表 4　浙江科技学院毕业设计（论文）外文资料翻译

基于模型的 Java EE web 安全错误配置分析

摘要

Java EE 框架是 web 应用程序开发的流行技术，它为开发人员提供了定义访问控制策略的方法，以保护应用程序资源免受未经授权的信息披露和操作。不幸的是，这种安全策略的定义和操作仍然是一个复杂且容易出错的任务，需要专家级的知识来了解 Java EE 访问控制机制的语法和语义。因此，可以很容易地引入可能导致无意安全性和/或可用性问题的错误配置。针对这个问题，我们提出了一种基于模型的逆向工程方法，该方法自动评估一组安全属性，用于反向工程 Java EE 安全配置，帮助检测异常的存在。我们通过将原型工具应用于从 GitHub 中提取的真实 Java EE 应用程序的示例中，评估了我们的方法的有效性和针对性。

关键词:模型驱动工程、安全、逆向工程

1 介绍

Java EE 是一种流行的技术，用于开发动态 web 应用程序(服务也是其他不太通用的框架的基础)，它向远程用户公开分布式信息和服务。在这种情况下，安全性是一个主要问题，因为构成 web 应用程序的 web 资源可能被许多用户通过不受信任的网络访问。因此，Java EE 框架为开发人员提供了指定访问控制策略的方法，以确保 web 应用程序公开的资源的机密性和完整性。

不幸的是，尽管有这些安全机制，但是实现安全配置仍然是一个复杂的、容易出错的活动，需要高级专家来避免错误配置问题，这可能会造成严重的业务损失。事实上，开放的 Web 应用程序安全项目(OWASP)文档将 Web 应用程序错误的配置排列在最关键安全缺陷的前 10 位，因为它们很容易被利用，并且具有很强

的业务影响。

对于访问控制和 Java EE 应用程序的具体情况，以及在应用程序代码中混乱的临时安全实现机制，基于角色的访问控制(RBAC)策略是通过使用具有两种不同的文本具体语法和相对复杂的执行语义的低级规则语言编写约束来指定的。具体地说，用户可以使用一组预定义的标记元素在 XML web 描述符文件中编写约束，直接在 Java Servlet 组件上编写注释(使用不同的语法和组织 w.r.t. XML 标记元素)，或者将这两种机制结合起来。然后，必须考虑约束和相应的执行语义之间的组合规则，以便理解有效执行的策略。

这种复杂性可能导致的异常和错误配置问题(例如,意想不到的规则的结果,意外的访问控制规则,之间的相互作用等)的影响不同的只是增加不必要的复杂性指定政策的引入等意想不到的行为给予资源访问未授权方或须授权的,正如第 3 节中所报告的在线调查的参与者所证实的那样。

为了解决这个问题，我们引入了逆向工程方法来自动检测 Java EE web 应用程序中的不一致和错误配置。首先，我们定义一个 web 应用程序必须满足的属性列表，以避免重要的异常，例如冗余(即:，说明不需要的约束，使策略过于复杂)和阴影(即:，规范的约束，从来没有执行过)。其次，我们提出了一种提取方法来解析给定 web 应用程序的安全配置(同时考虑到 web 描述符配置和 Java 注释)，并将其表示为针对 Java EE web 访问控制策略的特定于平台的安全模型(PSM)。然后，在该模型之上实现 OCL 查询和模型转换操作，以便在任何给定的 Java EE web 应用程序中自动评估定义的属性。此外，我们的工具会生成诊断报告，帮助识别对属性违规负责的源配置元素，从而帮助开发人员修复它们。

我们通过使用从 GitHub(基于 web 的 Git 存储库托管平台)中提取的可公开的 Java EE web 应用程序来评估我们的方法的有效性。这个评估表明，相关的安全配置数量确实违反了我们推荐的属性，并且我们的工具能够成功地检测到这些违规行为。

论文的其余部分组织如下。第 2 节描述了 Java EE 的访问控制机制。第 3 节介绍了 Java EE 项目中安全性使用的动机调查。第 4 节描述了一些安全属性。第 5 节展示了如何从 Java EE web 应用程序中提取访问控制模型，而第 6 节详细介绍了我们对其属性进行评估的自动方法。第 7 节对我们的方法提出了一些附加的应用程序。第 8 节展示了评估结果，第 9 节给出了工具实现的详细信息。有关工作将在第 10 节讨论。最后，我们在第 11 节中给出结论和今后的工作。

2 Java EE web 安全

粗略地说，在 Java EE 领域，当 web 客户端发出 HTTP 请求时，web 服务器将请求转换为 HTTP Servlet 调用，以执行一些业务逻辑操作(Servlet 和 Java server Pages)。

在这个模式中，一个非常重要的需求是确保 web 应用程序管理的资源的机密性和完整性，因为它们可以被许多用户访问并遍历未保护的网络。从这个意义上说，Java EE 框架提供了随时可用的访问控制工具。下面我们将简要描述 Java EE 为实现 web 应用程序中访问控制策略所提供的机制。

如前所述，Java EE 应用程序通常是由 jsp 和 Servlet 组成的(jsp 又被转换为 Servlet)。这个层中的访问控制机制负责控制对这些元素的访问，以及其他可访问的构件(纯 HTML 页面、多媒体文档等)。这些访问控制策略可以使用两种不同的机制来指定:声明性安全性和程序性安全性，后者提供了需要用户上下文评估的高级访问控制的情况。尽管如此，Java EE 规范建议在可能的情况下优先使用声明式安全性。

对于声明式访问控制策略，有两种选择:(1)在可移植的部署描述符(web.xml)中编写安全约束，以及(2)作为 servlet Java 代码的一部分编写安全注释(注意，不是所有的安全配置都可以通过注释来指定)。

代码清单 1 显示了在 web 中定义的安全约束。xml 描述符。它包含三个主要元素:一个 web 资源集合，指定受安全约束影响的资源的路径和用于该访问的 HTTP 方法(在本例中为/限制/雇员/n 路径和 GET 方法);一个自定义约束声明了哪些角色(如果有的话)可以访问资源(仅在示例中的角色 Employee)和一个用户数据约束，该约束决定了用户数据必须如何从 web 应用程序和 web 应用程序(例如，任何类型的传输都被接受)。此外，尽管它不是强制的，但是 web 描述符可能包含角色声明(参见代码清单 2)。

代码清单 1 在 web.xml 中的安全约束

```
<security-constraint>
    <display-name>GET To Employees</display-name>
    <web-resource-collection>
        <web-resource-name>Restricted</web-resource-name>
```

```
        <url-pattern>/restricted/employee/*</url-pattern>
        <http-method>GET</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>Employee<1role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>NONE/transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

代码清单 2 在 web.xml 的角色声明

```
<security-role>
    <role-name>Employee</role-name>
</security-role>
```