**二问题描述**

如前面所提到的，网络应用程序和控制器之间的交互在整个北界接口(NBI)中进行。该接口应该允许可信应用程序对网络进行编程，并从网络请求服务/信息。这种交互可以简化为:读取网络状态/编写网络策略。

读取网络状态:这涉及向控制器发送HTTP GET请求的应用程序。控制器解释并将请求转换为等效的OpenFlow请求，它与相关的数据平面元素进行通信。数据平面元素响应所请求的数据，控制器以HTTP响应的形式解释和提供给应用程序。

编写网络策略:类似地，要在交换机上安装流规则，应用程序向控制器发送一个HTTP POST请求。控制器解释并将请求转换为OpenFlow修改命令，指示相关开关将此流添加到其流表。然后，控制器向应用程序发送一个HTTP响应，确认流规则安装的成功/失败。

**这种方法有很多缺点:**

1 没有对RESTful API命令的验证。

对于HTTP请求的来源或对特定应用程序的请求的相关性的确认是无法控制的。因此，任何应用程序都可以读取或写入网络状态。

2没有计划来确保安装的规则不重叠或相互干扰。因此，有可能引入规则，从而破坏网络的预期行为。

3应用程序不必提供身份信息。

因此，没有办法规范哪个应用程序/策略应该具有更高的优先级。

4安装后不得使用任何应用程序或行为检查。这意味着合法安装的应用程序可能会在没有检测的情况下恶意地进行恶意攻击。

**这个评估有三个可能的解决方案:**

1)规则冲突检测和纠正

2)应用程序识别和优先执行

3)恶意活动的探测和缓解

已经提出了一些用于规则冲突检测和纠正的解决方案，并将在第三节中突出显示。本文介绍的工作重点是应用程序识别和优先执行。本文还提出了一种针对恶意活动检测和缓解的初步方法，并将在今后的工作中进行。

**三相关工作**

**在[1] [3]SDN平台上已经确定了一些安全问题，涉及到与北界接口的安全性相关的具体问题。**

第一个是应用程序策略冲突。当控制器从两个或多个应用程序接收不兼容的流规则时，问题就出现了。在文献[4]-[9]中提出了几个解决方案。这些解决方案在执行中不同，但本质上是监视网络中的变化，在网络行为中构建模型，使用自定义算法来派生网络是否包含错误并解决策略冲突。

Frenetic [10]是一个特定的向北的API，旨在解决策略冲突。它用于对由集中控制的网络交换机的集合进行编程控制器。运行时系统在指示控制器在交换机中安装流规则之前，将流规则转换为不重叠的策略。但是，Frenetic并没有将应用程序认证到网络。

另一种策略冲突解决方法是使用基于角色的授权方案，如在[11]中提出。该系统集成了FortNOX执行引擎，它处理可能与规则插入的冲突，从而规则的接受/拒绝依赖于作者的安全授权。与现有流规则冲突的新流程规则将被FortNOX检测到。如果新(冲突的)流规则请求是由更高优先级的作者生成的，那么现有的流规则将被替换。然而,如果新的流程规则由较低优先级的作者产生，然后将被忽略。这种方法的局限性在于确定适当的安全授权

的水平。FortNOX不解决应用程序标识和优先执行的问题。

在[12]中标识了向每个没有保护的应用程序打开OpenFlow的全部特权的问题。作者建议使用一组权限和一个隔离机制来强制API条目的权限。该解决方案将最小特权应用于保护网络免受控制飞机攻击的应用程序。

本论文所提供的解决方案最近的工作是由斯坦福研究所开发的SE(安全增强)泛光灯[13]。它是泛光灯的开放流量控制器的扩展。泛光灯引入了安全执行内核(SEK)，这是对FortNOX提供的改进检测可能的规则冲突，并协调控制和数据平面之间的通信。se泛光灯的实现还包括一个经过数字验证的北向API。管理员需要预先签署OpenFlow应用程序的java类，这可能在运行时由SEK进行数字验证。一旦签名和验证，应用程序有权修改或查询网络，或者网络上的流量。

我们的解决方案是通过方法的粒度来区分的。在se泛光标志和验证一个完整的应用程序的地方，我们的基于permission基于的方法允许将一组操作授予应用程序。这组权限可以增强或减少应用程序的生命，并基于监视应用程序活动。此外，我们的解决方案是可扩展的利用监视操作日志来应用入侵检测方法的潜力。我们的解决方案的进一步提供是对内部和外部应用程序的适用性，如第IV部分所述。

**四 系统设计**

安全要求来规范应用程序能够访问的网络信息以及应用程序可以执行的网络操作

在第二节中被定义。在本节中，我们描述了满足这种安全性需求的系统。

该系统是基于泛光架构(图1)设计的，考虑了两个应用程序类别（内部应用和外部应用）;1应用程序平面上的外部应用程序通过NBI的REST API与控制器进行通信，2将模块应用程序编写为java包，并直接合并到控制器中。这些应用程序是作为控制器的一部分进行编译的，可以直接访问各种控制器类、它们的方法和数据。

A:系统属性

下列系统属性已被识别，并在随后的部分中呈现:

1-6

**b .权限定义**

作为第一步，定义了一组权限。这些权限反映应用程序用于读取网络状态或写入网络政策的OpenFlow相关命令。权限集是在PermOF[12]中呈现的分类的扩展。泛光控制器提供了一组REST统一资源标识符(uri)，应用程序用于指定所需的资源和操作。为了使权限集包含所有相关的操作，引入了读取控制器信息的权限。这个权限封装了控制器内存使用调用。权限列在图2中。

C模块设计

由于必须保护内部和外部的应用程序，因此不可能仅通过获取REST调用来提供预期的安全功能。这是因为内部java模块可以直接访问REST调用所使用的底层方法。因此，这些方法本身必须得到保护。这个功能是在operation检查点使用权限检查方法引入的。在图2(筛选方法)中列出的每一个连接到请求权限的方法中都部署了operation检查点。

当一个与图2中标识的权限相关联的操作被请求时，调用operation检查点来确定是否授予了必要的权限。如果有，操作将正常执行。但是，如果应用程序没有适当的权限，则操作将不会这样就可以保护网络免受未经授权的访问和修改。

D 日志记录

日志功能被设计用来记录所有未经授权的尝试，以修改网络或获取关于网络的信息。所记录的信息如下:hdatei htimei happlicationIDi hdeniedpermissioni。日志函数嵌入在operation检查点，并在请求操作被拒绝时调用。允许入侵记录提供未授权的访问历史。图6提供了日志文件内容的说明。在网络安全的环境中，日志文件提供了一个重要的功能。可以根据尝试操作的顺序来确定入侵行为的模式记录在日志文件中，可以对恶意应用程序行为进行检测和保护。

五测试

测试环境包括一个Mininet仿真网络和一系列与OpenvSwitch OpenFlow开关相连接的主机，由修改后的泛光灯控制器控制，实现操作检查点。测试环境运行在运行ubuntu12.04 LTS的VirtualBox VM中。

性能：通过测量权限检查的执行所引入的延迟，可以评估引入权限系统的性能损失。

权限检查在所有筛选方法中都是一致的，并涉及实例化一个操作检查点的实例，以确定所请求的操作是否允许。因此，性能惩罚是通过测试REST调用之一(控制器内存使用)来评估的。执行时间被记录下来和没有权限检查的对比。测试重复了8次，结果详见表I。

观察到的延迟是可以忽略的。引入的平均延迟是367.125 s，这在泛光灯控制器上没有明显增加额外的开销。

六不足

首先，不可能阻止对默认java方法的访问。所访问的默认java方法是读取控制器信息操作的getRuntime。在通过REST调用访问时，可以确保该操作的安全性，但是由于内部Java模块的内部性，可以访问无法阻止的对象。第二，为了实现权限(在有效负载中读取pkt，发送pkt，流mod修改hdr和修改所有的流)，需要对控制器进行广泛的重新设计，而这并不是这项工作的目标。