doi:10.3772/j.issn.1002-0470.2024.08.002

基于数据包头序列的物联网恶意流量检测①

卫重波②[*,**] 谢高岗③[***] 刁祖龙[*,****] 张广兴[*]

([*]中国科学院计算技术研究所 北京 100190)

([**]中国科学院大学 北京 100190)

([***]中国科学院计算机网络信息中心 北京 100190)

([****]紫金山实验室 南京 211111)

摘 要 现有的基于机器学习(ML)的恶意流量检测方法,通常以高维的流量特征作为输入,并采用复杂模型,在实践中产生高误报率且资源占用较高。更重要的是,加密协议的广泛使用,使得数据包有效载荷特征很难被访问。幸运的是,物联网(IoT)设备的网络行为通常是有规律和周期性的,该特征反映在通信数据包序列上,每个数据包一定程度上描述了一次网络事件。基于此,本文提出了基于数据包头序列的恶意流量检测方法。它将流量序列转换为网络事件序列,并计算一组特征(即序列性、频率性、周期性和爆发性)来描述网络行为。实验环境包含一组真实的物联网设备,并将提出的方法部署在树莓派模拟的网关上。实验结果表明,与最新的检测方法相比,本文提出的方法能够在复杂网络环境下保持高准确性和低误报率,并提升了处理速率。

关键词 机器学习(ML);恶意流量检测;网络行为;物联网(IoT)安全;数据包头序列

近年来,物联网(Internet of Things, IoT)设备的应用越来越广泛。正如思科[1]预测的那样,到2023年底,物联网设备数量可能会超过290亿。智能家居等物联网设备在给人们的工作和生活带来便利的同时,也带来了风险。暴露在互联网上的设备由于自身的安全缺陷和用户的误用,容易受到攻击。被攻击设备可能进一步传播恶意流量,巨大的体量给网络安全带来威胁。因此,在物联网边缘阻断恶意流量的传播至关重要。

在本地网络中部署网络入侵检测系统(network intrusion detection system, NIDS)是一种常用的安全措施,它可以阻止恶意流量的早期传播。通常,NIDS部署在单个点上,它可以监视进入和离开本地网络的流量。实现这一目标的一种解决方案是将NIDS直接嵌入网关或路由器设备。

在过去的数十年,已经有许多NIDS被提出。一种流行的方法是使用人工神经网络(artificial neural network,ANN)来执行入侵检测。ANN学习复杂模式和行为的能力使其成为将网络流量分类为正常或某种类型攻击的合适解决方案[2-4]。然而,人工神经网络的计算复杂性随着神经元的数量呈指数级增长[5],而可能承载NIDS的网关和路由器设备限制了可用于处理流量数据的资源。此外,随着加密协议的广泛应用,一些基于流量数据包载荷的入侵检测系统逐渐不可用。这意味着深度包检测(deep packet inspection, DPI)方法不再适用,数据包头信息和时间相关信息成为主要可用的流量信息。

因此,基于轻量机器学习(machine learning, ML)的恶意流量检测方法被不断提出,它们以流级数据包头部的统计特征作为输入。然而,根据对几

① 国家重点研发计划(2022YFB3103000)和国家自然科学基金(62102397)资助项目。

② 男,1994年生,博士生;研究方向:计算机网络,物联网络安全;E-mail:weichongbo@ict.ac.cn。

③ 通信作者,E-mail:xie@cnic.cn。

(收稿日期:2023-04-06)

个真实数据集的测试结果,它们在复杂的网络环境下性能较弱且不稳定。具体地,当只有很少量的恶意流量混在高带宽流中,或是当网络抖动或产生大量背景流量时,检测性能急剧下降。

一方面,许多工作中提出的统计特征是冗余的、薄弱的。例如,一些 NIDS 分别计算每个数据包头字段的值,并进行计数、求和、方差等数学统计。然后用这些统计数据训练一些参数作为阈值,以确定接收的流量是否可疑。然而,它们没有考虑流的序列特征,这意味着输入数据对于表征网络行为和语义是很弱的。因此,攻击者很容易用许多单独的合法数据包来隐藏他们的活动。另一方面,计算统计特征的时间窗口通常较小,容易因网络噪声引起误报。而且,由于无法获得这些统计特征流量的详细语义信息,给网络管理员理解和处理这些告警带来了困难。

针对上述问题,本文提出了一种新的恶意流量检测方法,通过流量的数据包序列来表征网络行为。它根据数据包头字段为每个数据包分配一个整数编码(这里称为事件码),将数据包序列转换为事件序列。其思想是,数据包头本质上包括网络消息的语义。然后,从每个设备的事件序列中提取 4 个特征:序列性(sequence)、频率性(frequency)、爆发性(surge)和周期性(seasonality),并将其作为 ML 模型的输入。最重要的是,它将所有攻击事件序列收集为攻击指纹库,以准确检测隐藏攻击,减少误报。本文的主要贡献如下。

(1)提出一种根据数据包序列表征网络行为的新方法,该方法根据数据包头字段将数据包序列转换为事件序列。

(2)提出 4 个特征来描述事件序列。它们描述了物联网设备在很长一段时间内的网络行为,以及攻击事件的序列指纹特征。

(3)构建了一个由真实物联网设备组成的实验环境,并在树莓派上执行基准测试。结果表明,该方法在检测性能和运行时性能上均优于其他最新的方法。

本文共分为 4 节,第 1 节为相关工作,主要介绍现有的物联网异常流量检测方法;第 2 节详细描述了提出的方法;第 3 节介绍实验数据和环境配置,并给出实验结果;第 4 节对本文主要成果进行总结。

1 相关工作

本节首先介绍物联网恶意流量检测任务的网络拓扑环境,然后介绍并总结现有的针对物联网络的流量异常检测方案,最后讨论现有的物联网流量特征方案的不足。

1.1 物联网络攻击

智能家居是一个典型的小型物联网网络。有一些物联网设备,包括智能交换机、摄像头等,它们通过一个或多个无线接入点连接,并通过网关接入互联网。

不幸的是,由于用户的误用或安全漏洞,这种网络拓扑结构很容易受到各种攻击。如图 1 所示,①表示攻击者扫描暴露在互联网上的 IoT 设备,并试图控制它们;②表示一旦设备被破坏,它就会执行攻击者的攻击命令,如分布式拒绝服务攻击、垃圾邮件等。

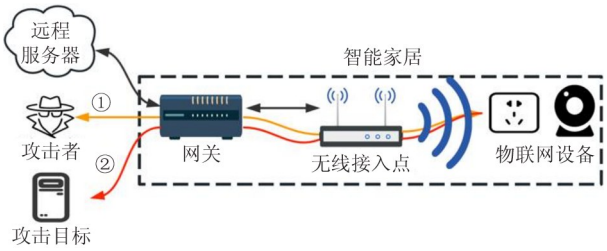

图 1 一个简单物联网络的攻击实例

1.2 针对物联网络的流量异常检测方案

基于网络流量的异常检测系统利用设备的通信流量信息来判断连接是否安全。一般情况下,如图 2所示,一个流量异常检测系统由以下几个部分组成。

(1)包捕获:捕获原始数据包。

(2)包解析:将原始数据包解析,获取元信息。

(3)特征提取:从实时流量中提取一组特征,该特征描述了特定设备近期的网络行为。

(4)异常检测器:它通常是一个多分类器,根据提取的特征集判断是否存在恶意流量。

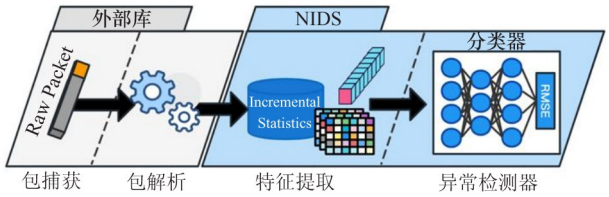

图 2　一个基于 ANN 的 NIDS 实例

有许多用于数据包捕获和解析器的外部库，例如 Tshark[6]。研究人员主要通过提出新的特征提取方案和异常检测器来提升检测性能。

一种流行的方法是将这样一个检测系统嵌入到网关设备中，它可以监视进出本地网络的流量[7-8]。然而，传统的基于神经网络的网络入侵检测系统资源消耗大，不适合部署在 Internet 网关和路由器上。

针对物联网网关，文献[9-11]提出了几种轻量级 ANN 模型或 ML 模型。自动编码器（AutoEncoder）是一种常用的在线异常检测算法。文献[6]提出了著名的 Kitsune，用于检测本地网络上的物联网攻击；它使用一组自动编码器来区分正常和异常流量的模式。文献[7]提出了一种称为 Bond 的简单的深度学习（deep learning，DL）模型，用于物联网网关上的僵尸网络检测。然而，根据在真实数据集上的测试，使用这些方法是不太实用的，原因包括以下 2 点。

（1）这些模型仅使用统计特征。统计特征基于对流数据的算术求和与平均，难以精确表征网络行为。因此，攻击者可以很容易地用许多单独的合法数据包隐藏他们的活动。

（2）它们的性能随着网络环境的变化而不稳定。轻微的网络波动可能会导致大量误报。

1.3　物联网流量特征

对于恶意流量检测系统，提取特征集以捕获每个数据包的上下文和目的信息是至关重要的。例如，一个 TCP SYN 数据包，可能是与服务器建立连接的良性尝试，也可能是试图引起分布式拒绝服务（distributed denial of service，DDoS）攻击而发送的数百万个类似数据包中的一个。

随着加密协议的广泛应用，数据包有效载荷逐渐变得不可访问。一些研究人员提出了基于每个数据包头字段的统计特征，例如数据包大小的计数、均值、标准偏差等[6,9]。一些知名的用于提取流量特征的外部库，例如 CICFlowMeter[12]，也只提取类似的统计特征，并不能精确表征网络行为和语义。

数据包序列是设备网络行为最直接的表现。如果将每个数据包视为一个攻击步骤，则恶意流中的数据包序列描述了攻击的过程。与传统计算机相比，物联网设备的网络行为通常具有规律性和周期性。因此，更好的方式是从设备的数据包序列中提取更准确的攻击指纹，该指纹为有效的攻击数据包序列。

2　基于数据包头序列的检测方法

本节提出了一种针对简单物联网络架构的流量异常检测方法。总体来说，该方法包括 5 个主要步骤：解析、映射、特征提取、训练 & 测试和更新。流量处理的流程如图 3 所示。第 1 步分析原始流量以获得包头字段，然后根据 IP 地址为每个 IoT 设备聚合这些数据包。第 2 步将每个数据包映射到一个事件代码。具体来说，根据包头字段分配事件码，将数据包序列转换为事件序列。第 3 步从事件序列的每个时间周期提取 4 类特征（序列性、爆发性、周期性和频率性）。特别地，序列性特征根据指纹库信息计算。第 4 步，每组特征实例被送入 ML 模型进行训练或测试。第 5 步，如果有新的攻击引起警报，指纹数据库将插入新的指纹信息。作为本文的主要贡献，本节将详细地描述映射、特征提取、训练和测试以及更新步骤是如何工作的。

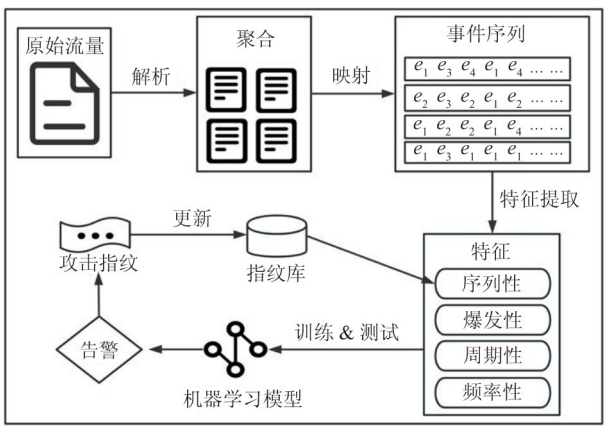

图 3　恶意流量检测系统示意图

2.1 问题描述

本文将整个时间区间切分成较小的时间片,每个时间片拥有固定的时间长度 ξ(比如 10 min),并以时间片的起始时刻来代表该时间片。因此,实时的检测任务转化为判断每个时间片的流量是否异常。具体地,对于开始于时间点 τ_h 的恶意流量,检测系统的目标是:根据时间长度为 $\Delta\tau_m$ 的区间 $[\tau_s, \tau_d]$ 的数据包信息,在时间片 τ_x 发出告警。恶意流量检测模型如图 4 所示。

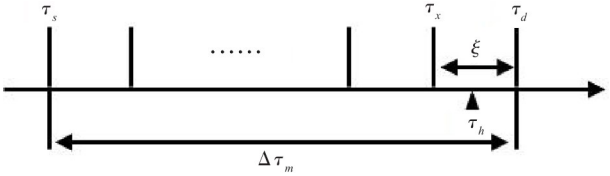

图 4 恶意流量检测模型

2.2 映 射

该步骤的目的是,对于每个数据包,根据其头部信息分配一个事件码 $e(e \in Z)$,将数据包序列转换为事件码序列。事件码代表了数据包级的网络事件,也可以将它视为数据包类型,通过对 IPv4 头部和传输控制协方式(transmission control protocol, TCP)头部中一些字段的哈希来实现分配,并结合了数据包的传输方向(上/下行)。该想法源于数据包的头部信息潜在地包含了流量的语义。一个事件码由 3 部分共 8 位(bit)组成,如图 5 所示。

(1)D:1 bit,0/1 代表上行/下行数据包。

(2)C _ IP:4 bit,由 IPv4 头部长度(Internet header length)、生存时间(time to live)和协议(protocol)字段值哈希计算。

(3)C _ TCP:3 bit,由 TCP 头部控制位(control flags,包括 URG、ACK、PSH、RST、SYN 和 FIN)值哈希计算。

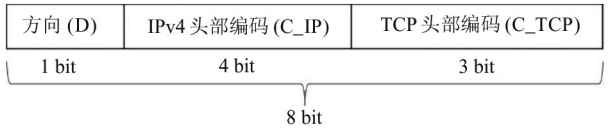

图 5 事件编码组成

有很多成熟的哈希算法可用,本工作使用快速且稳定的红黑树结构实现,通过动增量分配值(value)的方式可以避免哈希冲突。虽然数据包头部字段的值域分布可能很广,但是经过对真实数据的分析,正常情况下,实际出现的字段值是非常有限的(最多只有 79 种),8 bit 的值域足以为每种头部组合分配编码。实际上,编码 e 可以扩展到更多位,但是它会带来后续额外的计算开销。

注意到,哈希方式主要针对 TCP 数据包,原因包括:(1)TCP 报文头部包含更多有意义的信息;(2)非 TCP 连接的攻击比较容易通过数学统计方式检测。非 TCP 数据包对应哈希值被置 0。

2.3 特征提取

将流量转换为事件编码序列之后,系统将捕获每个事件序列的特征,以便最好地区分良性流量和恶意流量。本工作确定了以下 4 个特征。

(1)频率性。频率性是大多数机器学习应用中的一个基本特征。比如,正常情况下,一个设备其他网络设备的尝试连接行为很少发生。

(2)周期性。通常周期性或准周期性(quasi-periodically)出现的特定类型数据包是良性的。比如用户的日常登录请求。

(3)爆发性。爆发性是异常检测系统一个基本特性,因为特定类型的数据包(事件码)偶尔出现并不表示异常,但突然的爆发式增长表示异常。

(4)序列性。它是本工作中提出的重要特征,实验发现在同种攻击下的数据包遵循相似的序列模式。

如上所述,这 4 个特性对于流量异常检测都是必要的。根据经验,它们的组合足以捕获所有流量的关键特征。下面将介绍如何提取上述 4 个特征。

(1)频率性提取:对于每个时间片,使用一个向量记录该时间片内每种事件的出现频次。本工作中,一个特定的数据包被标记为一个特定的事件编码 $e \in [0,255]$。形式化地,对于时间片 τ_k,$\boldsymbol{P}_k = (p_1^k, p_2^k, \cdots, p_n^k)$ 是其时间片内的数据包序列,数据包 p_i^k 被映射为事件编码 $e_{p_i^k}$,然后得到其事件序列 $\boldsymbol{e}_k = (e_{p_1^k}, e_{p_2^k}, \cdots, e_{p_n^k})$。对于每个事件编码 $e_j (j \in \{1,2,\cdots,N\}$,$N$ 为事件编码的类型总数,这里为 256),本文标记 $C_\Delta^k(e_j)$ 为其在时间片 τ_k 内的出现频次。这样,对于每个事件 e_j,得到它在时间区间

$[\tau_s,\tau_d]$ 内连续时间片 $(\tau_1,\tau_2,\cdots,\tau_M)$ 上的频率性特征向量 $\boldsymbol{C}_\Delta(e_j)=(C_\Delta^1(e_j),C_\Delta^2(e_j),\cdots,C_\Delta^M(e_j))$，其中，$\tau_d=\tau_k+\xi$ 表示异常时间片 τ_k 的结束时间，$\tau_d-\tau_s=M\cdot\xi$ 表示时间 M 个连续的时间片。

(2)周期性特征提取：如前所述，某些类型的数据包是准周期性出现的。即同类型数据包(事件码)经常在每分钟/小时/天相似时间点出现。让 H_j 表示事件 e_j 的观测时间跨度，可以得到观测数列 $\boldsymbol{C}_\xi(e_j)=(C_\xi^1(e_j),C_\xi^2(e_j),\cdots,C_\xi^h(e_j))$，其中，$C_\xi^k(e_j)$ 是 e_j 在第 k 个时间片中出现的次数，h 是 H_j 中时间片的个数。显然，如果 e_j 是周期性的，$\boldsymbol{C}_\xi(e_j)$ 是一个周期性的时间序列。因此，需要进一步判断 $\boldsymbol{C}_\xi(e_j)$ 是否是周期性的。

已经有一系列的研究关于判断一个时间序列是否具有周期性：基于时域的自相关(autocorrelation)方法和基于频域的离散傅里叶变换(discrete Fourier transform，DFT)方法[13]。基于复杂度的考量，本工作采用简单有效的基于自相关的方法 YIN[14]，它整合了大量的改进方法，并提高了准确性。在 YIN 中，计算微分函数(difference function)：

$$D'(\rho,e_j)=\frac{D(\rho,e_j)}{\frac{1}{\rho}\sum_{k=1}^{\rho}D(k,e_j)} \tag{1}$$

$$D(\rho,e_j)=\sum_{k=1}^{h-\rho}(C_\xi^k(e_j)-C_\xi^{k+\rho}(e_j))^2 \tag{2}$$

其中，ρ 是根据领域知识确定的以分、时为单位的时间周期。更小的 $D'(\rho,e_j)$ 意味着更强的周期性，因此对于每个事件编码 $e_j(j\in 1,2,\cdots,N)$，计算 $\alpha_j=\min_\rho D'(\rho,e_j)$ 作为最终的周期性值，其在正常流量的训练数据集上是唯一的。

(3)爆发性特征提取：尽管当某些事件偶尔出现时不代表异常，但它们突然激增时很可能表示异常。比如一个尝试重连的 Telnet 会话可能是用户的操作失误，但高频的发生很可能是遭受暴力破解攻击。

对于每个事件 e_j，系统的目标是发现在当前时间区间 $[\tau_k-\Delta\tau_m,\tau_k]$ 内，是否存在 e_j 呈爆发性出现的时间片。通过统计 e_j 在每个时间片内的出现次数，得到时间序列 $\boldsymbol{C}_\xi(e_j)=(C_\xi^1(e_j),C_\xi^2(e_j),\cdots,C_\xi^h(e_j))$，其中，$C_\xi^k(e_j)$ 是 e_j 在第 k 个时间片中出现的次数。为了检测 $\boldsymbol{C}_\xi(e_j)$ 是否存在“阶跃”式变化，本工作使用一种检测时间序列行为变化的方法[15]。它提高了奇异谱变换的计算效率，且在真实的实验中得到了验证和评价。它计算每个事件 e_j 的时间序列 $\boldsymbol{C}_\xi(e_j)$ 的变化分数(change score) cs_j^k，当 $\boldsymbol{C}_\xi(e_j)$ 出现阶跃变化时，cs_j^k 会获得更大的值。最终，获得 e_j 在时间片 $(\tau_1,\tau_2,\cdots,\tau_M)$ 上爆发性特征向量 $\boldsymbol{cs}_j=(cs_j^1,cs_j^2,\cdots,cs_j^M)$。

(4)指纹 & 序列性特征提取：表 1 列出了某一设备在其遭受不同攻击(对应时间片 τ_1、τ_2、τ_3)时的事件码序列。可以发现，这些序列共享一个公共子序列(表 1 中的粗体符号)。因此，本文定义攻击序列指纹为特定攻击类型在不同次攻击下的事件序列的公共子序列。这些指纹可以显著帮助异常检测和攻击分类。

表 1 特定攻击下不同时间片的事件序列

时间片	事件序列
τ_1	e_1 $\boldsymbol{e_{10}}$ $\boldsymbol{e_{22}}$ e_7 e_3 $\boldsymbol{e_{18}}$ $\boldsymbol{e_{19}}$ e_2 e_7 $\boldsymbol{e_{15}}$ e_2 e_1
τ_2	$\boldsymbol{e_{10}}$ e_6 e_4 $\boldsymbol{e_{22}}$ $\boldsymbol{e_{18}}$ e_{17} e_6 e_1 e_2 $\boldsymbol{e_{19}}$ $\boldsymbol{e_{15}}$ e_2
τ_3	e_4 e_1 $\boldsymbol{e_{10}}$ e_7 $\boldsymbol{e_{22}}$ $\boldsymbol{e_{18}}$ e_{19} e_6 e_6 $\boldsymbol{e_{15}}$ e_{17} e_1

攻击序列指纹的好处是它可以准确地发现隐藏在非关键数据包(噪声流量)中的攻击数据。为了评估一个事件序列是否包含攻击序列，系统将计算其序列性特征，以衡量它与各个已知攻击序列指纹的相似性得分。

具体地，对于给定的时间片 τ_k，$\boldsymbol{P}_k=(p_1^k,p_2^k,\cdots,p_n^k)$ 是其时间片内的数据包序列，$\boldsymbol{e}_k=(e_{p_1^k},e_{p_2^k},\cdots,e_{p_n^k})$ 是其对应的事件序列。对于指纹库的任一指纹 $f^i,i\in\{1,2,\cdots,L\}$，L 为当前指纹库规模，系统计算 f^i 与 $\boldsymbol{e}_k$ 的最长公共子序列(longest common subsequence，LCS) $LCS_{f_i^k}$，然后 $\boldsymbol{e}_k$ 的序列性特征分数 $S_{\max}^k$ 为

$$S_{\max}^k=\max_i\frac{LCS_{f_i^k}}{len(f^i)} \tag{3}$$

其中，$len(f^i)$ 表示 f^i 的长度。最终，获得时间片 $(\tau_1,\tau_2,\cdots,\tau_M)$ 上的序列特征向量 $\boldsymbol{S}_{\max}=(S_{\max}^1,S_{\max}^2,\cdots,S_{\max}^M)$。

(5)指纹库更新:指纹库收集所有攻击事件序列指纹。最初,训练集的一部分攻击事件序列按照算法 1 逐个地添加至指纹库 Ω。算法中,ϕ 为人工设置的阈值,合理的阈值可以精简指纹库,同时适应攻击事件序列的变化。测试阶段,被检测器发现的攻击事件序列也会被使用同样的算法添加至 Ω。

算法 1 更新指纹库 Ω

输入:事件序列 $\boldsymbol{e}_k$, 指纹库 Ω;

输出:更新后的指纹库 Ω_{new}。

UPDATE($\boldsymbol{e}_k$,Ω)

BEGIN

1. FOR Ω 中每个指纹 f
2. 计算 $s = \dfrac{LCS(f,\boldsymbol{e}_k)}{len(f)}$
3. IF $s \geqslant \phi$ THEN
4. 更新 $f = LCS(f,\boldsymbol{e}_k)$
5. IF 没有 f 被更新 THEN
6. 将 $\boldsymbol{e}_k$ 插入 Ω
7. 返回更新后的指纹库 Ω_{new}

END

(6)特征聚合:在提取了上述 4 类特征后,在时间片 $(\tau_1,\tau_2,\cdots,\tau_M)$, 对于事件 e_j, 得到:频率性特征 $\boldsymbol{C}_\Delta(e_j) = (C^1_\Delta(e_j),C^2_\Delta(e_j),\cdots,C^M_\Delta(e_j))$, 爆发性特征 $\boldsymbol{cs}_j = (cs^1_j,cs^2_j,\cdots,cs^M_j)$, 和周期性特征值 α_j。对于所有事件,得到序列性特征 $\boldsymbol{S}_{\max} = (S^1_{\max},S^2_{\max},\cdots,S^M_{\max})$。将 α_j 作为权重赋予 $\boldsymbol{C}_\Delta(e_j)$ 和 $\boldsymbol{cs}_j$ 得到 $C'_\Delta(e_j) = C_\Delta(e_j) \times \alpha_j$, $\boldsymbol{cs}'_j = \boldsymbol{cs}_j \times \alpha_j$。最后,得到了时间片 $(\tau_1,\tau_2,\cdots,\tau_M)$ 上的特征集的整合矩阵 $\boldsymbol{A}$:

$$\boldsymbol{A} = (\boldsymbol{S}_{\max},\boldsymbol{C}'_\Delta(e_1),\cdots,\boldsymbol{C}'_\Delta(e_N),\boldsymbol{cs}'_1,\cdots,\boldsymbol{cs}'_N)^{\mathrm{T}} = (\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_M) \tag{4}$$

(7)训练与测试:多种轻量级的 ML 模型,包括 K 近邻(k-nearest neighbor, KNN)、决策树(decision tree, DT)和随机森林(random forest, RF)作为异常检测器。训练阶段,学习它们的参数,采集训练数据中攻击事件指纹到指纹库 Ω 中。测试阶段,运行模型,并使用算法 1 更新指纹库。

3 实验评估

本节评估提出的方法的检测性能(detection performance)和运行时性能(runtime performance),并与最先进的物联网流量异常检测方法 Kitsune[6] 和 LimNet[8] 进行对比。

3.1 数据集

实验环境为一个真实的物联网网络(如图 6 所示),将提出的恶意流量检测系统部署在树莓派(Raspberry Pi)4 B 上以进行基准测试。Pi 被连接到交换机,以获取实时流量。

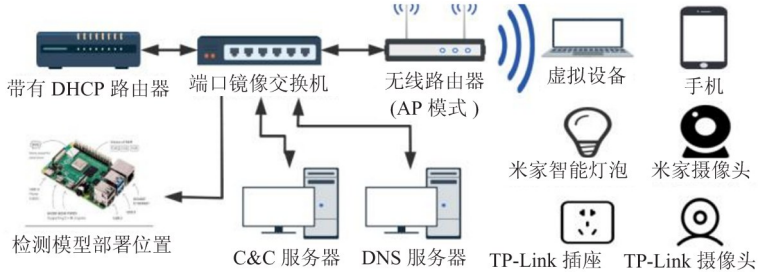

图 6 实验网络拓扑

C&C 服务器和 DNS 服务器配合模拟外部网络攻击。其上部署了 2 种开源的物联网僵尸病毒 BASHLITE[16] 和 Mirai[17],并对 3 个月内产生的攻击流量进行了标记。此外还使用了公开的物联网数据集 MedBIoT[18] 和 IoT-23[19]。它们是从一个中型网络收集的,包含多种真实物联网设备和数十种虚拟设备。实验中随机选择若干时间片上共计 5 000 万个数据包,按 1∶1 进行训练与测试。

3.2 实验配置

在 Raspberry Pi 4 B 上进行基准测试,使用 Python 3.7.9 和 Pytorch 1.12.0 编码,并在单核上执行。设置事件片长度 ξ =5 min, $\Delta\tau_m$ =1 h。

3.3 检测性能

使用精确性(Precision)、召回率(Recall)、综合

评价指标 F1 分数(F1 Score)和误报率(false positive ratio,FPR),将正确的恶意流量告警标记为真阳(true positive,TP),它们按以下方式计算:

$$Precision = \frac{TP}{TP + FP},\ Recall = \frac{TP}{TP + FN},$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall},\ FPR = \frac{FP}{FP + TN}$$

表 2 比较了提出的方法在使用不同 ML 模型分类器时的检测性能,以及与已有的最先进的方法 Kitsune 和 LiMNet 的对比。Kitsune 使用了一组自动编码器模型集合,LiMNet 使用带有新单元的循环神经网络(recurrent neural network,RNN)。用相关工作中提到的最佳设置对它们进行测试。实验结果表明,提出的方法结合 DT 模型在几乎所有指标上都取得了最佳性能。

表 2 原始数据集上的检测性能

方法	Precision	Recall	FPR	F1 Score
Kitsune	0.91	0.96	0.10	0.94
LiMNet	0.95	0.98	0.06	0.97
KNN	0.90	0.91	0.10	0.91
DT	**0.97**	**0.98**	**0.02**	**0.97**
RF	0.93	0.95	0.07	0.94

为了进一步测试这些方法应对复杂网络环境的鲁棒性,评估数据中加入“噪声”:(1)收集不稳定网络环境下的流量,例如发生网络抖动和数据包重传(假设略微的网络抖动是正常的);(2)在攻击流量中人工嵌入正常流量,增加攻击的隐匿性,即将攻击流量伪装成正常流量。最后,将 20% 的原始数据随机替换为噪声数据。

表 3 展示了这些方法在应对复杂网络状况及隐藏攻击时的性能。实验结果表明,提出的方法依然能取得优异的性能,特别是误报率与其他方法相比有明显的降低。综合来看,基于 DT 模型的方法对恶意流量的检测效果优于其他算法,特别是复杂网络环境下。

表 3 “噪声”数据集上的检测性能

方法	Precision	Recall	FPR	F1 Score
Kitsune	0.73	0.94	0.33	0.83
LiMNet	0.82	0.93	0.20	0.87
KNN	0.93	0.95	0.07	0.94
DT	**0.97**	**0.95**	**0.02**	**0.96**
RF	0.94	0.93	0.05	0.93

3.4 运行时性能

为了测试运行时性能,在 Raspberry Pi 4 B 模拟网关上执行了基准测试(详细信息见表 4)。

表 4 基准测试配置

	Raspberry Pi 4 B
Type	Broadcom BCM2711
CPU Clock	1.5 GHz
Cores	4
RAM	2 GB

表 5 展示了提出的方法和其他方法的运行时效率对比。新提出的方法使用决策树作为分类器时获得最快的数据包平均处理速率,是 Kitsune 的 1.5 倍,LiMNet 的 6.0 倍。

表 5 数据包处理速率对比

方法	速率/(数据包/s)
Kitsune	4 000
LiMNet	1 000
KNN	3 500
DT	**6 000**
RF	4 800

事实上,不同于 Kitsune 和 LiMNet,本文提出的方法将时间片内的流量一起处理。在数据包到达时,只需对数据包首部进行简单的解析和哈希操作,最终每个时间片生成一个特征矩阵 $\boldsymbol{A}$。整个流程的时间消耗基本与特征矩阵数量呈正比,所以更大的 ξ 会获得更快的数据包平均处理速率。

4 结 论

本文提出了基于数据包序列的物联网恶意流量检测方法。它构造物联网设备的事件序列,从中提取攻击指纹和 4 类特征,结合轻量的 ML 模型,在真实数据集上,获得了比最新方法更优的检测性能和

运行时性能,证明了本文方法的实用性。

根据实验结论分析,物联网设备流量具有明显的时间相关的序列特征,同时数据包头字段信息可以反映其网络行为语义。二者结合,可以从流量(数据包序列)中学习物联网设备的网络行为模型,不同于其他的基于统计的方法,序列"指纹"提供了更精确的网络攻击痕迹,大幅减少了实际应用中的高误报的缺陷。

未来的工作将集中在提出针对物联网设备网络事件序列的更优特征集和更轻量级的分类器上。

参考文献

[1] CISCO U. Cisco annual Internet report (2018-2023) white paper[R]. San Jose: Cisco, 2020:1-35.

[2] 孟献轲, 张硕, 熊诗, 等. 基于时空注意力特征的异常流量检测方法[J]. 计算机应用与软件, 2023,40(4):99-106.

[3] 宣萍, 房朝辉, 丁宏. 基于自注意力机制的网络流量异常检测方法[J]. 安徽大学学报(自然科学版), 2023,47(1):24-28.

[4] 刘祥军, 江凌云. 基于集成学习的物联网设备异常流量检测算法[J]. 计算机应用研究, 2022,39(6):1785-1789.

[5] SRIVASTAVA R K, GREFF K, SCHMIDHUBER J. Training very deep networks[J]. Advances in Neural Information Processing Systems, 2015,28:2377-2385.

[6] MERINO B. Instant traffic analysis with Tshark how-to [M]. Birmingham: Packt Publishing Ltd, 2013:1-68.

[7] LIN W H, LIN H C, WANG P, et al. Using convolutional neural networks to network intrusion detection for cyber threats[C]//2018 IEEE International Conference on Applied System Invention. Medan, Indonesia: IEEE, 2018:1107-1110.

[8] LIU J, AHMED E, SHIRAZ M, et al. Application partitioning algorithms in mobile cloud computing: taxonomy, review and future directions[J]. Journal of Network and Computer Applications, 2015,48:99-117.

[9] MIRSKY Y, DOITSHMAN T, ELOVICI Y, et al. Kitsune: an ensemble of autoencoders for online network intrusion detection [EB/OL]. (2018-05-27) [2023-04-04]. https://arxiv.org/pdf/1802.09089.

[10] GANDHI H, MEHRA M, RIBEIRO V. Bond: efficient and frugal DL model co-design for botnet detection on IoT gateways[C]//The 1st International Conference on AI-ML-Systems. Bangalore, India: ACM, 2021:1-7.

[11] GIARETTA L, LEKSSAYS A, CARMINATI B, et al. Limnet: early-stage detection of IoT botnets with lightweight memory networks[C]//Computer Security-ESORICS 2021: 26th European Symposium on Research in Computer Security. Darmstadt, Germany: Springer, 2021:605-625.

[12] LASHKARI A H, ZANG Y, OWHUO G, et al. Cicflowmeter[EB/OL]. (2021-08-10)[2023-04-04]. https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt.

[13] PARTHASARATHY S, MEHTA S, SRINIVASAN S. Robust periodicity detection algorithms[C]//Proceedings of the 15th ACM International Conference on Information and Knowledge Management. Kansas, USA: ACM, 2006:874-875.

[14] DE CHEVEIGNE A, YIN H K. A fundamental frequency estimator for speech and music[J]. The Journal of the Acoustical Society of America, 2002, 111(4): 1917-1930.

[15] ZHANG S, LIU Y, PEI D, et al. Funnel: assessing software changes in web-based services[J]. IEEE Transactions on Services Computing, 2016,11(1):34-48.

[16] ANGRISHI K. Turning Internet of Things (IoT) into Internet of vulnerabilities (IoV): IoT botnets[EB/OL]. (2017-02-13)[2023-04-04]. https://arxiv.org/pdf/1702.03681.

[17] ANTONAKAKIS M, APRIL T, BAILEY M, et al. Understanding the Mirai botnet[C]//The 26th USENIX Security Symposium. Vancouver, Canada: USENIX Association, 2017:1093-1110.

[18] GUERRA-MANZANARES A, MEDINA-GALINDO J, BAHSI H, et al. MedBIoT: generation of an IoT botnet dataset in a medium-sized IoT network[C]//Proceedings of the 2020 International Conference on Information Systems Security and Privacy. Valletta, Malta: Springer, 2020:207-218.

[19] PARMISANO A, GARCIA S, ERQUIAGA M J. A labeled dataset with malicious and benign IoT network traffic[EB/OL]. (2020-01-20)[2023-04-04]. https://www.stratosphereips.org/datasets-iot23.

IoT traffic anomaly detection based on header sequence

WEI Chongbo[*,**], XIE Gaogang[***], DIAO Zulong[*,****], ZHANG Guangxing[*]

([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100190)

([***] Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190)

([****] Purple Mountain Laboratories, Nanjing 211111)

Abstract

Existing malicious traffic detection methods based on machine learning (ML) usually take high-dimensional traffic features as input and use complex models. In practice, it generates high false alarm rates and has high resource consumption. More importantly, the widespread use of encryption protocols makes packet payload features difficult to access. Fortunately, the network behavior of Internet of Things (IoT) devices is usually regular and periodic, and the feature is reflected in the sequence of communication packets, each of which describes a network event to some extent. Based on this, this paper proposes a malicious traffic detection method based on packet header sequences. It converts traffic sequences into network event sequences and computes a set of features (namely sequence, frequency, surge, and seasonality) to describe the network behavior. The experimental environment contains a set of real IoT devices, and the proposed method is deployed on a Raspberry Pi simulated gateway. The experimental results show that the proposed method is able to maintain high accuracy and low false alarm rate in complex network environments and improve the processing rate compared to the state-of-the-art detection methods.

Key words: machine learning (ML), traffic anomaly detection, network behavior, Internet of Things (IoT) security, packet header sequence