# SVS Bachelor-Projekt Network Security

## Blatt 3: Datenkommunikation

**Louis Kobras**
**6658699**

**Utz Pöhlmann**
**6663579**

## 1 HTTP

### 1.1 [local]

Funktioniert so nicht, da die Website über das HTTPS-Protokoll läuft, welches SSL erfodert, welches wiederum nicht von telnet unterstützt wird. Alternativ kann OpenSSL verwendet werden.

Es konnte ein Stylesheet ausgelesen werden; (s. [1]).
Kommentar: Zeilenumbrüche schaden nicht.

## 2 SMPT (Mail Spoofing)

### 2.1 [local]

- Verbinden zum Mailserver mit `telnet mailhost.informatik.uni-hamburg.de 25` (25 ist der Port des Servers)
- Eingabe folgender Befehle:
  - `EHLO svs-labwall.informatik.uni-hamburg.de`
  - `MAIL FROM: <svsg07@informatik.uni-hamburg.de>`
  - `RCPT TO: <4kobras@informatik.uni-hamburg.de>`
  - `DATA`
- folgender Mail-Text: `‘‘Hier könnte Ihr Inhalt stehen’’`
- Quelltextvergleich mit “echter” Mail ergab fehlende Konfigurationsinformationen
- Mangel nicht offensichtlich, kann durch Ergänzung der obigen Eingabe zwischen `DATA` und dem Mail-Text angepasst werden
- Fehlende Informationen:
  - `MIME-Version: 1.0`
  - ```
    Content-Type: text/plain;
            charset=ISO-8895-1;
            DelSp="Yes";
            format="flowed"
    ```
  - `Content-Disposition: inline`
  - `Content-Transfer-Encoding: 7bit`
  - `User-Agent: Internet Messaging Program (IMP) H3 (4.1.5)`
- Absender `gmail.com` ebenfalls möglich; folgender Eintrag ist zu modifizieren:
  - `EHLO google.com`
- Fehlerfall: Shell terminiert `telnet`-Ausführung mit dem Kommentar `‘‘I can break rules, too. Goodbye.’’`
    ```

# 3 License Server (DNS-Spoofing)

## 3.1 [local]

Protokoll:

1. Key als User-Input
2. Übermitteln des Keys an den Server
3. Rückgabe vom Server, ob Key gültig oder nicht (`SERIAL_VALID=0` bzw. `SERIAL_VALID=1`)

4a Wenn gültig, Dank für Kauf

4b Wenn nicht gültig, FBI ist unterwegs

## 3.2

- Verhindern der Kommunikation der Software mit dem echten Auth-Server
- Geschehen durch Erweitern des Hosts um `127.0.0.1 license-server.svslab` in `/etc/hosts`
- Herunterladen der Java-Klasse `TCPClient.java` ([2])
- Manipulieren des Servers: `ServerSocket` auf svslab-Port (1337) gesetzt
- Manipulieren des Servers: Rückgabe des Servers auf statisch ‘‘`SERIAL_VALID=1`’’ gesetzt
- $\Longrightarrow$ alle Keys gültig, unabhängig von Eingabe

## 3.3

Es gibt zwei anmerkbare Mängel.

1. Es sollte nicht angegeben werden, ob die Serial-Länge korrekt ist.
2. Es könnte mithilfe einer eindeutigen Signatur o.Ä. eine Abfrage an den Server eingebunden werden, ob er “echt” ist (gehasht).

# 4 License Server (Brute-Force-Angriff)

## 4.1 [local]

Das Programm funktioniert an sich, wenn man aber an den Server sendet, kriegt man (scheinbar nach Zufall) entweder “invalid command” oder “invalid length” zurück, bei Eingabe von `serial=abcdefgh` ($a, b, c, d, e, f, g, h \in \{0, 1, ..., 9\}$).

Wir baten zwei Gruppen neben uns um Hilfe, jedoch konnten diese uns auch nicht weiterhelfen bzw. haben keinen Fehler in unserem Programm gefunden.

Als Ausgangspunkt wurde die Java-Klasse `TCPClient.java` (source: [2]) genommen.

Einige gültige Keys:

- 03133700
- 06264700
- 09401100
- 15668500
- 18802200
- 21935900
- 25069600
- 31337000
- 47005500
- 59540300
- 62674000
- 72075100
- 87743600
- 90877300
- 94011000
- 97144700

### 4.2

Möglichkeiten, sich zu verteidigen, enthalten, sind jedoch nicht beschränkt auf:

- Sperren des Absenders der Auth-Anfrage nach $n$ Fehlversuchen (Unterbrechen von Brute-Force-Attacken)[1]
- Prüfung der IP bzw. Prüfsumme, ob Empfänger und Absender korrekt sind (Zurechenbarkeit)
- Limitieren der Eingabe auf $k$ pro Minute (Verlangsamen von Brute-Force-Attacken)

### 4.3 [local]

Alle gefundenen gültigen Schlüssel sind durch 100 teilbar, liegen also in der Form $xxxxx00$ vor. Ist ein Schlüssel außderdem durch 1000 teilbar, also in der Form $xxxxx000$, so ist ebenso ein Schlüssel der Form $0xxxxx00$ gültig, der die gleiche Ziffernfolge anstelle der $x$ enthält.

## 5 Implementieren eines TCP-Chats

### 5.1 [local]

Es werden zwei URLs in Fragmenten gesendet. Zusammengesetzt sehen sie folgendermaßen aus:

- [URL1] `http://www.oracle.com/technetwork/java/socket-140484.html`
- [URL2] `http://code.google.com/p/example-of-servlet/source/browse/trunk/src/javaLanguage/basic/?r=56#basic%2Fsocket`

Da UDP ein unzuverlässiges Protokoll ist, war es wie erwartet nötig, einige Zeit zu warten, bis alle Fragmente empfangen wurden (jeweils 4).

### 5.2

### 5.3

### 5.4

### 5.5

## Literatur

[1] `https://www.inf.uni-hamburg.de/assets/application-11e3b49e605\ff8ba1f01d275bd36850edfdfc1fbbb8c22e55fae1baf643a00d0.css`

[2] `https://systembash.com/a-simple-java-tcp-server-and-tcp-client/`

[1] Je nach Art der Sperrung ist dies lediglich eine Bremse; wird z.B. nur die IP gesperrt, kann diese resettet werden, um wieder Zugang zu erlangen.

# Anhang 1: Quelltext zu Aufgabe 3

**Klasse GeneratorTool.java**

```
1  import java.util.ArrayList;
2
3  public class GeneratorTool {
4      // Halterung für die Walzen
5      private ArrayList<Integer> _walzen;
6      // Speicher für das aktuelle Passwort
7      private String _passwort;
8      // Halterung für alle gültigen Symbole//
9      private ArrayList<String> _charListe;
10     // Liste aller gültigen Symbole
11     private final String _symbols;
12     // Walze, die zur Zeit die letzte ist, welche bearbeitet wird.
13     private int _aktuelleWalze;
14
15     /**
16      * Konstruktor. Probiert automatisch alle Passwörter durch
17      */
18     public GeneratorTool() {
19         _walzen = new ArrayList<Integer>();
20         setupWalzenListe();
21         _passwort = "";
22         _symbols = "0123456789";
23         _charListe = new ArrayList<String>();
24         setupCharListe();
25         _aktuelleWalze = 0;
26     } // end Konstruktor
27
28     /**
29      * Iteriert über die Walzen, bis die Abbruchbedingung erfüllt ist oder
30      * alle Werte ausprobiert wurden
31      */
32     public void findeEinPasswort() {
33         _aktuelleWalze = 7;
34         _passwort = "";
35         if (_aktuelleWalze != _walzen.size()) {
36             tick(_walzen.get(_aktuelleWalze));
37             // System.out.println(_passwort);
38         }
39     } // end findePasswort()
40
41     /**
42      * Dreht die gegebene Walze um ein Feld weiter. Bei Überlauf wird die
           Walze
43      * zurückgesetzt und die nächste Walze rekursiv aufgerufen. Anschließ
           end
44      * wird das zum aktuellen Walzenstand gehörende Passwort generiert.
45      *
46      * @param walze
47      *             der Stand der aktuellen Walze
48      * @param walzenIndex
49      *             der Index der aktuellen Walze (wichtig zum Ändern des
           Wertes
50      *             im Walzenarray)
51      */
52     private void tick(int walze) {
```

```
53          walze += 1;
54          _walzen.set(_aktuelleWalze, walze); // setzt den Wert der Walze
                auch im
55          // Array
56          if (walze == _symbols.length()) {
57              _aktuelleWalze = rolleWalze(_aktuelleWalze); // setzt die nä
                    chste
58              // Walze weiter
59              // walze = _walzen.get(walzenIndex);
60              if (_aktuelleWalze < _walzen.size()) {
61                  walze = 0; // setzt die aktuelle Walze zurück
62                  _walzen.set(_aktuelleWalze, walze); // setzt den Wert der
                        Walze
63                  // auch im Array
64              }
65          }
66          if (_aktuelleWalze < _walzen.size()) {
67              for (int w : _walzen) {
68                  if (w >= 0) {
69                      _passwort += getSymbol(w);
70                  }
71              }
72          }
73      } // end tick(int)
74
75      /**
76       * Wird aufgerufen, sobald eine Walte komplett durchgedreht hat. Setzt
             die
77       * nächste Walze einen Index weiter. Methode: Walze 1 dreht durch.
             Walze 1
78       * wird wieder auf Wert 1 gesetzt. Walze 2 dreht durch. Walze 2 wird
             wieder
79       * auf Wert 1 gesetzt und Walze 1 wird einen Wert weiter gesetzt. Walte
              2
80       * dreht durch. usw.
81       *
82       * @param walzenIndex
83       *            der Index der aktuell durch gedrehten Walze
84       * @return der Index der neuen aktuellen Walze
85       */
86      private int rolleWalze(int walzenIndex) {
87          if (walzenIndex == 0) {
88              if (_walzen.get(walzenIndex) >= _symbols.length() - 1) {
89                  _walzen.set(walzenIndex, 0);
90                  return walzenIndex + 1; // gehe zur nächsten Walze
91              } else {
92                  _walzen.set(walzenIndex, _walzen.get(walzenIndex) + 1);
93                  return walzenIndex;
94              }
95          } else if (0 < walzenIndex && walzenIndex < _walzen.size()) {
96              if (_walzen.get(walzenIndex) >= _symbols.length() - 1) {
97                  _walzen.set(walzenIndex, 0);
98                  return rolleWalze(walzenIndex - 1) + 1; // setze letzte
                        Walze +1
99              } else {
100                 _walzen.set(walzenIndex, _walzen.get(walzenIndex) + 1);
101                 return walzenIndex;
102             }
103         }
```

```
104         return -1; // Hier kommt das Programm nie an
105     }
106 
107     /**
108      * Holt ein Symbol aus der Symbolliste
109      *
110      * @param index
111      *            der Index des Symbols
112      * @return das Symbol
113      */
114     private String getSymbol(int index) {
115         return "" + _symbols.charAt(index);
116     } // end getSymbol()
117 
118     /**
119      * setzt die gültigen Symbole in einer Liste auf
120      */
121     private void setupCharListe() {
122         for (int i = 0; i < _symbols.length(); i++) {
123             _charListe.add(i, _symbols.substring(i, i + 1));
124         }
125     } // end setupCharListe()
126 
127     /**
128      * setzt die Walzen auf und sortiert sie in einer Liste
129      */
130     private void setupWalzenListe() {
131         int walze0 = 0;
132         int walze1 = 0;
133         int walze2 = 0;
134         int walze3 = 0;
135         int walze4 = 0;
136         int walze5 = 0;
137         int walze6 = 0;
138         int walze7 = -1;
139         _walzen.add(0, walze0);
140         _walzen.add(1, walze1);
141         _walzen.add(2, walze2);
142         _walzen.add(3, walze3);
143         _walzen.add(4, walze4);
144         _walzen.add(5, walze5);
145         _walzen.add(6, walze6);
146         _walzen.add(7, walze7);
147     } // end setupWalzenListe()
148 
149     /**
150      * gettermethode für _passwort
151      *
152      * @return _passwort
153      */
154     public String getPasswort() {
155         return _passwort;
156     } // end get_passwort()
157 
158 } // end class
```

**Klasse TCPClient.java**

```java
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;
import java.util.Scanner;

class TCPClient {

    public static boolean _keyGefunden;

    public static void main(String argv[]) throws Exception {
        GeneratorTool gt = new GeneratorTool();
        _keyGefunden = false;

        String sentence;
        String modifiedSentence;
        Socket clientSocket = new Socket("localhost", 1337);
        do {
        DataOutputStream outToServer = new DataOutputStream(
                clientSocket.getOutputStream());
        BufferedReader inFromServer = new BufferedReader(new
            InputStreamReader(
                clientSocket.getInputStream()));

            //gt.findeEinPasswort();
            //sentence = "serial=".concat(gt.getPasswort());
            sentence = "help";
            System.out.println(sentence);
            //outToServer.writeBytes(sentence);
            outToServer.writeBytes(sentence + '\n');
            //outToServer.writeBytes(new Scanner(System.in).nextLine());
            try{Thread.sleep(500);}catch(Exception e){}
            modifiedSentence = inFromServer.readLine();
            if (modifiedSentence.contains("SERIAL_VALID=1")
                    || sentence.equals(""))
                _keyGefunden = true;
            System.out.println("FROM␣SERVER:␣" + modifiedSentence);
        } while (!_keyGefunden);
        clientSocket.close();
    }
}
```

**Klasse TCPServer.java**

```java
import java.io.*;
import java.net.*;

class TCPServer
{
   public static void main(String argv[]) throws Exception
      {
         String clientSentence;
         String capitalizedSentence;
         ServerSocket welcomeSocket = new ServerSocket(1337);

         while(true)
         {
```

```
14              Socket connectionSocket = welcomeSocket.accept();
15              BufferedReader inFromClient =
16                 new BufferedReader(new InputStreamReader(connectionSocket.
                       getInputStream()));
17              DataOutputStream outToClient = new DataOutputStream(
                    connectionSocket.getOutputStream());
18              clientSentence = inFromClient.readLine();
19              System.out.println("Received: " + clientSentence);
20              capitalizedSentence = clientSentence.toUpperCase() + '\n';
21              outToClient.writeBytes("SERIAL_VALID=1");
22           }
23        }
24 }
```

# Anhang 2: Quelltext zu Aufgabe 4

### Klasse GeneratorTool.java

wie in Aufgabe 3

### Klasse TCPClient.java

wie in Aufgabe 3

# Anhang 3: Quelltext zu Aufgabe 5

## Aufgabe 5.1

### Klasse UDPClient.java

```
1  package blatt03.fuenf.eins;
2
3  import java.net.DatagramPacket;
4  import java.net.DatagramSocket;
5
6  public class UDPClient {
7      public static void main(String args[]) throws Exception {
8          while(true){
9          DatagramSocket clientSocket = new DatagramSocket(9999);
10         byte[] receiveData = new byte[1024];
11         DatagramPacket receivePacket = new DatagramPacket(receiveData,
12                 receiveData.length);
13         clientSocket.receive(receivePacket);
14         String modifiedSentence = new String(receivePacket.getData());
15         System.out.println("FROM SERVER:" + modifiedSentence + "\n");
16         clientSocket.close();
17     }}
18 }
```

## Aufgabe 5.2

### Klasse ClientWorker.java

```
1  package blatt03.fuenf.zwei;
2
3  import java.awt.event.ActionEvent;
4  import java.io.BufferedReader;
5  import java.io.IOException;
6  import java.io.InputStreamReader;
7  import java.io.PrintWriter;
8  import java.net.Socket;
9  import java.net.UnknownHostException;
10
11 import javax.swing.JTextArea;
12
13 class ClientWorker implements Runnable {
14       private Socket _client;
15       private JTextArea _textArea;
16         private SocketClientUI _ui;
17         private Socket _socket;
18         private BufferedReader _in = null;
19         private PrintWriter _out = null;
20
21     //Constructor
22       ClientWorker(Socket client, JTextArea textArea) {
23         _client = client;
24         _textArea = textArea;
25       }
26
27       public void listenSocket(){
28
29             //Create socket connection
30                try{
31                  _socket = new Socket("kq6py", 4444);
32                  _out = new PrintWriter(_socket.getOutputStream(),
33                              true);
34                  _in = new BufferedReader(new InputStreamReader(
35                            _socket.getInputStream()));
36                } catch (UnknownHostException e) {
37                  System.out.println("Unknown␣host:␣kq6py");
38                  System.exit(1);
39                } catch  (IOException e) {
40                  System.out.println("No␣I/O");
41                  System.exit(1);
42                }
43             }
44
45         public void actionPerformed(ActionEvent event){
46                Object source = event.getSource();
47
48                if(source == _ui.getButton()){
49             //Send data over socket
50                   String text = _ui.getTextArea().getText();
51                   _out.println(text);
52                   _ui.getTextArea().setText(new String(""));
53                   _out.println(text);
54                }
55             //Receive text from server
56                try{
57                  String line = _in.readLine();
58                  System.out.println("Text␣received:␣" + line);
```

```
59                } catch (IOException e){
60                  System.out.println("Read failed");
61                  System.exit(1);
62                }
63              }
64
65        public void run(){
66          String line;
67          BufferedReader in = null;
68          PrintWriter out = null;
69          try{
70            in = new BufferedReader(new
71              InputStreamReader(_client.getInputStream()));
72            out = new
73              PrintWriter(_client.getOutputStream(), true);
74          } catch (IOException e) {
75            System.out.println("in or out failed");
76            System.exit(-1);
77          }
78
79          while(true){
80            try{
81              line = in.readLine();
82      //Send data back to client
83              out.println(line);
84      //Append data to text area
85              appendText(line);
86             }catch (IOException e) {
87              System.out.println("Read failed");
88              System.exit(-1);
89             }
90          }
91        }
92
93        public synchronized void appendText(String line){
94              _textArea.append(line);
95            }
96
97        protected void finalize(){
98          //Objects created in run method are finalized when
99          //program terminates and thread exits
100              try{
101                 _socket.close();
102               } catch (IOException e) {
103                 System.out.println("Could not close socket");
104                 System.exit(-1);
105               }
106           }
107     }
```

**Klasse Server.java**

```
1  package blatt03.fuenf.zwei;
2
3  import java.awt.event.ActionEvent;
4  import java.io.BufferedReader;
5  import java.io.IOException;
6  import java.io.InputStreamReader;
```

```
 7 import java.io.PrintWriter;
 8 import java.net.ServerSocket;
 9 import java.net.Socket;
10
11 public class Server {
12
13     private SocketServerUI _ui;
14     private String _line;
15
16     public static void main(String args[]) {
17         new Server().listenSocket();
18     }
19
20     public Server() {
21         _ui = new SocketServerUI();
22
23     }
24
25     public void listenSocket() {
26         ServerSocket server = null;
27         Socket client = null;
28         System.out.println("1");
29         try {
30             server = new ServerSocket(4444);
31         } catch (IOException e) {
32             System.out.println("Could␣not␣listen␣on␣port␣4444");
33             System.exit(-1);
34         }
35         System.out.println("2");
36         try {
37             client = server.accept();
38         } catch (IOException e) {
39             System.out.println("Accept␣failed:␣4444");
40             System.exit(-1);
41         }
42         System.out.println("3");
43         try {
44             new BufferedReader(new InputStreamReader(
45                     client.getInputStream()));
46             new PrintWriter(client.getOutputStream(), true);
47         } catch (IOException e) {
48             System.out.println("Read␣failed");
49             System.exit(-1);
50         }
51         System.out.println("4");
52
53         while(true){
54             ClientWorker w;
55             //server.accept returns a client connection
56                   System.out.println("5");
57                   w = new ClientWorker(client, _ui.getTextArea());
58                   Thread t = new Thread((Runnable) w);
59                   t.start();
60                   try{Thread.sleep(1000);}catch(Exception e){}
61           }
62     }
63
64     public void actionPerformed(ActionEvent event) {
65         Object source = event.getSource();
```

```
66
67          if (source == _ui.getButton()) {
68              _ui.getTextArea().setText(_line);
69          }
70      }
71  }
```

**Klasse SocketClient.java**

```
1   package blatt03.fuenf.zwei;
2
3   import java.awt.event.ActionEvent;
4   import java.io.BufferedReader;
5   import java.io.IOException;
6   import java.io.InputStreamReader;
7   import java.io.PrintWriter;
8   import java.net.Socket;
9   import java.net.UnknownHostException;
10
11  public class SocketClient {
12
13      private SocketServerUI _ui;
14      private Socket _socket;
15      private BufferedReader _in = null;
16      private PrintWriter _out = null;
17
18      public static void main(String args[]) {
19          new SocketClient().listenSocket();
20      }
21
22      public SocketClient() {
23          _ui = new SocketServerUI();
24
25      }
26
27      public void listenSocket(){
28
29          //Create socket connection
30             try{
31               _socket = new Socket("kq6py", 4321);
32               _out = new PrintWriter(_socket.getOutputStream(),
33                           true);
34               _in = new BufferedReader(new InputStreamReader(
35                           _socket.getInputStream()));
36             } catch (UnknownHostException e) {
37               System.out.println("Unknown␣host:␣kq6py");
38               System.exit(1);
39             } catch  (IOException e) {
40               System.out.println("No␣I/O");
41               System.exit(1);
42             }
43          }
44
45      public void actionPerformed(ActionEvent event){
46              Object source = event.getSource();
47
48             if(source == _ui.getButton()){
49          //Send data over socket
```

```
50                String text = _ui.getTextArea().getText();
51                _out.println(text);
52                _ui.getTextArea().setText(new String(""));
53                _out.println(text);
54             }
55          //Receive text from server
56             try{
57               String line = _in.readLine();
58               System.out.println("Text␣received:␣" + line);
59             } catch (IOException e){
60               System.out.println("Read␣failed");
61               System.exit(1);
62             }
63          }
64 }
```

**Klasse SocketClientUI.java**

```
1  package blatt03.fuenf.zwei;
2
3  import java.awt.BorderLayout;
4
5  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JTextArea;
8
9  public class SocketClientUI {
10     private JFrame _frame;
11     private JTextArea _text;
12     private JButton _button;
13     public SocketClientUI(){
14         _frame = new JFrame();
15         _text = new JTextArea();
16         _button = new JButton("Senden");
17         _frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18
19
20
21         _frame.setLayout(new BorderLayout());
22         _frame.add(_text, BorderLayout.CENTER);
23         _frame.add(_button, BorderLayout.SOUTH);
24         _frame.pack();
25         _frame.setVisible(true);
26     }
27     public JTextArea getTextArea() {
28         return _text;
29
30     }
31     public JButton getButton() {
32         return _button;
33
34     }
35 }
```

**Klasse SocketServer.java**

```java
package blatt03.fuenf.zwei;

import java.awt.event.ActionEvent;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class SocketServer {

    private SocketServerUI _ui;
    private String _line;

    public static void main(String args[]) {
        new SocketServer().listenSocket();
    }

    public SocketServer() {
        _ui = new SocketServerUI();

    }

    public void listenSocket() {
        ServerSocket server = null;
        Socket client = null;
        BufferedReader in = null;
        PrintWriter out = null;

        try {
            server = new ServerSocket(4321);
        } catch (IOException e) {
            System.out.println("Could not listen on port 4321");
            System.exit(-1);
        }
        try {
            client = server.accept();
        } catch (IOException e) {
            System.out.println("Accept failed: 4321");
            System.exit(-1);
        }
        try {
            in = new BufferedReader(new InputStreamReader(
                    client.getInputStream()));
            out = new PrintWriter(client.getOutputStream(), true);
        } catch (IOException e) {
            System.out.println("Read failed");
            System.exit(-1);
        }

        while (true) {
            try {
                _line = in.readLine();
                // Send data back to client
                out.println(_line);
            } catch (IOException e) {
                System.out.println("Read failed");
```

```
59                  System.exit(-1);
60              }
61          }
62      }
63
64      public void actionPerformed(ActionEvent event) {
65          Object source = event.getSource();
66
67          if (source == _ui.getButton()) {
68              _ui.getTextArea().setText(_line);
69          }
70      }
71  }
```

**Klasse SocketServerUI.java**

```
1   package blatt03.fuenf.zwei;
2
3   import java.awt.BorderLayout;
4
5   import javax.swing.JButton;
6   import javax.swing.JFrame;
7   import javax.swing.JTextArea;
8
9   public class SocketServerUI {
10
11      private JFrame _frame;
12      private JTextArea _text;
13      private JButton _button;
14      public SocketServerUI(){
15          _frame = new JFrame();
16          _text = new JTextArea();
17          _button = new JButton("Empfangen");
18          _frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19
20
21
22          _frame.setLayout(new BorderLayout());
23          _frame.add(_text, BorderLayout.CENTER);
24          _frame.add(_button, BorderLayout.SOUTH);
25          _frame.pack();
26          _frame.setVisible(true);
27      }
28      public JTextArea getTextArea() {
29          return _text;
30
31      }
32      public JButton getButton() {
33          return _button;
34
35      }
36
37      public void setText(String text){
38          _text.setText(_text.getText() + "\n" + text);
39          _frame.pack();
40      }
41  }
```