DAY 7、8——SSL 安全通信

问题 6：如何使用 SSL 实现安全网络通信？

7.1 学习内容

文件传输方法，请学习教程的**第 7 章 SSL 安全通信**。

7.2 任务清单

第一天：

① 试运行程序，掌握数据库的创建和使用。

② 阐述服务器端 Message、Translate、User、Member 四个类的作用。

③ 运行并阅读程序，描绘出主要函数的步骤和程序结构。

④ 将重要功能，记在电子笔记中。

第二天：

⑤ 学习和掌握 SSL socket 连接。分析程序中每条语句的作用。

⑥ 利用 wireshark 抓包验证自己的分析判断是否正确。

⑦ 一在小组内，抽签阐述服务器或客户端的软件结构，并录制视频。

7.3 关键技术

7.3.1 建立数据库文件

Netbeans 自带小型数据库 derby。可以创建本地的数据库，供程序使用。方法是以管理员身份运行 netBeans，在项目中的第三张卡片“服务”中，点击数据库下的 javaDB，右键选择创建数据库。在对话框中输入数据库的名称为 QQDB，用户名为“nbuser”，密码为“password”。创建成功后，出现 jdbc:derby://localhost:1527/QQDB,选中，点击连接。成功后，可以点击右键——执行命令，粘贴以下 SQL 语句，创建数据表中的内容：

```
/**
 * 定义 Member 表结构与初始化数据
 */
DROP TABLE MEMBER;
CREATE TABLE "MEMBER" (
    "ID" INTEGER not null primary key,
    "NAME" VARCHAR(32),
    "PASSWORD" VARCHAR(256),
    "EMAIL" VARCHAR(64),
    "HEADIMAGE" VARCHAR(128),
    "TIME" TIMESTAMP
);
INSERT INTO MEMBER VALUES(10000,'张三
','123456','zhangsan@163.com','i9000.jpg',current_timestamp);
INSERT INTO MEMBER VALUES(20000,'李四
','123456','lisi@163.com','i9001.jpg',current_timestamp);
```

```
    INSERT INTO MEMBER VALUES(30000,'王五
','123456','wangwu@163.com','i9002.jpg',current_timestamp);
    SELECT * FROM MEMBER
```

成功后，用 netbeans 打开\chap07\end\QQServer 项目。在源包中 cn.edu.ldu.db 下找到 DBTest.java 文件，点击鼠标右键——运行文件，会自动添加 ID=50000 的这个用户信息到数据库中。然后才可以双击 QQServer.jar 和 QQClient.jar 运行程序。QQ 用户 ID 输入“50000”，密码“password”，即可登录成功。

7.3.2 生成公钥和私钥

生成密钥的方法如下：

Java 自带 keytool 工具。首先，建议将 java 目录 C:\Program Files\Java\jdk1.8.0_111\bin 下的 keytool.exe 和运行时需要的动态连接库 jli.dll 拷贝到一个单独的目录下。例如，D:\keystore\ 下。

为了快速生成需要的所有密钥和证书，建议建立一个批处理文件 genkeyAll.bat，内容如下：

```
keytool -genkeypair -keystore d:\keystore\client.keystore -storepass 123456 -alias client -keyalg RSA -keysize 1024 -keypass 123456 -validity 365 < huida.txt
keytool -exportcert -rfc -alias client -file d:\keystore\dadong.cer -keystore d:\keystore\client.keystore -storepass 123456
keytool -importcert -alias client -file d:\keystore\dadong.cer -keypass 123456 -keystore d:\keystore\tserver.keystore -storepass 123456 <y.txt
```

keytool -genkeypair -keystore d:\keystore\server.keystore -storepass 123456 -alias server -keyalg RSA -keysize 1024 -keypass 123456 -validity 365 < serverhuida.txt

keytool -exportcert -rfc -alias server -file d:\keystore\server.cer -keystore d:\keystore\server.keystore -storepass 123456

keytool -importcert -alias server -file d:\keystore\server.cer -keypass 123456 -keystore d:\keystore\tclient.keystore -storepass 123456 <y.txt

由于在生成证书的过程中，需要输入一些单位、个人信息，因此将这些信息编辑成文件，采用重定向第方式提交给批处理中的命令。如<huida.txt 、<y.txt 。<号是重定向方向，表示从文件获取信息。默认是从键盘获取信息。

huida.txt 文件是客户端的用户信息，内容如下：

```
DaDong
School of software
Nanfang University
Nanchang
Jiangxi
CN
Y
```

Y.txt 内容如下，目的是给出确认回答 yes。

```
Y
```

serverhuida.txt 文件是服务器端的用户信息，内容如下：

```
Server
School of software
Nanfang University
Nanchang
Jiangxi
CN
Y
```

将生成文件，client.keystore、tclient.keystore 复制到自建的 client 目录下，server.keystore、tserver.keystore 放到自建 server 目录下，密钥生成完毕。

7.3.3 启动服务器 SSLSocket 监听

```
private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            //获取服务器工作地址端口
            String hostName=txtHostName.getText();
            int hostPort=Integer.parseInt(txtHostPort.getText());
            //创建 UDP 数据报套接字,在指定端口侦听
            DatagramSocket serverSocket=new DatagramSocket(hostPort);
            txtArea.append("服务器开始侦听...\n");
            //创建并启动 UDP 消息接收线程
            Thread recvThread=new ReceiveMessage(serverSocket,this);
            recvThread.start();
            //创建并启动文件接收线程
            new Thread(new Runnable() {
                public void run() {
                try {
                //获取客户机证书库
                InputStream key
=ServerUI.class.getResourceAsStream("/cn/edu/ldu/keystore/server.keystore");//私钥库
                InputStream tkey
=ServerUI.class.getResourceAsStream("/cn/edu/ldu/keystore/tserver.keystore");//公钥库
                String SERVER_KEY_STORE_PASSWORD = "123456"; //server.keystore 密码
```

```
String SERVER_TRUST_KEY_STORE_PASSWORD = "123456";//tserver.keystore 密码

SSLContext ctx = SSLContext.getInstance("SSL");//SSL 上下文

KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");

TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509");

KeyStore ks = KeyStore.getInstance("JKS");

KeyStore tks = KeyStore.getInstance("JKS");

//加载私钥证书库

ks.load(key, SERVER_KEY_STORE_PASSWORD.toCharArray());

//加载公钥证书库

tks.load(tkey, SERVER_TRUST_KEY_STORE_PASSWORD.toCharArray());

kmf.init(ks, SERVER_KEY_STORE_PASSWORD.toCharArray());

tmf.init(tks);

ctx.init(kmf.getKeyManagers(), tmf.getTrustManagers(), null);

//服务器侦听安全连接

SSLServerSocket sslListenSocket = (SSLServerSocket) ctx.getServerSocketFactory().createServerSocket(hostPort);

int processors=Runtime.getRuntime().availableProcessors();//CPU 数

ExecutorService  fixedPool=Executors.newFixedThreadPool(processors*2);// 创建固定大小线程池

while (true) { //处理所有客户机连接

    SSLSocket fileSocket=(SSLSocket)sslListenSocket.accept();//如果无连接，则阻塞，否则接受连接并创建新的会话套接字

    //文件接收线程为 SwingWorker 类型的后台工作线程
```

```
                    SwingWorker<Integer,Object>                    recver=new
RecvFile(fileSocket,ServerUI.this,tks,ks); //创建客户线程

                    fixedPool.execute(recver); //用线程池调度客户线程运行

                }//end while

                } catch (Exception ex) {

                    JOptionPane.showMessageDialog(null, ex.getMessage(), "错误提示",
JOptionPane.ERROR_MESSAGE);

                }//end try catch

                }//end run()

            }).start();

        } catch (NumberFormatException | SocketException ex) {

            JOptionPane.showMessageDialog(null, ex.getMessage(), "错误提示",
JOptionPane.ERROR_MESSAGE);

        }

        btnStart.setEnabled(false);

    }
```

7.3.4 客户端发送文件后台线程

```
public class FileSender extends SwingWorker<List<String>,String>{
    private File file; //文件
    private Message msg;//消息类
    private ClientUI parentUI; //父类
    private SSLSocket fileSocket; //传送文件的套接字
    private static final int BUFSIZE=8096; //缓冲区大小
    private int progress=0; //文件传送进度
    private String lastResults=null; //传送结果
    //构造函数
    public FileSender(File file,Message msg,ClientUI parentUI) {
        this.file=file;
```

```
            this.msg=msg;
            this.parentUI=parentUI;
        }
        @Override
        protected List<String> doInBackground() throws Exception {
            //用客户机密钥库初始化 SSL 传输框架
            InputStream key =ClientUI.class.getResourceAsStream("/cn/edu/ldu/keystore/client.keystore");//
私钥库
            InputStream                                                                    tkey
=ClientUI.class.getResourceAsStream("/cn/edu/ldu/keystore/tclient.keystore");//公钥库
            String CLIENT_KEY_STORE_PASSWORD = "123456"; //client.keystore 私钥库密码
            String CLIENT_TRUST_KEY_STORE_PASSWORD = "123456";//tclient.keystore 公钥库密码
            SSLContext ctx = SSLContext.getInstance("SSL"); //SSL 上下文
            KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509"); //私钥管理器
            TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509");//公钥管理器
            KeyStore ks = KeyStore.getInstance("JKS");//私钥库对象
            KeyStore tks = KeyStore.getInstance("JKS");//公钥库对象
            ks.load(key, CLIENT_KEY_STORE_PASSWORD.toCharArray());//加载私钥库
            tks.load(tkey, CLIENT_TRUST_KEY_STORE_PASSWORD.toCharArray());//加载公钥库
            kmf.init(ks, CLIENT_KEY_STORE_PASSWORD.toCharArray());//私钥库访问初始化
            tmf.init(tks);//公钥库访问初始化
            //用私钥库和公钥库初始化 SSL 上下文
            ctx.init(kmf.getKeyManagers(), tmf.getTrustManagers(), null);
            //用 SSLSocket 连接服务器
            fileSocket                                                                     =
(SSLSocket)ctx.getSocketFactory().createSocket(msg.getToAddr(),msg.getToPort());
            //构建套接字输出流
            DataOutputStream out=new DataOutputStream(
                                        new BufferedOutputStream(
                                        fileSocket.getOutputStream()));
            //获取客户机私钥
            PrivateKey privateKey=(PrivateKey)ks.getKey("client",
CLIENT_KEY_STORE_PASSWORD.toCharArray());
            //获取服务器公钥
            PublicKey publicKey=(PublicKey)tks.getCertificate("server").getPublicKey();
            //定义摘要算法
            MessageDigest sha256=MessageDigest.getInstance("SHA-256");//256 位
            //构建文件输入流
            DataInputStream din=new DataInputStream(
                                        new BufferedInputStream(
                                        new FileInputStream(file)));
```

```
        //基于输入流和摘要算法构建消息摘要流
        DigestInputStream in=new DigestInputStream(din,sha256);
                DataOutputStream out1=new DataOutputStream(
                                new BufferedOutputStream(
                                  new FileOutputStream("d:\\a.dat")));


        long fileLen=file.length();   //计算文件长度
        //1.发送文件名称、文件长度
        out.writeUTF(file.getName());
        out.writeLong(fileLen);
        out.flush();
        parentUI.txtArea.append("1.发送文件名称、文件长度成功！\n");
        //2.传送文件内容
        int numRead=0; //单次读取的字节数
        int numFinished=0; //总完成字节数
        byte[] buffer=new byte[BUFSIZE];
        while (numFinished<fileLen && (numRead=in.read(buffer))!=-1) { //文件可读
              out.write(buffer,0,numRead);   //发送
              out.flush();
                out1.write(buffer,0,numRead);   //发送
              out1.flush();

              numFinished+=numRead; //已完成字节数
              Thread.sleep(200); //演示文件传输进度用
              publish(numFinished+"/"+fileLen+"bytes");
              setProgress(numFinished*100/(int)fileLen);
        }//end while
        in.close();
        din.close();
        parentUI.txtArea.append("2.传送文件内容成功！\n");
        byte[] fileDigest=in.getMessageDigest().digest(); //生成文件摘要
        parentUI.txtArea.append("生成的摘要：
"+DatatypeConverter.printHexBinary(fileDigest)+"\n\n");
        //用私钥对摘要加密，形成文件的数字签名
        Cipher cipher=Cipher.getInstance("RSA/ECB/PKCS1Padding"); //加密器
        cipher.init(Cipher.ENCRYPT_MODE, privateKey);//用个人私钥初始化加密模式
        byte[] signature=cipher.doFinal(fileDigest);//计算数字签名

        //更新显示
```

```
            parentUI.txtArea.append("生成的数字签名：
"+DatatypeConverter.printHexBinary(signature)+"\n\n");

            //生成 AES 对称密钥
            SecretKey secretKey=Cryptography.generateNewKey();
            parentUI.txtArea.append("生成的密钥：
"+DatatypeConverter.printHexBinary(secretKey.getEncoded())+"\n\n");

            //对数字签名加密
            Cipher cipher2=Cipher.getInstance("AES");
            cipher2.init(Cipher.ENCRYPT_MODE, secretKey);//初始化加密器
            byte[] encryptSign=cipher2.doFinal(signature);//生成加密签名
            parentUI.txtArea.append("用密钥加密后的数字签名：
"+DatatypeConverter.printHexBinary(encryptSign)+"\n\n");
            //对密钥加密
            cipher.init(Cipher.ENCRYPT_MODE, publicKey);//用服务器公钥初始化加密模式
            byte[] encryptKey=cipher.doFinal(secretKey.getEncoded());//加密密钥
            parentUI.txtArea.append("对密钥加密：
"+DatatypeConverter.printHexBinary(encryptKey)+"\n\n");

            //3.发送加密后的数字签名
            out.writeInt(encryptSign.length);
            out.flush();
            out.write(encryptSign);
            out.flush();
            parentUI.txtArea.append("3.发送加密的数字签名成功！\n");

            //4.发送加密密钥
            out.write(encryptKey);//密文长度为 128 字节
            out.flush();
            parentUI.txtArea.append("4.发送加密的密钥成功！\n");

            //5.接收服务器反馈信息
            BufferedReader br=new BufferedReader(
                                    new InputStreamReader(
                                    fileSocket.getInputStream()));
            String response=br.readLine();//读取返回串
            if (response.equalsIgnoreCase("M_DONE")) { //服务器成功接收
                lastResults= "5."+ file.getName() +"    服务器成功接收！\n" ;
            }else if (response.equalsIgnoreCase("M_LOST")){ //服务器接收失败
                lastResults= "5."+ file.getName() +"    服务器接收失败！\n" ;
```

```
            }//end if
            //关闭流
            br.close();
            out.close();
                        out1.close();
            fileSocket.close();
            return null;
        } //doInBackground
        @Override
        protected void process(List<String> middleResults) {
            for (String str:middleResults) {
                parentUI.progressLabel.setText(str);
            }
        }
        @Override
        protected void done() {
            parentUI.progressBar.setValue(parentUI.progressBar.getMaximum());
            parentUI.txtArea.append(lastResults+"\n");
            parentUI.filePanel.setVisible(false);
        }
}
```

7.4 问题讨论

① 你是否能回忆程序的结构？

② 本例中的密码是否会被拦截？

③ 如何改进？