PORT SCANNER

Project Report

P - 1161

Submitted in partial fulfillment of the
Requirement for the award of the degree of the

**Bachelor of Information Technology of
Bharathiar University, Coimbatore.**

Submitted by

M. Devi Thulasimani
0027S0072

U. Kamalaja
0027S0082

Under the guidance of

Ms. P. Sudha B.E.
Lecturer

DEPARTMENT OF INFORMATION TECHNOLOGY

**KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE – 641006.**

MARCH 2004.

DEPARTMENT OF INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE

This is to certify that the project entitled

PORT SCANNER

Is done by

M. Devi Thulasimani
0027S0072

U. Kamalaja
0027S0082

And submitted in partial fulfillment of the
Requirement for the award of the degree of the

Bachelor of Information Technology of
Bharathiar University, Coimbatore.

Professor & Head Of the Department
(Dr.S.THANGASAMY)

Project Guide
(Ms. P. SUDHA)

Certified that the candidates were examined by us in the project work
Viva voce examination held on _____.

Internal Examiner

External Examiner

Declaration

We,

M. Devi Thulasimani	**0027S0072**
U. Kamalaja	**0027S0082**

declare that the project entitled **"Port Scanner"** , is done by us and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted on the partial fulfillment of the requirement for all awards of the degree of Bachelor Of Information Technology of Bharathiar University.

Place: Coimbatore. **[M. Devi Thulasimani]**

Date : 17-03-04. **[U. Kamalaja]**

Project Guided by

P. Sudha

Ms. P. Sudha B. E.

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with. We express our deep gratitude to our Principal **Dr.K.K.Padmanabhan** for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head Of the Department, **Dr.S.Thangasamy** whose consistent support and enthusiastic involvement helped us a great deal.

We are greatly indebted to our beloved guide **Ms. P. Sudha B.E.,** Lecturer, Department of Information Technology for her excellent guidance and timely support during the course of this project. As a token of our esteem and gratitude, we would like to honour her for her assistance towards this cause.

We also thank our Project Coordinator **Mrs. S. Devaki M.S.,** and our beloved Class Advisor Ms. P. Sudha B.E., for their invaluable assistance.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Information Technology.

We feel proud to pay our respectful thanks to our **Parents** for their enthusiasm and encouragement and also we thank our **friends** who have associated themselves to bring out this project successfully.

SYNOPSIS

Port Scanning is a testing of a computer connected to a network for open TCP and/or UDP ports. Port Scanner accurately determines active services on the appropriate host using TCP/UDP port interrogations. Systems administrators use port scanners to determine what TCP/UDP services are available on a system.

A cardinal rule of a system security is to disable any service that the system isn't using because any open TCP/UDP service offers intruders a possible entry into your system. Thus, a port scanner can be used to ensure that only the desired TCP/UDP services are running on a system. This information is rather critical for developing and/or verifying the security policies.

Port Scanning is pretty much a general practice technique when gathering information about a network. Also the Operating System used by the system plays a major role in its security.

Hence, our project aims at developing software for scanning the ports of the remote system and to identify the Operating System it is running on.

CONTENTS

1. INTRODUCTION

1.1 Existing System and Limitations:

System administrators to gather information about the network perform Port scanning. To manually perform Port Scanning Telnet is launched and manually Telnet to each Port. When you telnet to a port of a remote host, a full three-way handshake takes place, which means a complete TCP connection opens. This indeed is not the most convenient method of getting a list of open ports on a remote system. Such manual port scanning is highly and easily detectable.

1.2 Proposed System and Advantages:

Types:

Almost all port scans are based on client sending a packet to the target port of the remote system. There are two types of port scanning:

- TCP Port Scanning
- UDP Port Scanning

Given the IP Address and range of ports we test a successful connect to the given port. There are three steps in port scanning. They are:

- Create a socket
- Attempt to connect to the port of the remote system
- Check for success/failure of connect

If the connect step succeeds, then the port is opened. If the third step fails, then the port is closed for any incoming connections.

Knowing the Operating System of the remote system is necessary to know the security level of that system. Detection of the Operating System could be done in many ways. We have done this by pinging the remote host. From the TTL value of the ping response we could identify the Operating system of the remote host.

Socket Programming:

Socket programming is used in the project to connect to the remote host by creating sockets. A socket is a structure maintained by a BSD-UNIX system to handle network connections. A socket is an end point of communication. There are routines in C to create sockets, connect them to the peer, to transmit and receive via those sockets.

Advantages:

- No special privileges are needed.
- Speed is increased by scanning many ports parallel.
- Pinging is done first to check whether the host is up or not. Hence unnecessary wastage of time in scanning a down host is avoided.

2. SOFTWARE REQUIREMENT ANALYSIS

System study is an activity that encompasses most of the tasks that we have collectively called computer system engineering. System study is conducted with the following objectives.

- Identify the needs.
- Evaluate the system concept for feasibility.
- Perform economic and technical analysis.
- Allocate function to hardware, software, people and other system elements.
- Create a system definition that forms the foundation for all subsequent engineering works.

2.1 Product Definition:

Port Scanning is often needed for security reasons. If we know what are the ports are opened we could know that what are the services are running on the system. Also, this information is needed to form and verify the security policies. Security levels of the system also depend on the Operating System it is running on. This software package performs Port scanning and OS Detection.

2.2Project Plan:

This project entitled **"Port Scanner"** is to develop a software package to perform port scanning to find the active ports and the operating system detection which determines the security level.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Purpose:

The primary purpose of the Software Requirement Specification (SRS) is to document the previously agreed functionality, attributes and the performance of the "Port Scanning". This specification is the primary document upon which all of the subsequent design, source code and test plan will be based.

3.2 Scope:

The scope of the document is to describe the requirements definition effort. The SRS is limited to description of the port scanning and Operating System detection techniques.

3.3 Product Overview and Summary:

Our product provides a reliable, robust and efficient means of scanning the ports of the remote system for security purpose. Our product basically deals with the security of the remote system by knowing what are the active ports and the Operating System it is running on. The product is developed using the concept of interrogating the TCP/IP ports and services (Telnet or FTP, for example) and record the response from the target. The Operating System detection is implemented based on the TTL value.

Port Scanner is given a host and a range of port numbers. It connects to the host on each port, and displays all the ports it found a connection on. It also identifies the Operating System the remote system is running on

3.4 Development and Operating Environment:

The development environment gives the minimum hardware and software requirements.

➢ Hardware Specification

- Processor Pentium III
- RAM 64 MB
- Cache 128MB
- Hard Disk 10 GB
- Floppy Drive 1.44 FDD
- Monitor 14" Monitor

➢ Software Specification

- Operating System Linux
- Platform C

3.5 Functional Specifications:

The descriptions of the modules are as follows:

➢ Scanning Module:

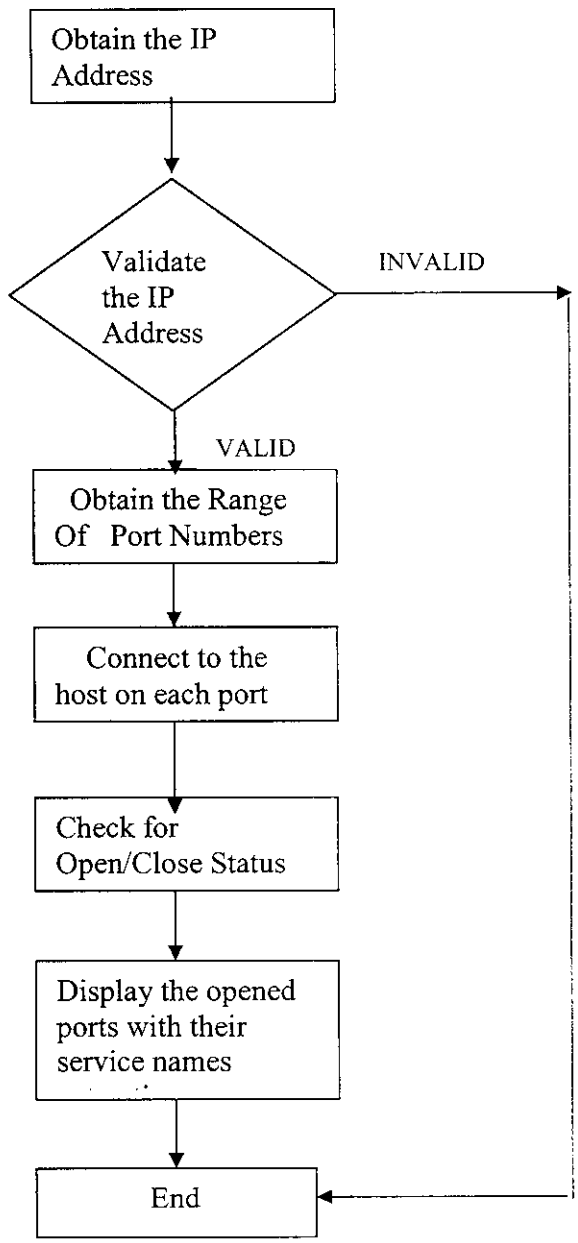
IP Address of the remote system is obtained from the user. This IP Address is checked for its validation as IP Address follows the unique standard notation. Then the ranges of port numbers to be scanned are obtained. This module connects to the host on each port, and displays all the ports, which are opened.

➢ OS Detection Module:

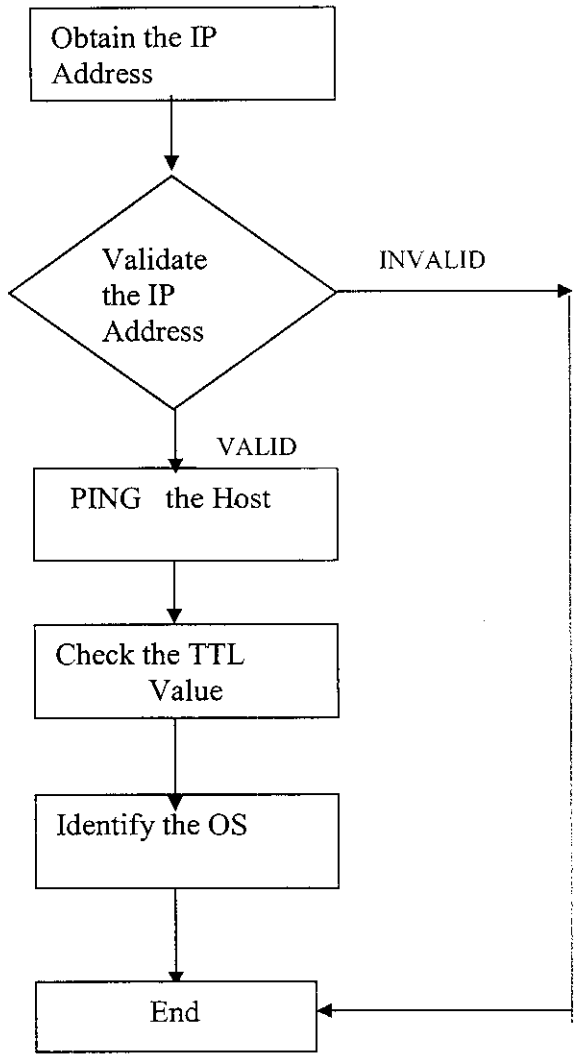
IP Address obtained from the end user is checked for its validity. The remote system identified by this IP Address is pinged. The TTL value from the ping response

3.6. Functional Flowchart:

3.6.1. Scanning Module:



3.6.1. OS Detection Module:



3.7 Exception Handling:

The package can handle only valid IP Address. If the user specifies an invalid IP Address of any other format as input the system flashes a warning message.

3.8 Future Enhancements:

The following modules could be included in the project in future:

- Finding the Data rate of the ports.
- Detection of any scanning of our ports.

4. SYSTEM DESIGN

The system design is the high level strategy for solving the problem and building a solution. System design includes decisions about the organization of the system into subsystems, allocation of subsystems to hardware and software components, and major conceptual and policy decisions that form the framework for the detailed design.

Design of Port scanner:

Scanning Module:

The following steps are involved in the design of this module:

- Get and Validate the IP Address:

  Obtain the IP Address from the user. Check the IP Address for its validity. If the IP Address is invalid, display an error message and exit. Else, proceed further.

- Obtain the range of Port Numbers:

  Get the starting and ending port numbers that are to be scanned.

- Create sockets and connect to the ports specified:

  Create sockets in the remote system and attempt to connect to the ports specified.

- Identify the status of the ports:

  If connection succeeds then the ports are active. Or else, they are closed for any attempted connections.

- Display the result:

  Display the opened ports with their number, service name.

OS Detection Module:

The following steps are involved in the design of this module:

- Get and Validate the IP Address:

  Obtain the IP Address from the user. Check the IP Address for its validity. If the IP Address is invalid, display an error message and exit. Else, proceed further.

- Pinging:

  Ping the host with the specified IP Address.

- TTL Magic:

  Check the TTL value to identify the Operating System of the remote host identified by the specified IP Address.

- Display the result:

  The result could be

  - Operating System Name
  - Destination Unreachable, if the host is down.

5. SYSTEM TESTING

Testing is an activity to verify that a correct system is being built and is performed with the intent of finding faults in the system. Testing is an activity, however not restricted to being performed after the development phase is complete. But this is to be carried out in parallel with all stages of system development, starting with requirement specification. Testing results once gathered and evaluated, provide a qualitative indication of software quality and reliability and serve as a basis for design modification if required.

System Testing is a process of checking whether the development system is working according to the original objectives and requirements. The system should be tested experimentally with test data so as to ensure that the system works according to the required specification. When the system is found working, test it with actual data and check performance.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and the attendant "cost" associated with a software failure are the motivation forces for a well planned, thorough testing.

5.1 Testing Objectives:

The testing objectives are summarized in the following three steps. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has high probability of finding an error. A successful test is one that uncovers as –yet-undiscovered errors.

5.2 Testing Principles:

All tests should be traceable to customer requirements. Tests should be planned long before testing begins, that is, the test planning can begin as soon as the requirements model is complete. Testing should begin "in the small" and progress towards resting "in large". The focus of testing will shift progressively from programs to individual modules and finally to the entire project. Exhaustive testing is not possible. To be more effective, testing should be one, which has highest probability of finding errors.
The following are the attributes of good tests:

- A good test has a high probability of finding an error.
- A good test is not redundant.
- A good test should be "best of breed"
- A good test should be neither too simple nor too complex.

5.3 Levels of Testing:

The details of the software functionality tests are given below. The testing procedure that has been used is as follow:

- Unit Testing
- Integration Testing
- Validation Testing
- Output Testing

Unit Testing:

Unit testing is carried out to verify and uncover errors within the boundary of the smallest unit or a module. In this testing step, each module was found to be working

satisfactory as per the expected output of the module. In the package development, each module is tested separately after it has been completed and checked with valid data. Unit testing exercise specific paths in the modules control structure to ensure complete coverage and maximum error detection.

The project is divided into three modules. These three modules are developed separately and verified whether they function properly.

Integration Testing:

Integration Testing address the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of higher-order tests are conducted. The main objective in this testing process is to take unit-tested modules and build a program structure that has been dictated by design.

The following are the types of integrated testing:

Top down Integration:

This method is an incremental approach to the construction of the program structure. Modules are integrated by moving downward the control hierarchy, beginning with the main program module. The module sub-ordinates to the main program module are incorporated to the structure in either a depth-first or breadth-first manner.

Bottom up Integration:

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to given level is always available and the need

for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

The low level modules are combined in to clusters that perform a specific software sub-function.

A driver i.e. the control program for testing is return to coordinate test case input and output.

The cluster is tested and drives are removed and clusters are combined moving upward in the program structure.

The three modules which are developed during unit testing are combined together and they are executed and verified whether the linking of the files and the corresponding modules without any error.

Validation Testing:

At the end of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and correction testing begins.

Validation Test Criteria:

Software testing and validation is achieved through series of black box tests that demonstrate conformity with the requirements is achieved, documentation is correct and other requirement is met. Here the three modules which are checked in the integration testing are assembled and programmed in the code is testing for any correction.

Output Testing:

Output testing is series of different test whose primary purpose is to fully exercise the computer based system. Although each test has a different purpose, all the work should be verified so that all system elements have properly integrated and perform allocated functions.

Output testing is the stage of implantation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. The input screens, output documents were checked and required modification made to suite the program specification. Then using rest data prepared, the whole system was tested and found to be a successful one.

Here the whole system is checked by giving sample inputs i.e. the password and it is checked for the whole function so that the system functions well and the requirements are met.

6. Future Enhancements

Our project is so flexible that it can be enhanced so easily. Some more features could also be added to our project. It includes:

- Determining the Geographical Location of the remote system using the router time information.
- Determining whether our ports are being scanned or not.

We will also focus our attention on making our code platform independent. This can be done by coding our project in any of the platform independent languages like Java. This makes the code to be executable globally.

7. Conclusion

The complete design and development of the system "Port Scanner" is presented in this tract. The system has user-friendly features. It is possible for any novice user to use the system.

The programming techniques used in the design of the system provide a scope for further expansion and implementation of any changes, which may occur in the future. The system has been tested for the hosts running on any kind of Operating System.

Since, the requirements of any organization and their standards are changing day by day; the system has been designed in such a way that its scope and boundaries could be expanded in the future with little modifications. As a further enhancement, this system can include the other features specified in the Future Enhancement Section.

8. Bibliography

1. "Advanced Linux Programming", by Mitchell & Samuel, First Edition, New Riders Publications.

2. "TCP/IP", by Comer, Fourth Edition, Prentice Hall Publications.

3. "Linux Programming Unleashed", by Kurt Wall, Second Edition, SAMS Publications.

Websites Visited:

www.advancedlinuxprogramming.com

www.linux.org

www.computercopson.com

www.insecure.org

9. Annexure

9.1. Sample Code:

9.1.1. Scanning Code:

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<errno.h>
#include<netdb.h>

void art()
{
        int j=0;
        while(j!=80)
        {
                printf("-");
                j++;
        }
        printf("\n");
}

struct sockaddr_in sa;
struct servent  *se;
int sock;

void checkout(char *ipa,int port1)
{
        char buf[255]={"Kalai"};
        sa.sin_family=AF_INET;
        sa.sin_port=htons(port1);
```

```
        sa.sin_addr.s_addr=inet_addr(ipa);
        sock=socket(AF_INET,SOCK_STREAM,0);
        memset(&(sa.sin_zero),'\0',8);
        se=(struct servent *)getservbyport(sa.sin_port,NULL);
        if((connect(sock,(struct sockaddr *)&sa,sizeof(sa)))==0)
        {
                printf("\t\t\t%d\t",port1);
                printf("Open\t");
                printf("%s\n",se->s_name);
        }
        close(sock);
}

int main()
{
        int sp,ep;
        char ipa[20];
        int j=0;
        if(!fork())
        {
                execl("/bin/sh","/bin/sh","-c","clear",NULL);
        }
        wait();
        art();
        printf("\t\t\t PORT      SCANNER\n");
        art();
        printf("\nIP ADDRESS : ");
        fgets(ipa,sizeof(ipa),stdin);
        printf("\n");
        if(inet_addr(ipa)==-1)
        {
                printf("INVALID IP ADDRESS\n");
                exit(1);
```

```
    }
    else
    {
            printf("\n");
            printf("STARTING PORTNO: ");
            scanf("%d",&sp);
            printf("\n");
            printf("ENDING   PORTNO: ");
            scanf("%d",&ep);
            printf("\n");
            printf("\nScanning started...\n");
            sleep(2);
            if(fork()==0)
            {
                execl("/bin/sh","/bin/sh","-c","clear",NULL);
            }
            wait();
            printf("\n\n");
            printf("\t\t\tSCANNING PORT\n");
            printf("\t\t\t******** ****\n");
            printf("\t\t\tSCANNING HOST %s\n\t\t\tFOR
PORTS BETWEEN %d AND %d \n",ipa,sp,ep);
            art();
            printf("\n\t\t\tPORT\tSTATE\tSERVICE
NAME\n");
            art();
            j=sp;
            printf("\n");
            while(j<=ep)
            {
                    switch(fork())
                    {
                            case 0:
```

```
                    checkout(ipa,j);
                    exit(0);
                    case -1:
                    perror("fork");
                    exit(1);
                    break;
                }
                j++;
            }
            wait();
            printf("\n\n");
            art();
            printf("\n\n\n");
            return;

    }
}
```

9.1.2. OS Detection Code:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
int j;
int s,fd;
char buf[350];
const char *b1="Destination Host Unreachable";
char *r,*f;
char r3[4];
char pc[30]="ping -c 1 ",ip[13];
char res[150];
void art()
{
        int i=0;
        while(i!=80)
        {
```

```
            printf("-");
            i++;
        }
        printf("\n");
}
int main()
{
        if(!fork())
        {
            execl("/bin/sh","/bin/sh","-c","clear",NULL);
        }
        wait();
        art();
        printf("\t\t\t\tOS DETECTION\n");
        art();
        printf("\nIP ADDRESS: ");
        scanf("%s",ip);
        if(inet_addr(ip)==-1)
        {
            printf("INVALID IP ADDRESS \n\n\n\n");
            art();
            exit(1);
        }
        else
        {
            if(ip!=NULL)
                strcat(pc,ip);
            fd=open("s3.c",O_RDWR);
        if(fd<0)
            {
                perror("open");
                exit(1);
            }
```

```
if(fork()==0)
{
        close(1);
        dup(fd);
        execl("/bin/sh","/bin/sh","-c",pc,NULL);
}
wait();
close(fd);
printf("\n");
printf("Detecting the OS...\n");
sleep(1);
fd=open("s3.c",O_RDWR);
if(fd<0)
{
        perror("open");
        exit(1);
}
s=read(fd,buf,250);
if(s<0)
{
        perror("read");
        close(fd);
        exit(1);
}
close(fd);
f=strstr(buf,b1);
if(f!=NULL)
{
art();
printf("\n%s\n",b1);
art();
        exit(1);
}
```

```
            else
            {
            r=strtok(buf,"=");
            r=strtok(NULL,"=");
            r=strtok(NULL,"=");
            j=0;
            while(j!=3)
            {
                  r3[j]=r[j];
                  j++;
            }
            r3[j]='\0';
            if(strcmp(r3,"64 ")==0)
                  strcpy(res,"LINUX");
            else if(strcmp(r3,"128")==0)
                  strcpy(res,"WINDOWS");
            printf("\n");
            printf("\nThe Host with IP ADDRESS %s runs on %s
OS\n",ip,res);
            printf("\n\n\n");
            art();
            printf("\n\n\n");
            }

      }
}
```

9.2 Output Screens:

PORT SCANNER

IP ADDRESS : 90.0.0.111

STARTING PORT NO : 1

ENDING PORT NO : 500

Scanning started...

SCANNING PORT

****

SCANNING HOST 90.0.0.111

FOR PORTS BETWEEN 1 AND 500

PORT	STATE	SERVICENAME
7	open	echo
13	open	daytime
19	open	chargen
21	open	ftp
22	open	ssh
23	open	telnet
25	open	smtp
37	open	time
53	open	domain
79	open	finger
80	open	http
109	open	pop-2
110	open	pop-3
111	open	sunrpc
113	open	auth
139	open	netbios-ssn
143	open	imap
199	open	smux
389	open	ldap

OS DETECTION

IP ADDRESS : 90.0.0.111

DETECTING OS...

The Host with IP ADDRESS 90.0.0.111 runs on LINUX OS.

OS DETECTION

IP ADDRESS : **90.0.0.1000**

INVALID IP ADDRESS
