基于 Jboss Seam 的 Web 应用程序的安全性探讨①

周安辉

(内江职业技术学院 电子信息工程系, 内江 641000)

摘 要: 预防攻击者利用 Web 应用程序安全漏洞入侵, 开发人员采用的一些传统做法已经不适合基于 Java 的新一代快速开发平台. 通过了解新概念和新技术, 结合 OWASP 对常规风险的建议, 本文探讨了开发人员使用 Jboss Seam 构建 Web 应用程序时应该遵守的一些通用安全规则.

关键词: 安全漏洞; 认证和授权; 敏感数据; Bean 验证

Research of Web Application Security Based on Jboss Seam

ZHOU An-Hui

(Electronic Information Engineering, Neijiang Vocational and Technical College, Neijiang 641000, China)

Abstract: Preventing an attacker to take advantage of Web application security vulnerabilities, some traditional practices have been used by Web developers is not suitable for the new generation of Java-based development platform quickly. By understanding the new concepts and technology, and combined with OWASP recommendations for conventional risk, this paper explore some common practices which should be followed by Web developers, when using Jboss Seam to build Web applications.

Key words: security vulnerabilities; authentication and authorization; sensitive data; bean validation

在过去的五年中, 互联网应用快速发展, 使 Web 应用程序成为攻击的主要目标. 尽管 Java 和 J2EE 包含了许多安全技术, 但要开发没有安全漏洞的 Web 应用程序并不容易, 因为大多数能够应用到其他环境的 Web 应用程序安全漏洞还是能够用于 Java 开发的 Web 应用程序. 然而, 作为下一代企业 Java 开发工具 Jboss Seam 框架, 为 Java 开发人员高效地构建安全的 Web 应用程序提供了强大的支持, 弥补了 J2EE 在 Web 应用程序安全方面的不足.

1 Web应用程序存在的主要风险

攻击通常是利用在一个 Web 应用程序的语法和语义中的漏洞, 进行故障注入. 使用一个标准浏览器, 一个攻击者能够利用掌握的 HTTP 和 HTML 等知识尝试一个特殊漏洞, 从而有可能触发一次成功的系统入侵.

OWASP 列出了 2013 年的十大 Web 应用程序风险排名[1]: ①注入; ②破坏认证和会话管理; ③跨站点脚本(XSS); ④不安全的直接对象引用; ⑤安全配置错误; ⑥敏感数据暴露; ⑦缺少功能级访问控制; ⑧跨站点请求假冒(CSRF); ⑨使用已知易受攻击的组件; ⑩未经验证的重定向和转发.

利用一个自动扫描工具能够发现已知漏洞的大多数, 但是潜在的漏洞却无能为力, 并且要发现逻辑漏洞需要手工分析 Web 应用程序源码和安全测试. 所以, 要降低 Web 应用程序面临的风险, 还是在于开发健壮的 Web 应用程序.

2 Jboss Seam简介

Seam 是一个强大的开源开发平台, 基于 Java 构建富裕 Internet 应用程序, 集成了异步 JavaScript 和 AJAX、JSF、JPA、EJB 3.0 和 BPM, 包括高级工具, 到一个统一的全栈式解决方案中. Seam 的设计从根本上消除了体系结构和 API 层的复杂性, 开发人员使用简

① 收稿时间:2013-04-08;收到修改稿时间:2013-05-02

单的注释 Java 类、一组丰富的 UI 组件和少量的 XML, 能够组装复杂的 Web 应用程序, 并且引入了对话和声明式状态管理等概念, 解决了传统 Web 应用程序中一些常见问题.

2012 年 9 月推出的 Seam 2.3.x, 集成了 J2EE 6 的一些新功能, 如 JSF 2、JPA 2 和 Bean 验证. 选它作为开发平台, 可享受 Seam 的成熟技术, 又能使用 J2EE 6 的新功能, 这有助于开发安全的 Web 应用程序.

3 Seam的Web应用程序安全问题处理

防范上述十大风险, 在使用 Seam 2.3.x 构建一个 Web 页应用程序时, 开发人员应该遵守一些通用的规则. 在这里的探讨, 旨在提高认识和了解技巧, 其并没有绝对的顺序:

① 使用 Bean 验证防范恶意输入. 通过对 Seam 组件的类、字段或方法添加注释的形式设定约束, 例如, 用 Bean 验证注释约束一个实体 bean 的 UserName 字段:

```
@NotNull
@Size(min = 3, max = 40)
@Pattern(regexp="[a-zA-Z]+", message="User name must only contain letters")
public String getUserName(){return UserName;}
```

再用 Seam 的<s:validateAll> 标签, 通知 JSF 根据在实体 bean 中的 Bean 验证注释验证所有包含的输入字段, 让验证输入变得高效而简明.

② 对于对话式操作, 使用 Seam 对话(conversation)上下文. 从用户的角度看, 一个对话是一个工作单元, 可以跨越几个与用户的交互、几个请求和几个数据库事务. Seam 特有的对话上下文确保来自不同对话的状态不会冲突和导致错误, 维持了对话状态的一致性, 降低了传统 Web 应用程序中对话状态传递带来的潜在风险, 完美地解决了处理后退按钮和多窗口操作时一次对话的会话断裂等问题.

③ 使用 Seam Security API 的认证和授权功能对 Web 应用程序资源进行细粒度的访问控制.

Seam 认证建立在 JAAS 上, 提供了强大的、高度可配置的 API, 能根据应用程序环境实现复杂的 JAAS 认证. 此外, Seam 还提供了一个 identityManager 组件, 隐蔽了 JAAS 的复杂性, 仅简单地配置它, 完全能满足普通的认证需求.

Seam 授权通过角色和权限两个核心概念, 使用一个可扩展的框架提供多种方法来保护应用程序的资源. 利用@Restrict 注释或 typesafe 注释(等价的, typesafe 注释能提供编译时安全)能够实现对组件和实体安全的保护; 根据用户的权限, 使用与组件安全相同的 EL 表达式, 能够实现有条件地控制页面片断或独立控件的渲染; 利用应用程序的 pages.xml 文件, 仅包含<restrict/>元素在你希望保护的 page 元素内, 能实现页面的保护和页面的安全. 详情见 Seam reference 文档[2].

④ 为了方便用户, 大多数 Web 应用程序提供了“记住我”功能. 糟糕的实践是用一个存储在客户端机器上的持久化 cookie 自动认证客户端, 如果攻击者窃取了 cookie, 其在任何时候能任意地登录应用程序, 故网站受到跨站点脚本安全漏洞的威胁将比平时严重很多.

Seam 提供了两种方法解决这个问题. 第一种方法, 支持在客户端用一个 cookie 存储用户名字, 而密码交给浏览器的记住(大多数现代浏览器都提供这个功能). 这种方式不需要配置, 在登录表单中用 rememberMe.enabled 捆绑“记住我”复选框即可, 如:

```
<h:outputLabel for="rememberMe" value="Remember me"/>
<h:selectBooleanCheckbox id="rememberMe" value="#{rememberMe.enabled}"/>
```

第二种方法, 支持在一个 cookie 中存储唯一令牌, 方便用户自动认证. 先要用@TokenUsername 和@TokenValue 注释创建一个包含令牌的实体, 如 AuthenticationToken, 将这些认证令牌存储在一个数据库内; 然后在 components.xml 中进行配置使用:

```
<security:jpa-token-store token-class="seam.security.AuthenticationToken"/>
<security:remember-me mode="autoLogin"/>
```

⑤ 敏感网页使用 SSL 协议, 让 Web 应用程序的数据传输更安全. Seam 提供了对 HTTPS 协议的支持. 在 pages.xml 中, 配置 page 元素的一个 scheme 参数, 例如配置 login.xhtml 页面使用 HTTPS:

```
<page view-id="/login.xhtml"?scheme="https"/>
```

当 HTTP 浏览配置有 scheme="https"的一个页面时, 会引发使用 HTTPS 重定到同一个网页.

然而, 一旦用户访问了要求 HTTPS 的页面, 当用户导航到其它非 HTTPS 页面时, HTTPS 会继续被使用(安全, 但性能太差!), 可以在 pages.xml 中定义 HTTP

为默认 scheme, 允许用户导航到其它非 HTTPS 页面时使用 HTTP:

```
<page view-id="*" scheme="http"/>
```

并且能在 components.xml 中配置 Seam 自动在每次 scheme 改变时使当前 HTTP 会话失效:

```
<web:session invalidate-on-scheme-change="true"/>
```

此选项让你的系统减少了嗅探会话 id 攻击, 或从使用 HTTPS 的页面转到使用 HTTP 的页面时泄露敏感数据.

为了更安全, 可在 pages.xml 的 pages 元素中指定 http-port 和 https-port 属性值覆盖默认端口号.

⑥ 使用 Seam 内置 CAPTCHA 能够预防自动化软件或脚本滥用你的站点, 防止攻击者暴力破解一个特定用户账号. 这个功能很简单:

在 web.xml 中, 配置 Seam Resource Servlet, 让它为页面提供 Captcha 图像:

```
<servlet>
<servlet-name>Seam?Resource?Servlet</servlet-name>
<servlet-class>org.jboss.seam.servlet.SeamResourceServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Seam Resource Servlet</servlet-name>
<url-pattern>/seam/resource/*</url-pattern>
</servlet-mapping>
```

增加一个 CAPTCHA 到一个表单中:

```
<h:graphicImage value="/seam/resource/captcha"/>
<h:inputText id="verifyCaptcha" value="#{captcha.response}" required="true">
<s:validate />
</h:inputText>
```

⑦ 如果不小心, 一个恶意用户能够从你的应用程序显示的错误消息推断有关它重要信息. 妥善处理安全异常, 防止用户收到响应它的默认错误页面, seam 可以在 pages.xml 中配置 exception 元素, 重定向到一个“完美的”页面, 防止信息的泄露. 例如, 对未登录异常和授权异常的处理:

```
<exception class="org.jboss.seam.security.NotLoggedInException" log="true" log-level="info">
    <redirect view-id="/register.xhtml">
        <message severity="warn">You must be a member to use this feature</message>
    </redirect>
</exception>
<exception class="org.jboss.seam.security.AuthorizationException">
    <end-conversation/>
    <redirect view-id="/security_error.xhtml">
    <message severity="error">You do not have permission to do this</message>
    </redirect>
</exception>
```

⑧ 如果在你的 Web 应用程序有文件上传模块, 为防止溢出, 应该设置一个上传文件的大小限制. Seam 的 JSF 组件<s:fileUpload>, 能够非常方便地控制文件的上传功能, 然而在使用时应注意:

在 web.xml 文件中, 为复合请求配置 Seam Multipart servlet 过滤器:

```
<filter>
  <filter-name>Seam?Filter</filter-name>
  <filter-class>org.jboss.seam.servlet.SeamFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>Seam?Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

在 components.xml 中, 可以为复合请求设置两个选项, createTempFiles, 设置为 true, 上传文件被传输到一个临时文件, 而不是在内存中; maxRequestSize, 设置一个文件上传请求的最大值. 例如:

```
<component class="org.jboss.seam.web.MultipartFilter">
    <property name="createTempFiles">true</property>
    <property name="maxRequestSize">1000000</property>
</component>
```

<s:fileUpload>控件必须在一个 multipart/form-data

编码类型的一个表单内部, 如:

<h:form enctype="multipart/form-data">

<s:fileUpload id="picture"data="#{register.picture}
"accept="images/png,images/jpg
"contentType="#{register.pictureContentType}"/>

</h:form>

⑨ 来自一个用户的任何输入, 再次显示在一个网站的页面上, 都可能成为一个漏洞的来源, 如果没有恰当地转义和过滤, 任何东西都可以被插入到页面的生成输出中, 例如, 插入引发跨站点脚本攻击的 HTML 限制子集. 但是, 对于要求人性化显示的页面, 如论坛帖子、wiki 页面等, 一般允许输入特殊格式化文本, 并正确回显.

Seam 的<s:formattedText/> JSF 组件, 利用 Seam 的文本解析器能够显示符合 Seam 文本语言的格式化文本, 包括对转义符(\)的支持、用反引号(`)引用代码块, 以及对 HTML 的某些限制子集的支持, 从根本上解决用户在页面输入/显示操作带来的潜在风险. 例如:

<h:inputTextarea id="text"
value="#{selectedBlog.text}"/>

<s:formattedText value="#{selectedBlog.text}"/>

⑩ Seam 组件在使用 EntityManager?API 与数据库进行交互时, 允许你在 EJB-QL 中使用 JSF EL 表达式优化 Java 代码, 例如:

User user=em.createQuery("from User
where username=#{user.username}").getSingleResult();

但是绝不允许:

User user=em.createQuery("from User
where username="+user.getUsername()).getSingleResult
();//易受到 SQL 注入攻击.

4 结语

在 Seam 框架开发 Web 应用程序时, 遵守上文提及的原则提升安全性, 能够使它在应用中免受各种常见 Web 应用程序风险的威胁, 从而为整个系统的安全提供更好的保障.

参考文献

1 OWASP.OWASP Top 10 for 2013. 2013-03-05 [2013-03-21]. https://www.owasp.org/ index.php/Top_Ten.

2 Samson Kittoli.Seam Reference Documentation. 2012-9-12 [2013-03-12]. http://docs.jboss.org/seam/2.3.0.Final/reference/en-US/html_single/.

3 Oracle and/or its affiliates.The Java EE 6 Tutorial. 2013-01-12 [2013-03-02]. http://docs.oracle.com/javaee/6/tutorial/doc/gijrp.html/.

4 mkyong.JSF 2.0 Tutorials. 2010-10-12[2013-03-02]. http://www.mkyong.com/tutorials/jsf-2-0-tutorials/.

5 Dan Allen.李鹏,韩智.译. Seam in Action.北京:人民邮电出版社,2010.

6 DocForge.Web application/Security. 2012-10-12[2013-02-10] .http://docforge.com/wiki/Web_application/Security.

7 Microsoft.Web 应用程序的基本安全实施策略. 2007-08-20[2013-01-15]. http://technet.microsoft.com/zh-cn/subscriptions/zdh19h94(v=vs.90).aspx.

8 John Conroy. How They Hack Your Website:Overview of Common Techniques. 2008-03-05[2013-01-15]. http://www.cmswire.com/cms/web-cms/how-they-hack-your-website-overview-of-common-techniques-002339.php.

(上接第 209 页)

5 Kim K. Financial time series forecasting using support vector machines. Neurocomputing, 2003, 55: 307–319.

6 Huang W, Nakamori Y, Wang S. Forecasting stock market movement direction with support vector machine. Computers and Operations Research, 2005, 32(10): 2513–2522.

7 张晨希,张燕平,张迎春,陈洁,万忠.基于支持向量机的股票预测.计算机技术与发展,2006,16(6):35–37.

8 Hsu CW, Chang CC, Lin CJ. A practical guide to support vector classification. Taiwan: Taiwan University, 2008.

9 Keerthi S, Lin C. Asymptotic behaviors of support vector machines with Gaussian kernel. Neuraleomputation, 2003, 15 (7): 1667–2689.

10 Scholkopf B, Smola A. Leaning with kenels. Citeseer, 2002.