Enhancing Network Security: NIDS based on Reconstruction Graph Network on PyQT platform

夏鸿睿、谭思远、翁正、张志胜、黄芃洋、蒲俊翰、杨雅雯、魏瑞敏、林锦蓥、王惟

September 2023

摘要

网络流量加密在保护企业数据和用户隐私的同时，也为恶意流量检测带来新的挑战。本文首先分析了 TLS 协议的特点以及在网络流量加密时的应用，然后在基于图卷积神经网络的基础上，创新使用了以时间为规则重构图的思路，给出了一种基于监督学习的恶意加密流量检测系统。该系统搭载在 PyQT 的基础上，使用 socket 进行文件传输，并将模型搭载在服务器端，解决不同主机算力的差异问题。

目录

1 TLS 协议分析

1.1 来历

SSL/TLS 是一种密码通信框架，他是世界上使用最广泛的密码通信方法。SSL/TLS 综合运用了密码学中的对称密码，消息认证码，公钥密码，数字签名，伪随机数生成器等，可以说是密码学中的集大成者。

SSL(Secure Socket Layer) 安全套接层，是 1994 年由 Netscape 公司设计的一套协议，并与 1995 年发布了 3.0 版本。

TLS(Transport Layer Security) 传输层安全是 IETF 在 SSL3.0 基础上设计的协议，实际上相当于 SSL 的后续版本。

1.2 底层框架

TLS 主要分为两层，底层的是 TLS 记录协议，主要负责使用对称密码对消息进行加密。

上层的是 TLS 握手协议，主要分为握手协议，密码规格变更协议，警告协议和应用数据协议 4 个部分。

握手协议负责在客户端和服务器端商定密码算法和共享密钥，包括证书认证，是 4 个协议中最最复杂的部分。密码规格变更协议负责向通信对象传达变更密码方式的信号警告协议负责在发生错误的时候将错误传达给对方应用数据协议负责将 TLS 承载的应用数据传达给通信对象的协议。[1]

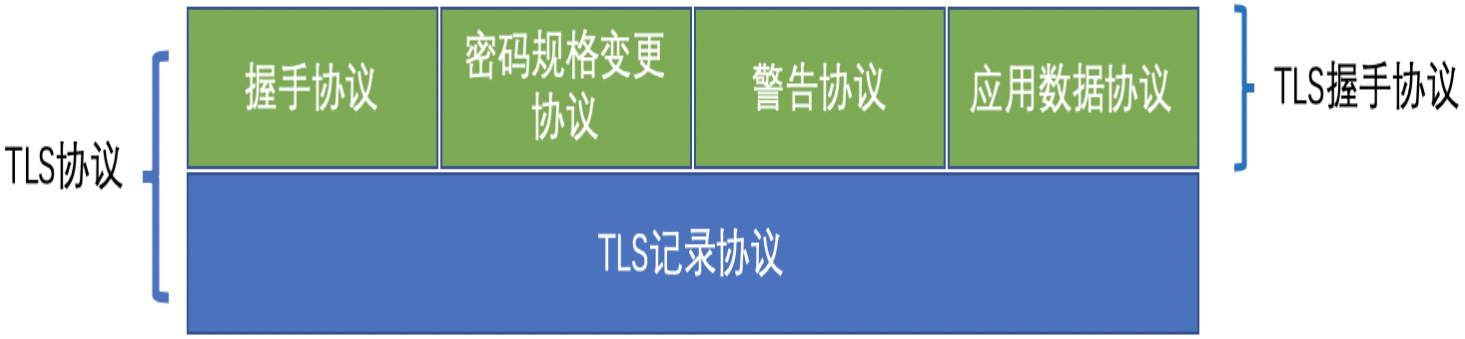


图 1: TLS 框架图

1.3 TLS 恶意流量特征分析

恶意加密流量的特征一般分为以下 3 类：内容特征、数据流统计特征和网络连接行为特征[2]。针对采用 TLS 协议的恶意加密流量，本次实验从数据元统计特征、上下文数据特征 2 个方面来分析其特征要素。

数据元统计特征：恶意流量与良性流量的统计特征差别主要表现在数据包的大小、到达时间序列和字节分布。数据包的长受 UDP、TCP 或者 ICMP 协议中数据包的有效载荷大小影响，如果数据包不属于以上协议，则被设置为 IP 数据包的大小。因到达时间以毫秒分隔，故数据包长度和到达时间序列，可以模拟为马尔科夫链，构成马尔科夫状态转移矩阵，从而统计分析数据包在时序上的特征。

上下文统计数据：上下文数据包括 HTTP 数据和 DNS 数据。过滤掉 TLS 流中的加密部分，可以得到 HTTP 流，具体包括出入站的 HTTP 字段、Content-Type、User-Agent、Accept-Language、Server、HTTP 响应码。DNS 数据包括 DNS 响应中域名的长度、数字以及非数字字符的长度、TTL 值、DNS 响应返回的 IP 地址数、域名在 Alexa 中的排名。

1.4 加密木马通信行为特征

HTTPS 隐蔽隧道利用 SSL(Secure Sockets Layer) 和 TLS(Transport Layer Security) 完成了对 HTTP 通信过程的加密。为了更好地检测加密木马，需要了解展现出与木马加密相关的网络行为特征，分析如下：

1、高下行小包数占比。木马交互过程中，控制端向客户端发送的大多数指令为操作指令，其载荷较小，故该行为具有高下行小包数占比。

[1]参考文献：骆子铭, 许书彬, 刘晓东. 基于机器学习的 TLS 恶意加密流量检测方案 [J]. 网络与信息安全学报, 2020,6(1): 77-83.

[2]参考文献：鲁刚, 郭荣华, 周颖, 等. 恶意流量特征提取综述 [J]. 信息网络安全, 2018, 213(9): 7-15.

2、非正常上下行大包数比。正常网络通信与木马通信的上下行流量比反差很大，在常规网络应用中，一般由内网主机发出资源请求命令，外部服务器响应命令后会回传多种多媒体资源。然而，内网主机被攻击者植入木马后，木马控制端发出指令会使得内网主机上传多类有价值信息。

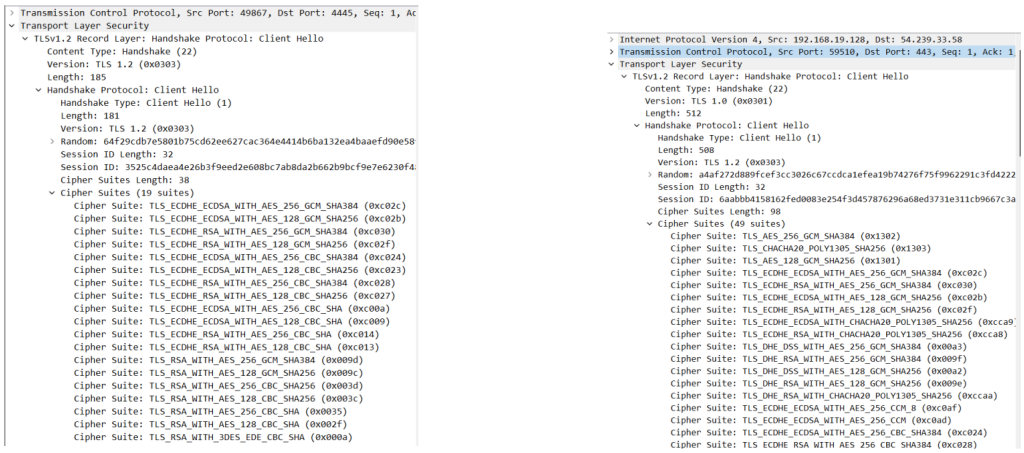
3、高上行数据包数比。为响应控制指令，被控端常上传大量数据信息使其具有高上行数据包数比。

4、高 PUSH 包比例。PUSH 是 TCP 报头中的标志位，若 PUSH 置为 1 则表示接收方尽快将数据包交付给接受应用进程。由于木马通信优先于其他进程发送和接受数据，其 PUSH 包占比高。

基于 HTTPS 的加密木马在通信过程中会尽可能模仿正常 HTTPS 通信，如使用 443 端口、构造私有证书握手和列举大量加密套件等，使现有 IDS 系统难以有效识别木马流量。通过分析公开的 8 种基于 HTTPS 的木马，重点关注以下两个方面:

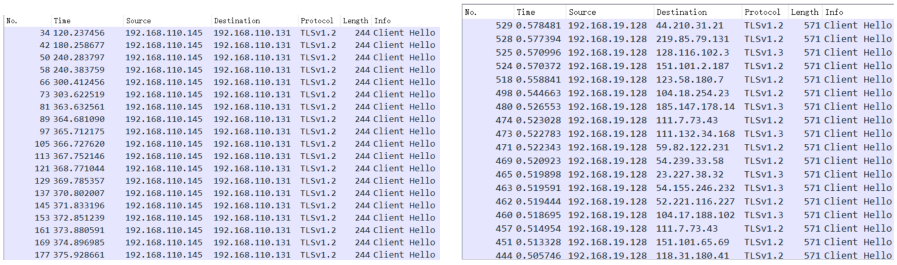
1、HTTPS 元数据交互阶段

此阶段一切报文载荷都是明文，通过判断 Compression Methods Length、Certificates Length、Cipher Suites 和 Extensions Length 等字段偏离情况的统计规律来进行网络行为的分析。下面以其中一种木马为例与正常应用进行比对分析。



(a) 木马 Cipher Suites　　(b) 正常 Cipher Suites

图 2: Cipher Suites 对比图



(a) 木马 Client Hello　　(b) 正常 Client Hello

图 3: Client Hello 对比图

首先，分析 Client Hello 字段的报文，可看出木马的报文长度明显比正常网络应用短。图 2 展示了木马的报文长度和正常的报文长度。图 3 展示了实际案例中木马和正常网络应用的 ClientHello 包。分析发现，木马 Client Hello 报文中 Cipher Suites 数量明显比正常网络应用少，扩展字段长度也比正常网络应用短。此外，木马通信的被控端和控制端之间没有合法的证书链，证书没有任何递归结构和签发机构，同时未经根证书机构认证。而正常网络流量中，Certificate 报文具有明显的多层递归结构，且很容易获取常见的根证书库，如 Microsoft、Mozilla、Apple、Chrome 等。为佐证上述分析，图 4 对 HTTPS 木马和正常网络应用在元数据交互阶段的关键字段载荷长度进行了统计。

2、HTTPS 加密应用数据交互阶段

此阶段，根据数据流的时序性特征，对木马控制端和被控端的交互时序和对正常控制端和被控端的交互时序分析可得图 8。正常应用具有内部控制特性，会话流量交互序列一般始于 1 个上行包;

	Length	Cipher Suites	Key Info	Signature Algorithms
Cobal Strike 木马	181	19	None	12
Htbot木马	205	20	None	8
SVChost木马	175	21	None	12
主机正常上网浏览	508	17	有	18
爬AlexaTop生成	512	49	有	20

图 4: TLS 字段统计表

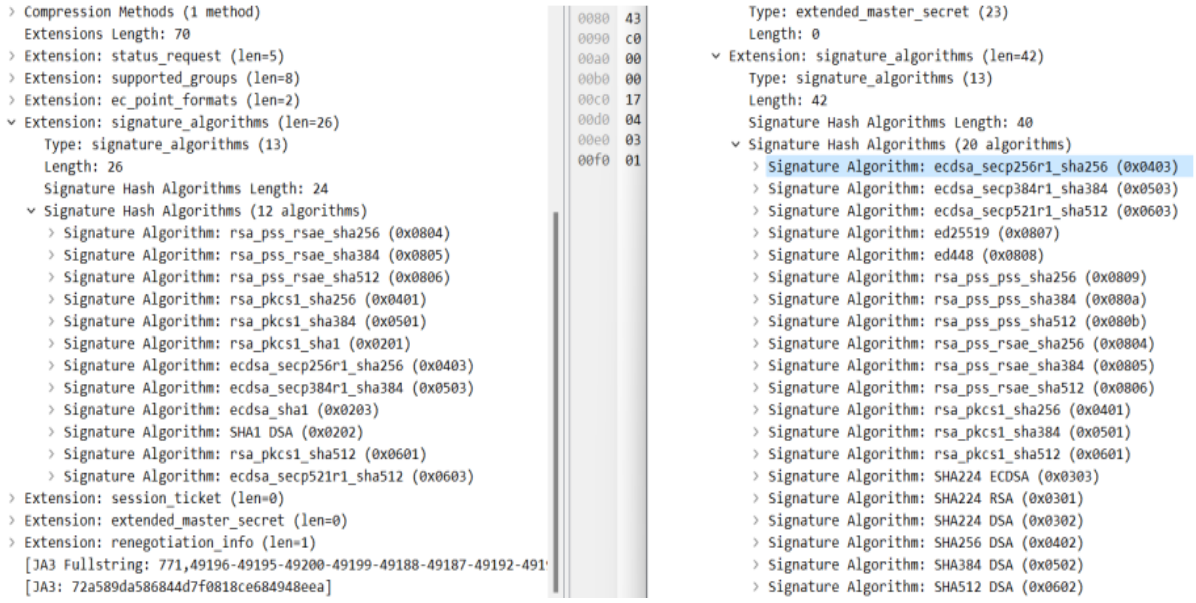


图 5: 关于签名算法 (左木马，右正常)

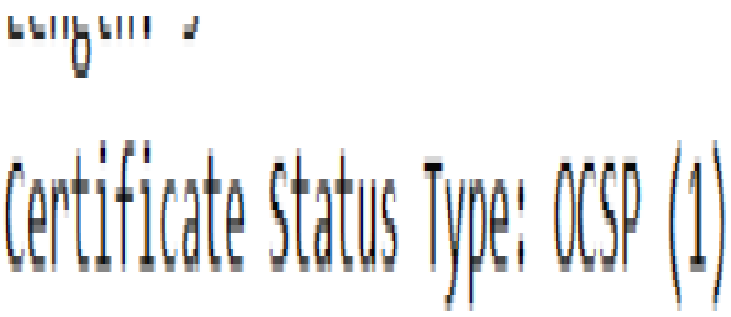
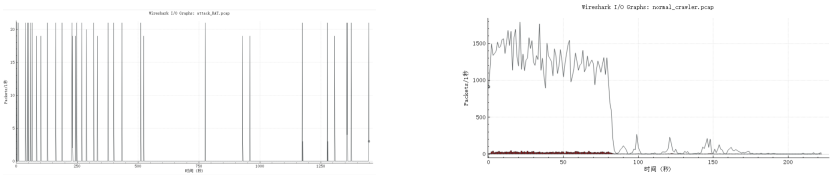


图 6: 木马的证书状态：OCSP（在线证书）

```
▾ Extension: psk_key_exchange_modes (len=2)
    Type: psk_key_exchange_modes (45)
    Length: 2
    PSK Key Exchange Modes Length: 1
    PSK Key Exchange Mode: PSK with (EC)DHE key establishment (psk_dhe_k
▾ Extension: key_share (len=38)
    Type: key_share (51)
    Length: 38
  ▾ Key Share extension
      Client Key Share Length: 36
    ▾ Key Share Entry: Group: x25519, Key Exchange length: 32
        Group: x25519 (29)
        Key Exchange Length: 32
        Key Exchange: bd8825fb2d686c32716bb02c7ac30edd8ed073de2a9cea1fb
▾ Extension: padding (len=173)
    Type: padding (21)
    Length: 173
    Padding Data: 000000000000000000000000000000000000000000000000000000
```

图 7: 正常的密钥信息，密码无)



(a) 木马交互操作间隔图　　(b) 正常交互操作间隔图

图 8: 交互操作对比图

木马行为则具有外部控制特性，会话流量交互序列一般始于 1 个下行包。通常的数据流中并无人为操作，但木马通信中会引入人的操作行为。这种人为操作需要一定的思考时间和随机因素，因而往往会诱发随机性的密集交互和时间间隔。系统可以通过获取适当的思考时间，以及滤出木马自动发送的心跳，从而利用密集交互的首个数据包判别控制方的来源。木马程序建立的网络通信，连接通常具有内网发起和外网控制的特征。然而，木马交互操作和正常交互操作在一定时间间隔内存在一定的相似性，所以此种特征无法作为主要判断依据。

2　模型框架分析

2.1　GAT 模型分析

GAT(Graph Attention Networks) 去解决 GCN(Graph Convolutional Network) 存在的问题并且在不少的任务上都取得了 **state of art** 的效果

2.1.1　GCN 的局限性

GCN 是处理 transductive 任务的一把利器（transductive 任务是指：训练阶段与测试阶段都基于同样的图结构），然而 GCN 有两大局限性是经常被诟病的：

1、无法完成 inductive 任务，即处理动态图问题。inductive 任务是指：训练阶段与测试阶段需要处理的 graph 不同。通常是训练阶段只是在子图（subgraph）上进行，测试阶段需要处理未知的顶点（unseen node）。

2、处理有向图的瓶颈，不容易实现分配不同的学习权重给不同的 **neighbor**。

2.1.2　图 (Graph) 数据结构特征

Graph 数据结构包含着边和点的关系，研究目标聚焦在顶点之上，边诉说着顶点之间的关系。对于任意一个顶点

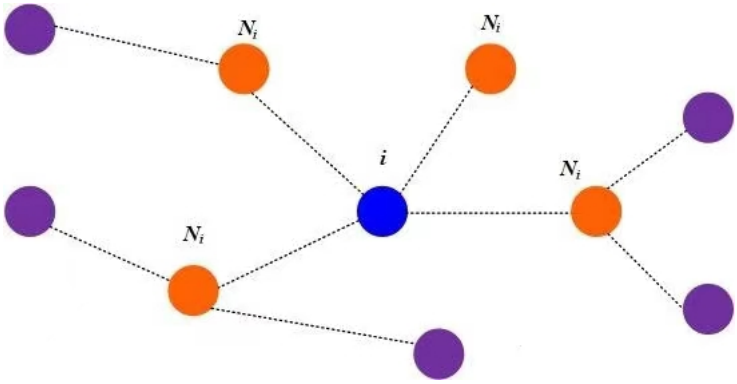


图 9: Graph 数据结构图

i，它在图上邻居 N_i 构成一种特征，即图的结构关系。除了图的结构之外，每个顶点还有自己的特征 h_i（通常是一个高维向量）

2.1.3 Mask graph attention or global graph attention

GAT 本质上可以有两种运算方式（本次实验采用了第 2 种运算方式）：

1、Mask graph attention

每一个顶点都对于图上任意顶点 i 都进行 attention 运算。可以理解为图 2 的蓝色顶点对于其余全部顶点进行一遍运算。

2、global graph attention

注意力机制的运算只在邻居顶点上进行，也就是说图 2 的蓝色顶点只计算和橙色顶点的注意力系数。a

2.2 创新点：重构图

本文针对所使用的模型和数据特征，对 graph 数据结构提出了一种重新构建的思路，即将边分类任务转化为点分类任务。讨论了 3 种重构图的方法：

1、在二维平面下，以距离关系为规则进行重构

2、以特定概率比例为规则进行重构

3、以时间顺序为规则进行重构

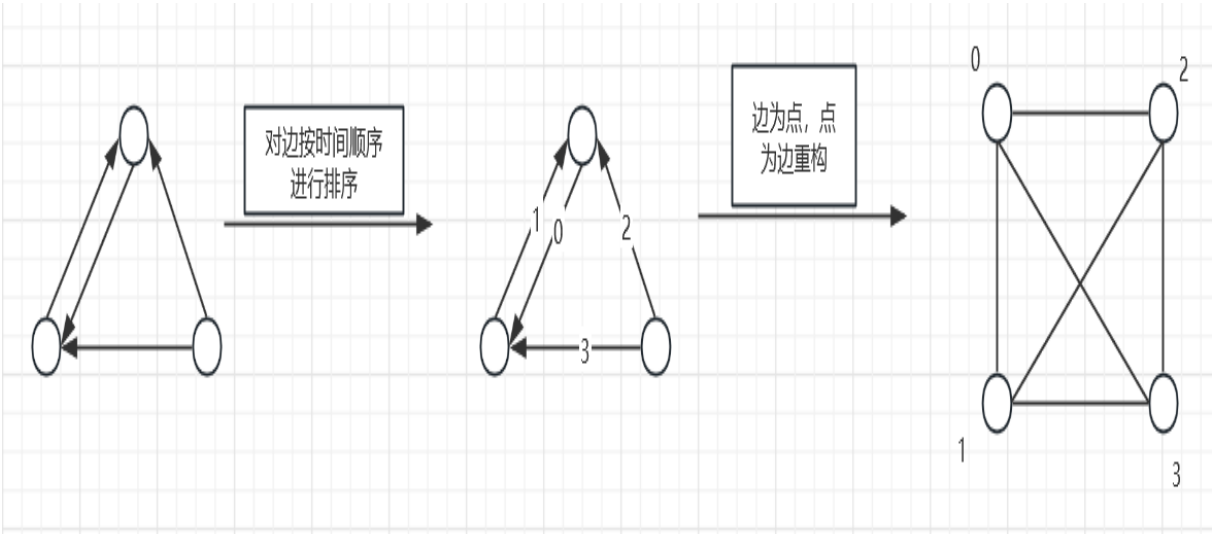
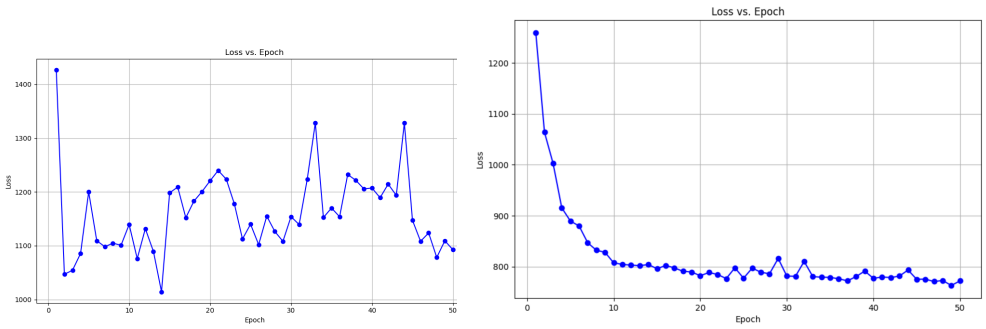


图 10: Reconstruction Graph

在经过画图、以及模型运行结果 (图 4、图 5) 来看，最终选择了第 3 种，以时间顺序为规则对图结构进行重构（将边和点关系重新打乱拼接）



(a) Before Reconstruction (b) After Reconstruction

图 11: Reconstruction 对比图

2.3 层级分析

本次实验采用稀疏连接降低模型参数量的同时, 保证了计算资源的使用效率。模块结构如图 6 所示, 该结构包含 2 个 74X1 的卷积层和 1 个全连接层, 输入图像进入该模块进行卷积或线性操作, 得到 13 种分类行为, 随后在通道维度上相并得到该模块的输出。每个卷积层和全连接层均有可训练的参数, 为恶意流量检测奠定了基础。

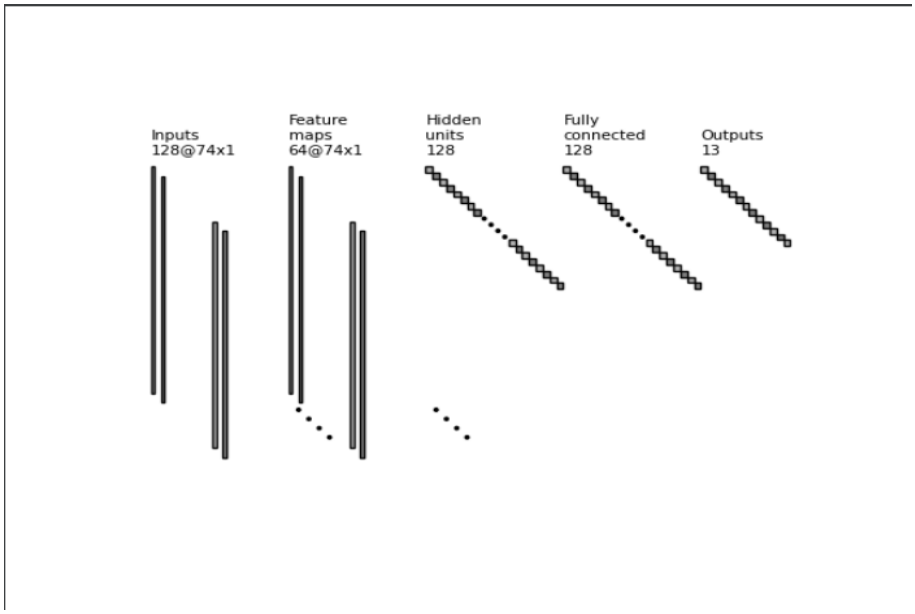


图 12: pool

3 恶意流量数据包抓取分析

3.1 网络流量数据包抓取

首先利用爬虫文件，对常用的 5 万个网站的正常浏览的网络流量进行了快速的抓取，同时，利用另一个主机利用远程木马 (图 13) 对 Rat 攻击、Botnet 攻击等进行了复现，利用 wireshark 对攻击时的网络流量进行了抓取，获得 pcap 包数据集。

获得初步数据集后，先利用源 ip 和目的 ip 对抓取的数据进行过滤，再根据 TLS 协议特征过滤，获得加密恶意流量数据集。

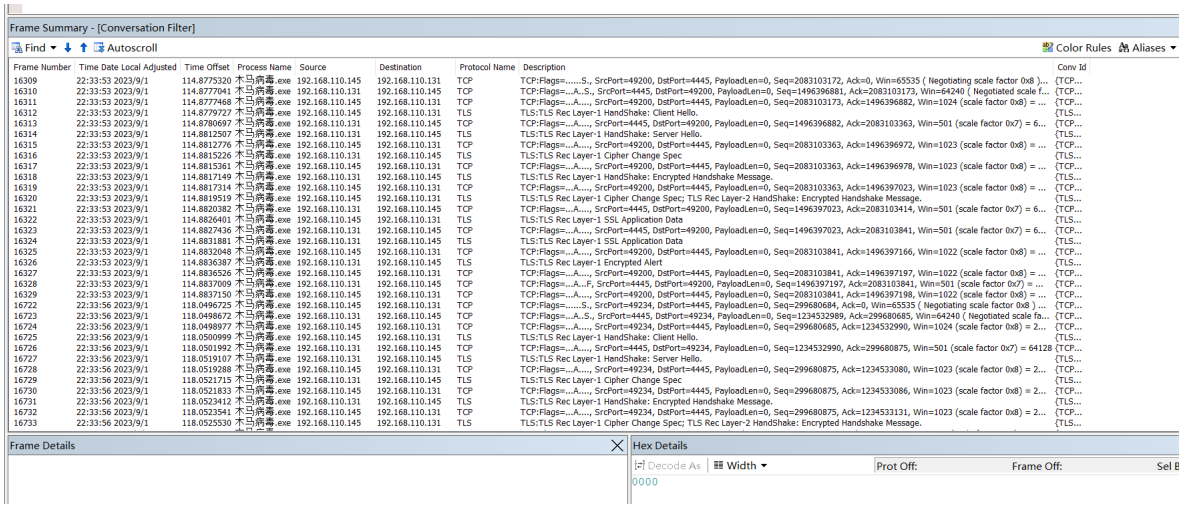


图 13: 木马攻击

3.2 pcap 包转化成 CSV 数据集文件

本次实验利用了 **CICFLOWMETER** 工具实现由 pcap 包向 CSV 数据集文件的转化[3]。

3.2.1 选择 CICFLOWMETER 的原因

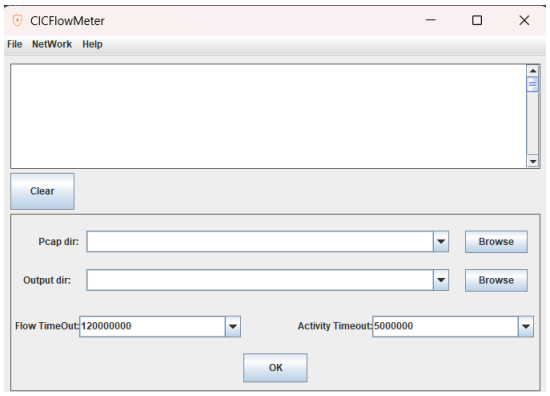


图 14: CICFLOWMETER

CICFLOWMETER 的优点：

灵活性：Cicflowmeter 可以适用于多种网络环境和流量类型。它支持处理各种协议，如 TCP、UDP、ICMP 等，并能够分析不同应用层协议的流量。

高效性：Cicflowmeter 使用了一些高效的算法和数据结构来进行流量分析和特征提取。它可以处理大规模的网络流量数据，并在短时间内提供准确的结果。

准确性：Cicflowmeter 采用了多种机器学习和数据挖掘技术来检测和识别通信。它通过分析流量中的特征和行为模式，能够有效地区分正常流量与恶意流量。

易于使用：Cicflowmeter 提供了简单易用的命令行界面和 API 接口，使得用户能够方便地集成它到自己的网络安全系统中。同时，它还提供了详细的文档和示例代码，帮助用户快速上手并解决问题。

跨平台性：可在 windows、Linux 等多平台使用，适合各种开发场景。

根据对加密木马通信行为特征的分析以及对 TLS 加密协议的分析，结合本次实验的开发环境，我们可以发现，加密木马通信行为在侧信道特征以及可提取明文特征中有着较为明显的异常特征，于是，我们使用了 CICFLOWMETER 从**众多特征中提取了 74 个关键特征**，以满足训练、检测、部署的需求。

[3]参考链接:https://github.com/datthinh1801/cicflowmeter

3.2.2 转化原理

从 pcap 文件中逐个读取 packet，将每个数据包添加到对应的流中在 currentFlows 存储当前还未结束得所有 TCP、UDP 流。在添加的过程中不断地更新每个流的统计特征，最终将统计特征写入 csv 文件。判断新加入的数据包是否属于当前所有未结束的流，如果属于当前流则判断正向还是反向，之后判断时间是否超时、不超时则判断是否含有 FIN 标志，如果两者都不满足，则声明一个 BasicFlow 对象，根据 id 从 currentFlows 中拿到与当前数据包对应的流，调用 addPacket 将该数据包加入到对应流中。如果前面判断不在当前所有未结束的流中，则直接创建一个新得流，里面只含当前数据包，存入到 currentFlows 中。如果属于当前某个未结束的流，且超时或存在 FIN 标志，则说明当前 flow 结束，超时则从 currentFlows 中移除对应流，新建 flow 存入 currentFlows 中，含 FIN 标志则直接从 currentFlows 中移除对应流。结束的 flow 直接调用 onFlowGenerated 函数将流打印存储起来。

在后续调试过程中，发现该工具中存在一处 bug（异常），通过 debug 定位，成功将代码修改正确。

```
        else:
            if self.feature.backward_bulk_count != 0:
                return (
                    self.feature.backward_bulk_size / self.feature.backward_bulk_count
                )
        return 0

    def get_packets_per_bulk(self, packet_direction):
        if packet_direction == PacketDirection.FORWARD:
            if self.feature.forward_bulk_count != 0:
                return (
                    self.feature.forward_bulk_packet_count
                    / self.feature.forward_bulk_count
                )
        else:
            if self.feature.backward_bulk_count != 0:
                return (
                    self.feature.backward_bulk_packet_count
                    / self.feature.backward_bulk_count
                )
        return 0

    def get_bulk_rate(self, packet_direction):
        if packet_direction == PacketDirection.FORWARD:
            if self.feature.forward_bulk_count != 0:
                return (
                    self.feature.forward_bulk_size / self.feature.forward_bulk_duration
                )
        else:
            if self.feature.backward_bulk_count != 0:
                return (
                    self.feature.backward_bulk_size
                    / self.feature.backward_bulk_duration
                )
        return 0
```

图 15: BUG

4 项目运行结构

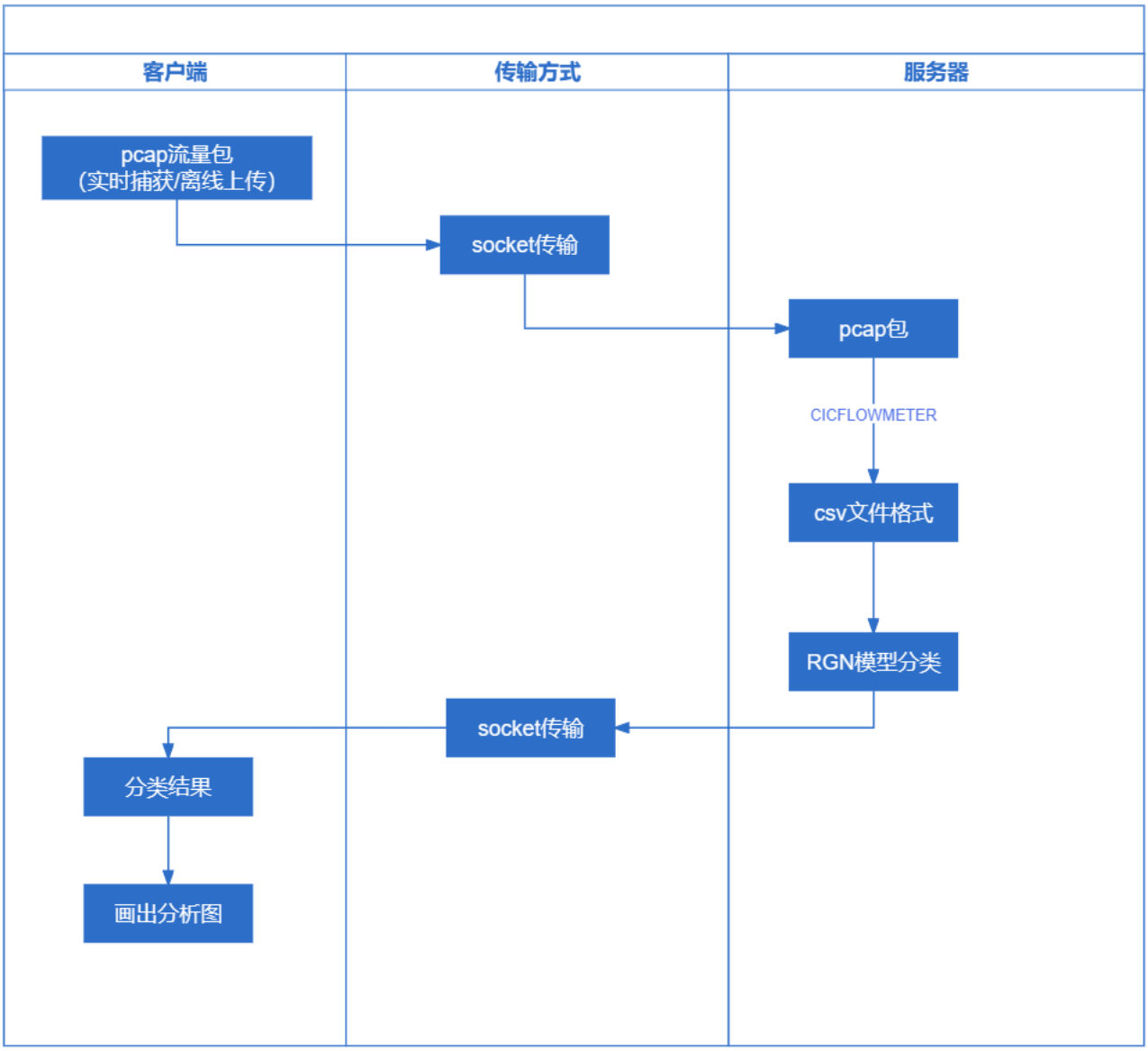


图 16: Process of Project

参考文献

[1] 鲁刚, 郭荣华, 周颖, 等. 恶意流量特征提取综述 [J]. 信息网络安全, 2018, 213(9): 7-15.

[2] 骆子铭, 许书彬, 刘晓东. 基于机器学习的 TLS 恶意加密流量检测方案 [J]. 网络与信息安全学报, 2020,6(1): 77-83.

[3] 谷勇浩, 徐昊, 张晓青. 基于多粒度表征学习的加密恶意流量检测 [J/OL]. 计算机学报:1-12[2023-09-07].http://kns.cnki.net/kcms/detail/11.1826.tp.20230421.1719.020.html

[4] Wang W, Zhu M, Wang J, et al. End-to-end encrypted traffic classification with one-dimensional convolution neural networks[C]//2017 IEEE international conference on intelligence and security informatics (ISI). IEEE, 2017: 43-48.

[5] Li Y, Guo H, Hou J, et al. A Survey of Encrypted Malicious Traffic Detection[C]//2021 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI). IEEE, 2021: 1-7.

[6] Li M, Song X, Zhao J, et al. TCMal: A Hybrid Deep Learning Model for Encrypted Malicious Traffic Classification[C]//2022 IEEE 8th International Conference on Computer and Communications (ICCC). IEEE, 2022: 1634-1640.