CORSO DI LAUREA IN INFORMATICA

Tecnologie Software per il Web

SECURITY

a.a. 2023-2024

Introduzione

I due aspetti di sicurezza principali nelle applicazioni web sono:

- **Impedire agli utenti non autorizzati di accedere a dati sensibili**
  Questo processo prevede la **restrizione degli accesso** (identificando quali risorse necessitano di protezione e chi dovrebbe accedervi) e l'**autenticazione** (identificando gli utenti per determinare se sono uno di quelli autorizzati)
- **Impedire agli attaccanti di rubare dati di rete mentre questi sono in transito**
  Questo processo prevede l'uso di TLS (Transport Layer Security) o SSL (Secure Sockets Layer) per crittografare il traffico tra il browser e il server
  - HTTPS: HTTP su una connessione crittografata SSL/TLS
  - SSL è uno standard superato da TLS

Access restriction

- The approaches to access restriction are the same regardless of whether or not you use SSL. They are:
- **Declarative security.** With declarative security none of the individual servlets or JSP pages need any security-aware code. Instead, both of the major security aspects are handled by the server (use web.xml)
- **Programmatic security.** With programmatic security, protected servlets and JSP pages at least partially manage their own security:
  - To prevent unauthorized access, each servlet or JSP page must either authenticate the user or verify that the user has been authenticated previously
  - *To safeguard network data, each servlet or JSP page has to check the network protocol used to access it (request.isSecure()). If users try to use a regular HTTP connection to access one of these URLs, the servlet or JSP page must manually redirect them to the HTTPS (SSL) equivalent*

Restrizione degli accessi: linee guida

1. Identificare quali Servlet e pagine JSP necessitano di protezione
2. Identificare chi può accedere ad un gruppo di Servlet e pagine JSP
   - Generalmente vi sono Servlet e pagine JSP riservate al solo amministratore, poi vi sono Servlet e pagine JSP alle quali vi può accedere l'utente ma anche l'amministratore (generalmente l'amministratore a accesso alle stesse funzionalità dell'utente, più altre a lui dedicate)
3. Le pagine JSP riservate all'amministratore vanno nella cartella "admin", mentre per le Servlet, l'URL pattern avrà la seguente forma "/admin/myServlet"
4. Le pagine JSP a cui può accedere sia l'utente che l'amministratore vanno nella cartella "common", mentre per le Servlet, l'URL pattern avrà la seguente forma "/common/myServlet"

Restrizione degli accessi: linee guida

- Fare attenzione agli URL relativi
  - Se la mia JSP si trova nella cartella “admin”, gli URL relativi nella pagina fanno riferimento a questa cartella
  - Discorso simile per le Servlet
- Esempio:
  - Se in “admin/protected.jsp” ho la chiamata **response.sendRedirect("/login.jsp")**, l’URL risolto è “myWebApp/admin/login.jsp” e non “myWebApp/login.jsp”
  - Posso risolvere nel seguente modo: **response.sendRedirect(request.getContextPath() + "/login.jsp");**

SESSION AND TOKEN

MANUAL APPROACH

Example 1: Token in session (page login-form.jsp)

...

```
<form action="Login" method="post">
<fieldset>
     <legend>Login</legend>
     <label for="username">Login</label>
     <input id="username" type="text" name="username" placeholder="enter login">
     <br>
     <label for="password">Password</label>
     <input id="password" type="password" name="password" placeholder="enter password">
     <br>
     <input type="submit" value="Login"/>
     <input type="reset" value="Reset"/>
</fieldset>
</form>
```

Token in session (servlet Login)

```
@WebServlet("/Login")
public class Login extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        {
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            String redirectedPage;
            try {
                checkLogin(username, password);
                request.getSession().setAttribute("adminRoles", new Boolean(true));
                redirectedPage = "/protected.jsp";
            } catch (Exception e) {
                request.getSession().setAttribute("adminRoles", new Boolean(false));
                redirectedPage = "/login-form.jsp";
            }
            response.sendRedirect(request.getContextPath() + redirectedPage);
        }
    }

    private void checkLogin(String username, String password) throws Exception {
        if ("root".equals(username) && "admin".equals(password)) {
            //
        } else
            throw new Exception("Invalid login and password");
    }
```

Si può accedere anche al DB

Token in session (page protected.jsp)

```
<%
// Check user credentials
Boolean adminRoles = (Boolean) session.getAttribute("adminRoles");
if ((adminRoles == null) || (!adminRoles.booleanValue()))
{
    response.sendRedirect("./login-form.jsp");
    return;
}
%>

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Protected Page</title>
</head>
<body>
<h1>Welcome to the Protected Page</h1>
Congratulations. You have accessed a protected document.
<br><br>
<form action="Logout" method="get" >
     <input type="submit" value="Logout"/>
</form>
</body>
</html>
```

Token in session (servlet Logout)

```
@WebServlet("/Logout")
public class Logout extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        request.getSession().removeAttribute("adminRoles");
        request.getSession().invalidate();

        String redirectedPage = "/login-form.jsp";
        response.sendRedirect(request.getContextPath() + redirectedPage);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
```

Si può fare anche il redirect alla Home page

- ***Filter can be used to control the access to resources***

Token nella session: verso una soluzione migliore

- Nell soluzione vista sinora, in ogni Servlet o pagina JSP protetta, devo assicurarmi che l'utente sia autenticato (ovvero il "token" è nella sessione) e che abbia il diritto di accedete a quella Servlet o pagina JSP
  - Svantaggio: stessa porzione di codice da ripetere in più Servlet o pagine JSP
  - Soluzione: uso di un filtro

Filtri

- Un filtro viene in genere utilizzato per eseguire una particolare funzionalità prima o dopo l'esecuzione della funzionalità principale di un'applicazione Web.
- Ad esempio, se viene effettuata una richiesta per una risorsa particolare come una servlet e viene utilizzato un filtro, il codice del filtro può essere eseguito e quindi passare l'utente alla servlet.
- I filtro potrebbe determinare che l'utente non dispone delle autorizzazioni per accedere a una particolare servlet e potrebbe inviare l'utente a una pagina di errore anziché alla risorsa richiesta.

Filtri (2)

- I filtri servlet sono configurati nel file descrittore di distribuzione (web.xml).
- Servlet e filtri non sono consapevoli l'uno dell'altro e possiamo aggiungere o rimuovere un filtro servlet semplicemente modificando web.xml.
- Possiamo avere più filtri per una singola risorsa e possiamo creare una catena di filtri per una singola risorsa in web.xml.
- Possiamo creare un filtro servlet implementando l'interfaccia javax.servlet.Filter.

login-form-filter.html

ProductView.jsp | ProductCont... | ProductStyle... | Login.java | login-form.jsp | web.xml | ProtectPage.... | login-form-f... | LoginFilter.... | AuthFilter.java | loginerror.html | pro

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2  <!DOCTYPE html>
3  <html>
4  <head>
5  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6  <title>Login form</title>
7  </head>
8  <body>
9
10 <form action="LoginFilter" method="post">
11 <fieldset>
12       <legend>Login Filter Custom</legend>
13       <label for="username">Login</label>
14       <input id="username" type="text" name="username" placeholder="enter login">
15       <br>
16       <label for="password">Password</label>
17       <input id="password" type="password" name="password" placeholder="enter password">
18       <br>
19       <input type="submit" value="Login"/>
20       <input type="reset" value="Reset"/>
21 </fieldset>
22 </form>
23
24 </body>
25 </html>
26
```

loginfilter.java

```java
import java.io.IOException;

@WebServlet("/LoginFilter")
public class LoginFilter extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        {
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            String redirectedPage;
            try {
                checkLogin(username, password);
                request.getSession().setAttribute("adminFilterRoles", true);
                redirectedPage = "/baseFilter.html";
                RequestDispatcher dispatcher = getServletContext().getRequestDispatcher(redirectedPage);
                dispatcher.forward(request, response);
            } catch (Exception e) {
                request.getSession().removeAttribute("adminFilterRoles");
                redirectedPage = "/login-form-filter.jsp";
                response.sendRedirect(request.getContextPath() + redirectedPage);
            }

        }
    }
```

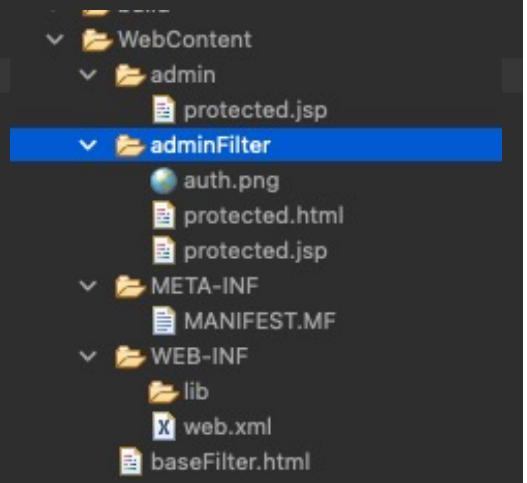
loginfilter.java (2)

```
    private void checkLogin(String username, String password) throws Exception {
        if ("root".equals(username) && "admin".equals(password)) {
            //
        } else
            throw new Exception("Invalid login and password");
    }

    private static final long serialVersionUID = 1L;

    public LoginFilter() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doPost(request, response);
    }


}
```

Filtro

```java
import java.io.IOException;

public class AuthFilter implements Filter {

    public void destroy() {
    }

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOEx
        HttpServletRequest hrequest = (HttpServletRequest) request;
        HttpServletResponse hresponse = (HttpServletResponse) response;

        String loginURI = hrequest.getContextPath() + "/adminFilter";
        boolean loginRequest = hrequest.getRequestURI().startsWith(loginURI);

        if(loginRequest) {
            System.out.println("Check role in the session");
            //check the token from session
            HttpSession session = hrequest.getSession(false);
            boolean loggedIn = session != null && session.getAttribute("adminFilterRoles") != null;

            if(!loggedIn) {
                System.out.println("Redirect to login form");
                hresponse.sendRedirect(hrequest.getContextPath()+ "/login-form-filter.jsp");
            } else {
                // admin resource
                chain.doFilter(request, response);
            }
        } else {
            // accessible resource
            chain.doFilter(request, response);
        }
    }

    public void init(FilterConfig fConfig) throws ServletException {
        System.out.println("Init the filter");
    }

}
```

Controlla se la pagina richiesta è nella cartella/adminFilter



web.xml

```
<filter>
    <filter-name>AuthFilter</filter-name>
    <filter-class>AuthFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>AuthFilter</filter-name>
    <url-pattern>/adminFilter/*</url-pattern>
</filter-mapping>
```

/baseFilter.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Login Page</title>
</head>
<body>

Protected image: <br>
<img src="./adminFilter/auth.png" onerror="this.src='noImage.png'" style="width:100px;"><br>
<br>

<a href="./adminFilter/protected.jsp">Access to adminFilter jsp resource</a><br>
<a href="./adminFilter/protected.html">Access to adminFilter html resource</a>

</body>
</html>
```

login-form-filter.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login form</title>
</head>
<body>

<form action="LoginFilter" method="post">
<fieldset>
      <legend>Login Filter Custom</legend>
      <label for="username">Login</label>
      <input id="username" type="text" name="username" placeholder="enter login">
      <br>
      <label for="password">Password</label>
      <input id="password" type="password" name="password" placeholder="enter password">
      <br>
      <input type="submit" value="Login"/>
      <input type="reset" value="Reset"/>
</fieldset>
</form>

</body>
</html>
```

Memorizzazione delle password

- Le password nel database non devono essere memorizzate in chiaro
- Anziché memorizzare la password nel database, va memorizzato l'hash della password
  - Hash: funzione non invertibile che mappa una stringa di lunghezza arbitraria in una stringa di lunghezza predefinita
- Quando si fa il login, occorre calcolare l'hash della password fornita dall'utente e poi confrontarla con l'hash della password memorizzato nel database

```
private String toHash(String password) {
    String hashString = null;
    try {
        java.security.MessageDigest digest = java.security.MessageDigest.getInstance("SHA-512");
        byte[] hash = digest.digest(password.getBytes(StandardCharsets.UTF_8));
        hashString = "";
        for (int i = 0; i < hash.length; i++) {
            hashString += Integer.toHexString(
                                (hash[i] & 0xFF) | 0x100
                            ).toLowerCase().substring(1,3);
        }
    } catch (java.security.NoSuchAlgorithmException e) {
        System.out.println(e);
    }
    return hashString;
}
```

L'algoritmo di hashing usato è lo SHA-512, più sicuro di SHA-1 e SHA-256

Esempio: l'hash della password "mypass" è:

"1c573dfeb388b562b55948af954a7b344dde1cc2099e978a992790429e7c01a4205506a93d9aef3bab32d6f06d75b7777a7ad8859e672fedb6a096ae369254d2"

Esempio

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        String username = request.getParameter("username");
        String password = request.getParameter("password");
        List<String> errors = new ArrayList<>();
        RequestDispatcher dispatcherToLoginPage = request.getRequestDispatcher("login.jsp");


        if(username == null || username.trim().isEmpty()) {
            errors.add("Il campo username non può essere vuoto!");
        }
        if(password == null || password.trim().isEmpty()) {
            errors.add("Il campo password non può essere vuoto!");
        }
        if (!errors.isEmpty()) {
            request.setAttribute("errors", errors);
            dispatcherToLoginPage.forward(request, response);
            return; // note the return statement here!!!
        }

        String hashPassword = toHash(password);
        String hashPasswordToBeMatch =
                "1c573dfeb388b562b55948af954a7b344dde1cc2099e978a992790429e7c01a4205506a93d9aef3bab32d6f06d75b7777a7ad8859e672fedb6a096ae369254d2";
        // this is the hash of "mypass"

        if(username.equals("admin") && hashPassword.equals(hashPasswordToBeMatch)){ //admin
            request.getSession().setAttribute("isAdmin", Boolean.TRUE); //inserisco il token nella sessione
            response.sendRedirect("admin/protected.jsp");
        } else if (username.equals("user") && hashPassword.equals(hashPasswordToBeMatch)){ //user
            request.getSession().setAttribute("isAdmin", Boolean.FALSE); //inserisco il token nella sessione
            response.sendRedirect("common/protected.jsp");
        } else {
            errors.add("Username o password non validi!");
            request.setAttribute("errors", errors);
            dispatcherToLoginPage.forward(request, response);
        }
}
```

Qui in realtà si dovrebbero caricare le credenziali dal database (ovvero nome utente e hash della password)