SDN与SDN网络安全

随着互联网的发展，基于IP的简洁网络架构却变的越来越臃肿以及无法满足目前的承载需求，网络发展面临着管理运维复杂，网络创新困难，设备日益臃肿等一系列问题。想要从根本上摆脱这个问题，就需要打破传统网络的原先封闭式架构，增强网络的灵活配置的特性，并利用好网络的可编程能力，于是经过多年的技术发展，SDN（软件定义网络）技术便应运而生[1]。它被誉为是继MPLS技术之后在网络技术领域的一次重大技术变革，从根本上对网络的架构产生革命性的冲击[2]。

SDN这个名词可能在两年前还完全没有人知道，但是如今，它却成为了一项在网络领域十分热门的词语。随着计算机网络的发展，云计算也成为了计算机网络技术中必不可少的一部分。云计算的发展，依赖的关键就是网络虚拟化技术，这种技术要求云计算服务商为用户提供具有高可用性，高扩展性的计算和管理服务。于是最近几年，计算机的虚拟化技术有了很大的发展，但是还有很多问题没有解决，网络虚拟化的技术还没有达到特别成熟的地步，而SDN和OpenFlow的出现，可以为网络虚拟化的实现带来很大的便利。

SDN的全称是Software Defined Network，翻译成中文就是软件定义网络，它一种新型网络创新架构，具有动态性，可控性，高适应性等特点，对于现在的高带宽网络环境有较好的适应性。在我们现在所使用的网络中，交换机和路由器是必不可少的成分，但还是这两者有一个共同的缺点，他们都是由厂家直接控制的，也就是说固件被直接锁定，无法进行差异化的修改。而SDN的目标是通过虚拟化的方法，把控制中心和数据中心分离开来，这样便可以站在更高的层面来监视整个网络，从而可以摆脱厂商对于网络设备的直接控制。这样可以使二者分别升级，于是便会更灵活缩短网络架构的迭代周期，并且还可以节省大量的成本。

由于SDN网络是一种新型的网络架构，所以会带来一些新的特性，除了相关的配置的更新，也会产生一些相关的新领域的问题。在评价一个网络系统的优劣的时候，首先要考虑到的就是网络系统是否安全可靠，是否能够支撑起网络用户的正常使用，是否能够防止网络被攻击，在面对攻击者的攻击时，能否快速做出反应，以解决网络中存在的漏洞和一些问题。

想要理解网络中的安全因素就要先了解SDN网络中的一些架构。SDN（软件定义网络）把传统的交换机（路由器）设备进行了“拆分”，传统的交换机功能从某种意义上来说，可以认为是由最底层的流量转发，以及更高级的其他处理功能（比方说网管控制、负载均衡等）这两部分组成。SDN剥离了交换机除流量转发之外的所有高级处理功能，并且将这部分高级处理功能移到了单独的设备中，称为“控制器”。

控制器实际上起到了“处理器”的作用，为交换机的数据转发进行最佳设计，以及提供更多的应用服务。控制器可以是一个专有硬件装置，也可以是一个虚拟机，或是一个物理服务器。那么，控制器和交换机之间是如何“交流”的呢，这时候就需要一种协议来定义二者之间的通信规则，如同TCP/IP等网络协议类似，OpenFlow就是这么一种网络协议。

交换机和SDN控制器之间使用OpenFlow协议进行“交流”，由SDN控制器来发出控制指令，交换机负责接收指令并进行相应的数据转发。控制器其实也只是一个“平台”，它的作用在于为网络应用提供一个统一的运行环境。我们所说的控制器所做的高级处理功能，都是控制器平台上的“应用”完成的，也就是SDN应用。

总体来说，一个SDN网络中包含三个架构层级：底层物理网络（交换机等），SDN应用以及SDN控制器：

　　物理网络处于最低层，包含组成整个IT基础架构的网络中的所有物理设备，比方说交换机、网线等等。但是这里的交换机已经不是我们现在意义上的交换机，而是“阉割”版的交换机，就是说已经被剥离控制处理功能，只剩下数据流量转发底层功能，这时候的交换机就如同一个通道，只负责流量的通行。（当然，为了网络设备的平滑演进，SDN网络被设计为能够同现有的交换机进行协同）

SDN应用是整个SDN生态环境的有机“活力”成分，就如同现在风靡在手机平台上的手机应用一样。SDN剥离交换机的处理控制功能的终极目的也在于此，这样用户就可以自由的开发定制“交换机”（网络）的功能，而不需要受制于网络设备厂商。同时，这也降低了网络设备行业的进入门槛，创业者可以不被底层的硬件阻碍，而从SDN应用入手做开发。

SDN控制器是整个SDN架构的“中间件”，控制器必须整合网络中所有物理和虚拟设备。控制器与网络设备之间高度融合，密切配合完成所有的网络任务。在SDN环境中，控制器会使用OpenFlow协议和NETCONF协议与交换机联系。(OpenFlow是将流数据发送到交换机的API，而NETCONF是网络配置API)。

大致了解了SDN的架构以后也就不难总结出针对SDN架构可以产生的相关安全方面的可研究的方向以及网络架构中相对薄弱的地方。我们可以预见几种针对SDN架构的攻击方法。SDN架构较为常见网络安全问题包括对SDN架构中各层的攻击。

1. 数据层攻击。攻击者可能将来自网络本身的某个节点（例如，OF交换机，接入SDN交换机的主机）作为攻击目标。从理论上说，攻击者可以先获得未经网络访问的权限，然后尝试进行攻击运行状态不稳定的网络节点。这可能是一个拒绝服务攻击（Denial of Service , DoS），或者是对交换机等网络基础设施进行的Fuzzing攻击。有许多南向接口协议用于控制器与数据层的交换机进行通信。每个协议虽然都有自己的安全通信机制。但因为这些协议都非常新，所以并没有能够实现综合安全部署的方法。攻击者也可以利用这些协议的特性在OF交换机中添加新的流表项。之所以这么做，是因为攻击者试图将这些特定服务类型的数据流进行欺骗“拦截”，不允许其在网络中传输。攻击者有可能引入一个新的数据流，并且指导引入的数据流绕过防火墙，从而使攻击者取得数据流走向的控制权。攻击者也有可能利用这些能力来进行网络嗅探，甚至可能引发中间人攻击。
2. 控制层攻击。SDN控制器是个很明显的攻击目标。攻击者可能会为了不同的目的而把SDN控制器作为攻击目标。攻击者可能会向控制器发送伪装的南向/北向接口对话消息，如果控制器回复了攻击者发送的南向/北向接口对话消息，那么攻击者就有能力绕过控制器所部署的安全策略的检测。攻击者可能会向控制器发起DoS或其他方式的资源消耗攻击，使得控制器处理Packet In消息变得非常缓慢。甚至可能导致整个网络崩溃。
3. 应用层攻击。攻击北向接口协议的行为也可以看做是一种攻击的方法。这些北向接口都由控制器管理。这些北向接口可以通过Python、Java、C、REST、XML、JSON等方式进行数据封装。如果攻击者利用了这些公开且没有任何认证机制的北向接口，那么攻击者就可以通过控制器来控制SDN网络的通信，并且可以制定自己的“业务策略”。

通过对以上攻击的思路的分析，我们也不难想到简单的应对策略以及解决思路。既然攻击主要可以分为3个层面，那么安全的防范的方式也可以从这三个方面展开。

1. 数据平面的安全强化。典型的SDN系统都是基于x86的处理器，并且使用TLS（原SSL）协议来保障控制平面的安全。这些生存时间很长的HTTP会话可能会引发大范围的攻击，从而危及控制层面的完整性，这将导致使用云服务的租户被暴露。组织机构可能更愿意在网络设备代理和控制器之间使用TLS协议建立认证加密机制。倘若使用TLS协议，控制器和网络设备/SDN代理之间可以互相验证其身份，可以避免网络嗅探和南向接口的欺骗通信。考虑到南向接口协议经常被使用，所以南向接口的通信安全必须被强化。一些协议可能会使用如之前所述的TLS会话机制，同样也可使用共享密钥密码或随机数的方式来防止再次发起的攻击。例如：SNMPv3协议的安全性比SNMPv2c和SSH协议高，更远比Telnet协议安全性高。其他的专有南向接口协议可能使用自己的方法来建立控制器和网络代理设备之间的加密认证通信机制，从而有效防止攻击者的网络嗅探和欺骗。同样，由于考虑到DCI协议经常被使用到，所以可能会有可配置的选项进行隧道端点的验证和隧道的安全通信。

2. 控制层的安全强化。控制器是一个关键点攻击目标。因此，它的安全性必须被强化。控制器和网络节点的安全性最终归结于主机操作系统的安全性。强化所有公共的Linux服务器的安全性是最好的选择。这样，机构就可通过其控制器密切监控任何可疑行为。如果一个控制器遭受了DoS攻击，那么它必须拥有一个高可靠性（High-Availability, HA）的架构。采用冗余控制器的SDN可能会丧失主控制器的选举功能。如果攻击者对所有控制器进行DoS攻击，那么攻击的危害性大大提升。此外，攻击者的倘若采用隐形攻击，那么受攻击目标尤其不易被发现。

3. 应用层的安全强化。另一个保护措施是使用带外数据（Out of Band，OOB）网络来控制流量传输。它在数据中心部署一个OOB网络，相比在企业广域网中部署成本更低更简单。使用一个OOB网络也可强化控制器的南/北向接口协议通信安全。采用TLS、SSH协议或其他的方法来强化北向接口的通信和控制器的管控将是最佳选择。来自应用程序、服务间的请求或者来自控制器中的数据，都应该有相应的加密认证方式来确保数据及通信的安全。给所有基于北向接口的应用程序进行请求SDN资源的安全编码非常有必要。安全编码不仅有利于因特网web程序的安全，而且也适用于SDN北向接口的安全强化。有些SDN系统有能力根据OF交换机中的数据流验证控制器的策略。这种类型的检查（类似于FlowChecker）有助于评估控制器对攻击的防御效果[3]。

通过对之前的网络安全思路的分析，可以产生很多的网络加固方式。最直观的方式也是最容易想到的方法应该就是设计并实现一个简单的静态包过滤防火墙。所以下面的内容会简单分析，如何实现一个在SDN网络中的简单防火墙，利用现有的开源代码以及开源控制器，就可以实现一个简单的控制器静态包过滤防火墙，可以阻断最简单最基础的网络流量攻击。

这个设计采用的控制器是最主流的OpenDayLight控制器，Opendaylight Controller提供了一个模块化的开放SDN控制器。该控制器提供了北向接口和南向接口，分别有着不同的作用。北向接口可以给应用程序使用，类似于API性质，可以通过Restful API进行调用。南向接口支持OpenFlow等各种协议，通过南向接口可以直接操纵交换机进行各种操作。

网络环境采用mininet进行模拟，通过Mininet可以几乎真实的模拟一个网络环境。Mininet 可以模拟一个完整的网络主机、链接和交换机在同一台计算机上且有助于互动开发、测试和演示，尤其是那些使用 OpenFlow 和 SDN 技术；同时也可将此进程虚拟化的平台下代码迁移到真实的环境中。

包过滤防火墙通过对于所流过数据的包头进行查看，通过包头的结果，来决定整个包的命运和去留。最简单的动作为丢弃和接受，也就是允许相应的包通过或者禁止包通过，当然，还有可能根据功能的要求进行一些其他的操作。

传统包过滤防火墙大致是通过对数据包的以下特性来进行检验的：

1. IP源地址

2. IP目的地址

3. 协议（TCP包、UDP包和ICMP包）

4. 目的端口号

5. 源端口号

6. 消息类型分类

7. 消息发出的端口

8. 消息到达的端口

根据防火墙的不同要求，还可以增加或减少相应的匹配域。这里的防火墙的实现思路也是这样。

以下是静态包过滤防火墙的工作流程，在ODL中的防火墙的设计也基于传统静态包过滤防火墙的设计思路。

1. 防火墙的规则要让硬件设备储存起来。

2. 当数据包到达端口时，要取出报头，对其中内容进行相应的语法分析。对于传统的包过滤防火墙，一般只过滤IP，TCP，UDP报头部分。

3. 若该数据包满足某一项接受规则，则这个数据包可以通过或者被进一步处理。

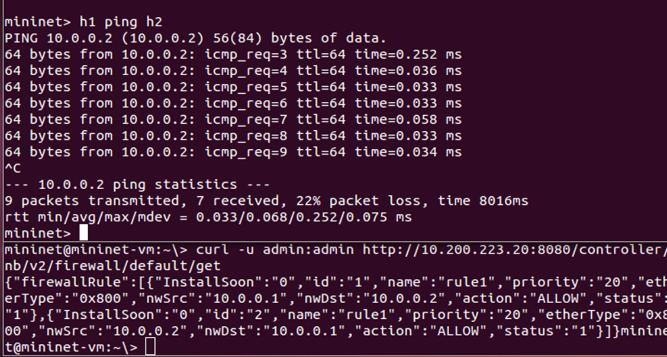
4. 若该数据包满足某一项不接受规则，即该规则阻止数据包的接收，则此包被禁止通过。

5. 若该数据包不满足任何一项规则，则按照要求，该数据包被阻塞。

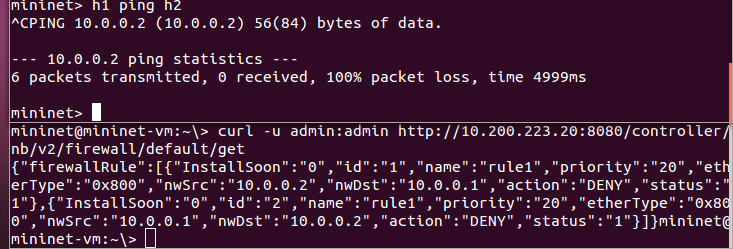
在逻辑的设计上，以及功能流程，可以采用这样的思路。首先会通过控制器，得到由交换机无法匹配而传来的packet in消息。得到这个消息之后，控制器便开始协议解析的工作，首先判定是否为packet in消息，如果是则进入下一步的操作，如果不是，那么久会忽略这个消息。因为控制器中可能有多个模块是对数据包进行操作的，别的模块可能也可以对该数据包进行相匹配的操作。接下来就会判断接收到包的协议类型，如果类型属于可以接受的类型，则第一步的匹配成功。接下来会取出之前定义的源IP目的IP端口号等信息，进行查表匹配，如果匹配就说明成功，反正则说明不匹配，会丢弃数据包。匹配也分为两种情况，一种是匹配通过，这个时候会调用相关函数修改流表中的项目。还有一种是匹配了但是防火墙要求丢弃数据包，这个时候也会提醒交换机，将相应的数据包丢弃，并且把丢弃的消息写入流表中，在流表有效的情况下，如果再次获得相同的数据包，则会按照流表的内容进行分发或者丢弃的操作，就不需要再次上交packet in消息了。关于具体的设计与实现，需要利用到OpenDayLight提供的一些基本功能以及接口，通过对于接口的调用，可以将功能以bundle的形式，加入到ODL的控制器当中，达到的目的也很明确，就是在控制器获取交换机传来的数据包的时候，可以让防火墙的模块也获得一份数据包的镜像，通过这个镜像，模块提取出自己需要识别的数据包内容，进一步的进行分析，最后根据结果产生不同的动作。

那么还会有一个问题，这些规则的来源是什么地方。这个可以采用RestFul API的方式来进行解决，通过RestFul API,以及相应的模块接口，可以将规则打包成json或者xml格式，传入到控制器的模块之中，从而产生期望的预定规则。这里采用的ODL控制器就使用了这种解决思路。在OpenDaylight里面有个很重要的功能模块就是SAL——Service Abstract Layer（服务抽象层），他的主要工作就是把南向接口的功能包装成统一的北向接口，不过在OpenDaylight里面有两种表现形式AD-SAL、MD-SAL，他们的工作就是提供REST API。AD-SAL比较好理解，看要实现什么样的API，去找相关的南向接口，然后做的工作就是将这些南向的功能变成统一的北向接口。

当把这些设计实现了以后便可以实现简单的静态包过滤防火墙功能，可以在mininet上布置一个简单的网络，然后连接上控制器进行测试，如图。让防火墙设置的规则是通过的时候，相应的规则是可以通过的，可以看到10.0.0.1节点和10.0.0.2节点是可以互通的，也就是达到了预期的效果。相反，当设定的规则是10.0.0.1和10.0.0.2之间的通信是拒绝的时候，通过ping测试，可以发现两个节点之间是无法通信的，这也就证明了设计达到了预期的效果。



防火墙允许通过



防火墙禁止通过

以上的部分是关于实现了一个在SDN中的简单防火墙，此外，在之前的网络安全的讨论内容中，我们提到了关于在数据平面与控制平面的一个监控问题，通过对架构的理解，可以知道，如果黑客可以一直监听这个信道，那么黑客便可以知道整个网络的所有状态，这对于保证网络的安全是不利的，所以便也有了一个设想，是不是当控制器知道了整个网络的拓扑的时候可以偶然的随机的下发一些少量的没有意义的流表，由于交换机只会机械的将这些流表加入自己的流表项，而当黑客在监听这些信道的时候可能他也不知道这些流表的意义是什么，通过这种方式，可能就可以误导黑客，对于网络的状态产生错误的判断，从而可以直接的保障网络系统的私密性，从而可以保证网络安全。但是这个方式也有他的局限性，比如说控制器需要知道哪些流表是可以随机下发的，因为下发流表的时候首先需呀保证，下发的流表对于正常合法的网络拓扑不能产生任何影响，此外还需要考虑到，采用了这种策略以后对网络的流量会不会产生一些消极影响，这些消极影响会不会直接影响到网络的性能。最重要的一点是，当下发了这些假的流表项，并且这些流表项被中间黑客截获以后，能否能真正的欺骗到黑客，能否正在的保障网络安全，这可能还是需要控制器对于流表的优化，以及对于网络拓扑认识程度，这样才可以下发足以以假乱真，但是又不会真正影响网络性能的流表。

1. 李呈.SDN产业发展白皮书[EB/OL]. <http://www.sdnap.com/sdn-downloads/5860.html> 2015-4-22.
2. 小武.我们能为SDN做什么[EB/OL]. [http://www.sdnap.com/sdnap-post/4064.html 2015-4-2](http://www.sdnap.com/sdnap-post/4064.html%202015-4-2).
3. Scott Hogg. SDN security attack vectors and SDN hardening [EB/OL]. http://www.computerworld.com.sg/