基于 OpenDaylight 防火墙的研究与实现

费 宁,刘春秋
(南京邮电大学 计算机学院 软件学院,江苏 南京 210003)

摘 要:随着网络规模的持续发展,传统的分布式网络已经不能满足网络配置和管理的要求,软件定义网络作为一种全新的网络架构给网络安全研究提供了新的方向。该架构将数据转发层和控制层相分离,并且在控制层之上开放了应用程序编程接口。在深入分析软件定义网络的系统原理和架构设计的基础上,提出了基于 OpenDaylight 平台的软件定义网络防火墙的实现方案 FireClient,并借助软件项目管理和依赖分析工具 Maven 和数据建模语言 Yang,开发了上层应用调用模块。FireClient 允许用户灵活修改策略,其在不同场景中的实际测试结果表明,使用基于软件定义网络的防火墙可以更为灵活的布置策略和快速实施。相比传统网络的配置和部署,软件定义网络使得第三方的快速应用开发成为可能,从而极大地推动了网络新业务的部署和拓展。

关键词:软件定义网络;OpenDaylight;防火墙;FireClient;Maven
中图分类号:TP393.1 **文献标识码**:A **文章编号**:1673-629X(2019)06-0112-04
doi:10.3969/j.issn.1673-629X.2019.06.023

Research and Implementation of Firewall Based on OpenDaylight

FEI Ning, LIU Chun-qiu
(School of Computer Science & Technology, School of Software, Nanjing University of Posts & Telecommunications, Nanjing 210003, China)

Abstract: With the continuous growth of network scale, the traditional distributed network can no longer meet the requirements of network configuration and management. As an evolutionary network framework, software defined networking (SDN) provides a new direction for network security research. This new framework separates the data forwarding plane from the control plane and also provides application programming interfaces on top of the control plane. Based on the in-depth analysis of the system principle and architecture design of SDN, we propose the implementation scheme FireClient of SDN firewall based on OpenDaylight platform, and with the help of Maven, a software project management and dependency analysis tool, and Yang, a data modeling language, we develop the upper application call module. The users can modify policies flexibly with FireClient, and its test in different scenarios shows that the use of firewalls based on SDN can arrange policies flexibly and implement quickly. Compared with the traditional network configuration and deployment, SDN makes it possible for the rapid application development of the third party, thus greatly promoting the deployment and expansion of new network services.

Key words: SDN; OpenDaylight; firewall; FireClient; Maven

0 引 言

软件定义网络(software defined network, SDN)将网络控制与数据转发分离,使用开放的、独立的应用层接口,使得通过软件集中配置和管理设备成为可能。软件定义网络作为一种全新的网络架构给网络安全研究提供了新的方向,SDN 在流转发、深度包检查、流量重定向等方面具有优势,基于 SDN 的网络安全新技术和新应用不断涌现[1-3]。不少学者提出了软件定义网络集中式架构下的安全技术框架,尝试提供更为完善的威胁分析和防御方法[4-5]。文中在深入分析软件定义网络和 OpenFlow 系统原理和架构设计的基础上,提出了基于 OpenDaylight 控制器的防火墙,并搭建了实验平台。该软件定义网络有利于新的网络应用的快速部署和实施,并且基于软件定义网络控制器的安全

收稿日期:2018-06-25 **修回日期**:2018-11-17 **网络出版时间**:2019-03-06
基金项目:国家自然科学基金(61003040);江苏省科技计划项目:未来网络前瞻性研究项目(BY2013095108);江苏省自然科学基金面上项目(BK20171447)
作者简介:费 宁(1977-),女,博士,副教授,研究方向为软件定义网络;刘春秋(1995-),男,研究方向为软件定义网络。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20190306.0907.030.html

策略切实可行。

1　基于 OpenDaylight 的安全设计与实现

1.1　OpenDaylight 的系统原理和架构设计

OpenDaylight 的软件定义网络的架构特性,使其可以方便地实现路由路径优化,大幅度降低网络维护成本,提高网络设备利用率,增加网络设备的可扩展性和稳定性,并能够解决传统网络中的传输性能、流量控制、访问控制等问题。

OpenDaylight 通过 SAL(service abstraction layer)将底层接口提供的功能封装成具体的服务提供给上层模块应用,屏蔽了多种南向协议之间的差异,为上层模块提供一致的服务,包括数据包服务、拓扑服务、流表编程服务等。

在路由转发部分,模块首先要注册 IListenDataPacket 服务,从 SAL 收到的数据包呈队列格式存放,并且将数据包 IP 目的地址放入 pendingPacketDestinations 集合中,run 线程从此集合中取出对象 IP。同时主机追踪模块解析数据包的目的主机,获取相关信息后,对交换机进行遍历,得到整个网络的拓扑结构图,再通过 Dijkstra 最短路径算法得到路由,将路径链路放到 rulesDB 中,接着将流表规则下发至路径经过的每个交换机,利用 dataPacketService. transmitDataPacket( )方法将数据包发送出去[6]。

OpenDaylight 提供了相应的应用层接口,应用程序向 OpenDaylight 发出调用指令,OpenDaylight 利用相应的接口与底层网络设备进行通信。同时,OpenDaylight 还向应用程序提供基础设施相关的功能接口,如流量监控、管理、入侵防御等。

1.2　基于 OpenDaylight 开发的防火墙模块的应用架构设计

传统网络中,处于服务器及应用程序之下的底层位置的是相应的网络设备,软件定义网络概念的提出,为集中式网络架构提供了可能性,内部网络功能也开始出现在新的层级中[7-8]。OpenDaylight 的作用就是将网络功能剥离硬件,重新放回到控制器当中,并且为 SDN 制定一套通用型框架[9]。文中设计的基于 OpenDaylight 平台的防火墙的整体流程包括:firewall 模块功能逻辑;FireClient 应用界面开发、网络请求逻辑和返回结果的数据处理;测试脚本对 firewall 模块请求接口的调用测试。整体设计流程如图 1 所示。

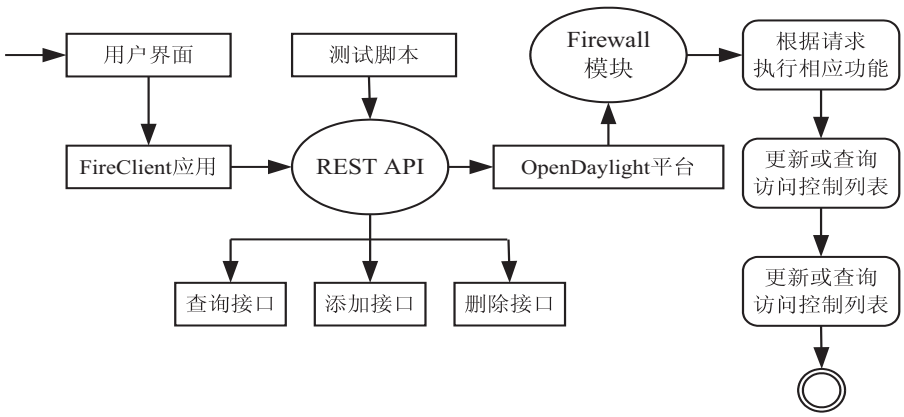

图 1　整体设计流程

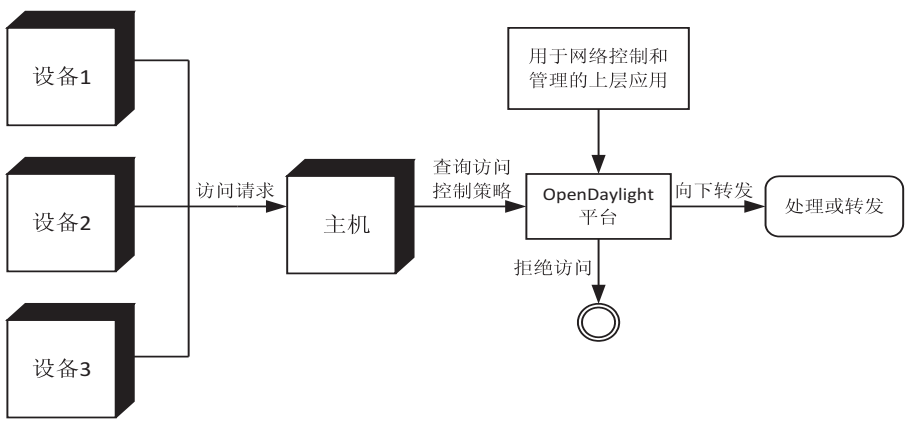

图 2　防火墙应用的控制流程

设计的防火墙应用控制流程如图 2 所示。当网络设备向主机发送网络请求时,装载于 OpenDaylight 平台上的防火墙应用会查询访问控制策略列表,如果请求访问的设备 IP 在列表内,则拒绝访问,否则对访问

请求做出处理或者向下层设备转发。基于此,首先需要基于 OpenDaylight 平台开发出防火墙应用(firewall 模块),用于访问控制,还需要开发出上层应用(FireClient)对访问控制策略进行添加、删除、查询等管理操作。

Firewall 模块的调用流程则是:用户通过上层应用调用 firewall 模块提供的标准接口,传入相应参数,firewall 模块本身对用户请求和参数进行对应的处理,更新访问控制列表,返回接口调用结果等。

FireClient 的功能流程则是:FireClient 应用提供发送请求、查询、增删策略功能,每个功能调用对应的 firewall 模块提供的 API,实现接口调用相关的网络请求发送,返回结果的 JSON 数据处理和展示等。

模块是 OpenDaylight 的功能单元,各个模块之间相互独立又彼此协调,组合起来构成了 OpenDaylight 对外提供的网络资源管理、数据分发等全部功能。比较常用的模块有 yangUI、dlux、l2switch 等。本次研究中设计开发的防火墙就是基于 OpenDaylight 平台的模块,该模块可以在开发测试完成后用 Maven 打包,并导入 OpenDaylight 的 release 版中,与其他模块一起使用。

2 实验环境搭建和测试结果

2.1 基于 OpenDaylight 开发的防火墙模块的应用架构设计

Maven 是基于项目对象模型(POM)的软件项目管理和依赖分析工具,可以通过描述信息进行创建项目、编译项目、加载项目依赖等[10-11]。基于 Maven 简化构建过程的思考,在本次项目开发中,将使用 Maven 构建 firewall 项目骨架,并且使用 Maven 对开发好的 firewall 进行编译、打包。

Yang 是一种数据建模语言[12-13],开发者通过对 Yang 模型的构建,指定标准访问接口和数据类型,通过远程调用或消息通知的方式实现模型配置与数据的分发处理。Yang 模型的类型之一为 Data,负责定义数据类型并用来进行数据存储,分为两种:Config 和 Operational。Config 为配置信息,是北向上层应用向南向网络发送的配置数据信息,Operational 为运行时信息,通常是 OpenDaylight 记录的南向网络动态信息。

实验设计中开发的 FireClient 是基于 Mac OS X 系统的 Cocoa 框架,开发语言是 Swift,集成开发环境是 Xcode[14]。主要步骤如下:

第一步创建项目;第二步开发用户界面,使用原生的 AppKit 框架[15],框架底层采用 OpenGL 绘制,用到了对象的延迟加载及 frame 相对布局,界面效果如图 3 所示;第三步调用 REST API,firewall 模块提供的 REST API 有:发送请求(request);查询访问控制 IP 列表(query);增加访问控制 IP(add);删除访问控制 IP(delete)。

图 3 FireClient 用户界面

REST API 调用代码,以 query 接口为例,执行步骤如下:

(1)构建 HTTP 请求头,请求头中包括 OpenDaylight 的 Dlux 界面的登录授权,content-type 等数据;

(2)构建 HTTP 请求体,请求体中包括向 query 接口发送的参数;

(3)申请并初始化 NSURLRequest 对象 request,设定请求方式为 POST,并指定已构建好的 HTTP 请求头和 HTTP 请求体;

(4)获取 URLSession 的单例对象 session,session 异步调用 dataTask 方法,传入 request 参数;

(5)在 dataTask 方法的 block 中对请求的响应结果的 JSON 数据进行解析,获取有效信息;

(6)获取主线程,在主线程中将有效信息展示在用户界面上;

(7)调用 session 的 resume( )方法开始执行 session 的 HTTP 请求。

2.2 测试结果及结果分析

模拟测试脚本用于模拟虚拟设备向 firewall 模块对外提供的 REST API,脚本代码如图 4 所示。

```
#!/bin/bash
echo "";
echo ">>>>>>>>senting request>>>>>>>>";

echo ">>request content: $1";
echo ">>       source ip: $2";

echo ">> request result:"
curl -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H
    "Content-Type: application/json" -H "Cache-Control: no-cache"
    -d '{"input":{"content":"$1", "ip":"$2"}}' "http://localhost:
    8181/restconf/operations/hello:request";
echo "";

echo ">>>>>>>>request end>>>>>>>>>>>>";
echo "";
```

图 4 脚本代码

测试过程按照如下用例依次进行:

用例 1:调用脚本向 request 发送请求,模拟 IP 为 192. 168. 2. 1,预期结果:访问成功。执行脚本./request. sh "hello" "192. 168. 2. 1",输出结果,如图 5 所示。

```
>>>>>>>sending request>>>>>>>
>>request content: hello
>>      source ip: 192.168.2.1
>>request result:
{"output":{"status":"Receipt success,content:hello"}}
>>>>>>>request end>>>>>>>
```

图 5　用例 1 测试结果

用例 2:用 FireClient 添加访问策略,禁止 IP 为 192. 168. 2. 1 设备访问,预期结果:访问拒绝。用 FireClient 增加访问策略,禁止 IP 为 192. 168. 2. 1 的设备访问,点击“增加”按钮之后,则添加成功。执行脚本./request. sh "hello" "192. 168. 2. 1",输出结果如图 6 所示:阻止了 IP 为 192. 168. 2. 1 的设备的访问。测试结果和预期一样。

```
>>>>>>>sending request>>>>>>>
>>request content: hello
>>      source ip: 192.168.2.1
>>request result:
{"output":{"status":"Access denied"}}
>>>>>>>request end>>>>>>>
```

图 6　用例 2 测试结果

用例 3:添加访问策略,禁止 IP 为 192. 168. 2. 2 的设备访问,用 FireClient 查询访问控制策略列表,查询值为 2,预期结果:返回两个禁止访问的 IP 地址。添加访问策略,禁止 IP 为 192. 168. 2. 2 的设备访问,操作步骤与用例 2 相同。用 FireClient 查询访问控制策略列表,查询值为 2。查询结果显示了添加的访问控制列表。

3　结束语

文中提出了一种基于 OpenDaylight 的防火墙设计与实现方案,并通过实验验证了其可行性。实验结果表明,由于 OpenDaylight 已经提供了丰富的应用程序接口,使得快速进行网络应用的二次开发成为可能。虽然该应用原型测试案例比较简单,但所有的防火墙策略都可以通过用户界面灵活修改。下一步将对用户策略的抽象和定制以及平台跨操作系统的移植等进行研究。

参考文献:

[1] 戴　彬,王航远,徐　冠,等. SDN 安全探讨:机遇与威胁并存[J]. 计算机应用研究,2014,31(8):2254-2262.

[2] 刘文懋,裘晓峰,陈鹏程,等. 面向 SDN 环境的软件定义安全架构[J]. 计算机科学与探索,2015,9(1):63-70.

[3] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow:enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[4] 王淑玲,李济汉,张云勇,等. SDN 架构及安全性研究[J]. 电信科学,2013,29(3):117-122.

[5] GREENE K. TR10: software-defined networking[N]. MIT Technology Review, 2011-10-07.

[6] 王雪梅. 基于 OpenDavlight 架构的路由组件的研究与实现[D]. 北京:北京邮电大学,2016.

[7] RADDA P. Brocade leads OpenFlow adoption to accelerate network virtualization and cloud application development[N]. Reuters, 2011-11-29.

[8] 文旭韬. SDN 安全控制器的优化设计与实现[D]. 北京:北京邮电大学,2015.

[9] 费　宁,陈春玲,毛燕琴. ASIC 芯片 OpenFlow 交换机设计与实现[J]. 北京邮电大学学报,2016,39(6):93-98.

[10] 许晓斌. Maven 实战[M]. 北京:机械工业出版社,2010:67-78.

[11] 钱言佳. 基于 Maven 的 CWAP 框架基础单元层和基础服务层的设计与实现[D]. 南京:南京大学,2016.

[12] 常亚楠. 基于 YANG 语言的 NETCONF 网络管理数据建模的研究与实现[D]. 武汉:华中师范大学,2009.

[13] 文俊浩,杨小义,谢　军. 扩展 UML 活动图在工作流建模中的应用[J]. 计算机应用研究,2007,24(12):244-245.

[14] MANNING J, BUTTFIELD-ADDISON P, NUGENT T. Swift development with Cocoa developing for the Mac and iOS App stores[M]. 北京:人民邮电出版社,2015.

[15] DOVEY J, FURROW A. Objective-C 开发经典教程[M]. 北京:清华大学出版社,2014.