J2EE Form-Based 主动认证和授权解决方案①

J2EE Form-Based Active Authentication and Authorization Solution

谯 石　董 文　刘 敏　(西华师范大学 计算机学院 四川 南充 637002)

摘　要： 在实现了J2EE规范的web容器所提供的三种认证和授权机制当中，Form-based的认证机制一直是企业级 web 应用的首选。然而，为了获得可定制的用户界面，Form-based 的认证机制对寻求健壮的企业级认证和授权解决方案的系统结构师提出了几个挑战。本文围绕这几个挑战提出了一个可靠的，经过实践检验的解决方案和实现。

关键词： J2EE 安全 Form-based 认证 授权 容器

1　前言

Web 应用程序在企业级开发中得到了广泛的运用，而在开发 Web 应用程序时实现它的安全机制却是结构师和开发人员必须完成的任务。在 J2EE 中，Web 容器为 Web 应用程序提供了内置的安全机制。Web 应用程序的安全机制有二种组件：认证和授权。基于 J2EE 的 Web 容器提供三种类型的认证机制：basic, form-based, 和 HTTPS 客户端认证而后授权。由于能够对认证用户界面进行定制，大多数的Web应用程序都使用 form-based 的认证。Web 容器使用在 Web 应用程序的部署描述符中定义的安全角色对应用程序的 Web 资源的访问进行授权。然而，尽管 form-based 的认证机制有定制用户界面的便利，但此种认证方式却对寻求建立强健的企业级认证方案的结构师提出了挑战。

2　Form-based的认证和授权机制的挑战

当 Web app 试图在基于表单的认证基础上扩展被保护资源的概念时，这些挑战就表现为登录页面访问出错。这种情况可能出现在系统需要在企业级应用中展现一些相当基本的要求时，比如：

・从默认的基于表单的登录页面进行认证，而不是首先访问受保护的资源时进行认证。

・从多个分离的页面(通过未受保护的页面，比如说，登录表单)获得认证信息。

上述的两种情况，仅仅采用基于表单的认证还不能够满足系统的需求。因为 Form-based 的认证机制是一个被动的认证机制，所谓被动认证是指当用户在访问被保护资源时容器才对用户的角色进行认证。如果强行使用现有的 Form-based 的认证机制实现上述的功能时就会导致"Invalid direct reference to form login page" 出错。

3　Form-Based 的认证和授权机制

Form-Based 认证，就其本质而言，是一个 Java-specific，容器实现的认证机制，允许定制登录界面。登录是通过包含两个字段(用户名和密码分别对应于j_username and j_password)的表单和一个特殊的容器识别的 action：_j_security_check(struts)完成的。

除了登录表单，基于表单的认证的特定应用程序实现，要依赖 Web 应用程序部署描述符里配置的两个非常重要的元素：login-config 和 security-constraint。前者用于表明应用程序正在使用基于表单的认证，并指明登录和出错页面的位置，安全约束元素用来定义将要保护的资源以及与这些资源相关的角色约束。

尽管基于表单认证的实现是 vendor-specific，在 java servlet 说明书里对其功能规格进行了简要的说明。当用户试图访问受保护的资源时，容器就检测用户的认证资格，如果用户已经被认证过并拥有授权

① 基金项目:西华师范大学科研启动基金(05B054)
收稿时间:2008-11-11

访问该资源的角色(在安全约束里定义的),被请求的资源被激活，并返回对该资源的引用。如果用户没有被认证过，就会产生下列事件:

1) 弹出登录表单,并将请求保存在容器里

2) 用户提交登录表单到服务器，然后服务器对该用户进行认证

3) 如果认证通过，就将该用户的角色和访问该资源所需的角色进行比较，如果用户的角色授权允许访问该资源，系统通过第 1 步中请求的 URL,让用户到达该资源进行访问。

4) 如果认证没通过，就让用户返回到出错页面(在登录配置元素里定义的)。

Form-based 认证的优点在于让容器处理大量的认证、授权、定向到登录页面、返回到出错页面等一系列事务。然而，在认证和授权方面，如果要满足高级的认证需求，这个级别的基于容器控制的认证和授权就会带来不利因素，会直接影响到使用基于表单认证的应用程序认证能力。

• 第一个不利因素，就是容器对用户的最初请求的 URL 拥有专断的权利，通常，以程序的方式，这个参数既不可访问，也不可设置。

• 第二个不利因素，就是在大多数情况下，它阻止了定制的安全过滤器,例如,对于跨越不同的服务器时，j_security_check action 被证明是非常不可靠的。

• 第三个也是最不利的因素就是容器控制的安全不支持认证，除非用户明确地要访问受保护的资源。这也是我们通常所说的懒惰认证(被动认证)，即，需要认证时才进行认证，而不是容器随时根据应用的需求去认证。

4 可选解决方案

我们可采取三种方案解决上述需求，通常被建议采用的第一种就是建立一个定制的，servlet-based 的认证机制，这种办法虽然很健壮，并且能够很好的文档化，但却没能很好的利用容器提供的安全解决机制，并且这种方法需要通过编程实现。第二个方案就是直接继承或者实现容器提供的认证 API 类，比如继承 Tomcat 提供的 AuthenticatorBase 认证类，这种办法仅仅建议经验丰富的 java 程序员采用，但如果采用这种方案，势必依赖于特定容器的 API，不利于在不同服务器之间移植。

第三种方案，即本文所提出的方案就是扩展现存的 J2EE form-based 认证机制以满足上述的需求。这种方案能解决通常基于表单的认证所出现的诸多问题，并且不牺牲容器自身提供的认证能力，同时也不会引起容器依赖。

5 主动认证和授权解决方案

许多使用基于表单认证的系统通常也希望在绝对需要认证之前也能进行认证，这种情况本文一开始就讨论过，包括从非保护的页面登录，或者在访问受保护的资源之前从登录页面直接登录，这两种情况都需要“先发”认证。一个必须考虑的特殊情况是，如果应用程序包含很少的订造页面或者想为特定的用户角色定制不受保护的页面时。此种情况下，应用程序需要主动参与认证， 而 form-based 的认证没有明确提供这些服务时。注意这不同于认证完全由应用程序负责，在这个意义上，主动认证指应用程序扩展由容器提供的 form-based 的认证服务。

要设计和实现主动认证系统，关键是要理解在一个系统里面只有有限数量的认证入口点。对于每一个入口点，form-based 的认证既要处理主动认证，也将使用默认的懒惰认证。下面，本文将通过一个具有三个入口点的例子来进行详尽的说明。

1) 直接访问被保护的页面。

2) 从没被保护的页面嵌入的登录页面进行主动认证。

3) 直接从 web 应用发布描述器所定义的登录页面登

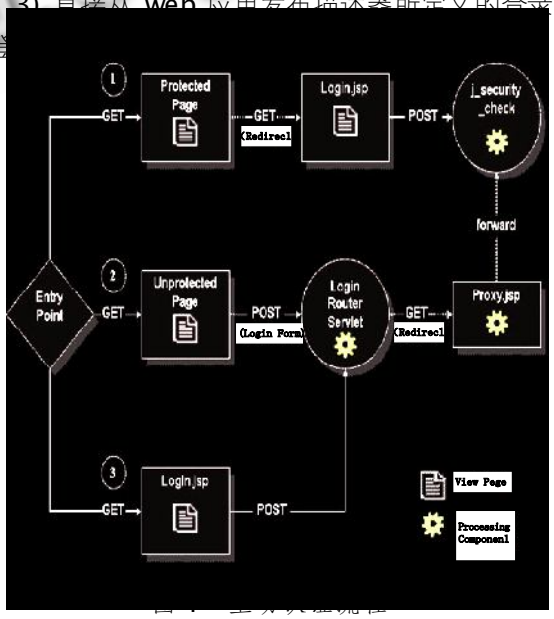

图 1 展示了对每个模块的三个入口点进行主动认证的流程，因为认证行为完全由容器控制，所以图 1 不包括被动认证

模块的三个认证入口点，仅入口点 1 是直接访问被保护的页面，其认证由 form-based 认证的现有实现处理，认证入口点 2 和 3 将不得不明确地由应用程序进行处理。我们可以断言，在访问任何受保护的资源之前，action j_security_check 将验证从表单输入的证书，即使证书有效，也将会导致“Invalid direct reference to form login page”的错误，换句话说，应用程序将不得不关注从表单登录，以防止这种类型的错误发生。

对于认证入口点 2，它涵盖了所有未受保护的页面，需要创建一个特殊的 login include 和提交用户证书到 Login Router Servlet 的特殊的 Action,该 Login Router Servlet 设置相应的登录会话(session)变量，标明未受保护的页面需要主动认证，并且改向到 JSP Proxy 页面。

这个 Proxy 作为被保护的资源，被定义在 Web 应用程序部署描述符里。因此，任何对 Proxy 的请求都将经过容器进行认证，Proxy 的任务是保留需要认证页面的跟踪，当容器返回控制权给 Proxy 时，Proxy 将重新返回到原来的页面。

对于认证入口点 3，是从默认登录页面直接登录的入口点，要求这个默认登录页面能够识别当前请求是主动认证请求还是被动认证请求。当收到主动认证请求是，登录页面应该具有一些可选的机制，以便用户在认证完成后能从定向到已选择的页面。当然，在访问受保护的资源时，不要求这种选择。

6 主动认证和授权的实现

上述表明，基于 form-based 的认证不能满足应用程序的所有需求，但它却提供了一个很好的可扩展基础，本文下面将集中就四个软件对象来综述基于表单的主动认证。

Login Include

未受保护的页面使用登录包含文件，让用户能够从应用程序中的任意入口点登录(保护的或未受保护的)。登录包含文件和标准的基于表单的认证登录很相似，此处只给出关键代码。

```
<form method="POST" action="loginRouter">
    <input type="hidden" name="j_security_check" value="/j_security_check">
    <input type="text" name="j_username">
    <input type="text" name="j_password">
    </form>
```

6.1 Login Router Servlet

Login Router Servlet 是主动认证的关键之一，它的任务有三点：

1) 它把从登录包含文件(login include)或默认登录页面(login.jsp)得到的值赋给 Session 变量(如 request 属性)，因为有些属性在改向之后就不再存在。

2) Login Router Servlet 得到最初请求的页面，以便在认证完成后页面能重新装入，应该注意到对从(login include 或默认登录页)两种来源的请求处理是不同。

3) 最后，当 Router 重定向到受保护的代理页面时，将触发容器对其进行认证。

6.2 Login Proxy (Proxy.jsp)

登录代理是最简单的主动认证部件，前面已经提到，登录代理的唯一目的是跟踪请求认证的页面，当认证和授权完成时重新返回到该页面。值得注意的是必须把登录代理作为受保护的资源定义在 Web 应用程序的发布描述器里，下面是登录代理的关键代码：

```
String redirectURL = session.getAttribute("originator").toString( );
response.sendRedirect(response.encodeRedirectURL(redirectURL));
```

6.3 Default Login (Login.jsp)

在主动认证里，默认的登录页面是最灵活也是最复杂的部分，它主要负责两点：

1) 默认登录页面需要判断它是否被应用程序里的另一个页面所请求，或者是不是通过 URL 的直接请求。基于这些情况，默认的登录页面可给出认证之后的可选择页面。

2) 默认的登录页面可以重定向到 j_security_check 进行适当的认证。

在进一步讨论默认的登录页之前，必须搞清该页面是怎样处理不同的状态的，在应用程序配置描述符里，Login.jsp 使用字符串"action=protected"和

"action=error"被定义为 form login 和 form error page，意味着需要进行初始认证或者随后需要进行认证。当用户想退出登录时，应用程序要设置"Action=logout"。给定这三种状态，默认的登录页面应该允许用户在认证的前提下给出选择机制让用户选择相关的页面。

1) 用户注销

2) 直接的 URL 页面请求

3) 页面的 Action 值不是“保护的”，“错误”，或者“注销”

如果确实要给出页面选择，通过设置“isSearchSet”变量为“true”即可，这在 Listing 2 里已经讨论过。如果确实需要提供页面选择机制，也应该提供 j_username 和 j_password 的输入认证，同时提供另一种机制，即为了重定向，在认证完成之后，将值"ORIGINATOR"赋给一系列的页面，它应该由 Login Router 来执行。登录或出错页面不需要选择机制。

默认登录页面的另外一个重要任务就是重定向到基于表单的认证，并且当页面首次装载时删除会话登录属性，如果这些属性不删除的话，由于不断的提交证书，应用程序将进入无限的重定向循环。

尽管使用的方法不同，但主动认证和 form-based 认证的最终结果都相同，如果用户通过认证和授权，他将定向到请求的页面，否则，用户将返回到默认的登录页面，并显示相关的出错信息。

7 结论

当为开发的 J2EE web 应用程序选择认证机制时，不要重复发明轮子。Form-based 的认证是一个很好的选择，它是 J2EE 安全策略规范(容器管理的安全策略)。容器管理的安全策略把安全掌握在容器手中，用配置的方式实现(不像应用控制的安全策略用代码实现)，因此，能节省开发时间，提高开发效率，便于扩展和维护。然而，Form-based 的认证是懒惰认证又叫被动认证，本文提出的基于 Form-based 的主动认证方案弥补了单纯 Form-based 的认证方案的不足，为系统结构师在设计 J2EE web 应用的安全策略时提出了新的思路。

参考文献

1 Almaer D. Web FORM-Based Authentication. http://www.onjava.com/pub/a/onjava/2001/08/06/webform.html, 2001.

2 Arumugam P. J2EE Form-Based Authentication. O'Reilly.http:www.onjava.com/pub/a/onjava/2002/06/12/form.html, 2002.

3 Java Servlet 2.3 Specification.http:www.jcp.org/aboutJava/communityprocess/final/jsr053/, 2003.

4 唐建平.与应用服务器平台无关的 Web 认证和授权方法.计算机应用, 2003,23(8):70－72.

5 王海龙,朱程荣. J2EE 中基于 CAPTCHA 的认证模块的实现.计算机技术与发展, 2006,16(1):228－230.