计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2024,33(12):55−66 [doi: 10.15888/j.cnki.csa.009691] [CSTR: 32024.14.csa.009691]
©中国科学院软件研究所版权所有.
E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 PFEC-Transformer 的 DNS 隐蔽隧道检测①

江　魁[1], 黄锐滨[2], 邓昭蕊[2], 伍　波[1], 朱思霖[2]

[1](深圳大学 信息中心, 深圳 518060)
[2](深圳大学 电子与信息工程学院, 深圳 518060)
通信作者: 江　魁, E-mail: jiangkui@szu.edu.cn

摘　要: DNS 作为互联网基础设施, 很少受到防火墙的深度监控, 导致黑客和 APT 组织通过 DNS 隐蔽隧道来窃取数据或控制网络, 对网络安全造成严重威胁. 针对现有检测方案容易被攻击者绕过以及泛化能力较弱的问题, 本研究改进了 DNS 流量的表征方法, 并提出了 PFEC-Transformer (pcap features extraction CNN-Transformer) 模型. 该模型以表征后的十进制数值序列作为输入, 在经过 CNN 模块进行局部特征提取后, 再通过 Transformer 分析局部特征间的长距离依赖模式并进行分类. 研究采集了互联网流量以及各类 DNS 隐蔽隧道工具生成的数据包构建数据集, 并使用包含未知隧道工具流量的公开数据集进行泛化能力测试. 实验结果表明, 该模型在测试数据集上取得了高达 99.97% 的准确率, 在泛化测试集上也达到了 92.12% 的准确率, 有效地证明了其在检测未知 DNS 隐蔽隧道方面的优异性能.

关键词: 网络安全; DNS 隐蔽隧道; 异常流量检测; 深度学习; 泛化能力

引用格式: 江魁,黄锐滨,邓昭蕊,伍波,朱思霖.基于 PFEC-Transformer 的 DNS 隐蔽隧道检测.计算机系统应用,2024,33(12):55–66. http://www.c-s-a.org.cn/1003-3254/9691.html

DNS Covert Tunnel Detection Based on PFEC-Transformer

JIANG Kui[1], HUANG Rui-Bin[2], DENG Zhao-Rui[2], WU Bo[1], ZHU Si-Lin[2]

[1](Information Center, Shenzhen University, Shenzhen 518060, China)
[2](College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China)

Abstract: As an Internet infrastructure, DNS is rarely subjected to deep monitoring by firewalls, allowing hackers and Asia-Pacific Telecommunity (APT) organizations to exploit DNS covert tunnels for data theft or network control and posing a significant threat to network security. In response to the easily bypassed nature of existing detection methods and their weak generalization capabilities, this study enhances the characterization method of DNS traffic and introduces the pcap features extraction CNN-Transformer (PFEC-Transformer) model. This model uses characterized decimal numerical sequences as input, conducts local feature extraction through CNN modules, and then analyzes long-distance dependency patterns between local features by using the Transformer for classification. The research builds datasets by collecting internet traffic and data packets generated by various DNS covert tunnel tools and conducts generalization testing with publicly available datasets containing traffic from unknown tunneling tools. Experimental results demonstrate that the model achieves an accuracy of 99.97% on the testing dataset and 92.12% on the generalization testing dataset, effectively showcasing its exceptional performance in detecting unknown DNS covert tunnels.

Key words: network security; DNS covert tunnel; anomaly traffic detection; deep learning; generalizability

① 基金项目: 教育部科技发展中心-中国高校产学研创新基金新一代信息技术创新项目 (2021ITA01009)
收稿时间: 2024-05-17; 修改时间: 2024-06-12; 采用时间: 2024-06-26; csa 在线出版时间: 2024-10-25
CNKI 网络首发时间: 2024-10-25

© 中国科学院软件研究所 http://www.c-s-a.org.cn

DNS (domain name system) 是互联网的关键服务之一, 它将易于理解的域名转换为计算机可识别的 IP 地址[1], 从而省去了记忆复杂 IP 地址的需求. 为避免干扰正常的 DNS 查询和响应, 大部分防火墙通常不深入检测 DNS 流量. 而 DNS 隐蔽隧道技术正是利用了这一特点, 利用 DNS 协议通信流量来隐藏实际传输的数据信息, 可用于实现敏感信息窃取和内网僵尸网络的命令与控制 (command and control, C&C) 通信. 自 1998 年首次发现攻击者利用 DNS 隐蔽隧道技术绕过付费 WiFi 门户验证以来[2], 这一技术已经成为高级持续性威胁 (APT) 的常用手段. 例如, 在 2020 年的 SolarWinds 供应链攻击中, 恶意软件 SUNBURST 就通过 DNS 隐蔽隧道传输 C&C 指令, 影响了包括美国政府部门在内的大约 1.6 万个 Orion 用户[3]. 因此, 对 DNS 隐蔽隧道的检测研究对于增强网络安全具有重要意义.

DNS 隐蔽隧道检测的研究可以分为基于特征工程的方法和不依赖特征工程的方法两大类. 前者包括传统的阈值检测和基于机器学习的检测方法, 这些方法依赖于人工提取的流量特征, 这一过程既关键又复杂, 并且存在被攻击者了解并规避的风险. 后者如基于深度学习的检测方法[4], 虽然不依赖人工特征, 但现有研究往往忽略了探索这些模型的泛化能力和运行效率.

为了弥补现有检测方法的不足, 本文首先通过改进 DNS 流量表征方法, 精确输入 DNS 隐蔽隧道的载荷特征, 减少干扰字段的影响, 增强模型对未知隧道的检测能力. 其次, 提出了一个新的 DNS 隐蔽隧道检测模型—PFEC-Transformer (pcap features extraction CNN-Transformer), 结合卷积神经网络 (CNN) 和 Transformer 技术, 有效提取和分析 DNS 流量中的局部特征和长距离依赖关系. 最后, 通过实验验证了改进的表征方法与传统 FQDN (fully qualified domain name, 完全限定域名) 的效果对比, 显示 PFEC-Transformer 模型在识别 DNS 隐蔽隧道流量方面的高效性和优越的泛化能力.

1 相关工作

1.1 基于统计特征的阈值检测

van Horenbeeck[5]于 2006 年发现 NSTX 所产生的隧道流量报文的头部有一个特殊的数值标志, 并为 Snort IDS 开发了针对该签名特征的检测规则. Sheridan 等[6]将关于 Iodine 的两个签名规则应用于 Snort, 通过 Boyer-Moore 模式来匹配包含给定的签名特征的 DNS 流量包数据. Farnham 等[7]将唯一主机名数量作为检测方案的指标, 当一个域名中的主机名数量超过设定的阈值就判定为 DNS 隐蔽隧道域名, 其设定阈值为 300. 与普通的域名相比, DNS 隐蔽隧道域名的字符频率分布更加随机, 不符合 Zipf 定律, 因此在 2010 年, Born 等[8]通过统计域名的 *n*-gram 频率, 根据 Zipf 定律识别 DNS 隐蔽隧道. 2013 年, Ellens 等[9]将统计方法和流量分析结合, 他们检测固定时间窗口内的 DNS 流量, 将每条流量表示为一个五元组, 并统计了每一个流的字节数、数据包数等特征, 使用 Brodsky-Darkhovsky 方法来确定 DNS 隐蔽隧道流量. 2019 年, Alharbi 等[10]统计一段时间内 DNS 请求报文的频率和 NXDomain 报文的比例, 通过设置阈值来检测 DNS 隧道流量.

上述基于 DNS 隐蔽隧道的签名特征和统计特征的检测方法具有较低的隧道误报率, 然而仅能检测已知隧道, 且其依赖的特征易受攻击者绕过. 后续出现的 DNS 隐蔽隧道工具可以避免出现签名特征, 或通过改变通信频率和响应包类型来绕过检测.

1.2 基于统计特征的机器学习检测

随着基于特征阈值的检测方法容易被规避以及机器学习应用的普及, 越来越多的网络安全研究者转向使用机器学习技术来检测 DNS 隐蔽隧道, 以弥补阈值检测方法的局限性. 机器学习算法主要分为有监督学习和无监督学习两类.

对于有监督学习算法, 2013 年, Aiello 等[11]通过提取 DNS 请求消息的大小、响应消息的大小和 DNS 响应延迟等统计特征并结合贝叶斯、SVM 和 KNN 等机器学习算法进行检测研究. 2016 年, Buczak 等[12]通过提取 DNS 通讯包中请求域名的字符特征和通信行为特征, 如 FQDN 的数字字符比例、DNS 包的大小、平均值和方差等, 使用随机森林算法进行隧道检测. 2018 年, Almusawi 等[13]提出利用 DNS 请求或响应的包长度和域名字符熵作为检测特征, 并构建 SVM 分类模型以检测 DNS 隐蔽隧道. 2020 年, Jiang 等[14]对 DNS 流量进行多维度分析, 提取多维特征并利用机器学习的方法对 DNS 隐蔽隧道进行分析. 2021 年, Mahdavifar 等[15]统计并提取了 30 个有状态和无状态特征, 并应用到 DNS 隧道检测的机器学习模型中, 比较了 SVM、MLP、SVM 和随机森林等的检测效果. 2022 年, 刁嘉文等[16]通过设计一种模拟 DNS 隐蔽隧道攻击生成算

© 中国科学院软件研究所 http://www.c-s-a.org.cn

法, 并结合多种机器学习算法进行对比研究, 最终文章确定随机森林算法的检测效果最优.

对于无监督学习算法, Do 等[17]提取 DNS 数据包中时间、数据包长度、源 IP 和目的 IP 等信息组成特征向量, 并使用 K-means 方法进行分类检测. 2019 年, Nadler 等[18]使用孤立森林算法通过训练正常 DNS 流量数据来检测隧道流量. 2020 年, Luo 等[19]关注于 DNS 流量包的记录类型与域名位置, 通过提取这些特征并利用孤立森林算法进行检测, 在对 DNS 隐蔽隧道数据泄露与渗透数据检测上取得了一定的效果.

虽然机器学习在一定程度上弥补了人工设置阈值检测的不足, 但其提取的特征仍属于“显性”特征, 容易受攻击者改变编码、通信频率、带宽等手段绕过检测. 此外, 不同场景下 DNS 通信流量特征各异, 导致算法无法实现较好的通用性.

1.3 深度学习检测

2009 年, Hind[20]提取域名的相关特征, 并利用神经网络构建的分类器来检测 DNS 隐蔽隧道. 2015 年, Zhang 等[21]构建了一个字符级的 CNN 模型, 对 DNS 流量包中的域名进行检测. 2019 年, Liu 等[22]将 DNS 流量包中的 DNS 协议部分的字节转换成一个向量, 并使用 CNN 模型对 DNS 流量进行分类检测, 并与 SVM、逻辑回归等机器学习算法进行检测效果对比, 取得了良好的检测效果. 2020 年, 张猛等[23]将特征工程与深度学习结合, 提出改进 CNN 模型 (RDCC-CNN), 通过提取 DNS 流量中有状态与无状态特征共 48 个组成一张灰度图片作为改进模型的输入, 在其数据集上获得 99.95% 的检测准确率. 2021 年, Chen 等[24]提取 DNS 流量包中的 FQDN 字符, 构建 LSTM 的检测模型, 并取得了较好的检测效果. 2023 年, 沈传鑫等[25]利用域名语义表示和图注意力网络, 提出 DSR-GAT 方法来检测 DNS 隐蔽隧道.

尽管在 DNS 隐蔽隧道检测方面, 深度学习相关研究已取得许多进展, 但现有研究仅集中于评估模型检测已知隧道工具的能力, 而忽视了对模型泛化能力的研究, 导致模型可能仅在已知隧道流量数据集上表现出色. 此外, 现有研究还缺乏对模型运行效率的比较和评估.

2 PFEC-Transformer 检测模型设计

受 Liang 等[26]利用聚类算法分析与评估 CNN 提取的多尺度局部特征质量的研究启发, CNN 提取序列的局部特征信息之间还存在着长距离依赖关系. 本研究首先利用 CNN 对 DNS 流量帧进行多尺度局部特征提取, 随后运用 Transformer 对提取的局部特征进行长距离依赖关系模式分析, 进一步提出一种新的 DNS 隐蔽隧道检测模型 PFEC-Transformer.

2.1 模型结构

PFEC-Transformer 模型的检测流程如图 1 所示, 主要分为 3 个模块, 分别是数据预处理模块、基于 CNN 的特征提取模块和基于 Transformer 的检测分类模块. 该模型接收原始 pcap 格式的 DNS 流量包中的数据内容, 经过数据处理模块将 DNS 流量包中的数据内容转换成为一个固定长度的十进制数值序列; 之后利用基于 CNN 的局部特征提取模块自动提取序列中隐含的多尺度的局部特征, 然后这些局部特征再由 Transformer 模块进行分析, 以挖掘局部特征间的长距离依赖关系, 最终输出分类预测结果. 其中 pcap 文件格式的原始 DNS 流量数据会先经过一个数据处理模块, 该模块的主要作用是将每一帧 DNS 流量转换为一个固定长度的十进制数值序列.

图 1 检测流程

模型总体架构如图 2 所示, 在 CNN 模块中, 将数据处理得到的向量进行多个尺度上的局部特征提取, 之后将提取得到的特征向量送入 Transformer 模块中进行长距离依赖关系模式分析并输出分类预测结果.

2.2 数据预处理

从网络中捕获的 DNS 流量数据以十六进制的格式保存在 pcap 文件的数据内容中, 这种数据格式无法直接用于模型训练, 因此在数据预处理阶段, 需要将 pcap 文件中的十六进制数据转换为模型可识别的数据格式.

2.2.1 DNS 流量帧协议分析

pcap (packet capture) 文件是用于存储网络流量数

© 中国科学院软件研究所 http://www.c-s-a.org.cn

据的文件格式, 通常用于捕获网络数据包. 在捕获 DNS 流量数据时, pcap 文件中不仅包含应用层 (DNS) 协议的数据, 还包括数据链路层、网络层等其他协议层信息, 如图 3 所示.

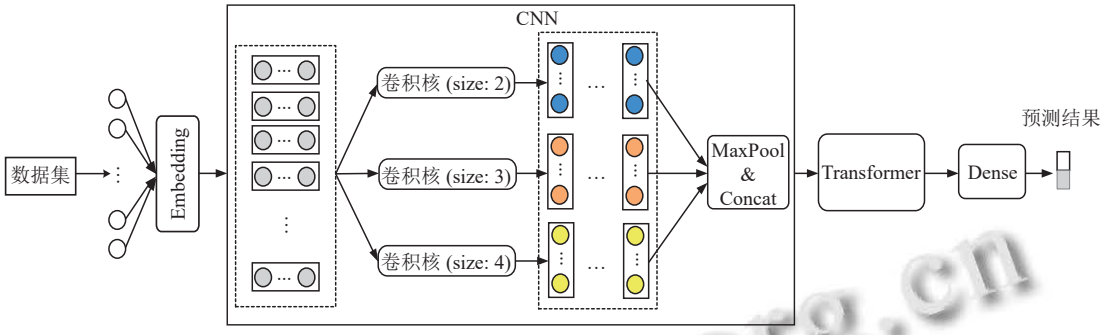

图 2 PFEC-Transformer 模型总体架构

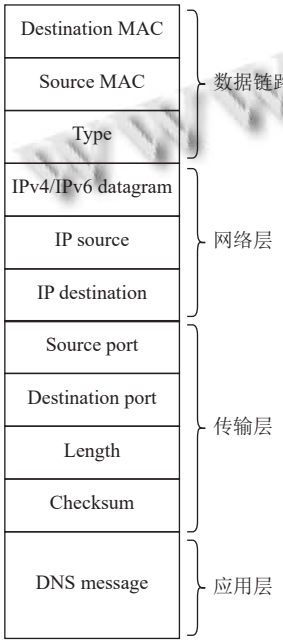

图 3 pcap 格式分析

对于一个 pcap 格式的 DNS 流量数据包, 从外到内依次为: 以太网帧 (包括目标 MAC 地址、源 MAC 地址和 Ether type 字段)、网络层 IP 数据包 (包括源 IP 地址、目标 IP 地址、IP 协议版本和长度等信息)、传输层协议 (通常为 UDP, 包含源端口、目的端口和 UDP 报文长度等信息), 最内部是 DNS 协议的应用层信息, 包括标头、查询/响应部分, 以及可能的附加部分, 如权威部分和附加部分.

2.2.2 DNS 流量表征方法

DNS 协议中的查询域名 (fully qualified name, FQDN) 是 DNS 隐蔽隧道通信载荷数据编码的关键位置, 也是检测中的重要对象. 以往的研究中, 基于阈值和机器学习的检测方式通常提取 FQDN 的统计特征, 而基于深度学习的研究则大多将 FQDN 字段作为唯一的检测输入. 然而, 仅关注 FQDN 字段的检测方法容易忽略流量数据中其他有助于 DNS 隐蔽隧道检测的字段, 同时也容易被攻击者规避, 导致检测方法失效.

此外, 基于深度学习的 DNS 隐蔽隧道检测中存在输入特征字段冗余的问题. 例如, Liang 等[26]提取了 DNS 协议部分的前 400 字节作为检测对象, 虽然考虑了除域名外的其他字段, 但忽略了受控域名 (如“hide-myself.org”) 无法作为有效的检测特征, 因为不同攻击者在不同时间使用的受控域名不同. 将受控域名用作检测字段输入模型, 容易导致模型对这些干扰特征过拟合, 降低模型的泛化能力, 使其仅在同库测试集 (仅包含训练中已知类型隧道的数据集) 上较好地完成分类任务, 而无法在实际部署中有效应用.

本文通过对先前研究进行总结, 并对 DNS 隐蔽隧道流量进行分析, 发现在 pcap 格式的数据内容中, 各协议层均包含一些干扰和诱导字段, 这些字段会影响模型检测的准确性和泛化能力. 例如, 在应用层, FQDN 的顶级域名、二级域名和三级域名由于黑客使用的受控域名不断变化, 因此成为检测的干扰字段. 在传输层, 源端口和目的端口也属于干扰字段, 因为隐蔽隧道工具的工作端口可变, 无法成为检测隐蔽隧道的特征. 类似地, 在网络层的源 IP 和目的 IP、数据链路层的源 MAC 地址和目的 MAC 地址也属于干扰字段.

为尽可能将 DNS 隐蔽隧道的载荷特征字段输入模型、同时避免干扰字段影响模型泛化能力, 本研究改进了 DNS 流量的表征方法, 使其更适用于 DNS 隐蔽隧道检测. 表征处理示意图如图 4 所示, 该方法基于

© 中国科学院软件研究所 http://www.c-s-a.org.cn

原始 DNS 流量, 在去除目标 MAC 地址、源 MAC 地址、源 IP 地址、目标 IP 地址、源端口、目的端口以及 FQDN 中的顶级域名、二级域名、三级域名等信息的基础上, 将余下信息按字节转换为 0–255 之间的十进制数, 使其变成可作为深度学习模型输入的数值序列, $DDP_n \to x_n$, $x_n \in R_l$.

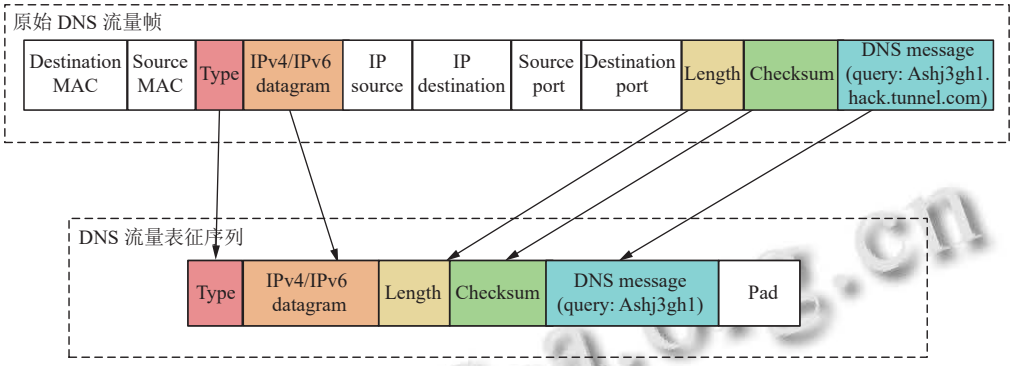

图 4 DNS 流量表征方法

以一个针对 google.com 的正常 DNS 请求为例, 其 pcap 文件格式中的十六进制数据内容如图 5 所示. 经过 DNS 流量特征表征处理后, 得到: x_n=[8 0 69 0 0 60 126 249 64 0 64 17 161 134 0 40 29 42 180 91 1 0 0 1 0 0 0 0 0 0 3 119 119 119 6 103 111 111 103 108 101 3 99 111 109 0 0 1 0 1].

```
0000  38 72 c0 5e 6b 22 78 e4  00 6c 39 cd 08 00 45 00
0010  00 3c 7e f9 40 00 40 11  a1 86 0a 00 00 22 08 08
0020  08 08 85 0a 00 35 00 28  1d 2a b4 5b 01 00 00 01
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c
0040  65 03 63 6f 6d 00 00 01  00 01
```

图 5 DNS 请求数据示例

对于任意一个 pcap 包, 本文表征方法详细处理步骤的伪代码如伪代码 1 所示.

伪代码 1. DNS 流量表征处理方法

输入: 原始 pcap 流量文件 P, 输出序列长度 l.
输出: x_n.

```
1 初始化空列表 x_n
2 for each DDP in P do
3    从数据报文 DDP 中移除源和目标的 MAC 地址
4    if DDP uses IPv4 then
5       移除源和目标的 IPv4 地址
6    else if DDP uses IPv6 then
7       移除源和目标的 IPv6 地址
8    end if
9    移除数据报文中的源和目标端口
10   fqdn=提取数据报文 DDP 中的 DNS 查询中的 FQDN
11   if Length(fqdn_labels)>3 then
12      fqdn=去除域名中的顶级域名、二级域名、三级域名信息
13   end if
14   将数据报文 DDP 中的 FQDN 替换为处理过的 fqdn
15   if Length of DDP<l then
16      填充占位数值使其数据报文长度达到 l
17   end if
18   将修改后的数据报文 DDP 转换为十进制字节序列 x
19   对 x 依照长度 l 进行截断并添加到 x_n 中
20 end for
21 return x_n
```

2.3 基于 CNN 的特征提取模块

本文中使用 CNN 来提取 DNS 流量中的局部特征. 经过表征处理的 DNS 流量数据转换为十进制数值序列, 在经过嵌入层后, 可以输入一维卷积神经网络进行特征提取. CNN 模型的特征提取过程如图 6.

由于每个 DNS 流量帧的大小不一, 而卷积神经网络模型要求每个样本的特征数量一致, 因此本文根据以往相关研究 (如文献[22]) 的经验以及多次调试测试比较, 设置 DNS 流量帧的表征序列长度固定为 400. 如果原始序列较短, 则填充“<pad>”符号 (在本文中将“<pad>”符号设置为十进制数值 256), 如果过长, 则进行截取, 以确保表征数值序列符合模型的输入要求.

在 CNN 模型中, 嵌入层用于将离散的输入 (例如单词、字符和类别标签) 映射到低维连续的向量空间. 生成的嵌入向量空间包含对输入的分布式表示, 使得相似的输入在向量空间中更加接近, 这有助于模型学习输入数据的语义关系和特征, 其嵌入公式如式 (1):

$$E(n,:) = Embed(x_i) \tag{1}$$

© 中国科学院软件研究所 http://www.c-s-a.org.cn

其中, E表示为一个维度为$V\times K$的嵌入矩阵, V是输入离散元素的数量, K是嵌入向量的维度. $E(n,:)$表示嵌入矩阵E中第 n 行, 其对应的是离散元素x_i的嵌入向量. 在本文中V为 257, K为 300.

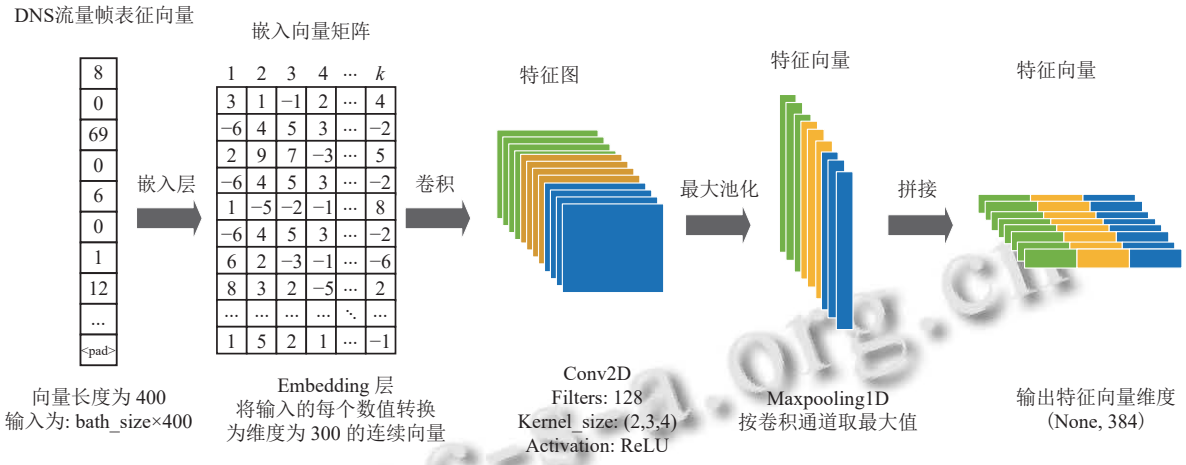

图 6 CNN 特征提取模型

本文使用⊗表示卷积运算, 一维卷积运算有一个长度为 h 的卷积核来实现, 卷积核表示为w, $w\in R^h$. 卷积后结果使用非线性激活函数生成新的序列值, 激活函数使用$f(x)$表示, 卷积公式如式 (2):

$$z_i = f(w\otimes x_{i:i+\mathrm{h}} + b_i) \tag{2}$$

为了更有效地提取原始序列中隐含的特征, 本文分别设计了卷积尺寸为 2、3 和 4 的 3 个卷积核. 随后进行最大池化处理, 保留每个卷积通道的最大值并将它们拼接在一起. 此外, 设定批大小为 128, 在 CNN 模块的输出维度为 (128, 384).

2.4 基于 Transformer 的检测分类模块

从 CNN 模块提取的多尺度局部特征图将作为 Transformer 模型的输入, 以优化模型性能. Transformer 模型的核心思想是运用自注意力机制处理序列数据, 其结构以编码器-解码器 (encoder-decoder) 架构为基础. 编码器负责处理输入序列, 解码器则基于编码器的输出和自身先前的输出生成最终序列. 经过多次调参实验, 本研究选择了最佳的 Transformer 模型参数设置: 编码器-解码器数量为 7, 随机失活率为 0.1, 前馈网络维度为 2 048, 多头注意力机制的头数为 5.

由于 Transformer 完全基于注意力机制, 不包含循环和卷积结构, 无法直接捕获序列中的位置信息. 为了解决这一问题, Transformer 引入了位置编码, 用于为每个位置的词嵌入添加关于位置的信息. 位置编码与词嵌入相加, 因此二者需要具有相同的维度. 位置编码过程的数学表示如式 (3) 和式 (4) 所示, 使用的是正弦和余弦函数的线性变化来编码位置信息, 目的是让位置编码在每个维度上有一定的周期性, 从而使模型学习到序列中位置间的长距离依赖关系. 其中, pos 表示的序列中被编码元素的位置, i 指的是词嵌入的维度, d_{model} 指的是模型的维度.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right) \tag{3}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right) \tag{4}$$

在编码器的工作流程中, 对于一个经过位置编码后的词嵌入矩阵 $X_{\mathrm{embedding}}$ 下一步是进行多头自注意力计算, 多头自注意力与自注意力在处理上的区别在于其是将向量矩阵分割为 h 个大小相同的小矩阵, 分别计算这些小矩阵的注意力之后再将其合并, 其数学表示如式 (5) 和式 (6) 所示:

$$head_i = Attention(Q_iW_i^Q, K_iW_i^K, V_iW_i^V) \tag{5}$$

$$MultiHead(Q,K,V) = Concat(head_1,\cdots,head_h)W^O \tag{6}$$

$X_{\mathrm{embedding}}$ 在应用多头注意力机制之后得到输出为 Z, 其应用过程可以由式 (7) 表示:

$$Z = MultiHead(Q,K,V)X_{\mathrm{embedding}} \tag{7}$$

Transformer 编码器-解码器之后对接 Dense 层, 其由两个全连接层组成, 主要作用是进行输出分类预测

© 中国科学院软件研究所 http://www.c-s-a.org.cn

结果, 其参数设置为 384、32、1.

3 实验评估

3.1 实验环境

本研究的实验在 CentOS 7.5.1211 操作系统下完成. 硬件环境配置如下: CPU 为 Intel Xeon E5-2695 v4 @ 2.10 GHz, GPU 为 NVIDIA Tesla P100, GPU 内存为 16 GB. 编程语言采用 Python 3.9, 深度学习框架为 PyTorch 1.2.1, 机器学习框架为 scikit-learn 1.3.0.

3.2 实验数据

由于公开的完整 pcap 文件格式的 DNS 隐蔽隧道数据较少, 本研究通过自建和收集互联网上公开流量数据的方式来构建数据集.

自建数据集包括隧道流量和正常流量两部分. 对于隧道流量, 本文在实验的虚拟网络环境中部署了 DNS 隧道, 并使用 DNS2TCP、DNScat2 和 Iodine 等常见的 DNS 隐蔽隧道工具生成隧道流量. 通过 Wireshark 和 TCPdump 等流量捕获工具抓取这些 DNS 隐蔽隧道流量, 其中涵盖了隧道建立初始化阶段和数据泄露阶段, 以及多种编码方式和查询类型. 对于正常的 DNS 流量数据, 则通过实验室和校园网内的正常 DNS 通信流量进行采集.

公开数据来源自 GitHub 上一个公开的 DNS 隧道数据集[27], 该项目作者在真实网络环境中部署了多种 DNS 隐蔽隧道工具, 进行通信并捕获隧道流量. 该数据集的独特之处在于更贴近 DNS 隐蔽隧道攻击实际发生时的环境. 除了包含自建数据集中的 3 种常见 DNS 隐蔽隧道工具外, 还包含了 DNSexfiltrator、DNSlivery、OzymanDNS 以及 cobalt_strike 等其他 7 个隐蔽隧道工具的流量数据.

最终, 本文共收集到 330 953 条数据样本, 并将收集到的数据划分为同库训练测试数据集和泛化性测试数据集. 同库训练测试数据集共 138 635 条样本, 其样本类型分布详情如表 1 所示, 其隧道流量数据全部来源于自建数据, 只包含 Iodine、DNScat2 和 DNS2TCP 这 3 种隧道工具流量样本. 在进行同库训练测试的时候, 本文将该数据集按照 6:2:2 的数量比例划分训练集、验证集和测试集, 其正常样本与隧道样本的详细分布如表 2 所示.

泛化性测试数据集共 192 318 条样本, 其隧道流量数据全部来自互联网公开的 DNS 隐蔽隧道流量, 正常 DNS 流量则通过采集实验室和校园网内正常 DNS 通信流量获得. 其数据集特点是其包含了同库训练测试数据集中所没有的隧道工具类型, 目的就是为了测试模型检测未知隧道工具流量样本的能力, 泛化性测试数据集中各个隧道工具样本和正常流量数量分布如表 3 所示.

表 1 同库训练测试数据集

样本类型	数量	总计样本数量
正常DNS流量样本	79 839	79 839
Iodine	15 178	58 796
DNScat2	22 738	
DNS2TCP	20 880	

表 2 同库训练测试数据集子集数量分布

子集名称	正常样本数量	隧道样本数量	总计样本数量
训练集	47 901	35 280	83 181
验证集	15 911	11 816	27 727
测试集	16 027	11 700	27 727

表 3 泛化性测试数据集

样本类型	数量	总计样本数量
正常DNS流量样本	143 051	143 051
Cobalt strike	2 086	49 267
DNS2TCP	5 630	
DNScat2	6 935	
DNSexfiltrator	20 202	
DNSlivery	1 122	
OzymanDNS	4 431	
Iodine	1 602	
TCP-over-DNS	3 571	
Tuns	2 336	
Reverse DNS shell	1 084	

3.3 评估指标

为了能更科学地对比评估各模型的检测效果, 使用准确率 (*ACC*)、平均精确率 (*Avg-PREC*)、平均召回率 (*Avg-REC*) 和平均 *F*1 值 (*Avg-F*1) 来评估一个模型的检测效果. 评价指标的计算公式如式 (8)–式 (11):

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{8}$$

$$Avg\text{-}PREC = \frac{TP}{2(TP+FP)}+\frac{TN}{2(TN+FN)} \tag{9}$$

$$Avg\text{-}REC = \frac{TP}{2(TP+FN)}+\frac{TN}{2(TN+FP)} \tag{10}$$

$$Avg\text{-}F1 = \frac{TP}{2TP+FN+FP}+\frac{TN}{2TN+FN+FP} \tag{11}$$

其中, *TP*、*FP*、*TN*、*FN* 分别为混淆矩阵中的 4 个评

© 中国科学院软件研究所 http://www.c-s-a.org.cn

价指标, *TP* (true-positive) 是模型正确地识别隧道样本的数量; *FP* (false-positive) 是模型错误地将正常 DNS 流量识别为隧道流量的数量; *TN* (true-negative) 是模型正确识别正常 DNS 流量样本数量; *FN* (false-negative) 指的是模型错误地将隧道样本识别成正常 DNS 流量样本.

3.4 实验分析

3.4.1 同库数据集检测实验

同库数据集检测实验仅在同库训练测试数据集中对模型进行训练和测试. 根据第 3.2 节中的训练、验证和测试子集划分, 对模型进行训练和评估, 实验的目的是比较模型在检测已知隧道类型流量数据方面的能力.

为了与机器学习算法进行对比, 本文参考先前基于机器学习的 DNS 隐蔽隧道检测相关研究, 提取了同库测试数据集中的 8 个特征, 包括报文长度、域名长度、域名熵以及域名中数字占比等, 表 4 详细列出了这 8 个特征. 在这些特征的基础上, 本文采用随机森林 (random forest, RF)、决策树 (decision tree, DT)、支持向量机 (support vector machine, SVM) 和逻辑回归 (logistic regression, LR) 这 4 种机器学习算法与深度学习模型进行了对比实验, 具体对比结果见表 5.

表 4 特征详情

特征类型	特征描述
域名特征	域名长度
	域名信息熵
	域名标签数
	域名中数字字符占比
	域名中英文字符占比
	域名中特殊符号占比
报文	DNS报文大小
	DNS报文请求记录类型

表 5 不同模型在同库数据集上的对比

模型	*ACC* (%)	*Avg-PREC* (%)	*Avg-REC* (%)	*Avg-F*1
DT	99.00	99.41	98.64	0.990 2
SVM	98.26	99.36	97.25	0.982 9
RF	99.39	99.43	99.38	0.994 1
LR	93.20	91.69	95.47	0.935 4
CNN	99.99	99.99	99.99	0.999 9
BiLSTM	99.96	99.96	99.96	0.999 6
Transformer	99.99	99.99	99.99	0.999 9
PFEC-Transformer	99.97	99.97	99.97	0.999 7

此外, 本文通过对同一模型进行多次参数调优, 选择每个模型中泛化检测效果最佳的参数设置. 各模型的参数设置说明如下.

机器学习算法的有关参数设置中, 决策树的 random_state 设置为 42, max_depth 设置为 15; 支持向量机的 *C* 参数设置为 2, 其他参数设置为默认数值; 随机森林的 n_estimators 设置为 100, 其余参数设置与决策树参数设置一样. 逻辑回归中的参数 *C* 同设置为 2, 其余参数为默认值.

基于 CNN 的模型: 滤波器的数量设置为 256, 卷积核大小共有 3 种, 分别为 2、3 和 4. 激活函数为 ReLU, 池化方式为最大池化, 全连接层大小为 768、128、1, 以二元交叉熵损失计算作为损失函数, 训练轮数为 20.

基于 LSTM 的模型: 隐藏层数设置为 2, 隐藏向量维度设置为 512, 激活函数为 tanh, 全连接层大小为 1 024、64、1, 以二元交叉熵损失计算作为损失函数, 训练轮数为 20.

基于 Transformer 的单一检测模型: 编码器-解码器数量设置为 2, 其余参数为默认设置, 以二元交叉熵损失计算作为损失函数, 训练轮数为 20.

PFEC-Transformer 的参数设置: 其 CNN 模块部分卷积通道数量设置为 128, 激活函数为 ReLU, Transformer 模块编码器-解码器数量设置为 7, 全连接层大小为 384、32、1, 学习率为 10^{-5}, 以二元交叉熵损失计算作为损失函数, 训练轮数为 20.

在实验结果中发现, 在 4 种机器学习算法中, 随机森林表现最佳, 其准确率、精确率、召回率和 *F*1 分数都超过了 99.30%. 然而, 本文提出的 PFEC-Transformer 模型在准确率、精确率、召回率方面均达到 99.97%, *F*1 分数为 0.999 7, 明显优于机器学习算法. 这表明所设计的模型能够有效学习 DNS 隐蔽隧道在流量中的隐藏特征, 在已知隧道类型的数据集上实现接近完美的分类预测.

此外, 深度学习模型相较于机器学习算法, 在拟合同库训练测试数据集中的隧道特征方面表现良好, 显著提升了检测效果. 然而, 在所有深度学习模型中, 并未发现本文模型相对于其他深度学习模型有明显优势.

3.4.2 模型泛化性实验

泛化性实验的目的是测试深度学习模型是否能正确识别未知隧道工具的流量并将其分类为 DNS 隐蔽隧道流量. 在该实验中, 所有模型均使用自建的同库训练测试数据集进行训练, 该数据集仅包含 3 种常见 DNS 隐蔽隧道工具的流量数据. 随后, 对经过训练并在同库

© 中国科学院软件研究所 http://www.c-s-a.org.cn

数据集上收敛的模型, 使用泛化性测试数据集进行泛化能力评估, 关键之处在于泛化性测试数据集中包含了另外 7 种隧道工具的流量数据. 在训练集中包含的 Iodine、DNScat2、DNS2TCP 这 3 种工具的流量为模型已知的隧道类型，而泛化性测试数据集中包含的 Cobalt Strike、DNSexfiltrator、DNSlivery、Ozyman-DNS、TCP-over-DNS、Tuns、Reverse DNS shell 其他 7 种隧道工具的流量为模型未知的隧道类型.

本文参考文献[24–26]等有关于基于 CNN 和 LSTM 的 DNS 隐蔽隧道研究工作, 并在此基础上对 CNN 和 LSTM 模型进行优化和结构组合改变, 引入了 7 种深度学习模型与 PFEC-Transformer 进行比较, 7 种模型均用自建的数据集进行训练, 对泛化数据集的分类预测结果如表 6 所示.

表 6　深度学习模型在泛化数据集检测结果对比

模型	*ACC* (%)	*Avg-PREC* (%)	*Avg-REC* (%)	*Avg-F*1
CNN	86.17	81.50	83.84	0.825 3
变长LSTM	73.44	67.04	70.32	0.679 0
BiLSTM	81.67	76.75	81.90	0.782 9
BiLSTM+Attention	90.98	87.86	88.18	0.880 2
BiLSTM+Pooling	87.09	82.67	84.72	0.835 9
CNN+Attention	85.12	80.21	82.65	0.812 6
Transformer	87.21	88.91	76.51	0.802 1
串行CNN-LSTM	92.04	91.73	86.89	0.889 5
PFEC-Transformer	92.12	94.32	85.13	0.885 5

CNN、LSTM 和 PFEC-Transformer 模型的参数设置如第 3.4.1 节所述. 串行 CNN-LSTM 模型的参数设置如下: BiLSTM 层包含 2 个隐藏层, 每个隐藏状态向量的维度为 512, 并经过两个全连接层, 参数设置分别为 1 024, 64, 1. 学习率为 10^{-5}, 以二元交叉熵损失计算作为损失函数, 训练轮次为 20.

通过观察表 6 的分类结果, 发现深度学习模型在同库数据集上能够有效完成分类任务, 但对于未知种类的 DNS 隐蔽隧道流量的识别能力仍然有限. 与其他深度学习模型相比, 本研究提出的 PFEC-Transformer 在准确率、平均精确率、平均召回率和平均 *F*1 分值分别达到 92.12%、94.32%、85.13% 和 0.885 5, 其检测准确率和精确率均处于最高水平.

在单一模型中, 添加注意力机制的 BiLSTM+Attention 模型比单一 BiLSTM 模型具有更好的检测效果, 这表明注意力机制的引入提高了 BiLSTM 模型的泛化能力. 此外, 在召回率方面, 带有注意力机制的 BiLSTM 模型要比本文提出的 PFEC-Transformer 模型高, 这是因为正常 DNS 流量和隧道流量序列中的长距离依赖关系差异更为显著. 因此, 注重原始序列长距离依赖模式分析的 BiLSTM+Attention 模型在召回率上表现更突出.

在对比模型中, 串行 CNN-LSTM 模型与 PFEC-Transformer 相似, 都首先利用 CNN 提取 DNS 流量中的多尺度局部特征, 然后对这些特征进行长距离依赖模式分析, 并输出分类预测结果. CNN 模块的参数设置与 PFEC-Transformer 一致. 然而, 与 PFEC-Transformer 不同的是, 串行 CNN-LSTM 模型使用 Att-BiLSTM 模块进行特征间的长距离依赖模式分析, 而 PFEC-Transformer 则采用 Transformer. 观察发现, 串行 CNN-LSTM 模型的检测效果与 PFEC-Transformer 相近, 特别在召回率上比 PFEC-Transformer 高 1.76%. 综合来看, 两个模型的泛化性能相差不大.

3.4.3　模型运行性能对比实验

为了评估模型的运行性能, 本次实验统计了 CNN、BiLSTM、Transformer、串行 CNN-LSTM 和 PFEC-Transformer 在训练过程中的显卡内存占用、平均每轮训练时间、检测耗时以及模型在泛化测试中的准确率, 共计 4 个指标, 以衡量模型的运行性能. 其中, 用于训练的共有 110 908 条样本, 每个模型的训练轮次设置为 10 轮, 测试所用共有 192 318 条样本. 不同模型的运行性能对比如表 7 所示.

表 7　模型运行性能对比

模型	准确率 (%)	平均每轮训练时长 (s)	训练占用GPU内存 (MiB)	测试耗时(s)
CNN	86.17	65	1 293	30
BiLSTM	81.67	298	3 037	194
Transformer	87.21	122	4 321	89
串行CNN-BiLSTM	92.04	367	4 831	218
PFEC-Transformer	92.12	97	1 809	24

通过观察实验对比结果, 发现本节提出的检测方法 PFEC-Transformer 在所有模型中测试耗时最短. 同时可以观察到, 基于 LSTM 的检测方法耗费的计算资源较大, 训练和测试时间也较长. 这是因为 LSTM 模型缺乏并行计算能力, 每个时间步的计算都依赖于上一时间步的结果. 另一方面, 基于 CNN 的检测方法计算资源消耗和时间成本较低, 因为采用了一维卷积, 在卷积层中共享卷积核参数, 无需重新分配存储参数, 使得基于 CNN 的检测模型表现优异. 相比于 CNN 模型, 虽然 PFEC-Transformer 在训练时长和 GPU 内存占用方

© 中国科学院软件研究所 http://www.c-s-a.org.cn

面不具优势, 但在模型准确率最高时, CNN 模型的卷积通道数量为 256, 而 PFEC-Transformer 仅为 128. 这种参数设置的差异导致 CNN 在推理时需要消耗更多时间, 因此在测试时间上, PFEC-Transformer 耗时更短. 此外, PFEC-Transformer 经过 CNN 提取后输入到 Transformer 模块的数据张量维度为 (None, 384), 相比于传统 Transformer 模型的输入张量 (None, 400, 300) 在数据处理维度上要更低. 因此, PFEC-Transformer 在平均每轮训练时长、CPU 内存占用和测试耗时方面均比 Transformer 更优秀.

此外, 对比串行 CNN-LSTM 与本文提出的 PFEC-Transformer 的运行性能对比如图 7 所示, 其中纵轴代表着模型指标占两个模型在该指标总和的百分比. 可以观察到, 虽然根据第 3.4.2 节的实验结果显示, 串行 CNN-LSTM 在泛化检测召回率上比 PFEC-Transformer 高 1.76%, 但 PFEC-Transformer 却极大地减少了模型运行所消耗的计算资源, 在模型运行平均每轮训练时长和测试耗时上减少 50% 以上, 在训练占用的 GPU 内存上相比也同样减少 50% 以上. 因此, 虽然串行 CNN-LSTM 在泛化检测效果的召回率指标上比 PFEC-Transformer 要高, 但在模型的运行效率上不如 PFEC-Transformer, 综合评比下 PFEC-Transformer 模型更具实用性.

3.4.4 DNS 流量表征对比实验

为了验证本文提出的 DNS 流量表征方法的有效性, 进行了一组表征方式对比的实验. 该实验通过提取自建数据集和泛化测试集的完全限定域名 (fully qualified domain name, FQDN), 将 FQDN 字符序列转换成数值序列输入深度学习网络进行分类预测, 实验结果如表 8 所示.

表 8 FQDN 表征检测结果

模型	*ACC* (%)	*Avg-PREC* (%)	*Avg-REC* (%)	*Avg-F*1
CNN	79.85	72.03	83.50	0.738 7
变长LSTM	70.45	69.03	81.74	0.665 9
BiLSTM	82.74	74.88	86.85	0.774 1
BiLSTM+Attention	84.44	76.92	90.11	0.797 4
BiLSTM+Pooling	86.89	78.41	88.03	0.815 0
CNN+Attention	82.26	73.97	73.76	0.738 7
Transformer	75.79	69.49	80.81	0.702 1
串行CNN-BiLSTM	87.39	79.80	75.46	0.773 1
PFEC-Transformer	88.37	85.53	83.13	0.842 2

观察结果显示, 基于 FQDN 的表征方式的模型检测效果不及本文提出的基于 pcap 的 DNS 流量表征方式. 以模型泛化检测准确率为例, 基于 FQDN 与基于 pcap 的泛化检测准确率对比见图 8. 通过对比实验结果发现, 除了单一 BiLSTM 之外, 大部分模型在使用 pcap 表征方式后的准确率都有显著提升. 这是因为 pcap 的 DNS 流量表征方法不仅考虑了 DNS 流量包的查询域名, 还包括了查询类型和其他协议层的字段信息, 使得模型学习到的特征更加丰富, 有助于深度学习模型做出更准确的分类预测. 而单一 BiLSTM 模型更适合用于分析字符串的 *n*-gram 信息, 所以在 FQDN 表征方法使其能够集中检测域名字符, 使其检测效果更佳.

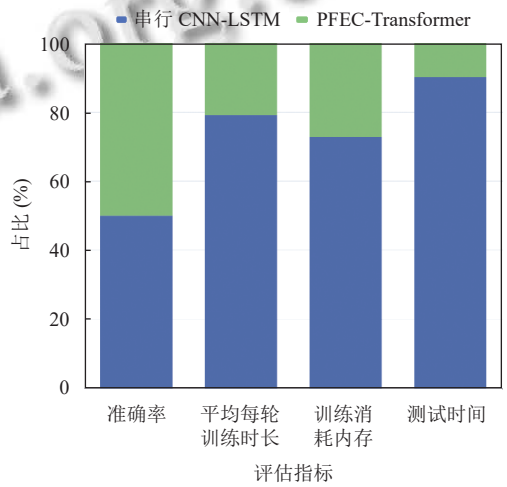

图 7 模型运行性能对比

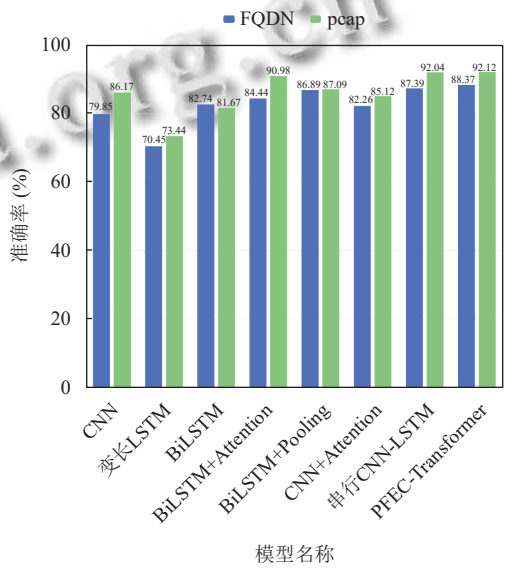

图 8 泛化检测准确率对比

4 总结

本文首先优化了 DNS 流量的表示方法, 通过消除

© 中国科学院软件研究所 http://www.c-s-a.org.cn

干扰字段提高了模型的泛化能力. 随后提出了全新的 DNS 隐蔽隧道检测模型 PFEC-Transformer. 最后, 利用自建和收集的公开数据, 建立了同库训练测试数据集和泛化性测试数据集, 验证了 PFEC-Transformer 模型和表示方法的有效性. 实验结果显示, PFEC-Transformer 模型在已知隧道流量检测方面明显优于机器学习算法, 在未知隧道流量检测方面具有最高的准确率和精确率, 同时具有比其他深度学习模型更高的运行效率, 并能有效提升模型的泛化能力. 这些实验结果突显了本文模型在实际应用中具有巨大潜力, 能够有效识别网络中的 DNS 隐蔽隧道, 为网络安全提供强有力支持. 下一步研究工作将继续优化 PFEC-Transformer 模型的泛化能力, 提高其在未知隧道流量检测方面的性能表现. 同时, 进一步考虑模型的可扩展性和实际部署, 实现更快速和精确的 DNS 隐蔽隧道检测.

参考文献

1 罗杰云, 贺敏伟. DNS 协议的安全浅析. 计算机系统应用, 2004, 13(1): 36–38, 35. [doi: 10.3969/j.issn.1003-3254.2004.01.009]

2 Pearson O. DNS tunnel-through bastion hosts. https://seclists.org/bugtraq/1998/Apr/79. (1998-04-13).

3 Mandiant. Highly evasive attacker leverages SolarWinds supply chain to compromise multiple global victims with SUNBURST backdoor. https://www.mandiant.com/resources/blog/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor. (2022-12-06).

4 Wang Y, Zhou AM, Liao S, *et al*. A comprehensive survey on DNS tunnel detection. Computer Networks, 2021, 197: 108322. [doi: 10.1016/j.comnet.2021.108322]

5 van Horenbeeck M. Detection of DNS tunneling. https://www.daemon.be/maarten/dnstunnel.html#detect. (2006-05-27).

6 Sheridan S, Keane A. Detection of DNS based covert channels. Proceedings of the 2015 European Conference on Cyber Warfare and Security. Academic Conferences International Limited, 2015. 267.

7 Farnham G, Atlasis A. Detecting DNS tunneling. SANS Institute InfoSec Reading Room, 2013, 9: 1–32.

8 Born K, Gustafson D. Detecting and visualizing domain-based DNS tunnels through *n*-gram frequency analysis. Journal of Information System Security, 2011, 7(2): 27–48.

9 Ellens W, Żuraniewski P, Sperotto A, *et al*. Flow-based detection of DNS tunnels. Proceedings of the 2013 IFIP International Conference on Autonomous Infrastructure, Management and Security. Barcelona: Springer, 2013. 124–135.

10 Alharbi T, Koutny M. Domain name system (DNS) tunnelling detection using structured occurrence nets (SONs). Proceedings of the 2019 International Workshop on Petri Nets and Software Engineering (PNSE 2019). Newcastle University, 2019.

11 Aiello M, Mongelli M, Papaleo G. Basic classifiers for DNS tunneling detection. Proceedings of the 2013 IEEE Symposium on Computers and Communications. Split: IEEE, 2013. 880–885.

12 Buczak AL, Hanke PA, Cancro GJ, *et al*. Detection of tunnels in PCAP data by random forests. Proceedings of the 11th Annual Cyber and Information Security Research Conference. Oak Ridge: ACM, 2016. 16.

13 Almusawi A, Amintoosi H. DNS tunneling detection method based on multilabel support vector machine. Security and Communication Networks, 2018, 2018: 6137098.

14 Jiang K, Wang F. Detecting DNS tunnel based on multidimensional analysis. Proceedings of the 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE). Harbin: IEEE, 2020. 272–275.

15 Mahdavifar S, Salem AH, Victor P, *et al*. Lightweight hybrid detection of data exfiltration using DNS based on machine learning. Proceedings of the 11th International Conference on Communication and Network Security. Weihai: ACM, 2021. 80–86.

16 刁嘉文, 方滨兴, 田志宏, 等. 基于攻击流量自生成的 DNS 隐蔽信道检测方法. 计算机学报, 2022, 45(10): 2190–2206. [doi: 10.11897/SP.J.1016.2022.02190]

17 Do VT, Engelstad P, Feng BN, *et al*. Detection of DNS tunneling in mobile networks using machine learning. Proceedings of the 2017 International Conference on Information Science and Applications. Singapore: Springer, 2017. 221–230.

18 Nadler A, Aminov A, Shabtai A. Detection of malicious and low throughput data exfiltration over the DNS protocol. Computers & Security, 2019, 80: 36–53.

19 Luo M, Wang QY, Yao YP, *et al*. Towards comprehensive detection of DNS tunnels. Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC). Rennes: IEEE, 2020. 1–7.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

20 Hind J. Catching DNS tunnels with A.I. https://defcon.org/images/defcon-17/dc-17-presentations/defcon-17-jhind-dns_tunnels_with_ai.pdf. (2009-07-31).

21 Zhang X, Zhao JB, LeCun Y. Character-level convolutional networks for text classification. Proceedings of the 28th International Conference on Neural Information Processing Systems. Montreal: MIT Press, 2015. 649–657.

22 Liu C, Dai L, Cui WJ, *et al*. A byte-level CNN method to detect DNS tunnels. Proceedings of the 38th IEEE International Performance Computing and Communications Conference (IPCCC). London: IEEE, 2019. 1–8.

23 张猛, 孙昊良, 杨鹏. 基于改进卷积神经网络识别 DNS 隐蔽信道. 通信学报, 2020, 41(1): 169–179. [doi: 10.11959/j.issn.1000-436x.2020017]

24 Chen SJ, Lang B, Liu HY, *et al*. DNS covert channel detection method using the LSTM model. Computers & Security, 2021, 104: 102095.

25 沈传鑫, 王永杰, 熊鑫立. 基于图注意力网络的 DNS 隐蔽信道检测. 信息网络安全, 2023, 23(1): 73–83. [doi: 10.3969/j.issn.1671-1122.2023.01.009]

26 Liang JB, Wang SX, Zhao S, *et al*. FECC: DNS tunnel detection model based on CNN and clustering. Computers & Security, 2023, 128: 103132.

27 IsGt93. DNS-tunnel-datasets. https://github.com/isGt93/dns-tunnel-datasets. (2022-03-13).

(校对责编: 孙君艳)

www.c-s-a.org.cn

www.c-s-a.org.cn

© 中国科学院软件研究所 http://www.c-s-a.org.cn