**PROJECT: NETWORK SCANNER USING SOCKETS**

## ****Project Overview****

The network scanner sequentially checks specified IP addresses and ports, attempting to establish a connection. The scanner reports whether each port is open, closed, or unresponsive. The project leverages Linux’s socket programming capabilities to perform the scanning operation and is designed to be lightweight and efficient, making it suitable for small-scale network scans.

### ****Introduction:****

This project implements a simple network scanner using sockets in C on a Linux platform. The purpose of the project is to scan a specified range of IP addresses and ports to check if they are open or closed. This tool can be used for network diagnostics and security auditing, helping to identify potential vulnerabilities or available services within a network.

## ****Project Scope****

The scope of this project includes:

* Scanning a range of IP addresses within a local network.
* Identifying active devices and their open ports.
* Supporting both TCP and UDP scanning.
* Providing output in a user-friendly format.
* Implementing the project on a Linux platform, specifically Ubuntu.

## ****Application Tools Used****

* **Programming Languages:** C, C++
* **Operating System:** Linux (Ubuntu)
* **Libraries:** POSIX Sockets API, Standard C Library
* **Development Tools:** GCC Compiler, GDB Debugger

## ****Modules Worked On****

### ****IP Range Scanning****

This module handles the specification of IP address ranges and performs scans on each IP to determine if it is active on the network.

### ****Port Scanning****

This module scans specified ports on active IP addresses to check if they are open or closed. It supports both TCP and UDP scans.

### ****Reporting****

This module collects scan results and formats them into a readable output, displaying active hosts and open ports.

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <string.h>

#include <errno.h>

#include <fcntl.h>

#define TIMEOUT 1

int ports[] = {80, 22, 109, 9100, 111};

int num\_ports = sizeof(ports) / sizeof(ports[0]);

const char\* ip\_addresses[] = {

"172.20.0.35",

"172.20.0.60",

"172.20.0.56"

};

int num\_ips = sizeof(ip\_addresses) / sizeof(ip\_addresses[0]);

void scan\_ip(const char\* ip) {

for (int i = 0; i < num\_ports; i++) {

int sockfd;

struct sockaddr\_in target;

struct timeval timeout;

fd\_set fdset;

printf("Scanning IP: %s, Port: %d\n", ip, ports[i]);

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0) {

perror("Socket creation failed");

continue;

}

fcntl(sockfd, F\_SETFL, O\_NONBLOCK);

target.sin\_family = AF\_INET;

target.sin\_port = htons(ports[i]);

inet\_pton(AF\_INET, ip, &target.sin\_addr);

connect(sockfd, (struct sockaddr \*)&target, sizeof(target));

timeout.tv\_sec = TIMEOUT;

timeout.tv\_usec = 0;

FD\_ZERO(&fdset);

FD\_SET(sockfd, &fdset);

if (select(sockfd + 1, NULL, &fdset, NULL, &timeout) > 0) {

int so\_error;

socklen\_t len = sizeof so\_error;

getsockopt(sockfd, SOL\_SOCKET, SO\_ERROR, &so\_error, &len);

if (so\_error == 0) {

printf("Port %d open on %s\n", ports[i], ip);

} else {

printf("Port %d closed on %s\n", ports[i], ip);

}

} else {

printf("No response from %s, Port %d\n", ip, ports[i]);

}

close(sockfd);

}

}

int main() {

for (int i = 0; i < num\_ips; i++) {

pid\_t pid = fork();

if (pid == 0) {

scan\_ip(ip\_addresses[i]);

exit(0);

} else if (pid > 0) {

} else {

perror("Fork failed");

exit(EXIT\_FAILURE);

}

}

while (wait(NULL) > 0);

return 0;

}

**Inputs:**

The three inputs are:-

**1.IP Address Range:**

* **Start IP:** The starting IP address to begin the scan.
* **End IP:** The ending IP address to stop the scan.
* Example: 172.20.0.59 to 172.20.0.61.

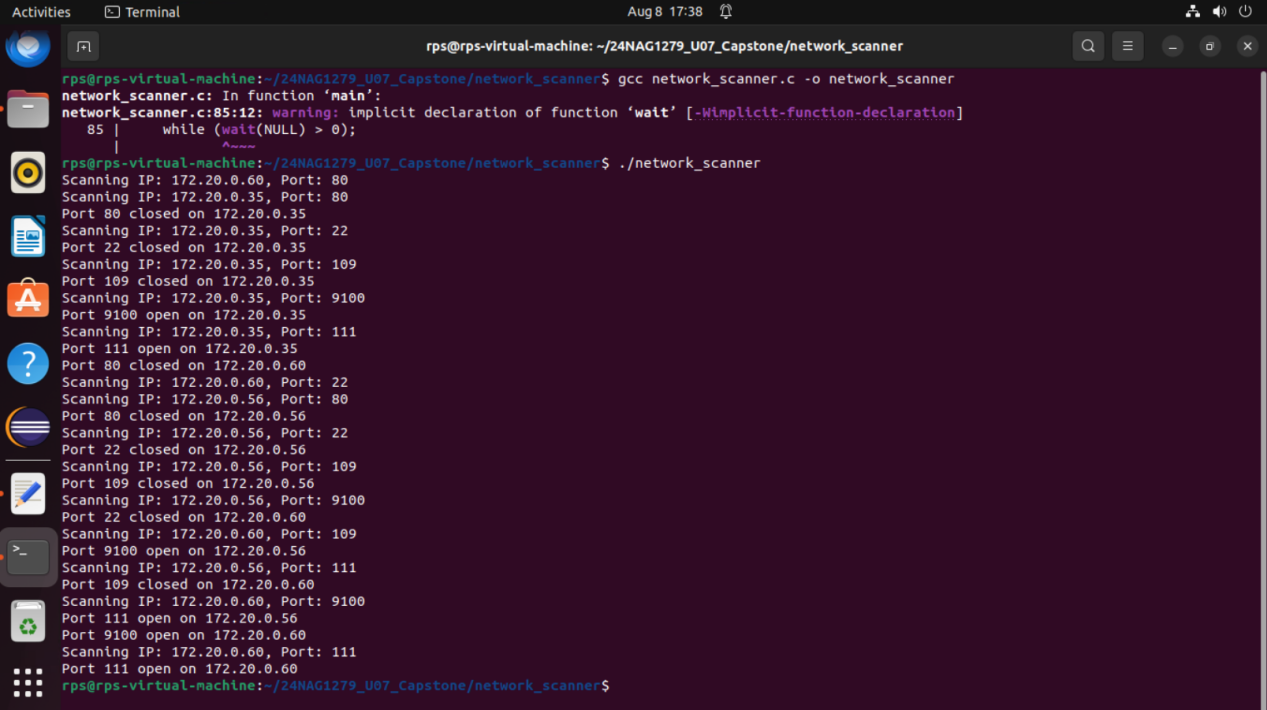
**2.Port Numbers:**

* A list of port numbers that the scanner will attempt to connect to.
* Example: 80, 22, 443, 9100, 111.

**3.Timeout Duration:**

* The duration in seconds for how long the program should wait for a response from each port.
* Example: 1 second.

**Output:**



## ****Conclusion****

The Network Scanner project successfully demonstrates the application of socket programming in network analysis. It provides a foundation for further development and customization according to specific network requirements.