Análisis del sistema de acceso a la Universidad Forestal de Beijing

Esta publicación fue escrita originalmente en chino y publicada en CSDN.

Esta publicación detalla el proceso de ingeniería inversa del sistema de inicio de sesión de red utilizado en la Universidad Forestal de Pekín.

El objetivo es simular el proceso de inicio de sesión de manera programática, imitando efectivamente las acciones de un usuario que se conecta a la red a través de un navegador web.

Usando las herramientas de desarrollo de Chrome, podemos observar las solicitudes de red realizadas durante el proceso de inicio de sesión.

La primera solicitud es a `CheckLogin.jsp`, seguida de una solicitud a `index.jsp` que es desencadenada por `CheckLogin.jsp`.

Vamos a examinar `CheckLogin.jsp` más de cerca.

Podemos ver que utiliza una solicitud POST e incluye varias parejas clave-valor. Pero, ¿cuál es la `acción`?

Inspeccionando los “Datos del formulario”en las herramientas de desarrollo, podemos ver la fuente:

Para encontrar el valor real de la `acción`, podemos inspeccionar la fuente de la página:

El valor de la `acción` es la cadena en la segunda línea.

Ahora, ¿cómo obtenemos la dirección IP? El sistema de inicio de sesión se puede acceder desde cualquier lugar del campus, ya sea a través de Wi-Fi o una conexión por cable en una habitación de la residencia. La dirección IP es dinámica. ¿Cómo podemos obtenerla automáticamente?

Recuerda la primera imagen?

La página web proporciona automáticamente la dirección IP. El sistema de inicio de sesión sabe qué IP lo está accediendo. Simplemente podemos extraerla de la página web.

Chrome proporciona herramientas convenientes para localizar rápidamente partes específicas del código de la página web. Firefox tiene un complemento similar llamado Firebug.

Usando la herramienta “inspeccionar elemento”(el icono de la lupa), podemos hacer clic en la dirección IP en la página.

Las herramientas de desarrollo mostrarán la estructura HTML, revelando que la dirección IP está dentro de una etiqueta `<span>` con la clase `login_txt`, específicamente dentro de su contenido de texto.

La función `select` en Jsoup puede encontrar elementos que coincidan con una consulta CSS, y `first()` devuelve el primer elemento coincidente.

Con todas las parejas clave-valor, ahora podemos simular el proceso de inicio de sesión.

La solicitud POST envía las parejas clave-valor en el cuerpo de la solicitud. La llamada `post.abort()` interrumpe la solicitud. Esto se debe a que reutilizaremos el `httpClient` para solicitudes posteriores. Las clases `org.apache.http` intentan reutilizar la conexión HTTP siempre que sea posible. Si una solicitud no se termina, enviar otra solicitud usando la misma conexión puede causar una excepción.

¿Por qué necesitamos reutilizar el `httpClient`? Lo explicaré más tarde.

Enviar a `checkLogin.jsp` no es suficiente para conectarse a la red. Este JSP solo verifica si el nombre de usuario y la contraseña son correctos.

Agregando una declaración de impresión, podemos ver el código de respuesta.

Un código de respuesta 200 indica un inicio de sesión exitoso, 303 indica credenciales incorrectas y 404 indica un error de conexión.

El segundo script JSP ejecutado es `index.jsp`, que es una solicitud GET.

Después de esto, aún no podemos conectarnos a la red. Necesitamos simular una solicitud a `connect_action.jsp`, que requiere un parámetro `userid` (por ejemplo, `userid=88888`), correspondiente a la identificación de un estudiante. Noté que el `userid` del estudiante con la identificación del estudiante anterior era simplemente uno menos que el mío. ¿Cómo podemos obtener este valor?

Afortunadamente, recordé que extrajimos la dirección IP de la página web. ¿Podemos hacer lo mismo para el `userid`?

Esta captura de pantalla es de la página desconectada, pero la misma lógica se aplica a `connect.jsp`. El código fuente de la página devuelta por el segundo JSP contiene la pareja clave-valor para el siguiente JSP que necesitamos solicitar.

Aquí, usamos una expresión regular. `group(0)` es la coincidencia completa (por ejemplo, `userid=88888`), y `group(1)` es el contenido dentro de los paréntesis (por ejemplo, `88888`). `\d` coincide con cualquier dígito, y `+` significa una o más ocurrencias. `find()` verifica si la expresión coincide con alguna parte de la cadena, mientras que `matches()` verifica si la expresión coincide con toda la cadena. Por lo tanto, si el `src` es como `<frame userid=88888&ip=`, `matches()` devolverá `false` porque la expresión regular no coincide con toda la cadena.

Aquí está el código en 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();
    }
}
```

El código fuente completo está disponible en GitHub. Tenga en cuenta que es posible que no pueda ejecutarlo sin una cuenta válida para la red de la universidad.

La razón para reutilizar el mismo `HttpClient` es que almacena automáticamente las cookies. Los scripts JSP utilizan cookies para determinar si las solicitudes provienen de la misma sesión. Esto es por lo que copiar una URL de Taobao a otro navegador puede resultar en un error de tiempo de espera.

`jsoup` y las expresiones regulares son herramientas muy útiles. Hay herramientas en línea para practicar expresiones regulares. ¡Diviértete usándolas!