**西安邮电大学**

**网络空间安全学院**

**实验报告**

**（2019/ 2020学年第1学期）**

实验名称： 信息安全专业课程设计A

学生姓名： 杨政和 c

专 业： 安全1802班

学 号： 03183045（9）

指导教师： 于海燕

**西安邮电大学网络空间安全学院XXXX（实验课程名）**

**考核表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学生姓名** |  | **班级/学号** | | **/** | |
| **承担任务实验室（单位）** | **信息安全与信息对抗实验教学中心** | **所在部门** | | **网安院安全1802** | |
| **实施时间** | **年 月 日 — 年 月 日** | | | | |
| **实验内容** | **字典破解密码** | | | | |
| **指导教师（师傅）姓名** |  | **职务或职称** | |  | |
| **指导教师（师傅）**  **对学生的评价** | **评价点** | **评价内容** | | | **评价得分** |
| **平时成绩**  **（20分）** | **课堂出勤、课堂纪律及学习态度**  **遵守实验室的规章制度等情况** | | |  |
| **实验能力**  **（30分）** | **文件检索或软硬件设计能力**  **分析、解决实际问题能力** | | |  |
| **实验验收**  **（20分）** | **实验最终完成质量情况**  **实验演示与答辩情况** | | |  |
| **实验报告**  **（30分）** | **语言及文字表达能力**  **实验报告结构及书写格式规范** | | |  |
| **总评成绩** |  | | **五级制** |  | |
| **指导教师（师傅）评语** | **指导教师（师傅）签字**  **年 月 日** | | | | |

1. **实验目的和要求**

**（实验报告正文可根据具体实验适当调整标题与格式。要求每次实验结束后需提交一次实验报告，全部实验结束后各次实验报告统一装订，附考核表一份，手写还是打印由课程教师确定）**

**正文要求小四号宋体，1.5倍行距。**

页面设置：上－2.5厘米，下－2厘米，左－2.5厘米，右－2厘米，装订线－0厘米，页眉－2厘米，页角－2厘米。

一级标题（章）：用1、2、3、4、5、6……的序号，小三号的加黑的黑体字，居中。一级标题（章）之间另起一页。

二级标题（节）：用1.1、1.2、1,3、1.4……的序号，四号的加黑的黑体字，居中。和一级标题之间空一行。

三级标题：用1.1.1、1.1.2、1.1.3、1.1.4……的序号，小四号的加黑的黑体字，靠左。和二级标题之间空一行。

四级标题：用1.1.1.1、1.1.1.2、1.1.1.3、1.1.1.4……的序号，小四号的宋体字，靠左。

作为章节和段落的序号划分至多到四级。

页码：用阿拉伯数字连续编页，用小五号Times New Roman字号，居中，数字两侧不加任何修饰。

**插图：**插图包括图解、示意图、构造图、曲线图、框图、流程图、布置图、地图、照片、图版等。插图注明项有图号、图题、图例。图号编码用章序号。如“图2.1”表示第2章第1图。图号与图题文字留一字空格置于无色文本框内，放在图的正下方居中，最好和图组合在一起，图题用5号或小5号字，字体用宋体。图中标注符号文字字号不大于图题的字号。

为确保打印足够清晰的图，还需注意以下几点：若是扫描图，则要保证有大于300dpi的分辨率；若是编程软件绘制的图，则应先画出足够大的图，再通过拷贝屏幕的方式，其它根据具体情况而定。

**表：**表的一般格式是数据依序竖排，内容和项目由左至右横读，通版排版。表号也用章序号编号码，如：表2.1是第2章中的第1表。表应有表题，与表号之间空1字，置于无色文本框内，放在表的上方居中，用5号或小5号宋体。表中的内容和项目字号不大于图题的字号。

**公式：**公式包括数学、物理和信息学等。正文中引用的公式、算式或方程式等可以按章序号用阿拉伯数字编号（式号），如：式（2.1）表示第2章第1式，公式一般单行居中排版与上下文分开，式号与公式同行，居右排版。

1. **实验原理**

**2.1 通信：这里涉及了服务端与客户端通信，同时也有客户端与客户端的通信；因此采用了socket的tcp模式与udp模式**

**2.2 破解**

**1.客户端（Client）：接收用户输入，包括IP,端口与账号信息，并将这些信息发送给破解工具，然后自己进入等待状态**

**2.破解工具（Tool）：在收到客户端请求后，根据字典库依次尝试密码去向服务登录；若登录成功，则表示破解成功，然后将密码发送给正在等待的客户端；若失败（账号错误或字典尝试完）就返回客户端一个空密码，客户端退出**

**3.服务器（Server）：运行在8080端口，可以读入写着用户信息（用户名，密码）的配置文件，对请求进行验证，并返回相应状态码；同时具有记录所有请求信息的功能**

1. **实验内容及步骤**

**3.1 服务器（Server）**

**ServerMain：服务器启动入口**

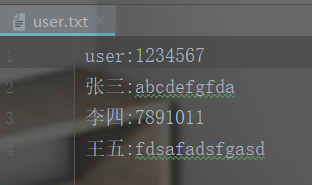
1. **import** java.io.IOException;
3. **public** **class** ServerMain {
5. **public** **static** **void** main(String[] args) {
6. **try** {
7. **new** Server().start();
8. } **catch** (IOException e) {
9. e.printStackTrace();
10. }
11. }
12. }

**Server：负责为所有客户端启动都启动一个线程去处理**

1. **import** java.io.IOException;
2. **import** java.net.ServerSocket;
3. **import** java.net.Socket;
4. **import** java.util.logging.Logger;
6. **public** **class** Server {
7. **private** **static** String name = Server.**class**.getName();
8. **private** **static** Logger log = Logger.getLogger(name);
10. **public** **void** start() **throws** IOException {
11. ServerSocket server = **new** ServerSocket(8080);
12. log.info("服务器在8080端口启动。。。");
14. **while** (**true**) {
15. Socket conn = server.accept();
16. String reqAddr = conn.getInetAddress().getHostAddress();
17. log.info(reqAddr + "尝试连接...");
18. **new** ServerThead(conn).start();
19. }
20. }
21. }

**ServerThread：具体的服务端线程**

1. **读取客户端请求的账号（username），与密码（pwd）**
2. **将账号密码与user.txt中记录的用户信息进行匹配**
3. **将匹配结果以状态码的形式发送给客户端**



1. **import** java.io.\*;
2. **import** java.net.Socket;
3. **import** java.util.logging.Logger;
5. **public** **class** ServerThead **extends** Thread {
7. **private** Socket conn;
8. **private** Integer code;
9. **private** Integer clientCount = 1;
11. **private** **static** **final** Integer ERROR\_USER = -1;
12. **private** **static** **final** Integer ERROR\_PWD = 0;
13. **private** **static** **final** Integer SUCCESS = 1;
15. **private** **static** String name = Server.**class**.getName();
16. **private** **static** Logger log = Logger.getLogger(name);
18. **public** ServerThead(Socket conn) {
19. **this**.conn = conn;
20. }
22. @Override
23. **public** **void** run() {
24. **try** {
25. **while** (**this**.code != ERROR\_USER || **this**.code != SUCCESS) {
26. read();
27. write();
28. }
29. conn.close();
30. } **catch** (IOException e) {
31. }
32. }
34. **public** **void** read() **throws** IOException {
35. InputStream inputStream = conn.getInputStream();
36. BufferedReader bufferedReader = **new** BufferedReader(**new** InputStreamReader(inputStream));
37. String msg = bufferedReader.readLine();
39. check(msg);
40. }
42. **private** **void** check(String msg) **throws** IOException {
43. File file = **new** File("server02/user.txt");
44. Reader fileReader = **new** FileReader(file);
45. BufferedReader bufferedReader = **new** BufferedReader(fileReader);
47. String info = bufferedReader.readLine();
48. String[] InfoSplit = info.split(":");
49. String[] msgSplit = msg.split(":");
51. String reqAddr = conn.getInetAddress().getHostAddress();
52. **if** (! InfoSplit[0].equals(msgSplit[0])) {
53. **this**.code = ERROR\_USER;
54. log.info(reqAddr + " 账号错误，拒绝连接！");
55. } **else** **if** (! InfoSplit[1].equals(msgSplit[1])) {
56. **this**.code = ERROR\_PWD;
57. log.info(reqAddr + " 密码错误。。。");
58. } **else** {
59. **this**.code = SUCCESS;
60. log.info("客户端" + clientCount + "（reqAddr" + " ）连接成功！！！");
61. clientCount++;
62. }
63. }
65. **public** **void** write() **throws** IOException {
66. OutputStream outputStream = conn.getOutputStream();
67. PrintWriter printWriter = **new** PrintWriter(outputStream);
68. printWriter.println(**this**.code);
69. printWriter.flush();
70. }
71. }

**3.2 破解工具（Tool）**

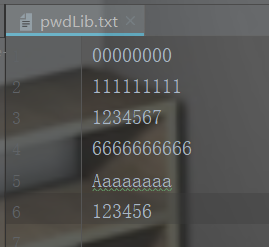
**ToolMain：启动该破解工具**

1. **package** tool;
3. **import** java.io.IOException;
5. **public** **class** ToolMain {
7. **public** **static** **void** main(String[] args) {
8. **try** {
9. **new** Tool().start();
10. } **catch** (IOException e) {
11. e.printStackTrace();
12. } **catch** (InterruptedException e) {
13. e.printStackTrace();
14. }
15. }
16. }

**ToolUtils：工具类，提供以下辅助方法**

1. **读取客户端信息（ip，port，username），udp模式**
2. **获取密码字典（pwdLib.txt）**
3. **将密码字典中的密码与username组装成访问服务器的token**
4. **将破解出的密码发送回客户端，udp模式**
5. **package** tool;
7. **import** java.io.\*;
8. **import** java.net.DatagramPacket;
9. **import** java.net.DatagramSocket;
10. **import** java.net.InetAddress;
11. **import** java.net.Socket;
12. **import** java.util.ArrayList;
13. **import** java.util.HashMap;
14. **import** java.util.List;
15. **import** java.util.Map;
17. **public** **class** ToolUtils {
19. **public** **static** List<String> generateInfo(String username) **throws** IOException {
20. List<String> pwdList = getPwdList();
21. List<String> infoList = **new** ArrayList<>();
23. **for** (**int** i = 0; i < pwdList.size(); i++) {
24. String info = username + ":"+ pwdList.get(i);
25. infoList.add(info);
26. }
27. **return** infoList;
28. }
30. **private** **static** List<String> getPwdList() **throws** IOException {
32. File file = **new** File("client02/src/tool/pwdLib.txt");
33. Reader reader = **new** FileReader(file);
34. BufferedReader bufferedReader = **new** BufferedReader(reader);
36. ArrayList<String> pwdList = **new** ArrayList<>();
37. String pwd;
38. **while** ((pwd = bufferedReader.readLine()) != **null**) {
39. pwdList.add(pwd);
40. }
42. bufferedReader.close();
43. reader.close();
45. **return** pwdList;
46. }
48. **public** **static** Map<String,Object> recvServerInfo() **throws** IOException {
49. DatagramSocket socket = **new** DatagramSocket(1234);
50. DatagramPacket packet = **new** DatagramPacket(**new** **byte**[256], 256);
51. System.out.println("等待客户端请求...");
52. socket.receive(packet);
54. String serverInfo = **new** String(packet.getData());
55. String[] split = serverInfo.split(":");
56. HashMap<String, Object> infoMap = **new** HashMap<>();
57. infoMap.put("ip",split[0]);
58. infoMap.put("port",Integer.valueOf(split[1]));
59. infoMap.put("username",split[2].trim());
61. socket.close();
62. **return** infoMap;
63. }
65. **public** **static** **void** sendPwd(String pwd) **throws** IOException {
66. DatagramSocket socket = **new** DatagramSocket(1234);
67. DatagramPacket packet = **new** DatagramPacket(pwd.getBytes(), pwd.length(), InetAddress.getByName("localhost"), 6666);
68. socket.send(packet);
69. System.out.println("密码已向客户端发送！");
70. socket.close();
71. }
72. }

**Tool：执行破解，依次将组装好的token发向服务端；若成功或账号错误则直接停止破解，返回错误；否则继续破解，直到找到或字典库用完；tcp模式**



1. **package** tool;
3. **import** java.io.\*;
4. **import** java.net.Socket;
5. **import** java.util.\*;
7. **public** **class** Tool {
9. **private** Socket client;
11. **private** **static** **final** Integer ERROR\_USER = -1;
12. **private** **static** **final** Integer ERROR\_PWD = 0;
13. **private** **static** **final** Integer SUCCESS = 1;

16. **public** **void** start() **throws** IOException, InterruptedException {
17. Map<String, Object> serverInfoMap = ToolUtils.recvServerInfo();
18. client = **new** Socket((String)serverInfoMap.get("ip"), (Integer)serverInfoMap.get("port"));
19. blast(ToolUtils.generateInfo((String) serverInfoMap.get("username")));
20. }
22. **public** **boolean** blast(List<String> infoList) **throws** IOException, InterruptedException {
23. OutputStream outputStream = client.getOutputStream();
24. PrintWriter printWriter = **new** PrintWriter(outputStream);
25. InputStream inputStream = client.getInputStream();
26. BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(inputStream));
28. **for** (String info : infoList) {
29. printWriter.println(info);
30. printWriter.flush();
31. Integer code = Integer.valueOf(reader.readLine());
32. **if** (code == ERROR\_USER) {
33. System.out.println("账号错误");
34. ToolUtils.sendPwd("");
35. **return** **false**;
36. } **else** **if** (code == ERROR\_PWD) {
37. System.out.println("破解中..." + info.split(":")[1] + " 错误");
38. } **else** **if** (code == SUCCESS) {
39. System.out.println("破解成功！密码：" + info.split(":")[1]);
40. ToolUtils.sendPwd(info.split(":")[1]);
41. **return** **true**;
42. } **else** {
43. System.out.println("未知异常");
44. **return** **false**;
45. }
46. }
48. ToolUtils.sendPwd("");
49. System.out.println("字典中无匹配密码。。。");
50. **return** **false**;
51. }
52. }

**3.3 客户端（Client）、**

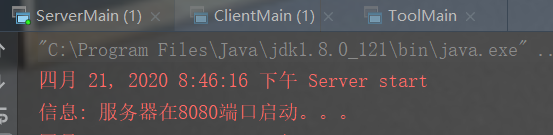
**ClientMain：启动客户端**

1. **package** client;
3. **import** java.io.IOException;
5. **public** **class** ClientMain {
7. **public** **static** **void** main(String[] args) {
8. **try** {
9. **new** Client().start();
10. } **catch** (IOException e) {
11. e.printStackTrace();
12. }
13. }
14. }

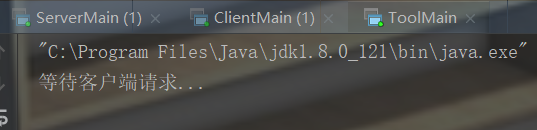
**Client**

1. **读取用户输入（IP，port，username）**
2. **将请求信息发送给Tool进行破解，udp模式**
3. **接收破解后的密码，udp模式**
4. **若破解成功则用该密码尝试登陆，失败则结束程序，tcp模式**
5. **package** client;
7. **import** java.io.\*;
8. **import** java.net.\*;
9. **import** java.util.Scanner;
11. **public** **class** Client {
13. **public** **void** start() **throws** IOException {
14. String serverInfo = input();
15. useTool(serverInfo);
16. String pwd = serverInfo.split(":")[2] + ":" + recvPwd();
17. login(pwd);
18. }
20. **public** String input() {
21. String serverInfo = "";
22. System.out.print("服务器IP：");
23. Scanner in = **new** Scanner(System.in);
24. serverInfo = serverInfo + in.nextLine() + ":";
25. System.out.print("端口：");
26. serverInfo = serverInfo + in.nextLine() + ":";
27. System.out.print("账号：");
28. serverInfo = serverInfo + in.next();
29. **return** serverInfo;
30. }
32. **public** **void** useTool(String serverInfo) **throws** IOException {
33. DatagramSocket socket = **new** DatagramSocket(6666);
34. DatagramPacket packet = **new** DatagramPacket(serverInfo.getBytes(), serverInfo.length(),
35. InetAddress.getByName("localhost"), 1234);
36. socket.send(packet);
37. System.out.println("等待破解密码中...");
39. socket.close();
40. }
42. **public** String recvPwd () **throws** IOException {
43. DatagramSocket socket = **new** DatagramSocket(6666);
44. DatagramPacket packet = **new** DatagramPacket(**new** **byte**[32], 32);
45. socket.receive(packet);
46. **if** (**new** String(packet.getData()).trim().isEmpty()) {
47. System.out.println("破解失败。。。");
48. System.exit(0);
49. }
50. String pwd = **new** String(packet.getData());
51. System.out.println("获取到破解密码：" + pwd);
52. **return** pwd;
53. }
55. **public** **void** login(String pwd) **throws** IOException {
56. Socket client = **new** Socket("127.0.0.1", 8080);
57. System.out.println("尝试连接...");
58. OutputStream outputStream = client.getOutputStream();
59. PrintWriter printWriter = **new** PrintWriter(outputStream);
60. printWriter.println(pwd.trim());
61. printWriter.flush();
63. InputStream inputStream = client.getInputStream();
64. BufferedReader bufferedReader = **new** BufferedReader(**new** InputStreamReader(inputStream));
65. **if** (Integer.valueOf(bufferedReader.readLine()) == 1 ) {
66. System.out.println("连接成功！");
67. } **else** {
68. System.out.println("连接失败！");
69. }
71. printWriter.close();
72. outputStream.close();
73. bufferedReader.close();
74. inputStream.close();
75. client.close();
76. }
77. }
78. **实验结果分析**

**启动服务器**

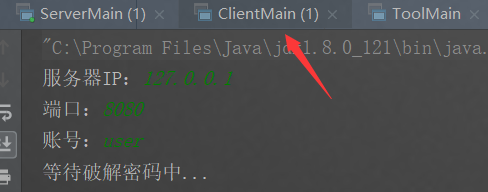


**启动破解工具**

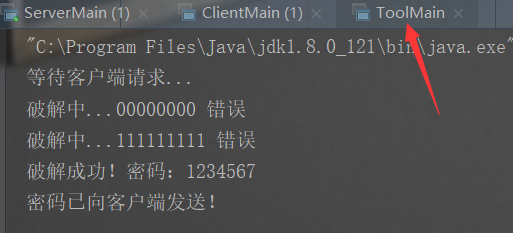
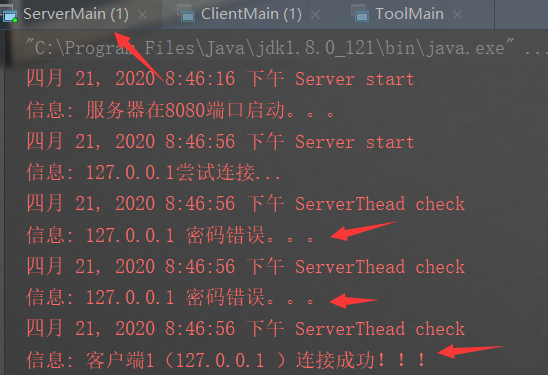


**4.1 情况一：可以破解成功，账户是user（存在这个账户，且密码在字典库中）**

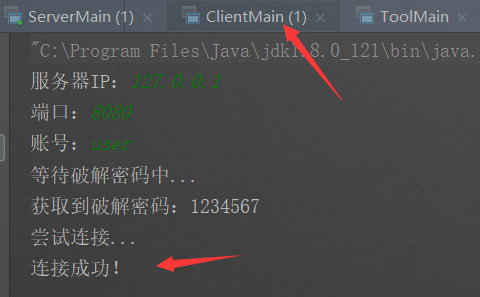
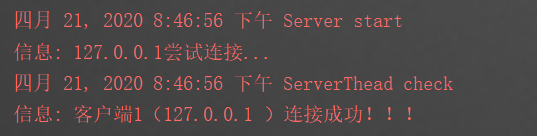
**客户端输入服务器信息后进入等待，将请求交给破解工具**



**破解工具接收到请求并开始破解，在经历两个失败后成功；同时服务器也记录了破解过程**

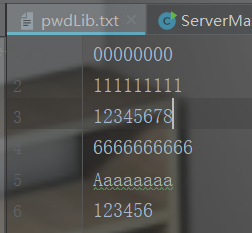
 

**最后客户端成功登陆，从服务端也能看到直接登陆成功**

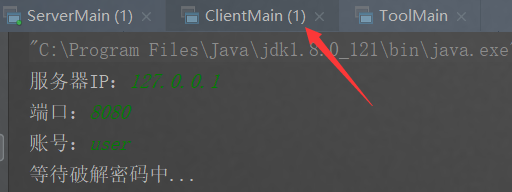
 

**4.2 情况二：登陆失败（字典用尽）**

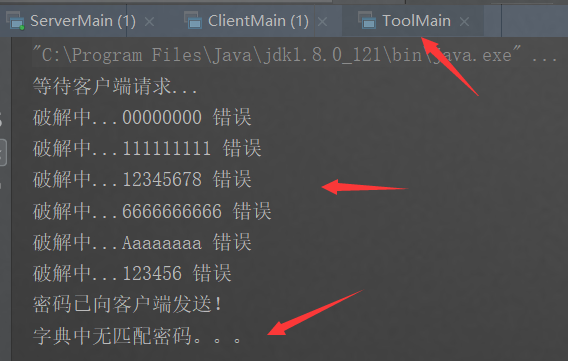
**首先将字典中的1234567改成12345678，保证字典中无user的密码**

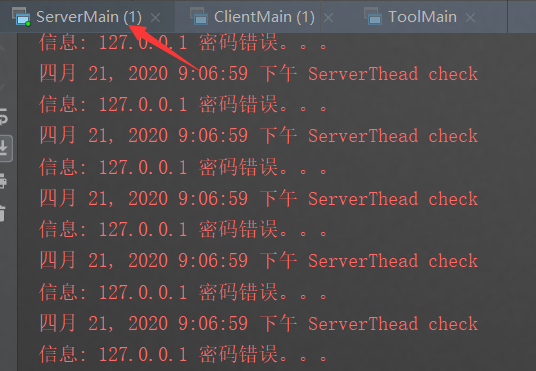


**客户端**

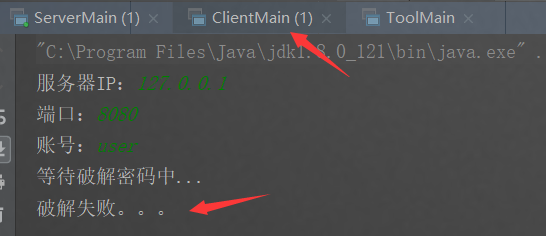


**破解工具 与 服务端信息**

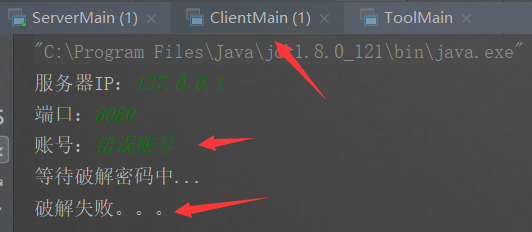


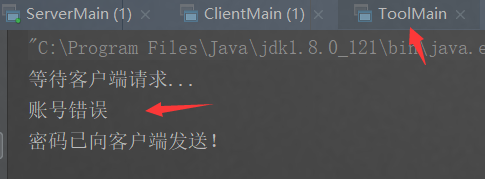


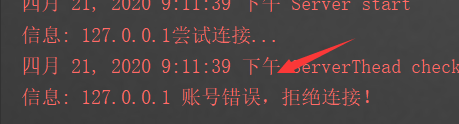
**客户端收到返回，结束程序**



**4.3 情况三：账号错误，输入一个服务器user.txt没有的账号**







**五．实验总结及心得体会**

**1.通过这次实验我更加熟悉了socket编程，体会了其中TCP模式与UDP模式的应用，提高了动手能力**

**2.我了解并学习了字典破解，并激发日后对于安全方面学习的热情**