北京林業大學登入系統分析

這篇文章原本用中文撰寫並發表在 CSDN。

這篇文章詳細描述了對北京林業大學網絡登錄系統的反向工程過程。

目的是通過程式模擬登錄過程，有效地模仿使用者經由瀏覽器連接網絡的行為。

利用 Chrome 的開發者工具，可觀察到登錄過程中的網絡請求。

第一個請求是到 `CheckLogin.jsp`，接著是 `CheckLogin.jsp` 觸發的請求到 `index.jsp`。

讓我們更仔細地檢查 `CheckLogin.jsp`。

可以看到它使用 POST 請求並包含幾個鍵值對。但是 `action` 是什麼？

通過檢查開發者工具中的“Form Data”，可以查看源代碼：

要找到 `action` 的實際值，可以檢查頁面的源代碼：

`action` 值是第二行中的字串。

那麼，我們如何獲取 IP 地址？登錄系統可以從校園內的任何地方訪問，無論是通過 Wi-Fi 還是寢室的有線連接。IP 地址是動態的。我們如何自動獲取它？

記住第一張圖片嗎？

網頁會自動提供 IP 地址。登錄系統知道哪個 IP 在訪問它。我們可以直接從網頁中提取它。

Chrome 提供了方便的工具，可以快速定位網頁代碼的特定部分。Firefox 有一個類似的插件，稱為 Firebug。

使用“檢查元素”工具（放大鏡圖標），可以點擊頁面上的 IP 地址。

開發者工具會顯示 HTML 結構，顯示 IP 地址在 `<span>` 標籤中，特定在其文本內容中，該標籤的類為 `login_txt`。

Jsoup 的 `select` 函數可以找到與 CSS 查詢匹配的元素，`first()` 返回第一個匹配的元素。

有了所有的鍵值對，我們現在可以模擬登錄過程。

POST 請求在請求主體中發送鍵值對。`post.abort()` 請求中斷請求。這是因為我們將重用 `httpClient` 進行後續請求。`org.apache.http` 類試圖盡可能重用 HTTP 連接。如果請求未終止，使用相同連接發送另一個請求可能會導致異常。

為什麼我們需要重用 `httpClient`？我稍後會解釋。

提交到 `checkLogin.jsp` 還不足以連接到網絡。這個 JSP 只檢查用戶名和密碼是否正確。

通過添加一個打印語句，可以查看響應代碼。

200 响應代碼表示成功登錄，303 表示錯誤的憑證，404 表示連接錯誤。

第二個執行的 JSP 構建是 `index.jsp`，這是一個 GET 請求。

然後我們仍然無法連接到網絡。我們需要模擬一個請求到 connect_action.jsp，它需要一個 userid 參數 (例如，userid=88888)，對應於學生的 ID。我注意到上一個學生的 userid 只比我的小一個。我們如何獲得這個值？

幸運的是，我記得我們從網頁中提取了 IP 地址。我們能否對 userid 做同樣的操作？

這張截圖是從斷開的頁面，但相同的邏輯適用於 connect.jsp。第二個 JSP 返回的頁面的源代碼包含下一個 JSP 需要請求的鍵值對。

這裡我們使用正則表達式。group(0) 是整個匹配 (例如，userid=88888)，group(1) 是括號內的內容 (例如，88888)。\d 匹配任何數位，+ 表示一個或多個發生。find() 檢查表達式是否與字符串的任何部分匹配，而 matches() 檢查表達式是否與整個字符串匹配。因此，如果 src 類似於 <frame userid=88888&ip="，matches() 會返回 false，因為正則表達式不匹配整個字符串。

這裡是 Java 代碼：

```
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();
    }
}
```

完整的源代碼可以在 GitHub 上找到。注意，您可能需要有效的大學網絡帳戶才能運行它。

重用相同的 `HttpClient` 的原因是它會自動存儲 cookie。JSP 構建使用 cookie 來確定請求是否來自相同的會話。這就是為什麼從淘寶複製 URL 到另一個瀏覽器可能會導致超時錯誤。

`jsoup` 和正則表達式是非常有用的工具。有在線工具可以練習正則表達式。祝您使用愉快！