Need for Security in Web Applications

- Potentially large number of users
- Multiple user types
- No operating system to rely on

Web Application Security

- **Client**
  - Request
  - Authentication
    - Who are you?
      - Username/password
    - You're not authorized to access

- **Server**

Connection Security

- **Secure Socket Layer (SSL)**
  - Server authentication
  - Client authentication
  - Connection encryption
- **Transport Layer Security (TLS)**
  - TLS 1.0 is based on SSL 3.0
  - IETF standard (RFC 2246)

HTTPS

- HTTP over SSL

Programmatic Security

- Security is implemented in the application code
- Example:
  - Login.jsp
  - Members.jsp

Pros?? Cons??
Security by Java EE Application Server

- HTTP Basic
- HTTP Digest
- HTTPS Client
- Form-based

HTTP Basic

- HTTP 1.0, Section 11.1 - http://www.w3.org/Protocols/HTTP/1.0/draft-ietf-http-spec.html
- Request for a restricted page
- Prompt for username/password
- Send request + username & password

HTTP Basic – Configuration

- AuthType Basic
- AuthName "Basic Authentication Example"
- AuthUserFile /home/cysun/etc/htpasswords
- Require user cs520

HTTP Basic – Request

GET /restricted/index.html HTTP/1.0
Host: sun.calstatela.edu
Accept: */*
Authorization: Basic Y3lzdW46YWJjZAo=

HTTP Basic – Server Response

HTTP/1.1 401 Authorization Required
Date: Tue, 24 Oct 2006 14:57:50 GMT
Server: Apache/2.2.2 (Fedora)
WWW-Authenticate: Basic realm="Restricted Access Area"
Content-Length: 484
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
<head><title>401 Authorization Required</title>
... ... ...
</html>

HTTP Basic – Request Again

GET /restricted/index.html HTTP/1.0
Host: sun.calstatela.edu
Accept: */*
Authorization: Basic Y3lzdW46YWJjZAo=

An online Base64 decoder is at http://www.opinionatedgeek.com/dotnet/tools/Base64Decode/
Improve HTTP Basic (I)

HTTP Basic
- Username and password are sent in plain text.
- Encrypt username and password.

Cryptographic Hash Function...
- String of arbitrary length → n bits digest
- Properties
  1. Given a hash value, it's virtually impossible to find a message that hashes to this value
  2. Given a message, it's virtually impossible to find another message that hashes to the same value
  3. It's virtually impossible to find two messages that hash to the same value

  A.K.A.
  - One-way hashing, message digest, digital fingerprint

...Cryptographic Hash Function
- Common usage
  - Store passwords, software checksum ...
- Popular algorithms
  - MD5 (broken, partially)
  - SHA-1 (broken, sort of)
  - SHA-256 and SHA-512 (recommended)

Encrypting Password is Not Enough
- Why??

Improve HTTP Basic (II)

HTTP Basic
- Username and password are sent in plain text.
- Encrypt username and password.
- Additional measures to prevent common attacks.

HTTP Digest
- RFC 2617 (Part of HTTP 1.1) -
  http://www.ietf.org/rfc/rfc2617.txt
  - request for a restricted page
  - prompt for username/password + nonce
  - resend request + message digest
HTTP Digest – Server Response

HTTP/1.1 401 Authorization Required
Date: Tue, 24 Oct 2006 14:57:50 GMT
Server: Apache/2.2.2 (Fedora)
WWW-Authenticate: Digest realm="Restricted Access Area",
qop="auth,auth-int", nonce="dcd98b7102dd2f0e8b11d0f600bf0c093",
algorithm="MD5",
opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-Length: 484
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
<head><title>401 Authorization Required</title></head>
…
</html>

HTTP Digest – Request Again

GET /restricted/index.html HTTP/1.0
Host: sun.calstatela.edu
Accept: */*
Authorization: Digest username="cysun",
realm="Restricted Access Area",
nonce="dcd98b7102dd2f0e8b11d0f600bf0c093",
uri="/restricted/index.html", qop=auth,
nc=00000001, cnonce="0a4f113b",
opaque="5ccc069c403ebaf9f0171e9517f40e41",
algorithm="MD5"
response="6629fae49393a05397450978507c4ef1"

Hash value of the combination of of username, password, realm, uri, nonce, cnonce, nc, qop.

Form-based Security

- Unique to J2EE application servers
- Include authentication and authorization, but not connection security

Form-base Security using Tomcat

- `$TOMCAT/conf/tomcat-users.xml`
  - Users and roles
- `$APPLICATION/WEB-INF/web.xml`
  - Authentication type (FORM)
  - Login and login failure page
  - URLs to be protected

Example – Users and Roles

```xml
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="admin"/>
  <role rolename="member"/>
  <user username="admin" password="1234" roles="admin,member"/>
  <user username="cysun" password="abcd" roles="member"/>
</tomcat-users>
```

Example – Directory Layout

```
/admin
  /index.html
/member
  /index.html
login.html
logout.jsp
error.html
```

WEB-INF
```
Example – Login Page

```html
<form action="/j_security_check" method="post">
    <input type="text" name="j_username">
    <input type="password" name="j_password">
    <input type="submit" name="login" value="Login">
</form>
```

Example – web.xml ...

```xml
<login-config>
    <auth-method>FORM</auth-method>
    <form-login-config>
        <form-login-page>/login.html</form-login-page>
        <form-error-page>/error.html</form-error-page>
    </form-login-config>
</login-config>
```

... Example – web.xml

```xml
<security-constraint>
    <web-resource-collection>
        <web-resource-name>AdminArea</web-resource-name>
        <url-pattern>/admin/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>admin</role-name>
    </auth-constraint>
</security-constraint>
```

Declarative Security

- Security constraints are defined outside application code in some metadata file(s)
- Advantages
  - Application server provides the security implementation
  - Separate security code from normal code
  - Easy to use and maintain

Limitations of Declarative Security by App Servers

- Application server dependent
- Not flexible enough
- Servlet Specification only requires URL access control

Security Requirements of Web Applications

- Authentication
- Authorization (Access Control)
  - URL
  - Domain object
  - Method invocation
    - Access to service layer, e.g. DAO
    - Access to web services
Spring Security (SS)
- A security framework for Spring-based applications
- Addresses all the security requirements of web applications
- Formerly known as Acegi Security
  - ABCDEFGHI

How Does Spring Security Work
- Intercept request and/or response
  - Servlet filters
  - Spring handler interceptors
- Intercept method calls
  - Spring method interceptors

Servlet Filter
- Intercept, examine, and/or modify request and response
  - [Diagram: Filter diagram with request and response]

Servlet Filter Example
- web.xml
  - <filter> and <filter-mapping>
- Modify request
- Modify response

Spring Handler Interceptor
- Serve the same purpose as servlet filter
- Configured as Spring beans, i.e. support dependency injection
  - [Diagram: Handler Interceptor diagram with request and response]

Intercept Request/Response
- Request
  - [Diagram: Request flow with Controller and intercept question]
  - What can we do by intercepting the request??
- Response
  - [Diagram: Response flow with Controller and intercept question]
  - What can we do by intercepting the response??
Intercept Method Call

Method Invocation
User getUserById(1)

BeforeAdvice
What can we do in BeforeAdvice??

AfterAdvice
What can we do in AfterAdvice??

Main Components of Spring Security
- Authentication
- URL Security
- Method invocation security
- Object access security
- Security tag library

Add Spring Security to a Web Application

**web.xml**

```xml
<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>
    org.springframework.web.filter.DelegatingFilterProxy
  </filter-class>
</filter>

<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Authentication Manager

Authentication Sources
- database
- LDAP
- Servlet Container

Authentication Sources Supported
- Database
- LDAP
- JAAS
- CAS
- OpenID
- SiteMinder
- X.509
- Windows NTLM

- Container-based
  - Jboss
  - Jetty
  - Resin
  - Tomcat

Authenticate Against a Database – Configuration

In the `security` namespace:

```xml
<authentication-manager>
  <authentication-provider>
    <jdbc-user-service data-source-ref="dataSource" />
  </authentication-provider>
</authentication-manager>
```
Authenticate Against a Database – Default Schema

create table users (
    username string primary key,
    password string,
    enabled boolean
);
create table authorities (
    username string references users(username),
    authority string -- role name
);
Some `<http>` Customizations

- `<form-login>`
  - login-page
  - authentication-failure-url
  - default-target-url
- `<remember-me>`

Enable Method and Object Security

In the `security` namespace:

```xml
<global-method-security secured-annotations="enabled"/>
```

- Use an Access Decision Manager for method security
- Use one or more After Invocation Providers for object security

Access Decision Manager

```
Access Decision Manager
-----------
<table>
<thead>
<tr>
<th>Access Decision Voter</th>
<th>Access Decision Voter</th>
<th>Access Decision Voter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role Voter</td>
<td>Role Voter</td>
<td>User-defined Voter</td>
</tr>
</tbody>
</table>
```

Each voter returns `ACCESS_GRANTED`, or `ACCESS_DENIED`, or `ACCESS_ABSTAIN`

Types of Decision Managers

- Affirmative based
- Consensus based
- Unanimous based

How Access Decision Voter Work

- `supports()` – determines whether the voter should participate a vote based on
  - The class type of the object to be authorized
  - Some configuration attributes, e.g. `ROLE_ADMIN`, `PERM_COURSE_WRITE`
- `vote()` – casts a vote based on
  - Authentication information of the current user
  - The object to be authorized
  - Configuration attributes

Method Security Example in CSNS2

- `Secure` `CourseDao.saveCourse()` so that administrators can create and edit courses, while course coordinators can edit their own courses
  - `MethodAccessVoter.java`
  - `CourseWriteVoter.java`
  - `CourseDao.java`
  - `security.xml`
Object Security Using After Invocation Provider

- Very similar to Access Decision Voter
  - supports()
  - decide()

Object Security Example in CSNS2

- Secure
  - AssignmentDao.getAssignmentById() to allow only the instructors and the students in a section to access an assignment
  - ObjectAccessVoter.java
  - AssignmentReadVoter.java
  - AssignmentDao.java
  - security.xml

Security Tag Library

- [http://static.springsource.org/spring-security/site/docs/3.1.x/reference/taglibs.html](http://static.springsource.org/spring-security/site/docs/3.1.x/reference/taglibs.html)
- `<authorize>`
  - access
- `<authentication>`
  - property

Security Taglib Examples in CSNS2

- Hide menus from the users who are not authorized to access them
  - menu.jsp

Conclusion

- Declarative security vs. Programmatic security
- Spring Security provides the best of both worlds
  - Declarative security framework
  - Portability and flexibility
  - Separate security code from regular code