目录

[第1章 绪论 5](#_Toc515051923)

[1.1 课题研究背景与研究现状 5](#_Toc515051924)

[1.2 课题研究意义 7](#_Toc515051925)

[1.3 论文的结构安排 7](#_Toc515051926)

[第2章 软件定义网络(SDN)与DDoS攻击 8](#_Toc515051927)

[2.1 软件定义网络 8](#_Toc515051928)

[2.1.1 SDN的模型架构 8](#_Toc515051929)

[2.1.2 SDN的优势 10](#_Toc515051930)

[2.2 分布式拒绝服务攻击 10](#_Toc515051931)

[2.2.1 DDoS的攻击类型 10](#_Toc515051932)

[2.2.2 DDoS的危害性 11](#_Toc515051933)

[2.2.3 已有的检测和防护算法 12](#_Toc515051934)

[第3章 基于流量监控的源IP攻击检测与防护方案 12](#_Toc515051935)

[3.1 方案模型 12](#_Toc515051936)

[3.1.1 流量阈值检测模块 13](#_Toc515051937)

[3.1.2 流量特征分析模块 13](#_Toc515051938)

[3.1.3 流量过滤模块 15](#_Toc515051939)

[3.2 流程图 16](#_Toc515051940)

[第4章 仿真实验 18](#_Toc515051941)

[4.1 仿真环境与工具介绍 18](#_Toc515051942)

[4.1.1 Ubuntu 18](#_Toc515051943)

[4.1.2 mininet仿真工具 18](#_Toc515051944)

[4.1.3 OpenDayLight 18](#_Toc515051945)

[4.1.4 云主机 18](#_Toc515051946)

[4.1.5 python模块 18](#_Toc515051947)

[4.1.6 sflow网络监测工具 19](#_Toc515051948)

[4.1.7 hping3 19](#_Toc515051949)

[4.2 环境搭建 20](#_Toc515051950)

[4.2.1 Ubuntu环境部署 20](#_Toc515051951)

[4.2.2 OpenDayLight环境部署 20](#_Toc515051952)

[4.2.3 mininet环境部署 23](#_Toc515051953)

[4.2.4 其它安装项 23](#_Toc515051954)

[4.3 仿真过程 24](#_Toc515051955)

[4.3.1 网络拓扑创建 24](#_Toc515051956)

[4.3.2 开启sflow 26](#_Toc515051957)

[4.3.3 网络连通性测试 28](#_Toc515051958)

[4.3.4 DDoS攻击模拟 29](#_Toc515051959)

[4.3.5 流量检测与防护 31](#_Toc515051960)

[4.4 功能模块介绍 33](#_Toc515051961)

[4.4.1 拓扑创建功能模块 33](#_Toc515051962)

[4.4.2 流量检测与防护模块 33](#_Toc515051963)

[第5章 总结与展望 36](#_Toc515051964)

[5.1 总结 36](#_Toc515051965)

[5.2 展望 36](#_Toc515051966)

软件定义网络中DDoS的检测与防护

专业班级：J通信工程1402 学生姓名：陆信宇

指导教师：蒋中秋 职称：讲师

摘要

软件定义网络(SDN)是一种新型的网络架构方案，代表了网络在未来的发展趋势。与传统网络相比，SDN的数据控制平面分离，并且网络可编程，这使得SDN架构下的网络变得更加灵活可控。Distributed Denial of Service(DDoS)，分布式拒绝服务攻击是网络信息安全领域重点关注的问题之一，如何快速的检测DDoS攻击以及作出精确有效的防御一直都是网络信息安全领域的研究热点之一。本文对基于SDN网络的DDoS攻击的检测与防护进行研究。

首先，需要对DDoS攻击进行检测，即判断是否遭受到了DDoS攻击。当大规模DDoS攻击流量进入网络时，必然会导致网络状态与未遭到攻击时有很大的差别。首当其冲就是网络带宽资源消耗成倍增加。本文利用网络带宽有限，通过给网络流量设定阈值的方法检测DDoS攻击，当网络流量超出阈值时即判断可能遭受DDoS攻击。

其次，需要分析正在遭受的DDoS攻击的类型。本文根据源IP地址是否随机生成，将攻击分为随机源IP地址的DDoS攻击和固定源IP地址的DDoS攻击。针对随机源IP地址的DDoS攻击，由于ipv4地址数目庞大，随机生成的ip地址重复率低，通过计算固定数目的数据包中，源IP地址的重复率以及源IP地址数目的平均值可判断是否遭受该攻击。针对固定源IP地址的DDoS攻击，需要收集一定数目数目包中的多个参数，如源目IP变化速率、源目端口变化速率、数据包平均大小等，据此进一步判断哪些源IP地址是攻击源。

最后，需要针对检测到的DDoS流量进行过滤拦截。本文根据两种不同的 DDoS攻击类型将过滤方案分为两种。针对随机源IP地址的DDoS攻击，将直接丢弃从接口收到的所有数据包，当流量正常时将放行流量；针对固定源IP地址的DDoS攻击，将丢弃来自检测到的异常源IP的数据包，当流量正常时将放行流量。

关键词：软件定义网络 SDN OpenFlow DDoS 分布式拒绝服务攻击 攻击检测 攻击防护

Detection and Protection of DDoS in Software-defined Networks

Abstract

Software-defined network (SDN) is a new type of network architecture that represents the future development of the network. Compared with the traditional network, the SDN's data plane is separated from control plane and the network is programmable. This makes the network under the SDN architecture more flexible and controllable. Distributed Denial of Service (DDoS), distributed denial of service (DDoS) attack is one of the most important issues in the field of network information security. How to quickly detect DDoS attacks and make accurate and effective defenses has always been one of the research hotspots in the field of network information security. This paper studies the detection and protection of DDoS attacks based on SDN networks.

First of all, the DDoS attack needs to be detected, that is, to determine whether it has suffered a DDoS attack. When large-scale DDoS attack traffic enters the network, it will inevitably lead to great differences between the state of the network and when it is not attacked. First and foremost is the exponential increase in network bandwidth resource consumption. In this paper, the network bandwidth is limited, and DDoS attacks are detected by setting thresholds for network traffic. When the network traffic exceeds the threshold, it is determined that it may suffer DDoS attacks.

Second, it is necessary to analyze the types of DDoS you are experiencing. This document classifies attacks into DDoS with random source IP addresses and DDoS with fixed source IP addresses based on whether the source IP address is randomly generated. For DDoS with random source IP addresses, because of the large number of ipv4 addresses, the randomly generated ip address has a low repetition rate. By calculating a fixed number of packets, the source IP address repetition rate and the average number of source IP addresses can be used to determine whether Suffered the attack. For DDoS with fixed source IP addresses, multiple parameters in a certain number of packets need to be collected, such as source IP change rate, source port change rate, and average packet size. Based on this, you can further determine which source IP addresses are attacks. source.

Finally, filter interception is required for the detected DDoS traffic. This article divides the filtering scheme into two types based on two different types of DDoS. For a DDoS with a random source IP address, all packets received from the interface will be directly discarded. When the traffic is normal, traffic will be released. For a DDoS with a fixed source IP address, packets from the detected abnormal source IP address will be discarded. The traffic will be released when the traffic is normal.

Key words：Software-defined network SDN OpenFlow DDoS Distributed Denial of Service detection protection

1. 绪论
   1. 课题研究背景与研究现状

在传统的网络架构中，数据平面和控制平面是紧耦合的，分布式的。扁平状的网络架构使得网络的控制节点分布在每一台网络设备(路由器/交换机)中。尽管如OSPF、EIGRP等动态路由协议可以动态学习路由，减少了手工配置转发规则的配置量，但这样的动态学习其实是不可控的。随着网络规模的扩大，网络的管理和维护将变得越来越困难，代价也越发的高昂。而且网络遇到变动或者遭到攻击的时候，不能快速感知并及时作出调整，以至于收敛时间过长而导致网络服务中断。

可编程能力弱则是传统网络的另一缺陷。尽管如Cisco、华为等网络设备供应商都提供了一套独立且功能丰富的接口来满足当前网络的需要，但随着网络功能需求的增长，这样的模式已经使得网络变得越来越臃肿，难以灵活管理。而且新协议的开发需要时间，这就导致网络功能的扩展受了到极大限制。

软件定义网络(SDN)的出现和发展，在某种程度上解决了上述传统网络的问题。与传统网络相比，SDN的数据平面和控制平面是分离的。控制平面从传统的网络设备中剥离出来，集成到可以纵观全网的控制器。这种控制平面集中的模式，能够更加灵活的调度网络流量。SDN的可编程性使得网络功能能够得到快速部署，极大的提高了网络的可扩展能力。

尽管与传统网络相比，SDN有着巨大的优势，但是SDN本身也存在一定的问题。例如SDN控制器的接口还未标准化，这就导致各大控制器厂商之间的设备不能很好的兼容。控制平面集中导致单点问题严重，如果控制器被打死，那么整个网络将陷入瘫痪，这也导致控制器成为DDoS攻击的主要目标。SDN的受攻击平面广，SDN的三层架构(应用层，控制层，数据层)中任何一层都有遭受DDoS攻击的可能。

无论是在传统网络中还是SDN网络中，安全问题一直都是研究的热点。其中分布式拒绝服务(DDoS)攻击则是最严重的的安全威胁之一。随着云计算技术的不断发展，DDoS破坏性也呈几何级增长。内容分发网络(CDN)提供商Akamai调查发现DDoS攻击的年增长率高达125%。

在传统网络中，根据攻击的TCP/IP层次不同，DDoS攻击主要有两种类型，分别是以网络为中心的DDoS攻击和应用层为中心的DDoS攻击。以网络为中心的DDoS攻击主要目标是网络设备和带宽，通过海量的垃圾数据打满目标带宽，导致正常流量堵塞，达到使目标对外提供的服务中断的目的。以应用层为中心的攻击主要目标是目标服务器，通过不断的发送垃圾请求来消耗服务器的资源，导致服务器对外提供的服务性能受到巨大影响。

而在SDN网络中，攻击者可以针对SDN的三个不同层面进行攻击。针对控制层的攻击有攻击时间短，流量大的特点，在短时间内使控制器失效，从而导致控制器控制的网络区域瘫痪。针对数据层的攻击有流量低，持续时间长，难以检测区分等特点，攻击者往往控制傀儡主机伪装成合法用户，使用低速的流量慢慢消耗目标主机资源。

* 1. 课题研究意义

SDN自从2009年被提出以来，短短10年不到的时间，就迅速得到业界的认可，即使是Cisco这样的传统网络设备供应商中的佼佼者也不得不去拥抱SDN。国内外的芯片厂商也推出了支持OpenFlow的芯片。使得SDN商业化得到了积极推进。随着时代向智能化的方向发展，全球接入互联网的设备急剧增多，传统的网络架构已经很难适应当前的网络需求。随着国内各大企业的业务逐渐向云上迁移，对于SDN的研究和完善必然会如火如荼的进行。

虽然SDN得到了众多IT企业巨头的青睐，几乎代表了网络未来发展的趋势，但是SDN的安全体系还处于初级阶段，对于DDoS这种老牌的攻击手段还没有一个很完善的解决方案。各大云计算厂商所采取的解决方案主要有路由黑洞和流量清洗两种，然而都是治标不治本，前者会误杀正常流量，后者的代价又过于高昂。所以，如何在SDN网络中有效的抑制DDoS攻击是网络安全领域的一个重点研究课题。

本次毕业设计的重点有两个方面，分别是SDN这种新型的网络架构思想以及在这种新型网络架构中的DDoS攻击的检测和防护。通过开发DDoS监控程序，部署在SDN网络中，能够根据不同类型的DDoS攻击，实时的监控并分析异常流量，并将分析的结果以web页面的方式呈现出来。程序主体使用Python编写，用户可以在页面中查看当前网络中的实时流量情况，可根据业务情况动态调整报警阈值。

* 1. 论文的结构安排

论文主要章节内容安排如下：

第1章：绪论，分析介绍SDN网络中DDoS攻击的研究背景、国内外研究现状以及研究的意义。

第2章：分析介绍SDN网络架构，归纳总结DDoS的不同攻击方式以及现有的检测和防护方案。

第3章：介绍基于流量监控的源IP攻击检测与防护方案，给出算法流程图。

第4章：仿真实验，简要介绍实验过程中使用到的工具软件，详细分析基本的仿真环境搭建、仿真过程以及最终的仿真结果。

第5章：总结全文，展望未来。

1. 软件定义网络(SDN)与DDoS攻击
   1. 软件定义网络

软件定义网络(SDN)最早诞生于斯坦福大学的Clean Slate课题。相比于传统网络，主要有两个方面的改进，一是网络的数据平面和控制平面分离，二是网络可编程。通过控制器的集中控制和网络可编程，可以大大提高网络功能部署的灵活性。本节主要介绍SDN的模型架构以及相比传统网络，SDN的优势。

* + 1. SDN的模型架构

SDN的基本模型架构如图2.1所示。从上到下共有三层，分别是业务层(应用层)、控制层、转发层(基础设施层)。







业务应用

业务层

API API API



网络服务

SDN Controller

控制层

控制—转发通信接口

网络设备

网络设备

网络设备

网络设备

网络设备

转发层

图2.1 SDN基本架构

1. 业务层：业务层也称应用层，不同的业务通过控制层提供的北向API与控制层进行通信，从而使得网络管理人员能够根据所管理的网络的需要灵活的控制网络流量。在传统网络中，由于网络功能都是由网络设备供应商进行开发并集成到各自的网络设备中的，所以在传统网络中，网络功能的可扩展性很差，成本高。而在SDN中，由于控制层提供了足够的API供开发调用，使得网络功能的扩展变得十分方便。而且由于这些API是与平台无关的，并不依赖于特定的硬件设备，所以网络功能的可移植性很高，大大推进了计算机网络的发展。
2. 控制层：控制层在逻辑上集中，是SDN的核心组成部分。SDN的核心之一就是控制平面与数据平面解耦。在传统网络中，其实控制层和数据层已经是分离的，但是在物理上两者依然是封装在一个硬件设备中，这就导致网络扩展变得很不方便。在SDN中，控制层面从转发设备中完全或者部分抽取出来，在远端单独部署，使得控制层与数据层分离，网络结构更加清晰。另外，SDN网络中，底层转发设备(路由器、交换机)不需要关心邻居是谁，不需要通过OSPF等动态路由协议建立邻居关系，它们仅仅具有数据转发功能，转发行为由远端的控制器通过下发流表进行控制。控制层通过控制—转发通信接口(南向接口，主要是OpenFlow)与底层网络硬件转发设备通信，通过北向API与上层应用进行通信，目前主流的控制器：Cisco等传统网络设备供应商联合开发的开源控制器OpenDayLight、ONF主导的开源控制器ONOS、斯坦福大学提出的第一款控制器NOX以及其Python重写的衍生版POX、FloodLight、Ryu、Beacon等。
3. 转发层：转发层也叫基础设施层。是SDN三层结构的最底层，由具有转发功能的硬件或者软件组成。该层的设备仅仅具有转发功能，不需要理解网络通信协议。转发层设备根据控制器下发的流表规则处理数据流。
   * 1. SDN的优势
4. 数据控制平面分离：控制平面拥有全网的拓扑信息以及设备状态信息，可以灵活控制网络流量。
5. 网络可编程：随着网络规模的扩大，网络功能的不断完善，企业业务逻辑也变得更加复杂多样，频繁的业务变动使得传统网络越来越难以满足企业需求，SDN开放的控制平面屏蔽了底层转发设备的差异，可以灵活智能的部署网络应用。
   1. 分布式拒绝服务攻击

分布式拒绝服务攻击是当前网络安全领域中比较突出的问题。攻击者通过病毒程序感染控制大量主机。被感染的主机称为僵尸主机，由僵尸主机组成的网络称为僵尸网络。然后由攻击者发送攻击指令，被控制的僵尸主机向被攻击的服务器发送大量非法请求，从而达到消耗目标服务器资源，使得目标所提供的服务中断的目的。

* + 1. DDoS的攻击类型

由于DDoS攻击的目的是消耗目标资源，中断目标提供的服务，所以根据攻击消耗的资源类型可以将攻击分为以下几种类型

1. 带宽资源消耗型攻击：

该攻击的目的是打满目标主机的带宽，使得目标主机带宽资源耗尽，无法对外提供正常服务。通常有直接攻击和反射型攻击两种。直接攻击是攻击者控制大量僵尸主机对目标直接发起大流量攻击，利用带宽资源的不对等，耗尽目标带宽，但是此种攻击不易隐藏攻击源。反射攻击是攻击者伪装成被攻击服务器的IP，向网络中的路由器、DNS服务器等设备发送大量请求数据，当这些设备收到请求数据时，会将响应数据发送给被攻击的服务器，使得被攻击服务器被大量响应数据淹没，服务中断。反射型攻击相比直接攻击，不但能够隐藏攻击者的真实地址，还能够对攻击流量进行放大，因为响应数据通常要比请求数据大的多，所以攻击者只需要发送少量的请求流量，就可以让被攻击者遭到巨大响应流量的攻击。

1. 系统资源消耗型攻击：

该攻击的目的是打满目标主机的系统资源，包括CPU、内存、磁盘IO等。比较常见的系统资源消耗型攻击是SYN FLOOD攻击。正常情况下客户端向服务端发送SYN包，客户端进入SYN\_SEND状态；服务端收到客户端的SYN包后，会回复一个SYN+ACK包，同时进入SYN\_RECV状态；客户端收到服务端的回复后，向服务端发送ACK报文确认，客户端服务端都进入ESTABLISHED状态，至此正常连接建立完成。在SYN FLOOD攻击中，攻击者控制大量僵尸主机在短时间内向服务器发送大量SYN包请求建立TCP连接，而服务器能够维护的连接数是有限的，这就导致正常的请求连接无法建立。

1. 应用服务资源消耗型攻击：

该攻击利用服务器处理请求能力有上限，通过发送需要消耗大量资源的请求来耗尽服务器资源，从而达到中断服务的目的。例如攻击者控制大量僵尸主机向被攻击服务器发送大量数据查询的报文，服务器每处理一个查询请求，就需要对后台数据库做一次查找，当大量的垃圾请求占满数据库资源时，将导致后台数据库宕机。

* + 1. DDoS的危害性

随着IT行业的发展，资源性能的不断提供，尤其是云计算时代的到来，使得DDoS攻击变得更加凶猛，危害更大。在公有云中，攻击者可以在短时间内创建出庞大的高资源僵尸网络，对攻击对象发起超大流量的DDoS攻击，对企业生产造成巨大破坏。就连技术实力雄厚的GitHub，也遭到了有史以来最严重的DDoS攻击，峰值流量达到了1.35Tbps。

* + 1. 已有的检测和防护算法

由于DDoS攻击的巨大破坏性，业界已经提出了很多检测与防护的解决方案。

论文[]根据流包数中位值、流字节数中位值、对流比、端口增速、源IP增速在内的5个流量特征，采用优化的KNN算法对所选的流量特征进行异常检测，识别率较高，误报率较低。

论文[]提出基于熵值算法的DDoS检测方法。在SDN的控制层使用该算法判断攻击，在转发层处理流表，结合ACL和流量管理，使用OpenFlow协议实现策略，提高了检测率，降低了误报率，能够快速响应。

论文[]采用SDN+BGP FLOWSPEC技术，精确控制丢弃异常攻击流量，保护正常流量正常转发，而且成本低廉，性能高。

1. 基于流量监控的源IP攻击检测与防护方案
   1. 方案模型

方案模型如图3.1所示。

流量方向

转发层

检测层

物理层

流量转发

流量过滤

流量特征分析

流量阈值检测

网络接口

图3.1 检测防护方案模型

本方案分为三个层次，共有三个核心模块，分别是流量阈值检测模块、流量特征分析模块和流量过滤模块。数据流进入网络接口，经过流量阈值检测模块，出现异常流量则进入流量特征分析模块，再通过流量过滤模块过滤异常流量，最后转发正常流量。

* + 1. 流量阈值检测模块

该模块主要用于流量检测，即实时监测每个OpenFlow交换机的接口流量。流量参数包括入接口字节速率(单位KB/s)、出接口字节速率(单位KB/s)，入接口包速率(pkgs/s)，出接口包速率(pkgs/s)，也是流量阈值检测模块中的四个阈值。之所以包含包速率作为流量参数是因为DDoS攻击中可能存在数据包长度很小但是数据包数量很多的情况。正常情况下，交换机接口的流量应该处于稳定状态，当遇到大规模DDoS攻击的时候，交换机接口流量将趋于饱和。

流量阈值可以自行设定，设定的阈值应该高于流量高峰期时的数值，避免由于误判而导致流量误杀。当接口流量大于设定阈值时，则初步判断可能遭到DDoS攻击，然后立刻进入流量特征分析模块进行流量特征分析。

* + 1. 流量特征分析模块

当流量阈值检测模块检测到异常流量时，将会进入该模块。该模块会启动抓包程序，通过抓取流经异常流量接口的指定数量的数据包，记录下抓包时间，分析包括源IP地址、目的IP地址、源端口、目的端口、数据包长度以及它们相应的统计数据，统计数据以源IP地址为键，其它元素为值，以键值对的形式保存。保证所存的数据中包含出现的所有源IP地址以及其对应的每一个目的IP地址、源端口、目的端口、数据包长度。统计数据数据结构如下：

{

sip1 : {

counts : xxx,

length : [length1, length2, length3,…],

dip : [dip1, dip2, dip3,…],

sports : [sport1, sport2, sport3,…],

dports : [dport1, dport2, dport3,…]

}

sip2 : {

…

}

}

根据统计数据，就可以判断DDoS攻击流量的源地址是否是随机生成的。

首先判断是否是随机源IP地址的DDoS攻击。众所周知，ipv4地址的数量有42亿之多，当攻击者利用随机生成的IP地址进行攻击的时候，IP地址的重复率会很低，同时源IP地址出现次数的平均值也会很低。由于抓取的数据包个数恒定，所以只要不同源IP地址个数占总的数据包个数的比例超过指定阈值，并且源IP地址出现次数的平均值低于指定阈值，则可判断正在遭受随机源IP的DDoS攻击。

如果不是随机源IP的DDoS攻击，则需要根据其它参数进行判断是否是固定源IP的DDoS攻击，可分为以下几种场景：

1. 攻击者针对某个特定主机进行攻击。
2. 攻击者针对多个主机进行攻击。
3. 攻击者针对某个主机的某个进程进行攻击。
4. 攻击者针对某个主机的多个进程进行攻击。
5. 攻击者针对多个主机的某个进程进行攻击。
6. 攻击者针对多个主机的多个进程进行攻击。

针对每个源IP地址所需参数如下：

1. 数据包平均长度

数据包平均长度指的是相同源IP地址的数据包的平均大小。如果数据包平均长度低于或者高于正常阈值，则可判断该源IP为异常IP。

1. 目的IP变化速率

目的IP变化速率指的是相同源IP地址的数据包，每秒目的IP地址去重之后的数量。如果该参数高于正常阈值，则可判断该源IP为异常IP。

1. 目的IP最大速率

目的IP最大速率指的是相同源IP地址的数据包，每秒目的IP地址重复出现次数中的最大值。如果该参数高于正常阈值，则可判断该源IP为异常IP。

1. 源端口变化速率

源端口变化速率指的是相同源IP地址的数据包，每秒源端口去重之后的数量。如果该参数高于正常阈值，则可判断该源IP为异常IP。

1. 源端口最大速率

源端口最大速率指的是相同源IP地址的数据包，每秒源端口重复出现次数中的最大值。如果该参数高于正常阈值，则可判断该源IP为异常IP。

1. 目的端口变化速率

目的端口变化速率指的是相同源IP地址的数据包，每秒目的端口去重之后的数量。如果该参数高于正常阈值，则可判断该源IP为异常IP。

1. 目的端口最大速率

目的端口最大速率指的是相同源IP地址的数据包，每秒目的端口重复出现次数中的最大值。如果该参数高于正常阈值，则可判断该源IP为异常IP。

七个参数共同参与到攻击流量检测中，可针对上述6种攻击场景进行攻击流量的检测。提高了识别的准确性，降低了误判率。七个阈值可根据现实环境动态调整，更具有普适性。

* + 1. 流量过滤模块

流量过滤模块接收来自流量特征分析模块的流量。根据DDoS的攻击方式，有两种过滤方式：

1. 针对随机源IP的DDoS攻击。由于攻击流量的源IP是伪造的，而且伪造的源IP地址中可能存在合法地址，所以根据源IP地址进行拦截意义不大。针对此种攻击方式本模块将会采取一刀切的方案，当交换机接口遭到随机源IP的DDoS攻击时，将会丢弃所有来自该接口的流量，直到该接口流量恢复正常的时候再放行。
2. 针对固定源IP的DDoS攻击。由于攻击流量的源IP是真实的，所以可以根据源IP地址进行拦截。当交换机接口遭到固定源IP地址的DDoS攻击时，本模块将会在交换机入接口丢弃检测到的异常IP地址并将其加入到黑名单，直到该IP地址的流量恢复正常时再放行。
   1. 流程图

流程图如图3.2所示：

开始

网卡流量监控

网卡流量异常

启动抓包程序

分析抓包文件

源IP地址随机

丢弃所有

源IP异常

丢弃异常IP

放行

是

否

是

否

是

否

图3.2DDoS攻击检测与防护流程图

1. 仿真实验
   1. 仿真环境与工具介绍
      1. Ubuntu

Ubuntu是基于Debian基本的架构的一个Linux操作系统，是Debian的衍生版本，也是目前主流Linux发行版之一。

* + 1. mininet仿真工具

mininet是一款强大的网络仿真工具。可以在个人PC、云主机、虚拟机等环境中方便快捷的创建一个真实的虚拟网络。只需要在CLI中执行mn命令，即可快速创建一个简单虚拟网络。同时mn命令后面可接多个参数，用于自定义拓扑、选择要连接的控制器等。

正由于其强大的网络仿真能力，Mininet对于开发，教学和研究有帮助。

* + 1. OpenDayLight

OpenDayLight(ODL)是由Linux基金会主导管理的开源项目，是一款高可用，模块化，可扩展的多协议控制器基础架构，专为在现代异构多供应商网络上部署SDN而构建，是推动SDN技术创新发展的重要力量。它提供了一个模型驱动的服务抽象平台，允许用户编写可轻松应用于各种硬件和南方协议的应用程序。整个项目使用Java编程语言进行开发，所以安装时候需要安装Java开发环境。

* + 1. 云主机

云主机是云计算基础设施层的重要组成部分，通过网络，以按需使用和付费的形式租赁给客户。用户可以根据自身需要，购买相应的计算资源、网络资源以及存储资源。

* + 1. python模块

本节简要介绍本论文仿真实验中将要用到的python模块。

1. psutil模块

psutil模块是一个第三方模块，主要用于获取系统信息，例如CPU状态、内存使用率、磁盘IO、网络IO等。

psutil.net\_io\_counters( pernic = True )是该模块提供的一个获取网络信息的方法，网络信息包括总的发送字节数bytes\_sent，总的接收字节数bytes\_recv，总的数据包发送数packets\_sent，总的数据包接收数packets\_recv等。pernic参数表示是否获取所有网卡的网络信息。

1. dpkt模块

dpkt是一个第三方模块，用于快速，简单的数据包创建/解析的python模块。该模块提供的一些方法可以方便的解析pcap抓包文件内容，获取抓包文件中的源IP地址，目的IP地址，源端口和目的端口等一系列数据。

1. os模块

该模块是一个python自带的系统模块，提供了大量的方法来处理文件和目录。其中os.system(‘command’)方法会执行引号中的命令，该命令将在shell中进行。通过该方法，可以方便的调用Linux下的tcpdump抓包工具，也可以调用OVS提供的ovs-ofctl命令下发流表。

* + 1. sflow网络监测工具

sflow是一款网络流量检测工具，采用数据流随机采样技术，能够通过web界面呈现全网的流量信息，是网络流量分析的重要工具。sflow是C/S架构，服务端的默认监听端口号是6343和8008，6343端口是agent的连接端口，8008端口是浏览器连接端口，用于web界面展示。

* + 1. hping3

hping3是一款可以发送自定义TCP/IP包的网络工具。主要用于防火墙规则测试、端口扫描、根据不同的协议，数据包大小，TOS和分片来进行网络性能测试、远程操作系统指纹识别、TCP/IP堆栈审计等。在本论文中用于模拟DDoS攻击。hping3用法：hping3 host [options]，其中options可选项主要用到的有以下几个：

--flood 表示尽可能快的发送数据包，并且没有回显。

--rand-source 表示发送数据包的时候，源地址是随机生成的。

* 1. 环境搭建
     1. Ubuntu环境部署

创建两台云主机，操作系统均为Ubuntu 16.04。其中一台云主机运行mininet仿真工具，另一台作为OpenDayLight控制器，如图4.1所示。



图4.1 云主机

* + 1. OpenDayLight环境部署

首先需要安装jdk作为其运行环境。Ubuntu命令行下执行命令：apt-get install openjdk-8-jdk，执行完成后可使用java -version和javac –version命令查看，如图4.2所示即表示安装成功。

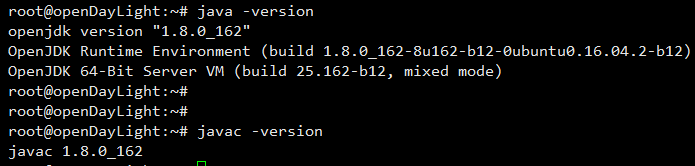


图4.2 jdk安装成功验证图

运行环境安装完成后，即可开始安装OpenDayLight，安装步骤如下：

1. 从OpenDayLight官网下载OpenDayLight的压缩包并解压，本论文使用的版本是karaf-0.6.3-Carbon。
2. 如图4.3所示，将org.apache.karaf.management.cfg配置文件中的rmiRegistryHost和rmiServerHost项的值改为127.0.0.1

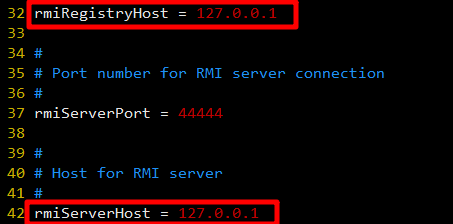


图4.3 opendaylight配置修改图

1. 如图4.4所示，在setenv文件结尾设置启动OpenDayLight时的环境变量。

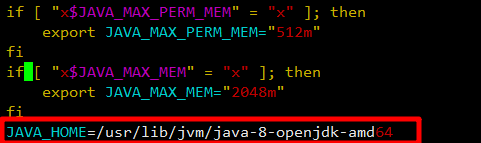


图4.4 setenv文件设置图

此时，OpenDayLight安装完成，但是要想使用其提供的功能，例如通过浏览器访问web界面，则还需要安装OpenDayLight的一些必要特性。安装步骤如下：

1. 启动OpenDayLight

在OpenDayLight安装目录下有bin目录，执行bin目录下的karaf启动程序，如图4.5所示：

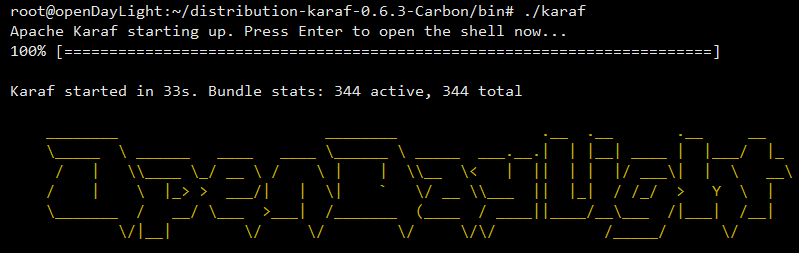


图4.5 OpenDayLight启动界面

1. 安装feature，安装命令为feature:install FEATURE1 FEATURE2…

按如下顺序安装feature：odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-core odl-openflowplugin-flow-services-ui

至此，OpenDayLight环境部署完成。可通过netstat –tanl | grep PORT命令查看相应端口是否正在监听，如图4.6所示。默认的控制器监听端口是6633，即OpenDayLight控制器通过6633端口与远程OpenFlow交换机进行通信，浏览器访问web界面的默认端口是8080，即在浏览器地址栏输入IP:8181/index.html，即可看到OpenDayLight的web登陆页面，默认用户名密码都是admin，如图4.7所示。

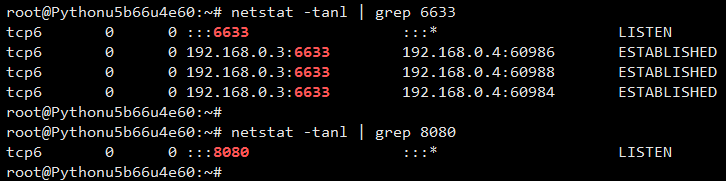


图4.6 监听端口验证图

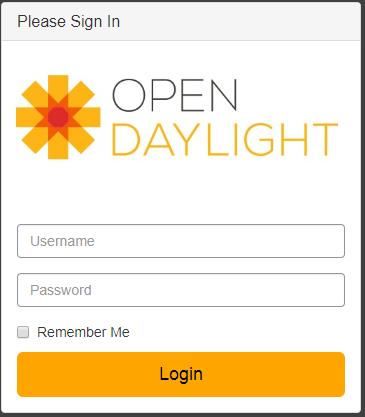


图4.7 OpenDayLight登陆界面

* + 1. mininet环境部署

mininet是基于Python的网络仿真工具。所以需要安装Python以及相应的开发库。Ubuntu命令行下执行命令：apt-get install python3.5 python3.5-dev。安装完成后开始安装mininet，安装步骤如下：

1. 执行命令git clone git://github.com/mininet/mininet从github网站获取mininet的源代码。
2. 进入mininet目录，如图4.8所示，使用git tag命令列出可安装的mininet版本。可通过git checkout命令选择要安装的mininet版本。

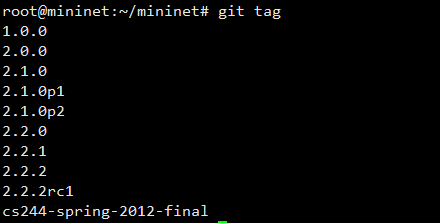


图4.8 mininet可用版本

1. mininet/util/目录下有一个install.sh的shell脚本文件，执行该脚本文件。后面可接-a参数表示安装所有内容，包括Open vSwitch等依赖项以及OpenFlow wireshark解析器等附加组件。
2. 如图4.9所示，命令行执行命令mn --version查看是否安装成功。



图4.9 mininet成功安装验证图

* + 1. 其它安装项

1. 安装sflow：在OpenDayLight云主机上，从sflow官网下载sflow-rt压缩文件并解压缩即可。
2. 安装hping3：Ubuntu命令行下执行命令apt-get install hping3
   1. 仿真过程

如上文所述，首先启动启动OpenDayLight控制器，查看相应端口是否监听。

* + 1. 网络拓扑创建

1. 网络拓扑如图4.10所示

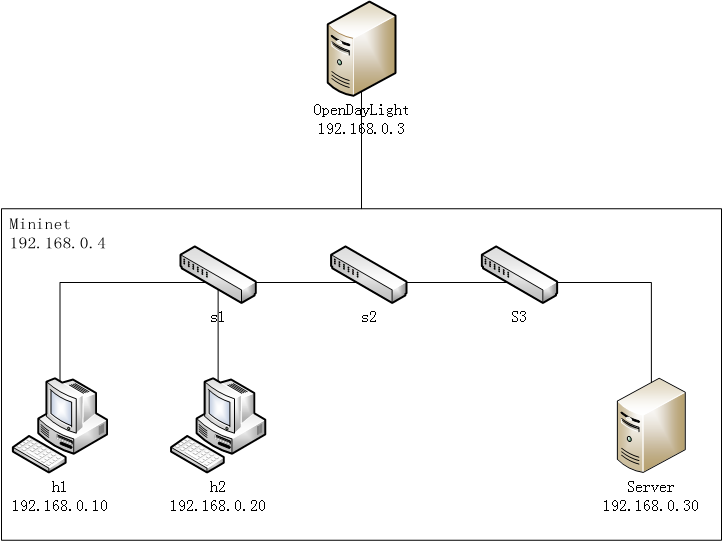


图4.10 仿真拓扑

在拓扑中，h1模拟攻击者，h2模拟合法用户，Server模拟受害服务器，它们通过三台OVS交换机相互通信，都由mininet仿真工具仿真。OpenDayLight通过云主机内网与mininet主机相连，作为该网络的SDN控制器。

1. 命令行执行命令mn --custom ~/topo/test1.py --topo=mytopo --controller=remote,ip=192.168.0.3,port=6633

--custom参数表示从python脚本中读取自定义拓扑的内容。~/topo/test1.py表示待创建拓扑的python脚本程序路径，--topo=mytopo用于指定SDN网络的拓扑，--controller表示要连接哪个SDN控制器，所连控制器的ip和端口需要指定。执行完该命令后进入mininet命令行界面，可执行links命令查看网络设备连接情况，如图4.11所示。

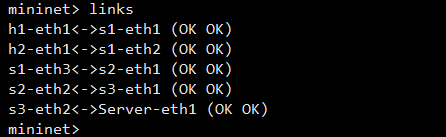


图4.11 网络设备连接状态

1. 登陆OpenDayLight提供的web控制台查看拓扑是否正常显示。

如图4.12所示，此时会发现界面中只显示了三台openflow交换机，终端主机并未显示，这是因为主机之间还没有流量通过，openDayLight无法感知主机存在，只需要在mininet命令行中执行pingall命令，再重新加载即可显示完整拓扑，如图4.13所示，h1的MAC地址为00:00:00:00:00:01，IP地址为192.168.0.10，h2的MAC地址为00:00:00:00:00:02，IP地址为192.168.0.20，Server的MAC地址为00:00:00:00:00:03，IP地址为192.168.0.30。

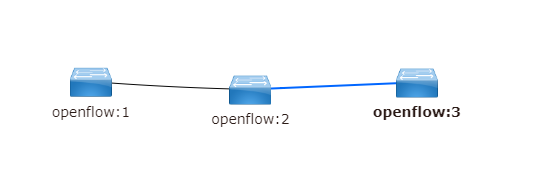


图4.12 OpenDayLight呈现拓扑(1)

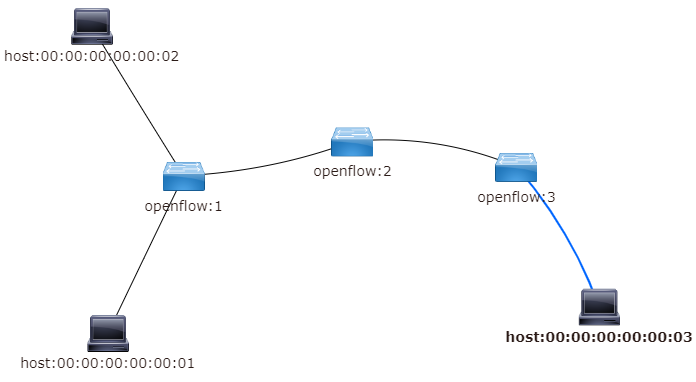


图4.13 OpenDayLight呈现拓扑(2)

* + 1. 开启sflow

1. 启动sflow-rt服务端。进入sflow-rt安装目录，在Linux命令行中执行命令./start.sh启动sflow-rt。
2. 开启OVS的sflow功能，配置sflow客户端。mininet的Linux命令行中执行命令ovs-vsctl -- --id=@sflow create sFlow agent=s1 target=\”192.168.0.3:6343\” header=128 sampling=64 polling=1 -- set bridge s1 sflow=@sflow。执行完成后，会生成一个UUID。如图4.14所示，可通过ovs-vsctl list sflow查看已经配置好的Agent信息，agent端是监听在s1上的。

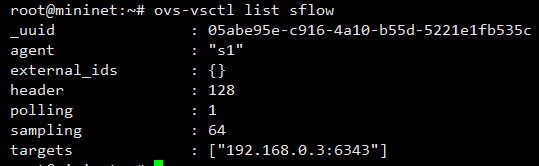


图4.14 sflow的agent配置信息

1. 使用ip link show INTERFACE命令查看交换机接口与接口编号的对应关系，如图4.15所示，127表示s1-eth1接口在sflow的web界面显示的编号：

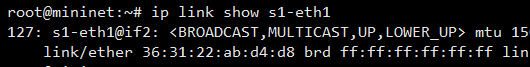


图4.15 接口对应编号

1. 浏览器访问sflow的web界面，如图4.16所示，点击192.168.0.4一行，出现该agent的所有监控项目，如图4.17所示，点击127.ifinpkts，出现监控图像，如图4.18所示，表示每秒流入s1-eth1接口的数据包个数。

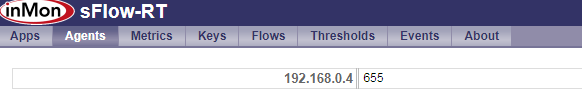


图4.16 sflow初始界面



图4.17 sflow监控项

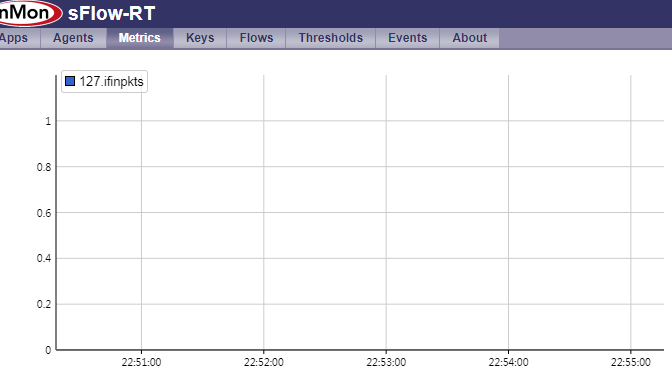


图4.18 sflow流量监控

* + 1. 网络连通性测试

1. h1与h2连通性测试，如图4.19所示：

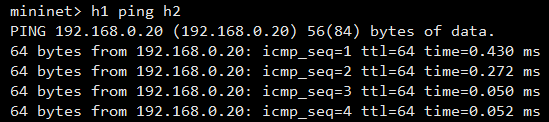


图4.19 初始h1与h2连通性

1. h1与Server连通性测试，如图4.20所示：

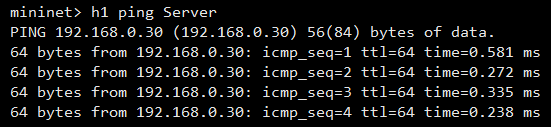


图4.20 初始h1与Server连通性

1. h2 与Server连通性测试，如图4.21所示：

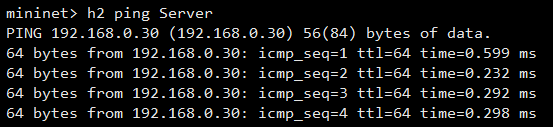


图4.21 初始h2与Server连通性

初始网络状态正常，h1 h2 Server之间可以正常通信。查看sflow的监控状态，如图4.22所示，可以看到有流量通过。

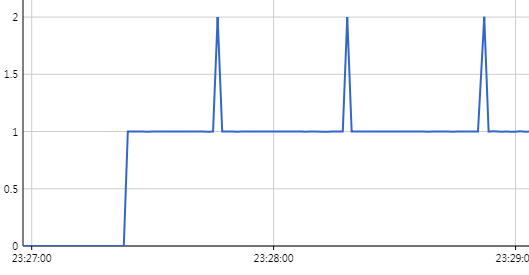


图4.22 sflow测试流量监控

* + 1. DDoS攻击模拟

1. 固定源IP

h1作为攻击源，如图4.23所示，在mininet命令行执行命令h1 hping3 --flood Server。查看sflow的web监控界面，可看到h1每秒发送的数据包个数达到了200k，如图4.24所示。

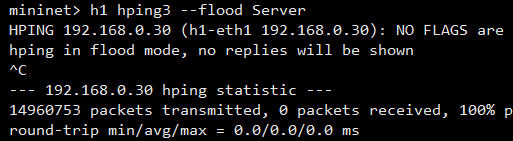


图4.23 固定源IP DDoS攻击



图4.24 固定源IP DDoS流量监控

1. 随机源IP

h1作为攻击源，如图4.25所示，执行命令h1 hping3 --flood --rand-source Server。查看sflow的web监控界面，可看到h1每秒发送的数据包个数达到了200k，如图4.26所示。

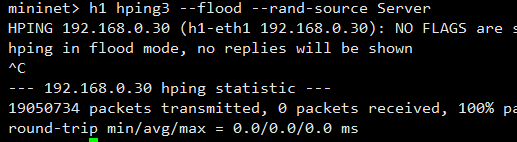


图4.25 随机源IP DDoS攻击

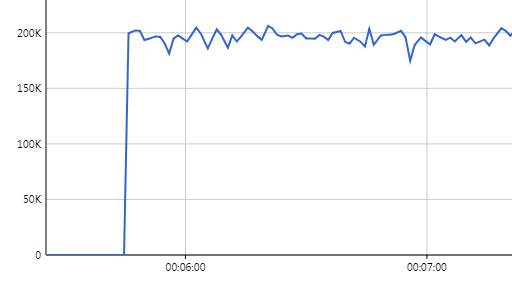


图4.26 随机源IP DDoS流量监控

* + 1. 流量检测与防护

1. 运行流量检测与防护的python脚本程序。

mininet的Linux命令行中执行命令python3 defend\_ddos\_sdn.py。defend\_ddos\_sdn.py是一个DDoS攻击流量检测与防护脚本。是3.2.1节中算法的python实现。

1. h1对Server进行固定源IP地址的DDoS攻击

在mininet命令行中执行命令h1 hping3 --flood Server，查看sflow监控结果如图4.27所示。

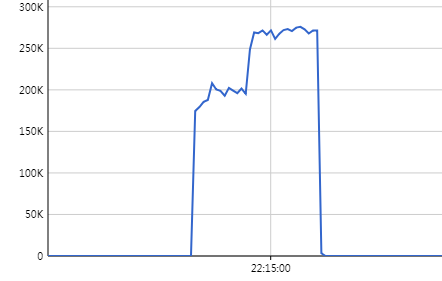


图4.27 固定源IP DDoS流量监控

使用ping命令测试h1与Server之间连通性，结果如图4.28所示，h1无法访问Server，表明攻击流量被拦截。使用ping命令测试h2与Server之间连通性，截图如图4.29所示，h2可以正常访问Server，表明合法流量正常通过。

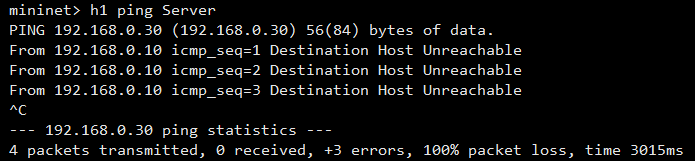


图4.28 h1与Server连通性

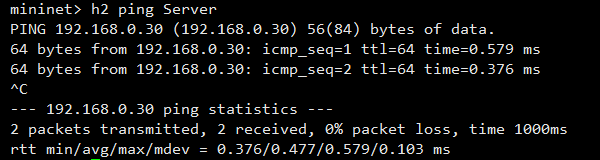


图4.29 h2与Server连通性

1. h1对Server进行随机源IP地址的DDoS攻击

在mininet命令行中执行命令h1 hping3 --flood --rand-source Server，查看sflow监控结果如图4.30所示。

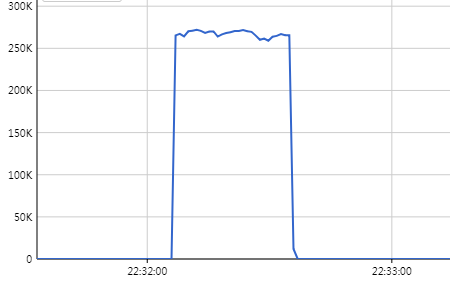


图4.30 随机源IP DDoS流量监控

使用ping命令测试h1与Server之间连通性，结果如图4.31所示，h1无法访问Server，表明攻击流量被拦截。使用ping命令测试h2与Server之间连通性，截图如图4.32所示，h2可以正常访问Server，表明合法流量正常通过。

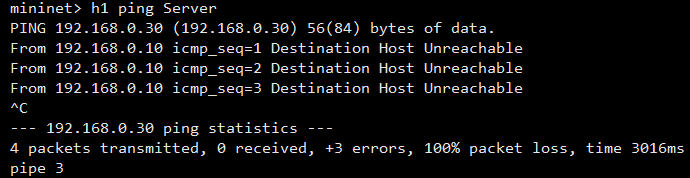


图4.31 h1与Server连通性

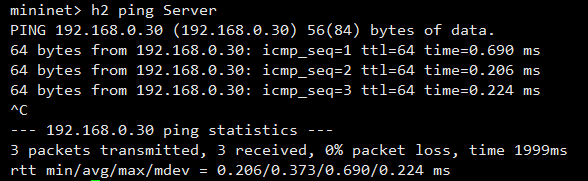


图4.32 h2与Server连通性

* 1. 功能模块介绍
     1. 拓扑创建功能模块

为了能够方便的自定义拓扑，mininet提供了一个供python调用的mininet模块，通过该模块，可以灵活添加主机、OpenFlow交换机等设备，自由构建网络拓扑。在python脚本中，通过from mininet.topo import Topo代码导入该模块。其中Topo是一个类，该类提供了多个方法来构建网络拓扑，主要用到以下几个方法。

1. addHost( hostname, ip = ’IP’ )

该方法返回一个主机对象，用于在拓扑中添加一个终端主机。可接收多个参数，hostname是主机名，ip如果不指定，则默认为10.0.0.0/8网络。

1. addSwitch( switchname )

该方法返回一个交换机对象，用于在拓扑中添加一个OpenFlow交换机。switchname是交换机名。

1. addLink( host1, switch1, PORT1, PORT2 )

该方法用于连接主机和交换机。其中host1是addHost方法返回的主机对象，switch1是addSwitch方法返回的交换机对象，PORT1和PORT2表示host1的port1端口与switch1的port2端口相连。

* + 1. 流量检测与防护模块

本节将简要介绍4.3.5节中提到的DDoS攻击流量检测与防护脚本，就其中一些核心函数作简要分析介绍：

1. nio():

该函数调用psutil模块中的net\_io\_counters方法，返回一个保存有网络信息的字典。字典结构如下：

{网卡接口1 : sentio( bytes\_sent=0, bytes\_recv=0, packets\_sent=0, packets\_recv=0, errin=0, errout=0, dropin=0, dropout=0 ), 网卡接口2 : sentio( … ), … }

1. monite( nio ):

该函数接收nio函数作为参数，调用nio()，根据nio()返回的数据，计算每个网卡的下行速率kB/s上行速率kB/s，下行包速率pkgs/s，上行包速率pkgs/s，以字典的形式返回流量异常的接口以及其对应的流量。字典结构如下：

{ 网卡接口1 : { 入接口字节速率 : 速率值, 入接口包速率 : 速率值, 出接口字节速率 : 速率值, 出接口包速率 : 速率值 }, 网卡接口2 : { … } }

1. capt( except\_if ):

该函数接收异常流量接口的列表，针对每个异常接口调用tcpdump分别抓包，将抓取的数据包保存为文件，以字典形式返回抓包文件名和抓包时长。字典结构如下：

{ { 异常接口1 : { 抓包文件名 : '2018-05-01-21:40:46-lo', 抓包时长 : 0.5504086017608643 }, 异常接口2 : {…}, …}

1. anlyPcap(pcap):

该函数调用dpkt模块解析pcap文件，以字典的形式返回解析出的内容。字典形式如下：

{

sip1 : {

counts : xxx,

length : [length1, length2, length3,…],

dip : [dip1, dip2, dip3,…],

sports : [sport1, sport2, sport3,…],

dports : [dport1, dport2, dport3,…]

}

sip2 : {

…

}

}

1. averDict(ip\_dict):

该函数接收抓包分析的结果，返回counts的平均值。

1. speed(t,list1,capt\_time):

该函数用于计算元素变化速率和元素最大速率。函数接收一个字符串，一个列表和抓包的时长作为参数。列表中是一串数字，表示在抓包过程中，抓取到的元素(dip,sports,dports)数目；字符串用于判断上述哪类速率，如果t = 'len'，则返回元素变化速率，如果t = 'max'，则返回元素最大速率。

1. rand\_estimate(num\_sip,average\_counts):

该函数用于判断是否正在遭受随机源IP的DDoS攻击，如果是，则返回True，否则返回False。num\_sip表示源IP数量，average\_counts表示counts的平均值。

1. dropAll():

该函数通过下发流表控制交换机丢弃所有数据包。

1. forwardAll():

该函数通过下发流表控制交换机转发所有数据包。

1. dropByIP(sip):
2. forwardByIP(sip):
3. addRemoveIP(ip\_dict):

该函数分析anlyPcap函数返回的结果，根据数据包平均长度、元素变化速率以及元素最大速率等7个参数 判断是否遭受固定源IP地DDoS攻击。如果是，则调用dropByIP(sip)函数丢弃源IP为sip的流量，否则调用forwardByIP(sip)函数转发源IP是sip的流量。

1. 总结与展望
   1. 总结

SDN作为未来网络发展趋势的代表，正在逐渐替代传统的网络架构，其独特的三层架构，控制平面逻辑集中，网络可编程等解决了传统网络遗留的很多问题。而DDoS攻击一直都是网络信息安全领域中十分突出的问题，如何在SDN网络中有效检测与防御DDoS攻击就显得尤为重要。本文根据DDoS的流量特征，提出的基于流量监控的源IP攻击检测方案，主要总结如下：

1. 流量阈值检测。

通过对网络接口流量的实时监控，设定流量阈值，当流量过载时可初步判断遭受DDoS攻击。仿真结果表明此方法针对大流量DDoS攻击有显著效果。

1. 流量特征分析。

首先根据源IP地址的重复率以及源IP地址数目的平均值判断遭受的是否是随机源IP地址的DDoS，再针对数据包平均长度，源目IP变化速率、源目IP最大速率、源目端口变化速率，源目端口最大速率等七个参数设定阈值，高于阈值则判断遭受的是固定源IP地址的DDoS攻击。

1. 流量过滤：

由于随机源IP攻击的不可预知性，针对源IP进行过滤没有意义，所以对于随机源IP地址的DDoS攻击采用的是一刀切的方式。对于固定源IP地址的DDoS，则是将检测到的异常IP纳入黑名单，直到检测其流量正常时再放行。

* 1. 展望

本文对于SDN网络中DDoS攻击的研究虽然取得了一些进展，但是对于该项研究还处于初级阶段，所做工作还需要进一步完善和提高。

1. 本文仅仅是针对大流量DDoS攻击下的攻击检测与防护。但是某些DDoS攻击流量具有流速低、持续时间长等特点，对于该种DDoS攻击，本文所提方案显得力不从心。虽然该种攻击难以达到拒绝服务的目的，但对网络带宽资源的消耗也不容小觑。
2. DDoS攻击是由大量僵尸主机发起的分布式拒绝服务攻击。但是由于实验条件限制，无法真实模拟一定量级的攻击场景，所以本文所提方案还有待真实网络场景的验证。

致谢

转眼进入大学已经四年，大学生涯也进入了尾声。回首这四年，失去很多，也收获很多，得失之间，是成长。在此，我要由衷的感谢一路上帮助我的师长、同学和朋友。

首先，我要感谢我毕业设计的指导老师，蒋中秋老师，感谢您在我毕业设计的过程中的悉心指导和耐心解答，及时指出我毕业设计中的错误和不足。

然后我要感谢我的同学，感谢他们在我毕业设计遇到瓶颈的时候给我提出合理化的建议。

士不可以不弘毅，任重而道远。毕业在即，前路漫漫，希望我们在同一片蓝天下，共同成长。

参考文献：

{Hyun, 2017 #5}