基于软件定义网络的 Crossfire 攻击防御方法

郭雷[1]，荆山[1*]，魏亮[2]，赵川[3]

（1. 济南大学信息科学与工程学院，山东 济南 250022；
2. 江苏省未来网络创新研究院，江苏 南京 211111；3. 泉城实验室，山东 济南 250103）

摘　要： 区别于常规的分布式拒绝服务攻击，利用僵尸网络发动的 Crossfire 攻击具有低速率不可区分的特性，这导致常规入侵检测系统难以防御此类攻击。针对该问题，设计一种检测防御 Crossfire 攻击的方法。该方法基于软件定义网络（SDN），首先利用网络瓶颈选择算法筛选出易受攻击的网络瓶颈节点与链路，在此基础上部署虚拟节点预防 Crossfire 攻击，虚拟节点应答可疑探测流，扰乱攻击者的攻击视图从而隐藏物理拓扑的网络瓶颈，并基于随机森林和双阈值自编码器检测僵尸网络，最后通过慢开始防御策略和局部快速重路由方法达到防御 Crossfire 攻击的目的。实验在 SDN 环境下进行，虚拟节点的部署能够使得瓶颈节点指标明显降低，构建的僵尸网络检测模型在精度、召回率、F1 值等方面相较于传统随机森林分类模型提高近 5 个百分点，防御方法能够在 10 s 内达到缓解 Crossfire 攻击的效果。实验结果表明，相对其他方法，所提方法能有效检测并缓解此类攻击，且在此过程中基本不会影响到合法流量在物理拓扑中的正常转发。

关键词： 软件定义网络；Crossfire 攻击；虚拟节点；僵尸网络检测；检测防御

中图分类号： TP309　　**文献标志码：** A　　**DOI：** 10.19678/j.issn.1000-3428.0067935

Crossfire Attack Defense Method Based on Software Defined Network

GUO Lei[1], JING Shan[1*], WEI Liang[2], ZHAO Chuan[3]

(1. School of Information Science and Engineering, University of Jinan, Jinan 250022, Shandong, China;
2. Jiangsu Future Networks Innovation Institute, Nanjing 211111, Jiangsu, China;
3. Quan Cheng Laboratory, Jinan 250103, Shandong, China)

【Abstract】 Unlike conventional Distributed Denial of Service(DDoS) attacks, Crossfire attacks launched by botnets are low-speed and indistinguishable, making them difficult for traditional intrusion detection systems to defend against. To address this issue, a method for detecting and defending against Crossfire attacks is proposed, based on a Software Defined Network(SDN). The method involves several steps. First, a network bottleneck selection algorithm identifies vulnerable network bottleneck nodes and links. On this basis, virtual nodes are deployed to prevent Crossfire attacks. These virtual nodes respond to suspicious probe flows, distorted the attacker's view, and obscured the network bottleneck in the physical topology. Botnet detection is performed using a random forest and a double-threshold autoencoder. Finally, a slow-start defense strategy and local fast rerouting method are adopted to mitigate crossfire attacks. Experiments conducted in an SDN environment show that deploying virtual nodes significantly reduces the bottleneck node index. The proposed botnet detection model performs approximately 5 percentage points better in terms of precision and recall compared to the traditional random forest classification model. The defense method effectively mitigates Crossfire attacks within 10 s. Experimental results show that the proposed method can effectively detect and mitigate such attacks in the SDN environment, with minimal impact on the normal forwarding of legitimate traffic in the physical topology.

【Key words】 Software Defined Network(SDN); Crossfire attack; virtual node; botnet detection; detection and defense

0　引言

随着计算机网络的迅速发展，全球各大组织机构加强了网络空间的安全防御，但网络攻击者也在不断改变其攻击手段，其中最为常见的攻击手段就是分布式拒绝服务（DDoS）攻击。传统计算机网络针对 DDoS 攻击的防御只能依赖于常规入侵检测系统[1]。软件定义网络（SDN）架构的出现弥补了传统网络的缺陷，该网络架构将控制与转发分离从而以更灵活的方式在网络控制层进行编

收稿日期： 2023-06-26　　**修回日期：** 2023-12-11

基金项目： 国家自然科学基金（62172258，61702218，61672262）；山东省自然科学基金（ZR2021LZH007）；山东省重点研发计划（2021CXGC010103）；泰山学者青年专家工程项目（tsqn202211280）。

通信作者 E-mail： *jingshan@ujn.edu.cn

程，相较于传统网络的分布式控制，SDN 有着能够快速获取全局网络视图的优势。所以，在 SDN 研究领域出现了很多针对 DDoS 攻击的有效解决方案，但多数方案仅针对常规 DDoS 攻击[2]，而忽略了针对网络骨干链路的 DDoS 攻击，这种攻击被称作链路泛洪攻击（LFA）。在 LFA 中，主要分为 Coremelt 和 Crossfire 两类攻击。Crossfire 攻击是由 Coremelt 攻击演变而来的，此类攻击占据 LFA 的主导地位，主要体现为 Crossfire 更易发动攻击且造成危害更大[3-4]。Crossfire 攻击者首先通过拓扑探测工具获取到攻击视图，确定将要攻击的网络瓶颈链路，然后利用僵尸网络发送大量低速率攻击流量，从而淹没骨干链路达到攻击的目的。在过去的几年里，此类攻击在实际互联网场景下影响了较多公司和机构[5]，攻击目标主要是网络骨干节点、企业网络以及校园网络等，造成了较大规模的网络故障。

由于 Crossfire 攻击发送低速率流量，此类攻击流量有着与合法流量极其相似的特点，因此传统机器学习分类算法的检测防御方法效果并不好。本文基于 SDN 的优势，提出了一种有效识别和防御 Crossfire 攻击的方法。针对其低速率不可区分的攻击特性，防御方案不再关注流速率等常规特征，而是重点进行提前预防，并且综合网络流在时间窗口中的疑似 Crossfire 攻击行为特征进行检测防御。该方法在筛选出拓扑中易受攻击的网络瓶颈之后，依据网络瓶颈部署虚拟节点从而提前预防攻击。同时，本文提出一种准确率与精度更高的僵尸网络检测模型，并应用一种效率更高的局部快速重路由方法，在此过程中维护一个黑名单，记录源 IP 是否为僵尸网络、路径探测次数以及被重路由次数等可疑行为的信息，由此来检测防御 Crossfire 攻击。本文的主要贡献如下：

1）引入虚拟节点预防 Crossfire 攻击，设计虚拟节点部署算法，隐藏网络瓶颈。

2）提出一种精度更高的基于随机森林和双阈值自编码器的僵尸网络检测模型。

3）基于黑名单应用一种慢开始检测防御策略，结合局部快速重路由算法，从根本上阻断 Crossfire 攻击，并将检测防御过程中对合法流量的伤害降到最低。

1 相关工作

以 Crossfire 攻击为主的 LFA 聚合低速率攻击流量，这就导致常规的入侵检测系统难以检测防御。LF-Shield 框架[6]基于 SDN，通过提取终端主机的流量特征来识别发起 LFA 流的恶意机器人。但该方法单纯使用卷积神经网络进行检测判断，容易对合法用户造成误判，尤其是针对未知的攻击数据流检测效果较差。CFADefense[7]方法采用了数理统计的方式，即通过丢包率、抖动、RTT 等识别攻击，但本质上是对网络状况进行监测，即只有当网络受到了一段时间的攻击，该方法才能识别攻击并进行防御。相关研究[8]表明，常见路径探测工具 traceroute 能够获取网络拓扑视图，所以文献[9]方法检测 traceroute 流量，其同样采用数理统计的方式。该方法设计了两个阈值检测 traceroute 数据包，通过判断阈值时间间隔内发送 traceroute 的次数来确定是否为攻击流量，但是该方案仅根据 Crossfire 攻击中可疑探测流这一项属性进行判断，在实际应用场景下不能保证检测准确率。

LFADefender[10]方案提出了两种检测思路：一种是通过多次重路由操作，最终确定恶意流量；另一种则是通过追踪 traceroute 数据包确定恶意流量。该方案能以较低开销快速检测防御 LFA，但发送 traceroute 探测的流量并不一定是恶意流量；另一方面，重路由模块开销较大，这使得攻击者有可能在重路由防御机制生效之前更快地改变目标链路。SPIFFY[11]开发快速流量工程算法以实现有效的带宽扩展，并提出可扩展的监控算法来跟踪可疑源的行为变化，但其本质还是被动式防御。Woodpecker[12]方法对于 Crossfire 攻击有较好的缓解效果，区别于现有的方法，该方法直接绕开恶意流量的检测，采用基于集中式流量工程的核心防御措施，使流量均衡，消除可能被对手利用的路由瓶颈。但该方法并没有考虑阻塞恶意流量，所以该方法无法根除 LFA。

近几年出现了利用主动防御技术来防御 Crossfire 攻击的方法。MVNAH[13]方法针对 Crossfire 攻击提出了一种多变体地址跳跃机制，该机制为受保护的诱饵服务器和具有多个随机网络地址的目标服务器创建变体，以增加不确定性从而迷惑敌手，以此降低敌手攻击效果。该方法不需要进行僵尸网络检测，但实际应用场景下消耗资源较多。文献[14-15]方法均采用了移动目标防御技术，文献[14]方法动态改变网络设置以欺骗攻击者，从而使发动攻击的成本非常高，文献[15]方法将可疑流量引入虚拟影子主机，使得 Crossfire 攻击无法按照预期执行，但这两种方法都存在检测防御成本较高

的缺点。SDHoneyNet[16]方法以及 BottleNet[17]方法提出了拓扑欺骗的概念。SDHoneyNet 方法首先提出软件定义蜜网的概念，在此基础上，BottleNet 方法基于图论动态改变蜜网吸引 Crossfire 攻击者，达到提前预防效果，但该方法并没有考虑到流量密度等动态指标的变化对攻击者的欺骗效果，该文献作者按照拓扑欺骗概念优化方案提出了 EqualNet[18]，该方法使每个网络转发节点虚拟出多个节点，这些虚拟节点均衡链路上的跟踪流分布，使敌手无法找出关键链路，从而显著增加执行 LFA 的成本，但是该方法要求每个网络转发节点具备虚拟化功能，实际场景下成本较高。文献[19]在传统网络背景下，考虑到网络拓扑变化通常代价高昂且缓慢，提出了评估拓扑变化对攻击检测效率影响的方法。

针对上述问题，本文首先从提前预防入手，部署虚拟节点，设计实施一套完整的防御方案，将检测防御过程中对合法流量的伤害降到最低。

2 系统概述

本文方法的整体设计如图 1 所示，主要分为网络瓶颈选择、虚拟节点部署及应用、Crossfire 攻击检测与防御三大模块。

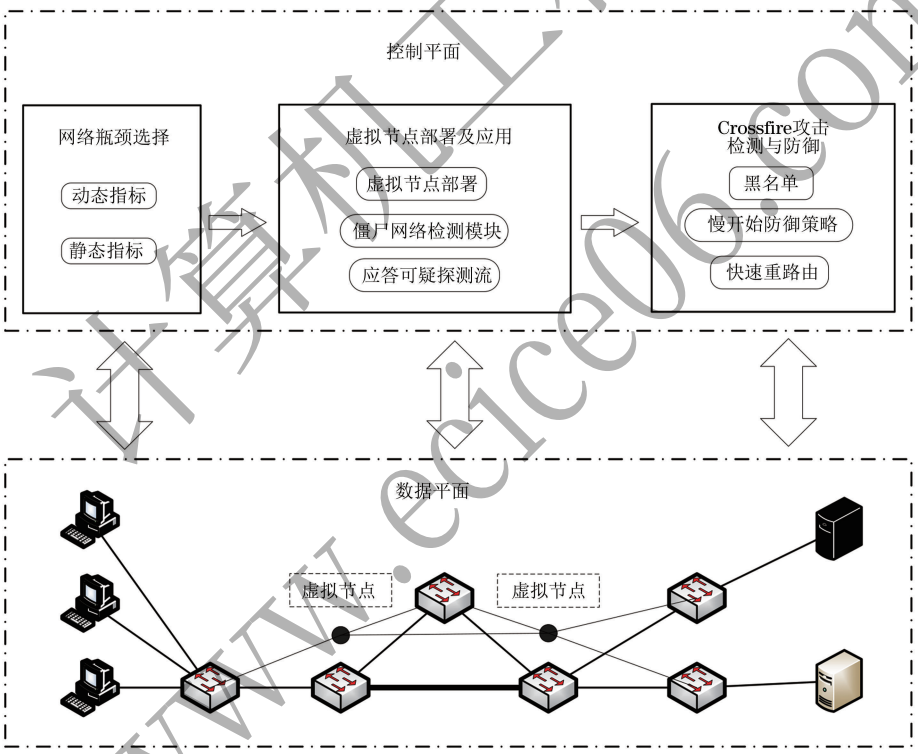

图 1 系统架构

Fig. 1 System architecture

系统首先会依据静态指标初步筛选网络瓶颈，之后根据动态指标最终确定网络瓶颈。在初步筛选网络瓶颈之后，部署虚拟节点，并控制其应答可疑探测流量，隐藏物理拓扑的网络瓶颈，从而达到提前预防 Crossfire 攻击的目的。考虑到攻击者会控制僵尸网络发送 Crossfire 攻击流量，系统拟采用一种基于随机森林和双阈值自编码器的分类模型来检测僵尸网络。在 Crossfire 攻击检测与防御模块中，维护黑名单，通过慢开始防御策略实现对 Crossfire 攻击的检测并下发丢弃流表完成防御。快速重路由部分则是对该系统补充加强的部分，是一种轻量级的方法，若攻击者绕过了系统的所有检测防御措施发动攻击，造成瓶颈链路拥塞，系统将采用快速重路由从而尽可能地减轻合法用户受损情况。

3 系统组成

3.1 网络瓶颈选择

在 SDN 架构下，控制平面能获取到数据平面的全局视图[20]，利用控制器的 REST API 可直接获取整个数据平面的多数网络性能指标，包括整体拓扑视图以及网络流表信息，而传统网络架构没有此类优势，网络管理员获取网络数据平面的性能指标大多依赖于特定设备厂商的网络管理软件，集中控制的优势较小。本文方法充分利用 SDN 全局视图的优势获取并计算 8 个网络瓶颈指标，由此快速筛选出网络瓶颈，为防御 Crossfire 攻击做准备。这些指标包括静态指标和动态指标，静态指标主要体现网络拓扑结构信息，动态指标主要体现拓扑中的流量

动态变化信息。8 个瓶颈指标如表 1 所示。

表 1 瓶颈指标

Table 1 Bottleneck indicators

名称	类别	类型
源-目的中心性(SDC)	静态	节点/链路
度中心性(DC)	静态	节点
接近中心性(CC)	静态	节点
中介中心性(BC)	静态	节点
流密度(FD)	动态	节点
消耗带宽比(CBR)	动态	链路
丢包率(PLR)	动态	链路
链路时延(LL)	动态	链路

1)源-目的中心性(SDC)。

Crossfire 攻击具有目标区域依赖性[4]。为了体现节点或链路针对目标区域的关键程度,本文提出了 SDC 指标,节点或链路的 SDC 指标越高,说明该节点或链路越易受到攻击。节点或链路 u 的 SDC 指标计算如下:

$$S_{\mathrm{SDC}}(u)=\frac{p_{\mathrm{path\_sd}}(u)}{p_{\mathrm{path\_sd}}} \tag{1}$$

式中:$p_{\mathrm{path\_sd}}$ 表示源区域 Source(S)到目标区域 Destination(D)的所有路径数目;$p_{\mathrm{path\_sd}}(u)$ 表示从 S 到 D 所有路径中经过某个节点或链路 u 的路径数量。

2)度中心性(DC)。

DC 指标也被应用在网络鲁棒性研究中[21],该指标主要衡量节点的度,节点的度越多,则在实际网络节点中拥有的链路越多,若攻击者破坏该节点,则整个网络会出现大面积故障。

3)接近中心性(CC)。

CC 指标描述了网络中某一节点与其他节点之间的接近程度。如果节点到图中其他节点的最短距离都很小,那么它的 CC 指标就很高。而在实际网络中,多数负载均衡方案的重路由路径会经过此类节点[22]。一个节点到其他节点的平均最短距离越小,该节点的 CC 指标就越大。

4)中介中心性(BC)。

BC 指标描述了一个节点处于任意两个节点最短路径之中的概率[23]。在实际网络拓扑中,该指标直接体现了某个节点在多个最短路径中的关键程度。

5)流密度(FD)。

在多数实际场景中,FD 至少在几个小时内基本保持不变,因此,FD 一般作为攻击者选择攻击目标时稳定可靠的指标[4]。流密度具有目标区域依赖性,结合该特性,节点 u 的 FD 计算如下:

$$F_{\mathrm{FD}}(u)=\frac{f_{\mathrm{flow\_sd}}(u)}{f_{\mathrm{flow}}(u)} \tag{2}$$

式中:$f_{\mathrm{flow\_sd}}(u)$ 表示节点 u 获取到的从源区域 Source(S)到目标区域 Destination(D)的所有流条目;$f_{\mathrm{flow}}(u)$ 表示节点 u 获取到的全部流条目。

6)消耗带宽比(CBR)。

CBR 越大,发起 Crossfire 攻击所需要的僵尸主机数量越少,相应的攻击成本越低,所以攻击者优先选择 CBR 较大的链路[4]。链路 u 的 CBR 计算如下:

$$C_{\mathrm{CBR}}(u)=\frac{(n_{\mathrm{TR}\_u}(t_1)-n_{\mathrm{TR}\_u}(t_0))\times 8}{(t_1-t_0)\times B_u} \tag{3}$$

式中:$n_{\mathrm{TR}\_u}(t)$ 表示 t 时刻链路 u 传输的字节总数;t_0 和 t_1 分别表示之前和当前时刻;B_u 表示链路 u 的带宽。

根据上述公式,本文提出了网络瓶颈选择算法,将所有的瓶颈指标归一到节点。算法所选的 8 个瓶颈指标依据来源于分析对应参考文献,同时在 4.2 节还进行了实验测试,测试结果证明各节点 8 个指标趋势与受到 Crossfire 攻击时各节点设备负载趋势基本一致,这表明 8 个指标均能反映节点瓶颈情况,所以本文瓶颈选择算法将综合该 8 个指标筛选网络瓶颈。算法首先获取 SDC、DC、CC 和 BC 指标,将所有链路的瓶颈指标累加到对应节点上并计算其平均数,由此计算出每个节点综合静态指标(SBI),依据 SBI 指标将各节点降序排序得到集合 SET_PSBI,依据 bp 比率(要选择出来的网络瓶颈数量占网络总节点数量的比率)初步选择网络瓶颈,为后续虚拟拓扑生成及应用做准备。在初步选出的瓶颈节点上获取动态指标 FD、CBR、PLR 和 LL,这其中涉及到如何将 CBR、PLR 以及 LL 这 3 种链路指标归一到对应节点上。以 CBR 为例,本文方法将节点连接的所有链路 CBR 累加,这样每个节点均有一个 CBR 累加和,将每个节点的 CBR 累加和在所有节点 CBR 累加和中的占比作为归一化到节点的 CBR_normalize,PLR 以及 LL 均为此种计算方法。之后综合所有动态指标计算初步选出的瓶颈节点的网络瓶颈指标(NBI),降序排列得到最终瓶颈节点及其瓶颈指标集合 Nodes_NBI,算法伪代码如算法 1 所示。

算法 1 瓶颈选择

输入 Source network-destination network, bp ratio

输出 Nodes_NBI

1. Find SDC, DC, CC, BC using Equations (1), (2), (3) and (4)

2. Compute SBI

3. Get the set of SBI: SET_PSBI

4. Sort(SET_PSBI, 'descend')

5. Compute FD, CBR, PLR, LL from SET_PSBI×bp ratio

6. Compute CBR_normalize, PLR_normalize, LL_normalize

7. Compute NBI

8. Get the set of each node's NBI: SET_NBI

9. Sort(Nodes_NBI, 'descend')

3.2 虚拟节点部署及应用

在依据 SBI 指标初步确定的网络瓶颈节点上部署虚拟节点,算法伪代码如算法 2 所示。网络管理员输入物理拓扑 G 以及瓶颈节点集合 $U_{\text{bottleneck}}$,算法输出虚拟节点部署策略。算法首先为瓶颈节点集合 $U_{\text{bottleneck}}$ 中所有瓶颈节点 v_{b0}～v_{bn} 各虚拟出一个虚拟节点分别对应 v_0～v_n,之后开始建立虚拟链路,算法会在瓶颈节点集合 $U_{\text{bottleneck}}$ 中选取瓶颈节点 v_{bi},首先为 v_{bi} 上的虚拟节点 v_i 添加虚拟链路,判断 v_{bi} 附近的所有节点 $v_{bi\_neighbor}$ 是否在 $U_{\text{bottleneck}}$ 中,若不在,则为 v_i 和 $v_{bi\_neighbor}$ 建立虚拟链路,若在,则找到 $v_{bi\_neighbor}$ 对应的虚拟节点 $v_{i\_neighbor}$,为 v_i 和 $v_{i\_neighbor}$ 建立虚拟链路。以此循环下去,最终为瓶颈节点集合 $U_{\text{bottleneck}}$ 中对应的所有虚拟节点建立虚拟链路。

算法 2 虚拟节点部署

输入 $G=(V, E)$, $U_{\text{bottleneck}}$

输出 虚拟节点部署策略

1. for v_{bi} in $U_{\text{bottleneck}}$
2. add a virtual node $v_i \rightarrow v_{bi}$
3. end for
4. for v_{bi} in $U_{\text{bottleneck}}$
5. get node set $N_{\text{neighbor}}(v_{bi})$
6. for $v_{bi\_neighbor}$ in $N_{\text{neighbor}}(v_{bi})$
7. if $v_{bi\_neighbor}$ in $U_{\text{bottleneck}}$
8. add a virtual link (v_i, $v_{i\_neighbor}$)
9. else
10. add a virtual link(v_i, $v_{bi\_neighbor}$)
11. end if
12. end for
13. end for

那么在实际工程应用下,如果网络节点设备支持虚拟化,则可直接在对应瓶颈节点上部署虚拟节点,应用该算法,若不支持,考虑到成本,则可以基于 SDN 控制器的优势,对可疑探测流量进行路径规划,即针对常见 traceroute 探测流,控制器下发策略至节点设备,使之构造虚假探测流回应,类似于虚拟节点应答 traceroute 流量,从而达到降低节点瓶颈指标的效果。

如图 2 所示,需要保护的瓶颈节点为 s_2 和 s_4,则分别在这两个节点上部署虚拟节点 v_0 和 v_1,并将其与邻居节点建立虚拟链路,在此过程中,监测可疑探测流量,控制虚拟节点 v_0 和 v_1 应答探测流,使瓶颈节点也暴露在攻击视图中,那么由于虚拟节点的作用,瓶颈节点指标也会降低,从而能够以较小代价混淆攻击视图,增加攻击者攻击成本。

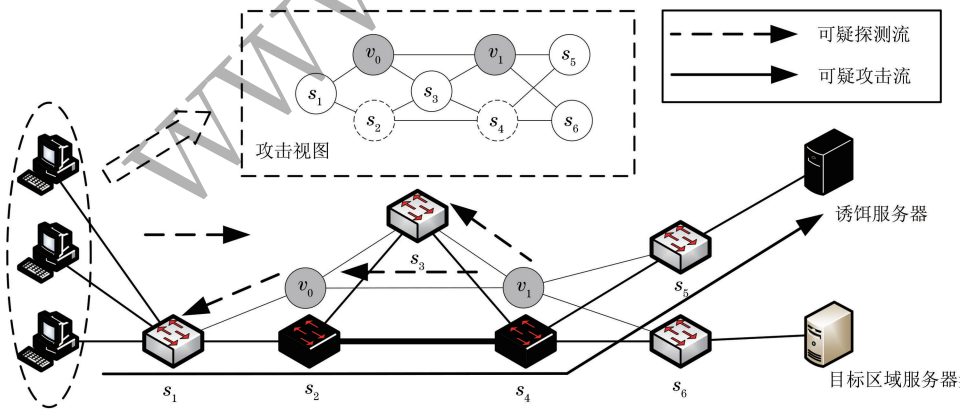

图 2 虚拟节点可疑流量重定向

Fig. 2 Redirection of suspicious traffic in the virtual node

因为发动 Crossfire 攻击需要调用大量的僵尸主机,所以本文方法在瓶颈节点部署僵尸网络检测模型,判断流量是否为僵尸主机,将判断结果作为后续黑名单中的一个重要属性。

检测算法采用随机森林和双阈值自编码器的混合集成算法。采用随机森林分类进行入侵检测的研究示例包括文献[24-25]算法。同时文献[26-27]的工作还得出结论,在他们的比较研究中,随机森林是入侵检测领域的顶级算法之一,但是随着近几年深度学习的迅速发展,其综合优势超过了像随机森林这种传统机器学习分类算法。随机森林在分类不平衡样本时效果不佳,而在本应用场景下,检测僵尸网络即为一种不平衡样本的处理情况,因为网络中多数流为合法流,极少数情况下才为恶意的僵尸网络

流，所以本文方案应用了自编码器来弥补随机森林分类过程中的劣势。方案利用 SDN 控制器提取到每一条流尽可能多的特征信息，包括流持续时间、协议类型、源端口、目的端口、流状态、TOS 值、流数据包总量、流字节总量、源字节总量等 9 种特征信息，检测模型利用这 9 种特征判断流的源是否为僵尸主机。

自编码器网络构造如图 3 所示，其由编码器(Encoder)和解码器(Decoder)组成，包括 1 个输入层、5 个隐藏层和 1 个输出层，自编码器选择 9 个输入输出维度对应 SDN 控制器获取到的 9 个特征，隐藏层分别对应 7、5、3、5 和 7 个神经元。自编码器可以重建输入，那么利用大量合法流量训练得到的自编码器模型，在应用过程中其重建输入输出的误差可以作为识别异常(僵尸网络)流量的异常分数，即重构误差较小时，可以判断其为正常流量，当重构误差较大时，可以判断其为异常(僵尸网络)流量。为了配合随机森林分类器提高整体检测模型的准确率、精度等，本文方案使用了双阈值判断，即采用低阈值确定合法流量，采用高阈值确定异常(僵尸网络)流量，双阈值选择由网络管理员根据自编码器模型训练情况确定。随机森林和自编码器的训练如图 4 所示，用训练集中合法流和异常(僵尸网络)流来训练随机森林，而仅用训练集中的合法流来训练自编码器。这样的训练方式，在保证随机森林正常训练的基础上，使得自编码器对合法流量的重构误差较小，应用过程中便于识别重构误差较大的异常(僵尸网络)流量。

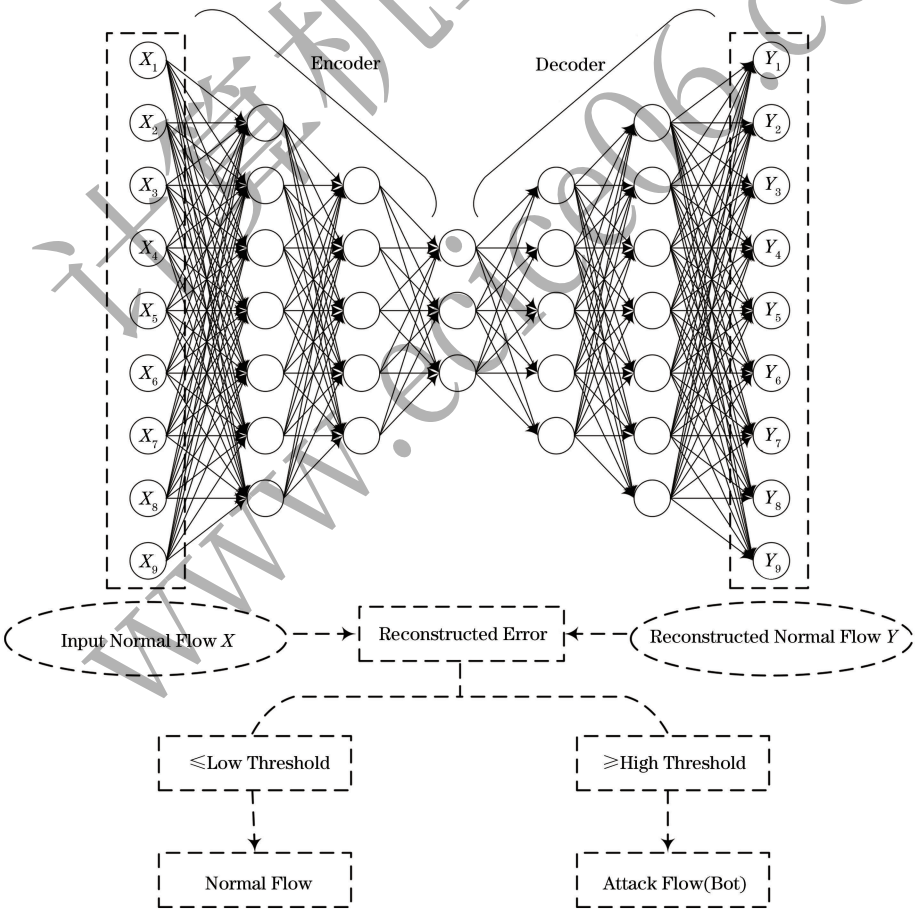

图 3 自编码器网络构造

Fig.3 Network construction of autoencoder

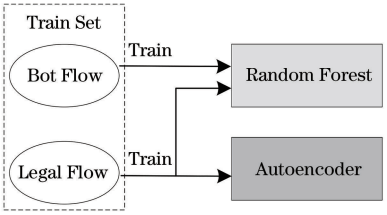

图 4 模型训练

Fig.4 Model training

本文方案模型应用过程如图 5 所示，当输入网络流之后，首先通过随机森林分类模型进行初步判断，若初步判断结果为合法流，则将其输入到自编码器得到重构误差，判断误差是否大于等于所设的高阈值，若是，则最终确定该流为异常(僵尸网络)流，否则最终确定该流为合法流；若初步判断结果为僵尸网络流，则还是将其输入到自编

码器得到重构误差，判断误差是否小于等于所设的低阈值，若是，则最终确定该流为合法流，否则最终确定该流为异常(僵尸网络)流。通过随机森林和自编码器双重保障以及双阈值，能够进一步提高精度，便于后期检测防御过程中维护更准确的黑名单，更迅速地完成防御。

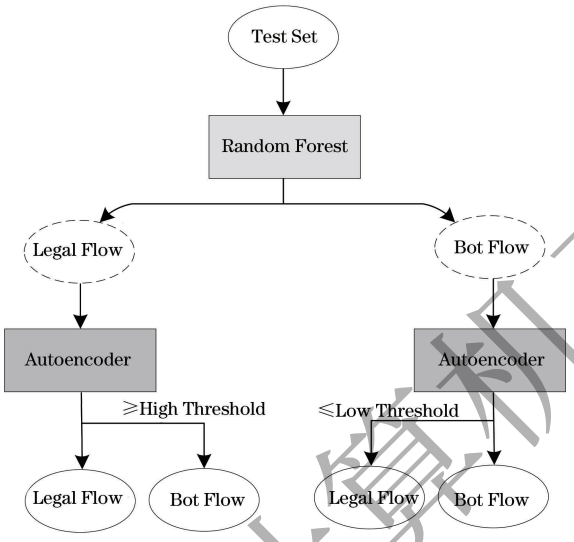

图 5 模型测试

Fig.5 Model testing

3.3 检测与防御

在可疑探测流被引入虚拟节点的同时，也将该探测流的源 IP 计入黑名单，如表 2 所示，同时，该表单还会记录该源 IP 是否被识别为僵尸网络、可疑探测次数以及该源 IP 经过瓶颈网络拥塞的次数等。根据以上指标维护的黑名单中的各属性计算对应 IP 的可疑分数，分数越高行为越可疑。表 2 中的前 2 个指标各占 10 分，第 3 个指标占 20 分。关于表中“是否为僵尸主机”属性分数计算，若源 IP 被检测为僵尸主机，则该项属性置为 10 分，否则置为 0 分。关于表中“可疑探测次数”属性分数计算，首先计算该源 IP 探测次数占据表中所有源 IP 探测次数的比率，由此比率分得该属性的 10 分，关于“经过瓶颈网络拥塞次数”的属性分数计算方式也是如此。若表中仅有 3 条可疑流，如表 2 所示，源 IP 为“207.43.26.123”的流，其可疑分数计算过程为 10+10×(1/4)+20×(1/5)=16.5。

表 2 黑名单

Table 2 Blacklist

源 IP	是否为僵尸主机	可疑探测次数/次	网络拥塞次数/次	可疑分数/分
207.43.26.123	是	10	3	16.5
186.21.34.162	否	20	6	13.0
202.194.68.10	是	10	6	20.5

本文根据黑名单应用一种慢开始防御策略。该策略由网络管理员输入正常流量转发时间窗口(NTW)、链路时延阈值(LLT)以及时间增量(Increment)，输出流表规则。首先，控制器结合表中源 IP 下发丢弃流表，流表超时选项对应可疑分数。判断源 IP 是否为僵尸网络流，若是则将流表超时选项设置为对应可疑分数的 2 倍，当所有丢弃流表失效之后，开启一个 NTW，利用 NTW 期间获取到的瓶颈链路时延，判断当前 NTW 的瓶颈链路时延(LL_Current)是否小于 LLT，若是，则清空黑名单，结束本轮的 Crossfire 攻击检测与防御，否则继续判断 LL_Current 是否小于上一个 NTW 的瓶颈链路时延(LL_Previous)，若是，则增加 Increment 到 NTW，并重新按照表中源 IP 下发对应可疑分数的超时丢弃流表，否则将减少 Increment 到 NTW，并重新按照表中源 IP 下发对应可疑分数的超时丢弃流表。算法核心是优先保障流量转发，并在此过程中逐渐过滤 Crossfire 攻击流量。算法伪代码如算法 3 所示。

算法 3 慢开始防御

输入 NTW，LLT，Increment

输出 Flow rules

```
while LL_Current≥LLT
    Distribute drop flow tables with a timeout option
    Select bot flow to distribute drop flow tables again
    Do normal forwarding for a NTW
    if LL_Current<LL_Previous
        NTW=NTW+Increment
    else
        NTW=NTW-Increment
    end if
end while
```

虽然本文方法有着多重防御，但考虑到最坏情况发生，即攻击者成功发动了对物理拓扑的攻击并造成了网络瓶颈的拥塞，所以本文提出了轻量级的局部快速重路由方法(Local Rerouting)，从而尽可能地降低攻击对网络的损害。区别于常见的最少连接负载均衡算法(Least Connections)[22]，本文方法在实际物理拓扑网络瓶颈的前一跳节点检测到可疑探测流之后，即触发预重路由模块，通过控制器获取瓶颈节点周围 2 跳距离以内的局部拓扑信息，从而得到该局部拓扑从源区域到目标区域的所有路径，并去除包含网络瓶颈的链路，然后通过控制器的南向接口获得交换机各个端口单位时间内的吞吐量以及交换机内存使用率、CPU 使用率等性能参数，根据这些参数计算每个网络节点的权值，设计基于

Dijkstra 的最短路径选择算法，最终选择权值最小的路径作为当前的最优路径。当物理拓扑网络瓶颈链路拥塞时，控制器根据提前计算好的最优路径在相应设备上下发流表规则，从而快速实现重路由。

4 实验结果与分析

4.1 实验环境与实施

本文所提出的方法在 Ubuntu 20.04.4 LTS 64 位和 8 GB 内存的主机上得到实现。本文利用 Mininet 模拟物理拓扑，使用 OpenDaylight 作为控制器，使用 Open vSwitch 部署虚拟节点，并使用 VXLAN 技术实现物理节点与虚拟节点的互联。所提方法基于 Python 编程语言实现。网络流量模拟采用 Python 中的 Scapy 模块。

网络瓶颈选择、虚拟节点部署及应用、Crossfire 攻击检测与防御三大核心模块均在控制平面基于 Python 语言编程实现。网络瓶颈选择模块利用 OpenDaylight REST API 中的“/network-topology”获取数据平面拓扑信息，并基于 Python 的 networkX 模块计算拓扑静态指标，利用 REST API 中的“/opendaylight-inventory:nodes/node/openflow:”计算流密度、消耗带宽比等动态指标。虚拟节点部署及应用模块利用 Python 代码运行 Open vSwitch 命令部署虚拟节点。Crossfire 攻击检测与防御模块利用 tcpdump 工具实时获取数据平面网络流量动态信息，通过 Python 的 rdpcap 模块解析提取关键特征，利用 OpenDaylight REST API 中的“/flow-node-inventory:table/tableid/flow/flowid”下发 JSON 格式的丢弃流表阻塞攻击源。针对网络时延、抖动、丢包率等难以通过 OpenDaylight REST API 直接计算获取的网络性能指标，实验方案在数据平面各交换机节点预置 Python 脚本作为网络监控代理与控制平面进行信息交互。

4.2 瓶颈选择实验评估

实验网络拓扑如图 6 所示，其中共有 8 个交换机，实验将所有链路带宽设置为 5 Mb/s。当拓扑正常运行时，H1、H2 主机区域分别向 Server1 区域发送 1 Mb/s 流量，H3 向 Server2 区域发送 1 Mb/s 流量，依据本文提出的网络瓶颈选择算法，可得到每个交换机对应的瓶颈指数。若选择网络瓶颈数量为该检测区域所有交换机的 30%，则检测到的网络瓶颈为 s_6 与 s_7。

在本实验方案中，网络时延超过 100 ms 认定网络拥塞。运行 Crossfire 攻击脚本，当 H1 区域发送速率增大至 3 Mb/s 之后，网络出现拥塞。瓶颈指标变化趋势如图 7 所示，其中：8 个虚折线分别表示在执行 Crossfire 攻击脚本之前，8 个瓶颈指标在各个交换机节点的变化趋势；实折线图表示在执行 Crossfire 攻击脚本之后，各交换机节点负载变化趋势。综合分析可知，8 个瓶颈指标以及综合瓶颈指标的变化趋势与各交换机节点负载变化趋势基本一致，综合瓶颈指标最高的 6 号交换机，在遭受 Crossfire 攻击时，设备负载也是最高的。该方案所计算出的设备瓶颈指标越高，由此在遭受 Crossfire 攻击时，处理的数据包越多，即设备负载越高，由此可以证明瓶颈指标选择的准确性以及瓶颈筛选方案的有效性。

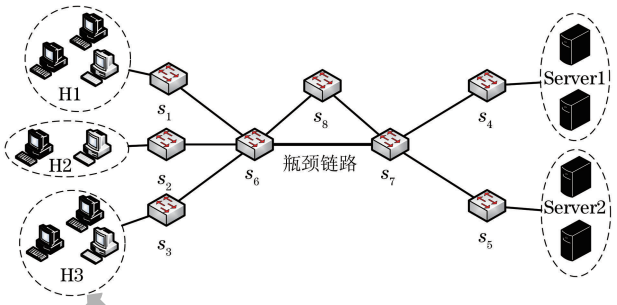

图 6 实验拓扑

Fig. 6 Experimental topology

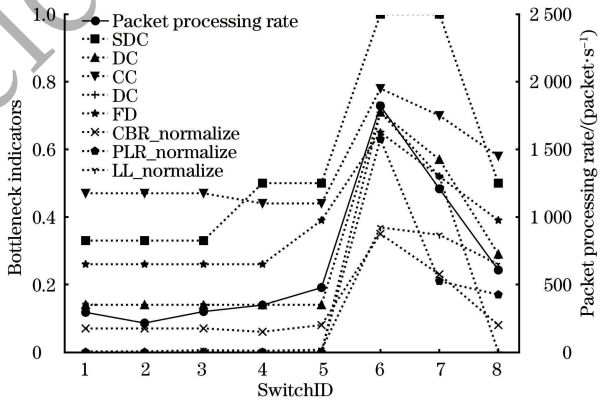

图 7 瓶颈指标变化趋势

Fig. 7 Trends of bottleneck indicators

4.3 虚拟节点部署及应用评估

利用瓶颈选择算法，若选择瓶颈比率为 0.3，同时应用虚拟节点部署算法，在 Topology Zoo 数据库[28]中的多个公开拓扑数据集上进行实验，分别在 8 个节点的 8_swicth、16 个节点的 peer1_2010_08 以及 37 个节点的 cernet_2006_11 等 3 个代表性拓扑上测试虚拟节点部署算法，结果分别如图 8～图 10 所示。从这 3 张实验结果图均可看出，在部署虚拟节点之后，各个节点瓶颈指标均明显降低。图中有部分节点的瓶颈指标得到了轻微提高，例如 8_swicth 中的节点 1～4 以及 peer1_2010_08 中的节点 3 等，此类节点在部署虚拟节点之前的瓶颈指标在整个拓扑中属于较低

的，在部署虚拟节点之后得到小幅提高，而原来瓶颈指标较高的节点明显降低，这样各节点瓶颈指标进一步拉近，更加有利于迷惑攻击者，达到了隐藏网络瓶颈的效果。

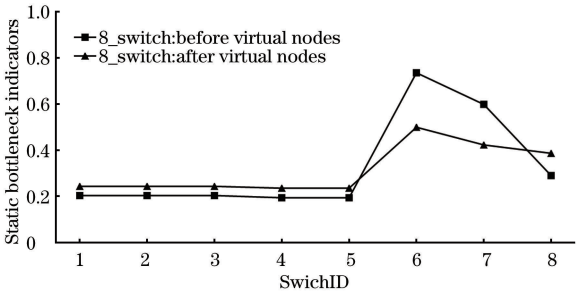

图 8 8_swicth 部署结果

Fig. 8 Deployment results of 8_swicth

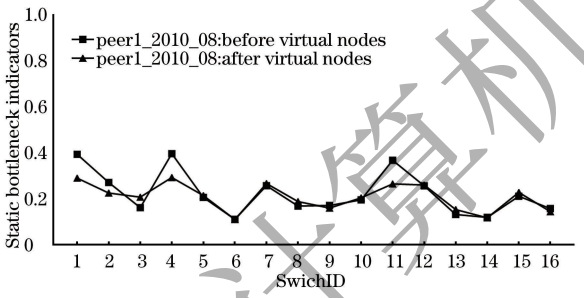

图 9 peer1_2010_08 部署结果

Fig. 9 Deployment results of peer1_2010_08

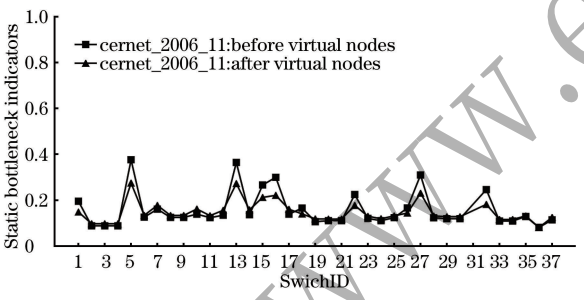

图 10 cernet_2006_11 部署结果

Fig. 10 Deployment results of cernet_2006_11

本文对进入虚拟节点的流量进行僵尸网络检测，采用CTU13数据集[29]中的capture20110811数据，共有1 780 700条数据，其中合法流有1 759 761条，僵尸网络流有20 939条。可以看出数据集中合法流量较多，而非法僵尸网络流量偏少，两者比例达到了84∶1，设置训练集测试集比例为7∶3，数据集划分情况分布如图11所示。

如果通过传统方法，仅利用随机森林对测试集进行分类得到混淆矩阵，测试集僵尸网络流共6 282条，但其中有348条被判断为合法流；测试集合法流量共527 928条，但其中有301条被判断为僵尸网络流，由此可以看出，随机森林在该测试集上的准确率较高，能够达到99%，但是其精度、召回率

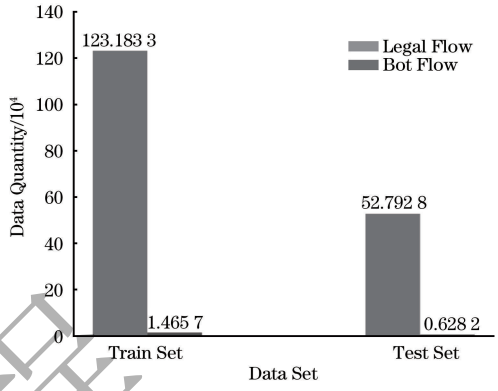

图 11 数据集划分

Fig. 11 Dataset partitioning

以及F1值等评估指标均在94%左右，尤其是对僵尸网络流样本的检测敏感性较差。

本文方案提出用自编码器来弥补随机森林的不足，仅利用训练集中的合法流来训练自编码器模型，设置训练批量(Batch Size)大小为256，学习率为0.001，采用MSE损失函数计算重构损失值。如图12所示，训练轮数15轮，损失值降低并稳定到0.32左右。

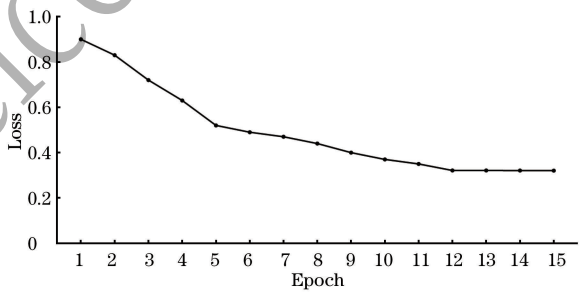

图 12 自编码器训练损失值

Fig. 12 Loss of training of autoencoder

自编码器在测试集上的应用效果如图13所示，可以看出：合法流量的重构误差分布在0.3～0.6，大多数的合法流量重构误差分布在0.4；僵尸网络流量的重构误差分布在0.6以上，分布范围较广，大多数的僵尸网络流量重构误差分布在0.7以上，所以基于自编码器重构误差分类合法流和僵尸网络流的可行性较大，且方案采取双阈值，可进一步提升精

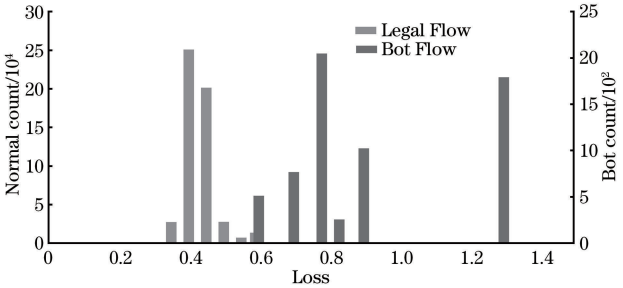

图 13 自编码器测试效果

Fig. 13 Testing effect of autoencoder

度。本文方法所提的整套僵尸网络检测模型综合评估效果与传统随机森林方法的对比如图 14 所示，自编码器低阈值设置为 0.45，高阈值设置为 0.7，由于本文方法增加了双阈值自编码器，因此在保证准确率的基础上，提高精度、召回率以及 F1 值等指标约 5 个百分点，弥补了随机森林算法检测小样本异常（僵尸网络）流量方面的劣势。

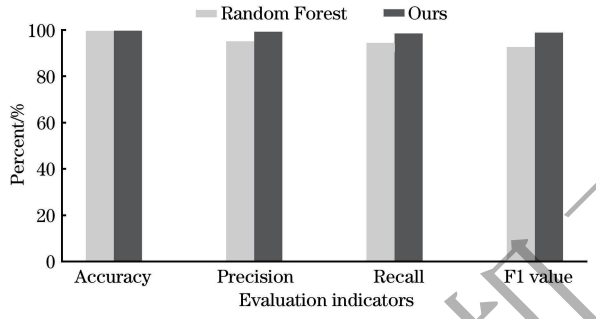

图 14 模型性能对比

Fig. 14 Model performance comparison

4.4 检测防御实验评估

为了测试基于黑名单的慢开始检测防御方法，实验选用图 6 所示的拓扑，瓶颈链路带宽设置为 2 Mb/s，运行含有 12 个随机源 IP 的 Crossfire 攻击脚本以及含有 12 个随机源 IP 的合法流量脚本，每个攻击的源 IP 发送速率随机且不超过 800 bit/s，并在发送攻击流量之前随机发送 6～12 个探测数据包，合法流量随机发送 1～2 个探测数据包。在模拟发动攻击之前，瓶颈选择模块以及虚拟节点部署及应用模块执行完毕，并开启基于黑名单的慢开始检测防御算法，由于仅执行合法流量脚本时，瓶颈链路延迟为 2 ms 左右，因此慢开始防御算法输入的链路时延阈值（LLT）设置为 5 ms，正常流量转发时间窗口（NTW）设置为 10 s，时间增量（Increment）设置为 2 s。

实验将本文方法与 LFADefender[10] 中的基于多次重路由方法（Rerouting）以及基于追踪探测包方法（Traceroute）进行比较分析，3 种方法的瓶颈链路的网络时延对比如图 15 所示，网络抖动对比如图 16 所示。实验在执行 Crossfire 攻击脚本之后，瓶颈链路网络延迟增大并触发慢开始检测防御算法。在第 5 s 时执行 Crossfire 攻击脚本，在第 10 s 时慢开始检测防御算法被触发，在 10 s 内完成初步的检测防御，并随着时间增加使得瓶颈链路网络时延稳定到 5 ms 左右，在第 80 s 时，攻击脚本发动了第 2 轮攻击，由于黑名单及慢开始防御策略的作用，第 2 轮攻击瓶颈链路几乎不受影响，时延稳定在 5 ms 以内。在本实验中，假设 LFADefender 每次重路由的时间忽略不计，LFADefender 追踪探测包的时间忽略不计且准确率为 100%。两轮攻击时间还是分别在第 5 s 和第 80 s。从网络时延对比来看，本文方法对比 LFADefender 的两种方法具有更快的检测防御速度。从网络抖动对比来看，本文方法在检测防御过程中能够将网络抖动稳定在一个较低的值，这使得在检测防御过程中，网络能够保证稳定传输。LFADefender 中 Rerouting 方法虽然在 15 s 左右达到缓解 Crossfire 攻击的效果，但瓶颈链路时延在此之后开始增加，出现链路拥塞现象。LFADefender 中 Traceroute 方法在网络抖动方面虽然维持在较低的值，但是缓解 Crossfire 攻击过程所需时间超过 30 s，并且在最开始的防御缓解过程中，瓶颈链路时延甚至有着增大的趋势，这说明该方法没有起到较好的防御缓解效果，这是由于大量合法用户也可能发送 traceroute 探测包，而该方法仅判断可疑 traceroute 探测包，所以该方法有着较高误报率从而无法达到理想的防御缓解效果。而本文所提方法能够在 10 s 内达到缓解 Crossfire 攻击的效果并能够稳定保持该防御效果，这是由于本文方法使用了一种准确率及精度更高的僵尸网络检测模型，同时提前维护的黑名单综合了多种 Crossfire 攻击判断属性，并且在此基础上的慢开始检测防御策略能够在保证不影响合法流量传输的基础上可持续的防御缓解此类攻击。

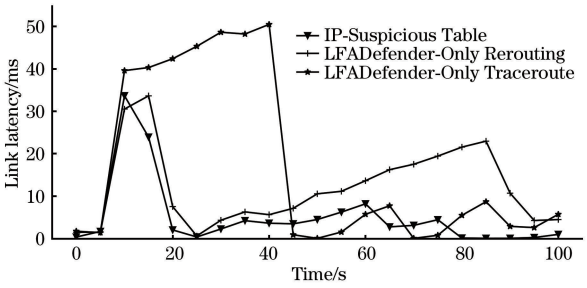

图 15 链路延迟对比

Fig. 15 Comparison of link latency

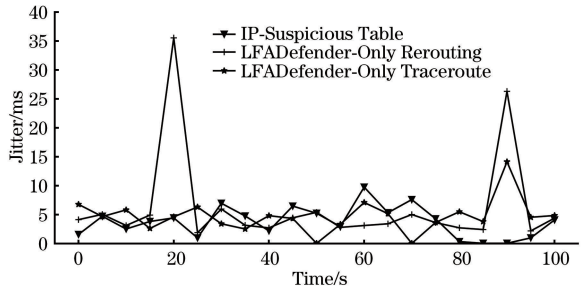

图 16 网络抖动对比

Fig. 16 Comparison of network jitter

关于局部快速重路由方法的测试效果，本实验选择了含有 36 个节点的 cernet_2006_11 拓扑，并在

该拓扑上对比了本文方法(Local Rerouting)与最少连接负载均衡算法(Least Connections)[22]。图 17、图 18 分别显示了随着时间的变化,瓶颈链路的网络时延与抖动变化,实验开始增加流量,使得网络出现拥塞,在第 10 s 启动 Local Rerouting 与 Least Connections,综合两图对比发现,Local Rerouting 能够在 30 s 内完成快速重路由,并且重路由后网络抖动较小,而 Least Connections 则需要耗费近 60 s 时间完成重路由,且重路由之后的网络抖动较大(网络抖动大于 25 ms),这表明网络传输不稳定。Local Rerouting 相较于 Least Connections 有着明显优势,这是因为 Local Rerouting 的核心优势在于局部快速重路由,该方法将计算范围缩小为瓶颈链路周围的局部拓扑,极大地提高了计算效率,在缓解 Crossfire 攻击造成的链路拥塞方面有较强的针对性。

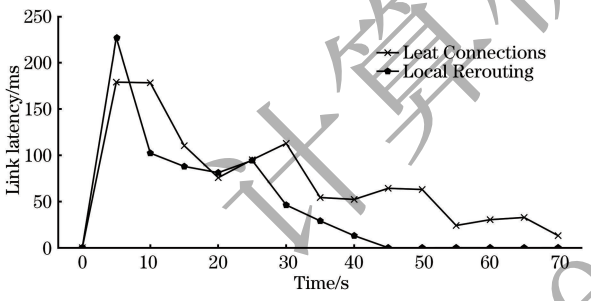

图 17 负载均衡链路延迟变化

Fig. 17 Variation in link latency during load balancing

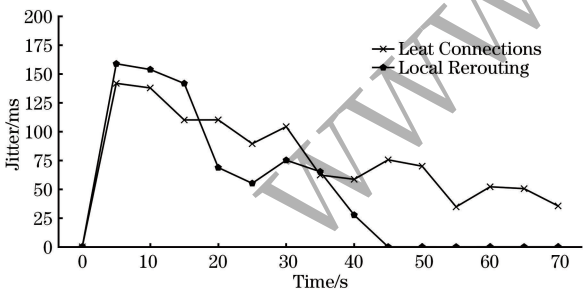

图 18 负载均衡网络抖动变化

Fig. 18 Variation in network jitter during load balancing

5 结束语

本文提出了一种预防和检测防御 Crossfire 攻击的方法,在筛选网络瓶颈的基础上,部署虚拟节点以预防 Crossfire 攻击,并提出一种基于随机森林和双阈值自编码器的僵尸网络检测模型,在此过程中,维护黑名单并结合慢开始防御策略实现对 Crossfire 攻击的检测防御,若攻击者实现 Crossfire 攻击并造成链路拥塞,本文方法中的局部快速重路由能够对拥塞链路进行快速缓解。实验结果表明,虚拟节点的部署可以隐藏网络瓶颈,满足提前预防 Crossfire 攻击的需求。本文方法所使用的僵尸网络检测模型相较于仅使用随机森林的模型在精度、召回率以及 F1 值等方面提高了近 5 个百分点。本文方法所提的基于黑名单的慢开始防御策略能够快速稳定地检测防御 Crossfire 攻击,局部快速重路由与常见的最少连接负载均衡算法相比有着更高的效率。在未来的工作中,将重点改进重路由模块,提高整个检测防御流程的时间效率。

参考文献

[1] 李可欣,王兴伟,易波,等. 智能软件定义网络[J]. 软件学报, 2021, 32(1): 118-136.
LI K X, WANG X W, YI B, et al. Intelligent software defined networking[J]. Journal of Software, 2021, 32(1): 118-136. (in Chinese)

[2] 徐玉华,孙知信. 软件定义网络中的异常流量检测研究进展[J]. 软件学报, 2020, 31(1): 183-207.
XU Y H, SUN Z X. Research development of abnormal traffic detection in software defined networking[J]. Journal of Software, 2020, 31(1): 183-207. (in Chinese)

[3] STUDER A, PERRIG A. The coremelt attack[C]// Proceedings of European Symposium on Research in Computer Security. Berlin, Germany: Springer, 2009: 37-52.

[4] KANG M S, LEE S B, GLIGOR V D. The Crossfire attack[C]//Proceedings of the IEEE Symposium on Security and Privacy. Washington D. C., USA: IEEE Press, 2013: 127-141.

[5] GitHub. February 28th DDoS incident report[EB/OL]. (2018-03-01)[2023-04-25]. https://github.blog/2018-03-01-ddos-incident-report/.

[6] XING J C, CAI J J, ZHOU B Y, et al. A deep ConvNet-based countermeasure to mitigate link flooding attacks using software-defined networks[C]//Proceedings of the IEEE Symposium on Computers and Communications. Washington D. C., USA: IEEE Press, 2019: 1-6.

[7] RAFIQUE W, HE X, LIU Z F, et al. CFADefense: a security solution to detect and mitigate crossfire attacks in software-defined IoT-edge infrastructure[C]//Proceedings of the 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems(HPCC/SmartCity/DSS). Washington D. C., USA: IEEE Press, 2019: 500-509.

[8] KANG M S, GLIGOR V D. Routing bottlenecks in the Internet: causes, exploits, and countermeasures[C]// Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. New York, USA: ACM Press, 2014: 321-333.

[9] NAKAHARA M, KAMIYAMA N. Detecting crossfire-attack hosts in search phase[C]//Proceedings of the 23rd Asia-Pacific Network Operations and Management Symposium(APNOMS). Washington D. C., USA: IEEE Press, 2022: 1-4.

[10] WANG J, WEN R, LI J Q, et al. Detecting and mitigating target link-flooding attacks using SDN[J]. IEEE Transactions on Dependable and Secure Computing, 2019, 16(6): 944-956.

[11] KANG M S, GLIGOR V D, SEKAR V. SPIFFY: inducing cost-detectability tradeoffs for persistent link-flooding attacks[C]// Proceedings of 2016 Network and Distributed System Security

Symposium. Reston, USA: Internet Society, 2016: 53-55.
[12] WANG L, LI Q, JIANG Y, et al. Woodpecker: detecting and mitigating link-flooding attacks via SDN[J]. Computer Networks, 2018, 147: 1-13.
[13] ZHOU B Y, PAN G N, WU C M, et al. Multi-variant network address hopping to defend stealthy crossfire attack [J]. Science China Information Sciences, 2020, 63(6): 169301.
[14] AYDEGER A, MANSHAEI M H, RAHMAN M A, et al. Strategic defense against stealthy link flooding attacks: a signaling game approach[J]. IEEE Transactions on Network Science and Engineering, 2021, 8(1): 751-764.
[15] HYDER M F, FATIMA T. Towards crossfire distributed denial of service attack protection using intent-based moving target defense over software-defined networking[J]. IEEE Access, 2021, 9: 112792-112804.
[16] KIM J, SHIN S. Software-defined HoneyNet: towards mitigating link flooding attacks[C]//Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). Washington D. C., USA: IEEE Press, 2017: 99-100.
[17] KIM J, NAM J, LEE S, et al. BottleNet: hiding network bottlenecks using SDN-based topology deception[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 3138-3153.
[18] KIM J, MARIN E, CONTI M, et al. EqualNet: a secure and practical defense for long-term network topology obfuscation [C] // Proceedings of 2022 Network and Distributed System Security Symposium. Reston, USA: Internet Society, 2022: 24-28.
[19] LIASKOS C, IOANNIDIS S. Network topology effects on the detectability of crossfire attacks[J]. IEEE Transactions on Information Forensics and Security, 2018, 13(7): 1682-1695.
[20] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow [J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.
[21] 穆俊芳,郑文萍, 王杰, 等. 基于重连机制的复杂网络鲁棒性分析[J]. 计算机科学, 2021, 48(7): 130-136.
MU J F, ZHENG W P, WANG J, et al. Robustness analysis of complex network based on rewiring mechanism [J]. Computer Science, 2021, 48(7): 130-136. (in Chinese)
[22] 周桐庆,蔡志平, 夏竞, 等. 基于软件定义网络的流量工程[J]. 软件学报, 2016, 27(2): 394-417.
ZHOU T Q, CAI Z P, XIA J, et al. Traffic engineering for software defined networks [J]. Journal of Software, 2016, 27(2): 394-417. (in Chinese)
[23] FREEMAN L C. A set of measures of centrality based on betweenness[J]. Sociometry, 1977, 40(1): 35.
[24] MARTEAU P F. Random partitioning forest for point-wise and collective anomaly detection—application to network intrusion detection[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 2157-2172.
[25] MISHRA A K, PALIWAL S. Mitigating cyber threats through integration of feature selection and stacking ensemble learning: the LGBM and random forest intrusion detection perspective[J]. Cluster Computing, 2023, 26(4): 2339-2350.
[26] CHOUDHURY S, BHOWAL A. Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection [C] // Proceedings of the International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials(ICSTM). Washington D. C., USA: IEEE Press, 2015: 89-95.
[27] ANBAR M, ABDULLAH R, HASBULLAH I H, et al. Comparative performance analysis of classification algorithms for intrusion detection system[C]//Proceedings of the 14th Annual Conference on Privacy, Security and Trust(PST). Washington D. C., USA: IEEE Press, 2016: 282-288.
[28] KNIGHT S, NGUYEN H X, FALKNER N, et al. The Internet topology zoo[J]. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1765-1775.
[29] GARCÍA S, GRILL M, STIBOREK J, et al. An empirical comparison of botnet detection methods[J]. Computers & Security, 2014, 45: 100-123.

编辑 金胡考