Defeating XSS and XSRF with JSF Frameworks

Steve Wolf

Vice President, Application Security

AsTech Consulting, Inc.

steve.wolf@astechconsulting.com

www.astechconsulting.com

OWASP Chapter Lead – Sacramento, CA

steve.wolf@owasp.org

www.linkedin.com/in/swolf530/

Follow @SteveWolf11

JSF Based Frameworks

- Oracle/Java Reference Implementation
- Apache MyFaces Family
- Other Third Party
- Build your own

Cross Site Scripting

- Hacker Favorite
- Persistent XSS
- Non-persistent XSS

Preventing Cross Site Scripting

- Validate ALL User Input
- Context Appropriate Encoding
- Protect the Cookies

JSF Based Frameworks

- Validation Frameworks
- Context Appropriate Encoding
- Anti-Samy Input Filters
- Web Application Firewalls

JSF Validation Framework

- JSF 2.2 Server Side Validation
- Pre-defined Validation Mechanisms
- Custom Validators

Pre-defined Validators

LongRange Validator

```
<h:inputText id=“num1" value="#{myDataBean.num1}">
  <f:validateLongRange minimum="10" maximum="133" />
</h:inputText>
<h:message for=“num1" style="color:red" /><br/>
```

Other Pre-defined Validators

LengthValidator

```
<f:validateLength minimum="5" maximum="10" />
```

DoubleRangeValidator

```
<f:validateDoubleRange minimum="10.11" maximum="1000.11”/>
```

RegexValidator

```
<f:validateRegex pattern="((?=.*[a-z])(?=.*[A-Z]))" />
```

RequiredValidator

```
<f:validateRequired />
```

Date Time Validator

```
<f:convertDateTime pattern="d-M-yyy"/>
```

Custom Validators

Validator Class

```
Public class EmailValidator implements Validator {
  public void validate(FacesContext context, UIComponent
    component, Object value) throws ValidatorException {
        matcher = "^[_A-Za-z0-9-]+(\\." +
          "[_A-Za-z0-9-]+)*@[A-Za-z0-9]+(\\.[A-Za-z0-9]+)*"
          + "(\\.[A-Za-z]{2,})$";
        if(!matcher.matches()){
          FacesMessage msg = new FacesMessage("E-mail validation
          failed.", "Invalid E-mail format.");
          msg.setSeverity(FacesMessage.SEVERITY_ERROR);
          throw new ValidatorException(msg);
        }
  }
}
```

Custom Validators

Info.xhtml

```
<h:panelGrid columns="3">
 Enter your email :
 <h:inputText id="email" value="#{user.email}"
 size="20" required="true" label="Email Address">
 <f:validator
 validatorId="com.myvalidators.EmailValidator" />
 </h:inputText>
 <h:message for="email" style="color:red" />
</h:panelGrid>
```

Output Encoding

JSF 1.1 Literal text output

```
<h:outputText value="#{user.name}" />
```

JSF 2.0 Literal text output using EL

```
#{user.name}
```

Encoding turned off

```
<h:outputText value="#{user.name}" escape="false" />
```

Another Technique

Jboss SeamTextParser

```
<s:formattedText value="<b>#{user.name}</b>" />
```

Some Early Flaws

Select Items Renderer

```
<f:selectItems value="#{bean.selectItems}" var="obj"
itemValue="#{obj}" itemLabel="#{obj.name}"/>
```

JSF Version 1.2 before 1.2_08

Some tags were not rendering escape=true by default.

Websphere JSF Widget Library before 7.0.0.10

TreeControl and ResourceServlet allowing XSS.

GlassFish Admin Console 2.1

Injection via query string on some pages.

Apache MyFaces Tomhawk before 1.1.6

Injections in autoscroll parameter.

Cross Site Request Forgery

- Tricks the Browser into Sending Requests
- Susceptible Pages are those that Alter Data
- Inherits Identity and Privileges of the Victim
- Usually Initiated through Fiche or XSS

Cross Site Request Forgery

- Full Protection in JSF 2.2
- Post vs. Get
- Protecting the View State
- Some earlier JSF based Frameworks

Protecting a Postback Request

- Post is always Protected
- Non-Postback Require Config

Protecting a non-Postback Request

faces-config.xml

```
<protected-views>
  <url-pattern>my_protected_page.xhtml</url-pattern>
</protected-views>
```

Protecting a non-Postback Request

URL when Calling the Protected Page

```
http://localhost/faces/my_protected_page.xhtml?javax
.faces.Token=9879179836156547230934 2
```

Using JSF 2.2 Built-in Protection

web.xml

```
<env-entry>
  <env-entry-name>
       com.sun.faces.ClientStateSavingPassword
  </env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>MY_PASSWORD</env-entry-value>
</env-entry>
```

Encrypting MyFaces Viewstate

web.xml

```
<context-param>
  <param-name>org.apache.myfaces.USE_ENCRYPTION</param-name>
  <param-value>true</param-value>
</context-param>

<context-param>
  <param-name>org.apache.myfaces.ALGORITHM</param-name>
  <param-value>AES</param-value>
</context-param>
```

Be Informed About the Implementation

- Implementations Differ
- Doc is Not always Good
- Unit Test your Implementation
- Corporate Standards

Wrap-up

- JSF Validation Framework
- Output Encoding
- Protect the View State

Defeating XSS and XSRF with JSF

Steve Wolf

Vice President, Application Security

AsTech Consulting, Inc.

steve.wolf@astechconsulting.com

www.astechconsulting.com

OWASP Chapter Lead – Sacramento, CA

steve.wolf@owasp.org

www.linkedin.com/in/swolf530/

Follow @SteveWolf11