Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

隧道流量识别研究

硕士研究生 李国浩

2026年02月01日

问题回溯

- 总结反思
  - 声音太小
  - 论文算法原理不够深入
  - TIPO中输入部分不明确

- 相关内容
  - 2025.06.02 李国浩《基于GNN的加密流量方法》
  - 2025.02.10 李国浩《预训练加密流量分类方法》
  - 2023.08.07 巩锟《预训练加密流量表征方法》

内容提要

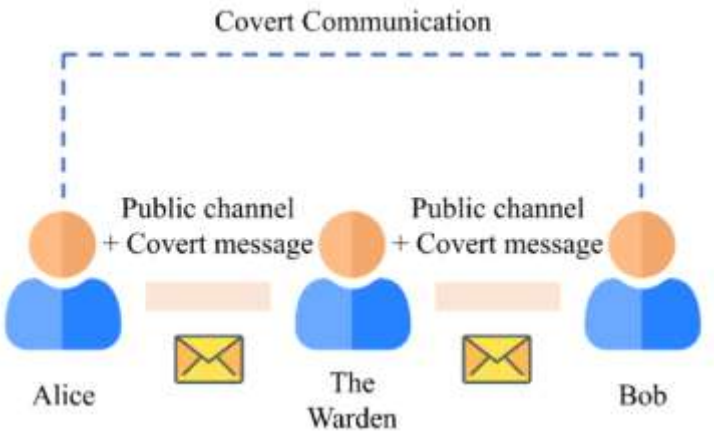
- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - DecETT
  - GraphTunnel
- 特点总结与知识展望
- 参考文献

预期收获

- 预期收获
  - 1.了解隧道流量识别的基本概念和研究方向
  - 2.理解两种隧道流量识别方法的基本原理
  - 3.了解现有隧道流量识别方法的缺陷及未来的发展方向

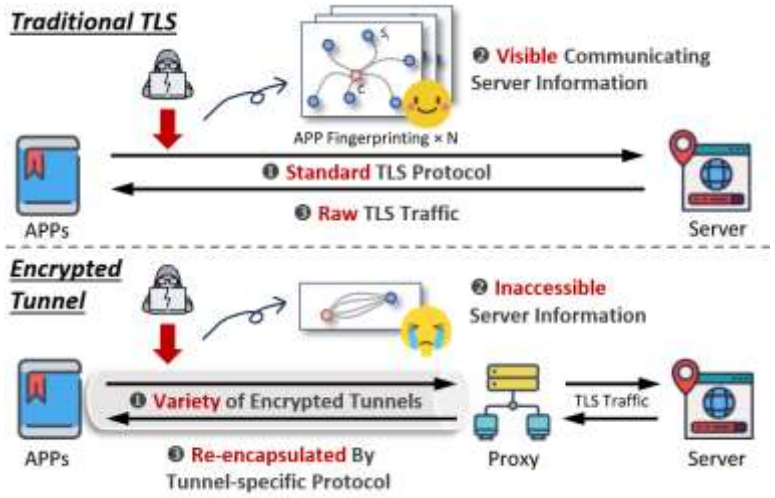
题目内涵解析

- 隐蔽隧道
  - 定义：一种利用网络协议中头部或有效负载的字段，或侧信道来传输信息的机制，使得监测系统难以发现其真实目的
  - 类型
    - 隐蔽存储隧道：通过修改协议中某些“存储型”字段来携带信息
    - 隐蔽时序通道：不改变数据本身，而是通过控制网络行为的时间特征来传递信息
  - 常见隧道
    - DNS 隧道，DoH/DoT隧道，SSH隧道， Shadowsocks隧道等
- 隧道流量识别
  - 识别是否为隧道流量以及隧道所使用的隧道协议
  - 识别隧道中的passenger协议
  - 识别隧道流量中的应用程序类型



研究背景及意义

- 研究背景
  - 隧道技术已广泛用于APT攻击与数据泄露
  - 攻击者利用加密隧道进行隐蔽数据传输，使检测系统很难感知异常
  - 新协议（QUIC/HTTP3、DoH）和新工具的出现增加检测难度
- 研究意义
  - 在加密化和隐蔽化的复杂网络环境中及时发现隐藏的恶意通信，从而有效防止数据泄露与攻击者的潜伏控制



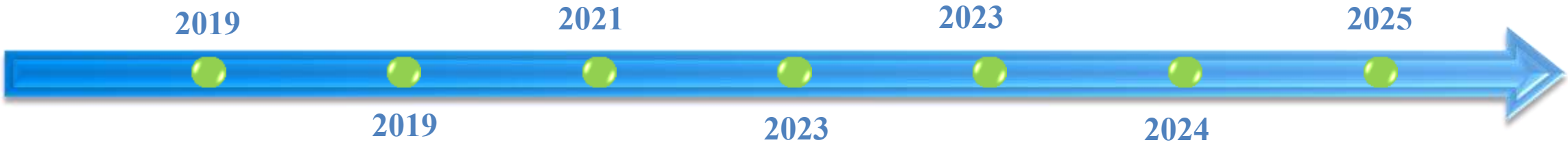
研究历史与现状

Lin等人提出基于规则和机器学习的应用层隧道检测，利用规则化DGA模块过滤明显的隧道流量，集成多种机器学习算法进行DNS，Http，Https隧道检测

Ding等人提出了一种基于变分自编码器的端到端异常检测模型，融合了注意力机制，在流级别对原始流序列数据进行建模，重构误差检测异常，实现DoH识别

Liang等人提出了一种基于CNN和聚类的DNS隧道检测模型，基于CNN模块提取特征，基于聚类的损失函数来优化模型参数，提高了未知的DNS隧道检测准确率

Gu等人提出一种双解耦语义增强的的隧道流量应用指纹识别方法，在隧道流量中分离协议语义表征和应用程序语义表征，并增强应用程序语义表征



Liu等人提出一种基于字节级卷积神经网络的DNS隧道检测方法，将字节数据转为可计算数据输入到CNN模型，解决人工特征提取困难的情况

Lv等人提出一种名为AAE-DSVDD的VPN流量识别方法，引入对抗自编码器（AAE）对VPN流量进行初步建模，通过深度支持向量数据描述（DSVDD）实现VPN流量的表征学习

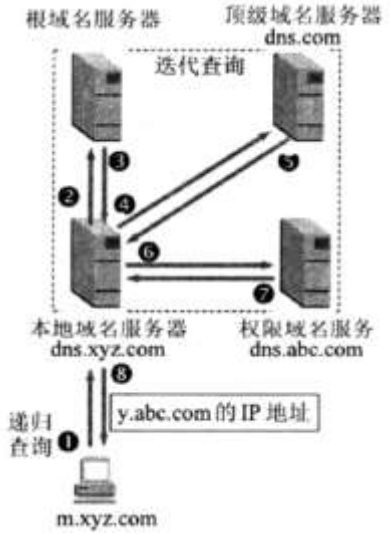
Gao等人提出了一种基于DNS递归解析图的稳健DNS隧道识别方法，设计隧道流量特征，构建DNS解析图结构，实现隧道流量识别以及隧道工具识别

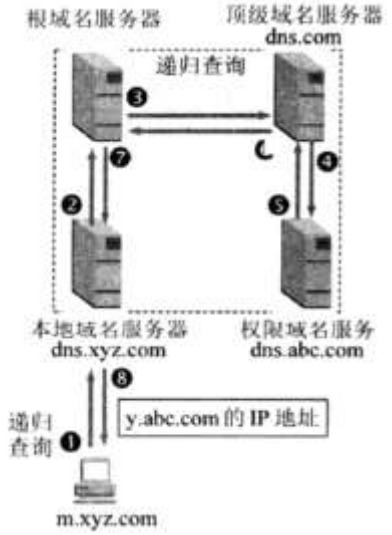
知识基础

- 流量粒度
  - 包级别（Packet）
    - 头部（header），有效负载（payload）
  - 流级别（Flow，单向流）
    - 五元组：(源IP, 源端口,目的IP, 目的端口, 传输层协议)
  - 会话级别（Session，双向流）
    - （源IP, 源端口）和（目的IP, 目的端口）互换，传输层协议不变

```
> Frame 19: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface lo, id 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 34012, Dst Port: 4443, Seq: 333, Ack: 6118, Len: 24
> Transport Layer Security
```

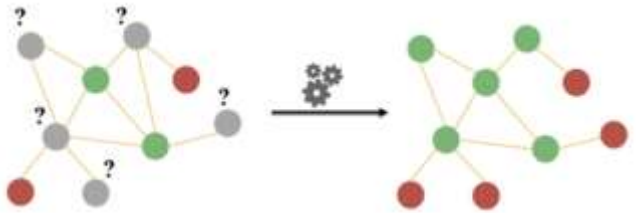
- DNS协议
  - 递归查询
  - 迭代查询





知识基础

- 图神经网络（GNN）
  - 通过消息传递（Message Passing）机制，让节点之间交换信息并更新自身表示
    - 聚合（Aggregate）：每个节点聚合邻居节点的信息
      - 均值聚合、池化聚合、LSTM聚合等
    - 更新（Update）：结合自身和邻居信息，生成新节点表示
      - 拼接、线性变换、非线性激活
    - 循环迭代：多次重复上述步骤，使信息在整个图中传播



$$h_v^{(k)} = update(h_v^{(k-1)}, aggregate(\{h_u^{(k-1)} \mid u \in \mathcal{N}(v)\}))$$

- 节点分类与图分类
  - 节点分类
    - 目标：已知图结构和部分节点的标签，预测其他节点的标签
  - 图分类
    - 目标：给定多个图样本，预测每个图的整体类别

DecETT: Accurate App Fingerprinting Under Encrypted Tunnels via Dual Decouple-based Semantic Enhancement

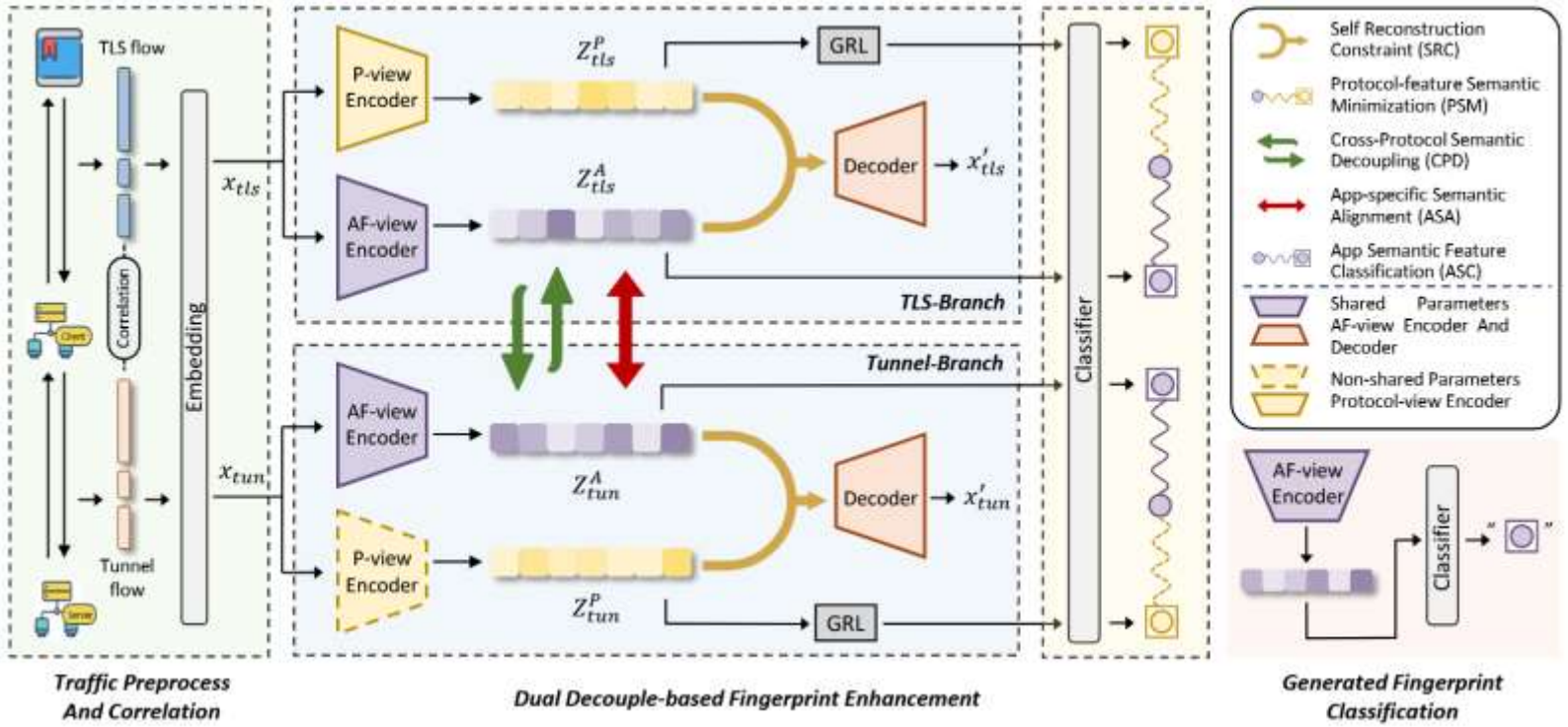
目标与难点

T	目标	加密隧道下的应用指纹精确识别
I	输入	TLS会话流*1，隧道会话流*1，会话流对应的应用标签*1
P	处理	1.构建并行的TLS流与隧道流的流量对，提取流序列并转为嵌入向量 2.设计流表征模块解耦协议语义和应用语义 3.设计应用语义增强模块增强应用语义 4.利用训练好的应用指纹编码器提取应用指纹并分类
O	输出	应用指纹分类结果

P	问题	现有方法将传统的应用指纹识别方法直接转移到加密隧道中，忽略了加密隧道的重新封装和混淆机制
C	条件	可以获取并行的加密流与隧道流
D	难点	如何解耦协议语义与应用语义，并增强应用语义
L	水平	CCF A（2025 WWW会议）

算法原理 总体结构

- DecETT
  - 流量预处理与关联
  - 基于双解耦的应用指纹增强
  - 生成指纹分类



算法原理

- 流量预处理与关联
  - 利用映射规则关联TLS流与其重新组装的隧道流

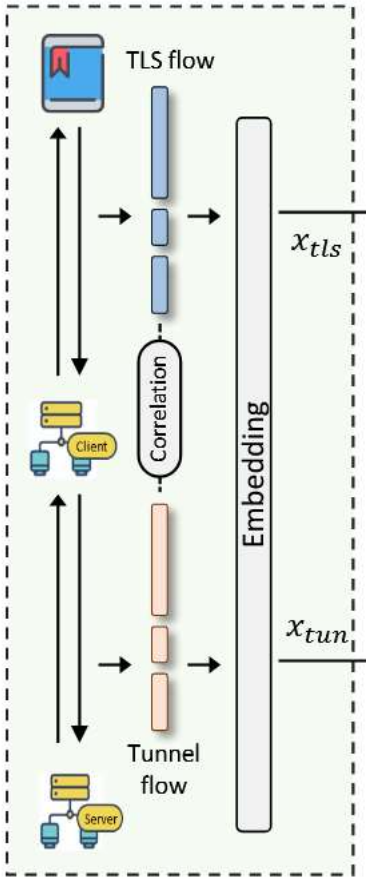
$$\begin{cases} (S^{tls}_{port}, S^{tun}_{port}) == (inbound, outbound) \in M \\ |t_{tls} - t_{tun}| \leq \varepsilon \end{cases}$$

  - 将TLS流与隧道流的流序列填充或截断为统一长度n

$$F_{tls-tun} = \{[p_{1,tls}, \cdots p_{n,tls}], [p_{1,tun}, \cdots p_{n,tun}]\}$$

  - 利用嵌入层将流对序列映射为嵌入向量

$$x_{tls-tun} = \{[e_{1,tls}, \cdots e_{n,tls}], [e_{1,tun}, \cdots e_{n,tun}]\}$$
$$= Emb(F_{tls-tun})$$



Traffic Preprocess And Correlation

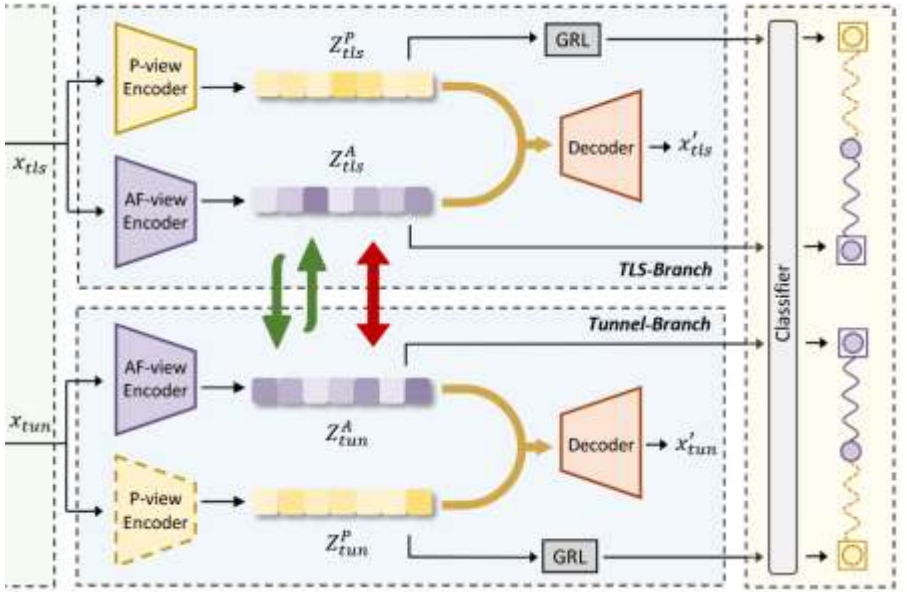
算法原理

- 基于双解耦的指纹增强
  - 目标：解耦纠缠的协议语义和应用语义特征，以TLS流量为锚点增强应用语义特征，减少隧道重新封装的影响
  - TLS分支与隧道分支
    - 协议视角编码器（不共享参数） $\mathrm{Enc}^P(\cdot)$
    - 应用指纹（AF）视角编码器（共享参数） $\mathrm{Enc}^A(\cdot)$
    - 解码器（共享参数） $Dec(\cdot)$
  - 分类器 MLP
  - 协议视角编码器旨在学习独立于应用程序的协议特征

$$Z^P = \mathrm{Enc}^P(x)$$

  - AF视角编码器旨在从原始流量中提取应用程序语义特征

$$Z^A = \mathrm{Enc}^A(x)$$



算法原理

- 基于双解耦的指纹增强
  - 流表征解耦（FRD）
    - 目标：强制应用程序语义特征与隧道协议特征解耦，以减少隧道重新封装的负面影响
    - 自重构约束（SRC）

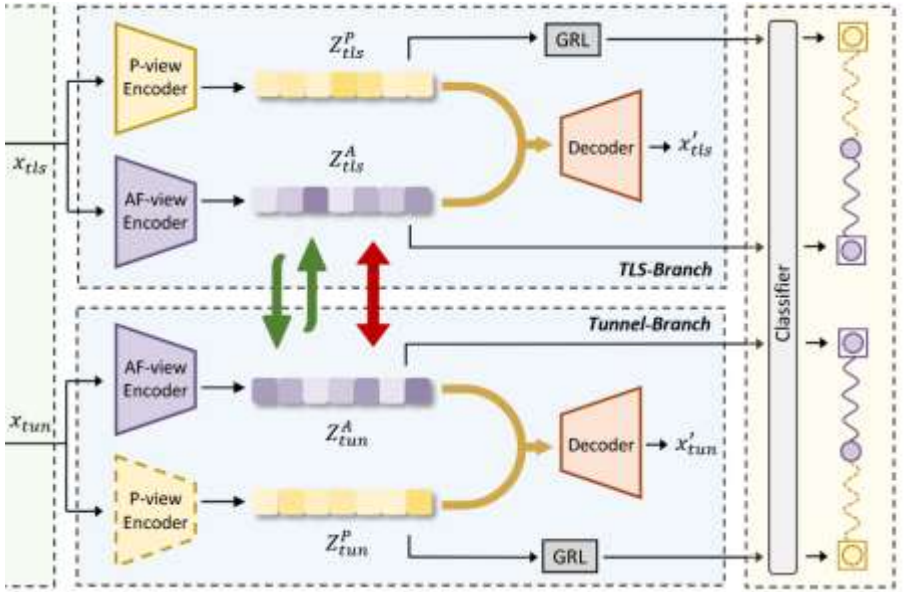
$$\grave{x}_{i,tls} = Dec(Z^P_{i,tls}, Z^A_{i,tls})$$

$$\grave{x}_{i,tun} = Dec(Z^P_{i,tun}, Z^A_{i,tun})$$

$$\mathcal{L}_{SRC} = -\frac{1}{N}\sum_{i=1}^{N}(\|\grave{x}_{i,tls} - x_{i,tls}\|^2 + \|\grave{x}_{i,tun} - x_{i,tun}\|^2)$$

    - 协议特征语义最小化（PSM）

$$\mathcal{L}_{PSM} = -\frac{1}{N}\sum_{i=1}^{N} y_i(\log(\hat{y}^P_{i,tls}) + \log(\hat{y}^P_{i,tun}))$$



算法原理

- 基于双解耦的指纹增强
  - 流表征解耦
    - 跨协议语义解耦（CPD）

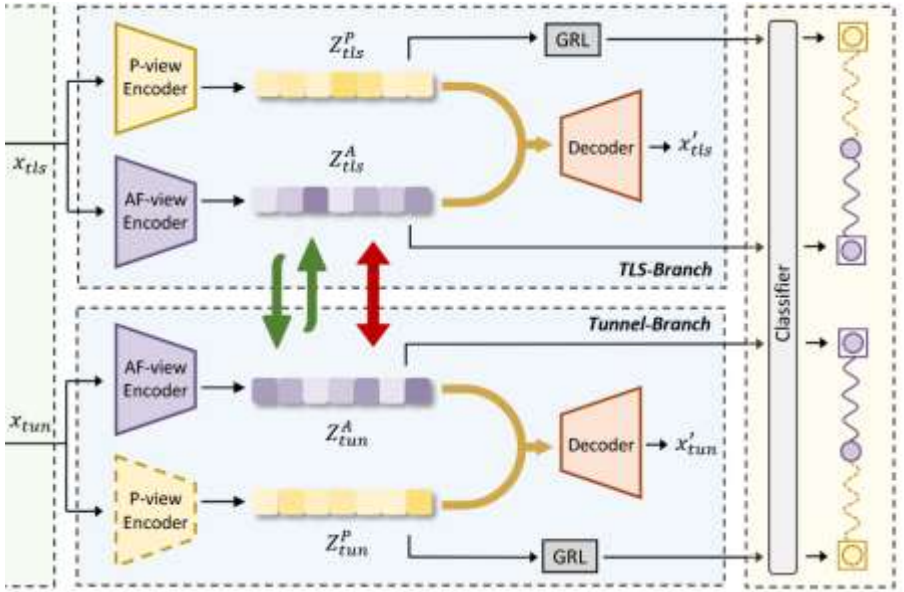
$$\hat{x}_{i,tls} = Dec\left(Z^{P}_{i,tls}, Z^{A}_{i,tun}\right)$$

$$\hat{x}_{i,tun} = Dec\left(Z^{P}_{i,tun}, Z^{A}_{i,tls}\right)$$

$$\mathcal{L}_{CPD} = -\frac{1}{\mathrm{N}}\sum_{i=1}^{N}\left(\left\|\hat{x}_{i,tls} - x_{i,tls}\right\|^{2} + \left\|\hat{x}_{i,tun} - x_{i,tun}\right\|^{2}\right)$$

    - 流表征解耦子模块总损失

$$\mathcal{L}_{\mathrm{FRD}} = \lambda_1\mathcal{L}_{\mathrm{FRD}} + \lambda_2\mathcal{L}_{\mathrm{PSM}} + \lambda_3\mathcal{L}_{\mathrm{CPD}}$$



算法原理

- 基于双解耦的指纹增强
  - 应用语义特征增强（AFA）
    - 目标：通过与两个具有强语义的监控信号对齐，进一步增强了与隧道流量解耦的应用程序语义特征
    - 应用语义对齐（ASA）

$$\mathcal{L}_{\mathrm{ASA}} = -\frac{1}{\mathrm{N}}\sum_{i=1}^{N}\left(1 - \frac{Z_{i,tls}^{A} \cdot Z_{i,tun}^{A}}{\left\|Z_{i,tls}^{A}\right\| \cdot \left\|Z_{i,\mathrm{tun}}^{A}\right\|}\right)$$

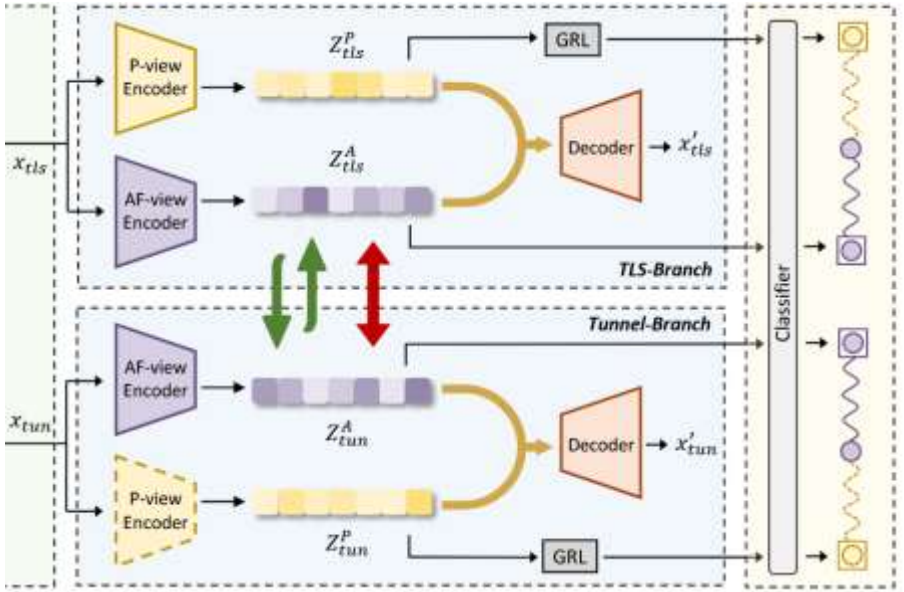
    - 应用语义特征分类（ASC）

$$\mathcal{L}_{ASC} = -\frac{1}{\mathrm{N}}\sum_{i=1}^{N} y_i\left(\log\left(\hat{y}_{i,tls}^{A}\right) + \log\left(\hat{y}_{i,tun}^{A}\right)\right)$$

    - 应用语义特征增强子模块总损失

$$\mathcal{L}_{\mathrm{AFA}} = \lambda_4\mathcal{L}_{\mathrm{ASA}} + \lambda_5\mathcal{L}_{\mathrm{ASC}}$$

  - DecETT总损失为$\mathcal{L}_{\mathrm{DecETT}} = \mathcal{L}_{\mathrm{FRD}} + \mathcal{L}_{\mathrm{AFA}}$

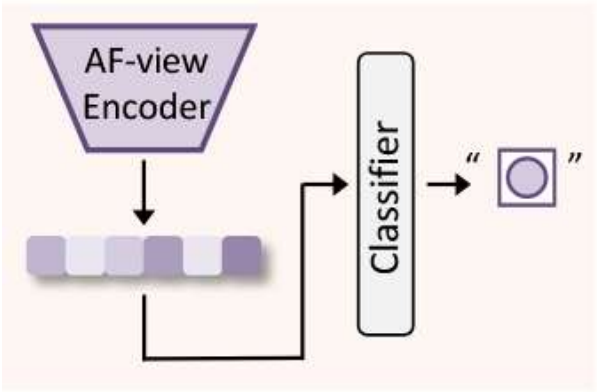


算法原理

- 生成指纹分类
  - 只需要$Emb(\cdot)$、 $Enc_{tun}^{A}(\cdot)$和隧道流F_{tun}，无需F_{tls} ，使得DecETT可以在真实的网络环境中使用

$$FP_{tun} = Enc_{tun}^{A}(Emb(F_{tun}))$$

$$y_{pred} = Classifier(FP_{tun})$$



Generated Fingerprint Classification

实验设计 数据资源

- 数据集收集
  - 选取5个具有代表性的加密隧道和54个广泛使用的应用程序进行流量收集

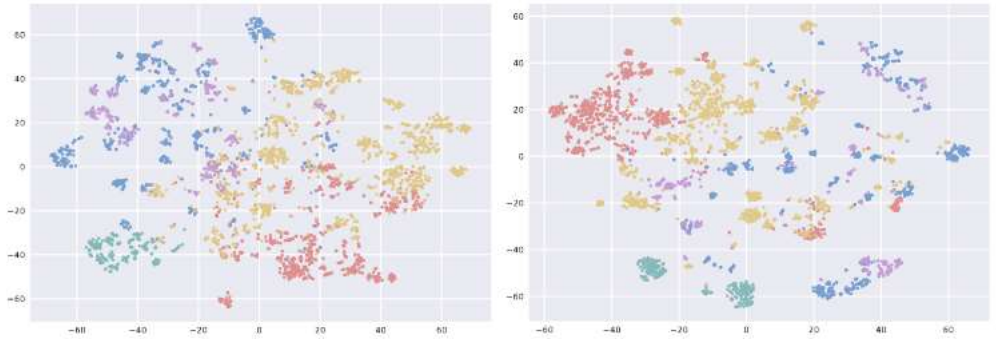
Tunnel	EA	Protocol	Obfs	Notes
Shadowsocks	AES-256-GCM	SOCKS	-	-
ShadowsocksR	AES-256-CFB	Origin	tls1.2_ticket_auth	-
V2Ray	AES-128-GCM	Vmess	-	
Trojan	AES-128-GCM	HTTPS	-	-
OpenVPN	AES-128-GCM	OpenVPN	-	TUN Mode

No.	Package Name
1	air.tv.douyu.android
2	cn.xdf.woxue.student
3	com.amazon.mShop.android.shopping
4	com.bilibili.app.in
5	com.bilibili.comic
6	com.bittorrent.client
7	com.duowan.kiwi
8	com.duowan.mobile
9	com.facebook.katana

- 对比方法
  - 基于统计的方法：AppScanner（2016）
  - 基于服务器信息的方法：FlowPrint（2020）
  - 基于有效负载的方法：ET-BERT（2022），YaTC（2023）
  - 基于序列的方法：DF（2018）、FS-Net（2019）和GraphDApp（2021）

实验设计 对比实验

- 结果表明
  - DecETT的表现明显优于所有其他比较方法
  - DecETT在V2Ray隧道下指标较低
  - 基于序列（FS-Net）的方法次于DecETT，基于基于服务器信息的方法最差

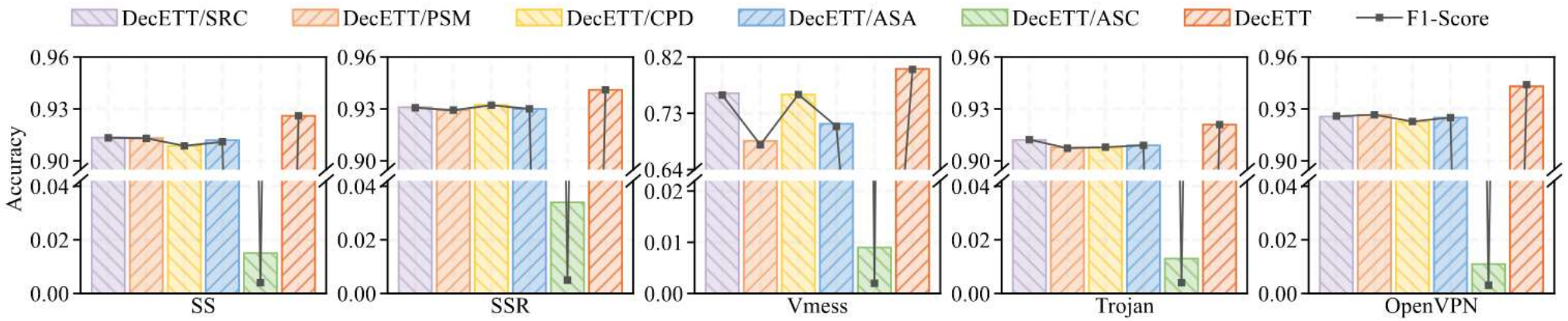


(a) FS-Net (b) DecETT

Method	Dataset	Shadowsocks				ShadowsocksR				V2Ray				Trojan				OpenVPN			
	Metric	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1
Statistic	AppScanner[30]	0.630	**0.995**	0.630	0.764	0.631	**0.996**	0.631	0.767	0.295	**0.993**	0.295	0.429	0.609	**0.996**	0.609	0.748	0.582	**0.995**	0.582	0.725
Server	FlowPrint[33]	0.122	0.015	0.122	0.027	0.053	0.003	0.053	0.005	0.103	0.013	0.103	0.022	0.050	0.008	0.050	0.012	0.027	0.001	0.027	0.002
Payload	ET-BERT[17]	0.079	0.085	0.079	0.045	0.098	0.134	0.098	0.085	0.055	0.072	0.055	0.032	0.280	0.216	0.203	0.203	0.265	0.300	0.265	0.256
	YaTC[42]	0.596	0.656	0.596	0.592	0.771	0.825	0.771	0.785	0.436	0.496	0.436	0.407	0.602	0.678	0.602	0.606	0.884	0.934	0.884	0.899
Sequence	DF[29]	0.739	0.746	0.739	0.738	0.762	0.764	0.762	0.760	0.656	0.659	0.656	0.651	0.726	0.730	0.726	0.724	0.816	0.818	0.816	0.816
	FS-Net[18]	0.845	0.837	0.838	0.837	0.856	0.849	0.850	0.849	0.610	0.610	0.606	0.610	0.822	0.828	0.822	0.823	0.876	0.874	0.873	0.874
	GraphDApp[28]	0.786	0.800	0.786	0.789	0.817	0.812	0.811	0.812	0.503	0.516	0.501	0.516	0.767	0.763	0.760	0.763	0.810	0.805	0.806	0.805
Ours	DecETT	**0.925**	0.926	**0.925**	**0.925**	**0.942**	0.942	**0.942**	**0.942**	**0.802**	0.803	**0.802**	**0.801**	**0.920**	0.922	**0.920**	**0.921**	**0.941**	0.941	**0.941**	**0.941**

实验设计 消融实验

- 结果表明
  - 移除SRC后，DecETT/SRC在5条隧道中的准确率下降了1%至4%
  - DecETT/PSM和DecETT/CPD的F1平均得分分别下降了3.56%和2.01%
  - 去除ASA对DecETT的影响显著，在V2Ray下F1得分最高下降了9%
  - 去除ASC后，DecETT的性能急剧下降



算法原理

- 算法总结
  - 算法贡献
    - 提出了一种基于双解耦的语义增强方法DecETT，在各种加密隧道下实现准确AF
    - 考虑到重新封装导致的应用程序特定信息的混淆，引入了具有更强、更稳定的应用程序语义信息的TLS流量作为锚点，以指导和增强有效的指纹生成
    - 设计了一个基于双解耦的语义增强模块，将隧道相关特征和应用程序特定的语义特征解耦，从而减轻了重新封装对准确指纹提取的负面影响
  - 算法不足
    - 加密流分支的加密协议单一，仅验证TLS，未验证其他加密协议

GraphTunnel: Robust DNS Tunnel Detection Based on DNS Recursive Resolution Graph

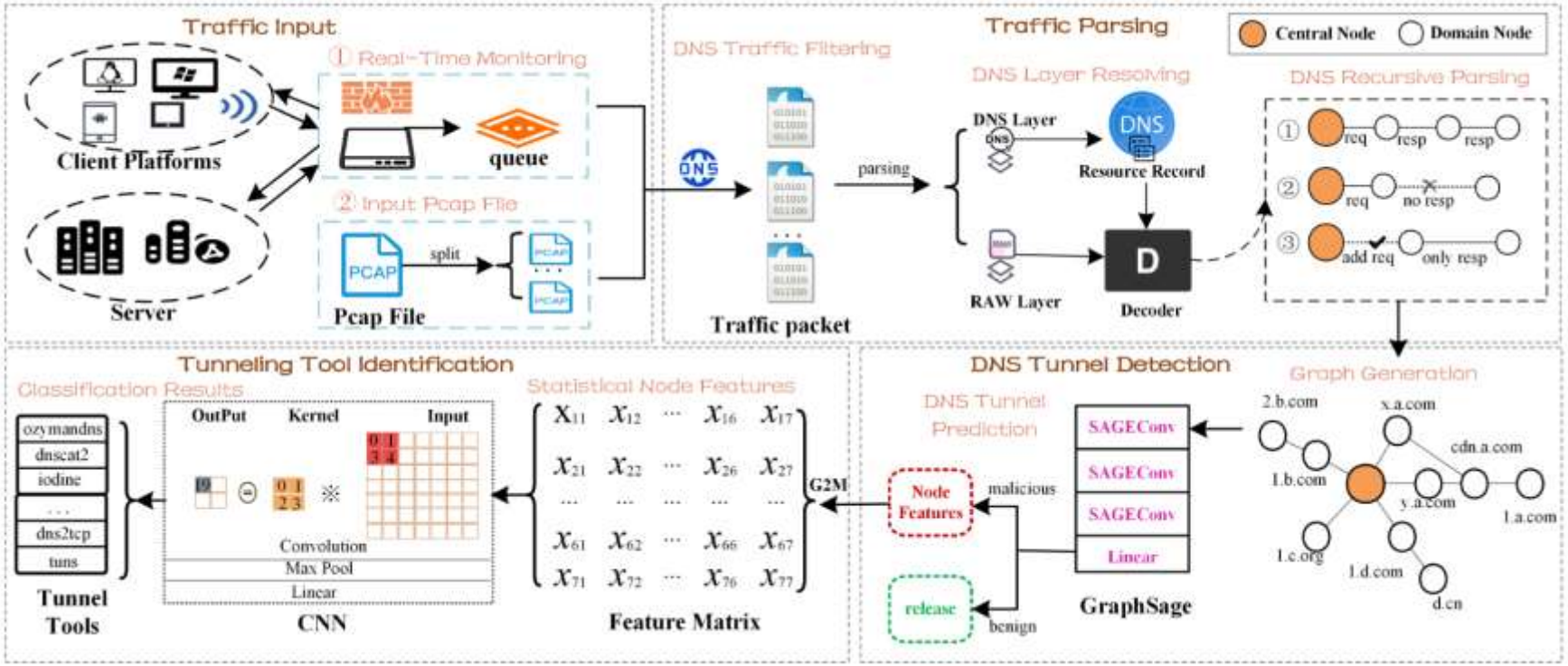
目标与难点

T	目标	检测是否为DNS隧道流量，以及使用的隧道工具
I	输入	DNS会话流*1，会话流对应的隧道标签*1，隧道工具标签*1
P	处理	1.提取会话流中各响应的DNS层，构建特征 2.根据查询响应构建解析路径，组合为图 3.通过GNN聚合并更新节点特征，得到图特征向量 4.利用图特征向量进行隧道检测，并设计G2M算法进行隧道工具检测
O	输出	是否为隧道流量（2分类），隧道工具类别（多分类）

P	问题	现有方法集中在域名或数据包有效载荷的表层特征上，缺乏对DNS隧道建立和攻击过程的行为结构特征的关注，导致在处理未知DNS隧道攻击和来自wildcard DNS的流量时检测准确率较低
C	条件	会话流中响应满足一定数目
D	难点	如何根据查询响应构建递归解析路径
L	水平	CCF A（2024 TIFS期刊）

算法原理 总体结构

- GraphTunnel
  - 流量输入
  - DNS流量解析
  - DNS隧道检测
  - 隧道工具识别



算法原理

- DNS流量解析
  - DNS流量筛选：UDP协议以及53端口
  - DNS层提取：提取数据包的DNS层或最高层，头部数据丢弃，使用scapy解决跨平台数据包格式不统一的问题
  - 特征提取
    - 记录类型分数：根据每种记录类型的用户覆盖率分配一个分数作为特征

Type	Score	Type	Score
A	99.948	NS	0.906
AAAA	82.082	TKEY	0.026
PTR	50.078	NAPTR	0.363
SRV	69.653	SPF	0.026
MX	1.165	CNAME	0.233
ANY	39.513	AXFR	0.026
SOA	9.089	NULL	0.026
TXT	17.607	Others	0

    - 子域名长度：DNS隧道在二级域名之后构建长的子域名，以增加数据量

$$F_2 = |D_{sub}| = \left|D_{FQDN} - D_{second}\right|, 0 < \left|D_{FQDN}\right| < 255$$

算法原理

- DNS流量解析
  - 特征提取
    - 子域名最长长度：良性DNS查询在子域名结构中表现出相对简单的层次结构，每级别的子域名长度都较短，提取每个子域名的长度，并确定之间的最大长度，从而区分DNS隧道和良性流量

$$F_3 = |D_{max}| = max_{i=a}^{n} |D_{sub_i}|, 0 < |D_{sub_i}| < 63$$

    - 连续辅音字符串数目：DNS隧道内的子域名通常包含大量随机字母组合，以最大限度地传输信息，从而产生大量的长的辅音串$F_4 = \sum_{i=1}^{m} I(|C_i > 4|), C_i \in \{D_{max}\}$
    - 最长子域名的熵：DNS隧道中，编码技术通常用于隐藏传输的数据，导致更多的随机字母组合，提高子域的熵值$F_5 = -\sum_{i=1}^{n}(p(x_i)logp(x_i))$

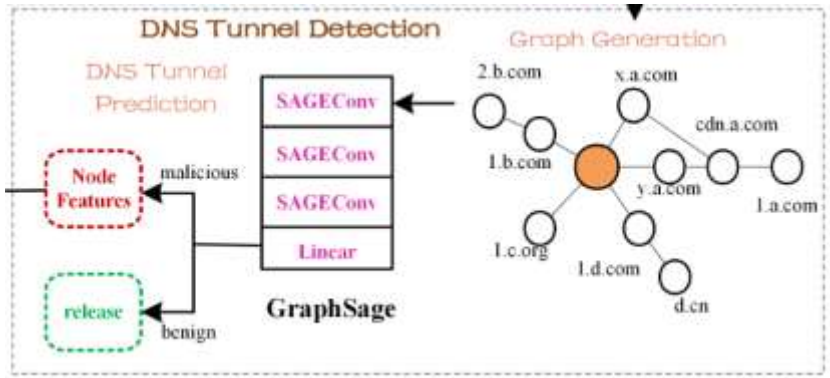
算法原理

- DNS流量解析
  - 特征提取
    - TTL值：隧道流量中的TTL值分布往往集中在较短的时间范围内，而良性流量中的TTL值相对较长，分布范围更广
    - 数据包字节大小：DNS隧道需要在DNS数据包中隐藏数据，与良性DNS流量相比，相应的请求和响应消息的大小要大得多
    - 平均响应时间：良性递归DNS解析需要与多个权威域名服务器进行交互，每个查询都需要一定的时间，从而影响整体响应时间

$$F_8 = \frac{T_{res} - T_{req}}{L_n}$$

算法原理

- DNS隧道检测
  - 图构建
    - 节点
      - 设置中间节点，连接各查询路径
      - 每个节点代表唯一的域名，节点封装了7个属性，包括信息熵等
    - 边
      - 每当递归查询被转发到新的权威域服务器时，确定该域名节点是否预先存在于图中，如果存在，建立直接连接，否则实例化一个新节点连接
      - 边属性对应于域名解析过程的平均响应时间
  - 隧道检测
    - 堆叠GraphSAGE聚合并更新节点消息
    - 中心节点聚合并编码整个DNS查询关系图的拓扑结构信息和每个节点的特征信息
    - 中心节点的综合特征向量作为整个图的嵌入表达式，输入多层感知机进行二分类



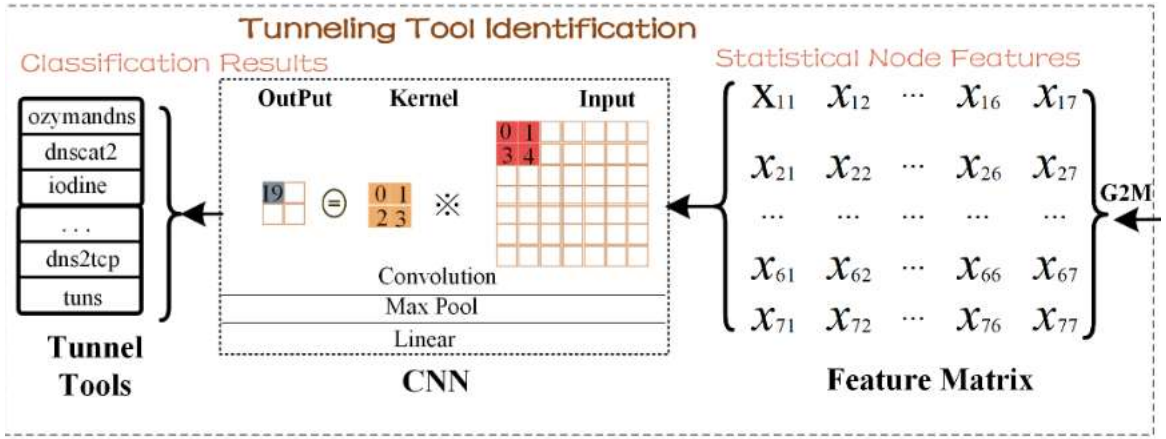
算法原理

- DNS隧道工具检测（G2M算法）
  - 统计属性提取：分析传入图嵌入向量中的每一类特征，提取7个统计属性：方差、均值、标准差、范围、中值、偏度和峰度
  - 分类：排列统计属性得到7*7的二维矩阵，输入到卷积神经网络，提取和压缩统计特征，利用线性层进行隧道工具分类

Algorithm 1 G2M Algorithm

Input: Graph G
Output: Matrix M

```
V ← GraphEmbedding(G)
M ← InitializeEmptyMatrix(7 × 7)
for each feature F in V do
    M[i] ← [var(F), mean(F), std(F), range(F),
        median(F), skewness(F), kurtosis(F)]
end for
return M
```



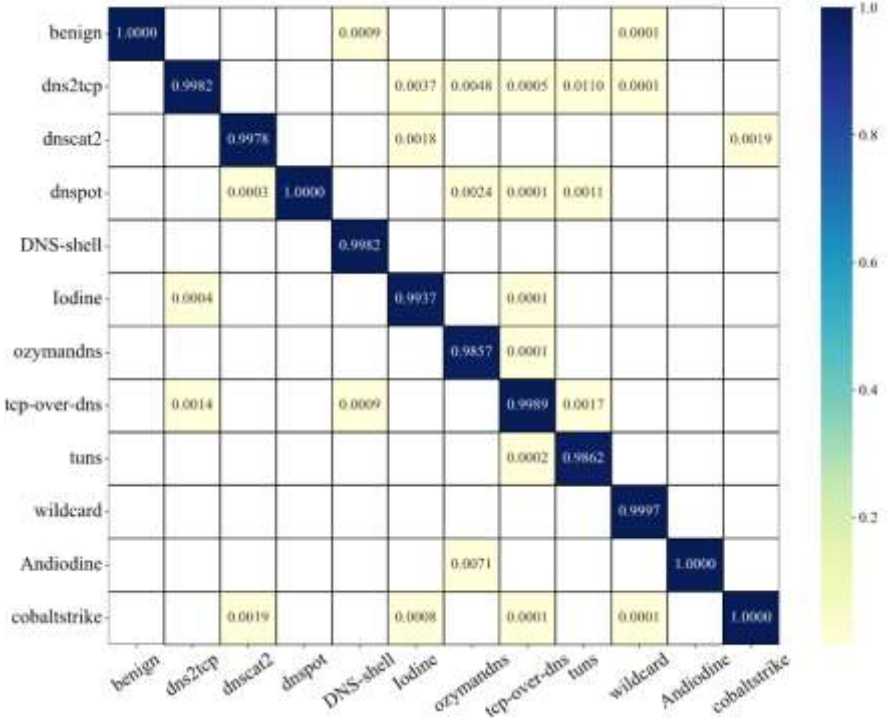
实验设计 数据资源

- 数据集
  - $Dataset_{our}$
    - 良性：从Cloudfare上获得了前1000000个域名，wireshark被用来监视和收集生成的标准DNS流量数据
    - 隧道：使用了10种DNS隧道工具，包括iodid、dnscat2和dns2tcp等，在本地内联网和公共服务器之间建立DNS隧道
  - $Dataset_{CIC}$
    - CIC-Bell-DNS-EXF-2021，良性流量与DNS流量为6：4
  - $Dataset_{korving}$
    - 三种DNS隧道工具产生的流量
  - 对比方法
    - D’Angelo(2022)，Mahdavifar(2021)，Suman(2023)，Filippo(2023)

实验设计 对比实验

- 结果表明
  - 所有方法在隧道流量监测上都呈现高准确率和F1值
  - D’Angelo在跨数据集上Acc与F1值下降50%，GraphTunnel在跨数据集上表现稳定
  - 10种DNS隧道工具检测上表现出高准确率

Model	Dataset	Accuracy	Precision	F1
Samaneh et al.	$Dataset_{CIC}$	0.9997	0.9997	0.9997
Suman et al.	$Dataset_{CIC}$	0.989	0.992	0.989
Filippo et al.	$Dataset_{CIC}$	0.971	0.945	0.956
D'Angelo et al.	$Dataset_{CIC}$	0.9977	0.9995	0.9971
D'Angelo et al.	$Dataset_{korving}(C2)$	0.3999	**1.0**	0.5714
D'Angelo et al.	$Dataset_{korving}(file)$	0.4913	**1.0**	0.6589
GraphTunnel	$Dataset_{CIC}$	**1.0**	**1.0**	**1.0**
GraphTunnel	$Dataset_{korving}(C2)$	0.9876	**1.0**	0.9937
GraphTunnel	$Dataset_{korving}(file)$	0.9994	**1.0**	0.9997



算法原理

- 算法总结
  - 算法贡献
    - 构建了DNS递归解析图，并通过分析正常DNS解析和DNS隧道解析之间的不同行为图模式，采用图神经网络来检测DNS隧道
    - 开发了G2M算法，通过统计分析节点特征向量并将其组织成灰度图像矩阵进行有效的卷积聚合，来改进DNS隧道工具的多分类
  - 算法不足
    - 通配符DNS解析对准确检测构成了相当大的挑战
    - 未对图的路径数目做参数实验验证

特点总结与未来展望

研究工作总结

- 特点总结
  - DecETT
    - 提出了一种基于双解耦的语义增强方法DecETT，将隧道相关特征和应用程序特定的语义特征解耦，可以在各种加密隧道下实现准确的应用指纹识别
  - GraphTunnel
    - 构建了DNS递归解析图，并通过分析正常DNS和DNS隧道之间的不同行为图模式，设计G2M算法，通过统计分析节点特征向量并将其转为灰度图像矩阵进行有效的卷积聚合，实现隧道流量及隧道工具的识别
- 未来展望
  - 针对多类隧道进行识别
  - 在识别过程引入字节序列特征

参考文献

- [1] GU Z, LIU C, ZHANG X, et al. DecETT: Accurate App Fingerprinting Under Encrypted Tunnels via Dual Decouple-based Semantic Enhancement[C]. International World Wide Web Conference, 2025: 2413-2423.
- [2] GAO G, NIU W, GONG J, et al. GraphTunnel: Robust DNS Tunnel Detection Based on DNS Recursive Resolution Graph[J]. IEEE Transactions on Information Forensics and Security, 2024, 19: 7705-7719.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！