Programmatic Security

Originais of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/msajsp.html

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at your location.

For live Java training, please see training courses at http://courses.coreservlets.com/. Servlets, JSP, Struts, JSF, Ajax, GWT, Java 5, Java 6, Spring, Hibernate, JPA, and customized combinations of topics.

Taught by the author of Core Servlets and JSP, More Servlets and JSP, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization. Contact hall@coreservlets.com for details.
Agenda

- Problems with declarative security
  - The advantages of declarative security usually outweigh the disadvantages. But not always.
- Combination security: mixing server-managed and servlet-managed (programmatic) security
  - Solve one of the drawbacks of declarative security with only a little bit of extra work.
- Pure programmatic security
  - Solve the other drawbacks, but with a very lot of extra work.

Problems with Pure Declarative Security

- Access is all-or-nothing
  - Users can access a resource or be denied access to it.
  - No changes depending on who accesses resource.
- Access based on exact password matches
  - Controlled by server.
- Involves server-specific component
  - Thus is not completely portable
    - Servers must support some way to designate users, passwords, and roles. But different servers do it different ways.
- All pages use same mechanism
  - Can't mix form-based and BASIC in same Web app
- Requires web.xml entries
  - This might matter when deploying in preset Web app
Combining Container-Managed and Programmatic Security

- Rely on the container (server) for usernames, passwords, and roles
  - Form-based or BASIC authentication as before
- Manage access explicitly from within the servlets or JSP pages
  - For example, you can change the result of a particular page depending on who accesses it. With pure declarative security, it is all or nothing.
- Use the following HttpServletRequest methods
  - isUserInRole
  - getRemoteUser
  - getUserPrincipal

Combination Security: Example

- Following steps shared with the hotdotcom Intranet example (from BASIC authentication)
  - Definition of usernames, passwords, and roles
  - web.xml entry to designate BASIC authentication
    `<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Intranet</realm-name>
    </login-config>`
  - web.xml entries to designate protected resources
    - security-constraint element with url-pattern
    - auth-constraint element with role-name
- Server still tracks users and sends dialog box if they are unauthenticated
Combination Security: Example (web.xml Excerpt)

```xml
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      Compensation Plan
    </web-resource-name>
    <url-pattern>
      /employee-pay.jsp
    </url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>employee</role-name>
    <role-name>executive</role-name>
  </auth-constraint>
</security-constraint>
```

Combination Security: Example (Continued)

- employee-pay.jsp
  - web.xml limits access to employees or executives

  <H3>Regular Employees</H3>
  Pay for median-level employee (Master's degree, eight year's experience):
  <UL>
    <LI><B>2007:</B> $50,000.</LI>
    <LI><B>2008:</B> $30,000.</LI>
    <LI><B>2009:</B> $25,000.</LI>
    <LI><B>2010:</B> $20,000.</LI>
  </UL>
Combination Security: Example (Continued)

- **employee-pay.jsp continued**
  - Specific code further limits access to executives

```jsp
<% if (request.isUserInRole("executive")) { %>
  <H3>Executives</H3>
  Median pay for corporate executives:
  <UL>
  <LI><B>2007:</B> $600,000.
  <LI><B>2008:</B> $700,000.
  <LI><B>2009:</B> $800,000.
  <LI><B>2010:</B> $900,000.
  </UL>
<% } %>
```

Combination Security: Example (Results)

- **Access by regular employee**

---

Due to temporary financial difficulties, we are scaling back our very generous plans for salary increases. Don’t worry, though: your valuable stock options more than compensate for any small drops in direct salary.

**Regular Employees**

Pay for medium-level employee (Master’s degree, eight year’s experience):
- 2007: $50,000.
- 2008: $30,000.
- 2009: $25,000.
- 2010: $20,000.
Combination Security: Example (Results)

- Access by executive

Pure Programmatic Security

- Idea
  - Each protected resource authenticates users and decides what (if any) access to grant

- Advantages
  - Totally portable
    - No server-specific component
  - Permits custom password-matching strategies
  - No web.xml entries needed (except maybe url-pattern)

- Disadvantages
  - Much harder to write and maintain
  - Each and every resource has to use the code
    - You can build reusable infrastructure (e.g., servlets that inherit from certain classes or custom JSP tags), but it is still a lot of work.
Pure Programmatic Security

1. Check whether there is an Authorization request header.
   – If there is no such header, go to Step 5.

2. Get the encoded username/password string.
   – If there is an Authorization header, it should have the following form:
     • Authorization: Basic encodedData
   – Skip over the word Basic and the space—the remaining part is the username and password represented in base64 encoding.

3. Reverse the base64 encoding of the username/password string.
   – Use the decodeBuffer method of the BASE64Decoder class. This method call results in a string of the form username:password. The BASE64Decoder class is bundled with the JDK; in JDK 1.3+ it can be found in the sun.misc package in jdk_install_dir/jre/lib/rt.jar.

Pure Programmatic Security (Continued)

4. Check the username and password.
   – The most common approach is to use a database or a file to obtain the real usernames and passwords. For simple cases, it is also possible to place the password information directly in the servlet. If the incoming username and password match one of the reference username/password pairs, return the page. If not, go to Step 5. With this approach you can provide your own definition of “match.” With container-managed security, you cannot.

5. When authentication fails, send the appropriate response to the client.
   – Return a 401 (Unauthorized) response code and a header of the following form:
     WWW-Authenticate: BASIC realm="some-name"
   – This response instructs the browser to pop up a dialog box telling the user to enter a name and password for some-name, then to reconnect with that username and password embedded in a single base64 string inside the Authorization header.
Example: A Servlet That Generates Stock Tips

```java
class ExampleServlet {
    public void doGet(...) {
        String authorization = request.getHeader("Authorization");
        if (authorization == null) {
            askForPassword(response);
        } else {
            String userInfo = authorization.substring(6).trim();
            BASE64Decoder decoder = new BASE64Decoder();
            String nameAndPassword =
                new String(decoder.decodeBuffer(userInfo));
            int index = nameAndPassword.indexOf("":"");
            String user = nameAndPassword.substring(0, index);
            String password = nameAndPassword.substring(index+1);
            if (areEqualReversed(user, password)) {
                showStock(request, response);
            } else {
                askForPassword(response);
            }
        }
    }

    private void askForPassword (HttpServletResponse response) {
        // SC_UNAUTHORIZED is 401
        response.setStatus(response.SC_UNAUTHORIZED);
        response.setHeader("WWW-Authenticate", "BASIC realm="Insider-Trading");
    }
}
```
// Returns true if s1 is the reverse of s2.
// Empty strings don't count.

private boolean areEqualReversed(String s1, String s2) {
    s2 = (new StringBuffer(s2)).reverse().toString();
    return((s1.length() > 0) && s1.equals(s2));
}
Using Programmatic Security with SSL

- **Determining If SSL Is in Use**
  - request.getScheme (returns "https" or "http")
  - request.isSecure (returns true or false)
- **Redirecting Non-SSL Requests**
  - response.sendRedirect
- **Discovering the Number of Bits in the Key**
  - request.getAttribute("javax.servlet.request.key_size")
- **Looking Up the Encryption Algorithm**
  - request.getAttribute("javax.servlet.request.cipher_suite")
- **Accessing Client X509 Certificates**
  - request.getAttribute("javax.servlet.request.X509Certificate")

Example: Programmatic Security and SSL

```java
/** Servlet that prints information on SSL requests. * Non-SSL requests get redirected to SSL. */
public class SecurityInfo extends HttpServlet {
  public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
      throws ServletException, IOException {
    // Redirect non-SSL requests to the SSL equivalent.
    if (request.getScheme().equalsIgnoreCase("http")) {
      String origURL =
          request.getRequestURL().toString();
      String newURL = httpsURL(origURL);
      String formData = request.getQueryString();
      if (formData != null) {
        newURL = newURL + "?" + formData;
      }
      response.sendRedirect(newURL);
    ...
```

... boolean isSecure = request.isSecure();
if (isSecure) {
    String keyAttribute = "javax.servlet.request.key_size";
    // Available only with servlets 2.3
    Integer keySize = (Integer)request.getAttribute(keyAttribute);
    String sizeString = replaceNull(keySize, "Unknown");
    String cipherAttribute = "javax.servlet.request.cipher_suite";
    // Available only with servlets 2.3
    String cipherSuite = (String)request.getAttribute(cipherAttribute);
...
Summary

- **Combination security obviates "all or nothing" restriction**
  - Use isUserInRole or getRemoteUser to change content depending on who accesses resource
  - Still rely on server for authentication
- **Pure programmatic security**
  - Check for Authorization header
  - If missing, send 401 status code
  - If there, reverse base64 encoding and check password
  - Much more work, but gets around following restrictions:
    - Server-specific component
    - Exact password match
    - web.xml entries