**VIRTUAL PRIVATE NETWORK (VPN)**

**ABSTRACT:**

A virtual private network (VPN) extends a private network across a public network, such as the Internet. It enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network. Applications running across the VPN may therefore benefit from the functionality, security, and management of the private network.

**INTRODUCTION:**

* In a VPN, private communication between two or more devices is achieved through a public network the Internet. Therefore, the communication is virtually but not physically there.
* The two devices that are communicating with each other in a public environment will have no third party who can interrupt this communication or receive any data that is exchanged between them.
* Simply put, a VPN, Virtual Private Network, is defined as a network that uses public network paths but maintains the security and protection of private networks.
* The two devices that are communicating will have a tunnel setup virtually. The sender will send the packets to the server which in turn sends the packet to the destination device after changing it. The server changes the IP Address in the ARP packets before forwarding it to the receiver.
* The server will have a database containing the original IP Address and the changed IP Address to retrace the packets back to the sender. Thus when the receiver sends an acknowledgement or data to the server, the server will trace the original IP and send the packet to the sender.

**WORKFLOW:**

* Initially, the client will send an authentication request to the VPN.
* The VPN on receiving an authentication request from the client, will securely authenticate via unique authentication key.
* When the client is successfully authenticated, the VPN will store the corresponding IP and Port address of the client in the database. It will also store the VPN assigned IP and port address.
* The database is implemented using the JDBC Api where the database is stored in the Localhost itself.
* On the client side, once a connection is established with the VPN, it asks for the server IP address and Port address.
* The user will then enter the message to be sent to the server.
* A datagram packet consisting the Server’s IP, Port address and data is encapsulated and sent to the VPN.
* The VPN will then decapsulate the packet to extract the Server’s IP and port address as well as the data to be sent.
* A new datagram packet is made consisting of the data from the client and is sent to the server with the VPN’s assigned addresses.
* The server on receiving the packet will display the message and then ask for a response. This response is then sent to the VPN i.e. the address from where the packet came.
* The VPN will then map the received packet to the corresponding client.
* On a “bye” message sent either by client or server, the entire established connection will be terminated.

**CODE:**

Server:

import java.util.\*;

import java.net.\*;

import java.io.\*;

public class UDPServer{

public static void main(String[] args) {

System.out.println();

Scanner sc=new Scanner(System.in);

String inMsg="",outMsg="";

byte[] inData,outData;

InetAddress clientAddress;

int clientPort;

try{

DatagramSocket ds=new DatagramSocket(8581);

try{

//ds=new DatagramSocket(8515);

System.out.println("Server IP : "+InetAddress.getLocalHost().getHostAddress());

System.out.println("Server Port : "+ds.getLocalPort());

DatagramPacket dsp,drp;

do{

inData=new byte[1024];

drp=new DatagramPacket(inData,inData.length);

ds.receive(drp);

clientAddress=drp.getAddress();

clientPort=drp.getPort();

inMsg=new String(drp.getData(),0,drp.getLength());

System.out.println("Client msg : "+inMsg);

if (inMsg.equalsIgnoreCase("bye")) {

break;

}

System.out.print("Enter something :");

outMsg=sc.nextLine();

outData=new byte[1024];

outData=outMsg.getBytes();

dsp=new DatagramPacket(outData,outData.length,clientAddress,clientPort);

ds.send(dsp);

}while (!outMsg.equalsIgnoreCase("bye"));

ds.close();

}

catch (Exception e) {

System.out.println(e.toString());

System.out.println();

}

finally{

ds.close();

}

}

catch(Exception e){}

}

}

VPN:

import java.util.\*;

import java.net.\*;

import java.io.\*;

import java.sql.\*;

public class UDP\_VPN{

static String encode(String[] array){

StringBuilder output=new StringBuilder();

for (String i:array ) {

output.append((i.length()+1)+"-"+i);

}

return output.toString();

}

static String[] decode(String encoded){

StringBuilder en=new StringBuilder(encoded);

ArrayList<String> output = new ArrayList<String>();

int len,i;

for (i=0,len=0;en.length()!=0;++i,len=0) {

while(Character.isDigit(en.charAt(0))){

len=len\*10+Integer.parseInt(en.charAt(0)+"");

en.deleteCharAt(0);

}

output.add(en.substring(1,len));//consider region after -

en.delete(0,len);

}

return output.toArray(new String[0]);

}

public static void main(String[] args) {

System.out.println();

Scanner sc=new Scanner(System.in);

String inMsg="",outMsg="";

//for time being they will be same

String secretAuthenticationKey = "AbraKaDabra", validAuthenticationRespnse="readyToUse";

byte[] inData,outData;

InetAddress clientAddress,serverAddress;

int clientPort,serverPort,vpnPublicPort=3333,vpnPrivatePort=5555;

String[] clientRequest,responseToClient= new String[3];

boolean validUser=false;

String vpnPublicIP="",vpnPrivateIP="";

try{

vpnPublicIP=InetAddress.getLocalHost().getHostAddress();

vpnPrivateIP=InetAddress.getLocalHost().getHostAddress();

}catch(Exception e){

e.printStackTrace();

}

try{

DatagramSocket ds=new DatagramSocket(vpnPublicPort);

DatagramPacket dsp,drp;

inData=new byte[1024];

drp=new DatagramPacket(inData,inData.length);

ds.receive(drp);

clientAddress=drp.getAddress();

clientPort=drp.getPort();

String clientaddr;

clientaddr=clientAddress.toString();

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/vpn","root","");

PreparedStatement stmt=con.prepareStatement("insert into ipad values(?,?,?,?)");

stmt.setString(1,clientaddr);

stmt.setInt(2,clientPort);

stmt.setString(3,vpnPrivateIP);

stmt.setInt(4,vpnPrivatePort);

int i=stmt.executeUpdate();

con.close();

inMsg=new String(drp.getData(),0,drp.getLength());

clientRequest = decode(inMsg);

System.out.println("Client msg : "+Arrays.toString(clientRequest));

if (clientRequest[2].equals(secretAuthenticationKey)) {

responseToClient[0]=vpnPrivateIP;//private ip address of vpn

responseToClient[1]=vpnPrivatePort+"";

responseToClient[2]=validAuthenticationRespnse;

validUser=true;

System.out.println("Client Authenticated\n");

}

else{

responseToClient[0]=vpnPrivateIP;

responseToClient[1]=vpnPublicPort+"";

responseToClient[2]="requestDenied";

validUser=false;

System.out.println("Unauthorized client");

}

outMsg=encode(responseToClient);

outData=new byte[1024];

outData=outMsg.getBytes();

dsp=new DatagramPacket(outData,outData.length,clientAddress,clientPort);

ds.send(dsp);

ds.close();

}

catch (Exception e) {

System.out.println(e.toString());

}

if (validUser) {

try{

DatagramSocket ds=new DatagramSocket(vpnPrivatePort);

try{

DatagramPacket dsp,drp;

do{

//take message from client

inData=new byte[1024];

drp=new DatagramPacket(inData,inData.length);

ds.receive(drp);

clientAddress=drp.getAddress();

clientPort=drp.getPort();

inMsg=new String(drp.getData(),0,drp.getLength());

clientRequest=decode(inMsg);

System.out.println("Client msg : "+Arrays.toString(clientRequest));

//forwarding client's message to server

serverAddress=InetAddress.getByName(clientRequest[0]);

serverPort=Integer.parseInt(clientRequest[1]);

outMsg=clientRequest[2];

outData=new byte[1024];

outData=outMsg.getBytes();

dsp=new DatagramPacket(outData,outData.length,serverAddress,serverPort);

ds.send(dsp);

if (outMsg.equalsIgnoreCase("bye")) {

break;

}

//take response msg from server

inData=new byte[1024];

drp=new DatagramPacket(inData,inData.length);

ds.receive(drp);

serverAddress=drp.getAddress();

serverPort=drp.getPort();

inMsg=new String(drp.getData(),0,drp.getLength());

//forwarding server response to client

responseToClient[0]=serverAddress.getHostAddress();

responseToClient[1]=serverPort+"";

responseToClient[2]=inMsg;

System.out.println("Server msg : "+Arrays.toString(responseToClient));

outMsg=encode(responseToClient);

outData=new byte[1024];

outData=outMsg.getBytes();

dsp=new DatagramPacket(outData,outData.length,clientAddress,clientPort);

ds.send(dsp);

}while (!inMsg.equalsIgnoreCase("bye")&&validUser);

ds.close();

}

catch (Exception e) {

System.out.println(e.toString());

}

finally{

ds.close();

System.out.println();

}

}catch(Exception e){}

}

}

}

CLIENT:

import java.util.\*;

import java.net.\*;

import java.io.\*;

public class UDPClient{

static String encode(String[] array){

StringBuilder output=new StringBuilder();

for (String i:array ) {

output.append((i.length()+1)+"-"+i);

}

return output.toString();

}

static String[] decode(String encoded){

StringBuilder en=new StringBuilder(encoded);

ArrayList<String> output = new ArrayList<String>();

int len,i;

for (i=0,len=0;en.length()!=0;++i,len=0) {

while(Character.isDigit(en.charAt(0))){

len=len\*10+Integer.parseInt(en.charAt(0)+"");

en.deleteCharAt(0);

}

output.add(en.substring(1,len));//consider region after -

en.delete(0,len);

}

return output.toArray(new String[0]);

}

public static void main(String[] args) {

System.out.println();

String inMsg = "", outMsg = "";

int format = 3, serverPort, vpnPublicPort = 3333,vpnPrivatePort;

boolean vpnConnect=false;

String vpnPublicIP="",serverIP;

InetAddress vpnPrivateIP;

String secretAuthenticationKey = "AbraKaDabra",validAuthenticationRespnse="readyToUse";

String[] authentication = new String[format];

String[] vpnDetails = new String[format];

String[] vpnResponse, server;

Scanner sc = new Scanner(System.in);

byte[] inData, outData;

try{

vpnPublicIP="10.120.102.227";//you can also make it as user input

DatagramSocket ds = new DatagramSocket();

DatagramPacket dsp, drp;

authentication[0] = vpnPublicIP;//public ip address of vpn -- dummy

authentication[1] = ""+vpnPublicPort;//public port of vpn

authentication[2] = secretAuthenticationKey;//key is required to establish a connection

outMsg = encode(authentication);

outData = outMsg.getBytes();

dsp = new DatagramPacket(outData, outData.length, InetAddress.getByName(vpnPublicIP), vpnPublicPort);

ds.send(dsp);

System.out.println("Asking for authentication from vpn "+secretAuthenticationKey);

inData = new byte[1024];

drp = new DatagramPacket(inData, inData.length);

ds.receive(drp);

inMsg = new String(drp.getData(), 0, drp.getLength());

vpnDetails = decode(inMsg);

if(vpnDetails[2].equals(validAuthenticationRespnse)){

vpnConnect = true;

vpnPrivateIP=InetAddress.getByName(vpnDetails[0]);//this can be different

vpnPrivatePort=Integer.parseInt(vpnDetails[1]);

System.out.println("Connection is established with vpn");

System.out.println("VPN Details : " + Arrays.toString(vpnDetails));

}

else{

System.out.println("Connection refused by vpn "+Arrays.toString(vpnDetails));

System.out.println();

}

}

catch(Exception e){

e.printStackTrace();

}

if (vpnConnect==true) {

//below part will run after connection is established with the vpn

System.out.print("Enter Initial Server IP : ");

serverIP = sc.next();

System.out.print("Enter Initial server Port : ");

serverPort = sc.nextInt();

sc.nextLine();

System.out.println();

server = new String[]{serverIP, "" + serverPort,""};

try{

DatagramSocket ds=new DatagramSocket();

DatagramPacket dsp,drp;

do{

System.out.print("Enter something : ");

server[2] = sc.nextLine();

outMsg = encode(server);

outData = outMsg.getBytes();

dsp = new DatagramPacket(outData, outData.length, InetAddress.getByName(vpnPublicIP), Integer.parseInt(vpnDetails[1]));

ds.send(dsp);

if (server[2].equalsIgnoreCase("bye")) {

break;

}

inData = new byte[1024];

drp = new DatagramPacket(inData,inData.length);

ds.receive(drp);

inMsg = new String(drp.getData(), 0, drp.getLength());

vpnResponse = decode(inMsg);

System.out.println("Server msg : " + vpnResponse[2]);

}while (!vpnResponse[2].equalsIgnoreCase("bye")&&vpnConnect);

ds.close();

}

catch (Exception e) {

System.out.println(e.toString());

}

}

}

}





