File: SimpleB2BUAProxyServlet.java

```
 1 package no.ubisafe.sipproxy;
 2
 3 import javax.annotation.Resource;
 4 import java.io.BufferedInputStream;
 5 import java.io.IOException;
 6 import java.io.InputStream;
 7 import java.net.InetAddress;
 8 import java.net.UnknownHostException;
 9 import java.util.List;
10 import java.util.Properties;
11
12 import javax.servlet.Servlet;
13 import javax.servlet.ServletException;
14 import javax.servlet.sip.B2buaHelper;
15 import javax.servlet.sip.SipErrorEvent;
16 import javax.servlet.sip.SipErrorListener;
17 import javax.servlet.sip.SipFactory;
18 import javax.servlet.sip.SipServlet;
19 import javax.servlet.sip.SipServletMessage;
20 import javax.servlet.sip.SipServletRequest;
21 import javax.servlet.sip.SipServletResponse;
22 import javax.servlet.sip.SipURI;
23
24 import no.ubisafe.unifid.supplicant.serviceSupplicant.core.*;
25
26 public class SimpleB2BUAProxyServlet
27             extends SipServlet
28             implements SipErrorListener, Servlet
29 {
30   @Resource
31   public SipFactory sipFactory;
32   //Instead of
33   //SipFactory sipFactory = (SipFactory)
     getServletContext().getAttribute("javax.servlet.sip.SipFactory");
34
35   private Properties settings = null;
36     private String pincode = null;
37
38   /** Creates a new instance of SimpleB2BUAProxyServlet */
39     public SimpleB2BUAProxyServlet()
40     {
41       loadProperties();
42     }
43
44     private void copyContent(SipServletMessage source,
       SipServletMessage dest) throws IOException {
45         if (source.getContentLength() > 0)
46         {
47           try
48           {
49             dest.setContent(source.getContent(),
               source.getContentType());
50               String enc = source.getCharacterEncoding();
51               if (enc != null && enc.length() > 0)
52               {
53                   dest.setCharacterEncoding(enc);
54               }
55           } catch (Exception ex)
56           {
57             System.out.println("INFO: copyContent; Exception: " +
               ex.getMessage());
58           }
59         }
```

```
60      }
61
62      private void replaceLocalhost(SipServletMessage message)
63      {
64          // Get IP Address
65        InetAddress addr = null;
66        try
67        {
68          addr = InetAddress.getLocalHost();
69          }
70        catch (UnknownHostException ex)
71        {
72          System.out.println("INFO: UnknownHostException: " +
          ex.getMessage());
73          return;
74        }
75          byte[] ipAddr = addr.getAddress();
76          String ipAddrStr =
77            (ipAddr[0] & 0xff) + "." +
78            (ipAddr[1] & 0xff) + "." +
79            (ipAddr[2] & 0xff) + "." +
80            (ipAddr[3] & 0xff);
81
82          String value = null;
83
84          value = message.getHeader("Contact");
85          value = value.replace("127.0.0.1", ipAddrStr);
86          message.setHeader("Contact", value);
87      }
88
89      protected void doRegister(SipServletRequest request)
90        throws ServletException, IOException
91    {
92        System.out.println("\n\nINFO: doRegister; Got request:\n" +
          request);
93
94        B2buaHelper helper = request.getB2buaHelper();
95
96      SipServletRequest nextRequest = helper.createRequest(request);
97      helper.linkSipSessions(request.getSession(),
        nextRequest.getSession());
98        System.out.println("INFO: request.getRequestURI: " +
          request.getRequestURI());
99
100       try
101       {
102           // First send TRYING to CLIENT
103           int response_code = SipServletResponse.SC_TRYING;
104     SipServletResponse response_trying =
        request.createResponse(response_code);
105
106           System.out.println("INFO: sending TRYING");
107           response_trying.send();
108       }
109       catch (Exception ex)
110       {
111         System.out.println("INFO: sending TRYING; Exception: \n");
112         ex.printStackTrace();
113       }
114
115       try
116       {
117         // Then send request to SIP server
```

```
118           nextRequest.getSession().setAttribute("originalRequest",
              request);
119
120           nextRequest.setRequestURI(request.getRequestURI());
121           replaceLocalhost(nextRequest);
122
123           System.out.println("INFO: register next: \n"+nextRequest);
124         nextRequest.send();
125         }
126         catch (Exception ex)
127         {
128           System.out.println("INFO: nextRequest; Exception: \n");
129           ex.printStackTrace();
130         }
131
132       System.out.println("\n");
133     }
134
135       protected void doAck(SipServletRequest request)
136       throws ServletException, IOException
137     {
138         System.out.println("\n\nINFO: doAck; Got request:\n" +
          request);
139
140       try
141       {
142           SipServletResponse okResponse =
            (SipServletResponse)request.getSession().getAttribute("okResp
            onse");
143           SipServletRequest ack = okResponse.createAck();
144         System.out.println("INFO: nextRequest; send Ack:\n " + ack);
145           ack.send();
146         }
147       catch (Exception ex)
148       {
149         System.out.println("INFO: nextRequest; send Ack;
          Exception:\n");
150         ex.printStackTrace();
151       }
152
153       System.out.println("\n");
154     }
155
156       protected void doRequest(SipServletRequest request)
157         throws ServletException, IOException
158     {
159         if (request.getMethod().equals("REGISTER"))
160         {
161           super.doRequest(request);
162         }
163         else
164         if (request.getMethod().equals("ACK"))
165         {
166           super.doRequest(request);
167         }
168         else
169         {
170               System.out.println("\n\nINFO: Got request; Method: " +
                  request.getMethod() + ":\n" + request);
171
172               try {
173               B2buaHelper helper = request.getB2buaHelper();
174               SipServletRequest nextRequest;
```

```
175             if (request.isInitial()) {
176               nextRequest = helper.createRequest(request, true,
                  null);
177             } else {
178               nextRequest =
                  helper.createRequest(helper.getLinkedSession(request.ge
                  tSession()), request, null);
179             }
180             copyContent(request, nextRequest);
181             System.out.println("\nINFO: Sending Next request:\n" +
                nextRequest);
182             nextRequest.send();
183           }
184           catch (Exception ex)
185           {
186             System.out.println("INFO: Sending Next request
                (Exception)");
187             ex.printStackTrace();
188           }
189       }
190   }
191
192     protected void doResponse(SipServletResponse response)
193         throws ServletException, IOException
194     {
195       System.out.println("INFO: Got response:\n" + response);
196       System.out.println("INFO: Method == " + response.getMethod() +
          "; Status: " + response.getStatus() + " "
197         + response.getReasonPhrase());
198
199         // Intercept response with method REGISTER
200         boolean blnChallenge = false;
201         boolean blnRegister = false;
202
203         if (response.getMethod().equals("REGISTER")) {
204             boolean firstResponseRecieved =
                "true".equals(getServletContext().getAttribute("FirstResp
                onseRecieved"));
205
206         // ... and Status == "401" (UNAUTHORIZED) and
            Www-Authenticate
207           if ( response.getStatus() ==
              SipServletResponse.SC_UNAUTHORIZED) {
208             AuthHeaderProcessor ahp = new AuthHeaderProcessor();
209
210             // WWW-Authenticate in header?
211             List<String> headers = ahp.getHeaderValues(response,
                "WWW-Authenticate");
212           // Avoid re-sending if the authentication repeatedly fails.
213               // Challenge only if first response not handled
214             if (headers.size()>0 && !firstResponseRecieved)
              blnChallenge = true;
215           }
216
217           // Only return RESPONSE to original request if first
              response already handled
218             if (firstResponseRecieved) blnRegister = true;
219         }
220
221         // Intercept response with method INVITE
222         boolean blnInvite = false;
223         if (response.getMethod().equals("INVITE")) {
224         // ... and Status == "401" (UNAUTHORIZED) and
```

```
            Www-Authenticate
225           if ( response.getStatus() == SipServletResponse.SC_OK) {
226             blnInvite = true;
227           }
228         }
229
230         if (blnChallenge)
231         {
232       getServletContext().setAttribute("FirstResponseRecieved",
          "true");
233
234           System.out.println("INFO: Challenge == true");
235
236             // Then create new REQUEST with CHALLENGE-RESPONSE and
                send to server
237
238           System.out.println("INFO: RemoteAddr == " +
              response.getRemoteAddr());
239           String username = null;
240
241           // Use IMSI for username
242           username = settings.getProperty("imsi");
243
244             if (response.getTo().getURI().isSipURI()) {
245               username = ((SipURI)
                  response.getTo().getURI()).getUser();
246
247               System.out.println("INFO: username == " + username);
248             }
249
250           System.out.println("INFO: challengeRequest");
251       SipServletRequest challengeRequest =
          response.getSession().createRequest(
252         response.getRequest().getMethod());
253
254       SipServletRequest originalRequest = (SipServletRequest)
          response.getSession().getAttribute("originalRequest");
255
256       String contact = originalRequest.getHeader("Contact");
257       if (contact != null)
258       {
259             System.out.println("INFO: Contact==" + contact);
260         challengeRequest.addHeader("Contact", contact);
261       }
262
263           System.out.println("INFO: AuthInfoImpl");
264
265           // Create authentication data here - must include data from
              SIM authentication instead of static password
266           String password = authenticate();
267           System.out.println("INFO: password == " + password);
268
269           AuthInfoImpl ai = new AuthInfoImpl();
270             ai.addAuthInfo(
271             -1, // NOT USED
272             response.getRemoteAddr(),
273             username,
274             password
275           );
276
277           System.out.println("INFO: ClientDigestCreator");
278             ClientDigestCreator cdc = new ClientDigestCreator();
279         cdc.createDigest(ai, challengeRequest, response);
```

```
280
281           replaceLocalhost(challengeRequest);
282
283           try
284             {
285               System.out.println("\nINFO: challengeRequest
                  (sending):\n" + challengeRequest);
286                 challengeRequest.send();
287             }
288             catch (Exception ex)
289             {
290               System.out.println("INFO: challengeRequest; Exception:
                  \n");
291               ex.printStackTrace();
292             }
293           }
294           else if (blnRegister)
295           {
296             getServletContext().setAttribute("FirstResponseRecieved",
                "false");
297
298         SipServletRequest originalRequest = (SipServletRequest)
            response.getSession().getAttribute("originalRequest");
299         SipServletResponse responseToOriginalRequest =
            originalRequest.createResponse(response.getStatus());
300         System.out.println("INFO: Sending response on first REGISTERED
            request: \n" +  responseToOriginalRequest);
301
            responseToOriginalRequest.setContentLength(response.getContentL
            ength());
302         responseToOriginalRequest.send();
303           }
304           else if (blnInvite)
305           {
306           System.out.println("INFO: 200 OK response to INVITE, storing
              response");
307
308             B2buaHelper helper =
                response.getRequest().getB2buaHelper();
309               SipServletRequest otherReq =
                  helper.getLinkedSipServletRequest(response.getRequest());
310
311               otherReq.getSession().setAttribute("okResponse",
                  response);
312
313               SipServletResponse other =
                  otherReq.createResponse(response.getStatus(),
                  response.getReasonPhrase());
314             try
315             {
316               System.out.println("INFO: response.getContent(): \n" +
                  response.getContent());
317               System.out.println("INFO: getContentType(): \n" +
                  response.getContentType());
318                   other.setContent(response.getRawContent(),
                      response.getContentType());
319             }
320             catch (Exception ex)
321             {
322               System.out.println("INFO: Sending (Exception)");
323               ex.printStackTrace();
324             }
325             try
```

```
326             {
327               System.out.println("INFO: Sending B2buaHelper generated
                  response: \n" +  other);
328               other.send();
329             }
330             catch (Exception ex)
331             {
332               System.out.println("INFO: Sending (Exception)");
333               ex.printStackTrace();
334             }
335           }
336           else
337           {
338             B2buaHelper helper =
                  response.getRequest().getB2buaHelper();
339               SipServletRequest otherReq =
                  helper.getLinkedSipServletRequest(response.getRequest());
340               SipServletResponse other =
                  otherReq.createResponse(response.getStatus(),
                  response.getReasonPhrase());
341             try
342             {
343               System.out.println("INFO: Sending B2buaHelper generated
                  response: \n" +  other);
344               other.send();
345             }
346             catch (Exception ex)
347             {
348               System.out.println("INFO: Sending (Exception)");
349               ex.printStackTrace();
350             }
351           }
352
353       System.out.println("\n");
354       }
355
356       // SipErrorListener
357
358       public void noAckReceived(SipErrorEvent ee)
359       {
360           System.out.println("INFO: Error: noAckReceived.");
361       }
362
363       public void noPrackReceived(SipErrorEvent ee)
364       {
365           System.out.println("INFO: Error: noPrackReceived.");
366       }
367
368       /**
369        * Performs authentication with SIM and returns authentication
          token for use in the
370        * final REGISTER request.
371        *
372        */
373       private String authenticate()
374       {
375         System.out.println("INFO: settings == " + settings);
376         System.out.println("INFO: authenticator_host == " +
          settings.getProperty("authenticator_host"));
377         System.out.println("INFO: authenticator_port == " +
          settings.getProperty("authenticator_port"));
378         System.out.println("INFO: supplicant_minseclevel == " +
          settings.getProperty("supplicant_minseclevel"));
```

```
379         System.out.println("INFO: supplicant_service_id == " +
            settings.getProperty("supplicant_service_id"));
380
381         try {
382           Credentials.pincode = (new
            String(pincode)).getBytes("US-ASCII");
383               ServiceCore core = new ServiceCore(
                  settings.getProperty("authenticator_host"),
384           Integer.parseInt(settings.getProperty("authenticator_port")),
385
              Integer.parseInt(settings.getProperty("supplicant_minseclevel
              ")),
386           ServiceCore.DAEMON,
387           ServiceCore.TOKEN_TYPE_USB,
388           settings.getProperty("supplicant_service_id")
389         );
390
391         System.out.println("INFO: core == " + core);
392         if (core != null)
393           return Credentials.imsi_password.toUpperCase();
394
395         } catch (Exception e) {
396           System.out.println("DEBUG: authenticate(): " +
              e.getMessage());
397           e.printStackTrace();
398         }
399         return null;
400       }
401
402     /**
403      * Loads settings from bundled properties file.
404      *
405      */
406     private void loadProperties()
407     {
408         System.out.println("INFO: Loading properties for supplicant");
409
410         pincode = System.getProperty("PINCODE");
411
412       settings = new Properties();
413       Object obj =
          getClass().getClassLoader().getResourceAsStream("supplicant.prope
          rties");
414       obj = new BufferedInputStream((InputStream) obj);
415       try {
416         settings.load((InputStream) obj);
417       } catch (IOException e) {
418         System.out.println("Exception in loadProperties(): " +
          e.getMessage());
419       }
420     }
421 }
```