Declarative Web Application Security

Prepared By : Vaishali Bharvada

Topics

- **Major security concerns**
- **Declarative vs. programmatic security**
- **Using form-based authentication**
- – Steps
- – Example
- **Using BASIC authentication**
- – Steps
- – Example

Major Issues

- **Preventing unauthorized users from accessing sensitive data.**
- – Access restriction
- • Identifying which resources need protection
- • Identifying who should have access to them
- – Authentication
- • Identifying users to determine if they are one of the authorized ones
- • **Preventing attackers from stealing network**
- **data while it is in transit.**
- – Encryption (usually with SSL)

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.
- • **To prevent unauthorized access**
- – Use the Web application deployment descriptor (*web.xml) to declare* that certain URLs need protection.
- – Designate authentication method that server uses to identify users.
- – At request time, the server automatically prompts users for
- usernames and passwords when they try to access restricted
- resources, automatically checks the results against a server-specific
- set of usernames and passwords, and automatically keeps track of
- which users have previously been authenticated. This process is
- completely transparent to the servlets and JSP pages.
- • **To safeguard network data**
- – Use the deployment descriptor to stipulate that certain URLs should
- be accessible only with SSL. If users try to use a regular HTTP
- connection to access one of these URLs, the server automatically
- redirects them to the HTTPS (SSL) equivalent.

Programmatic Security

- **Protected servlets and JSP pages at least partially manage their own security.**
- – *Much more work, but totally portable.*
- • No server-specific piece. Also no web.xml entries needed and a bit more flexibility is possible.
- **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.
- – 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.

Form-Based Authentication

- **When a not-yet-authenticated user tries to access a protected resource:**
- – Server automatically redirects user to Web page with an
- HTML form that asks for username and password
- – Username and password checked against database of
- usernames, passwords, and roles (user categories)
- – If login successful and role matches, page shown
- – If login unsuccesful, error page shown
- – If login successful but role does not match, 403 error
- given (but you can use error-page and error-code)
- **When an already authenticated user tries to**
- **access a protected resource:**
- – If role matches, page shown
- – If role does not match, 403 error given
- – Session tracking used to tell if user already authenticated

BASIC Authentication

- **When a not-yet-authenticated user tries to**
- **access a protected resource:**
- – Server sends a 401 status code to browser
- – Browser pops up dialog box asking for username and
- password, and they are sent with request in Authorization
- request header
- – Username and password checked against database of
- usernames, passwords, and roles (user categories)
- – If login successful and role matches, page shown
- – If login unsuccesful or role does not match, 401 again
- • **When an already authenticated user tries to**
- **access a protected resource:**
- – If role matches, page shown
- – If role does not match, 401 error given
- – Request header used to tell if user already authenticated

Form-Based Authentication (Declarative Security)

- **1) Set up usernames, passwords, and roles.**
- – Designate a list of users and associated passwords and
- abstract role(s) such as normal user or administrator.
- – This is a completely server-specific process.
- – Simplest Tomcat approach: use
- ***install_dir/conf/tomcat-users.xml:***

- *<tomcat-users>*
- *<user username="abc" password="abc"roles="registered-user" />*
- *<user username="admin" password="admin"roles="administrator" />*
- *<user username="vmb" password="vmb"roles="administrator,registered-user" />*
- *<role rolename="manager-gui"/><user username="tomcat" password="tomcat" roles="manager-gui"/>*
- *</tomcat-users>*

Continue..

- 2) Tell server that you are using form-based authentication. Designate locations of login and login-failure page.
- - Use the *web.xml login-config element with authmethod* of FORM and form-login-config with
- locations of pages.
- <web-app> …
- <login-config>
- <auth-method>FORM</auth-method>
- <form-login-config>
- <form-login-page>/login.jsp</form-login-page>
- <form-error-page>/login-error.html</form-error-page>
- </form-login-config>
- </login-config>
- … </web-app>

Continue…

- 3) Create a login page (HTML or JSP)
- – HTML form with ACTION of j_security_check,
- METHOD of POST, textfield named j_username, and
- password field named j_password.
- <FORM ACTION="j_security_check" METHOD="POST">
- …
- <INPUT TYPE="TEXT" NAME="j_username">
- …
- <INPUT TYPE="PASSWORD" NAME="j_password">
- …
- </FORM>
- – For the username, you can use a list box, combo box, or set of radio buttons instead of a textfield.

Continue….

- 4) Create page for failed login attempts.
- – No specific content is mandated.
- – Perhaps just “username and password not found” and give a link back to the login page.
- – This can be either an HTML or a JSP document.

Continue..

- 5) Specify URLs to be password protected.
- – Use security-constraint element of *web.xml. Two*
- subelements: the first (web-resource-collection)
- designates URLs to which access should be restricted; the second
- (auth-constraint) specifies abstract roles that should have
- access to the given URLs. Using auth-constraint with no
- role-name means no *direct access is allowed.*
- <web-app ...>...
- <security-constraint>
- <web-resource-collection>
- <web-resource-name>Sensitive</web-resource-name>
- <url-pattern>/sensitive/*</url-pattern>
- </web-resource-collection>
- <auth-constraint>
- <role-name>administrator</role-name>
- <role-name>executive</role-name>
- </auth-constraint>
- </security-constraint>
- <login-config>...</login-config>...
- </web-app>

Continue….

- 6) List all possible abstract roles (categories
- of users) that will be granted access to *any*
- resource
- – Many servers do not enforce this, but technically required
- <web-app …>
- …
- <security-role>
- <role-name>administrator</role-name>
- </security-role>
- <security-role>
- <role-name>executive</role-name>
- </security-role>
- </web-app>

Continue…

- **7) Specify which URLs require SSL.**
- – If server supports SSL, you can stipulate that certain resources are available only through encrypted HTTPS (SSL) connections.
- Use the user-data-constraint subelement of security-constraint. Only full J2EE servers are *required to support SSL.*
  - <security-constraint>
  - …
  - <user-data-constraint>
  - <transport-guarantee>
  - CONFIDENTIAL
  - </transport-guarantee>
  - </user-data-constraint>
  - </security-constraint>

Continue..

- **8) Turn off the invoker servlet.**
- – You protect certain URLs that are associated with registered servlet or JSP names.
- The *http://host/prefix/servlet/Name format of default servlet* URLs will probably not match the pattern. Thus, the security restrictions are bypassed when the default URLs are used.
- – Disabling it
- • In each Web application, redirect requests to other servlet by normal *web.xml method*
- **<url-pattern>/servlet/*</url-pattern>**
- • Globally
- – Server-specific mechanism (e.g. *install_dir/conf/server.xml for Tomcat).*

Example

- Example: Access Rules
- **Student Page**
- – Anyone
- **Faculty page**
- –Faculty
- – Administrators
- **Admin page**
- –administrator

Example

- Example: Access Rules
- **Home page**
- – Anyone
- **Investing page**
- – Registered users
- – Administrators
- **Stock purchase page**
- – Registered users
- – Via SSL only
- **Delete account page**
- – Administrators

Form-Based vs. BASIC Authentication

- **Advantages of form-based**
- - Consistent look and feel
- - Fits model users expect from ecommerce sites
- **Disadvantage of form-based**
- - Can fail if server is using URL rewriting for session tracking. Can fail if browser has cookies disabled.
- **Advantages of BASIC**
- - Doesn't rely on session tracking
- - Easier when you are doing it yourself (programmatic)
- **Disadvantage of BASIC**
- - Small popup dialog box seems less familiar to most users
- **Other auth-method options**
- - CLIENT-CERT (X 509 certificates)
- - DIGEST (Not widely supported by browsers)

BASIC Authentication

- **1. Set up usernames, passwords, and roles.**
- – Same as for form-based authentication. Server-specific.
- **2. Tell the server that you are using BASIC**
- **authentication. Designate the realm name.**
- – Use the *web.xml login-config element with an*
- auth-method subelement of BASIC and a realmname subelement (generally used as part of the title of the dialog box that the browser opens).
- **<login-config>**
- **<auth-method>BASIC</auth-method>**
- **<realm-name>Some Name</realm-name>**
- **</login-config>**

Continue..

- **3. Specify which URLs should be password**
- **protected.**
- – Same as with form-based authentication.
- **4. List all possible roles (categories of users)**
- **that will access any protected resource**
- – Same as with form-based authentication
- **5. Specify which URLs should be available**
- **only with SSL.**
- – Same as with form-based authentication.
- **6. Turn off the invoker servlet.**
- – Same as with form-based authentication.

Example

- Example: Access Rules
- **Student Page**
- – Anyone
- **Faculty page**
- –Faculty
- – Administrators
- **Admin page**
- –administrator

Example : Basic Authentication

- **Home page**
- – Anyone
- **Financial plan**
- – Employees or executives
- **Business plan**
- – Executives only

Summary

- **Main security issues**
- – Preventing access by unauthorized users
- – Preventing attackers from stealing network data
- Declarative security
- – Much less work than programmatic security
- – Requires server-specific password setup
  - Form-based authentication
- – Attempts to access restricted resources get redirected to login page. HTML form gathers username and password.
- Session tracking tracks authenticated users.
  - BASIC authentication
- – Attempts to access restricted resources results in dialog box. Dialog gathers username and password. HTTP headers track authenticated users.

- THANK YOU