Die Analyse des Login-Systems der Beijing Forestry University

Dieser Beitrag wurde ursprünglich auf Chinesisch geschrieben und auf CSDN veröffentlicht.

Dieser Beitrag beschreibt den Prozess des Reverse Engineering des Netzwerks-Login-Systems, das an der Beijing Forestry University verwendet wird.

Das Ziel besteht darin, den Login-Prozess programmiert zu simulieren und somit die Handlungen eines Benutzers zu imitieren, der sich über einen Webbrowser mit dem Netzwerk verbindet.

Mit den Entwicklertools von Chrome können wir die Netzwerkanfragen während des Login-Prozesses beobachten.

Die erste Anfrage erfolgt an `CheckLogin.jsp`, gefolgt von einer Anfrage an `index.jsp`, die durch `CheckLogin.jsp` ausgelöst wird.

Lassen Sie uns `CheckLogin.jsp` genauer untersuchen.

Dabei sehen wir, dass es eine POST-Anfrage verwendet und mehrere Schlüssel-Wert-Paare enthält. Aber was ist der `action`?

Durch die Untersuchung der “Form Data”in den Entwicklertools können wir die Quelle einsehen:

Um den tatsächlichen Wert des `action` zu finden, können wir die Seitenquelle untersuchen:

Der `action`-Wert ist der String in der zweiten Zeile.

Nun, wie erhalten wir die IP-Adresse? Das Login-System kann von überall auf dem Campus aus zugreifen, sei es über Wi-Fi oder eine verdrahtete Verbindung in einem Wohnheim. Die IP-Adresse ist dynamisch. Wie können wir sie automatisch erhalten?

Erinnern Sie sich an das erste Bild?

Die Webseite liefert automatisch die IP-Adresse. Das Login-System kennt die IP-Adresse, die darauf zugreift. Wir können sie einfach aus der Webseite extrahieren.

Chrome bietet praktische Tools, um schnell spezifische Teile des HTML-Codes der Webseite zu lokalisieren. Firefox verfügt über ein ähnliches Plugin namens Firebug.

Mit dem “Element untersuchen”Werkzeug (das Lupe-Symbol) können wir auf die IP-Adresse auf der Seite klicken.

Die Entwicklertools zeigen die HTML-Struktur an und verraten, dass die IP-Adresse sich innerhalb eines `<span>`-Tags mit der Klasse `login_txt`, insbesondere innerhalb ihres Textinhalts, befindet.

Die `select`-Funktion in Jsoup kann Elemente finden, die einer CSS-Abfrage entsprechen, und `first()` gibt das erste passende Element zurück.

Mit allen Schlüssel-Wert-Paaren können wir nun den Login-Prozess simulieren.

Die POST-Anfrage sendet die Schlüssel-Wert-Paare im Anfragekörper. Der `post.abort()`-Aufruf unterbricht die Anfrage. Dies geschieht, weil wir den `httpClient` für nachfolgende Anfragen wiederverwenden. Die `org.apache.http`-Klassen versuchen, die HTTP-Verbindung so weit wie möglich wiederzuverwenden. Wenn eine Anfrage nicht beendet wird, kann das Senden einer weiteren Anfrage über dieselbe Verbindung zu einer Ausnahme führen.

Warum müssen wir den `httpClient` wiederverwenden? Das werde ich später erklären.

Das Senden an `checkLogin.jsp` reicht nicht aus, um eine Verbindung zum Netzwerk herzustellen. Dieses JSP überprüft nur, ob der Benutzername und das Passwort korrekt sind.

Durch das Hinzufügen einer Druckausgabe können wir den Antwortcode sehen.

Ein 200er-Code bedeutet eine erfolgreiche Anmeldung, 303 bedeutet falsche Anmeldeinformationen und 404 bedeutet einen Verbindungsfehler.

Das zweite JSP-Skript, das ausgeführt wird, ist `index.jsp`, das eine GET-Anfrage ist.

Danach können wir uns immer noch nicht mit dem Netzwerk verbinden. Wir müssen eine Anfrage an `connect_action.jsp` simulieren, die einen `userid`-Parameter (z.B. `userid=88888`) erfordert, der der Studentennummer entspricht. Ich bemerkte, dass der `userid` des vorherigen Studenten nur einer weniger war als meiner. Wie können wir diesen Wert erhalten?

Glücklicherweise erinnerte ich mich daran, dass wir die IP-Adresse aus der Webseite extrahiert haben. Können wir dasselbe mit dem `userid` tun?

Dieses Screenshot stammt von der Seite der getrennten Verbindung, aber dieselbe Logik gilt auch für `connect.jsp`. Der Quellcode der Seite, die vom zweiten JSP zurückgegeben wird, enthält das Schlüssel-Wert-Paar für das nächste JSP, das wir anfordern müssen.

Hier verwenden wir einen regulären Ausdruck. `group(0)` ist der gesamte Treffer (z.B. `userid=88888`), und `group(1)` ist der Inhalt innerhalb der Klammer (z.B. `88888`). `\d` entspricht jeder Ziffer und `+` bedeutet eine oder mehrere Vorkommen. `find()` überprüft, ob der Ausdruck einem Teil der Zeichenkette entspricht, während `matches()` überprüft, ob der Ausdruck der gesamten Zeichenkette entspricht. Daher, wenn die `src` wie `<frame userid=88888&ip=` ist, wird `matches() false` zurückgeben, weil der reguläre Ausdruck der gesamten Zeichenkette nicht entspricht.

Hier ist der 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();
    }
}
```

Der vollständige Quellcode ist auf GitHub verfügbar. Beachten Sie, dass Sie ihn möglicherweise nicht ohne einen gültigen Account für das Universitätsnetzwerk ausführen können.

Der Grund für die Wiederverwendung desselben `HttpClient` besteht darin, dass er automatisch Cookies speichert. Die JSP-Skripte verwenden Cookies, um zu bestimmen, ob Anfragen aus derselben Sitzung stammen. Daher könnte das Kopieren einer URL von Taobao in einen anderen Browser zu einem Timeout-Fehler führen.

`jsoup` und reguläre Ausdrücke sind sehr nützliche Tools. Es gibt Online-Tools, um reguläre Ausdrücke zu üben. Viel Spaß bei der Verwendung!