# 基于SDN的DDoS攻击防御实现

武仕沛

北京航空航天大学计算机学院

摘 要：分布式拒绝服务攻击（Distributed Denial of Service）即利用大量合法的分布式服务器对目标发送请求，从而导致正常合法用户无法获得服务。传统分布式的网络不具备状态性，难以集中管理，在转发包时，直接采取查表的方式进行转发，而不对数据流进行分析，这就给DDoS攻击流留下可乘之机。而集中式的、基于软件定义网络（SDN）体系架构的网络能够迅速有效地收集南向接口传来的数据，通过自定义控制器模块实现对网络数据的监管和辅助传输。本实验着重研究SDN网络下DDoS的攻击方的行为，并辅以行之有效的防御措施，使得SDN网络变得安全。此次实验攻击方实现链路攻击和单目标攻击两种攻击方式，针对这两种攻击分别在SDN网络中部署packet-in协调转发和攻击流量检测模块，试验结果表明，在采取防御措施以后，针对SDN的DDoS攻击影响得到明显抑制。

关键词：分布式拒绝服务攻击；软件定义网络；链路攻击；packet-in协调转发；攻击流量检测

0 引言

分布式拒绝服务攻击（DDoS）通过大量合法分布式节点对服务器发送请求，从而淹没服务器临近链路带宽或耗尽服务器本身计算资源，最终使正常用户无法获得服务器的响应。

由于现行网络本身具有分布式的特点，因此在面对DDoS攻击时，往往难以及时协调一致的进行响应。通常分布式的网络设备是不具备状态的，也很难进行集中管理以记录状态，因此在收到转发包时，直接采取查表的方式进行转发，而不对数据流进行分析，这就给DDoS攻击流留有可乘之机。因此，当网络检测到DDoS攻击时，被攻击服务器临近链路已经被消耗了大量带宽，且服务器也已经收到了大量非法的请求。可见传统的网络在面临该攻击行为时，其反制措施往往有一定后滞性。

传统的采用分布式协议的网络如不能如期对DDoS攻击进行有效的防御，那么便考虑集中控制式的网络模型。因此，本实验采用SDN网络模型，结合mininet搭建起虚拟网络环境，控制器采用FloodLight，控制器和交换机之间采用OpenFlow协议传输报文和下发流控制规则，网络流量监视器采用sFlow进行监控。

本实验的工作可以分为两大部分，一是通过在mininet中搭建虚拟网络环境，并结合FloodLight开发，通过源码级的调试，来测试诸如控制器模块、报文交换模块、链路发现模块、拓扑管理模块和转发模块，进而对控制器有一个全面而详细的认识，并结合网络流量监测工具sFlow对网络流量进行实时监控，通过设置转发规则并查看前后流量变化，从而对SDN网络架构及其工作流程有一个宏观的认识。

另一部分的工作内容在于DDoS的攻防对抗上，正如前文所述，DDoS攻击可分为两大类，一是对服务器临近链路带宽的占用，二是对服务器本身计算资源的耗费。因此，这部分工作重点就在于如何设计攻击流量以达到占用链路带宽和耗尽服务器计算资源的效果，以及设计相应的算法或者逻辑规则，并部署到控制器上，针对这二者的攻击，起到良好的防御效果。

1 相关工作

此次实验由小组三人共同完成，我涉及到的部分为项目整体环境搭建，FloodLight控制器开发，链路攻击流量产生，packet-in协调转发策略/模块的设计和实现，并通过测试检验了本人工作的有效性。

2 实验环境搭建

本实验采用Windows+Linux双系统联调开发，具体的，Windows上部署FloodLight开发和运行环境，借助于IDEA强大的编辑功能，可以方便的进行模块调试，数据通路的查看。而在Linux上部署MiniNet平台，实现虚拟网络环境的搭建，同时借助于sFlow代理工具，实现网络环境中各节点、各端口的流量监测。

双系统联调开发首要保证的就是两平台的联通性，首先是在Linux上配置IP地址，设置为192.168.137.181/24，Windows配置IP为10.4.9.251/24，之后在mininet启动过程中，将远端控制器IP设置为Windows的IP地址即可。

FloodLight作为控制器，启动之初仅充当了一个作为SDN网络协调者的黑盒功能，如果要针对本实验的DDoS攻击做检测和防御的话，需另行实现检测和处理模块。因此除了使用FloodLight外，还应当深入了解该工具源码，学会如何二次开发。FloodLight源码可在github.com/floodlight/上下载，之后在IDEA中导入该工程，即可进行开发。初始版本可直接进行编译运行，FloodLight自带前端模块，通过可视化的形式对所管理网络进行展示。通过maven打包发布后，通过jar命令进行部署。

另在Linux环境下，通过apt-get方式安装MiniNet组网软件，所需的Java和Python环境以及sFlow监测软件均可通过源码包的方式进行安装。

MiniNet组网软件使用较简单，可通过命令式或图形界面创建拓扑结构，本实验以一个度为8，深度为2的树形网络为例，创建命令为sudo mn --switch ovsk --topo tree,depth=2,fanout=8 --controller=remote,ip=10.4.9.251,port=6633。各参数意义如下：

sudo：由于虚拟组网环境需要调用到内核网卡接口，因此需要管理员权限运行。

--switch：指定网络中交换机的类型，由于该实验交换机需要与控制器进行交互，因此需选用能够支持OpenFlow协议的交换机，ovsk中的ovs即代表Open vSwitch。

--topo：网络拓扑结构类型，常见的有single（单点）、linear（总线）、tree（树形）结构，根据选择的类型不同，需要指定具体的细分参数，如树形结构中需指明树深度，节点度数。

--controller：指定控制器来源，需指明ip地址和端口号。

在FloodLight前端界面中，可以图示化的形式展示该网络拓扑，如图1所示，图中每个节点都标识了其mac地址，方便在MiniNet中进行管理。

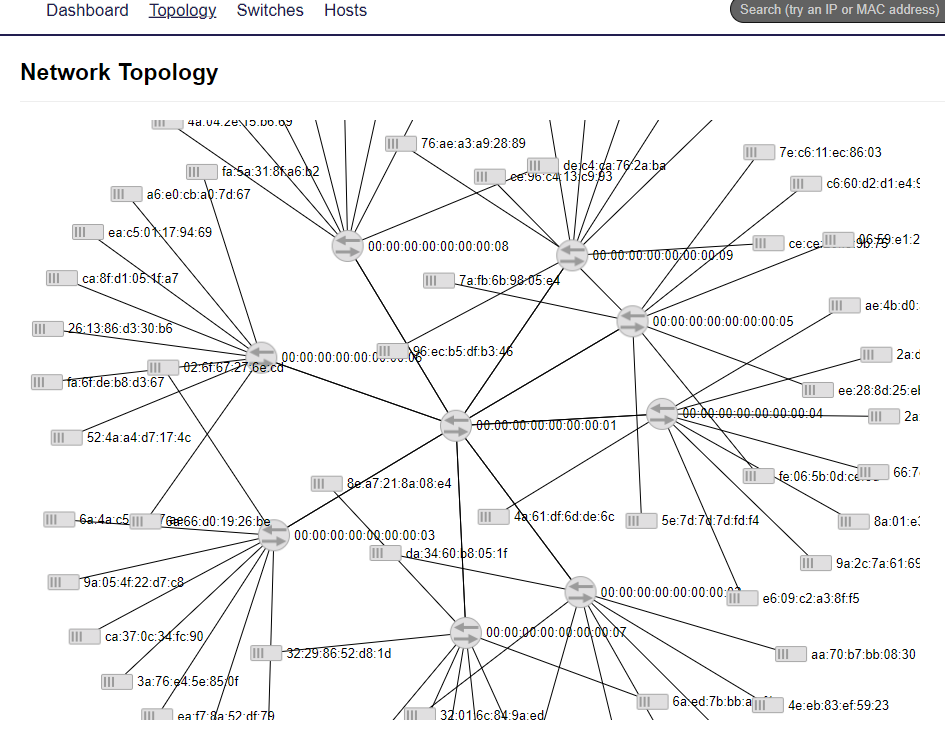


图1 FloodLight拓扑结构可视化

mininet除了可以进行虚拟网络搭建以外，还可以针对交换机、主机进行单独管理，例如ping报文测试，这也为后面的攻击行为提供了环境基础。使用方式实在mininet交互式命令行中，通过xterm -name单独对打开的节点进行管理，（-name代表交换机或主机节点），这里以xterm h1为例，打开h1主机的管理窗口，并进行ping报文的测试，被ping主机为h32（10.0.0.32），ping命令结果如图2所示。后续的攻击流量测试就以该环境为基础，通过编写python脚步，实现链路带宽攻击。

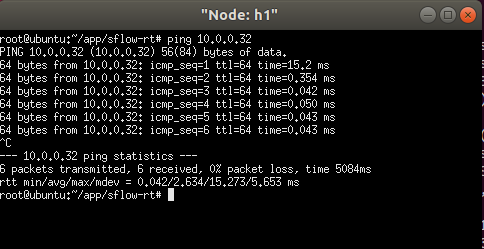


图2 虚拟主机节点通过xterm进行ping通测试

流量监控软件sFlow可以实现对网络中单个或多个被管交换机进行流量监控，这取决于配置的sFlow-agent的数量。首先启动sFlow服务端，命令为：./sflow-rt/start.sh。之后需要针对被管交换机设置sFlow-agent，设置了agent的交换机会将自身链路接口处的流量信息上传给sFlow-agent，agent配置命令为：sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0 target=\"127.0.0.1:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow。命令参数意义如下：

agent：指定采样的网卡，由于虚拟组网环境中，所有的流量都会经过虚拟机的主网卡，因此网卡为eth0。

target：sFlow服务端地址，默认端口号为6343。

sampling：采样率，即每隔N个Packet采样一次。

polling：轮询时间，即每隔N秒轮询一次。

bridge：被监控的网络设备，这里以h1所在的交换机s1为例，待会儿以h1主机进行ping报文测试，查看流经s1的流量。需说明的是，如果要针对全网环境测试的话，该命令需要多次配置，以覆盖所有交换机/路由器。

后端配置端口默认为8008，总揽界面如图3所示。

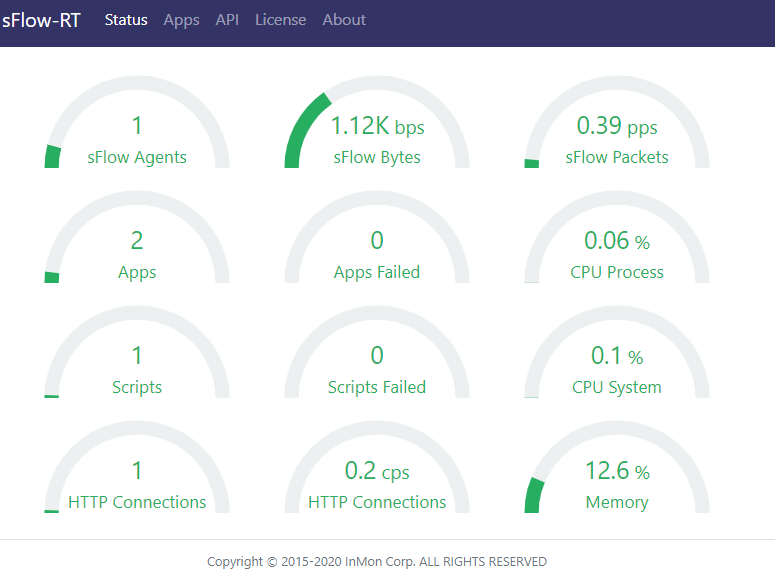


图3 sFlow流量监控仪表盘

3 链路带宽攻击

链路带宽攻击在DDoS环境下相较于传统网络较为新颖，其攻击的原理不同，但效果相同，在业界已经有相关论文对该种攻击做过详细阐述，其攻击原理是通过发送大量源、目标IP等特征伪造的数据包，当OVS设备收到该数据包后，查看流表发现无法匹配时，就会向控制器发送packet-in报文，请求控制器为该报文计算一条传输路径，当接收到控制器的指示后，就会根据该流表转发该报文，若同一时刻接收到大量的“未知”数据包时，就会因流表缓存溢出从而向控制器发送大量的packet-in报文，若超过控制器处理阈值，就会产生时延/丢包，后续真正的合法请求包就会得不到转发。

由于该实验设备有限，同一时刻下无法产生足以抑制控制器处理的packet-in报文，所以为了方便复现该攻击，人为的在控制器Forwarding模块中调低了packet-in处理速率，同时减小了OVS设备的缓存大小，让其更快产生缓存溢出。

得知攻击原理后，相应的防御机制就比较好实现了，造成链路带宽被占用的原因无非就是packet-in大量拥塞，那么解决方案要么从源头上减少packet-in的产生，要么将packet--in进行分流，相当于做了个负载均衡处理。针对该攻击，由于无法事先得知何种攻击源会产生该攻击报文，因此也就无法采用源头控制的方案了，而是考虑采用后者，后文称该种防御机制为packet-in协调转发粗略。

4 packet-in报文协调转发策略