VPN MONITORING

Project Report P-1207

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

Bachelor of Computer Science and Engineering
Of
Bharathiar University, Coimbatore.

Submitted by

Ashwini Venkatesan
0027K0162

Candy Sarah George
0027K0168

Under the guidance of

Mrs. V. Vanitha M.E.
Senior Lecturer

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE – 641006.

MARCH 2004.

CERTIFICATE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE

This is to certify that the project entitled

VPN MONITORING

is done by

Ashwini Venkatesan — **0027K0162**

Candy Sarah George — **0027K0168**

and submitted in partial fulfillment of the

requirements for the award of the degree of the

Bachelor of Computer Science and Engineering

Of

Bharathiar University, Coimbatore.

S. Thangasamy

Professor & Head of the department

(Dr. S. Thangasamy)

17/3/04

Guide

(Mrs. V. Vanitha M.E.)

Certified that the candidates were examined by us in the project work viva voce examination held on 23-03-2004.

D. Chandrakumar 23/3/'04

Internal Examiner

S. Thiruvasagam 23/3/04

External Examiner

DECLARATION

DECLARATION

We,

Ashwini Venkatesan	**0027K0162**
Candy Sarah George	**0027K0168**

hereby declare that the project entitled **"VPN Monitoring"**, submitted to Bharathiar University as the project work of **Bachelor of Computer Science & Engineering Degree**, is a record of original work done by us under the supervision and guidance of **Mrs. V. Vanitha M.E.,** *Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore* and this project work has not found the basis for the award of any Degree, to any candidate of any University.

Place: COIMBATORE

Date: 23-03-2004

Ashwini Venkatesan

Candy Sarah George

Countersigned By

17/3/04

Ms. V. Vanitha M.E.

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We express our profound respect and sincere gratitude to, **Dr. K. K. Padmanabhan PhD,** *Principal, Kumaraguru College of Technology,* for ushering us in the path of triumph.

We are thankful to **Dr. S. Thangasamy PhD**, our beloved *Professor and the Head of the Department, Computer Science and Engineering,* whose consistent support and enthusiastic involvement helped us a great deal.

We record our sincere thanks to our project coordinator **Mrs. D. Chandrakala M.E**, Senior Lecturer, Department of Computer Science and Engineering, for her invaluable advice and encouragement throughout the course of the project.

We are greatly indebted to our beloved guide **Mrs. V. Vanitha M.E.**, *Lecturer, Department of Computer Science and Engineering* for her excellent guidance and timely support in completing the project. As a token of our esteem and gratitude, we honour her for her assistance towards this cause.

We also thank our beloved class advisor **Mrs. M.S. Hema B.E.**, for her invaluable assistance.

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

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

SYNOPSIS

The project is aimed to establish a virtual private network (VPN), which ensures that the network is secure; and then enable the administrator to monitor any client system from the server. VPN technology is remarkably secure. Data that is sent over VPN requires authorization if it has to be received or replayed. The project will provide a systems control tool, which can be run on the server system, and can control almost any function of any computer on the network. This created network helps send data in the encrypted form, which disables any intruders from viewing the files being transferred.

The software created is used in a network of computers, where there exists a server and many clients. It enables the administrator to monitor the entire network. This can be done by enabling him and only him, to view, in his system; all activities being performed at the client side. This right is provided only to the administrator at the server side and not to the client side. Since it is a network, the administrator can choose the required system or client by just having the knowledge of the IP address of that particular system. The administrator not only has the right to view, but also has the facility to control the user rights. The power of the client side system can also be controlled by the will of the administrator. This project also enables the communication between the client and server side systems.

CONTENTS

CONTENTS

INTRODUCTION

1) INTRODUCTION

1.1) Existing system:

Virtual private Network (VPN) is the kind of network created between the systems where the data sent between them is extremely secure. Hence this network does not allow the hacking of private data in an organization. Here the data sent is said to undergo the tunneling effect. This is done by creating a tunneling effect for all the packets being sent. The packets are wrapped around to create another packet with some special headers and sent to the server side. At the client side the packets are received and the data is extracted from the packet

Most VPN systems use IPSec technologies. IPSec is useful because it is compatible with most different VPN hardware and software, and is the most popular for networks with remote access clients.

IPSec requires very little knowledge for clients, because the authentication is not user-based. Instead, the security comes from the workstation's IP address or its certificate, establishing the user's identity and ensuring the integrity of the network.

A drawback of IPSec-compliant products is that they provide access control over the network and transport layers only, and not a great deal of measures to selectively regulate access to individual resources within these hosts. If customers given access to particular company information on a server, highly selective controls are needed to make sure they access only the information they have been authorized to see.

.

1.2) Proposed System:

The proposed system helps the administrator using the server to easily view the client side system, access information from it, lock up its desktop or even shut down the system.

Unlike the previous system, with this system, if there is some kind of breach in security, only the destination network is affected. This is achieved using L2TP or PPTP protocol. It also helps to handle virtually any authentication and encryption standards. Here, the protocols of the communicating systems should be compatible.

PPTP (Point-to-Point Tunneling Protocol) is a protocol that allows corporations to extend their own corporate network through private tunnels over the public Internet.

L2TP (Layer Two Tunneling Protocol) is an extension of PPTP used by an Internet service provider to enable the operation of a virtual private network over the Internet. It uses packet-switched network connections to make it possible for the endpoints to be located on different machines.

SYSTEM REQUIREMENT ANALYSIS

2) SYSTEM REQUIREMENT AND ANALYSIS

2.1) Product Definition:

The objective of VPN Monitoring is to create Remote Task Manager and provide network security. A random user cannot simply log in to a VPN, as some information is needed to allow a remote user access to the network. This kind of network helps the administrator to monitor the network and the operations being carried out. It enables the administrator to block any part of any of the client systems, in that particular network.

2.2) Project Plan:

Network monitoring is a software tool used to monitor the systems connected to the network. This software monitors each and every system that is connected to the network and maintains the record of each user who enters the system. The modules in the software are as follows:

i) Connection Establishment

ii) Application Viewing

iii) Keyboard Spy

iv) Screen Shots

v) Chatting

vi) System Shutdown

Connection establishment

The network that is present on which exists many clients and one server is used for establishing the connection between the server and the required client. This modules is a server side activated module where the server connects itself to the

client it wishes to monitor over. This is done by establishing the connection by sending a request to the required IP address. Once a request is sent for connection establishment the client side gets connected, if the connection is not already established or the software is running.

Keyboard spy

When server requests for the current action of the user from a particular client, the client sends the required file to the server, which contains the current action over the keyboard. It contains a notepad file which is a report generated having the details of the keys pressed by the user. The file can also be saved for future proof.

Process viewing

When server request for the current action of the user from a particular client the client sends the required file to the server, which contains the current action of the user logged in. it contains the process name both the background and foreground process, process time, the location. So the server can view the current action. The file sent by the client can be saved if needed for future proof.

System shutdown

When the server finds the need to shutdown the client side system, he has to input the details like the system name, the domain name and the user name. When the server sends the command to shutdown the client system, the client does not have the capability of stopping the process. This module is used when the extreme measures have to be taken for monitoring purpose.

Screen capturing

The server sends request to the client to capture the current active screen or the entire screen of the client and is stored as .bmp file and the client reads the bmp file and transfers it in form of bytes when the server request the server then

receive the bytes and it writes in a separate bmp file in the server so that the current active or entire screen of the client can be seen from the server.

Chatting

This module is used for communication between the server and the client. The connection is established by mentioning the IP address of the client. Then the messages are sent between each other through the network. Hence the server can send some sort of warning messages to the required client or any sort of clarification can be done by the systems. This connection can be established and disconnected only by the network.

SOFTWARE REQUIREMENT SPECIFICATION

3) SOFTWARE REQUIREMENTS SPECIFICATION

Case Study:

The case study portion in the section 1 and 2 covers the server/client requirements. Section 3 and 4 contain specific requirements.

3.1) Introduction

3.1.1) Purpose of the document:

The purpose of this document is to provide the requirements for the monitoring of network. Sections 1 and 2 are primarily intended for customers of the application, but will also be of interest for network programmers. Section 3 is of use for network engineers, software people and customers of this application.

3.1.2) Scope:

The document is limited to the description of the modules, and the different ways of monitoring the client systems from administrator side.

3.2) Overall description:

The main function associated with this product is described in this section. The characteristics of a user of this product are indicated.

3.2.1) Product perspective:

The objective of the project is network security by creating a Remote Task Manager. People often think that Remote Task Manager (RTM) is a remote desktop program. This is not true. RTM is a systems control tool. A random user

cannot simply log in to a VPN, as some information is needed to allow a remote user access to the network. This kind of network helps the administrator to monitor the network and the operations being carried out. It enables the administrator to block any part of any of the client systems, in that particular network.

3.2.2) Product function:

A systems control tool can be run from any remote Windows computer and can control almost any function of any computer on the network. It is the primary tool used to perform routine network administration tasks.

3.2.3) User characteristics:

Those using this product are common users, who should be able to understand every aspect. They need training to understand and use the system effectively.

3.2.4) Constraints:

The product shall operate as a console system.

3.3) Specific requirements:

In this section we will see about the specific requirements like interface, functional requirements, performance requirement of the product.

3.3.1) Interface requirements:

User interface:

The user interface of this product requires a text area for information transfer during one of the modules.

3.3.2) Functional requirements:

This will define the fundamental actions, which take place in the product.

Information flow:

The information flow is in the form of images, files and text .The server will first connect to the client and on connecting, it serves the information to the server.

Process description:

A tunnel is created between the server and the client. All information that has to be sent between the two will be passed through this tunnel. After the VPN has been setup, the administrator can monitor the client system, by viewing the keys pressed, the processes running on the client machine, the snapshot of client's screen; and if required he can shutdown the client machine from the server.

SYSTEM DESIGN

4) SYSTEM DESIGN & IMPLEMENTATION

4.1) Introduction

This document will specify the logical structure of the program. This will show the detailed description of the product.

4.1.1) Purpose:

The purpose of the software design document (SDD) is to provide an overall description of the secure network created between server and client, and the monitoring of client by the administrator on the server.

4.1.2) Scope:

The scope of the software design document is constrained by the requirements stated in the "VPN Monitoring" software requirement specification.

4.2) Decomposition Descriptions

This section describes the modules involved in the project and the descriptions of their functions.

4.2.1) Module Decomposition:

In this the modules are classified based on the operation. They are:

i) Network Establishment

ii) Application Viewing

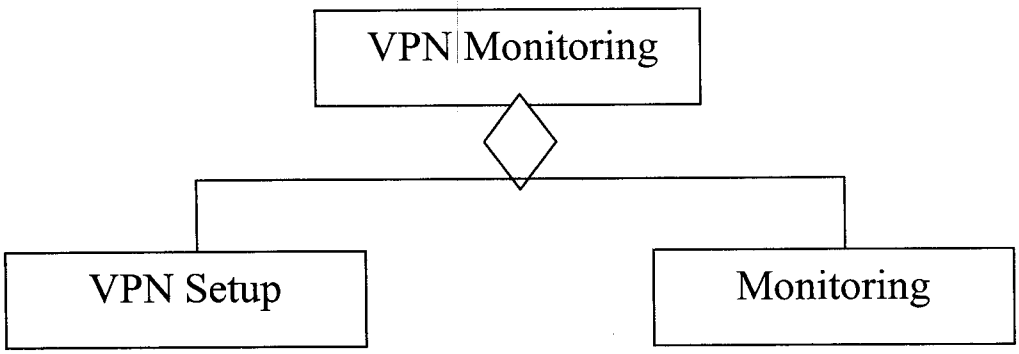
iii) Keyboard Spy

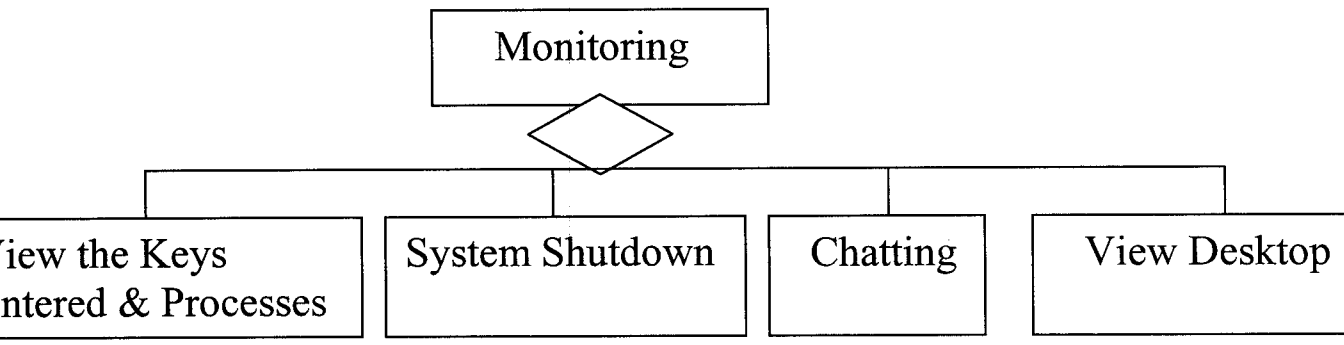
iv) Screen Shots

v) Chatting

vi) System Shutdown

Aggregation diagrams:





The aggregation diagrams show that this product (VPN Monitoring) consists of VPN Setup and the monitoring process of client by the administrator.

4.2.2) Data decomposition:

Monitoring: The monitoring consists of identifying the keys pressed and viewing processes of client system. It also has options to chat, shutdown and view desktop.

- **View Keys Entered:** The administrator can view the keys entered by the client. It is captured and stored into file, which is retrieved by the server.
- **View Processes:** The administrator can view the current processes running on the client side system. The processes are stored into a file, which is retrieved by the server.
- **System Shutdown:** The client side system can be shutdown if the administrator finds that it is necessary.
- **Chatting:** This enables the administrator and client to communicate with each other easily.
- **View Desktop:** The administrator is capable of viewing the client's desktop, by retrieving the snap shot of the desktop stored in a file.

4.3) Dependency Description

4.3.1) Intermodule Dependencies:

The use case is a view of the system from a specific User's view. This considers actions initiated by the user on the system, and consequently, the actions initiated by the system on the user.

The use case diagrams illustrate the effects on the client with the typical uses of the product. These diagrams provide a lower- level implementation, such as the Order, Direction and Information of use case.

High-Level Analysis Models:

A process of refinements starting from the Use Case Models has generated the Analysis Model. Each use case role has been devised into the stages of the analysis model. This model is a very general and high-level collaboration

diagram. It takes the role from the use case diagram and expands each of the use cases to show how they will collaborate within the components of the design. The detailed use case descriptions are described using stereotypes. These are as follows:

Boundary Class

Models interaction between the system and its user (users and external systems).

Control Class

Represents coordination, sequencing, transition and control of other objects.

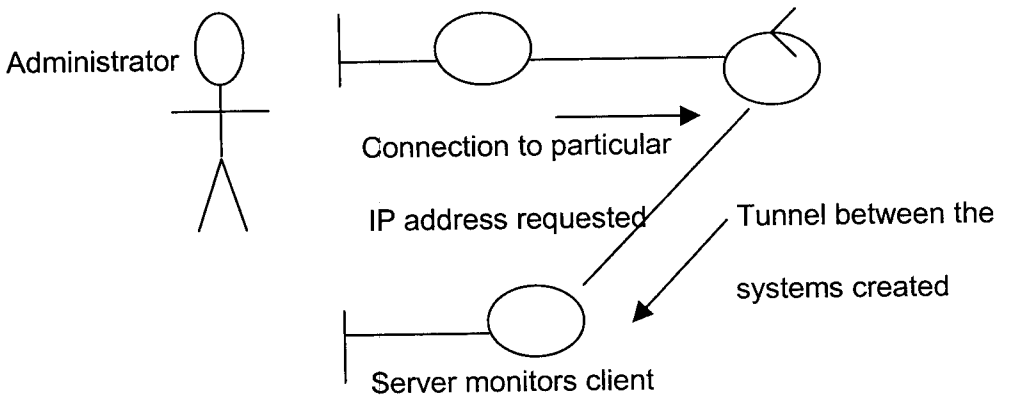
Entity Class

Modules information long-lived (persistent) data.

Stereotypes

i) VPN Setup

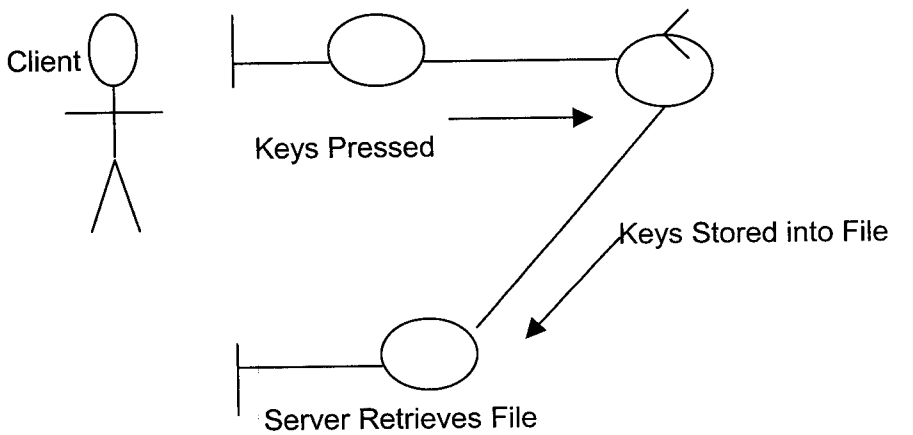
Connection:



The above diagram shows that the administrator can setup a VPN by using the IP address of the client side system. After the tunnel is created, packets are sent over the network, as the administrator monitors client.

ii) Monitoring

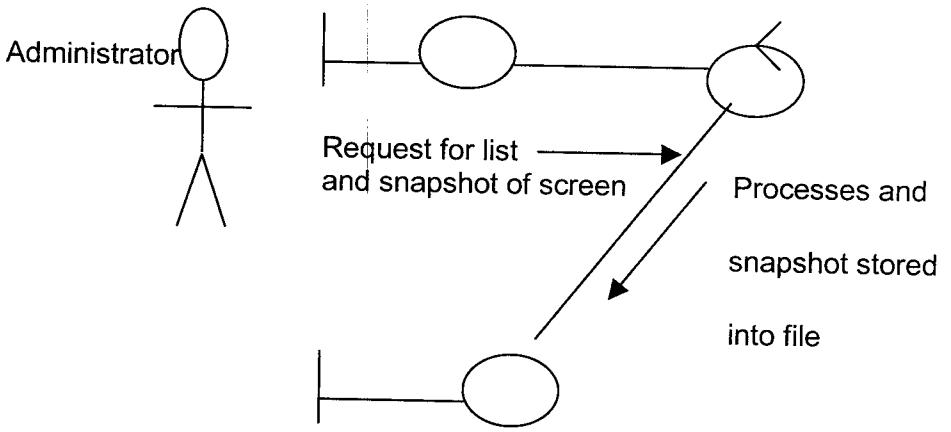
Keyboard Spy:



The above diagram shows that when the administrator needs to view the keys entered on the client side system, then he invokes the operation; which

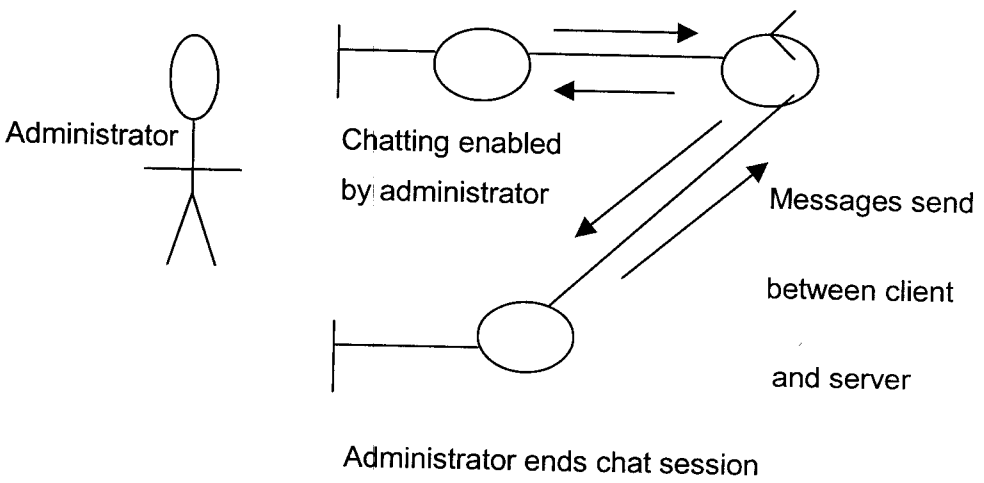
stores the keys entered into file. This file is then retrieved and viewed by the administrator on the server machine.

View Process List & Screen:



The above diagram shows that when the administrator needs to view the processes running on the client side system or view client side screen, he invokes the operation; which stores the processes and snapshot of screen into file. This file is then retrieved and viewed by the administrator on the server machine.

Chatting:



The administrator can start a chat session with any client. Both server and client will be able to send messages, till the administrator ends the session.

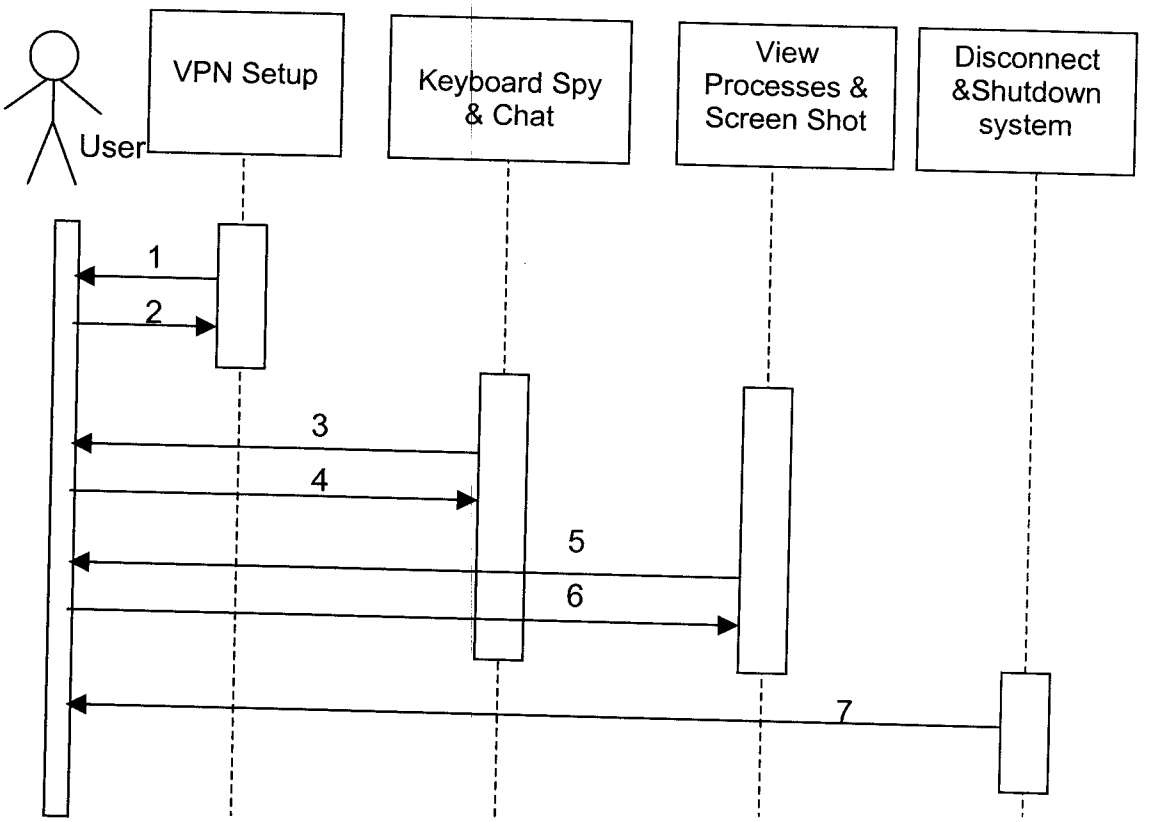
4.3.2) Interprocess Dependencies:

Sequence Diagrams

A sequence diagram consists of a collection of objects (named rectangles), messages (solid-line arrows) and time, represented as a vertical progression extending downward from each object as a dashed line. This line is the object's lifeline. Along the lifeline is a narrow rectangle that is called activation. The activation represents an execution of an operation that the object carries out. The length of the rectangle signifies the activation's duration. Messages are passed between objects. Within the sequence diagram, message passing is represented as an arrow between one objects lifeline and another. There are three kinds of messages that can be passed between objects.

The full arrow is a simple message and symbolizes the transfer of control from one object to another. A synchronous message is represented by an open arrow that has a cross (x) at its arrowhead. "Half arrowheads" denotes the opposite one, known as the Asynchronous message. An asynchronous message is one that returns immediately after spawning a new thread in the receiving object.

The diagram below illustrates that the administrator can start monitoring the client side system only after setting up of the VPN. This is achieved by using the client's IP address. After this administrator can view the keys pressed, the processes running on the client machine, the snapshot of the screen. He can also start up a chat session with the client. Finally if he requires, he can disconnect the VPN link or even shutdown the client's system.



The processes numbered in the diagram are:

1 Use IP address to initiate VPN

2 VPN established

3 Use IP address to chat and retrieve file which contains keys pressed

4 Client chats with administrator

5 Retrieval of process file and snapshot

6 Required files sent

The network is implemented by using the point-to-point protocol where the required data is in a packet which is taken and sent to the server side as an encrypted data through a tunnel which is created. This is achieved by adding some special header to the packet. The packet is taken and the length of the data field is calculated. This length is attached to the data sent along the network. This newly created packet is encrypted into the network as a secure

data. Since the data has been encrypted along with the header of the packet the data cannot be retrieved by any outsider who tries to indulge in the network, since to extract the data, he needs to know the length attached to it.

At the server side of the network, the data is extracted from the received packet by first decrypting the packet then extracting the data from the header attached. First the header is read, which indicates the length of the data. Hence the specified number of bytes is extracted.

As far as the connection establishment is concerned the server has to mention the IP address of the client side system. Address is sent as the socket data to the client side. If the address is correct then the connection is established between the client and server. The administrator then chooses the option from the menu, to monitor the client. The client receives this request and then sends the required file to the server, which on reception is decrypted and displayed.

SYSTEM TESTING

5) SYSTEM TESTING

5.1) Testing Activities

This section deals with the planning of the test case, resources, schedule, and features to be tested. Here the tests to be conducted are designed with sample test cases. The test results are then recorded for future reference and improvements.

5.1.1) Plan the general approach, resources & schedule:

The general approach for testing the product is to find the requirements, design document details and then plan the test case. We should also consider the environment and design the test case for specialized environments. In this case the software is a Graphical User Interface (GUI), hence the test case should be designed in such a way that it tests all the properties and functions of a GUI.

Required Resources for the testing process:

System Requirement Specification (SRS) document.

System Design Description (SDD) document.

.

5.1.2) Determine Features to be tested:

This is a test for a specialized environment like the GUIs and so we should take care in testing the qualities of GUI interfaces and other properties like interfacing of the system with the environment.

The test includes:

Test for application window properties

Test for pull down menus.

Test for the mouse operations.

5.1.3) Design the set of tests:

The language used for the development of the product is VC++, so that we can test every action and hope that we get the expected response. We will create a template for every test.

Test Set 1: For displayed windows

1. Will the window open properly based on menu-based commands?
2. Can the window be resized, moved, and scrolled?
3. Does the window properly regenerate when it is overwritten and then re-called?
4. Are all functions that relate to the window available when needed?
5. Are all functions that relate to the window operational?
6. Are all relevant pull-down menus, controls, scroll bars, dialog boxes, and buttons, available and properly displayed for the window?

Test Set 2: For pull-down menus and mouse operations

1. Do pull-down operations work properly?
2. Are all functions and pull-down sub functions properly listed?
3. Do multiple or incorrect mouse clicks within the window cause unexpected side effects?

5.1.4) Execute the test procedures:

The above set of tests is executed in sequence & if any errors or negative results occurred for the test, then it is corrected immediately and the test is preceded further.

5.1.5) Evaluate the test effort:

The following are the test results with the errors that have occurred & corrected:

Test Set 1:

1. Yes, the window opens properly according to the context.
2. Yes, all can be done.

3. Yes, the window redraws works properly.
4. Yes, all functions are available when needed.
5. Yes, all functions relate to that window.
6. Yes, all controls and objects are displayed at their appropriate places.

Test Set 2:

1. Yes, the pull-down menus work properly.
2. Yes, all are listed properly.
6. No, incorrect mouse picks within the window do not generate unexpected side effects.

The above results are evaluated and accepted to be satisfactory and hence the test process is terminated.

5.2) Functional testing

The testing refers to the operation of some important function, present in the product. In this project the administrator on the server machine should be able to retrieve files from client side.

Files Retrieved Should be Correct:

Keys Pressed by client:
The keys pressed by the client should be correctly stored in the file that is to be retrieved by the administrator.

Processes Running on client system:
The processes that are currently running on the client machine has to be stored into a file that will be viewed by the administrator.

Snapshot of client's screen:

A snapshot of the current client screen has to be stored into a bmp file that will be retrieved and viewed by the administrator on the server machine.

Chat Session Should Work Properly:

The server and client should be able to chat without any delays or problems. Both the administrator and client should be able to see, their own messages as well as the other person's messages.

Functional test result:

Yes, keys presses correctly retrieved.

Yes, processes running on the client system are stored into file properly.

Yes, the bmp file with the snapshot of screen is correctly retrieved.

Yes, the chat session is functioning correctly.

FUTURE ENHANCEMENTS

6) FUTURE ENHANCEMENTS

- ✓ Make the monitoring process fully protected by the administrator. The client should not be allowed to close the program at any point.

- ✓ Use the public Internet to create a VPN, and hence connect systems at different geographical locations and enable the administrator to still monitor the client.

- ✓ Enable video conferencing.

CONCLUSION

7) CONCLUSION

VPN Monitoring thus enables the administrator to effectively monitor the entire client systems currently connected in the LAN on a one-to-one basis. The network created is very safe, as a tunneling effect was created and it can provide security for organizations that have the requirement of monitoring the client systems.

BIBLIOGRAPHY

8) BIBLIOGRAPHY

BOOKS REFERRED

- George Sheperd, Scot Wingo and David Kruglinski; "Programming in Visual C++"; Released by Microsoft Press

- Andrew S. Tannenbaum; "Computer Networks"; Amsterdam Vrije University; Third Edition

- James F. Peters & Pedrycz;" Software Engineering & Engineering Approach" ;Third Edition

WEBSITES VISITED

- **www.ieee.org**

- **www.vpn.com**

ANNEXURE

9) ANNEXURE

9.1) Sample Source Code:

```
#include<stdio.h>
#include<dos.h>
#include<windows.h>
#include<winsock.h>
char buf[1029];
char str[50];
int sockfd;
char data[1000];
FILE *stream;
char* decrypt(int length)
{
 char st;
 char dat[500];
 int i;
 for(i=0;i<length;i++)
 {
        st=(*(data+i))-6;
        data[i]=st;
 }
 data[i]='\0';
 return data;
}

void main()
{
```

```
WSADATA WSAData;
struct sockaddr_in  serv;
int ret,num;
char senddata[1000],ip[100];
int choice;
int length;
char command[4];
char file_name[20];
char sys_str[100];
char len[20];
int p_length;
int tempint;
char stdata[2000];
printf("Enter The IP Address to connect:");
scanf("%s",ip);
WSAStartup (MAKEWORD(1,1), &WSAData);
sockfd=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);


serv.sin_family=AF_INET;
serv.sin_port=htons(1098);
serv.sin_addr.s_addr=inet_addr(ip);


ret=connect(sockfd,(struct sockaddr*)&serv,sizeof(serv));
if (ret==0)
{
        printf("\nConnected Successfully");
}
```

```
else
{
        printf("\nError in Connection");
        getch();
        return 0;
}

while(1)
{
        printf("\n 1. Chatting");
        printf("\n 2. Remote Shutdown");
        printf("\n 3. Keyboard Spy");
        printf("\n 4. Process List");
        printf("\n 5. Screen Shot");
        printf("\n 6. URL Spy\n");
        printf("\n 7. Exit\n");

        printf("\nEnter Your Choice:");
        scanf("%d",&choice);

        if (choice==1)
        {
                printf("\nChatting");
                strcpy(senddata,"<clientchat>û");
                ret=send(sockfd,senddata,lstrlen(senddata)
+1,0);
                system("CSocketcli.exe");
        }
        else if (choice==2)
        {
                printf("\nRemote Shutdown");
```

```
                strcpy(senddata,"<shutdown>û");
                ret=send(sockfd,senddata,lstrlen(senddata) +
1,0);
        }
        else if (choice==3)
        {

                printf("\nKeyboard Spy");
                strcpy(senddata,"<keyspy>û");
                ret=send(sockfd,senddata,lstrlen(senddata) +
1,0);


                length=recv(sockfd,buf,sizeof(buf),0);
                printf("\nThe received data is %s",buf);

                i=0;

                while (buf[i]!='²')
                {
                        len[i]=buf[i];
                        i++;
                }

                len[i]='\0';

                for(i=0;i<strlen(len);i++)
                {
                        len[i]=len[i]-6;
                }
```

```
p_length=atoi(len);

tempint=i+1;
printf("\nThe length is %d",tempint);
getch()
for(i=tempint;i<(p_length+tempint+1);i++)
{
        stdata[i-tempint]=buf[i];
}
stdata[i-tempint]='\0';
printf("\nThe received data is %s",stdata);
for(i=0;i<p_length;i++)
        data[i]=stdata[i];

strcpy(buf,decrypt(p_length));
printf("\nThe data is %s",buf);
for(i=0;i<3;i++)
{
        command[i]=buf[i];
}
command[i]='\0';
if (strcmp(command,"BOF")!=0)
{
        printf("Error");
        return 0;
}

for(i=3;i<length;i++)
{
        file_name[i-3]=buf[i];
        if (buf[i]='\0') break;
```

```
        }

        printf("\n File name is %s",file_name);

        if ((stream=fopen(file_name,"wb"))==NULL)
        {
                printf("\nError in opening");
                return 0;
        }

        while(1)
        {
                strcpy(senddata,"NEXTû");
ret=send(sockfd,senddata,lstrlen(senddata) + 1,0);
                length=recv(sockfd,buf,sizeof(buf),0);
                printf("\nThe received length is
                %d\n",length);
                i=0;
                while (buf[i]!='²')
                {
                        len[i]=buf[i];
                        i++;
                }
                len[i]='\0';

                for(i=0;i<strlen(len);i++)
                {
                        len[i]=len[i]-6;
                }
                printf("\nThe length string is %s",len);
```

```
                    p_length=atoi(len);
                    tempint=i+1;
                    printf("\nThe length is %d",p_length);
                    getch();
                    for(i=tempint;i<length+tempint;i++)
                    {
                            stdata[i-tempint]=buf[i];
                    }
                    stdata[i-tempint]='\0';
                    printf("\nThe message is %s",stdata);
                    for(i=0;i<p_length;i++)
                            data[i]=stdata[i];
                    decrypt(p_length);
                    data[p_length]='\0';
                    printf("\nThe Decrypted data is
                    %s\n",data);
                    for(i=0;i<3;i++)
                    {
                            command[i]=data[i];
                    }
                    command[i]='\0';
                    if (strcmp(command,"EOF")==0)  break
            fwrite(data,sizeof(char),p_length,stream);
            }
            fclose(stream);
            strcpy(sys_str,"notepad.exe ");
            strcat(sys_str,file_name);
            system(sys_str);
}
else if (choice==4)
{
```

```
    printf("\nProcess ");
    strcpy(senddata,"<processspy>û");
ret=send(sockfd,senddata,lstrlen(senddata) + 1,0);
    //receiving the first command
    length=recv(sockfd,buf,sizeof(buf),0);
    printf("\nThe received data is %s",buf);
    i=0;
    while (buf[i]!='²')
    {
            len[i]=buf[i];
            i++;
    }

    len[i]='\0';
    for(i=0;i<strlen(len);i++)
    {
            len[i]=len[i]-6;
    }
    p_length=atoi(len);
    tempint=i+1;
    printf("\nThe length is %d",tempint);
    getch();

    for(i=tempint;i<(p_length+tempint+1);i++)
    {
            stdata[i-tempint]=buf[i];
    }
    stdata[i-tempint]='\0';
    printf("\nThe received data is %s",stdata);
    for(i=0;i<p_length;i++)
            data[i]=stdata[i];
```

```
strcpy(buf,decrypt(p_length));
printf("\nThe data is %s",buf);
for(i=0;i<3;i++)
{
        command[i]=buf[i];
}
command[i]='\0';
if (strcmp(command,"BOF")!=0)
{
        printf("Error");
        return 0;
}
for(i=3;i<length;i++)
{
        file_name[i-3]=buf[i];
        if (buf[i]='\0') break;
}
printf("\n File name is %s",file_name);
if ((stream=fopen(file_name,"wb"))==NULL)
{
        printf("\nError in opening");
        return 0;
}

while(1)
{
        strcpy(senddata,"NEXTû");


ret=send(sockfd,senddata,lstrlen(senddata) + 1,0);
```

```
length=recv(sockfd,buf,sizeof(buf),0);
printf("\nThe received length is %d\n",length);
i=0;

while (buf[i]!='²')
{
                len[i]=buf[i];
                i++;
        }
len[i]='\0';
for(i=0;i<strlen(len);i++)
{
        len[i]=len[i]-6;
}
printf("\nThe length string is %s",len);
p_length=atoi(len);
tempint=i+1;
printf("\nThe length is %d",p_length);
getch();

for(i=tempint;i<length+tempint;i++)
{
stdata[i-tempint]=buf[i];
}

stdata[i-tempint]='\0';
 printf("\nThe message is %s",stdata);

for(i=0;i<p_length;i++)
data[i]=stdata[i];
decrypt(p_length);
```

```
data[p_length]='\0';
printf("\nThe Decrypted data is %s\n",data);
for(i=0;i<3;i++)
{
        command[i]=data[i];
}
        command[i]='\0';

if (strcmp(command,"EOF")==0)  break;

fwrite(data,sizeof(char),p_length,stream);
                }
                fclose(stream);
                strcpy(sys_str,"notepad.exe ");
                strcat(sys_str,file_name);
                system(sys_str);

        }
        else if (choice==5)
        {
                printf("Screen Shot");
                strcpy(senddata,"<screenspy>û");
                ret=send(sockfd,senddata,lstrlen(senddata) +
1,0);

                length=recv(sockfd,buf,sizeof(buf),0);
                printf("\nThe received data is %s",buf);
                i=0
                while (buf[i]!='2')
                {
                        len[i]=buf[i];
```

```
        i++;
}

len[i]='\0';

for(i=0;i<strlen(len);i++)
{
        len[i]=len[i]-6;
}

p_length=atoi(len);

tempint=i+1;
printf("\nThe length is %d",tempint);


for(i=tempint;i<(p_length+tempint+1);i++)
{
        stdata[i-tempint]=buf[i];
}

stdata[i-tempint]='\0';

printf("\nThe received data is %s",stdata)
for(i=0;i<p_length;i++)
        data[i]=stdata[i];

strcpy(buf,decrypt(p_length));
printf("\nThe data is %s",buf);
for(i=0;i<3;i++)
{
```

```
                command[i]=buf[i];
        }
        command[i]='\0';

        if (strcmp(command,"BOF")!=0)
        {
                printf("Error");
                return 0;
        }

        for(i=3;i<length;i++)
        {
                file_name[i-3]=buf[i];
                if (buf[i]='\0') break;
        }

        printf("\n File name is %s",file_name);
        if ((stream=fopen(file_name,"wb"))==NULL)
        {
                printf("Error in opening");
                return 0;
        }

        while(1)
        {
                strcpy(senddata,"NEXTû");
ret=send(sockfd,senddata,lstrlen(senddata) + 1,0);
length=recv(sockfd,buf,sizeof(buf),0);
printf("\nThe received length is %d\n",length);

                i=0
```

```
while (buf[i]!='²')
{
        len[i]=buf[i];
        i++;
}
len[i]='\0';

for(i=0;i<strlen(len);i++)
{
        len[i]=len[i]-6;
}

printf("\nThe length string is %s",len);

p_length=atoi(len);

tempint=i+1;
printf("\nThe length is %d",p_length);

for(i=tempint;i<length+tempint;i++)
{
        stdata[i-tempint]=buf[i];
}

stdata[i-tempint]='\0';

printf("\nThe message is %s",stdata);

for(i=0;i<p_length;i++)
        data[i]=stdata[i];
```

```
                    decrypt(p_length);

                    data[p_length]='\0';

                    printf("\nThe Decrypted data is
                    %s\n",data);


                    for(i=0;i<3;i++)
                    {
                            command[i]=data[i];
                    }
                    command[i]='\0';

                    if (strcmp(command,"EOF")==0)  break;


fwrite(data,sizeof(char),p_length,stream);
                    }
                    fclose(stream);
                    strcpy(sys_str,"mspaint.exe ");
                    strcat(sys_str,file_name);
                    system(sys_str);

            }
            else if (choice==6)
            {
                    printf("URL Spy");
                    strcpy(senddata,"<urlspy>û");
                    ret=send(sockfd,senddata,lstrlen(senddata) +
                    1,0);
```

```
length=recv(sockfd,buf,sizeof(buf),0);
printf("\nThe received data is %s",buf);
i=0;
while (buf[i]!='2')
{
        len[i]=buf[i];
        i++;
}

len[i]='\0';

for(i=0;i<strlen(len);i++)
{
        len[i]=len[i]-6;
}

p_length=atoi(len);

tempint=i+1;
printf("\nThe length is %d",tempint);
getch();

for(i=tempint;i<(p_length+tempint+1);i++)
{
        stdata[i-tempint]=buf[i];
}

stdata[i-tempint]='\0';

printf("\nThe received data is %s",stdata);
```

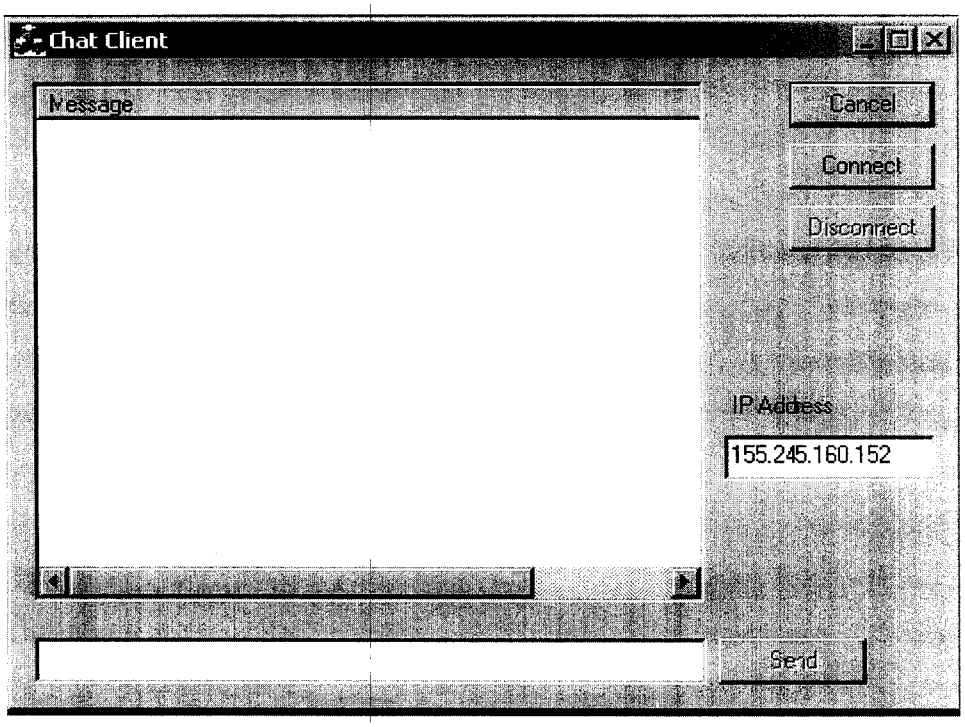
```
for(i=0;i<p_length;i++)
        data[i]=stdata[i];

strcpy(buf,decrypt(p_length));

printf("\nThe data is %s",buf);


for(i=0;i<3;i++)
{
        command[i]=buf[i];
}
command[i]='\0';

if (strcmp(command,"BOF")!=0)
{
        printf("Error");
        return 0;
}

for(i=3;i<length;i++)
{
        file_name[i-3]=buf[i];
        if (buf[i]='\0') break;
}

printf("\n File name is %s",file_name);

if ((stream=fopen(file_name,"wb"))==NULL)
{
```

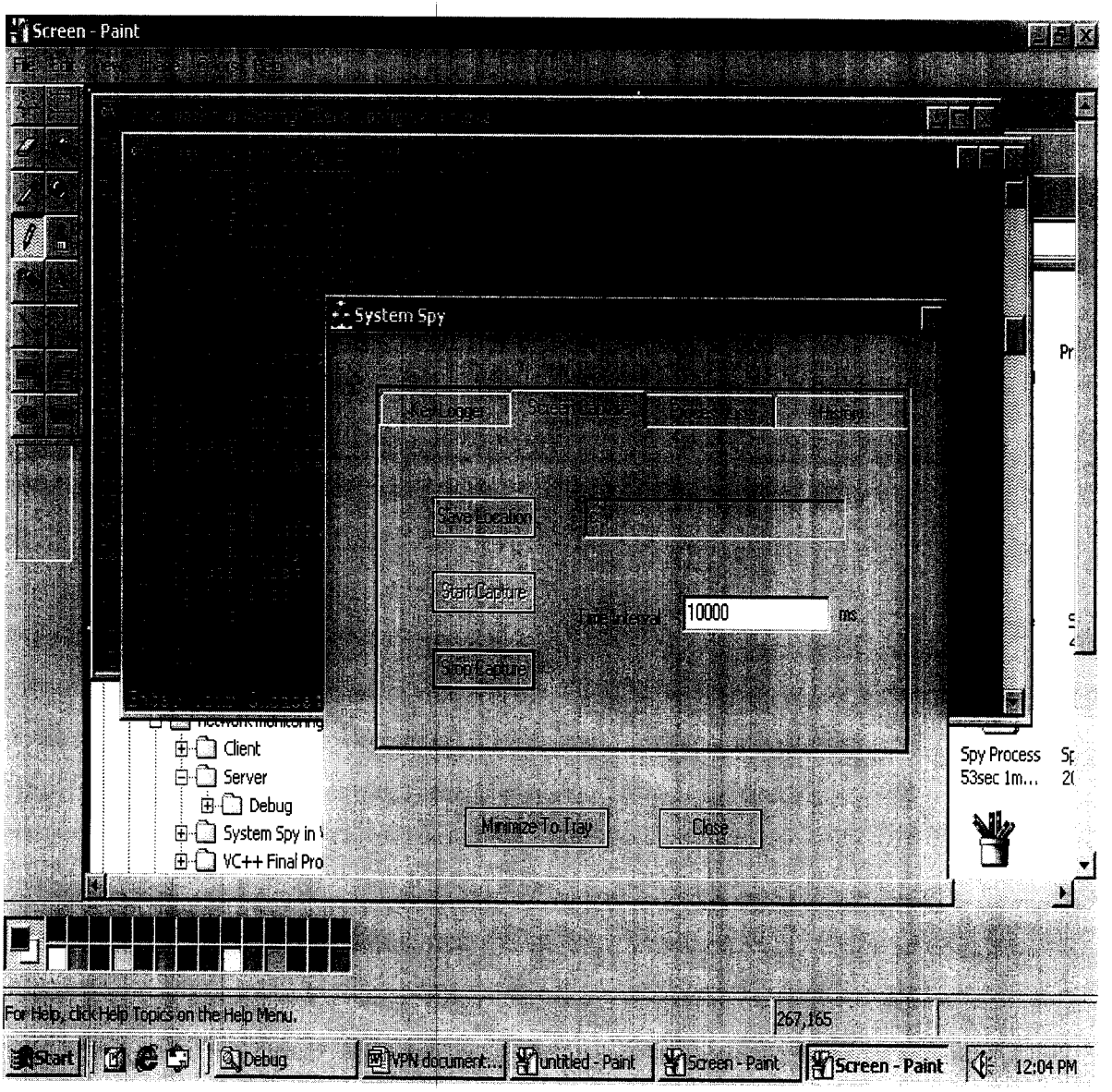
```
                printf("Error in opening");
                return 0;
        }

        while(1)
        {
                strcpy(senddata,"NEXTû");


ret=send(sockfd,senddata,lstrlen(senddata) + 1,0);
length=recv(sockfd,buf,sizeof(buf),0);
printf("\nThe received length is %d\n",length);

                i=0;

                while (buf[i]!='²')
                {
                        len[i]=buf[i];
                        i++;
                }
                len[i]='\0';

                for(i=0;i<strlen(len);i++)
                {
                        len[i]=len[i]-6;
                }
                printf("\nThe length string is %s",len);

                p_length=atoi(len);
```

```
                printf("\nThe length is %d",p_length);
                getch();

                for(i=tempint;i<length+tempint;i++)
                {
                        stdata[i-tempint]=buf[i];
                }

                stdata[i-tempint]='\0';

                printf("\nThe message is %s",stdata);

                for(i=0;i<p_length;i++)
                        data[i]=stdata[i
                decrypt(p_length);
        data[p_length]='\0';
        printf("\nThe Decrypted data is %s\n",data);



                for(i=0;i<3;i++)
                {
                        command[i]=data[i];
                }
                command[i]='\0';

                if (strcmp(command,"EOF")==0)

fwrite(data,sizeof(char),p_length,stream);
                }
                fclose(stream);
```

```
                strcpy(sys_str,"notepad.exe ");
                strcat(sys_str,file_name);
                system(sys_str);

        }
        else if (choice==7)
        {
                strcpy(senddata,"EXITû");
                ret=send(sockfd,senddata,lstrlen(senddata) +
1,0);
                return 0;
        }
 }
}
```

9.2) Sample Screen




Screen - Paint
System Spy
Key Logger
Screen Capture
Save Location
Start Capture
10000
ms
Stop Capture
Minimize To Tray
Close
Client
Server
Debug
System Spy in
VC++ Final Pro
Spy Process
53sec 1m...
For Help, click Help Topics on the Help Menu.
267,165
Start
Debug
VPN document...
untitled - Paint
Screen - Paint
Screen - Paint
12:04 PM


plist - Notepad
File Edit Format Help
SystemRoot\System32\smss.exe 448
\??\C:\WINDOWS\system32\winlogon.exe 528
C:\WINDOWS\system32\services.exe 572
C:\WINDOWS\system32\lsass.exe 584
C:\WINDOWS\system32\svchost.exe 780
C:\WINDOWS\System32\svchost.exe 844
C:\Program Files\Common Files\Symantec Shared\ccEvtMgr.exe 1072
C:\WINDOWS\system32\spoolsv.exe 1216
C:\WINDOWS\Explorer.EXE 1432
C:\Program Files\Common Files\Symantec Shared\ccApp.exe 1716
C:\WINDOWS\System32\ctfmon.exe 1752
C:\WINDOWS\System32\inetsrv\inetinfo.exe 2016
C:\Program Files\Common Files\Microsoft Shared\VS7Debug\mdm.exe 2032
C:\Program Files\Norton AntiVirus\navapsvc.exe 132
C:\Program Files\Norton AntiVirus\AdvTools\NPROTECT.EXE 180
C:\WINDOWS\System32\nvsvc32.exe 204
D:\Oracle\Ora81\BIN\TNSLSNR.exe 248
d:\oracle\ora81\bin\ORACLE.EXE 480
D:\Oracle\Ora81\BIN\OWASTSVR.EXE 1300
c:\windows\system32\srvany.exe 1248
C:\WINDOWS\System32\snmp.exe 1412
C:\service\XYNTService.exe 1492
D:\Oracle\Ora81\bin\oradim.exe 1616
C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\msdev.exe 2244
C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\MSDEV.EXE 2384
C:\KlSCPlH.exe 3368
C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\vcspawn.exe 3512
F:\Rajesh\Network Monitoring\ver6\ver5\Server\Debug\read.exe 3524
C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\vcspawn.exe 3596
F:\Rajesh\Network Monitoring\ver6\ver5\Client\Debug\configuration.exe 3612
Start
Server
plist - Notepad
12:09 PM