

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 erfordert, welches wiederum nicht von telnet unterstützt wird. Alternativ kann OpenSSL verwendet werden.

Es konnte ein Stylesheet ausgelesen werden; (s. [1]).

Kommentar: Zeilenumbrüche schaden nicht (Anzeige des Quelltextes enthielt weder Zeilenumbrüche noch (dementsprechend) Einrückung).

2 SMTP (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’`
- Beenden der Text-Eingabe mit einer Leerzeile, gefolgt von einer Zeile, die nur einen Punkt (‘.’) enthält
→ Mail wird abgesendet
- 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 `TCPCClient.java` ([2])
- Manipulieren des Servers: `ServerSocket` auf `svslab`-Port (1337) gesetzt
- Manipulieren des Servers: Rückgabe des Servers auf statisch `"SERIAL_VALID=1"` gesetzt
- \implies 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` ($a, b, c, d, e, f, g, h \in \{0, 1, \dots, 9\}$).

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

Einige gültige Keys:

- | | | | |
|------------|------------|------------|------------|
| • 03133700 | • 18802200 | • 47005500 | • 87743600 |
| • 06264700 | • 21935900 | • 59540300 | • 90877300 |
| • 09401100 | • 25069600 | • 62674000 | • 94011000 |
| • 15668500 | • 31337000 | • 72075100 | • 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 $xxxxxx00$ vor. Ist ein Schlüssel außerdem 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

- Starten der Klasse `Server.java` mit Port 4321
- Verbinden mit dem Server durch `telnet localhost 4321`
- Kommunikation zwischen Server und Terminal erfolgreich

5.3

Wir waren nicht in der Lage, den Server derart zu restrukturieren, dass er sich mit mehreren Clients verbindet.

5.4

Fehlschlag, da Vorbedingung (A5.3) fehlt.

5.5

Fehlschlag, da Vorbedingung (A5.3, A5.4) fehlt.

Eine solche Sicherheitsmethode hilft bei einem menschlichen Gegner, der schließlich dem Verdruss der Langsamkeit erliegt. Wenn ein Algorithmus 10.000 Passwörter pro Sekunde prüfen kann, wird die Laufzeit um den Faktor 10.000 verlängert. Einen Menschen kann man damit mittelmäßig gut abwehren, denn er wird irgendwann das Interesse verlieren (Wenn er derart motiviert ist, sich davon nicht aufhalten zu lassen, kommt er auch so an das Passwort ran). Eine Maschine jedoch kann (noch) nicht der Langeweile erliegen und lässt sich somit von einer Drosselung nicht auf Dauer aufhalten.

¹Je 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.

Literatur

- [1] <https://www.inf.uni-hamburg.de/assets/application-11e3b49e605\ff8ba1f01d275bd36850edfdcf1fbbb8c22e55fae1baf643a00d0.css>
- [2] <https://systembash.com/a-simple-java-tcp-server-and-tcp-client/>

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
43     * Walze
44     * zurückgesetzt und die nächste Walze rekursiv aufgerufen. Anschließend
45     * wird das zum aktuellen Walzenstand gehörende Passwort generiert.
46     *
47     * @param walze
48     *           der Stand der aktuellen Walze
49     * @param walzenIndex
50     *           der Index der aktuellen Walze (wichtig zum Ändern des
51     *           Wertes
52     *           im Walzenarray)
53     */
54    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 Walze 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. Walze
    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

```
1 import java.io.BufferedReader;
2 import java.io.DataOutputStream;
3 import java.io.InputStreamReader;
4 import java.net.Socket;
5 import java.util.Scanner;
6
7 class TCPClient {
8
9     public static boolean _keyGefunden;
10
11     public static void main(String argv[]) throws Exception {
12         GeneratorTool gt = new GeneratorTool();
13         _keyGefunden = false;
14
15         String sentence;
16         String modifiedSentence;
17         Socket clientSocket = new Socket("localhost", 1337);
18         do {
19             DataOutputStream outToServer = new DataOutputStream(
20                 clientSocket.getOutputStream());
21             BufferedReader inFromServer = new BufferedReader(new
22                 InputStreamReader(
23                     clientSocket.getInputStream()));
24
25             //gt.findeEinPasswort();
26             //sentence = "serial=".concat(gt.getPasswort());
27             sentence = "help";
28             System.out.println(sentence);
29             //outToServer.writeBytes(sentence);
30             outToServer.writeBytes(sentence + '\n');
31             //outToServer.writeBytes(new Scanner(System.in).nextLine());
32             try{Thread.sleep(500);}catch(Exception e){}
33             modifiedSentence = inFromServer.readLine();
34             if (modifiedSentence.contains("SERIAL_VALID=1")
35                 || sentence.equals(""))
36                 _keyGefunden = true;
37             System.out.println("FROM_SERVER: " + modifiedSentence);
38         } while (!_keyGefunden);
39         clientSocket.close();
40     }
41 }
```

Klasse TCPServer.java

```
1 import java.io.*;
2 import java.net.*;
3
4 class TCPServer
5 {
6     public static void main(String argv[]) throws Exception
7     {
8         String clientSentence;
9         String capitalizedSentence;
10         ServerSocket welcomeSocket = new ServerSocket(1337);
11
12         while(true)
13         {
```



```
14      Socket connectionSocket = welcomeSocket.accept();
15      BufferedReader inFromClient =
16          new BufferedReader(new InputStreamReader(connectionSocket.
17              getInputStream()));
18      DataOutputStream outToClient = new DataOutputStream(
19          connectionSocket.getOutputStream());
20      clientSentence = inFromClient.readLine();
21      System.out.println("Received:␣" + clientSentence);
22      capitalizedSentence = clientSentence.toUpperCase() + '\n';
23      outToClient.writeBytes("SERIAL_VALID=1");
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     public void listenSocket(){
27
28         //Create socket connection
29         try{
30             _socket = new Socket("kq6py", 4321);
31             _out = new PrintWriter(_socket.getOutputStream(),
32                                     true);
33             _in = new BufferedReader(new InputStreamReader(
34                                     _socket.getInputStream()));
35         } catch (UnknownHostException e) {
36             System.out.println("Unknown host: kq6py");
37             System.exit(1);
38         } catch (IOException e) {
39             System.out.println("No I/O");
40             System.exit(1);
41         }
42     }
43
44     public void actionPerformed(ActionEvent event){
45         Object source = event.getSource();
46
47         if(source == _ui.getButton()){
48             //Send data over socket
49         }
```

```
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

```
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 SocketServer {
12
13     private SocketServerUI _ui;
14     private String _line;
15
16     public static void main(String args[]) {
17         new SocketServer().listenSocket();
18     }
19
20     public SocketServer() {
21         _ui = new SocketServerUI();
22     }
23
24     public void listenSocket() {
25         ServerSocket server = null;
26         Socket client = null;
27         BufferedReader in = null;
28         PrintWriter out = null;
29
30         try {
31             server = new ServerSocket(4321);
32         } catch (IOException e) {
33             System.out.println("Could not listen on port 4321");
34             System.exit(-1);
35         }
36
37         try {
38             client = server.accept();
39         } catch (IOException e) {
40             System.out.println("Accept failed: 4321");
41             System.exit(-1);
42         }
43
44         try {
45             in = new BufferedReader(new InputStreamReader(
46                 client.getInputStream()));
47             out = new PrintWriter(client.getOutputStream(), true);
48         } catch (IOException e) {
49             System.out.println("Read failed");
50             System.exit(-1);
51         }
52
53         while (true) {
54             try {
55                 _line = in.readLine();
56                 // Send data back to client
57                 out.println(_line);
58             } catch (IOException e) {
59                 System.out.println("Read failed");
60             }
61         }
62     }
63 }
```

```
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 }
```