基于 eBPF 与 LSTM 的 DDoS 攻击检测系统

昌武洋, 付 雄, 王俊昌
南京邮电大学 计算机学院,南京 210000

摘 要:针对网络异常流量检测中的 DDoS 攻击检测,以往的基于深度学习的解决方案都是在脱离系统实体的数据集上构建模型和优化参数,提出并实现一种使用 Linux 内核观测技术 eBPF(extended Berkeley Packet Filter)与深度学习技术结合的基于网络流量特征分析的网络异常流量检测系统。系统采用 eBPF 直接从 Linux 内核网络栈最底层高效地采集网络流量特征数据,然后使用基于长短记忆网络 LSTM(Long Short Term Memory)构建的深度学习系统检测网络异常流量。在具体实现中,系统首先通过 Linux 内核网络栈最底层 XDP(eXpress Data Path)中的 eBPF 程序挂载点采集网络流量特征数据。之后,使用 LSTM 构建神经网络模型和预测分类。将系统应用于一个仿真实验网络环境得出的实验结果表明,系统的识别精确度达到 97.9%,同时,在使用该系统的情况下,网络中的 TCP 与 UDP 通信的吞吐率仅平均下降 8.53%。结果表明:系统对网络通信影响较低,同时也实现了较好的检测效果,具有可用性,为网络异常流量检测提供了一种新的解决方法。

关键词:攻击检测; Linux 内核观测技术;长短记忆网络;深度学习
中图分类号:TP393.0 **文献标识码**:A doi:10.16055/j.issn.1672-058X.2023.0002.006

DDoS Attack Detection System Based on eBPF and LSTM

CHANG Wuyang, FU Xiong, WANG Junchang
School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210000, China

Abstract: For DDoS attack detection in abnormal network traffic detection, previous deep learning-based solutions construct models and optimize parameters on datasets separated from system entities. This paper proposed and implemented a network anomaly traffic detection system based on network traffic characteristic analysis that combined Linux kernel observation technology eBPF (extended Berkeley Packet Filter) with deep learning technology. The system used eBPF to efficiently collect network traffic feature data directly from the bottom layer of the Linux kernel network stack, and then used a deep learning system based on the Long Short Term Memory (LSTM) to detect abnormal network traffic. In the specific implementation, the system first collected network traffic characteristic data through the eBPF program mount point in the bottom XDP (eXpress Data Path) of the Linux kernel network stack. LSTM was used to build neural network model and predict classification. The experimental results obtained by applying the system to a simulated experimental network environment showed that the recognition accuracy of the system reached 97.9%. At the same time, in the case of using this system, the throughput rate of TCP and UDP communication in the network dropped by only 8.53% on average. The results show that the system has a low impact on network communication, achieves better detection results, has the availability, and provides a new solution for abnormal network traffic detection.

Keywords: attack detection; Linux kernel observation technology; long short-term memory network; deep learning

收稿日期:2022-03-05 **修回日期**:2022-05-18 **文章编号**:1672-058X(2023)02-0036-08

作者简介:昌武洋(1998—),男,湖南益阳人,硕士研究生,从事高性能计算研究.
通讯作者:王俊昌(1982—),男,云南个旧人,博士,讲师,从事并行与分布式计算研究. Email:wangjc@njupt.edu.cn.

引用格式:昌武洋, 付雄, 王俊昌. 基于 eBPF 与 LSTM 的 DDoS 攻击检测系统[J]. 重庆工商大学学报(自然科学版),2023,40(2):36—43.
CHANG Wuyang, FU Xiong, WANG Junchang. DDoS attack detection system based on eBPF and LSTM[J]. Journal of Chongqing Technology and Business University (Natural Science Edition),2023,40(2):36—43.

1 引　言

DDoS 攻击根据攻击原理可分为两类,第一类是利用通信协议的漏洞及其实现系统的缺陷,通过发送一些异常的数据包让受害服务器在处理这些异常数据包的过程中出现异常情况,导致受害服务器无法为正常的请求服务。第二类就是攻击者向受害服务器发送大量的恶意数据包请求,其目的是占用受害服务器处理网络请求的资源,使其无法处理正常的网络请求。据此,可以设计一个系统从混杂的网络流量中分出网络异常流量。

eBPF 是一种通过在 Linux 内核函数置入观测点获取内核运行数据的技术,被广泛地用于网络分析开发[1]。其在 Linux 内核网络栈的挂载点 XDP[2]是其在内核中最早接触网络数据包的地方。eBPF 具有使用 Python 作为开发语言的前端工具集 BCC(BPF Compiler Collection)。通过 BCC 可以将内核中收集的网络流量数据交由分类模型处理。

sbulb[3]开发了一个使用 XDP 开发的 UDP 负载平衡器原型。以便在集群中的每个服务器实例之间共享状态。这个负载平衡器是一种通过修改 IP 地址工作的传输层 NAT 负载平衡器。

MicroBPF[4]利用 eBPF 从内核捕获 TCP 协议网络通信过程中的相关参数,用于微服务架构中的性能诊断。它探测两个级别的统计信息:流和数据包。流级统计目前有 16 个指标,如已发送但未确认的数据包、接受窗口大小等。数据包级别的统计数据是往返延迟的细分,包括 TCP 层、IP 层、MAC 层,网络的延迟和应用层延迟。

文献[5]研究的两个案例表明,eBPF 可用于多功能和高性能的数据包过滤应用。XDP 挂载在网络堆栈之前的最低级别,非常适合粗包过滤,如 DoS 预防。测量结果表明,与在内核中使用常见的包过滤工具执行类似任务相比,XDP 可以产生四倍的性能。

面对海量应用的问题排查和故障定位变得越来越复杂的情景,文献[6]提出了一种容器网络可观测性分析系统。这个系统基于 eBPF 非侵入式采集用户应用 L7/L4 层网络协议交互信息,然后使用机器学习方法对应用网络性能和问题进行分析诊断,分析不同应用的网络性能瓶颈并定位具体的实例 pod,实现与协议无关的网络性能问题定位和分析。

因此,本文基于 eBPF 的特性,提出并实现一种基于 eBPF 的服务软件无关的非入侵的 Linux 网络异常流量检测系统。它是一种非入侵式的,不需要被部署主机上其他服务系统额外操作的使用 Linux 内核可观测技术的网络异常流量检测系统。

机器学习及深度学习技术已经被使用到异常检测。拥有可靠的、公开可用的入侵检测系统评估数据集是该领域研究人员和开发者的基本关注点之一。Sharafaldin[7]通过分析自 1998 年以来的 11 个公开可用的用于入侵检测系统的数据集,发现这些数据集对攻击的识别效率受缺乏流量多样性等原因限制。Sharafaldin 生成一个新的包含 12 种网络攻击类型的可用于入侵检测系统的数据集,并使用随机森林回归算法从包含 80 个流量特征的数据集中提取最佳短特征集。之后,又使用七种常见的机器学习算法检测它们在新数据集上的性能和准确性。

Finelame[8]描述并评估了一种新的细粒度应用程序级 DoS 检测框架。这是一种使用了 K-means 算法的用于检测非对称 DoS 攻击的实用框架。在 Finelame 中,用户只需要标注自己的代码来标记请求处理的开始和结束。根据标注,Finelame 自动跟踪整个应用程序中的 CPU、内存、存储和网络使用情况。通过使用 eBPF 技术和,Finelame 以低开销和超细粒度做到这一点,使得它能够在网络请求离开系统之前以及在网络请求试图耗尽资源时检测到不同的请求。

LUCID[9]是一个实用的、轻量级的深度学习 DDoS 检测系统,它利用卷积神经网络(CNN)构建了一个将流量分为恶意流量或良性流量的系统,基于卷积神经网络的模型进行训练,通过截取一个网络数据流在某个时间窗口的数据包来提取特征数据提取并将它用于模型的输入。通过使用最新的数据集来验证后,LUCID 在低处理时间下能保证检测的精度。贾婧[10]设计的一个基于注意力机制的双向长短期记忆网络(LSTM)检测方法,实现了在检测 DDoS 攻击检测时,通过在双向 LSTM 神经网络结构中加入注意力机制,在关注相邻网络数据流的同时也关注了长距离网络数据流。通过使用 CAIDA-2007 数据集验证后,验证结果表明方法可在有限的资源下提取高价值特征信息,能够提高检测 DDoS 攻击的准确率。

统计网络报文的数据特征的传统方法[8]决定流量异常的阈值困难。与此同时,机器学习方法[6]需进行复杂的特征工程。通过在数据集上对检测模型训练与测试,有许多基于深度学习的方法可以准确地区分正常流量与异常流量。使用深度学习技术搭建检测模型是解决网络异常检测问题的一种高效方式,但许多现有的工作如 LVCID[9]和贾婧[10]都是基于现有公开的数据集搭建了表现优秀的检测模型。一个可以在实际生产环境中工作的网络异常流量检测系统应当在系统内

实现网络流量特征提取功能和检测功能。

本系统支持在 Linux 内核网络栈收集网络流量信息,如网络请求类型,IP 数据报首部,TCP 报文段首部和 UDP 用户数据报首部的信息。针对不同的网络协议,系统可分类收集其数据报头的信息和记录它们到达 Linux 内核网络栈的时间。然后,系统将使用 LSTM 神经网络构建的分类模型分析收集的数据,找出异常网络流量。本系统所做的主要工作如下所述:

第一,通过使用 eBPF 及其相关技术构建了一个在 Linux 内核网络栈的最底层收集网络流量数据的工具,这个工具可细粒度,低消耗地将网络流量数据从 Linux 内核中取出,并交由用户态下相关程序接口供其取用。

第二,为从内核中取出的数据构建了一个高效的检测模块,通过使用 LSTM 构建的深度学习检测模型,可以与基于 eBPF 的网络流量数据提取工具组合为一种新的网络异常流量检测系统。

2 系统设计与实现

本部分将讲述系统的设计与实现细节,主要包含数据采集与处理和数据分析两个部分。系统的运行逻辑和数据流向示意图如图 1 所示,虚线为数据流向,实线为系统运行逻辑。系统的运行分为三个阶段,第一阶段为训练分类模型,这个阶段的程序只会运行一次,负责将分类模型训练出来交给第三阶段使用。第二阶段为获取网络流量数据,系统在这个阶段将睡眠等待 10 秒以收集网络流量数据。第二阶段的程序将与第三阶段的程序在系统运行过程中周期性地交替运行。第三阶段为预测分类阶段,在这个阶段,系统将利用分类模型和获取到的数据进行预测分类工作,然后输出结果,再转到第二阶段运行。

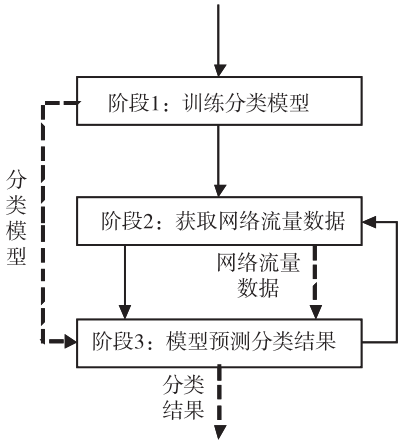

图 1 系统运行逻辑示意图

Fig. 1 Schematic diagram of system operation logic

2.1 数据采集与处理

可以利用 eBPF 这个工具做到详细地收集数据,但它的易用性方面却有很多的问题。原生的 eBPF 程序如果要执行,需要编写代码,加载进内核,从内核读取结果数据,卸载退出等一系列的动作,这些使得从内核空间取出数据变得繁琐。针对这个问题,可使用 BCC (BPF Compiler Collection)[11] 这个工具集,它是基于 eBPF 的 Linux 内核分析、跟踪、网络监控工具集。在 BCC 中可以使用 Python 作为入口进行编程,BCC 调用 LLVM Clang 编译器,这个编译器具有 eBPF 后端,可以将 C 代码转换成 eBPF 字节码,然后调用 bpf 系统调用,将 eBPF 字节码加载到内核中。使用这个工具集后,开发人员可以通过相关的帮助函数访问 eBPF 的存储数据结构 map,并将其映射为 Python 中的相关数据结构。BCC 的以上特点让开发人员利用 eBPF 的难度降低。系统使用 BCC 的另一个原因是现在常见的深度学习神经网络搭建工具都选择 Python 作为开发语言,其中也包括了本系统选择的 Tensorflow。

如图 2 所示,系统的数据收集工作由被挂载在 Linux 内核网络栈里的 eBPF 程序完成。然后数据交由 eBPF 的前端交互工具集 BCC 调用 RNN 检测模块中的 LSTM 检测模型完成数据分析与分类预测。

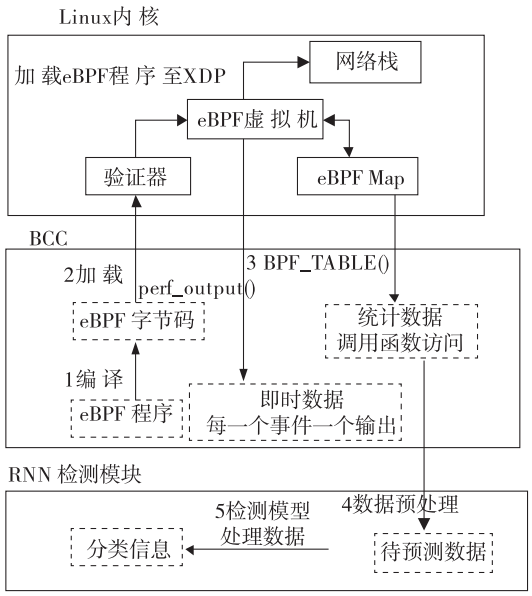

图 2 数据收集流程图

Fig. 2 The flowchart of data collection process

在系统中,使用了 XDP 技术来将 eBPF 程序挂载到 Linux 内核网络栈来收集数据,这种方式能让系统在勿需修改其他服务系统或 Linux 系统代码的情况下高细粒度地收集网络流量数据。

如图 3 所示,XDP 挂载点位于 Linux 内核网络栈的最底端,作为 eBPF 程序的一个可挂载点,它只负责过滤入口网络流量。就像 IEEE 的计算机网络分层模型一样,在 Linux 内核网络栈中也有相对应的数据结构用

于表示各种报文协议的头结构。如图 3 所示,它们分别是:使用 Struct ethhdr 结构体来表示以太网帧的头部,使用 Struct iphdr 结构体来表示 IP 协议的头部,使用 Struct tcphdr 结构体来表示 TCP 协议的头部,使用 Struct udphdr 结构体来表示 UDP 协议的头部。

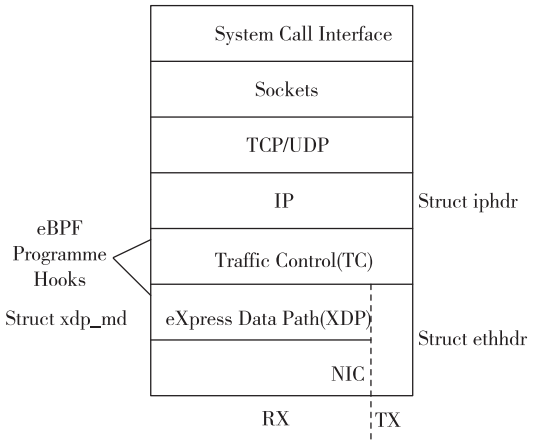

图 3　Linux 内核网络栈

Fig. 3　Linux kernel network stack

通过 XDP 技术运行的 eBPF 程序的运行环境是一个由内核传递的参数 Struct xdp_md。在 Struct xdp_md 的成员中,Data 和 Data_end 这两个变量分别包含了存储从设备驱动层传递上来的以太网帧的区域的头地址和尾地址。通过这两个变量可以获取在 Linux 内核网络栈中表示 IP 层网络报文的结构体和 TCP/UDP 层表示网络报文的结构体。虽然 Data 和 Data_end 中存储地址,但是它们的 C 语言类型是常规的 32 位无符号 Integer。为了取出 Data 和 Data_end 指向的存储区域中存储的数据,首先需要对它们进行类型转换为指针。

将 Data 和 Data_end 转换为指针后,就可以获得表示以太网帧的头部的结构体来对网络流量进行从以太网帧到 IP 数据包再到 TCP/UDP 数据包的层层拆包,层层获取数据的工作了。开发人员编写的程序被 BCC 挂载到了内核执行后,内核的 eBPF 验证器会十分严格地检查内存越界,为避免被报告错误,程序每拆一层都要进行越界检查。

常见的 DDoS 攻击方式如 SYN flood 和 ACK flood 利用了协议设计的漏洞,在某段时间内,向受害服务器发送与正常的网络流量的 TCP/UDP 和 IP 协议特征不一样的流量。所以,可以利用以上特点,从 Linux 内核网络栈的最底端获取网络流量,然后交由 LSTM 检测模型处理。如表 1 所示,根据对常见 DDoS 攻击的分析和参考 Sharafaldin[7] 的工作,系统提取了以下网络流量特征信息用于分析。

表 1　原始数据特征表

Table 1　Characteristics of original data

特征名	描　述	数据来源	类　型
saddr	源地址	iphdr	u32
daddr	目的地址	iphdr	u32
source	源主机端口	tcphdr/udphdr	u16
dest	目的主机端口	tcphdr/udphdr	u16
source_mac	源主机 MAC 地址	ethhdr	unsigned char *
syn_nums	统计 TCP 通信中 syn 标志出现的个数	tcphdr	u32
fin_nums	统计 TCP 通信中 fin 标志出现的个数	tcphdr	u32
total_length	统计 IP 地址所有数据包的长度	iphdr	u32
packets_length_max	统计某 IP 地址所有数据包中包长度的最大值	iphdr	u16
packets_length_min	统计某 IP 地址所有数据包中包长度的最大值	iphdr	u16
packets_nums	用于统计当前 IP 地址所有数据包的数量	iphdr	u32
first_t	记录连接的时间	BPF 帮助函数	u64
newest_t	记录最新一次收到数据包的时间	BPF 帮助函数	u64
last_t	记录上一次收到数据包的时间	BPF 帮助函数	u64
idle_t_min	记录最小的等待时间	BPF 帮助函数	u64
idle_t_max	记录最大等待时间	BPF 帮助函数	u64
macs	用于统计同一 MAC 地址下有多少 IP 连入,变量类型为 BPF_HASH 表,键为 MAC 地址,值为 IP 连接数目	ethhdr	BPF_HASH

2.2 数据分析

系统采用了基于 LSTM 搭建的神经网络模型作为数据分析预测的工具。如图 1 所示,系统由 eBPF 的相关部件完成网络流量数据的收集工作后,将数据交由检测模块完成数据分类预测工作。在数据交由分类器预测前,先对原始数据进行特征选择,其处理结果如表 2 所示。最后,为提升检测模块的处理速度,需要将特征选择后的数据做归一化处理。

表 2 正式数据特征表

Table 2 Characteristics of formal data

特征名	描 述
syn_nums	原始 syn_nums
fin_nums	原始 fin_nums
total_length	原始 total_length
packets_length_max	原始 packets_length_max
packets_length_min	原始 packets_length_ min
packets_nums	原始 packets_nums
average_length	数据包平均长度
dur	newest_t -last _t
idle_t_min	原始 idle_t_min
idle_t_max	原始 idle_t_max
ips_count_with _same_mac	与这个对象有相同的 mac 地址的数目由 source_mac 与 macs 查询获得
label	分类标签,在为构建预测模型收集数据时会被标注,在实际分类预测时这个特征将不存在。

为收集构建分类模型所需的仿真环境,本文构建了一个基于 VirtualBox 搭建的虚拟机的仿真实验环境。如图 4 所示,搭建了虚拟机作为实验机器,其中为本系统运行服务器安装 Ubuntu 系统,为另外的将用于向服务器发送异常流量和正常流量的客户端安装 Windows10 或 Ubuntu 系统。为构建 DDoS 的实验环境,本文选用 HOIC 和 SlowHTTPTest 作为 DDoS 攻击模拟软件作为构建攻击环境的工具。

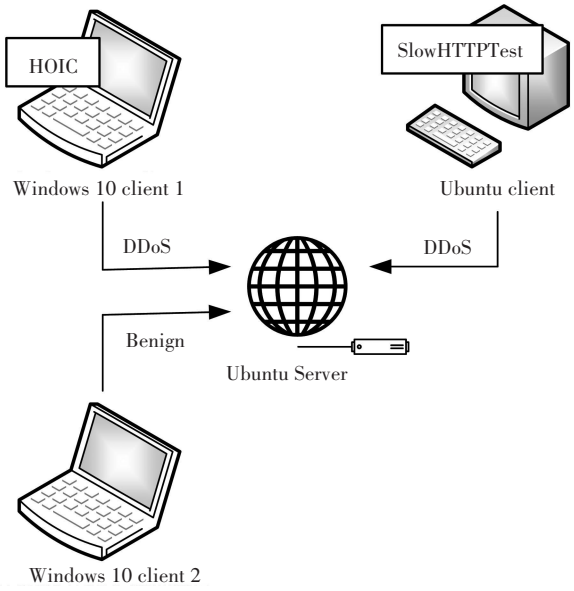

图 4 实验环境示意图

Fig. 4 Schematic diagram of experimental environment

HOIC 是一种可以使用多线程对目标服务器进行高速 HTTP 洪水攻击的 DDOS 攻击软件。常见的洪水攻击有 UDP Flood、TCP Flood 和 SYN Flood 。SlowHTTPTest 是一种可以对目标服务器进行慢速 HTTP 攻击的 DoS 攻击软件。SlowHTTPTest 攻击的原理是服务器在接收到完整的请求后才会进行处理,这时如果客户端的发送速度缓慢或者发送不完整,服务器会为其保留占用的连接资源,大量此类请求将导致因服务器的资源消耗殆尽引起的拒绝服务。SlowHTTPT est 可模拟常见的 slowloris DoS 攻击。

对于正常流量数据的收集,在收集异常流量的同时,服务器上模拟正常使用系统时的网络浏览,从而采集正常流量信息。采集到数据集后,就可开始构建用于检测分类的神经网络并调整其相关参数以求得最好的分类表现。

系统的异常检测使用基于循环神经网络 RNN 搭建的检测模型。为了获取能够搭建出表现最优的模型,选择了 RNN 的 3 种实现形式 SimpleRNN,长短记忆网络(LSTM)和门控循环单元(GRU)作为候选神经单元。通过在仿真环境中进行多次测试并调整各神经网络层以及各神经网络层的参数得到了 3 种实现形式的较优模型的精确率。其详细过程为:第一步,预先在仿真环境中采集建立初始模型的数据集。第二步,使用初始的数据集搭建出各类模型的一个初始可行的模型。第三步,将检测模型与 eBPF 数据收集程序整合为一个系统,放入仿真实验环境中。第四步,根据系统在

仿真环境中的表现,调整检测模型的各神经网络层以及各层的参数,同时将在调整各类检测模型过程中收集到的数据合并到训练模型的数据集中。

经过多轮对检测模型的调整,测得在仿真环境中 SimpleRNN 较优模型准确率为 0.92,LSTM 较优模型准确率为 0.98,GRU 较优模型准确率为 0.97。LSTM 的神经网络模型如图 5 所示,在模型中,为避免过拟合,设置了两个值为 0.2 的 Dropout 层;使用交叉熵作为损失函数,使用 Adam 作为优化算法。在系统的检测模型中,两个 LSTM 层的神经单元数量被分别置为 100 和 80,两个全连接层的神经元个数分别为 128 和目标类的数目。

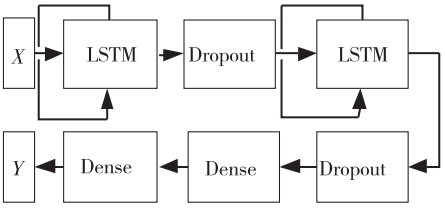

图 5　系统检测模型神经网络结构

Fig. 5　Neural network structure of system detection model

3　实验与结果分析

3.1　实验环境

本次的测试工作的环境为虚拟机仿真测试,本文选取了图 4 所示实验环境中的服务器和 Ubuntu 系统的客户端机器。如图 6 所示,测试时选取了一种名为 Qperf 的网络性能测试软件作为 Linux 网络状态的测试工具,将服务器作为 Qperf 的客户端,Ubuntu 客户端作为 Qperf 的服务器。本次测试是为了测试系统的数据收集的 XDP 程序启动后对 Linux 服务器网络及系统性能的影响,将 Snort 作为网络性能及系统性能消耗的对比。Snort 是一种能够抓包网络数据并根据自定义规则进行响应处理的入侵检测系统。

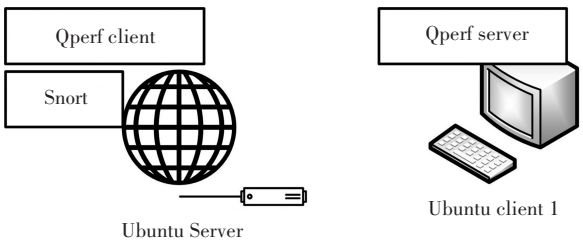

图 6　测试环境示意图

Fig. 6　Schematic diagram of test environment

3.2　网络性能消耗分析

网络带宽与网络延迟是网络性能的典型指标,实验以 Snort 嗅探器模式为对比,对 TCP 和 UDP 协议的这两个指标进行了多轮测试。对 TCP 协议通信下的测试中,每一轮测试都会按数据包大小从 1 bytes 到 64 kB,每次增长两倍,对其带宽和延迟进行多次测试,最后将每轮的结果取其平均值。对 UDP 协议通信下的测试中,每一轮测试都会按数据包从 1 bytes 到 32 kB,每次增长两倍,对其带宽和延迟进行多次测试,最后将每轮的结果取其平均值。

对于网络带宽而言,不论是 TCP 的带宽,还是 UDP 接收带宽和 UDP 发送带宽,XDP 数据收集程序对系统的网络带宽的影响是随着数据包大小的增大而变得明显。设启动系统的 XDP 数据收集程序后系统带宽为 x,不启动 XDP 数据收集程序时系统带宽为 y。如图 7 所示,其在启动系统的 XDP 数据收集程序后,与不启动 XDP 数据收集程序相比,根据网络协议种类将不同大小数据包的带宽增长率($[x-y]/y$)求得平均后发现,XDP 数据收集程序使得 TCP 协议的网络带宽(TCP)降低了 7.4%,使得 UDP 协议的接收带宽(UDP_recv)降低了 8.6%,使得 UDP 协议的发送带宽(UDP_send)降低了 9.6%。作为对比,Snort 嗅探器模式下对 TCP 协议和 UDP 协议的网络带宽影响程度比系统的程序小,由图 7 可知,在实验环境中,Snort 嗅探器模式对网络影响使得 TCP 协议的网络带宽(TCP_sn)降低了 2.9%,使得 UDP 协议的接收带宽(UDP_re_sn)降低了 2.9%,使得 UDP 协议的发送带宽(UDP_se_sn)降低了 4.5%。

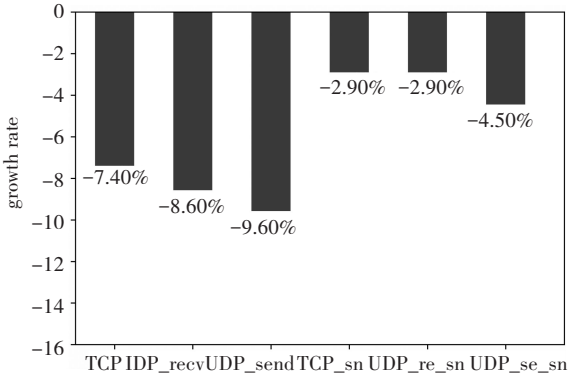

图 7　网络带宽影响对比示意图

Fig. 7　Comparison diagram of network bandwidth impact

对于网络延迟,系统和 Snort 在 TCP 与 UDP 两种通信协议下的延迟表现对比如图 8 所示。图 8 显示了两个系统在 TCP 协议通信下数据包大小从 1 bytes 到 64 kB 和 UDP 协议通信下数据包大小从 1 bytes 到 32 kB 的最小延迟,最大延迟和平均延迟。由图 8 可看出,系统的 TCP 最小延迟(TCP_min),TCP 平均延迟(TCP

_ave),UDP 最小延迟(UDP_min)和 UDP 平均延迟(UDP_ave)明显低于 Snort 的相应延迟。系统两种通信协议下的延迟平均比 Snort 低 17.8%。在最明显的 TCP 平均延迟(TCP_ave)处,系统的延迟比 Snort 的延迟低 23.7%。

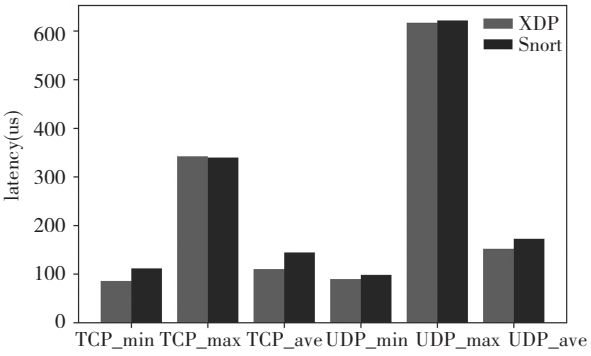

图 8 网络延迟影响对比示意图

Fig. 8 Comparison diagram of network delay impact

3.3 分类效果分析

在如图 4 所示系统的实验环境中,对系统的分类效果进行测试。在这个仿真实验环境中,使用 DDoS 攻击模拟软件 HOIC 和 SlowHTTPTest 产生网络异常流量数据。在上述的仿真实验环境中,通过 HOIC 和 SlowHTTPTest 对部署本系统的服务器进行多轮攻击,同时使用另外的机器正常访问服务器,然后记录系统对攻击访问和正常访问的检测分类结果。通过在测试环境中收集分类模型对两类流量多次的处理结果后,得到的数据经过统计得出系统的分类表现如表 3 所示。真阳性(True Positive,TP):预测正确样本的真实类别是异常网络流量,并且模型预测的结果也是异常网络流量。真阴性(True Negative,TN):样本的真实类别是正常网络流量,并且模型将其预测成为正常网络流量。假阳性(False Positive,FP):样本的真实类别是正常网络流量,但是模型将其预测成为异常网络流量。假阴性(False Negative,FN):样本的真实类别是异常网络流量,但是模型将其预测成为正常网络流量。经过计算后可得出系统的分类模型准确度(Accuracy),精确度(Precision),召回率(recall)和 $F1$ 值。

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

$$Precision=\frac{TP}{TP+FP} \tag{2}$$

$$Recall=\frac{TP}{TP+FN} \tag{3}$$

$$F1=\frac{2*precision*recall}{precision+recall} \tag{4}$$

在仿真系统中,经过人工检查 DDoS 攻击模拟软件 HOIC 产生的洪水攻击数据后发现其特征比较明显,所以系统的检测模型可以达到 100% 的准确率。在实际的生产环境中,攻击者的产生数据不会这么明显。但是系统在仿真实验环境中对 SlowHTTPTest 产生的 HTTP 慢攻击的表现依然可以说明本系统的可行性。受限于攻击模拟软件的局限性,在仿真实验环境中无法完全还原实际生产环境中攻击的复杂性,但是本实验说明了系统可以高度准确地分类攻击模拟软件模拟的攻击和正常请求。为提高系统的检测模型的可用性,还需要在更多种类网络攻击的仿真实验环境和实际生产环境不断优化。

表 3 系统分类效果示意表

Table 3 Schematic diagram of system classification effect

测试项目	对比项目			
	Accuracy	*Precision*	*Recall*	*F*1
HOIC	100%	100%	100%	100%
SlowHTTPTest	97%	97.9%	96%	96.9%

3.4 实验结果分析

通过使用系统的 eBPF 网络流量数据采集程序,系统可以对网络流量数据进行细粒度的采集。在实验环境中测试后,发现数据采集程序平均降低 8.53% 的系统网络带宽同时,系统网络延迟表现较好。系统在仿真环境中可以准确地分出异常网络流量和正常网络流量。实验说明了使用 eBPF 技术在 Linux 内核采集网络流量数据,使用 BCC 作为前端工具与深度学习技术的系统可以高效地执行异常流量检测的任务,使得深度学习对于网络攻击检测的研究不止于在已有数据集上进行,提供了一种可以在生产环境中实践的方法。

4 结束语

本文提出并实现了一种非入侵式的网络异常流量分析系统,这个系统通过 XDP 在 Linux 内核网络栈的最底端挂载 eBPF 程序的方式细粒度地收集以太网帧,IP 帧以及 TCP/UDP 报文头结构的信息。然后,使用深度学习方法构建检测模型对收集的数据进行分析,最后找出异常流量。运行本系统勿需修改内核或其他服务系统的程序即可实现对网络异常流量的分析。优秀的数据收集工具与高效率的分类检测模型使得系统在工作时以较低的网络资源消耗实现良好的分类效果。

参考文献(References):

[1] GREGG B. BPF Performance tools: Linux system and application observability[M]. New Jersey: Addison Wesley, 2019: 56—78.

[2] VIEIRA M, CASTANHO M, PACÍFICO R, et al. Fast packet processing with eBPF and XDP: concepts, code, challenges, and applications[J]. ACM Comput. Surv, 2020, 53(1): 1—36.

[3] SIERRA W A. Udploadbalancer[EB/OL]. https://github.com/AirVantage/sbulb, 2021-6-20.

[4] ALVENWONG. MicroBPF[EB/OL]. https://github.com/alvenwong/MicroBPF, 2021-6-20.

[5] SCHOLZ D, DANIEL R, EMMERICH P, et al. Performance implications of packet filtering with Linux eBPF[A]. International Teletraffic Congress[C]//New York: IEEE, 2018: 209—217.

[6] LIU C, CAI Z, WANG B, et al. A protocol-independent container network observability analysis system based on eBPF[A]. IEEE 26th International Conference on Parallel and Distributed Systems[C]//New York: IEEE, 2020: 697—702.

[7] SHARAFALDIN I, LASHKARI A, GHORBANI A. Toward generating a new intrusion detection dataset and intrusion traffic characterization[A]. 4th International Conference on Information Systems Security and Privacy[C]// Lda: SCITEPRESS, 2018: 108—116.

[8] DEMOULIN H, PEDISICH I, VASILAKIS N, et al. Detectin gasymmetric application-layer denial-of-service attacks in-flight with finelame[A]. USENIX Annual Technical Conference[C]// Washington: USENIX, 2019: 693—707.

[9] DORIGUZZI-CORIN R, MILLAR S, SCOTT-HAYWARD S, et al. LUCID: a practical, lightweight deep learning solution for DDoS attack detection[J]. IEEE Transactions on Network and Service Management, 2020, 17(2): 876—889.

[10] 贾婧, 王庆生, 陈永乐, 等. 基于注意力机制的攻 DDoS 击检测方法[J]. 计算机工程与设计, 2021, 42(9): 2440—2445.
JIA Jing, WANG Qing-sheng, CHEN Yong-Le, et al. Detection method of DDoS attack based on attention mechanism[J]. Computer Engineering and Design, 2021, 42(9): 2440—2445.

[11] HØILAND-JØRGENSEN T, BROUER J D, BORKMANN D, et al. The express data path: fast programmable packet processing in the operating system kernel[A]. International Conference on Emerging Networking Experiments and Technologies[C]// New York: ACM, 2018: 54—66.

[12] IO VISOR PROJECT. BPF compiler collection (BCC)[EB/OL]. https://github.com/iovisor/bcc, 2020-10-5.

[13] 高玉龙. DDOS 攻击及检测技术研究[D]. 扬州: 扬州大学, 2011.
GAO Yu-long. Research on DDoS attack and detection technology[D]. Yangzhou: Yangzhou University, 2011.

[14] DOULIGERIS C, MITROKOTSA A. DDOS attacks and defense mechanisms: classification and state-of-the-art[J]. Comput. Networks, 2004, 44(5): 643—666.

[15] 姜欧涅. 基于 eBPF 的网络数据包捕获与分析系统的设计与实现[D]. 武汉: 华中科技大学, 2020.
JIANG Ou-nie. Design and implementation of network packet capture and analysis system based on eBPF[D]. Wuhan: Huazhong University of Science and Technology, 2020.

[16] ENBERG P, RAO A, TARKOMA S. Partition-aware packet steering using XDP and eBPF for improving application-level parallelism[A]. the 1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms[C]//New York: ACM, 2019: 27—33.

[17] HUUB VAN WIEREN. Signature-based DDOS attack mitigation: automated generating rules for extended Berkeley packet filter and express data path[D]. Netherlands: UNIVERSITY OF TWENTE, 2019.

[18] HUA Y. An efficient traffic classification scheme using embedded feature selection and light GBM[A]. Information Communication Technologies Conference[C]//New York: IEEE, 2020: 125—130.

[19] BERTRONE M, MIANO S, RISSO F, et al. Accelerating Linux security with eBPF iptables[A]. the ACM SIGCOMM 2018 Conference on Posters and Demos[C]//New York: ACM, 2018: 108—110.

[20] TU NV, YOO J, HONG JW. Accelerating virtual network functions with fast-slow path architecture using express data path[J]. IEEE Transactions on Network and Service Management. 2020, 17(3): 1474—1486.

[21] BAIDYA S, CHEN Y, LEVORATO M. EBPF-based content and computation-aware communication for real-time edge computing[A]. IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops[C]//New York: IEEE, 2018: 865—870.

[22] GHIGOFF Y, SOPENA J, LAZRI K, et al. BMC: accelerating memcached using safe in-kernel caching and pre-stack processing[A]. Networked Systems Design and Implementation[C]//Washington: USENIX, 2021: 487—501.

责任编辑:陈　芳