PROJECT REPORT

NETWORK MONITORING TOOL

Navpreet Singh
(Supervisor)

Chief Engineer
Computer Center, IIT Kanpur

Devansh Gupta
(Student)

B.S. Computer Science
Arizona State University, USA

ACKNOWLEDGMENT

It has been an immense privilege to have Mr. Navpreet Singh, Chief Engineer at the Computer Centre, IIT Kanpur, as my mentor for this summer project. I would like to take this opportunity to express my deepest gratitude to him for his timely and invaluable advice.

Additionally, I am profoundly grateful to Mr. Saurabh Malhotra for his help, regular supervision of my work, and unwavering guidance throughout this project.

- Devansh Gupta
B.S. Computer Science
Arizona State University, USA

INDEX

Institute Profile

Established in 1960, the Indian Institute of Technology Kanpur (IIT Kanpur) quickly gained prominence, with the Parliament of India recognizing all IITs as "Institutions of National Importance" through the 'Institutes of Technology Act 1961'. From its inception, the foundation of IIT Kanpur was strengthened through significant collaboration with the United States under the Kanpur Indo-American Programme (KIAP), leveraging the expertise of industrially advanced Western nations to build a world-class technical institute.

IIT Kanpur is renowned for its rigorous selection process at undergraduate, postgraduate, and research levels, ensuring it admits the brightest minds in India. The institute offers advanced courses, access to eminent faculty, well-equipped laboratories, comprehensive libraries, and excellent hostel facilities, all contributing to outstanding opportunities for research and development. Its commitment to scientific and innovative teaching methodologies fosters an environment where both faculty and students consistently achieve recognition through research publications, projects, fellowships, and industrial exposure. This continual pursuit of excellence solidifies IIT Kanpur's status as a prestigious institution producing graduates with exceptional career prospects and universal acclaim.

Research Areas

IIT Kanpur has consistently demonstrated excellence in a wide array of research fields, thriving in a vibrant academic atmosphere where strong academic and research activities complement each other. This synergy, nurtured by students at all levels under the expert guidance of faculty members, drives the institute to the forefront of knowledge and innovation. Key research areas include Finite Element Methods using Domain Decomposition, Flow-Induced Vibrations, Computational Chemistry, Genomics and Bioinformatics, Molecular Dynamics, Tomography, Robotics, Neural Networks, and Genetic Algorithms. Recently, IIT Kanpur has established a robust research group focused on Nanoscience and Nanotechnology.

A Hallmark of Educational Excellence

IIT Kanpur excels in pioneering original research and advancing technology at the cutting edge. The institute is dedicated to training students to become competent and motivated engineers and scientists. It fosters freedom of thought, cultivates vision, encourages growth, and instills human values along with a concern for the environment and society.

As a key player in shaping technical education in India, IIT Kanpur has been instrumental in conceptualizing and implementing a grand vision for engineering education. This premier institute has frequently been ranked as the top engineering institution in the nation and holds the distinction of being ranked third among the world's leading engineering educational systems. IIT Kanpur is committed to maintaining its leadership not only nationally but also internationally.

Objective

The objective of this project is to develop a Network Monitoring Tool (NMT), which oversees and ensures the availability and overall performance of computers (hosts) and network services within a network. The tool monitors the network to identify issues caused by overloaded or crashed servers, network connections, and other devices. In the event of an outage, it promptly notifies the network administrator, enabling immediate resolution of the problem.

The NMT is designed to detect abnormal behavior in any host. When such behavior is detected, the tool inspects the host's activities by decrypting data packets sent across the network by the host machine. By analyzing this packet information, the tool can determine the nature of the host's communications. Based on this analysis, appropriate actions can be taken to ensure the host functions properly.

The NMT leverages tcpdump, a powerful packet analyzer, to capture packets received by a network interface. The scope of packets captured can be fine-tuned using logical operators and parameters such as source and destination MAC or IP addresses, protocol types (IP and Ethernet), and TCP/UDP port numbers. This selective packet capture allows for precise monitoring and analysis of network traffic.

Abstract

The Network Monitoring Tool is a local service that logs and monitors data flowing in and out of your network, helping administrators detect congestion and identify its root causes. This tool is essential for troubleshooting packet loss and latency issues.

Designed to support TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and ARP (Address Resolution Protocol), the Network Monitoring Tool captures key details such as timestamp, source IP, destination IP, source port, destination port, and packet length. These metrics are crucial for tracking all communication and data transfers within the network.

By constantly monitoring the network for slow or failing components, the tool notifies the network administrator of any issues. It provides insights into network performance during regular operations and facilitates the analysis of traffic between various terminals and systems. All monitored data is recorded and stored in data files and databases, allowing for thorough analysis.

If a host machine exhibits abnormal behavior, administrators can easily retrieve the relevant data to diagnose and address the problem. Additionally, if further analysis is needed, decryption tools can be used to examine the data in more detail.

Overall, the Network Monitoring Tool is an invaluable resource for maintaining network health, ensuring smooth operations, and swiftly addressing any anomalies.

Introduction

Network monitoring is a critical IT function that can save money by improving network performance, employee productivity, and preventing infrastructure cost overruns. This can be achieved using various software tools or a combination of plug-and-play hardware and software appliance solutions. A network monitoring tool identifies and helps resolve issues such as slow webpage downloads, lost emails, questionable user activity, and file delivery problems caused by overloaded or crashed servers and unstable network connections.

When failures, slow responses, or other unexpected behaviors are detected, these tools send notifications to designated locations to alert system administrators, thus providing greater operational visibility. Virtually any type of network can be monitored, whether it's wireless or wired, a corporate LAN, VPN, or service provider WAN. Devices running different operating systems can be monitored, including servers, routers, switches, and cell phones.

The Network Monitoring Tool I have designed is simple yet highly effective, based on the **LAMP** architecture:
- **L** stands for Linux, the operating system.
- **A** stands for Apache, the web server.
- **M** stands for MySQL, the Database Management System.
- **P** stands for Perl/PHP/Python, the server-side scripting languages.

The first and most crucial step in designing a network monitoring tool is capturing all the data sent across the network. Data is transmitted in the form of packets. Every action on the Internet involves packets, such as receiving web pages or sending emails. These packets carry information needed to reach their destination, including the sender's IP address, the receiver's IP address, the total number of packets, and the sequence number of the specific packet. They use protocols such as TCP/IP and typically contain 1,000 to 1,500 bytes.

Packets are generally split into three parts:
1. **Header**: Contains instructions about the data carried by the packet, such as packet length, packet number, protocol, and destination address.
2. **Payload**: Also called the body or data, this is the actual data being delivered. Fixed-length packets may have the payload padded with blank information to reach the required size.
3. **Trailer**: Sometimes called the footer, it contains bits that indicate the end of the packet.

Captured data packets are categorized by type (ARP, TCP, or UDP), stored in respective tables in local files in a specific format, and then analyzed. This tool provides detailed packet information, including timestamp, source IP address, destination IP address, packet (payload) length, source port number, and destination port number. This allows for easy identification of any problematic machines. Additionally, the top-talker feature enhances the tool’s utility by identifying hosts that send the most packets and storing the details of each packet from these hosts in a separate table to aid network administrators.

Implementation

For the design process of this NMT, I have followed the below listed steps in the same order:

1. Packet Sniffing
2. Packet Processing
3. Data Storage
4. User Interface

- **PACKET SNIFFING:**
    - ⇒ The first and most fundamental step is sniffing packets traversing the network. For this purpose, we utilize the command-line packet capture tool, **tcpdump**.
    - ⇒ tcpdump is a widely used packet sniffer that operates from the command line, allowing users to intercept and display TCP/IP and other packets being transmitted or received over a network connected to the computer.
    - ⇒ tcpdump is compatible with most UNIX-like operating systems, including Linux, Solaris, BSD, Mac OS X, and others. There is also a Windows version known as Windump.
    - ⇒ tcpdump offers detailed insights into any network conversation traversing the wires. One of its key advantages is the ability to run it remotely via an SSH or TELNET session. Moreover, the machine running tcpdump does not need to have an x-windows environment. This application is lightweight and efficient, as it operates with minimal overhead due to its non-graphical interface.

- ## PACKET PROCESSING:
  - ⇒ The next step is to process the captured packets so they can be written to the database. To proceed, we will write a shell script using the vi editor. This shell script will include the tcpdump command at the beginning. As previously mentioned, this script is invoked at runtime when the user signs in to use the tool.

Some key UNIX commands used for Packet Processing:

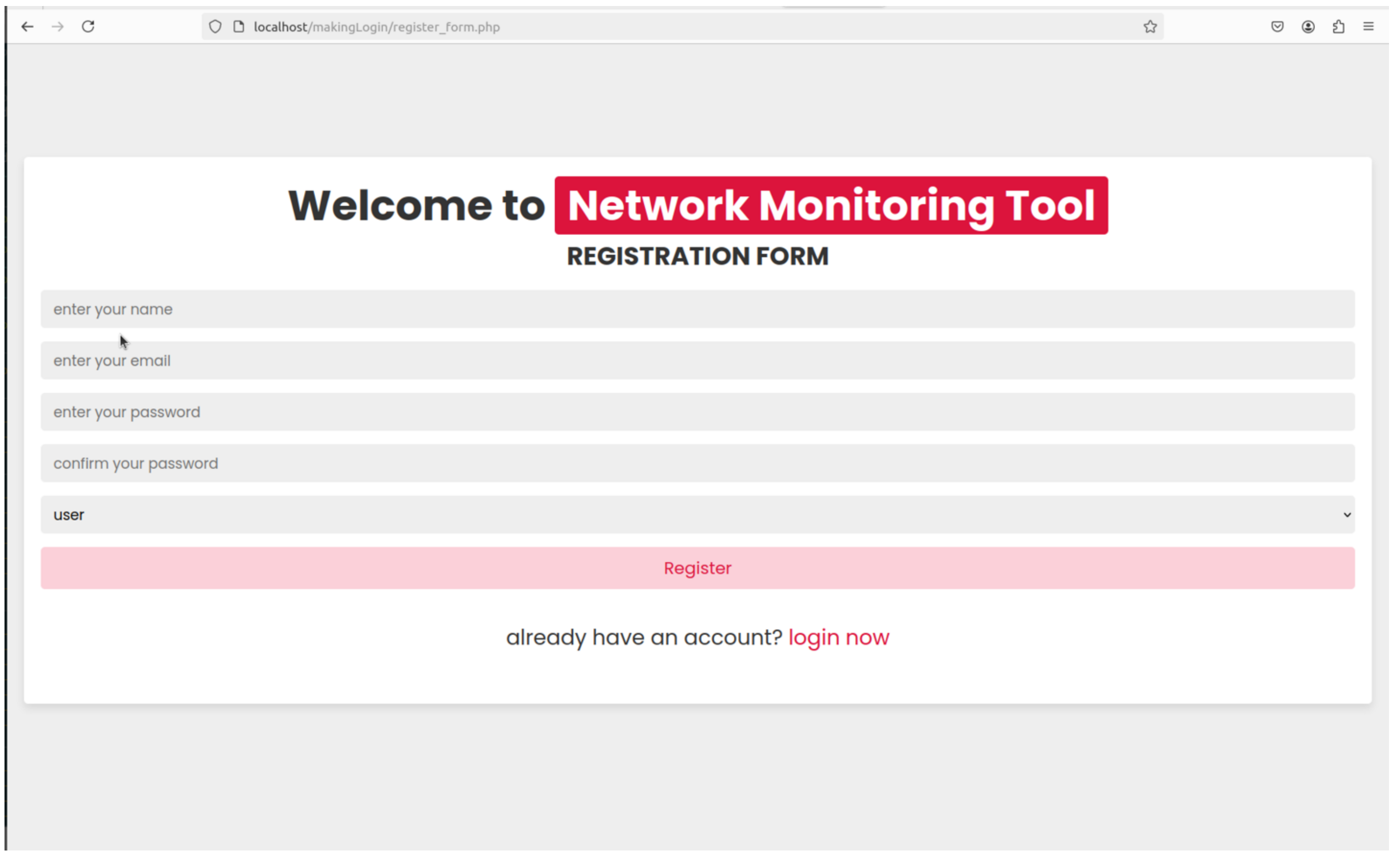
- **grep:** Searching for a pattern or keyword put in as filter and save then into a file.
- **awk:** Helps split/extract each input line into fields
- **cut:** Extracts desired information from a string of the fragmented line
- **tr:** Helps remove unwanted characters to get clean & formatted data
- **echo:** Helps to write the information extracted to the respective output files

- **DATA STORAGE:**
  - ⇒ I have used MySQL to save the login credentials of the users.
  - ⇒ After the data is processed (i.e. filtered), I have used the local file system to save the data.
  - ⇒ The data which consists of different segments of the captured data packets, such as the source IP, destination IP etc. is saved in several .txt files.
  - ⇒ There is also an all-info file that has all the packets captured filtered out for a certain protocol i.e. for ARP, TCP and UDP.

- **USER INTERFACE:**
  - ⇒ The representation of the data extracted and stored is done using HTML, CSS and PHP.
  - ⇒ PHP scripts have been particularly implemented for fetching the data from the respective data files.

Below are some screenshots of the user interface:

- **register_form.php:**


localhost/makingLogin/register_form.php

Welcome to Network Monitoring Tool

REGISTRATION FORM

enter your name

enter your email

enter your password

confirm your password

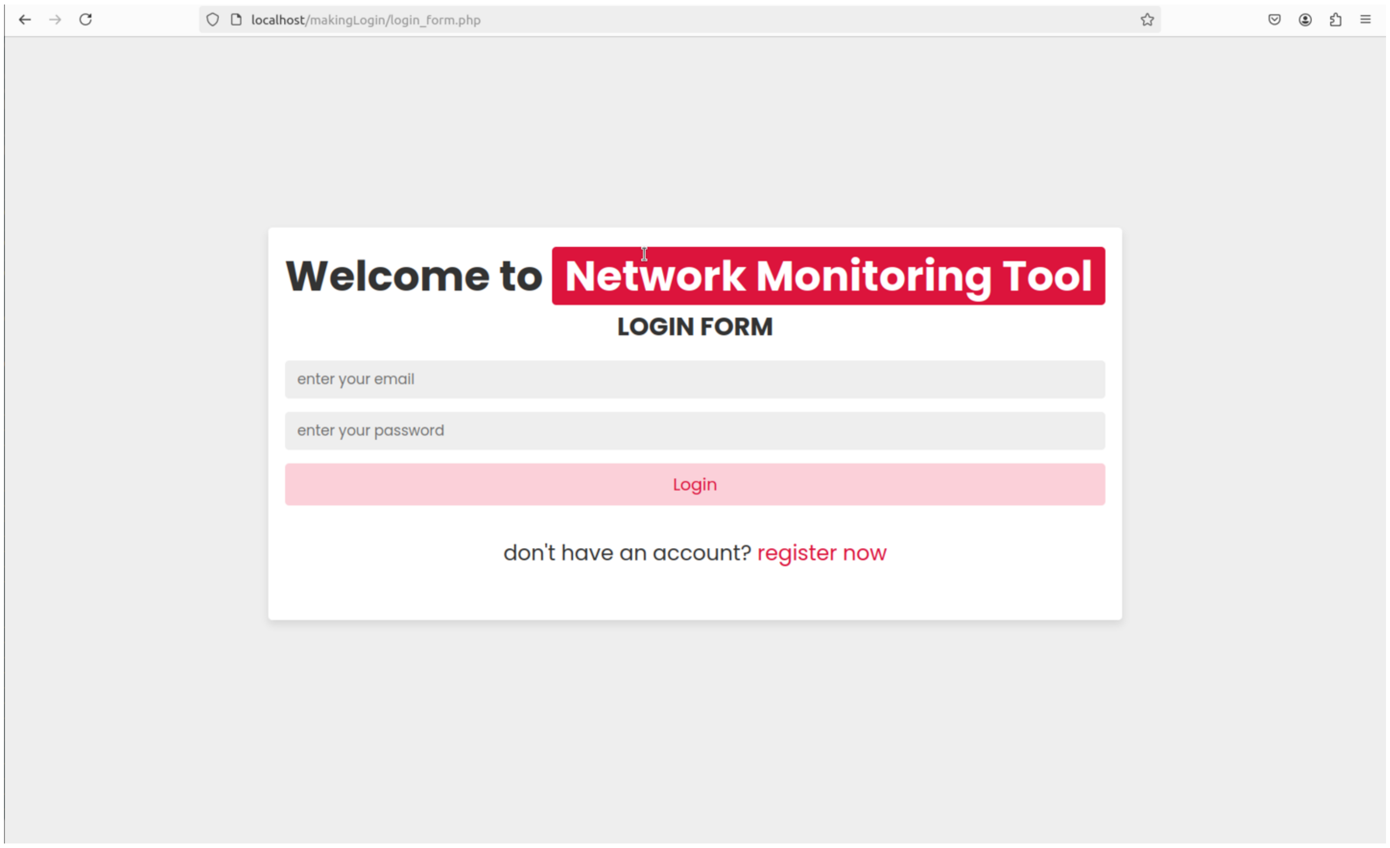
user

Register

already have an account? login now

⇒ A new user can create their account in the NMT's login database by entering the required information.

- **login_form.php:**


localhost/makingLogin/login_form.php

Welcome to Network Monitoring Tool

LOGIN FORM

enter your email

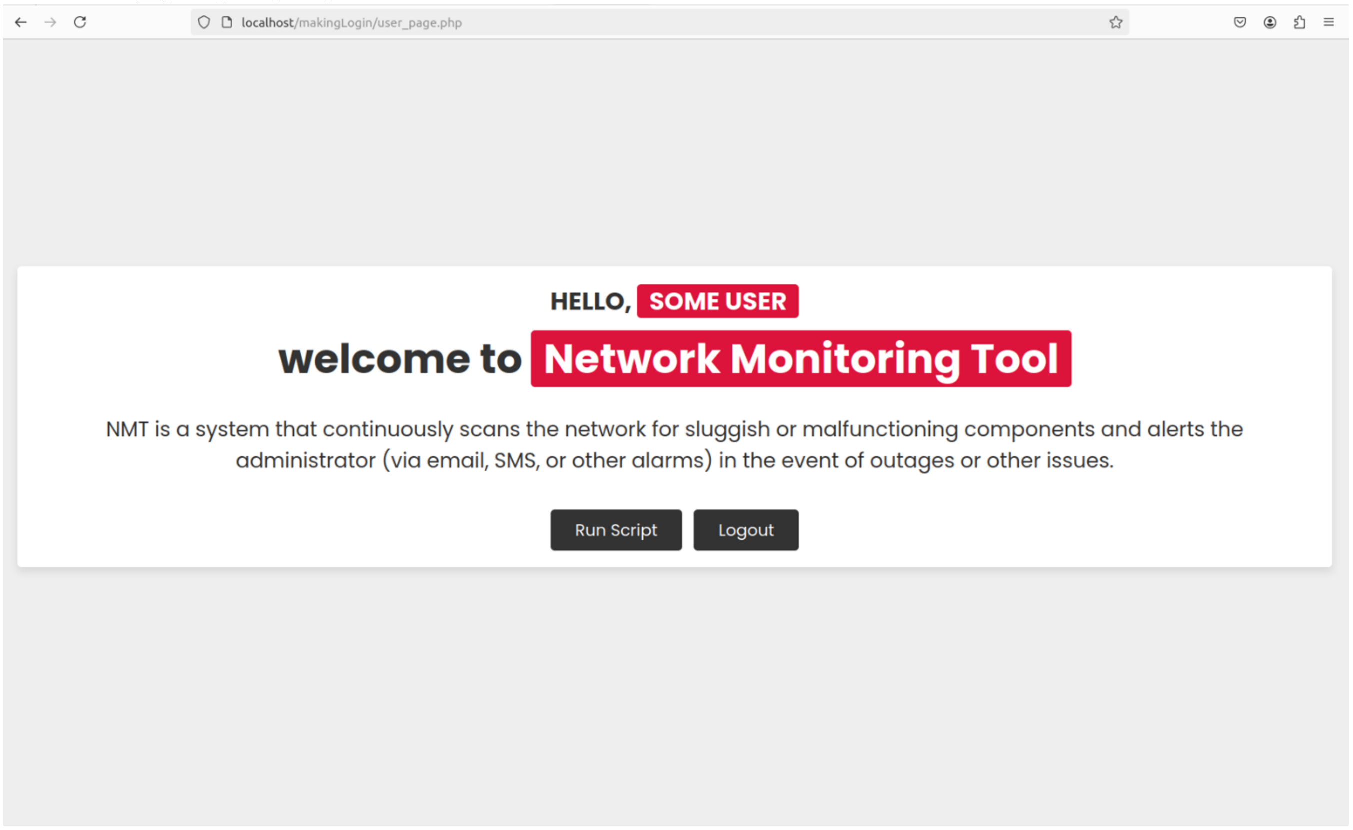
enter your password

Login

don't have an account? register now

- The user can login by entering their credentials to use the NMT.

- **user_page.php:**



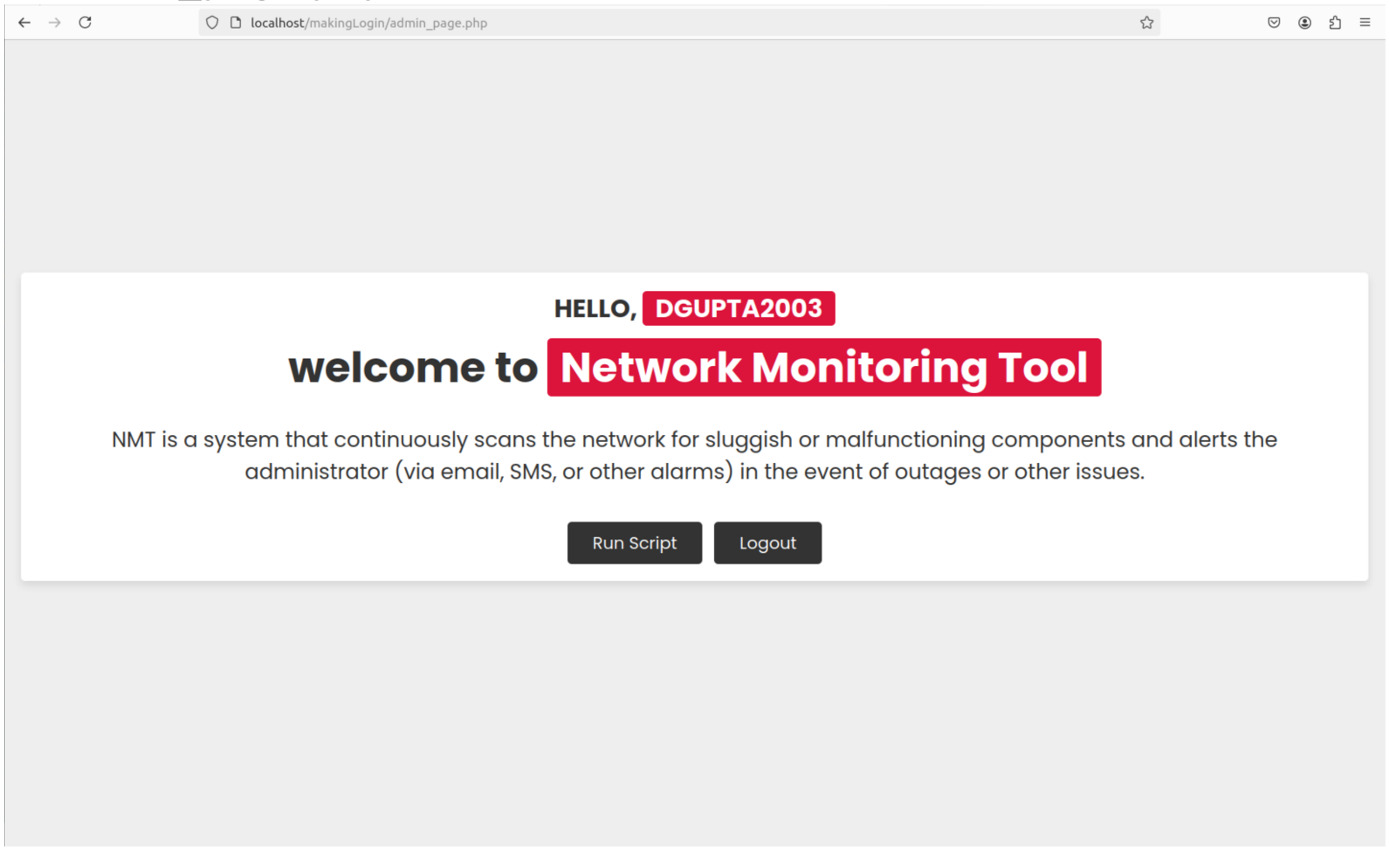
localhost/makingLogin/user_page.php

HELLO, SOME USER

welcome to Network Monitoring Tool

NMT is a system that continuously scans the network for sluggish or malfunctioning components and alerts the administrator (via email, SMS, or other alarms) in the event of outages or other issues.

- **admin_page.php :**



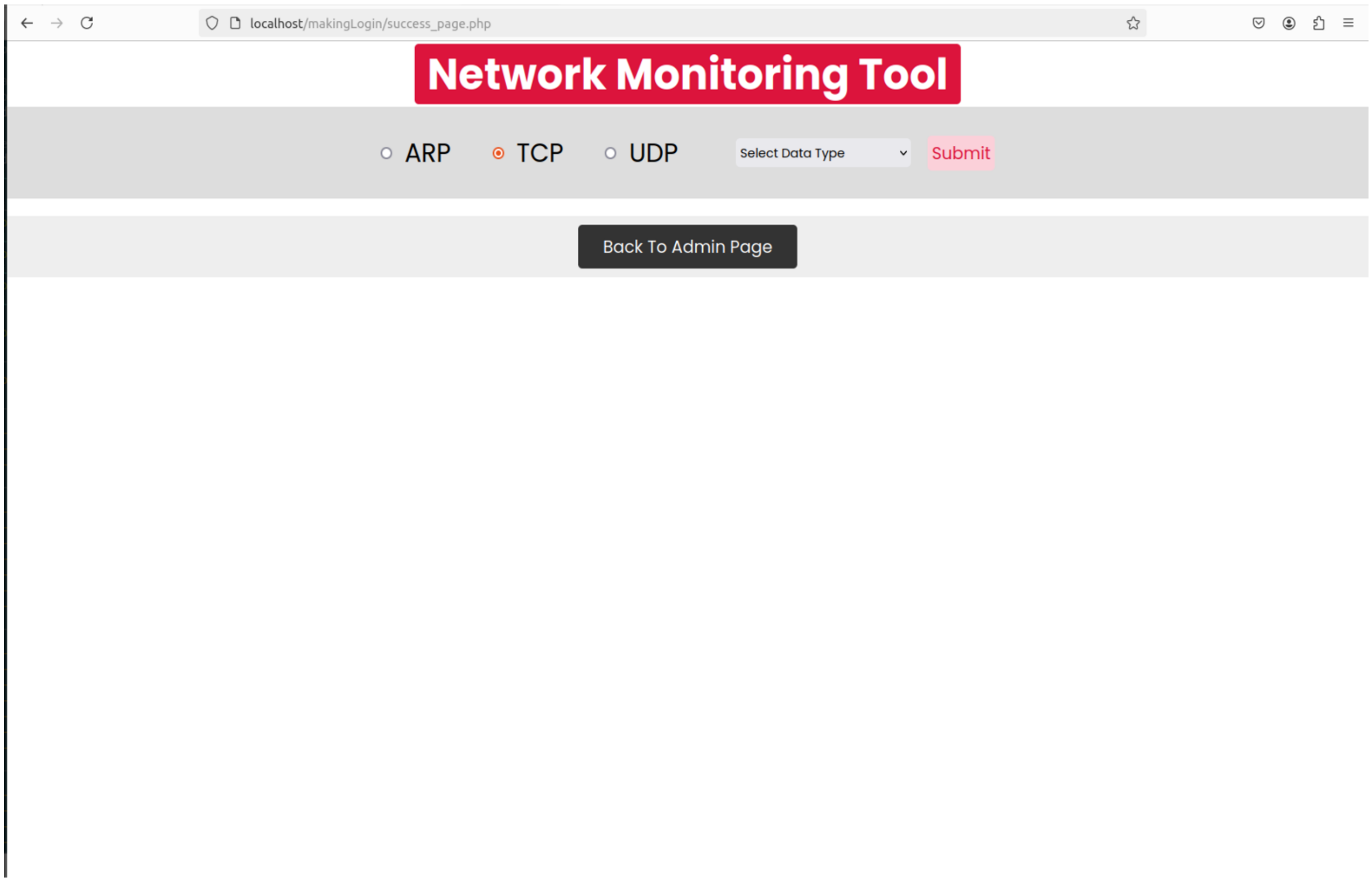
localhost/makingLogin/admin_page.php

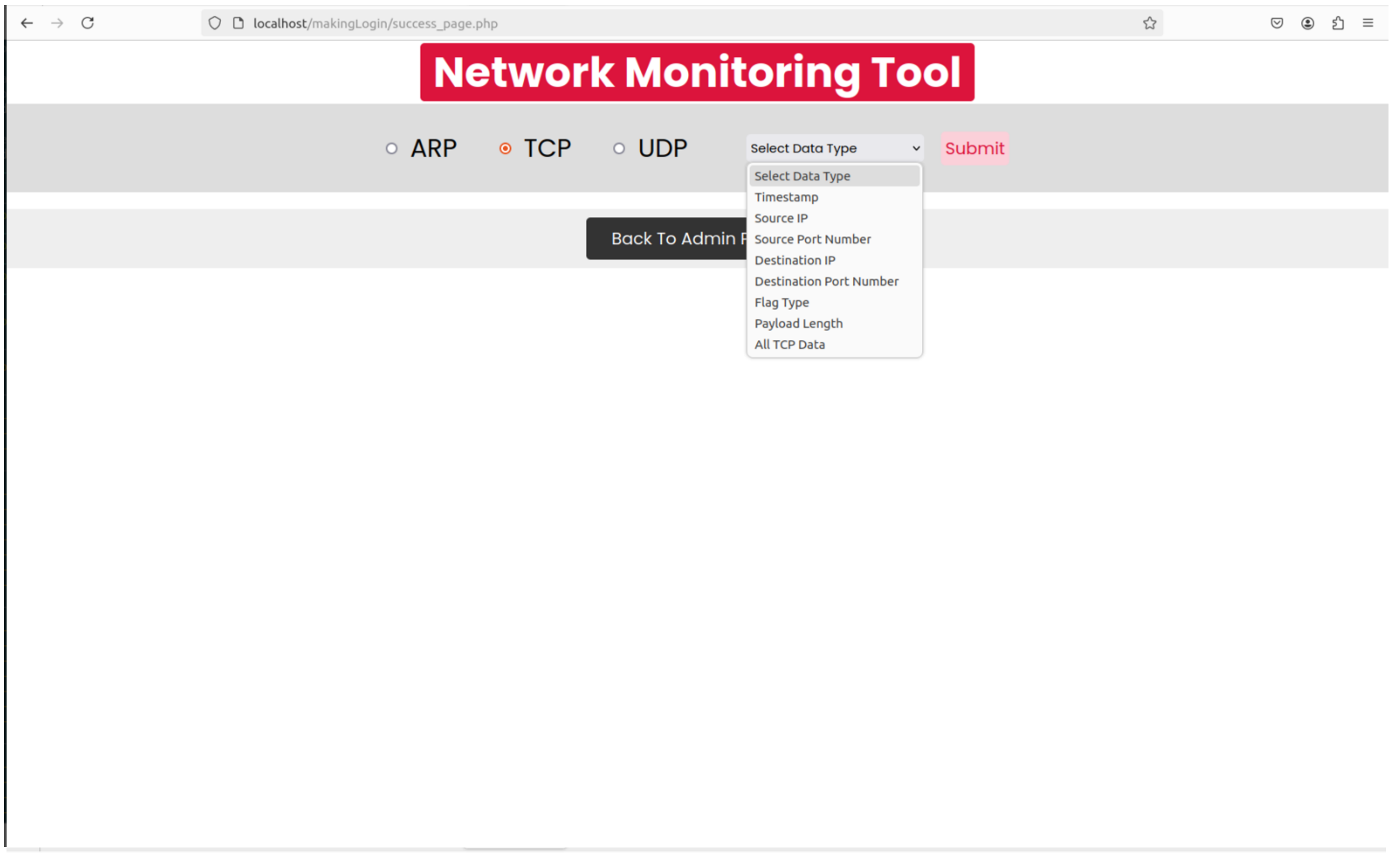
HELLO, DGUPTA2003

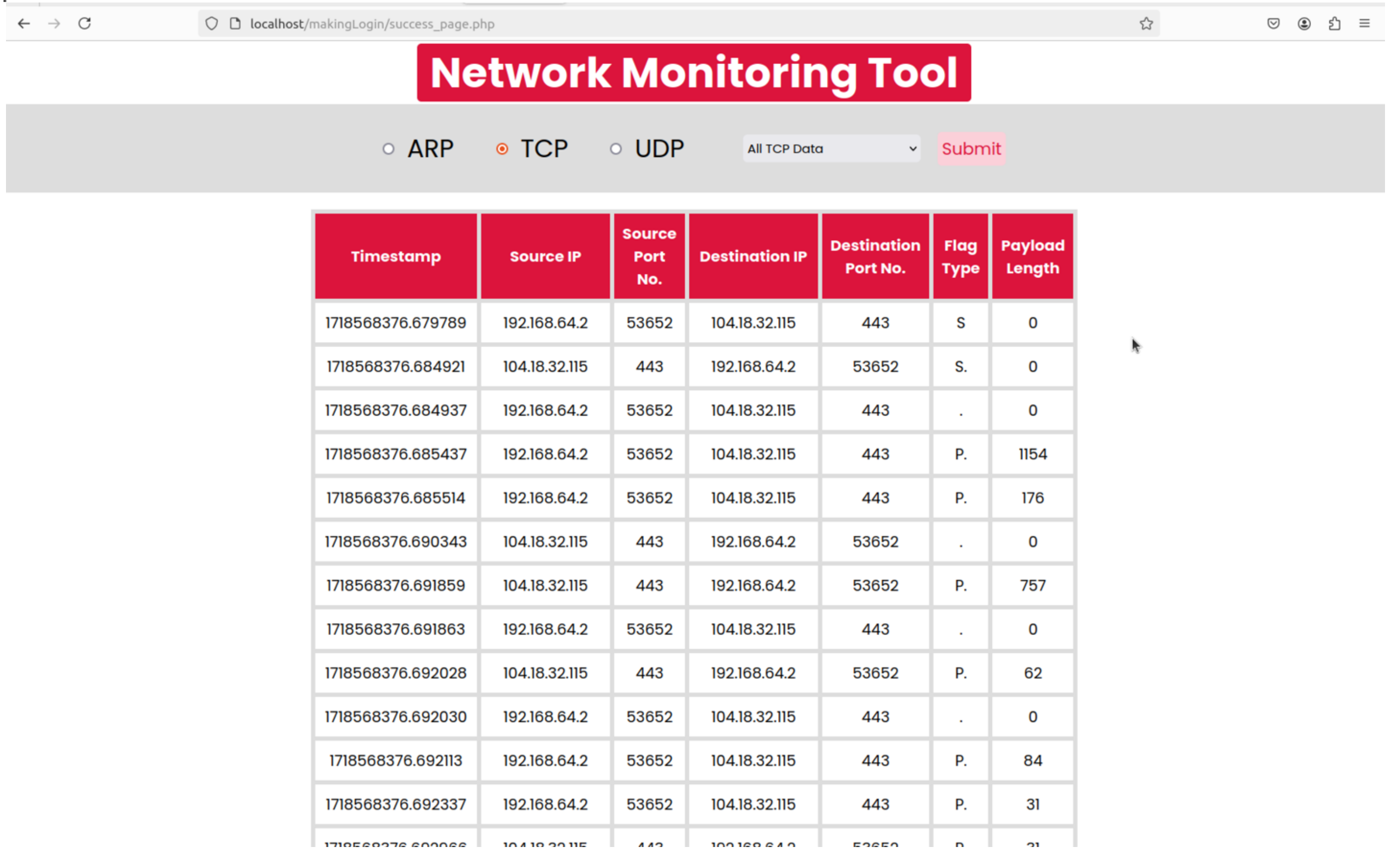
welcome to Network Monitoring Tool

NMT is a system that continuously scans the network for sluggish or malfunctioning components and alerts the administrator (via email, SMS, or other alarms) in the event of outages or other issues.

- **success_page.php:**







Timestamp	Source IP	Source Port No.	Destination IP	Destination Port No.	Flag Type	Payload Length
1718568376.679789	192.168.64.2	53652	104.18.32.115	443	S	0
1718568376.684921	104.18.32.115	443	192.168.64.2	53652	S.	0
1718568376.684937	192.168.64.2	53652	104.18.32.115	443	.	0
1718568376.685437	192.168.64.2	53652	104.18.32.115	443	P.	1154
1718568376.685514	192.168.64.2	53652	104.18.32.115	443	P.	176
1718568376.690343	104.18.32.115	443	192.168.64.2	53652	.	0
1718568376.691859	104.18.32.115	443	192.168.64.2	53652	P.	757
1718568376.691863	192.168.64.2	53652	104.18.32.115	443	.	0
1718568376.692028	104.18.32.115	443	192.168.64.2	53652	P.	62
1718568376.692030	192.168.64.2	53652	104.18.32.115	443	.	0
1718568376.692113	192.168.64.2	53652	104.18.32.115	443	P.	84
1718568376.692337	192.168.64.2	53652	104.18.32.115	443	P.	31

- The above screenshot shows that the NMT successfully displays the filtered data extracted from the captured packets.

Tools/Softwares Used

This project is built on the LAMP architecture, which stands for a stack of software typically comprised of free and open-source tools essential for running dynamic websites or servers:

- **Linux:** Serving as the operating system foundation.
- **Apache:** Functioning as the web server software.
- **MySQL:** Providing the database management system.
- **PHP:** Serving as the programming language for server-side scripting.

Together, these components form a robust environment for developing and deploying web applications and managing data-driven websites efficiently.

How does LAMP work?

LAMP, which stands for Linux, Apache, MySQL, and PHP/Perl/Python, is a widely adopted architecture specifically designed for web applications. Its design is straightforward and revolves around several key components interacting seamlessly to deliver dynamic and static content over the web.

At its core, **Linux** serves as the foundation operating system, managing HTTP connections and directing them to **Apache**, the web server. Apache handles requests by serving static content directly from the Linux kernel and routing dynamic page requests to **PHP** (or Perl/Python), where server-side scripting languages process and generate HTML content based on user interactions.

MySQL plays a crucial role in the LAMP stack by managing database queries. PHP interacts with MySQL to store and retrieve data, making it integral for storing and accessing dynamic content for web applications. This relational database management system ensures efficient data handling and storage capabilities.

Administrative tasks within the LAMP environment are typically managed through tools like **phpMyAdmin**, providing a graphical interface for database management and other server configurations. This simplifies the process of maintaining and monitoring the LAMP stack, ensuring smooth operation and scalability.

While LAMP traditionally refers to a setup on Linux, the stack's flexibility allows developers to implement similar configurations on Windows platforms, known as "WAMP," using Apache, MySQL, and PHP/Perl/Python. This versatility, combined with extensive online resources and quick setup times, has made LAMP a preferred choice for developing robust and scalable web applications across various platforms.

Where is LAMP?

The LAMP stack comprises several independent open-source projects, each supported by major commercial or nonprofit entities:

- **Linux:** Supported by Red Hat and Novell.

- **Apache:** Maintained by Covalent.

- **MySQL:** Developed by MySQL AB.

- **PHP/Python/Perl:** Supported by Zend, Python Software Foundation, and the Perl Foundation.

While assembling a LAMP stack can be complex, several vendors offer integrated solutions, including ActiveGrid, BitRock, SpikeSource, and

SourceLabs. These vendors provide pre-configured LAMP stacks underneath their application servers.

Although each project operates under distinct open-source licenses, developers are not required to release their application code as open source. However, any modifications made to components like the Linux kernel or MySQL database must be distributed under the GPL license.

Why LAMP?

LAMP, an acronym for Linux, Apache, MySQL, and PHP/Perl/Python, is highly favored for web applications due to its proven advantages in speed, cost-effectiveness, flexibility, and ease of use compared to other alternatives. Major technology players from IBM to Oracle, along with numerous startups, are actively promoting LAMP and enhancing it with enterprise-grade capabilities and management tools.

LAMP is not merely a passing trend; it has become a mainstream choice, rivaling established platforms like J2EE and .NET. Understanding its components, its widespread adoption across industries, and its operational mechanics sets the stage for diving into building your own LAMP applications. Who knows, you might discover a genuine affinity for its capabilities.

Conclusion

The Network Monitoring Tool serves as a crucial utility for centralized server systems within organizations. It plays a vital role in monitoring and safeguarding the internet facilities against misuse and unauthorized activities. By recording and analyzing all data transmissions between terminals and systems, this tool ensures comprehensive oversight and enables prompt action in cases of faults, misuse, congestion, or legal issues associated with any connected terminal.

Installed on any server machine, this utility software quickly identifies and resolves various networking issues. It provides insights into network communication patterns, data transfers, and potential security breaches, ensuring application servers remain secure and operational. Administrators benefit from real-time alerts for any abnormalities or interruptions, facilitating swift problem resolution and maintaining system integrity.

Moreover, the tool's scope extends to user assistance and administrative support. Once deployed, it conducts a range of checks to detect illegal activities or service disruptions, generating alerts that prompt administrators to address faults promptly. Comprehensive search reports are generated, offering detailed insights based on specific search criteria.

In essence, this project delivers a versatile utility tool that enhances network management efficiency, safeguards against threats, and supports seamless communication across organizational networks.

Future Scope

Data packets transmitted across a network contain headers that include the addresses of the source and destination computers. This information is accessible to anyone monitoring the network. To ensure security, data is often encrypted before transmission. Each packet includes a 'cipher key' alongside the header and payload. At the receiving end, the cipher key is used to decrypt the data, which is then decoded for use.

In future development, I plan to enhance the project to decode these ciphers. This enhancement will allow the tool not only to identify infected IPs using TOPTALKER but also to understand the nature of the infection. Additionally, protocols like telnet transmit data in plain text, which still requires decryption to be human-readable. The goal is to decode data exchanged between two IPs into a readable format, empowering a centralized server with comprehensive network control through this tool.

Appendix

- *final_script2.sh*

```
#!/bin/bash
# Capture 2000 packets from the network interface 'enp0s1' and save them in a pcap
file
tcpdump -i enp0s1 -c 2000 -w "/var/www/html/makingLogin/all_capture.pcap"

# Read the pcap file and save the packet data to a text file with timestamps
tcpdump -r "/var/www/html/makingLogin/all_capture.pcap" -nn -tt >
"/var/www/html/makingLogin/all_capture.txt"

# Filter the packets and save to respective files
grep -E " Flags| ack" "/var/www/html/makingLogin/all_capture.txt" >
"/var/www/html/makingLogin/tcp_capture.txt"
grep "UDP" "/var/www/html/makingLogin/all_capture.txt" >
"/var/www/html/makingLogin/udp_capture.txt"
grep "ARP" "/var/www/html/makingLogin/all_capture.txt" >
"/var/www/html/makingLogin/arp_capture.txt"

echo "Captures saved in /var/www/html/makingLogin"

# creating the output directory
output_dir="/var/www/html/makingLogin"

# creating input files for each protocol
arp_input_file="$output_dir/arp_capture.txt"
udp_input_file="$output_dir/udp_capture.txt"
tcp_input_file="$output_dir/tcp_capture.txt"

# creating output files for UDP
udp_timestamp_file="$output_dir/udp_timestamp.txt"
udp_source_ip_file="$output_dir/udp_sourceIP.txt"
udp_source_port_file="$output_dir/udp_sourcePort.txt"
udp_dest_ip_file="$output_dir/udp_destinationIP.txt"
udp_dest_port_file="$output_dir/udp_destinationPort.txt"
udp_payload_length_file="$output_dir/udp_payloadLength.txt"
udp_all_info_file="$output_dir/udp_all.txt"

# creating output files for ARP
arp_timestamp_file="$output_dir/arp_timestamp.txt"
arp_type_file="$output_dir/arp_type.txt"
arp_source_ip_file="$output_dir/arp_sourceIP.txt"
arp_receiver_ip_file="$output_dir/arp_receiverIP.txt"
arp_receiver_mac_file="$output_dir/arp_receiverMAC.txt"
arp_payload_length_file="$output_dir/arp_payloadLength.txt"
arp_all_info_file="$output_dir/arp_all.txt"

# creating output files for TCP
tcp_timestamp_file="$output_dir/tcp_timestamp.txt"
```

```
tcp_source_ip_file="$output_dir/tcp_sourceIP.txt"
tcp_source_port_file="$output_dir/tcp_sourcePort.txt"
tcp_dest_ip_file="$output_dir/tcp_destinationIP.txt"
tcp_dest_port_file="$output_dir/tcp_destinationPort.txt"
tcp_flag_type_file="$output_dir/tcp_flagType.txt"
tcp_payload_length_file="$output_dir/tcp_payloadLength.txt"
tcp_all_info_file="$output_dir/tcp_all.txt"

# clearing the content of the output files
> "$udp_timestamp_file"
> "$udp_source_ip_file"
> "$udp_source_port_file"
> "$udp_dest_ip_file"
> "$udp_dest_port_file"
> "$udp_payload_length_file"
> "$udp_all_info_file"

# heading for the all info file
echo -e "Timestamp\tSource IP\tSource Port No.\tDestination IP\tDestination Port
No.\tPayload Length" >> "$udp_all_info_file"

# while loop to process each line of the UDP input file
while IFS= read -r line; do
    # Extract the required fields using awk
    timestamp=$(echo "$line" | awk '{print $1}')
    source_ip=$(echo "$line" | awk '{print $3}' | cut -d'.' -f1-4)
    source_port=$(echo "$line" | awk '{print $3}' | cut -d'.' -f5 | tr -d ':')
    destination_ip=$(echo "$line" | awk '{print $5}' | cut -d'.' -f1-4)
    destination_port=$(echo "$line" | awk '{print $5}' | cut -d'.' -f5 | tr -d ':')
    payload_length=$(echo "$line" | awk '{print $NF}')

    # Save each field into the respective files
    echo "$timestamp" >> "$udp_timestamp_file"
    echo "$source_ip" >> "$udp_source_ip_file"
    echo "$source_port" >> "$udp_source_port_file"
    echo "$destination_ip" >> "$udp_dest_ip_file"
    echo "$destination_port" >> "$udp_dest_port_file"
    echo "$payload_length" >> "$udp_payload_length_file"

    # Save all information into the all_info_file
    echo -e
"$timestamp\t$source_ip\t$source_port\t$destination_ip\t$destination_port\t$payload
_length" >> "$udp_all_info_file"
done < "$udp_input_file"

echo "UDP Data extracted to files successfully."

# clearing the content of the output files
> "$arp_timestamp_file"
> "$arp_type_file"
> "$arp_source_ip_file"
> "$arp_receiver_ip_file"
> "$arp_receiver_mac_file"
> "$arp_payload_length_file"
> "$arp_all_info_file"
```

```
# heading for the all info file
echo -e "Timestamp\tType\tSource IP\tReceiver IP\tReceiver MAC\tPayload Length" >>
"$arp_all_info_file"

# while loop to process each line of the ARP input file
while IFS= read -r line; do
    timestamp=$(echo "$line" | awk '{print $1}')
    type=$(echo "$line" | awk '{print $3}' | tr -d ',')
    source_ip=$(echo "$line" | awk -F'tell ' '{print $2}' | awk '{print $1}' | tr -
d ',')
    receiver_ip=$(echo "$line" | awk -F'who-has ' '{print $2}' | awk '{print $1}' |
tr -d ',')
    receiver_mac=$(echo "$line" | awk -F'(' '{print $2}' | awk -F')' '{print $1}' |
tr -d ',')
    payload_length=$(echo "$line" | awk '{print $NF}')

    # Handle receiver IP and MAC for Replies
    if [[ "$type" == "Reply" ]]; then
        receiver_ip=$(echo "$line" | awk -F'Reply ' '{print $2}' | awk '{print
$1}')
        receiver_mac=$(echo "$line" | awk -F'is-at ' '{print $2}' | awk -F','
'{print $1}')
    fi

    # Save each field into the respective files
    echo "$timestamp" >> "$arp_timestamp_file"
    echo "$type" >> "$arp_type_file"
    echo "$source_ip" >> "$arp_source_ip_file"
    echo "$receiver_ip" >> "$arp_receiver_ip_file"
    echo "$receiver_mac" >> "$arp_receiver_mac_file"
    echo "$payload_length" >> "$arp_payload_length_file"

    echo -e
"$timestamp\t$type\t$source_ip\t$receiver_ip\t$receiver_mac\t$payload_length" >>
"$arp_all_info_file"
done < "$arp_input_file"

echo "ARP Data extracted to files successfully."


# clearing the content of the output files
> "$tcp_timestamp_file"
> "$tcp_source_ip_file"
> "$tcp_source_port_file"
> "$tcp_dest_ip_file"
> "$tcp_dest_port_file"
> "$tcp_flag_type_file"
> "$tcp_payload_length_file"
> "$tcp_all_info_file"

# heading for the all info file
echo -e "Timestamp\tSource IP\tSource Port No.\tDestination IP\tDestination Port
No.\tFlag Type\tPayload Length" >> "$tcp_all_info_file"
```

```
# while loop will process each line of the input TCP file
while IFS= read -r line; do

    timestamp=$(echo "$line" | awk '{print $1}')
    source_ip=$(echo "$line" | awk '{print $3}' | cut -d'.' -f1-4)
    source_port=$(echo "$line" | awk '{print $3}' | cut -d'.' -f5 | tr -d ':')
    destination_ip=$(echo "$line" | awk '{print $5}' | cut -d'.' -f1-4)
    destination_port=$(echo "$line" | awk '{print $5}' | cut -d'.' -f5 | tr -d ':')
    flag_type=$(echo "$line" | awk '{print $6}' | tr -d '[]')
    payload_length=$(echo "$line" | awk '{print $NF}')

      # Handle different format when the flag type is "Flags"
      if [[ "$flag_type" == "Flags" ]]; then
            flag_type=$(echo "$line" | awk '{print $7}' | tr -d '[],')
      fi

    # Save each field into the respective files
    echo "$timestamp" >> "$tcp_timestamp_file"
    echo "$source_ip" >> "$tcp_source_ip_file"
    echo "$source_port" >> "$tcp_source_port_file"
    echo "$destination_ip" >> "$tcp_dest_ip_file"
    echo "$destination_port" >> "$tcp_dest_port_file"
    echo "$flag_type" >> "$tcp_flag_type_file"
    echo "$payload_length" >> "$tcp_payload_length_file"

    # Save all information into the all_info_file
    echo -e "$timestamp\t$source_ip\t$source_port
\t$destination_ip\t$destination_port\t$flag_type\t$payload_length" >>
"$tcp_all_info_file"
done < "$tcp_input_file"

echo "TCP Data extracted to files successfully."
```

- ***style.css:***

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0
,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;
1,700;1,800;1,900&display=swap');

*{
    font-family: 'Poppins', sans-serif;
    margin: 0; padding: 0;
    box-sizing: border-box;
    outline: none; border:none;
    text-decoration: none;
}

.top-bar {
      min-height: 5vh;
      display: flex;
```

```
            align-items: center;
            margin-bottom: 20px;
            justify-content: center;
            padding: 20px;
            background: #dddddd;
       }
             .top-bar .radio-buttons {
                  display: flex;
                  margin-right: 10px;

              }
              .top-bar .radio-buttons label {
                  margin-right: 40px;
                  font-size: 30px;
                  color: #crimson;
                  padding: 10px;
              }
              .top-bar .dropdown-menu {
                  margin-left: 10px;
                  font-size: 15px;
                  padding: 5px;
                  border-radius: 5px;
              }
              .top-bar .dropdown-menu:hover{
                  background: #fbd0d9;
                  color: black;
              }
              .top-bar .radio-buttons input[type="radio"] {
                  margin-right: 5px;
              }
            .top-bar .btn-submit-button {
                  margin-left: 20px;
                  font-size: 20px;
                  padding: 5px;
                  border-radius: 5px;
                  background: #fbd0d9;
                color: crimson;
                  text-transform: capitalize;
                  cursor: pointer;
              }
              .top-bar .btn-submit-button:hover {
                  color: #fff;
                  background: crimson;
              }


      .successhead h1{
            font-size: 50px;
```

```
        color: #333;
        text-align: center;
    }

    .successhead h1 span{
          background: crimson;
        color: #fff;
        border-radius: 5px;
        padding:0 15px;
    }


    .some-block{

        display: flex;
        align-items: center;
        justify-content: center;
        padding: 20px;
        padding-bottom: 10px;
        padding-top: 10px;
        background: #eee;

    }

    .some-block .btn{
        display: inline-block;
        padding: 10px 30px;
        font-size: 20px;
        background: #333;
        color: #fff;
        margin:0 5px;
        text-transform: capitalize;
        border-radius: 5px;
    }

    .some-block .btn:hover{
        background: crimson;
    }

    .container{
        min-height: 100vh;
        display: flex;
        align-items: center;
        justify-content: center;
        padding: 20px;
        padding-bottom: 60px;
        background: #eee;
    }
```

```
.container .content{
    text-align: center;
}

.container .content h3{
    font-size: 30px;
    color: #333;
}

.container .content h3 span{
    background: crimson;
    color: #fff;
    border-radius: 5px;
    padding:0 15px;
}

.container .content h1{
    font-size: 50px;
    color: #333;
}

.container .content h1 span{
    background: crimson;
    color: #fff;
    border-radius: 5px;
    padding:0 15px;
}

.container .content p{
      padding: 20px;
    font-size: 25px;
    margin-bottom: 20px;
}

.container .content .btn{
    display: inline-block;
    padding: 10px 30px;
    font-size: 20px;
    background: #333;
    color: #fff;
    margin:0 5px;
    text-transform: capitalize;
    border-radius: 5px;
}

.container .content .btn:hover{
    background: crimson;
```

```
}

.loading-overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    z-index: 9999;
    display: none;
}

.loading-spinner {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    border: 16px solid crimson;
    border-top: 16px solid gray;
    border-radius: 50%;
    width: 120px;
    height: 120px;
    animation: spin 2s linear infinite;
}

@keyframes spin {
    0% {
        transform: rotate(0deg);
    }

    100% {
        transform: rotate(360deg);
    }
}

/* .formcontainer{
    min-height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    padding: 20px;
    padding-bottom: 60px;
    background: #eee;
} */

.container form{
    padding: 20px;
```

```
        border-radius: 5px;
        box-shadow:0 5px 10px rgba(0,0,0,.1);
        background: #fff;
        text-align: center;
    }

    .container form h3{
        font-size: 30px;
        text-transform: uppercase;
        margin-bottom: 10px;
        color: #333;
    }

    .container form input, .container form select{
        width: 100%;
        padding: 10px 15px;
        font-size: 17px;
        margin: 8px 0;
        background:#eee;
        border-radius: 5px;
    }

    .container form select option{
        background: #fff;
    }

    .container form .form-btn{
        background: #fbd0d9;
        color: crimson;
        text-transform: capitalize;
        font-size: 20px;
        cursor: pointer;
    }

    .container form .form-btn:hover{
        background: crimson;
        color:#fff;
    }

    .container form p{
        margin-top: 10px;
        font-size: 20px;
        color: #333;
    }

    .container form p a{
        color: crimson;
    }
```

```
.container form .error-msg{
      margin: 10px 0;
      display: block;
      background: crimson;
      color: #fff;
      border-radius: 5px;
      font-size: 20px;
      padding: 10px;
}

table {
    width: 50%;
    border-collapse: collapse;
    margin: auto;
    font-size: 18px;
    text-align: center;
    margin-bottom: 20px;
}
th, td {
    border: 5px solid #dddddd;
    padding: 10px;
}
th {
    color: white;
    background-color: crimson;
}
```

- ***config.php:***

```
<?php

$conn = mysqli_connect('localhost','devansh','1234','user_lgnDB');

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

?>
```

- ***register_form.php:***

```
<?php
    @include 'config.php';
    $error = array(); // Initialize error array

    if (isset($_POST['submit'])){
        $name = mysqli_real_escape_string($conn, $_POST['name']);
        $email = mysqli_real_escape_string($conn, $_POST['email']);
        $pass = md5($_POST['password']);
        $cpass = md5($_POST['cpassword']);
        $user_type = $_POST['user_type'];

        $select = " SELECT * FROM user_form WHERE email = '$email' &&
password = '$pass' ";

        $result = mysqli_query($conn, $select);

        if(mysqli_num_rows($result) > 0) {
            $error[] = 'user already exists!';
        } else {
            if($pass != $cpass){
                $error[] = 'passwords do not match!';
            } else {
                $insert = "INSERT INTO user_form(name, email, password,
user_type) VALUES('$name','$email','$pass','$user_type')";
                mysqli_query($conn, $insert);
                header('location:login_form.php');
            }
        }
    }
?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>register form</title>

    <link rel="stylesheet" href="style.css">
</head>
<body>

    <div class="container">
            <div class= "content">
        <form action="" method="post">
```

```
            <h1>Welcome to <span>Network Monitoring Tool</span></h1>
                <h3>registration form</h3>
                <?php
                if(isset($error)){
                      foreach($error as $err){
                            echo '<span class="error-msg">'.$err.'</span>';
                      }
                }
          ?>

                <input type="text" name="name" required placeholder="enter your
name">
                <input type="email" name="email" required placeholder="enter your
email">
                <input type="password" name="password" required
placeholder="enter your password">
                <input type="password" name="cpassword" required
placeholder="confirm your password">
                <select name="user_type">
                    <option value="user">user</option>
                    <option value="admin">admin</option>
                </select>
                <input type="submit" name="submit" value="register" class="form-
btn">
                <p>already have an account? <a href="login_form.php">login
now</a></p>
            </form>
    </div>
    </div>


</body>
</html>
```

- ***login_form.php:***

```
<?php
    @include 'config.php';

    $error = array();

    session_start();

    if (isset($_POST['submit'])){
        $name = mysqli_real_escape_string($conn, $_POST['name']);
        $email = mysqli_real_escape_string($conn, $_POST['email']);
        $pass = md5($_POST['password']);
        $cpass = md5($_POST['cpassword']);
        $user_type = $_POST['user_type'];

        $select = " SELECT * FROM user_form WHERE email = '$email' && password =
'$pass' ";

        $result = mysqli_query($conn, $select);

        if(mysqli_num_rows($result) > 0) {

           $row = mysqli_fetch_array($result);

           if($row['user_type'] == 'admin'){
            $_SESSION['admin_name'] = $row['name'];
            header('location:admin_page.php');
           }elseif($row['user_type'] == 'user'){
            $_SESSION['user_name'] = $row['name'];
            header('location:user_page.php');
           }
        }else{
            $error[] = 'incorrect email or password!';
        }
    };
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>login form</title>

    <link rel="stylesheet" href="style.css">
</head>
<body>

    <div class="container">
            <div class = "content">
        <form action="" method="post">

        <h1>Welcome to <span>Network Monitoring Tool</span></h1>
            <h3>login form</h3>
            <?php
            if(isset($error)){
                  foreach($error as $err){
                  echo '<span class="error-msg">'.$err.'</span>';
                  }
            }
      ?>
```

```
            <input type="email" name="email" required placeholder="enter your
email">
            <input type="password" name="password" required placeholder="enter
your password">
            <input type="submit" name="submit" value="login" class="form-btn">
            <p>don't have an account? <a href="register_form.php">register
now</a></p>
        </form>
        </div>
    </div>


</body>
</html>
```

- ***user_page.php:***

```
<?php

      @include 'config.php';

      session_start();

      if(!isset($_SESSION['user_name'])){
            header('location:login_form.php');
      }

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>user page</title>

    <link rel="stylesheet" href="style.css">
</head>
<body>
      <div class="loading-overlay" id="loading-overlay">
            <div class="loading-spinner"></div>
      </div>

    <div class="container">
        <div class="content">
        <form>
            <h3>hello, <span><?php echo $_SESSION['user_name'] ?></span></h3>
            <h1>welcome to <span>Network Monitoring Tool</span></h1>
            <p>NMT is a system that continuously scans the network for
sluggish or malfunctioning components and alerts the administrator (via
email, SMS, or other alarms) in the event of outages or other issues. </p>
            <button id="run-script-btn" class="btn">Run Script</button>

            <a href="logout.php" class="btn">logout</a>

        </form>
        </div>
    </div>

<script>

        document.getElementById('run-script-btn').addEventListener('click',
function() {
        // Show loading animation
        document.getElementById('loading-overlay').style.display = 'block';

        fetch('run_script.php')
            .then(response => response.text())
            .then(data => {
                if (data.includes('success')) {
                    // Redirect to success page
                    window.location.href = 'success_page.php';
                } else {
                    // Display error message
                    alert(data);
                }
            })
            .catch(error => console.error('Error:', error))
            .finally(() => {
                // Hide loading animation when script finishes executing
                document.getElementById('loading-overlay').style.display =
'none';
            });
    });

    </script>


</body>
</html>
```

- ***admin_page.php:***

```
<?php

      @include 'config.php';

      session_start();

      if(!isset($_SESSION['admin_name'])){
            header('location:login_form.php');
      }

?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>admin page</title>

    <link rel="stylesheet" href="style.css">
</head>
<body>
      <div class="loading-overlay" id="loading-overlay">
            <div class="loading-spinner"></div>
      </div>


    <div class="container">
        <div class="content">
        <form>
            <h3>hello, <span><?php echo $_SESSION['admin_name']
?></span></h3>
            <h1>welcome to <span>Network Monitoring Tool</span></h1>
            <p>NMT is a system that continuously scans the network for
sluggish or malfunctioning components and alerts the administrator (via
email, SMS, or other alarms) in the event of outages or other issues. </p>

            <!-- Button to run the Bash script -->
            <button id="run-script-btn" class="btn">Run Script</button>
            <a href="logout.php" class="btn">logout</a>
      </form>
        </div>
    </div>

    <script>
```

```
        document.getElementById('run-script-btn').addEventListener('click',
function() {
        // Show loading animation
        document.getElementById('loading-overlay').style.display = 'block';

        fetch('run_script.php')
            .then(response => response.text())
            .then(data => {
                if (data.includes('success')) {
                    // Redirect to success page
                    window.location.href = 'success_page.php';
                } else {
                    // Display error message
                    alert(data);
                }
            })
            .catch(error => console.error('Error:', error))
            .finally(() => {
                // Hide loading animation when script finishes executing
                document.getElementById('loading-overlay').style.display =
'none';
            });
    });

    </script>
</body>
</html>
```

- ***success_page.php:***

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>success page</title>
    <link rel="stylesheet" href="style.css">

</head>
<body>
      <div class="successhead">
      <h1><span>Network Monitoring Tool</span></h1>
      </div>
      <form id="data-form">

   <div class="top-bar">

            <div class="radio-buttons">

                    <input type="radio" id="arpradio" name="protocol"
value="ARP" onclick="populateDropdown('ARP')"><label
for="arpradio">ARP</label>

                    <input type="radio" id="tcpradio" name="protocol"
value="TCP" onclick="populateDropdown('TCP')"><label
for="tcpradio">TCP</label>

                    <input type="radio" id="udpradio" name="protocol"
value="UDP" onclick="populateDropdown('UDP')"><label
for="udpradio">UDP</label>

                </div>

              <select id="dropdown-menu" class="dropdown-menu" name="data-
type">
                  <option value="">Select Data Type</option>
              </select>
            <button type="submit" class="btn-submit-button">Submit</button>

                  </div>
        </form>
      <div id="table-container"></div>

            <div class="some-block">

            <a href="admin_page.php" class="btn">Back to Admin Page</a>
        </div>

    <script>



        document.addEventListener('DOMContentLoaded', function() {
            const protocolRadios =
document.querySelectorAll('input[name="protocol"]');
            const dropdown = document.getElementById('dropdown-menu');
            const dataOptions = {
                ARP: ["Timestamp", "Source IP", "Receiver IP", "Receiver
MAC", "ARP Type", "Payload Length", "All ARP Data"],
                TCP: ["Timestamp", "Source IP", "Source Port Number",
"Destination IP", "Destination Port Number", "Flag Type", "Payload Length",
"All TCP Data"],
                UDP: ["Timestamp", "Source IP", "Source Port Number",
"Destination IP", "Destination Port Number", "Payload Length", "All UDP
Data"]
            };

            protocolRadios.forEach(radio => {
                radio.addEventListener('change', function() {
                    populateDropdown(this.value);
                });
            });

            function populateDropdown(protocol) {
                dropdown.innerHTML = ''; // Clear existing options

                const defaultOption = document.createElement('option');
                defaultOption.value = "";
                defaultOption.textContent = "Select Data Type";
                dropdown.appendChild(defaultOption);

                if (dataOptions[protocol]) {
                    dataOptions[protocol].forEach(option => {
                        const opt = document.createElement('option');
                        opt.value = option;
                        opt.textContent = option;
                        dropdown.appendChild(opt);
                    });
                }
            }

            document.getElementById('data-form').addEventListener('submit',
function(event) {
                event.preventDefault();

                const formData = new FormData(this);

                fetch('fetch_fileData.php', {
                    method: 'POST',
                    body: formData
                })
                .then(response => response.text())
                .then(data => {
                    document.getElementById('table-container').innerHTML =
data;
                })
                .catch(error => console.error('Error:', error));
                });
            });
        </script>
    </body>
    </html>
```

- ***run_script.php:***

```
<?php
session_start();

// Define the path to your bash script
$script = '/var/www/html/makingLogin/final_script2.sh';

// Execute the script and capture the output
$output = shell_exec($script);

// Add a success indicator to the output
if ($output) {
   echo "success: " . $output;
} else {
    echo "failure: Script did not run successfully.";
}
?>
```

- ***fetch_fileData.php:***

```
<?php
function getFileName($protocol, $dataType) {
    $fileMap = [
        'ARP' => [
            'Timestamp' => 'arp_timestamp.txt',
            'Source IP' => 'arp_sourceIP.txt',
            'Receiver IP' => 'arp_receiverIP.txt',
            'Receiver MAC' => 'arp_receiverMAC.txt',
            'ARP Type' => 'arp_type.txt',
            'Payload Length' => 'arp_payloadLength.txt',
            'All ARP Data' => 'arp_all.txt'
        ],
        'TCP' => [
            'Timestamp' => 'tcp_timestamp.txt',
            'Source IP' => 'tcp_sourceIP.txt',
            'Source Port Number' => 'tcp_sourcePort.txt',
                'Destination IP' => 'tcp_destinationIP.txt',
                'Destination Port Number' => 'tcp_destinationPort.txt',
                'Flag Type' => 'tcp_flagType.txt',
                'Payload Length' => 'tcp_payloadLength.txt',
                'All TCP Data' => 'tcp_all.txt'
            ],
            'UDP' => [
                'Timestamp' => 'udp_timestamp.txt',
                'Source IP' => 'udp_sourceIP.txt',
                'Source Port Number' => 'udp_sourcePort.txt',
                'Destination IP' => 'udp_destinationIP.txt',
                'Destination Port Number' => 'udp_destinationPort.txt',
                'Payload Length' => 'udp_payloadLength.txt',
                'All UDP Data' => 'udp_all.txt'
            ]
        ];

        return $fileMap[$protocol][$dataType] ?? null;
    }

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $protocol = $_POST['protocol'];
        $dataType = $_POST['data-type'];

        //error_log("Protocol: " . $protocol);
        //error_log("Data Type: " . $dataType);
        //echo "Protocol: " . htmlspecialchars($protocol) . "<br>";
        //echo "Data Type: " . htmlspecialchars($dataType) . "<br>";

        if ($protocol && $dataType) {
            $fileName = getFileName($protocol, $dataType);

            if ($fileName) {
                $filePath = "/var/www/html/makingLogin/$fileName";
            }

            if (file_exists($filePath)) {
                $data = file($filePath, FILE_IGNORE_NEW_LINES);
            } else {
                echo "<p>File not found.</p>";
                exit;
            }
        } else {
            echo "<p>Invalid input.</p>";
            exit;
        }
    } else {
        echo "<p>Invalid request method.</p>";
    exit;
}


echo '<table border="1">';
if ($dataType === 'All ARP Data' || $dataType === 'All TCP Data' || $dataType
=== 'All UDP Data') {
    $headers = explode("\t", array_shift($data));
    echo '<tr>';
    foreach ($headers as $header) {
        echo '<th>' . htmlspecialchars($header) . '</th>';
    }
    echo '</tr>';

    foreach ($data as $line) {
        $columns = explode("\t", $line);
        echo '<tr>';
        foreach ($columns as $column) {
            echo '<td>' . htmlspecialchars($column) . '</td>';
        }
        echo '</tr>';
    }
} else {
    echo '<tr><th>' . htmlspecialchars($dataType) . '</th></tr>';
    foreach ($data as $line) {
        echo '<tr><td>' . htmlspecialchars($line) . '</td></tr>';
    }
}
echo '</table>';
?>
```

- ***logout.php:***

```
<?php

      @include 'config.php';

      session_start();
      session_unset();
      session_destroy();

      header('location:login_form.php');

?>
```