分 类 号：TP391
学　　号：202121100438

济南大学
UNIVERSITY OF JINAN

硕 士 学 位 论 文

（全 日 制 研 究 生）

基于域名解析行为分析的网络恶意行为检测

作　　者　　李辉
指 导 教 师　　彭立志
合 作 导 师　　潘泉波

学 科 名 称　　计算机科学与技术
学 位 类 别　　工学硕士
答 辩 时 间　　2024 年 6 月 1 日

Network Malicious Behavior Detection Based on Domain Name Resolution Behavior Analysis

By

LI Hui

Under the Supervision of

PENG Li Zhi

A Thesis Submitted to the University of Jinan

In Partial Fulfillment of the Requirements

For the Degree of Master of Engineering

University of Jinan

Jinan, Shandong, P. R. China

June，2024

目 录

摘 要

近年来，日新月异的互联网技术及其应用极大地提升了人类工作生活质量。但互联网的开放性、多样化的用户群体与应用却同时使得互联网面临着日益严峻的安全挑战，新的安全漏洞、新的攻击手段层出不穷，给安全防护带来了更大的难度。网络行为分析是指通过对网络流量、用户活动等进行监控和分析，旨在提高网络的安全性，是解决这一系列网络安全问题的有效手段。传统的网络行为检测方法是通过分析通信流量（如TCP、UDP）实现对恶意流量的检测，这种方法主要针对低速、低 QoS 网络环境。由于恶意流量会混杂在正常的网络流量中，在终端网络防护设备资源有限的情况下，对当前高速网络中庞大的数据进行分析难度大且效率低下，使得传统网络行为检测方法在真实的网络环境中逐渐失效。

相比之下，DNS 解析流量体量非常小，通过分析 DNS 解析流量可以对大多数的网络行为进行检测，并且部分网络恶意行为会通过 DNS 建立隐蔽通信隧道来逃避检测，这种情况必须通过分析 DNS 解析流量来检测此类恶意应用。因此，从域名解析行为分析的角度进行网络恶意行为检测是一种有效的途径。本文以 DNS 流量分析和机器学习方法为基础，从域名解析行为分析的角度对网络恶意行为检测进行了如下研究：

（1）首先针对以虚拟货币挖矿行为检测问题，对其非加密 DNS 流量进行检测，提出了一种基于词素的矿池域名检测方法 Mining Vanguard，解决传统检测方法实时性低的问题。该方法包含快速匹配和动态推理两个模块，在快速匹配模块中设计了一个基于词素的哈希表，优化了正则表达式的匹配过程，能够实现对知识库中矿池域名的快速发现；在动态推理模块中通过构建一个包括传统 DNS 解析特征和词素特征在内的综合特征集，将网络特性与语义特性结合在一起，并根据此特征集训练一个能够区分正常域名和矿池域名的机器学习模型，实现了对未知矿池域名的发现。在真实网络环境下采集的数据集上实验，表明该方法在挖矿行为识别问题上具有良好的性能。

（2）设计了一种阶层式网络恶意应用识别系统，该系统主要由四大模块组成，其中，数据采集预处理模块负责流量的数据采集和特征提取；域名匹配推理模块通过分析 DNS 流量实现对恶意应用的早期发现，匹配模块和推理模块相结合，兼顾了系统的实时性和准确率；基于 LTC-Sketch 的初筛模块能够通过分析网络流的持久度和频繁度，筛选出特定频率的流量，过滤掉大部分正常流量，降低正常流量和恶意流量的非平衡比率，提高后续模型的识别效率，降低模型算力消耗。机器学习模块能够对疑似恶意流量进行

识别，做出最后的决策。以挖矿行为检测实际任务为实例进行了实验验证，实验结果显示该系统对挖矿行为检测的表现可以达到较为理想的效果。

（3）针对加密 DNS 流量，提出了一种基于加密 DNS 流量的网站指纹识别方法，通过分析加密 DNS 流量实现对恶意网站的识别。该方法根据加密 DNS 流量的特点，通过大津法（OTSU）进行数据划分，提高在真实网络环境下的识别效率和准确率。然后设计了一种基于直方图的包长特征和基于时间序列的特征集，结合传统网络统计特征，形成了一个针对加密 DNS 流量网站指纹识别的特征集合。本文设计了一个加密 DNS 流量采集平台，基于该平台采集制作了一个涵盖多种加密 DNS 协议和 DNS 解析服务器的数据集。在该数据集上进行了验证实验，实验结果显示该方法在 DNS-over-QUIC 和 DNS-over-HTTPS 网站指纹识别任务上均有较好的表现。最后，将该方法应用在网络恶意应用识别任务上，通过分析网络恶意应用产生的加密 DNS 流量来进行网络恶意应用检测，实验结果显示该方法对网络恶意应用检测也具备较好的效果。

关键词：网络行为分析；恶意流量识别；DNS；机器学习

Abstract

In recent years, rapidly evolving internet technologies and their applications have significantly improved the quality of human work and life. However, the openness of the internet, along with its diverse user base and applications, has also led to increasingly severe security challenges. New security vulnerabilities and attack methods continuously emerge, making security protection more difficult. Network behavior analysis, which involves monitoring and analyzing network traffic and user activities to enhance network security, is an effective means to address these network security issues.

Traditional network behavior detection methods rely on analyzing communication traffic (such as TCP and UDP) to detect malicious traffic. These methods are primarily designed for low-speed, low-QoS network environments. As malicious traffic often blends with normal network traffic, analyzing the massive amount of data in high-speed networks with limited resources on terminal network protection devices is challenging and inefficient. Consequently, traditional network behavior detection methods are becoming less effective in real network environments.

In contrast, DNS resolution traffic is relatively small in volume. By analyzing DNS resolution traffic, most network behaviors can be detected, and some network malicious applications use DNS to establish covert communication tunnels to evade detection. Therefore, analyzing DNS resolution traffic is essential for detecting such malicious applications. This thesis focuses on network malicious application detection from the perspective of DNS resolution behavior analysis using DNS traffic analysis and machine learning methods:

(1)Firstly, targeting the detection of virtual currency mining behavior, a morpheme-based mining pool domain name detection method called Mining Vanguard is proposed for detecting non-encrypted DNS traffic. This method addresses the low real-time performance of traditional detection methods. It includes two modules: fast matching and dynamic inference. The fast matching module employs a morpheme-based hash table, optimizing the regular expression matching process to quickly identify mining pool domain names in the knowledge base. The dynamic inference module constructs a comprehensive feature set that combines traditional DNS resolution features with morpheme features, training a machine learning model to

distinguish between normal and mining pool domain names. Experiments on real network data demonstrate the method's effectiveness in identifying mining behavior.

(2)A hierarchical network malicious application recognition system is designed, comprising four main modules. The data collection and preprocessing module handles traffic data collection and feature extraction. The domain name matching and inference module detects malicious applications early by analyzing DNS traffic, balancing real-time performance and accuracy. The initial screening module, based on LTC-Sketch, analyzes the persistence and frequency of network flows, filtering out most normal traffic and reducing the imbalance ratio between normal and malicious traffic, thereby improving the efficiency and reducing the computational load of subsequent models. The machine learning module identifies suspected malicious traffic, making the final decision. Experiments on mining behavior detection demonstrate the system's effectiveness.

(3)For encrypted DNS traffic, a website fingerprinting method based on encrypted DNS traffic is proposed to identify malicious websites. This method utilizes OTSU for data segmentation to enhance identification efficiency and accuracy in real network environments. It combines histogram-based packet length features and time series-based features with traditional network statistical features to form a feature set for website fingerprinting in encrypted DNS traffic. A platform for collecting encrypted DNS traffic was designed, creating a dataset encompassing various encrypted DNS protocols and DNS resolution servers. Validation experiments on this dataset show the method's strong performance in DNS-over-QUIC and DNS-over-HTTPS website fingerprinting tasks. Finally, the method is applied to network malicious application detection by analyzing encrypted DNS traffic generated by malicious applications. Experimental results indicate the method's effectiveness in detecting network malicious applications.

Key Words: network behavior analysis; malicious traffic identification; dns; machine learning

第一章 绪 论

1.1 研究背景与意义

近年来，随着互联网的高速发展和互联网用户群体的迅速增长，生活在地球上的每个人几乎都可以享受到互联网给生活带来的各种便利。根据 We Are Social 发布的 Digital 2024 报告[1]显示，截止到 2024 年 1 月，全世界网民数量高达 53 亿，相当于全世界人口总数的 66%。云计算、大数据、物联网、工业互联网等新技术应用大规模发展，各种类型的终端设备都具备了接入互联网，进行信息交换和智能化调控的能力，为人们的生活带来了前所未有的便利。

然而，新技术的发展也使得网络安全问题日益复杂，近些年出现了较多关于网络安全的案例。2020 年 6 月，日本理化学研究所的一台超级计算机遭到了勒索软件攻击。攻击者采用了 Ryuk 勒索软件进行攻击，导致该计算机的大量数据被加密，要求受害者支付赎金才能解密[2]；2019 年 SSH（Secure Shell）漏洞 CVE-2019-6110[3]，攻击者可以利用该漏洞通过中间人攻击的方式，执行远程命令操纵客户端输出，影响了许多诸如路由器、工控机等基于嵌入式系统的设备。近期，国家高度关注的一个网络安全焦点是恶意虚拟货币挖矿行为，这已成为一个典型的网络安全案例。特别是那些企业级互联网数据中心（IDC）、ISP 骨干网络交换中心以及国家级超级计算中心（CSA）等，它们因拥有丰富的网络资源和强大的算力资源，成为了恶意挖矿活动的主要目标。恶意挖矿活动不仅会对计算机的 CPU、GPU 等硬件资源造成占用，导致计算机性能下降甚至崩溃，还会显著增加计算机的能耗，从而无端浪费电力资源。这种浪费不仅影响了个体用户和企业，更对我国正在推进的“双碳”战略构成了重大挑战。

2018 年 9 月，NSAFtpMiner 使用 Eternalblue(永恒之蓝)、Doubleplsar(双脉冲星)等漏洞攻击工具感染了约 3 万台电脑进行恶意挖矿活动。企业、政府机构、事业单位的高性能服务器也成为攻击目标，挖矿木马甚至入侵多家医院的医疗计算机系统进行恶意挖矿活动[4]。2020 年 11 月，4SHMiner 挖矿木马团伙控制约 1.5 万台服务器进行挖矿，根据算力突变数据显示其在 16 至 17 日一天之内就新增感染近 1 万台机器[5]。2022 年 5 月，国家互联网应急中心监测数据显示，恶意挖矿团伙“8220”渗透了四千台左右的设备并传播挖矿木马窃取其算力[6]。11 月 10 日，国家发改委组织召开治理虚拟货币挖矿

专题视频会议，要求严查严处国有单位机房涉及的挖矿活动[7]。尽管遭受严厉政策的打击，但在巨大利益的诱惑之下，挖矿活动由地上转为地下。众多的黑产团伙为了逃避监管，通过网页挂马、系统漏洞等方式将挖矿木马部署到高性能计算设备上，窃取计算资源进行恶意挖矿活动，造成极大的网络安全威胁以及恶劣的社会影响。因此，对网络行为的分析，特别恶意行为的检测成为近年来的一个研究热点。

网络行为分析通过对网络流量、用户活动等进行监控和分析提高网络的安全性。DNS 是一种常常被应用于网络攻击的网络协议，并且几乎所有网络行为都会利用 DNS 协议完成主机名和互联网地址的转换服务。关于 DNS 的网络恶意行为可分为对 DNS 通信过程或数据可靠性进行破坏的攻击活动，以及利用 DNS 正常解析的特性辅助攻击者进行破坏的攻击活动。无论是否破坏 DNS 解析的正常过程，攻击者目的都是为了让受害主机与攻击者指定的主机进行通信，从而达到其目的。但是，上述的恶意攻击行为，都会在域名解析流量中留下痕迹。比如，在挖矿木马攻击时，挖矿木马会让受害主机发送 DNS 请求获得矿池的 IP 地址；网络恶意行为通过加密 DNS 协议建立隐蔽通信隧道进行通信时产生的加密 DNS 流量与正常访问网页产生的加密 DNS 流量时间特征存在差异。当网络恶意行为进行攻击时，攻击流量会混杂在正常的网络流量中，在终端网络防护设备资源有限的情况下，对当前网络的全网流量进行分析难度大且效率低下；相比之下，DNS 解析流量体量非常小，通过分析 DNS 解析流量可以对大多数的网络行为进行监控，而且存在部分网络恶意行为会通过 DNS 建立隐蔽通信隧道（DoH 隧道等），这种情况必须通过检测 DNS 解析流量来检测此类恶意应用。因此本课题研究目的为通过分析域名解析行为流量，针对网络恶意行为检测具体问题展开研究。一方面利用域名解析流量进行虚拟货币挖矿检测能够实现该恶意行为的早期识别，弥补传统检测方法实时性较低的问题。另一方面，加密 DNS 协议的出现给网络监管带来了新的挑战和问题，需针对加密 DNS 协议的特点，设计新型的特征提取和流量检测的方法。另外，随着近年来面向传统低速网络环境中的测量方法逐渐失效，需要设计在新型高速网络环境下更为复杂的网络流量检测框架，成为了网络行为分析的重点。

1.2 本文创新点与组织结构

本文的主要工作与创新点如图 1.1 所示，主要有以下几个内容：

（1）提出了一种基于词素的矿池域名识别方法 Mining Vanguard：从域名解析行为分析的角度检测加密货币挖矿行为，根据矿池域名中词素相似度较高的特点，设计了一

种将快速匹配和动态推理相结合的矿池域名识别框架，基于词素信息设计了一种面向矿池正则表达式匹配的哈希结构和用于检测矿池域名的机器学习模型，并且给出了对应的证明过程和充分的实验验证。

（2）采集了大量真实环境下的数据集，涵盖了多种类型的网络流量，包括矿池域名、挖矿流量以及访问网页产生的加密 DNS 流量。这些数据集的采集覆盖了多种网络环境和应用场景，通过对这些真实环境数据集的分析和研究，验证了基于域名解析行为分析的网络恶意行为或行为检测的有效性和可靠性，为相关研究提供了更丰富的数据支持和实证分析基础。

（3）设计了一种阶层式网络恶意应用识别系统，以挖矿行为检测实际任务为依托进行了实验验证。在该方法中使用多种网络流量测量方法与早期网络行为检测技术，探索了基于域名解析行为分析在综合流量分析系统中的应用，通过多模态的网络数据实现对加密货币挖矿行为的识别，兼顾了检测方法的时效性和可靠性。

（4）提出了一种基于加密 DNS 流量的网站指纹识别方法，为 DNS 加密后的网络监管提供了新的思路。本文根据 DNS 流量特点提出了一种基于直方图和时间序列的网络流量特征集合，在真实的加密 DNS 网页流量数据集上进行了测试和验证，证明了该方法的有效性。另外将该方法应用在网络恶意应用识别任务上，通过分析网络恶意应用产生的加密 DNS 流量来进行网络恶意应用检测，获得了较好的实验效果。

本文共分为五章，各章安排如下：

第一章，全面概述了本文主要研究的基于域名解析行为分析的网络恶意行为检测的研究背景、意义和动机，给出本文的主要研究内容和创新点，并介绍本文的主体结构与章节安排。

第二章，全面介绍了课题所涉及的 DNS 流量以及加密 DNS 流量、网络流量识别、虚拟货币挖矿及网站指纹识别领域中的概念、发展和研究现状，其中着重介绍了互联网流量识别技术和加密货币挖矿检测技术的研究现状及其和本文的关系。

第三章，介绍了本文提出的基于域名解析行为分析的加密货币挖矿检测方法 Mining Vanguard 及以 Mining Vanguard 为基础的阶层式网络恶意应用识别系统，特征库的构建、模型匹配检测的具体流程、矿池域名和网络流量数据集的采集及阶层式网络恶意应用识别系统的各个模块的作用和原理，构造了性能评估实验和消融实验等，在真实的网络环境下进行了测试，验证了 Mining Vanguard 及阶层式网络恶意应用识别系统的在检测加密货币挖矿行为的性能。

第四章，介绍了基于加密 DNS 流量的网站指纹识别方法，分别介绍了加密 DNS 流量现存问题、加密 DNS 流量测采集平台和采集流程、直方图特征的提取过程，结合采集的数据集验证了基于加密 DNS 流量的网站指纹识别方法的有效性的同时，对加密 DNS 流量的安全性进行了进一步的分析。

第五章，用于总结本课题的整体研究内容和和研究结果，并对本课题部分细节结合最新研究成果给出了下一步的研究方向。

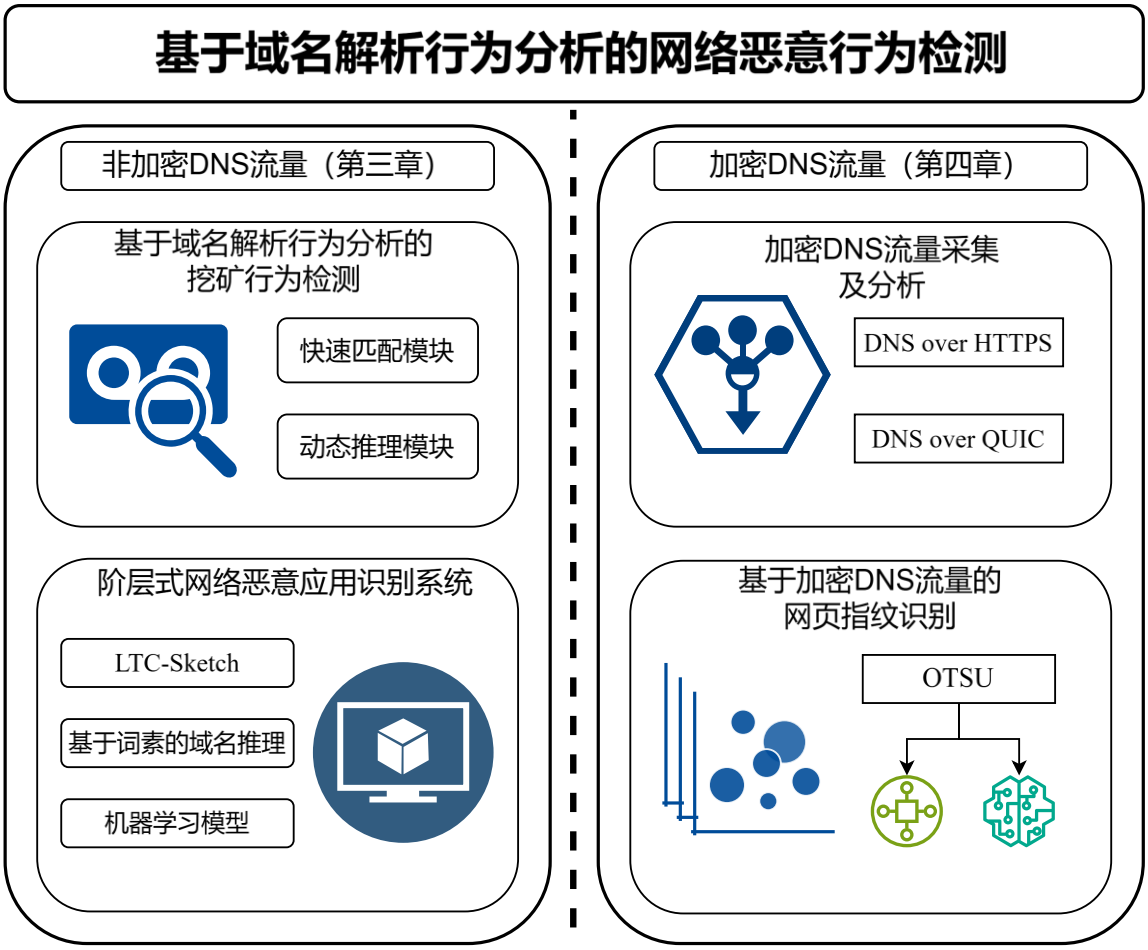

图 1.1 本文各研究内容间关系架构

第二章　相关技术与研究综述

在本章中，首先介绍与本文相关的 DNS 技术的概念和理论，其次分别介绍虚拟货币挖矿及其检测技术、互联网流量识别技术以及网站指纹识别技术三个领域的基本原理和研究现状。

2.1 DNS 技术

域名系统（Domain Name System，DNS）是一种分层的、全球分布的目录服务[8]。它在互联网通信过程扮演着关键的角色，在网络通信中，设备主要通过 IP 地址进行通信，然而 IP 地址这串数字难以轻松记忆，于是域名系统应运而生。DNS 服务器负责建立域名与 IP 地址的映射关系。当主机访问某个域名时，DNS 服务器可以将该域名转换为相应的 IP 地址，使得主机能够更加方便地访问目标网站。本文主要介绍域名解析和加密 DNS 协议。

2.1.1 域名解析

DNS 系统作为将域名映射到与之相关联的 IP 地址的重要网络基础设施，地址转换则是通过 DNS 解析实现的。标准的 DNS 协议基本格式如图 2.1 所示，DNS 将字符串形式的域名转换成数字形式的 IP 地址，方便了主机用户的记忆，也为互联网服务提供了灵活的资源配置方式。

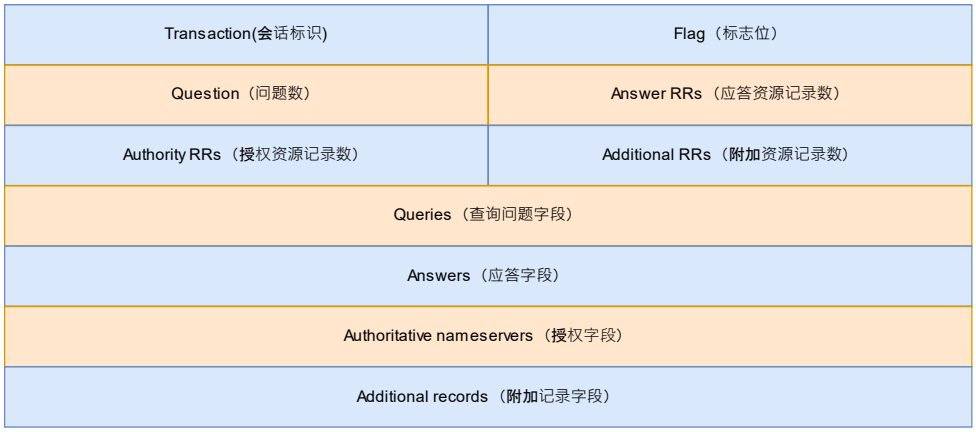

图 2.1 DNS 协议基本格式

域名解析是指将用户输入的域名转换为对应的 IP 地址的过程，域名递归解析过程如图 2.2 所示。当用户在浏览器中输入一个域名时，首先会向本地域名解析器发起查询

请求。本地解析器会检查本地缓存中是否有该域名的解析记录，如果有且未过期，则直接返回对应的 IP 地址。

如果本地缓存中没有相应的记录或者记录已过期，本地解析器将向根域名服务器发送查询请求。根域名服务器负责存储顶级域名（如 com、net、org）的 IP 地址信息。根域名服务器会返回对应顶级域名服务器的 IP 地址。

接着，本地解析器向顶级域名服务器发送查询请求，获取目标域名所属的权威域名服务器的 IP 地址。权威域名服务器是负责管理特定域名的服务器，存储着该域名的 IP 地址信息。

最后，本地解析器向权威域名服务器发送查询请求，获取目标域名的确切 IP 地址，并将结果缓存到本地，以便下次快速响应相同的查询。这样，用户的浏览器就能通过获取到的 IP 地址与目标服务器建立连接，实现对该域名的访问。整个域名解析过程通过多级的查询和响应完成，确保用户能够方便快速地访问互联网上的各个网站。

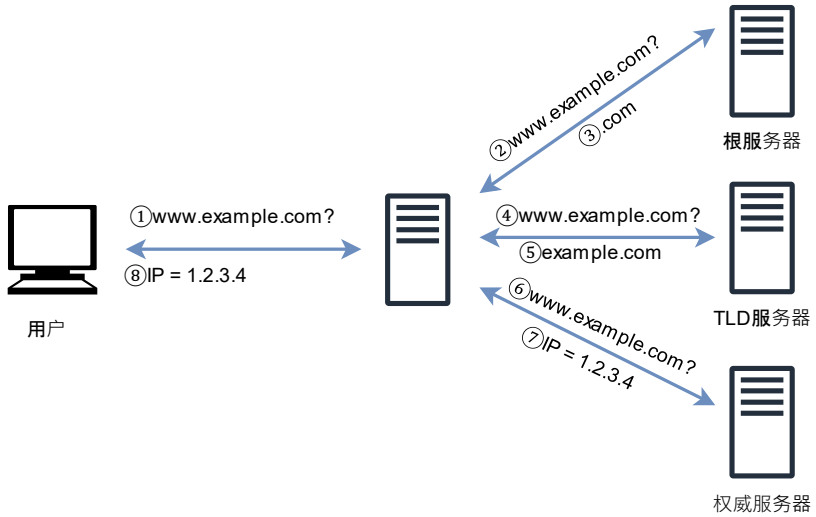

图 2.2 域名递归解析过程

2.1.2 加密 DNS 流量

常规的域名请求大多以明文形式发送，这种情况使得互联网服务提供商（Internet Service Provider，ISP）、自治系统（Autonomous System，AS）或国家级机构等实体能够进行用户跟踪、大规模监视和审查。广泛监视的风险及其后果促使互联网治理者、行业参与者和标准化机构加强隐私保护。目前较为主流的加密 DNS 协议有 DNS-over-TLS（DoT）、DNS-over-HTTPS（DoH）和最近成为标准协议的 DNS-over-QUIC（DoQ）。这些协议加密了客户端与递归解析器之间的通信，以防止网络窃听者检查域名。在 2018

年期间，谷歌[9]和 Cloudflare[10]推出了公共 DoH 解析器，而 Mozilla 也为 Firefox 添加了 DoH 支持[11]。这些努力都旨在利用 DoH 和 DoT 的加密能力来增强对用户的浏览器流量安全保障[12]。

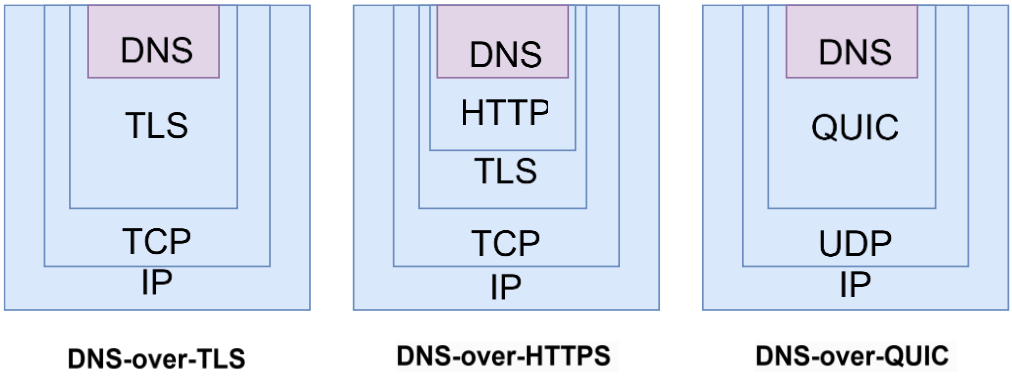

图 2.3 三种加密 DNS 协议网络分层结构

DNS 加密是通过将 DNS 查询和 DNS 响应(客户机和解析器之间)的内容封装在上层协议中实现的。目前三种标准化的 DNS 加密方法（DoT，DoH 和 DoQ），每种方法都使用特定的上层协议。如图 2.3 所示，与直接嵌入在 UDP 传输层下的基于明文的 DNS 不同，三种 DNS 加密方法分别将 DNS 封装在其启用的加密层中，即 TLS、TLS with HTTP（即 HTTPS）和 QUIC。

DoT 是第一个针对 DNS 安全提出的 DNS 加密标准协议，其 RFC[13]在 2016 年正式发布，DoT 使用传输层安全协议 TLS 对 DNS 查询流量和 DNS 响应流量进行加密。TLS 协议广泛应用于电子邮件（SMTP）、超文本（HTTP）、IP 语音（VoIP）等流行的网络应用中，证明了其加密效果。

基于 DoT 的 DNS 服务具体地域名解析过程如下。首先主机端向支持 DoT 的 DNS 解析服务器发送连接请求，使用指定的 DoT 服务端口号（TCP/853）。接下来通过典型的 TLS 握手过程与 DNS 解析服务器建立 TLS 连接，主机端与 DNS 解析服务器交换加密密钥[14]。在 TLS 会话建立成功之后，主机端可以通过 TLS 加密 DNS 流量。一般情况下，TLS 连接会保持连接状态，从而减少后续 DNS 查询过程中可能需要进行额外的 TLS 握手而产生的网络延迟。虽然 DoT 是一种可行的 DNS 加密方法，但它面临着一些可能限制其使用的挑战。一方面，据 Kartch 等人[15]报告指出，专用于 DoT 服务的端口 TCP/853 目前并没有被安全设备所认可，因此很可能被防火墙设备阻止；另一方面，虽然 DoT 查询和响应的内容是加密的，但通过监听固定的端口号可以相对容易地获得加密的 DNS 数据包。

DoH 于 2018 年标准化[16]，它利用使用较为广泛的 HTTPS 协议来封装 DNS 的内容，通过 TCP/443 端口来传输信息，从而使得现有的安全措施不会阻碍 DoH 的传输，DoH 目前是使用最为广泛的 DNS 加密协议[17]，首先因为 HTTPS 是浏览器默认使用的超文本协议，使得 DoH 在主机端更容易被接受。其次从隐私的角度来看，DoH 比 DoT 的安全性更强，因为 DoH 流量和 HTTPS 流量会混杂在一起，这使得通过端口无法直接获得 DoH 流量，并且日后对 HTTPS 的任何性能和安全性的增强都会使得 DoH 受益。

DNS over TLS（DoT）和 DNS over HTTPS（DoH）都使用 TCP 作为传输层协议。考虑到建立连接所需的 TCP 和 TLS 握手，加上 TCP 确认机制，DoT 和 DoH 的 DNS 请求相比于基于明文的 DoUDP 的方式来说，响应时间可能会更长，这种时间开销是不可忽视的。因此，为了进一步提高加密 DNS 的性能，DoQ[18]在 2017 年由 Google 首次作为互联网草案提出，它利用 QUIC 协议[19]（建立在 UDP 传输层之上），通过零 RTT 握手和数据流多路复用，使得加密通信更快。据 G. Carlucci 等人的报告[20]显示，QUIC 在页面加载时间等响应质量指标上，均优于其竞争对手 HTTPS。

尽管 DoQ 在性能上有其优点，但与 DoT 类似，DoQ 使用专用服务端口（UDP/784 和 UDP/8853），目前这些端口并不被安全系统广泛认可，因此其流量在传输过程中可能被防火墙默认阻止。

2.2 互联网流量识别

互联网流量识别技术的发展历程可划分为三个关键阶段，每个阶段都伴随着不同的主流识别技术。早期，端口映射识别技术基于端口号的动态映射规则进行流量识别，然而，随着网络应用模式的日益动态化和多样化，这种方法的适用性逐渐减弱。随后，深度包检测技术（DPI）崭露头角，以其较高的准确性受到关注。然而，DPI 技术的实施依赖于数据包载荷的可见性，这在一定程度上可能侵犯用户的隐私。此外，对于加密流量的识别，DPI 技术面临较大的挑战，因为其难以直接解析加密数据的内容。

在此基础上，Sherry 等人提出使用一种中间件 BlindBox[21]在不损害用户隐私的情况下来检查 TLS 加密的流量中是否存在恶意流量，但由于建立检查通道的巨大延迟，其使用受到限制。Ning 等人提出的 PrivDPI[22]通过在后续会话中重用先前会话中生成的中间结果，将 BlindBox 的效率提高了 288 倍，使检测场景更加可行。尽管近年来 DPI 技术随着专用硬件（如 FPGA 智能网卡[23]）的发展，凭借其高度的可靠性，又重新被工业界所认可。比如，在 Nvidia 公司于 2022 年在最新一代产品 BlueField-3[24]系列智能网

卡设计的开发架构 DOCA 中，为了方便运维人员对云原生 DPU 流量的测量需求，也提供了对 DPI 技术 SDK 支持[25]。但不可避免的是，由于 DPI 方法规则依赖性较强，故而泛化能力较弱，这就需要使用基于机器学习的识别方法。

基于机器学习的流量识别方法相比于 DPI 技术，无论是识别准确性还是泛化能力上，都有极大的优势[26]。机器学习方法包括有监督学习[27,28]、无监督学习[29,30]，还包括近年来研究成果较多的基于半监督学习[31,32]、深度学习[33]和强化学习[34]的识别方法。Shen 等人[35]提出了一种动态优化流量识别过程中特征选择的方法，针对不同的时间复杂度需求、不同的准确性要求，给出最为优化的特征选择方式。在半监督、无监督学习方面，Wu 等人[30]提出 BCAC，将流量压缩存储为 Sketch 结构后，使用聚集聚类完成流量分类任务。近年来，在流量识别领域被广泛使用的卷积神经网络（Convolutional Neural Networks ，CNN）[36,37]，识别准确率高，而且不需要人工选择特征。典型的工作如：Lim 等人[38]的方法使用 CNN 结合残差网络（ResNet[39]），只根据数据包大小而不使用包头其他信息，将将 payload 转换为图片的格式完成分类任务。类似地，Wang 等人[40]将 CNN 应用到特征提取、特征选择和分类任务，运用一维卷积神经网络，整合到了一个端到端的框架中。Lotfollahi 等人[33]提出的 Deep packet 在使用 CNN 的同时还结合了 SAE，不仅能完成流量协议识别，还能识别不同的应用。该方法在南京理工大学陆一飞团队[41]的改进下，进一步实现了在未知流量分类时有较高的泛化能力。

2.3虚拟货币挖矿及其检测

2.3.1 虚拟货币挖矿原理

虚拟货币是一种使用密码学原理来确保交易安全及控制交易单位创造的交易介质。比特币（Bitcoin，BTC）是最早出现的基于去中心化，以区块链为底层技术的虚拟货币，中本聪（Satoshi Nakamoto）于 2008 年 10 月 31 日发表比特币论文[42]，2009 年 1 月 3 日发布创世区块。十多年来，虚拟货币发展迅猛，根据 CoinMarketCap 的数据，截止到 2021 年 12 月，全球虚拟货币种类高达 8600 多种[43]。

从技术视角来看，区块链不仅是虚拟货币的基石，更是一个网络分布式账本。与传统银行采用的中心化记账模式不同，区块链的去中心化 P2P 网络结构通过独特的共识机制确保每笔交易都被记录在区块链上，从而验证交易的合法性，并有效防范双重支付的风险。目前主流的共识机制就是工作量证明（Proof of Work，PoW）机制[44]。根据该机

制的规定，用户需要进行一系列耗时且复杂的运算，得到运算结果后，该结果需要被服务方验算，以耗费的算力和时间作为担保成本，以确保区块的链接并保持交易的完整性。

为了激发人们参与虚拟货币分布式账本中交易记录的验证工作，中本聪创建了记账激励机制，即“挖矿”。当前，“矿工”通常通过公共矿池协作进行“挖矿”，这类似于一种集体努力。在矿池中，每个矿工贡献自己的哈希率，一旦矿池成功挖掘到一个区块，奖励将根据矿工们各自的哈希率贡献进行公平分配。为了确保奖励的公正性，矿池需要采用特定机制来准确识别每位参与挖矿的矿工。一些矿池使用专有的站点密钥，比如 CoinHive[45]，它在 2019 年 3 月之前一直是基于浏览器的挖掘服务的主要提供商。还有一些矿池使用电子邮件或钱包地址。每个矿工和矿坑之间的通信是通过使用 Straum[46] 协议完成的。这是一个明文协议，使用 JSON-RPC 格式传输信息，包括身份验证令牌、块问题和挖矿结果[47]。由于区块链的挖矿机制要求第一个完成特定工作量证明（Proof of Work, PoW）的矿工才能获得奖励，这使得挖矿活动成为了一场高度依赖矿工哈希率（即每秒计算哈希数的能力）的竞赛。哈希率越高，矿工挖掘到新区块并获取奖励的概率就越大。因此，加密虚拟货币的高价值吸引了不法分子的注意。他们采用网络入侵等手段，对他人的计算资源进行非法利用，通过在浏览器中嵌入恶意脚本或以恶意软件的形式分发恶意代码，将他人的计算资源转化为自己的“矿机”，以窃取虚拟货币作为报酬。这种行为被称为恶意挖矿，它不仅侵犯了他人的计算资源，还严重威胁了这些资源的安全性。这一威胁在最近几年有了很大的增长[48]，特别是在我国政府强力打压公开的矿场之后，主动的“挖矿”行为已不再是主流，更多的是被恶意劫持导致的被动挖矿。

2.3.2 虚拟货币挖矿的检测

被动的恶意挖矿通常由两种模式进行。一种是使用基于浏览器的加密挖掘程序（称为 Crypto-Jacking），其中“挖矿”过程在嵌入 web 内容的脚本（通常是 JavaScript）中运行；另一种使用恶意挖矿软件，挖掘过程嵌入在连接到互联网的受感染机器上运行的恶意软件的有效负载中。在这两种情况下，恶意挖矿的主体通过使用数百台被劫持的机器，可以获得接近中型矿场的哈希率。针对两种不同的挖矿模式，近年来学术界和工业界分别提出了主机侧检测和网络侧检测两类方法[49,50]。

主机侧检测技术通常是在单一目标主机上安装检测程序，使用系统调用和操作码作为特征，并将特征输入到识别模型[51]。Karn 等人[52]用在容器云的场景下，通过监视 Linux 内核系统调用情况来判断云容器中是否存在挖矿行为。类似地，Darabian 等人[53]也是通

过系统调用和操作码作为判断依据，并设计了基于注意力机制的 ATT-LSTM 作为检测模型。Gangwal 等人[54]使用高性能硬件计数器和 iPerf[55]（Linux 操作系统中用于测试网络性能的工具）测量网络带宽、QoS 等性能，结合从处理器中提取执行事件特征，如指令特征、缓存特征等指标构建模型，并使用随机森林和支持向量机作为检测模型。此外，Azmoodeh 等人[56]创新地提出检测电能的消耗水平，通过时间序列分析的方法来识别挖矿行为。这是因为挖掘应用程序需要巨大的计算资源，挖矿时的瞬时电功耗更大。类似的，Kelton 等人设计的内置于浏览器中的工具 CoinSpy[57]，构建了计算、内存和网络资源使用负载和占比的多维度时间序列，利用深度学习模型构建检测挖矿活动。

使用主机检测方法可以准确检测目标主机上的挖掘行为，但是这些方法部署较为困难，且难以同时对大量关注主机同时检测。为了弥补这些缺点，研究人员开始在网络上部署检测系统，设计网络侧检测方案，尝试从网络流量特征中提取信息来检测挖掘行为。例如，Caprolu 等人[58]从网络流量中提取简单的统计特征，如包到达时间间隔和数据包大小的滑动平均和标准差等，使用这些特征数据训练机器学习模型来检测挖掘行为。这些方法已被证明在检测加密劫持方面是有效的。Pastor 等人[59]则提取了更多种的特征，添加了诸如数据包总数、有效负载等信息，利用多种决策模型来进行检测，并证明了他们方法的有效性。这些方法从完整的流中提取特征，比真实检测更有意义。此外，针对现有挖矿行为流量样本较少的问题，Mozo 等人[60]提出利用生成对抗网络（GAN）生成挖矿流量样本，还能模拟挖矿行为中各种不同的事件（如断网重连）产生的流量，进一步地支持了其他机器学习模型的训练。Radhesh 等人[61]结合了主机端检测和网络端检测各自的优势，对现有常见挖矿宿主程序 WebMine[62]、Coinhive[45]和 CrypotoLoot[63]等分析其通信协议和会话规律，提炼关键字指纹库，并基于主机端检测系统调用的模块结合，设计了 Mining Sweeper，识别挖矿软件的通信流量。

对于国内日渐严峻的恶意挖矿形势，我国也有代表性团队快速地研发出了行之有效的检测方法。东南大学程光团队提出了一种识别加密货币以太坊（Ethereum）的方法[64,65]，使用流量中典型特征的统计信息，如端口号、报文大小、报文到达顺序等，对活跃节点库中记录的节点进行分析，以识别以太坊的 TCP 流量。在此基础上，该团队还构建了机器学习模型，以较高精度识别加密的比特币挖矿僵尸网络流量[66]。

2.4 网站指纹识别

网站指纹识别指通过机器学习分类器学习每个网站的工作模式，具体流程为监听者

通过从敏感网站(被监控的网站)和其他用户可能访问的网站(未被监控的网站)中收集大量客户端访问网页的网络流量来训练分类器。有了训练好的分类器，监听者就可以在被监听者浏览网页产生的加密连接中拦截流量，并使用分类器确定用户访问的具体网站。

网站指纹识别的起源可以追溯到 1996 年 Wagner 等人[67]的研究，他们发现，通过分析加密的 HTTP 请求传递的数据包总量可以揭示出用户访问的具体网站。随后的网站指纹识别主要针对加密系统薄弱的部分，如 HTTPS[68]、OpenSSH[69]、VPN[70]以及加密的网页代理[71]，直到 2009 年，Herrmann 等人[72]首次尝试将网站指纹应用于 Tor（洋葱路由）。在接下来的十年里，网站指纹识别通常基于一组手动选择的特征来训练分类器，分类器的准确率可以达到 80%甚至超过 90%。Bhat 等人[73]提出了一种基于 ResNet 的洋葱路由场景下的网站指纹识别，这种识别方法是一个半自动特征提取过程，因为它引入了累积特征作为额外的特征（例如，数据包的总数、总传输时间），他们将该方法应用到 Rimmer 等人[74]的数据集时，观察到准确率在非防御和防御场景下都有所提高。Eun 等人[75]进一步研究了深度学习技术在网站指纹识别上的应用，探索自动编码器（AutoEncoder，AE）作为特征提取器的潜力，他们去除了 AE 的 SoftMax 分类器，发现 AE 提取的特征比手动提取的特征集更有力。但是深度学习技术深受诟病的一点就是不可解释性，因此在准确率相差不大的时候，轻量级机器学习模型在网络流量识别领域更受欢迎。

上述工作都是只关注到访问网页产生的 HTTPS 流量，并没有在加密 DNS 流量上进行相关测试。关于基于加密 DNS 流量的网站指纹识别是一个较为崭新且富有挑战性的领域，但它对整个 DNS 安全和网络监管有着十分重要的意义。Houser 等人[76]通过对加密 DNS 流量包长信息和时间信息进行建模来推断用户访问的具体网站。在 DNS 无填充策略的情况下，该方法在基于 DoT 的网页指纹识别任务上可以达到假阴性率小于 17%、假阳性率小于 0.5%的效果。Siby 等人[77]通过使用比 HTTPS 流量少了 124 倍的 DoH 流量实现了网站指纹识别，证明了通过加密 DoH 流量进行网站指纹识别的可能性。Bushart 等人[78]通过结合了加密 DoH 流量和加密 DoT 流量的包长信息和时间信息来推断用户访问的网站，并且证明了加密 DNS 协议仅靠填充策略并不能有效的阻止网站指纹识别。

2.5本章小结

本章首先介绍了 DNS 技术以及三种主流的 DNS 加密协议，然后概述了近年来网络流量识别的研究现状和虚拟货币挖矿的原理以及近期工业界和学术界提出的检测技术，

最后介绍了网站指纹识别的概念及其相关研究现状和基于加密 DNS 流量的网站指纹识别技术研究现状，为后续章节中使用到的概念、方法及其研究现状做了简要的说明和介绍。

第三章　基于域名解析行为分析的加密货币挖矿行为检测

本章节，重点介绍本文提出的基于词素的矿池域名识别框架 Mining Vanguard 及其在阶层式网络恶意应用识别系统中的应用。首先介绍了 Mining Vanguard 的整体框架和检测流程，对其中的数学问题给出了阐述和证明，并且依据真实网络环境下采集的数据集进行了有效性实验；然后介绍了阶层式网络恶意应用识别系统整体识别流程和各个模块的具体作用；最后进行了整体性部署和各个模块有效性实验。

3.1 Mining Vanguard 方法

本文提出了一种基于词素的矿池域名识别方法，目的是实现对加密货币挖矿行为的早期识别，在下文中称为 Mining Vanguard。Mining Vanguard 的最大特点在于其分析的数据对象不是传统加密货币挖矿行为检测方法所关注的 TCP 流量，而是从 DNS 请求流量入手。挖矿软件运行过程如图 3.1 所示，执行挖矿任务的挖矿软件需要通过连接矿池执行挖矿操作，所以大多数挖矿软件在开始阶段会发送 DNS 请求来获取矿池的 IP 地址，因此通过确认主机是否有连接矿池的 DNS 请求，就可以实现对加密货币挖矿行为的早期识别。

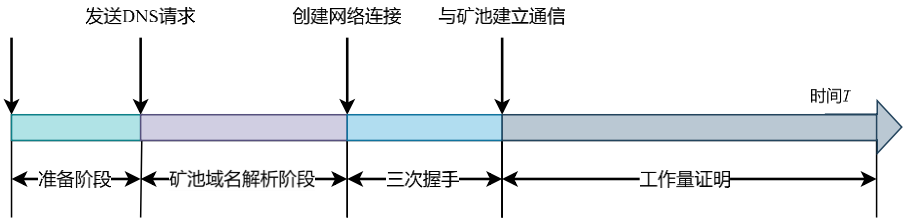

图 3.1 挖矿软件运行流程

本文发现矿池域名中往往包含特定的词汇，比如“pool”等，因此 Mining Vanguard 的另外一个创新点是将语言学概念“词素”作为分割域名的基础，并提出了几种基于词素的域名统计特征。与单词不同，词素是语言中最小的有意义的词汇单位[79]。相比于基于单词的分割或机械地使用 N-gram 进行统计，基于词素的方法更加动态，更符合语言规律。以域名“internationalreview.com”为例，在去除顶级域名“com”后，第二级域名可以通过词素分割为“inter”、“nation”、“al”、“re”和“view”。而对于单词分割，则仅能分为“inter”、“national”和“review”。因此，词素分割的粒度更加细致，而单词分割的粒度较为粗糙。

Mining Vanguard 的架构如图 3.2 所示。Mining Vanguard 包括两个主要的功能模块，

分别是快速匹配模块（黄色部分）和动态推理模块（蓝色部分）。当捕获到一个域名时，首先对该域名进行词素分割，并提取网络解析特征和字符特征。然后执行正则表达式匹配，如果匹配成功则确定该域名是一个矿池域名，并停止后续操作。否则，提取词素特征，并将所有特征拼接在一起输入到分类器，进行进一步的识别。如果分类器预测为矿池域名，则认定该域名为矿池域名并根据一定策略更新匹配库。

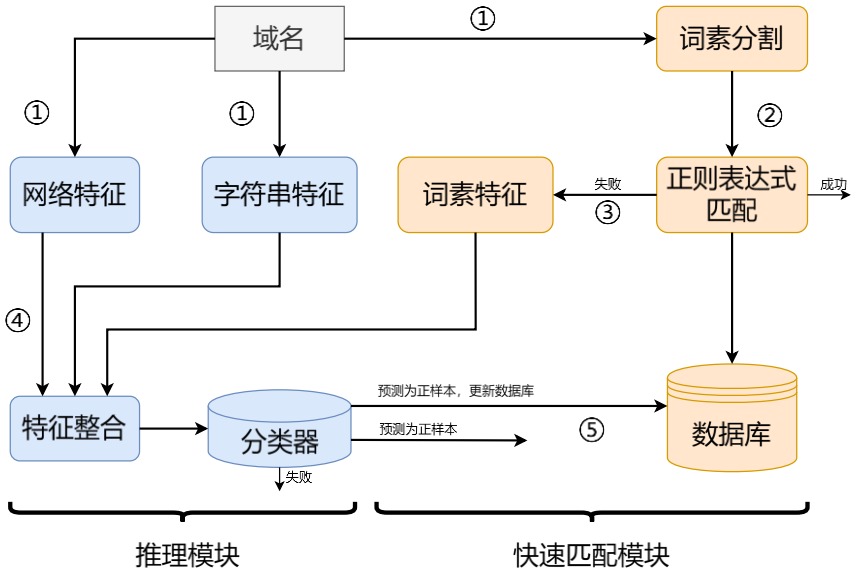

图 3.2 Mining Vanguard 系统整体框架

本节从矿池数据库的搭建、正则表达式匹配方法、机器学习模型的构建，这三个方面对 Mining Vanguard 方法及其中的科学问题予以全面介绍。

3.1.1 矿池数据库的构建

本文构建了三个数据集来搭建检测系统中的矿池数据库，表 3.1 显示了本文所选取的三个数据集的大小和示例，下面本文将对每个数据集进行详细的介绍。

在 RE 数据集中，本文收集了来自奇安信威胁情报平台[96]提供的用于检测矿池域名的矿池域名正则表达式，该数据集包含了 337 个具有代表性的正则表达式，将这些正则表达式存储在数据库中，作为评估 Mining Vanguard 快速匹配模块的初始数据。

Qianxin2023 数据集中的矿池域名来自奇安信威胁情报平台提供的矿池列表，本文对该矿池列表进行了去重处理，最后选取了 5238 个矿池域名作为 Qianxin2023 数据集中的正样本。此外，Qianxin2023 数据集从 Alexa 2023 网站访问量排名列表[89]中，选取了访问量最高的前 100000 条域名作为正常域名，本文对排名前 10 万的域名进行了检查，如果包含矿池域名或者其他恶意域名，将剔除该域名并往下顺延，确保其中均为良

性正常域名。

SDCERT2023 是由国家计算机网络应急技术处理协调中心山东分中心(SDCERT)提供，该数据集是完全在真实网络环境下采集，流量数据采集地点为山东省，分别于 2021 年 12 月和 2022 年 2 月进行了两次捕获，包含了山东省在这两个时间段产生的矿池域名的 DNS 请求，该数据集旨在获取更为全面和具有代表性的挖矿池行为数据，用于测试 Mining Vanguard 在真实环境下的表现，并且有利于后续对加密货币挖矿行为进行深入分析和研究。

表 3.1 矿池数据集种类及其示例

数据集	种类	数量	示例
RE	正则表达式	337	^.*\\.?f2pool\\.com ^.*\\.?alwayshashing\\.com ^.*\\.?webcoin\\.com
Qianxin2023	矿池域名	5238	ca01.supportxmr.com mun.suprnova.cc pool.0cash.org
	正常域名	100000	youtube.com baidu.com google.com
SDCERT2023	矿池域名	2012329	btc.f2pool.com bsty.hashlink.eu etc2.poolgpu.com
	正常域名	2086230	drive.wps.com edge.microsoft.com bilibili.com

3.1.2 快速匹配模块

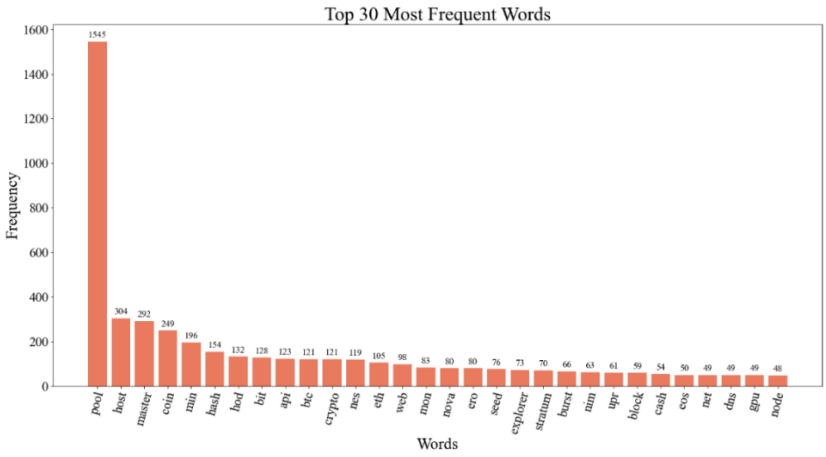

图 3.3 矿池域名词素出现频次统计

快速匹配模块如图 3.2 的黄色部分所示。对于一个包含大量矿池域名且动态更新的庞大数据库而言，如果仅采用顺序匹配的方式，随着新的正则表达式不断更新，所需时间将呈线性增长。假设正则表达式的数量为n，则顺序匹配算法的时间复杂度为$O(n)$。为了降低时间复杂度并加速匹配过程，本文设计了一种基于词素信息的哈希存储方法。

该方法利用了词素信息的特性，将每个正则表达式转换为一个哈希值，并将其存储在哈希表中。这样一来，即使正则表达式数量增加，匹配过程仍能保持稳定的时间复杂度。通过引入基于词素信息的哈希存储方法，成功地降低了匹配过程的时间复杂度，并实现了更快速、更可靠的域名匹配。

通过观察，可以发现在矿池域名中可能包含相同的词素mor_x，如图 3.3 所示。首先对 RE 数据集中出现的所有词素进行统计，当正则表达式中mor_x的数量满足公式

$$count(mor_x) \geq th \tag{3.1}$$

算法 1. $RE-Hash$初始化
输入：挖矿域名的正则表达式列表$REList$，矿池词素列表LM
输出：初始化后的$RE-Hash$哈希表$REHash$

```
1  foreach mor_f in LM do //遍历词素列表
2      initialized list l:=None
3      foreach re in REList do
4          if re has mor_f then
5              l.append(re) //将该正则表达式添加到词mor_f所领导的列表中
6              remove re from REList
7          end if
8      end foreach
9      REHash[h(mor_f)] := l //将该词素所领导的列表插入到哈希表中
10 end foreach
```

其中th是预先设定的阈值，当满足式（3.1）的条件时mor_x被定义为矿池词素mor_f，所有的mor_f可以构成一个矿池词素列表LM。然后从所有的正则表达式中筛选出包含矿池词素mor_f的正则表达式，将其分成一个由矿池词素mor_f所领导的顺序列表。此过程循环执行，直到没有剩余的矿池词素mor_f为止。针对剩余的没有被任何矿池词素所领导的正则表达式，需要将其划分到一个独特的顺序列表，本文将其称为词素mor_n所领导的正

则表达式，本文将所有词素领导的顺序列表称为$REList$。这种按照矿池词素进行分组的方式能够更加有效地利用矿池正则表达式之间的相似性和重复性，从而提高数据结构的利用效率。

利用哈希表来存储构建完成的$REList$，定义该哈希表为 RE-Hash，RE-Hash 的结构如图 3.4 所示。RE-Hash 中每个节点的结构为$< key, value >$，其中$key = h(mor_f)$，代表矿池词素mor_f哈希计算后的关键字，而$value$代表由mor_f领导的正则表达式顺序列表，将所有包含矿池词素mor_f的正则表达式依次插入到哈希表中，再将所有剩余的正则表达式插入到 RE-Hash 的最后一个节点中，具体的 RE-Hash 初始化过程如算法 1 所示。

算法 2. $RE-Hash$的匹配过程
输入：待匹配域名$DName$，$REHash$
输出：匹配结果$MFlag$

```
1  MFlag:= False
2  Segment DName into morphemes MorpDName
3  foreach mor in MorpDName do
4  |  MFlag:=False
5  |  if REHash[h(mor)] != None then
6  |  |  foreach re in REHash[h(mor)] do
7  |  |  |  if DName matches re then
8  |  |  |  |  MFlag:= True  //将域名与词素mor所领导的正则表达式匹配
9  |  |  |  |  return MFlag
10 |  |  |  end
11 |  |  end
12 |  end
13 end
13 if MFlag:= False then
14 |  foreach re in REHash[h(mor_n)] do
15 |  |  if DName matches re then
16 |  |  |  MFlag:= True  //将域名与mor_n所领导的正则表达式匹配
17 |  |  |  return MFlag
18 |  |  end
19 |  end
20 end
21 reutrn MFlag
```

根据词素的特点，设计了一个针对矿池词素的哈希函数，设c_i表示词素信息中第i个

字母的 ASCII 码，如果第i个字母为空（即矿池词素长度小于i），则设c_i为 0，针对矿池词素的哈希函数如式（3.2）所示。

$$key=(c_1-97)\times 27^3+(c_2-97)\times 27^2+(c_3-97)\times 27+c_4 \tag{3.2}$$

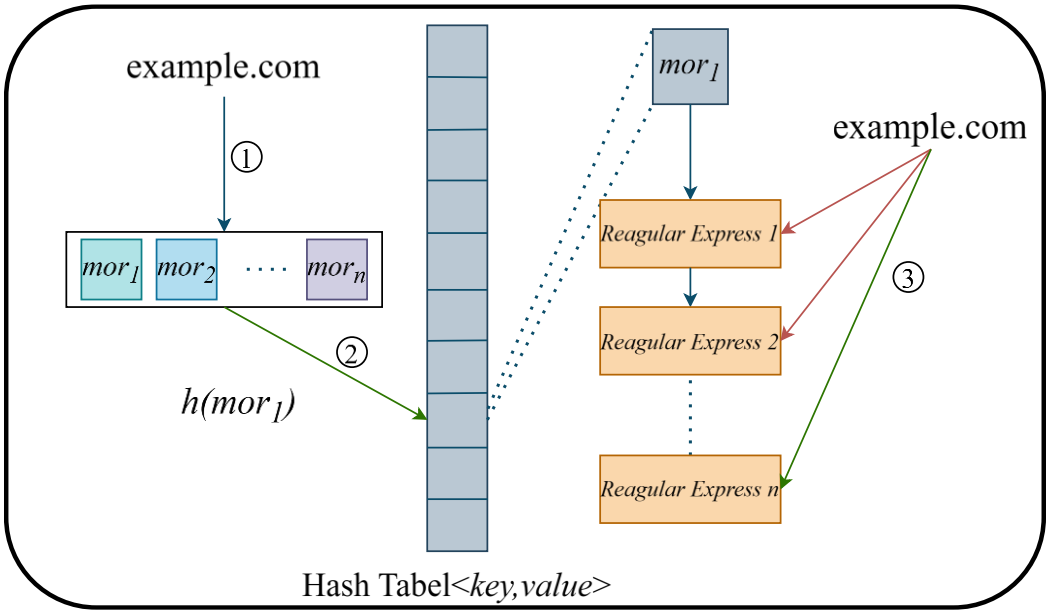

图 3.4 RE-Hash 架构及匹配过程

图 3.4 展示了基于矿池词素的正则表达式匹配过程。首先，对目标域名进行词素分割，然后检索分割后的词素列表中是否含有矿池词素，如果不含有矿池词素，则直接顺序匹配哈希表中最后一个节点所存储的正则表达式顺序列表；如果含有矿池词素，则如图 3.4 中示例所示提取了第一个矿池词素“mor_1”并执行哈希搜索，搜索结果为M_i，因此目标域名需要按顺序匹配由矿池词素“mor_1”领导的正则表达式顺序列表$RE_x|M_i$，如果匹配不成功，则继续重复此操作，直到遍历完所有矿池词素后，顺序匹配哈希表中最后一个节点所存储的正则表达式顺序列表；具体过程如算法 2 所示。

表 3.2 使用的符号及其含义

变量	含义
$\boldsymbol{f(l)}$	匹配l个正则表达式所需要的时间
$\boldsymbol{L}$	数据库中原始矿池正则表达式的数量
$\boldsymbol{\Delta L}$	数据库正则表达式的增加量
$\boldsymbol{N}$	哈希表头的原始数量
$\boldsymbol{p}$	$Pr[count(mor_x)\geq th]$,详见公式 3.1

本文将证明基于词素的正则匹配方法相对于数据库中正则表达式数量具有次线性时间复杂度的论断。为了更清晰地呈现，首先列出了本节中使用的符号，并在表 3.2 中进行了解释。

如图 3.5 所示，线性函数$f(x)$满足$f(x_1+x_2)=f(x_1)+f(x_2)$，这意味着随着输入规模的增加，函数的增长呈线性关系。而函数$g(x)$具有次线性的性质，即$g(x_1+x_2)\leq g(x_1)+g(x_2)$，其增长速率低于线性函数。因此，当算法的时间复杂度为$O(n^m)$时，若$m=1$，则该算法随着输入规模n的增加呈线性增长。然而如果能够证明算法具有次线性特征，就可以得出结论，该算法的时间复杂度为$O(n^m)$，其中$0\leq m\leq 1$。

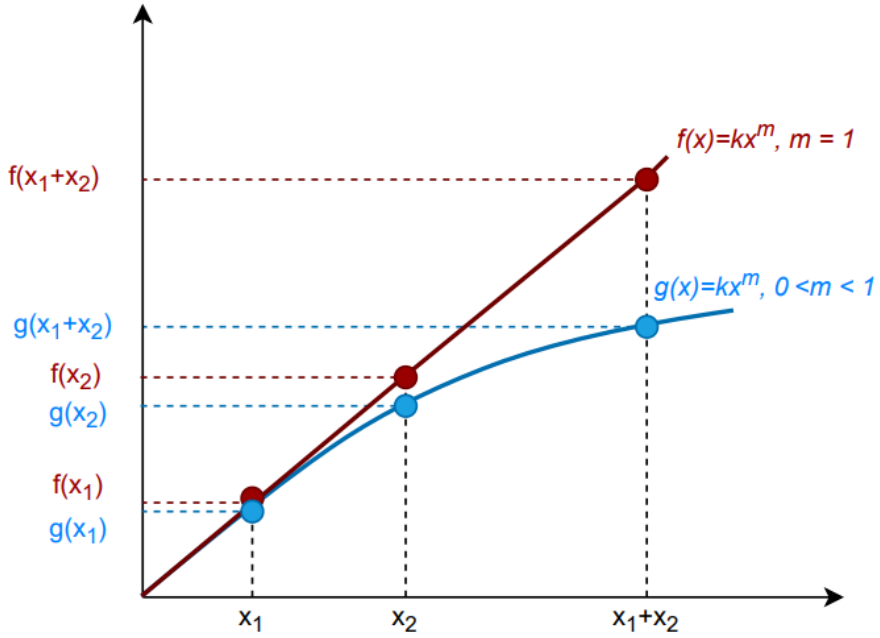

图 3.5 次线性说明

设p代表正则表达式更新集中含有矿池词素的概率，即

$$p=Pr[count(mor_x)]\geq th \tag{3.3}$$

每个节点包含的正则表达式长度不同，但所有节点的数量总和等于L(更新后为$L+\Delta L$)。因此$f(L)$可以被估计为

$$\hat{f}(L)=E[(f(L)]=\frac{c}{N}\times L \tag{3.4}$$

这里c表示单个正则表达式匹配所需时间，更新后新增正则表达式匹配所需时间为

$$\hat{f}(\Delta L)=p\times\frac{c(\Delta L)}{N+1}+(1-p)\frac{c(\Delta L)}{N}=(\frac{p}{N+1}+\frac{1-p}{N})\times c(\Delta L) \tag{3.5}$$

同理可得

$$\hat{f}(L+\Delta L)=(\frac{p}{N+1}+\frac{1-p}{N})\times c\times(L+\Delta L) \tag{3.6}$$

式（3.6）−式（3.5）−式（3.4）可得

$$\begin{aligned}&\hat{f}(L+\Delta L)-\hat{f}(\Delta L)-\hat{f}(L)\\&=(\frac{p}{N+1}+\frac{1-p}{N})\times c\times(L+\Delta L)-(\frac{p}{N+1}+\frac{1-p}{N})\times c\times(\Delta L)-\frac{c}{N}\times L\\&=(\frac{p}{N+1}-\frac{p}{N})\times c\times L\le 0\end{aligned}\tag{3.7}$$

因此证明了$\hat{f}(L+\Delta L)\le\hat{f}(L)+\hat{f}(\Delta L)$，这表明随着正则表达式列表的增长，该方法仍具有较低的时间复杂度。这意味着随着数据集的扩大，本文提出的匹配方法能够更有效地处理更多的正则表达式，从而提高了系统的性能和扩展性。

3.1.3 动态推理模块

由于匹配模块无法在没有任何先验知识的情况下构建正则表达式，所以对未知或零日矿池域名没有识别能力，为了解决这一难题，本文提出了一种基于机器学习的未知的矿池域名的检测方法。首先对待测域名进行特征提取，特征集合如表 3.3 所示，域名D的特征提取主要包括三个部分，分别是词素特征、网络特征和字符串特征。通过结合这些特征，利用机器学习算法来训练模型，从而识别未知的矿池域名。下面本文根据特征类型对提取的每个特征进行详细的介绍。

表 3.3 动态推理模块使用的特征集合

特征类型	特征名称
词素特征	词素的数量
	强/弱词素的数量
	强/弱词素的比例
字符串特征	分隔符（.）的数量
	字符串熵
	字母数字的转换频次
	元音字母的比例
	域名字符串的长度
网络特征	解析 IP 地址的数量
	域名解析响应时间
	顶级域名的长度
	顶级域名的 ASCII 码和
	TTL 平均值

（1）词素特征：在特征提取阶段，将继续深入使用基于词素的特征。与快速匹配阶

段不同的是，在此阶段将词素划分得更加详细。当词素mor_x满足公式

$$th \le mor_x < th^* \tag{3.8}$$

本文将其定义为弱词素，这些词素是在矿池域名中较为常见的词素，但其可能并不带有很明显的矿池语义暗示。当词素mor_x满足公式

$$th^* \le mor_x \tag{3.9}$$

这表明该词素在挖矿域名中具有很高的频率，而且它们一般具有很强的矿池语义暗示（比如 pool），因此被定义为强词素。下面本文将详细介绍每个词素特征的含义和选择的理由。

- 词素的数量：指域名D中包含词素的数量。一般来说，非矿池域名通常比矿池域名包含更多富有语义的词素。
- 强/弱词素的数量：包含强词素的域名通常有很强烈的挖矿行为相关的语义暗示，弱词素在域名中通常挖矿行为相关的语义暗示较弱。将强弱词素分别统计有助于区分矿池词素和不相关词素。
- 强/弱词素的比例：域名D中强/弱词素占所有词素的比例。挖矿域名可能包含更多强词素。对于非挖矿域名，尽管它们可能包含一些矿池词素，但它们通常所占比例较低。

（2）字符串特征：为了避免矿池域名使用域名生成算法（DGA）随机生成域名以逃避监管，增加了以下字符串特征，以使分类器能够对随机生成的域名有一定的识别能力。这些特征能够捕捉域名中的模式和结构，使得分类器能够更好地区分合法域名和恶意生成的域名。

- 分割符的数量：指域名D中分隔符（.）的数量。
- 字符串熵：字符串熵代表域名D中字符分布的随机性程度。熵定义为 uni-gram 频率分布，并由式(3.10)计算得出，其中p_c代表字符c在域名D中所占的比例。

$$f = -\sum_{c \in D} p_c \cdot \log_2 p_c \tag{3.10}$$

- 字母数字的转换频次：域名D字符串中字母和数字之间的切换次数。非随机生成的域名通常具有更好的可读性，因此它们较少切换字母和数字。
- 元音字母的比例：表示域名D字符串中元音字母所占的比例。非随机生成的域名通常具有适当的元音字母比例，这是因为非随机生成的域名为了保持较高的可读性。而随机生成的域名通常不会关注域名字符串的可读性，因此元音字母

的比例可能会偏高或者偏低。

- 域名字符串的长度：由于注册短域名的经济成本较高，而随机生成的域名通常具有短期使用的特点，因此随机生成的域名更倾向于选择较长的域名，这些域名可能更难记忆，但在一些恶意行为中具有一定的优势。

（3）网络特征：网络特征能够反映域名和 IP 地址之间的关系，能够为模型识别矿池域名提供更为丰富的信息。

- 解析 IP 地址的数量：指域名D在 DNS 服务器解析后获得的 IP 地址的数量，一些矿池搭建者通过快速更改域名和 IP 地址之间的映射关系来规避检测。因此，这些矿池域名可能解析到更多的 IP 地址。
- 域名解析响应时间：指域名D在 DNS 服务器解析所需要的时间。如果解析失败，则记录为-1。此特征反映了域名解析的效率和稳定性，部分矿池域名可能存在解析效率低或者不稳定的情况，而大部分正常域名则较少出现此类情况。
- 顶级域名的长度和 ASCII 码之和：不同的顶级域名的注册费用有较大差异，而矿池搭建者由于资金问题比较倾向于注册廉价的非常见的顶级域名。
- TTL（time to live）平均值：每个 DNS 响应都有一个 TTL，指示域名解析结果应该被缓存的时间，通常正常域名的 TTL 设置为 1 至 5 天之间。

提取上述特征的目标是训练一个能够区分正常域名和矿池域名的分类器。对分类器进行训练的输入是带有标签的数据集，输出是一个经过训练能够识别矿池域名的分类器，为了获得更好的检测模型，本文对不同的分类算法进行了交叉验证和控制实验，下文中的实验部分展示了该实验的详细过程和结果，最后选择效果最佳的分类器作为 Mining Vanguard 推理模块的机器学习模型。

3.2 Mining Vanguard 方法性能实验

3.2.1 正则表达式匹配速度实验

为了验证基于词素的正则表达式匹配方法和传统的顺序匹配方法在时间性能上的差异，本文进行了对比实验，测试了两种方法在匹配过程中所需的时间。由于每个域名完成匹配所需时间极短，观察单个域名正则匹配的时间并不容易，将每 100 个域名捆绑成一个批次来计算时间会更直观的体现两种方法匹配速度的差异。

实验结果如图 3.6 所示。结果显示，在每个批次上，基于词素的方法平均比传统方

法快约 40 毫秒，这表明了基于词素的匹配方法在匹配速度上的优势。此外，为了进一步评估基于词素的匹配方法的时间性能，实验构建了一个正则表达式数量逐渐增加的正则表达式库，并测试了匹配时间随着正则表达式数量增加的变化情况。

从结果中可以看出，当正则匹配列表变得更大时，基于词素的匹配方法呈现出次线性时间复杂度的优势。这意味着基于词素的匹配方法能够有效地处理大型匹配列表，而不会导致时间复杂度的线性增长。这一结果进一步验证了基于词素的匹配方法在处理大规模数据时的高效性和可扩展性。

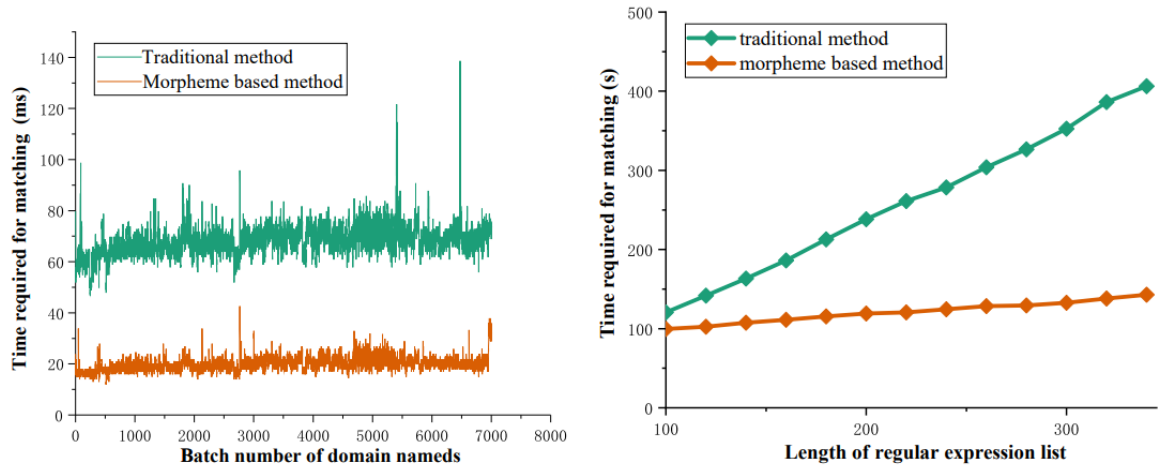

图 3.6 匹配时间比较

3.2.2 分类器算法选择实验

该实验的目标是挑选最合适的分类器算法，由于 Mining Vanguard 的动态推理模块是一个基于监督学习的分类器，因此需要向训练模型提供带标签的数据集，对于这部分实验，使用前文提到的 Qianxin2023 数据集，该数据集包含正常域名和矿池域名，基于奇安信平台信息和 Alexa 对这些域名的标签进行了标记。另外在推理模块的特征提取中，本文将划分强弱词素的阈值t和t^*分别设置为 10 和 50。

在本实验中，利用 Python 中的 Scikit-learn 库训练了四种常见的机器学习模型：支持向量机（SVM）、随机森林（RF）、Adaboost 和 K 近邻（KNN），这些模型是在监督学习任务中广泛应用的经典算法，保留这些模型的默认参数。该实验使用 10 折交叉验证，这是机器学习中的一种标准方法，来测量分类器的泛化误差。在交叉验证中，每一类的样本被分成十个不相交的分区。分类器在 9 个分区的每一组上进行训练，并在剩下的分区中进行测试。在训练完成后，评估了这四种模型在测试集上的性能，比较四者在准确率（Accuracy）、查准率（Precision）、召回率（Recall）、F1-Score 以及 G-Mean

的差异，选择效果最好的模型用于后续的预测任务。Accuracy、Precision、Recall、F1-Score 和 G-Mean 的计算方式如下：

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.11}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.12}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.13}$$

$$F1-score = \frac{2 \times Recall \times Precision}{Recall \times Precision} \tag{3.14}$$

$$G-Mean = \sqrt{\frac{TP}{TP + FN} + \frac{TN}{TN + FP}} \tag{3.15}$$

其中，TP代表被正确预测为正类别（矿池域名）的样本个数，TN代表被正确预测为负类别（正常域名）的样本个数，FP代表被错误预测为正类别的样本个数，FN代表被错误预测为负类别的样本个数。

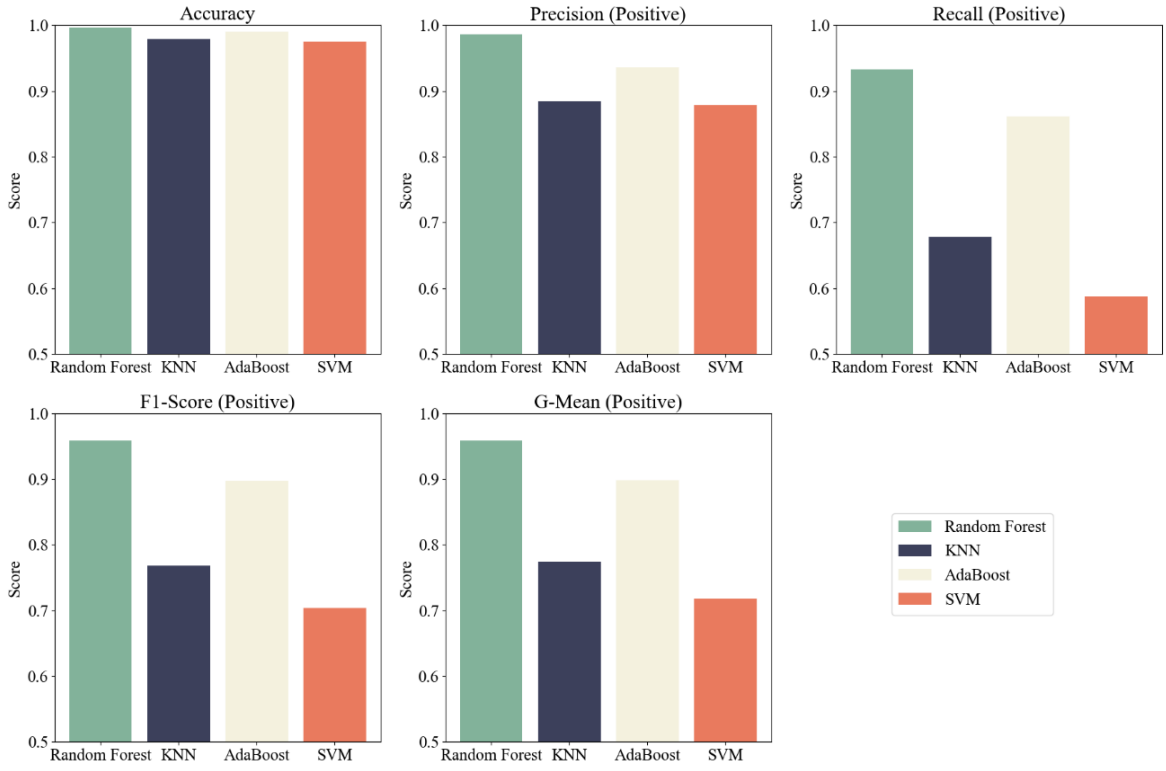

图 3.7 分类器算法选择实验结果

如图 3.7 所示，与其它分类器相比，随机森林能够以超过 98%的查准率和超过 93%的召回率检测到矿池域名。这表明随机森林模型具有最优异的性能。因此，最终选择随机森林作为 Mining Vanguard 中的分类器。

3.2.3 非平衡比例实验

在真实的网络环境中，正常域名占据整个域名空间的绝大多数，而矿池域名所占的比例是极小的，和大多数网络流量分析的任务类似，矿池域名识别任务是一个数据集高度非平衡的任务，因此本文为了验证 Mining Vanguard 在不同的非平衡比例下的表现进行了进一步的实验。

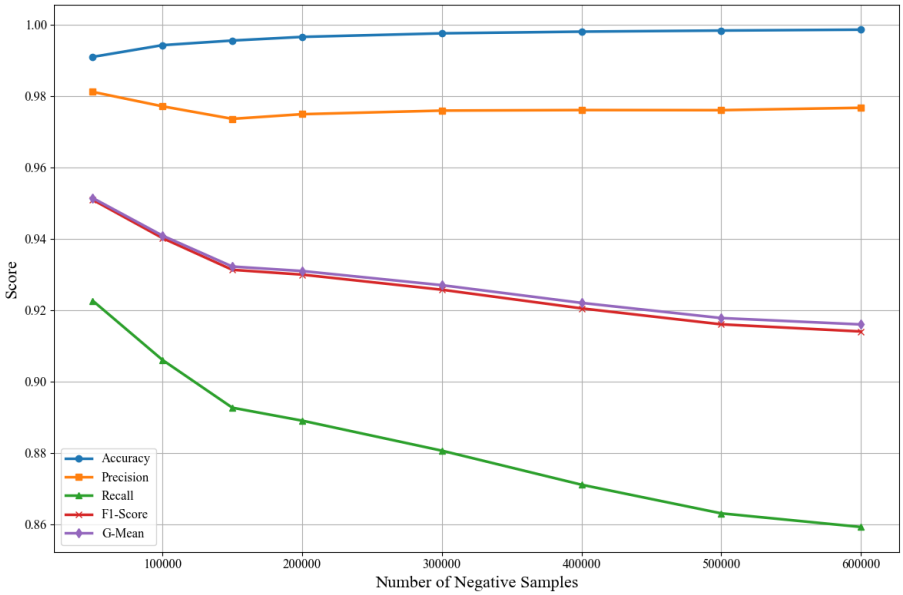

图 3.8 Mining Vanguard 非平衡比例实验结果

该实验使用随机森林模型，数据集使用 Qianxin2023，共 5238 个矿池域名，负样本使用 Alex 网页访问量排名的前 60 万个域名，分别使用前 5 万个、前 10 万个、前 15 万个、前 20 万个、前 30 万个、前 40 万、前 50 万和 60 万个正常域名样本作为负样本进行 8 次实验，虽然 60 万个域名在整个域名空间中仍然只占较小的一部分，但是绝大部分域名的访问量极低，将访问量排名前 60 万的域名作为负样本几乎已经涵盖了人们日常生活访问频率较高的所有域名。图 3.8 展示了非平衡比例实验的结果。从实验结果中可以看出随着非平衡比例的不断提高对于模型检测结果中 G-Mean 并没有大幅下降，而是始终保持在 92%以上，并且面对 60 万的正常域名负样本，Recall 依然能够保持在 85%以上。这意味着在面临较大的非平衡比率时，模型对矿池域名和正常域名仍能保持较好的区分度。

3.2.4 消融实验

为了验证所提出的基于词素的特征的有效性，本文进行了特征消融实验。该实验分

别使用包含词素特征和不包含词素特征的特征集训练了两个分类器（模型均为随机森林）。从图 3.9 中可以看出，基于词素的特征能够将查准率提高 7.3%，召回率提高 18.8%，F1 分数提高 13.8%，G-mean 提高 13.5%。实验结果表明，基于词素的特征能够显著提高 Mining Vanguard 推理模块中矿池域名识别模型的性能。

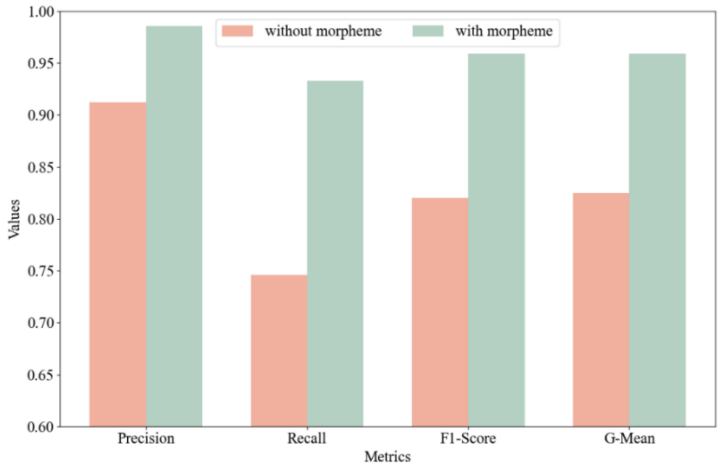

图 3.9 消融实验结果

3.2.5 真实网络环境下评估实验

将 Mining Vanguard 部署在真实的网络环境用于检测挖矿行为，图 3.10 展示了该平台的架构，该平台是包括一台搭载 Nvidia GeForce GTX 3060 图像处理器和 Intel Core I7-12700 CPU 的主动挖矿服务器、一台搭载 Intel Core I5-8300 CPU 的处理器、8GB 内存和 Realtek PCIe GbE 以太网卡的流量采集监测主机。

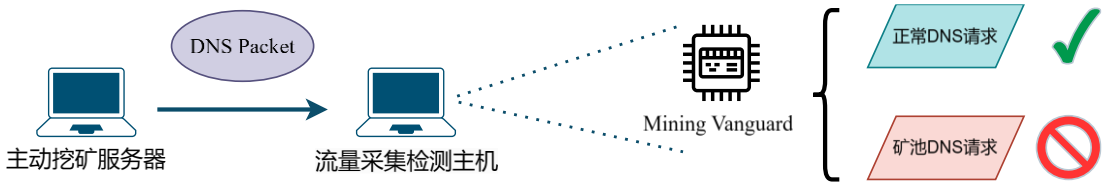

图 3.10 Mining Vanguard 平台架构

该实验使用的数据集为 SDCERT2023 数据集，同时运行检测平台的主动挖矿服务器和流量采集检测主机。主动挖矿服务器对 SDCERT2023 数据集中的每一个域名向 DNS 服务器发送域名解析请求，流量采集监测主机通过捕获 DNS 请求流量监测挖矿行为。该实验一共进行了两次，在第一次实验中，使用 2021 年 12 月份山东省发送的矿池域名请求作为正样本，本文将该正样本集合称为 cert202112，在第二次实验中，使用 2022 年 2 月份山东省内发送的矿池域名请求作为正样本，本文将该正样本集合称为 cert202202，数据集 SDCERT2023 中的其余正常域名作为负样本。

图 3.11 展示了两次实验的结果，从图中可以看出 Mining Vanguard 在两次实验中的检测效果均十分优秀，两次实验中正确率均超过了 99.5%，并且假阳率小于 0.02%。这表明 Mining Vanguard 在真实网络环境中可以检测出大部分的挖矿行为，并且有着较小的误报率。另外两次实验中正样本的捕获时间相隔 3 个月，这意味着较长时间跨度的矿池域名更新并不会对 Mining Vanguard 造成较大的影响，这表明 Mining Vanguard 具有较强的鲁棒性。

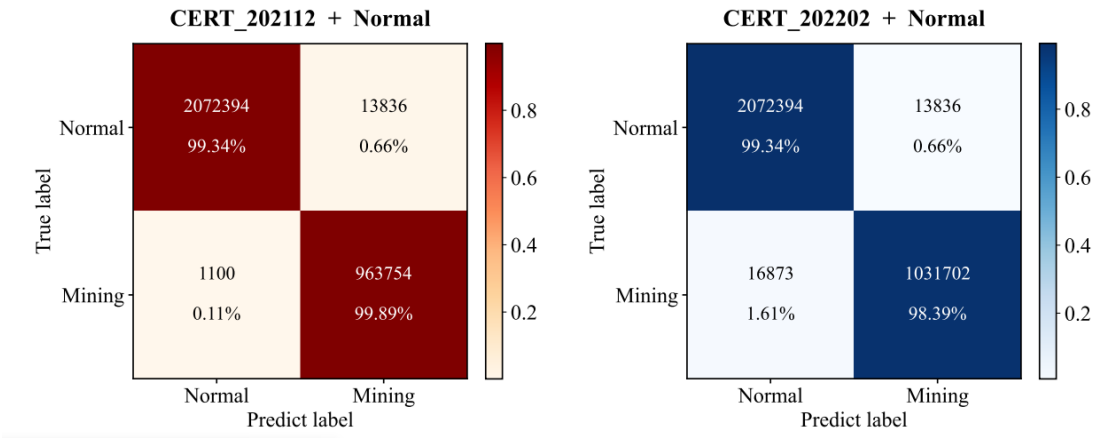

图 3.11 真实网络环境下评估实验结果

3.2.6 与先前的工作对比实验

在本节实验将 Mining Vanguard 与先前工作的方法进行比较，Mining Vanguard 这种基于域名解析行为的挖矿行为识别系统与常见的基于通信流量的挖矿行为识别模型检测及时性的比较。

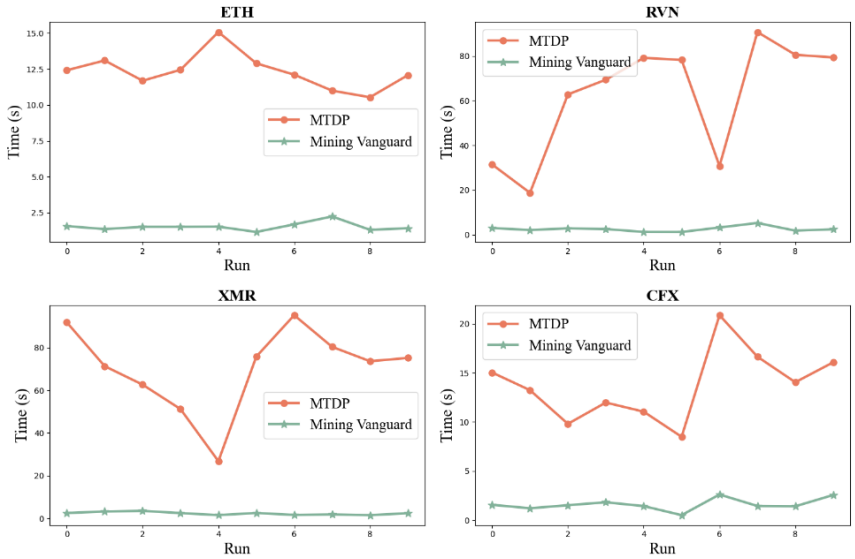

图 3.12 检测时间比较

本文将加密货币挖矿行为开始到检测平台识别出该行为为加密货币挖矿行为这段时间定义为检测所需时间。Sun 等人[80]提出了一种基于卷积的特征提取方法用于加密货币挖矿流量的早期识别，并且搭建了挖矿行为检测平台 MTDP，这种方法需要提取挖矿行为建立的 TCP 连接的前 30 个数据包的包长。在该实验中，平台的主动挖矿服务器分别挖掘了四种如今较为主流的加密货币，币种分别为 ETH、RVN、XMR、CFX，并且每个币种重复挖掘十次。

图 3.12 显示了两种系统的检测所需时间。实验结果显示基于域名解析行为的加密货币挖矿行为识别在早期识别效果上更有优势，检测所需时间较为稳定，并且 Mining Vanguard 比基于通信流量的检测平台 MTDP 平均提前 40 多秒检测到挖矿行为。

3.3 阶层式网络恶意应用识别系统

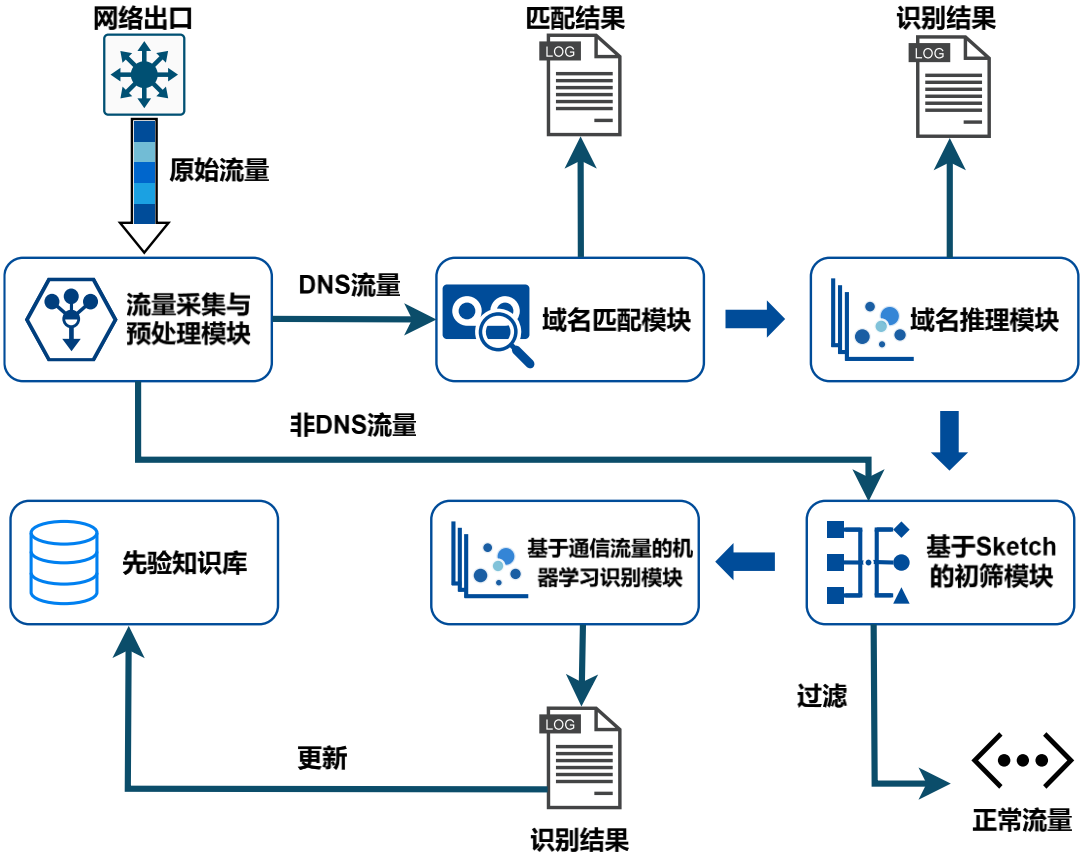

图 3.13 阶层式网络恶意应用识别系统整体框架

基于 3.1 节提出的矿池域名识别方法 Mining Vanguard，本文设计了一个阶层式网络恶意应用识别系统，并且依托于挖矿行为识别这个具体任务进行了相关实验，该系统一方面能够从更多维度检测网络恶意应用，提供检测的准确性，另一方面弥补了仅靠域名

匹配推理模块无法识别加密后的 DNS 流量的缺陷，能够在恶意应用加密通信的情况下依旧实现网络恶意应用行为的准确识别，不受通信协议的限制，具有显著的普适性。

本文以挖矿行为检测这个具体任务为依托，对阶层式网络恶意应用识别系统进行介绍。该系统能够在真实的网络环境下实时监测加密货币挖矿行为，系统架构如图 3.13 所示，该系统主要包括流量采集模块、域名匹配推理模块、基于 Sketch 的初筛模块、机器学习模型模块、先验知识库五个模块组成。该系统代码环境分别为 C 语言和 Python。因为在真实的网络环境下，流量的采集和处理对对时间的要求较高，因此除了域名匹配推理模块和机器学习模块使用 Python 语言完成，其余模块均使用具有高效性能的 C 语言来完成。下面本文将详细介绍该系统的部分核心模块。

3.3.1 流量采集和特征提取

在流量采集模块和特征提取模块，采集分析的设备为装载 Window 系统的主机，且主机网卡的工作模式为混杂模式。在该流量采集主机上采用 WinPcap 开发工具包开发数据采集脚本，对网络园区中的混杂网络流量进行实时捕获。在这个环节需要区分 DNS 流量和非 DNS 流量，因为后续模块中只有域名匹配推理模块需要使用 DNS 流量，并且域名匹配推理模块和机器学习模块所提取的流量特征不同，事先对流量进行筛选，有利于提高后续步骤的分析效率。

具体的分类方法则是通过读取数据包的包头信息，若数据包的目的端口号为 53（DNS 服务的默认端口号为 53），则将其判断为 DNS 请求流量数据包，并且使用哈希表存储该数据包的到达时间。如果数据包的源端口号为 53，则该数据包为 DNS 响应流量数据包，并且将响应数据包的到达时间映射到请求流量数据包同一位置并且计算相隔时间，提取解析后的 IP 地址信息，并将上述信息分发给域名匹配推理模块。如果数据包的源端口和目的端口均不为 53，则判断该数据包为非 DNS 流量数据包，对于非 DNS 数据包，系统会根据其五元组信息（源 IP、目的 IP、源端口、目的端口、协议）汇聚成数据流，并且会将该数据流的包负载大小、包到达时间等结构化信息交付给统计规则筛选模块供后续分析使用。

3.3.2 域名匹配推理模块

Mining Vanguard 部署在该模块，其具体细节和检测流程见 3.1 节。与 3.1 节不同的是，Mining Vanguard 接收到的不再是离线数据，而是由数据采集模块提供的实时网络流

量数据。通过域名匹配推理模块可以实现对挖矿行为的早期识别，如果该模块判断域名为矿池域名，则判定该流量为挖矿流量，不需要再由下面的模块进行检测，直接触发预警。若该模块判断域名非矿池域名，则由下一模块进行检测。

如果利用该系统检测其他网络恶意应用，需要对该模块的数据库和机器学习模型进行更新。

3.3.3 统计规则筛选模块

模块基于 LTC（Long-Tail Clock sketch）[81,82]算法设计而成。与传统的“最频繁项”[83]或“最持久项”[84]算法不同，LTC 旨在从流式数据中寻找前k个重要性最高的元素。根据 LTC 算法的定义，元素的重要性s由其出现频繁度f和出现持久度p共同决定，被表示为二者的加权和，即：

$$s = \alpha f + \beta p \tag{3.16}$$

其中α和β是由特定问题决定的权重参数。算法设计的原则基于网络流量的长尾分布（Long-Tail Distribution）。基于这一原则，算法设计了一个长尾分布时钟和一个二维数组计数器，当一个元素e到达后，会被按照特定的哈希规则存到二维数组计数器中，计数器中会记录其出现的频繁度f；长尾时钟扫描模块用于在一个周期内扫描二维数组中所有元素，并对当前周期内未出现过的元素持久度p降低，从而实现在高速网络环境中对特定重要性元素的无偏估计。

如图 3.14 所示，一些常见网络行为在频繁度和持久度上的分布有明显特点。通常，诸如网络监控流媒体传输等需要高实时性的网络行为，其通信会话的频繁度较高；而网络聊天等需要长时间保持会话通信的行为，其会话的持久度会较高，但是短时间内通信频率不高。根据此特点，本文采用了 LTC 算法用于恶意流量分析的早期筛选。以挖矿行为检测为例，根据对挖矿行为通信模式和矿池-矿机通信协议的长期观察，挖矿行为的通信规则是需要长时间保持通信，且每次通信的信息速率和包数不高，属于典型的“高持久度、低频繁度”模式。因此，通过调整 LTC 算法中两个重要参数α和β，根据式（3.16）计算出各个流量样本的$significance$值大小，根据此度量值进行排序，可以有效地过滤掉大部分$significance$值较小的持续通信的正常流量，仅保留少部分$significance$值较大的有挖矿通信模式特征的流量，用于后续模块进一步检测，大大降低了后续模块的工作负载，提高了系统整体的有效性和可靠性。

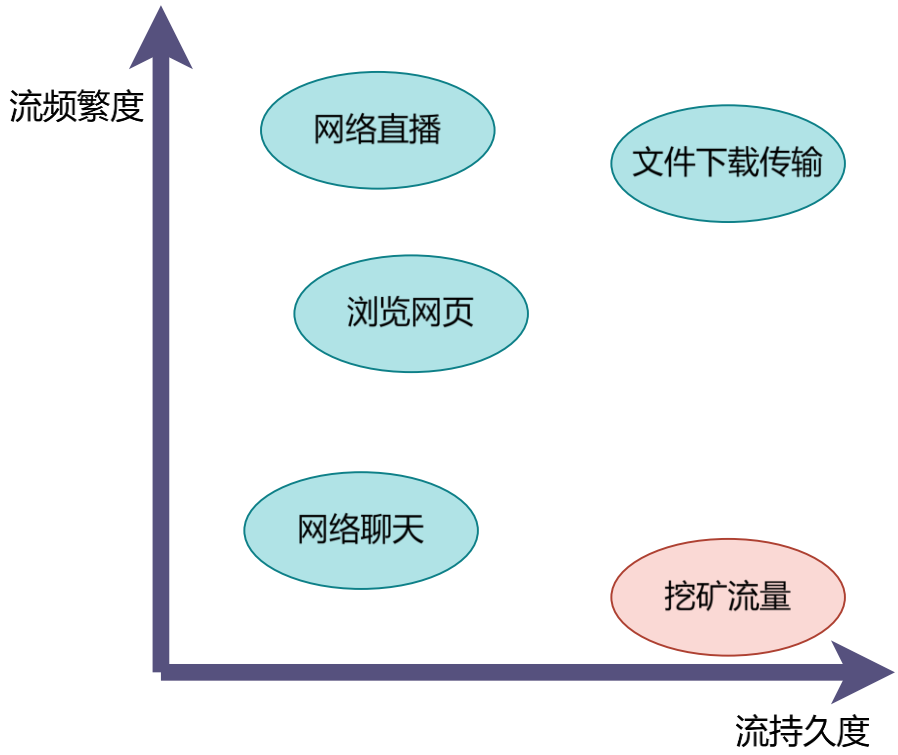

图 3.14 常见网络行为频繁度和持久度示例

3.3.4 机器学习模型模块

该模块的目的是利用机器学习模型对筛选后的网络流进行最终的识别，如果识别为恶意流量，则对恶意流量进行清洗整理，并且将该网络流对应的域名和流量信息录入先验知识库，实现对先验知识库的动态更新。

另外，本文以挖矿行为检测具体任务为依托，基于挖矿行为的通信模式进行针对性的特征提取，目的是训练出一个能够区分挖矿流量和非挖矿流量的机器学习模型。图 3.15 展现了挖矿行为通信时产生的消息类型。挖矿行为的通信模式通常需要遵守一定的应用层协议（如 Stratum 协议）进行节点注册、分发任务和提交结果等事件，但是不管矿池采用何种协议类型，必然会有四种类型的消息来进行必要的挖矿操作：登录或注册消息msg_l、登录确认消息msg_c、任务分配消息msg_{allo}和任务提交消息msg_r。因此可以将挖矿行为的通信划分为两个阶段，分别是矿工登录阶段和矿工计算阶段。矿工登录阶段主要用于矿池和二级矿工之间的握手和矿工登录过程，进行必要的哈希算法协商。msg_l和msg_c通常出现在此阶段且仅出现一次。矿工计算阶段交替发送上行和下行数据包用于任务的分发和的提交，该部分是进行挖矿行为识别的关键。msg_{allo}和msg_r在该阶段穿插出现，且分别主导了下行流量和上行流量。

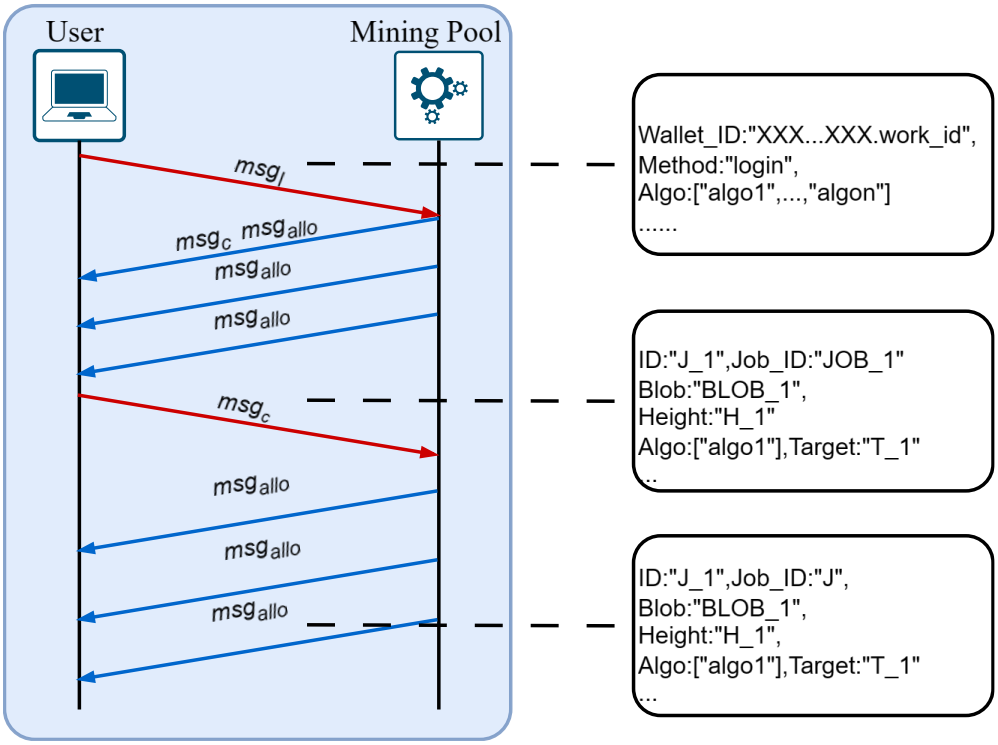

图 3.15 矿工与矿池之间通信产生的网络消息

本文将流量采集模块分发到机器学习模块的网络流定义为$f_i = (p_{i1}, p_{i2}, \ldots, p_{in})$，包长序列$Len_i = len(f_i) = (l_{i1}, l_{i2}, \ldots, l_{in})$，$l_{in} > 0$表示$p_{in}$为上行流量数据包，$l_{in} < 0$表示$p_{in}$为下行流量数据包。通过采集 8 种主流的加密货币挖掘过程中产生的通信流量，然后提取每条 IP 流的包长序列，如图 3.16 所示，其中 X 轴代表流f_i的包序号，删除了每条流的握手包，得到预处理后的 IP 流$\tilde{f} = f|_{st}^{n}$，Y 轴代表数据包有效负载大小l_{in}，绿色散点代表下行流量，红色散点代表上行流量。

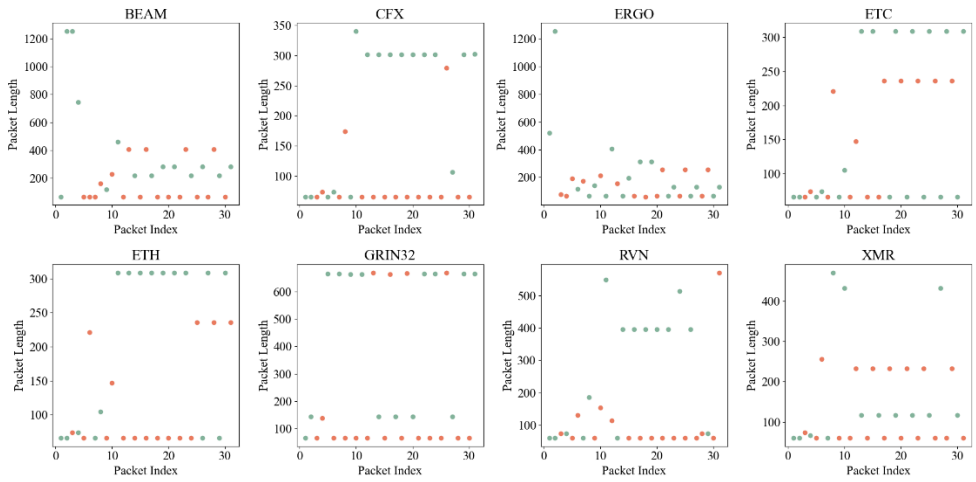

图 3.16 8 种加密货币的包长序列

从图 3.16 可以看出挖矿流量包长序列较为整齐，且具有一定的周期性，这和上文提

到的挖矿行为在进行通信时发送的消息类型相符合，同一币种由于分配的任务计算复杂度相似，因此包长序列较为整齐。不同币种之间矿池与矿工的通信流量存在差异，主要体现在下行流量的有效负载和上下行流量的穿插频率两个方面，这可能是因为不同的币种分配的任务计算复杂度不同导致的。

基于上述的发现，尝试构建基于包长的流量指纹。该流量指纹不需要使用时间序列特征，因为挖矿行为产生的网络流量中的时间序列特征容易被矿机的性能所影响，导致。msg_{allo}和msg_r的发送频率不同，因此该特征不具备较好的泛化性。为了更加关注挖矿行为中 IP 流的上下行流量交替和上下行流量中包长较为稳定的特征，本文设计了三个方面的关键特征构建流量指纹。

（1）N-Gram 特征：基于包长序列Len_i进行更加精准的特征提取，目的是为了提取出矿工计算阶段的主要特征。本文采用 N-Gram 算法对包长序列Len_i进行分割，将包长序列Len_i切割成连续的 bigram，得到 bigram 序列$B = \{b_1, b_2, \dots, b_n\}$，然后统计$B$中每个 bigram 的出现次数，并将它们记录在频次表C中，$C[b_i]$表示b_i在序列B中出现的次数，$\tilde{B}$表示从集合B中选择前x个出现次数最多的 bigram，综上，$\tilde{B}$可由式（3.17）得出。

$$\tilde{B} = top(x, \{b \mid C[b]\}) \tag{3.17}$$

$top(x, A)$表示从集合A中选择前x个最大值。

（2）交互频率特征：基于包长序列Len_i提取任务分配消息msg_{allo}和任务提交消息msg_r的交互频次，对于挖矿行为产生的网络流量$\tilde{f}$中上下行流量的交互频次一定程度上反映了挖矿行为中任务分配消息msg_{allo}和任务提交消息msg_r的交互频次，所以可以通过设计一个计数器num，用来统计包长序列Len_i中上下行流量的交替次数。另外本文提出了一个流量块的概念，当包长序列Len_i中连续出现两次或两次以上相同数值时，这段子序列被称为流量块，流量块的出现代表该阶段上行流量或者下行流量占据主导地位，分别统计上行流量块的最大长度upl和下行流量块的最大长度$downl$作为挖矿行为识别的流量块特征。

（3）统计特征：基于包长序列Len_i分成三个序列，分别是上行流量包长序列，下行流量包长序列以及完整的包长序列。分别计算每个包长序列中以下统计特征：最小值min、最大值max、平均值avg、标准偏差sd、中值绝对偏差mad、方差var、峰度kur、百分位数per。对于这些统计特征，使用随机森林分类器来获得基尼指数，在分类器构造期间计算基尼指数，用来评估每个特征的贡献度，然后可以通过特征贡献分数来评估每个

统计特征的贡献度，选取贡献分数超过一定阈值的统计特征来减低特征维度和保证分类的准确性。

3.4 阶层式网络恶意应用识别系统性能实验

3.4.1 数据集采集

当前学术界和工业界对于虚拟货币挖矿行为的分析工作尚处于初级阶段，相关研究较为稀缺，因此公开的流量数据集极少。为了完成数据采集，需要建立一个安全可控的仿真环境。这个环境包括以下设备：一台主动挖矿服务器，配置有 Nvidia GeForce GTX 3060 图像处理器和 Intel Core i7-12700 CPU；一台流量采集预处理主机，配置有 Intel Core i5-8500 处理器、16GB 内存以及 Realtek PCIe GbE 以太网卡；以及一台 Huawei AR100-S 企业接入级无线管理路由器。

表 3.4 挖矿流量数据集说明

	XMR	RVN	GRIN32	ETH	ETC	ERGO	BEAM	CFX
流量样本（个）	26	24	18	28	26	23	25	26
数据大小（KB）	418	494	318	3120	676	314	590	4977

本文共采集了八种加密货币币种，每种币种的挖掘都是通过主动挖矿服务器连接到公共矿池，然后流量采集预处理主机根据矿池的 IP 地址和端口号对混杂流量进行过滤，从而得到挖矿流量的 Pcap 文件。最终的数据集如表 3.4 所示。为了丰富数据集并增加数据的多样性，本文采用了一个公共数据集 CJ-Sniffer[85]，值得注意的是，在使用公共数据集时遵循了该数据集的许可和使用规定，并且将其添加到自行采集的数据集中，以提高研究的可信度和有效性。CJ-Sniffer 中包含了由用户发起的加密货币挖矿和非法的加密货币挖矿活动生成的经过标记的、数据包级别的网络流，该数据集中的加密货币币种均为门罗币（XMR），一共包含 526 条流量样本。需要注意的是自采的数据集中存在部分币种（GRIN32、ETC）已经不被支持，这些币种的流量为其停止运营之前被采集的。另外本文还在真实的网络环境中采集了正常流量样本，共计正常行为流量 12315 例。

3.4.2 统计规则模块实验

在该部分实验中，通过统计规则模块的 LTC-Sketch 技术对网络流量进行处理和分析，旨在解决流量分布的非平衡性问题。实验前，为了测试统计规则模块的初筛性能，本文将正常流量和挖矿流量的非平衡比率调整到 1442:1，这表示流量分布极为不均，导

致数据处理过程中的资源分配效率低下。

通过将 LTC-Sketch 应用于流量数据的预筛选过程中，成功将非平衡比率显著降低至 22:1。图 3.17 展示了非平衡比率的变化，这一显著的改进展示了 LTC-Sketch 在处理高度非平衡流量数据方面的强大能力，平衡比率的下降会降低后续机器学习模块从混杂流量中识别出挖矿流量的压力，基于此比例数据集设计了一项对比实验，旨在对比使用 LTC-Sketch 和不使用 LTC-Sketch 在算力消耗指标（本实验将其简化为识别所需时间）上的差异，图 3.18 展示了这次对比实验的结果，从实验结果中可以看出，通过 LTC-Sketch 对混杂网络流量的过滤，机器学习模块在检测所算力消耗上约减少了 82.5%。

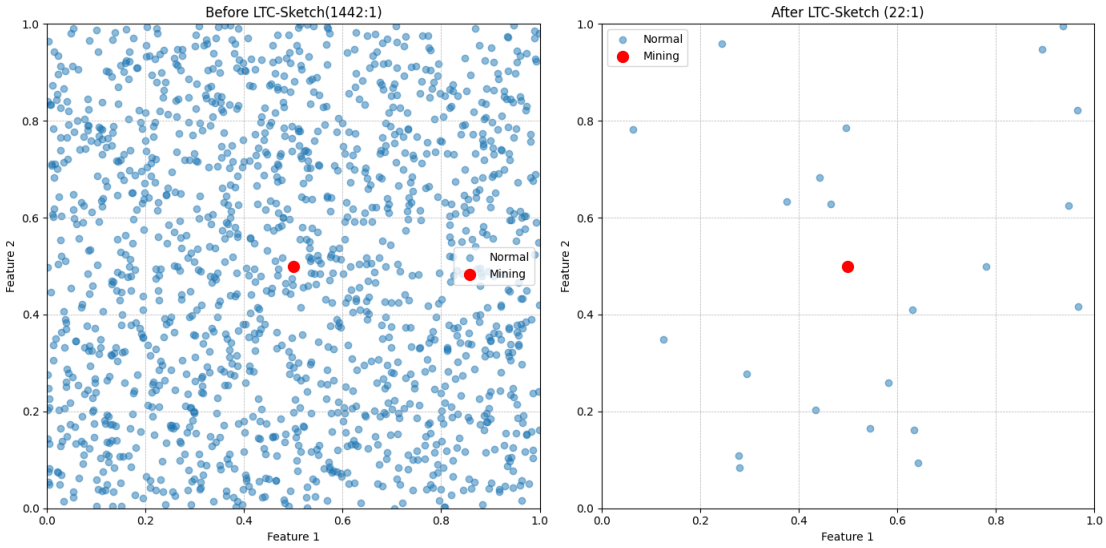

图 3.17 统计规则模块实验结果

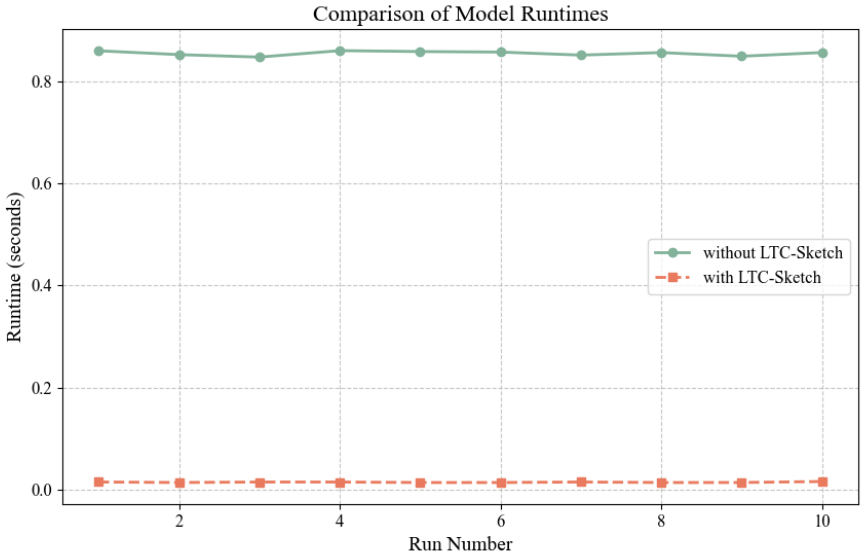

图 3.18 LTC-Sketch 对比实验

3.4.3 机器学习模型识别效果评估及模型选择

该实验的目标是挑选最合适的分类器算法部署在机器学习模块识别模块。在该实验过程中，本文针对四种不同的分类器（Random Forest、SVM、AdaBoost 和 KNN）进行了十折交叉验证实验。在每个折叠中，将数据集划分为训练集和测试集，并使用训练集对模型进行训练，然后在测试集上进行预测，并计算出召回率（Recall）、精确率（Precision）、准确率（Accuracy）和 G-Mean 的值。

为了评估实验效果并选出最佳的分类器，分别计算了每个分类器在每个指标上的平均值和标准差。通过比较四种分类器在四个指标上的平均值可以确定哪个分类器在整体上表现最好。此外，通过考虑每个指标的标准差可以评估分类器的稳定性，以确保其在不同数据集上的性能一致性。实验结果如图 3.19 所示。

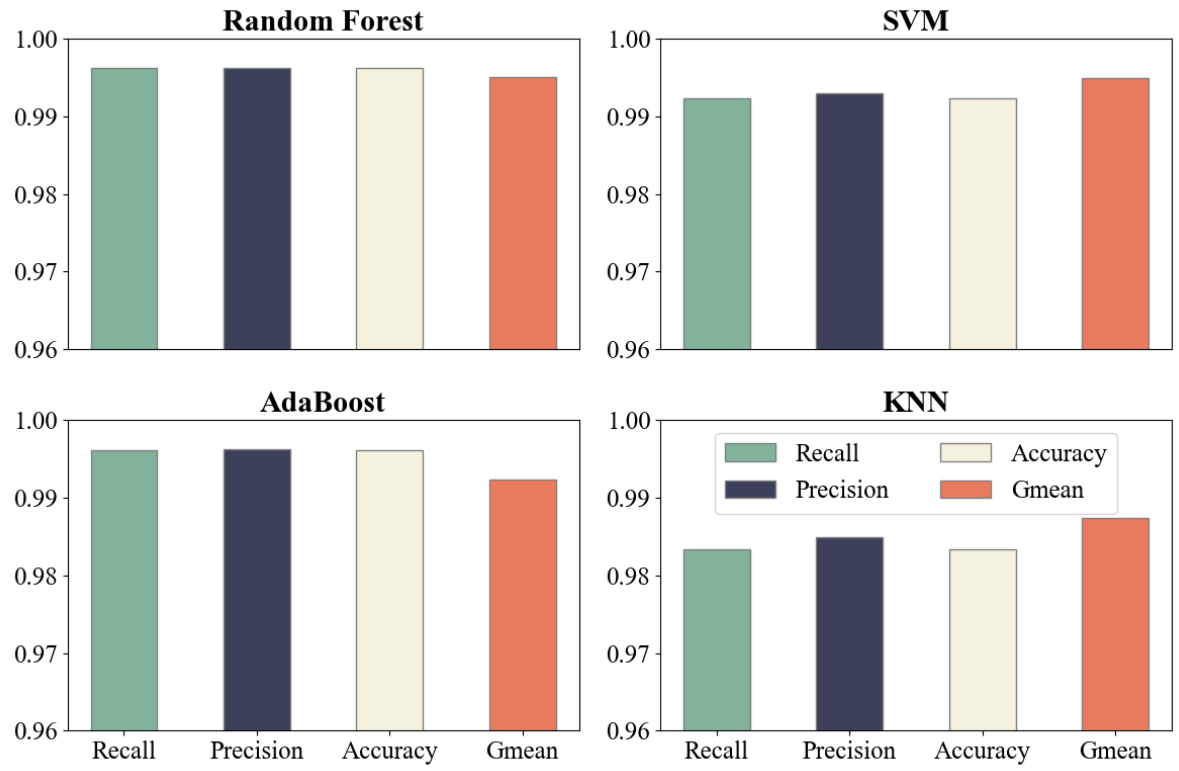

图 3.19 机器学习模块模型性能比较

可以看出四种分类器的性能均比较好，这体现出了提取的特征的稳定性和有效性。横向对比来看随机森林的效果明显高于 SVM 和 KNN 分类器，和 Adaboost 分类器性能基本一致，另外在模型稳定性方面，通过十折交叉验证实验显示，随机森林和 Adaboost 的各项指标标准差基本在 0.005 左右，而 SVM 和 KNN 的各项指标标准差在 0.01 左右，这证明了随机森林和 Adaboost 在性能的稳定性上也比 SVM 和 KNN 更加优秀。

3.4.4 阶层式网络恶意应用识别系统整体性实验

本实验将阶层式网络恶意应用识别系统部署在真实网络环境下评估该系统的整体性能。首先流量采集器对测试终端产生的网络流量进行采集，该测试终端会进行一些日常网络活动（比如浏览网页、观看网络视频和下载文件等）和主动挖矿行为。实验整体结果如表 3.5 所示。

模块一代表域名匹配推理模块，模块二代表机器学习模型模块。从实验结果中可以看到正常流量 483 条中有 5 条被识别为挖矿流量，误报率仅为 0.10%。对于 6 种币种的挖矿行为共产生流量 30 条，全部识别为正样本，域名匹配推理模块识别出 16 个矿池域名，机器学习模型模块识别出 14 条挖矿流量，需要注意的是在该系统的运行逻辑是当域名匹配推理模块识别出该条流量所对应的 DNS 请求种的域名为矿池域名，则机器学习模型模块则不检测该条流量，直接视为挖矿行为。

表 3.5 系统整体实验结果

	流数量（条）	模块一预测为正类	模块二预测为正类	预测为负类
Web	6	0	0	6
Video	10	0	0	10
File downloading	5	0	0	5
Cloud-gaming	8	0	0	8
Remote-control	2	0	0	2
Other	452	0	5	447
XMR	5	3	2	0
ERGO	5	2	3	0
RVN	5	1	4	0
BEAM	5	2	3	0
ETH	5	3	2	0
ETC	5	5	0	0

3.5 本章小结

本章中重点介绍了本文提出的 Mining Vanguard 方法以及阶层式网络恶意应用识别系统，该方法创新地从域名解析行为分析的角度进行挖矿行为检测，并且还基于该方法提出了一种阶层式网络恶意应用识别系统，兼顾了系统的实时性和准确性。在本章节中首先介绍了矿池域名的采集过程和 Mining Vanguard 的整体框架；然后详细地介绍了快速匹配和动态推理两个模块的具体流程，以及各个环节的具体方法及其对应的科学问题。其中，针对矿池域名的特点，创新地提出了基于词素的快速匹配方法和动态推理方法，

通过在真实的网络数据集上进行大量实验验证了 Mining Vanguard 的有效性；然后为了让整体检测框架更加完善，提出了一种阶层式网络恶意应用识别系统。该方法在纯机器学习模型之上增加了更有针对性的检测工作，一方面通过监测 DNS 流量，提取待测域名进行规则匹配和模型推理判断是否存在发送挖矿请求域名解析请求，以此来推断主机是否已存在或即将存在挖矿行为。另一方面，该系统持续观察 IP 会话流，并使用 LTC 技术过滤掉大部分非挖矿流量，仅保留疑似挖矿流量。通过设计的大量实验表明，本文设计识别方法在恶意挖矿行为早期就可以及时产生预警响应，通过阶层式的设计，降低了机器学习模型面临的数据非平衡率，进而在识别准确率、漏报率等指标上表现突出。

3.6 本章附录

表 3.6 阶层式网络恶意应用识别系统机器学习模块模型性能比较

KNN	Average Accuracy=0.9834, Average Recall=0.9834 Average Precision =0.9849, Average G-mean =0.9874									
Accuracy	1.000	1.000	0.9873	0.9620	0.9747	0.9872	1.000	0.9744	0.9615	0.9872
Precision	1.000	1.000	0.9884	0.9670	0.9778	0.9881	1.000	0.9769	0.9624	0.9881
Recall	1.000	1.000	0.9873	0.9620	0.9747	0.9872	1.000	0.9744	0.9615	0.9872
G-mean	1.000	1.000	0.9926	0.9742	0.9845	0.9915	1.000	0.9832	0.9555	0.9915
SVM	Average Accuracy=0.9924(std:0.0102), Average Recall=0.9924(std:0.0102) Average Precision =0.9930(std:0.0093), Average G-mean =0.9950(std:0.0067)									
Accuracy	1.000	1.000	1.000	0.9747	0.9873	1.000	0.9872	1.000	0.9744	1.000
Precision	1.000	1.000	1.000	0.9770	0.9882	1.000	0.9878	1.000	0.9769	1.000
Recall	1.000	1.000	1.000	0.9747	0.9873	1.000	0.9872	1.000	0.9744	1.000
G-mean	1.000	1.000	1.000	0.9829	0.9923	1.000	0.9915	1.000	0.9832	1.000
Adaboost	Average Accuracy=0.9962(std:0.0058), Average Recall=0.9962(std:0.0058) Average Precision =0.9963(std:0.0057), Average G-mean =0.9924(std:0.0135)									
Accuracy	1.000	0.9873	1.000	1.000	0.9873	1.000	1.000	1.000	1.000	0.9872
Precision	1.000	0.9875	1.000	1.000	0.9875	1.000	1.000	1.000	1.000	0.9879
Recall	1.000	0.9873	1.000	1.000	0.9873	1.000	1.000	1.000	1.000	0.9872
G-mean	1.000	0.9682	1.000	1.000	0.9636	1.000	1.000	1.000	1.000	0.9916
RF	Average Accuracy=0.9962(std:0.0058), Average Recall=0.9962(std:0.0058) Average Precision =0.9963(std:0.0057), Average G-mean =0.9924(std:0.0096)									
Accuracy	1.000	0.9873	1.000	0.9873	1.000	1.000	1.000	1.000	1.000	0.9872
Precision	1.000	0.9875	1.000	0.9875	1.000	1.000	1.000	1.000	1.000	0.9879
Recall	1.000	0.9873	1.000	0.9873	1.000	1.000	1.000	1.000	1.000	0.9872
G-mean	1.000	0.9682	1.000	0.9636	1.000	1.000	1.000	1.000	1.000	0.9916

表 3.7 LTC-Sketch 对比实验

	Without LTC-Sketch Average Runtime =0.853, With LTC-Sketch Average Runtime=0.0145									
Without LTC-Sketch	0.851	0.854	0.847	0.866	0.852	0.854	0.858	0.856	0.845	0.852
With LTC-Sketch	0.015	0.014	0.015	0.015	0.013	0.014	0.015	0.014	0.014	0.016

表 3.8 Mining Vanguard 模型性能比较

KNN	Average Accuracy=0.9795, Average Recall=0.6785 Average Precision =0.8848, Average G-mean =0.7747，F1-Score=0.7678									
Accuracy	0.981	0.9807	0.9773	0.9796	0.9802	0.9778	0.9802	0.9802	0.9776	0.9805
Precision	0.9039	0.8575	0.8575	0.8894	0.8891	0.8683	0.8930	0.9136	0.8714	0.8785
Recall	0.7003	0.6545	0.6545	0.6755	0.6889	0.6851	0.6851	0.6660	0.6481	0.7055
G-mean	0.7003	0.6545	0.6545	0.6755	0.6889	0.6851	0.6851	0.6660	0.6481	0.7055
F1-Score	0.7892	0.7492	0.7424	0.7751	0.7763	0.7753	0.7753	0.7704	0.7434	0.7826
SVM	Average Accuracy=0.9754, Average Recall=0.5874 Average Precision =0.8782, Average G-mean =0.7181，F1-Score=0.7036									
Accuracy	0.9762	0.9750	0.9737	0.9764	0.9771	0.9731	0.9762	0.9748	0.9739	0.9772
Precision	0.8914	0.8517	0.8757	0.8750	0.8879	0.8452	0.8914	0.8912	0.8784	0.8944
Recall	0.5954	0.6030	0.5515	0.6145	0.6202	0.5629	0.5954	0.5629	0.5525	0.6156
G-mean	0.7285	0.7166	0.6949	0.7332	0.7421	0.6898	0.7285	0.7083	0.6967	0.7420
F1-Score	0.7139	0.7061	0.6768	0.7219	0.7303	0.6758	0.7139	0.6900	0.6784	0.7293
Adaboost	Average Accuracy=0.9901, Average Recall=0.8615 Average Precision =0.9359, Average G-mean =0.8979，F1-Score=0.8971									
Accuracy	0.9905	0.9900	0.9905	0.9900	0.9903	0.9895	0.9910	0.9904	0.9882	0.9908
Precision	0.9417	0.9249	0.9381	0.9301	0.9432	0.9312	0.9479	0.9380	0.9235	0.9402
Recall	0.8645	0.8702	0.8683	0.8645	0.8568	0.8530	0.8683	0.8664	0.8317	0.8718
G-mean	0.9023	0.8971	0.9025	0.8967	0.8990	0.8912	0.9072	0.9015	0.8764	0.9054
F1-Score	0.9014	0.8967	0.9018	0.8961	0.8980	0.8904	0.9063	0.9007	0.8752	0.9047
RF	Average Accuracy=0.9942, Average Recall=0.9060 Average Precision =0.9770, Average G-mean =0.9408，F1-Score=0.9402									
Accuracy	0.9945	0.9941	0.9941	0.9941	0.9943	0.9948	0.9943	0.9919	0.9942	0.9960
Precision	0.9814	0.9733	0.9772	0.9695	0.9754	0.9776	0.9874	0.9600	0.9832	0.9858
Recall	0.9083	0.9064	0.9026	0.9103	0.9103	0.9179	0.8988	0.8738	0.8988	0.9330
G-mean	0.9442	0.9393	0.9392	0.9394	0.9423	0.9473	0.9420	0.9159	0.9401	0.9587
F1-Score	0.9435	0.9387	0.9384	0.9389	0.9417	0.9468	0.9410	0.9149	0.9391	0.9597-

第四章 基于加密 DNS 流量的网站指纹识别

4.1 基于加密 DNS 流量的网站指纹识别概述

近年来为了加强对 DNS 隐私性的保护，业界有关机构和研究人员提出了 DNS over HTTPS（DoH）、DNS over TLS（DoT）加密传输协议，许多 DNS 服务提供商也对这些加密协议进行了支持。最近，DNS over QUIC（DoQ）这一旨在以最小延迟提供 DNS 隐私保护的新协议正式成为拟议标准，QUIC 协议解决了队头阻塞，提供多路复用并且通过减少握手次数和优化连接建立过程，降低了连接建立的延迟。因此，使用 QUIC 传输协议提供 DNS 服务不仅是传统的以性能为导向的 DNS 协议（DoUDP、DoTCP）的进化，也是隐私保护 DNS 协议（DoT、DoH）的进化。但是随着加密 DNS 协议的兴起，传统的基于域名黑名单的网络监管方法也逐渐失效，比如通过部署加密 DNS 协议来访问某些恶意网站，可以绕过中国防火长城而成功访问，而使用非加密 DNS 协议则会访问失败。这意味这加密 DNS 协议加强了用户隐私安全的同时，也给一些不法分子带来了可乘之机。

通过网络流统计特征或时间序列特征[71,72,68,73]对加密流量进行广义识别甚至揭示其内容的技术已经较为成熟，Siby 等人[77]证明了基于加密 DNS 流量构建网站指纹，识别用户通过浏览器访问的具体网页进一步推断用户行为的可能性。网站指纹攻击相关工作集中在对 HTTP/HTTPS 进行流量分析，用户在访问某个网页产生的 DNS 流量远远小于产生的 HTTP 流量，构建 DNS 流量网站指纹难度相对 HTTP 流量具有更大的挑战性，但由于 DNS 流量相比于 HTTPS 流量更小的特点使得监管者在固定的监测能力下能够监管更多的网络用户，一方面通过基于加密 DNS 流量的网站指纹识别可以提高网络检测的效率；另一方面恶意网站通过加密 DNS 协议封装自己的请求来逃避检测，需要一种新的检测方式来应对这种恶意行为，基于加密 DNS 流量的网页指纹识别是一个有效的检测手段。

4.2 加密 DNS 流量采集平台

本文通过部署虚拟机来访问网页并捕获响应的加密 DNS 流量。加密 DNS 流量采集平台框架如图 4.1 所示，该台虚拟机部署在中国山东地区，并且搭建 Window 10 系统。由于需要收集大量的网页访问流量，所以为了让采集流程更加智能，平台使用了 Python

中的 Selenium 来进行自动化访问，并且以无头模式（即浏览器不显示网页 GUI 情况下运行）驱动浏览器访问网页。另外使用一个名为 YogaDNS[86]的 DNS 代理软件，它用于转发浏览器的 DNS 请求，YogaDNS 通过监听回环地址，并将加密后的 DNS 流量转发给递归 DNS 解析器。在 Selenium 调用浏览器访问网页的同时，Python 会调用 Scapy 库监听虚拟机产生的网络流量，并且根据 DNS 服务器的 IP 地址对虚拟机产生的网页流量进行采集，每次网页访问结束后会生成一个单独的 Pacp 文件。

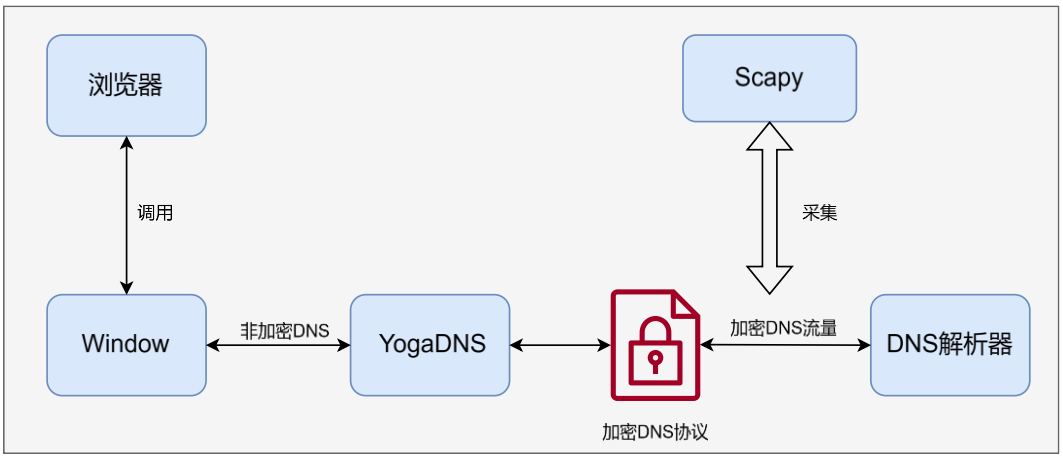

图 4.1 加密 DNS 流量采集平台框架

浏览器访问网页并捕获相关的加密 DNS 流量的具体步骤如下。首先浏览器开始访问某一网页，当页面开始加载的同时，Scapy 开始进行捕获流量，并且当网页访问结束后，Scapy 停止流量的捕获。每个页面有 20 秒的加载时间，并且在网页访问之间等待额外的 5 秒。这种采集方法能够确定与单个网页关联的加密 DNS 流量确切开始和结束的时间。另外，该采集环境不会在本地缓存 DNS 响应，以避免访问之间的干扰。

4.3 基于加密 DNS 流量的网站指纹识别方法

网站指纹识别使监听者能够确定用户通过加密或匿名通道访问哪些网站。它利用了这样一个事实，即网络数据包的大小、时间和顺序是网站内容的反映。由于每个网站的资源都是唯一的，因此即使流量被加密，网页内容也可以被识别。

网站指纹识别所利用的模式与 DNS 流量中的模式存在相关性：加载的资源以及它们的顺序决定了相应 DNS 查询的顺序。在访问某个网页时，浏览器会在网页加载期间短时间内发送多个 DNS 请求，比如 CDN、广告服务商和 JavaScript 资源等，本文将一次点击网页操作产生的所有的 DNS 流量称为 DNS Burst，通过 DNS Burst 构建加密 DNS 流量网站指纹。

与网站指纹识别一样，基于加密 DNS 流量的网站指纹识别视为一个监督学习问题：

首先收集一组访问网站产生的加密 DNS 流量的训练数据集，与加密 DNS 流量对应的网站(标签)是已知的，从这些加密 DNS 流量中提取特征(例如，网络数据包的长度)，并训练一个分类器。该分类器的目标是能够从收集到的网站访问产生的加密 DNS 流量中分析出用户正在访问的具体网站。

4.3.1 识别方法的整体框架

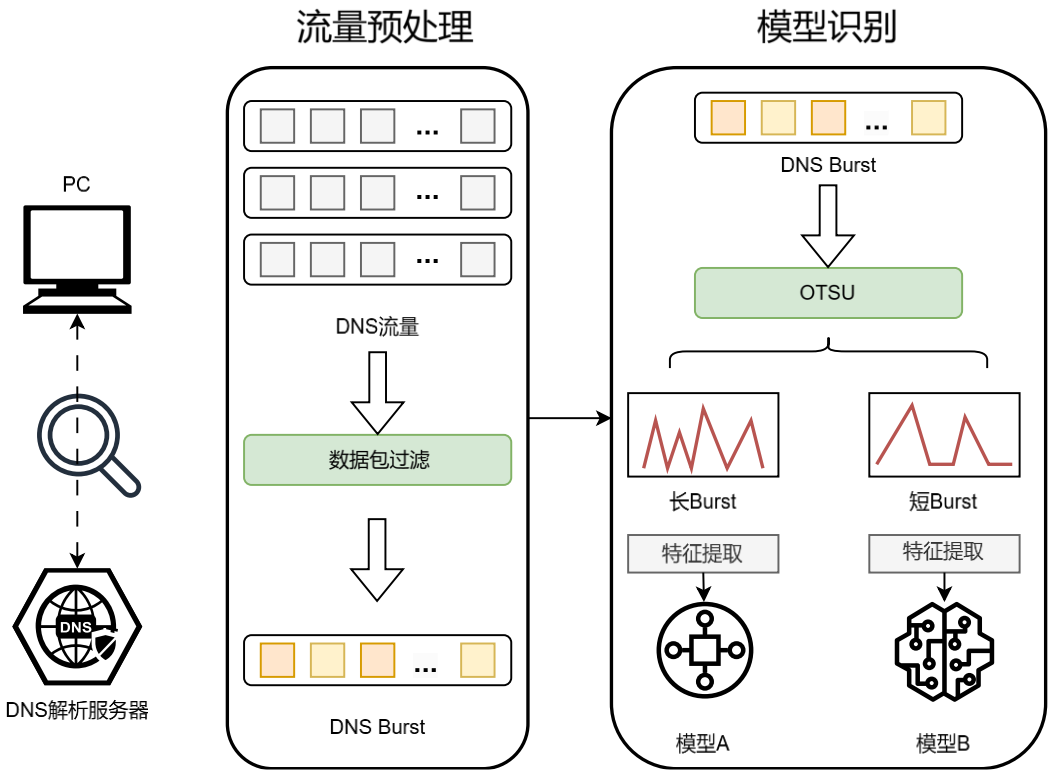

图 4.2 基于加密 DNS 流量的网站指纹识别框架

本文设计了一种基于加密 DNS 流量的网站指纹识别框架，如图 4.2 所示。首先采集监管范围内的 PC 与加密 DNS 解析服务器之间的加密 DNS 流量，然后进行流量预处理和模型识别两个模块。流量预处理模块的目标是从混杂的园区网络流量中提取出加密 DNS 流量，根据加密 DNS 流量协议的不同，提取的策略也会有所区别。DoQ 和 DoT 协议采用固定的端口提供 DNS 服务（DoT 使用 TCP/853 端口，DoQ 使用 UDP/784 和 UDP/8853），因此只需要根据端口号过滤出 DoQ 流量和 DoT 流量，而 DoH 流量则是基于 HTTPS 协议传输 DNS 流量，因此 DoH 所使用的 443 端口不单单含有 DoH 流量，这就需要训练一个轻量级的机器学习模型从混杂的 HTTPS 流量中提取 DoH 流量，这部分工作已有团队研究且准确率较高[87,88]，因此从混杂的 HTTPS 流量中进行 DoH 流量识别不是本文的研究重点，本文假设监听者可以从混杂的网络流量中筛选出加密 DNS 流

量。

通过对采集完成的DoH流量进行分析，大部分网页DoH Burst的包数量集中在1000以下，而包数量越多通常意味着该样本蕴含的信息较多且存在包数量相似的网页较少，识别的难度则会降低，而包数量较少且存在较多包数量相似的网页则会增加识别的难度。因此本文采用了分治的思想，根据网页产生的 DNS Burst 包数量的大小，对所有加密DNS流量样本进行划分，经过划分后可以在网络流量实时处理时提供冗余，有利于缓解在园区网络环境下单个模型识别处理能力有限导致过载的问题，并且划分后对于识别长Burst网页流量，可以采用更少的特征集合，提高识别效率和节省算力。划分的方法是大津法（OTSU）。大津法是一种用于图像处理和计算机视觉中的图像分割技术，它的基本思想就是找到一个阈值，将所有样本分为两个部分，使得分割后的样本中两个部分的类内方差最小化，而两个部分之间的类间方差最大化，具体方法如下。

首先计算流量数据集中所有样本产生的DNS Burst的包数量的直方图$p(i)$，其中i表示不同的包数量，DNS Burst中平均包数量m通过公式（4.1）得出。

$$m = \sum_{i=1}^{L} i \cdot p(i) \tag{4.1}$$

将每一个可能出现的阈值定义为t，根据阈值t可以将流量数据集划分为两个类别，分别为

$$class_1 = \{Flow_i \mid L_i \le t\}, class_2 = \{\{Flow_i \mid L_i > t\}\} \tag{4.2}$$

对于每个可能出现的包数量阈值t，计算两个类别的权重$w_1(t)$和$w_2(t)$，通过公式

$$w_1(t) = \sum_{i=1}^{t} p(i), w_2(t) = \sum_{i=t+1}^{L} p(i) \tag{4.3}$$

对于每个可能出现的包数量阈值t，计算两个类别的平均包数量$m_1(t)$和$m_2(t)$，通过公式

$$m_1(t) = \frac{1}{w_1(t)} \sum_{i=1}^{t} i \cdot p(i), m_2 = \frac{1}{w_2(t)} \sum_{t+1}^{L} i \cdot p(i) \tag{4.4}$$

根据上述公式可以计算阈值t划分出的两个类别的类间方差$between(t)$

$$between(t) = w_1(t) \cdot w_2(t) \cdot [m_1(t) - m_2(t)]^2 \tag{4.5}$$

遍历所有可能的阈值，找到使得类间方差$between$最大的阈值t^*，即

$$t^* = \arg\max_{0<t<L} between(t) \tag{4.6}$$

根据包数量阈值t^*可以将数据集中的网页划分为两类，一类产生的 DNS Burst 包数量小于阈值t^*，一类 DNS Burst 包数量大于阈值t^*，对于这两类流量依据它们的特点分别设计特征集合，训练两个不同的机器学习模型对用户访问的具体网站进行识别。

4.3.2 特征提取

本文针对加密 DNS 流量设计了一种基于直方图特征的加密 DNS 流量包长特征提取方法。DNS Burst 记为网络流B，其中B包含N个数据包，提取B的包长序列$len(B)$，每个数据包在包长序列中的值可以记为d_i，其中$i = 1,2,\dots,N$。

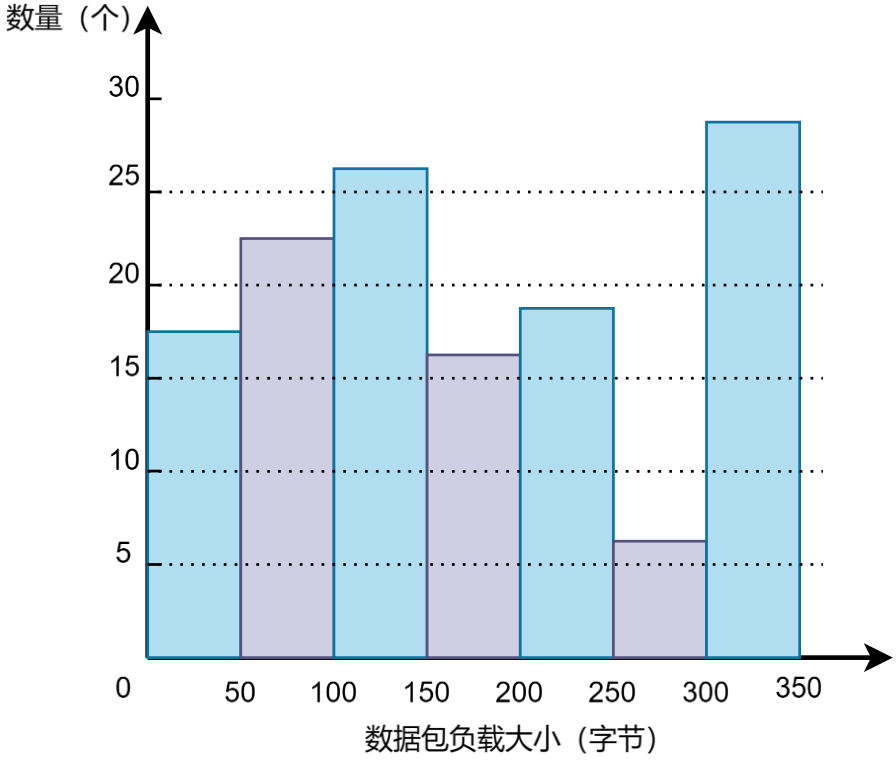

图 4.3 直方图特征提取示例

首先根据每个数据包特征序列的值转换为直方图H。直方图H表示每个数据点d_i的分布情况，直方图每一条柱子代表在一个特定的特征区间内数据包的数量（比如在包长 0-100 这个区间数据包的数量为 20），设直方图有k个区间，每个区间范围为$[a_1, b_1), [a_2 b_2), \dots, [a_k, b_k)$，则在直方图$H$可以表示为$H = \{h_1, h_2, \dots, h_k\}$，其中$h_i$表示落在

第i个区间$[a_i, b_i)$内数据包的数量，$H = \{h_1, h_2, \dots, h_k\}$被提取作为加密 DNS 流量指纹的特征向量，记为$F_1 = H = \{h_1, h_2, \dots, h_k\}$。

然后基于此直方图H继续深入挖掘其数据包的特征数据的分布情况，定义一组直方图H中纵轴数量的区间，区间的数量定义为m，记为$R = \{[r_0, r_1), [r_1, r_2), \dots, [r_{m-1}, r_m)\}$。对于每个包长区间$[a_i, b_i)$，将其对应到纵轴数量区间$R$上，并统计落在该区间内的横轴包长区间$[a_i, b_i)$的数量，即如果某个包长区间$[a_i, b_i)$内数据包的数量$h_i$落在了某个特定的纵轴数量区间$[r_{j-1}, r_j)$内，则纵轴该数量区间$[r_{j-1}, r_j)$的计数加一。经过上述统计，可以得到一个长度为$m$的特征序列，记为$F_2$。图 4.3 展示了一个 DNS Burst 进行上述直方图特征提取的例子。

DoH 和 DoQ 协议都制定了相应的填充策略[16,18]，尽管大部分公共 DNS 解析服务器并没有进行相应的部署[77]，但 DNS 的填充策略导致包长特征贡献度降低是必须要考虑的问题，所以本文除了针对加密 DNS 流量的特点提取了包长特征外，还需要对针对时间序列特征进行一步的特征挖掘。网页在加载过程中往往会有一定的时间顺序，比如先加载广告资源或者图片资源[76]。所以可以从此入手根据加密 DNS 流量在不同时间段产生的数据包的数量来推测用户在访问的具体网页，特征提取过程如下。将流持续时间划分为时间区间T，设有n个区间，$T = \{[t_0, t_1), [t_1, t_2), \dots, [t_{n-1}, t_n)\}$。统计每个区间的数据包的数量，即可以得到包数量序列$P = \{p_1, p_2, \dots p_n\}$，提取该序列做为加密 DNS 流量网站指纹的一部分，记为F_3。图 4.4 展示了在一个 DNS Burst 上进行F_3特征提取的过程。

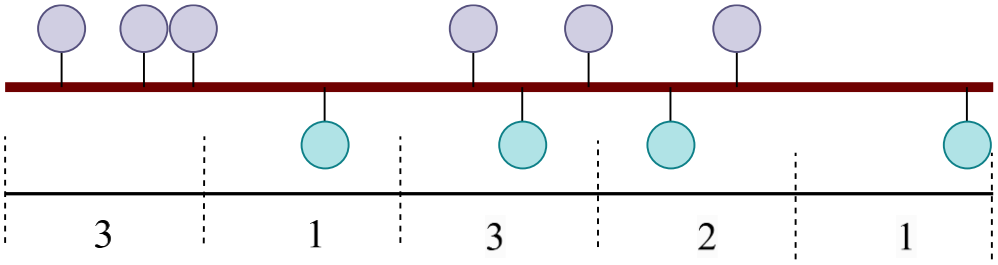

图 4.4 时间序列特征提取示例

上述特征（F_1、F_2、F_3）均提取三次，分别在全部的数据包序列、上行数据包序列和下行数据包序列。另外为了从数据包中提取更多的有效信息，需要添加更多的统计特征来提高网站指纹识别的准确率，本文从经典网络流量分析和基于 HTTPS 流量的网站指纹分析工作中得到启发，提取了一系列针对加密 DNS 流量的统计特征，表 4.1 展示了本文所使用的统计特征。

选择这些统计特征的理由如下：上下行流量的包数量和包负载大小特征一定程度反

映了 DNS 查询和 DNS 响应的数量和长度；上下行流量的包到达时间间隔特征一定程度反映了两个连续的 DNS 查询和两个连续的 DNS 响应之间的时间间隔；不同的网页由于资源数量的差异导致加载速度上存在差异，传输持续时间一定程序上反映了网页加载速度；统计特征构成的特征向量记为F_4。在后续实验中对比不同的监督学习分类算法的性能表现，选择效果最好的分类器作为最终的机器学习模型。

表 4.1 统计特征及其描述

特征名	特征描述
UIAT_mean，min，max，std	上行数据包的包到达时间间隔的均值、最大值、最小值、标准差
DIAT_mean，min，max，std	下行数据包的包到达时间间隔的均值、最大值、最小值、标准差
AllIAT_mean，min，max，std	所有数据包的包到达时间间隔的均值、最大值、最小值、标准差
ULen_mean, min, max, std	上行数据包的包载荷大小的均值、最大值、最小值、标准差
DLen_mean, min, max, std	下行数据包的包载荷大小的均值、最大值、最小值、标准差
AllLen_mean，min，max，std	上行数据包的包载荷大小的均值、最大值、最小值、标准差
UPacketCount	上行数据包的数量
DPacketCount	上行数据包的数量
AllPacketCount	数据包的总数量
DuraTime	传输持续时间

4.4 基于加密 DNS 流量的网站指纹识别实验

4.4.1 流量采集方法与数据集

本文于 2023 年 7 月份到 8 月份收集 Alexa 域名访问排行列表[89]中前 700 个网页产生的加密 DNS 流量，尽管 700 个网页与全世界整个网页规模相比较小，但是类似的网站指纹评估工作中网页数量都集中在 500 个[94,95]。

基于前文介绍的加密 DNS 流量采集平台，每个网页访问 50 次。考虑到由于公共 DNS 解析器的不同可能对实验的结果产生影响，本文从多个维度进行了数据采集工作，并且设计了多个大小不同的数据集方便后续的实验分析。表 4.2 展示了数据集的详细信息。

表 4.2 加密 DNS 流量数据集

数据集名称	DNS 解析器	加密协议	浏览器	网页数量	样本数量
DoHAli	ALIDNS	DNS over HTTPS	Chrome	200	50
DoHCloud	CloudFlare	DNS over HTTPS	Firefox	700	50
DoQAd	AdGuard	DNS over QUIC	Edge	378	50

另外需要说明的是由于网络因素、地理因素和反机器人机制等影响，部分网页可能存在响应错误等情况，对于这类网页本文的处理办法为直接将其从数据集中删除。还存

在同一网页不同域名的情况，比如 amazon.co.jp 和 amazon.co.uk 本质是同一个网页内容，但是域名为不同国家的不同版本，对于这类网页本文将其视为同一个网页。

此外，本文将重点分析 DoQ 流量在网站指纹识别上的表现，首先，基于 DoQ 流量的网站指纹识别是本文在该领域做出了创新性的探索，虽然以往已有关于 DoH 和 DoT 的相关研究，但本文首次尝试将 DoQ 流量引入这一领域，为相关研究提供了全新的视角和思考；其次，尽管目前关于 DoQ 的研究和应用尚处于起步阶段，但其前瞻性和发展潜力已经引起了业界的广泛关注。正如 Kosek 等人[90]所指出 DoQ 协议的出现解决了 DNS 效率和安全的问题，DoQ 的查询效率完全优于 DoT 和 DoH，甚至在某些网页上的表现已经超过了 DoUDP，这无疑使得 DoQ 成为当前加密 DNS 领域的最佳选择。

4.4.2 OTSU 阈值划分实验

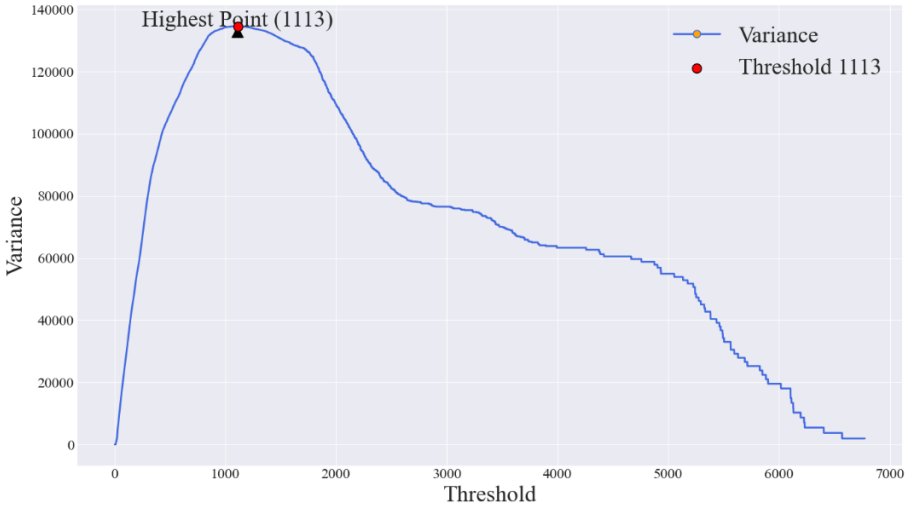

图 4.5 DoH 流量不同阈值的类间方差值

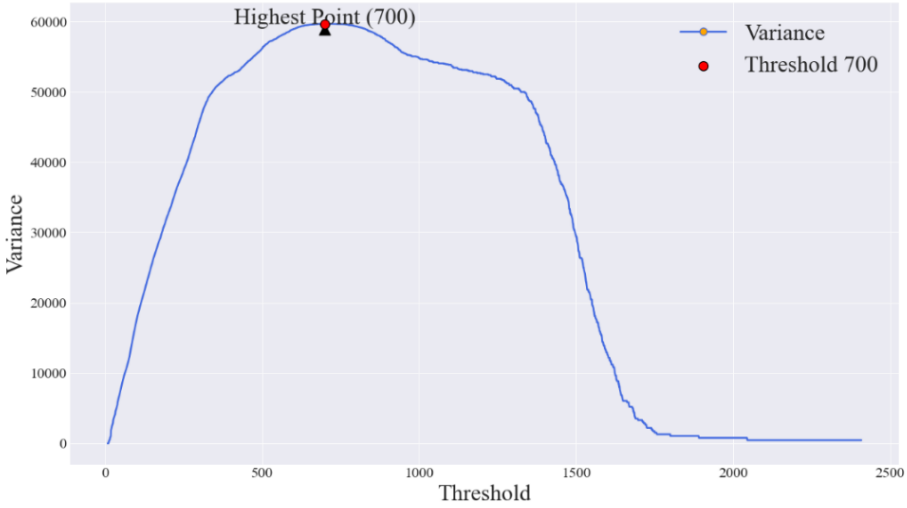

图 4.6 DoQ 不同阈值的类间方差值

该实验分别使用 DoHAli 和 DoQAd 两个数据集来获得 DoH 流量和 DoQ 流量的阈值，通过 OSTU 算法针对每个数据集，分别找到了其最佳的阈值。图 4.5 和图 4.6 分别显示了网页 DoH 流量和网页 DoQ 流量每一个可能的阈值划分后两个类别的类间方差。可以看到 DoH 在阈值选定为 1113 时，类间方差达到最大，对于 DoQ 流量则是阈值为 700 时，类间方差达到最大。后续实验中将依据该实验结果所获得的阈值对加密 DNS 流量进行划分。在 DoHAli 数据集中 200 个网页中有 164 个网页产生的 Burst 长度小于阈值，36 个网页产生的 Burst 长度大于阈值。DoQAd 数据集 378 个网页中 330 个网页产生的 Burst 长度小于阈值，48 个网页产生的 Burst 长度大于阈值。

4.4.3 阈值参数的选取

表 4.3 阈值参数及取值

参数	参数含义	取值
h	包长区间划分	50(byte)
x	时间序列划分	0.8(s)
t_{x1}	x秒前区间划分	0.05(s)
t_{x2}	x秒后区间划分	0.5(s)

本实验的目的是选取包长直方图横轴的划分区间和时间序列特征F_3的划分区间。需要注意的是对于时间序列特征F_3，本文发现网页产生的大部分加密 DNS 流量通常发生在点击网页后极短的时间内。为了更加关注点击网页所产生的这一段流量，本文对整个时间序列区间进行了两部分的划分，分为x秒前和x秒后。x秒前每一个区间相比于x后划分的更加细致。经过实验确定了每一项参数的取值，参数及其具体取值如表 4.3 所示。

4.4.4 无头浏览器与普通浏览器对比

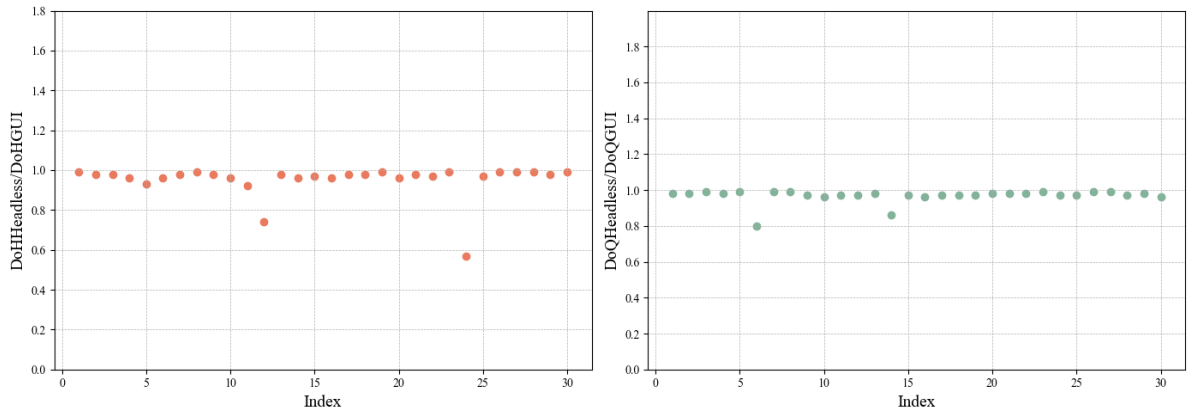

图 4.7 浏览器两种模式对比实验结果

本文数据集中的加密 DNS 流量均是通过浏览器的无头模式进行采集的，需要考虑的是这种采集模式和真实网络环境中采集的加密 DNS 流量之间的差异是多少。尽管无头浏览器和普通浏览器之间操作几乎一致，但有可能会存在一些时间等方面的差异。因此本文根据 Houser 等人[76]提出的对比方法来评估无头浏览器和普通浏览器之间的差异，即比较二者在查询数据包数量上的差异。在无头模式和有头模式下分别采集了 30 对 DoH 流量和 DoQ 流量，分别记为$DoQHeadless$、$DoHHeadless$、$DoQGUI$、$DoHGUI$，通过计算$DoQHeadless/DoQGUI$和$DoHHeadless/DoHGUI$来估计二者的差异，图 4.7 展示了比值的分布情况。

从图中可以看出几乎所有的网页的$DoQHeadless/DoQGUI$和$DoHHeadless/DoHGUI$值都处在 1.0 左右，虽然有部分网页的比率下降到了 0.6 左右，但是这和访问时间或广告资源等诸多因素有关，总体来看使用无头浏览器采集加密 DoH 和 DoQ 流量数据是合理的，这和 Houser 等人在 DoT 流量上得出来的结论一致。

4.4.5 OTSU 划分后模型效果

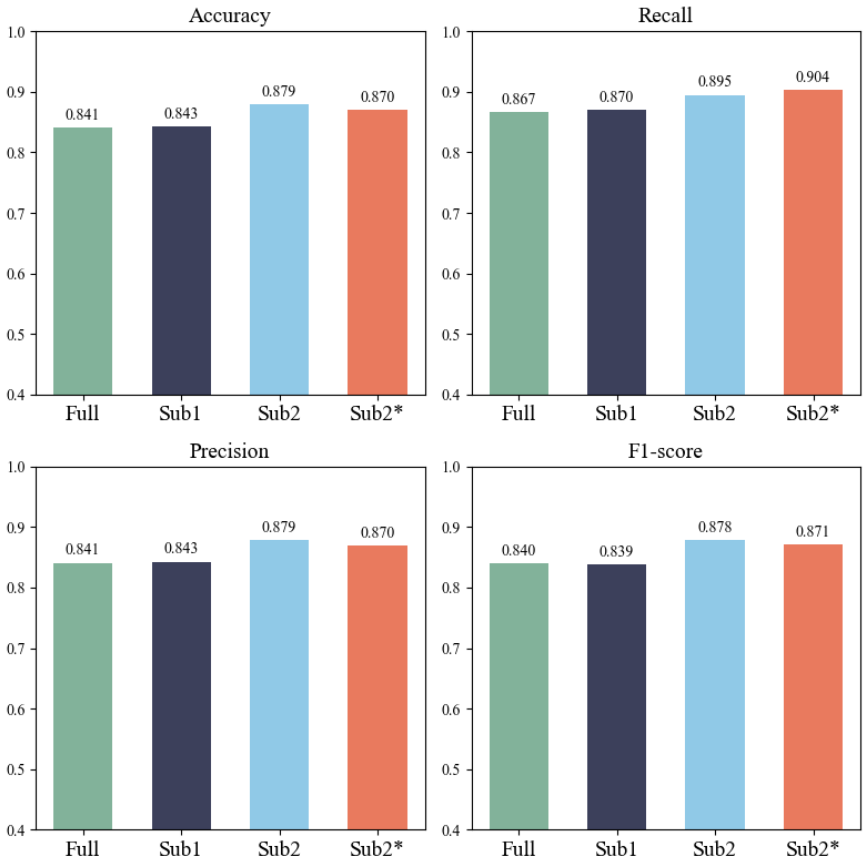

图 4.8 OTSU 划分前后模型性能对比

该实验目的是比较不经过 OTSU 和经过 OTSU 划分后模型识别准确率的差异，以及验证大于阈值的长 Burst 流量在较少的特征集合下是否能够保持较高的准确率。通过使用 DoQAd 数据集进行两次实验。第一次实验将完整的 DoQAd 数据输入到随机森林模型中进行训练，训练集和测试集的划分比例为 7：3；第二次实验使用 OTSU 算法对 DoQAd 数据集进行划分得到，将划分后的数据子集分别使用随机森林模型进行训练，训练集和测试集的划分比例同第一次实验相同。模型测试结果如图 4.8 所示，Full 代表使用了完整的数据集，Sub1 代表经过 OTSU 划分后低于阈值的数据子集，Sub2 代表高于阈值的数据子集，而 Sub2*则是代表高于阈值且只使用统计特征和部分直方图特征的数据子集。

实验结果显示，使用完整数据集的准确率在 84%左右，而经过 OTSU 划分后的两个数据子集的准确率均高于使用完整数据集的准确率，低于阈值的数据子集准确率为 84.3%，而高于阈值的数据子集准确率则是高达 87.9%，这也验证了产生长 Burst 的加密 DNS 网页流量识别难度更低一些。另外在 Sub2*删除了 40%左右的特征，只保留了直方图特征和时间序列特征，识别准确率仍然可以达到 87%以上，特征维度的减少有利于增加模型在高速网络环境下的处理能力。

4.4.6 模型选择实验

该实验的目标是挑选最合适的分类器算法，向训练模型提供带标签的数据集，该实验使用的数据集为 DoHCloud 和 DoQAd，分别评估本文提出的方法在 DoH 流量和 DoQ 流量上的表现。

在本实验中，利用 Python 中的 Scikit-learn 库训练了四种常见的机器学习模型：支持向量机（SVM）、随机森林（RF）、Adaboost 和 K 近邻（KNN），这些模型是在监督学习任务中广泛应用的经典算法。该实验使用 10 折交叉验证，在交叉验证中，每一类的样本被分成十个不相交的分区。分类器在 9 个分区的每一组上进行训练，并在剩下的分区中进行测试。在训练完成后，评估了这四种模型在测试集上的性能，比较四者在准确率、查准率、召回率以及 F1-Score 的差异。

如图 4.9 所示，与其它分类器相比，随机森林在 DoHCloud 和 DoQAd 上的表现均比其他分类器的效果好，在 DoHCloud 上 700 个网站指纹识别的任务中准确率达到 80%，在 DoQAd 上 378 个网站指纹识别的任务中准确率可以达到 85%左右。从该实验结果可以看出本文提出的基于加密 DNS 流量的网页指纹识别方法对于加密 DoH 流量和加密

DoQ 流量均有良好的识别效果。

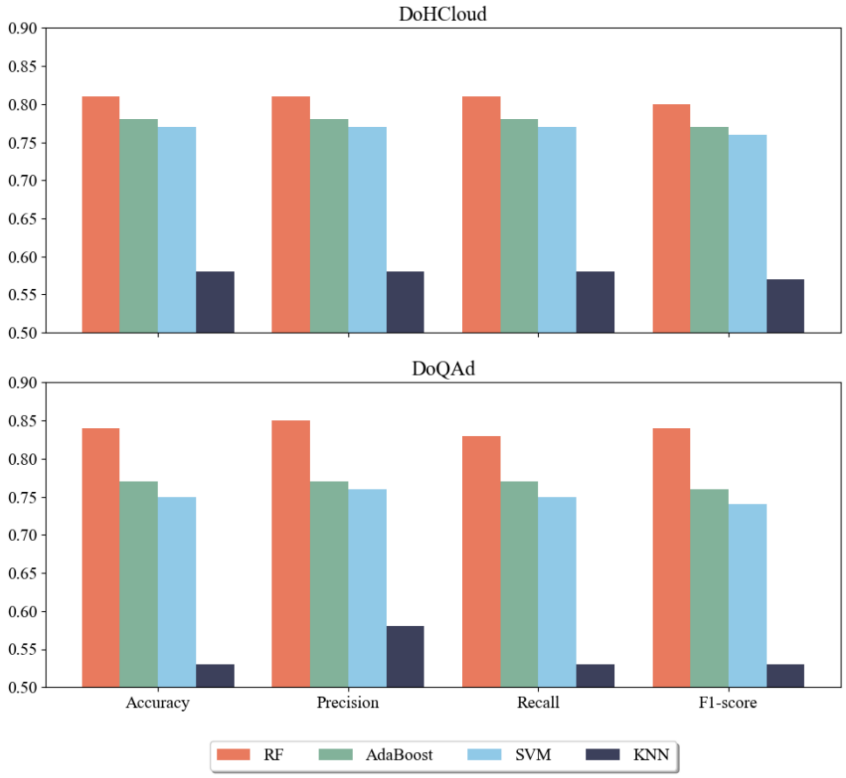

图 4.9 四种机器学习模型的性能表现

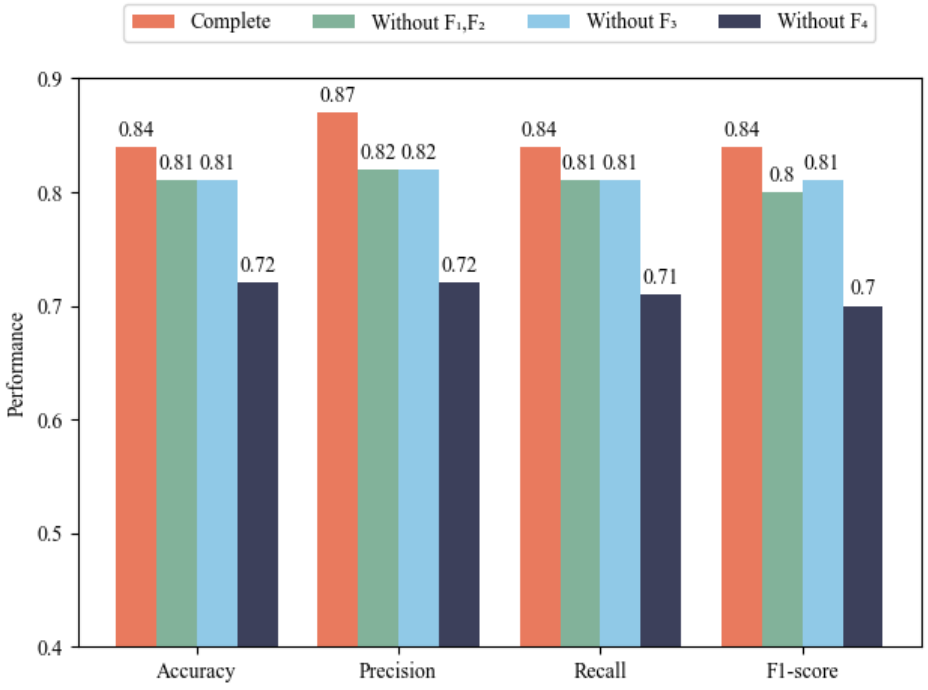

图 4.10 消融实验结果

4.4.7 消融实验

本实验旨在分析本文提出的四种特征序列（F_1、F_2、F_3、F_4）在模型预测中的作用。构建四种特征集，分别为完整特征集、不包含F_1、F_2的特征集（F_1、F_2均为包长直方图特征）、不包含F_3的特征集以及不包含F_4的特征集。采用随机森林模型，评估指标由准确率、召回率、查准率和 F1 值，图 4.10 展示了消融实验的结果。通过对各类特征消融后的性能变化进行分析，可以看出F_4对模型性能的影响最为显著，这是因为F_4包含 DNS 流量的包长、时间等各个维度的特征。F_1、F_2、F_3贡献度差距较小，它们使得模型的准确率提高了约 12%。

此外，根据随机森林模型提供的基于基尼指数的特征重要度排序，表 4.4 显示了贡献度排名前十的特征。DoQAd 数据集中的流量没有经过填充，可以从特征贡献度排名中发现包长特征是一个贡献度相当显著的特征，但是在应对经过填充后的数据集时，包长特征的贡献度会明显下降，在 4.4.8 节的实验中会有详细的描述。

表 4.4 贡献度排名前十的特征

排名	特征	贡献度
1	下行流量第三个包长区间（100-150byte）的包数量	0.043
2	全部流量第三个包长区间（100-150byte）的包数量	0.040
3	全部流量第四个包长区间（150-200byte）的包数量	0.040
4	下行流量第四个包长区间（150-200byte）的包数量	0.039
5	下行流量第五个包长区间（200-250byte）的包数量	0.038
6	全部流量第五个包长区间（200-250byte）的包数量	0.036
7	上行流量包到达时间间隔的标准差	0.032
8	上行流量包到达时间间隔的最小值	0.031
9	上行流量包到达时间间隔的最大值	0.030
10	全部流量包到达时间间隔的标准差	0.028

4.4.8 填充策略下识别效果

在采集 DoHAli 数据集时，阿里云 DNS 服务器部署了相关的填充策略[91]，通过提取五个网页产生的 DNS Burst 的包长序列进行分析，图 4.11 展示了这 6 个网页的包长序列分布情况，散点的大小代表了同一包长出现的频次，5 个不同的网页产生的包长序列分布基本一致，绝大部分的数据负载大小均为 60、94 和 153。

本实验通过设计两个特征集，来评估模型在应对 DNS 服务器使用填充策略下的性能表现。一个特征集合包含完整的特征集，另一个特征集合仅包含包长特征，图 4.12 展

示不同特征集合下的模型表现。

通过实验结果显示，经过填充后的 DoH 流量在仅使用包长特征时，准确率降低到了 50%以下，这比较符合本实验的预期。但通过添加时间序列特征后，准确率和 Recall 依然可以保持在 81%以上，这意味着仅依靠填充策略并不能有效地阻止本文提出的基于加密 DNS 流量的网站指纹识别方法。

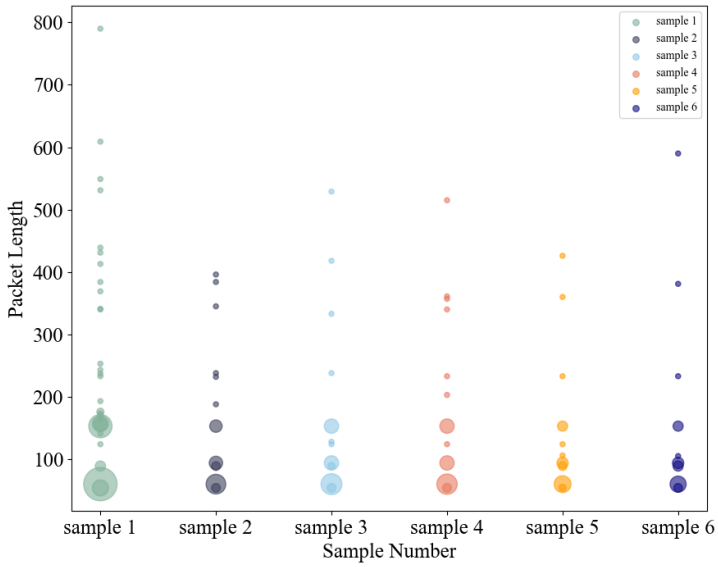

图 4.11 6 个网页的包长序列分布

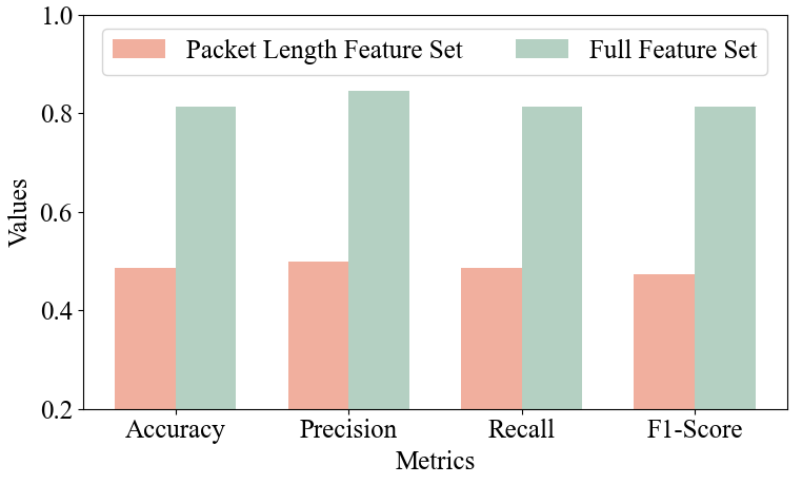

图 4.12 经过填充策略后的网站指纹识别效果

4.4.9 对比实验

该实验的目的是将本文提出的基于加密 DNS 流量的网页指纹识别方法与先前工作提出的基于加密 DNS 流量的网站指纹识别方法进行对比，验证本文提出的方法的有效

性。Siby 等人[77]提出了一种基于 N-Gram 的加密 DoH 流量网站指纹识别方法，该方法利用加密 DoH 流量中N个 N-Gram 包长序列来表征网站指纹。比如一个网页的加密 DNS 流量包长序列为（-64，88，33，-33），那么 uni-grams 和 bi-grams 分别为（-64），（88），（33），（-33）和（-64，88）（88，33）（33，-33）；Houser 等人[76]则是针对 DoT 流量，提出了一种根据数据包大小和到达时间序列进行建模的网站指纹识别方法，该方法的特征集中包括上下行数据包数量、上下行累计字节数等一系列统计特征。因为 DoQ 协议是一个比较新的协议，2022 年 5 月才成为标准协议，所以据目前了解本工作是第一个进行基于 DoQ 流量的网站指纹识别的研究。

图 4.13 展示了三种方法在 DoQAd 上的表现，可以看出本文提出的方法比 Houser 等人的方法在准确率上高了约 2.9%，比 Siby 等人的方法约高出了 58%。这里需要注意的是 Siby 等人提出的方法仅关注包长特征，并没有使用时间序列特征，这有可能是该方法在 DoQAd 数据集上表现较差的原因之一。

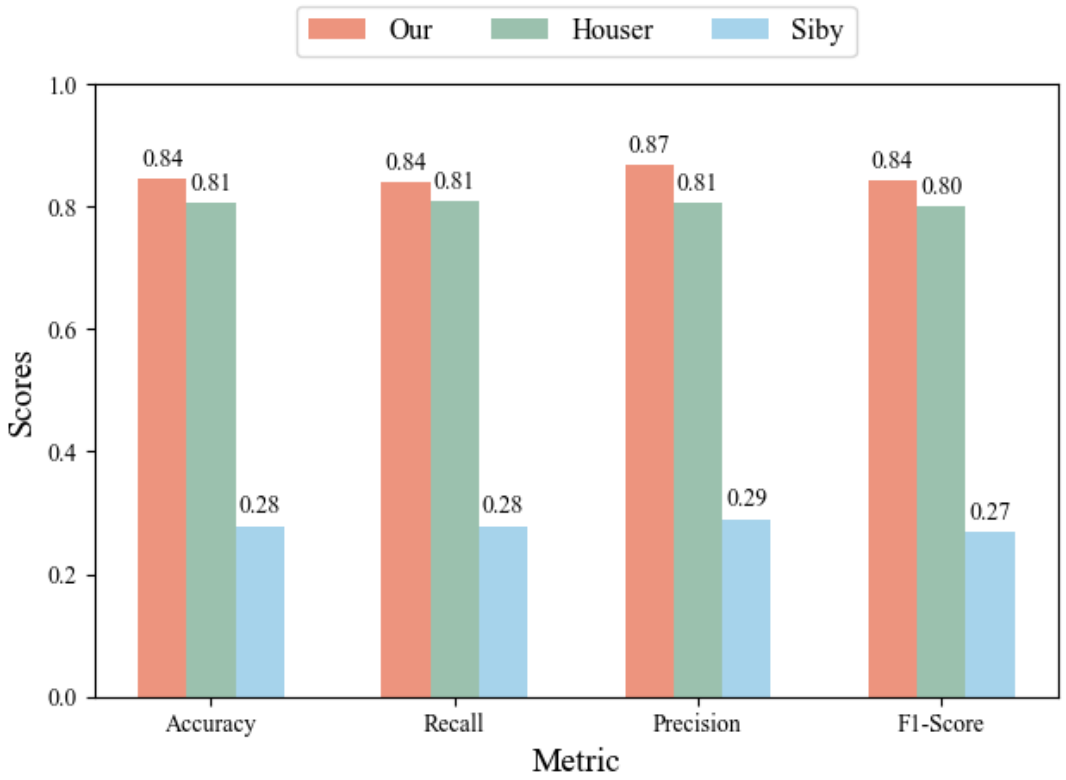

图 4.13 对比试验结果

4.4.10 在网络恶意应用检测上的表现

本实验的目的是为了评估本文提出的基于加密 DNS 流量的网站指纹识别方法在检测网络恶意应用上的表现。该实验的正样本使用了 Mitsuhashi 等人提供的 DoH-DGA-

Malware-Traffic-HKD 数据集[92]，该数据集是通过运行四种基于域名生成算法（DGA）的网络恶意应用（PadCrypt、Sisron、Tinba、Zloader）所捕获的，DGA 是一种恶意软件（如僵尸网络、勒索软件等）常用的技术，它在运行时随机生成大量域名，用于控制服务器进行通信。实验数据通过 VirusShare[93]进行收集的真实网络环境下产生的数据集，各个恶意应用的流数量如表 4.5 所示；该实验的负样本则是使用本文采集的访问正常网页所产生的 DoH 流量。由于两个数据集的采集时间和采集地点不同，所以为了实验的有效性，不使用特征集中的时间特征，仅使用包长特征。

表 4.5 数据集种类及数量

数据集	类型	流数量
DoH-DGA-Malware	Sisron	744
	Tinba	1808
	Zloader	820
	PadCrypt	840
CIRA-CIC-DoHBrw	良性 DoH	18290
	恶意 DoH	249467

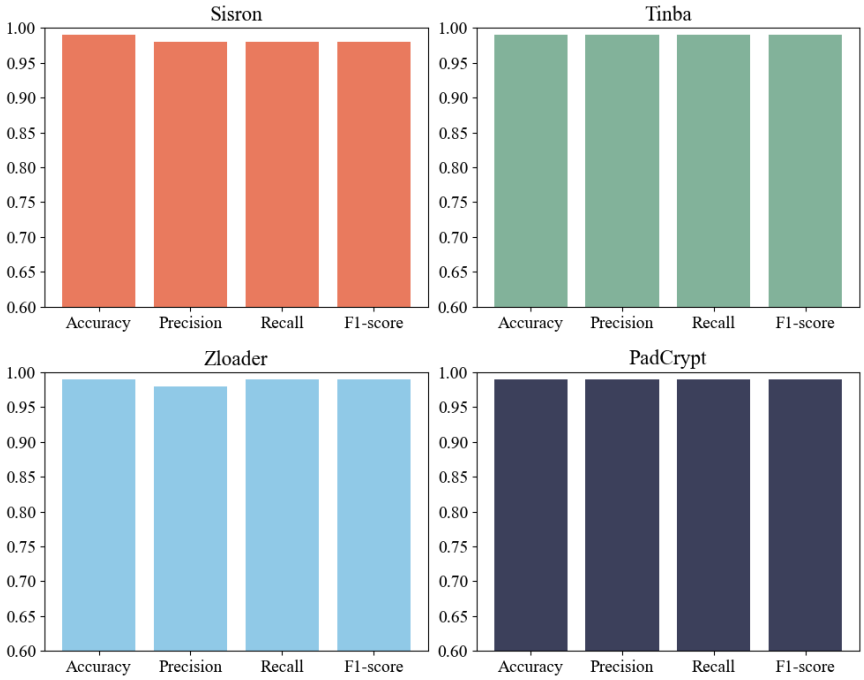

图 4.14 基于 DGA 的网络恶意应用检测的性能表现

基于此数据集，重新训练一个二分类任务的随机森林分类器，目的是获得一个能够从区分正常的 DoH 流量和网络恶意应用产生的恶意 DoH 流量的分类器。使用 10 折交叉验证，计算十次结果中 Accuracy、Recall、Precision 和 F1-score 四个指标的平均值来评估模型的性能。实验结果如图 4.14 所示，本文所提出的识别方法对四种基于 DGA 的网络恶意应用识别准确率都达到了 99%以上，Recall、Precision 以及 F1-score 也达到了

99%，这说明本文所提出的识别方法有能力检测基于 DGA 的网络恶意应用，有较好的准确性。

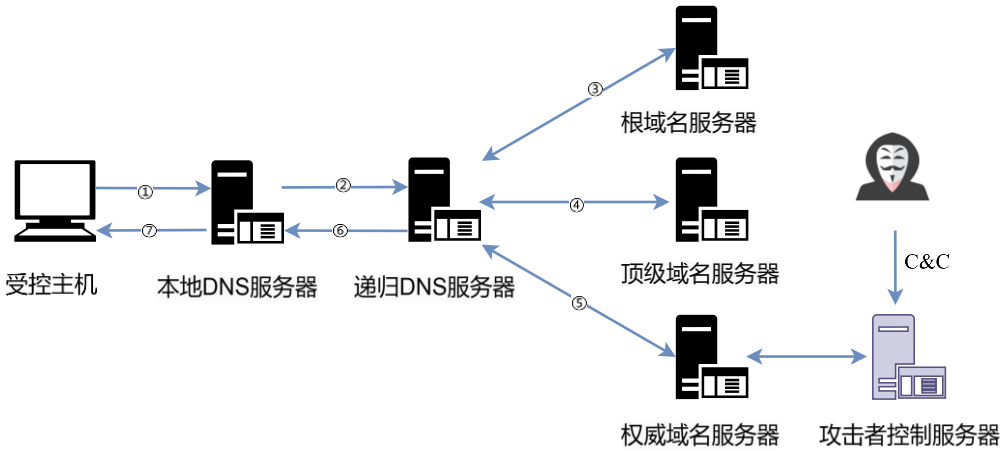

图 4.15 DoH 隧道攻击

另外本文还评估了该方法在检测 DoH 隧道攻击上的表现，DoH 隧道是指建立在受控主机和伪装成权威域名服务器的主服务器之间，可作为恶意活动的秘密数据通信通道。具体攻击流程如图 4.15 所示，攻击者通过把 C&C 数据或目标数据通过编码、加密、混淆等方法嵌入到一个 DNS 请求的子域名中达到控制目标主机和窃取信息的目的[87]。该实验使用加拿大网络安全研究所提出的 CIRA-CIC-DoHBrw-2020 数据集[88]，该数据集分为非 DoH 流量、良性 DoH 流量和恶意 DoH 流量，该实验只评估模型对良性 DoH 流量和恶意 DoH 流量的识别能力，具体地流数量如表 4.5 所示。模型使用随机森林模型，按照 7：3 的比例划分训练集和测试集，模型性能表现如图 4.16 所示。

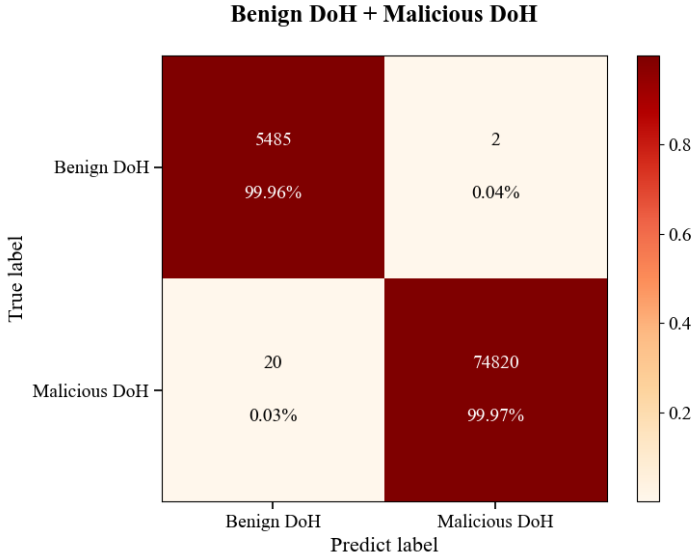

图 4.16 基于 DoH 隧道的恶意 DoH 流量检测的性能表现

从图中可以看出该方法对于基于 DoH 隧道的恶意 DoH 的识别准确率可以达到 99%

以上，仅有 20 条恶意 DoH 流量被错误识别为良性 DoH 流量。虽然验证的数据集有限，但这说明了本文提出的识别方法对于恶意 DoH 流量检测同样具有一定的适用性。

4.5本章小结

本章主要阐述了一种基于加密 DNS 流量的网站指纹识别方法，该方法提出了一种基于 OTSU 的数据划分策略，并且针对加密 DNS 流量的特点提出了基于直方图划分的数据包包长特征和时间序列特征。通过设计的大量实验表明，该方法在加密 DNS 流量网站指纹识别上具有较好的表现，在识别经过填充策略后的加密 DNS 流量任务上仍然具有较好的性能，另外还通过实验证明了该方法在网络恶意应用检测上的适用性。

第五章 总结与展望

5.1 研究总结

本文针对基于域名解析行为分析的网络恶意行为检测开展了研究，对于非加密 DNS 流量，以挖矿行为检测问题作为研究依托，通过域名解析行为分析的角度实现了挖矿行为的早期识别，并在真实的网络环境下进行了大量实验，验证了该方法的及时性和有效性；以挖矿行为检测作为进一步的研究对象，设计了一种阶层式网络恶意应用识别系统，并根据挖矿流量的特性对各个模块进行了针对性设计，通过一系列实验验证了该框架可以实现对挖矿行为的有效检测；对于加密 DNS 流量，通过设计一种自动化加密 DNS 流量采集平台获得了一个完整的加密 DNS 流量数据集，依托此数据集设计了一种基于加密 DNS 流量的网站指纹识别方法，并进行了一系列相关实验验证了该方法得有效性，最后探究了该方法在网络恶意应用检测任务上的效果。

本文的研究内容可以总结为以下三个部分：

（1）提出了一种基于词素的挖矿行为检测框架 Mining Vanguard，通过检测主机用户发送的 DNS 请求数据包来推测是否存在挖矿行为。该系统分为快速匹配和动态推理两个模块：在快速匹配模块，设计了一种基于词素的哈希表，优化了正则表达式的匹配过程，能够实现对知识库中矿池域名的快速发现；在动态推理模块，通过构建一个包括传统 DNS 解析特征和词素特征在内的综合特征集，将网络特性与语义特性结合在一起，并训练一个能够区分正常域名和矿池域名的机器学习模型，实现了对未知矿池域名的识别。通过在真实环境下采集的数据集进行了大量的实验，发现 Mining Vanguard 方法的召回率可以达到 90%以上，通过对比试验发现该方法比传统基于通信流量的挖矿行为检测方法在识别所需时间上平均减少了 40 多秒。

（2）设计了一种阶层式网络恶意应用识别系统，该系统主要由四大模块组成，其中，数据采集预处理模块负责流量的数据采集和特征提取；域名匹配推理模块通过 DNS 流量实现对恶意应用的早期发现，匹配模块和推理模块相结合，兼顾了系统的实时性和准确率；基于 LTC-Sketch 的初筛模块能够通过分析网络流的持久度和频繁度，筛选出特定频率的流量进行保留，过滤掉大部分正常流量，降低正常流量和恶意流量的非平衡比率，提高后续模型的识别效率，降低模型算力消耗。机器学习模块能够对疑似恶意流量进行识别，做出最后的决策。以挖矿行为检测实际任务为依托进行了实验验证，实验结

果显示该系统对挖矿行为检测的表现可以达到较为理想的效果。

（3）针对加密 DNS 流量，提出了一种基于加密 DNS 流量的网站指纹识别方法，通过分析加密 DNS 流量实现对恶意网站的识别。该方法根据加密 DNS 流量的特点，通过大津法（OTSU）进行数据集划分，提高在真实网络环境下的识别效率和准确率。然后设计了一种基于直方图的包长特征和基于时间序列的特征集，结合传统网络统计特征形成了一个针对加密 DNS 流量网站指纹识别的特征集合。本文设计了一个加密 DNS 流量采集平台，基于该平台采集制作了一个涵盖多种加密 DNS 协议和 DNS 解析服务器的数据集。在该数据集上进行了验证实验，实验结果显示该方法在 DNS-over-QUIC 和 DNS-over-HTTPS 网站指纹识别任务上均有较好的表现。最后，将该方法在网络恶意应用识别任务上，通过分析网络恶意应用产生的加密 DNS 流量来进行网络恶意应用检测，实验结果显示该方法对网络恶意应用检测也具备较好的效果。

5.2 研究展望

尽管本文中的研究在现阶段能够表现出较为可靠的性能，但仍有一些改进空间。由于现实世界中的网络环境还在不断更新发展，对应地，网络恶意行为也会层出不穷地产生新的模式。对此，主要有以下两个方面的改进：

（1）对于基于词素的矿池域名识别，虽然目前 Mining Vanguard 的性能表现较好，但随着恶意挖矿软件的发展，部分软件已经开始启用 DoH 协议对矿池 DNS 请求加密来逃避检测。对于这种情况，需要发掘一种针对矿池加密 DNS 请求的识别方法，可以借助深度学习方法，尽管深度学习模型的可解释性不如 Mining Vanguard，但是可以借助深度模型庞大的体量针对加密 DNS 流量提取更有效的特征，从而提高识别的准确率。

（2）对于基于加密流量的网站指纹识别，目前实验结果都是基于小规模的网页数据集，面对开放的网络世界，网站的数量比实验中使用的数据集中网站的数量多得多。

面对庞大的网页数量，精准识别的难度非常大。对此，面对开放的网络世界，可以发掘一种基于加密 DNS 流量的网页类型识别方法，实现对恶意网站（博彩、色情等）的有效检测。

参考文献

[1] We Are Social：数字 2024[R/OL]. (2024-03-06). https://accesspath.com/report/5915128/.

[2] 错综复杂:Ryuk 勒索软件攻击事件背景整理与总结[EB/OL]. (2019-01-16). https://www.freebuf.com/articles/blockchain-articles/194353.html.

[3] Pan Y, Lin W, Jiao L, et al. Model-Based Grey-Box Fuzzing of Network Protocols[J]. Security and Communication Networks, 2022, 2022(1): 6880677.

[4] 精通 NSA 十八般兵器的 NSAFtpMiner 矿工来了，已有 3 万台电脑中招[EB/OL]. (2018-09-09). https://www.freebuf.com/articles/es/183365.html.

[5] RunMiner 挖矿木马攻击，约 1.6 万台服务器沦陷[EB/OL]. (2020-12-07). https://ti.dbappsecurity.com.cn/info/1443.

[6] 关于“8220”黑客攻击团伙近期活跃情况的挖掘分析报告[R]. 国家互联网应急中心, 2022.

[7] 国家发展改革委等部门关于整治虚拟货币“挖矿”活动的通知 (发改运行〔2021〕1283 号)[EB/OL]. (2021-09-25). http://www.gov.cn/zhengce/zhengceku/2021-09/25/content_ 5639225.html.

[8] Newaz A I, Sikder A K, Rahman M A, et al. A survey on security and privacy issues in modern healthcare systems: Attacks and defenses[J]. ACM Transactions on Computing for Healthcare, 2021, 2(3): 1-44.

[9] Vale M, Dupuy A. Google Public DNS over HTTPS (DoH) supports RFC 8484 standard[J]. Google Security Blog, 2019.

[10] Hounsel A, Borgolte K, Schmitt P, et al. Analyzing the costs (and benefits) of DNS, DoT, and DoH for the modern web[C]//Proceedings of the Applied Networking Research Workshop. 2019: 20-22.

[11] D Chhabra R, Murley P, Kumar D, et al. Measuring DNS-over-HTTPS performance around the world[C]//Proceedings of the 21st ACM Internet Measurement Conference. 2021: 351-365.

[12] Csikor L, Singh H, Kang M S, et al. Privacy of DNS-over-HTTPS: Requiem for a dream?[C]//2021 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2021: 252-271.

[13] Hu Z, Zhu L, Heidemann J, et al. Specification for DNS over transport layer security (TLS)[R]. 2016.

[14] Dierks T. The transport layer security (TLS) protocol[J]. IETF RFC4346, 2006.

[15] Kartch R. Best practices for network border protection[J]. Carnegie Mellon University’s Software Engineering Institute Blog, 2017.

[16] Borgolte K, Chattopadhyay T, Feamster N, et al. How dns over https is reshaping privacy, performance, and policy in the internet ecosystem[C]//TPRC47: The 47th Research Conference on Communication,

Information and Internet Policy. 2019.

[17] Lyu M, Gharakheili H H, Sivaraman V. A survey on DNS encryption: Current development, malware misuse, and inference techniques[J]. ACM Computing Surveys, 2022, 55(8): 1-28.

[18] Huitema C, Shore M, Mankin A, et al. Specification of DNS over dedicated QUIC connections[J]. Internet Engineering Task Force, Internet-Draft draft-huitema-quic-dnsoquic-05, 2018.

[19] Iyengar J, Thomson M. QUIC: A UDP-based multiplexed and secure transport[M]//RFC 9000. Fremont, CA, USA: Internet Engineering Task Force (IETF), 2021.

[20] Carlucci G, De Cicco L, Mascolo S. HTTP over UDP: an experimental investigation of QUIC[C]//Proceedings of the 30th Annual ACM Symposium on Applied Computing. 2015: 609-614.

[21] Sherry J, Lan C, Popa R A, et al. Blindbox: Deep packet inspection over encrypted traffic[C]//Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. 2015: 213-226.

[22] Ning J, Poh G S, Loh J C, et al. PrivDPI: Privacy-preserving encrypted traffic inspection with reusable obfuscated rules[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 1657-1670.

[23] Salva-Garcia P, Ricart-Sanchez R, Chirivella-Perez E, et al. Xdp-based smartnic hardware performance acceleration for next-generation networks[J]. Journal of Network and Systems Management, 2022, 30(4): 75.

[24] NVIDIA BlueField DPU Platform Operating System v3.9.3.1 Documentation[EB/OL]. (2023-03-02). https://docs.nvidia.com/networking/display/BlueFieldDPUOSLatest.

[25] NVIDIA DOCA DPI Sample Guide[EB/OL]. (2021-04-21). http://docs.nvidia.com/ doca/dpi-samples/index.html.

[26] 张蕾, 崔勇, 刘静, 等. 机器学习在网络空间安全研究中的应用[J]. 计算机学报, 2018, 41: 1943-1975.

[27] Sun G, Chen T, Su Y, et al. Internet traffic classification based on incremental support vector machines[J]. Mobile Networks and Applications, 2018, 23: 789-796.

[28] Peng L, Zhang H, Chen Y, et al. Imbalanced traffic identification using an imbalanced data gravitation-based classification model[J]. Computer Communications, 2017, 102: 177-189.

[29] Erman J, Arlitt M, Mahanti A. Traffic classification using clustering algorithms[C]//Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data. 2006: 281-286.

[30] Wu H, Chen X, Cheng G, et al. Bcac: Batch classifier based on agglomerative clustering for traffic classification in a backbone network[C]//2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS). IEEE, 2021: 1-10.

[31] Mahdavifar S, Kadir A F A, Fatemi R, et al. Dynamic android malware category classification using semi-supervised deep learning[C]//2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE, 2020: 515-522.

[32] Aamir M, Zaidi S M A. Clustering based semi-supervised machine learning for DDoS attack classification[J]. Journal of King Saud University-Computer and Information Sciences, 2021, 33(4): 436-446.

[33] Lotfollahi M, Jafari Siavoshani M, Shirali Hossein Zade R, et al. Deep packet: A novel approach for encrypted traffic classification using deep learning[J]. Soft Computing, 2020, 24(3): 1999-2012.

[34] Aceto G, Ciuonzo D, Montieri A, et al. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges[J]. IEEE Transactions on Network and Service Management, 2019, 16(2): 445-458.

[35] Shen M, Liu Y, Zhu L, et al. Optimizing feature selection for efficient encrypted traffic classification: A systematic approach[J]. IEEE Network, 2020, 34(4): 20-27.

[36] 张小莉, 程光, 张慰慈. 基于改进深度卷积神经网络的网络流量分类方法[J]. 中国科学:信息科学, 2021, 51(01): 56-74.

[37] 郭益民, 张爱新. 基于卷积神经网络的 Android 流量分类方法[J]. 通信技术, 2020, 53(02): 432-437.

[38] Lim H K, Kim J B, Heo J S, et al. Packet-based network traffic classification using deep learning[C]//2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). IEEE, 2019: 046-051.

[39] Fang W, Yu Z, Chen Y, et al. Deep residual learning in spiking neural networks[J]. Advances in Neural Information Processing Systems, 2021, 34: 21056-21069.

[40] 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.

[41] 陆一飞, 李艳. 一种基于 CNN 和 SAE 的加密流量识别方法[M]. CN111611280A 版. 2020.

[42] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. 2008.

[43] CoinMarketCap[EB/OL]. (2023-03-02). https://coinmarketcap.com.

[44] Gervais A, Karame G O, Wüst K, et al. On the security and performance of proof of work blockchains[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 3-16.

[45] Varlioglu S, Gonen B, Ozer M, et al. Is cryptojacking dead after coinhive shutdown?[C]//2020 3rd International Conference on Information and Computer Technologies (ICICT). IEEE, 2020: 385-389.

[46] Stratum mining protocol[EB/OL]. (2023-03-02). https://stratumprotocol.org.

[47] Recabarren R, Carbunar B. Hardening stratum, the bitcoin pool mining protocol[J]. arXiv preprint arXiv:1703.06545, 2017.

[48] Charles M. Threat report: Don’t join blockchain revolution without ensuring security[EB/OL]. (2018-06-13). https://www.mcafee.com/blogs/other-blogs/mcafee-labs.

[49] 黄子依, 秦玉海. 基于多特征识别的恶意挖矿网页检测及其取证研究[J]. 信息网络安全, 2021, 21: 87-94.

[50] 高见, 孙懿, 王润正, 等. 基于机器学习的浏览器挖矿检测模型研究[J]. 计算机工程与应用, 2021, 57: 125-130.

[51] Rodriguez J D P, Posegga J. Rapid: Resource and api-based detection against in-browser miners[C]//Proceedings of the 34th Annual Computer Security Applications Conference. 2018: 313-326.

[52] Karn R R, Kudva P, Huang H, et al. Cryptomining detection in container clouds using system calls and explainable machine learning[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 32(3): 674-691.

[53] Darabian H, Homayounoot S, Dehghantanha A, et al. Detecting cryptomining malware: A deep learning approach for static and dynamic analysis[J]. Journal of Grid Computing, 2020, 18: 293-303.

[54] Gangwal A, Piazzetta S G, Lain G, et al. Detecting covert cryptomining using hpc[C]//Cryptology and Network Security: 19th International Conference, CANS 2020, Vienna, Austria, December 14–16, 2020, Proceedings 19. Springer International Publishing, 2020: 344-364.

[55] Iperf: The TCP/UDP bandwidth measurement tool[EB/OL]. (2001-06-08). http://dast. nlanr.net/Projects.

[56] Azmoodeh A, Dehghantanha A, Conti M, et al. Detecting crypto-ransomware in IoT networks based on energy consumption footprint[J]. Journal of Ambient Intelligence and Humanized Computing, 2018, 9:

1141-1152.

[57] Kelton C, Balasubramanian A, Raghavendra R, et al. Browser-based deep behavioral detection of web cryptomining with coinspy[C]//Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb). NDSS, 2020: 1-12.

[58] Caprolu M, Raponi S, Oligeri G, et al. Cryptomining makes noise: Detecting cryptojacking via machine learning[J]. Computer Communications, 2021, 171: 126-139.

[59] Pastor A, Mozo A, Vakaruk S, et al. Detection of encrypted cryptomining malware connections with machine and deep learning[J]. IEEE Access, 2020, 8: 158036-158055.

[60] Mozo A, González-Prieto Á, Pastor A, et al. Synthetic flow-based cryptomining attack generation through generative adversarial networks[J]. Scientific Reports, 2022, 12(1): 2091.

[61] Konoth R K, Vineti E, Moonsamy V, et al. Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 1714-1730.

[62] Rüth J, Zimmermann T, Wolsing K, et al. Digging into browser-based crypto mining[C]//Proceedings of the Internet Measurement Conference 2018. 2018: 70-76.

[63] Bian W, Meng W, Wang Y. Detecting webassembly-based cryptocurrency mining[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 2685-2687.

[64]胡晓艳, 舒卓卓, 童钟奇, 等. 一种结合活跃节点库和机器学习的高效以太坊流量识别方法[M]. CN114024748A 版. 2022.

[65] 胡晓艳, 童钟奇, 程光, 等. 一种轻量化以太坊加密流量识别方法[M]. CN111865823B 版. 2022.

[66] 胡晓艳, 舒卓卓, 程光, 等. 一种比特币挖矿僵尸网络流量的快速识别方法[M]. CN113518073B 版. 2022.

[67] Wagner D, Schneier B. Analysis of the SSL 3.0 protocol, he Second USENIX Workshop on Electronic Commerce Proceedings[J]. 1996.

[68] Hintz A. Fingerprinting websites using traffic analysis[C]//International Workshop on Privacy Enhancing Technologies. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002: 171-178.

[69] Liberatore M, Levine B N. Inferring the source of encrypted HTTP connections[C]//Proceedings of the 13th ACM Conference on Computer and Communications Security. 2006: 255-263.

[70] Bissias G D, Liberatore M, Jensen D, et al. Privacy vulnerabilities in encrypted HTTP streams[C]//Privacy Enhancing Technologies: 5th International Workshop, PET 2005, Cavtat, Croatia,

May 30-June 1, 2005, Revised Selected Papers 5. Springer Berlin Heidelberg, 2006: 1-11.

[71] Lu L, Chang E C, Chan M C. Website fingerprinting and identification using ordered feature sequences[C]//Computer Security–ESORICS 2010: 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings 15. Springer Berlin Heidelberg, 2010: 199-214.

[72] Herrmann D, Wendolsky R, Federrath H. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier[C]//Proceedings of the 2009 ACM Workshop on Cloud Computing Security. 2009: 31-42.

[73] Bhat S, Lu D, Kwon A, et al. Var-CNN: A data-efficient website fingerprinting attack based on deep learning[J]. Proceedings on Privacy Enhancing Technologies, 2019, 4: 292-310.

[74] Rimmer V, Preuveneers D, Juarez M, et al. Automated website fingerprinting through deep learning[J]. arXiv preprint arXiv:1708.06376, 2017.

[75] Oh S E, Sunkam S, Hopper N. p1-FP: Extraction, classification, and prediction of website fingerprints with deep learning[J]. Proceedings on Privacy Enhancing Technologies, 2019, 2019(3): 191-209.

[76] Houser R, Li Z, Cotton C, et al. An investigation on information leakage of DNS over TLS[C]//Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies. 2019: 123-137.

[77] Siby S, Juarez M, Diaz C, et al. Encrypted DNS double right arrow privacy? A traffic analysis perspective[C]//27Th Annual Network And Distributed System Security Symposium (NDSS 2020). INTERNET SOC, 2020.

[78] Bushart J, Rossow C. Padding ain't enough: Assessing the privacy guarantees of encrypted {DNS}[C]//10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20). 2020.

[79] 张维维, 龚俭, 刘茜, 等. 基于词素特征的轻量级域名检测算法[J]. 软件学报, 2016, 27: 2348-2364.

[80] Sun P, Lyu M, Li H, et al. An early stage convolutional feature extracting method using for mining traffic detection[J]. Computer Communications, 2022, 193: 346-354.

[81] Yang T, Zhang H, Yang D, et al. Finding significant items in data streams[C]//2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE, 2019: 1394-1405.

[82] Cheng S, Yang D, Yang T, et al. LTC: A fast algorithm to accurately find significant items in data

streams[J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 34(9): 4342-4356.

[83] Zhou Y, Yang T, Jiang J, et al. Cold filter: A meta-framework for faster and more accurate stream processing[C]//Proceedings of the 2018 International Conference on Management of Data. 2018: 741-756.

[84] Ting D. Data sketches for disaggregated subset sum and frequent item estimation[C]//Proceedings of the 2018 International Conference on Management of Data. 2018: 1129-1140.

[85] Feng Y, Li J, Sisodia D. Cj-sniffer: Measurement and content-agnostic detection of cryptojacking traffic[C]//Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses. 2022: 482-494.

[86] Li X, Xu W, Liu B, et al. TuDoor attack: Systematically exploring and exploiting logic vulnerabilities in DNS response pre-processing with nalformed packets[C]//2024 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2024: 181-181.

[87] Zebin T, Rezvy S, Luo Y. An explainable AI-based intrusion detection system for DNS over HTTPS (DoH) attacks[J]. IEEE Transactions on Information Forensics and Security, 2022, 17: 2339-2349.

[88] Banadaki Y M, Robert S. Detecting malicious dns over https traffic in domain name system using machine learning classifiers[J]. Journal of Computer Sciences and Applications, 2020, 8(2): 46-55.

[89] Top websites ranking in the world[R/OL]. Alexa. https://www.alexa.com/topsites.

[90] Kosek M, Schumann L, Marx R, et al. DNS privacy with speed? Evaluating DNS over QUIC and its impact on web performance[C]//Proceedings of the 22nd ACM Internet Measurement Conference. 2022: 44-50.

[91] Yang D, Li Z, Jiang H, et al. A deep dive into DNS behavior and query failures[J]. Computer Networks, 2022, 214: 109131.

[92] Mitsuhashi R, Jin Y, Iida K, et al. Detection of dga-based malware communications from doh traffic using machine learning analysis[C]//2023 IEEE 20th Consumer Communications & Networking Conference (CCNC). IEEE, 2023: 224-229.

[93] Kouliaridis V, Kambourakis G, Peng T. Feature importance in android malware detection[C]//2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2020: 1449-1454.

[94] Sirinam P, Imani M, Juarez M, et al. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and

Communications Security. 2018: 1928-1943.

[95] Hayes J, Danezis G. k-fingerprinting: A robust scalable website fingerprinting technique[C]//25th USENIX Security Symposium (USENIX Security 16). 2016: 1187-1203.

[96] 奇安信威胁情报平台[R/OL]. https://www.qianxin.com/product/detail/pid/390.

单位代码：10427
密　　级：公　开