Programmazione Web

Lezione del 30 Aprile 2013

Docente: Novella Bartolini

Protezione delle risorse - Autenticazione

- Un utente tenta di accedere ad una risorsa protetta (es. una pagina jsp).
- Se l'utente è già stato autenticato la risorsa viene resa disponibile, altrimenti viene chiesto all'utente di digitare nome utente e password.
- Se nome e password non possono essere autenticati, viene visualizzato un errore e l'utente ha la possibilità di scrivere nuovamente i dati, altrimenti la risorsa viene resa disponibile.

Autenticazione dichiarativa vs. programmata

- L'approccio dichiarativo all'autenticazione è basato interamente sul servlet container che gestisce i nomi degli utenti, le password e i ruoli
- L'approccio programmato implica invece la gestione diretta della sicurezza da parte di servlet e pagine jsp (come nell'esempio del tag EnforceLogin)

Autenticazione dichiarativa

- Gli aspetti relativi alla sicurezza sono interamente gestiti dal serlvet container.
- Per prevenire accessi non autorizzati
  - Si utilizza il descrittore della web application (*web.xml*) per dichiarare che certe risorse sono riservate a utenti che rivestono certi ruoli.
  - Si definisce un metodo di autenticazione per identificare gli utenti e rispettivi ruoli.
  - Quando viene richiesta una risorsa protetta, il server richiede all'utente username e password, li confronta con un set predefinito e, automaticamente, tiene traccia degli utenti già autenticati. Questo processo è completamente trasparente alle servlet e alle pagine JSP.
- Per preservare la sicurezza dei dati sulla rete
  - Si usa il descrittore della web application per specificare che a certe risorse si può accedere solo attraverso una connessione https. Se gli utenti provano ad usare una connessione HTTP vengono forzati ad usare HTTPS con meccanismi di redirezione.

Autenticazione programmata

- Le risorse protette (servlet e pagine JSP) sono responsabili della gestione della propria sicurezza.
  - Portabilità del codice. Non ci sono elementi della web application che dipendono dal particolare server utilizzato. Non sono necessarie ulteriori specifiche nel descrittore.
- Per prevenire accessi non autorizzati
  - Ciascuna servlet o pagina JSP deve autenticare l'utente o verificare che sia già stato autenticato.
- Per preservare la sicurezza dei dati sulla rete
  - Ogni servlet o pagina JSP deve controllare il protocollo usato
  - Se gli utenti provano ad usare una connessione HTTP la servlet o la pagina JSP devono redirigere le richieste sul protocollo HTTPS.

Tipi di autenticazione dichiarativa

1. Autenticazione BASIC
2. Autenticazione basata su form
3. Autenticazione DIGEST

- La scelta del meccanismo viene specificata nel file web.xml in corrispondenza dei tag
  - **<login-config>** e **<auth-method>**

Principali e ruoli

- L'utente è un *principal*
- I *principal* sono entità denominate, che in genere rappresentano singoli individui o società.
- I principal possono ricoprire uno o più *ruoli (roles)*
  - Es: un cliente può essere nello stesso tempo un dipendente
- Le restrizioni di sicurezza della specifica delle servlet, associano i ruoli con le risorse protette

Restrizioni di sicurezza: /WEB-INF/web.xml

```
<web-app>
    <display-name>Web App x test meccanismi di sicurezza </display-name>
    <description> Test dei meccanismi di sicurezza </description>
    <login-config> … vedremo tra poco …  </login-config>
    <security-constraint>
    <web-resource-collection> <!-- risorse protette -->
        <web-resource-name>Una pagina protetta </web-resource-name>
        <url-pattern>/jsps/pagina-protetta.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint><!-- ruoli associati alle risorse -->
        <role-name>tomcat</role-name>
        <role-name>role1</role-name>
    </auth-constraint>
    </security-constraint>
</web-app>
```

Elementi di <security-constraint>

Elemento	Tipo*	Descrizione
`web-resource-collection`	+	Un sottoinsieme delle risorse di un'applicazione web al quale vengono applicate le restrizioni di sicurezza
`auth-constraint`	?	Le restrizioni di autorizzazione collocate su una o più raccolte di risorse web (conterrà i tag che identificano i ruoli)
`user-data-constraint`	?	Una specifica sul meccanismo di trasporto dei dati inviati tra un client e un servlet container

* Tipo: += uno o più

?= uno, facoltativo

Elementi <web-resource-collection>

Elemento	Tipo*	Descrizione
web-resource-name	**1**	Il nome di una risorsa web
description	**?**	La descrizione di una risorsa web
url-pattern	*****	Uno schema url associato ad una risorsa web
http-method	**?**	Un metodo http associato ad una risorsa web

* Tipo: 1= uno, obbligatorio

?= zero o più

*=uno o più

Elementi <auth-constraint>

Elemento	Tipo*	Descrizione
description	**?**	Una descrizione della restrizione di un'autorizzazione
role-name	*****	Il ruolo al quale si applica la restrizione

* Tipo: 1= uno, obbligatorio

?= zero o più

*=uno o più

Risorse – Ruoli - Principali

- Le restrizioni specificate nel file web.xml associano i ruoli alle risorse.
- L'associazione dei ruoli ai principali è compito del servlet container
  - (file `<catalina-home>/conf/tomcat-users.xml`).

File di configurazione
<CATALINA-HOME>/conf/tomcat-users.xml

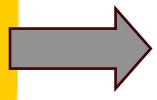
```
<tomcat-users>
 . . .
        <role rolename="cliente"/>
        <user name="rossi" password="tomcat" roles="cliente", "altro" />
 . . .
</tomcat-users>
```

Metodi di `HttpServletRequest`

- `Principal getUserPrincipal()`
  - Restituisce un riferimento a `java.security.Principal`
- `Boolean isUserInRole(String)`
  - Stabilisce se un utente ricopre un ruolo specificato da una stringa
- `String getRemoteUser()`
  - Restituisce il nome utente utilizzato per il login

Metodi di `HttpServletRequest` (cont.)

- Le API della servlet non forniscono i metodi "setter" corrispondenti ai metodi "getter" getUserPrincipal() e getRemoteUser()



Le applicazioni non possono impostare principali e ruoli; questi possono essere impostati solo dai contenitori di servlet

Metodi di `HttpServletRequest`(cont.)

- String `getAuthType`()
  - Restituisce il tipo di autenticazione BASIC, FORM, DIGEST.
- Boolean `isSecure`()
  - Restituisce true se la connessione è http o https
- String `getScheme`()
  - Fornisce lo schema che rappresenta il meccanismo di trasporto: http, https, ftp o altro.

/WEB-INF/web.xml (aut. dichiarativa)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
. . .
<web-app>
. . .
    <security-constraint>
    . . .
    </security-constraint>

    <login-config>
       <auth-method> BASIC </auth-method>
       <realm-name>Esempio di autenticazione Basic </realm-name>
    </login-config>
</web-app>
```

Il metodo **`HttpServletRequest.getAuthType()`** fornisce il metodo di autenticazione utilizzato

Autenticazione BASIC

- Quando un client tenta di accedere ad una risorsa protetta, il servlet container invia automaticamente una finestra di richiesta di nome utente e password
- L'invio della finestra di richiesta avviene in modo trasparente alle pagine JSP e alle servlet
- Questo metodo manca completamente di sicurezza. Le password vengono trasmesse con la codifica base64

Codifica base64

- La password viene trasformata in una sequenza di caratteri appartenenti ad un sottogruppo del set di caratteri ASCII. In questo modo, ogni carattere codificato può essere rappresentato con sei bit.
- Ogni gruppo di 24 bit viene diviso in quattro gruppi di sei bit, ad ognuno dei quali si associa il corrispondente carattere ASCII appartenente al sottogruppo specificato.
- Le password sono estremamente vulnerabili

Esempio sull’autenticazione BASIC

Web Application di test (aut. BASIC)
/JSPS/pagina-protetta.jsp

```
<html><head><title>Una pagina protetta </title></head>
<body>
<%@ include file="show-security.jsp" %>

<%  if (request.isUserInRole("tomcat")){ %>
        Appartieni al ruolo <i>tomcat</i> <br>
<%      } else { %>
        Non appartieni al ruolo <i>tomcat</i><br>
<%      }  %>

<% if (request.isUserInRole("role1")) { %>
       Appartieni al ruolo <i>role1</i> <br>
<%     } else { %>
       Non appartieni al ruolo <i>role1</i><br>
<%     }  %>

</body></html>
```

Web Application di test (aut. BASIC)

/JSPS/show-security.jsp

```
<font size="4" color="blue">
   Informazioni sulla sicurezza
</font> <br>
User principal: <%= request.getUserPrincipal().getName() %>. <br>
Request authenticated with: <%= request.getAuthType() %>. <br>

<% if (request.isSecure()) { %>
      This connection is secure. <br>
<% } else { %>
      This connection is NOT secure. <br>
<% } %>

Server Address: <%= request.getServerName() %> <br>
Remote Host: <%= request.getRemoteHost() %> <br>
Remote Addr; <%= request.getRemoteAddr() %>
```

Web Application di test (aut. BASIC)
/WEB-INF/web.xml

```
<web-app>
    <display-name>Web App x test meccanismi di sicurezza </display-name>
    <description>
    Test dei meccanismi di sicurezza
    </description>

    <security-constraint>
    <!-- risorse protette -->
    <web-resource-collection>
        <web-resource-name>Una pagina protetta </web-resource-name>
        <url-pattern>/jsps/pagina-protetta.jsp</url-pattern>
    </web-resource-collection>
. . .
```

Web Application di test (aut. BASIC) /WEB-INF/web.xml (cont.)

```
. . .
    <auth-constraint>
        <!-- ruoli associati alle risorse indicate sopra -->
        <role-name>tomcat</role-name>
        <role-name>role1</role-name>
    </auth-constraint>
    </security-constraint>

    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>Basic Authentication Example</realm-name>
    </login-config>
</web-app>
```

<CATALINA-HOME>/conf/tomcat-users.xml

```
<tomcat-users>
    <user name="tomcat" password="tomcat" roles="tomcat" />
    <user name="role1" password="tomcat" roles="role1" />
    <user name="both" password="tomcat" roles="tomcat","role1" />
</tomcat-users>
```

Il file tomcat-users.xml appartiene al servlet container e realizza l'associazione tra il principal e il ruolo da lui ricoperto

Autenticazione basata su form

- Permette di controllare l'aspetto e il comportamento della pagina di login
- Il metodo basato su form funziona come il metodo basic ma viene visualizzata la pagina di login anzichè una finestra di dialogo
- Viene inoltre specificata una pagina di errore per il caso di mancata autenticazione
- La password viene trasmessa con la codifica base64

Autenticazione basata su form (cont.)

- Implementare una pagina di login
- Implementare una pagina di errore, che verrà visualizzata in caso di mancata autenticazione
- Nel descrittore della web application specificare che si adotta l'autenticazione basata su form e il nome delle pagine di login e di errore

Web Application di test (aut. FORM)
/JSPS/login.jsp

```
<html><head><title>Pagina di login</title></head>
<body>
<font size="4" color="blue">
    Per favore inserisci i tuoi dati:<br></font>

<form action="j_security_check" method="POST">
<table>
<tr><td>Name:</td>
    <td><input type="text" name="j_username"></td></tr>
<tr><td>Password:</td>
    <td><input type="password" name="j_password" size="8"></td>
</tr>
</table>
<br>
  <input type="submit" value="login">
</form>
</body>
</html>
```

Web Application di test (aut. FORM)
/JSPS/login.jsp

- **`j_username`**, **`j_password`**, **`j_security_check`** sono nomi dettati dalla servlet specification
- `j_username`: il nome del campo username
- `j_password`: il nome del campo password
- `j_security_check`: l'azione del modulo di login

- La pagina di login così definita viene automaticamente richiamata dal servlet container quando si tenta di accedere alla risorsa protetta specificata nel descrittore.

Web Application di test (aut. FORM)
/JSPS/error.jsp

```
<html><head><title>ERRORE !!!</title></head>

<body>

<font size="6" color="red">
    I dati inseriti non sono validi!<br>
</font>

Clicca <a href=' <%=response.encodeURL("login.jsp") %>'>qui</a> per
riprovare!

</body>
</html>
```

Web Application di test (aut. FORM)
/WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <display-name>test di autenticazione mediante form</display-name>
    <description>Test </description>
    <security-constraint>
    <!-- risorse protette -->
    <web-resource-collection>
        <web-resource-name>Una pagina protetta </web-resource-name>
        <url-pattern>/jsps/pagina-protetta.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <!-- ruoli associati alle risorse indicate sopra -->
        <role-name>tomcat</role-name>
        <role-name>role1</role-name>
    </auth-constraint>
    </security-constraint> . . .
```

Web Application di test (aut. FORM)
/WEB-INF/web.xml (cont.)

```
. . .
<login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
            <form-login-page>/jsps/login.jsp</form-login-page>
            <form-error-page>/jsps/error.jsp</form-error-page>
        </form-login-config>
    </login-config>
</web-app>
```

/WEB-INF/web.xml

```
. . .
<web-app>
. . .
    <security-constraint>
    <web-resource-collection> . . .
        <url-pattern>/jsps/risorsaA.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint> . . .
        <role-name>ruolo A</role-name>
    </auth-constraint>
    </security-constraint>
    <security-constraint>
    <web-resource-collection> . . .
        <url-pattern>/jsps/risorsaB.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint> . . .
        <role-name>ruolo B</role-name>
    </auth-constraint>
 /security-constraint> <
</web-app>
```

Meccanismi di autenticazione dichiarativa

- BASIC (equiv. FORM) e DIGEST
- Sono definiti nel documento RFC 2617, http://ftp.isi.edu/in-notes/rfc2617.txt
- BASIC e FORM: la password viene trasmessa in chiaro
- DIGEST: la password non viene trasmessa, ma viene trasmesso un valore hash.

Autenticazione DIGEST

- Quando un client tenta di accedere ad una risorsa protetta, il servlet container invia automaticamente una finestra di richiesta di nome utente e password
- L'invio della finestra di richiesta avviene in modo trasparente alle pagine JSP e alle servlet
- La password viene codificata con metodi di hashing

Codifica con il metodo DIGEST

- Lo schema Digest è basato su un meccanismo di domanda-risposta.
- L'entità responsabile dell'autenticazione (il server) propone all'utente un valore *nonce* (ogni volta diverso e valido solo per la richiesta corrente)
- Affinchè l'utente possa essere autenticato, deve rispondere con un valore di checksum calcolato in base a
  - Username e Password
  - Il valore nonce proposto
  - Il metodo HTTP
  - URI

/WEB-INF/web.xml (aut. dichiarativa)

Per realizzare l'autenticazione secondo lo schema Digest ci si comporta esattamente come per il metodo basic, ma si specifica il metodo **DIGEST** nel file /WEB-INF/web.xml

```
<web-app>
. . .
    <security-constraint>
    . . .
    </security-constraint>

   <login-config>
     <auth-method> DIGEST </auth-method>
     <realm-name>Esempio di autenticazione Digest </realm-name>
   </login-config>
</web-app>
```

Autenticazione dichiarativa vs. programmata

- L'approccio dichiarativo all'autenticazione è basato interamente sul servlet container che gestisce i nomi degli utente, le password e i ruoli
- L'approccio programmato implica invece la gestione diretta della sicurezza da parte di servlet e pagine jsp

Problemi con meccanismi dichiarativi

- L'accesso è tutto-o-niente
  - Gli utenti possono accedere alle risorse protette oppure essere bloccati, senza alternative ulteriori
  - Non è possibile proporre agli utenti contenuti diversi a seconda dei ruoli ricoperti.
- L'accesso, basato esclusivamente su password è completamente controllato dal server.

Problemi con meccanismi dichiarativi

- La soluzione non è sempre portabile
- I server devono supportare *qualche* metodo per la definizione di utenti, password e ruoli, ma server diversi lo fanno in modo diverso.
- E' necessario modificare il file web.xml

Autenticazione programmata

- Le risorse protette (servlets e pagine JSP) sono responsabili della gestione della propria sicurezza.
  - Protabilità del codice. Non ci sono elementi della web application che dipendono dal particolare server utilizzato. Non sono necessarie ulteriori specifiche nel descrittore.
- Per prevenire accessi non autorizzati
  - Ciascuna servlet o pagina JSP deve autenticare l'utente o verificare che sia già stato autenticato.
- Per preservare la sicurezza dei dati sulla rete
  - Ogni servlet o pagina JSP deve controllare il protocollo usato
  - Se gli utenti provano ad usare una connessione HTTP la servlet o la pagina JSP devono redirigere le richieste sul protocollo HTTPS.

Approccio combinato: dichiarativo + programmatico

- Si fa affidamento sul server per la gestione di nomi, password e ruoli attraverso metodi dichiarativi
  - Autenticazione BASIC, basata su form oppure DIGEST
- Si gestisce l'accesso alle risorse in modo esplicito dalle servlet o dalle pagine JSP
  - Esempio: si può cambiare il contenuto della pagina secondo l'identità del richiedente (impossibile con metodi puramente dichiarativi).

Approccio combinato: dichiarativo + programmatico (cont.)

- Si utilizzano i seguenti metodi

  `HttpServletRequest`

  - **`boolean isUserInRole(String)`**
  - **`String getRemoteUser()`**
  - **`Principal getUserPrincipal()`**

  per definire il contenuto della risorsa protetta sulla base dell'identità del richiedente.

Approccio combinato: /WEB-INF/web.xml (cont.)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
. . .
<web-app>
    <display-name>Web App x test meccanismi di sicurezza </display-name>
    <description> Test dei meccanismi di sicurezza </description>

    <security-constraint>
    <web-resource-collection> <!-- risorse protette -->
        <web-resource-name>Una pagina protetta </web-resource-name>
        <url-pattern>/jsps/pagina-protetta.jsp</url-pattern>
    </web-resource-collection>
    <auth-constraint><!-- ruoli associati alle risorse -->
        <role-name>pagante</role-name>
        <role-name>nonpagante</role-name>
    </auth-constraint>
    </security-constraint>
</web-app>
```

Approccio combinato: /WEB-INF/pagina-protetta.jsp (cont.)

- Notare che il file web.xml consente l'accesso alla risorsa protetta sia agli utenti paganti che a quelli non paganti

```
<html><head><title>Una pagina protetta </title></head>
<body>
<%  if (request.isUserInRole("pagante")){ %>
        Informazioni riservate agli utenti abbonati
<%      } else if (request.isUserInRole("nonpagante")) {  %>
        Informazioni per chi non si è ancora abbonato
<%      }  %>
</body></html>
```

Approccio combinato: /WEB-INF/pagina-protetta.jsp (cont.)

- Oppure se gli utenti non paganti cercano di accedere ad una risorsa riservata agli abbonati si può decidere di deviare la richiesta verso una pagina di registrazione (o di login):

```
<html><head><title>Una pagina protetta </title></head>
<body>
<%  if (request.isUserInRole("pagante")){ %>
        Informazioni riservate agli utenti abbonati
<%      } else if (request.isUserInRole("nonpagante")) {
        response.sendRedirect("PaginaDiLogin.jsp");
        }  %>
</body></html>
```