Project Report On: Network Monitoring Tool

Indian Institute of Technology, Kanpur

Submitted By:
Harsimran Singh
Third Year
Thapar University
Patiala

CERTIFICATE

This is to certify that Harsimran Singh of Thapar Institute of Engineering and Technology,Kanpur has worked for his winter training project for the session of 2018-2019 on **Network Monitoring Tool** under my supervision in 3rd year .

(Mr. Navpreet Singh)

Chief Computer Engineer

Computer Center

IIT Kanpur

ACKNOWLEDGEMENT

Being students utmost important for us to know how to implement our theoretical knowledge in our daily lives. Even for veteran engineers it sometimes becomes difficult to choose the correct resources while developing a basic tool/software. In such a case it is important that they have a preliminary knowledge about prerequisites which we require. Roped into the project gave us an opportunity to have an exposure of the actual world.

We take this momentous opportunity to express our heartfelt gratitude, ineptness and regards to highly esteemed guide, Dr. Navpreet Singh for providing me an opportunity to work on Project that focuses on creating a Network Monitoring Tool. We with full pleasure converge our heartiest thanks to our project guide, Dr. Saurabh Malhotra for his invaluable advice and cooperation without which this project would not have seen the light of day.

Harsimran Singh

Thapar University

Patiala

Punjab

Index

Institute Profile

About IIT Kanpur

The Indian Institute of Technology Kanpur was established in the year 1960. The Parliament of India passed the ‘Institutes of Technology Act 1961’ declaring all the IITs as “Institutions of National Importance”. When the foundation stone of the IIT system was laid, it was perceived that taking help from the industrially advanced western nations might be a lot more effective in achieving the status of a world class technical institute. Hence, IIT Kanpur had a massive collaboration with USA through a specially created programme – KIAP (Kanpur Indo-American Programme).

IIT Kanpur is a prestigious institute of higher learning producing meritorious students with excellent career growth and universal recognition. The students get the best of opportunities in the form of highly advanced courses, eminent faculty members, well-equipped laboratories, libraries, hostels, and immense facilities to excel in research and development. The selection procedure for students at undergraduate, postgraduate, and research level is highly stringent so that the institute gets the best brains of India. Highly scientific and innovative technology is used for teaching and carrying out research activities. Every year IIT Kanpur is enriched by the laurels brought by the faculty members and the students in the form of research publications, projects, fellowships and industrial exposure.

Research Area

IIT Kanpur has demonstrated its excellence in research in many areas. The institute has a vibrant university atmosphere with a combination of strong academic and research activities. Both compliment each other and the students at all levels under the able guidance of the faculty members maintain this brilliant combination and help the institute to reach the zenith of knowledge and innovation. To cite a few areas: Finite Element Methods using Domain Decomposition, Flow Induced Vibrations, Computational Chemistry, Geonomics and Bioinformatics, Molecular Dynamics, Tomography, Robotics, Neural Networks and Genetic Algorithms, and so on. The most recent initiative of IIT Kanpur has been the formation of a strong research group in the areas of Nanoscience and Nanotechnology.

A HALLMARK OF EDUCATIONAL EXCELLENCE

IIT Kanpur is carrying out original research of significance and technology development at cutting edge. It imparts training for students to make them competent, motivated engineers and scientists. The institute not only celebrates freedom of thought, cultivates vision and encourages growth, but also inculcates human values and concern for the environment and society.

IIT Kanpur has been a part of an initiative to conceive a grand design for technical education in India and implement a concept true to its spirit. This premier institute has been ranked as the no.1 engineering institute of the nation on several occasions. The entire system is ranked third among the top engineering educational systems at the international level. IIT Kanpur cherishes to maintain its leadership role not only at the national but also at the international level.

Objective

The objective of this project is to develop a network monitoring tool. A network monitoring tool oversees the operations of a network. Such a tool is used to ensure availability and overall performance of computers (hosts) and network services. It monitors the network for the problems caused by overloaded and/or crashed servers, network connections and other devices. It notifies the network administrator in case of outages so that he can handle the problem immediately.

This Network Monitoring Tool essentially detects abnormal behavior of any host and in case of such abnormality checks for its activities by decrypting data (packets) sent across the network by this host machine. Thus, by reading this packet information we can analyze the nature of its conversation. Then we can take necessary action against that host in order to make it function properly.
Network Monitoring Tool (NMT) works with the help of tcpdump. tcpdump can be used to capture some or all packets received by a network interface. The range of packets captured can be specified by the using a combination of logical operators and parameters such as source and destination Mac or IP addresses, protocol types (IP and Ethernet) and TCP/UDP port numbers.

Abstract

Network Monitoring Tool enables you to log and monitor data flowing in and out of your network. The program runs as a local service. It helps administrator to check for congestion in the network and the root cause for it. It can be useful in troubleshooting packet loss and latency.

The tool has been designed for TCP (Transmission Control Protocol), UDP (User Datagram Protocol) and ARP (Address Resolution Protocol).It traces timestamp, source IP, destination IP, source port, destination port and packet length.To keep the track record of all the communication and all the data transfer taking place, we need various network tools. These tools constantly monitor a network for slow or failing components and notifies the network administrator.

Network Monitoring Tool lets us know how well the network is running during the course of ordinary operations. It is mainly for monitoring and for analyzing the network traffic between various terminals and systems and then getting a record of all those data into the data files and then to the database. After dumping of data it is then analyzed for any problem in the data transferred by the host machines connected across the network. If any host behaves abnormally we can easily retrieve the dumped data associated with that host in order to tackle the problem related to that host. If still needed yet another tool **decrypting of data** can be used to analyze the problem.

Introduction

Network monitoring for a network is a critical IT function that can save money in network performance, employee productivity and infrastructure cost overruns. Network monitoring can be achieved using various software tools or a combination of plug-and-play hardware and software appliance solutions. A network monitoring tool monitors an internal network for problems. It can find and help resolve snail-paced webpage downloads, lost-in-space e-mail, questionable user activity and file delivery caused by overloaded, crashed servers, dicey network connections or other devices.

When failures, unacceptably slow response or other unexpected behavior is detected these tools send messages to designate locations to notify system administrators, thus providing more operational visibility. Virtually any kind of network can be monitored. It doesn't matter whether it's wireless or wired, a corporate LAN, VPN (Virtual Private Network) or service provider WAN. We can monitor devices on different operating systems with a multitude of functions, ranging from servers, routers and switches to cell-phones.
The Network Monitoring Tool that I have designed is a simple but a very effective tool. Different monitoring tools are designed using different architectures. My Network Monitoring Tools is based on the **LAMP** architecture.

- L stands for Linux operating system.
- A stands for Apache, the web server.
- M stands for the Mysql, the Database Management System.
- P stands for Perl/PHP/Python, the server-side scripting languages.

As stated earlier, in order to design a network monitoring tool the first and the most important step is to capture all the data sent across the network. Any data that is sent across the network is sent in the form of data units that are called **packets**. It turns out that everything one does on the Internet involves packets. For example, every Web page that we receive comes as a series of packets, and every e-mail we send leaves as a series of packets. Networks that ship data around in small packets are called **packet switched networks**.

(To study about packets refer to APPENDIX A1)

Each packet carries the information that will help it get to its destination i.e. the sender's IP address, the intended receiver's IP address, something that tells the network how many packets this e-mail message has been broken into and the number of this particular packet. The packets carry the data in the protocols that the Internet uses: Transmission Control Protocol/Internet Protocol (TCP/IP). Each packet contains part of the body of your message. A typical packet contains perhaps 1,000 or 1,500 bytes.

Most packets split into three parts:

1). Header the header contains instructions about the data carried by the packet. For e.g. Length of the packet, packet number, protocol, destination address etc.

2). Payload also called the **body** or **data** of a packet. This is the actual data that the packet is delivering to the destination. If a packet is fixed-length, then the payload may be **padded** with blank information to make it the right size.

3). Trailer the trailer, sometimes called the **footer**, typically contains a couple of bits that tell the receiving device that it has reached the end of the packet.

In short, the captured data packets are separated on the basis of its type – ARP, IP or UDP, stored in their respective tables in Mysql in a particular format and then finally this traffic is analysed. This tool is capable of providing all packet details sent by any source IP like timestamp, source MAC address, destination MAC address, packet length, destination IP, source port, destination port etc. In this way we can easily catch the infected machine. The top-talker facility makes this tool even more attractive since we can easily find out which hosts are sending the maximum number of packets and also the details of each packet of such hosts are stored in a separate table to help the network administrator.

Methodology

In designing this Network Monitoring Tool, I followed the following steps in sequence: -

- Packet Sniffing
- Packet Processing
- Data Storage

- User Interface

PACKET SNIFFING:

First and most basic step is sniffing packets passing throughout the network. For this we have used command-line packet capture tool **tcpdump. Tcpdump** is a common packet sniffer that runs under the command line. It allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. Tcpdump works on most UNIX like operating systems: Linux, Solaris, BSD, Mac OS X and many others. There is also a port of tcpdump for windows called Windump.

In some operating systems like Fedora, a user must have super user privileges to use tcpdump because the packet capturing mechanism on those systems requires elevated privileges. However, the -z option may be used to drop privileges to a specific unprivileged user after capturing has been set up.

TCPDUMP can provide very detailed information about any network conversation that runs across the wires. An advantage of TCPDUMP is that it can be run remotely through a SSH or TELNET session, and the machine running TCPDUMP does not have to be running x-windows. The application uses very little overhead, since it's a non-graphical interface.

PACKET PROCCESSING:

Second step is to process the captured packets so that they can be written in database. Hence to move further we will process the packets by writing shell script on **gedit editor**. This shell script is the same script that also contains the tcpdump command in the beginning. As stated earlier, this script is invoked at runtime when the user signs in to use the tool.

For packet processing following UNIX commands have been used:

gedit third.sh :: open new vi editor for writing shell script. Now if required to write a plain script to be run on terminal it starts with

#! /bin/sh

Before executing we will be required to change file permission using 'chmod'. Then to execute write 'bash traffic.sh' on terminal (assuming file name to be traffic.sh)

grep: (searching for a pattern) to search for all IP packets and put them into a separate file.

cut: (slitting a file vertically) to appropriately cut the data into file so that they can be written into database.

paste: (pasting files) to paste all cut files to one so that they can be loaded directly to database.

sort: (ordering a file) for use in toptalker to order the ip packets Option n compare according to string numeric value and option r sort in reverse order.

uniq: (locate repeated and non-repeated lines) to count number of packets for toptalker and remove duplicate entries.

head: (displaying the beginning of a file) to display first ten top talkers.

rm: (permanently deletes a file) to remove all the files when the user logs out for privacy purpose.

DATA STORAGE:

After processing, data is needed to be stored into mysql database. It does so by using the following sql commands.

- CREATE DATABASE database name //(to create the database)
- USE database name //(to use the database)
- CREATE TABLE table name(table attributes) //(to create any table in database)
- LOAD DATA LOCAL INFILE “filename” INTO TABLE “table name” //(to load the data of a file into a table in the database)

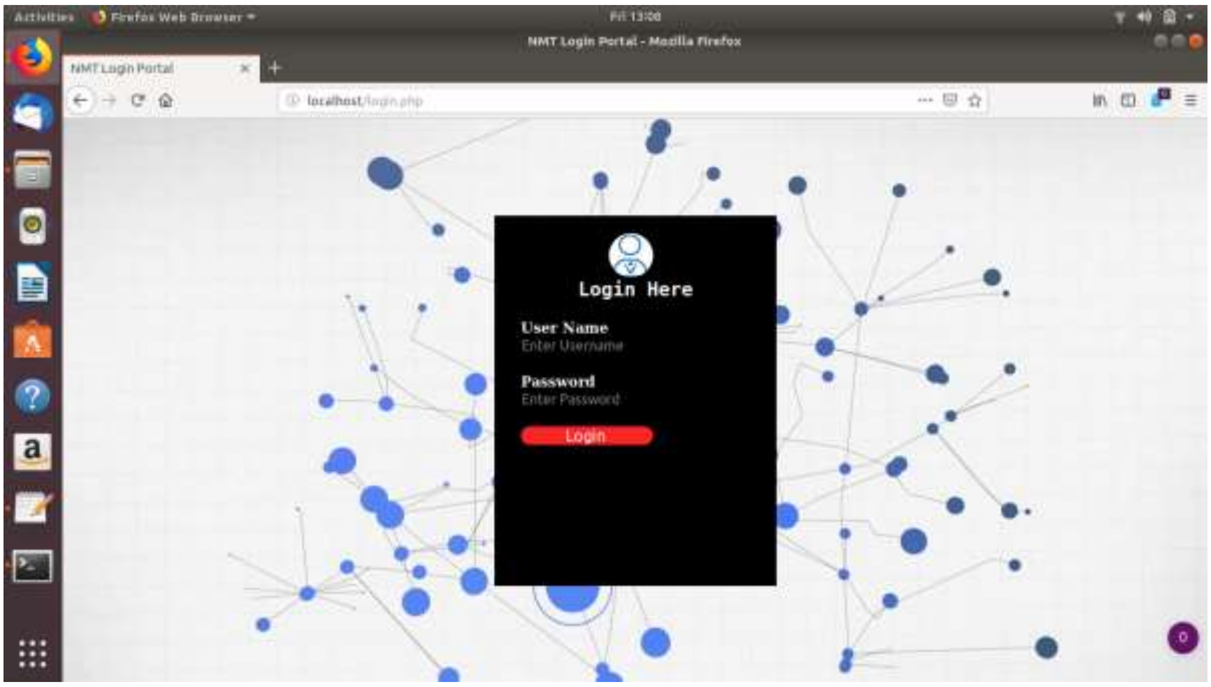
Clearly database is first created then used in which table is created and then the file is loaded to that table. The database commands are stored in the same shell script that is called when the user signs in to use the network monitoring tool.

Another shell script is written to delete all the packets information stored in files and MySQL database. This script is invoked when the user logs out after using the services of the tool. Thus no outsider other than the administrator can access the captured packets information without signing in. This script is simply meant for security purpose.

USER INTERFACE:

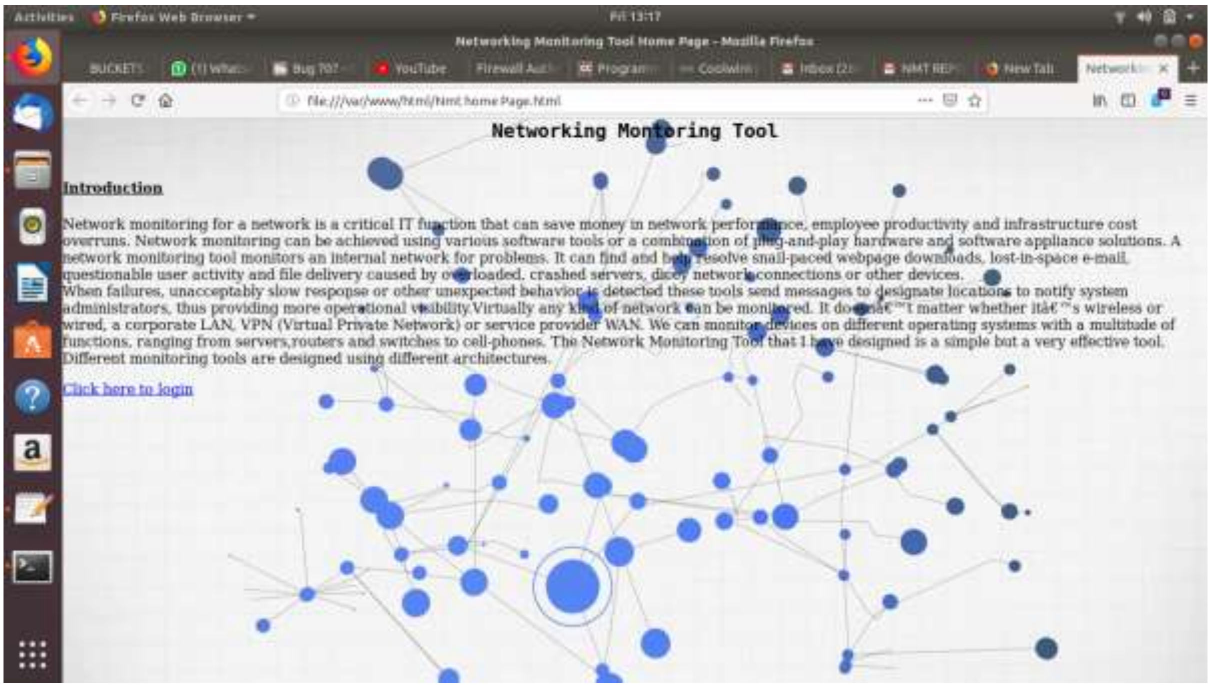
This is next and very essential part of monitoring tool. It is implemented using php and HTML.

Login.Php



- Provides login fields i.e username and password for the registered users.

HomePage.html


Activities
Firefox Web Browser
Fri 13:17
Networking Monitoring Tool Home Page - Mozilla Firefox
file:///var/www/html/Nmt home Page.html

Networking Montoring Tool

Introduction

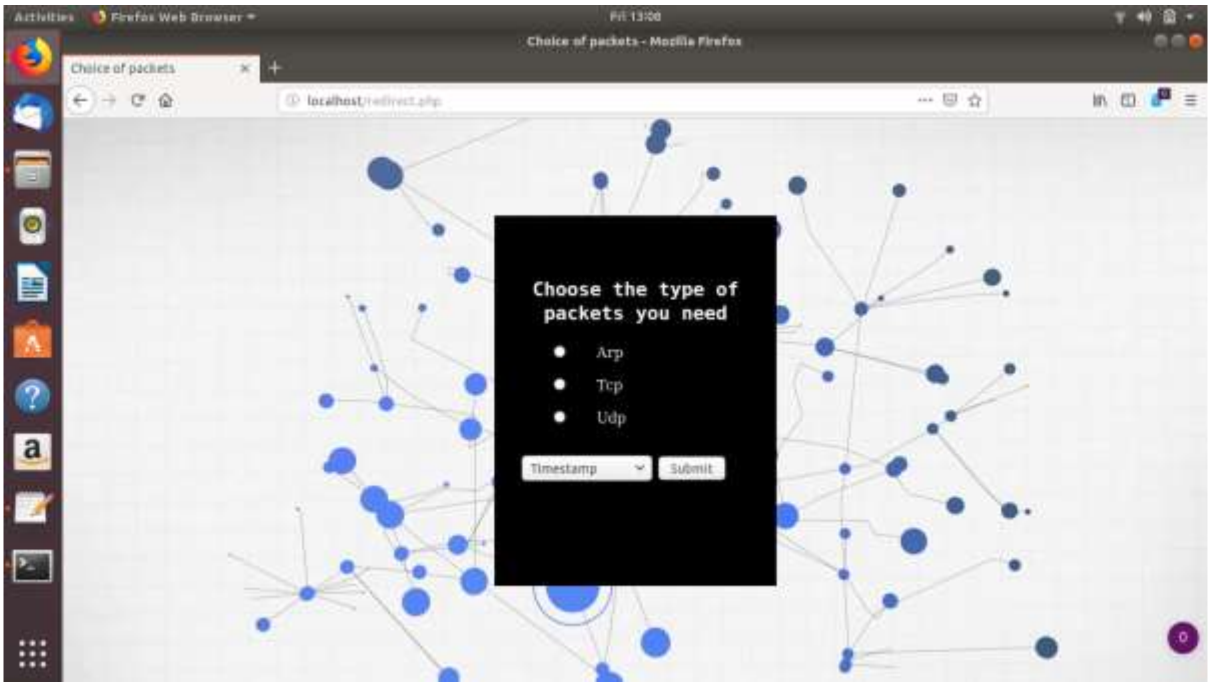
Network monitoring for a network is a critical IT function that can save money in network performance, employee productivity and infrastructure cost overruns. Network monitoring can be achieved using various software tools or a combination of plug-and-play hardware and software appliance solutions. A network monitoring tool monitors an internal network for problems. It can find and help resolve snail-paced webpage downloads, lost-in-space e-mail, questionable user activity and file delivery caused by overloaded, crashed servers, dicey network connections or other devices.
When failures, unacceptably slow response or other unexpected behavior is detected these tools send messages to designate locations to notify system administrators, thus providing more operational visibility.Virtually any kind of network can be monitored. It doesnâ€™t matter whether itâ€™s wireless or wired, a corporate LAN, VPN (Virtual Private Network) or service provider WAN. We can monitor devices on different operating systems with a multitude of functions, ranging from servers,routers and switches to cell-phones. The Network Monitoring Tool that I have designed is a simple but a very effective tool. Different monitoring tools are designed using different architectures.

Click here to login

- Provides a brief information to the newly registered user about Network Monitoring Tool and LAMP architecture.

- Link to login after registering

Redirect.php



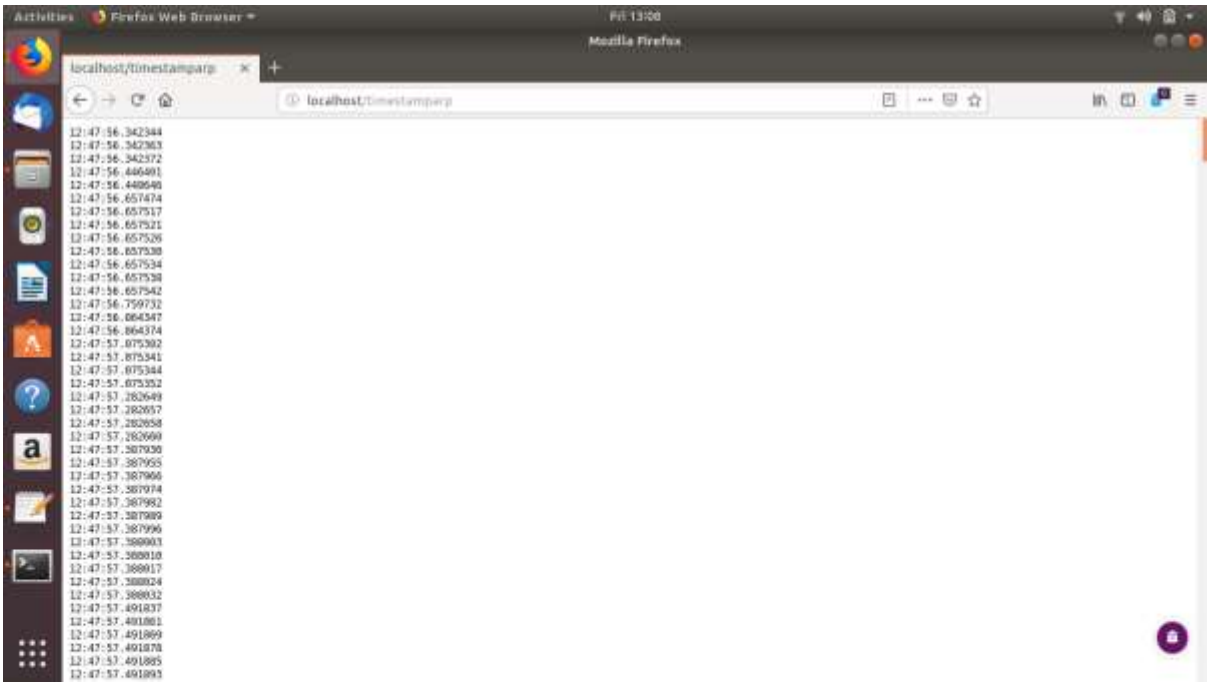
Choose the type of packets you need

- Arp
- Tcp
- Udp

Timestamp Submit

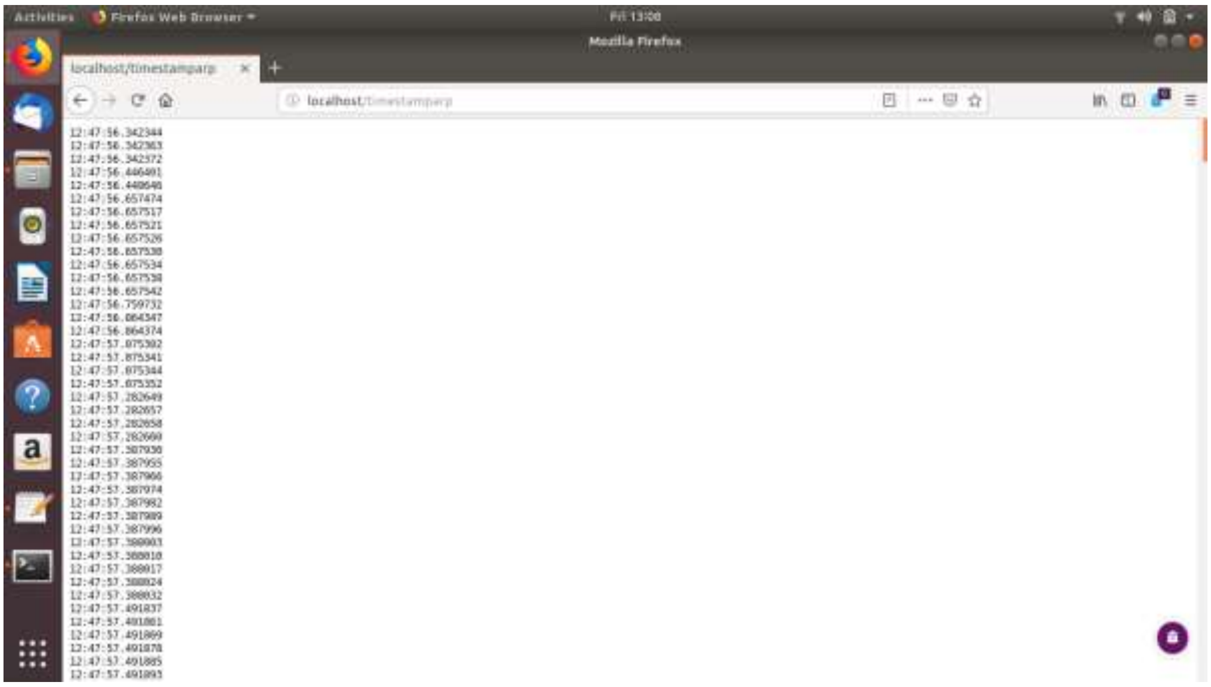
- If the login is successful and authenticated then the above page is displayed otherwise an error message is displayed.
- The user selects the desired packet and field to be displayed in the next page.
- The fields are timestamp, source mac, destination mac, length, type, source ip, destination ip, source port and destination port for the packets ARP, IP, UDP or ALL of them.

TimeStamp Output



```
12:47:56.342344
12:47:56.342363
12:47:56.342372
12:47:56.446491
12:47:56.446646
12:47:56.657474
12:47:56.657517
12:47:56.657521
12:47:56.657526
12:47:56.657530
12:47:56.657534
12:47:56.657538
12:47:56.657542
12:47:56.759732
12:47:56.864347
12:47:56.864374
12:47:57.075302
12:47:57.075341
12:47:57.075344
12:47:57.075352
12:47:57.282649
12:47:57.282657
12:47:57.282658
12:47:57.282660
12:47:57.387930
12:47:57.387955
12:47:57.387966
12:47:57.387974
12:47:57.387982
12:47:57.387989
12:47:57.387996
12:47:57.388003
12:47:57.388010
12:47:57.388017
12:47:57.388024
12:47:57.388032
12:47:57.491837
12:47:57.491861
12:47:57.491869
12:47:57.491878
12:47:57.491885
12:47:57.491893
```

SourceIP Output



```
12:47:56.342344
12:47:56.342363
12:47:56.342372
12:47:56.446491
12:47:56.446646
12:47:56.657474
12:47:56.657517
12:47:56.657521
12:47:56.657526
12:47:56.657530
12:47:56.657534
12:47:56.657538
12:47:56.657542
12:47:56.759732
12:47:56.864347
12:47:56.864374
12:47:57.075302
12:47:57.075341
12:47:57.075344
12:47:57.075352
12:47:57.282649
12:47:57.282657
12:47:57.282658
12:47:57.282660
12:47:57.387930
12:47:57.387955
12:47:57.387966
12:47:57.387974
12:47:57.387982
12:47:57.387989
12:47:57.387996
12:47:57.388003
12:47:57.388010
12:47:57.388017
12:47:57.388024
12:47:57.388032
12:47:57.491837
12:47:57.491861
12:47:57.491869
12:47:57.491878
12:47:57.491885
12:47:57.491893
```

Tools/Softwares Used

This project is based on L.A.M.P. Architecture. The LAMP refers to a solution stack of software, usually free and open source software, used to run dynamic Web sites or servers.

- Linux, the operating system
- Apache, the web server
- MySQL, the database
- PHP, the programming language.

1) LINUX OPERATING SYSTEM

Linux, is a generic term referring to a family of Unix-like computer operating systems based on the Linux kernel. Their development is one of the most prominent examples of free and open source software collaboration; typically all the underlying source code can be used, freely modified, and redistributed by anyone

Users can control a Linux-based system through a command line interface (or CLI), a graphical user interface (or GUI), or through controls attached to the associated hardware (this is common for embedded systems). For desktop systems, the default mode is usually graphical user interface (or GUI).

A Linux system typically provides a CLI of some sort through a shell, which is the traditional way of interacting with a Unix system. A Linux distribution specialized for servers may use the CLI as its only interface. A “headless system” run without even a monitor can be controlled by the command line via a remote-control protocol such as SSH or telnet.

Advantages of using Linux:

Upgradeability

- The core of the Linux operating system is free and is uploaded constantly with new features and support for new hardware.
- Many of Linux’s large quantity of programs are “open source”, allowing the holder of the software to improve the software in whatever way they want as long as credit is given.

Security

- Well-done administration of a computer with Linux allows for a very secure multi-user.
- Workstation.
- Virus is less of a threat now.

LINUX ARCHITECTURE:

DESKTOP (KDE, Gnome, XFCF)

|

Windows Manager

|

X Windows system/X11

|

CLI/SHELL

|

Kernel

Hardware

Linux system command is “tcpdump”. (Refer to APPENDIX A1 to see manual of TCPDUMP)

2) APACHE HTTP SERVER:

The **Apache HTTP Server**, commonly referred to as **Apache**, is a web server notable for playing a key role in the initial growth of the World Wide Web and in 2009 became the first web server to surpass the 100 million web site milestone [2]. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently known as Sun Java System Web Server), and has since evolved to rival other Unix-based web servers in terms of functionality and performance. The majority of all web servers using Apache are Linux web servers.

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support mod_perl, mod_python, Tcl, and PHP. Popular authentication modules include mod_access, mod_auth, mod_digest, and mod_auth_digest, the successor to mod_digest. A sample of other features include SSL and TLS support (mod_ssl), a proxy module, a URL rewriter (also known as a rewrite engine, implemented under mod_rewrite), custom log files (mod_log_config), and filtering support (mod_include and mod_ext_filter).

Apache is primarily used to serve both static content and dynamic Web pages on the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides.

Apache is the web server component of the popular LAMP web server application stack, alongside Linux, MySQL, and the PHP/Perl/Python (and now also Ruby) programming languages.

SHELL PROGRAMMING:A SHELL is a command line interpreter which takes commands at the command line and then executes them. In general, a shell executes a programming language. The Bourne Shell in Linux can be used to create shell a script i.e. programmers that are interpreted and executed by the shell.

3) <u>MySQL</u>

MySQL is popular for web applications and acts as the database component of the LAMP software stack. Its popularity for use with web applications is closely tied to the popularity of PHP, which is often combined with MySQL.

Advantages of Mysql

The following features are implemented by MySQL but not by some other RDBMS software:

- Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.
- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.
- To administer MySQL databases one can use the included command-line tool (commands: mysql and mysqladmin).

Features OF Mysql:

- Stored procedures
- Triggers
- Cursors
- Updatable Views , Query caching
- True VARCHAR support
- INFORMATION_SCHEMA, SSL support

4) PHP/HTML

PHP, which stands for “Hypertext Preprocessor”, is a server-side, HTML embedded scripting language used to create Dynamic Web Pages. Much of its syntax is borrowed from C, Java and Perl with unique features thrown in. The goal of the language is to allow Web developers to write dynamically generated pages quickly.

PHP generally runs on a web server, taking PHP code as its input and creating web pages as output. It can also be used for command-line scripting and client-side GUI applications. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

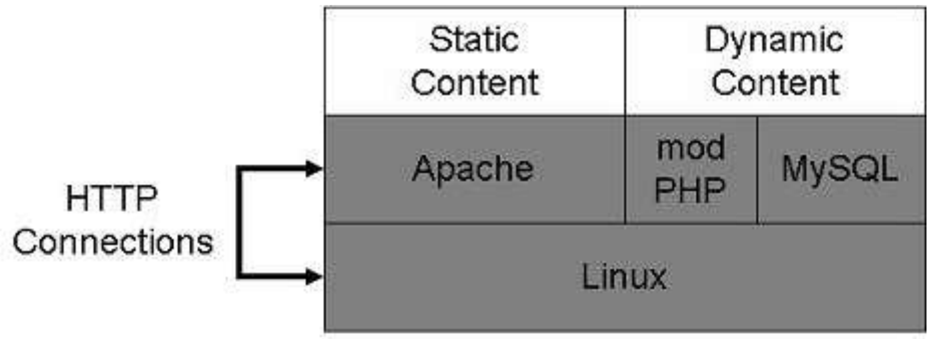
As with many scripting languages, PHP scripts are normally kept as human-readable source code, even on production web servers.[39] In this case, PHP scripts will be compiled at runtime by the PHP engine, which increases their execution speed. PHP scripts are able to be compiled before runtime using PHP compilers as with other programming languages such as C.

What is a PHP file?

- PHP files may contain text, HTML tags and scripts.
- PHP files are returned to the browser as plain HTML.
- PHP files have a file extension of .php, .php3 or .P.HTML
- PHP + MySQL

How does LAMP Work?

LAMP is singularly focused towards Web applications. The architecture is very straightforward, as illustrated in Figure. Linux forwards HTTP connections to Apache, which serves static content directly from the Linux kernel. Dynamic pages are forwarded by Apache to PHP, which runs the PHP code to design the page. Database queries are sent to MySQL through PHP. Administration is commonly handled through phpMyAdmin, and every major enterprise management system can manage Apache and Linux.



There are numerous resources on the Web explaining how to quickly build a LAMP application. Most developers are productive within hours of installing the LAMP stack. Linux isn't a necessary standard because development often occurs on a Windows platform running Apache, MySQL, and either PHP, Python, or Perl (this configuration is called "WAMP").

Where Is LAMP?

LAMP is composed of several separate open source projects, each with its own owners, packages, and distributors. Each of the core components of LAMP has a major commercial or nonprofit foundation behind it:

- Linux: Red Hat, Novell
- Apache: Covalent
- MySQL: MySQL AB
- PHP/Python/Perl: Zend, Python Software Foundation, the Perl Foundation

Because it can be onerous to piece together a LAMP stack, a few stack vendors have emerged, including: (Disclosure: this includes ActiveGrid, which provides the LAMP stack underneath its application server) ActiveGrid, BitRock, SpikeSource, and SourceLabs

Although the projects are each under distinct open source licenses, the code that a developer writes for his or her applications does not need to be released as open source. However, if the software contains modifications to some of the projects (including the Linux kernel and the MySQL database), those modifications need to be published under the GPL license.

Why Lamp?

For Web applications, LAMP has been proven faster, cheaper, more flexible, and easier than any alternative. There is a strong push to LAMP by vendors ranging from IBM to Oracle to numerous startups—and these vendors are adding enterprise-grade capabilities and management to LAMP. There is no question that LAMP is not a passing trend, but now entering the mainstream as a serious contender to J2EE and .NET. Now that you know what it is, who uses it and how it works, it's time to start building a LAMP application yourself! Who knows, you just might love it.

Conclusion

This Network Monitoring Tool is an essential networking utility tool used by the centralized server system for any organization. The internet facility of any organization has to be monitored regularly so as to check for any cases of misuse and misapplication. So to keep a track of such things, all the data being transmitted from one terminal or system to another has to be recorded and analyzed. It can be used as a utility which can be installed on any server machine and thus is useful in finding the faults, misuse, congestion or any legal or illegal problem held at any attached terminal and proper action can be taken.

It gives us a tool, utility software which can give us solution to many networking problems in no time. With this only its objective area widens and so the scope. The end result of this project is that it will act as a utility tool and can be easily installed on any application server and there it can be analyzed as on what time which packet was communicating with the server, what data has being transferred, whether the application server is not being hacked or misused, and which of the terminals is unable to communicate with server due to crash of the system or any fault on any system terminal.

The scope also includes the help this project will provide to the users or the administrator. After being installed on any terminal a variety of checks can be done. To every illegal or any interrupted service occurs, alter is generated and is sent to the administrator to recover the faults. A complete search report is maintained and generated according to the search option.

Future Scope

The data packets being transmitted across a network, including the addresses of the source and destination computers in its header part, is accessible to others who may be monitoring the network. For security, the data is often encrypted before being sent on the network. Along with the header and payload a 'cipher key' is passed along with each packet. The receiving end first decodes this cipher and then by use of cipher it decodes the data. Whole data is broken into several packets each of which contains header, cipher, payload, and footer.

In future work I need to extend the project so that it can decode the cipher and use that cipher to decode the data so that along with knowing infected IP using TOPTALKER we can also know what the infection is. Also telnet send data in plain text. But it is not in human readable form it still needs cipher to be decoded. In future we should be able to decode the data sent between two IPs in proper human readable format so that the centralized server has full power on network through this tool.

Appendix A

1) **PACKETS:**

It turns out that everything you do on the Internet involves **packets**. For example, every Web page that you receive comes as a series of packets, and every e-mail you send leaves as a series of packets. Networks that ship data around in small packets are called **packet switched networks**.

On the Internet, the network breaks an e-mail message into parts of a certain size in bytes. These are the packets. Each packet carries the information that will help it get to its destination -- the sender's IP address, the intended receiver's IP address, something that tells the network how many packets this e-mail message has been broken into and the number of this particular packet. The packets carry the data in the protocols that the Internet uses: Transmission Control Protocol/Internet Protocol (TCP/IP). Each packet contains part of the body of your message. A typical packet contains perhaps 1,000 or 1,500 bytes.

Each packet is then sent off to its destination by the best available route -- a route that might be taken by all the other packets in the message or by none of the other packets in the message. This makes the network more efficient. First, the network can balance the load across various pieces of equipment on a millisecond-by-millisecond basis. Second, if there is a problem with one piece of equipment in the network while a message is being transferred, packets can be routed around the problem, ensuring the delivery of the entire message.

Depending on the type of network, packets may be referred to by another name:

- Frame
- Block
- Cell
- Segment

2) TCPDUMP:

TCPDUMP required ROOT ACCESS or the program must have suid of root. It is usually located in '/usr/sbin/tcpdump'.
Tcpdump is a common packet sniffer that runs under the command line. It allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. It was originally written by Van Jacobson, Craig Leres and Steven McCanne who were, at the time, working in the Lawrence Berkeley Laboratory Network Research Group.

Tcpdump works on most Unix-like operating systems: Linux, Solaris, BSD, Mac OS X, HP-UX and AIX among others. In those systems, tcpdump uses the libcap library to capture packets.

There is also a port of tcpdump for Windows called WinDump; this uses WinPcap, which is a port of libpcap to Windows.

In some Unix-like operating systems, a user must have superuser privileges to use tcpdump because the packet capturing mechanisms on those systems require elevated privileges. However, the -Z option may be used to drop privileges to a specific unprivileged user after capturing has been set up. In other Unix-like operating systems, the packet capturing mechanism can be configured to allow non-privileged users to use it; if that is done, superuser privileges are not required.

SYNOPSIS

tcpdump [-adeflnNOpqStvx] [-c count] [-F file][-i interface] [-r file] [-s snaplen][-T type] [-w file] [expression]

OPTIONS

-A Print each packet (minus its link level header) in ASCII. Handy for capturing web pages.

-c Exit after receiving *count* packets.

-C Before writing a raw packet to a savefile, check whether the file is currently larger than *file_size* and, if so, close the current savefile and open a new one. Savefiles after the first savefile will have the name specified with the **-w** flag, with a number after it, starting at 1 and continuing upward. The units of *file_size* are millions of bytes (1,000,000 bytes, not 1,048,576 bytes).

-d Dump the compiled packet-matching code in a human readable form to standard output and stop.

-dd Dump packet-matching code as a **C** program fragment.

-ddd Dump packet-matching code as decimal numbers (preceded with a count).

-D Print the list of the network interfaces available on the system and on which *tcpdump* can capture packets. For each network interface, a number and an interface name, possibly followed by a text description of the interface, is printed. The interface name or the number can be supplied to the **-i** flag to specify an interface on which to capture. This can be useful on systems that don't have a command to list them (e.g., Windows systems, or UNIX systems lacking **ifconfig -a**); the number can be useful on Windows 2000 and later systems, where the interface name is a somewhat complex string. The **-D** flag will not be supported if *tcpdump* was built with an older version of *libpcap* that lacks the **pcap_findalldevs ()** function.

-e Print the link-level header on each dump line.

-E Use *spi@ipaddralgo:secret* for decrypting IPsec ESP packets that are addressed to *addr* and contain Security Parameter Index value *spi*. This combination may be repeated with comma or newline separation. Note that setting the secret for IPv4 ESP packets is supported at this time. Algorithms may be **des-cbc**, **3des-cbc**, **blowfish-cbc**, **rc3-cbc**, **cast128-cbc**, or **none**. The default is **des-cbc**. The ability to decrypt packets is only present if *tcpdump* was compiled with cryptography enabled. *secret* is the ASCII text for ESP secret key. If preceded by 0x, then a hex value will be read. The option assumes RFC2406 ESP, not RFC1827 ESP. The option is only for debugging purposes, and the use of this option with a true `secret' key is discouraged. By presenting IPsec secret key onto command line you make it visible to others, via **ps**(1) and other occasions. In addition to the above syntax, the syntax *file name* may be used to have tcpdump read the provided file in. The file is opened upon receiving the first ESP packet, so any special permissions that tcpdump may have been given should already have been given up.

-f Print `foreign' IPv4 addresses numerically rather than symbolically (this option is intended to get around serious brain damage in Sun's NIS server usually it hangs forever translating non-local internet numbers). The test for `foreign' IPv4 addresses is done using the IPv4 address and netmask of the interface on which capture is being done. If that address or netmask are not available, available, either because the interface on which capture is being done has no address or netmask or because the capture is being done on the Linux "any" interface, which can capture on more than one interface, this option will not work correctly.

-F Use *file* as input for the filter expression. An additional expression given on the command line is ignored.

-i Listen on *interface*. If unspecified, *tcpdump* searches the system interface list for the lowest numbered, configured up interface (excluding loopback). Ties are broken by choosing the earliest match. On Linux systems with 2.2 or later kernels, an *interface* argument of ``any'' can be used to capture packets from all interfaces. Note that captures on the ``any'' device will not be done in promiscuous mode. If the **-D** flag is supported, an interface number as printed by that flag can be used as the *interface* argument.

-l Make stdout line buffered. Useful if you want to see the data while capturing it. E.g.,``tcpdump -l | tee dat'' or ``tcpdump -l >dat& tail -f dat''.

-L List the known data link types for the interface and exit.

-m Load SMI MIB module definitions from file *module*. This option can be used several times to load several MIB modules into *tcpdump*.

-M Use *secret* as a shared secret for validating the digests found in TCP segments with the TCP-MD5 option (RFC 2385), if present.

-n Don't convert addresses (i.e., host addresses, port numbers, etc.) to names.

-N Don't print domain name qualification of host names. E.g., if you give this flag then *tcpdump* will print ``nic'' instead of ``nic.ddn.mil''.

-O Do not run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer.

-p *Don't* put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason; hence, `-p' cannot be used as an abbreviation for `ether host {local-hw-addr} or ether broadcast'.

-q Quick (quiet?) output. Print less protocol information so output lines are shorter.

-R Assume ESP/AH packets to be based on old specification (RFC1825 to RFC1829). If specified, *tcpdump* will not print replay prevention field. Since there is no protocol version field in ESP/AH specification, *tcpdump* cannot deduce the version of ESP/AH protocol.

-r Read packets from *file* (which was created with the **-w** option). Standard input is used if *file* is ``-''.

-S Print absolute, rather than relative, TCP sequence numbers.

-s Snarf*snaplen* bytes of data from each packet rather than the default of 68 (with SunOS's NIT, the minimum is actually 96). 68 bytes is adequate for IP, ICMP, TCP and UDP but may truncate protocol information from name server and NFS packets (see below). Packets truncated because of a limited snapshot are indicated in the output with ``[|*proto*]'', where *proto* is the name of the protocol level at which the truncation has occurred. Note that taking larger snapshots both increases the amount of time it takes to process packets and, effectively, decreases the amount of packet buffering. This may cause packets to be lost. You should limit *snaplen* to the smallest number that will capture the protocol information you're interested in. Setting *snaplen* to 0 means use the required length to catch whole packets.

-T Force packets selected by "*expression*" to be interpreted the specified *type*. Currently known types are **aodv** (Ad-hoc On-demand Distance Vector protocol), **cnfp** (Cisco NetFlow protocol), **rpc** (Remote Procedure Call), **rtp** (Real-Time Applications protocol), **rtcp** (Real-Time Applications control protocol), **snmp** (Simple Network Management Protocol), **tftp** (Trivial File Transfer Protocol), **vat** (Visual Audio Tool), and **wb** (distributed White Board).

-t *Don't* print a timestamp on each dump line.

-tt Print an unformatted timestamp on each dump line.

-ttt Print a delta (in micro-seconds) between current and previous line on each dump line.

-tttt Print a timestamp in default format proceeded by date on each dump line.

-u Print undecoded NFS handles.

-U Make output saved via the **-w** option ``packet-buffered"; i.e., as each packet is saved, it will be written to the output file, rather than being written only when the output buffer fills. The **-U** flag will not be supported if *tcpdump* was built with an older version of *libpcap* that lacks the **pcap_dump_flush ()** function.

-v When parsing and printing, produce (slightly more) verbose output. For example, the time to live, identification, total length and options in an IP packet are printed. Also enables additional packet integrity checks such as verifying the IP and ICMP header checksum. When writing to a file with the **-w** option, report, every 10 seconds, the number of packets captured.

-vv even more verbose output. For example, additional fields are printed from NFS reply packets, and SMB packets are fully decoded.

-vvv Even more verbose output. For example, telnet **SB** ... **SE** options are printed in full. With **-X** Telnet options are printed in hex as well.

-w Write the raw packets to *file* rather than parsing and printing them out. They can later be printed with the -r option. Standard output is used if *file* is ``-''.

-W Used in conjunction with the **-C** option, this will limit the number of files created to the specified number, and begin overwriting files from the beginning, thus creating a 'rotating' buffer. In addition, it will name the files with enough leading 0s to support the maximum number of files, allowing them to sort correctly.

-x When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex. The smaller of the entire packet or *snaplen* bytes will be printed. Note that this is the entire link-layer packet, so for link layers that pad (e.g. Ethernet), the padding bytes will also be printed when the higher layer packet is shorter than the required padding.

-xx When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, *including* its link level header, in hex.

-X When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex and ASCII. This is very handy for analysing new protocols.

-XX When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, *including* its link level header, in hex and ASCII.

-y Set the data link type to use while capturing packets to *datalinktype*.

-Z Drops privileges (if root) and changes user ID to *user* and the group ID to the primary group of *user*. This behavior can also be enabled by default at compile time.

About last three lines of when tcpdump stops:

- packets “captured'' (this is the number of packets that tcpdump has received and processed)

- packets ``received by filter'' (the meaning of this depends on the OS on which you're running tcpdump, and possibly on the way the OS was configured - if a filter was specified on the command line, on some OSes it counts packets regardless of whether they were matched by the filter expression and, even if they were matched by the filter expression, regardless of whether tcpdump has read and processed them yet, on other OSes it counts only packets that were matched by the filter expression regardless of whether tcpdump has read and processed them yet, and on other OSes it counts only packets that were matched by the filter expression and were processed by tcpdump).

  packets ``dropped by kernel'' (this is the number of packets that were dropped, due to a lack of buffer space, by the packet capture mechanism in the OS on which tcpdump is running, if the OS reports that information to applications; if not, it will be reported as 0).

3) Time to Live:

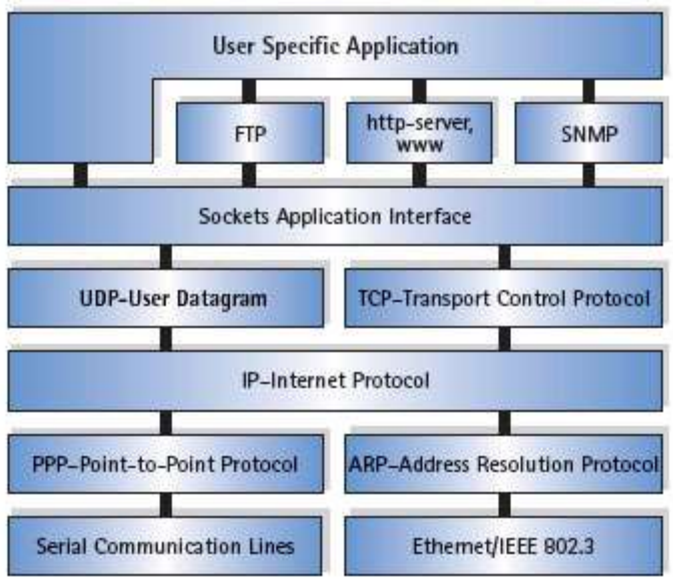
In IPv4, time to live (TTL) is an 8-bit field in the Internet Protocol (IP) header. It is the 9th octet of 20. The time to live value can be thought of as an upper bound on the time that an IP datagram can exist in an internet system. The TTL field is set by the sender of the datagram, and reduced by every host on the route to its destination. If the TTL field reaches zero before the datagram arrives at its destination, then the datagram is discarded and an ICMP error datagram (11 - Time Exceeded) is sent back to the sender. The purpose of the TTL field is to avoid a situation in which an undeliverable datagram keeps circulating on an internet system, and such a system eventually becoming swamped by such immortal datagram.

In theory, time to live is measured in seconds, although every host that passes the datagram must reduce the TTL by at least one unit. In practice, the TTL field is reduced by one on every hop. To reflect this practice, the field is named **hop limit** in IPv6.

4) SEQUENCE NUMBER:

A fundamental notion in the design is that every octet of data sent over a TCP connection has a sequence number. Since every octet is sequenced, each of them can be acknowledged. The acknowledgment mechanism employed is cumulative so that an acknowledgment of sequence number X indicates that all octets up to but not including X have been received.

5) TRANSMISSION CONTROL PROTOCOL (TCP):


User Specific Application
FTP
http-server, www
SNMP
Sockets Application Interface
UDP-User Datagram
TCP-Transport Control Protocol
IP-Internet Protocol
PPP-Point-to-Point Protocol
ARP-Address Resolution Protocol
Serial Communication Lines
Ethernet/IEEE 802.3

The **Transmission Control Protocol** (**TCP**) is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components, with Internet Protocol (IP), of the suite, so that the entire suite is commonly referred to as *TCP/IP*. Whereas IP handles lower-level transmissions from computer to computer as a message makes its way across the Internet, TCP operates at a higher level, concerned only with the two end systems, for example, a Web browser and a Web server. In particular, TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. Besides the Web, other common applications of TCP include e-mail and file transfer. Among its other management tasks, TCP controls message size, the rate at which messages are exchanged, and network traffic congestion.

6) About Encryption And Decryption:

When a firewall computer receives a network packet from the network, it determines whether the transmission is authorized. If so, the computer examines the header within the packet to determine what encryption algorithm was used to encrypt the packet. Using this algorithm and a secret key, the computer decrypts the data and addresses of the source and destination computers and sends the data to the destination computer. If both the source and destination computers are firewall computers, the only addresses visible (i.e., unencrypted) on the network are those of the firewall computers. The addresses of computers on the internal networks, and, hence, the internal network topology, are hidden. This has been termed "virtual private networking" (VPN).

More about TCPDUMP -d Option:

As specified in manual -d option of TCPDUMP dump the compiled packets in human readable form and then stops, but it actually converts compiled code into assembly language that is more human readable in comparison to machine language code. Load is denoted using **ldh,** Comparison is made using **jeq,**If true jumps shown by **jt,** If false jump shown by **jf,**

0x800 ethertype of IPV4**, 0x86dd** ethertype of IPV6.

(007) ret #96 is equivalent to TRUE i.e. 96 bytes of packet data will be copied to calling application.

(008) ret #0 is equivalent of FALSE i.e. Zero bytes of packet data will be copied to application.

Appendix B

third.sh

```
#!/bin/sh


/usr/sbin/tcpdump -i wlo1 -ne -c 1000  > packets

grep -i "tcp" packets > tcppackets
grep -i "arp" packets > arppackets
grep -i "udp" packets > udppackets

#timestamp

cut -d " " -f 1 tcppackets>timestamptcp
cut -d " " -f 1 arppackets>timestamparp
cut -d " " -f 1 udppackets>timestampudp


#smac


cut -d " " -f 2 tcppackets>smactcp
cut -d " " -f 2 arppackets>smacarp
cut -d " " -f 2 udppackets>smacudp
```

```
#dmac

cut -d " " -f 4 tcppackets | cut -d "," -f 1 >dmactcp

cut -d " " -f 4 udppackets | cut -d "," -f 1 >dmacudp

cut -d " " -f 4 arppackets | cut -d "," -f 1 >dmacarp

#dip

cut -d " " -f 12 tcppackets | cut -d ":" -f 1,2,3,4 >diptcp
cut -d " " -f 12 udppackets | cut -d ":" -f 1 |cut -d "." -f 1,2,3,4>dipudp
cut -d " " -f 14 arppackets | cut -d "," -f1 >diparp

#sip
cut -d " " -f 10 tcppackets>siptcp
cut -d " " -f 10 udppackets|cut -d "." -f 1,2,3,4>sipudp
cut -d " " -f 12 arppackets | sed 's/is-at//g' >siparp

#length

cut -d " " -f 9 tcppackets | cut -d ":" -f1 >lengthtcp

#alternate

#cut -d " " -f16  arppackets >lengtharp

cut -d " " -f 9 arppackets | cut -d ":" -f1 >lengtharp

cut -b88-89 udppackets>lengthudp


#dport

cut -d " " -f 12 tcppackets | cut -d "." -f 5 | cut -d ":" -f1 >dporttcp

#d
cut -d " " -f 12 udppackets| cut -d "." -f 5 | cut -d ":" -f1 >dportudp
#sport

cut -d " " -f10 tcppackets | cut -d "." -f5 >sporttcp
cut -d " " -f10 udppackets | cut -d "." -f5 >sportudp
```

For Tabular Representation

```
service mysqld start

mysql -u root -e"

drop database if exists dump;

create database dump;

use dump;

drop table if exists arp;

drop table if exists ip;

drop table if exists udp;

drop table if exists all_packets;


create table arp(time varchar(20),src_mac varchar(25),dest_mac varchar(25),src_ip varchar(25),dest_ip varchar(25),length varchar(20));


create table ip(time varchar(20),src_mac varchar(25),dest_mac varchar(25),src_ip varchar(25),dest_ip varchar(25),length varchar(20),src_port varchar(20),dest_port varchar(20));


create table udp(time varchar(20),src_mac varchar(25),dest_mac varchar(25),src_ip varchar(25),dest_ip varchar(25),length varchar(20),src_port varchar(20), dest_port varchar(20));


create table all_packets(time varchar(20),src_mac varchar(25),dest_mac varchar(25),src_ip varchar(25),dest_ip varchar(25),length varchar(20),src_port varchar(20), dest_port varchar(20));
```

```
LOAD DATA LOCAL INFILE 'finalip' INTO TABLE ip;

LOAD DATA LOCAL INFILE 'finaludp' INTO TABLE udp;

LOAD DATA LOCAL INFILE 'finalarp' INTO TABLE arp;

LOAD DATA LOCAL INFILE 'all' INTO TABLE all_packets;";
```

Login.php

```
<?php
ini_set('display_errors','1');
$host="localhost";
$user="phpmyadmin";
$password="Hstqwerty@05";
$db="demo";

$cont=mysqli_connect($host,$user,$password);
mysqli_select_db($cont,$db);

/*
mysqli_connect($host,$user,$password);
mysqli_select_db($db);
*/

if(isset($_POST['username'])){

   $uname=$_POST['username'];
   $password=$_POST['password'];

   $sql="select * from loginform where user='".$uname."'AND
Pass='".$password."' limit 1";

   $result=mysqli_query($cont,$sql);

   if(mysqli_num_rows($result)==1){
     //  echo " You Have Successfully Logged in";
        //echo shell_exec('sudo tcpdump -i any  -c100 -n -v');
        //       echo $ot;
      //echo $fil;

system("/var/www/html/third.sh");
        header("Location: redirect.php");
        exit();
```

```
    }
    else{
        echo " You Have Entered Incorrect Password";
        exit();
    }

}
?>
<!DOCTYPE html>
<html>
        <title>NMT Login Portal</title>
        <link rel="stylesheet" type="text/css" href="style.css">
        <body>
                <div class="loginbox">
                <img src="loginbot.png" class="avatar">
                        <h1>Login Here</h1>
                        <form id = "login" method="POST" action="#">
                                <p>User Name</p>
                                <input id = "inputUsername" value=" type="text"
name="username" placeholder="Enter Username">
                                <p>Password</p>
                                <input id = "inputPassword" value=""
type="password" name="password" placeholder="Enter Password"><br>
                                <input type = "submit"  value="Login
"onclick="loginOnClick()" ><br><br>
                                </form>
                <div>
        </body>
</html>
```

Style.css

```
body{
	margin:0;
	padding:0;
	background:url("h1.png");
	background-size:cover;
	background-position:center:
	font-family:sans-serif;
}

.loginbox{
	width:320px;
	height:420px;
	background:#000;
	color:#fff;
	top:50%;
	left:50%;
	position:absolute;
	transform:translate(-50%,-50%);
	box-sizing:border-box;
	padding: 70px 30px;
}

.box{
	width:320px;
	height:420px;
	background:#000;
	color:#fff;
	top:50%;
	left:50%;
```

```
        position:absolute;
        transform:translate(-50%,-50%);
        box-sizing:border-box;
        padding: 70px 30px;
}

.avatar{
        width:50px;
        height:50px;
        top:20px;
        left:130px;
        position:absolute;
        border-radius:50%;
        box-sizing:border-box;
}

h1{
        margin:0px;
        padding:0 0 20px;
        text-align:center;
        font-size:22px;
        font-family:monospace;

}

.loginbox p{
        margin:0;
        padding:0;

        font-weight:bold;
}
```

```
.loginbox input{
        width:100%;
        margin-bottom:20px;
}
.box input{
        width:30%;
        margin-bottom:20px;
}

.loginbox input[type="text"],input[type="password"]{
        width:180px;
        border:none;
        background:transparent;
        outline:none;
        color:#fff;
        font-size:16px;
}

.loginbox input[type="submit"]{
        width:150px;
        border:none;
        background:#fb2525;
        outline:none;
        color:#fff;
        font-size:18px;
        border-radius:20px;
}


.loginbox input[type="submit"]:hover{
        cursor:pointer;
        background:#ffc107;
        color:#000;
}


.loginbox input[type="button"]{
        width:150px;
        border:none;
        background:#fb2525;
        outline:none;
        color:#fff;
        font-size:18px;
        border-radius:20px;
}



.loginbox input[type="button"]:hover{
        cursor:pointer;
        background:#ffc107;
        color:#000;
}
```

Redirect.php

```
<?php
ini_set('display_errors','1');
if(isset($_POST['r1']) && isset($_POST['dropdown'])){
$rad = $_POST["r1"];
$sel = $_POST["dropdown"];

        if($rad=="arp"){
                if($sel=="Timestamp"){
                echo "Arp Packets";
                echo "Timestamp";
                header("Location: timestamparp");

                }
                else if($sel=="SIP"){
                echo "Arp Packets";
                echo "SIP";
                header("Location: siparp");
                }
                else if($sel=="DIP"){
                echo "Arp Packets";
                echo "DIP 0 byte file";
                //header("Location: diparp");
                }
                else if($sel=="DPort"){
                echo "Arp Packets";
                echo "DPort not possible";
                //header("Location: dportarp");
                }
                else if($sel=="SPort"){
                echo "Arp Packets";
                echo "SPort";
                //header("Location: sportarp");

                }
                else if($sel=="SMAC"){
                echo "Arp Packets";
                echo "SMAC";
                header("Location: smacarp");
                }
                else if($sel=="DMAC"){
                echo "Arp Packets";
                echo "DMAC";
```

```
		header("Location: dmacarp");
		}
		else if($sel=="Plength"){
		echo "Arp Packets";
		echo "Plength";
		header("Location: lengtharp");
		}
		else if($sel=="All"){
		echo "Arp Packets";
		echo "Detailed View";
		header("Location: arppackets");
		}


	}
	else if($rad=="udp"){
		if($sel=="Timestamp"){
		echo "UDP Packets";
		echo "Timestamp";
		header("Location: timestampudp");
		}
		else if($sel=="SIP"){
		echo "UDP Packets";
		echo "SIP";
		header("Location: sipudp");
		}
		else if($sel=="DIP"){
		echo "UDP Packets";
		echo "DIP";
		header("Location: dipudp");
		}
		else if($sel=="DPort"){
		echo "UDP Packets";
		echo "DPort";
		header("Location: dportudp");
		}
		else if($sel=="SPort"){
		echo "UDP Packets";
		echo "SPort";
		//header("Location: sportudp");
		}
		else if($sel=="SMAC"){
		echo "UDP Packets";
		echo "SMAC";
		header("Location: smacudp");
		}
```

```
        else if($sel=="DMAC"){
        echo "UDP Packets";
        echo "DMAC";
        header("Location: dmacudp");
        }
        else if($sel=="Plength"){
        echo "UDP Packets";
        echo "Plength";
        header("Location: lengthudp");
        }
        else if($sel=="All"){
        echo "UDP Packets";
        echo "Detailed View";
        header("Location: udppackets");
        }


}
else if($rad=="tcp"){
        if($sel=="Timestamp"){
        echo "TCP Packets";
        echo "Timestamp";
        header("Location: timestamparp");

        }
        else if($sel=="SIP"){
        echo "TCP Packets";
        echo "SIP";
        header("Location: siptcp");
        }
        else if($sel=="DIP"){
        echo "TCP Packets";
        echo "DIP";
        header("Location: diptcp");
        }
        else if($sel=="DPort"){
        echo "TCP Packets";
        echo "DPort";
        header("Location: dporttcp");
        }
        else if($sel=="SPort"){
        echo "TCP Packets";
        echo "SPort";
        //header("Location: sporttcp");
        }
        else if($sel=="SMAC"){
```

```
			echo "TCP Packets";
			echo "SMAC";
			header("Location: smactcp");
			}
			else if($sel=="DMAC"){
			echo "TCP Packets";
			echo "DMAC";
			header("Location: dmactcp");
			}
			else if($sel=="Plength"){
			echo "TCP Packets";
			echo "Plength";
			header("Location: lengthtcp");
			}
			else if($sel=="All"){
			echo "TCP Packets";
			echo "Detailed View";
			header("Location: tcppackets");
			}


		}
	}
	?>
	<!DOCTYPE html>
	<html>
		<title>Choice of packets</title>
		<link rel="stylesheet" type="text/css" href="style.css">
		<body>
			<div class="box">

				<h1>Choose the type of packets you need</h1>
				<form id = "login" method="POST" action="#">

					<input  value="arp" type="radio" name="r1">Arp<br>
					<input  value="tcp" type="radio" name="r1">Tcp<br>
					<input  value="udp" type="radio" name="r1">Udp<br>

	<p><select name="dropdown">
		<option value="Timestamp">Timestamp</option>
		<option value="SIP">Source IP</option>

		<option value="SMAC">Source MAC</option>
		<option value="DIP">Destination IP</option>
		<option value="SPort 0">Source Port</option>
		<option value="DPort">Destination Port</option>
```

```
            <option value="DMAC">Destination MAC</option>
            <option value="Plength">Plength</option>
            <option value="All">all</option>


</select>
                                                <input type = "submit"  value="Submit "onclick="loginOnClick()"
><br><br>
                                                </form>
                        <div>
            </body>
</html>
```