SDN 环境下的 DDoS 检测与缓解机制

贾 锟[1,2], 王君楠[1,2], 刘 峰[2]

[1] 中国科学院大学网络空间安全学院 北京 中国 100093
[2] 中国科学院信息工程研究所 北京 中国 100093

摘要 软件定义网络(Software-defined Network, SDN)以可编程的形式定义路由, 对传统网络架构进行了一次彻底颠覆。通过采用中心化的拓扑结构, SDN 有效实现了对网络基础设施的全局控制。然而这种中心化的拓扑极易受到网络攻击的威胁, 如分布式拒绝服务攻击(Distributed Denial of Service, DDoS)。传统的 DDoS 通过堵塞交换机带宽, 消耗控制器计算资源的方式实现拒绝服务。近年来, 又有新型的 DDoS 变种通过攻击控制器与交换机通信的南向通道, 攻击交换机流表的方式实现拒绝服务。为了缓解传统 DDoS 和新型 DDoS 带来的安全问题, 本文提出了一个面向 SDN 的轻量化 DDoS 检测防御框架 SDDetector(Software Defined Detector)。可以在粗粒度和细粒度两种模式下运行, 粗粒度模式通过提取 SDN 交换机中的统计特征对可疑的攻击行为进行阈值警报; 触发警报后, 细粒度模式再进行二次特征提取, 并利用熵检测算法和 SVM 检测算法做进一步地攻击判别。研究发现, 熵检测算法擅长处理采用源 IP 伪造技术的 DDoS 攻击以及针对 SDN 的新型 DDoS 攻击; 而 SVM 检测算法擅长处理基于应用层协议的、需要交互的 DDoS 攻击。SDDetector 以近似并行的模式运行两种算法, 自动使特征提取速度最快的算法来完成攻击检测, 从而大幅降低了系统对攻击的响应时间。经过实验验证发现, 在特定场景下, 本文提出的模型能够比单一的检测方案少用 75%的响应时间。

关键词 分布式拒绝服务攻击; 软件定义网络; RYU; MiniNet; 轻量化; 机器学习; 人工智能
中图法分类号 TP393.1 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2021.01.02

DDoS detection and mitigation Framework in SDN

JIA Kun[1,2], WANG Junnan[1,2], LIU Feng[2]

[1] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Abstract Software-defined Network (SDN) overturns traditional network framework thoroughly with programmability and centralized management. While history tells us that centralized topology may easily incur single-failure considering the key role of controller. Distributed Denial of Service (DDoS) is such a threat to SDN. Traditional DDoS forces servers to stop service through flooding bandwidth or exhausting computing resource. This can still make sense if the attacking target is SDN switches or controllers. Besides, recently some new DDoS type try to specially make use of vulnerability of SDN, such as southern channel and rule tables. The new DDoS can achieve denial of service with less traffic and less time. To solve this problem, we present a lightweight detection framework called SDDetector (Software Defined Detector). It can work in coarse or finely modes. The former collects coarse statistic information and check whether they are exceeding thresholds. If it's determined as an anomaly, controller will send an alarm to switches and ask for a finely statistic information. Then the entropy detection algorithm and SVM detection algorithm start to work. Research shows that entropy detection algorithm can give a faster response in new DDoS and traditional DDoS with IP-Spoofing technique. While the SVM works better when facing with DDoS based on the application layer, because real IP needs to be used to communication. SDDetector runs two detection algorithms almost simultaneously, and the one who gives faster response domains the result. After that, controller will push a new flow rule in order to redirect the attacking traffic. Experiment shows that our method can give a significant 75% reduction in reaction time comparing to others in some attacking types.

Key words distributed denial of service; software defined network; RYU; MiniNet; lightweight; machine learning; artificial intelligence

通讯作者: 刘峰, 博士, 博士生导师, Email: liufeng@iie.ac.cn。
本课题得到国家重点研发计划(No. 2018YFC0806900)和北京市科学技术委员会(No. Z191100007119009)支持。
收稿日期: 2020-09-30; 修改日期: 2020-11-16; 定稿日期: 2020-11-25

1 引言

软件定义网络(Software-defined Network, SDN)是一种新型的网络架构, 旨在协助网络运营商更好地管理基础设施。基于这个目标, SDN 构建了一个中心化的结构, 赋予控制器全局视角, 使其得以动态化地管理旗下所有交换机。通过软件编程的形式, 控制器可以快速定义和控制路由规则、设置网络拓扑、安装网络应用等。鉴于其良好的可编程性和全局性, SDN 被广泛应用于具有大量服务设施, 需要统一配置管理的环境, 如云端, 医院和学校。但是其中心化的架构在网络攻击面前又极其脆弱。一旦核心控制器被攻陷, 整个 SDN 网络都会崩塌[1]。针对这种架构最常见的攻击手段就是分布式拒绝服务攻击(Distributed Denial of Service, DDoS)。

传统 DDoS 一般由攻击者控制的僵尸网络发起。大量受控终端, 通过向指定受害者发送泛洪式地应用请求, 消耗目标的带宽和计算资源, 迫使目标服务器停止正常的应用服务。如果目标位于 SDN 架构, 带宽耗尽时, 同链路的其他服务器也会受到影响。

除了传统的 DDoS, 研究人员也发现了针对 SDN 本身的新型 DDoS 攻击手法, 如针对控制器中心架构的 Packet-In 泛洪攻击[2], 针对交换机的流表溢出攻击[2], 针对流表计时器的慢速 DoS 攻击[3]及针对南向通道的 CrossPath 攻击[4]等。

为了缓解这些问题, 研究人员提出了大量方案, 包括基于异常检测的方案[5-9], 基于架构革新的方案[10-11]和基于负载均衡的方案[12-13]等。SDN 独有的可编程特性和全局视角使得这些方案能够快速地实现和应用。

SDN 环境下的 DDoS 检测方案关键要点在于能否保障轻量化和实时性。因为 SDN 本质上是通过建立一个中心架构来实现控制层与数据层的分离, 所以它的控制层需要承接旗下所有交换机的通信任务。如果检测机制不能做到轻量化, 就会占据太多宝贵的计算资源; 而实时性是 DDoS 防御的基本要求。现在很多主流的攻击方式都采用间歇式爆发攻击策略, 即短时间快速发起打击, 在瘫痪目标网络之后迅速停止。如果不能做到实时检测, 防御机制就无法从根本上阻止 DDoS 的破坏。

调研发现, 现行的 DDoS 检测方案大都着眼于某些特定的攻击手段, 无法在多种 DDoS 攻击场景下保持自身的轻量化与实时性; 此外, 研究人员常忽略了在正常工作状态时或者检测告警前降低 SDN 工作负荷的重要性, 导致 SDN 极可能在检测响应之前就被瘫痪。为改善以上问题, 本文提出了兼顾轻量化与实时性, 适用于广泛场景的快速检测方案。

本文的主要贡献包括以下三点:

1) 提出了轻量化的 SDDetector 检测方案。该方案完全基于 SDN 自身的 API 实现攻击检测, 不依赖任何外部的辅助工具; 其轻量化不仅体现在攻击发生时检测响应的及时性, 还体现在无攻击时对 SDN 控制器的减负效果。

2) 创新性地提出在不同状态下采用不同粒度的路由规则。具体来说, 在没有攻击警报时, 执行的是粗粒度的路由规则; 在攻击警报发送到控制器时, 执行细粒度的路由规则。这种设定能够过滤掉大量的重复流量和合法流量, 从而极大地减轻控制器的负载, 提升方案的反应速度。

3) 显著提高多种攻击场景下的检测效率。SDDetector 以近似并行的方式运行 Packet-In 的检测手段和基于流表统计信息的检测手段, 依据特征提取速度最快的算法来完成攻击检测。实验表明, 在采用源 IP 伪造技术的 DDoS 攻击场景下, SDDetector 能够比参照的单一 SVM 检测机制少用 75%的反应时间。

文章结构安排如下: 第二章简要阐述 SDN 的背景信息, 并介绍了当前的研究现状; 第三章详细介绍了 SDDetector 的设计细节以及工作流程; 第四章分别进行了熵检测方案、SVM 检测方案、特征有效性检验的相关实验, 最后在实战场景下进行了模型的工作场景再现; 第五章总结 SDDetector 在多场景下提高检测效率的方式, 提出仍面临的部分问题, 并在最后说明方案未来的改进方向。

2 背景与相关工作

本章简单介绍软件定义网络和OpenFlow协议的基础概念, 并对现有的研究工作做一个总结。

2.1 软件定义网络与 OpenFlow

SDN 是一种颠覆传统网络基础架构的新型框架。由美国斯坦福大学 clean-slate 研究组率先提出, 以中心化代替分布式, 软件代替硬件。通过从各设备中剥离出数据层和控制层, 并整合控制层功能于高层级的控制器, 将传统的分布式拓扑变成集中管理的层级结构。并在控制层引入可编程的特性, 以简化网络配置等操作, 实现了定义路由规则, 改变网络拓扑等功能的程序化, 打破了传统架构的封闭性和僵硬性。

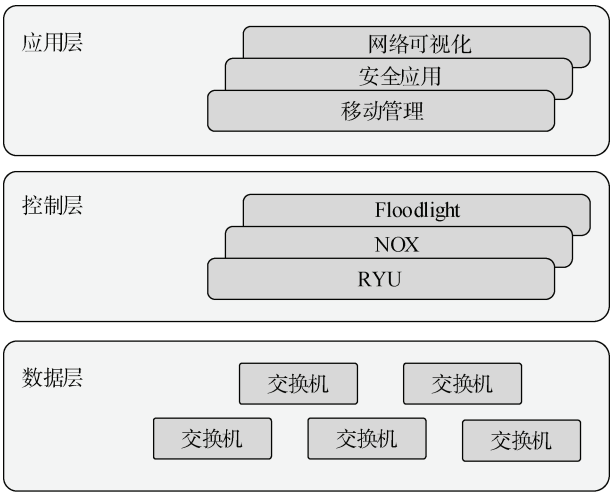


图 1　软件定义网络框架
Figure 1　SDN Framework

OpenFlow 是一种实现 SDN 框架的具体协议, 或者准确来说, 它构建了数据层的基本运行规则, 以及数据层与控制层的通信模式。OpenFlow 通过在交换机中引入流表的概念, 实现传统的路由功能。在流表与端口配置统计功能, 并授予控制器请求访问的特权, 使控制器获得全局视野。关于该协议的具体信息可参考 OpenFlow Specification[14]。

2.2　相关工作

SDN 大量应用在云环境和医疗环境等基础核心产业平台中, 因而针对 SDN 的 DDoS 容易连带地干扰寄生在平台上的大量服务, 造成重大网络安全事故。为了预防这个问题, 学术界提出了大量的检测防御方案。根据检测的方法可以分为基于统计的检测方案和基于机器学习的检测方案[15]。

基于统计的检测方案: 通过提取统计特征对 DDoS 攻击进行区分的方案。通常会设定一个阈值, 超过阈值的流量则会判断为 DDoS. Rui Wang 等人[5]提出了一个基于熵的检测方法, 通过在交换机端部署一个数据收集器, 并根据目的 IP 的地址计算流的熵值。但是这个方案依赖于五元组(源 IP, 目的 IP, 源端口, 目的端口, 协议)定义的流, 这种检测方案快速且高效, 但它需要单独部署一个额外的收集器, 因为 OpenFlow 自带的流表可能不会指定具体的某个元素, 如目的 IP, 这种情况检测方案就失效了。Mousavi 等[16]选择了一个相似的算法, 但他是每 50 个数据包计算一次熵值。这种方法在真实环境中的应用效果不是很好, 因为正常数据包的流量也很大时, 往往会导致误判。Shariq[17]等人对每个数据包进行评分, 评分原则基于源 IP 是否在流表中, 是否存在成功的 TCP 连接过, 协议类型, 数据通信的速率。并采用动态自适应的阈值算法判定攻击是否发生。Xiang You 等[9] 选择 Packet-In 信息作为攻击的信号。我们前面提到了当交换机遇到无法匹配的数据包时, 会触发 Table-Miss 选项, 从而向控制器发送 Packet-In 信息。在攻击发生时, 因为随机源 IP 的技术, 这种 Table-Miss 情况会激增, 从而导致大量 Packet-In 消息。但是真实环境中, 控制器极易在尚未来得及反应时就被瞬发的超大流量攻陷, 因为这些 Packet-In 信息都会通过南向接口发送到控制器。此外, 采用真实源 IP 的 HTTP 泛洪攻击、SSL 泛洪攻击, 因为不会激活大量 Packet-In 信息, 检测器就难以触发告警。

基于机器学习的检测方案: 通过提取流经交换机的流量特征, 并用机器学习方法进行恶意流量识别。在这之中, SOM 方法、SVM 方法和神经网络被使用得最为广泛。Braga 等[18]应用了一个轻量级的 SOM 方法, 在一个设定的时间窗口提取流量规则的六维特征, 包括流量规则下的平均数据包数, 平均持续时间等, 然后用 SOM 方法判断攻击是否发生。Jin Ye 等[7]通过 Onp_Flow_Stats 信息收集流表的统计信息, 并基于此提取源 IP 的增长速度, 源端口的增长速度, 数据包数量的标准差, 数据包字节量的标准差, 配对流的比例等特征, 最后用 SVM 算法作为分类依据。但是特征配对流的比例在面对通配符时无法得到准备的 计算结果, 影响了这个方法的普适性; 如果为了解决这个问题又必须明确指定五元组的信息, 又容易导致 Table-Miss 的情况。类似的, Mehr[19]等人也采用了 SVM 作为他们的分类算法, 但是他们生成的攻击流量仅仅每秒有 25 个数据包。这可以在简单的实验环境下得到正确的检测, 但在实际环境中, 结果的可信度仍值得检验。

基于统计的检测方案和基于机器学习的检测方案都绕不开对 OpenFlow 环境下流量特征的提取。主流的特征提取方式分为, 基于南向通道的 Packet-In 特征提取和基于流表统计信息的特征提取。

基于 Packet-In 特征提取的方案: 交换机触发 Table-Miss 选项后, 会发送 Packet-In 信息到控制器端。常见的 SYN Flood、UDP Flood、ACK Flood 为了保护受控端, 往往采用源 IP 伪造的技术, 因而会触发大量的 Packet-In,基于此可以提取相应的异常特征。Da Yin[8]、Xiang You[9]、Niyaz[20]等人的工作都采用的这种机制。此外针对 SDN 的 Packet-In 和流表溢出攻击也有比较好的效果, 因为他们本质上都需要随机生成匹配域, 从而触发 Packet-In。在以上攻击场景下, 如果采用基于流表统计信息的方案, 响应时间会因为南向带宽的堵塞以及交换机流表溢出而

大幅增加。

基于流表统计信息特征提取的方案: 在面对 HTTP 泛洪攻击和不采用随机化匹配域中元素的 DDoS 攻击表现卓越。因为他们能快速获取到流表特征, 并能提取到更接近真实攻击流量的特征, 如 Rodrigo Braga[18], Jin Ye[7] 等人的工作。但是在 MiniNet 仿真环境下实践发现, 流表中流规则数目过高时, 会极大地增加流表统计信息获取的耗时。所以, 在面对 Packet-In 泛洪和流表溢出等新型攻击手段时, 这种检测效率会大大降低。

表 1 SDN 环境下的 DDoS 检测研究汇总

Table 1 Research of DDoS in SDN 类型

类型	作者	解决方案大致描述	存在问题
基于 Packet-In 特征提取的方案	Seyed et al[16] I Gde et al[21] Xiang et al[9] Yin et al[15]	在控制器端部署信息收集模块, 捕获交换机触发 Table-Miss 选项后发送的 Packet-In 信息, 并进行统计特征提取	触发 Packet-In 效率低时, 无法快速检测攻击; 爆发性正常行为 Flash Event 也容易干扰检测效果;在 HTTP、HTTPS 等泛洪攻击场景表现不佳,慢速攻击也难以招架
基于流表统计信息特征提取的方案	Rui et al[5] Braga et al[18] Jin et al[7] Cui et al[22]	在控制器端部署信息收集模块, 定时发送 Onp_Flow_Stats 指令, 请求流表及端口的统计信息, 再进行特征提取	不适用于新型的 Packet-In 泛洪攻击和流表溢出攻击, Onp_Flow_Stats 指令执行阶段会消耗大量的时间

研究发现, 现有的检测方案往往存在以下问题:

1) 基于 Packet-In 特征提取的方案和基于流表统计信息进行特征提取的方案都不能很好地适用于广泛的攻击场景。

2) 研究大多针对于提高检测准确率而忽略了快速响应的需求, 尤其是在面对不同的攻击场景时, 系统的响应时间往往出入很大。

3) 研究均假定流规则是以细粒度五元组(源 IP、目的 IP、源端口、目的端口、协议)的形式呈现, 而实际环境中, 调整流规则的匹配域本应该是 SDN 的常态。

4) 有些方案实际上是从传统 DDoS 检测方式转移过来, 没有很好地利用 SDN 自身的全局特性。配置传统的检测方案通常又需要额外部署数据采集器, 使方案变得异常复杂还增加了额外的开销。

5) 现有的研究方案大都着眼于在攻击发生时进行预警, 从而针对性地遏止攻击流量以实现控制器的减负效果; 忽略了正常工作状态时对控制器的降负。

为了尽可能在广泛的攻击场景仍能实现快速检测响应, 并在正常工作状态时对控制器进行降负, 本文提出了融合基于 Packet-In 和基于流表统计的检测方案。方案重点考虑了如何在多种攻击手段下实现快速响应的能力, 为此, 我们以近似并行的机制运行两种检测方案, 以响应速度最快的方式来判定攻击。推翻五元组的匹配域设定, 改为在不同场景下应用不同粒度的匹配域, 以尽可能降低交换机与控制器的运行负担。此外, 本文还利用了控制器的全局视野, 定位攻击源, 并针对性地部署流规则以防御 DDoS。

3 方案设计

围绕 SDN 环境下的 DDoS 检测与缓解需求, 本文设计了面向多攻击场景的并行检测框架 SDDetector。

SDDetector 核心功能均部署在 OpenFlow 架构的控制器端。

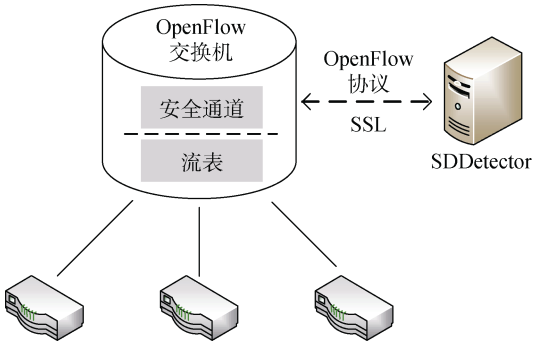


图 2 OpenFlow 设计框架

Figure 2 Framework of OpenFlow

SDN 的中心化架构赋予控制器遥控中枢的职能, 在保障交换机正常路由功能的条件下, 可以多样化定制其他的需求。开源的控制器 API 又给了开发者一个很好的平台, 可以快速且不干扰 SDN 基础功能的情况下部署安全应用。

SDDetector 核心功能由五个模块组成, 分别是数据监测模块、流规则模块、阈值警报模块、熵检测模块和 SVM 检测模块。方案整体的工作流程如下:

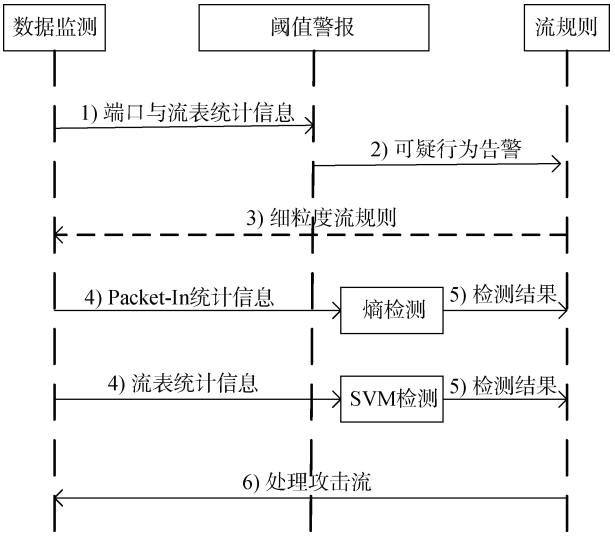


图 3 SDDetector 整体流程

Figure 3 Processes of SDDetector

1) 数据监测模块部署在控制器端, 设定固定的时间间隔发起 Onp_Flow_Stats 请求, 获取交换机端的统计信息。

2) 阈值警报模块判定是否激发细粒度的流规则。其判定的依据为正向和反向数据包数量的比例, 数据包的速率以及基于 MAC 地址的熵值。如果超出阈值, 该模块会尝试定位可疑受害者的 MAC 和可疑攻击者的 MAC。

3) 流规则模块负责下发针对性的流规则。有两种情况控制器会发送不同的流则。第一种情况, 阈值警报模块触发, 流规则模块删除初始的粗粒度流规则, 改成细粒度的流规则。粗粒度的流规则指, 只有交换机的输入端口、目的 MAC 地址为具体的值, 其他匹配域均为通配符 ANY; 细粒度的流规则指, 除了上面两个元素以外, 还有源 IP 和目的 IP, 以及传输层协议为具体值。第二种情况, 在进一步检测模块确认 DDoS 攻击发生后, 流规则模块将相应的数据流导流至流量过滤中心或简单的丢弃该流量。

4) 熵检测模块负责对发送到控制器的 Packet-In 消息进行相应的攻击检测。主要适用于流表溢出攻击和 Packet-In 泛洪攻击等随机化源 IP 的场景。

5) SVM 检测模块负责对可疑流量进行更多维的检测, 其依赖的特征来源于控制器对细粒度流表统计的请求。只处理针对可疑者受害者的流量, 以减少系统的工作量。

OpenFlow 交换机以流表(Flow Table)的方式实现传统交换机的路由功能。流表在 SDN 架构中类似于现今传统网络路由器中的路由表, 由很多条流规则(Flow Entry)组成。一条流规则定义一组路由指令, 包括了匹配域、优先级、处理指令和统计数据等字段。当一个数据包流经交换机时, 交换机根据匹配域寻找该数据包对应的流规则, 如果有多个匹配的规则, 则选择最高优先级的, 随后根据该流规则的指令对数据包进行处理。

流规则的匹配域包含以下元素:

表 2 流规则匹配域

Table 2 Match Fields of Flow Entry

入端口	源 MAC	目的 MAC	优先级	源 IP
目的 IP	IP 层协议	TCP/UDP 源端口	TCP/UDP 目的端口	……

匹配域中的元素可以是具体的某个值, 也可以是 ANY 通配符, 也就是可以根据需要定义一个宽泛的匹配域。如果数据包找到了匹配的流规则, 那么它根据定义的指令进行相应处理; 如果没有匹配的规则, 则会触发一个 Table-Miss 选项。这个选项会将该数据包的信息以 Packet-In 模式转发给控制层。控制层再根据需要进行适当的处理, 一般是添加匹配该数据包的流规则。

SDDetector 在初始阶段应用了一个粗粒度的路由规则, 即匹配域中只指定 In_Port 字段和目的 MAC 字段, 分别代表从交换机的进入端口和流量的目标 MAC 地址。当一个无法匹配的数据包到达 SDN 交换机时, 相应的 Packet-In 信息会发送到控制器。如果我们初始就定义一个细粒度的流表规则的话, 就可能导致大量不同源 IP, 不同端口信息的数据包触发 Tale-Miss 选项, 从而淹没交换机流表和控制器。粗粒度场景下的工作流程如图 4 所示。

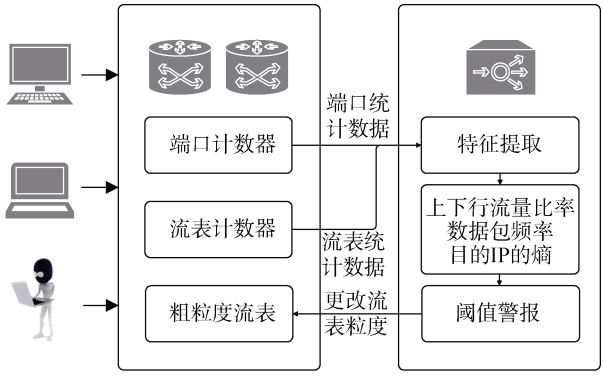


图 4 粗粒度场景的工作流程

Figure 4 Process of Coarse Condition

3.1 第一阶段统计信息请求

SDN 的端口和流表都配置了相应的统计功能。端口计数器负责统计流经该端口的数据包数量以及字节数。流表计数器记录匹配数据包的数量和字节信息之外, 还需要记录流规则存活的时间。每一条流规则都设定了一个硬性存活时间和一个软性存活时间。在软性存活时间内如果没有接收到匹配数据包时, 该流量规则就会自动清除; 而硬性存活时间截止时, 无论期间是否有匹配, 都会强制清除该规则。

控制器通过 Onp_Flow_Stats 指令获取 SDN 网络的全局视野。当交换机收到这个指令后, 会将所有的统计信息上传到控制器, 许多安全功能、限流功能都基于此。

统计信息请求最重要的两个问题分别是, 访问请求的时间间隔, 信息获取之后的特征提取。

访问请求的时间间隔: 数据监测模块基于预定义的时间间隔周期性地发起流表统计数据请求。预定义的时间间隔对于模型的运行效率具有非常大的影响。过长的时间间隔容易导致对攻击的响应过慢, 从而无法真正抵御突发性的 DDoS 攻击; 而过短的时间间隔容易加重控制器和交换机的负担。我们经过多番实验以及参考 Braga[18]等人的工作, 选择了与之相同的设计, 以 3s 作为预设的时间间隔。

特征提取: Dayal[23]等人在一次 DDoS 仿真实验中展示了他们总结的多个异常特征。可以看到常见的 DDoS 攻击, 如 Smurf, UDP flood, HTTP flood 和 SYN flood 都暴露出了基于目的 IP 的熵值异常。所以, 相似的针对局域网内部的 MAC 地址, 也有这样的异常特征。考虑到 SDN 通常应用于无限网络, 移动网络, 企业网络和校园网络, 这样的局域网假设是符合现实情况的。

DDoS 通常应用随机源 IP、随机目的端口的技术以躲避溯源追踪, 因而 DDoS 流量通常只有很少的交互信息。我们收集的一个真实环境 DDoS 流量显示仅仅有 3%~4%的下行流量, 也就是服务器到众多攻击源的流量只占据总流量的极少数, 但是在一个正常的流量集中, 下行流量占到了 45%~50%, 所以我们也监测在交换机端的上下行数据包数量的比率。本文用 *tx* 数量数据包表示从交换机视角发送的数据包, 用 *rx* 表示从交换机视角接收的数据包数量。

此外流经交换机的数据包速度是反映大流量异常的最直接特征。

以上三个特征目的 MAC 的熵、上下行流量的比率、数据包速度作为本文阈值检测模块的基本判断依据。值得一提的是, 以上三个特征均可以用来定位可疑攻击源和可疑攻击目标, 后文提及的针对性流规则部署即依赖于此。

OpenFlow 协议的 Onp_Flow_Stats 指令可以向控制器旗下的所有交换机请求端口、流表统计信息。

端口统计信息典型范例:

port"s3-eth1": *rx pkts*=143, *bytes*=11952, *drop*=0, *errs*=0, *frame*=0, *over*=0, *crc*=0; *tx pkts*=116, *bytes*=9844, *drop*=0, *errs*=0, *coll*=0

流表统计信息典型范例:

cookie=0x0, *duration*=101.711s, *table*=0, *n_packets*=39, *n_bytes*=2982, *priority*=1, *in_port*="s3-eth1", *dl_dst*=00:00:00:00:00:06, *actions*=output:"s3-eth3"

得到端口统计的数据与流表统计的数据之后, 根据下面的公式即可得到目的 MAC 的熵、上下行流量比率、与数据包速度。

$$(1) E = -\sum_i P_{MAC_i} \log_2 P_{MAC_i}$$

$$(2) ratio = rx / tx$$

$$(3) V_{packet} = Total\_Packets / Time\_Interval$$

3.2 阈值警报

在提取了目的 IP 的熵, 上下行流量的比率以及数据包频率三个特征之后, 我们对攻击威胁作一个初步判断。如果之中有一个特征超过了预设的阈值, 就会触发阈值警报。我们选取的阈值基于真实环境中 DDoS 的流量信息和在 MiniNet 环境下模拟的 DDoS 流量信息得到。我们依据异常的统计信息, 建立了一个简单的攻击定位和溯源机制。一旦后续我们的深度检测模块判定 DDoS 事件发生, 攻击源就可以确定。

3.3 修改路由规则

当阈值警报发出时, 流规则模块会立马生成相应的细粒度路由规则替换原先的粗粒度规则。具体来说, 控制器首先会指示交换机删除老的(攻击源端口, 攻击目标 MAC, action=output)的流规则。这样下一个指向受害者的流量会触发 Table-Miss 选项并被发送到控制器。然后控制器会针对地定义一个新的流规则(源 IP, 协议, 目的 IP, 目的 MAC, action=output)。这里同样为了避免 Packet-In 泛洪, 和 Dayal 等[23]相比, 细粒度的流规则并不考虑加入 TCP 或者 UDP 的端口信息, 且只有针对受害者 IP 的流量才会适用于新的路由规则, 这个措施可以大大地减少 controller 的负担。在用户众多、服务器众多的云环境和校园网环境下, 这种“降负”效果更明显。

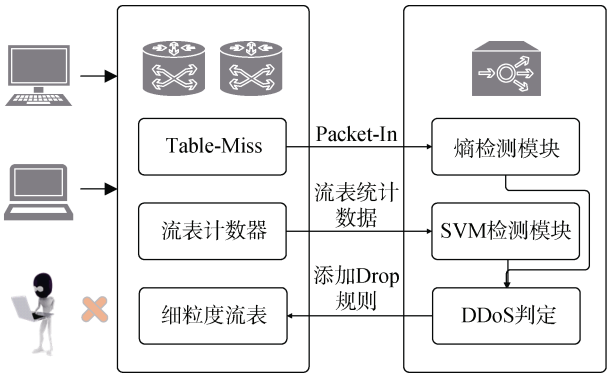


图 5　细粒度场景的工作流程

Figure 5　Process of Finely-grained Condition

3.4　第二阶段统计信息请求

这一次的统计信息依赖于 SDN 自带的 Table-Miss 功能和流表计数器。前者会在新流到来后发送 Packet-In 信息到控制器, 后者同第一阶段基本相同, 也是通过发送 Onp_Flow_Stats 指令进行请求, 只不过这次只提取细粒度流表的特征与端口特征, 且特征的定义也不一样。详细描述如下。

1) 流规则的数量: 每一个不同 IP 的数据包都会请求新的流规则, 所以我们可以发现在一次 DDoS 攻击中, 流规则的数量大幅上升, 我们用字符 n 代表流规则的数量。

2) 流规则下的数据包平均数量: OpenFlow 交换机会自动统计匹配该流规则的数据包数量。攻击者通常会生成随机的源 IP 地址以防防御方追踪溯源。这个方法被广泛应用于 SYN 泛洪攻击、UDP 泛洪攻击和 ACK 泛洪攻击。所以当匹配域扩展到包括源 IP 时, 流规则下的平均数据包数量会急剧下降。与之相对的, 如果攻击者不采用这种源 IP 伪造技术, 那么流规则下的平均数据包数量又会大幅上升。

$$Average = \frac{1}{n}\sum_{1}^{n} Packets\_of\_Flow_i$$

3) 流规则数据包数量的熵值: 我们以一个细粒度的视角去定义熵值, 而且我们只针对流向目标 IP 的流量, 也就是我们统计符合目标 IP 为受害者的流规则的离散度。

$$E_P = -\sum_i P_{Packets\_number=i} \log_2 P_{Packets\_number=i}$$

4) 目的 IP 的熵: 攻击发生时, 流量往往集中在受害者, 而目标为其他服务器的数据包占比会较小, 这就导致了目的 IP 熵值的异常。

$$E_{\mathrm{IP}} = -\sum_i P_{\mathrm{IP}_i} \log_2 P_{\mathrm{IP}_i}$$

5) 上下行流量的比率: 当海量的流量涌向受害者, 受害者通常不能正常地进行反馈回应。此外, 如果目标端口不在服务端口中, 那么服务器不会响应 RST 数据。这些因素都导致了 DDoS 期间的上下行流量的比率异常。上下行流量都是从交换机的视角去定义, 也就是从交换机到终端的流量为上行流量, 从终端到交换机的流量为下行流量, 所以实际上在发生 DDoS 攻击时靠近攻击源的端口, 这个比率会低于 1。

$$ratio = rx / tx$$

3.5　熵检测

与上面流表统计特征获取同步进行的是熵检测。熵检测模块部署在控制器端, 在第一阶段的阈值告警之后就会自动触发, 开始收集经由控制器的 Packet-In 统计特征。因为我们的设计同时采用了基于 Packet-In 的检测方案和基于流表统计信息的检测方案, 我们希望尽可能突出两种检测方案的各自优势, 减少两种检测方案的重复部分, 因此此处我们只针对采用随机源 IP 技术的攻击手段, 如 Packet-In 泛洪攻击和流表溢出攻击。并针对性地只选择了两个特征, 即源 IP 的熵和目的 IP 的熵。Xiang You 等[9]的方案选择了包括源 IP 的熵、目的 IP 的熵和目的端口的熵。并以 1000 个数据包作为窗口, 进行相应的熵值计算。本文的细粒度规则不涉及传输层端口, 所以端口的熵值对于模型检测意义不大。

熵是反映数据分布离散度的指标, 熵值比较高说明分布的集中度比较低, 也就是更加离散; 熵值比较低说明分布比较集中。极端情况, 如分布在 0 点的概率为 1, 那么该分布的熵为 0。熵的数学定义如下:

$$S = -\sum_i P_i \log_2 P_i$$

根据 Xiang You 等[9]的论述, 数据包的熵近似满足正态分布。正态分布具有高度对称性和集中性, 绝大多数数据分布集中在均值附近。以明确的概率方式阐述即:

——以 68.2%的概率分布在均值附近一个标准差的范围内;

——以 95.4%的概率分布在均值附近两个标准差的范围内;

——以 99.7%的概率分布在均值附近三个标准差的范围内。

本文选择两个标准差作为异常判定的标准, 因为随机源 IP 的攻击策略会导致源 IP 熵值急剧增大, 而针对性的泛洪攻击又会导致目的 IP 熵值急剧减小, 所以实际上我们只选择单一方向作为异常判定的标

准, 即:

$$v_{\text{src}} > \mu_{\text{src}} + 2\sigma_{\text{src}}$$
$$v_{\text{dst}} < \mu_{\text{dst}} - 2\sigma_{\text{dst}}$$

同时满足这两个条件时, 认为发生了 Packet-In 泛洪攻击。

3.6 SVM 检测

基于 Packet-In 的熵检测方案也有它的劣势, 其一是只有在拿到 1000 个 Packet-In 数据包之后检测才能进行, 但是在交互式的 DDoS 或者无源 IP 伪造的攻击场景下, 如 HTTP DDoS、HTTPS DDoS,交换机并不一定能很快触发足够的 Packet-In 消息。此外 Packet-In 消息只反映新流的报文信息, 也就是说已经有相应匹配流规则的数据包, 是无法通过 Packet-In 进行统计特征的提取的, 而数据包的数量, 流规则存活时间, 流规则的数目等又是反映 DDoS 的关键特征。因而单纯依赖 Packet-In 的熵很可能不能准确反映实际攻击的某些特征。

为了应对上面提到的交互式 DDoS 和无源 IP 伪造的攻击手段, 同时选取更能代表真实流经交换机流量的特征, 本文同步引入了一个 SVM 检测方案来分类正常流量和攻击流量。这里说的正常是指在此期间不存在 DDoS 行为, 攻击是指在此期间存在 DDoS 行为。文中针对的流量最小单元是时间间隔内的所有流量, 而不是单个数据包或单个流规则。因为在攻击发生时, 通常也会存在有正常的通信流量。得益于 RYU 控制器的开源 API, 我们的 SVM 检测模块可以轻松地部署在控制器端。SVM 的基本原理介绍如下。

SVM 分类模型:

SVM 最初是一种对数据进行二分类的线性分类器, 通过计算得到一个超平面, 以最低的误差将数据分割到超平面的两端。如图 6:

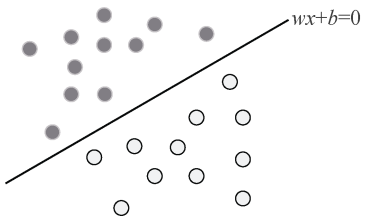


图 6 SVM 超平面
Figure 6 Hyperplane of SVM

超平面的定义为:

$$wx + b = 0$$

判定数据所属的分类是依赖于:

$$wx_i + b > 0 : Y_i = 0$$
$$wx_i + b \leqslant 0 : Y_i = 1$$

但是这种常规的 SVM 模型只对线性可分割的数据有效。因而在 20 世纪 90 年代, 一批学者提出了改进 SVM 的方法, 引入了核函数的概念, 使得 SVM 也能处理非线性可分的数据。

核函数的本质是将源数据从线性不可分的空间投影到线性可分的高维空间。常见的核函数包括高斯核函数(RBF), 多项式核函数和 *Sigmoid* 核函数。高斯核函数通过升维使原先线性不可分的数据变得线性可分, 在现实场景中应用非常广泛。相对于其他核函数, 高斯核函数更适用于样本数量可观, 而特征维度较少的场景, 因而本文最终选择了高斯核函数作为 SVM 检测方案的基础。

3.7 丢弃或导流

根据分类结果, 我们可以定制将攻击流量丢弃还是将其引导至专业的 DDoS 清洗设备。值得一提的是, 因为只有针对目标地址的流量才会引导至清洗设备, 所以实际上 SDDetector 还能降低外置的清洗设备的工作负荷。

本文整体方案的伪代码如下:

```
0 初始化时间窗口 Δt
1 time.sleep(Δt)
2 data=[]
3 for switch in SDN:
4     data.append(Request_Stats(switch))
5 (entropy,ratio,velocity)= Extract(data)
6 if DDoSalarm(entropy,ratio,velocity):
7     {
8       Change_RouteRule()
9           if Packet_In_Detection():
10              DropAttackFlow()
11          elif SVM_Detection():
12              DropAttackFlow()
13        }
14 goto 1
```

4 实验

本文的实验环境通过 MiniNet 构建。MiniNet 是基于 Linux Container 架构开发的一个进程虚拟化网络仿真工具。可以创建包含主机, 交换机, 控制器和链路的虚拟网络, 其交换机支持 OpenFlow1.1 和 1.3 协议。实验构造的拓扑结构如图 7 所示。

根据传统的路由规则, 所有流经路由设备的数据包都会修改源 MAC 地址为该路由设备的 MAC 地址, 目的 MAC 地址为下一个路由设备的 MAC 地址。

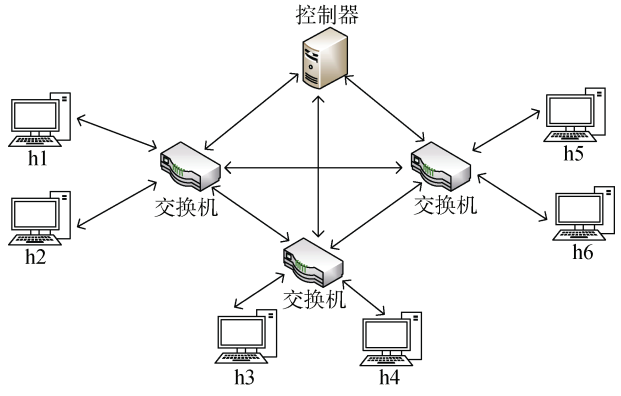


图 7　SDN 实验拓扑

Figure 7　Topology of SDN Experiment

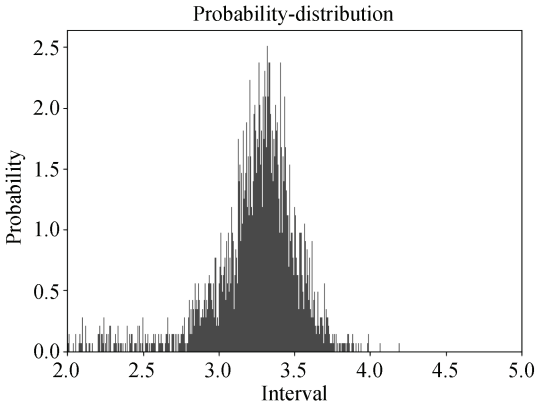


图 8　目的 IP 的熵分布

Figure 8　Entropy Distribution of Destination IP

但是通过 WireShark 抓包发现, SDN 网络内部并不会有这样的机制, 也就是说数据包在 SDN 环境下传输时, 源 MAC 和目的 MAC 从始至终都不会发生改变, 这对于我们的检测方案有着至关重要的作用。SDN 模拟环境运行在 Ubuntu 18.04.4 LTS 上。处理器为 Intel core i7-8550U, 2.00GHz, 内存为 8GB RAM。SDN 的控制器部署在另一台虚拟机, 内存为 4GB RAM。Hping3 作为生成攻击流量的工具, 它可以定制从链路层到应用层的几乎所有报文信息, 也可以自由设置攻击频率, 数据段长度, 还能选择是否随机化端口和 IP。实验阶段生成了 SYN DDoS、UDP DDoS、ACK DDoS, RST DDoS 和混杂型的 DDoS 流量。并且在某些时间阶段, 应用了源 IP 伪造的技术。背景流量在用户终端抓取, 并通过 TCPRePlay 工具在 SDN 模拟环境下进行重放。

为了全面地论证本文的方案设计, 我们进行了多个实验, 包括熵检测方案的实验, SVM 检测方案的实验, 特征有效性检验的实验, 攻防实战的实验。

4.1　熵检测方案的实验

3.5 节中提到, 源 IP 的熵和目的 IP 的熵符合正态分布, 为了验证这个结论, 实验采集了 4.5GB 的合法流量, 计算得到了正常流量目标 IP 分布的熵均值约为 3.14, 标准差为 0.515; 源 IP 分布的熵均值为 3.49, 标准差为 0.68。

通过将数据频次以直方图的形式描绘, 如图 8, 我们可以近似看到目的 IP 的熵的确呈现明显的正态分布特性。

4.2　SVM 检测方案的实验

为了验证 SVM 检测方案的有效性, 我们进行了 SVM 检测方案的相关实验。该环节收集的 DDoS 数据集来源于 MiniNet 模拟器下的模拟攻击。其中的流量类型包括 SYN DDoS、UDP DDoS、ACK DDoS、RST DDoS 和 Mixed DDoS。Mixed DDoS 指混杂了以上多种 DDoS 类型的攻击流量。为了较好地模拟真实环境, 实验还在多个终端上收集了 40GB 的正常通信流量作为实验的背景流量, 并通过 TcpReplay 工具在实验阶段进行重放。测试的 DDoS 数据集来源于真实环境下的攻击流量和模拟环境下的攻击流量。真实的攻击数据采集于 2018 年, 包含常见的 UDP DDoS, ACK DDoS 和 HTTP DDoS, 在 17min 发出了总共 22.3GB 的流量。将原始数据输入模拟的 SDN 网络, 交换机流表就会自动记录得到统计信息, 再经过一系列运算处理, 控制端才正式提取出来 SVM 检测方案所需的五大特征。我们最初在选择使用哪些特征时, 确立了一个基本准则, 即特征的值应该尽量不受用户规模的影响, 这样才能增加特征在不同应用场景下的普适性。在输入 SVM 分类模型之前, 提取的特征还需要经过 *sigmoid* 函数做一个标准化处理, 这能够协助 SVM 模型实现更快速的收敛速度和更高的分类准确率。方案设置了 50 轮的训练周期, 直到模型达到收敛。

Jin Ye[7]等人的方案和本文的方案原理类似, 均采用了基于 SVM 的分类算法, 但与我们不同的是, 他们的模型特征从始至终都基于细粒度的路由规则。即便如此, 我们的分类结果, 也显示出优于他们测试结果的特性。表 3 列出了 SDDetector 与 Jin Ye 方案的分类结果对比。

表 3　SVM 检测结果对比

Table 3　Jin Ye's Plan & Our Plan

		Jin Ye 的方案	SDDetector
TCP DDoS	准确率	96.83%	100%
	误报率	0/0	0
UDP DDoS	准确率	93.65%	100%
	误报率	0.9%	0

同时也在表 4 列举了 SDDetector 针对真实攻击下的检测结果, 以此证明实验结果的可靠性。

表 4　真实环境下的检测结果

Table 4　Detection Result in Real World

真实环境	结果
准确率	100%
误报率	0

检测结果证明 SVM 的检测方案可以很好地预警 DDoS 流量, 无论是实验环境下生成的模拟 DDoS,还是真实环境下的 DDoS。

4.3　特征有效性检验实验

决定检测方案有效性的一个核心因素就是特征的区分度, 为此我们分别采集了合法流量, 采用源 IP 伪造技术的 DDoS, 以及采用真实 IP 的 DDoS,并计算流规则的数量、流规则下的数据包平均数量、流规则数据包数量的熵值、目的 IP 的熵、上下行流量的比率具体值, 得到以下实例图(图 9~图 13)。

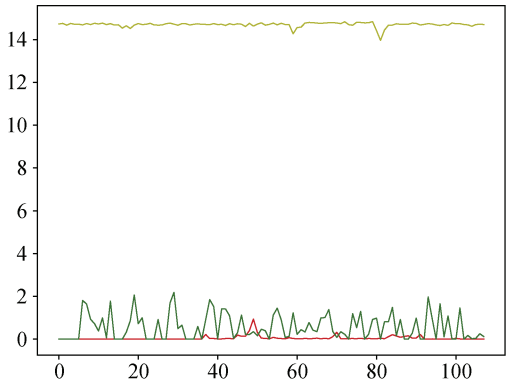


图 9　流规则的熵

Figure 9　Entropy of Flow Entry

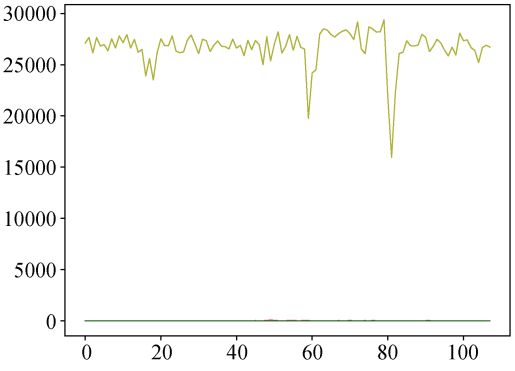


图 10　流规则数目

Figure 10　Number of Flow Entries

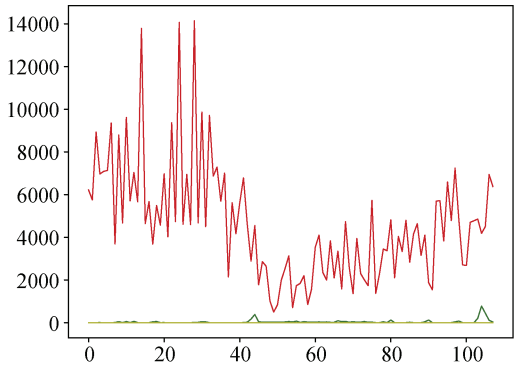


图 11　流规则下数据包的平均数量

Figure 11　Average of Flow Entry's Packet Number

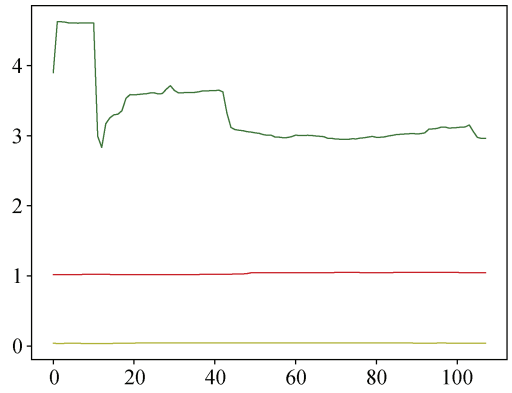


图 12　目的 IP 的熵

Figure 12　Entropy of Destination IP

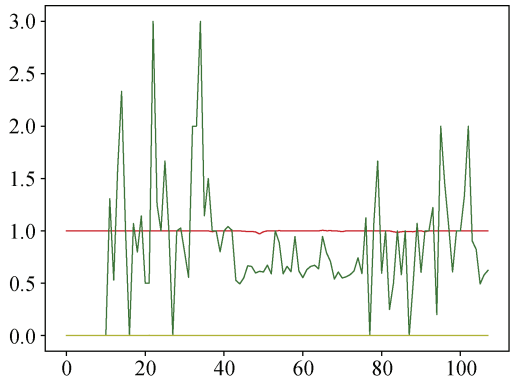


图 13　上下行流量比率

Figure 13　Ratio of Outgoing and Incoming Traffic

其中绿色代表合法流量, 黄色代表采用源 IP 伪造技术的 DDoS, 红色代表采用真实 IP 的 DDoS。图 10 中, 因为实验条件受限, 发起 DDoS 攻击的主机数目较少, 因而采用真实 IP 的 DDoS 与合法流量产生的流规则数目都太少, 在比例尺下发生了重叠, 所以只能看到一种颜色。

可以看到, 在绝大多数情况下, 三种特征的区分度都非常明显, 因而能够证明其用来进行分类的有效性。

4.4 攻防实战实验

控制器在 SDN 框架中扮演了核心中枢的角色。它负责监控整个网络的流量信息, 部署新的流量规则, 处理异常情况等。针对 SDN 的攻击手段很多都利用其中心化的架构, 达到瘫痪控制器就攻陷整个 SDN 网络的效果, 比如 Packet-In 泛洪攻击。交换机的流表也是容易被针对的点, 如恶意规则注入攻击[3], 对流规则算法的探测攻击[24], 以及流表溢出攻击[3]。

相对于传统 DDoS, 针对 SDN 的 Packet-In 泛洪攻击和数据层流表溢出攻击破坏力更大, 也更难检测和防御。4.4.1 和 4.4.2 对这两种新型 DDoS 的原理进行详细介绍。

4.4.1 Packet-In 泛洪攻击

OpenFlow 作为实现 SDN 框架的具体协议, 针对流经交换机的数据包定义了一套完备的处理机制。首先, 根据数据包的报文信息, 找到优先级最高的匹配流规则, 再执行流规则下定义的指令; 如果没有找到匹配的流规则, 则触发 Table-Miss 选项, 向控制器发送 Packet-In 信息, 请求控制器下发一套适用于该数据包的流规则。鉴于 SDN 网络的中心化架构, 数据包流经的所有交换机都需要向控制器发送一次 Packet-In 信息, 这对脆弱的控制器来说是一个极大的挑战。Packet-In 泛洪攻击的过程如图 14。攻击者

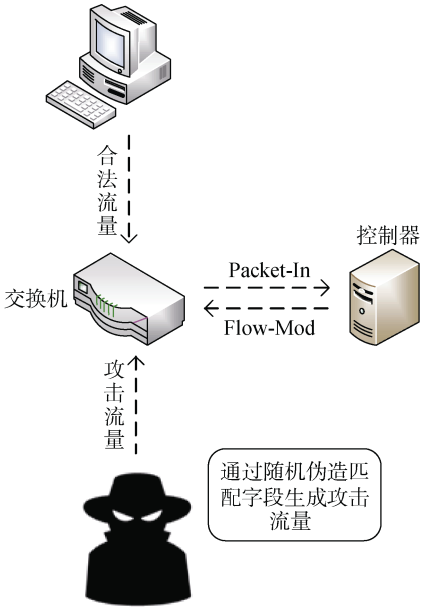


图 14 Packet-In 泛洪攻击

Figure 14 Packet-In flooding

通过伪造匹配域中的元素, 生成大量激发 Table-Miss 的数据包, 迫使交换机发送 Packet-In 数据包到控制器。控制器需要响应每一个 Packet-In 信息, 并下发相应的流规则。这不仅会堵塞南向通道的带宽, 还会附带耗尽控制器的计算资源。据 DevoFlow 测量, HP ProCurve 5406zl 型号的交换机仅仅可以支持 17Mbps 大小的南向通道带宽[3]。

4.4.2 数据层流表溢出攻击

OpenFlow 交换机的流表也可能成为 DDoS 攻击的目标。我们知道一个流规则必然指定某几个具体的元素, 如果新的数据包到来不能匹配这些流规则的话, 控制器会通过 Flow-Mod 消息下发一个新的流规则。但是交换机能够维持的流规则数量是有限的。每个流规则本身会占据内存, 同时他们还会维系一个计数器, 负责统计流经的数据包数和字节数等特征; 系统还需要对每个流规则进行相应的硬性存活时间和软性存活时间检查。每添加一个流规则都是对交换机的一次资源占用。目前大多数商用交换机都使用 TCAM 存储流表。TCAM 是一种基于硬件的表项查找方法, 支持对流表的并行访问, 也就是说可以进行一次比对就完成对流表中所有规则的匹配, 因而面对海量的数据流量, 效率可以大大提高, 且几乎不受表项数目增加的影响。但是其高昂成本和能耗导致很多商用交换机的存储容量极其受限。如 Pica 8 商用交换机只能够存放 8192 条流规则[25]。采用随机化数据包字段的 DDoS 攻击能够迅速占满交换机的流表, 导致新的规则无法添加, 从而无法处理到来的新的合法数据。

鉴于 Packet-In 泛洪攻击与数据层流表溢出攻击的巨大威胁, 我们在设计 DDoS 检测防御框架时, 需要重点考虑尽量少地触发 Table-Miss 选项, 在之前研究中只有很少人提及这个想法。

4.4.3 两阶段攻击

为了验证 SDDetector 在多种攻击场景下的检测效果, 实验需要构造针对 SDN 的 Packet-In 泛洪攻击和流表溢出攻击。因为触发 Packet-In 数据包只需要修改匹配域中的某个元素即可; 触发 Flow-Mod 修改流表的指令也是同样的原理。而本文方案细粒度流规则的匹配域包括了源 IP, 因而实验采用了源 IP 伪造技术模拟针对 SDN 的 Packet-In 泛洪攻击和流表溢出攻击。实际上, 如果匹配域包括了其他元素, 实验也可以通过伪造其他元素触发检测。

实验分为两个阶段分别进行。第一个阶段引入源 IP 伪造技术, 模拟 Packet-In 泛洪和流表溢出攻击, 第二个阶段, 采用真实源 IP, 模拟传统的 ACK 泛洪

攻击。下图 15 表示的从攻击源抓到的流量分布图, 图 16 表示从受害者端抓到的流量分布图。图中显示攻击源抓到的流量会随着时间起伏, 研究发现这是由于 CPU 和网络适配器的性能不足导致的, 即使在非 MiniNet 环境下测试的流量也呈现这种状况。鉴于其不会影响整体的实验进程, 我们大可以忽略不管。图 15 展示实验在 16:32:37 的时候激活了一次随机源 IP 的 SYN DDoS, 与此同时, 受害者端的下行流量开始迅速激增。随后模型探测到了可疑的攻击, 删掉了相应的流规则, 这导致在图 16 中 16:32:39 处流量的快速下降。紧接着, 新的细粒度的流规则开始创建。引入的时间函数告知攻击在 16:32:41 的时候正式被检测发现, 并立即添加了丢弃来自攻击源数据包的流规则。然而图 16 发现, 流量并没有在 16:32:41 直接下降到 0。经过多番测试和验证, 发现新的丢弃规则发送前已经进入交换机等待路由的数据包会继续向目标终端发送, 因而仍会维持一段时间的数据传输。可以看到在受害者端的流量 16:32:46 开始趋于 0, 而实际上从攻击源端看到的流量并未停止。在第二个阶段, 实验采用了真实源 IP 机制。流程图参见图 17, 图 18。攻击发起于 16:14:26 左右, 被检测于 16:14:32, 与前面情况相同, 受害者端的流量持续到 16:14:37 才结束。但是从攻击源的流量可以看到实际上在 16:15:02 的时候攻击才停止。

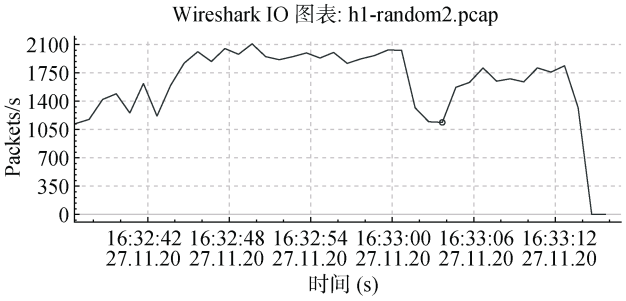


图 15 伪 IP 攻击源流量

Figure 15 Attacker Side with IP-spoofing

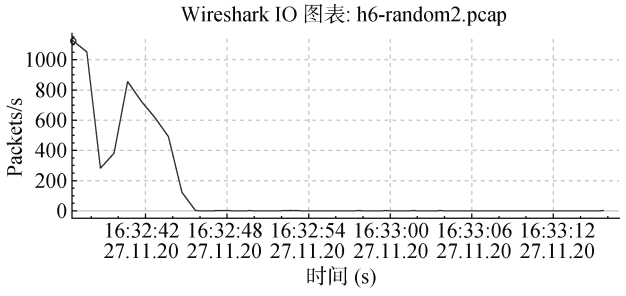


图 16 伪 IP 攻击目标流量

Figure 16 Victim Side with IP-spoofing

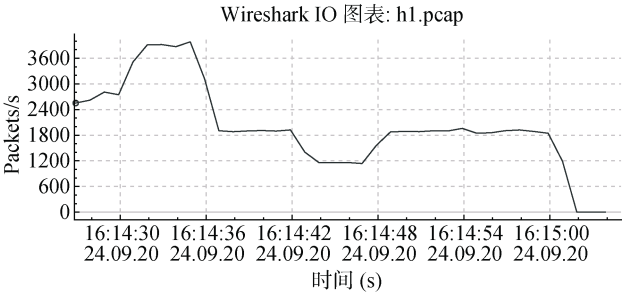


图 17 真实 IP 攻击源流量

Figure 17 Attacker Side with Real IP

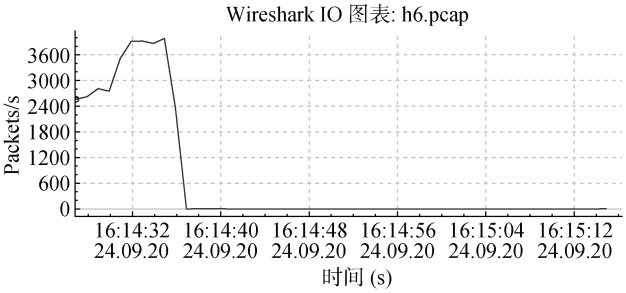


图 18 真实源 IP 攻击目标流量

Figure 18 Victim Side with Real IP

实验证明, 本文提出的融合熵检测算法和 SVM 检测算法的方案能够在多种 DDoS 环境下有效运行。Jin Ye[7]等人的方案也采用了 SVM 的检测算法, 但是他们从初始阶段就以细粒度的视角去做整体的检测, 这就意味着 DDoS 攻击时, 控制器需要处理比我们多得多的 Packet-In 信息, 交换机也会运维比我们更多的流规则。而且, 经过第一阶段的过滤, 我们的模型只对流向目标终端的流量进行进一步检测, 这能排除掉大量不必要的流量, 节省运算资源, 同时提高模型的检测准确率。

为了展示本文提到的方案在多种 DDoS 攻击场景的效率, 我们复现了 Jin Ye 的方案, 即从初始阶段就开始采用细粒度的流表规则, 并直接获取所有的流表统计信息进行检测。同时引入计时函数, 获取从收集统计数据到输出检测结果的耗时。

表 5 方案耗时比较

Table 5 Time consuming comparison of schemes

		Jin Ye 的方案	SDDetector
攻击频率	伪造源 IP	47.80s	1.5s
1000 pkt/s	真实源 IP	0.13s	0.13s
攻击频率	伪造源 IP	8.34s	2.01s
500 pkt/s	真实源 IP	0.13s	0.12s

可以看到在面对采用随机源 IP 技术的 Packet-In 泛洪攻击时, Jin Ye 的方案用时大大超过了我们的模

型; 尤其是攻击频率指数级增大的情况。原因很简单。拥塞的南向通道使得 Jin Ye 等人设想的统计信息获取阶段耗时太久。实验发现, 单纯算法提取特征并分类的耗时不超过 0.1s, 也就是说, 方案的核心负荷来源于获取流表统计信息的过程。而 SDDetector 在面对 Packet-In 泛洪攻击时, 优先基于熵的算法进行检测, 只需要采集到 1000 个 Packet-In 数据包就可以完成, 并在检测结果出来后, 迅速就阻断了攻击源, 防止持续的 Packet-In 数据包冲击控制器。这些因素共同作用就使得我们的处理时间只占到他们处理时间的四分之一。

5 结论与未来工作

5.1 结论

本文提出了一个 SDN 环境下的轻量级 DDoS 检测与缓解方案。特别考虑了细粒度的流规则会给交换机和控制器带来沉重的负担, 因而在两种不同的情形下配置了不同粒度的流规则。同时, 模型也融合了适用于不同攻击场景下的检测方案, 即针对 Packet-In 泛洪攻击的熵值检测方案, 和针对无源 IP 伪造的攻击手段的 SVM 检测方案。并且, 模型只针对流向可疑的攻击受害者的流量进行相应的细粒度流规则部署。这些举措都是为了尽可能地去降低交换机与控制器的运行负荷。通过实验发现, 常规直接检测方案面对源 IP 随机化时影响模型运行效率的最大因素来源于流表信息的获取, 这导致检测的速率与保留真实源 IP 时大相径庭。在这种情况下, 我们的模型会优先选择更高效的基于 Packet-In 的检测方案。但是面对如 HTTP 泛洪攻击和 HTTPS 泛洪攻击场景时, Packet-In 方案又会因为需要等待足够检测基准的 Packet-In 数据包, 而浪费时间, 此时 SVM 检测方案反而能脱颖而出了。而且熵检测方案提取的特征, 并不能很好地反映真实流经交换机的流量的绝大多数特性。SDDetector 还具备双层保险, 即使瞬发的正常流量, 如大热的电视剧播出时间, 或者假日的网络购物促销, 触发了阈值警告, 熵检测模型和 SVM 检测模型也能对其进行可靠的再分类。SVM 算法中选择的 5 个特征和熵检测算法选择的 2 个特征都能够明显地反映流量的异常。

5.2 未来工作

1) 实验发现如果交换机部署的是细粒度的流规则, 那海量的数据极易引发 Packet-In 泛洪, 从而导致控制器的处理压力巨幅上升, 这是 SDN 的一个巨大隐患, 但是绝大多数 SDN 环境下的 DDoS 研究都是基于单控制器[26]。未来我们需要尝试去解决这种由于中心化导致的单点失效的问题, 通过引入多控制器的平台, 实现控制器的负载均衡, 从而有效增加 SDN 网络的弹性生存能力。

2) 此外, 因为 RYU API 的限制, 本文的两个检测手段实际上是以串联的形式布局, 也就是需要先按照 Packet-In 的模式做检查, 如果熵检测模块没有得到检测结果, 且耗时超过正常值时, 我们会立马判定不存在 Packet-In 泛洪, 同时掐断 Packet-In 检测, 转而为 SVM 检测。未来需要把这个串联的布局, 改为完全并行的布局, 两个方案同时实施, 并且互相通信, 一方完成攻击检测之后, 另一方立即停止, 降低负荷的同时, 达到快速响应的目标。

3) 随着人们对 SDN 的进一步了解, 除了传统的泛洪式 DDoS 外, 还有一些精巧的 DoS 手段也会对 SDN 造成巨大的影响, 如利用流规则 Duration 的慢速攻击手段[27], 因为每个流规则都定义了一个软性存活时间, 在这个时间段如果没有匹配的数据包, 则流规则会被自动删除。攻击者于是每次都在软性存活时间截止前发送一个数据包, 从而能够长期维系流规则, 还能减少 Packet-In 和数据包数量上的异常。此外, 还有针对 SDN 拓扑发现阶段的攻击手段[28], 针对 OpenFlow 处理机制漏洞 buffered-packet 的攻击手段[29]都可能会造成 SDN 的拒绝服务。未来还需要补充针对这些攻击手段的检测机制。

致 谢 在此向对本文工作提出指导的老师、同学表示感谢, 以及对提出建议的评审专家表示感谢。

参考文献

[1] Maninder Pal Singh, Abhinav Bhandari. New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges[J]. *Computer Communications,* 2020: 509-527.

[2] Rajat Kandoi, Markku Antikainen. Denial-of-service attacks in OpenFlow SDN networks[C]. *Integrated Network Management(IM),*2015: 1322-1326.

[3] Xu J F, Wang L M, Xu Z. Survey on Resource Consumption Attacks and Defenses in Software-Defined Networking[J]. *Journal of Cyber Security,* 2020, 5(04): 72-95.
(徐建峰,王利明,徐震.软件定义网络中资源消耗型攻击及防御综述[J].*信息安全学报*,2020,5(04): 72-95.)

[4] Jiahao Cao, Qi Li, Renjie Xie, et al. The CrossPath Attack: Disrupting the SDN Control Channel via Shared Links[C]. *USENIX Security Symposium*, 2019: 19-36.

[5] Rui Wang, Zhiping Jia, Lei Ju. An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking[C].

TrustCom/BigDataSE/ISPA (1), 2015: 310-317.

[6] Qiao Yan, F. Richard Yu, Qingxiang Gong, et al. Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges[J]. *IEEE Communications Surveys and Tutorials* 2016, 18(1): 602-622

[7] Jin Ye, Xiangyang Cheng, Jian Zhu, et al. A DDoS Attack Detection Method Based on SVM in Software Defined Network[J]. *Security and Communication Networks*, 2018: 9804061:1-9804061:8

[8] Da Yin, Lianming Zhang, Kun Yang. A DDoS Attack Detection and Mitigation with Software-Defined Internet of Things Framework[J]. *IEEE Access 6,* 2018: 24694-24705

[9] Xiang You, Yaokai Feng, Kouichi Sakurai. Packet in Message Based DDoS Attack Detection in SDN Network Using OpenFlow[C]. *International Symposium on Computing and Networking (CANDAR)*, 2017: 522-528

[10] Lingyu Zhang, Ying Wang, Xuxia Zhong, et al. Resource-saving replication for controllers in multi controller SDN against network failures[C]. *NOMS,* 2018: 1-7

[11] Xupeng Luo, Qiao Yan, Mingde Wang, et al. Using MTD and SDN-based Honeypots to Defend DDoS Attacks in IoT[C]. *ComComAP,* 2019: 392-395

[12] Yan-Ting Chen, Chi-Yu Li, Kuochen Wang. A Fast Converging Mechanism for Load Balancing among SDN Multiple Controllers[C]. *IEEE Symposium on Computers and Communications*, 2018: 682-687

[13] Alexander Craig, Biswajit Nandy, Ioannis Lambadaris, et al. Load balancing for multicast traffic in SDN using real-time link cost modification[C]. *IEEE International Conference on Communications*, 2015: 5789-5795

[14] OpenFlow specification: *http://opennetworking.org/.*

[15] Da Yin, Lianming Zhang, Kun Yang. A DDoS Attack Detection and Mitigation With Software-Defined Internet of Things Framework[J]. *IEEE Access*, 2018: 24694-24705

[16] Seyed Mohammad Mousavi, Marc St-Hilaire. Early detection of DDoS attacks against SDN controllers[C]. *International Conference on Computing, Networking and Communications*, 2015: 77-81

[17] Shariq Murtuza, Krishna Asawa. Mitigation and Detection of DDoS Attacks in Software Defined Networks[C]. *International Conference on Contemporary Computing*, 2018: 1-3

[18] Rodrigo Braga, djard de Souza Mota, Alexandre Passito. Lightweight DDoS flooding attack detection using NOX/OpenFlow[C]. *IEEE Conference on Local Computer Networks*, 2010: 408-415.

[19] Shideh Yavary Mehr, Byrav Ramamurthy. An SVM Based DDoS Attack Detection Method for Ryu SDN Controller[C]. *CoNEXT Companion*, 2019: 72-73

[20] Quamar Niyaz, Weiqing Sun, Ahmad Y. Javaid. A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)[J]. *EAI Endorsed Transactions on Security and Safety*, 2016, 4(12).

[21] I. Gde Dharma Nugraha, Fiqri Muthohar, J. D. Alvin Prayuda, et al. Time-based DDoS detection and mitigation for SDN controller[C]. *17th APNOMS* 2015: 550-553

[22] Yunhe Cui, Lianshan Yan, Saifei Li, et al. SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks[J]. *Journal of Network and Computer Applications*, 2016: 65-79

[23] Neelam Dayal, Shashank Srivastava. Analying behavior of DDoS attacks to identify DDoS detection features in SDN[C]. *International Conference on Communication Systems and Networks*, 2017: 274-281

[24] Heng Cui, Ghassan O. Karame, Felix Klaedtke, et al. On the Fingerprinting of Software-Defined Networks[J]. *IEEE Transactions on Information Forensics and Security*, 2014, 11(10): 2160-2173

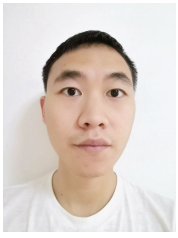
[25] X. Jin, H.H. Liu, R. Gandhi, et al. Dynamic scheduling of network updates[C]. *ACM SIGCOMM Conference*, 2014: 539-550.

[26] Shahzeb Haider, Adnan Akhunzada, Iqra Mustafa, et al. A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks[J]. *IEEE Access*, 2020, 8: 53972-53983

[27] X. Wu, M. Liu, W. Dou, et al. DDoS Attacks on data plane of software-defined network: are they possible[J] *Security and Communication Networks,* 2016: 5444–5459.

[28] Eduard Marin, Nicola BucciolMauro Cont. An In-depth Look Into SDN Topology Discovery Mechanisms: Novel Attacks and PracticalCountermeasures[C]. *Conference on Computer and Communications Security*, 2019: 1101-1114

[29] Jiahao Cao, Renjie Xie, Kun Sun, et al. When Match Fields Do Not Need to Match: Buffered Packets Hijacking in SDN[C], *Network and Distributed System Security Symposium*, 2020



贾锟 于 2012 年在北京大学基础数学专业获得学士学位。现在中国科学院大学网络安全专业攻读博士学位。研究领域为僵尸网络、DDoS、SDN 网络的安全防护等。Email:jiakun@iie.ac.cn

王君楠 于 2017 年在南开大学获得信息安全与法学双学位。现在中国科学院大学网络安全专业攻读博士学位。研究兴趣包括: 对抗机器学习, 恶意流量检测等。Email: wangruonan@iie.ac.cn

刘峰 于 2003 年获得山东大学计算机学院博士学位。现在中国科学院大学信息工程研究所担任博士生导师。研究兴趣包括: 网络攻防, 视觉安全, 网络空间战略等。Email: liufeng@iie.ac.cn