Analyse du système de connexion de l'Université Forestière de Pékin

Ce message a été initialement rédigé en chinois et publié sur CSDN.

Ce message détaille le processus de l'ingénierie inverse du système de connexion réseau utilisé à l'Université Forestière de Pékin.

L'objectif est de simuler le processus de connexion de manière programmatique, en imitant efficacement les actions d'un utilisateur se connectant au réseau via un navigateur web.

En utilisant les outils de développement de Chrome, nous pouvons observer les requêtes réseau effectuées pendant le processus de connexion.

La première requête est envoyée à `CheckLogin.jsp`, suivie d'une requête vers `index.jsp` déclenchée par `CheckLogin.jsp`.

Examinons de plus près `CheckLogin.jsp`.

On peut voir qu'il utilise une requête POST et inclut plusieurs paires clé-valeur. Mais quelle est l'action `action`?

En inspectant les "Form Data"dans les outils de développement, nous pouvons voir la source :

Pour trouver la valeur réelle de `action`, nous pouvons inspecter le code source de la page :

La valeur de `action` est la chaîne dans la deuxième ligne.

Maintenant, comment obtenir l'adresse IP ? Le système de connexion peut être accès de n'importe où sur le campus, que ce soit via Wi-Fi ou une connexion filaire dans une chambre. L'adresse IP est dynamique. Comment pouvons-nous l'obtenir automatiquement ?

Souvenez-vous de la première image ?

La page web fournit automatiquement l'adresse IP. Le système de connexion connaît l'IP qui y accède. Nous pouvons simplement l'extraire de la page web.

Chrome fournit des outils pratiques pour localiser rapidement des parties spécifiques du code de la page web. Firefox dispose d'un plugin similaire appelé Firebug.

En utilisant l'outil "inspecter l'élément"(l'icône de la loupe), nous pouvons cliquer sur l'adresse IP sur la page.

Les outils de développement montreront la structure HTML, révélant que l'adresse IP se trouve dans une balise `<span>` avec la classe `login_txt`, précisément dans son contenu texte.

La fonction `select` de Jsoup peut trouver les éléments correspondant à une requête CSS, et `first()` renvoie le premier élément correspondant.

Avec toutes les paires clé-valeur, nous pouvons maintenant simuler le processus de connexion.

La requête POST envoie les paires clé-valeur dans le corps de la requête. L'appel `post.abort()` interrompt la requête. Cela est dû au fait que nous réutiliserons le `httpClient` pour les requêtes suivantes. Les classes `org.apache.http` tentent de réutiliser la connexion HTTP autant que possible. Si une requête n'est pas terminée, l'envoi d'une autre requête en utilisant la même connexion peut provoquer une exception.

Pourquoi devons-nous réutiliser le `httpClient` ? Je vais l'expliquer plus tard.

Soumettre à `checkLogin.jsp` ne suffit pas pour se connecter au réseau. Ce JSP vérifie seulement si le nom d'utilisateur et le mot de passe sont corrects.

En ajoutant une instruction d'impression, nous pouvons voir le code de réponse.

Un code de réponse 200 indique une connexion réussie, 303 indique des informations d'identification incorrectes, et 404 indique une erreur de connexion.

Le deuxième script JSP exécuté est `index.jsp`, qui est une requête GET.

Après cela, nous ne pouvons toujours pas nous connecter au réseau. Nous devons simuler une requête vers `connect_action.jsp`, qui nécessite un paramètre `userid` (par exemple, `userid=88888`), correspondant à une carte d'étudiant. J'ai remarqué que le `userid` de l'étudiant ayant l'identifiant étudiant précédent était simplement un de moins que le mien. Comment pouvons-nous obtenir cette valeur ?

Heureusement, je me suis souvenu que nous avons extrait l'adresse IP de la page web. Pouvons-nous faire de même pour le `userid` ?

Cette capture d'écran provient de la page déconnectée, mais la même logique s'applique à `connect.jsp`. Le code source de la page retournée par le deuxième JSP contient la paire clé-valeur pour le prochain JSP que nous devons demander.

Ici, nous utilisons une expression régulière. `group(0)` est la correspondance entière (par exemple, `userid=88888`), et `group(1)` est le contenu à l'intérieur des parenthèses (par exemple, `88888`). `\d` correspond à tout chiffre, et `+` signifie une ou plusieurs occurrences. `find()` vérifie si l'expression correspond à une partie de la chaîne, tandis que `matches()` vérifie si l'expression correspond à la chaîne entière. Par conséquent, si le `src` est comme `<frame userid=88888&ip="`, `matches()` retournera `false` car le regex ne correspond pas à la chaîne entière.

Voici le code Java :

```
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();
    }
}
```

Le code source complet est disponible sur GitHub. Notez que vous pourriez ne pas pouvoir l'exécuter sans un compte valide pour le réseau de l'université.

La raison pour laquelle nous réutilisons le même `HttpClient` est qu'il stocke automatiquement les cookies. Les scripts JSP utilisent les cookies pour déterminer si les requêtes proviennent de la même session. C'est pourquoi copier une URL de Taobao dans un autre navigateur peut entraîner une erreur de délai d'attente.

`jsoup` et les expressions régulières sont des outils très utiles. Il existe des outils en ligne pour pratiquer les expressions régulières. Amusez-vous à les utiliser !