**Introduction**

Network monitoring is a computer network's systematic effort to detect slow or failing network components, such as overloaded or crashed/frozen servers, failing routers, failed switches or other problematic devices. In the event of a network failure or similar outage, the network monitoring system alerts the network administrator (NA). Network monitoring is a subset of network management.

Network monitoring is generally carried out through software applications and tools. Network monitoring services are widely used to detect whether a given Web server is functioning and connected properly to networks worldwide. Many servers that perform this function provide a more complete visualization of both the Internet and networks.

## While constantly monitoring the health/reliability of a network and searching for trends, the monitoring system tracks and logs network parameters. These include data transmission rate (throughput), error rates, downtime/uptime, use-time percentages, and response time to user and automated inputs and requests. When predetermined parameter thresholds are reached, alarms are triggered, and network fault management processes are initiated.

In the following project we have developed a packet analyser tool which is used to capture the packets and provide filtered information about the packets. A packet analyser (also known as packet sniffer) is a computer program or piece of computer hardware that can intercept and log traffic that passes over a digital network or a part of network. Packet capture is a process of intercepting and logging traffic. As data streams flow across the network, the sniffer captures each packet, and if needed, decodes the packet’s raw data, showing the values of various fields in the packets, and analyses its content according to the appropriates RFC or other specifications.

A packet analyzer used for intercepting traffic on wireless networks is known as a wireless analyzer or Wi-Fi analyzer. A packet analyzer can also be referred to as a network analyzer or protocol analyzer through these terms also having other meanings.

**Why packet Analyzers are used?**

Packet sniffers are used for both legal and illegal activity. A legal packet sniffer is a commercial device used to assist with network management and maintenance and to provide network security. It is also used as a diagnostic tool for network backup systems and to examine the network system for any security breaches.

An illegal packet sniffer is used by a hacker to gain unauthorized access to sensitive information and data on a network. An illegal packet sniffer is installed without the knowledge of the IT administrator and hides in different areas of the network for the purpose of spying on and stealing the information packets that pass over the network.

**Who all uses packet analyzer?**

When you connect to the Internet, you are joining a network maintained by your **Internet service provider** (ISP). The ISP's network communicates with networks maintained by other ISPs to form the [foundation of the Internet](https://computer.howstuffworks.com/internet-infrastructure.htm). A packet sniffer located at one of the servers of your ISP would potentially be able to monitor all your online activities, such as:

* Which Web sites you visit
* What you look at on the site
* Whom you send e-mail to
* What's in the e-mail you send
* What you download from a site
* What streaming events you use, such as audio, video and [Internet telephony](https://computer.howstuffworks.com/ip-telephony.htm)

Advertising agencies or internet advertising agencies are paid according to:

* Number of ads shown by them
* Number of clicks on their ads also called PPC

Government Agencies uses packet sniffing

* Ensure security of data over the internet
* Track an organisation’s & unsecured data

**Project Goals**

Major objective of the project was to develop a packet analyzer tool which can analyse LAN network packets. There is different type of internet packets such as TCP (Transmission Control Protocol), UDP (User Datagram Protocol), ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol), etc. In this project we are analysing only **TCP, UDP, ARP** packets. There is variety of information inside packet such as Flags, Sequence Number, Packet Length, Acknowledgement, Source Mac Address, Destination Mac Address, Source IP Address, Destination IP Address, Source Port, etc. A packet analyzer used for **intercepting** and logging traffic on wireless network are known as wireless analyzer or Wi-Fi analyzer.

We have developed a webpage which is used to create systematic table for different type of packets captured from LAN network. Shell Script plays an important role in background to filter the packets according to the user requirement. I have implemented PHP for server-side scripting language.

**Software Used**

**Bundled Software:** Bundled software can be either a set of single software programs that are sold together, or one or more software programs sold together with a piece of hardware. Common types of bundled software include operating systems, utilities and accessories sold with desktop or laptop computers, as well as mobile devices. Other types of bundled software are multiple programs sold as a single software service or product that can provide more than one use.

Bundled Software **LAMP** is being used to develop the project.

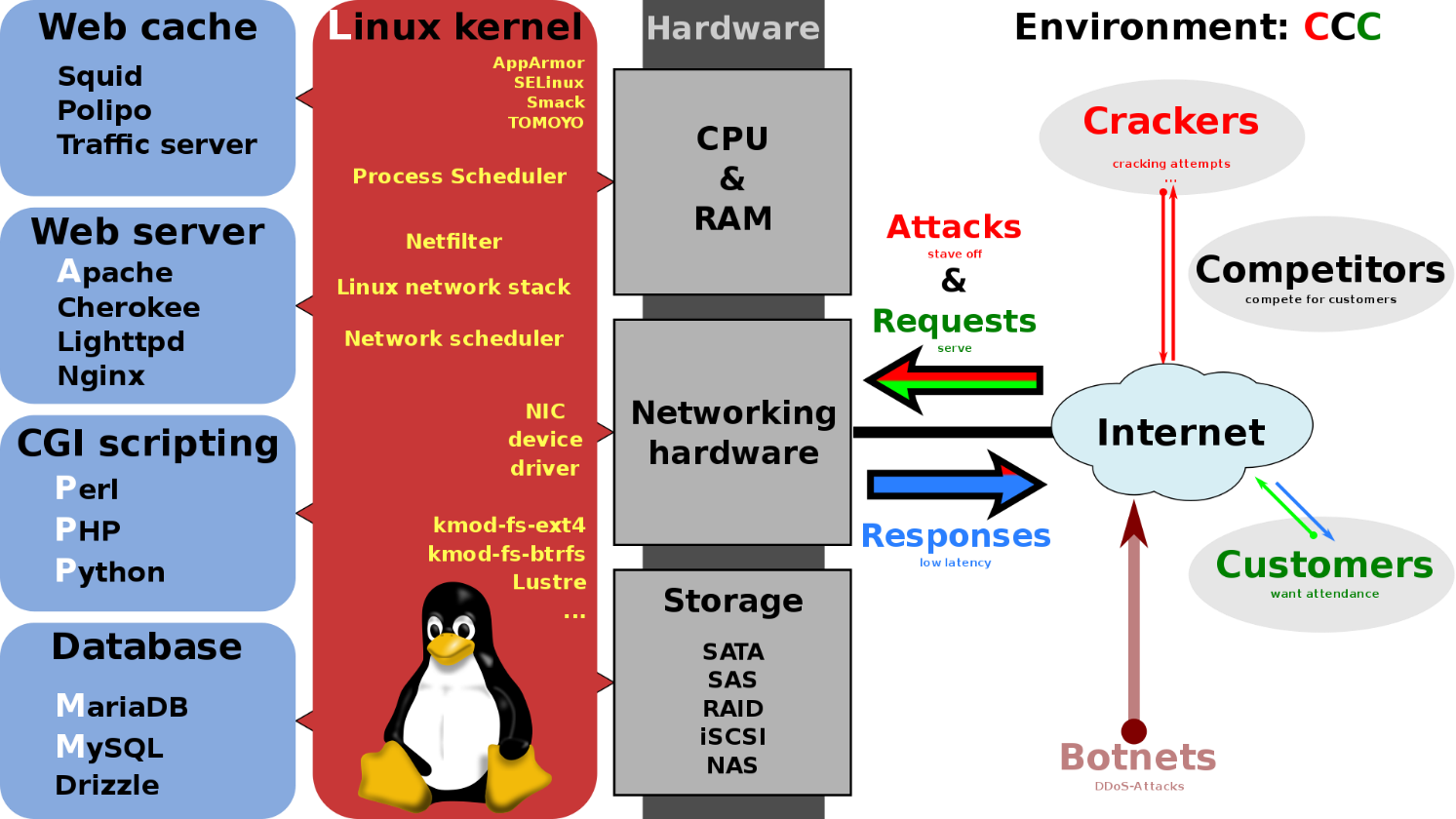
**LAMP** is an archetypal model of web service [stacks](https://en.wikipedia.org/wiki/Solution_stack), named as an [acronym](https://en.wikipedia.org/wiki/Acronym) of the names of its original four [open-source](https://en.wikipedia.org/wiki/Open-source_software) components: the [Linux](https://en.wikipedia.org/wiki/Linux) [operating system](https://en.wikipedia.org/wiki/Operating_system), the [Apache HTTP Server](https://en.wikipedia.org/wiki/Apache_HTTP_Server), the [MySQL](https://en.wikipedia.org/wiki/MySQL) relational database management system (RDBMS), and the [PHP](https://en.wikipedia.org/wiki/PHP) programming language. The LAMP components are largely interchangeable and not limited to the original selection. As a solution stack, LAMP is suitable for building [dynamic web sites](https://en.wikipedia.org/wiki/Dynamic_web_site) and [web applications](https://en.wikipedia.org/wiki/Web_application).

**L Linux** (OS)

**A Apache** (http server)

**M MySQL** (RDBMS)

**P PHP** (server-side Scripting Language)



**Figure 1: A high-level overview of LAMP's building blocks and overall system environment**

**LINUX:** Just like Windows XP, Windows 7, Windows 8, and Mac OS X, Linux is an operating system. An operating system is software that manages all the hardware resources associated with your desktop or laptop. To put it simply – the operating system manages the communication between your software and your hardware. Without the operating system (often referred to as the “OS”), the software wouldn’t function.

The most popular Linux distributions are:

* Ubuntu Linux
* Linux Mint
* Arch Linux
* Deepin
* [Fedora](http://fedoraproject.org/)
* Debian
* [openSUSE](http://www.opensuse.org/en/)

Ubuntu is a free and open-source Linux distribution based on Debian. Ubuntu is officially released in three editions: *Desktop*, *Server*, and *Core*[(for internet of things devices and robots). All the editions can run on the computer alone, or e.g. in Windows. Ubuntu is a popular operating system for cloud computing, with support for OpenStack.](https://en.wikipedia.org/wiki/Ubuntu#cite_note-12)

Ubuntu is released every six months, with long-term support (LTS) releases every two years. The latest release is 19.04 ("Disco Dingo"), and the most recent long-term support release is 18.04 LTS ("Bionic Beaver"), which is supported until 2023 under public support and until 2028 as a paid option.

**Ubuntu version 18.04.2 LTS is being used in the project to develop it.**

**Apache:** Apache Web Server is an open-source web server creation, deployment and management software. Initially developed by a group of software programmers, it is now maintained by the Apache Software Foundation. Apache is the most commonly used Web server on Linux systems. Web servers are used to serve Web pages requested by client computers. Clients typically request and view Web pages using Web browser applications such as *Firefox*, *Opera*, *Chromium*, or *Internet Explorer*.

Users enter a Uniform Resource Locator (URL) to point to a Web server by means of its Fully Qualified Domain Name (FQDN) and a path to the required resource.

The most common protocol used to transfer Web pages is the Hyper Text Transfer Protocol (HTTP). Protocols such as Hyper Text Transfer Protocol over Secure Sockets Layer (HTTPS), and File Transfer Protocol (FTP), a protocol for uploading and downloading files, are also supported.

Apache Web Server is designed to create web servers that have the ability to host one or more HTTP-based websites. Notable features include the ability to support multiple programming languages, server-side scripting, an authentication mechanism and database support. Apache Web Server can be enhanced by manipulating the code base or adding multiple extensions/add-ons.

Apache Web Servers are often used in combination with the *MySQL* database engine, the HyperText Preprocessor (*PHP*) scripting language, and other popular scripting languages such as *Python* and *Perl*. This configuration is termed LAMP (Linux, Apache, MySQL and Perl/Python/PHP) and forms a powerful and robust platform for the development and deployment of Web-based applications.

**MySQL:** MySQL was a free-software database engine originally developed and first released in 1995. It was originally produced under the GNU General Public License, in which source code is made freely available.

MySQL is very popular for Web-hosting applications because of its plethora of Web-optimized features like HTML data types, and because it's available for free. It is part of the Linux, Apache, MySQL, PHP (LAMP) architecture, a combination of platforms that is frequently used to deliver and support advanced Web applications. MySQL runs the back-end databases of some famous websites, including Wikipedia, Google and Facebook- a testament to its stability and robustness despite its decentralized, free-for-all philosophy.

**PHP:** The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web-based software applications.

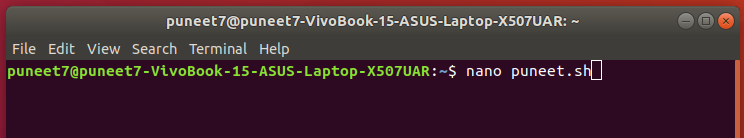
PHP code may be executed with a command line interface (CLI), embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in a web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control.

**After the installation of LAMP next step is to capture the packets from the LAN, here shell script come into play. There are certain command which are used to capture the packets.**

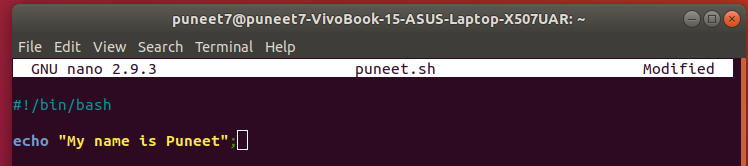
**Shell Scripting**

A shell script is a text [file](https://whatis.techtarget.com/definition/file) that contains a sequence of commands for a [UNIX](https://searchdatacenter.techtarget.com/definition/Unix)-based [operating system](https://whatis.techtarget.com/definition/operating-system-OS). It's called a shell script because it combines into a "script" in a single file a sequence of commands that would otherwise have to be presented to the system from a keyboard one at a time. The [shell](https://searchdatacenter.techtarget.com/definition/shell) is the operating system's command interpreter and the set of commands you use to communicate with the system. A shell script is usually created for command sequences for which a user has a repeated need. You initiate the sequence of commands in the shell script by simply entering the name of the shell script on a command line.

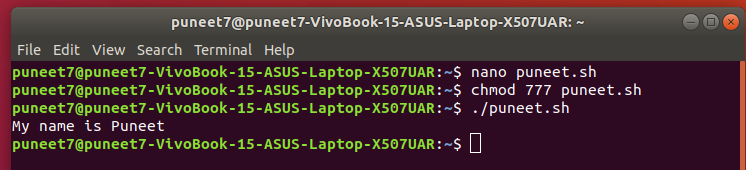
Ubuntu, Linux Mint or any other Linux distribution, are interacting to shell every time you use terminal. Each shell script is saved with **.sh** file extension example puneet.sh



**Figure 2 Creating Shell Script File in Command Prompt**



**Figure 3 Writing Command in Shell Script File**



**Figure 4 Giving Permission to file & executing shell script in Command Prompt**

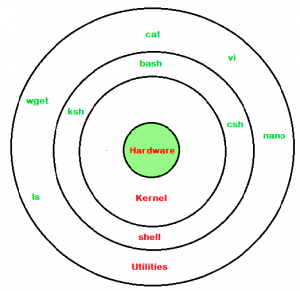
**What is Kernel ?**

The kernel is a computer program that is the core of a computer’s operating system, with complete control over everything in the system. It manages following resources of the Linux system –

* + File management
  + Process management
  + I/O management
  + Memory management
  + Device management etc.

**What is Shell ?**

A shell is special user program which provide an interface to user to use operating system services. Shell accept human readable commands from user and convert them into something which kernel can understand. It is a command language interpreter that execute commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or start the terminal.



**Figure 5 Linux shell**

**Command Line Shell:**

Shell can be accessed by user using a command line interface. A special program called Terminal in linux/macOS or Command Prompt in Windows OS is provided to type in the human readable commands such as “cat”, “ls” etc. and then it is being execute. The result is then displayed on the terminal to the user.

**To filter the data from file following commands are used:**

**Cut:** The cut command is for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field. Basically the cut command slices a line and extracts the text. It is necessary to specify option with command otherwise it gives error. If more than one file name is provided then data from each file is not precedes by its file name.

It cuts the file column wise.

**Syntax** cut OPTION... [FILE]...

$cut -d "delimiter" -f (field number) file.txt

where,

-d option is used then it considered space as a field separator

-f field to be extracted from file

**Grep:** The grep command is used to search text file for patterns. A pattern can be a word, text, numbers and more. It is one of the most useful commands on Ubuntu/ Linux like operating systems.

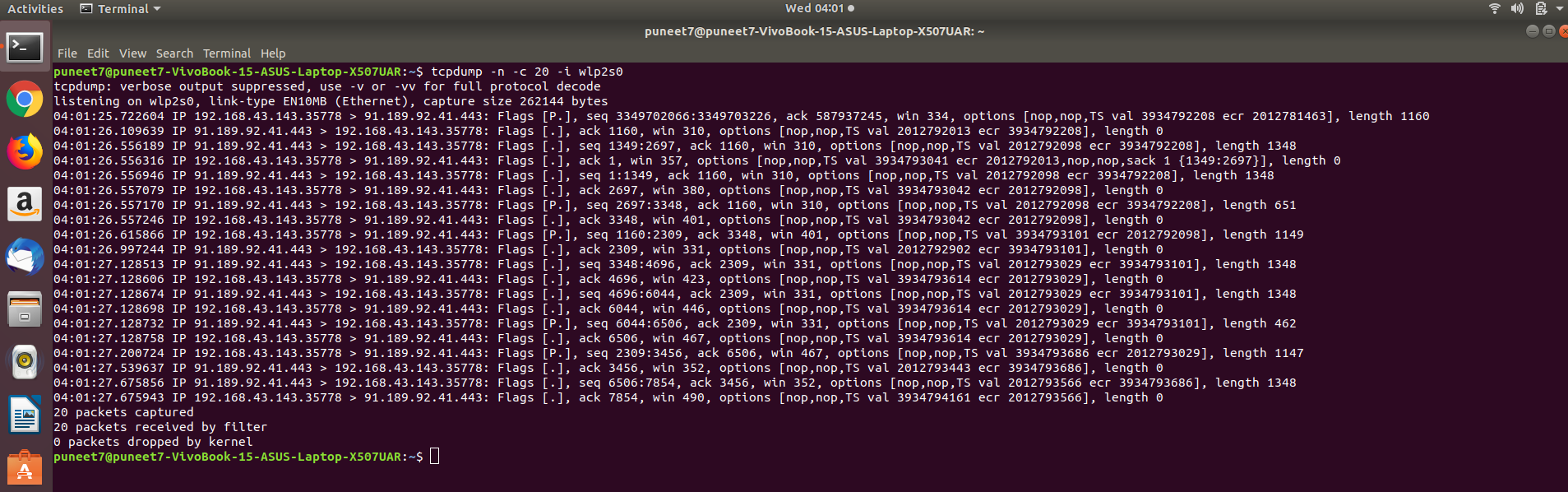
**Syntax**: grep [options] pattern {files}

**TCPdump**

Tcpdump is an open source command-line tool for monitoring (sniffing) network traffic. TCPdump works by capturing and displaying [packet](https://searchnetworking.techtarget.com/definition/packet) [header](https://whatis.techtarget.com/definition/header)s and matching them against a set of criteria. It understands boolean search operators and can use host names, [IP address](https://searchwindevelopment.techtarget.com/definition/IP-address)es, network names, and protocols as arguments.

It prints out a description of the contents of packets on a network interface that match the boolean expression, the description is preceded by a timestamp, printed, by the default, as hours, minutes, seconds, and fractions of a second.

**Using wlp2s0 Network Interface**

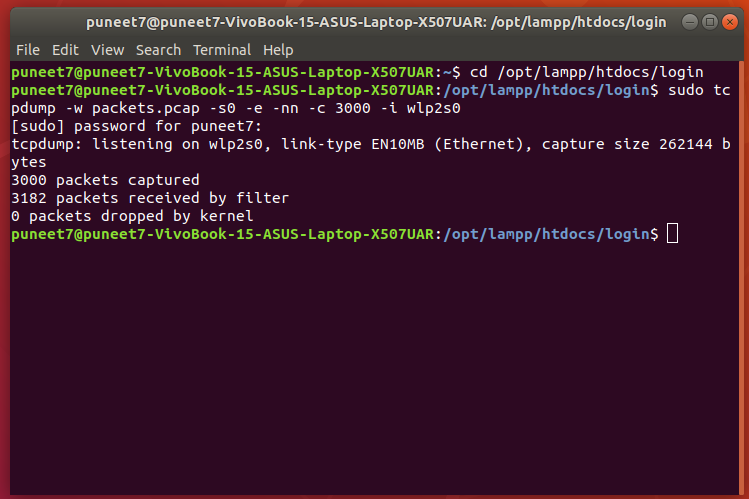


**Figure 6 Packet captured using Tcpdump**

* **Install TCPdump on Ubuntu:** apt-get install tcpdump
* **Display available interfaces:** tcpdump -D

### **Capture packets from a specific interface:** tcpdump -i wlp2s0

* **Capture only specific number of packets:** tcpdump -i wlp2s0 -c 20
* **Capture and save packets in a file:** tcpdump -w cap.pcap -i wlp2s0
* **Capture IP address packets:** tcpdump -n -i wlp2s0
* **Capture only TCP packets:** tcpdump -i wlp2s0 -c 20 -w tcp.pcap tcp
* **Capture only UDP packets:** tcpdump -i wlp2s0 -c 20 -w udp.pcap udp
* **Capture only ARP packets:** tcpdump -i wlp2s0-c 20 -w arp.pcap arp



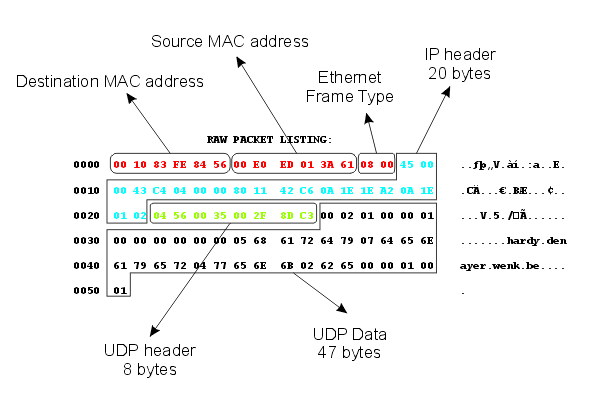
**Figure 7 Show that 3000 packets captured & stored in packets.pcap file**

**About Packets**

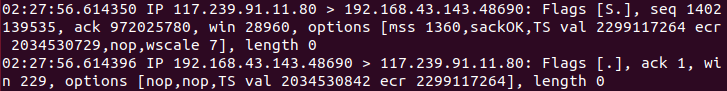
**Packets:** A packet is the unit of data that is routed between an origin and a destination on the Internet or any other packet-switched network.

**TCP/IP:** Transmission Control Protocol/Internet Protocol (TCP/IP) is the language a computer uses to access the internet. It consists of a suite of protocols designed to establish a network of networks to provide a host with access to the internet.

TCP/IP is responsible for full-fledged data connectivity and transmitting the data end to end by providing other functions, including addressing, mapping and acknowledgment. TCP/IP contains four layers, which differ slightly from the OSI model.



**Figure 8 TCP/IP Data Packet**



**Figure 9 Showing captured headers fields of TCP/IP example using TCPdump in command line**



**Figure 10 Wireshark showing TCP/IP Headers**

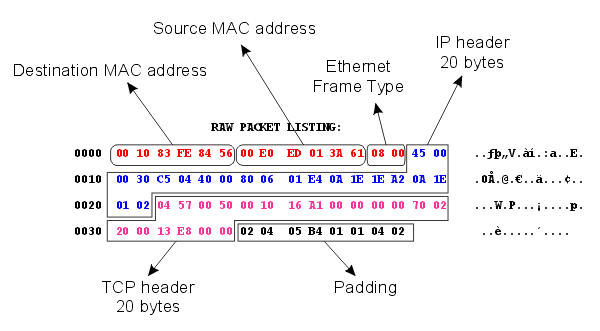
**Explanation of TCP Header Fields:**

* **Source Port Address –** 16 bit field that holds the port address of the application that is sending the data segment.
* **Destination Port Address –** 16 bit field that holds the port address of the application in the host that is receiving the data segment.
* **Sequence Number –** 32 bit field that holds the sequence number, i.e, the byte number of the first byte that is sent in that particular segment. It is used to reassemble the message at the receiving end if the segments are received out of order.
* **Acknowledgement Number –** 32 bit field that holds the acknowledgement number, i.e, the byte number that the receiver expects to receive next. It is an acknowledgment for the previous bytes being received successfully.
* **Header Length (HLEN) –** This is a 4 bit field that indicates the length of the TCP header by number of 4-byte words in the header, i.e, if the header is of 20 bytes(min length of TCP header), then this field will hold 5 (because 5 x 4 = 20) and the maximum length: 60 bytes, then it’ll hold the value 15(because 15 x 4 = 60). Hence, the value of this field is always between 5 and 15.
* **Control flags –** These are 6 1-bit control bits that control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc. Their function is:
* **URG:** Urgent pointer is valid
* **ACK:** Acknowledgement number is valid (used in case of cumulative acknowledgement)
* **PSH:** Request for push
* **RST:** Reset the connection
* **SYN:** Synchronize sequence numbers
* **FIN:** Terminate the connection
* **Window size –** This field tells the window size of the sending TCP in bytes.
* **Checksum –** This field holds the checksum for error control. It is mandatory in TCP as opposed to UDP.
* **Urgent pointer –** This field (valid only if the URG control flag is set) is used to point to data that is urgently required that needs to reach the receiving process at the earliest. The value of this field is added to the sequence number to get the byte number of the last urgent byte.

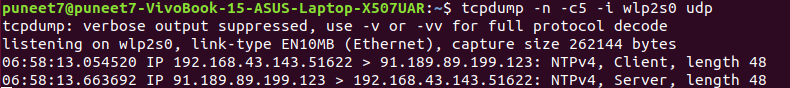
**UDP:** User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of Internet Protocol suite, referred as UDP/IP suite. Unlike TCP, it is unreliable and connectionless protocol. So, there is no need to establish connection prior to data transfer.

Though Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of Internet services; provides assured delivery, reliability and much more but all these services cost us with additional overhead and latency. Here, UDP comes into picture. For the realtime services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also save bandwidth.

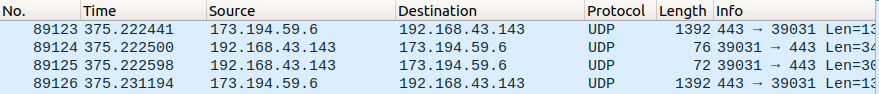
User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.



**Figure 11 UDP Data Packet**



**Figure 12Showing captured headers fields of UDP example using TCPdump in command line**



**Figure 13 Wireshark showing UDP Headers**

**Explanation of TCP Header Fields:**

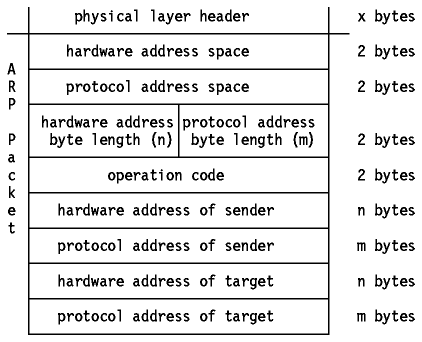
* **Source Port :** Source Port is 2 Byte long field used to identify port number of source.
* **Destination Port :** It is 2 Byte long field, used to identify the port of destined packet.
* **Length :** Length is the length of UDP including header and the data. It is 16-bits field.
* **Checksum :** Checksum is 2 Bytes long field. It is the 16-bit one’s complement of the one’s complement sum of the UDP header, pseudo header of information from the IP header and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

**ARP:** The address resolution protocol (arp) is a protocol used by the Internet Protocol (IP) [RFC826], specifically IPv4, to map IP network addresses to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when IPv4 is used over Ethernet.

The term address resolution refers to the process of finding an address of a computer in a network. The address is "resolved" using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer. The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore to provide the required address. The address resolution procedure is completed when the client receives a response from the server containing the required address.

An Ethernet network uses two hardware addresses which identify the source and destination of each frame sent by the Ethernet. The destination address (all 1's) may also identify a broadcast packet (to be sent to all connected computers). The hardware address is also known as the Medium Access Control (MAC) address, in reference to the standards which define Ethernet. Each computer network interface card is allocated a globally unique 6 byte link address when the factory manufactures the card (stored in a PROM). This is the normal link source address used by an interface. A computer sends all packets which it creates with its own hardware source link address, and receives all packets which match the same hardware address in the destination field or one (or more) pre-selected broadcast/multicast addresses.

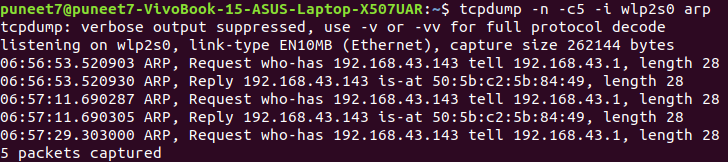
The Ethernet address is a link layer address and is dependent on the interface card which is used. IP operates at the network layer and is not concerned with the link addresses of individual nodes which are to be used.The address resolution protocol (arp) is therefore used to translate between the two types of address. The arp client and server processes operate on all computers using IP over Ethernet. The processes are normally implemented as part of the software driver that drives the network interface card.



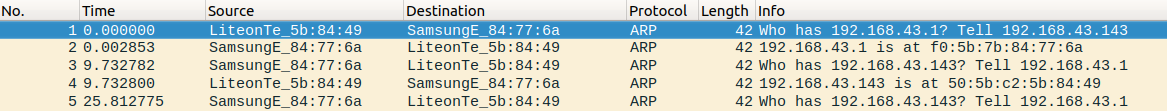
**Figure 14 ARP Packet Format**

There are four types of arp messages that may be sent by the arp protocol. These are identified by four values in the "operation" field of an arp message. The types of message are:

* ARP request
* ARP reply
* RARP request
* RARP reply



**Figure 15 Showing captured headers fields of ARP example using TCPdump in command line**



**Figure 16 Wireshark showing ARP Headers**

**Command Used (command line)**

**To install LAMP**

**Step 1: Update Software Packages**

sudo apt update

**Step 2: Install Apache Web Server**

sudo apt install apache2

**Check if Apache web server is running or not:**

sudo systemctl status apache2

**Step 3: Install MySQL Database Server**

sudo apt install mysql-server

**Verify if MySQL service is running or not using command:**

sudo systemctl status mysql

**Setup Database Administrative User (Root) Password**

sudo mysql\_secure\_installation

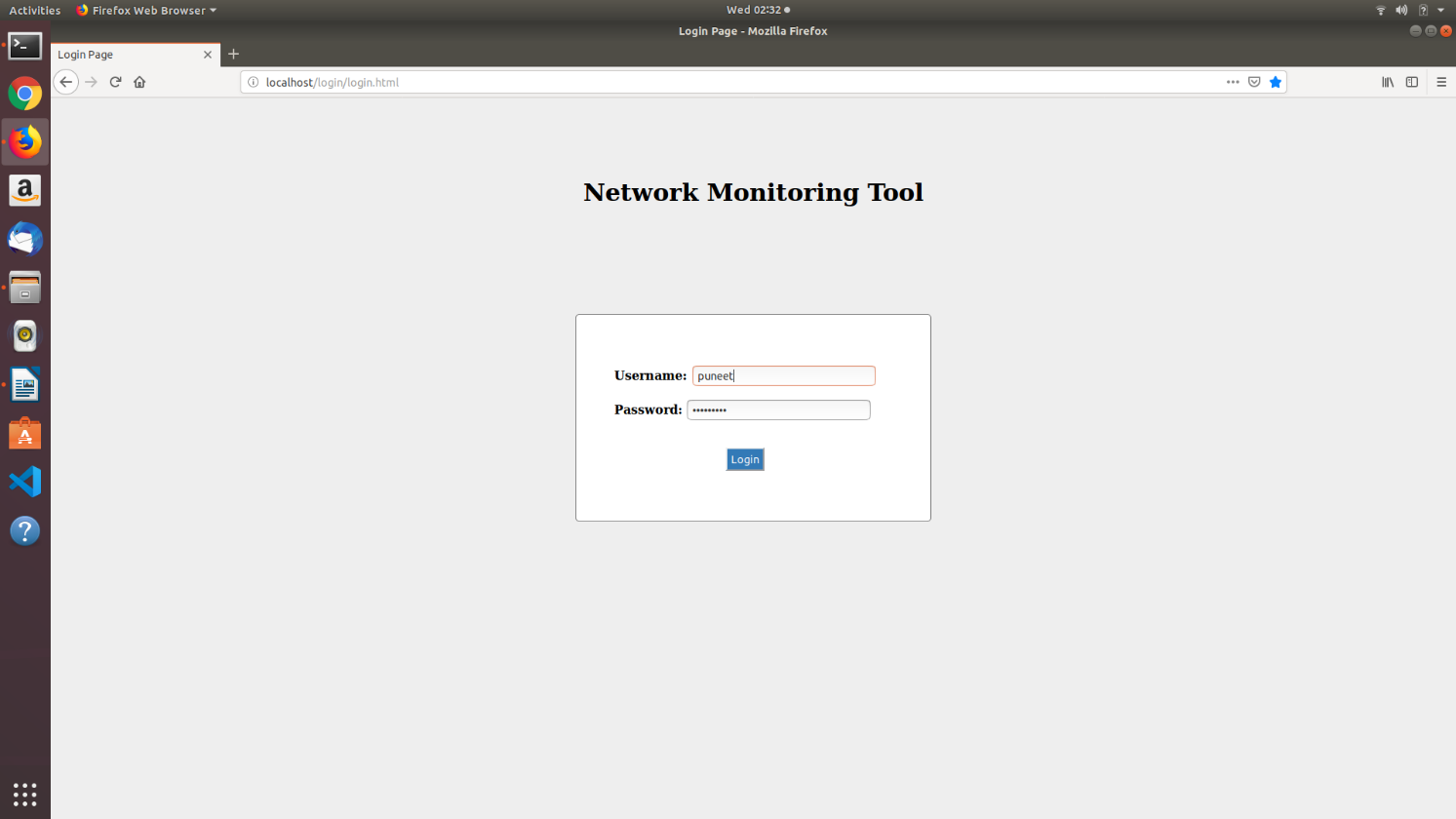
**Step 4: Install PHP**

sudo apt install php libapache2-mod-php php-mysql

**Change Directory**

cd /opt/var/www/html/nmt

**Login Page**



**Source Code**

**File Name: login.html**

<html>

<head>

<title> Login Page</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<br><br><br><br><h1 align="center" >Network Monitoring Tool</h1><br>

<div id="frm">

<form action="logindb.php" method="POST">

<p>

<B><label>Username:</label> </B>

<input type="text" id="user" name="user" placeholder="Username"/>

</p>

<p>

<B><label>Password:</label></B>

<input type="password" id="pass" name="pass" placeholder="Password"/>

</p>

<p> <br>

<input type="submit" id="btn" value="Login" />

</p>

</form>

</div>

</body>

</html>

**File Name: style.css**

body{

background: #eee;}

#frm{

border: solid gray 1px;

width: 20%;

border-radius: 5px;

margin: 100px auto;

background: white;

padding: 50px;

}

#btn{

color: #fff;

background: #337ab7;

padding: 5px;

margin-left: 40%;}

**Database**

**Step 1: Open Command Line**

**Step 2 : Sign into MySQL**

mysql -u root -p

**Step 3: create a database**

CREATE DATABASE login;

**Step 4: list of the current databases that you have created**

SHOW DATABASES;

**Step 5: select a database to use for subsequent operations**

USE login;

**Step 6: Creating Table**

CREATE TABLE users (user varchar (20) not null, pass varchar(20) not null);

**Step 7: Insert Data**

INSERT INTO users (user, pass) VALUES (“puneet”, “puneet917”);

**Step 8: Selecting Data**

Select \* from users;

**PHP Script**

**File Name: logind.php**

<?php

include "config.php";

ini\_set('display\_errors', '1');

$myusername = $\_POST['user'];

$mypassword = $\_POST['pass'];

$myusername = stripslashes($myusername);

$mypassword = stripslashes($mypassword);

$myusername = mysqli\_real\_escape\_string($con, $\_POST['user']);

$mypassword = mysqli\_real\_escape\_string($con, $\_POST['pass']);

shell\_exec("./cp.sh");

$sql\_query = "select \* from users where username='".$myusername."' and password='".$mypassword."'";

$result = mysqli\_query($con,$sql\_query);

$count = mysqli\_num\_rows($result);

if ($count == 1)

{

$var = shell\_exec("./cp.sh");

header("Location:selection.html");

}

else {

echo "YOU HAVE ENTERED WRONG USERNAME OR PASSWORD";

}

?>

**File Name: config.php**

<?php

session\_start();

$host = "localhost";

$user = "root";

$password = "puneet3959";

$dbname = "login";

$con = mysqli\_connect($host, $user, $password,$dbname);

if (!$con) {

die("Connection failed: " . mysqli\_connect\_error());

}

**File name: cp.sh**

#!/bin/bash

tcpdump -ne -c 3000 > pack (Capture & Store 3000 Packets)

grep "Flags" pack > tcp (Search TCP packets & stores Packets)

grep "UDP" pack > udp (Search UDP packets & stores Packets)

grep "ARP" pack > arp (Search ARP packets & stores Packets)

chmod 777 pack

chmod 777 tcp

chmod 777 udp (Gives permission to a file to read, write & execute )

chmod 777 arp

**Shell permission & execution command in php script**

**To give permission shell script in php script**

chmod 777 cp.sh

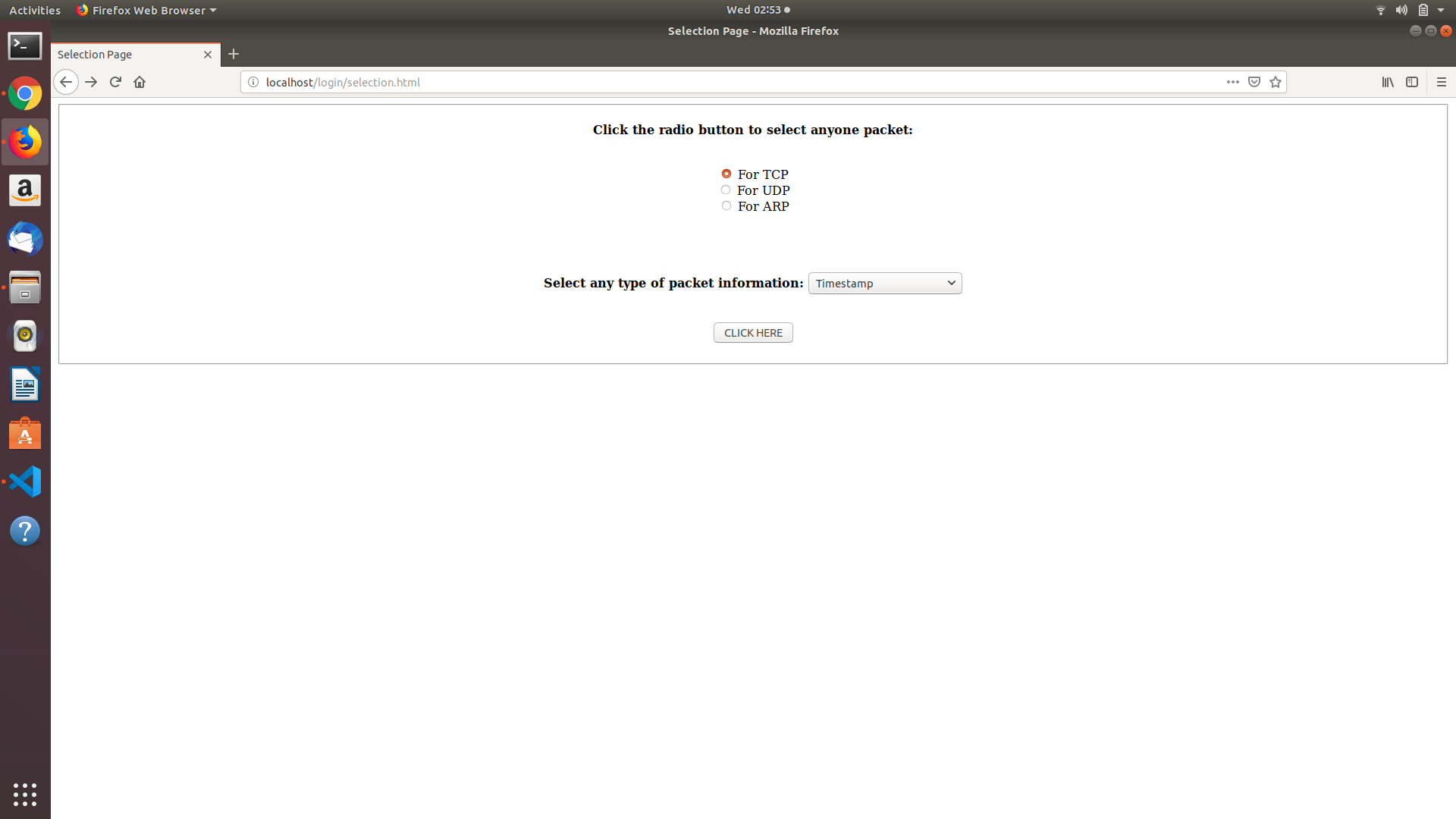
**To execute shell script in php script**

./cp.sh

Once the packet are being separated, next step is to filter out important attributes from packets which are mentioned as below:

* **TimeStamp**
* **Source MAC Address**
* **Destination MAC Address**
* **Source IP Address**
* **Destination IP Address**
* **Source Port**
* **IP Version**
* **Length of Packet**

**User Option Form**



**Figure 17 Select Packets & Information**

**File Name: selection.html**

<!DOCTYPE html>

<html>

<head>

<title> Selection Page</title>

</head>

<body>

<form action="monitoring.php" method="POST">

<fieldset>

<p style="text-align: center">

<b>Click the radio button to select anyone packet:</b> <br>

<br><br>

<input type="radio" name="networktype" value="TCP" checked> For TCP<br>

<input type="radio" name="networktype" value="UDP"> For UDP<br>

<input type="radio" name="networktype" value="ARP"> For ARP<br>

<br><br>

<br><br>

<b>Select any type of packet information:</b>

<select name="choice">

<option selected="true" disabled="disabled">Select one</option>

<option value="Timestamp">Timestamp</option>

<option value="Ip version">Ip version</option>

<option value="Source Ip">Source Ip Address</option>

<option value="Destination Ip">Destination Ip Address</option>

<option value="Source Port">Source Port</option>

<option value="Source Mac Address">Source Mac Address</option>

<option value="Destination Mac Address">Destination Mac Address</option>

<option value="Length of Packet">Length of Packet</option>

</select>

<br><br> </p>

<p style="text-align: center">

<input type="submit" name="submit" value="CLICK HERE">

</form>

</fieldset>

</body>

</html>

**File Name: monitoring.php**

<?php

if(isset($\_POST['submit']))

$networktype = $\_POST['networktype'];

$choice = $\_POST['choice'];

echo $choice;

echo "<br>";

if($networktype == "TCP")

{

if($choice == "Timestamp")

{

$var = shell\_exec("./tcptime.sh");

echo $var;

echo "<br>"."TCP\_TIMESTAMP PRINTED SUCCESSFULLY";

}

elseif($choice == "Ip version")

{

$var = shell\_exec("./tcpipv.sh");

echo $var;

echo "<br>"."TCP\_IP\_VERSIONS PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Ip")

{

$var = shell\_exec("./tcpsrc.sh");

echo $var;

echo "<br>"."TCP\_SOURCE\_IP PRINTED SUCCESSFULLY";

}

elseif($choice == "Destination Ip")

{

$var = shell\_exec("./tcpdst.sh");

echo $var;

echo "<br>"."TCP\_DESTINATION\_IP PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Port")

{

$var = shell\_exec("./tcpsport.sh");

echo $var;

echo "<br>"."TCP\_SOURCE\_PORT PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Mac Address")

{

$var = shell\_exec("./tcpsmac.sh");

echo $var;

echo "<br>"."TCP\_SOURCE\_MAC\_ADDRESS PRINTED SUCCESSFULLY";

}

elseif($choice == "Destination Mac Address")

{

$var = shell\_exec("./tcpdmac.sh");

echo $var;

echo "<br>"."TCP\_DESTINATION\_MAC\_ADDRESS PRINTED SUCCESSFULLY";

}

elseif($choice == "Length of Packet")

{

$var = shell\_exec("./tcplpac.sh");

echo $var;

echo "<br>"."TCP\_LENGTH\_OF\_PACKET PRINTED SUCCESSFULLY";

}

}

elseif($networktype == "UDP")

{

if($choice == "Timestamp")

{

$var = shell\_exec("./udptime.sh");

echo $var;

echo "<br>"."UDP\_TIMESTAMP PRINTED SUCCESSFULLY";

}

elseif($choice == "Ip version")

{

$var = shell\_exec("./udpipv.sh");

echo $var;

echo "<br>"."UDP\_IP\_VERSIONS PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Ip")

{

$var = shell\_exec("./udpsrc.sh");

echo $var;

echo "<br>"."UDP\_SOURCE\_IP PRINTED SUCCESSFULLY";

}

elseif($choice == "Destination Ip")

{

$var = shell\_exec("./udpdst.sh");

echo $var;

echo "<br>"."UDP\_DESTINATION\_IP PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Port")

{

$var = shell\_exec("./udpsport.sh");

echo $var;

echo "<br>"."UDP\_SOURCE\_PORT PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Mac Address")

{

$var = shell\_exec("./udpsmac.sh");

echo $var;

echo "<br>"."UDP\_SOURCE\_MAC\_ADDRESS PRINTED SUCCESSFULLY";

}

elseif($choice == "Destination Mac Address")

{

$var = shell\_exec("./udpdmac.sh");

echo $var;

echo "<br>"."UDP\_DESTINATION\_MAC\_ADDRESS PRINTED SUCCESSFULLY";

}

elseif($choice == "Length of Packet")

{

$var = shell\_exec("./udplpac.sh");

echo $var;

echo "<br>"."UDP\_LENGTH\_OF\_PACKET PRINTED SUCCESSFULLY";

}

}

elseif($networktype == "ARP")

{

if($choice == "Timestamp")

{

$var = shell\_exec("./arptime.sh");

echo $var;

echo "<br>"."ARP\_TIMESTAMP PRINTED SUCCESSFULLY";

}

elseif($choice == "Ip version")

{

$var = shell\_exec("./arpipv.sh");

echo $var;

echo "<br>"."ARP\_IP\_VERSION PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Ip")

{

$var = shell\_exec("./arpsrc.sh");

echo $var;

echo "<br>"."ARP\_SOURCE\_IP PRINTED SUCCESSFULLY";

}

elseif($choice == "Destination Ip")

{

$var = shell\_exec("./arpdst.sh");

echo $var;

echo "<br>"."ARP\_DESTINATION\_IP PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Port")

{

$var = shell\_exec("./arpsport.sh");

echo $var;

echo "<br>"."ARP\_SOURCE\_PORT PRINTED SUCCESSFULLY";

}

elseif($choice == "Source Mac Address")

{

$var = shell\_exec("./arpsmac.sh");

echo $var;

echo "<br>"."ARP\_SOURCE\_MAC\_ADDRESS PRINTED SUCCESSFULLY";

}

elseif($choice == "Destination Mac Address")

{

$var = shell\_exec("./arpdmac.sh");

echo $var;

echo "<br>"."ARP\_DESTINATION\_MAC\_ADDRESS PRINTED SUCCESSFULLY";

}

elseif($choice == "Length of Packet")

{

$var = shell\_exec("./arplpac.sh");

echo $var;

echo "<br>"."ARP\_LENGTH\_OF\_PACKET PRINTED SUCCESSFULLY";

}

}

?>

**To CUT ARP Packets Information**

**File Name : arpdmac.sh**

#!/bin/bash

cut -f 4 -d " " arp

**File Name : arpdst.sh**

#!/bin/bash

grep "Request" arp | cut -f 14 -d " " | cut -f 1 -d "," && grep "Reply" arp | cut -f 13 -d " " | cut -f 1 -d ","

**File Name : arpipv.sh**

#!/bin/bash

echo "NO IPV VERSION FOUND";

**File Name : arplpac.sh**

#!/bin/bash

cut -f 9 -d " " arp | cut -f 1 -d ":"

**File Name : arpsmac.sh**

#!/bin/bash

cut -f 2 -d " " arp

**File Name : arpsport.sh**

#!/bin/bash

echo "no source port";

**File Name : arpsrc.sh**

#!/bin/bash

grep "Reply" arp | cut -f 11 -d " " && grep "Request" arp | cut -f 12 -d " "

**File Name : arptime.sh**

#!/bin/bash

cut -f 1 -d " " arp

**To CUT TCP Packets Information**

**File Name : tcpdmac.sh**

#!/bin/bash

cut -f 4 -d " " tcp | cut -f 1 -d ","

**File Name : tcpdst.sh**

#!/bin/bash

grep "IPv4" tcp | cut -f 12 -d " " | cut -f 1-4 -d "."

grep "IPv6" tcp | cut -f 12 -d " " | cut -f 1 -d "."

**File Name : tcpipv.sh**

#!/bin/bash

cut -f 6 -d " " tcp

**File Name : tcplpac.sh**

#!/bin/bash

cut -f 9 -d " " tcp | cut -f 1 -d ":"

**File Name : tcpsmac.sh**

#!/bin/bash

cut -f 2 -d " " tcp

**File Name : tcpsport.sh**

#!/bin/bash

grep "IPv4" tcp | cut -f 10 -d " " > tsrc

grep "IPv6" tcp | cut -f 10 -d " " > tsrc1

cut -f 5 -d "." tsrc

cut -f 2 -d "." tsrc1

rm tsrc

rm tsrc1

**File Name : tcpsrc.sh**

#!/bin/bash

grep "IPv4" tcp | cut -f 10 -d " " | cut -f 1-4 -d "."

grep "IPv6" tcp | cut -f 10 -d " " | cut -f 1 -d "."

**File Name : tcptime.sh**

#!/bin/bash

cut -f 1 -d " " tcp

**To CUT UDP Packets Information**

**File Name : udpdmac.sh**

#!/bin/bash

cut -f 4 -d " " udp | cut -f 1 -d ","

**File Name : udpdst.sh**

#!/bin/bash

cut -f 12 -d " " udp | cut -f 1 -d ":" | cut -d '.' -f 1-4

**File Name : udpipv.sh**

#!/bin/bash

cut -f 6 -d " " udp

**File Name : udplpac.sh**

#!/bin/bash

cut -f 9 -d " " udp | cut -f 1 -d ":"

**File Name : udpsmac.sh**

#!/bin/bash

cut -f 2 -d " " udp

**File Name : udpsport.sh**

#!/bin/bash

grep "IPv4" udp | cut -f 10 -d " " > usrc

grep "IPv6" udp | cut -f 10 -d " " > usrc1

# cut -f 12 -d " " udp > usrc

cut -f 5 -d "." usrc

cut -f 2 -d "." usrc1

rm usrc

rm usrc1

# cut -f 10 -d " " udp > usrc

# cut -f 2 -d "." usrc

# rm usrc

**File Name : udpsrc.sh**

#!/bin/bash

grep "IPv4" udp | cut -f 10 -d " " | cut -f 1-4 -d "."

grep "IPv6" udp | cut -f 10 -d " " | cut -f 1 -d "."

**File Name: udptime.sh**

#!/bin/bash

cut -f 1 -d " " udp

**Permission of file**

**CHMOD:** In Linux and Unix, everything is a file. Directories are files, files are files and devices are files. Devices are usually referred to as a node; however, they are still files. All of the files on a system have permissions that allow or prevent others from viewing, modifying or executing. If the file is of type Directory then it restricts different actions than files and device nodes. The super user "root" has the ability to access any file on the system. Each file has access restrictions with permissions, user restrictions with owner/group association. Permissions are referred to as bits.

To change or edit files that are owned by root, **sudo** must be used

There are three types of access restrictions:

|  |  |  |
| --- | --- | --- |
| **Permission** | **Action** | **chmod option** |
| read | (view) | r or 4 |
| write | (edit) | w or 2 |
| execute | (execute) | x or 1 |

**Permission & Execution are required in Project are:**

chmod 777 cp.sh

./cp.sh

chmod 777 arpdmac.sh

./arpmac.sh

chmod 777 arpdst.sh

./arpdst.sh

chmod 777 arpipv.sh

./ arpipv.sh

chmod 777 arplpac.sh

./ arplpac.sh

chmod 777 arpsmac.sh

./ arpsmac.sh

chmod 777 arpsport.sh

./ arpsport.sh

chmod 777 arpsrc.sh

./ arpsrc.sh

chmod 777 arptime.sh

./ rptime.sh

chmod 777 tcpdmac.sh

./ tcpdmac.sh

chmod 777 tcpdst.sh

./ tcpdst.sh

chmod 777 tcpipv.sh

./ tcpipv.sh

chmod 777 tcplpac.sh

./ tcplpac.sh

chmod 777 tcpsmac.sh

./ tcpsmac.sh

chmod 777 tcpsport.sh

./ tcpsport.sh

chmod 777 tcpsrc.sh

./ 777 tcpsrc.sh

chmod 777 tcptime.sh

./ tcptime.sh

chmod 777 udpdmac.sh

./ udpdmac.sh

chmod 777 udpdst.sh

./ udpdst.sh

chmod 777 udpipv.sh

./ udpipv.sh

chmod 777 udplpac.sh

./ udplpac.sh

chmod 777 udpsmac.sh

./ udpsmac.sh

chmod 777 udpsport.sh

./ udpsport.sh

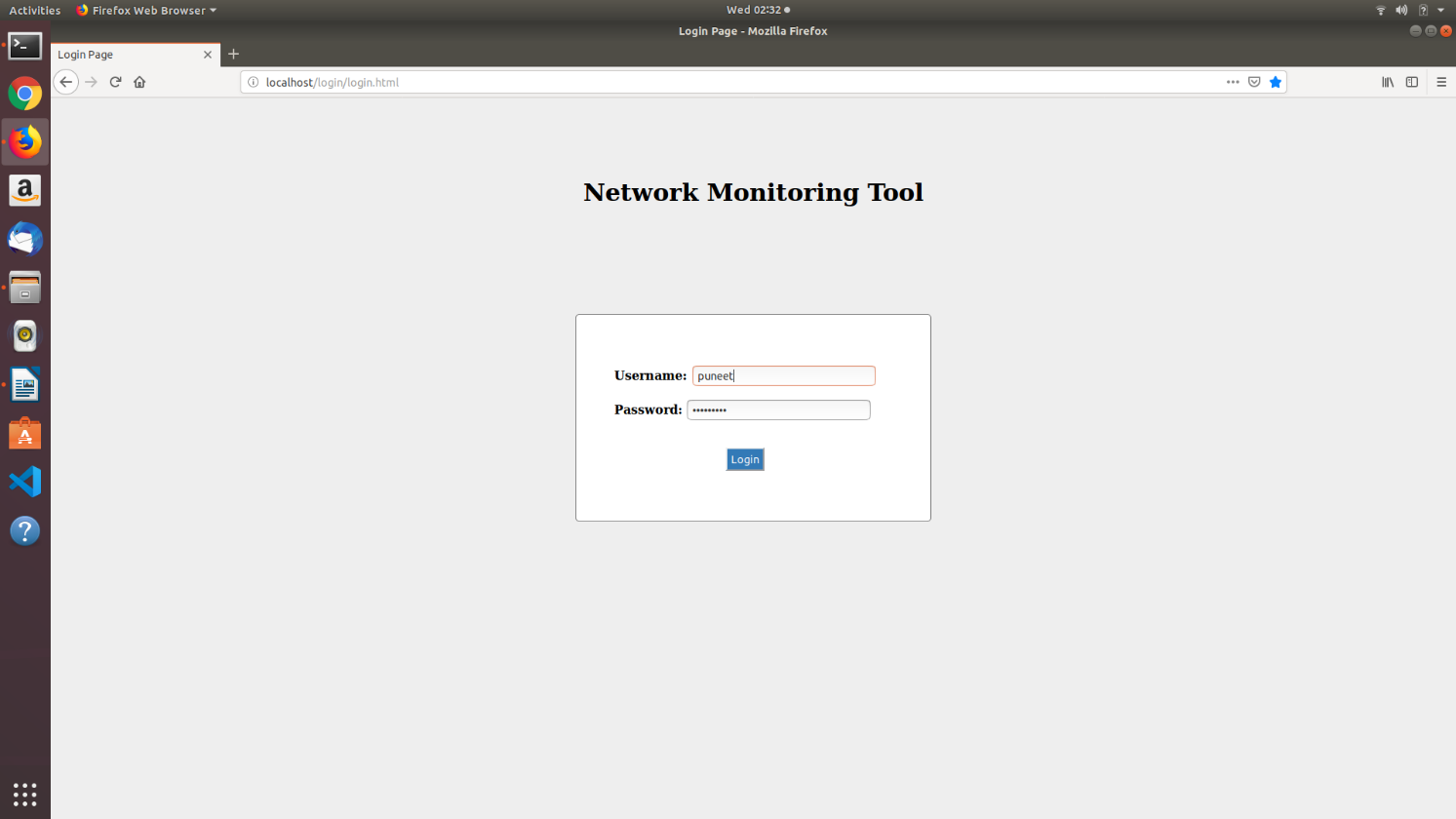
chmod 777 udpsrc.sh

./ udpsrc.sh

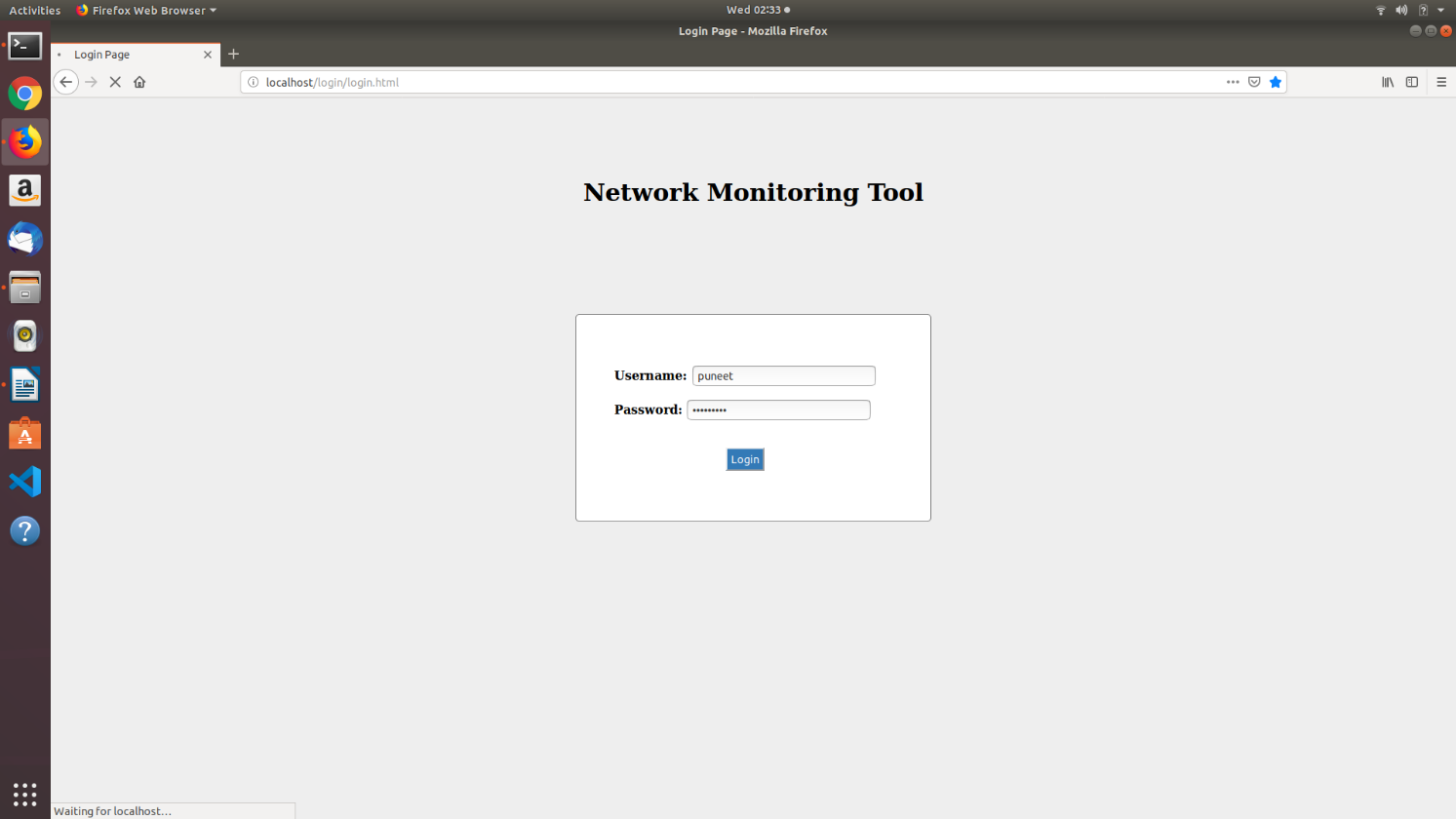
chmod 777 udptime.sh

./ udptime.sh

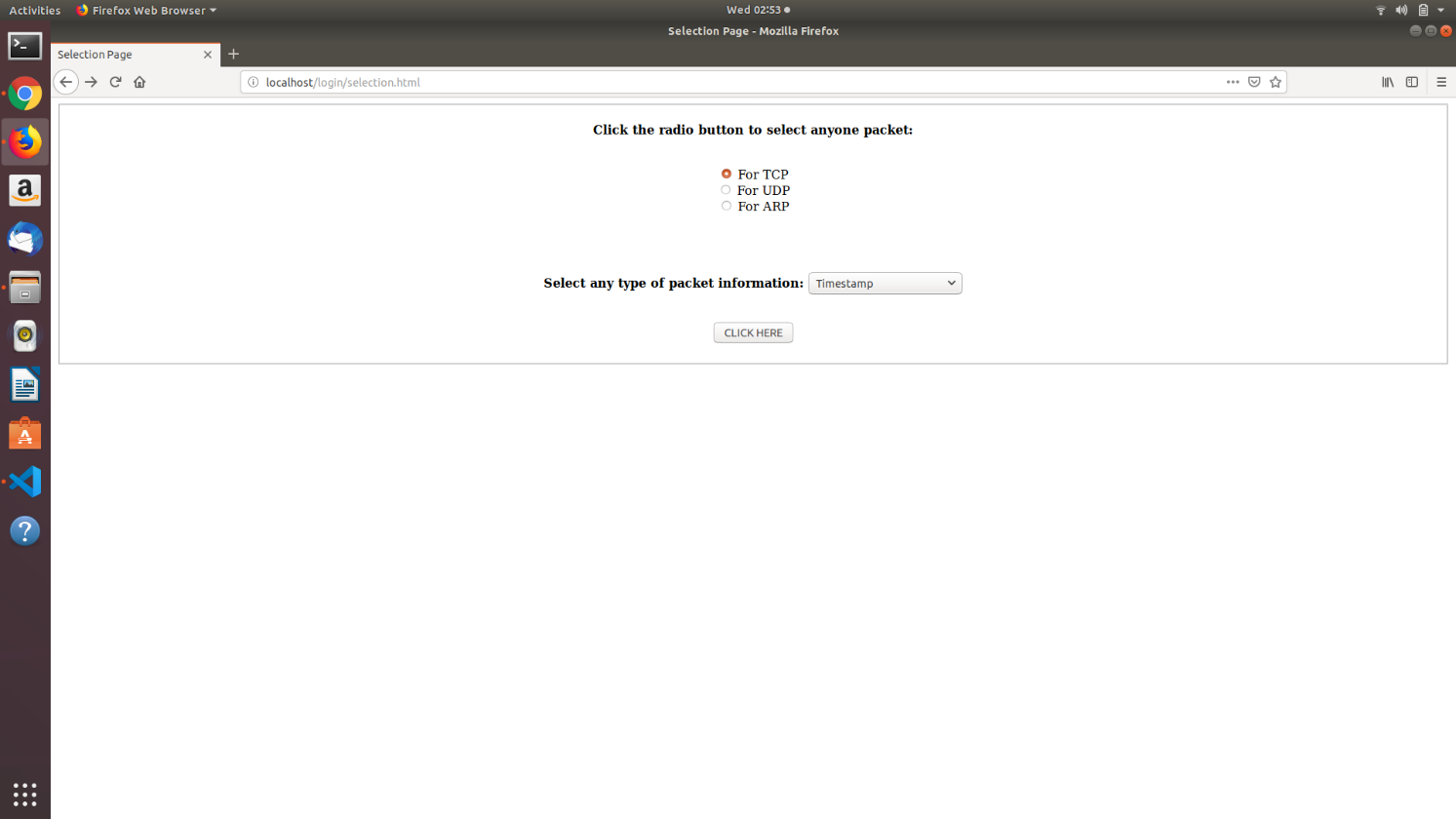
**Final Working of the Project**



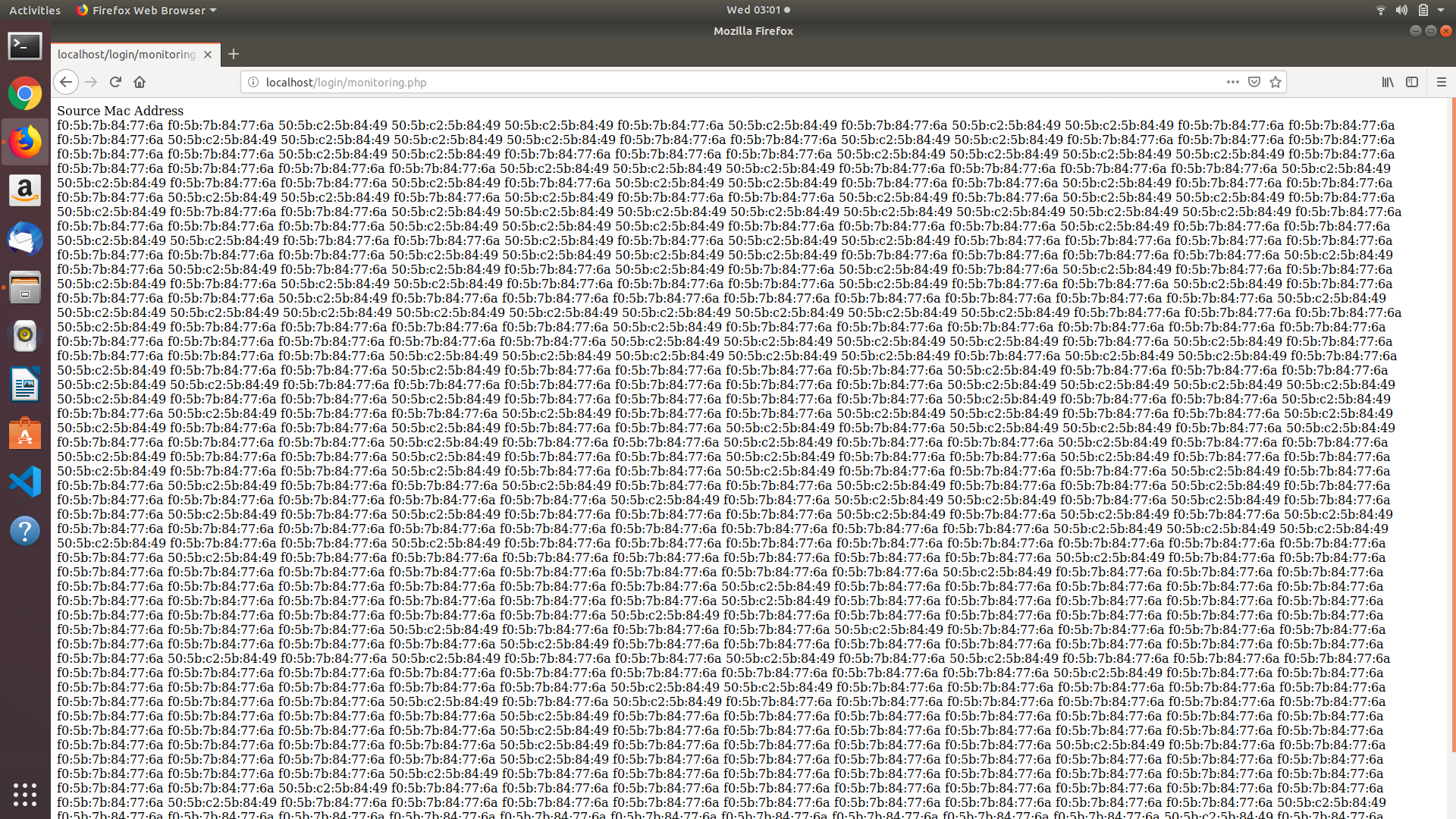
**Figure 18 Open Login Page in Browser**



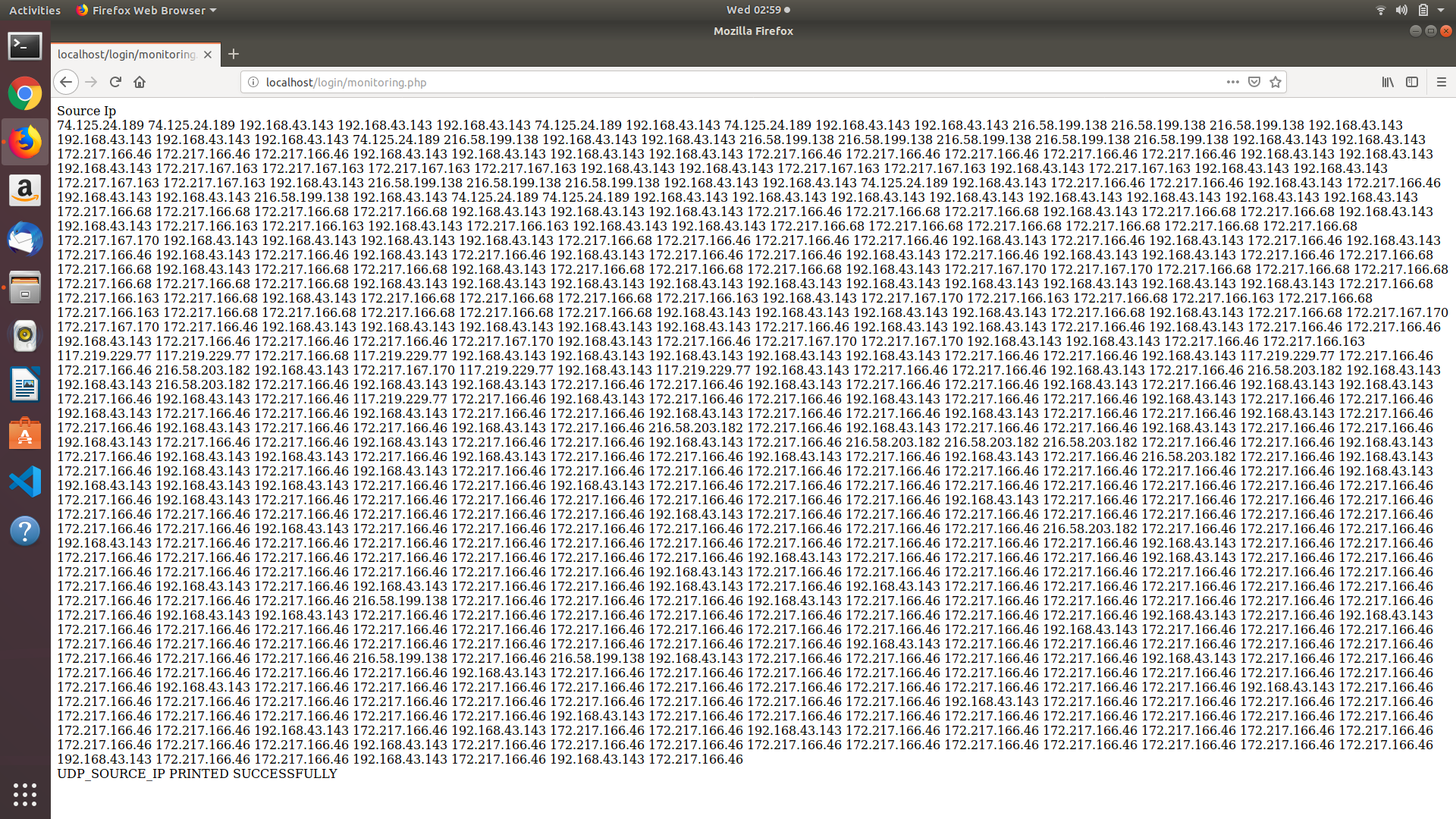
**Figure 19 Login Page is Capturing & Storing Packets also Grep ARP, UDP, TCP packets**



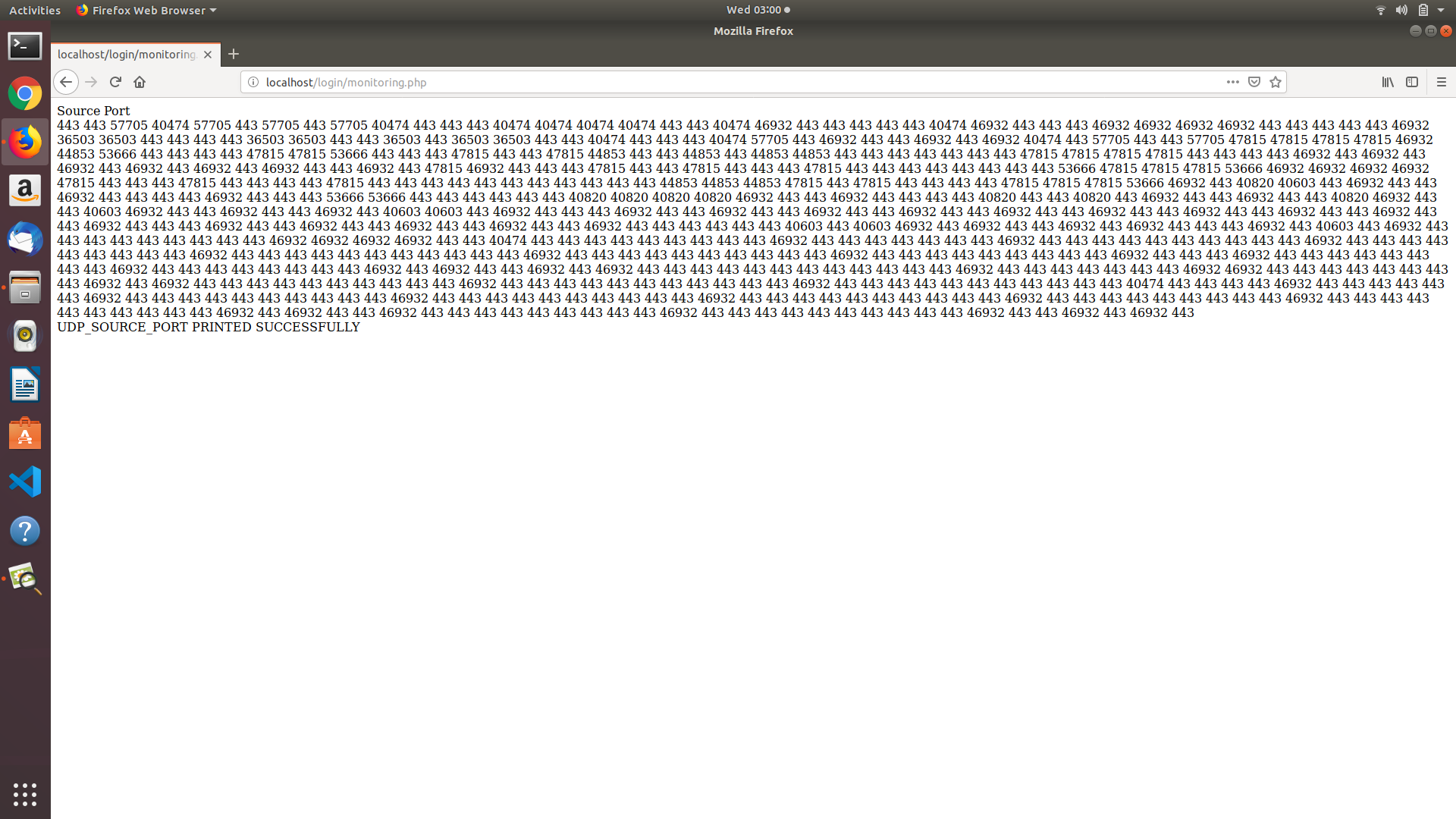
**Figure 20 Select Packet & Information**



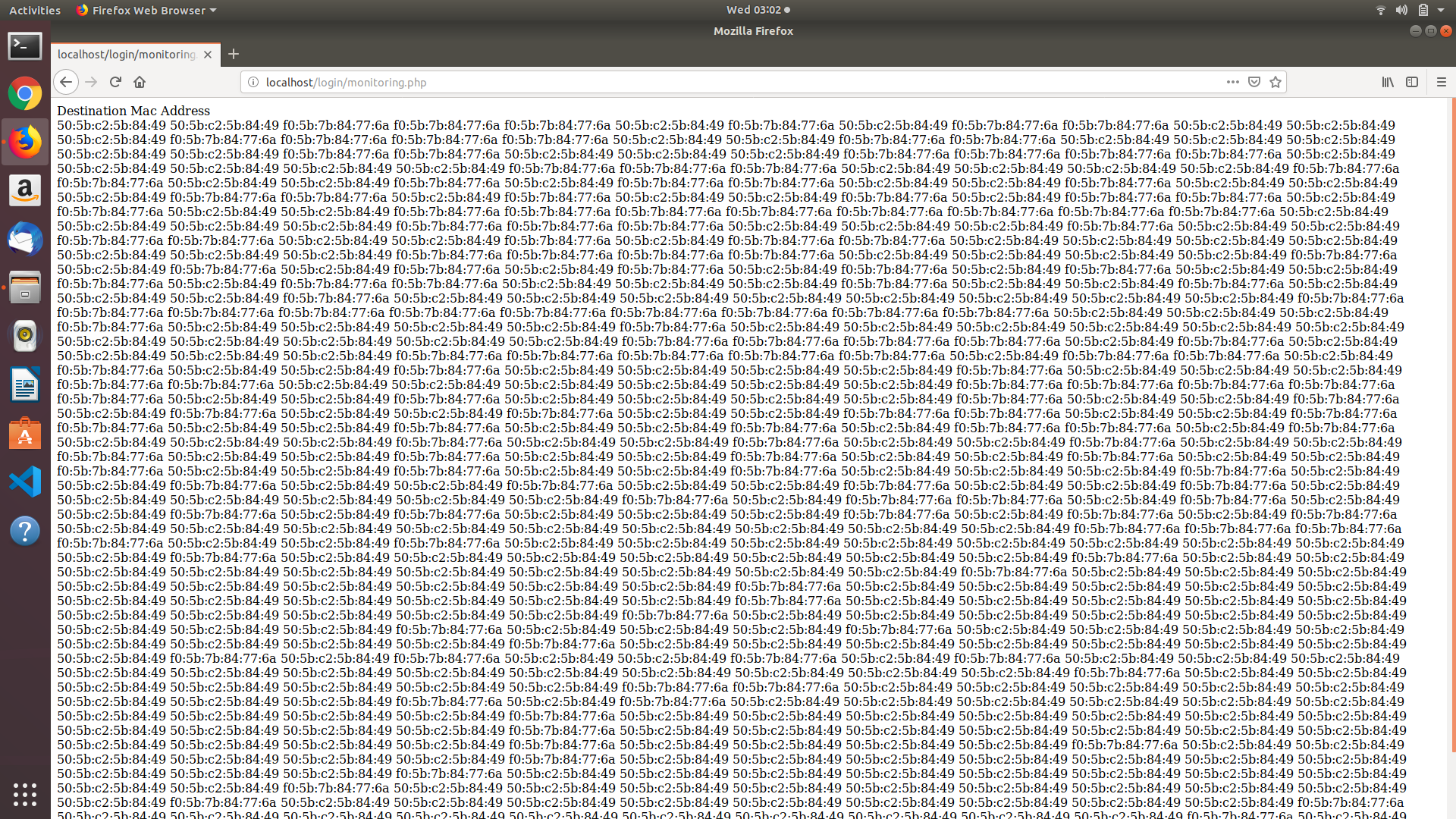
**Figure 21 UDP SOURCE MAC ADDRESS**



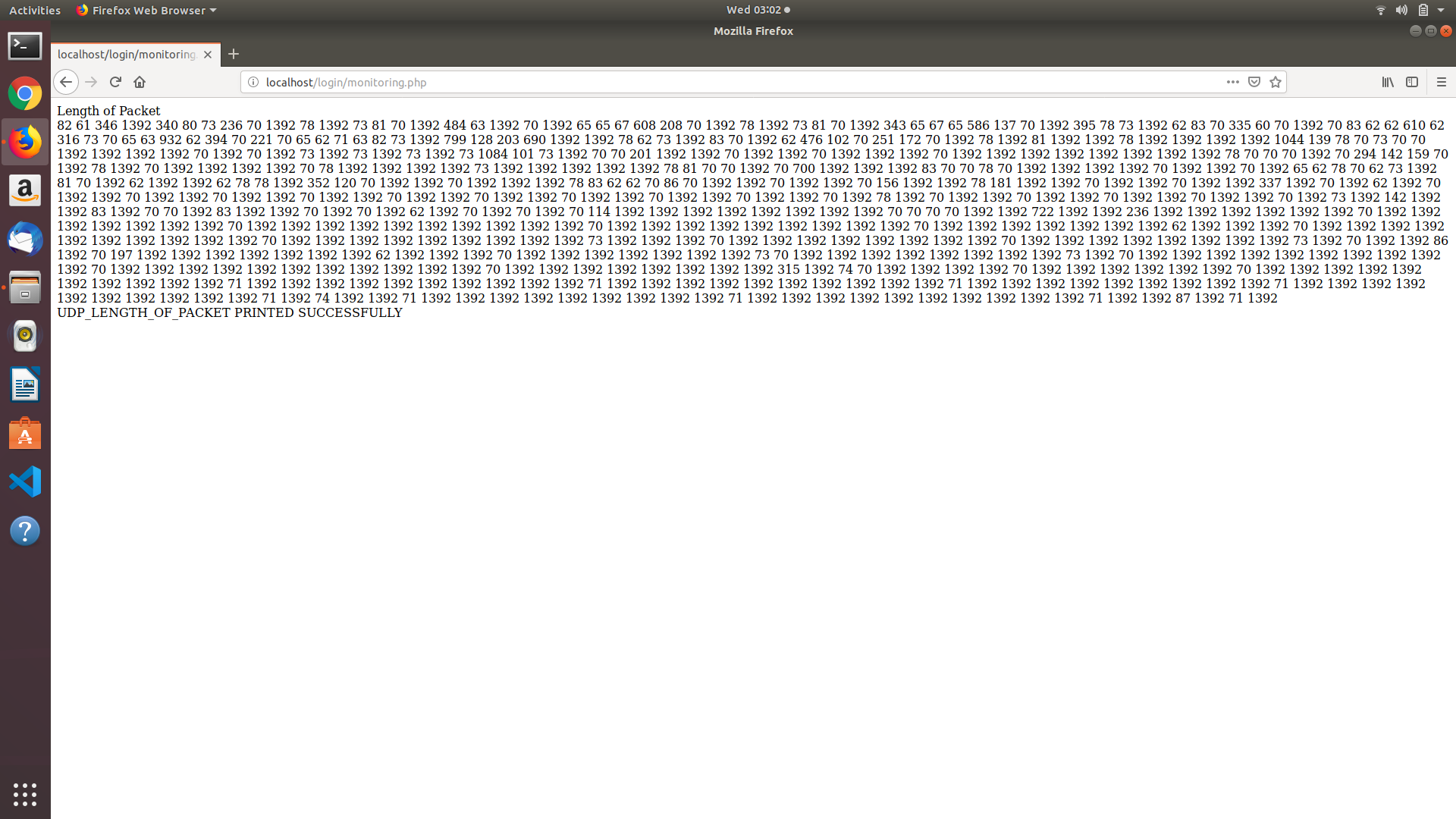
**Figure 22 UDP SOURCE IP ADDRESS**

****

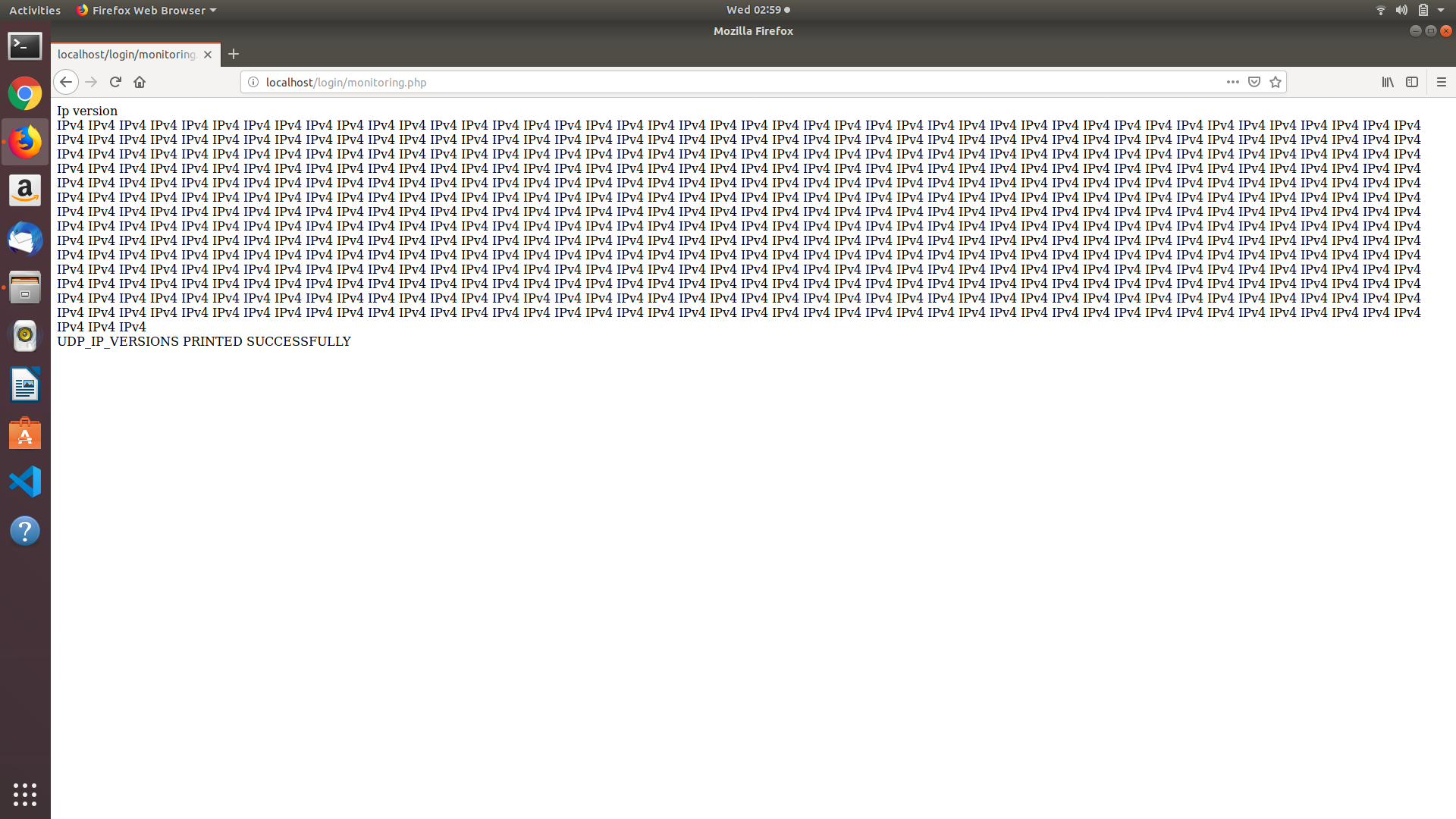
**Figure 23 UDP SOURCE PORT**

****

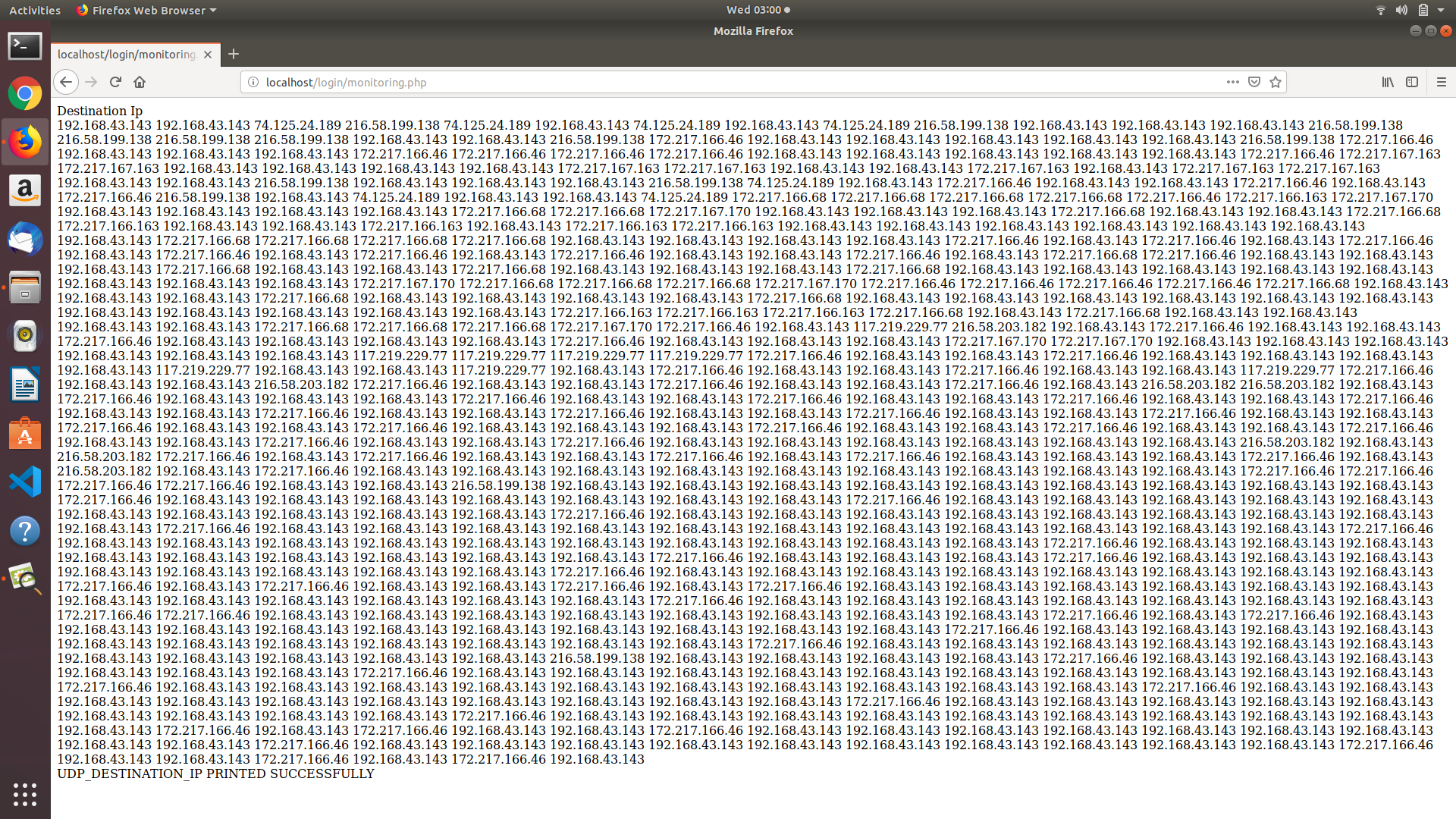
**Figure 24UDP DESTINATION MAC ADDRESS**

****

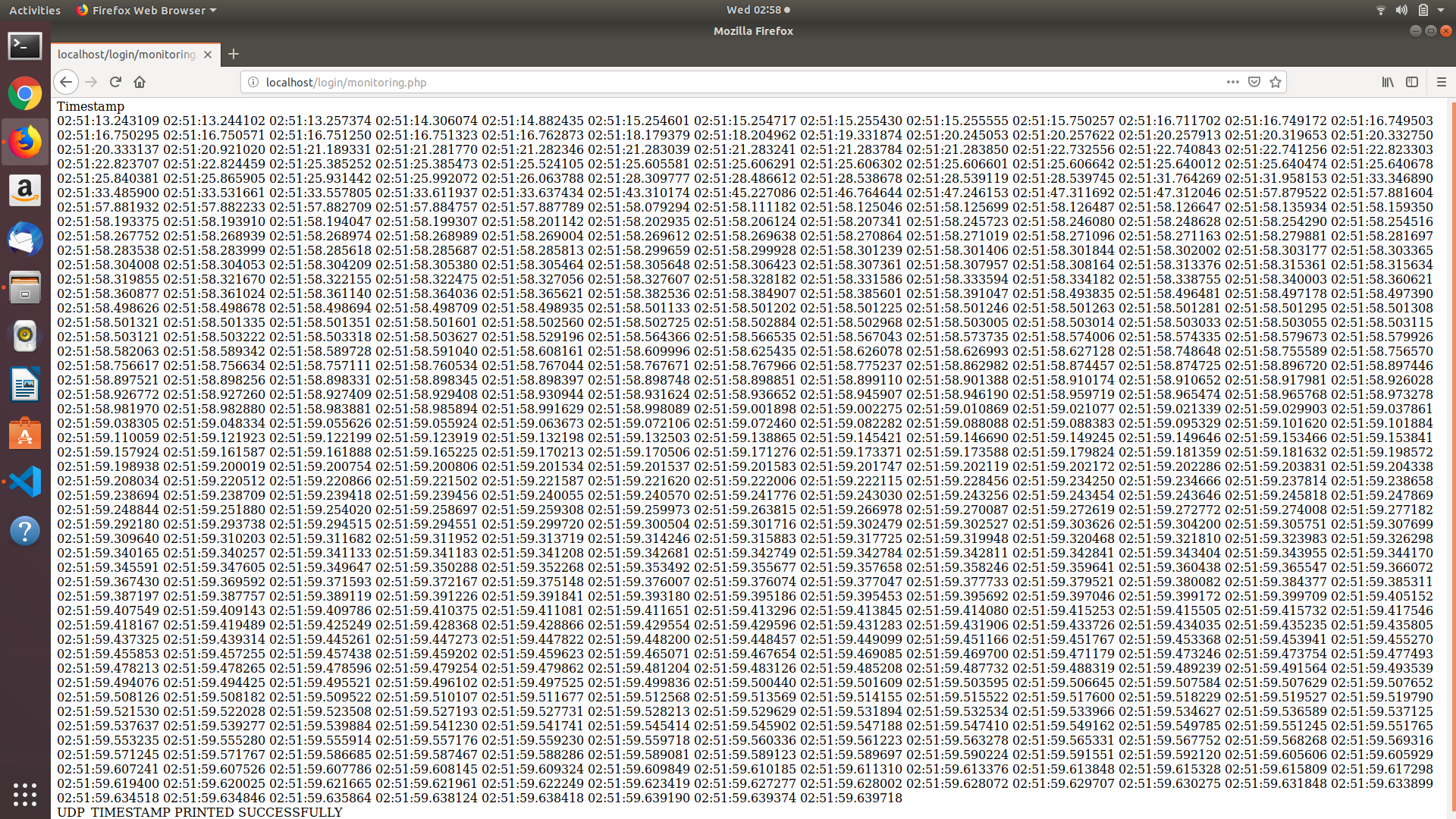
**Figure 25UDP LENGTH OF PACKET**

****

**Figure 26UDP IP VERSION**

****

**Figure 27UDP DESTINATION IP ADDRESS**

****

**Figure 28UDP TIME STAMP**

Summary

A packet analyzer used for intercepting traffic on wireless network is known as wireless analyzer or Wi-Fi analyzer. A packet analyzer can also be referred to as a network analyzer or protocol analyzer through these terms also have other meaning.

Other Notable packet analyzer are:

* **Wireshark**
* **SolarWinds Packet Analysis Bundle**
* **PRTG Network Monitor**
* **Steel Central Packet Analyzer**