基于联邦学习的第三方库流量识别

崔华俊[1,2], 孟国柱[1,2], 李玥琦[1,2], 张 棪[1,2], 代玥玥[3], 杨慧然[1,2],
朱大立[1,2], 王伟平[1,2]

[1]中国科学院信息工程研究所, 北京 中国 100093
[2]中国科学院大学网络空间安全学院, 北京 中国 100049
[3]华中科技大学网络空间安全学院, 湖北 中国 430074

摘要 第三方库(Third-party Library, TPL)已经成为移动应用开发的重要组成部分, 开发者通常在应用中集成 TPL 以实现诸如广告、消息推送、移动支付等特定功能, 从而提高开发效率并降低研发成本。然而, 由于 TPL 与其所在的移动应用(宿主应用)共享相同的系统权限, 且开发者对 TPL 自身的安全隐患缺乏了解, 导致近年来由 TPL 引起的安全问题频发, 给公众造成了严重的信息与隐私安全困扰。TPL 的流量识别对于精细化流量管理与安全威胁检测具有重要意义, 是支撑对宿主应用与 TPL 之间进行安全责任判定的重要能力, 同时也是促进 TPL 安全合规发展的重要检测方法。然而目前关于 TPL 的研究主要集中于 TPL 检测、TPL 引起的隐私泄漏问题等, 关于 TPL 流量识别的研究十分少见。为此, 本文提出并实现了一种用于 TPL 流量识别的框架——LibCapture, 该框架首先基于动态插桩技术与 TPL 检测技术设计了自动生成 TPL 加密流量数据集的方法。其次, 针对隐私保护以及数据共享的问题, 构建了基于卷积神经网络的联邦学习模型, 用于识别 TPL 流量。最后, 通过对 2327 个真实应用的流量测试证明了本文所提框架具有较高的流量识别准确率。此外, 本文分析了联邦学习参与方本地样本数据差异性给全局模型聚合带来的具体影响, 指出了不同场景下的进一步研究方向。

关键词 加密流量识别; 第三方库; 联邦学习; 动态插桩
中图法分类号 TN915.08 DOI 号 10.19363/J.cnki.cn10-1380/tn.2023.05.09

A Third-party Library Traffic Identification Framework Using Federated Learning

Cui Huajun[1,2], Meng Guozhu[1,2], Li Yueqi[1,2], Zhang yan[1,2], Dai Yueyue,
Yang Huiran, Zhu Dali[1,2], Wang Weiping[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
[3] School of Cyber Science and Engineering, Huazhong University of Science and Technology, Hubei 430074, China

Abstract Third-party Library (TPL) has become a vital component in mobile app development. Developers usually integrate TPLs into apps to realize specific functions such as advertising, message pushing, and mobile payment, thus improving development efficiency and reducing research and development costs. However, since TPLs share the same system permissions with its mobile application (host application), and developers always lack understanding of TPL's security risks, security and privacy leakage threats caused by TPLs have occurred frequently in recent years, causing serious information security and privacy leakage problems to the public. TPL traffic identification is of great significance for fine-grained traffic management and security threat detection. It is an important capability to support the determination of security responsibilities between the host apps and TPLs, and a critical detection method to promote the development of TPL security compliance. Unfortunately, the existing studies on TPL mainly focus on TPL detection, privacy leakage caused by TPLs, etc. To the best of our knowledge, there is little research on TPL traffic identification. To this end, we propose a new framework, named LibCapture, to identify TPL traffic. The framework first designs a method to automatically generate TPL encrypted traffic datasets based on dynamic hooking and TPL detection techniques. Secondly, for privacy protection and data sharing, we propose a CNN-based federated learning (FL) model to identify TPL traffic. Finally, we apply our framework to 2327 real-world apps, and the results show that our proposed framework can achieve high TPL traffic identification accuracy, and demonstrate that FL can achieve similar accuracy compared with the non-FL method. In addition, to study how the local datasets of participants influence the global model during model aggregation, we analyze the impact made on the global model when the participants in FL have different local datasets and point out the further

通讯作者: 张棪, 博士, 副研究员, Email: zhangyan80@iie.ac.cn。
本课题得到国家重点研发计划(No. 2019YFB1005205)资助。
收稿日期: 2022-09-06; 修改日期: 2022-11-18; 定稿日期: 2023-03-31

research direction in different scenarios.

Key words encrypted traffic identification; third-party library; federated learning; dynamic hooking

1 引言

在移动应用开发中, 第三方库(Third-party Library, TPL)已经成为重要组成部分。安卓应用 TPL 通常以“.jar”文件或“.aar”文件的形式呈现, 开发者通过集成 TPL 来完成某些特定功能, 从而提高研发效率, 降低研发成本。例如, 谷歌 AdMob 可以实现 app 内广告的呈现, 蚂蚁金服 AliPay 可以实现 app 内支付宝付款功能。已有工作表明, app 中超过 60% 的代码属于 TPL[1], 且平均每个 app 集成 6-9 个 TPL[2]。由此可见, TPL 已成为移动应用中的重要基础组件。

然而, TPL 的广泛使用也带来了许多网络安全隐患。第一, TPL 可能会引起隐私泄露问题。由于 TPL 与其所处 app(宿主 app)共享进程环境和系统权限, 用户无法区分 app 申请的权限是用于自身业务还是由 TPL 使用于其它业务(例如用户信息采集), 因此很难在 app 申请权限时做出正确的判断。目前已有许多 TPL 隐私泄露问题方面的研究[3-5], 如 Feal[6]等人讨论了针对不同类别 TPL 的权限搭载(privilege piggybacking)和隐私数据监管问题。Wang[7]等人指出了一种新型的隐私窃取攻击方法——跨库数据采集(cross library data harvest, XLDH)。该文发现一些恶意 TPL 会主动检测宿主 app 中是否有特定的目标 TPL, 并利用 Java 反射机制从目标 TPL 中获取用户隐私数据。Recon[8]利用机器学习算法结合用户人工标注的方式, 提出了可检测 app 隐私泄露问题的模型。第二, TPL 对其他研究领域也会造成影响。以 app 流量识别为例, 该类研究的目标是判断网络中的流量来自于哪个 app, 从而为网络运营商的网络结构优化和服务质量(Quality of Service, QoS)监测提供分析依据。虽然已有大量工作对 app 流量识别技术开展了研究[9-11], 但 Taylor[12-13]等人指出 TPL 流量会对 app 流量识别的模型性能产生负面影响。因为不同的 app 中可能会出现相同的 TPL 流量(如广告库流量), 这些 TPL 流量会让模型误认为它们同时属于多个分类, 从而导致分类器模型性能下降。除了隐私泄露和 app 流量识别问题, TPL 流量也会对恶意 app 检测、app 重打包检测等产生影响[14]。由上述分析可知, 识别 TPL 流量对于流量管理、恶意 TPL 识别、隐私泄露问题检测等研究具有重要意义。

图 1 展示了两个真实 app 的网络访问行为, 可以看出, 集成相同 TPL 的 app 均会访问该 TPL 服务端域名, 集成不同 TPL 的 app 会访问不同 TPL 服务端域名, 此外, 在访问 TPL 域名的同时, app 还会访问自身服务端(app 服务端)的域名。基于以上分析, 从流量产生者的角度来看, 可将移动流量分为两部分: 宿主 app 流量与 TPL 流量。宿主 app 流量即由 app 为实现自身业务功能所产生的流量, 该类流量通常存在于 app 与其服务端(app 服务端)之间; TPL 流量即由 app 内集成的 TPL 产生的流量, 该类流量通常存在于 TPL 与其服务端(TPL 服务端)之间。然而, 由于越来越多流量基于加密协议传输, 导致研究人员难以从混合的流量中区分出宿主 app 流量和 TPL 流量。这种流量混合的情况对安全责任的界定以及其他安全研究产生了不利影响。

关于 TPL 的已有研究工作主要集中于 TPL 检测(2.1 节)以及 TPL 安全或者隐私问题分析(2.3 节)。然

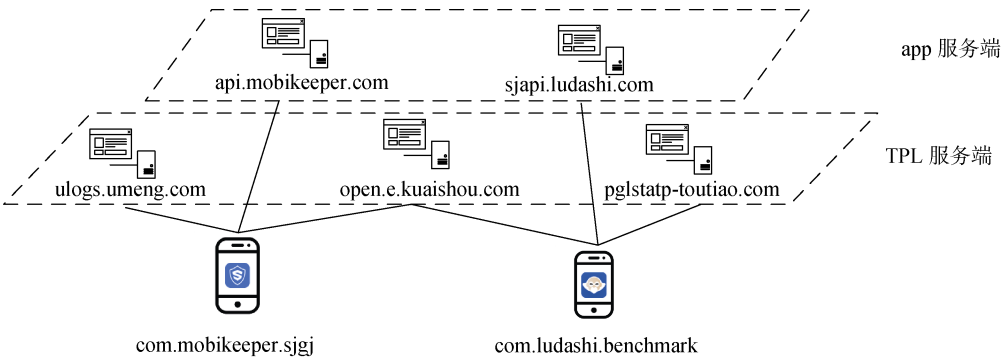

图 1 移动应用网络访问行为示意图

Figure 1 Visiting graph of mobile apps

而, 关于 TPL 流量识别的研究工作十分少见。其原因有两点: 第一, 公开数据集缺失。TPL 流量与宿主 app 流量混合在一起, 且大多数是 HTTPS 加密流量, 缺乏区分 TPL 流量的方法, 导致目前暂未出现具有标签信息的 TPL 流量公开数据集; 第二, 隐私保护与数据共享。移动设备以及 app 包含大量用户隐私信息, 采集app流量数据涉及数据安全与隐私相关法律法规, 导致科研工作者或技术研发公司在从事这方面研究工作时, 难以共享来自于不同用户的app流量数据来共同推进研究。

为解决 TPL 流量识别的问题, 本文提出了一个 TPL 流量识别的框架——LibCapture。针对目前 TPL 流量识别问题存在的数据集缺失以及隐私保护与数据共享问题, 所提 LibCapture 可解决以下两方面难点: 1)LibCapture 可基于动态插桩技术与 TPL 识别方法, 在加密流量中准确标记出 TPL 流量, 并自动生成 TPL 数据集; 2)LibCapture 可构建基于卷积神经网络的联邦学习模型(联邦 CNN 模型), 用于识别 TPL 流量。联邦 CNN 模型可实现在不共享参与方本地数据的情况下, 利用所有参与方本地数据共同训练全局模型, 从而获得比非联邦模型更多的数据量, 并取得更好的模型性能。此外, 其在用户数据隐私保护与可扩展性方面具有天然优势, 十分适合移动应用流量分析场景。总体而言, 本文提出的 LibCapture 可在app运行时同时采集app发出的加密流量及该流量被加密之前对应的明文流量, 并识别 app 中集成的 TPL, 关联 TPL 的明文流量与加密流量, 从而生成 TPL 流量数据集。基于 TPL 数据集, 结合联邦学习和卷积神经网络(Convolutional Neural Network, CNN), 本文提出了 TPL 加密流量识别方法, 并基于超过 2000 个真实 app 的流量验证了 LibCapture 的有效性。

本文的贡献总结如下:

1) 设计并实现了一个 TPL 流量识别框架——LibCapture。该框架可自动爬取 app、运行 app、采集流量并自动生成 TPL 流量数据集。此外, 提出了基于联邦学习的 TPL 流量识别模型, 实现了隐私保护下的多数据源联合训练, 为 LibCapture 提供了良好的可扩展性。本着“开源科研”的思想理念, 我们在 Github 公开了本文的核心代码, 仓库地址为: https://github.com/BlcDing/LibCapture。

2) 提出了一种基于动态插桩技术的TPL加密流量数据集生成方法。LibCapture 采用动态插桩技术捕获 app 流量在加密之前的明文流量, 同时利用 tcpdump 捕获app发出的加密流量, 并结合 TPL 信息与明-密文流量对应信息将明文流量关联至加密流量, 从而自动生成 TPL 加密流量数据集。

3) 提出了一种基于联邦学习的TPL加密流量识别模型。结合联邦学习理念与 CNN 卷积神经网络, LibCapture 实现了在本地原始数据不共享的前提下, 分布式训练与更新 TPL 流量识别模型, 并分析了联邦学习不同参与方本地数据集差异性对模型的影响, 指出了对应的解决方法与未来的研究方向。我们将 LibCapture 应用于 2327 个 app, 采集 app 流量并生成 TPL 数据集。实验结果证明 LibCapture 可取得 92.34% 的分类准确率。

本文剩余部分的组织结构如下: 第 2 章介绍了与本文相关的研究工作, 包括 TPL 检测、联邦学习以及加密移动流量分析的现状; 第 3 章详细介绍了 LibCapture 的系统设计与实现原理, 包括网络流量采集模块、数据集生成模块和基于联邦学习的 TPL 流量识别模块; 第 4 章介绍了实验结果与分析, 包括 LibCapture 的实现、数据集、TPL 识别模型的效果验证与分析及与已有工作的对比; 第 5 章对本文的不足之处以及方法特点进行讨论; 第 6 章对本文工作进行了总结, 并指出了下一步的研究计划和未来的研究方向。

2 相关工作

2.1 TPL 检测

TPL 检测是指识别一个 app 中集成了哪些 TPL, 这是一类基础性质的研究工作, 因为研究人员可基于 TPL 检测的结果进行更深层次的分析, 常见的应用场景有 TPL 与宿主 app 代码分离、TPL 代码缺陷检测等。目前 TPL 检测的研究方法主要可以分为三类: 第一类是基于白名单的检测, 其基本思想是根据 TPL 包名[7]或其所访问域名[15]的白名单列表进行匹配。该类方法的优势在于简单快速, 缺点是需要维护并更新白名单列表, 且该类方法在面临代码混淆或者流量加密等技术时将会失效; 第二类是基于机器学习的特征匹配方法, 该类方法通过提取 TPL 的源代码特征形成 TPL 特征库, 从而训练 TPL 检测模型, 特征可以包括代码控制流信息(control flow graph)、系统 API 调用信息等[14]。该类方法的优点在于通过提取源代码级别的抽象特征, 可以应对代码混淆等源码保护手段, 缺点是针对未知 TPL 缺乏检测能力, 当面对特征库中未出现的 TPL 时, 检测效率并不理想, 代表性研究工作有 LibRadar[16]、LibScout[17-18]、LibD[19]、LibPecker[2]等; 第三类是基于流量聚类的方法, 该类方法的基本思想是集成在

不同app中的TPL产生的流量具有相似性, 因此通过采集大量app流量找到不同app产生的相似流量来检测 TPL。该类方法的优点在于基于聚类的方法可以检测未知 TPL, 缺点是只能检测出具有网络流量行为的 TPL, 对于没有网络流量行为的 TPL 不具备检测能力, 代表性研究工作有 LibHunter[20]。

2.2 联邦学习

联邦学习(Federated Learning, FL)的思想由谷歌最早提出, 其本质是一种分布式协作的模型训练方法。FL 系统结构由一个服务端与多个参与训练的客户端组成, 各客户端利用自身本地的数据训练本地模型, 并将模型的相关参数传递至服务端, 服务端根据特定算法对来自客户端的参数信息进行融合与更新, 并将新的模型参数下发至客户端, 从而实现多个客户端在不共享本地数据的情况下共同协作训练模型[21], 这样的优势使得联邦学习在隐私保护领域应用十分广泛。谷歌将联邦学习技术应用在其手机键盘(Google keyboard)中, 通过大量用户的手机输入习惯共同训练输入法模型[22], 为用户提供智能的输入推荐与提示。与传统集中式的人工智能模型相比, 联邦学习无需将所有训练数据都上传至云端, 只是将模型相关参数进行传递, 既保护了用户的隐私信息, 又能够结合多方的数据优势共同训练高性能模型。

2.3 加密移动流量分析

移动流量分析的研究已有较长的时间, 但是经过我们调研发现, 针对 TPL 流量识别的研究十分少见。与 TPL 流量识别关联性最高的是 app 流量识别, 即区分网络流量来自于哪个 app。在这方面的代表性工作有 AppScanner[12]和 NetworkProfiler[23], app 流量识别在恶意流量检测、网络资源分配与管理方面具有重要意义。然而, 上述工作均没有解决所谓“背景噪声流量”或“模棱两可流量”的问题——即如何鉴别由 app 中的 TPL 所产生的网络流量的问题。

针对隐私信息泄露的流量分析是与本文相关的另一个研究领域, 其主要目标是评估app向远端服务器传输了什么隐私数据。已有的研究工作[8, 24-25]通常使用中间人工具(Man-In-The-Middle, MITM)来解密 HTTPS 流量, 并基于解密后的明文流量进行隐私泄露问题分析, 常见的 MITM 工具有 Fiddler[26]、Meddle[27]等。然而, MITM 工具无法区分流量来自于宿主 app 还是 TPL, 因此这类工作在涉及到 TPL 流量分析时, 通常基于 2.1 节中提到的白名单列表来进行区分, 显然这种方法无法解决 TPL 加密流量识别的问题。其他与移动加密流量分析相关的研究还有恶意软件检测[28-29]、广告欺诈检测[30-31]以及位置分析[32]等。

综上所述, 当前对于移动加密流量分析已有的研究工作主要集中于app流量识别, 基于解密后明文流量的隐私泄露风险分析、广告欺诈、位置分析以及恶意软件流量检测等。关于 TPL 流量识别的研究工作尚未出现, 然而, TPL 流量识别却对上述研究领域有着重要的支撑和促进作用。

3 方法

3.1 系统框架

LibCapture 由三部分组成, 图 3 展示了系统各部分之间的关系与总体工作流程, 系统组成说明如下:

1) 流量采集模块: 该模块从应用市场自动爬取app, 自动运行 app 并采集运行过程中产生的流量。针对某一具体 TCP 流, 流量采集模块基于动态插桩技术能够同时捕获其在加密之前(明文流量)与加密之后的流量(密文流量)。我们将在 3.2 节中进行详细介绍。

2) 数据集生成模块: 该模块首先检测出流量中存在的 TPL, 并根据明文流量自动标记密文流量, 从而生成 TPL 加密流量数据集。我们将在 3.3 节中进行详细介绍。

3) 基于联邦学习的 TPL 流量识别模块: 该模块利用联邦学习技术与 CNN 卷积神经网络, 对 TPL 加密流量数据集进行分布式训练与建模, 并识别 TPL 加密流量。我们将在 3.4 节中进行详细介绍。

3.2 流量采集模块

进行 TPL 加密流量识别需要克服的第一个困难就是公开数据集缺失的问题。为了采集 TPL 的加密流量, 本文设计并实现了一种基于动态插桩(hooking)技术的明文流量与密文流量捕获方法。

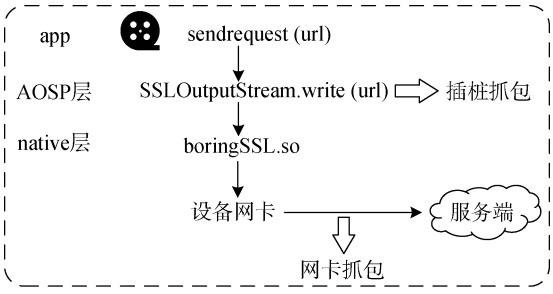

图 2 流量采集模块示意图

Figure 2 Traffic capturing module

图 2 展示了 app 发出网络请求过程的流程图, 首先, app 会将其要访问的 URL 作为参数传递给

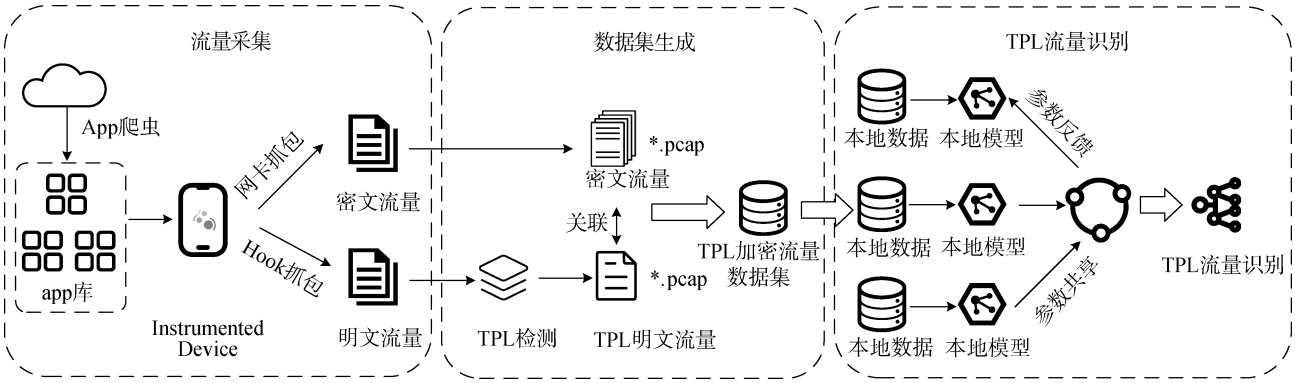

图 3　LibCapture 系统工作流程图

Figure 3　System workflow of LibCapture

“sendrequest()”方法, 随后调用安卓 Framework 层的默认系统接口“SSLOutputStream”, 由“SSLOutputStream”调用 native 层的 boringSSL.so 对 URL 进行加密, 并最终经由移动设备网卡将请求发送出去。

也就是说, app 发送的网络请求数据在被送入 boringSSL.so 之前是明文, 由 boringSSL.so 加密之后变成密文并最终从 Android 设备网卡发送至服务端; 类似的, app 接收的请求在进入 boringSSL.so 之前是密文, 经由 boringSSL.so 解密之后变成明文进入“SSLInputStream”并最终由 app 读取数据。因此, 在“SSLOutputStream”和“SSLInputStream”中的数据是未被加密的(明文流量)。在上述过程中, “sendrequest()”方法可由不同的网络库实现[33-36], 其方法名与参数名均可能不同, 但无论具体如何实现, 均会调用安卓 Framework 层的“SSLOutputStream”和“SSLInputStream”接口。如果能够在此处捕获到该数据, 并且在移动设备网卡处捕获到加密数据(密文流量), 便能够将明文流量与密文流量关联, 从而为数据集生成模块提供标签信息。

基于上述分析, 为了在 app 运行过程中同时捕获加密流量以及与加密流量对应的明文流量, 我们设计并实现了流量采集模块。具体来说, 流量采集模块由三个服务组成: 调度服务、加密流量采集服务(Encrypted Traffic Capturing, ETC)和明文流量采集服务(Unencrypted Traffic Capturing, UTC)。对每一个测试 app, LibCapture 首先将该 app 安装进测试设备, 启动该 app 并模拟各类用户行为与 app 进行交互使得 app 产生网络流量。与此同时, 调度器会同时启动 ETC 和 UTC 服务进行流量采集, 测试结束后, 关闭 app 进程, 并关闭 ETC 和 UTC。我们基于 tcpdump 实现了 ETC 服务, 用于捕获从 Android 设备网卡中发出的加密流量, 基于 LibHunter 中的动态插桩技术, 对安卓 Framework 层中的“SSLOutputStream”和“SSLInputStream”进行插桩, 从而捕获 app 运行时产生的明文流量。

通过上述方式, 我们从不同的采集点捕获了来自于该 app 的两份流量, 一份为从设备网卡上捕获的加密流量, 另一份为基于动态插桩技术捕获的明文流量。

3.3　数据集生成模块

在捕获了 app 加密流量与明文流量的基础上, 我们需要解决的第二个问题是如何检测流量中包含了哪些 TPL 并将 TPL 流量与宿主 app 流量进行区分, 从而生成 TPL 流量数据集。为此, 我们设计并实现了数据集自动生成模块。如图 4 所示, 对于检测流量中存在的 TPL 的问题, 我们基于先前工作 LibHunter 对采集的明文流量以及函数调用栈进行训练, 得到流量中包含的 TPL 列表。LibHunter 是一种使用无监督的方式进行 TPL 检测的方法, 其基本思想是集成了相同 TPL 的 app 会产生一部分相似流量, 这部分流量很有可能是该 TPL 产生的, 因此 LibHunter 提出了基于 URL 与函数调用栈的双向聚类方法, 对 URL 与函数调用栈进行聚类, 并基于投票方式将 URL 与调用栈进行关联, 从而得到了函数调用栈-URL 的对应关系, 并进一步从调用栈中提取出 TPL 的包名。

基于 LibHunter 的分析结果, LibCapture 从明文流量函数调用栈中依据 TPL 包名筛选出所有 TPL 请求, 筛选的逻辑是如果一条请求所对应的函数调用栈中出现了 TPL 的包名, 则认为该请求是由 TPL 发出的, 从而标记该请求是 TPL 请求, 否则认为是宿主 app 请求。进一步地, 在标记明文流量请求的基础

之上, LibCapture 针对每一条请求提取了 6 元组信息——源地址、源端口、目的地址、目的端口、app 包名以及请求时间戳。将 6 元组相同的明文与密文流量一一对应, 并根据明文流量的 TPL 标签信息对密文流量进行标记, 从而生成 TPL 加密流量数据集。

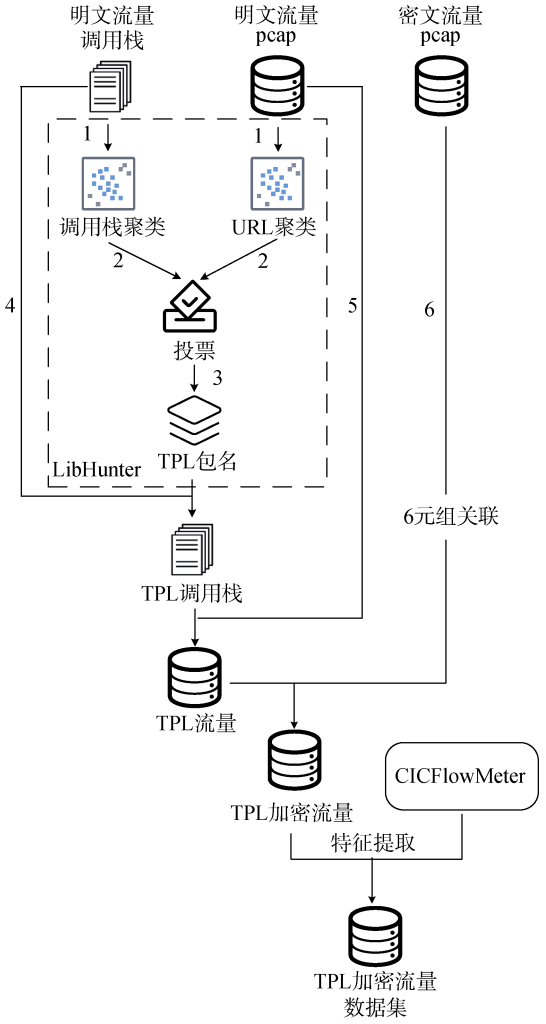

图 4　数据集生成模块示意图

Figure 4　Dataset construction

此外, LibCapture 使用 CICFlowMeter[37]从 pcap 文件中提取 TPL 流量特征, 包含数据包长度、TCP 标志位等 79 维特征, 具体特征信息请见我们的 Github 仓库 https://github.com/BlcDing/LibCapture。

3.4　基于联邦 CNN 的 TPL 流量识别模型

本节介绍基于联邦学习的 TPL 流量识别模型。联邦学习中参与方的本地模型采用卷积神经网络(Convolutional Neural Network, CNN)。针对数据集生成模块中所提取的流量特征, 我们设计了一个一维 CNN 模型, 其包括 2 个卷积层、2 个池化层和 1 个全连接层, 模型结构如图 5 所示。

其中输入数据为 79x1 的特征矩阵, 一次输入 32 条样本数据, 卷积层输入通道为 1, 输出通道为 16, 卷积核大小为 5×5, 步长为 1, 填充为 0, 池化层池化核大小为 3×3, 步长为 3, 填充为 0, 使用 Tanh 和 ReLU 两种线性激活函数, 全连接层输入尺度为 32×7, 输出尺度为 128, 使用 ReLU 激活函数, 在训练过程的前向传播中, 每个神经元以 50%的概率处于不激活的状态, 另外使用梯度下降优化算法和交叉熵损失函数, 学习率设置为 1e-2。模型详细情况如表 1 所示。

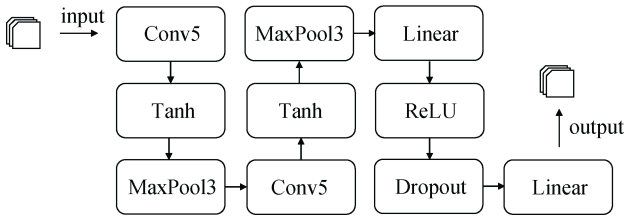

图 5　一维 CNN 模型结构

Figure 5　1D-CNN model structure

表 1　一维 CNN 模型结构详述

Table 1　1D-CNN model structure details

Layer(type)	Output Shape	Param
Conv1d-1	[–1, 16, 75]	96
Tanh-2	[–1, 16, 75]	0
MaxPool1d-3	[–1, 16, 25]	0
Conv1d-4	[–1, 32, 21]	2592
Tanh-5	[–1, 32, 21]	0
MaxPool1d-6	[–1, 32, 7]	0
Linear-7	[–1, 128]	28800
ReLU-8	[–1, 128]	0
Dropout-9	[–1, 128]	0
Linear-10	[–1, 5]	645
Total params: 32, 133		
Trainable params: 32, 133		
Non-trainable params: 0		

CNN 模型的参数主要包含每一层的权重(weight)和偏置(bias), LibCapture 将每个参与方每一层的权重和偏置上传至联邦学习服务端, 服务端使用 FedAvg[39]算法进行全局参数更新, 从而产生新的全局参数, 并将新参数下发至客户端。具体流程如图 6 所示。

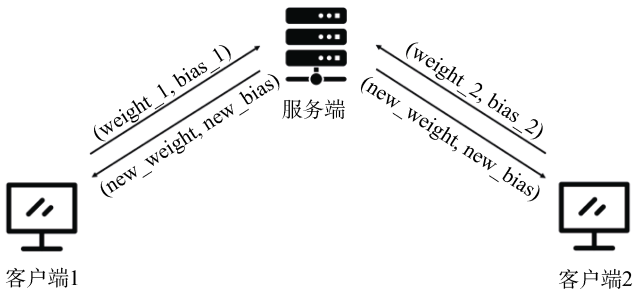

图 6　联邦学习流程图

Figure 6　Workflow of federated learning

4 实验结果与分析

4.1 系统实现与实验设置

4.1.1 系统实现

LibCapture 由流量采集、数据集生成以及联邦 CNN 模型三部分组成。流量采集模块基于 Python requests 库以及 bs4 库实现, 动态插桩能力基于 Frida 实现, 加密流量采集基于 tcpdump 实现, 采集控制器基于 Python 实现; 数据集生成模块基于 LibHunter 二次开发实现; 联邦 CNN 模型基于 Flower 和 PyTorch 实现。

4.1.2 实验环境与设置

本文于 2021 年 12 月根据 360 应用市场下载量排行榜爬取了 2327 个 app。对于每一个 app, LibCapture 将其安装到测试终端, 利用 Android Monkey 工具模拟用户点击、滑动、输入等事件与 app 进行交互, 并利用流量采集控制器协调 ETC 服务和 UTC 服务同时进行流量捕获。每个 app 运行 10 分钟, 测试时间结束后, LibCapture 将关闭并卸载 app, 同时将采集到的明文流量与密文流量传输至服务端, 进行 TPL 检测与流量标记, 并供联邦 CNN 模型训练与识别。实验环境由两台 Pixel3 手机和两台服务器组成。其中 Pixel3 手机用于运行、测试 app 并进行流量采集; 1 台服务器用于 TPL 检测与 TPL 流量标记, 从而生成 TPL 流量数据集; 1 台服务器用于运行联邦 CNN 模型服务端。

4.1.3 数据集

LibCapture 共捕获 962MB 明文 pcap 文件、3.3GB 加密 pcap 文件以及 569MB 调用栈文件。由于明文 pcap 文件是基于动态插桩技术采集的流量, 该技术是针对特定 app 进程的抓包技术, 只会捕获由该进程产生的流量; 而 tcpdump 是基于网卡的流量采集技术, 其会捕获经过设备网卡的所有数据包, 即包含了测试 app 以及所有系统背景流量的数据。因此, 在我们捕获的流量中加密 pcap 文件要多于明文 pcap 文件。我们使用 LibHunter 来检测明文 pcap 文件中包含的 TPL, 发现其中包含 38 个 TPL, TPL 概况如表 2 所示。

我们将第一次收集的数据集记为原始数据集 Ⅰ, 其中包含 38 个 TPL, 由于数据主要集中于“com.qq.e.comm”、“com.bytedance.sdk”等 5 类 TPL, 其余 TPL 样本量过少, 因此我们针对这 5 类 TPL 进行了第二次收集, 记为原始数据集 Ⅱ。

根据实验所需, 分别对原始数据集 Ⅰ 和 Ⅱ 做以下处理, 形成的数据集细节如表 3 所示:

1) 将原始数据集 Ⅰ 记为数据集 a, 均分为 3 份, 分别记为 a1、a2、a3;

表 2 TPL 概况

Table 2 Overview of TPL

TPL				
com.qq.e.comm	com.baidu.speech	com.oppo.upgrade	cn.jpush.android	com.mopub.volley
com.bytedance.sdk	com.alibaba.sdk	com.flurry.sdk	com.alipay.sdk	com.iflytek.cloud
com.amap.api	com.tencent.bugly	com.anythink.core	com.weibo.ssosdk	com.sensorsdata.analytics
com.kwad.sdk	com.google.analytics	com.tapjoy.TapjoyURLConnection	com.huawei.hms	com.adjust.sdk
com.baidu.mobads	com.kuaishou.weapon	cn.magicwindow.common	com.evernote.edam	com.google.firebase
com.umeng.commonsdk	com.qihoo.sdk	com.uc.sdk	com.google.ads	com.igexin.push
com.uc.crashsdk	com.tencent.tbs	com.baidu.push	cn.jiguang.jmlinksdk	com.pgl.sys
com.baidu.mobstat	com.appsflyer.AppsFlyerLib	com.sigmob.volley		

表 3 数据集说明

Table 3 Description of the dataset

数据集	划分方法	划分数量	子数据集	TPL 数量	说明
数据集 a	随机划分	3	a1、a2、a3	38	原始数据集 I
数据集 b	随机划分	3	b1、b2、b3	5	样本数量不平衡, 类别数量平衡
数据集 c	随机划分	3	c1、c2、c3	5	样本数量平衡, 类别数量平衡
数据集 d	按类别划分	2	d1、d2	5	样本数量不平衡, 类别数量不平衡
数据集 e	按类别划分	2	e1、e2	5	样本数量平衡, 类别数量不平衡
数据集 f	随机划分	2	f1、f2	5	样本数量平衡, 类别数量平衡
数据集 g	按训练集和测试集的比例为 7∶3 划分	2	g1、g2	5	样本数量平衡, 类别数量平衡

2) 原始数据集Ⅱ中有 5 类 TPL 加密流量样本, 通过随机抽取使这 5 类的样本数量不平衡, 记为数据集 b, 再将其均分为 3 份, 分别记为 b1、b2、b3;

3) 从原始数据集Ⅱ中随机选取数据, 使每个类别的数量接近 1∶1∶1∶1∶1, 形成样本数量平衡的数据集 c, 再将其均分为 3 份, 分别记为 c1、c2、c3;

4) 将数据集 b 按类别分为 2 份, 分别记为 d1、d2, 其中数据集 d1 包含 3 个类别, 数据集 d2 包含 2 个类别, 将数据集 d1、d2 的总和记为 d;

5) 将数据集 c 按类别分为 2 份, 分别记为 e1、e2, 其中数据集 e1 包含 3 个类别, 数据集 e2 包含 2 个类别, 将数据集 e1、e2 的总和记为 e;

表 4 实验细节说明

Table 4 Description of experimental details

编号	实验方法	对应章节	训练集	测试集	类别数	Accuracy	Precision	Recall	F1-Score
1	CNN	4.3.1	a1	a3	38	60.52%	3.00%	8.57%	4.44%
2			b1	b3	5	69.29%	55.15%	65.36%	59.82%
3			c1	c3	5	72.73%	71.40%	71.39%	71.40%
4		4.3.2	d1	d	5	36.25%	48.04%	25.85%	33.61%
5			e1	e	5	47.65%	48.98%	28.67%	36.17%
6		4.3.3	f1	f	5	83.05%	81.62%	82.49%	82.06%
7	联邦 CNN	4.3.1	a1、a2	a3	38	62.01%	26.00%	20.91%	23.18%
8			b1、b2	b3	5	70.09%	57.09%	63.62%	60.18%
9		4.3.2	c1、c2	c3	5	80.53%	78.88%	79.00%	78.94%
10			d1、d2	d	5	54.06%	42.54%	46.14%	44.27%
11		4.3.2 & 4.3.4	e1、e2	e	5	60.01%	59.56%	60.12%	59.84%
12		4.3.3 & 4.3.4	f1、f2	f	5	92.34%	91.28%	91.66%	91.47%
13	AppScanner	4.3.4	e1	e	5	57.49%	78.59%	60.25%	46.05%
14			f1	f	5	95.97%	95.65%	95.68%	95.65%
15			g1	g	5	95.04%	94.63%	94.74%	94.57%

6) 将数据集 c 均分为 2 份, 分别记为 f1、f2, 将数据集 f1、f2 的总和记为 f;

7) 取数据集 c 中的 70%作为训练集, 记为 g1, 另外 30%作为测试集, 记为 g2, 将数据集 g1、g2 的总和记为 g。

4.2 分类模型评价指标

本文从四个方面: 准确率(accuracy)、精确率(precision)、召回率(recall)和 f1 值(f1-score)对提出的模型进行评估。假设某个样本类别为“1”, 我们规定类别“1”的样本为正样本, 其他标签样本为负样本。设 TP(True Positive)为将标签为“1”的数据判断为“1”的样本数, TN(True Negative)为将其他标签的数据判断为对应标签的样本数, FP(False Positive)为将其他标签的数据判断为“1”的样本数, FN(False Negative)为将标签为“1”的数据判断为其他标签的样本数。因此四个指标的标准计算公式如下所示:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$f1 = 2 * \frac{precision * recall}{precision + recall}$$

4.3 对比实验和分析

4.3.1 样本数量不平衡对模型性能的影响

真实世界采集的各类 TPL 加密流量普遍存在样本数量不平衡的情况, 为了模拟真实世界情况, 本文分别使用数据集 a、b、c, 对比了在联邦学习与非联邦学习的情况下, 样本数量不平衡与否对实验的影响。

参与对比的实验为实验 1-3 和实验 7-9, 其中实验 1-3 基于经典 CNN 模型, 实验 7-9 基于联邦 CNN 模型, 分别使用数据集 a、b、c。

实验 1 和实验 7 使用了数据集 a, 分别训练 500 轮, 两次实验的结果较差, CNN 模型准确率为 60.52%, 精确率为 3.00%, 召回率只有 8.57%, 联邦 CNN 准确率为 62.01%, 精确率为 26.00%, 召回率只有 20.91%。对应的混淆矩阵和 ROC 曲线图如图 7-8 所示。

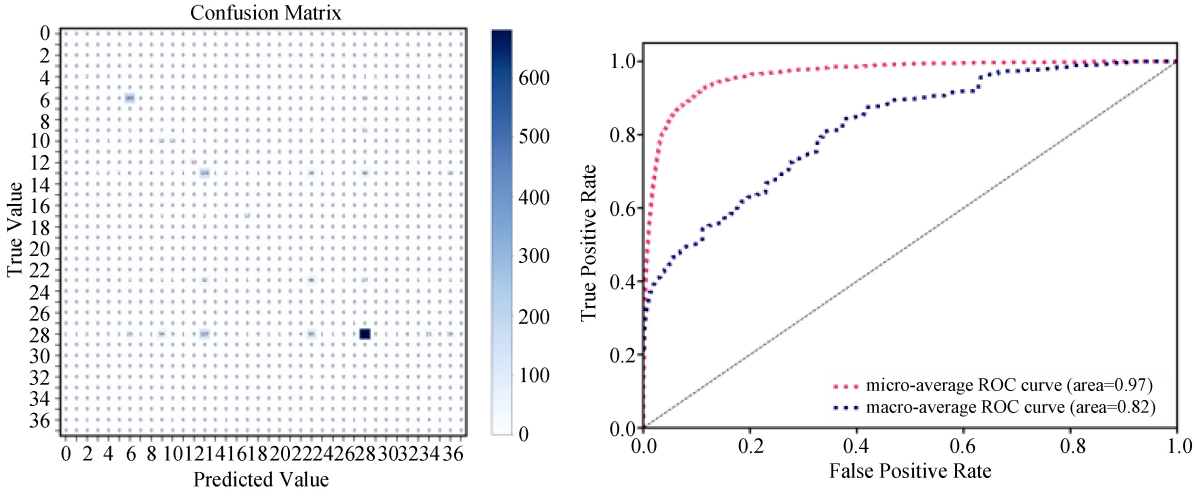

图 7　实验 1 混淆矩阵和 ROC 曲线

Figure 7　Confusion matrix & ROC of Experiment 1

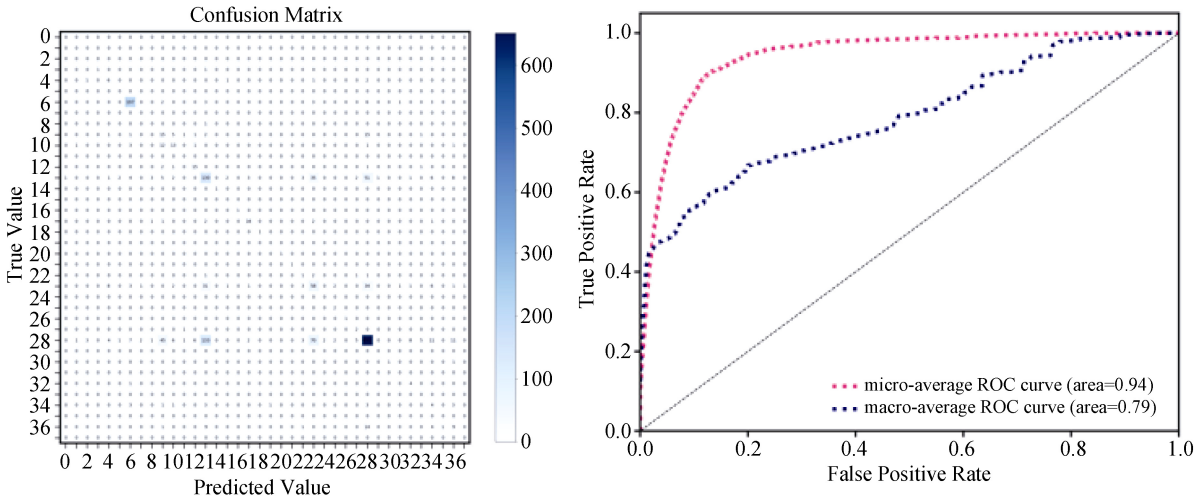

图 8　实验 7 混淆矩阵和 ROC 曲线

Figure 8　Confusion matrix & ROC of Experiment 7

实验 1 和实验 7 使用的数据集 a 主要集中于“com.qq.e.comm”、“com.bytedance.sdk”等 5 类 TPL, 其余 TPL 样本量过少, 因此我们针对在 app 中大量集成的 5 类 TPL 进行了第二次数据采集, 并在新的数据集上额外进行了实验。

实验 2 与实验 8 使用了样本数量不平衡的数据集, 分别训练 500 轮, 实验 3 与实验 9 使用了样本数量平衡的数据集, 分别训练 500 轮, 对应的混淆矩阵和 ROC 曲线图如图 9-12 所示。根据表 4 的实验结果可以看出, 无论样本数量平衡与否, 联邦 CNN 的各项评估指标均高于单独 CNN 模型。当样本数量平衡时, 单独 CNN 模型的准确率相较于样本数量不平衡的情况提升了大约 3%, 联邦 CNN 的准确率提升了大约 10%。因此可以得到, 样本数量是否平衡对于联邦 CNN 的学习有较大影响, 使用样本数量不平衡的数据集, 会导致训练模型侧重样本数目较多的类别, 因此模型在测试数据上的泛化能力就会受到影响。当样本数量平衡时, CNN 模型达到了 72.73%的准确率, 联邦 CNN 模型可以达到 80.53%的准确率。

尽管联邦 CNN 在样本数量平衡的情况下, 预测准确率相较于样本数量不平衡时能够获得更大的提升, 但是在现实环境中, 各类别样本数量不平衡的情况广泛存在。因此在这种情况下, 研究人员应尝试其他方法, 例如更广泛地收集数据或增加特征数量等。

4.3.2　类别数量不平衡对模型性能的影响

在不同应用场景下, 可能存在客户端参与训练的数据集包含不同类别样本的情况, 为了探究类别数量不平衡对模型性能的影响, 我们分别使用数据集 d、e, 对比了在联邦学习与非联邦学习的情况下, 类别数量不平衡对实验的影响。

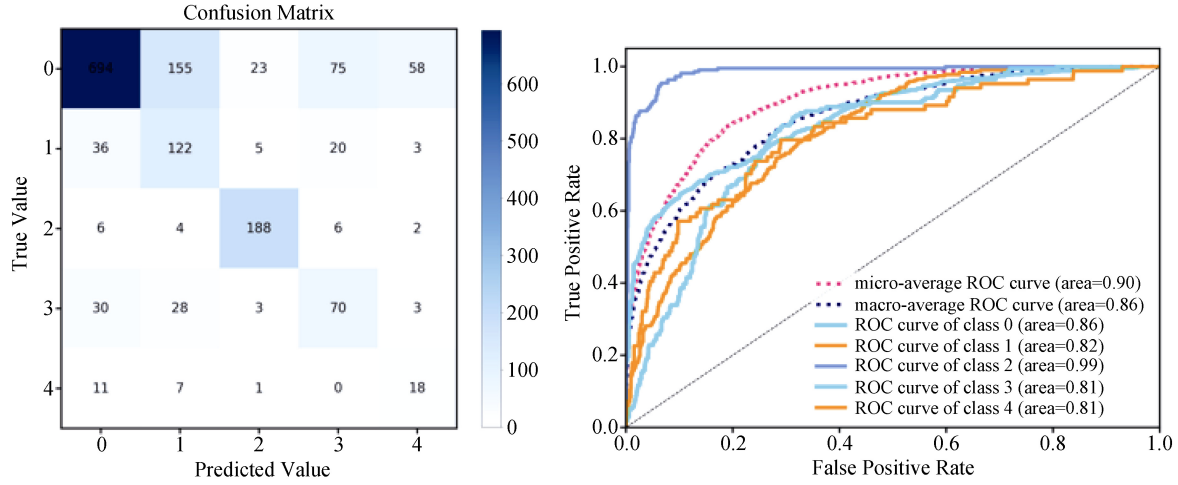

图 9　实验 2 混淆矩阵和 ROC 曲线

Figure 9　Confusion matrix & ROC of Experiment 2

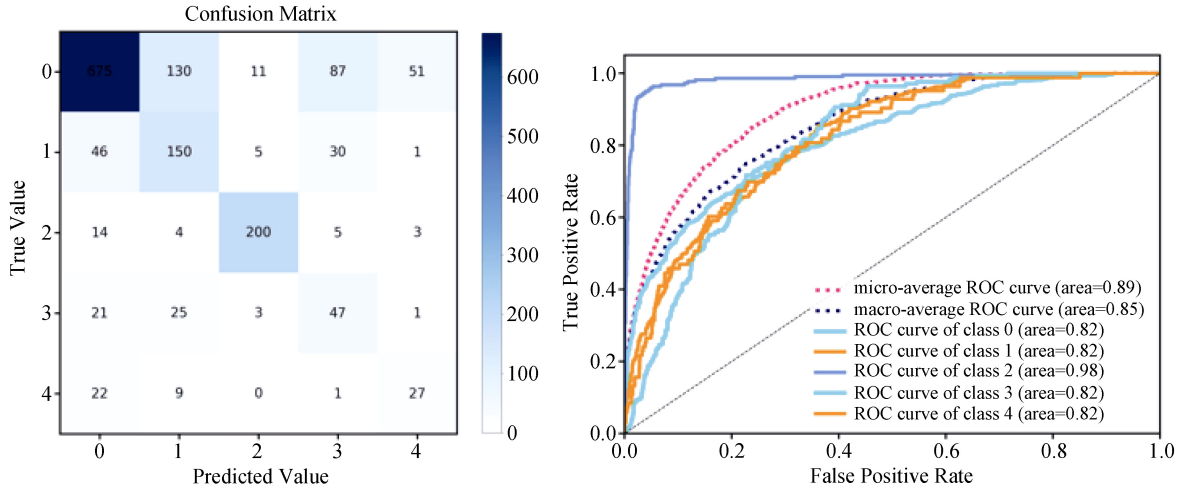

图 10　实验 8 混淆矩阵和 ROC 曲线

Figure 10　Confusion matrix & ROC of Experiment 8

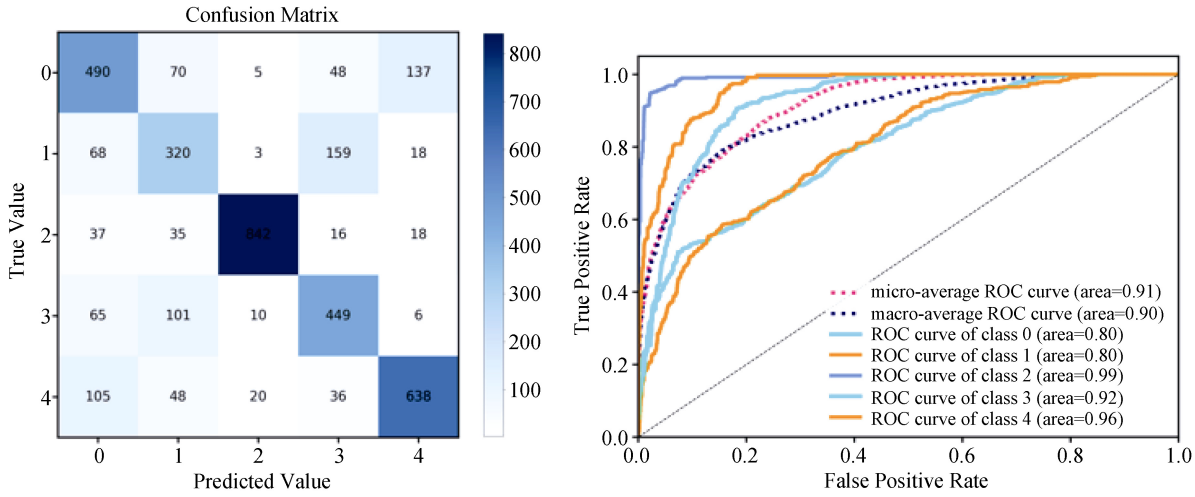

图 11　实验 3 混淆矩阵和 ROC 曲线

Figure 11　Confusion matrix & ROC of Experiment 3

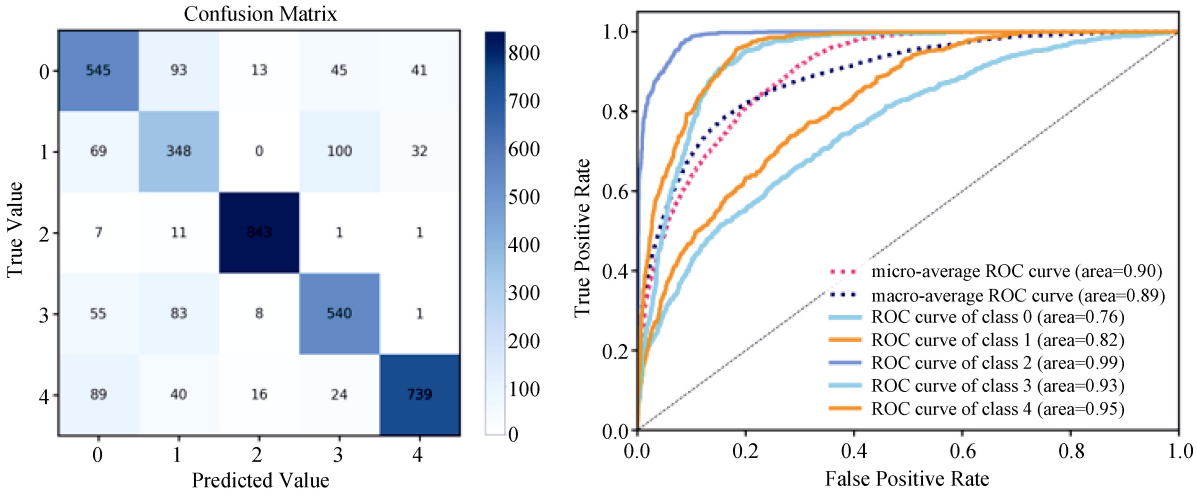

图 12　实验 9 混淆矩阵和 ROC 曲线

Figure 12　Confusion matrix & ROC of Experiment 9

参与对比的实验为实验 4-5 和实验 10-11, 其中实验 4 与实验 5 基于经典 CNN 模型, 分别训练 500 轮, 实验 10 与实验 11 基于联邦 CNN 模型, 分别训练 500 轮, 两组实验分别使用数据集 d、e, 实验细节与实验结果如表 4 所示。

实验 4 与实验 10 使用了样本数量不平衡、类别数量不平衡的数据集, 实验 5 与实验 11 使用了样本数量平衡、类别数量不平衡的数据集, 对应的混淆矩阵和 ROC 曲线图如图 13-16 所示。根据表 4 的实验结果可以看出, 经典 CNN 模型的准确率最高仅为 47.65%, 联邦 CNN 的准确率最高仅为 60.01%, 在类别数量不平衡的情况下, 尽管联邦模型的性能明显优于不联邦的模型, 但无论样本数量是否平衡, 两种方法均不能取得较好的预测效果。因此, 类别数量不平衡对于模型的性能影响较大。

我们分析认为, 当数据集样本类别不平衡时, 会导致训练模型只能学习到训练集已知的类别, 尽管联邦学习可以通过客户端与服务端之间参数的传递, 学习到其他客户端的参数, 但是由于本文设计的联邦学习模型在服务端使用 FedAvg 算法更新参数, 而 FedAvg 算法本身并未根据客户端样本类别的差异性对来自不同客户端的参数进行针对性处理, 最终联邦 CNN 取得的准确率最高仅为 60.01%。由实验可见, 针对客户端样本类别不平衡的情况, 联邦学习模型服务端的参数聚合算法需要进行更加有针对性的设计, 从而将客户端数据集的差异性进行合理融合。

4.3.3　样本及类别数量平衡时模型性能评估

由 4.3.1 节和 4.3.2 节的实验结果表明, 当样本数量平衡或类别数量平衡时, 能够得到较好的训练效果, 因此本文使用数据集 f 在样本数量及类别数量均平衡时, 对比了联邦 CNN 模型与经典 CNN 模型的效果。

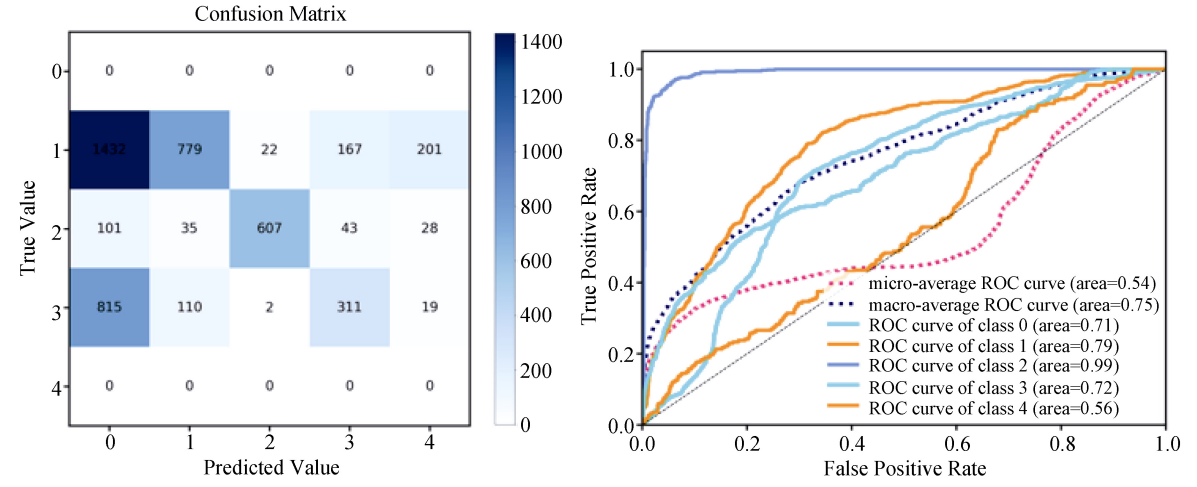

图 13　实验 4 混淆矩阵和 ROC 曲线

Figure 13　Confusion matrix & ROC of Experiment 4

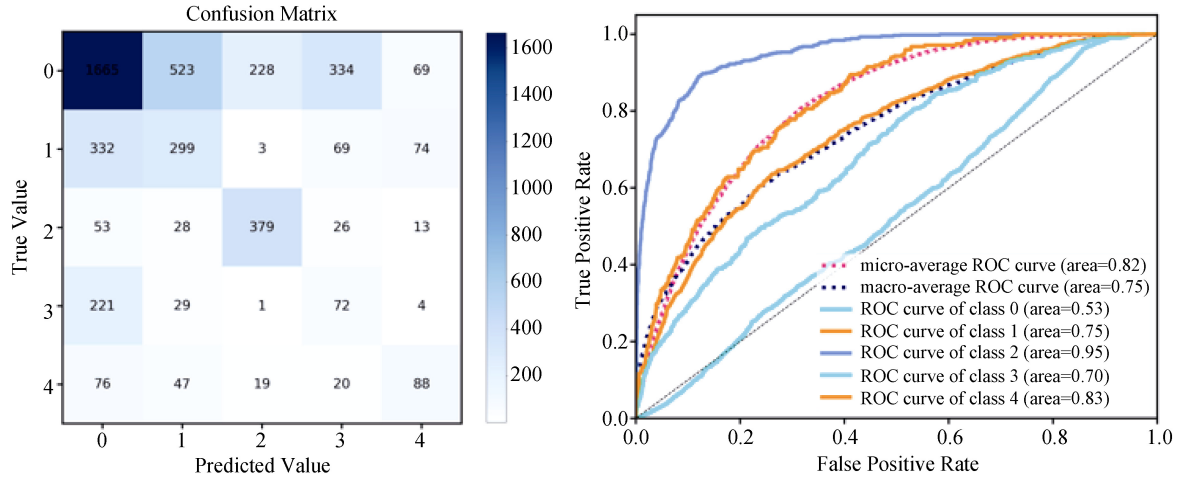

图 14　实验 10 混淆矩阵和 ROC 曲线

Figure 14　Confusion matrix & ROC of Experiment 10

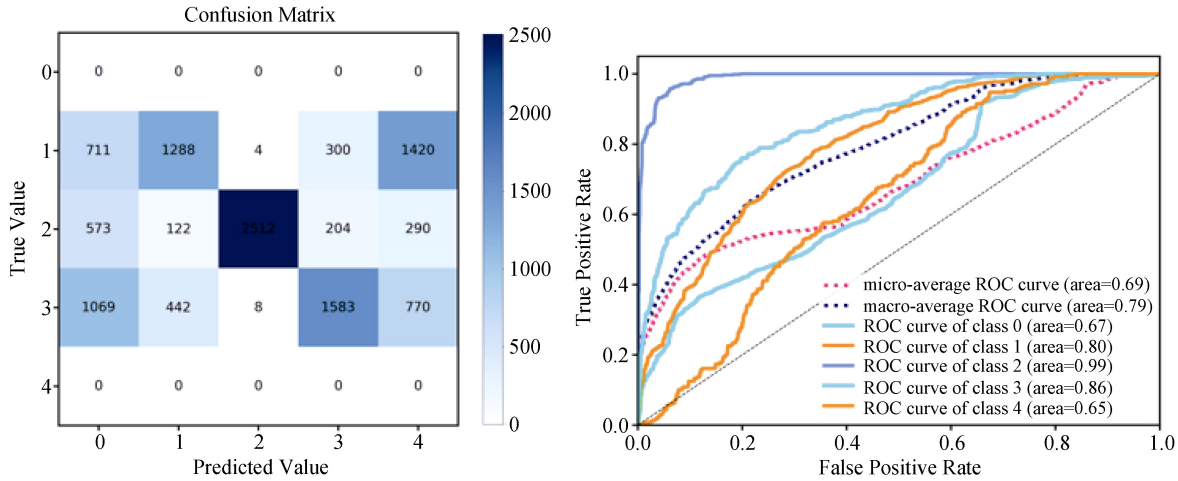

图 15　实验 5 混淆矩阵和 ROC 曲线

Figure 15　Confusion matrix & ROC of Experiment 5

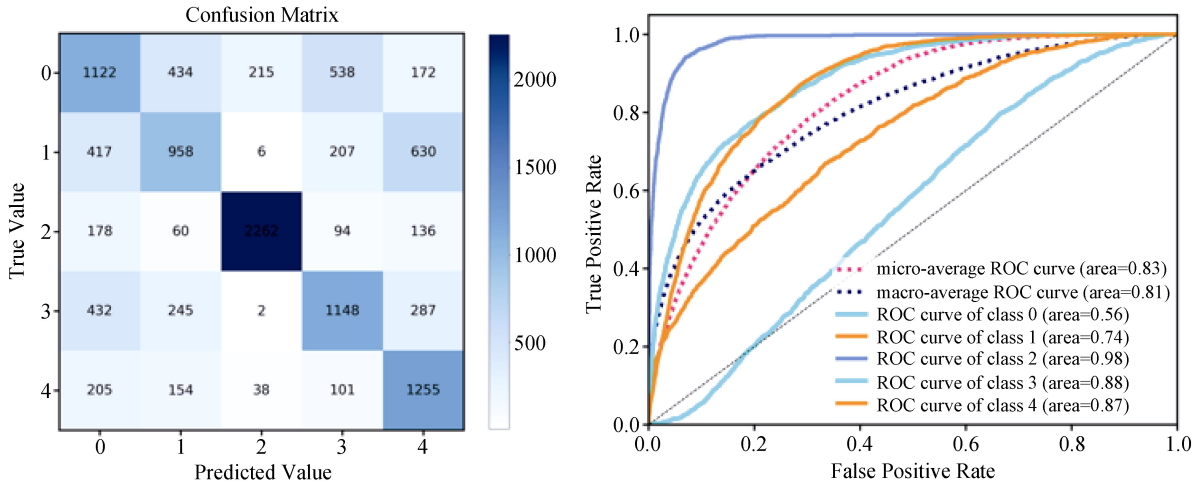

图 16　实验 11 混淆矩阵和 ROC 曲线

Figure 16　Confusion matrix & ROC of Experiment 11

参与对比的实验为实验 6 和实验 12，其中实验 6 基于 CNN 模型训练 500 轮，实验 12 基于联邦 CNN 训练 500 轮，分别使用数据集 f，实验细节与实验结果如表 4 所示。

实验 6 和实验 12 对应的混淆矩阵和 ROC 曲线图如图 17-18 所示。根据表 4 的实验结果可以看出, 经典 CNN 模型和联邦 CNN 都取得了最好的实验效果, 预测准确率分别为 83.05%和 92.34%。实验 6、12 与实验 3、9 的区别在于, 数据集 c 将数据集分为 3 份, 而数据集 f 将数据集分为 2 份, 因此实验 6、12 使用的训练集数据量更大, 测试集则包含了所有数据。从实验结果来看, 在样本数量和类别数量均平衡的情况下, 当参与实验的数据量更大的时候, 能够获得更好的效果。

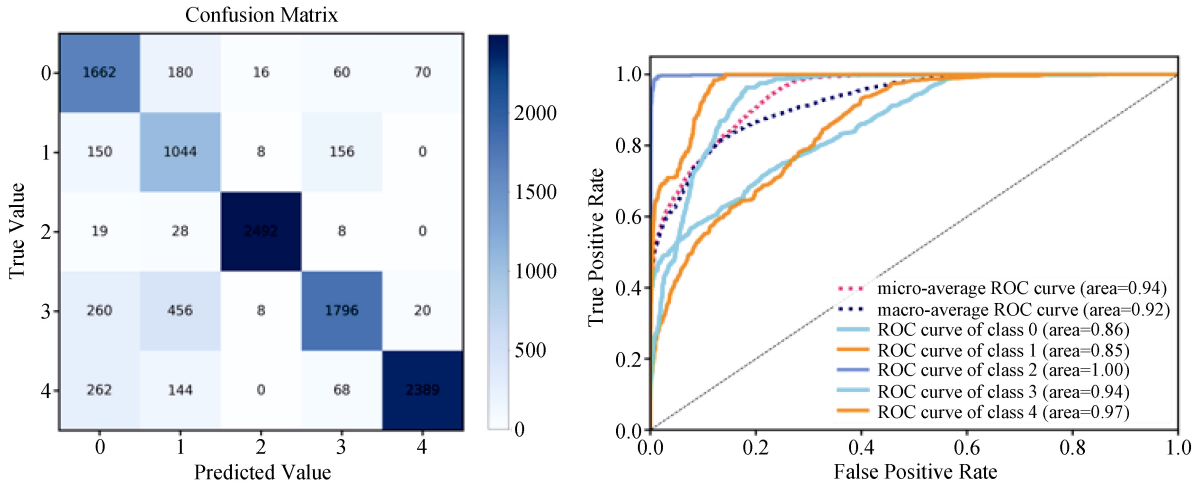

图 17　实验 6 混淆矩阵和 ROC 曲线

Figure17　Confusion matrix & ROC of Experiment 6

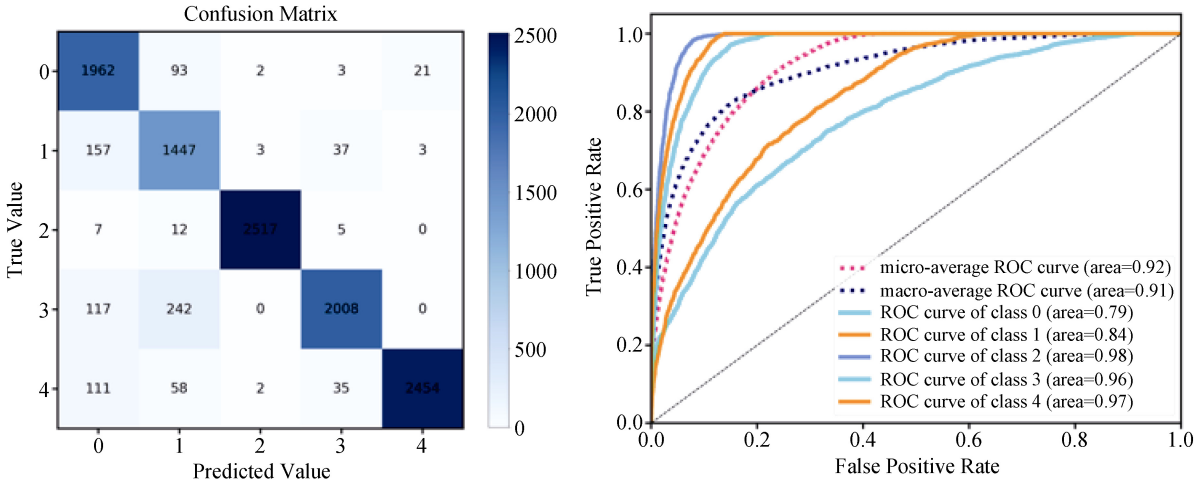

图 18　实验 12 混淆矩阵和 ROC 曲线

Figure 18　Confusion matrix & ROC of Experiment 12

4.3.4　与已有工作对比

AppScanner[12]是 app 流量识别研究的经典方法, 其能够从加密网络流量中对 app 流量进行自动指纹提取与识别。从本质上来说, AppScanner 使用随机森林分类器对流量进行分类, 其使用的随机森林分类器的主要参数设置如表 5 所示:

为了测试在类别数量不平衡的情况下联邦 CNN 与 AppScanner 的效果, 我们分别使用了样本数量平衡但类别数量不平衡的数据集 e 和样本数量与类别数量均平衡的数据集 f。作为对比, 我们按照常见划分训练集和测试集的比例, 即 7∶3 的划分方式处理数据集 g, 从而探索 AppScanner 的识别效果。

表 5　随机森林各项参数及数值

Table 5　Parameters of Random Forest

参数	数值
criterion	gini
max_features	sqrt
n_estimators	150

参与对比的实验为实验 11-15, 其中实验 11-12

使用联邦 CNN 模型训练 500 轮，使用数据集 e、f 进行训练，实验 13-15 使用 AppScanner 模型，使用数据集 e、f、g 进行训练，各项结果取平均值，实验参数设置与实验结果如表 4 所示。

实验 11 与 13 使用了样本数量平衡、类别数量不平衡的数据集，实验 12、14、15 使用了样本数量和类别数量均平衡的数据集，其中实验 11-14 使用的数据集均分为 2 份，实验 15 使用的数据集将训练集和测试集按 7∶3 的比例分为 2 份。对应的混淆矩阵和 ROC 曲线图如图 16、18-21 所示。根据表 4 的实验结果可以看出，在类别数量不平衡的情况下，联邦 CNN 的预测准确率为 60.01%，AppScanner 的预测准确率为 57.49%，均不能取得较好的预测效果；当样本数量和类别数量均平衡的情况下，联邦 CNN 能够取得 92.34%的准确率，AppScanner 能够取得 95.97%的准确率；此外，使用 7∶3 的比例划分数据集对 AppScanner 进行测试，取得的准确率为 95.04%。也就是说，本文提出的基于联邦学习的 TPL 加密流量识别模型能够获得与 AppScanner 的预测准确率接近的效果。需要注意的是，AppScanner 取得略高于联邦 CNN 模型的准确率的前提是，其获取了与联邦 CNN 模型相同的样本类别。但真实生产环境中，联邦 CNN 模型由于其天然的分布式特性，很容易容纳更多的参与方共同参与训练，从而超越 AppScanner 的准确率。此外，将所有用户数据上传至云端集中训练存在严重的隐私数据保护问题，联邦 CNN 模型在用户隐私保护方面的优势远超 AppScanner。

经过上述分析可见，在样本类别不平衡时，联邦 CNN 模型取得了优于 AppScanner 的模型准确率。

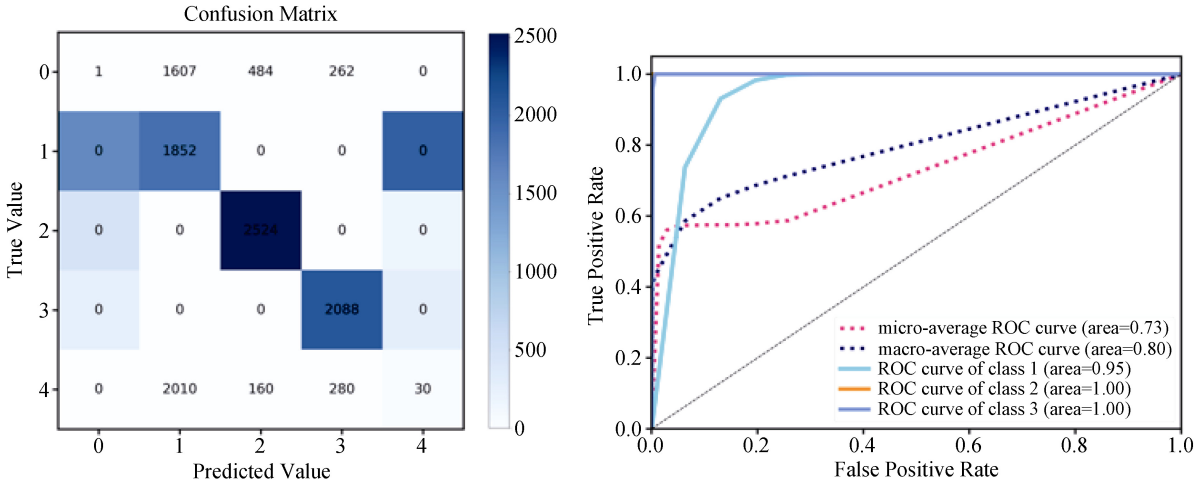

图 19　实验 13 混淆矩阵和 ROC 曲线

Figure 19　Confusion matrix & ROC of Experiment 13

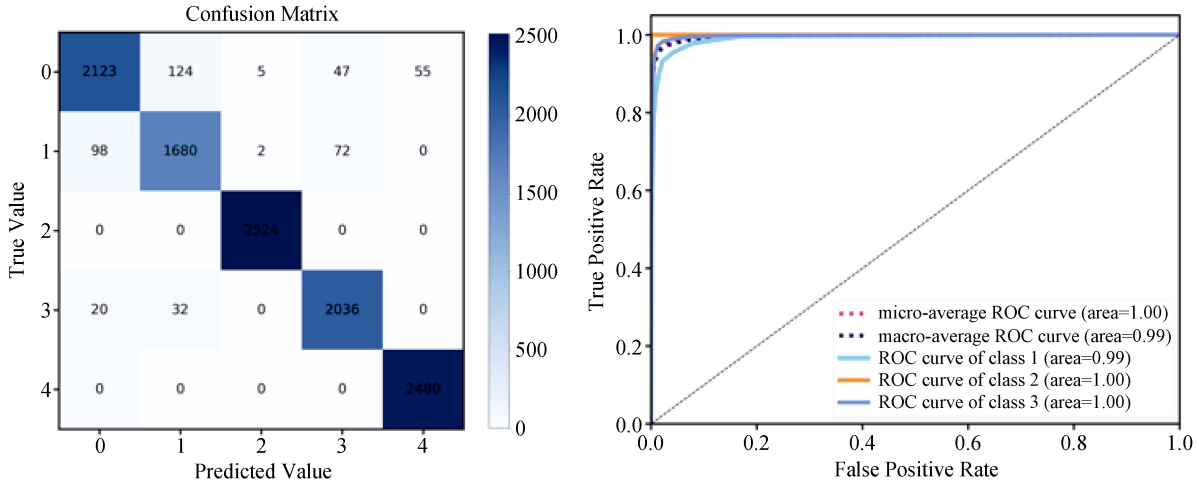

图 20　实验 14 混淆矩阵和 ROC 曲线

Figure 20　Confusion matrix & ROC of Experiment 14

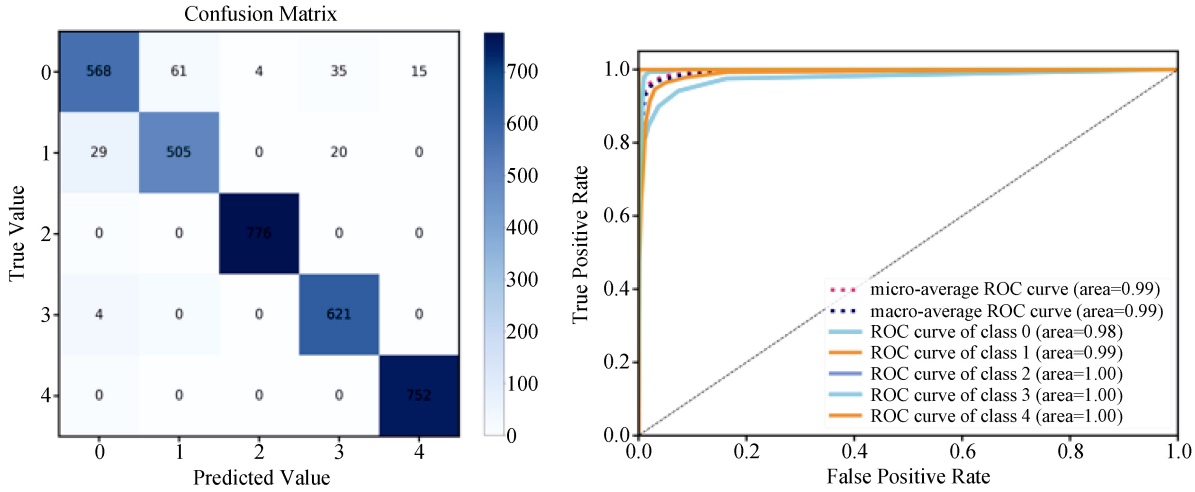

图 21　实验 15 混淆矩阵和 ROC 曲线

Figure 21　Confusion matrix & ROC of Experiment 15

即使当我们假设 AppScanner 能够获取所有联邦参与方本地的样本类别时，联邦 CNN 模型也取得仅比 AppScanner 低 3%左右的识别准确率。但由于联邦 CNN 模型具有用户隐私保护、分布式训练与更新等优势，我们认为联邦 CNN 模型更加适用于移动应用场景下的 TPL 加密流量识别问题。

值得一提的是，由于 AppScanner 使用了随机森林分类器，而随机森林在分类任务中能够获得较高的准确率，因此在未来的工作中，可以探索随机森林或其它机器学习模型结合联邦学习进行 TPL 加密流量识别的可行性。

综上，本文提出了一种基于联邦 CNN 的 TPL 加密流量识别模型，实现了 TPL 流量分类模型的分布式训练与模型更新，通过大量对比实验证明了本文所提模型对于 TPL 加密流量能够取得较高的识别率。进一步地，通过与现有研究工作 AppScanner 进行对比，验证了即使在数据集相同的情况下，联邦 CNN 仅比不联邦的 AppScanner 准确率低 3%左右，但却在隐私保护与分布式训练方面有明显优势。此外，本文分析了联邦学习不同参与方本地数据集差异性对全局模型聚合带来的影响，针对不同场景分别指出了可能的解决方法和未来的研究方向。

5　讨论

5.1　联邦学习的模型选择

随着人工智能技术的迅速发展，新的人工智能模型不断涌现，各类模型优化方法也层出不穷。本文选择经典人工智能模型 CNN 作为联邦学习的客户端模型，且在未对 CNN 模型的结构及参数等过多深入调优的情况下，取得了 92.34%的准确率，证明了联邦学习在 TPL 加密流量识别领域的有效性。由于联邦学习天然的隐私保护与协同训练能力，十分适合于移动应用中分布式模型的场景，我们认为在 TPL 加密流量识别领域联邦学习是一种合理且可行的技术路线。

另一方面，虽然本文选择了经典 CNN 模型作为联邦学习的客户端模型，但这并不代表其它模型不适用于本场景。由于本文主要探讨联邦学习技术在 TPL 加密流量识别领域的可行性以及联邦客户端数据分布对模型的影响和启示，设计新的人工智能模型并不是本文的主要研究目标。我们相信未来会出现其它效果更加理想的人工智能模型，或者经过更大数据量训练以及调优之后能够取得更好识别效果的人工智能模型。

5.2　联邦学习参与方的选择

在现实生产环境中，参与联邦学习的各客户端除了本地样本数据可能存在差异外，在计算存储能力、网络通信能力、电池功耗等方面均存在天然差异。本文研究了客户端本地数据差异性对全局模型的性能影响，需要注意的是，客户端中上述其它因素依然会对全局模型聚合产生影响，因此在基于联邦学习的 TPL 加密流量识别模型中，需结合客户端的多种能力因素进行实时客户端选择，让更适合参与协同训练的客户端上传训练参数至服务端，让不适合参与训练的客户端接收更新后的全局模型参数，从而全面提升模型性能。

5.3　流量采集方法

本文中流量采集方法基于动态插桩技术实现，其基本思想是对安卓操作系统 Framework 层中与 SSL 相关的系统默认 API 进行插桩，插入流量捕获代

码, 在流量数据被加密之前将数据进行提取和保存, 从而实现加密流量的明文采集。该方法可对所有调用了 3.2 节中提到的 API 的 app 进行流量捕获, 但如果 app 不调用系统默认 API 而是采用自定义 so 库来发送和接收 HTTPS 流量, 则本文所提方法将会失效。考虑到自定义 so 库的必要性及研发成本, 我们认为大多数 app 均会采用操作系统默认 API 来收发 HTTPS 流量。在实验过程中, 我们尚未发现采用自定义so库绕过本文所提流量采集方法的app, 但并不排除有少量 app 会出于高安全性、高性能等方面因素的考虑, 会实现自定义 so 库来替代安卓系统默认 API。针对自定义 so 库的 app 流量采集, 需要进行针对性分析, 确定正确的插桩点来进行流量捕获。

6 总结与展望

本文指出了移动应用流量分析研究领域中一个被长期忽视的问题——第三方库(Third-party Library, TPL)流量识别。TPL 由于其广泛的被集成性, 一旦出现安全问题将直接影响所有集成该 TPL 的应用, 涉及面广且影响面大。本文首先分析了该项研究的意义与重要性, 并分析了目前该方面研究极少的原因与难点, 即缺乏公开数据集与数据共享困难。基于此, 本文提出了一个新的框架解决上述难点并识别 TPL 加密流量。首先基于动态插桩技术与 TPL 检测技术提出了 TPL 流量自动标注的方法, 从而生成 TPL 加密流量数据集。此外, 利用联邦学习技术结合 CNN 卷积神经网络模型实现了在保护用户隐私的前提下的联邦 CNN 模型来识别 TPL 流量。我们对 2000 多个真实移动应用进行实验, 研究了联邦 CNN 参与方本地数据差异性对全局模型聚合带来的具体影响, 针对不同场景分别分析造成影响的原因并指出了可能的解决方法。此外, 我们将本文所提框架与已有 app 流量识别经典研究工作 AppScanner 进行实验对比, 验证了即使在掌握相同数据集的情况下, 联邦 CNN 仅比 AppScanner 的准确率低 3%左右, 但却具有隐私保护与分布式训练等优势, 更加适合移动应用流量分析问题, 进一步证实了所提框架的有效性。

最后, 我们提出了一些该领域值得进一步挖掘的研究点:

1) 新的联邦学习模型。由本文实验可知, 在获知与联邦 CNN 模型相同的样本类别的前提下, 以 AppScanner 为代表的经典机器学习方法随机森林取得了准确率优于联邦 CNN 约 3%的效果, 但由于联邦学习的多方参与和分布式训练特性, 使得联邦学习的模型容易获取更多样本与更多类别。因此, 本文计划下一步探索联邦随机森林及其他模型在 TPL 加密流量识别领域的应用方法。

2) 联邦学习模型的客户端选择算法。联邦学习模型的客户端分散于用户终端, 无法确定用户的使用习惯以及会产生何种数据, 也无法预估用户手机的电量、计算能力、存储能力与网络通信能力等。因此在进行联邦模型训练与参数更新时, 联邦服务端需能够智能选择参与训练与上传参数的部分客户端, 避免选择数据质量差、计算性能弱的客户端, 从而获得高效准确的模型效果。

3) 新的 TPL 流量触发方法。流量分析类研究的基础是采集到更多的真实流量, 目前主流的app流量触发方法是本文采用的 Android Monkey 事件模拟器。此外, 有部分流量触发方案的研究提高了 app 流量触发的代码覆盖率, 例如 SmartDroid[38]等。但此类工作有两类不足, 一是其以app代码覆盖率更高为触发目标, 会带来触发耗时更长等问题, 而 TPL 的流量有其特殊性, 以广告库为例, 此类流量倾向于在 app 启动时以及运行过程中主动出现, 无需深度挖掘代码进行触发; 二是现有的此类研究工作已多年不维护, 难以在现在的 app 中成功运行。因此, 研究针对 TPL 流量的新触发方法有一定意义。

致 谢 本文得到了国家重点研发计划项目(2019YFB1005205)资助。

参考文献

[1] Wang H Y, Guo Y. Understanding Third-Party Libraries in Mobile App Analysis[C]. *2017 IEEE/ACM 39th International Conference on Software Engineering Companion*, 2017: 515-516.

[2] Zhang Y, Dai J R, Zhang X H, et al. Detecting Third-Party Libraries in Android Applications with High Precision and Recall[C]. *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering*, 2018: 141-152.

[3] Ren, J., M. Lindorfer, D.J. Dubois, et al. A longitudinal study of pii leaks across android app versions[C]. *NDSS*, 2018: 139-154.

[4] Demetriou, S., W. Merrill, W. Yang, et al. Free for all! assessing user data exposure to advertising libraries on android[C]. *NDSS*, 2016: 35-49.

[5] Son, S., D. Kim, and V. Shmatikov What Mobile Ads Know About Mobile Users[C]. *NDSS*, 2016: 144-158.

[6] Feal, Á., J. Gamba, J. Tapiador, et al. Don't Accept Candy from Strangers: An Analysis of Third-Party Mobile SDKs[J]. *Data Protection and Privacy, Volume 13: Data Protection and Artificial Intelligence*, 2021, 13(1): 1-28.

[7] Wang, J., Y. Xiao, X. Wang, et al. Understanding malicious cross-library data harvesting on android[C]. *30th USENIX Security Symposium*, 2021: 4133-4150.

[8] Ren J J, Rao A, Lindorfer M, et al. ReCon: Revealing and Controlling PII Leaks in Mobile Network Traffic[C]. *The 14th Annual International Conference on Mobile Systems*, *Applications*, *and Services*, 2016: 361-374.

[9] Alan H F, Kaur J. Can Android Applications be Identified Using only TCP/IP Headers of Their Launch Time Traffic? [C]. *The 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016: 61-66.

[10] Mongkolluksamee S, Visoottiviseth V, Fukuda K. Combining Communication Patterns & Traffic Patterns to Enhance Mobile Traffic Identification Performance[J]. *Journal of Information Processing*, 2016, 24(2): 247-254.

[11] Chen Y, You W, Lee Y, et al. Mass Discovery of Android Traffic Imprints through Instantiated Partial Execution[C]. *The 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017: 815-828.

[12] Taylor V F, Spolaor R, Conti M, et al. AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic[C]. *2016 IEEE European Symposium on Security and Privacy*, 2016: 439-454.

[13] Taylor V F, Spolaor R, Conti M, et al. Robust Smartphone App Identification via Encrypted Network Traffic Analysis[J]. *IEEE Transactions on Information Forensics and Security*, 2018, 13(1): 63-78.

[14] Zhan X, Fan L L, Liu T M, et al. Automated Third-Party Library Detection for Android Applications: Are we there Yet? [C]. *The 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020: 919-930.

[15] Liu T M, Wang H Y, Li L, et al. MadDroid: Characterizing and Detecting Devious Ad Contents for Android Apps[C]. *Proceedings of The Web Conference 2020*, 2020: 1715-1726.

[16] Ma Z A, Wang H Y, Guo Y, et al. LibRadar: Fast and Accurate Detection of Third-Party Libraries in Android Apps[C]. *2016 IEEE/ACM 38th International Conference on Software Engineering Companion*, 2017: 653-656.

[17] Derr E, Bugiel S, Fahl S, et al. Keep me Updated: An Empirical Study of Third-Party Library Updatability on Android[C]. *The 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017: 2187-2200.

[18] Backes M, Bugiel S, Derr E. Reliable Third-Party Library Detection in Android and Its Security Applications[C]. *The 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016: 356-367.

[19] Zhang J X, Beresford A R, Kollmann S A. LibID: Reliable Identification of Obfuscated Third-Party Android Libraries[C]. *The 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019: 55-65.

[20] Cui H J, Meng G Z, Li Y Q, et al. LibHunter: An Unsupervised Approach for Third-Party Library Detection without Prior Knowledge[C]. *2022 IEEE Symposium on Computers and Communications*, 2022: 1-7.

[21] Yang Q, Liu Y, Chen T J, et al. Federated Machine Learning: Concept and Applications[J]. *ACM Transactions on Intelligent Systems and Technology*, 2019, 10(2): 1-19.

[22] Yang T, Andrew G, Eichner H, et al. Applied Federated Learning: Improving Google Keyboard Query Suggestions[EB/OL]. 2018: arXiv: 1812.02903. https://arxiv.org/abs/1812.02903

[23] Dai S F, Tongaonkar A, Wang X Y, et al. NetworkProfiler: Towards Automatic Fingerprinting of Android Apps[C]. *2013 Proceedings IEEE INFOCOM*, 2013: 809-817.

[24] He Y Z, Yang X J, Hu B H, et al. Dynamic Privacy Leakage Analysis of Android Third-Party Libraries[J]. *Journal of Information Security and Applications*, 2019, 46: 259-270.

[25] Reardon J, Feal Á, Wijesekera P, et al. 50ways to Leak your Data: An Exploration of Apps' Circumvention of the Android Permissions System[C]. *The 28th USENIX Conference on Security Symposium*, 2019: 603-620.

[26] Telerik. Web Debugging Proxy. https://www.telerik.com/fiddler. Jul. 2022.

[27] Meddle. Web Debugging Proxy Application. https://meddle.mobi/. Jul. 2022.

[28] Narudin F A, Feizollah A, Anuar N B, et al. Evaluation of Machine Learning Classifiers for Mobile Malware Detection[J]. *Soft Computing*, 2016, 20(1): 343-357.

[29] Arora A, Peddoju S K. Minimizing Network Traffic Features for Android Mobile Malware Detection[C]. *The 18th International Conference on Distributed Computing and Networking*, 2017: 1-10.

[30] Crussell J, Stevens R, Chen H. MAdFraud: Investigating Ad Fraud in Android Applications[C]. *The 12th Annual International Conference on Mobile Systems*, *Applications*, *and Services*, 2014: 123-134.

[31] Sun S B, Yu L, Zhang X K, et al. Understanding and Detecting Mobile Ad Fraud through the Lens of Invalid Traffic[C]. *The 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021: 287-303.

[32] Sivan N, Bitton R, Shabtai A. Analysis of Location Data Leakage in the Internet Traffic of Android-Based Mobile Devices[EB/OL]. 2018: arXiv: 1812.04829. https://arxiv.org/abs/1812.04829

[33] Google. Volley is an HTTP library that makes networking for Android apps easier. https://google.github.io/volley/. Jul. 2022.

[34] OpenFeign. Feign is a Java to HTTP client. https://github.com/OpenFeign/feign. Jul. 2022.

[35] square. OkHttp is an HTTP client. https://square.github.io/okhttp/. Jul. 2022.

[36] square. A type-safe HTTP client for Android and Java. https://square.github.io/retrofit/. Jul. 2022.

[37] ahlashkari. CICFlowmeter-V4.0 is an Ethernet traffic Bi-flow generator and analyzer. https://github.com/ahlashkari/CICFlowMeter. Jul. 2022.

[38] boyliang. SmartDroid. https://github.com/boyliang/SmartDroid. Jul. 2022.

[39] McMahan H B, Moore E, Ramage D, et al. Communication-Efficient Learning of Deep Networks from Decentralized Data[EB/OL]. 2016: arXiv: 1602.05629. https://arxiv.org/abs/1602.05629

崔华俊 于 2015 年在南京师范大学计算机技术专业获得硕士学位。现任中国科学院信息工程研究所第三研究室工程师。研究领域为移动应用安全。研究兴趣包括: 移动应用流量分析、协议逆向、软件供应链安全。Email: cuihuajun@iie.ac.cn

孟国柱 于 2017 年在新加坡南洋理工大学计算机科学与工程学院获得博士学位。现任中国科学院信息工程研究所副研究员。研究领域为移动安全、人工智能安全。研究兴趣包括: 安卓恶意软件、AI 安全与隐私。Email: mengguozhu@iie.ac.cn

李玥琦 于 2020 年在电子科技大学软件工程专业获得学士学位, 现于中国科学院大学攻读硕士学位。研究领域为移动应用安全。研究兴趣包括流量分析、零信任技术。Email: liyueqi@iie.ac.cn

张棪 于 2009 年在电子科技大学信息与通信系统专业获得博士学位。现任中国科学院信息工程研究所副研究员。研究领域为移动通信网络安全。研究兴趣包括: 移动应用流量分析、信令安全、网络虚拟化。Email: zhangyan80@iie.ac.cn

代玥玥 于 2019 年在电子科技大学获得博士学位, 现任华中科技大学副教授。主要研究方向包括移动通信网络安全、边缘智能和区块链。担任 IEEE Network 客座编委, Digital Communications and Networks 客座编委。Email: yueyuedai@ieee.org

杨慧然 于 2016 年在电子科技大学信息与通信系统专业获得硕士学位。现任中国科学院信息工程研究所工程师。研究领域为移动通信安全。研究兴趣包括网络流量分析、网络虚拟化。Email: yanghuiran@iie.ac.cn

朱大立 于 2007 年在华中科技大学获得计算机应用技术专业博士学位。现任中国科学院信息工程研究所正研级高级工程师, 博士生导师。研究领域为移动互联网安全和无线网络攻防技术。研究兴趣包括: 智能终端安全、应用安全。Email: zhudali@iie.ac.cn

王伟平 出生于 1975 年, 中国科学院信息工程研究所博士生导师、研究员, 主要研究方向为大数据与人工智能。Email: wangweiping@iie.ac.cn