Need for Security in Web Applications

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

Web Application Security

Authentication
- who are you?
- username/password
- you're not authorized to access

Authorization (Access Control)

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
- Configure SSL in Tomcat 5.5 - http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html

Programmatic Security

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

Pros?? Cons??
Security by J2EE 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
  
  Client prompt for username/password
  
  resend request + username & password

HTTP Basic – Request

GET /restricted/index.html HTTP/1.0
Host: sun.calstatela.edu
Accept: */*

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></head>
…
</html>

HTTP Basic – Request Again

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

Base64 Encoding of "cysun:abcd"

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>
<body>
...</body>
</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

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 matches 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.
  1. One-way hashing, message digest, digital fingerprint

...Cryptographic Hash Function

- Common usage
  - Store passwords, software checksum ...
- Popular algorithms
  - MD5 (broken, sort of)
  - SHA-1 (expected to be broken soon)
  - SHA-256 and SHA-512 (recommended)

Form-based Security

- Unique to J2EE application servers
- Username/password are passed as clear text
- Login page instead of login prompt

Form-base Security using Tomcat

- STOMCAT/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 – Directory Layout

```
/admin
  /restricted
  index.htm
  login.jsp
  logout.jsp
  404.htm
  error.htm
  home.jsp
  secret.jsp
```

Example – Users and Roles

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

Example – web.xml ...

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

... Example – web.xml

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

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>
```

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
    - Access to web services

Acegi

- A security framework for Spring applications
- Addresses all the security requirements of web applications
- ABCDEFGHI

How Does Acegi Work

- Intercept request and/or response
  - Servlet filters
    - E.g. RequestEncodingFilter in Evelyn
  - Spring interceptors
  - Intercept method calls
    - Spring AOP support

Intercept Request/Response

Intercept Method Call

```
Request

Controller
/member/index.html

Response

BeforeAdvice
What can we do in BeforeAdvice??

Method Invocation
User getUserById(1)

AfterAdvice
What can we do in AfterAdvice??
```
Authentication Processing Filter

Login Form

Configure Authentication Filter Beans

Authentication Manager

Authenticate Against a Database ...

... Authenticate Against a Database
Access User Details in Application Code

- User details
  - Username
  - Password
  - Authorities (Roles)
- Example: `AcegiUtils.getUsername()` in CSNS

Access Decision Manager

Types of Decision Managers

- Affirmative based
- Consensus based
- Unanimous based

How Decision Voter Works

- `AccessDecisionVoter` Interface
- Given
  - Object to be accessed
  - User information: username, roles
  - Configuration attributes, typically are roles names and/or access types like READ, WRITE etc.
- Return
  - `ACCESS_GRANTED`, or `ACCESS_DENIED`, or `ACCESS_ABSTAIN`

Secure URL Access

- `FilterSecurityInterceptor`
- Example: CSNS
  - Using `RoleVoter`: mapping from URL patterns to roles

Secure Method Invocation

- `MethodSecurityInterceptor`
- Example: CSNS
  - Using `RoleVoter`: mapping from method name patterns to roles
Secure Object Access

- Implemented by checking the returned object of a method call
- Access decision is managed by AfterInvocationManager

Secure Object Access Example

- CSNS
  - MethodSecurityInterceptor
  - AfterInvocationManager
  - Customized AfterInvocation providers to provide application-specific access control
    - SectionAccessVoter
    - AssignmentAccessVoter
    - SubmissionAccessVoter
    - FileAccessVoter

Conclusion

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