Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

加密代理流量识别

硕士研究生 李国浩

2026年08月23日

问题回溯

- 存在的问题
  - 部分内容讲解语速过快
  - 对于知识基础部分讲解比较拖沓
  - 演讲时间较短
- 相关内容
  - 2026.02.03 李国浩《隧道流量识别研究》
  - 2025.06.04 李国浩《基于GNN的加密流量识别方法》

内容提要

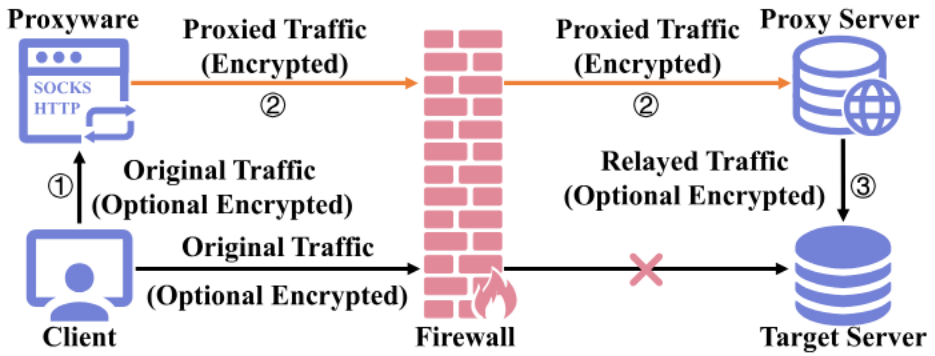
- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - ProxyKiller
  - EO-EPTC
- 特点总结与工作展望
- 参考文献

预期收获

- 预期收获
  - 了解常见的代理隧道连接形式
  - 了解代理隧道识别的应用场景及未来发展方向
  - 理解两种代理流量识别方法

内涵解析与研究目标

- 题目内涵解析
  - 代理流量：在客户端与目标服务之间引入了代理节点，代理节点会对连接进行封装、复用或加密
  - 识别难点：加密与封装削弱了内容和协议的可见性
- 研究目标
  - 研究不依赖有效载荷解密的代理流量表征与识别方法
  - 识别代理流量的代理协议，应用指纹、网站指纹等



研究背景与意义

- 研究背景
  - 代理工具：各类代理工具通常利用加密、隧道封装和协议伪装技术，将代理通信隐藏在HTTPS、WebSocket等正常流量中
  - 传统方法：有效载荷不可见，传统依赖端口、明文特征和固定协议指纹的检测方法难以有效识别这类流量
- 研究意义：
  - 发现违规代理访问、隐蔽通信和恶意流量隧道，为网络安全监测和异常行为分析提供技术支持
  - 引起代理开发社区关注，促进相关协议设计和发展，更好地保护用户隐私

研究历史

2020

Alice等人利用首个数据包的数据包长度、字节熵、加密块长度余数确定可疑流，向可疑服务器发送多种探测数据，确定可疑服务器是否运行SS代理协议

2021

Wang等人在时间、长度等统计特征的基础上引入滑动窗口JS散度特征，分析不同时间窗口内流量分布的变化，通过分布变化刻画隧道内部应用行为

2023

Wu等人采用正常流量豁免规则，分析数据包的平均Popcount、可打印字符数量与比例、是否匹配TLS协议头等代理协议共同属性，检测是否为加密代理协议

2024

ProxyKiller提出利用字节特征和流量行为图的方法，在传统时空特征上，增加字节特征，根据节点相关性构建多流交互行为图，弥补了缺乏针对不同代理流连接行为特征的设计

2024

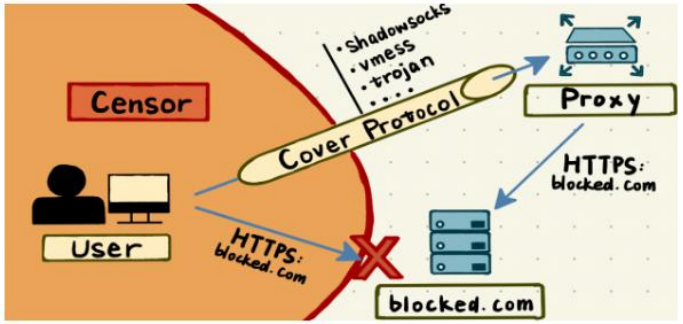
Xue等人通过内层握手产生的突发大小、往返次数、包方向变化、时序相似性等提取封装TLS握手指纹，从协议专用指纹发展为协议无关的代理机制指纹，实现代理流量监测

2026

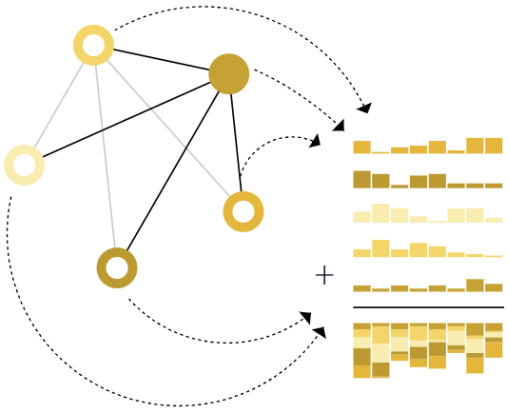
EO-EPTC从原始流量和代理流量中解析出长度序列，进行预对齐和转换关系学习，缓解因传输和封装造成的分布差异，根据原始流量序列生成代理流量序列，缓解代理流量标签依赖

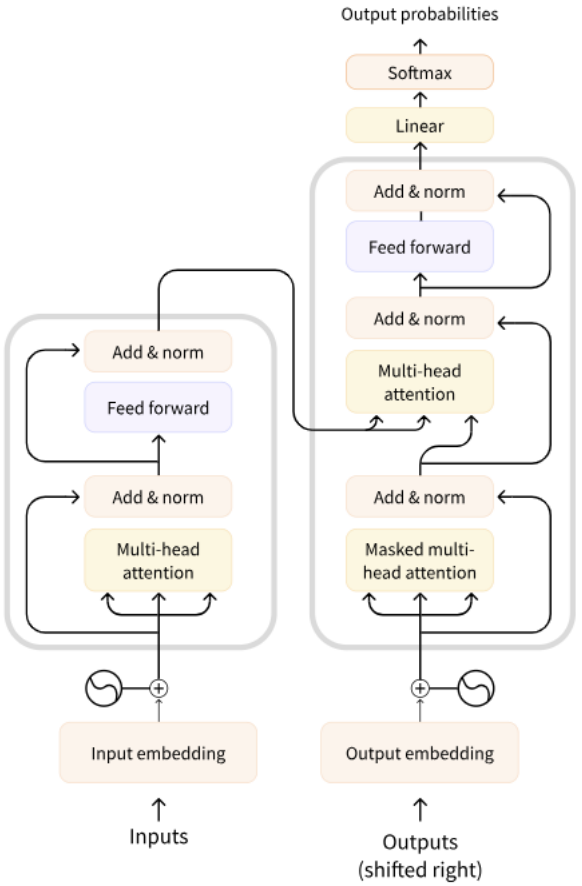
知识基础

- 代理协议
  - 基本过程：本地应用→代理客户端→加密代理隧道→代理服务器→目标服务器
- 分类粒度
  - 数据包、流（会话），多流，主机，网络
- 协议类型
  - SS、Vmess、Trojan、Vless
- SSH反向代理隧道
  - 命令：
    - ssh -R 7890:localhost:7890 -p 22 <remote_username>@<server_ip>
    - export http_proxy=http://127.0.0.1:7890
    - export https_proxy=http://127.0.0.1:7890
  - 作用：远程服务器的流量通过SSH加密隧道，转发到本地计算机的7890端口



- **Transformer**
  - **Encoder：编码输入序列**
    - **多头自注意力**
      - $Attention(Q,K,V) = softmax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V$
    - **前馈神经网络**
    - **残差连接和层归一化**
  - **Decoder：生成输出序列**
    - **Masked Self-Attention**
    - **Cross-Attention**
- **GIN**
  - **聚合邻居节点：** $h_v^{(k)} = MLP^{(k)}\left(\left(1+\varepsilon^{(k)}\right)h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right)$





ProxyKiller: An Anonymous Proxy Traffic Attack Model Based on Traffic Behavior Graphs

ProxyKiller TIPO

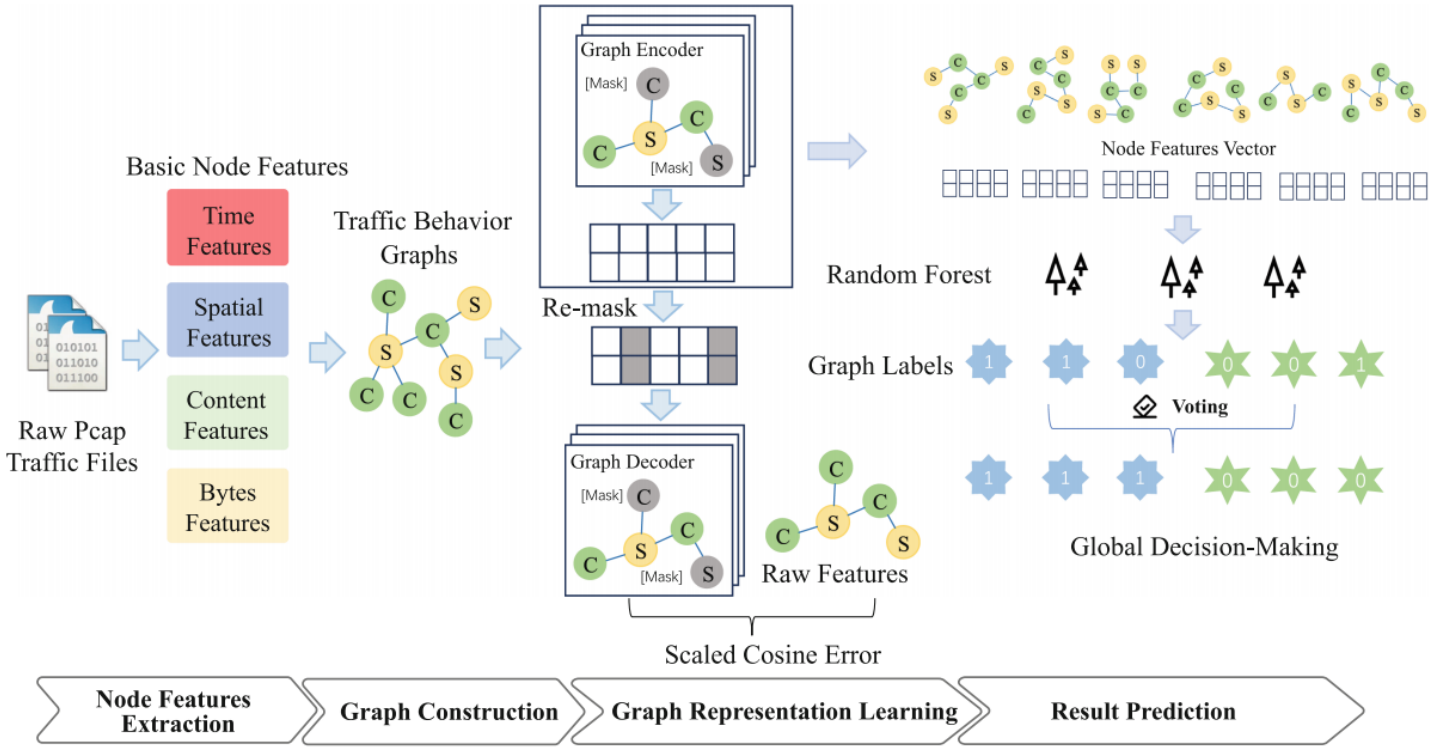
T	目标	识别代理协议类型
I	输入	局域网出口捕获的pcap文件*1
P	处理	1.会话粒度分割，标识节点，节点特征提取 2.根据实际通信边和关联边构造流量行为图 3.利用GraphMAE进行掩码节点特征重构 4.节点分类与全局决策修正
O	输出	流量行为图对应的代理协议标签*1
P	问题	1.时空特征难以充分刻画匿名代理流量 2.单流分类忽略多条流之间的关联行为
C	条件	需要区分流中客户端和服务端方向
D	难点	1. 如何构建多流的流量行为图 2.如何控制图规模与保持行为完整性
L	水平	2024 CCF-B

ProxyKiller 研究问题

- 两个问题
  - 单条流量的特征表达不足
    - 应用问题：在真实网络出口，普通流量与匿名代理流量具有部分相似的统计特征，且代理工具能够通过包长、时间和协议头伪装规避检测，导致现有方法产生误报和漏报
    - 科学问题：代理协议的封装与加密削弱了时空统计特征的稳定判别能力，时间、空间、内容及字节分布等侧信道特征在语义层面存在差异，难以形成统一表征
  - 单流建模无法利用多流连接行为
    - 应用问题：单条流的持续时间有限且统计特征不稳定，仅依据孤立流进行判断容易产生偶然误判，难以稳定识别用户整体的匿名代理行为
    - 科学问题：传统方法通常将五元组流视为相互独立的分类样本，忽略同一代理行为产生的多条流在通信对象、服务器网段、协议指纹和访问域名等方面存在的结构依赖

ProxyKiller 算法原理图

- 思路分析
  - 节点特征提取：时间、空间、内容、字节特征提取
  - 流量行为图构建：实际通信边和关联边
  - 图表征学习：提取隐式拓扑特征



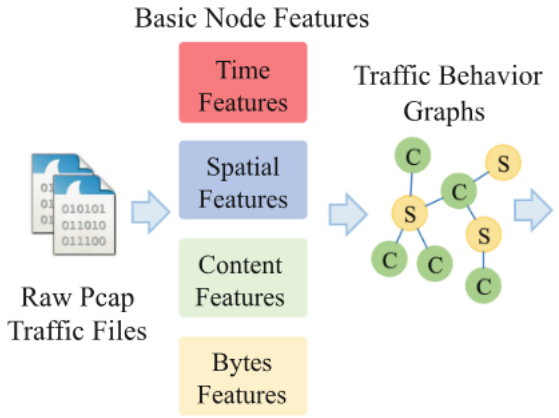
算法原理 节点特征提取

- 节点特征提取
  - 以会话粒度切分流量文件，将每条会话标识为三元组$(flow_{id}, ip, port)$，将会话的客户端和服务端表示为两个节点
  - 时间、空间、内容特征提取
  - 字节特征提取
    - 内容分布特征
    - payload长度分布
    - payload大小、平均popcount、平均可打印字符数
  - 每个节点对应一个381维的初始特征向量

ID	Feature Name	Directions	Category
1	Protocol	-	Content Features
2	IP Version	-	Content Features
3-4	Duration (ms)	Bidirectional, Single	Temporal Features
5-14	Packet Count	Bidirectional, Single	Spatial Features
15-16	Bytes Size	Bidirectional, Single	Spatial Features
17-24	Packet Interval	Bidirectional, Single	Temporal Features
25-26	SYN Packet Count	Bidirectional, Single	Content Features
27-28	CWR Packet Count	Bidirectional, Single	Content Features
29-30	ECE Packet Count	Bidirectional, Single	Content Features
31-32	URG Packet Count	Bidirectional, Single	Content Features
33-34	ACK Packet Count	Bidirectional, Single	Content Features
35-36	PSH Packet Count	Bidirectional, Single	Content Features
37-38	RST Packet Count	Bidirectional, Single	Content Features
39-40	FIN Packet Count	Bidirectional, Single	Content Features
41	Application Name	-	Content Features
42	Application Category Name	–	Content Features
43	Application is Guessed	–	Content Features
44	Payload Size in Bytes	–	Byte Features
45-300	Payload Bytes Content Distribution	–	Byte Features
301-379	Payload Length Distribution	–	Byte Features
380	Bytes Average Popcount	–	Byte Features
381	Bytes Average Printable Count	–	Byte Features

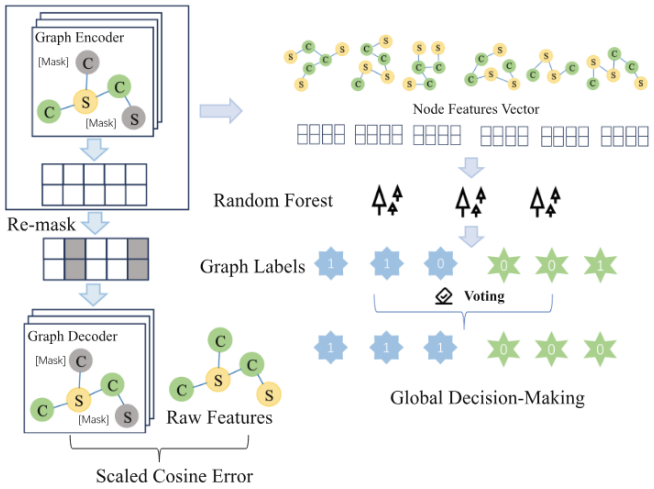
算法原理 流量行为图构建

- 流量行为图构建
  - 实际通信边：存在从节点a到节点b的流，则表示实际交互，可建立无向边
  - 关联边：连接具有相似或相同属性的节点
    - 相同服务器IP
    - 服务器IP属于同一C类子网
    - 客户端相同JA3指纹
    - 相同域名
  - 图大小限制：如果相关节点超过L个，切分成多张流量行为图
  - $G = (X, A)$
    - $X \in \mathbb{R}^{|N| \times 381}$：节点特征矩阵
    - A：由实际边和关联边构成的邻接关系
    - $|N| \leq L$



算法原理 图表征学习

- 图表征学习
  - 随机掩码：随机选择部分节点，将节点特征掩码，得到掩码输入图$\tilde{G} = (\tilde{X}, A)$
  - 编码：使用GIN作为基模型，输出节点嵌入矩阵$\boldsymbol{H} = GraphEncoder(\tilde{G})$
    - 节点自身及相邻节点4类特征
    - 多流之间连接行为
  - 解码：输出重建节点嵌入矩阵$\hat{\boldsymbol{H}} = GraphDecoder(\boldsymbol{H}, A)$
    - 利用邻居节点之间未屏蔽特征的分布来重建屏蔽节点的特征向量
  - 缩放余弦误差计算：$Loss_{SCE} = \frac{1}{|V_M|}\sum_{v \in V_M}(1 - \frac{x_v^{\mathrm{T}}\hat{x}_v}{||x_v||||\hat{x}_v||})^{\gamma}$



- 图预测
  - 随机森林节点级预测，图级多数投票
  - 全局决策修正

- 数据集
  - Datacon2021：6种代理工具生成流量
  - AnonProxy2023：4类匿名代理协议流量
  - LFETT2021：2类代理协议下不同应用流量
- 评价指标
  - Accuracy、Recall、Precision、F1
- 基线方法
  - 机器学习： CUMUL、BIND
  - 深度学习： GraphDapp、FS-Net
  - 专有模型： SSAPPIDENTIFY

实验设计　对比实验

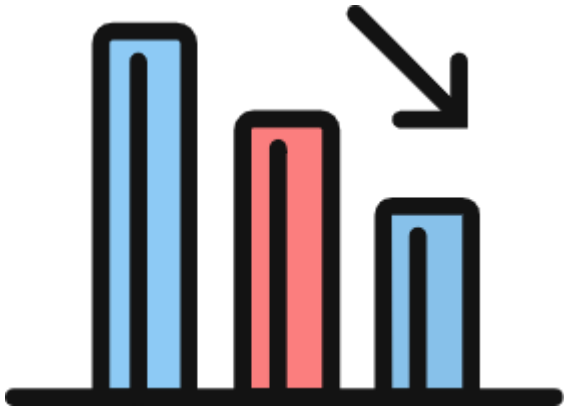
- **Datacon2021上F1的提升范围为2.77%~34.37%**
- **AnonProxy2023上F1的提升范围为1.76%~10.82%**
- **LFETT2021上F1的提升范围24.07%~53.1%**

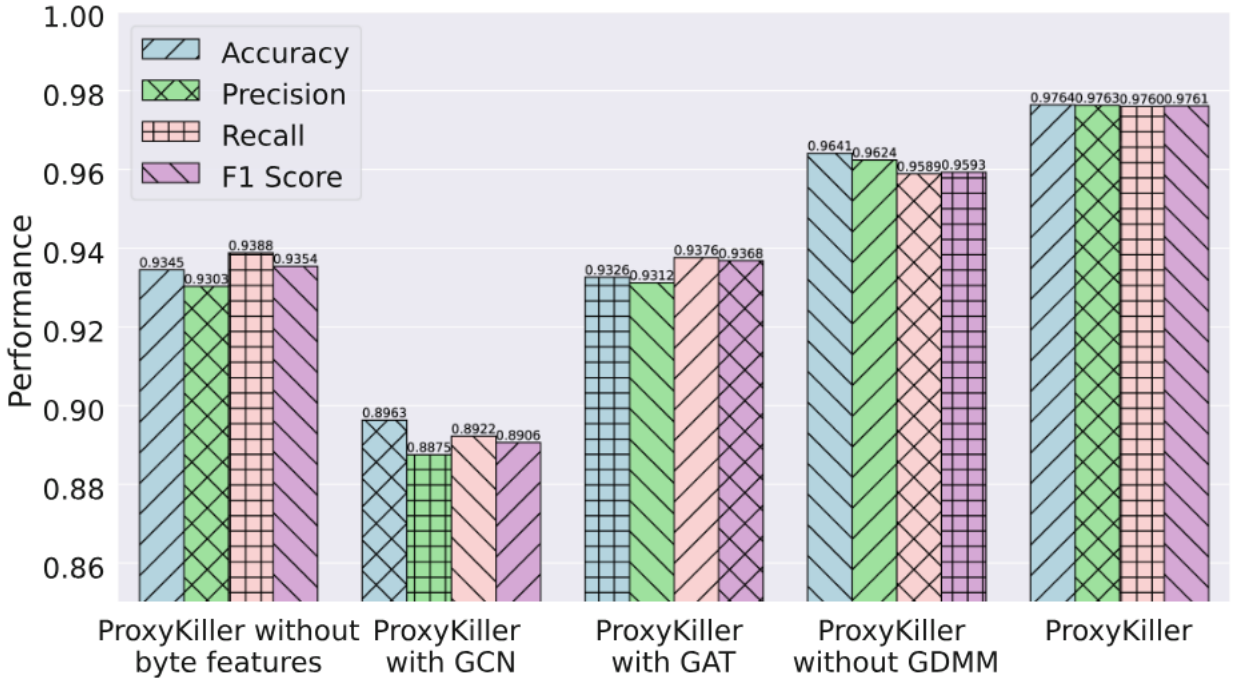
Methods	Datacon2021				AnonProxy2023			
	Acc	Pr	Re	F1	Acc	Pr	Re	F1
CUMUL [19]	0.6742	0.6488	0.6654	0.6527	0.9125	0.8959	0.9072	0.9013
BIND [1]	0.7892	0.8156	0.8093	0.8107	0.8842	0.8891	0.8816	0.8853
FS-Net [18]	0.9679	0.9686	0.9603	0.9642	0.9677	0.9643	0.9682	0.9661
GraphDapp [23]	0.7436	0.6994	0.6845	0.6858	0.9175	0.9052	0.9134	0.9087
SSAPPIDENTIFY [25]	0.9783	0.9752	0.9645	0.9687	0.9783	0.9764	0.9731	0.9759
ProxyKiller	**0.9924**	**0.9967**	**0.9962**	**0.9964**	**0.9949**	**0.9938**	**0.9912**	**0.9935**

Methods	LFETT2021			
	Acc	Pr	Re	F1
CUMUL [19]	0.5472	0.4731	0.4386	0.4451
BIND [1]	0.5764	0.5639	0.5547	0.5582
FS-Net [18]	0.5026	0.5178	0.5142	0.5149
GraphDapp [23]	0.5204	0.4925	0.4874	0.4896
SSAPPIDENTIFY [25]	0.7675	0.7212	0.7387	0.7354
ProxyKiller	**0.9764**	**0.9763**	**0.9760**	**0.9761**

实验设计　消融实验

- 消融字节特征
  - 评估指标下降3.72%~4.6%
- 替换GNN模型
  - 评估指标下降3.84%~8.88%
- 消融全局决策
  - 评估指标下降1.23%~1.71%





实验设计　可解释实验

- 使用随机森林模型的可解释性评估节点特征
  - 字节分布特征对代理协议的分类有显著影响
  - 连接持续时间等相关时间特征在代理行为和代理工具分类任务中有显著影响

Rank	AnonProxy2023	Datacon2021	LFETT2021
1	Payload Bytes Content Distribution (130)	Packet Interval (max)	Duration
2	Bytes Average Popcount	Duration	Packet Interval (max)
3	Payload Bytes Content Distribution (84)	Packet Interval (mean)	Packet Interval (stddev)
4	Payload Bytes Content Distribution (32)	Packet Interval (min)	Packet Interval (mean)
5	Payload Bytes Content Distribution (47)	Packet Count (max)	Packet Interval (max)
6	Bytes Average Printable Count	Packet Count (stddev)	Packet Count (max)
7	Payload Bytes Content Distribution (1)	Packet Count (mean)	Duration
8	Payload Bytes Content Distribution (3)	Bytes Size	Packet Count (mean)
9	Packet Count (mean)	Packet Count (min)	Packet Count (stddev)
10	Payload Bytes Content Distribution (0)	Bytes Average Popcount	Packet Interval (mean)
11	Payload Bytes Content Distribution (37)	ACK Packet Count	Packet Interval (stddev)
12	Payload Bytes Content Distribution (69)	Packet Count (max)	Bytes Size
13	Packet Count (max)	Payload Bytes Content Distribution (3)	Payload Size in Bytes
14	Payload Bytes Content Distribution (254)	Payload Bytes Content Distribution (0)	Bytes Average Popcount
15	Payload Bytes Content Distribution (23)	Bytes Size	Packet Count (max)
16	Packet Count (stddev)	Packet Interval (stddev)	Packet Count
17	Packet Count (max)	Payload Bytes Content Distribution (23)	Packet Count (min)
18	Payload Bytes Content Distribution (2)	ACK Packet Count	ACK Packet Count
19	Packet Count (min)	PSH Packet Count	Packet Count (mean)
20	Payload Bytes Content Distribution (10)	Payload Size in Bytes	Packet Interval (min)

算法总结 DrillAgent

- 算法流程
  - 原始流量预处理与节点特征提取：提取时间、空间、内容、字节特征
  - 构建流量行为图：实际通信边与跨流连接边
  - 自监督图表示学习：节点特征掩码与节点特征重建
  - 全局决策修正：节点预测、图预测、图修正
- 算法优势
  - 融合多维侧信道特征，增强加密代理流量的表示能力
  - 利用多流连接关系，突破单流独立分类的局限
- 算法不足
  - 跨流关系依赖人工规则，图结构的稳定性和准确性有限
  - 图表示学习与分类目标分离，图级决策方式较为粗糙

EO-EPTC: End-to-End Original Traffic-Based Encrypted Proxy Traffic Classification Framework

EO-EPTC TIPO

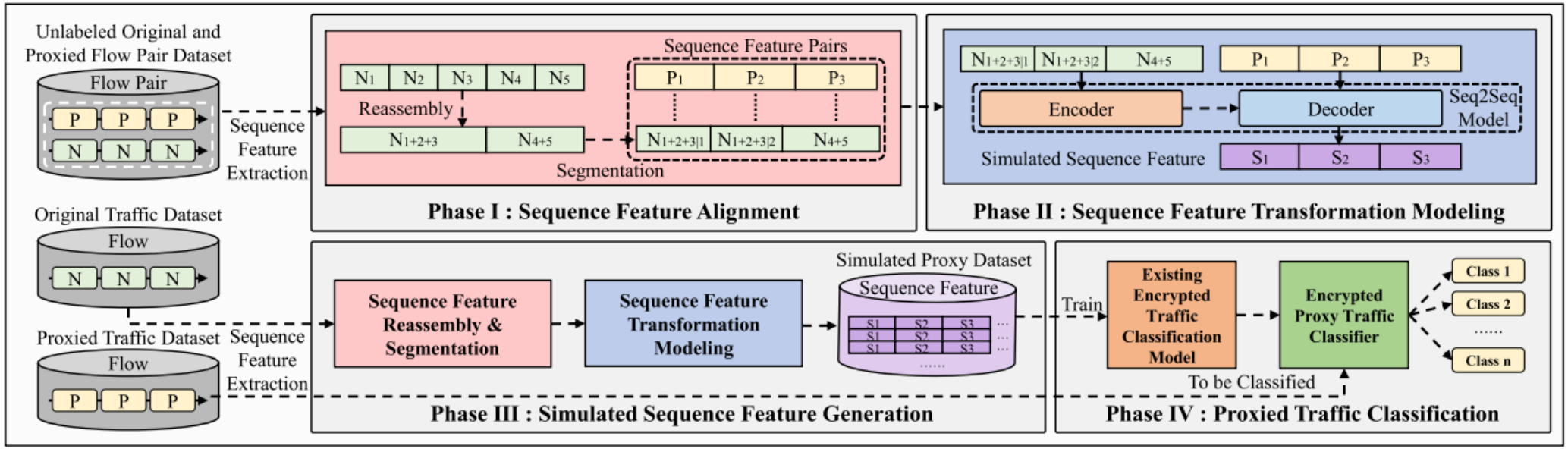
T	目标	利用原始流量，训练出可识别加密代理流量的分类器
I	输入	原始流量与对应的代理流量*1
P	处理	1. 从原始流量和代理流量中解析出长度序列 2. 预对齐原始流量和代理流量的序列特征对 3. 利用序列对训练Seq2Seq模型，学习代理协议底层语义 4.根据原始流量序列，利用Seq2Seq模型生成模拟序列特征 5.利用模拟序列特征训练加密流量分类器
O	输出	加密流量分类器*1
P	问题	1.原始流量与代理流量之间存在特征分布差异 2.分类模型对特定协议数据依赖严重
C	条件	原始流量与代理流量需要对应，且为会话形式
D	难点	1. 如何将原始流量序列与代理流量序列对齐 2. 如何弥合原始流量和代理流量之间的分布差距
L	水平	2026 CCF-A

EO-EPTC 研究问题

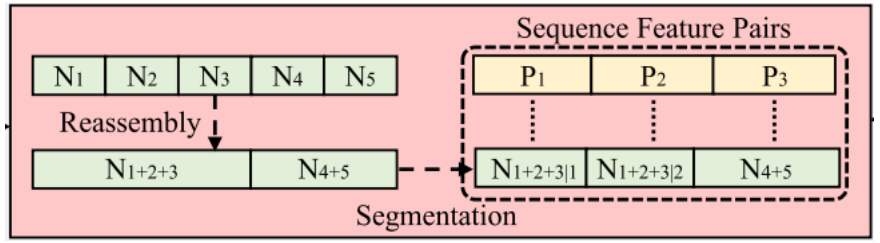
- 两个问题
  - 特征分布存在差异
    - 应用问题：原始流量与代理流量的特征分布不一致，使基于原始流量训练的分类器难以直接部署于代理流量识别场景，分类性能显著下降
    - 科学问题：代理协议封装与网络传输扰动共同造成原始流量和代理流量在序列边界、数值偏移、分组结构及位置关系等方面的差异
  - 分类模型对特定协议数据严重依赖
    - 应用问题：现有方法需要针对不同代理协议、网站类别和运行环境反复采集带标签代理流量，导致数据构建和模型更新成本高，难以适应代理协议快速演化
    - 科学问题：带标签原始流量和无标签原始-代理配对流量提供的监督信息不一致，现有方法难以将原始流量的类别信息迁移到不同代理流量

EO-EPTC 算法原理图

- 思路分析
  - 序列特征对齐：通过序列重组和分割，减少TCP分段、MTU等造成的序列差异
  - 序列特征转换建模：学习“原始流量特征→代理流量特征”的转换关系
  - 模拟序列特征生成：将带标签原始流量转换为带标签的模拟代理流量

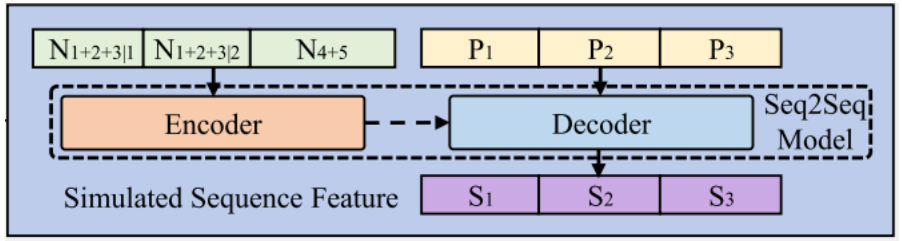


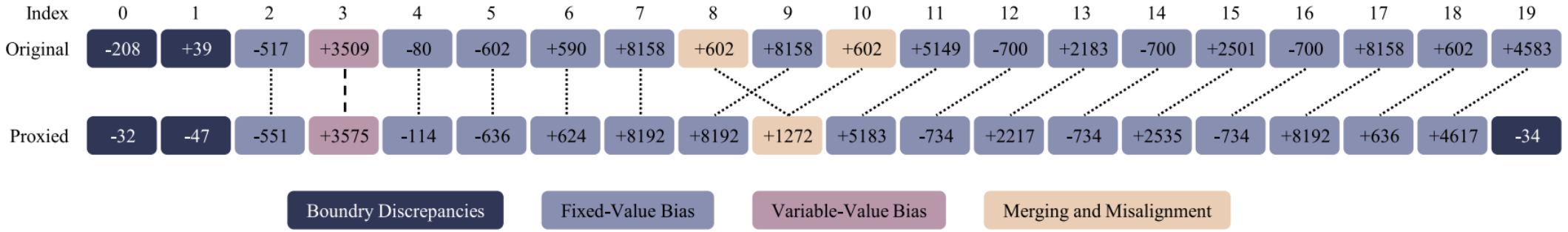
- 序列特征对齐
  - 提取TCP和UDP流量的数据包Payload长度序列作为序列特征，使用“+”表示客户端到服务器（C2S），使用“-”表示服务器到客户端（S2C）
  - 重组
    - 输入：会话的数据包序列$\boldsymbol{P} = \{p_1, p_2, \dots, p_n\}$
    - 处理：按同方向连续累加TCP分段长度；若遇到Push标志，结束当前数据单元，将累计长度写入新序列并重置该方向累加；清零后继续处理后续数据包
    - 输出：重组序列$\boldsymbol{S}$
  - 分割
    - 输入：重组后的所有序列$\boldsymbol{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_m\}$
    - 处理：确定分割阈值τ；对重组后的每个序列元素s_i，如果$|s_i| \leq \tau$直接保留；如果$|s_i| > \tau$则按阈值τ拆分；分割时保留原来的方向符号，最后不足τ的余数单独保留
    - 输出：分割后的序列$\boldsymbol{S}'$



算法原理　序列特征转换建模

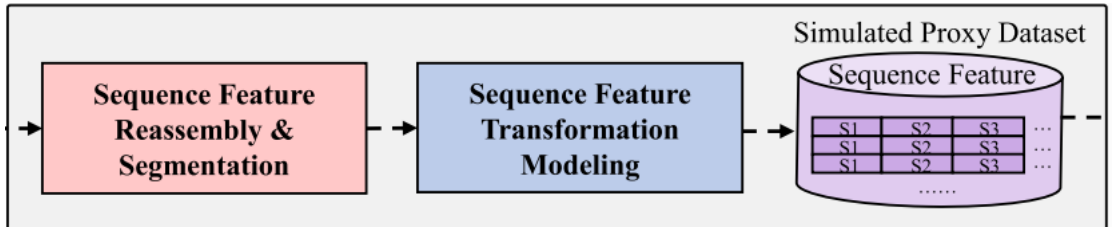
- 序列特征转换建模
  - 封装机制造成的序列差异：边界差异、固定值偏差、可变值偏差、合并和错位
  - 输入：原始-代理流量对$\mathcal{D}_{pair} = \{(\boldsymbol{S}_i^{ori}, \boldsymbol{S}_i^{proxy})\}_{i=1}^{N}$
  - 处理：
    - 特征嵌入与位置编码$\boldsymbol{Z}_i^{ori} = Embedding(\boldsymbol{S}_i^{ori})$
    - Transformer Encoder提取原始流量的序列结构，输出中间表示$\boldsymbol{H}_i = Encoder(\boldsymbol{Z}_i^{ori})$
    - Transformer Decoder根据编码器的中间表示，生成代理流量序列$\widehat{\boldsymbol{S}}_i^{proxy} = Decoder(\boldsymbol{H}_i)$
    - 损失计算：$Loss = MaskedCE(\widehat{\boldsymbol{S}}_i^{proxy}, \boldsymbol{S}_i^{proxy})$
  - 输出：训练好的Seq2Seq模型



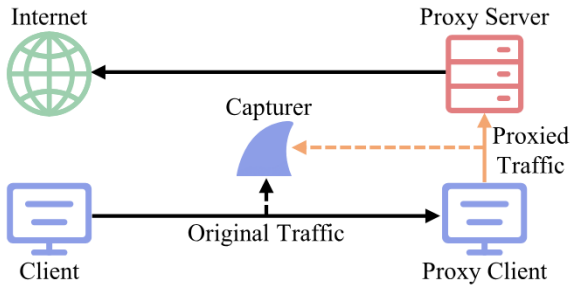


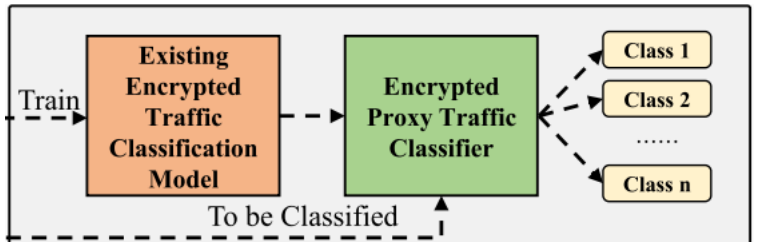
算法原理 模拟序列特征生成

- 模拟序列特征生成
  - 输入：带标签的原始流量、训练好的Seq2Seq模型
  - 处理：
    - 序列特征对齐
    - 序列特征转换建模
  - 输出：带标签的模拟代理流量数据集



- 代理流量分类
  - 利用生成的模拟代理流量数据集训练现有加密流量分类器
  - 利用训练好的加密流量分类器识别真实代理流量





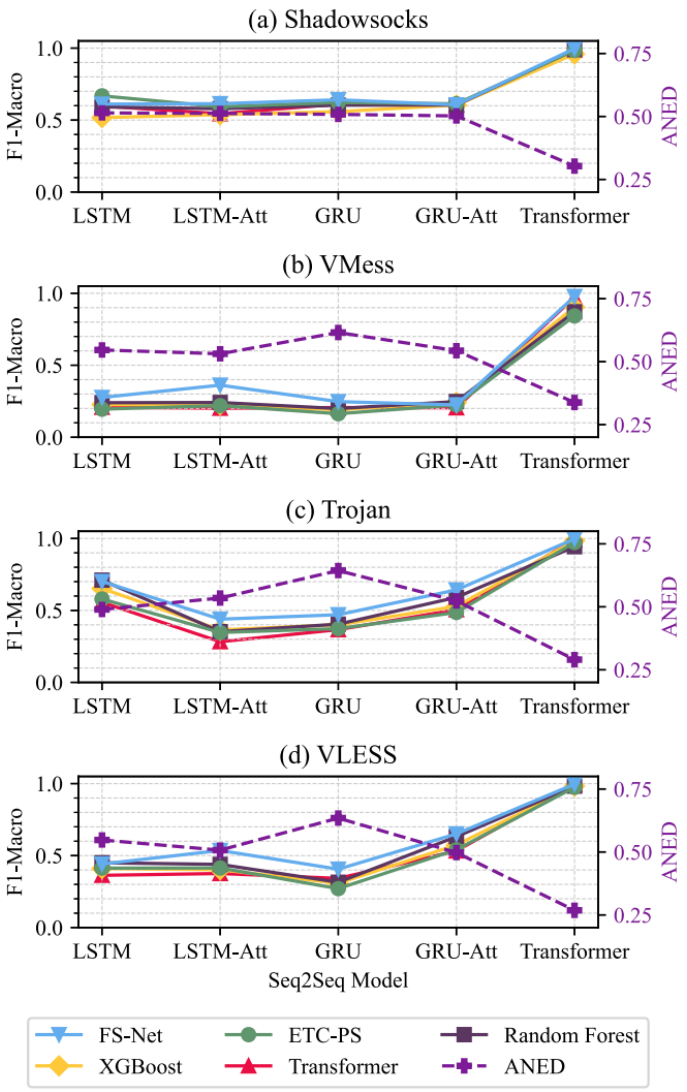
- 数据集
  - 4类代理协议
  - 60个网站

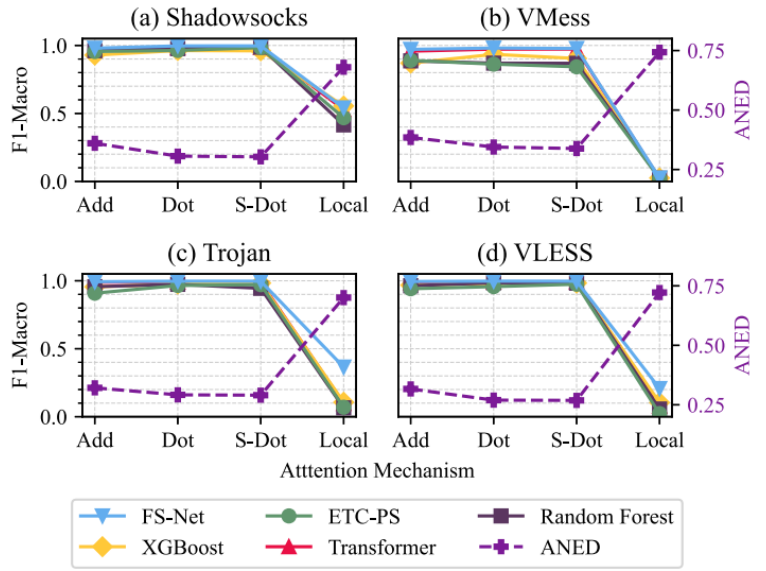
Proxy Protocol	# of Webs	# of Flow Pairs	# of Original Packets	# of Proxied Packets	File Size
SS	60	1,924,994	129,188,898	143,144,593	417.43GB
VMess	60	1,447,650	116,895,522	36,831,908	466.85GB
Trojan	60	1,497,140	119,277,887	38,063,133	457.47GB
VLESS	60	1,504,976	124,366,713	37,862,431	427.98GB

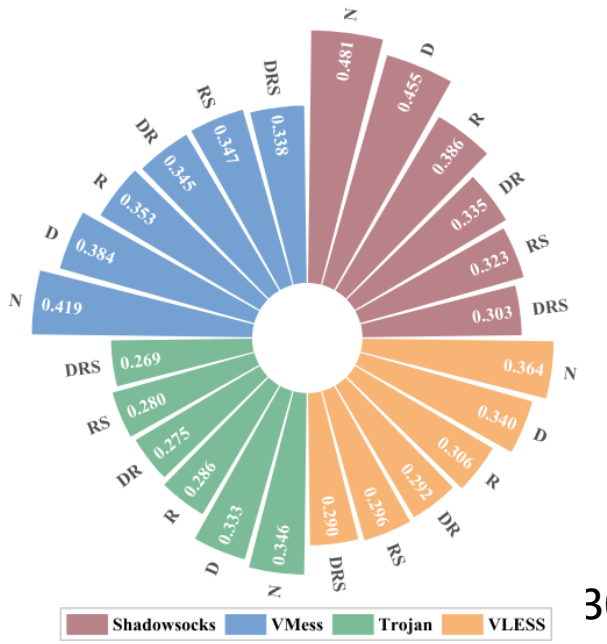
- 评价指标
  - $F1_{macro}$：衡量生成数据训练出的分类器是否有效
  - $Average\ Normalized\ Edit\ Distance(ANED)$：
    - $ANED = \frac{1}{M}\sum_{i=1}^{M} \frac{ED(\hat{S}_i^{proxy}, S_i^{real-proxy})}{\max(|\hat{S}_i^{proxy}|, |S_i^{real-proxy}|)}$
    - 衡量生成序列与真实序列的相似性
- 加密流量分类器
  - FS-Net、ETC-PS、Random Forest、XGBoost、Transformer
- 基线方法
  - Flowpic、Flowformers、mini-FlowPic

实验设计　对比实验

- Seq2Seq基模型对比
  - Transformer
- 注意力机制对比
  - Scaled Dot-Product Attention
- 对齐特征对比
  - 三种处理：
    - 方向特征编码
    - 序列特征重组
    - 序列特征分割
  - 方向编码＋重组＋分割(DRS)



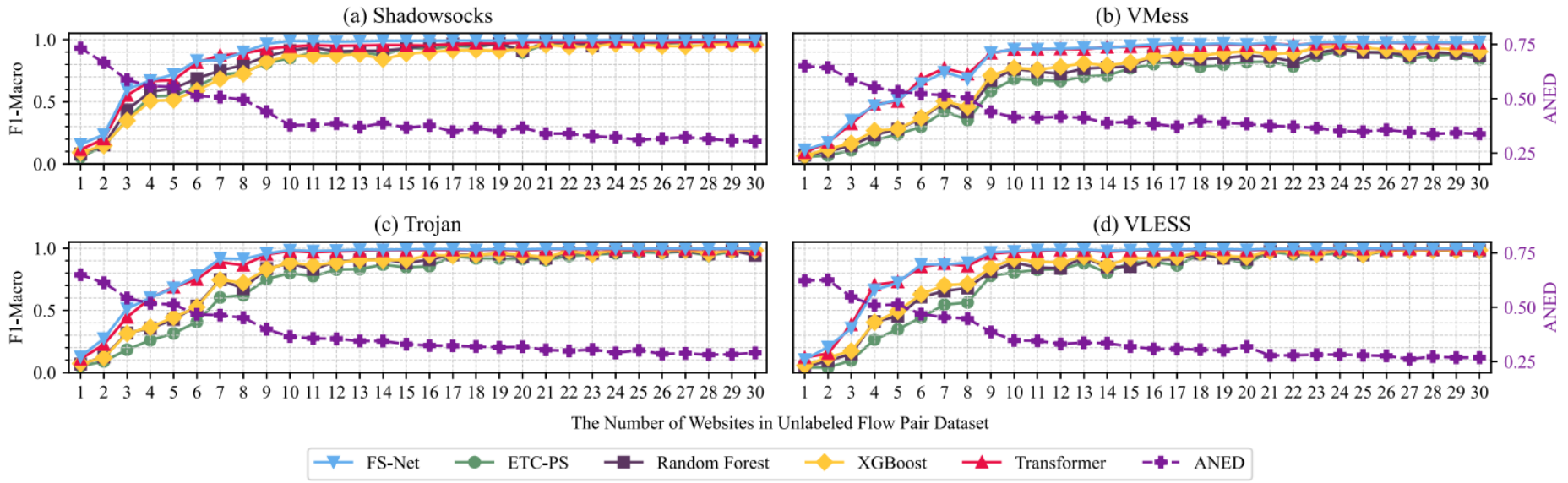




实验设计　性能评估

- 数据集规模影响
  - 超过26个网站后性能趋于稳定
- 基线方法比较
  - 使用原始流量训练，再在真实代理流量上测试
  - EO-EPTC直接学习原始序列到代理序列的转换关系，缓解原始流量与代理流量之间的分布差异

Approach	SS	VMess	Trojan	VLESS
FlowPic	8.39%	8.31%	8.46%	8.74%
Flowformers	7.61%	6.39%	7.69%	8.51%
mini-FlowPic	17.99%	26.94%	15.69%	25.22%
FS-Net+EO-EPTC	99.70%	97.75%	99.67%	99.67%



算法总结 EO-EPTC

- 算法流程
  - 序列特征提取与对齐：缓解传输造成的分布差异
  - 训练序列特征转换模型：缓解协议封装造成的分布差异
  - 生成模拟代理流量特征：生成带标签代理流量，与现有分类器兼容
  - 训练和部署分类器
- 算法优势
  - 有效缩小特征分布差异
  - 减少带标签代理流量依赖
- 算法不足
  - 不同代理协议需要单独适配，未实现跨协议识别
  - 只使用有效载荷长度序列，未使用其他侧信道信息

特点总结与未来展望

特点总结与未来展望

- **Proxykiller**
  - 从时间、空间、内容和字节等多个维度刻画代理流量
  - 利用多条流共同形成的连接行为模式，降低孤立流信息不足所造成的误判
- **EO-EPTC**
  - 有效缓解原始流量与代理流量之间的分布差异
  - 使用少量无标签配对数据学习代理转换规律，再把大量带标签原始流量转换为模拟代理训练数据
- 未来发展
  - 协议版本细粒度识别
  - 对抗混淆条件下的鲁棒协议识别

参考文献

[1] Hongbo Xu, Zhenyu Cheng, Shuhao Li, et al. ProxyKiller: An Anonymous Proxy Traffic Attack Model Based on Traffic Behavior Graphs[C]. ESORICS, Cham, 2024:162-181.

[2] Yige Chen, Huajie Jia, Zhenzhou Tang, et al. EO-EPTC: End-to-End Original Traffic-Based Encrypted Proxy Traffic Classification Framework[J]. IEEE Transactions on Information Forensics and Security, 2026, 21:490-505.

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！