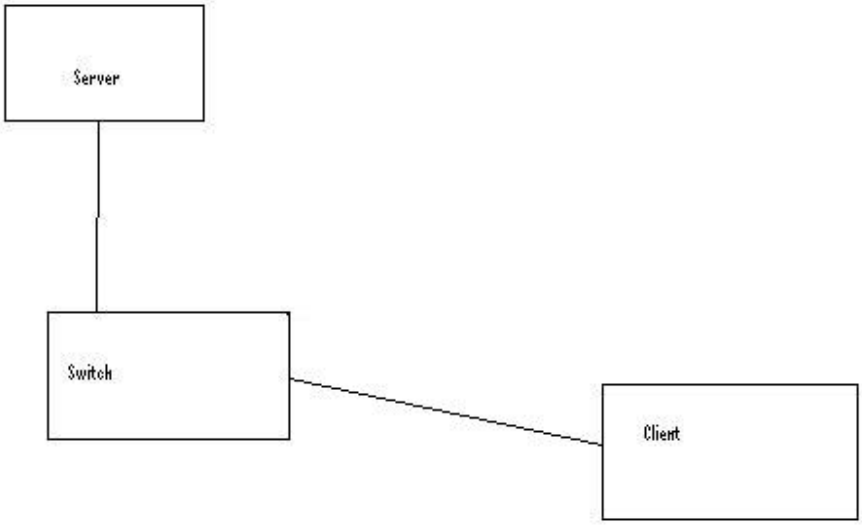



**Figure 6 Experiment Setup Without Net4801**

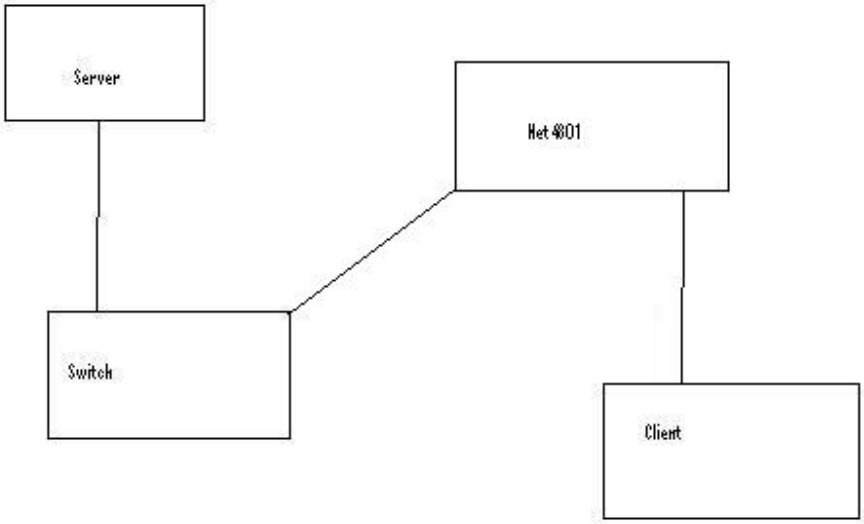



**Figure 7 Experiment Setup With Net4801**

We first set the network up without the NET4801 as shown in figure 6. Once this was done, we executed a program on the client, which uploaded files to the web server by calling a PHP script. This program uploaded the specified number of files to the server in parallel mode using the pthread library. We calculated the time taken by each thread to upload its respective file and we plotted the graphs of the thread number Vs time taken by the thread. We did this for a variable number of files both with the NET4801 and without it. The results are explained with the help of graphs for each different setup in the later section. The packet scanner was supposed to perform better in low traffic conditions and the performance eventually degrades as the network traffic increases. The first setup had five threads to upload five files of 2MB each. First, we did it without the NET4801 and next we did it with NET4801. To summarize the above mentioned setup and the results, the following graphs are shown.

### *3.1 Graphs:*

Number of Files: 5

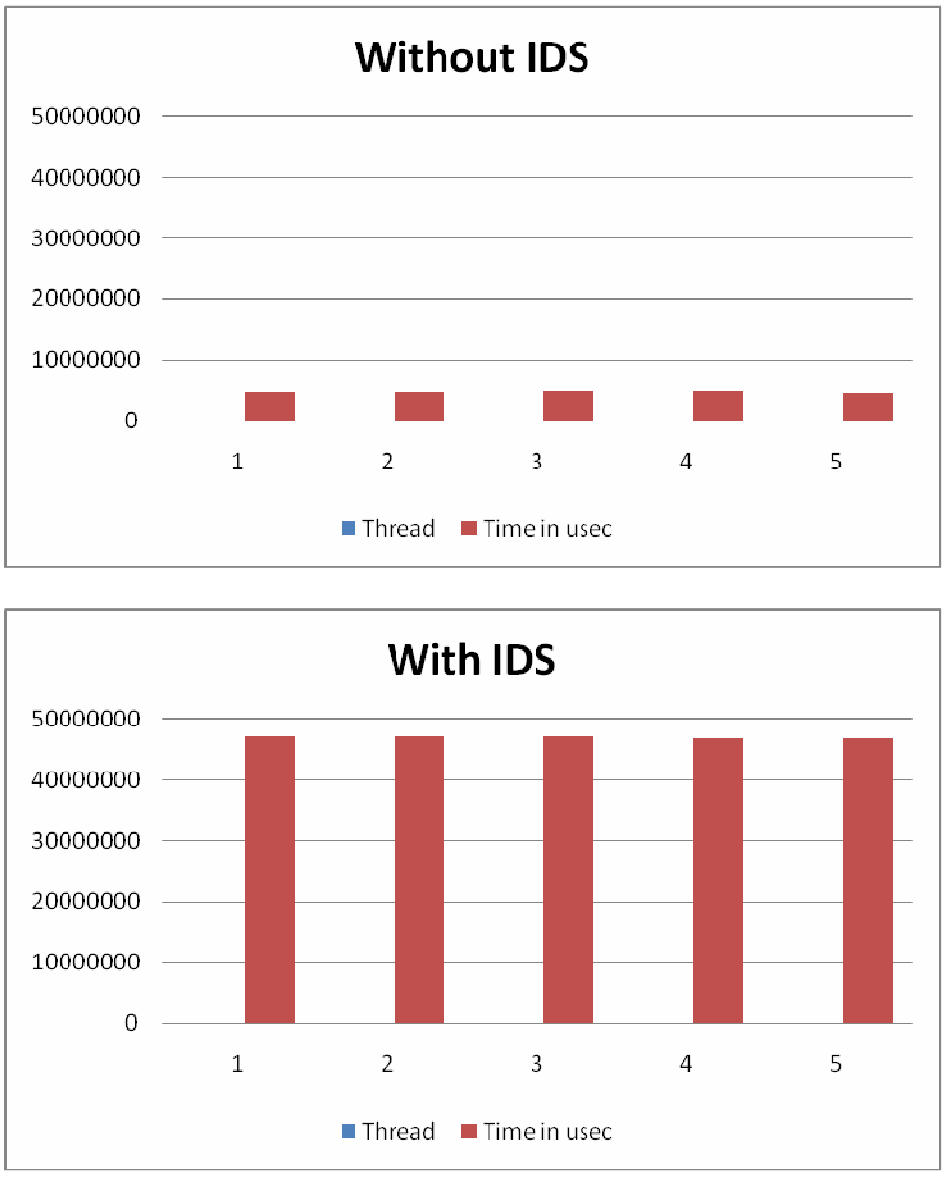



**Figure 8 Five Threads (a) Without IDS (b) With IDS**

Number of Files: 10

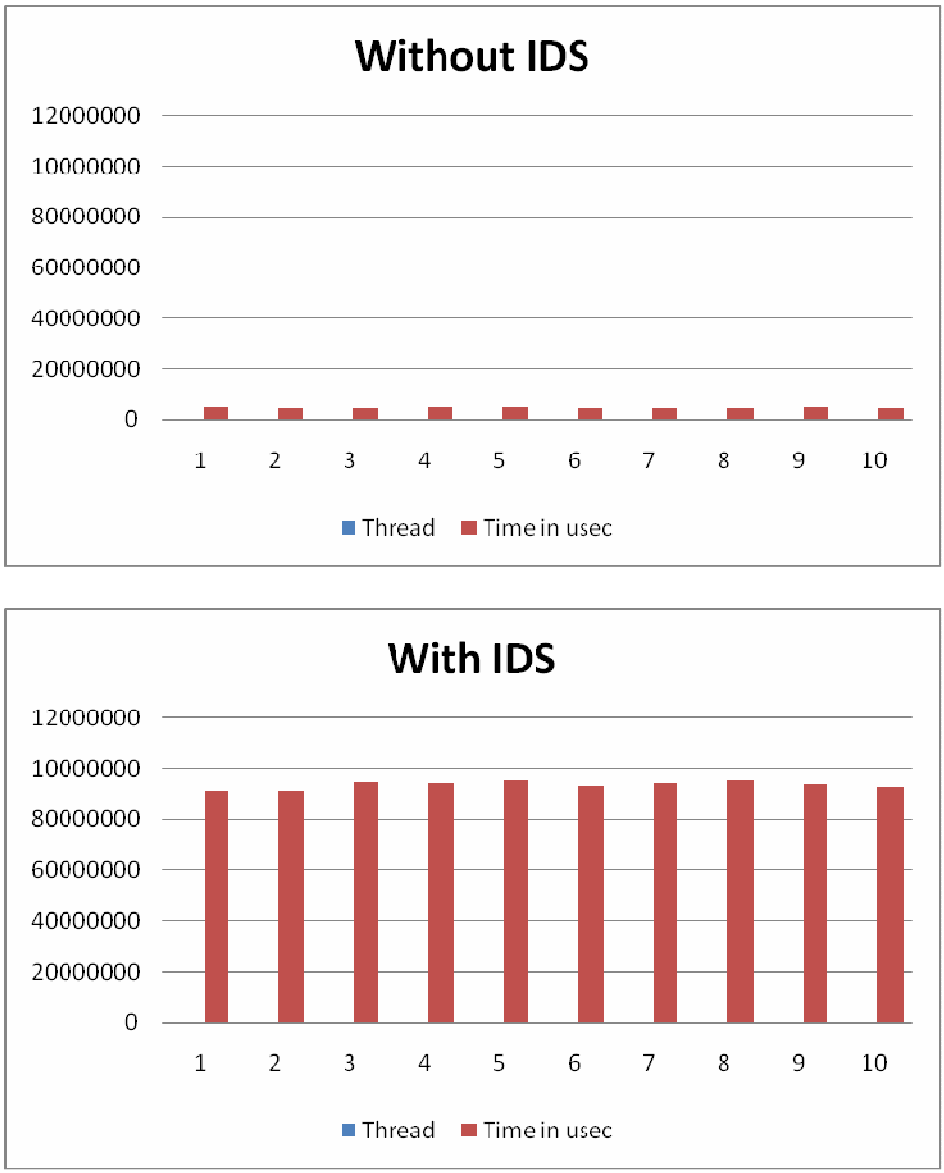
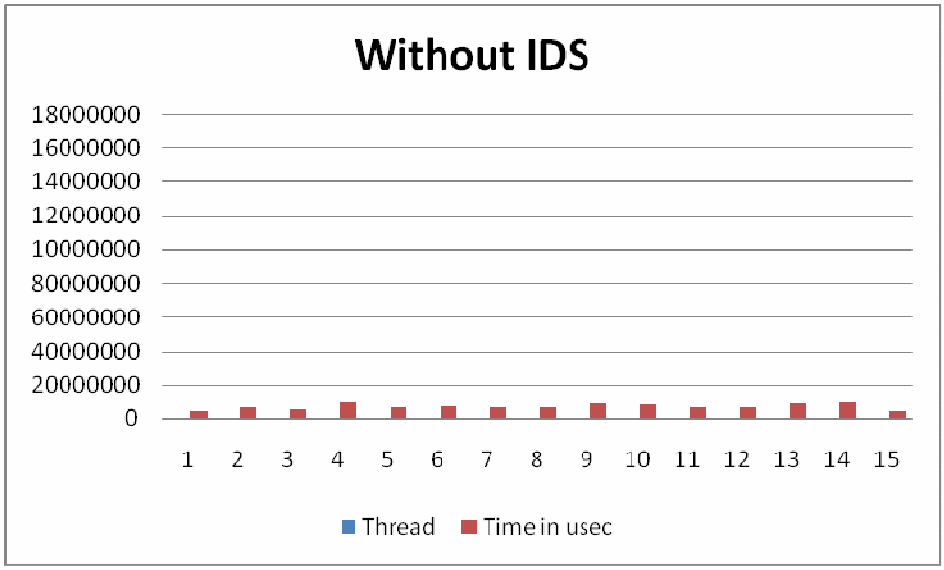



**Figure 9 Ten Threads (a) Without IDS (b) With IDS**

Number of files: 15



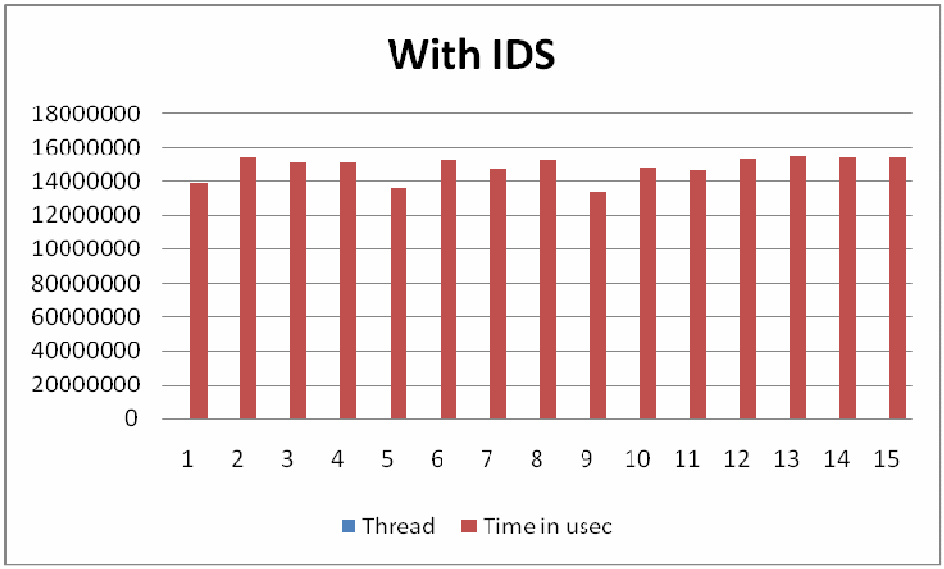




**Figure 10 Fifteen Threads (a) Without IDS (b) With IDS**

Number Of Files: 20

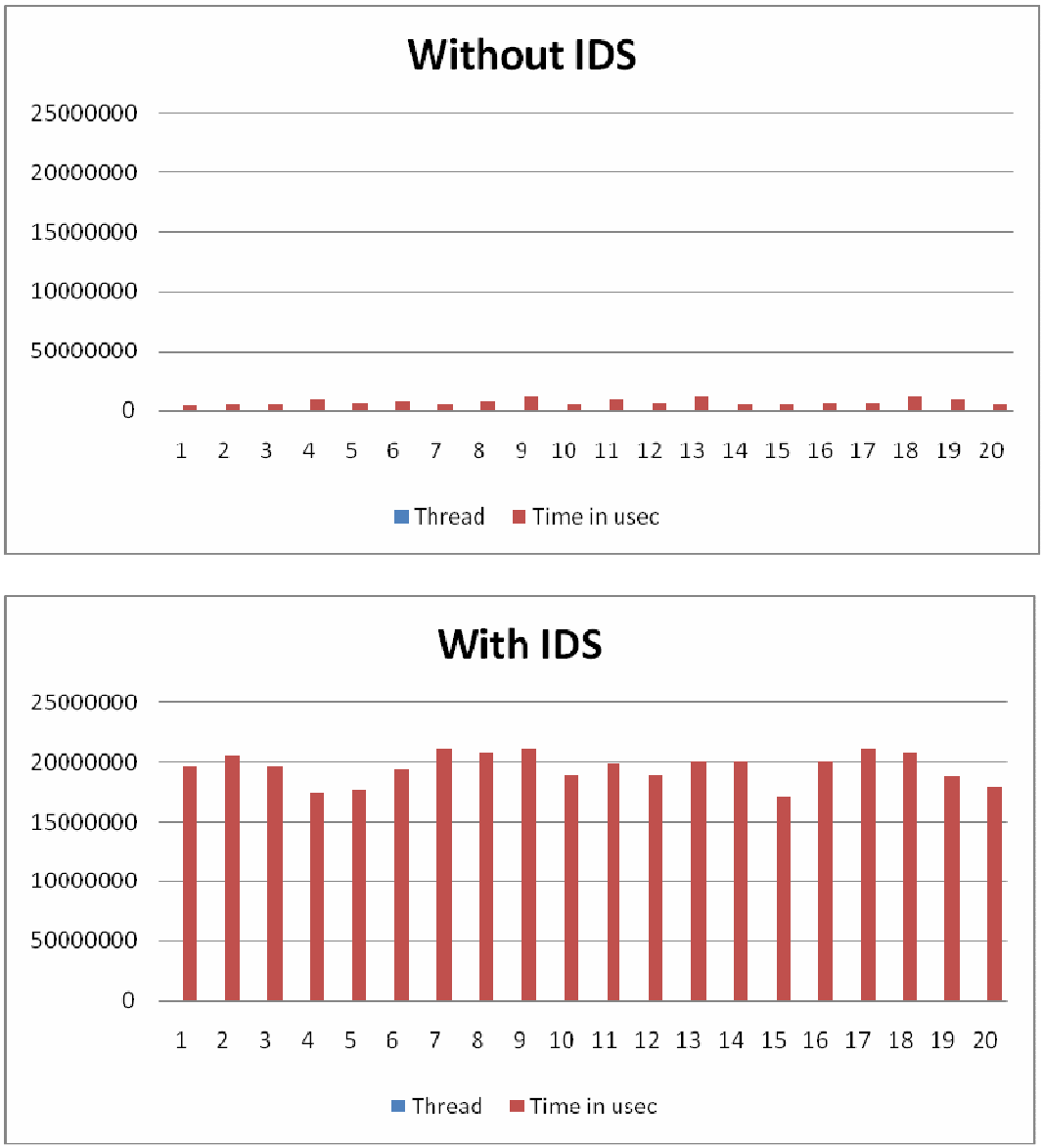



**Figure 11 Twenty Threads (a) Without IDS (b) With IDS**

Number Of Files: 25

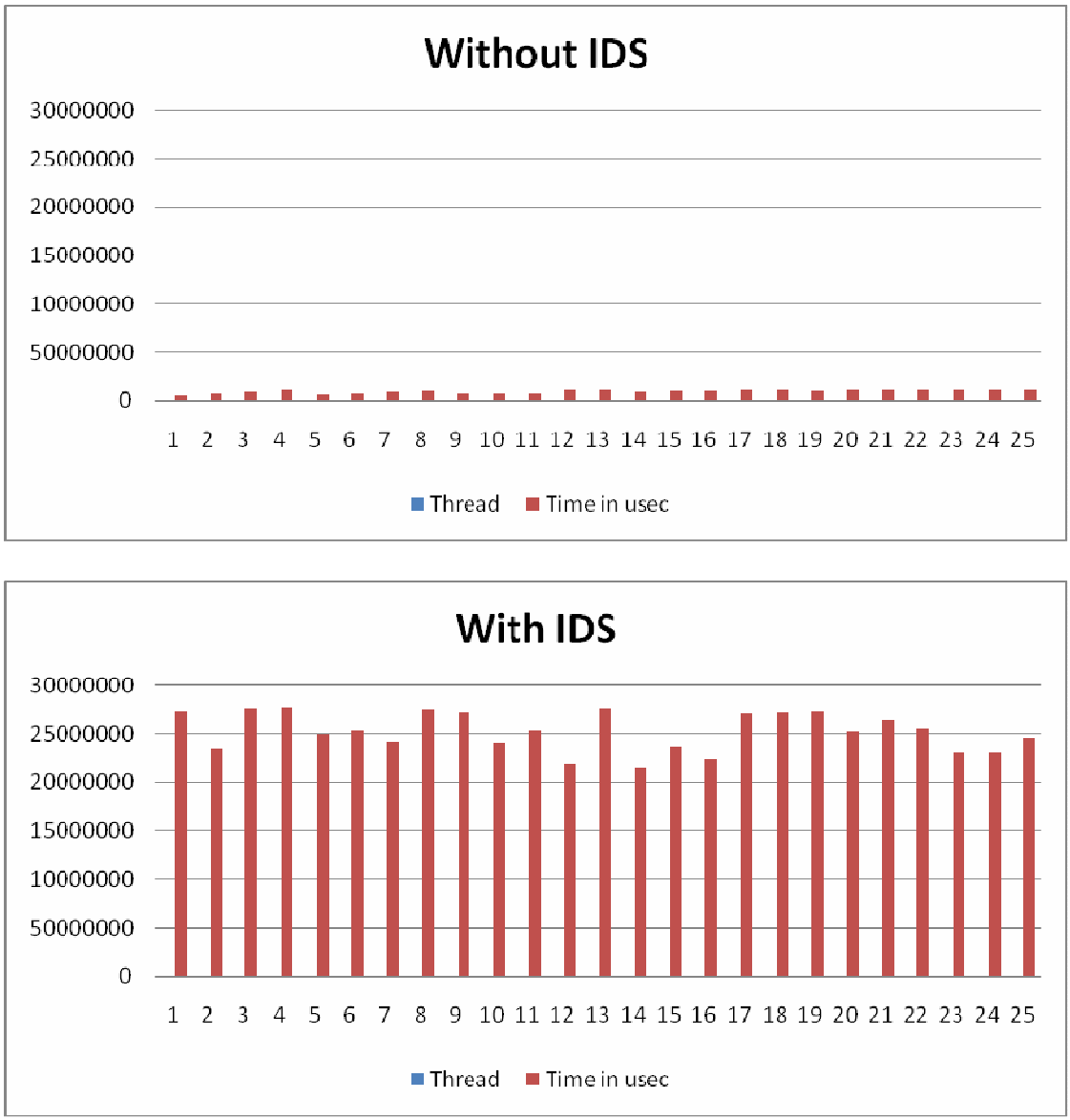
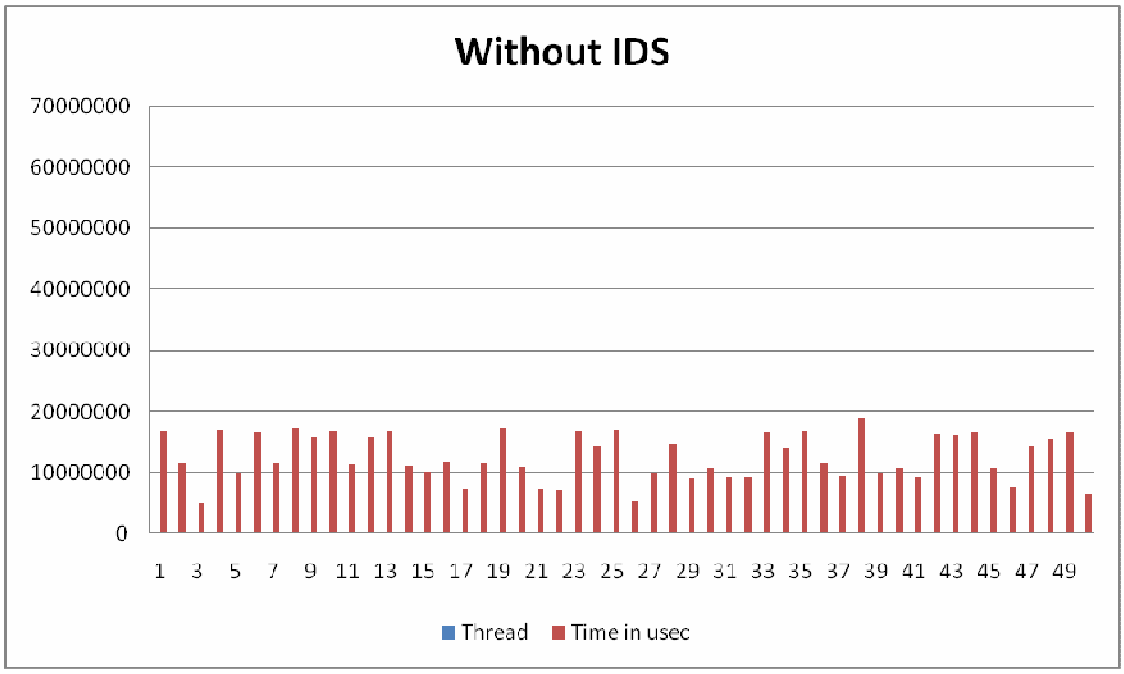



**Figure 12 Twenty Five Threads (a) Without IDS (b) With IDS**

Number Of Files: 50



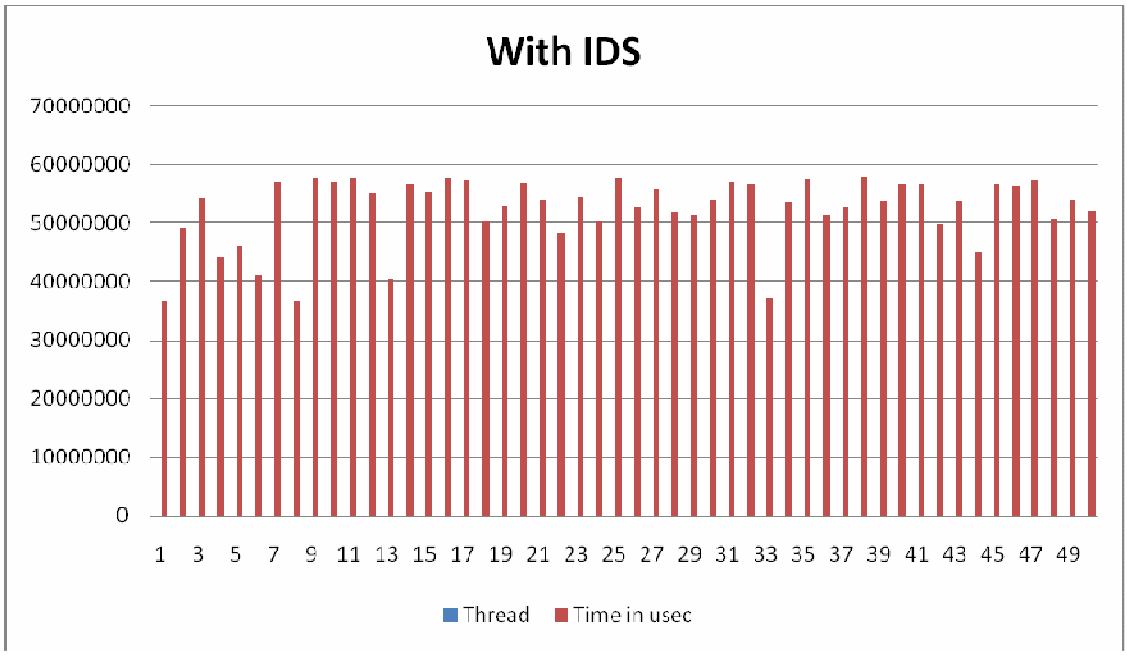




**Figure 13 Fifty Threads (a) Without IDS (b) With IDS**

As clearly seen form the above graphs, the time taken by each thread increses with the introduction of an IDS in the network. Also, since the IDS was a lightweight software, that is, it worked well under low traffic conditions, the performance was better with fewer threads.

# 4. Conclusion

The performance of a network with a firewall in it definitely degrades. In this experiment the firewall was only for name sake. That is, although the packets were scanned by the firewall, it was effectively, only parsing the packet. There was no signature database to actually compare the signature of the packet to the signatures in the database. The performance of the network was still degraded by the packet scanner. If the comparison mechanism were to be involved, the performance of the packet scanner and hence the network would be further hampered as in addition to parsing the packet there would also be comparison.

# Bibliography

Courtney, David. 8 2005. https://neon1.net/misc/minibsd.html#t1 (accessed 2007).

Distrubution, Berkley Software. *FreeBSD Handbook.* 1995. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ (accessed 2008).

Fanglu Guo, Tzi-cker Chiueh. "Traffic Analysis: From Stateful Firewall to Network Intrusion Detection System." Stony Brook: Stony Brook University.

Gerzo, Daniel. "Introduction to NanoBSD." *The FreeBSD Documentation Project.* 2006.

James Harris, Americo J. Melara, Hugh Smith and Phillip Nico. *Performance analysis of the Linux firewall in a host.* California Polytechnic State University, 2002.

Juan Pablo Pereira. *Comparison of Firewall, Intrusion Prevention and Antivirus Technologies.* Sunnyvale: Juniper Networks, Inc., 2006.

Kostic, Chris, and Matt Mancuso. *Firewall Performance Analysis Report.* Hanover: Computer Sciences Corporation, 1995.

Lau, Michael R. Lyu and Lorrien K. Y. "Firewall Security: Policies, Testing and Performance Evaluation." Shatin: The Twenty-Fourth Annual International Computer Software and Applications Conference, 2000.

Liang, Brian. *Intrusion Detection Systems.* Sovereign House, 2000.

Marinova-Boncheva, Vera. "A Short Survey of Intrusion Detection Systems." Bulgaria: Ministry of Education, Project No MU-MI-1601/2006., 2007.

Molitor, Andrew. "Measuring Firewall Performance."

Rebecca Bace, Peter Mell. *Intrusion Detection Systems.* Special Publication, Gaithersburg: National Institute of Standards and Technology , 2001.

Soekris, Engineering. *Soekris Engineering net4801 series boards and systems.* 2004.

Stenberg, Daniel. *Curl Manpage.* August 2004. http://curl.haxx.se/docs/manpage.html (accessed 2008).

VMware. "Installing and Configuring Linux Guest Operating Systems." *VMware Resources.* VMware, 2008.

# Appendix A – Configuration Files and Source Code

```
/* upl.c
 * Author - Paras Gandhi
 * Uploads a Files to a Web Server in parallel mode.
 */

#include <stdio.h>
#include <inttypes.h>
#include <pthread.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/time.h>
#define  __USE_UNIX98
#include <unistd.h>
#include <string.h>


#define MAXFILENAME 20
#define MAXCOMMANDSIZE 255
#define MAXTHREADS 10
#define HOSTNAME 128
```

```
typedef struct
{
      uint64_t microsec;
      char command[MAXCOMMANDSIZE];
      int thnum; //thread number

}thread_struct;

extern void *thread_upl(void *tfu);


int main(int argc, char *argv[])
{
      int maxthreads = MAXTHREADS;
      thread_struct *tfu;
      pthread_t *tid;
      int cret;
      char hostname[HOSTNAME] = "localhost";
      char filename[MAXFILENAME];
      int errflag = 0;
      int i;
      int fd;
      loff_t offset;
```

```
ssize_t res;
ssize_t bytes;
void *buf;
FILE *fp;

while((cret = getopt(argc, argv, "f:h:"))!= -1)
{
     switch (cret)
     {
          case 'f':
               if (optarg == " ")
               {
                    errflag = 1;
                         break;
               }
               maxthreads = atoi(optarg);
               break;
          case 'h':
               if (optarg == " ")
               {
                    errflag = 1;
                    break;
               }
               strcpy(hostname, optarg);
               break;
```

```
                default:
                     printf("\nArgument(s) not supported
!!!\n");
                        errflag = 1;
          }
     }

     if(errflag == 1)
     {
          printf("Using Defaults for wrong arguments!\n");
     }
     tid = malloc(sizeof(pthread_t) * maxthreads);
     tfu = malloc(sizeof(thread_struct) * maxthreads);

     printf("Hostname\t:%s\n",hostname);
     printf("Files\t\t:%d\n",maxthreads);
     for(i = 0; i < maxthreads; i++)
     {

          res = 0x400;
          buf = malloc(0x400);
          offset = 0x200000 - 0x400;
          snprintf(tfu[i].command, MAXCOMMANDSIZE,
     "curl -F ufile=@uplo/%d -F Submit=Upload
http://%s/upload_ac.php > j 2>&1",i,hostname);
          snprintf(filename, MAXFILENAME, "uplo/%d",i);
```

```
            tfu[i].thnum = i;


            if ((fd = open(filename, O_RDWR | O_CREAT |
O_SYNC, 0666)) < 0)
            {
                  perror(filename);
                  exit(0);
            }
            else
            {
                  lseek(fd, offset, SEEK_SET);
                  while(res > 0)
                  {
                        bytes = pwrite(fd, buf, res, offset);
                        if (bytes <= 0)
                        {
                              perror("Write Failed!\n");
                              exit(0);
                        }
                        offset +=bytes;
                        res -= bytes;
                  }
                  close(fd);
            }
            free(buf);
            //printf("%s\n",tfu[i].command);
```

```
        }

        for(i = 0; i < maxthreads; i++)
        {
             pthread_create((&tid[i]),
                       NULL,
                       thread_upl,
                       (void *) &tfu[i]);

        }

        for(i = 0; i < maxthreads; i++)
        {
             pthread_join(tid[i], NULL);
             printf("%d\n",tfu[i].microsec);
        }

        if((fp = fopen("out","w+"))!= NULL)
        {
             fprintf(fp,"Thread\tThreadID\tTime in usec\n");
             for(i = 0; i < maxthreads; i++)
             {
                  fprintf(fp,"%d\t%lld\t%ld\n",
                  i,(long long int)tid[i],(long
int)tfu[i].microsec);
             }
```

```
            fclose(fp);
      }

}


void *thread_upl(void *tfu)
{
      thread_struct *thu;
      int j;
      struct timeval before;
      struct timeval after;


      thu = (thread_struct *)tfu;
      printf("Thread %d created!\n", thu->thnum);

      gettimeofday(&before, NULL);
      system(thu->command);
      gettimeofday(&after, NULL);
      timersub(&after, &before, &after);

      thu->microsec = (uint64_t)after.tv_sec * 1000000
                  + (uint64_t)after.tv_usec;
```

```
        printf("Thread %d Completed!\n", thu->thnum);
        pthread_exit(0);

}
```

```
# MINIBSD
# Author - Paras Gandhi
# Kernel Configuration File for the Nanobsd Image.
# Based on the Generic kernel configuration file for
# FreeBSD
# GENERIC -- Generic kernel configuration file for
# FreeBSD/i386
#

machine         i386
cpu         I586_CPU
ident           MINIBSD

# Options Specific to the Soekris NET48XX
options         CPU_GEODE
#options        CPU_SOEKRIS


# To statically compile in device wiring instead of
/boot/device.hints
#hints "GENERIC.hints"          # Default places to look for
devices.

options    SCHED_4BSD           # 4BSD scheduler
options    INET             # InterNETworking
options    INET6                # IPv6 communications
protocols
options    FFS              # Berkeley Fast Filesystem
options    SOFTUPDATES          # Enable FFS soft updates
support
options    UFS_ACL              # Support for access control
lists
options    UFS_DIRHASH          # Improve performance on big
directories
options    MD_ROOT              # MD is a potential root
device
options    NFSCLIENT        # Network Filesystem Client
options    NFSSERVER        # Network Filesystem Server
```

```
options   NFS_ROOT       # NFS usable as /, requires
NFSCLIENT
options   MSDOSFS             # MSDOS Filesystem
options   CD9660              # ISO 9660 Filesystem
options   PROCFS              # Process filesystem
(requires PSEUDOFS)
options   PSEUDOFS       # Pseudo-filesystem framework
options   GEOM_GPT       # GUID Partition Tables.
options   COMPAT_43      # Compatible with BSD 4.3 [KEEP
THIS!]
options   COMPAT_FREEBSD4          # Compatible with
FreeBSD4
options   SCSI_DELAY=15000    # Delay (in ms) before
probing SCSI
options   KTRACE              # ktrace(1) support
options   SYSVSHM             # SYSV-style shared memory
options   SYSVMSG             # SYSV-style message queues
options   SYSVSEM             # SYSV-style semaphores
options   _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B
real-time extensions
options   KBD_INSTALL_CDEV    # install a CDEV entry in
/dev
options   AHC_REG_PRETTY_PRINT     # Print register
bitfields in debug
# output.  Adds ~128k to driver.
options   AHD_REG_PRETTY_PRINT     # Print register
bitfields in debug
# output.  Adds ~215k to driver.
options   ADAPTIVE_GIANT      # Giant mutex is adaptive.

device          apic          # I/O APIC

# Bus support.  Do not remove isa, even if you have no isa
slots
device          isa
device          eisa
device          pci

# ATA and ATAPI devices
```

```
device          ata
device          atadisk         # ATA disk drives
options   ATA_STATIC_ID  # Static device numbering

# Floating point support - do not disable.
device          npx

# Power management support (see NOTES for more options)
#device         apm
# Add suspend/resume support for the i8254.
device          pmtimer

# Serial (COM) ports
device          sio       # 8250, 16[45]50 based serial
ports

# PCI Ethernet NICs that use the common MII bus controller
code.
# NOTE: Be sure to keep the 'device miibus' line in order
to use these NICs!
device          miibus          # MII bus support
device          sis       # Silicon Integrated Systems SiS
900/SiS 7016

# Wireless NIC cards
#device         wlan      # 802.11 support
#device         an        # Aironet 4500/4800 802.11
#wireless NICs.
#device         awi       # BayStack 660 and others
#device         wi        # WaveLAN/Intersil/Symbol 802.11
#wireless NICs.
#device         wl        # Older non 802.11 Wavelan
#wireless NIC.

# Pseudo devices.
device          loop      # Network loopback
device          mem       # Memory and kernel memory devices
device          io        # I/O device
device          random          # Entropy device
```

```
device          ether           # Ethernet support
#device         sl         # Kernel SLIP
#device         ppp        # Kernel PPP
#device         tun        # Packet tunnel.
device          pty        # Pseudo-ttys (telnet etc)
device          md         # Memory "disks"
device          gif        # IPv6 and IPv4 tunneling
device          faith           # IPv6-to-IPv4 relaying
(translation)

# The `bpf' device enables the Berkeley Packet Filter.
# Be aware of the administrative consequences of enabling
this!
# Note that 'bpf' is required for DHCP.
device          bpf        # Berkeley packet filter

# USB support
#device         uhci       # UHCI PCI->USB interface
#device         ohci       # OHCI PCI->USB interface
#device         ehci       # EHCI PCI->USB interface (USB
2.0)
#device         usb        # USB Bus (required)
#device         udbp       # USB Double Bulk Pipe devices
#device         ugen       # Generic
#device         umass           # Disks/Mass storage -
Requires scbus and da

options    BRIDGE
options    IPDIVERT
options    IPSTEALTH
options    IPFIREWALL
options    IPFIREWALL_VERBOSE
options    IPFIREWALL_VERBOSE_LIMIT=5
options    IPFIREWALL_DEFAULT_TO_ACCEPT
```

```
# myconf.nano
# Custom configuration file for adding functionality
# to the nanobsd image.
NANO_NAME=custom
NANO_SRC=/usr/src
NANO_KERNEL=CDP
NANO_IMAGES=2
NANO_CODESIZE=409600
NANO_CONFSIZE=20480
NANO_DATASIZE=81920
NANO_PACKAGE_DIR=Pkg

CONF_BUILD='
NO_KLDLOAD=YES
NO_NETGRAPH=YES
NO_PAM=YES
'

CONF_INSTALL='
NO_ACPI=YES
NO_BLUETOOTH=YES
NO_CVS=YES
NO_FORTRAN=YES
NO_HTML=YES
NO_LPR=YES
NO_MAN=YES
NO_SENDMAIL=YES
NO_SHAREDOCS=YES
NO_EXAMPLES=YES
NO_INSTALLLIB=YES
NO_CALENDAR=YES
NO_MISC=YES
NO_SHARE=YES
'

CONF_WORLD='
NO_BIND=YES
NO_MODULES=YES
```

```
NO_KERBEROS=YES
NO_GAMES=YES
NO_RESCUE=YES
NO_LOCALES=YES
NO_SYSCONS=YES
NO_INFO=YES
'

FlashDevice SanDisk 512MB

cust_nobeastie() (
    touch ${NANO_WORLDDIR}/boot/loader.conf
    echo "beastie_disable=\"YES\"" >>
${NANO_WORLDDIR}/boot/loader.conf
)

cust_autoboot() (
    touch ${NANO_WORLDDIR}/boot/loader.conf
    echo "autoboot_delay=\"-1\"" >>
${NANO_WORLDDIR}/boot/loader.conf
)

cust_terminal() (
    echo "setenv TERM vt100" >>
${NANO_WORLDDIR}/root/.cshrc
)

cust_lighttpd_conf() (
    sed -i "" -e '/"mod_fastcgi"/s/#/ /'
${NANO_WORLDDIR}/usr/local/etc/lighttpd.conf

    sed -i '' -e
'/server\.errorlog/s/\/var\/log\/lighttpd\.error\.log/\/dev
\/null/' ${NANO_WORLDDIR}/usr/local/etc/lighttpd.conf

    sed -i '' -e
'/accesslog\.filename/s/\/var\/log\/lighttpd\.access\.log/\
/dev\/null/' ${NANO_WORLDDIR}/usr/local/etc/lighttpd.conf
```

```
    sed -i '' -e
'/#fastcgi.server/,/####/s/\/var\/run\/lighttpd\//\/tmp\//'
${NANO_WORLDDIR}/usr/local/etc/lighttpd.conf

    sed -i '' -e '/#fastcgi.server/,/####/s/php-cgi-
cgi/php-cgi/' ${NANO_WORLDDIR}/usr/local/etc/lighttpd.conf

    sed -i '' -e '/#fastcgi.server/,/####/s/^# /  /'
${NANO_WORLDDIR}/usr/local/etc/lighttpd.conf

    sed -i '' -e '/#fastcgi.server/s/#/ /'
${NANO_WORLDDIR}/usr/local/etc/lighttpd.conf

    sed -i '' -e '/^server.document-
root/s/\/usr\/local\/www\/data\//\/www\//'
${NANO_WORLDDIR}/usr/local/etc/lighttpd.conf

)

cust_rc_conf() (
    echo "cloned_interfaces=\"bridge0\"" >>
${NANO_WORLDDIR}/etc/rc.conf
    echo "ifconfig_bridge0=\"addm sis0 stp sis0 addm sis2
stp sis2 up\"" >> ${NANO_WORLDDIR}/etc/rc.conf
    echo "ifconfig_sis0=\"DHCP\"" >>
${NANO_WORLDDIR}/etc/rc.conf
    echo "ifconfig_sis1=\"inet 192.168.1.1 netmask
255.255.255.0\"" >> ${NANO_WORLDDIR}/etc/rc.conf
    echo "ifconfig_sis2=\"up\"" >>
${NANO_WORLDDIR}/etc/rc.conf
    echo "defaultrouter=\"NO\"" >>
${NANO_WORLDDIR}/etc/rc.conf
    echo "lighttpd_enable=\"YES\"" >>
${NANO_WORLDDIR}/etc/rc.conf
    echo "dhcpd_enable=\"YES\"" >>
${NANO_WORLDDIR}/etc/rc.conf
    echo "dhcpd_ifaces=\"sis1\"" >>
${NANO_WORLDDIR}/etc/rc.conf
    echo "sshd_enable=\"YES\"" >>
${NANO_WORLDDIR}/etc/rc.conf
```

```
    echo "syslogd_enable=\"NO\"" >>
${NANO_WORLDDIR}/etc/rc.conf
    echo "filter_enable=\"YES\"" >>
${NANO_WORLDDIR}/etc/rc.conf
)

cust_sysctl_conf() (
    echo "net.link.bridge.ipfw=0" >>
${NANO_WORLDDIR}/etc/sysctl.conf
    echo "net.link.bridge.pfil_member=0" >>
${NANO_WORLDDIR}/etc/sysctl.conf
    echo "net.link.bridge.pfil_bridge=1" >>
${NANO_WORLDDIR}/etc/sysctl.conf
)

cust_sudoers() (
    sed -i "" -e '/^root/i\
www  ALL=(ALL) NOPASSWD: ALL'
${NANO_WORLDDIR}/usr/local/etc/sudoers
)

cust_cdp() (
    echo "/dev/${NANO_DRIVE}s4 /ext ufs rw 1 1" >>
${NANO_WORLDDIR}/etc/fstab
    mkdir -p ${NANO_WORLDDIR}/ext

    cp -f dhcpd.conf
${NANO_WORLDDIR}/root/dhcpd.conf.default
    cp -f filter.conf
${NANO_WORLDDIR}/root/filter.conf.default
    cp -f rc.d.filter
${NANO_WORLDDIR}/root/rc.d.filter.default

    cd ${NANO_DEVICE_DIR}
    chmod -R 700 ${NANO_WORLDDIR}/www
    chown -R www:wheel ${NANO_WORLDDIR}/www
    chmod -R 700  ${NANO_WORLDDIR}/root
```

```
    cp -f ${NANO_WORLDDIR}/etc/rc.conf
${NANO_WORLDDIR}/root/rc.conf.default

    cp -f ${NANO_WORLDDIR}/root/rc.d.filter.default
${NANO_WORLDDIR}/usr/local/etc/rc.d/filter
    cp ${NANO_WORLDDIR}/root/filter.conf.default
${NANO_WORLDDIR}/usr/local/etc/filter.conf
    cp ${NANO_WORLDDIR}/root/dhcpd.conf.default
${NANO_WORLDDIR}/usr/local/etc/dhcpd.conf

    chmod 555 ${NANO_WORLDDIR}/usr/local/etc/rc.d/filter
    chmod 666 ${NANO_WORLDDIR}/usr/local/etc/filter.conf
    chmod 666 ${NANO_WORLDDIR}/usr/local/etc/dhcpd.conf

    chroot ${NANO_WORLDDIR} sh -c 'echo admin | pw useradd
-n admin -h 0'

    cp /usr/share/misc/termcap
${NANO_WORLDDIR}/usr/share/misc/termcap
)

customize_cmd cust_comconsole
customize_cmd cust_install_files
customize_cmd cust_pkg
customize_cmd cust_allow_ssh_root
customize_cmd cust_nobeastie
customize_cmd cust_autoboot
customize_cmd cust_terminal
customize_cmd cust_lighttpd_conf
customize_cmd cust_rc_conf
customize_cmd cust_sysctl_conf
customize_cmd cust_sudoers
customize_cmd cust_cdp
```