**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： XXXXXX c

专 业： 安全XXXX班

学 号： 8位学号（班内序号）

指导教师： XXX

**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XXX** | |
| **实施时间** | **年 月 日 — 年 月 日** | | | | |
| **实验内容** |  | | | | |
| **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. **通信：通过socket进行远程远程通信，其中使用tcp协议，一个服务端服务多台客户端**

**其中通过使用多线程技术，使服务端可以同时服务多个客户端**

1. **防火墙：每个客户端都有IP与账号（account）作为其标识，因此可以建立黑名单，将所有禁止的IP与账户都记录在配置文件中，在提供服务前先对客户端进行验证**
2. **实验内容及步骤**

**服务端代码**

1. **配置文件**
2. #拒绝的IP，多个IP以逗号隔开
3. reject.ip=39.105.136.112 #127.0.0.1
4. #允许的账号，多个账号以逗号隔开
5. allow.account=123456,abcdefg,78910
6. #拒绝的账号，多个账号以逗号隔开
7. reject.account=654321,gfedcba,109876

**2.入口类ServerMain，启动服务端**

1. **import** java.io.IOException;
3. **public** **class** ServerMain {
5. **public** **static** **void** main(String[] args) {
6. Server server = **new** Server();
7. **try** {
8. server.start();
9. } **catch** (IOException e) {
10. e.printStackTrace();
11. }
12. }
13. }

**3.Server类，服务端核心代码：**

**3.1建立多个线程为所有请求服务，并记录所有尝试连接的客户端个数**

**3.2对客户进行IP 与 账号验证，并返回相应的Code**

**3.3 这里服务端会一直运行**

1. **import** java.io.\*;
2. **import** java.net.ServerSocket;
3. **import** java.net.Socket;
4. **import** java.util.concurrent.TimeUnit;
5. **import** java.util.logging.Logger;
7. **public** **class** Server {
9. **private** **static** Integer clientCount = 1;
11. **private** **static** String name = Server.**class**.getName();
12. **private** **static** Logger log = Logger.getLogger(name);
14. **private** **static** **final** Integer REJECT\_IP = -2;
15. **private** **static** **final** Integer REJECT\_ACCOUNT = -1;
16. **private** **static** **final** Integer ERROR\_ACCOUNT = 0;
17. **private** **static** **final** Integer SUCCESS = 1;
19. **public** **void** start() **throws** IOException {
20. ServerSocket server = **new** ServerSocket(8080);
21. log.info("服务器在8080端口启动。。。");
23. **while** (**true**) {
24. Socket conn = server.accept();
25. String requestAddress = conn.getInetAddress().getHostAddress();
26. log.info("客户端-" + clientCount + "(" + requestAddress + " )尝试建立连接ing");
27. **new** Thread(() -> {
28. OutputStream outputStream = **null**;
29. InputStream inputStream = **null**;
30. PrintWriter printWriter = **null**;
31. BufferedReader bufferedReader = **null**;
32. **try** {
33. outputStream = conn.getOutputStream();
34. printWriter = **new** PrintWriter(outputStream);
35. inputStream= conn.getInputStream();
36. bufferedReader = **new** BufferedReader(**new** InputStreamReader(inputStream));
38. **if** (PropertiesUtil.getRejectIps().contains(requestAddress)) {
39. printWriter.println(REJECT\_IP);
40. printWriter.flush();
41. log.info("客户端-" + requestAddress + "属于黑名单IP，已拒绝");
42. **return**;
43. }
45. String account = bufferedReader.readLine();
46. **if** (PropertiesUtil.getRejectAccounts().contains(account)) {
47. printWriter.println(REJECT\_ACCOUNT);
48. printWriter.flush();
49. log.info("客户端-" + clientCount + "(" + requestAddress + " )属于黑名单用户，已拒绝");
50. } **else** **if** ( ! PropertiesUtil.getAllowAccounts().contains(account)) {
51. printWriter.println(ERROR\_ACCOUNT);
52. printWriter.flush();
53. log.info("客户端-" + clientCount + "(" + requestAddress + " )账号错误，本次连接中断");
54. } **else** {
55. printWriter.println(SUCCESS);
56. printWriter.flush();
57. log.info("客户端-" + clientCount + "(" + requestAddress + " )连接成功，准备服务！");
59. TimeUnit.SECONDS.sleep(2);
60. System.out.println("客户端-" + clientCount + "：" + bufferedReader.readLine());
61. printWriter.println("好的好的~~");
62. printWriter.flush();
63. System.out.println("服务端：" + "好的好的~~");
64. }
65. **return**;
66. } **catch** (IOException e) {
67. e.printStackTrace();
68. } **catch** (InterruptedException e) {
69. e.printStackTrace();
70. } **finally** {
71. clientCount++;
72. **try** {
73. bufferedReader.close();
74. inputStream.close();
75. printWriter.close();
76. outputStream.close();
77. } **catch** (IOException e) {
78. e.printStackTrace();
79. }
80. }
81. }).start();
82. }
83. }
84. }

**4.工具类PropertiesUtil，用来读取properties文件，并返回**

**4.1禁止登陆IP集合**

**4.2允许的账户（account）集合**

**4.3禁止的账户集合**

1. **import** java.io.\*;
2. **import** java.util.Arrays;
3. **import** java.util.List;
4. **import** java.util.Properties;
6. **public** **class** PropertiesUtil {
8. **private** **static** Properties properties = **new** Properties();
9. **private** **static** **final** String  FILE\_PATH = "server01/src/root.properties";
11. **static** {
12. **try** {
13. FileInputStream fileInputStream = **new** FileInputStream(FILE\_PATH);
14. InputStream inputStream = **new** BufferedInputStream(fileInputStream);
15. properties.load(inputStream);
16. } **catch** (IOException e) {
17. e.printStackTrace();
18. }
19. }
21. **public** **static** List<String> getRejectIps() {
22. String ips = (String)properties.get("reject.ip");
23. **return** Arrays.asList(ips.split(","));
24. }
26. **public** **static** List<String> getAllowAccounts() {
27. String accounts = (String)properties.get("allow.account");
28. **return** Arrays.asList(accounts.split(","));
29. }
31. **public** **static** List<String> getRejectAccounts() {
32. String ips = (String)properties.get("reject.account");
33. **return** Arrays.asList(ips.split(","));
34. }
35. }

**客户端（Client）**

**1.客户端启动入口ClientMain**

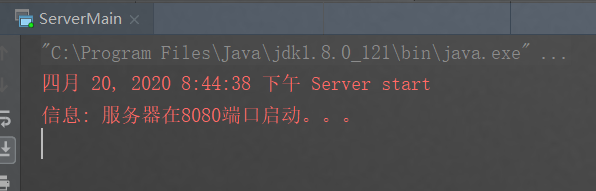
1. **1. public** **class** ClientMain {
3. **public** **static** **void** main(String[] args) {
4. **new** Client().start();
5. }
6. }

**2客户端核心类Client**

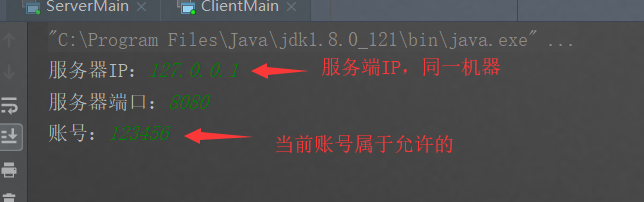
* 1. **读取控制台的输入，包括IP，端口（port），账号（account）**
  2. **尝试与服务端建立连接**
  3. **获取响应Code，并进行判断，然后返回相应的消息**
  4. **若连接成功，则向服务端发送一条请求（模拟请求服务），然后断开连接**

1. **import** java.io.\*;
2. **import** java.net.Socket;
3. **import** java.util.HashMap;
4. **import** java.util.Map;
5. **import** java.util.Scanner;
6. **import** java.util.concurrent.TimeUnit;
7. **import** java.util.logging.Logger;
9. **public** **class** Client {
10. **private** **static** String name = Client.**class**.getName();
11. **private** **static** Logger log = Logger.getLogger(name);
13. **private** **static** **final** Integer REJECT\_IP = -2;
14. **private** **static** **final** Integer REJECT\_ACCOUNT = -1;
15. **private** **static** **final** Integer ERROR\_ACCOUNT = 0;
16. **private** **static** **final** Integer SUCCESS = 1;
18. **public** **void** start() {
19. Map<String, Object> input = input();
20. Socket client = **null**;
21. **try** {
22. client = **new** Socket((String)input.get("ip"),(Integer)input.get("port"));
23. } **catch** (IOException e) {
24. log.info("连接失败！IP或端口可能错误");
25. **return**;
26. }
27. log.info("客户端 " + client.getInetAddress().getHostAddress() + " 启动");
29. OutputStream outputStream = **null**;
30. PrintWriter printWriter = **null**;
31. **try** {
32. outputStream = client.getOutputStream();
33. printWriter = **new** PrintWriter(outputStream);
34. printWriter.println((String)input.get("account"));
35. printWriter.flush();
36. log.info("请求连接中...");
38. getConnectCode(client);
39. } **catch** (IOException e) {
40. e.printStackTrace();
41. }
43. **try** {
44. printWriter.close();
45. outputStream.close();
46. client.close();
47. } **catch** (IOException e) {
48. e.printStackTrace();
49. }
50. }
52. **public** Map<String,Object> input() {
53. HashMap<String, Object> map = **new** HashMap<>();
55. Scanner in = **new** Scanner(System.in);
56. System.out.print("服务器IP：");
57. map.put("ip",in.next());
58. System.out.print("服务器端口：");
59. map.put("port",in.nextInt());
60. System.out.print("账号：");
61. map.put("account",in.next());
63. **return** map;
64. }
66. **public** **void** getConnectCode(Socket client) **throws** IOException {
67. InputStream inputStream = client.getInputStream();
68. BufferedReader bufferedReader = **new** BufferedReader(**new** InputStreamReader(inputStream));
69. **int** code = Integer.valueOf(bufferedReader.readLine());
70. **if** (code == REJECT\_IP) {
71. log.info("黑名单IP，拒绝连接！");
72. } **else** **if** (code == REJECT\_ACCOUNT) {
73. log.info("黑名单用户，拒绝服务！");
74. } **else** **if** (code == ERROR\_ACCOUNT) {
75. log.info("账号错误，想清楚再来吧。。。");
76. } **else** **if** (code == SUCCESS){
77. log.info("连接成功，准备干活！");
78. **try** {
79. TimeUnit.SECONDS.sleep(1);
80. } **catch** (InterruptedException e) {
81. e.printStackTrace();
82. }
83. work(client);
84. } **else** {
85. log.info("未知问题。。。");
86. }
87. }
89. **public** **void** work(Socket client) **throws** IOException {
90. PrintWriter printWriter = **new** PrintWriter(client.getOutputStream());
91. printWriter.println("快给我服务！！！");
92. printWriter.flush();
93. System.out.println("客户端：快给我服务！！！");
95. BufferedReader bufferedReader = **new** BufferedReader(**new** InputStreamReader(client.getInputStream()));
96. String response = bufferedReader.readLine();
97. System.out.println("服务端：" + response);
99. bufferedReader.close();
100. printWriter.close();
101. }
102. }
103. **实验结果分析**

1.服务端启动，在8080端口

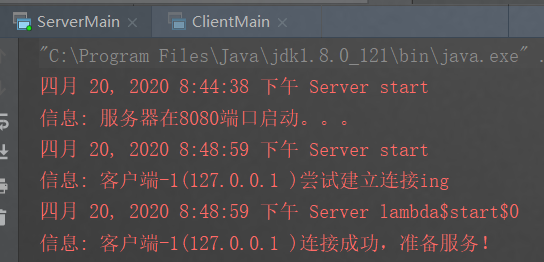


**2.启动一个客户端 ：可以正常访问，IP与账号（account）都没在黑名单中**

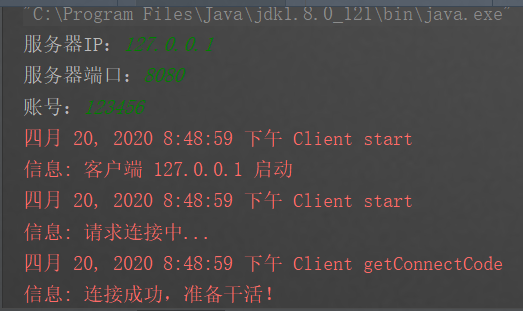


**3.观察服务端与客户端状态**

**服务端：有一台客户端请求连接，1号（127.0.0.1），成功通过验证，连接成功**

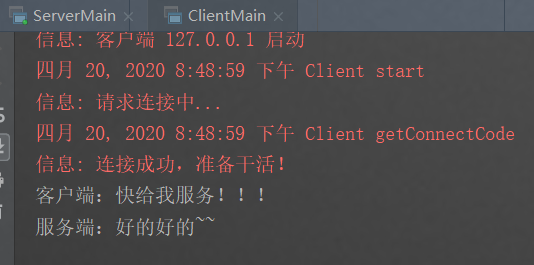


**客户端：连接成功，并提示**

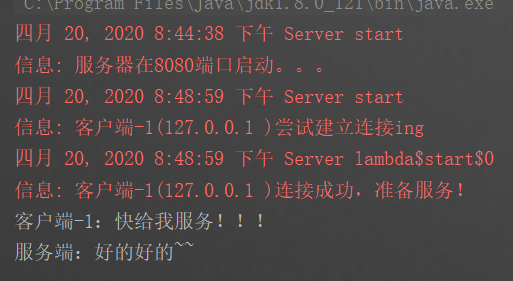


**4.连接成功后进行服务**

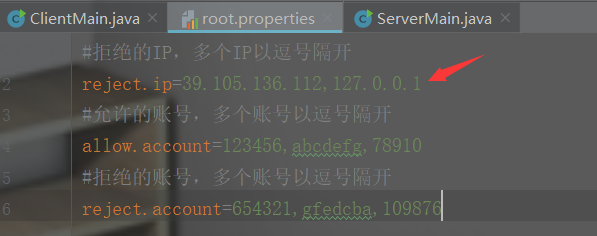
**客户端：发送消息，服务端响应**

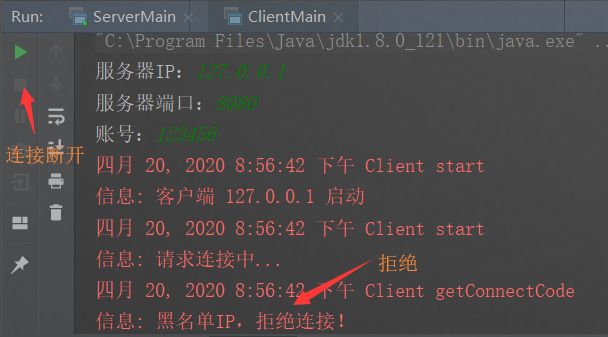


**服务端**

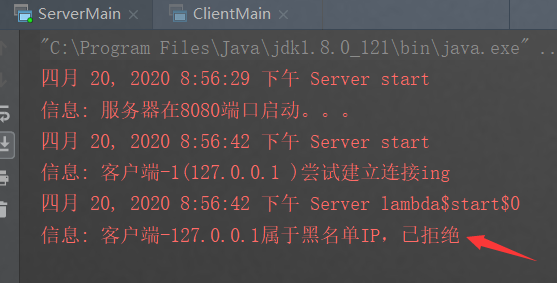


**5.修改黑名单，将127.0.0.1加入禁止的IP，看响应**



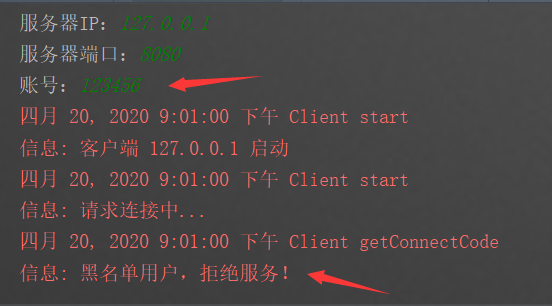


**再看服务端**



**7.修改黑名单，加入账号123456**

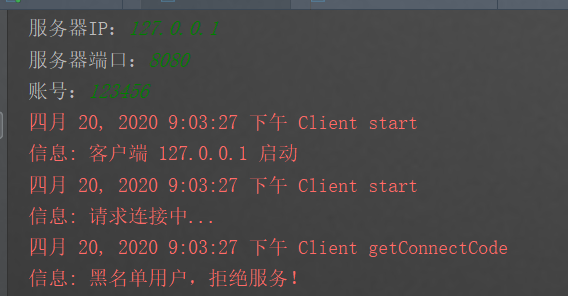






**8.多个客户端同时请求**

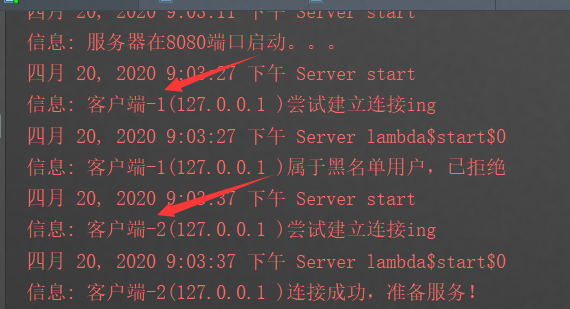
**客户端1**



**客户端2**



**服务端**



1. **实验总结及心得体会**