Http 服务程序设计文档

分工说明

黄珉(3120000156)：

服务器 Socket 编程

胡亮泽(3120102116)：

数据加密和签名

编程环境

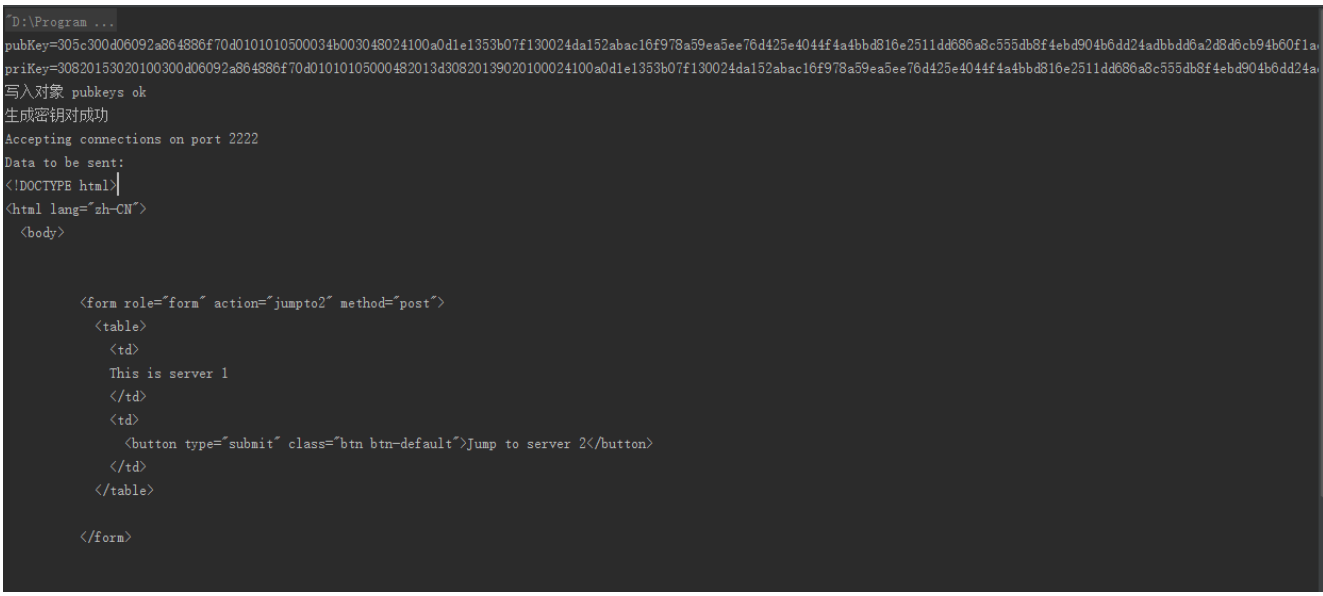
电脑配置： Lenovo ideapad, Vaio E 系列

操作系统： Windows 7

开发工具： Jetbrain IDEA

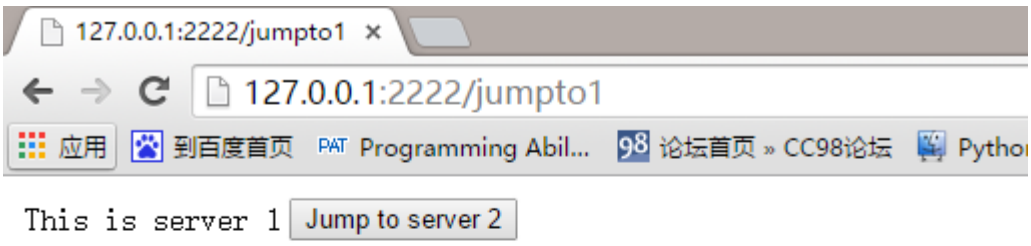
运行效果

1. 我们的程序一共由两个 HTTP 服务器组成，两个服务器都能够相应外部和本地的数据请求，完成加密和签名，解密和检查签名正确性，并向另外一个服务器发出请求。运行服务器，可以看到输出如下信息，分别是生成的本地公玥和私钥的编码，表明密钥成功生成，接下来是一行信息，表明我们的一号服务器开始在 2222 号端口监听请求的信息，最后是我们要发送的 html 数据信息。



2. 两个服务器的源码基本相同，除了 1 号服务器监听 2222 号端口，2 号服务器监听 81 号端口，html 页面的内容有些许差别，请求信息不同。因此，整个过程的描述信息围绕其中一个服务器（服务器 1）进行，另外一个不再赘述。

3. 运行两个服务器后，打开浏览器，输入 127.0.0.1:2222，即访问本机的 2222 号端口，前面已经提到过，这是 1 号服务器监听的端口，因此，我们可以获得来自浏览器的 GET 请求，返回如下信息：



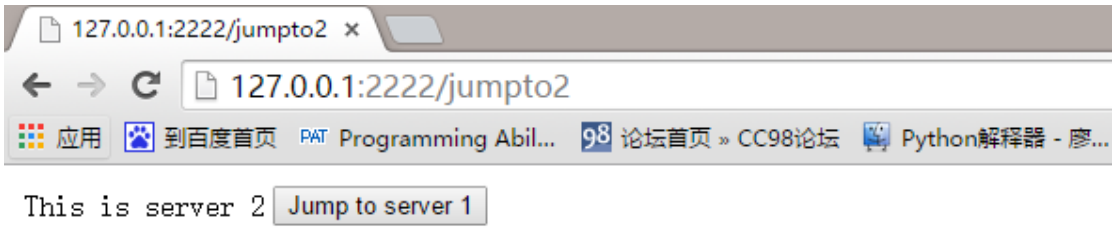
从图中可以看出，浏览器显示的页面是 1 号服务器提供的页面数据。

4. 接下来我们观察整个过程中服务器程序输出的提示信息，这些信息主要包括浏览器发送的请求信息。从图中我们可以看到获得了浏览器发送的 GET 请求，因此我们返回一个 HTML 页面的数据，也就是我们刚才看到的 HTML 页面。

```
        <form role="form" action="jumpto2" method="post">
          <table>
            <td>
            This is server 1
            </td>
            <td>
              <button type="submit" class="btn btn-default">Jump to server 2</button>
            </td>
          </table>

        </form>




  </body>
</html>
Request GET / HTTP/1.1
Get process the message
Request GET /favicon.ico HTTP/1.1
Request GET /favicon.ico HTTP/1.1
```

5. 接下来，我们点击页面中提供的”Jump to server 1”，按钮来跳转到 2 号服务器的主

页。该按钮可以向 1 号服务器发出一个获得 2 号服务器数据的 POST 请求，从而由 1 号服务器处理请求并连接到二号服务器，获得数据。



6. 点击后，我们观察 2 号服务器的程序输出的提示信息：首先是一条来自 1 号服务器的提示信息，观察到”Another”, 这表明该请求并不是来自本地的。 接下来，2 号服务器需要响应 1 号服务器的请求。首先需要通过私钥生成签名信息，以此表明自己的身份，这里我们约定签名内容为“Right Server“,双方都知道这个约定。
   在命令窗口中，输出了签名后的内容以及签名前的原值。接下来，二号服务器会向一号服务器发送自己的公玥。最后发送 DES 加密后的 HTML 数据。

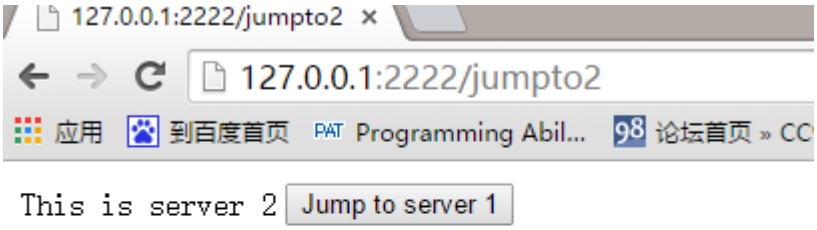
```
Another POST /server2.html HTTP/1.1
[B@351fe21b
signed(签名内容)原值=94769c8b705a7a3f3ecfd1e1507381c1da7928584e4352b086d81f128c50aad33eb551d450f3e2a3beab85b39b75f7fee03a0d853583f2e72bbba0755ff62733
info（原值）=Right Server
签名并生成文件成功
signed :94769c8b705a7a3f3ecfd1e1507381c1da7928584e4352b086d81f128c50aad33eb551d450f3e2a3beab85b39b75f7fee03a0d853583f2e72bbba0755ff62733
```

7. 观察 1 号服务器中输出的提示信息：首先我们输出 1 号服务器接收到的公玥的编码。这是 2 号服务器的公玥内容，接下来服务器会通过这个 16 进制的编码重新生成一个公玥对象，这个对象就是 2 号服务器的公玥。
   接下来，我们输出 2 号服务器发送的签名后的数据以及 DES 加密后的 HTML 数据。最后，通过验证签名是否正确，决定输出解密后的 HTML 页面或者错误信息。这里根据图中的输出信息，我们的签名是正确的，因此，我们将输出正确的 HTML 页面。

```
client wants to jump!
From server2 socket:
305c300d06092a864886f70d0101010500034b00304802410 0b7a93ca3737ecab3862bd82bea4153e1d6bffd92b225937ae0ad79a78ec3eb073d60a00af464f4ef181c544a8985d12cca056b6d7dba0ced32ac23bcc406c1
Key get complete
94769c8b705a7a3f3ecfd1e1507381c1da7928584e4352b086d81f128c50aad33eb551d450f3e2a3beab85b39b75f7fee03a0d853583f2e72bbba0755ff62733completesigned value:94769c8b705a7a3f3ecfd1e1507
ff8f8f33ffa202def552163ba94893f7301fd412d2b85d4678c831277b7c8184f86dccf8c2a4a93a63efa56bb2d6a92703eeca58c7bb5162286aafce0199fa69ed085525a83ebac50b8cc091e1dd9f4dcd24f0bc2f26d21e
info=Right Server
签名正常

From server2: null
Request GET /favicon.ico HTTP/1.1
```

8. 以下是输出的 HTML 页面信息，可以看到我们已经成功跳转到了 2 号服务器的页面，

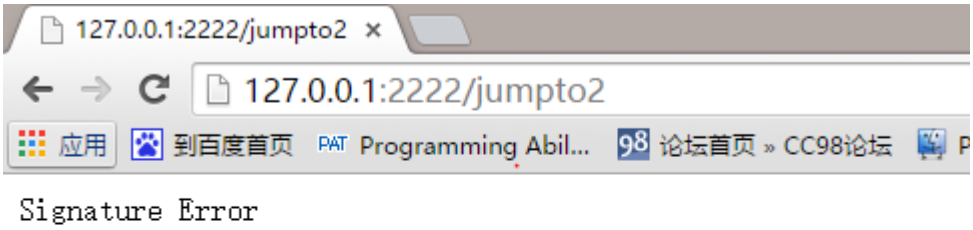
并且页面出现了一个 Jump to Server 1 的按钮，点击按钮可以跳转回 1 号服务器



9. 如图，成功跳转后显示如下页面

10. 需要注意的是，由于这个请求是由本地服务器响应的，因此我们可以发现并没有从 2 号服务器接收数据。

11. 另外，我们修改 1 号服务器代码中的签名内容，就可以发现跳转到错误页面，如图所示：



心得体会

虽然已经有 socket 编程的基础，我组仍遇到不少困难，以下是我们遇到的问题以及解决方案：

1. 在服务器运行时，经常会出现无法正常发送数据或者接受数据。经检查，发现 java 非阻塞发送数据时，需要手动检查接受的内容，从而使一次接受的数据内容只包括我们自己需要的，或者在服务端再次发送数据前，需要先接受一个从客户端发送的确认接受完毕的信号
2. 由于在加密过程中使用了 java 的库函数，而 socket 发送的数据确是 byte[]类型的，因此我们需要使用函数先将公钥和签名信息转换成相应的 byte[], 我们将其转换为 16 进制的字符串，在成功发送数据后再转换回来，生成公钥。否则数据会发送失败。
3. 服务器在接收请求信息并返回 html 数据时，需要判断本次的请求是来自本地还是来自其他的服务器，否则会出现错误，因为本地的数据并不需要加密。解决方案是在协议的 POST 请求的头中表明这是一个来自其他服务器的请求信息，这样就可以正确加密和签名数据。

我组虽然在本次实验中遇到了不少困难，但获益良多，主要收获有以下几点：

1. 熟悉了 JAVA 的 socket 编程，并了解了 java 的 socket 编程和 C++的不同。
2. 对 HTTP 协议有了更加深入和清晰的 认识，通过编程实践理解了 header 等内容在协议中的地位和意义
3. 通过编程实现加密和签名，理解了两者在数据传送中的价值和作用。