Analysis of the Beijing Forestry University Login System

This post was originally written in Chinese and published on CSDN.

This post details the process of reverse engineering the network login system used at Beijing Forestry University.

The goal is to simulate the login process programmatically, effectively mimicking the actions of a user connecting to the network through a web browser.

By using Chrome's developer tools, we can observe the network requests made during the login process.

The first request is to `CheckLogin.jsp`, followed by a request to `index.jsp` which is triggered by `CheckLogin.jsp`.

Let's examine `CheckLogin.jsp` more closely.

We can see that it uses a POST request and includes several key-value pairs. But what is the `action`?

By inspecting the "Form Data"in the developer tools, we can view the source:

To find the actual value of the `action`, we can inspect the page source:

The `action` value is the string in the second line.

Now, how do we obtain the IP address? The login system can be accessed from anywhere on campus, whether it's via Wi-Fi or a wired connection in a dorm room. The IP address is dynamic. How can we get it automatically?

Remember the first image?

The webpage automatically provides the IP address. The login system knows which IP is accessing it. We can simply extract it from the webpage.

Chrome provides convenient tools to quickly locate specific parts of the webpage's code. Firefox has a similar plugin called Firebug.

Using the "inspect element"tool (the magnifying glass icon), we can click on the IP address on the page.

The developer tools will show the HTML structure, revealing that the IP address is within a `<span>` tag with the class `login_txt`, specifically within its text content.

The `select` function in Jsoup can find elements matching a CSS query, and `first()` returns the first matching element.

With all the key-value pairs, we can now simulate the login process.

The POST request sends the key-value pairs in the request body. The `post.abort()` call interrupts the request. This is because we will reuse the `httpClient` for subsequent requests. The `org.apache.http` classes try to reuse the HTTP connection as much as possible. If a request is not terminated, sending another request using the same connection can cause an exception.

Why do we need to reuse the `httpClient`? I'll explain that later.

Submitting to `checkLogin.jsp` is not enough to connect to the network. This JSP only checks if the username and password are correct.

By adding a print statement, we can see the response code.

A 200 response indicates a successful login, 303 indicates incorrect credentials, and 404 indicates a connection error.

The second JSP script executed is `index.jsp`, which is a GET request.

After this, we still cannot connect to the network. We need to simulate a request to `connect_action.jsp`, which requires a `userid` parameter (e.g., `userid=88888`), corresponding to a student's ID. I noticed that the `userid` of the student with the previous student ID was just one less than mine. How can we obtain this value?

Fortunately, I remembered that we extracted the IP address from the webpage. Can we do the same for the `userid`?

This screenshot is from the disconnected page, but the same logic applies to `connect.jsp`. The source code of the page returned by the second JSP contains the key-value pair for the next JSP we need to request.

Here, we use a regular expression. `group(0)` is the entire match (e.g., `userid=88888`), and `group(1)` is the content within the parentheses (e.g., `88888`). `\d` matches any digit, and `+` means one or more occurrences. `find()` checks if the expression matches any part of the string, while `matches()` checks if the expression matches the entire string. Therefore, if the `src` is like `<frame userid=88888&ip="`, `matches()` will return `false` because the regex does not match the entire string.

Here's the Java code:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
```

```
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.cookie.Cookie;
import org.apache.http.impl.client.AbstractHttpClient;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

public class Login{

    static void print(String format, Object... args) {
        System.out.println(String.format(format, args));
    }

    public static void main(String[] args) {
        HttpClient httpClient = new DefaultHttpClient();
        String ip;
        String userId;
        String username="130888888";
        String password = "88888888";

        ip=cssQueryFirstText("http://login.bjfu.edu.cn/index.jsp","span.login_txt");

        doPost(httpClient,"http://login.bjfu.edu.cn/checkLogin.jsp",
                "username",username,"password",password,
                "ip",ip,"action","checkLogin.jsp");
        String content=doGet(httpClient,"http://login.bjfu.edu.cn/user/index.jsp",
          "ip",ip,"action","connect");
        userId=userId(content);
        doGet(httpClient,"http://login.bjfu.edu.cn/user/network/connect_action.jsp",
            "userid",userId,"ip",ip,"type","2");
    }

    static String userId(String html){
        Document doc=Jsoup.parse(html);
        Element elem=doc.select("frame#main").first();
```

```
    String src=elem.attr("src");
    Pattern pattern=Pattern.compile("userid=(\\d+)");
    Matcher matcher=pattern.matcher(src);
    String ans="";
    if(matcher.find()){
      ans=matcher.group(1);
    }
    return ans;
}

static String cssQueryFirstText(String url,String cssQuery) {
    String ip=null;
    try{
        Document doc=Jsoup.connect(url).get();
        Elements elems=doc.select(cssQuery);
        Element elem=elems.first();
        ip=elem.text();
    }catch(Exception e){
        e.printStackTrace();
    }
    return ip;
}

static String doGet(HttpClient httpClient, String url, String... pairs) {
    String entityContent=null;
    try {
        url = makeGetSrl(url, pairs);
        HttpGet get = new HttpGet(url);
        HttpResponse response = httpClient.execute(get);
        //print("%d", response.getStatusLine().getStatusCode());
        entityContent = entity(response);
        EntityUtils.consume(response.getEntity());
    } catch (IOException e) {
        e.printStackTrace();
    }
    return entityContent;
}

static void printEntity(HttpResponse rp){
    print("%s",entity(rp));
```

```
}

static String entity(HttpResponse response) {
    StringBuilder sb = new StringBuilder("");
    try {
        HttpEntity entity = response.getEntity();
        if (entity != null) {
            BufferedReader br = new BufferedReader(new InputStreamReader(
                    entity.getContent()));
            String line = null;
            while ((line = br.readLine()) != null) {
                sb.append(line + "\n");
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return sb.toString();
}

static String makeGetSrl(String url,String... pairs) {
    if(!url.endsWith("?")){
        url+="?";
    }
    List<NameValuePair>params=new LinkedList<NameValuePair>();
    int len=pairs.length;
    for(int i=0;i<len/2;i++){
        params.add(new BasicNameValuePair(pairs[2*i],pairs[2*i+1]));
    }
    String paramsStr=URLEncodedUtils.format(params,"utf-8");
    url+=paramsStr;
    return url;
}

static String doPost(HttpClient httpClient, String url, String... pairs) {
    String entityContent = null;
    try {
        HttpPost post = new HttpPost(url);
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        for (int i = 0; i < pairs.length / 2; i++) {
```

```
                params.add(new BasicNameValuePair(pairs[2 * i], pairs[2 * i + 1]));
            }
            post.setEntity(new UrlEncodedFormEntity(params, HTTP.UTF_8));
            HttpResponse response = httpClient.execute(post);
            entityContent = entity(response);
            //print("%d", response.getStatusLine().getStatusCode());
            post.abort();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return entityContent;
    }

    private static String getCookies(HttpClient client) {
        StringBuilder sb = new StringBuilder();
        List<Cookie> cookies = ((AbstractHttpClient)
                client).getCookieStore().getCookies();
        for(Cookie cookie: cookies)
            sb.append(cookie.getName() + "=" + cookie.getValue() + ";");
        return sb.toString();
    }
}
```

The complete source code is available on GitHub. Note that you may not be able to run it without a valid account for the university network.

The reason for reusing the same `HttpClient` is that it automatically stores cookies. The JSP scripts use cookies to determine if requests are from the same session. This is why copying a URL from Taobao to another browser might result in a timeout error.

`jsoup` and regular expressions are very useful tools. There are online tools to practice regular expressions. Have fun using them!