Unmasking the Internet: A Survey of Fine-Grained Network Traffic Analysis

Yebo Feng, Jun Li, and Jelena Mirkovic

***Abstract*—Fine-grained traffic analysis (FGTA), as an advanced form of traffic analysis (TA), aims to analyze network traffic to deduce information related to application-layer activities, fine-grained user behaviors, or traffic content, even in the presence of traffic encryption or traffic obfuscation. Different from traditional TA, FGTA approaches are usually based on sophisticated classification approaches such as machine learning and high-dimensional clustering, enabling them to discover subtle differences between different network traffic sets. Nowadays, with the increasingly complex Internet architecture, the increasingly frequent transmission of user data, and the widespread use of traffic encryption, FGTA is becoming an essential tool for both network administrators and attackers to gain different levels of visibility over the network. It plays a critical role in intrusion and anomaly detection, quality of experience investigation, user activity inference, website fingerprinting, location estimation, etc. To help scholars and developers research and advance this technology, in this survey paper, we examine the literature that deals with FGTA, investigating the frontier developments in this domain. By comprehensively surveying different approaches toward FGTA, we introduce their input traffic data, elaborate on their operating principles by different use cases, indicate their limitations and countermeasures, and raise several promising future research avenues.**

***Index Terms*—Network traffic, traffic analysis, traffic classification, traffic monitoring, fine-grained traffic analysis, intrusion detection, user behavior identification.**

I. Introduction

In the context of Internet, protocols and applications are usually built upon hierarchical models [1] (e.g., TCP/IP and OSI), where the communication functions of a telecommunication or computing system are categorized into several abstraction layers. Higher layers only encapsulate high-level methods, protocols, and specifications, operating with the support of lower layers [2]. With such design, programmers can easily develop interoperable Internet applications regardless of diverse underlying protocols and technologies. However, this convention also makes cross-layered network analysis feasible. As developers of higher layer applications usually only take higher-layer measures (e.g., encryption, anonymization, etc.) to preserve the user privacy regardless of leaving traceable patterns on lower layers, analyzers can capture network features from the lower layers to infer higher-layer knowledge in communication [3], even in the presence of message encryption. Such a process is called traffic analysis (TA), a technique widely used in today's Internet.

TA has been studied for decades, with myriad systems, tools, and algorithms [4]–[9] developed to serve different types of purposes, such as traffic measurement, traffic engineering, anomaly detection, and network surveillance. In early development of TA, traditional TA approaches were mainly designed for network traffic measurement/forecast [10]–[12], anomaly detection [13], and basic traffic classification [14]. These approaches are usually rule-based, statistics-based, or clustering-based, can separate traffic of different network protocols or conduct basic modeling of traffic flow changes. Later, with the adoption of cutting-edge data processing techniques and algorithms, such as harnessing the power of machine learning on big data, TA is able to deduce more information from network traffic data regarding application-layer activities, fine-grained user behaviors, and message content. For instance, researchers have developed advanced TA techniques to detect application-layer threats, infer the specific websites that people are visiting over HTTPS, or even dig users' private data from network-layer knowledge. We define such advanced TA techniques as fine-grained traffic analysis (FGTA), the process of application-layer behavior modeling, fine-grained user activities inferring, or traffic content decoding, only through link-layer or network-layer traffic data, with or without encryptions.

As a subset of TA, FGTA is mainly different from traditional TA in the following ways:

- The most notable difference is the goals of analysis. Traditional TA can coarsely distinguish or model traffic from different types of network device, protocols, or applications. However, FGTA aims to analyze traffic at a finer granularity, such as traffic from different application-layer activities (e.g., Twitter post vs. Tweeter read), different groups of application users (e.g., online social network (OSN) bots vs. normal users), or different user content (e.g., the visiting website).
- The analysis pipelines of traditional TA and FGTA are usually different. FGTA, aiming at more granular information, usually takes the traditional TA as a prerequisite step to "preprocess" the traffic before the final inference. For example, a FGTA approach that tries to identify the web page the user is visiting needs to first leverage traditional TA to extract all the web browsing traffic.
- As for analysis algorithms, most FGTA approaches depend on sophisticated modeling or classification methods, such as deep machine learning or high-dimensional clustering, to tackle the challenging fine-grained object identification tasks. While, traditional TA, dealing with

Yebo Feng and Jun Li are with the Computer Science Department, University of Oregon, Eugene, OR, 97403. (E-mail: yebof@uoregon.edu and lijun@cs.uoregon.edu)

Jelena Mirkovic is with the Information Sciences Institute, University of Southern California, Marina del Rey, CA, 90292. (E-mail: mirkovic@isi.edu)

Manuscript received XXXX XX, 202X; revised XXXX XX, 202X.

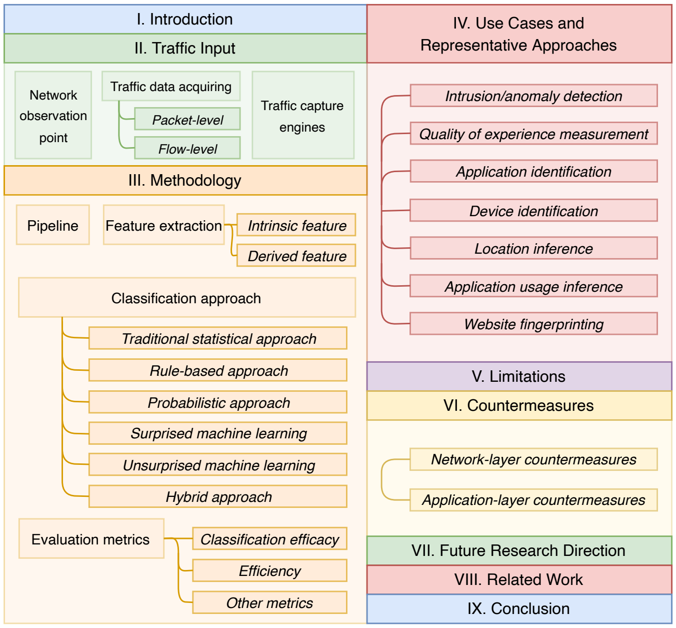
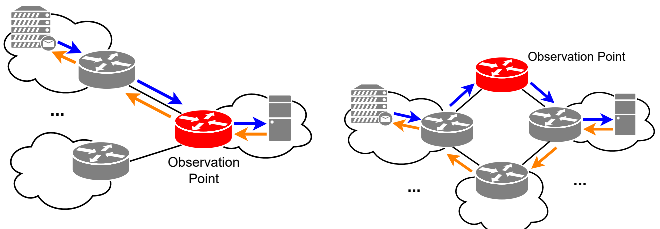


Fig. 1: The paper organization and snapshots of proposed taxonomies.

easier tasks, can utilize a number of different analytical methods, such as rule-based, statistics-based, or soft-computing-based approaches.

With the increasingly complex Internet architecture, increasingly frequent transmission of user data, and the widespread use of traffic encryption [6], FGTA is becoming a more and more important research topic. Compared with traditional TA, FGTA can reveal more information from network traffic and can achieve high efficacy even in various complicated network environments [15]. Besides, as network traffic data become more easily accessible than before, the applicable scenarios of FGTA are more extensive compared with directly analyzing traffic content. Furthermore, FGTA is efficient and portable in discovering application-layer knowledge [16]–[18]. By analyzing a small amount of metadata or statistical information of traffic, FGTA can obtain almost the same level of visibility as decoding large amount of message content. Therefore, FGTA has a wide range of usage scenarios. As for network managements, FGTA can help measure application usage [19], detect complicated network intrusions or anomalies [20], investigate edge user experience [21], etc. As for the attacker side, FGTA can help eavesdrop on private information of users [22], model user behaviors [23], estimate user locations [24], etc. Studying FGTA is essential for comprehensive network inspection, safeguarding information transmission, and precise network configuration.

In this paper, we examine the literature that deals FGTA. By including more than 230 citations, we first discuss the network traffic data and its collection in real-world environments. We then elaborate on frontier developments of FGTA, demonstrating and comparing different FGTA approaches' operating pipelines, classification approaches, and use cases. We further indicate the limitations and countermeasures to current FGTA approaches. In the end, based on our observations and reflections on this field, we propose several avenues for future research, thereby helping future academics and developers to advance FGTA. To our best knowledge, this paper is the first survey paper that focuses on FGTA and compares the state-of-the-art approaches in this field.



(a) The observation point is the gateway of the network. The traffic capture engine can collect bidirectional traffic data.

(b) The observation point is in the network. The captured traffic can be asymmetric.

Fig. 2: Network visibility with different observation locations.

The rest of this report is organized as follows. After describing the input data of FGTA in Section II, we discuss and summarize the methodologies of FGTA in Section III. We elaborate on frontier developments of FGTA by their use cases in Section IV. We then point out the limitations of existing FGTA in Section V and introduce the countermeasures in Section VI In the end, we propose some avenues for future research in Section VII and conclude this report in Section IX. Figure 1 illustrates the organization of this survey paper and give snapshots of the proposed taxonomies.

II. Traffic Input

Like traditional TA, FGTA approaches use network traffic data from some vantage points in the network as input to synthesize knowledge. Network traffic data refers to the information exchanged between devices on a computer network. Such data can be in diverse formats and include a wide range of information, such as communication logs, packet headers, and payload. The network traffic data is the inference object for all TA approaches. In this section, we compare different types of network traffic data and survey their capture engines. We also introduce these capture engines' deployments and application scenarios in FGTA.

A. Network Observation Point

The observation point of the traffic capture engine will significantly impact the integrity of the captured data and the network visibility. Different observation points are suitable for different types of TA tasks.

The ideal observation point for most FGTA tasks is located at the gateway of a network (illustrated in Figure 2a), which enables them to capture both inbound and outbound traffic of the network. Such a bidirectional traffic dataset is suitable to infer the interactions between the observed network and rest of the Internet. However, analyzers cannot learn about the traffic in the rest of the Internet according to this dataset.

Sometimes, the observation point can be in the middle of the network (illustrated in Figure 2b), especially when the traffic capture engine is deployed by an ISP or IXP. In this case, the capture engine is able to collect a large amount of traffic that pass by it. However, it also raises the following concerns:

- Due to asymmetric packet routing [25], in-network observation point sometime may only capture traffic in one direction (illustrated in Figure 2b).
- It cannot guarantee the robustness of captured traffic because of the deployment of various traffic engineering techniques [26], [27]. The routing path for any packet can be dynamic in today's networks.

Therefore, in-network-based observation points may be more suitable for traditional TA tasks such as Internet measurement and network-layer anomaly detection. As for FGTA, many approaches (e.g., user behavior inference, website fingerprinting) prefer to use the gateway-based observation point to capture more complete traffic data from endpoints. However, wherever the observation point is located, it is difficult to capture all the relevant traffic in the network.

To capture comprehensive traffic data from the network with complex topology, we can deploy multiple observation points at different vantage points if conditions permit. By using a pool of metering processes to collect network packets at multiple observation points, optionally filter them and aggregate information about these packets, a traffic exporter can gather each of the observation points together into an observation domain and sends this information to a traffic capture engine [28]. Then we can obtain relatively comprehensive network traffic data without redundancy. However, this approach is expensive to deploy and not always feasible due to real-world constraints.

B. Traffic Data Acquiring

Since the birth of the Internet, various traffic capture engines have been developed to log traffic information. TA approaches can further leverage these "log information" to measure network events, detect anomalies, and analyze network behaviors. Based on different information captured, these traffic capture engines can be classified into either packet-level or flow-level [4].

1) Packet-level capture: Packet-level capture is widely used in local networks and endpoint devices. As its name states, it copies or makes a snapshot of all the network packets that pass by the network interface and forwards the collected data to a collector. The agent that takes charge of the capture is called a packet-level traffic capture engine or a "sniffer", which can be either software-based (e.g., Snoop [29], Wireshark [30], etc.) or hardware-based (e.g., Sniffer InfiniStream [31]). It can be as simple as an IP table rule on a route that copies all the traffic to a cloud disk besides normal forwarding.

Packet-level capture can collect raw network traffic, containing both packet headers and packet payloads. Theoretically, it can support all types of FGTA tasks because it basically logs all the information flowing on networks. However, in most cases, packet-level traffic capture might not be the right solution to deploy for the following reasons:

- Packet-level traffic capture is expensive, not only because the interface needs to copy all the packets that pass by it, but also because the interface needs to forward all the captured traffic to an analysis node through a link. All these operations will double the workload of the network interface and occupy a considerable amount of link bandwidth. Packet-level traffic capture is therefore not scalable.
- The information contained in packet-level traffic data is sometimes an "overkill" for TA, as many TA approaches only require statistical information from the packet headers to complete the analysis. Moreover, user messages, website content, and video streaming are usually contained in packet payloads in encrypted forms, making most information captured in packet-level traffic meaningless for all TA approaches.
- Packet-level traffic may contain sensitive information (i.e., payload) of users. Thus, network service providers are cautious about capturing and analyzing such data.

2) Flow-level capture: To address the aforementioned issues of packet-level traffic captures and make traffic capturing affordable, scalable, and practical for network service providers, researchers and developers have proposed myriad flow-level traffic capture engines.

In flow-level traffic capture systems, the capture engines no longer copy or make snapshots of each packet, instead, they first aggregate relevant packets into a flow and then capture metadata or statistical information to represent that flow. Here, the concept of flow has been around for a long time. Typically, a flow can be identified by either a 5-tuple (i.e., source IP address, source TCP/UDP port, destination IP address, destination TCP/UDP port, and IP protocol) or a 3-tuple (i.e., source IP address, destination IP address, and IP protocol) [32]. However, with the development of flow capture engines, researchers have proposed many other formal and informal definitions of network traffic flows (e.g., RFC 2722 [33], RFC 3697 [34], RFC 3917 [35], etc.). In this paper, we define a network traffic flow as a sequence of relevant network packets from a source to a destination for the same application. In most instances, the network system will process packets within a flow in the same manner. Besides, each application-layer behavior will generate one or multiple flows in both directions.

By capturing traffic at flow-level, traffic capture engines no longer suffer from high system overhead and high bandwidth usage. Figure 3 illustrates the workflows of a network interface with and without Cflowd as the flow-level traffic captured engine [36]. Unlike packet-level traffic capture that will copy and forward any packet entirely to the collector port, flow-level traffic capture only copies information from headers to assemble traffic flows. The volume of data to process is then largely reduced in such a procedure. According to existing literature [37], NetFlow, the most frequently used flow-level traffic capture engine, only creates 1-1.5% of throughput (without sampling) on the interface it is exported on [38]. With a great deal of data reduction, network administrators can store, process, inspect and analyze large amounts of network data efficiently. Furthermore, when combining this procedure with packet sampling, it becomes feasible to capture and store traffic flows at an ISP or IXP scale, thereby extending the usage scenarios of TA. As we can see from a study, NetFlow

TABLE I: Comparisons of selected widely-used traffic capture engines (●: fully support; ◐: partially support; ○: not support.).

Traffic Capture Engine	Data Captured	Granularity	Open or Proprietary	Layer (OSI)	Hardware Acceleration	Sampling
SNMP [40]	High-level statistical information about the interface.	Flow-level (aggregated)	Open	2, 3	○	○
IPFIX [41]	Metadata and statistical information about the flow.	Flow-level	Open	3, 4	●	●
NetFlow v9 [42]	Metadata and statistical information about the flow.	Flow-level	Proprietary	3, 4	●	●
NetFlow v5 [43]	Metadata and statistical information about the flow.	Flow-level	Proprietary	3, 4	●	●
Argus [44]	Metadata and statistical information about the flow.	Flow-level	Open	2, 3, 4	●	○
sFlow [45]	Complete packet headers and partial packet payloads.	Packet-level	Partially Open	2 - 7	●	●
Tcpdump [46]	Network information pass through the observation point.	Packet-level	Open	2 - 7	○	○
Wireshark [30]	Network information pass through the observation point.	Packet-level	Open	2 - 7	○	○
PF_RING [47]	Network information pass through the observation point.	Packet-level	Open	2 - 7	●	◐
Netmap [48]	Network information in the memory of the observation point.	Packet-level	Open	2 - 7	●	◐

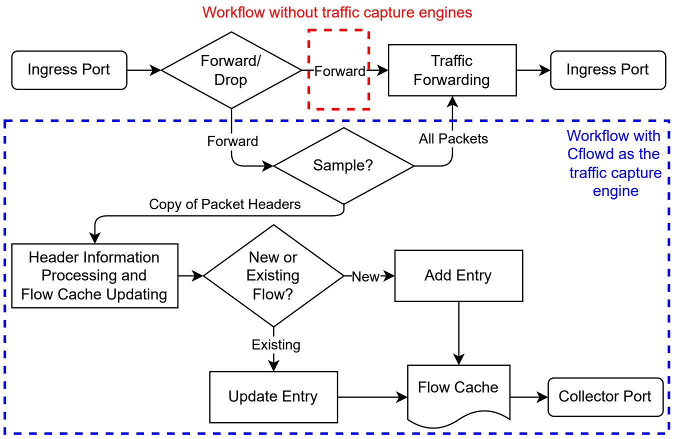


Fig. 3: Workflow of a network interface when Cflowd serves as the traffic captured engine.

only occupies around 15% of the router/switch's CPU load when capturing sampled network traffic [39]. Compared with packet-level traffic capture that sometimes may double the system overhead and link usage, flow-level traffic capture is a huge improvement regarding efficiency and deployability.

However, the shortcoming of flow-level traffic capture is also obvious—it will decrease the visibility of the network traffic because people only see metadata and aggregated statistical information about the traffic rather than each packet. This is especially troublesome for FGTA as many approaches require at least inter-packet information (e.g., packet interval time). To make up for this, we can shorten the lifecycle for each flow in traffic capture engines to let them generate flows more frequently, thereby increasing the network visibility.

C. Widely used traffic capture engines

Here, we introduce widely-used traffic capture engines in academia and industry (Table I shows comparisons of them).

1) Packet-level traffic capture engines: Back in the early days of Internet, developers had realized the importance of capturing network packets for troubleshooting. Thus, Tcpdump [46], a software-based packet-level traffic capture engine (sniffer), was proposed in 1988. It allows users to store and display TCP/IP and other packets being transmitted or received over a network. Nowadays, Tcpdump has been ported to several operating systems (e.g., Unix with libpcap library, Windows with WinPcap) and is still frequently used in network studies. Similar software-based sniffers were also proposed to meet different needs. For example, Snoop [29], a simple packet capture tool that is bundled on Solaris operating system; Wireshark [30], a free packet capture and analysis software that not only supports multiple operating systems (e.g., Linux, Solaris, Windows, FreeBSD, Mac OS, etc.), but also comes with a user-friendly interface; PF_RING [47], a high speed packet capture library that can turn a commodity PC into an efficient and cheap network measurement box suitable for both packet capture and TA. As for routers and switches, traffic mirroring [49]–[51] is also well-studied, with many software or hardware-based approaches [31], [52] proposed to support real-time packet capture for enterprise-level networks.

However, as capturing the entire packet is expensive and sometimes impractical, people began to make a snapshot of each packet rather than storing it entirely. The most frequently-used approach is sFlow [45], an industrial method (defined in RFC 3176 [45]) originally developed by InMon Inc., to capture packet-level snapshot from switches and routers. Compared with previous packet-level traffic capture engines, sFlow has the following features, making it the ideal input for most FGTA approaches:

- Without capturing the entire packet, sFlow can just copy the first N bytes of a packet to save computing and transmission resource. This is especially useful for TA tasks as packet payloads are useless in such scenarios but the entire packet headers are still preserved for fine-grained analysis.
- As an industrial standard, sFlow is compatible on many different platforms of network switches and routers and utilizes a dedicated chip built into the devices to operate, which removes the burden of the CPU and memory of the router or switch when capturing the traffic.
- By introducing time-based or packet-based sampling techniques, sFlow can capture traffic on all interfaces simultaneously at wire speed.

Therefore, sFlow can reach a good balance between data integrity and velocity—being able to capture all the packet headers and simultaneously create less burden on the router or switch.

2) Flow-level traffic capture engines: Flow-level traffic capture engines also have a long history. Back in 1984, the Audit Record Generation and Utilization System (Argus flow [44]) was proposed as the first implementation of network flow monitoring, and is still an ongoing open source network flow monitor project now. Argus can monitor all network traffic, including Internet Protocol (IP) traffic, data plane, control plane and management plane. It captures much of the packet dynamics and semantics in each flow, providing reachability, availability, connectivity, duration, rate, load, delay metrics for all network flows. It also captures most attributes that are available from the packet headers [53]. Later, in 1988, Simple Network Management Protocol (SNMP) [40] was proposed as a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). Unlike Argus flow that provides rich information about ongoing traffic, SNMP only provides statistical information per interface, such as link utilization, interface bandwidth, and some other information if the device provides. SNMP is thus less applicable in TA compared with Argus, especially in the domain of FGTA.

With rapid development and popularization of the Internet, the industry had realized the importance of flow-level traffic capture engine and many solutions were proposed. The most typical example is NetFlow [42], so far the most widely-used flow-level capture engine with many TA approaches built upon. Just like Argus, NetFlow uses a flow record to represent a set of packets. However, unlike Argus, which is a bidirectional monitoring approach, NetFlow is a unidirectional flow monitor, reporting flow information of each direction of conversations independently. This feature allows NetFlow to have a finer granularity than Argus. Since NetFlow was developed by Cisco, it is bundled with most Cisco routers and switches, making it the object of imitation of the entire industry. Following NetFlow, many similar systems were proposed by both research institutions and commercial companies, such as Cflowd [36], J-Flow [54], NetStream [55], Remote Network Monitoring (RMON) [56], etc. NetFlow itself also has evolved into different variations. The most famous one is Internet Protocol Flow Information Export (IPFIX) [41], an IETF protocol built upon NetFlow v9.

The most recent development of traffic capture and traffic handling have been mainly focusing on the velocity issue. Researchers have proposed multiple approaches to capture large volume of network traffic at line speed without having any effect on data plane. For example, Netmap [48] a memory-based framework that enables commodity operating systems to handle millions of packets per seconds without the support of custom hardware; eXpress Data Path (XDP) [57], a fast programmable packet processing approach based on the operating system kernel, supports high speed packet logging and processing; hXDP [58], an efficient software network packet processing approach written in extended Berkeley Packet Filter (eBPF) on Field Programmable Gate Arrays (FPGA) network interface controllers (NICs); NetSeer [59], a flow event telemetry (FET) monitor, which aims to discover and record all performance-critical events on the programmable data plane. However, those approaches do not change the pipeline of TA or FGTA, as they only make it faster to capture and handle network traffic.

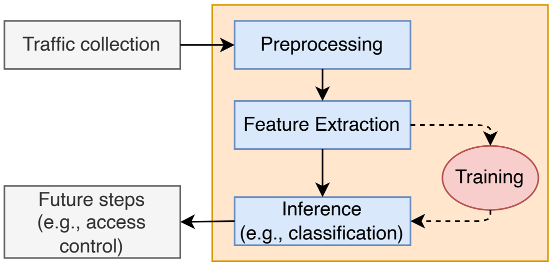


Fig. 4: A general data processing pipeline for FGTA.

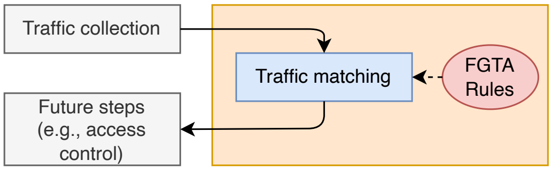


Fig. 5: Example of a simplified FGTA data processing pipeline.

III. Methodology

In this section, we delve into the methodology of FGTA and explore this field from the perspectives of data processing pipelines, feature extraction approaches, classification approaches, and evaluation metrics. These components are integral to the success of FGTA and play a crucial role in achieving accurate results.

A. Pipeline

The process of generating fine-grained analysis results from raw network traffic collected from network infrastructures typically involves several necessary steps. These data processing procedures are known as the FGTA pipeline. Different FGTA approaches may have different pipelines, with different steps and different orders. In this subsection, we discuss three types of FGTA pipelines (illustrated in Figure 5, 4, and 6).

Figure 4 illustrates the most commonly-used pipeline for FGTA. Regardless of whether the input traffic is in flow-level or packet-level format, it usually cannot be directly processed by analysis algorithms. Therefore, the first step of the FGTA pipeline is usually to preprocess the raw traffic data. The preprocessing step typically involves the following tasks:

- **Data decoding:** the raw network traffic data is usually encoded in a format that is not easily processable (e.g., binary format or encrypted form). This task converts the raw traffic data into a readable and processable form.
- **Data cleaning:** the raw traffic data may contain some noise, invalid data, or control messages. This task extracts only the valuable data for further analysis.
- **Data refactoring:** this task refactors the raw network traffic data and make it suitable for the subsequent analysis or maintenance. For example, indexing the raw traffic data by socket pairs, or converting the flow records to a B tree structure [60].

- **Other tasks necessary for subsequent steps:** depending on different FGTA pipelines, there may be other tasks necessary for subsequent steps. For example, extracting marked packets from the raw traffic data, anonymizing the raw traffic data for General Data Protection Regulation (GDPR) compliance [61], or compressing the data for efficient storage.

After the preprocessing step, FGTA approaches usually move to feature extraction, which refers to the process of selecting and transforming raw network traffic data into a set of relevant features that are suitable for machine learning, inference, or other analysis steps. For both traditional TA and FGTA approaches, the feature extraction is a particularly important step for representing the ongoing network events and achieving accurate results. We further discuss more details about feature extraction in Section III-B.

After relevant features are extracted, FGTA approaches are typically ready for inference. The inference goals of these approaches can vary, including identifying specific network events, classifying traffic flows based on different application behaviors, or detecting network anomalies. We further discuss the use cases of FGTA approaches in Section IV. The inference results of FGTA approaches can be used for a variety of purposes, including network monitoring, access control, device management, data center protection, etc. However, regardless of the inference goal, the inference step of FGTA always operates in the form of fine-grained classification. For example, classifying outlier traffic flows from normal traffic flows (i.e., anomaly detection), or classifying traffic flows according to different applications (i.e., application identification). Therefore, we use the term *classification* to refer to the inference step of FGTA approaches. Section III-C discusses the classification approaches used in FGTA.

The previously mentioned pipeline outlines the general steps for FGTA approaches. However, depending on the specific goals, system design, and operational environments, the FGTA pipeline can be simplified or extended, with specific steps omitted or added.

Figure 5 illustrates a simplified FGTA pipeline, where the raw network traffic data is directly used for rule-based traffic pattern matching. A short data processing pipeline is very efficient to operate and can still generate accurate results if the pre-defined matching rules are effective. It is useful when target traffic pattern is distinct or well-defined (i.e., location inference [24]). However, this short pipeline is not suitable for most of the FGTA approaches, as they often require additional steps to more thoroughly analyze the traffic data for generating fine-grained, application-layer classification results.

To infer high-level, fine-grained information from content-agnostic network traffic data, most FGTA approaches employ more complicated pipelines to mine hidden knowledge. Figure 6 illustrates such an example. Many application usage inference approaches apply similar pipelines [23], [62], [63] because they need to extract features and classify traffic for multiple times at different phases to derive detailed user behavior information of specific applications. The sample pipeline include two different classification steps, with one for narrowing down the analysis scope and the other for generating fine-grained classification. More importantly, this pipeline reassemble traffic flows into sessions (some papers may call them transactions [18] or bursts [19], [64]) before extracting features for the final classification. This step is very helpful for digging fine-grained behavior information from the traffic data because the target network behavior or event usually consist of multiple packets or traffic flows. Simply analyzing the network traffic flow by flow or packet by packet may not be able to capture the whole picture of the ongoing network events. Therefore, session assembly is used to aggregate adjacent, relevant, or similar traffic data into an analysis unit, which is a more representative data structure to present the ongoing network events and makes it possible to infer fine-grained application-layer information. Figure 7 illustrates an example of session assembly [63], where flow records are divided into flow points and then aggregated into sessions according to the traffic density. Based on current literature, the following approaches are commonly used for session assembly:

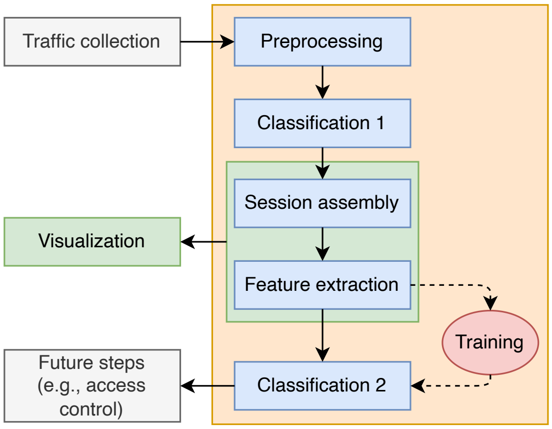


Fig. 6: Example of a more complicated FGTA data processing pipeline.

- **Time-based session assembly:** this approach aggregates traffic flows into sessions based on the timing or the intervals of ongoing network traffic.
- **Clustering-based session assembly:** this approach utilize clustering algorithms to group traffic flows into sessions.
- **Index-based session assembly:** this approach aggregates traffic flows into sessions by specific indexes (e.g., socket pair, packet ID ranges, time to live (TTL), etc).
- **Rule-based session assembly:** this approach aggregates traffic flows based on pre-defined rules (e.g., rules on the hash value of packet payload, rules on TCP flags, etc.).

After sessions are assembled, representative features can be properly extracted for fine-grained classifications.

B. Feature extraction

Feature extraction is a term refers to the process of selecting and generating relevant features from the raw data in order to create a representation that can be used for machine learning, statistical modeling, or other analysis procedures [65]. In the context of FGTA approaches, feature extraction involves

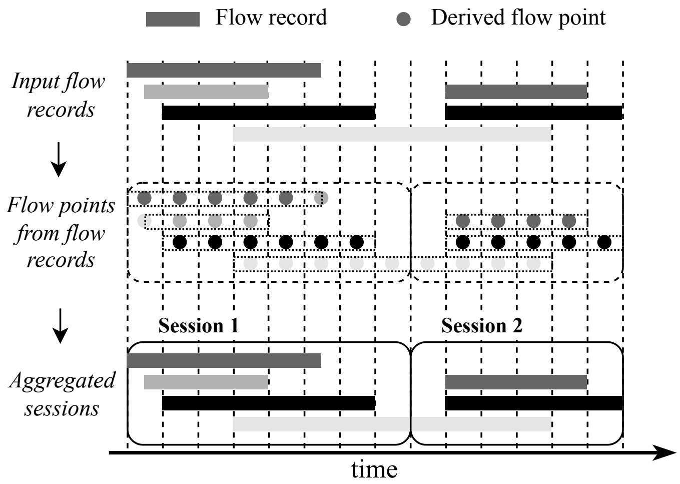


Fig. 7: Example of a session assembly procedure, where relevant flow records are aggregated into a traffic session to represent a network event.

TABLE II: Examples for intrinsic and derived features.

Category	Example	
Intrinsic feature	Flow-level	Packet-level
	Flow size, number of packets, AS number, protocol type, flow duration, etc.	TCP flag, ToS, packet size, packet interval, first n bytes of the payload, etc.
Derived feature	Flow/packet-based	Session-based
	Interval deviation, size deviation, interval distribution, inbound/outbound packet ratio, packet similarity, etc.	Session duration, density distribution, session image, normalized session vector, round-way communication number, etc.

inspecting network traffic data to identify relevant features that can be used for the corresponding classification tasks. This process typically involves techniques such as packet inspection, data fusion, and statistical modeling to identify and derive patterns or characteristics in the data that are relevant to the specific inference goal. The resulting set of features is then used as input to the classification model.

Due to the nature of network traffic collection, all the extracted features can be categorized into two types: intrinsic features and derived features. Intrinsic features are directly contained in the raw network traffic data, such as packet length, packet header fields, etc. The process of fetching intrinsic features is simple and straightforward. The analysis system can directly select, slice, or generate intrinsic features from the raw data, requiring little to no additional processing. On the other hand, derived features are not directly contained in the raw network traffic data. They are generated by applying some data processing techniques to the raw data, such as statistical modeling, feature transformation, information assembly, data fusion, etc.

Table II lists some typical examples of intrinsic and derived features. Different features are suitable for different FGTA tasks. Typically, some relatively easy FGTA tasks may only require intrinsic features to operate. For example, some application identification or anomaly detection approaches can generate accurate results by directly inputting intrinsic features. Because the network traffic of such applications or anomalies can already be distinguishable by intrinsic features [17], [66], [67]. However, some more complex FGTA tasks may require derived features for finer granularity analysis, especially for tasks that infer detailed, application-layer user behaviors [19], [63], [68]. The target network traffic of these FGTA tasks is less distinguishable and may only show obvious patterns with sophisticated feature engineering techniques. We discuss more details about suitable features for different FGTA tasks in Section IV.

Although derived features are more powerful than intrinsic features in mining fine-grained information from network traffic data, they are also more complex and expensive to generate. One may need to apply sophisticated data processing techniques, such as traffic buffering, data fusion, session assembly, statistical modeling, etc., to fetch these derived features. Such processes are time-consuming and may require significant computational resources. As time-sensitive tasks, it is vital for FGTA procedures to be efficient and scalable, thereby outputting analysis results in a timely manner. Thus, carefully selecting and generating necessary features is a critical step for all FGTA approaches.

C. *Classification approach*

The key step of FGTA is to classify the target network traffic from others. Designing a proper classification approach determines the efficacy and performance of FGTA approaches. In many cases, constructing a classification model requires labeled data. In the context of FGTA, labeled data refers to network traffic data that has been manually labeled or annotated with ground truth information. This ground truth information typically includes information such as the application type, user behavior type, or whether the traffic is generated by malicious behavior or not. Obtaining labeled data can be a challenging and resource-intensive process. It typically requires a significant amount of manual effort and expertise to accurately label network traffic data. Researchers may be able to automate the labeling process with the help of other state-of-the-art classification approaches, but the accuracy of labels may not be ideal [69]. On the other hand, some classification approaches can be trained without labeled data or with other forms of prior knowledge. In the remaining of this subsection, we discuss the classification approaches that are commonly used in FGTA approaches (summarized in Table III).

1) Traditional statistical approach: Traditional statistical approaches leverage statistical properties, statistical models or some other mathematical methods to identify subtle differences or patterns in different groups of network traffic [80]. Typical examples of statistical approaches include distribution fitting [70], logistic regression [71], linear regression [20], etc. Traditional statistical approaches are widely used in traditional TA tasks because they are explainable, easy to implement, usually efficient to operate, and good at tackling relatively easy tasks. However, as FGTA tasks becoming more and more challenging, traditional statistical approaches are not sufficient to identify subtle differences in network traffic. Thus,

TABLE III: Summary of widely-used classification approaches in FGTA.

Category	Description	Representative algorithms/approaches	Pros	Cons	Use in FGTA	Reference
Traditional statistical approach	Leverage statistical properties or statistical models for FGTA.	Distribution fitting, regression, variance matching, etc.	Explainable, easy to implement, and efficient.	Poor efficacy.	Limited	[20], [70], [71]
Rule-based approach	Utilize a set of pre-defined rules to locate the target network traffic group.	Session signatures, traffic thresholds, predefined packet header fields, etc.	Explainable, easy to implement, and efficient.	Hard to define rules, poor efficacy, and poor flexibility.	Limited	[16], [21], [72]
Probabilistic approach	Approaches based on probability theories to analyze network traffic.	Bayesian classifier, Markov model, HMM, etc.	Flexibility, adaptability, and ease of use.	Complexity, sensitivity to assumptions, limited accuracy, and relatively poor explainability.	Popular	[66], [73], [74]
Supervised machine learning	ML methods that rely on learning knowledge from labeled data.	KNN, SVM, LSTM, etc.	Great efficacy, ease of use, and good flexibility.	Limited explainability, overfitting, dependency on labeled data, and limited scalability.	Most popular	[75]–[77]
Unsupervised machine learning	ML methods that do not require labeled training data and can discover patterns and relationships in the data on its own.	K-means, PCA, DBSCAN, etc.	Discovering unknown patterns, flexibility, no labels needed, and no training time.	Limited result interpretability, limited efficacy, poor scalability in inference, and overfitting.	Popular	[63], [78], [79]
Hybrid approach	Combine multiple classification approaches for better performance.	Ensemble model, semi-supervised machine learning, combining statistical approaches with rule-based approaches, etc.	Inherits advantages of multiple classification approaches.	Complicated to design, and computationally expensive.	Popular	[17], [70], [79]

traditional statistical approaches are gradually replaced by more sophisticated classification approaches, such as machine learning approaches. Still, traditional statistical approaches are commonly used in feature extraction, pre-analysis, and pre-classification. For example, many FGTA approaches use traditional statistical approaches to narrow down the analysis scope before fine-grained analysis, thereby reducing the computational complexity of the subsequent procedures [70].

2) Rule-based approach: Rule-based approaches are based on a set of pre-defined rules that are manually designed by experts to locate the target network traffic group [81]. Before defining the classification rules, the experts usually need a thorough understanding of the target network traffic and the network environment. Typical examples of classification rules include session signatures [21], traffic thresholds [72], pre-defined packet header fields [16], etc. Similar to traditional statistical approaches, rule-based approaches are explainable, easy to implement, and efficient to operate, thereby being widely used in traditional TA tasks. However, in the era of FGTA, the analysis tasks are in finer granularity and becoming more and more challenging. Thus, the pre-defined rule sets are becoming larger, more complex, making them more difficult to define, verify, and maintain. Moreover, the rule-based approaches are not able to adapt to the dynamic network environment, which is a common feature of modern networks. Therefore, in recent trends, rule-based approaches are less used in FGTA tasks. But they are still powerful tools in some specific FGTA tasks, pre-classification, and accelerating the analysis process.

3) Probabilistic approach: Probabilistic approaches are based on probability theory and statistical inference to identify the target network traffic group [82]. They model the traffic data probabilistically for classification tasks. For instance, typical probabilistic approaches like Bayesian classifier [83], Markov model [84], or hidden Markov model (HMM) [85] are widely used to model network traffic first. These models can then be utilized to identify traffic patterns of specific applications, protocols, anomaly, or behaviors. Benefitting from the following advantages, a variety of FGTA approaches have been proposed based on probabilistic approaches to tackle different FGTA tasks [66], [73], [74]:

- Flexibility: probabilistic approaches can be used to model a wide range of traffic patterns and behaviors. Besides, they can tolerate noise and uncertainty in the data, making them powerful tools for analyzing complex and heterogeneous traffic data.
- Adaptability: probabilistic approaches can be easily adapted to changes in traffic patterns over time, allowing them to detect new or previously unseen threats.
- Ease of use: with supports of various libraries, probabilistic approaches are relatively easy to implement and does not require extensive domain knowledge or expertise.

However, probabilistic approaches feature the following disadvantages, resulting limited performance and application especially in complicated FGTA tasks (e.g., user behavior inference):

- Complexity: probabilistic approaches are usually computationally expensive, especially when the network traffic data is large and complex.
- Sensitivity to assumptions: probabilistic approaches are sensitive to the assumptions (labels) made during model training or development, and incorrect assumptions can lead to inaccurate results.
- Limited accuracy: probabilistic approaches may not achieve the highest accuracy compared to other more advanced methods, such as deep learning, in some scenarios. Also, they are relatively weak in handling nondiscrete data.
- Explainability: probabilistic approaches may not provide as much interpretability as other methods, making it difficult to understand how the models arrived at their conclusions.

4) Supervised machine learning: Supervised machine learning is a widely used machine learning method that can be applied to almost any FGTA tasks with reliable prior knowledge [15]. In supervised machine learning, a classifier is trained using a labeled training dataset that includes known classification labels. The trained classifier is then used to classify or detect anomalies in new traffic data. Supervised machine learning approaches typically involve two main phases: training and inference. During the training phase, the classifier is trained on the labeled dataset (i.e., labeled network traffic) to learn the relationship between the input features and the classification labels. The inference phase involves using the trained classifier to infer the classification labels of ongoing network traffic.

With decades of development, researchers have proposed a variety of supervised machine learning approaches [86], from traditional machine learning methods, such as k-nearest neighbor (KNN), decision tree, Support Vector Machine (SVM), to advanced deep learning methods [87], [88], such as multilayer perceptron (MLP), recurrent neural network (RNN), long short-term memory (LSTM). Each of the proposed supervised machine learning approaches has its own advantages and disadvantages, making them suitable for different FGTA tasks. Selecting the most suitable supervised machine learning approach is the key to designing an effective ML-based FGTA approach. We discuss more details about the ML algorithm select by use case in Section IV.

Overall, due to the following advantages, supervised machine learning approaches are the most widely used approaches in FGTA [20], [75]–[77]:

- High accuracy: supervised machine learning can achieve high accuracy in FGTA, especially when compared to other methods.
- Ease of use: on the one hand, supervised machine learning approaches are relatively easy to implement, with supports of various libraries and tools [89]–[91]. On the other hand, they do not require extensive domain knowledge or expertise to manually identify distinguishable rules or patterns.
- Flexibility: supervised machine learning approaches can be used to model a wide range of traffic patterns and behaviors.

However, supervised machine learning approaches also have many shortcomings that limit their performance and use cases:

- Limited explainability: many supervised machine learning algorithms, such as deep neural network (DNN), can be difficult to interpret, which can limit their usefulness in some applications, especially in anomaly or attack detection.
- Overfitting: supervised machine learning models can overfit to the training data, which can result in poor performance on new data.
- Dependency on labeled data: supervised machine learning requires high-quality labeled training data, which can be time-consuming and expensive to collect, making them less effective than unsupervised or semi-supervised methods in some cases.
- Limited scalability: many supervised machine learning approaches may not scale well to extremely large or complex datasets. Both the training and inference phases may be computationally expensive.

5) Unsupervised machine learning: Unlike supervised machine learning, unsupervised machine learning is a machine learning method that does not require labeled training data and can discover patterns and relationships in the data on its own [92]. Unsupervised machine learning algorithms typically involve clustering [93] or dimensionality reduction [94] techniques that can help identify similarities and differences between traffic flows. These algorithms do not directly output labeled classification results, but can be used to group similar traffic flows together or identify anomalous traffic flows that do not fit into any of the existing clusters. Widely used unsupervised machine learning algorithms include K-means [95], DBSCAN, principal component analysis (PCA) [96], hierarchical clustering [97], etc.

Unsupervised machine learning algorithms feature the following advantages in FGTA:

- Discovering unknown patterns: unsupervised machine learning approaches can identify previously unknown patterns and behaviors in the traffic data, which can be useful for detecting new or emerging threats.
- Flexibility: unsupervised machine learning approaches can be more flexible and adaptable than supervised machine learning as they do not require labeled data, making them easy to work with a wide variety of traffic datasets.
- No training time: unsupervised machine learning approaches usually takes zero training time, making them more efficient than supervised machine learning approaches regarding model development.

They also inevitably have the following disadvantages:

- Limited result interpretability: interpreting the results of the clustering or dimensionality reduction algorithms used in unsupervised machine learning can be difficult without prior domain knowledge.
- Limited accuracy: unsupervised machine learning may not achieve the same level of accuracy as supervised machine learning, especially when dealing with complex or noisy traffic datasets.
- Scalability in inference: although taking no time for training, some unsupervised machine learning algorithms are computationally expensive in the inference phase, which can limit their scalability.
- Overfitting: unsupervised machine learning models can also suffer from overfitting or underfitting, which can result in poor performance on certain datasets.

In conclusion, unsupervised machine learning is a powerful tool for FGTA, but it may have limitations in terms of result interpretability and accuracy. Due to such natures, unsupervised machine learning approaches are not widely used in tasks such as anomaly detection and attack detection [63], [78].

6) Hybrid approach: Hybrid approaches combine the advantages of multiple classification approaches to achieve better adaptability, explainability, or performance. People can use different approaches as different procedures in the FGTA

TABLE IV: Widely used classification efficacy metrics for FGTA, where TP denotes the number of true positives, TN denotes the number of true negatives, FP denotes the number of false positives, and FN denotes the number of false negatives.

Category	Description	Calculation
TPR	The probability that an actual positive will test positive.	$\frac{TP}{TP+FN}$
TNR	The probability that an actual negative will test negative.	$\frac{TN}{TN+FP}$
PPV	The probability that an item with a positive test result is truly positive.	$\frac{TP}{TP+FP}$
NPV	The probability that an item with a negative test result is truly negative.	$\frac{TN}{TN+FN}$
FNR	The probability of positives which yield negative outcomes.	$\frac{FN}{FN+TP}$
FPR	The probability of negatives which yield positive outcomes.	$\frac{FP}{FP+TN}$
FDR	The probability that an item with a positive test result is truly negative.	$\frac{FP}{FP+TP}$
FOR	The probability that an item with a negative test result is truly positive.	$\frac{FN}{FN+TN}$
F1	The harmonic mean of precision (PPV) and recall (TPR).	$\frac{2TP}{2TP+FP+FN}$
ACC	How close a given set of analysis results are to their true value.	$\frac{TP+TN}{TP+TN+FP+FN}$
ROC	A graph showing the performance of a classification model at all classification thresholds.	Through TPR and FPR.
AUC	The area under the entire ROC curve.	Through ROC.

pipeline, enhancing the feature extraction or pre-classification phase, or simply use an ensemble classification model to increase the robustness. For example, combining supervised and unsupervised machine learning approaches for semi-supervised traffic classification [17], combining statistical approaches with machine learning approaches better FGTA performance [70], or utilizing a variety of approaches in the FGTA pipeline for more comprehensive attack coverage [79].

Although hybrid approaches can usually achieve better performance, they are more complex to design and assemble. Besides, hybrid approaches are usually more computationally expensive than using single approaches.

D. Evaluation metrics

In FGTA, evaluation metrics are important measures of the performance, efficiency, and usability of proposed approaches. In this section, we discuss some commonly used evaluation metrics.

1) Classification efficacy: The most important evaluation metric for FGTA is the classification efficacy, which measures the accuracy of the proposed approach in classifying the target traffic flows. The classification efficacy is well-defined in the domain of data mining [98]. It can be measured by true positive rate (TPR), true negative rate (TNR), positive predictive value (PPV), negative predictive value (NPV), false negative rate (FNR), false positive rate (FPR), false discovery rate (FDR), false omission rate (FOR), F1 score (F1), accuracy (ACC), receiver operating characteristic (ROC), area under the curve (AUC), etc. Table IV lists the calculations and descriptions of these metrics. For specific FGTA tasks, some metrics may be more important than others. For example, in anomaly detection, FPR and FNR are more important than PPV and NPV because FPR determines the false alarm rate, reflecting the usability of proposed approaches, and FNR determines the miss rate, reflecting the detection effectiveness of proposed approaches. While in tasks such as webpage fingerprinting, PPV and NPV are more important than FPR and FNR because PPV and NPV are more relevant to the classification efficacy.

2) Efficiency: Efficiency is another important evaluation metric for FGTA, which is usually measured by the time cost of finishing analyzing a certain amount of traffic flows by the proposed approach. In addition to accuracy and other performance metrics, the time cost can have a significant impact on the proposed approach's practicality and applicability. In real-world scenarios, the majority of FGTA tasks are performed on a large amount of traffic flows in real time. Therefore, the bottom line of these FGTA tasks is to reach the line speed in processing traffic flows. Here, the line speed refers to the maximum speed at which a FGTA approach can process incoming traffic data without dropping or losing packets. It is typically measured in terms of bits per second (bps) or packets per second (pps).

To increase efficiency, many FGTA approaches optimize the feature extraction procedure, classification algorithms, or the general pipeline. Some approaches also choose to design dedicated hardware architectures to accelerate the FGTA process. For example, FlowLens [99] utilizes programmable switch to support ML-based flow classification at hardware level, making ML-based flow classification efficient enough to catch up with line-speed traffic. We discuss more details about this issue in Section IV.

3) Other metrics: According to different use cases, FGTA approaches may need to consider other metrics, such as the memory cost, the storage cost, compatibility, etc. Intrusion detection focused FGTA approaches may also need to consider the explainability of the outputs to help network administrators understand the reasons for the detection results, thereby facilitating the network security management with confidence. In addition, some FGTA approaches may need to consider the privacy of the users for compliance with certain regulations (i.e., GDPR). Thus, developers need to select suitable metrics for specific use cases.

IV. Use Cases and Representative Approaches

As discussed in Section I, FGTA has a wide range of uses. FGTA can be leveraged by both attackers and network administrators, for both illegal purposes and social good. According to their use cases, we propose a taxonomy for FGTA approaches (illustrated in Figure 8). In the rest of this section, We further examine typical FGTA approaches in each of the category.

A. Attack/Anomaly Detection

TA approaches are widely used by both the industry and academia to detect anomalies or attacks. With more than two decades of research, we have seen a myriad of solutions (e.g., [100]–[103]) targeting at different types of threats. However, as networks attacks become more and more sophisticated and traffic encryption is widely used by all the parties, detection

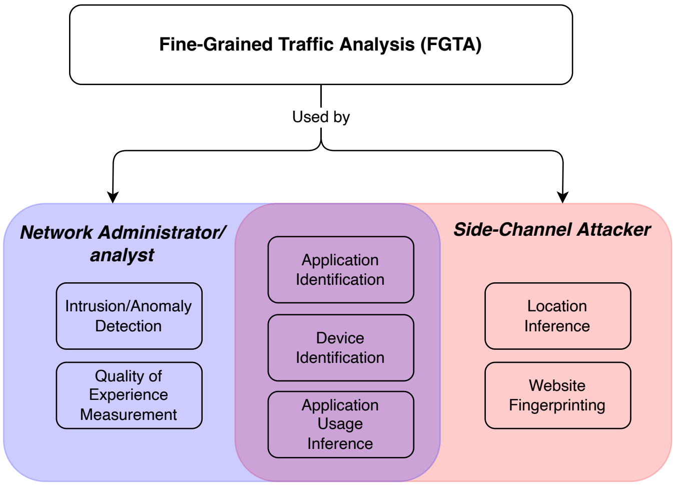

Fine-Grained Traffic Analysis (FGTA)
Used by
Network Administrator/ analyst
Intrusion/Anomaly Detection
Quality of Experience Measurement
Application Identification
Device Identification
Application Usage Inference
Side-Channel Attacker
Location Inference
Website Fingerprinting

Fig. 8: A taxonomy for FGTA by use case.

approaches based on traditional TA gradually become incompetent to tackle modern attacks. Therefore, researchers began adopting FGTA to model hosts and clients' application-layer behaviors to detect such attacks/anomalies. In this subsection, we elaborate on FGTA-based attack/anomaly detection approaches, introducing their applicable scenarios and operation mechanisms (Table V shows an overview).

1) Intrusion detection: Many FGTA approaches focus on detecting complicated intrusions in the network by examining the characteristics of the underlying network traffic. Most of them apply machine learning models to perform the detection.

Amoli et al. [78] leveraged an unsupervised machine learning model (i.e., density-based spatial clustering of applications with noise (DBSCAN)) to distinguish subtle differences between historic traffic and intrusion traffic. Their approach is able to detect zero-day and complex attacks without much prior knowledge of these attacks. Papadogiannaki et al. [112] generated traffic signatures from packet metadata sequences and then used these to detect intrusions in the UNSW-NB15 dataset [113].

Many researchers also focus on utilizing supervised deep learning models to detect intrusions. Tang et al. [67] extracted six basic features from traffic flows and trained a DNN model with the NSL-KDD dataset to detect intrusions. Shone et al. [79] first leveraged nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning. Then, they implemented stacked NDAEs with GPU-based architectures for quick and accurate intrusion detection on labeled datasets (i.e., KDD Cup '99 and NSL-KDD). Mirsky et al. [104] monitored the statistical patterns of network traffic and designed an ensemble of neural networks called autoencoders to collectively differentiate between normal and abnormal traffic patterns. Their approach is able to detect various attacks (e.g., video injection, ARP MitM, OS scan, etc.). Besides, unlike many other approaches that are only evaluated in closed-world environments, this approaches was tested with a real-world test bed.

2) Malware detection: Today's malware is becoming more and more challenging to be detected by traditional TA due to traffic hiding and the increasing adoption of traffic encryption. FGTA could be an ideal tool to detect such malware.

Shabtai et al. [105] proposed a framework for malware detection on Android platforms. It can identify attacks or masquerading applications installed on a mobile device and injected applications with malicious code by semi-supervised machine-learning methods. Wang et al. [106] leveraged a machine learning algorithm (i.e., C4.5 decision tree) in analyzing mobile traffic, which is capable of identifying Android malware with high accuracy—more than 98%.

Later, some researchers collectively evaluated the efficacy of different machine learning models in detecting malware. Lashkari et al. [107] detected malicious and masquerading applications with five different classifiers—random forest, KNN, decision tree, random tree, and regression. They found that these models can achieve similar performances in malware detection. Besides, they published a labeled dataset that contains both benign Android applications and injected applications' network traffic. Anderson et al. [20] designed and carried out experiments that show how six machine learning algorithms (e.g., linear regression, logistic regression, decision tree, random forest, SVM, and MLP) perform when confronted with real network data. They found the random forest ensemble classifier to be the most robust for the domain of malware detection.

3) Data exfiltration detection: FGTA can also be used in detecting data exfiltration, thereby protecting personal sensitive data from leakage. Different from directly detecting anomalies or attacks, approaches in this domain usually profile user behaviors or model normal application usage to identify abnormal data transfer.

Wei et al. [114] proposed ProfileDroid, which is the first approach to profile mobile application at four layers: (a) static, or application specification, (b) user interaction, (c) operating system, and (d) network. At network-layer, this approach can capture essential characteristics of application communications, including but not limited to the ratio of incoming traffic and outgoing traffic, number of distinct traffic sources, traffic intensity, the percentage of HTTP and HTTPS traffic, etc. The profiling information can help identify inconsistencies and surprising behaviors, thereby detecting data exfiltration. A similar work is TaintDroid [115]. It leverages dynamic information-flow tracking to identify private data leaks of Android applications. The authors indicated that network traffic is useful to help monitor the behavior of popular third-party Android applications and discover potential misuse cases of user private information across applications. These two approaches do not only leverage network traffic, but their ideas inspired a lot of subsequent work in this domain.

Later, researchers began to investigate purely using network traffic to profile application usage and report possible data exfiltration. Razaghpanah et al. [116] monitored network communications on mobile phones from user-space. The proposed approach facilitates user-friendly, large-scale deployment of mobile traffic measurements and services to illuminate mobile application performance, privacy and security. Song et al. [117] proposed a VPN-based approach to detect sensitive information leakage Le et al. [118] proposed AntMonitor,

TABLE V: Comparisons of selected attack/anomaly detection approaches (○: not support; ◐: partially support; ●: support).

Category	Approach	Year	Goal of Analysis	Feature	Method	Real-World Evaluation
Intrusion Detection	Amoli et al. [78]	2016	Mail-bomb, SSH-process-table, botmaster, etc.	Flow-level traffic feature such as duration, number of packets, smallest packet size, largest packet size, etc.	DBSCAN	○
	Tang et al. [67]	2016	R2L, U2R, Probe, DoS	Duration, protocol type, src bytes, dst bytes, count, srv count	DNN	○
	Shone et al. [79]	2018	R2L, U2R, Probe, DoS, guess password, portsweep, buffer overflow, etc.	Features extracted with NDAE	NDAE for unsupervised feature learning, stacked NDAEs for detection	○
	Mirsky et al. [104]	2018	Video injection, ARP MitM, OS scan, etc.	Damped incremental statistics and 23 other features from packet-level data	Kitsune's core algorithm (KitNET), a type of autoencoders	●
Malware Detection	Shabtai et al. [105]	2014	Malicious attacks or masquerading/injected mobile applications	2 best feature subsets selected from 20 manually defined feature subsets of various sizes	Linear regression, decision table, SVM for regression, Gaussian processes for regression, isotonic regression, and decision/regression tree	◐
	Wang et al. [106]	2016	Android malware such as plankton, FakeInstall, FakeRun, MobileTx, etc.	Six TCP flow features and four HTTP request features	C4.5 decision tree	○
	Lashkari et al. [107]	2017	Malicious and masquerading applications such as Airpush, Kemoge, AVpass, FakeAV, etc.	24 features extracted from both packet and flow-level traffic	Random forest, KNN, decision tree, random tree, and regression	◐
	Anderson et al. [20]	2017	Detecting malicious, encrypted malware network traffic	22 and 319 data features in the standard and enhanced feature set extracted from NetFlow and IPFIX data	Linear regression, logistic regression, decision tree, random forest, SVM, and MLP	●
Data Exfiltration Detection	Ren et al. [108]	2016	Cross-platform information leak identification	Raw network packets with payload	Decision tree, AdaBoost, bagging, blending, and Naïve Bayes	●
	Continella et al. [109]	2017	PII leakage detection, even in the presence of obfuscation techniques	Raw network packets with payload	Behavior modeling and differential analysis	◐
	Rosner et al. [110]	2019	Information leaks in TLS-encrypted network traffic	A feature space that includes observations about individual packets and sequences of packets; additional features from the phase detection and the full original traces.	Trace alignment, phase detection, feature selection, feature probability distribution estimation and entropy computation	○
Others	Feng et al. [63]	2021	Online social network bot detection	Traffic fingerprint images converted from NetFlow data	DBSCAN, CNN	●
	Coulter et al. [111]	2019	A data-driven cyber security system that can identify high-level application-layer attacks or anomalies such as Twitter spam	Statistical features extracted from the network traffic and content (optional)	A varity of classification approaches	○
	Feng et al. [70]	2022	Cryptojacking activity	Packet size, timing, direction, and protocol from sFlow data	LSTM	○

which passively monitors and collects packet-level measurements from Android devices to provide a fine-grained analysis. By inspecting the network traffic data, it can provide users with control over how their data is shared by applications. Ren et al. [108] proposed ReCon, a cross-platform system that reveals personally identifiable information (PII) leaks by inspecting network packets and gives users control over them without requiring any special privileges or custom operating system (OS). The authors leveraged the Weka data mining tool [119] to train classifiers that predict PII leaks. Continella et al. [109] proposed an approach to privacy leak detection that is resilient to obfuscation techniques (e.g., encoding, formatting, encryption). To achieve the goal, the authors first established a baseline of the network behavior of applications, and then utilized black-box differential analysis on application usages.

However, the aforementioned approaches still require inspections on traffic content to detect data exfiltration. The ideal FGTA-based solution should be content-agnostic. In 2019, Rosner et al. [110] presented a black-box approach for detecting and quantifying side-channel information leaks in TLS-encrypted network traffic. Given a user-supplied profiling-input suite in which some aspect of the inputs is marked as secret, it combines network trace alignment, phase detection, feature selection, feature probability distribution estimation and entropy computation to quantify the amount of information leakage that is due to network traffic.

4) Others: A few research works have been focusing on using FGTA to detect other types of application-layer anomalies. By harnessing the power of machine learning on big data, such approaches can model fine-grained application-layer anomalies only with flow-level traffic or packet headers. For example, BotFlowMon [18], [63] detects online social network bot traffic by converting NetFlow records to images and training a convolutional neural network (CNN)-based classification model; Coulter et al. [111] proposed a data-driven cyber security system that can detect Twitter spam or other high-level application-layer anomalies through machine-learning-based flow analysis; Feng et al. [70], [120] detects cryptojacking traffic by inferring the hash rate stability with cryptomining traffic in sFlow format. Table V lists their analysis features and methodologies.

TABLE VI: Comparisons of selected fine-grained QoE investigation approaches.

Approach	Goal	Method	Feature
[124]	Identify QoE degradation in YouTube	Random forest	Three feature sets selected by information gain.
[125]	Estimate QoE in YouTube	Random forest, J48, Naïve Bayes, OneR, and SMO	Five hand-crafted feature sets
[126]	Estimate video streaming QoE over HTTPS and QUIC protocols	Decision tree	A packet-level feature set extracted from network and transport-layers
[127]	Estimate QoE in YouTube	Random forest and linear regression	Three feature sets (inbound, outbound, and inbound + outbound)
[21]	Estimate mobile ABR video adaptation behavior over HTTPS and QUIC protocols	Traffic fingerprinting with chunk sizes	Packet size and timing

B. Fine-Grained Quality of Experience Investigation

Quality of experience (QoE) is a well-studied topic in the development of the Internet. Unlike quality of service (QoS), which refers to the network parameter settings configured by service providers to deliver various levels of service to their customers, QoE measures how the service is experienced by individual users at the edge of the network [121]. To score well in QoE, service providers need to analyze network traffic to conduct QoE investigations. The investigation results can help them revise the network configurations accordingly, thereby providing decent service to users.

Plenty of works have been proposed to conduct QoE investigations with traditional TA (e.g., [121]–[123]). They can roughly classify network traffic into several groups (e.g., video, voice, data transfer, etc.) using statistical, DPI-based, or rule-based approaches and measure the service experience according to some metrics. However, such approaches may not be able to tackle today's increasingly complicated network traffic, since different types of traffic may be encrypted by different protocols (e.g., HTTPS, Quick UDP Internet Connection (QUIC)) and sent from different devices (e.g., Internet of things (IoT), smartphone, server) by different applications. Besides, service providers may want to conduct more granular management of network traffic. For example, residential areas' network administrators may want to increase the priority of video streaming traffic related to YouTube for certain users; network administrators of companies may want to ensure the quality of online meeting traffic for some offices. Therefore, researchers began to leverage FGTA to conduct QoE investigation in finer granularities in the past ten years.

Usually, fine-grained QoE investigations are performed in two steps:

1) Extract the target traffic using traffic classification.
2) Measure the extracted traffic to check if it meets certain criteria.

Some approaches may combine these two steps into one and directly identify potential QoS/QoE problems. Table VI shows a comparison of some selected QoE methods.

In 2016, Dimopoulos et al. [124] proposed a random-forest-based detection model to identify QoE issues related to YouTube video streaming. By selecting three sets of features with information gain, the proposed model is able to directly detect different levels of QoE degradation that is caused by three key influence factors (i.e., stalling, the average video quality, and the quality variations). The authors demonstrated that it can detect QoE problems with an accuracy of 92% by evaluating this approach using collected traffic At the same year, Orsolic et al. [125] also studied using different machine learning algorithms (i.e., random forest, J48, Naïve Bayes, OneR, and Sequential Minimal Optimization (SMO)) to detect YouTube QoE issues under different bandwidth scenarios. In 2019, Khokhar et al. [127] proposed the first work that not only can identify YouTube QoE issues related to objective factors (e.g., startup delay, stalling, resolution change, etc.), but also can identify QoE issues related to the subjective Mean Opinion Score (MOS).

Mazhar et al. [126] further extends QoE investigation to all encrypted video streaming traffic (transferred over HTTPS or QUIC) by using a classification model trained by decision tree. They demonstrated that their approach is able to achieve a 90% classification accuracy for HTTPS and an 85% classification accuracy for QUIC Xu et al. [21] infers mobile Adaptive Bitrate (ABR) video adaptation behavior using packet size and timing information in encrypted environments.

C. Website Fingerprinting

Website fingerprinting (WFP) is used to identify what web page the user is visiting, even in the presence of traffic encryption or encrypted tunnels established by Tor [128], [129], Shadowsocks (i.e., a popular secure socks5 proxy) [130], VPN, etc. It is a FGTA technique that widely-used by attackers to eavesdrop on user activities online. In this subsection, we survey and compare well-known WFP approaches (Table VII), elaborating on the history of WFP and investigating its capability.

1) Early development of WFP: WFP has a long history. The early WFP attacks simply focused on using data sizes to infer the URL the user is visiting through encrypted SSL connections. Back in 1998, Mistry et al. [131] demonstrated that the size of HTML files is a critical feature to specific web pages. They proposed an attack that simply uses the transmitted data volumes to identify certain websites. Although this attack is not feasible anymore after the launch of connection pipelining and connection parallelization by HTTP 1.1 (RFC 2616 [141]), this research inspired many other WFP researches in the next two decades. In 2002, Hintz [142] defined "fingerprints" of websites as the histograms of transferred files' sizes. He recorded some website fingerprints and successfully recognize some websites transferred through HTTPS with these fingerprints. However, Hintz's WFP attack only works for a small number of websites. Later, Sun et al. [132] extends size-based WFP to thousands of websites. They proposed a WFP approach based on Jaccard's coefficient, which can correctly identify 75% of the websites in their

TABLE VII: Comparisons of selected WFP approaches (○: not support; ◐: partially support; ●: support).

Approach	Year	Method	Feature	Effectiveness			
				HTTP/1.1	VPN	Tor	Multi-tab
Mistry et al. [131]	1998	Size matching	Size of HTML file	○	○	○	○
Sun et al. [132]	2002	Similarity score calculation (Jaccard's coefficient)	HTTP object count, sizes, etc.	◐	○	○	○
Bissias et al. [72]	2005	Cross correlation of two value sequences	Packet size and inter-arrival time distributions	●	◐	○	○
Liberatore et al. [133]	2006	Similarity score calculation (Jaccard's coefficient)	Direction and length for each packet	●	◐	○	○
Herrmann et al. [134]	2009	Multinomial Naïve Bayes	Frequency distribution of the IP packet size	●	●	○	○
Panchenko et al. [135]	2011	SVM	Volume, time, and direction of the traffic	●	●	◐	○
Cai et al. [73]	2012	Damerau-Levenshtein distance and Hidden Markov Model	Packet size, time, and direction	●	●	●	○
Wang et al. [136]	2014	KNN	A large feature set generated from packet-level traffic	●	●	●	○
Hayes et al. [137]	2016	Random decision forests	Features selected by gini coefficient	●	●	●	◐
Rimmer et al. [138]	2017	SDAE, CNN, and LSTM	Automatically learned feature sets from packet-level network traffic	●	●	●	○
Sirinam et al. [139]	2018	CNN	Packet-level traffic data	●	●	●	○
Sirinam et al. [140]	2019	N-shot learning with triplet networks	Selected by a neural-network-based feature selector	●	●	●	○
Yin et al. [22]	2021	Split point finding and BalanceCascade-XGBoost	Packet size, time, and direction	●	●	●	●

collected dataset. However, a common drawback of file-based attacks is that they cannot tackle traffic hidden in encrypted tunneling protocols (e.g., VPN, OpenSSH), not to mention Tor.

2) Defeat encrypted tunnel: To extend WFP to handle encrypted tunneling protocols, multiple "more advanced" WFP approaches had been proposed. Both Bissias et al. [72] and Liberatore et al. [133] proposed improved forms of WFP. Rather than using the data size as the feature, they extract sets of traffic patterns from encrypted IP packet headers, such as packet inter-arrival time, size, etc. These approaches have some efficacy in identifying websites transferred by encrypted tunneling services. However, the accuracies of page identification is still not ideal in reality. In 2009, by using packet-level features, Herrmann et al. [134] proposed a multinomial Naïve Bayes classifier that can identify up to 97% of web requests on a sample of 775 sites and over 300,000 real-world traffic dumps recorded over a two-month period. The authors demonstrate that this approach is effective in tackling website traffic in encrypted tunnels. Lu et al. [143] pointed out that packet ordering information, though noisy, can be utilized to enhance website fingerprinting. In addition, the ordering information is effective for WFP even under traffic morphing. By calculating the Levenshtein distance between different network traffic, their approach can perform WFP over OpenSSH for 2,000 profiled websites. The identification accuracy of the proposed scheme reaches 81%, which is 11% better than the approach proposed by Liberatore et al. [133].

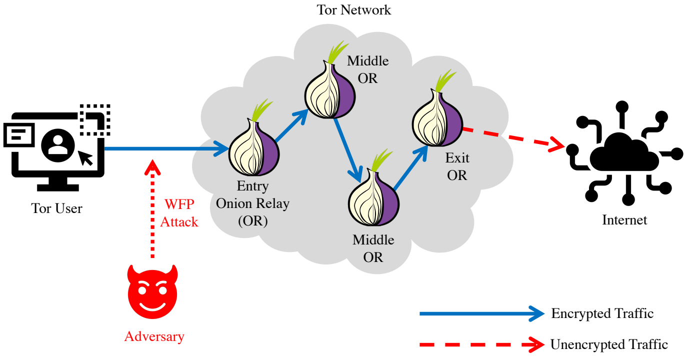


Fig. 9: Threat model for WFP attacks over Tor network.

3) WFP in Tor era: To safeguard personal information and avoid Internet censorship in an increasingly dangerous network environment, many people began to use The Onion Router (Tor), a free and open-source software for enabling anonymous communication, to visit the Internet. Different from traditional encrypted tunneling protocols, Tor reroutes Internet traffic through a worldwide, volunteer overlay network, consisting of more than six thousand relays [144], for concealing a user's actual location and Internet usage from anyone conducting network surveillance or TA.

Figure 9 illustrates the operation model of Tor. To protect user's identity, each Tor user creates an encrypted virtual tunnel to its destination through a chain of several volunteer nodes—onion relays (ORs). According to their positions in the virtual tunnel, ORs can be classified into entry OR, middle OR, and exit OR. Each of the ORs only knows its predecessor and its successor [145]. When forwarding network traffic, the user's network packets will be encrypted in multiple layers and each of the ORs can only decrypt one layer of encryption. Thus, Tor ensures that none of the ORs in the circuit knows the user and its destination at the same time. Besides, to prevent TA, the user data is encapsulated in chunks of a fixed size, called cells, before transmission [146]. The WFP attacks above are thus ineffective against Tor network, as they rely heavily on packet-size-related features.

Indeed, it is almost impossible to find any useful knowledge inside a Tor network. However, the virtual tunnel between the Tor user and the entry OR does provide attackers with an interface and makes WFP possible (illustrated in Figure 9).

In 2011, Panchenko et al. [135] are the first to demonstrate that it feasible to use WFP to identify web pages visited by Tor users. They trained a SVM classifier with features extracted from volume, time, and direction of network packets, with a classification accuracy of 55% when testing with their web page dataset. Panchenko et al. are also the first to evaluate their WFP attack in a real-world setting. The result shows that their approach is able to achieve a true positive rate of up to 73% and a false positive rate of 0.05%. Based on this work, a significant amount of improved WFP approaches were proposed to use different algorithms and features (e.g., VNG++ [147], Hidden Markov Models [73], Levenshtein-like distance [148], etc.) to tackle web page identification in Tor. In 2014, Wang et al. [136] proposed a KNN WFP classifier and applied it on a large feature set with weight adjustment. Their approach achieved an accuracy of 91% in a closed-world setting and a true positive rate of 85% for a false positive rate of 0.6% when testing with more than 5,000 background pages in a real-world setting.

Nevertheless, these WFP approaches still have some obvious flaws according to an evaluation made by Juarez et al. [149]:

- Previous WFP attacks assume single-tab browsing behavior of users. However, multi-tab browsing is widely used in reality.
- WFP attacks highly depend on the coverage of training dataset, but existing datasets cannot include web page traffic from all versions of Tor browser, user habits, or user locations.
- Previous WFP attacks cannot detect dynamic or personalized web pages, as they traffic of these pages is polytropic.
- Many countermeasures for WFP have been proposed (which will be discussed later in Section VI), making many of previous WFP attacks non-effective.

To further increase the success rate of WFP attacks and defeat countermeasures, researchers began to collect more comprehensive training datasets, use more complicated feature sets, and apply more sophisticated classification algorithms for WFP.

Wang et al. [148] described how they collect the training dataset in a much more thorough manner than previous works. They gathered the data in different Tor settings and with different defense approaches. Later, Panchenko et al. [150] collected the first Internet-scale WFP dataset to develop and evaluate WFP comprehensively. Based on the dataset, they proposed CUMUL, a web page classifier that has a higher recognition rate and a smaller computational overhead than previous approaches. They also demonstrated that although CUMUL is more efficient and superior in terms of detection accuracy, still, it cannot scale when applied in realistic settings. As for WFP feature set, Cai et al. [151] systematically analyzed previous WFP approaches to understand which traffic features convey the most information; Hayes et al. [137] utilized the gini coefficient index to select a feature set and designed a random decision forests classifier based upon them; Wang et al. [152] evaluated the classification accuracy of each feature category by using KNN.

In the recent five years, the development of WFP has been focusing on conducting attacks in the presence of effective countermeasures, with little encrypted data, or under complicated circumstances. Many of recent approaches also investigated the applicability of deep learning techniques in WFP. Rimmer et al. [138] trained three classification model with Stacked Denoising Autoencoder (SDAE), CNN, and Long Short-Term Memory (LSTM) respectively. These deep learning models are capable of automatically learning the best features to conduct WFP. The authors further demonstrated that automatically-created features are more effective especially in tackling constantly changing web content. In 2018, Sirinam et al. [139] presents a very powerful WFP attack—Deep Fingerprinting (DF). By employing a CNN model with a sophisticated architecture design, the authors claim that this attack can defeat many WFP countermeasures (e.g., WTF-PAD [153] and Walkie-Talkie [152]) and works well in very complicated real-world scenarios (95% accuracy for 20,000 URLs in a real-world setting). Sirinam et al. [140] further proposed an approach based on N-shot learning with triplet networks in 2019, which can achieve decent efficacy with relatively small training data. Besides these approaches, Abe et al. [154] also applied SDAE in WFP; Bhat et al. [155] leveraged ResNets [156], a CNN architecture, to reach high success rates in WFP; Oh et al. [157] used unsupervised DNN to generate low-dimensional features and trained different machine learning classification models based upon them. In 2021, Wang et al. [158] leveraged adversarial domain adaption (a transfer learning technique) to achieve high WFP accuracy with little encrypted data; Yin et al. [22] proposed a WFP attack that is able to identify websites in multi-tab environments, which means it can achieve usable accuracies regardless of the number of simultaneously opened web pages; Hoang et al. [159] found that even in the presence of domain name encryption technologies or content delivery network (CDN), WFP based on IP addresses is still feasible. They exploited the complex structure of most websites, which load resources from several domains besides their primary one, and further applied the generated domain fingerprints to conduct WFP at large.

D. Location Inference

Location inference is a widely studied topic by computer scientists. We have seen myriad works focusing on using social network information [160], [161], smartphone accelerometer [162], image content [163], etc. to infer users' locations. In the past decade, a few researchers began to use FGTA to conduct location inference. The location we discuss here can be either a geographical location or a contextual location, the later one means the type of location the user is sending packets from, such as an airport, a campus, or a residential building. This subsection examines inference approaches for these two types of locations.

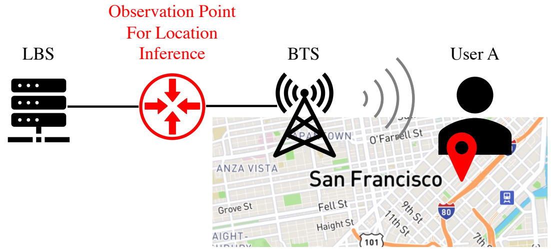


Fig. 10: Operation model for geographical location inference.

1) Contextual location inference: The intuition behind contextual location inference with FGTA is straightforward—users from different types of locations tend to generate different traffic because they need to use different web applications at different locations. Besides, different locations (e.g., campus, company, residential area) may process network traffic in different manners. Contextual location inference using FGTA aims to measure and analyze sets of network traffic and infer where these sets of traffic are coming from.

Back in 2009, Trestian et al. [164] conducted a detailed study on applications accessed by users at different locations. They demonstrated that users are more likely to show interest in a particular class of applications than others at certain locations, which is irrespective of the time of day. They indicated that we can further use the traffic generated by these applications to identify the type of locations (e.g., work versus home). In 2014, Das et al. [165], [166] collected around 100 GBs of real-world network traffic from more 1700 users at different types of locations (e.g., cafeteria/restaurant, university campus, airport/travel, etc.). By measuring and analyzing this dataset, Das et al. selected sets of features for packet-level, flow-level traffic and built a decision-tree-based classification model to predict contextual location with an overall accuracy of 87%. Later, a few similar works also demonstrated that mobile traffic from different cellular towers [167]–[169] tends to have different characteristics.

The drawback of contextual location inference is that it only works on a group of network traffic sending from many endpoints. It cannot infer a device's contextual location by only analyzing its own network traffic.

2) Geographical location inference: Purely using network traffic to infer a user's geographical location seems impossible. However, in 2015, Ateniese et al. [24] demonstrated that it is actually feasible under certain assumptions.

Nowadays, location-based applications (LBA), such as Facebook, Yelp, Google Map, etc., are widely used. These LBAs obtain user locations through location-based services (LBS). LBS providers usually use a base transceiver station (BTS) to locate a user and send real-time location information to the user. Ateniese et al. proposed an approach (illustrated in Figure 10) that simply monitors the traffic between the BTS and the LBS to identify user locations. They found that different locations will trigger LBS packets of different sizes. An adversary can potentially create a location knowledge base of different locations' packet sizes and their corresponding timestamps to conduct geographical location inference. Still, this approach has many limitations (e.g., low accuracy, difficult to build the location knowledge base at large, etc.). This work is more about demonstrating the feasibility of geographical location inference with FGTA than launching a full-fledged, ready-to-use approach.

E. Device/OS Identification

TA has been used to identify user devices or the OS running on the device for a long time. For example, Lippmann et al. [170] focused on extracting TCP or IP packet metadata to recognize different OSes in 2003. However, with the increase in the variety and complexity of user device and OS, simply identifying the device/OS type according to rules in packet header is no longer effective. Thus, researchers turn to use FGTA to investigate if specific traffic patterns can be correlated with some OSes or devices, which not only can recognize a few rough device/OS types, but also can pinpoint the device model or OS for various IoT and mobile devices. In this subsection, we introduce such FGTA approaches that deal with device/OS identification.

1) OS identification: Chen et al. [171] perform OS identification and detection of NAT and tethering (i.e., multiple devices sharing the Internet connection of a mobile device, which can lead to multiple OSes sharing a single IP address) by inspecting TCP/IP headers of packet traffic. They leverage a probability-based method by applying the Naïve Bayes classifier to effectively combine multiple features (e.g., TTL value, IP ID monotonicity, TCP timestamp, clock frequency, etc.), thereby fingerprinting and recognizing different OSes in different environments. Laštovička et al. [172] also proposed an OS identification method by constructing a decision tree with the TLS handshake, HTTP headers, and TCP/IP features. However, these approaches cannot distinguish between minor versions of the same OS. To tackle this problem, Ruffing et al. [173] identify different versions of smartphone OSes by using the frequency spectrum of packet timing from encrypted traffic. By identification through correlations of the feature-extracted spectra, the authors demonstrate that even a network traffic input of 30 seconds can be enough for high-accuracy identification results.

2) IoT device identification: Compared with OS identification, IoT device identification can be more challenging due to the complexity of their network environments and the devices' wide variety. Lopez-Martin et al. [174] extract a time-series feature vectors from network traffic, where each element of the time-series vector contains the features of a packet in the flow. They then proposed a classifier that is based on both a RNN model and a CNN model to separate heterogeneous IoT traffic using the features. Meidan et al. [175] collected and labeled network traffic from nine distinct IoT devices (e.g., baby monitor, motion sensor, printer, security camera, etc.), PCs, and smartphones. They then utilized a multi-stage machine-learning-based classifier to classify traffic of IoT devices in two phases. In the first stage, the classifier can distinguish traffic between IoT and non-IoT devices. In the second stage, the classifier can further identify traffic from different IoT devices. The authors demonstrate that their approach is able to classify IoT traffic with an accuracy of 99.281%.

However, these two researches do not consider complicated network environments (e.g., smart homes, enterprises, and cities) of IoT devices. Sivanathan et al. [176] addressed this challenge by developing a robust framework for IoT device traffic classification with a multi-stage machine-learning-based algorithm. The authors instrumented a smart environment with 28 different IoT devices that consist of spanning cameras, lights, plugs, motion sensors, appliances, and health-monitors. They then collected and synthesized network traffic traces from this infrastructure for a period of six months. By extracting statistical features such as activity cycles, port numbers, signaling patterns, and cipher suites from the traffic and using Naïve Bayes and random forest as the identification models, they are able to classify heterogeneous IoT devices with an accuracy over 99%. Yao et al. [177] further proposed an end-to-end IoT traffic classification method that eliminates the multi-stage classification for high accuracy and efficiency. It relies on a deep-learning-aided capsule network to construct an efficient classification mechanism that integrates feature extraction, feature selection, and classification model. One drawback of these approaches is that their evaluations are all based on closed-world datasets, which may not be able to precisely reflect their true efficacy in the real world.

F. Application Identification

Using the network traffic from a device to identify the applications that are running on the device, even in the presence of traffic encryption, is one of the most classic use cases of TA. Decades ago, people have investigated using traditional TA approaches to classify traffic from different applications. Before 2000, many researchers simply used traffic ports to identify some popular applications that have well-established ports (e.g., port 443 for HTTPS, port 110 for POP3). Port-based approaches fail for most emerging applications such as gaming, streaming, and messaging [178]. Later, Karagiannis et al. proposed BLINC [179], which not only looks at port-based features, but also inspects the host's social behavior and its community behavior to determine the applications. Bernaille et al. [180] observe the sizes of the first few packets of an SSL connection to identify the web application, which can achieve an accuracy of more than 85%. There are also many machine-learning-based traditional TA approaches [14], [181]–[183] that classify application traffic according to the traffic patterns.

However, application identification with traditional TA can hardly adapt to the current network environment and meet current needs due to several limitations:

- Traditional TA can only identify some high-level protocols (e.g., HTTP, HTTPS, SMTP, POP3, etc.) and a few frequently used applications that have obvious traffic patterns (e.g., MySQL, BitTorrent, MSN, etc.).
- Traditional TA-based application identifications only work in relatively simple network environments. For example, endpoints only consist of servers, clients, and peers; devices communicate without encrypted tunneling protocols (e.g., virtual private network (VPN)).

Nowadays, network environments are becoming far more complicated that before. Different types of nodes (e.g., smartphone, IoT, middlebox) may communicate through complicated network environments (e.g., VPN, network address translation (NAT), WiFi). Besides, millions of web applications are used on different platforms, with more complex communication mechanisms and much less regular traffic patterns. Therefore, people started to leverage FGTA in identifying specific applications among miscellaneous traffic from different types of devices. In this subsection, we introduce typical FGTA-based application identification approaches (Table VIII shows an overview).

1) Application identification for general-purpose devices: General-purpose devices, such as personal computers and servers, support the operation of countless web applications. Recently, FGTA-based application identification for general-purpose devices focuses on identifying more specific applications in more complicated network environments.

Chen et al. proposed Seq2img [184], an application traffic classification framework based on an online CNN model. Seq2img employs a data fusion method based on Reproducing Kernel Hilbert Space (RKHS) to convert flow sequences into images, which can fully capture the static and dynamic behaviors of different applications. Then, Seq2img utilizes a CNN model to recognize network traffic of popular applications, such as Facebook, Instagram, Wechat, etc.

Rezaei et al. [68] investigated using a few labeled, sampled packet-level datasets to train a comprehensive application identification model. They first pre-train a CNN-based model on a large unlabeled dataset, where the input is the time series features of a few sampled packets. Then, the learned weights are transferred to a new CNN model that is re-trained on a small labeled dataset. They demonstrated that this semi-supervised approach achieves almost the same accuracy as a fully-supervised method with a large labeled dataset. The proposed approach is able to identify applications like Google Drive, Google Doc, Google Search, Google Music, etc.

In 2020, Lotfollahi et al. [77] proposed an application identification method that can work in both VPN and non-VPN networks. After extracting features from packet headers, they used both CNN and stacked autoencoder (SAE) to train the classification models. Evaluation results show that this approach can achieve a recall score of 0.98 in application identification tasks.

2) Mobile application identification: With the raising of mobile network, mobile application identification becomes an emerging research topic in recent years. Unlike general-purpose devices, mobile devices are less regularized in port usage. In addition, a wide variety of mobile applications may utilize some common libraries in communication, generating similar network traffic patterns. Thus, mobile application identification can be more challenging.

Wang et al. [185] use random forest algorithm to analyze packet-level traffic in wireless networks. Their approach is able to detect the usage of 13 selected popular mobile applications on IOS platform, such as Snapchat, Tecent QQ, Mint, Tinder, YouTube, etc., with an accuracy of more than 87.23%. They demonstrate that by using the mobile applications the privacy of the user is more at risk compared to using online services through browsers on mobile devices.

TABLE VIII: Comparisons of selected FGTA approaches for application identification (○: not support; ◐: partially support; ●: support).

Category	Approach	Year	Target Application	Traffic Feature	Method	Real-World Evaluation
App Identification for General-Purpose Devices	Chen et al. [184]	2017	Instagram, Skype, Facebook, Wechat, Youtube, etc.	Images converted from flow sequences	RKHS-based data fusion and CNN	○
	Rezaei et al. [68]	2018	Google Drive, Youtube, Google Docs, Google Search, Google Music, etc.	Time series features extracted from sampled packets	Semi-supervised CNN	○
	Lotfollahi et al. [77]	2020	Vimeo, YouTube, VoipBuster, Spotify, Netflix, Hangouts, Facebook, etc. (with or without VPN)	Normalized features extracted from packet headers	CNN and SAE	○
Mobile App Identification	Wang et al. [185]	2015	Snapchat, Tecent QQ, Mint, Tinder, YouTube, etc.	Statistical features from packets (e.g., STD time, average size, STD size, etc.)	Random forest	◐
	Alan et al. [66]	2016	1595 applications on four different devices	Features from TCP/IP headers (e.g., packet size, timing, direction)	Jaccard's coefficient and Naïve Bayes	◐
	Taylor et al. [186]	2016	110 most popular applications in Google Play Store	Two sets of features from flow-level traffic-flow vector and statistical features	SVM and random forest	◐
	Aceto et al. [187]	2018	49 mobile applications (i.e., QQ, SayHi, eBay, 6Rooms, NetTalk, PureVPN, etc.)	Statistical features (e.g., packet length, percentiles, deviation, etc.) for incoming, outgoing, and bidirectional packets	A multi-classification (viz. fusion) model consists of Naïve Bayes, random forest, SVM, and decision tree	○
	Aceto et al. [76]	2019	Facebook, Facebook Messenger, and other 49 apps on both Android and IOS	Automatically-extracted features using neural networks	Multiple machine learning models (e.g., CNN, LSTM, MLP, etc.)	○
	Van et al. [17]	2020	More than 1M apps from three datasets.	Packet and flow-level features selected by adjusted mutual information	A semi-supervised fingerprinting with destination-based clustering, browser isolation, and pattern recognition	○
Decentralized App Identification	Shen et al. [64]	2019	Aragon, Bancor, Canwork, Chainy, Cryptopepes, Eth_town, Etheremon, etc.	57 features of packet lengths, 72 features of bursts, and 54 features of time series, fused by kernel functions	KNN, SVM, and random forest	◐
	Aiolli et al. [188]	2019	BTC.com. BitPay, Bread, Wirex, Copay, etc.	Vectors of statistical features about the packet length from traffic flow	SVM and random forest	◐

Many researchers also studied application identification on Android platform. Inspired by some WFP approaches (Section IV-C), Alan et al. [66] use Jaccard's coefficient and Naïve Bayes to analyze features (e.g., packet size, timing, direction) from TCP/IP headers to identify 1595 applications on four different devices. Taylor et al. [186] proposed AppScanner, a framework that can automatically fingerprint and identify Android applications from their encrypted network traffic. The authors extracted two sets of features (i.e., flow vector and statistical features) from flow-level network traffic and implemented this approach using both SVM and random forest algorithms. The evaluations show that AppScanner can identify the 110 most popular applications in Google Play Store with more than 99% accuracy. In the next year, Taylor et al. further extended AppScanner in a follow-up research [189]. They investigated how application fingerprints change over time, across different devices, and across different application versions.

Recently, many similar works (e.g., [17], [76], [187], [190]) have been proposed to enhance the efficacy, efficiency, and coverage of mobile application identifications.

3) Application identification on other platforms: A few researches have been focusing on identifying decentralized applications on blockchain systems. Shen et al. [64] proposed an encrypted traffic classification of decentralized applications (e.g., Cryptopepes, Matchpool, Lordless, etc.) on Ethereum with features like packet lengths, bursts, and time series. Aiolli et al. [188] focused on identifying user activities on Bitcoin wallet applications (e.g., BTC.com, Bitcoin Wallet, Coinbase, etc.). The authors used SVM and random forest models to conduct the identification.

We also studied the application identification approaches for IoT devices. However, as each IoT device is usually bundled with a IoT application, the identification of IoT application is equal to the identification of IoT devices in most cases. Therefore, we introduce these approaches in Section IV-E (IoT device identification).

TABLE IX: Comparisons of selected application usage inference approaches (○: not support; ◐: partially support; ●: support).

Category	Approach	Year	Analysis Object	Feature	Method	Real-World Evaluation
Messager/OSN	Coull et al. [191]	2014	Apple iMessage: language, control, read, start, stop, image, text, etc.	Payload length and the message length; a binary feature vector of packet length and direction pairs	Linear regression, Naïve Bayes, and rule lookup table	○
	Fu et al. [74]	2016	Wechat and WhatsApp: stream video call, news feed, location sharing, etc.	Discriminative features from the perspectives of packet length and time delay	Traffic segmentation with hierarchical clustering and thresholdingheuristics; HMM-based classifier.	◐
	Liu et al. [23]	2017	Facebook, Wechat, and WhatsApp: short video, video call, text, picture, etc.	A selected feature set extracted from traffic packet sequences by a Maximizing Inner activity similarity and Minimizing Different activity similarity measurements.	A recursive time continuity constrained K-means clustering algorithm for traffic flow segmentation and a random forest classifier for segmented traffic classification.	●
	Feng et al. [63]	2021	Facebook and Twitter: post, chat, read, etc.	Images converted from NetFlow records	Clustering-based traffic segmentation; CNN	●
Streaming Service	Wright et al. [192]	2008	Identify the phrases spoken within a call from a standard speech corpus.	The lengths of encrypted VoIP packets	HMM	○
	Schuster et al. [19]	2017	Identify the videos streamed by YouTube, Netflix, Amazon, and Vimeo.	Time series data of the following flow attributes: down/up/all bytes per second, down/up/all packet per second, and down/up/all average packet length.	Time-based burst; CNN	○
General-Purpose	Conti et al. [193]	2015	User activities in Gmail, Facebook, Twitter, Tumblr, Dropbox, etc.	Features from TCP/IP packet fields (e.g., IP address, port number, packet size, direction, and timing)	Dynamic time warping, random forest, and a hierarchical clustering algorithm called agglomerative	○
	Saltaformaggio et al. [194]	2016	User activities on Android and IOS platforms	Features extracted from IP packet headers, divided by behavior measurements (a small time window)	A K-means clustering model and an SVM model	◐
	Papadogiannaki et al. [16]	2018	User activities (e.g., voice call, video call, messaging, etc.) in popular Over-The-Top mobile applications (e.g., WhatsApp, Skype, Viber, etc.)	Customizable	A pattern language to identify application events, rule mining	●
Others	Yan et al. [195]	2018	Red packet transactions and fund transfers in Wechat	Overall statistics, packet length, number of TCP handshakes, inbound and outbound statistics	Threshold-based traffic segmentation, random forest	○
	Wang et al. [196]	2019	Classify specific actions (e.g., transfer payment, transfer receipt, QR code payment, etc.) on the mobile payment application, and then detect the detailed steps (e.g., click the button, receive the fund, open the red packet, etc.) within the action	Overall statistics of the packet length, range statistics of the packet length, flow statistics, incoming and outgoing statistics.	Threshold-based traffic segmentation, hierarchical identification with random forest, AdaBoost, GBDT, and XGBoost	○
	Jiang et al. [71]	2019	Application usage information (e.g., reading documents, surfing webs, editing documents, etc.) on remote desktop	Statistic features of flow burst	Threshold-based traffic segmentation, logistic regression, SVM, GBDT, random forest	○
	Wang et al. [75]	2020	Identify DApp (e.g., Superrare, Editional, John Orion Young, etc.) user behaviors (e.g., open DApps, open market, view detail, etc.)	Selected DApps features, behavior-sensitive features, and improved inter-arrival time series	Random forest, decision tree, and GBDT	○

G. Application Usage Inference

Application usage inference aims to analyze encrypted network traffic to identify certain application events, infer user behaviors, and measure specific service usage. It is one of the most challenging FGTA tasks, as it not only classifies the network traffic that is associated with different applications, device, or web pages, but also leverages the traffic patterns to recognize the application-layer activities that users conducted with the applications, devices, or web pages. Therefore, many application usage inference approaches may take extra steps (e.g., clustering, pre-filtering, etc.) to narrow down the scope before the final traffic classification. Besides, they need to perform traffic segmentation to locate different traffic bursts, where each burst represents a group of adjacent packets that support an application event.

In this subsection, we introduce representative application usage inference approaches, demonstrating their applicable scenarios and methodologies (Table IX shows a comparison).

1) Messager/Online social network usage inference: User activities on messaging or OSN applications are very private and sensitive. However, although being encrypted, a third party can still infer the rough messaging/OSN activities that users have performed only through content-agnostic network traffic data.

Back in 2009, Schneider et al. [197] investigated OSN usages from the perspective of network traffic for four different platforms—Facebook, LinkedIn, Hi5, and StudiVZ. The authors studied how users actually interact with OSNs by extracting clickstreams from passively monitored network traffic. They found that different OSN operations (e.g., login, open friend list, logout, select profile, etc.) will trigger statistically different network traffic. This research later lead many researchers to dig deeper into using the traffic differences to classify different user actions on OSNs. Coull et al. [191] analyzed the network traffic of encrypted messaging services such as Apple iMessage. The authors demonstrated that an eavesdropper can learn information about user actions (e.g., control, read, start, stop, image, and text), the language of messages, and even the length of those messages with greater than 96% accuracy simply by observing the sizes of encrypted packets. They used three algorithms to perform the inference—linear regression, Naïve Bayes, and rule lookup table. However, they only evaluated their approach in closed-world environments with a small dataset. Fu et al. [74] extended the inference to more messaging applications (i.e., Wechat and WhatsApp) and more activities (e.g., stream video call, news feed, location sharing, etc.). By segmenting Internet traffic into sessions with a number of dialogs, extracting discriminative features from the perspectives of packet length and time delay, and leveraging multiple machine learning models to conduct the classification, the proposed approach can achieve 96% and 97% accuracy in WeChat and WhatsApp respectively. Liu et al. [23] further extended the inference coverage to more OSN applications (e.g., Facebook, Wechat, and WhatsApp) and evaluated their approach in a real-world environment with real-time traffic data streaming. Real-world evaluation is essential to reveal the true performance and efficacy of application usage approaches, but many approaches were only evaluated through closed-world off-line cases, leaving the inference throughput and abilities to handle noise mysteries. Feng et al. [63], [198] developed and evaluated their OSN usage inference approach in a larger network environment—a campus network. Although their approach is mainly built for social bot detection, it can identify some commonly seen user activities (i.e., posting, reading, liking, etc.) on Twitter and Facebook.

2) Streaming service usage inference: There are a few works focusing on leveraging FGTA to extract behavioral information from network traffic of streaming service (e.g., VoIP, audio streaming, and video streaming). Researchers have demonstrated the feasibility of revealing voice information from encrypted VoIP conversations or identifying encrypted video streams [6].

Wright et al. [192] demonstrated that when the audio is encoded using variable bit rate codecs, the lengths of encrypted VoIP packets can be used to identify the phrases spoken within a call. By leveraging a HMM, the authors indicated that an eavesdropper can identify phrases from a standard speech corpus within encrypted calls with an average accuracy of 50%, and with accuracy greater than 90% for some phrases. Schuster et al. [19] demonstrated that many video streams are uniquely characterized by their burst patterns, and classifiers based on CNN models can accurately identify these patterns given very coarse network measurements. The authors only extracted features from flow attributes, such as inbound/outbound bytes per second, inbound/outbound packet per second, and inbound/outbound average packet length. They have examined this approach on Netflix, YouTube, Amazon, and Vimeo.

3) General-purpose application usage inference: The approaches discussed in this subsection aim at inferring all types of application-layer events rather than only recognizing certain event categories.

Conti et al. [193], [199] analyzed encrypted mobile traffic to infer user actions on Android devices, such as email exchange, posting a photo online, publishing a tweet, etc. They extracted features from TCP/IP packet fields (e.g., IP address, port number, packet size, direction, and timing) and use a random forest to perform the inference. They trained and evaluated their approach using a dataset that is associated with several Android applications with diverse functionalities, such as Gmail, Facebook, Twitter, Tumblr and Dropbox. The evaluation results demonstrate that it can achieve more than 95% of accuracy and precision for most of the actions within the dataset. However, this approach was not evaluated in the real-world environments. In 2016, Saltaformaggio et al. [194] proposed NetScope, a framework that can perform robust inferences of user activities for both Android and IOS devices by only inspecting IP packet headers. NetScope leverages a K-means model and an SVM model to learn and detect network traffic generated by different application behaviors. By testing the approach in a lab environment, the authors demonstrated that despite the widespread use of fully encrypted communication, NetScope can distinguish subtle traffic behavioral differences between user activities (e.g., Instagram browse versus post, Yelp browse versus search, Facebook feed versus post, etc.). Papadogiannaki et al. [16] further pushed application usage inference to a much larger scale. They proposed OTTer, a highly scalable engine that identifies fine-grained user actions (e.g., voice call, video call, messaging, etc.) in popular Over-The-Top mobile applications, such as WhatsApp, Skype, Viber, and Facebook Messenger with encrypted network traffic connections. By evaluating OTTer is a real-world test bed, the authors demonstrated that it can operate at traffic loads with an average of 109 Gbps.

4) Others: There are a few application usage inference approaches tackling different problems. For instance, Yan et al. [195] segmented the network traffic into several bursts and trained a random forest model to identify red packet transactions and fund transfers in Wechat; Wang et al. [196] proposed an approach to identify the mobile payment applications from traffic data, then classify specific actions (e.g., transfer payment, transfer receipt, QR code payment, etc.) on the mobile payment application, and finally, detect the detailed steps (e.g., click the button, receive the fund, open the red packet, etc.) within the action; Jiang et al. [71] studied encrypted remote desktop traffic and found that an eavesdropper can reveal application usage information (e.g., reading documents, surfing webs, editing documents, etc.)

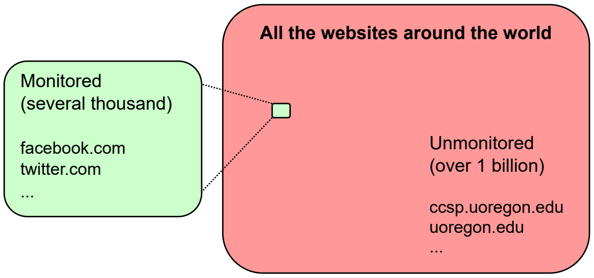


Fig. 11: Training data coverage for WFP.

due to side-channel privacy leakage. Wang et al. [75] aimed at identifying DApp (e.g., Superrare, Editional, John Orion Young, etc.) user behaviors (e.g., open DApps, open market, view detail, etc.) on Ethereum by using random forest, decision tree, and gradient boosting decision tree (GBDT).

V. Limitations

Although it seems effective in inferring different high-level, fine-grained behaviors, FGTA still has many limitations. In fact, FGTA approaches are far from what they have promised in the real world, and their efficacies depend on many conditions. In this section, we discuss the limitations of FGTA.

A. Coverage of train data

As most FGTA approaches are based on machine learning algorithms or prior knowledge about specific traffic, the efficacy of such approaches is highly dependent on the coverage of training datasets or rules learned beforehand. Unfortunately, existing datasets or rules can only represent a small fraction of real-world scenarios. It is actually impossible to collect a dataset to cover all possible scenarios. Take the WFP attack discussed in Section IV-C as an example, as shown in Figure 11, state-of-the-art datasets from public repositories can only cover less than 0.001% of all websites around the world. FGTA approaches built upon such datasets then have little effect in practice. Furthermore, network traffic of websites, applications, or OSes is dynamic. For instance, the layouts of Facebook websites have been changed for several times since its birth, and so has the network traffic associated with Facebook. Therefore, a FGTA approach that worked before may no longer be effective, if we do not update its classification model with the latest training datasets.

B. Uncertainties in real-world environments

As can be seen from our previous discussion (Section III-D and Section IV), many approaches were only evaluated in closed-world environments, which means they were only tested with a small amount of labeled traffic, with a little noise or without noise. Such closed-world evaluations cannot objectively reveal the efficacy of proposed approaches in the real world. Network traffic in real-world environments can be very quite different from traffic in laboratory environments:

- Real-world network configurations can be complicated, with traffic going through NATs, Wi-Fi connections, or special middle boxes. All these factors can significantly change the original traffic characteristics.
- Edge users have different habits of using web applications. Some may send traffic with VPN, Shadowsocks, or Tor. Although many FGTA approaches claim to be effective even with traffic tunneling techniques, many researchers found their efficacy will actually be reduced under such circumstances [200].
- The ratio of different network traffic in the real world is different from that in the laboratory environment, making accuracies obtained from closed-world evaluations hardly representative.

Therefore, real-world evaluations or large-scale pilot studies are essential for developing and polishing a usable FGTA approach.

C. False alarms

FGTA aims to identify specific types of user activities from network traffic. Usually, the analysis object only occupies a very tiny proportion of the whole traffic (e.g., less than 0.01%). Thus, a very small false positive rate can be amplified in deployment, making the proposed FGTA approach hardly usable.

D. Integrity of network traffic

As discussed in Section II, the integrity of the network traffic collected in the real-world cannot be guaranteed. The traffic flow can be asymmetric or highly sampled, which will certainly reduce the efficacy of existing FGTA approaches. However, FGTA with incomplete network traffic was not widely discussed in existing papers.

VI. Countermeasure

Besides the aforementioned limitations of FGTA approaches, Internet users can also adopt various tricks or methods to escape inference. From the perspective of illegitimate users, these countermeasures can make them stay stealthy and avoid being discovered when conducting malicious activities. On the other hand, legitimate users can also leverage these countermeasures to perturb FGTA, thereby protecting their privacy. This section investigates FGTA countermeasures, comparing their efficacy and use cases.

Naïve countermeasures send individual or aggregated traffic through encrypted channels to escape the inferences of traditional TA approaches, such as VPN, Shadowsocks, and Tor. However, these approaches are proven to be vulnerable to many FGTA approaches [206]–[209]. Therefore, people began to modify the features of traffic flows to perturb FGTA approaches' classification models. Such perturbations can be conducted from either network layer or application layer [145]. Table X shows a comparison of some well-known countermeasure approaches.

TABLE X: Comparisons of selected well-known FGTA countermeasures (*None*: 0; *Low*: 0-30%; *Medium*: 30%-60%; *High*: more than 60%).

Category	Approach	Usage Scenarios	Time Overhead	Bandwidth Overhead	Additional Requirements
Network-Layer	Pinheiro et al. [201]	All web applications	None	Medium	Middlebox and SDN controller
	FRONT and GLUE [202]	Tor	None	Low	None
	Traffic morphing [203]	All web applications	None	Low	Knowledge about other traffic classes
	Walkie-Talkie [152]	Web browsing	Medium	Low	Knowledge about some web traffic
	WTF-PAD [153]	Web browsing	None	Low	None
	Liberatore et al. [133]	All web applications	None	High	None
	BuFLO [147]	Web browsing	High	High	Network Transfer with fixed rates
	TrafficSliver [145] (network-layer mode)	Tor	None	None	Multiple entry ORs in Tor network
Application-Layer	HTTPOS [204]	Web browsing	None	Low	None
	TrafficSliver [145] (L7 mode)	Tor	None	None	Multiple entry ORs in Tor network
	LLaMA and ALPaCA [205]	Tor	Server-side: Medium; Client-side: Low	Server-side: Medium; Client-side: Low	None

A. Network-layer Countermeasures

Network-layer FGTA countermeasures directly modifying the network traffic by adding padding packets, changing packet bytes, or delaying existing packets, thereby obfuscating specific features that FGTA approaches rely on, making the current traffic look like other activities', or regularizing the traffic patterns of different applications [202]. Such approaches usually come with some side effects. They might increase the overheads of the network system, including time overhead, bandwidth overhead, and potentially computational overhead.

Among all the network-layer countermeasures, traffic obfuscation is the most classic approach. Back in 2006, Liberatore et al. [133] leveraged per-packet padding (i.e., increasing the bytes of packets) in an attempt to defeat host profiling system. They found that per-packet padding is reasonably effective, which can lower predictive accuracy to less than 8% with a cost of increasing traffic volume by 145%. However, per-packet padding cannot defend against many WFP attacks [137], [147] because this approach still preserves some key traffic features that can help classify the traffic. To fix the drawbacks, WTF-PAD [153] extends per-packet padding to link-based padding to modify more traffic features. It detects large time gaps between packets and covers them by adding dummy packets. Further, to obscure traffic bursts, it also adds delays between packets to make them statistically different. Due to its low computational overhead and time overhead, WFP-PAD has been used in many real-world FGTA defense systems [210], [211]. Still, WTF-PAD leaks a portion of information in transmission and can be broken by some FGTA approaches [139], [212]. Gong et al. [202] proposed FRONT and GLUE. FRONT focuses on obfuscating the trace front with dummy packets. It also randomizes the number and distribution of dummy packets to impede the attacker's inferring process. GLUE adds dummy packets between separate traces so that they appear to the attacker as a long consecutive trace, making the attacker unable to find the start or end points.

Compared with traffic obfuscation that freely modifies traffic features, traffic confusion mimic other groups of traffic to let FGTA approaches generate wrong outputs, which is sometimes more effective, especially when defending against WFP attacks. Wright et al. proposed traffic morphing [203]. It can thwart statistical TA approaches by morphing one class of traffic to look like another class using convex optimizations. Although it cannot defend against some types of FGTA approaches [137], [147], this approach inspired many subsequent countermeasure approaches. For example, Glove [213] first leverages a clustering algorithm to group web pages with similar traffic, and then inserts only a small amount of dummy traffic to hide the web page traffic in a close group; Supersequence [136] also clusters network traffic traces of different web pages and extracts the shortest common supersequence to cover current web traffic; Walkie-Talkie [152] modifies the browser to communicate in half-duplex mode (buffer traffic and send in bursts) rather than the usual full-duplex mode (immediately send available data). By combining with dummy packets, Walkie-Talkie can modify the traffic of monitored sensitive pages and benign non-sensitive pages, so that each page's packet sequences are exactly the same (each packet has the same timing, length, direction and sequence number). However, a traffic-confusion-based approach requires a priori knowledge about popular web pages' network traffic. It cannot tackle traffic of dynamic content or unpredictable activities. Moreover, such approaches can lead to noticeable computational overhead.

Another obfuscation direction is to regularize the network traffic, making different groups of traffic have relatively uniform patterns. For instance, Buffered Fixed-Length Obfuscation (BuFLO) [147] obfuscates page transmissions by sending packets of a fixed size at a fixed interval and using dummy packets to both fill in and potentially extend the transmission. Thus, the traffic generated by different websites has a similar continuous traffic flow. However, BuFLO can cause very high time and bandwidth overhead, sometimes can even bring congestion problems to the network [73]. To alleviate the problem, Congestion-Sensitive BuFLO (CS-BuFLO) [214] was proposed to vary the packet transmission rate. Tamaraw [151] achieves a better security/bandwidth trade-off by using smaller fixed packet sizes and treating incoming and outgoing packets differently to avoid unnecessary padding and dummy traffic. DynaFlow [215] morphs packets into fixed bursts, dynamically changes packet inter-arrival times to generate constant traffic flows, and pads the number of bursts. Theoretically, DynaFlow leads to less network overhead compared with BuFLO, CS-BuFLO, and Tamaraw.

The recent development of FGTA countermeasures mainly focuses on two aspects:

1) The countermeasure should lead to nearly zero overhead to both the data plane and the endpoints.
2) The countermeasure should be applicable to various web applications (e.g., web page visiting, video streaming, VoIP, etc.) and scenarios.

For instance, Henri et al. [216] split traffic exchanged between the user and Tor nodes over two different, unrelated network connections (e.g., DSL, Wi-Fi, or cellular networks) to protect against FGTA by a malicious ISP; TrafficSliver [145] limits the data a single observation point can observe and distorts repeatable traffic patterns exploited by FGTAs with user-controlled splitting of traffic over multiple Tor entry nodes. TrafficSliver also offers an application-layer solution, which will be discussed in Section VI-B; Wang [217] points out that an attacker may only need to successfully identify a single web page (which they define as the one-page setting) in reality, and a WFP countermeasure must still thwart that attempt. Based on this assumption, Wang fortifies WFP countermeasures by exploring randomness and regularization options for several existing countermeasures. To protect IoT networks, Pinheiro et al. [201] implement a middlebox to modify the outbound and inbound traffic's packet size. They also leverage an SDN application to obtain information of network traffic from both sides (source and destination) to manage the size-based padding mechanism.

B. Application-layer Countermeasures

Unlike network-layer countermeasures that directly modify network traffic to cover user activities, application-layer countermeasures use dummy applications to generate unnecessary traffic, thereby indirectly perturbing FGTA approaches. However, most application-layer countermeasures are limited in covering traffic of web page being visited.

Panchenko et al. [135] proposed a browser plug-in that adds traffic noise by loading another random web page in parallel. However, it may fail to defend against some WFP attacks if users lower the page loading frequency to decrease the bandwidth overhead [136]. Another Tor-based countermeasures approach [218] randomizes the order of requests for embedded website content and the pipeline size (i.e., the number of requests processed in parallel) to perturb WFPs. Cherubin et al. [205] propose LLaMA and ALPaCA, defenses for client side and server side. LLaMA reorders outgoing HTTP requests by randomly delaying them and adding dummy HTTP requests. On the server side, ALPaCA conducts traffic morphing by padding web objects of a page and inserting invisible dummy web objects. The three methods above only work in Tor environments.

HTTP Obfuscation (HTTPOS) [204] is countermeasure that can be used in environments other than Tor. By modifying HTTP requests and basic TCP features, it manipulates four fundamental network flow features, including packet size, web object size, flow size, and timing of packets. It can also modify and reorder HTTP headers and insert dummy HTTP requests. Another general countermeasure is TrafficSliver's application-layer defense [145]. This approach is on the client side. By sending single HTTP requests for different web objects over distinct Tor entry nodes, this application-layer defense can reduce the detection rate of WFP classifiers by almost 50 percentage points.

VII. Future Research Direction

Although FGTA has been developed for decades, there still exists room for further development, enhancement, and exploration. In this section, we discuss avenues for future research according to our observations about recent research trends, existing literature, industry deployments, and major problems to be solved in this domain.

A. Improvement of Analysis Efficacy and Coverage

FGTA has been used in many different subfields of computer network, including attack detection, traffic measurement, side-channel attack, and network management, etc. Researchers have constructed myriad analysis models and collected plenty of datasets specifically for different categories of tasks. But there are still many use cases or scenarios that have not been covered by existing approaches. For example, with the raising of Unmanned Aerial Vehicle (UAV), FGTA can be potentially refined for UAV anomaly detection [219]; FGTA can also be adopted to monitor and safeguard network traffic of automatic vehicles; with the emergence of new forms of applications, attacks, and communication protocols, existing FGTA approaches may not be able to handle today's traffic. Therefore, researchers can gather more updated traffic datasets to enhance the coverage of existing FGTA approaches, so that they can be used in more types of tasks and scenarios.

In addition, the efficacy of many current FGTA approaches are not ideal for real-world deployments. Depending on the observation points, FGTA approaches may easily see millions of traffic flows over a short time period in the real world. Under such circumstances, an FGTA approach could generate large numbers of false positives or false negatives, even if it achieves more than 95% accuracies in closed-world evaluations. Thus, increasing the efficacy of FGTA is a timeless topic for researchers and developers.

B. Evaluation Enhancement

As we elaborate in Section V, current closed-world evaluation methods are far away from revealing an FGTA approach's real capability and many open-world evaluations are not very standardized and effective. It is therefore suitable to propose a new, operable, and effective evaluation paradigm for FGTA. Such an evaluation paradigm should contain a testing dataset similar to a real-world test case in terms of volume, environment, and data distribution. Simultaneously, the dataset should have comprehensive labels for almost all traffic flows, not only for analysis targets. This can be achieved by either constructing a large scale sandbox to simulate and collect all types of traffic from a white box view, or collect a large-scale, real-world traffic dataset and carefully label it using knowledge of endpoints from all perspectives. Besides, the testing data portion that is visible to the observation point should be consistent with the real-world deployment conditions.

C. Dealing with Complex Network Environments

In real-world deployments, the network environments and configurations can be different from researchers' assumptions. The following factors were not widely discussed in previous papers, but can be common for network service providers.

- Many observation points can only see asymmetric network traffic, which can challenge most FGTA approaches.
- Some networks are composed of multiple subnets, including but not limited to wireless network, optical network, or radio frequency network. Traffic flows collected from such a network can have different delays and congestion control mechanisms. Tackling this type of traffic can be challenging.
- Due to deployments of modern traffic engineering approaches, traffic captured from some observation points is dynamic, posing difficulties to many FGTA methods.

We believe designing and implementing new FGTA approaches that can work under these circumstances are directions worthy of future research.

D. Integrating FGTA into Other Analytical Systems

Information contained in network traffic is essentially limited. Even though FGTA can already reveal considerable amount of information, the detailed behavior models of endpoints are still hidden behind the curtain. To more comprehensively investigate the network situation, researchers can try to combine FGTA with information from other dimensions (e.g., application-layer activities, server specifics, hardware conditions, etc.), which can provide a better situational awareness. So far, there are a few researches that combine TA with information from other layers for more accurate attack/anomaly detection and timely threat response (e.g., [220]–[222]). Researchers can push this idea forward by further integrating FGTA into this idea.

Furthermore, cyber threat intelligence (CTI) [223], allowing entities to share attack/anomaly information with trusted partners and peers, is becoming a powerful tool to quickly and accurately tackle intractable attacks. By embedding results from FGTA into CTI systems, participating entities can raise awareness of the current situation, thereby more quickly responding to incoming attacks. Designing attack defense systems with both FGTA and CTI is thus a promising research direction.

VIII. Related Work

In this section, we introduce related work on TA and compare our paper with them. Table XI summarizes the most related and representative ones.

Our work differs from existing works in the following aspects:

- we have a clear and focused survey topic: the whole paper focuses on FGTA, which aims to analyze network traffic to deduce information related to high-layer activities, fine-grained user behaviors, or application-layer message content;
- to investigate FGTA comprehensively, our paper utilizes multiple methodologies, including literature survey, summarization, and taxonomization, to cover different subjects, such as traffic capture, application identification, website fingerprinting, countermeasures, etc. Most existing related survey papers only cover a subset of the aforementioned topics;
- although studied for many years, TA is still iterating rapidly and continuously, especially for FGTA. Compared with other earlier literature, this paper sorts out and examine the most recent development of FGTA at the time of writing this paper.

In the early development of TA (i.e., before 2010), the survey papers in this field mainly focus on traditional TA [232]–[234], including protocol-level traffic classification, TA approaches based on deep packet inspection (DPI), distinguishing server or peer-to-peer nodes from clients, and coarse grained application identifications.

Later, due to increasingly diverse web-applications and widespread use of traffic encryption, there is an increasing need for more sophisticated TA approaches to monitor and analyze the modern networks. Meanwhile, the evolution of classification algorithms and easy access to big data also effectively stimulate the development of TA. Therefore, survey papers began to examine works that leverage big data [5] or machine learning [7] to tackle TA.

On the other hand, the network is also becoming more and more specialized, which has spawned many TA approaches with specific design goals. To track such a trend, many of the recent survey papers only survey a certain type of TA approaches, such as TA for IoT devices [225], [226], encrypted TA [6], [15], TA for mobile devices [227], [228], etc. Similar to these papers, our work focuses on a new and specific topic—FGTA, which has not been systematically studied before.

IX. Conclusion

With the increasing complexity of network transmission technology, FGTA is becoming a crucial tool to gain a finer granularity of visibility over the network. From the perspective of attackers, it can analyze the content-agnostic metadata and statistical information of network traffic to infer the website visited by users, estimate locations of traffic sender, or decode the video content streamed in the link. As for the network administrators, FGTA can be used to detect application-layer threats even with layer 3 or layer 4 data, investigate quality of experience without collect sensitive user data, or perform fine-grained traffic measurement to better configure the network.

In this paper, we analyze literature that deal with FGTA to help researchers and developers learn the latest developments in this area. After comparing different FGTA approaches by

TABLE XI: Overview of related literature (○: not include; ◐: partially include; ●: include).

Ref.	Year	Summary	Focus	Subjects covered			
				General TA	FGTA	Traffic capture	Counter-measure
This paper	2022	A survey of FGTA, which aims to analyze network traffic to deduce information related to high-layer activities, fine-grained user behaviors, or application-layer message content.	FGTA	◐	●	●	●
[15]	2022	The most recent survey on achievements in machine learning-powered encrypted traffic analysis, including the workflow, feature extraction, and algorithms.	Machine-learning-based encrypted TA	◐	◐	◐	◐
[224]	2022	A survey consists of an analysis of IoT traffic data acquisition approaches, a classification of public datasets, a literature evaluation of IoT traffic processing, and a comparison of ML approaches for IoT device classification.	Machine-learning-based IoT TA	○	◐	●	○
[6]	2021	A survey of literature that deals with network traffic analysis and inspection after the ascent of encryption in communication channels.	Encrypted TA	◐	◐	○	●
[225]	2021	An extensive analysis of the communications channels of 32 IoT consumer devices, including traffic measurement and modelling.	IoT TA	◐	◐	◐	○
[226]	2020	This survey looks at the emerging trends of network traffic classification in IoT and the utilization of traffic classification in its applications. It also compares the legacy of traffic classification methods.	IoT TA	●	◐	◐	○
[5]	2019	This survey mainly focuses on approaches and technologies to manage the big traffic data, additionally briefly discussing big data analytics (e.g., machine learning) for the sake of TA.	General TA	●	◐	●	○
[227]	2018	A review of works that contributed to the network traffic analysis targeting mobile devices, including a systematic classification of such works according to their goal, traffic capture point, and targeted platforms.	Mobile device TA	◐	◐	◐	●
[7]	2018	A systematic review based on the steps to achieve traffic classification by using machine learning techniques, including their workflow, feature extraction, deployment, etc.	Machine-learning-based TA	◐	◐	○	○
[228]	2016	An examination of the literature on analyses of mobile traffic collected by operators within their network infrastructure.	Mobile device TA	◐	◐	●	○
[229]	2015	A survey of approaches for classification and analysis of encrypted traffic, including widespread encryption protocols and payload and feature-based classification approaches.	Encrypted TA	◐	◐	○	○
[230]	2014	A survey in which a complete and thorough analysis of the most important opensource deep packet inspection modules is performed.	Payload-based TA	◐	○	◐	○
[231]	2013	A survey of peer-to-peer traffic detection and classification, with an extended review of the related literature.	P2P TA	◐	○	○	○
[232]	2009	A survey explains the main techniques and problems known in the field of IP traffic analysis and focuses on application detection.	General TA	●	○	●	○
[233]	2009	A report attempts to provide an overview of some of the widely used network traffic models, highlighting the core features of the model and traffic characteristics they capture best.	General TA	◐	○	○	○

their use cases, we found that most existing approaches are based on machine learning or high-dimensional clustering. They are effective in capturing the subtle differences between network traffic generated by different activities. However, many FGTA approaches still come with limitations related to training data coverage, false positive rates, and real-world usability. In addition, edge users of the network can adopt a variety of countermeasures to defend against FGTA, with some overheads regarding network bandwidth and delay. Researchers can further study this domain to increase the coverage of FGTA or make the approaches more practical in complex real-world network environments.

ACRONYMS

ABR	Adaptive Bitrate
ACC	accuracy
AUC	area under the curve
BTS	base transceiver station
CDN	content delivery network
CNN	convolutional neural network
CTI	cyber threat intelligence
DNN	deep neural network
DPI	deep packet inspection
F1	F1 score
FDR	false discovery rate
FGTA	fine-grained traffic analysis
FNR	false negative rate
FOR	false omission rate
FPR	false positive rate
GBDT	gradient boosting decision tree
GDPR	General Data Protection Regulation
HMM	hidden Markov model
IoT	Internet of things
KNN	k-nearest neighbor
LBA	location-based applications
LBS	location-based services

LSTM	long short-term memory
MLP	multi-layer perceptron
NAT	network address translation
NDAE	nonsymmetric deep autoencoder
NPV	negative predictive value
OS	operating system
OSN	online social network
PCA	principal component analysis
PII	personally identifiable information
PPV	positive predictive value
QoE	quality of experience
QoS	quality of service
QUIC	Quick UDP Internet Connection
RKHS	Reproducing Kernel Hilbert Space
RNN	recurrent neural network
ROC	receiver operating characteristic
SAE	stacked autoencoder
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
TA	traffic analysis
TNR	true negative rate
TPR	true positive rate
TTL	time to live
UAV	Unmanned Aerial Vehicle
VPN	virtual private network
WFP	website fingerprinting

ACKNOWLEDGMENTS

The authors would like to thank Zhangxiang Hu, Lei Jiao, Yingjiu Li, and Thanh Nguyen from the University of Oregon for their constructive suggestions on this work.

REFERENCES

[1] C. A. Sunshine, *Computer network architectures and protocols*. Springer Science & Business Media, 2013.

[2] Wikipedia, "Internet protocol suite," https://en.wikipedia.org/wiki/Internet_protocol_suite, date of visit: 2021-10-05.

[3] J.-F. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *Designing Privacy Enhancing Technologies*. Springer, 2001, pp. 10–29.

[4] C. So-In, "A survey of network traffic monitoring and analysis tools," *Cse 576m computer system analysis project, Washington University in St. Louis*, 2009.

[5] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, and P. Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 800–813, 2019.

[6] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[7] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2018.

[8] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[9] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of the 2008 ACM CoNEXT conference*, 2008, pp. 1–12.

[10] A. Sang and S.-q. Li, "A predictability analysis of network traffic," *Computer networks*, vol. 39, no. 4, pp. 329–345, 2002.

[11] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, "Structural analysis of network traffic flows," in *Proceedings of the joint international conference on Measurement and modeling of computer systems*, 2004, pp. 61–72.

[12] J. Mirkovic, Y. Feng, and J. Li, "Measuring changes in regional network traffic due to covid-19 stay-at-home measures," *arXiv preprint arXiv:2203.00742*, 2022.

[13] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 71–82.

[14] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, 2006, pp. 281–286.

[15] M. Shen, K. Ye, X. Liu, L. Zhu, J. Kang, S. Yu, Q. Li, and K. Xu, "Machine learning-powered encrypted network traffic analysis: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2022.

[16] E. Papadogiannaki, C. Halevidis, P. Akritidis, and L. Koromilas, "Otter: A scalable high-resolution encrypted traffic identification engine," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 315–334.

[17] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Network and Distributed System Security Symposium (NDSS)*, vol. 27, 2020.

[18] Y. Feng, J. Li, L. Jiao, and X. Wu, "BotFlowMon: Learning-based, content-agnostic identification of social bot traffic flows," in *2019 IEEE Conference on Communications and Network Security (CNS)*, June 2019, pp. 169–177.

[19] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1357–1374.

[20] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*, 2017, pp. 1723–1732.

[21] S. Xu, S. Sen, and Z. M. Mao, "Csi: Inferring mobile abr video adaptation behavior under https and quic," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

[22] Q. Yin, Z. Liu, Q. Li, T. Wang, Q. Wang, C. Shen, and Y. Xu, "Automated multi-tab website fingerprinting attack," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[23] J. Liu, Y. Fu, J. Ming, Y. Ren, L. Sun, and H. Xiong, "Effective and real-time in-app activity analysis in encrypted internet traffic streams," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 335–344.

[24] G. Ateniese, B. Hitaj, L. V. Mancini, N. V. Verde, and A. Villani, "No place to hide that bytes won't reveal: Sniffing location-based encrypted traffic to track a user's position," in *International Conference on Network and System Security*. Springer, 2015, pp. 46–59.

[25] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, "On routing asymmetry in the internet," in *GLOBECOM'05. IEEE Global Telecommunications Conference, 2005.*, vol. 2. IEEE, 2005, pp. 6–pp.

[26] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.

[27] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[28] B. Trammell and E. Boschi, "An introduction to ip flow information export (ipfix)," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 89–95, 2011.

[29] "Solaris snoop packet sniffer," http://www.softpanorama.org/Net/Sniffers/snoop.shtml, accessed: 2022-02-10.

[30] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier, 2006.

[31] "Data sheet of sniffer infinistream," http://mavin.com/pictures/InfiniStream.pdf, accessed: 2022-02-10.

[32] M. Bagnulo, P. Matthews, and I. van Beijnum, "Stateful nat64: Network address and protocol translation from ipv6 clients to ipv4 servers," Tech. Rep., 2011.

[33] N. Brownlee, C. Mills, and G. Ruth, "Traffic flow measurement: Architecture," RFC 2722, Tech. Rep., 1999.

[34] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering, "Ipv6 flow label specification," *RFC3697*, 2004.

[35] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for ip flow information export (ipfix)," RFC 3917 (informational), Tech. Rep., 2004.

[36] D. W. McRobb, "Cflowd design," *CAIDA, Sept*, 1998.

[37] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[38] J. Lee, "Calculating netflow volume," https://www.plixer.com/blog/calculating-netflow-volume/, posted: 2013-03-26.

[39] B.-Y. Choi and S. Bhattacharyya, "On the accuracy and overhead of cisco sampled netflow," in *Proceedings of ACM SIGMETRICS Workshop on Large Scale Network Inference (LSNI)*, 2005, pp. 1–6.

[40] C. Hare, "Simple network management protocol (snmp)." 2011.

[41] B. Claise, B. Trammell, and P. Aitken, "Specification of the ip flow information export (ipfix) protocol for the exchange of flow information," pp. 2070–1721, 2013.

[42] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes, "Cisco systems netflow services export version 9," 2004.

[43] "Netflow v5 formats," https://www.ibm.com/docs/en/npi/1.3.0?topic=versions-netflow-v5-formats, accessed: 2022-03-14.

[44] "Argus project," https://openargus.org/, date of visit: 2021-11-25.

[45] P. Phaal, S. Panchen, and N. McKee, "Rfc3176: Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks," 2001.

[46] "Tcpdump & libpcap," https://www.tcpdump.org/, accessed: 2022-02-10.

[47] ntop, "Pf_ring documentation," https://www.ntop.org/guides/pf_ring/, accessed: 2022-02-12.

[48] L. Rizzo, "netmap: a novel framework for fast packet i/o," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 101–112.

[49] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca, "Planck: Millisecond-scale monitoring and control for commodity networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 407–418, 2014.

[50] J. Svoboda, I. Ghafir, V. Prenosil *et al.*, "Network monitoring approaches: An overview," *Int J Adv Comput Netw Secur*, vol. 5, no. 2, pp. 88–93, 2015.

[51] L.-M. Wang, T. Miskell, J. Morgan, and E. Verplanke, "Design of a real-time traffic mirroring system," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 793–796.

[52] "ProfiShark Network TAPs," https://www.profitap.com/profishark-network-taps/, 2019, accessed: 2022-02-11.

[53] Wikipedia, "Argus – audit record generation and utilization system," https://en.wikipedia.org/wiki/Argus_%E2%80%93_Audit_Record_Generation_and_Utilization_System, accessed: 2022-02-11.

[54] Í. Cunha, F. Silveira, R. Oliveira, R. Teixeira, and C. Diot, "Uncovering artifacts of flow measurement tools," in *International Conference on Passive and Active Network Measurement*. Springer, 2009, pp. 187–196.

[55] Huawei, "Configuration guide - network management and monitoring," https://support.huawei.com/enterprise/en/doc/EDOC1000178174/986bf11e/overview-of-netstream, accessed: 2022-02-11.

[56] S. Waldbusser, R. Cole, C. Kalbfleisch, and D. Romascanu, "Introduction to the remote monitoring (rmon) family of mib modules," *RFC3577, Network Working Group*, 2003.

[57] T. Høiland-Jørgensen, J. D. Brouer, D. Borkmann, J. Fastabend, T. Herbert, D. Ahern, and D. Miller, "The express data path: Fast programmable packet processing in the operating system kernel," in *Proceedings of the 14th international conference on emerging networking experiments and technologies*, 2018, pp. 54–66.

[58] M. S. Brunella, G. Belocchi, M. Bonola, S. Pontarelli, G. Siracusano, G. Bianchi, A. Cammarano, A. Palumbo, L. Petrucci, and R. Bifulco, "{hXDP}: Efficient software packet processing on {FPGA}{NICs}," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 973–990.

[59] Y. Zhou, C. Sun, H. H. Liu, R. Miao, S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi *et al.*, "Flow event telemetry on programmable data plane," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 76–89.

[60] D. Comer, "Ubiquitous b-tree," *ACM Computing Surveys (CSUR)*, vol. 11, no. 2, pp. 121–137, 1979.

[61] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.

[62] Y. Feng, J. Luo, C. Ma, T. Li, and L. Hui, "I can still observe you: Flow-level behavior fingerprinting for online social network," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 6427–6432.

[63] Y. Feng, J. Li, L. Jiao, and X. Wu, "Towards learning-based, content-agnostic detection of social bot traffic," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2149–2163, 2021.

[64] M. Shen, J. Zhang, L. Zhu, K. Xu, X. Du, and Y. Liu, "Encrypted traffic classification of decentralized applications on ethereum using feature fusion," in *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*. IEEE, 2019, pp. 1–10.

[65] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.

[66] H. F. Alan and J. Kaur, "Can android applications be identified using only tcp/ip headers of their launch time traffic?" in *Proceedings of the 9th ACM conference on security & privacy in wireless and mobile networks*, 2016, pp. 61–66.

[67] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 international conference on wireless networks and mobile communications (WINCOM)*. IEEE, 2016, pp. 258–263.

[68] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets," *arXiv preprint arXiv:1812.09761*, 2018.

[69] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. Claffy, "Gt: picking up the truth from the ground for internet traffic," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 12–18, 2009.

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

[71] M. Jiang, G. Gou, J. Shi, and G. Xiong, "I know what you are doing with remote desktop," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2019, pp. 1–7.

[72] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted http streams," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2005, pp. 1–11.

[73] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 605–616.

[74] Y. Fu, H. Xiong, X. Lu, J. Yang, and C. Chen, "Service usage classification with encrypted internet traffic in mobile messaging apps," *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2851–2864, 2016.

[75] Y. Wang, Z. Li, G. Gou, G. Xiong, C. Wang, and Z. Li, "Identifying dapps and user behaviors on ethereum via encrypted traffic," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2020, pp. 62–83.

[76] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Mimetic: Mobile encrypted traffic classification using multimodal deep learning," *Computer networks*, vol. 165, p. 106944, 2019.

[77] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[78] P. V. Amoli, T. Hamalainen, G. David, M. Zolotukhin, and M. Mirzamohammad, "Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets," *JDCTA (International Journal of Digital Content Technology and its Applications*, vol. 10, no. 2, pp. 1–13, 2016.

[79] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[80] K. Park and W. Willinger, "Self-similar network traffic: An overview," *Self-Similar Network Traffic and Performance Evaluation*, pp. 1–38, 2000.

[81] W. Duch, R. Setiono, and J. M. Zurada, "Computational intelligence methods for rule-based data understanding," *Proceedings of the IEEE*, vol. 92, no. 5, pp. 771–805, 2004.

[82] N. Alon and J. H. Spencer, *The probabilistic method*. John Wiley & Sons, 2016.

[83] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[84] M. H. Davis, *Markov models & optimization*. CRC Press, 1993, vol. 49.

[85] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[86] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[87] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[88] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[89] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *Osdi*, vol. 16, no. 2016. Savannah, GA, USA, 2016, pp. 265–283.

[90] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[91] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[92] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, "A systematic review on supervised and unsupervised machine learning algorithms for data science," *Supervised and unsupervised learning for data science*, pp. 3–21, 2020.

[93] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, "Unsupervised machine learning for networking: Techniques, applications and research challenges," *IEEE access*, vol. 7, pp. 65 579–65 615, 2019.

[94] L. Van Der Maaten, E. Postma, J. Van den Herik *et al.*, "Dimensionality reduction: a comparative," *J Mach Learn Res*, vol. 10, no. 66-71, p. 13, 2009.

[95] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[96] G. H. Dunteman, *Principal components analysis*. Sage, 1989, no. 69.

[97] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[98] J. Lever, "Classification evaluation: It is important to understand both what a classification metric expresses and what it hides," *Nature methods*, vol. 13, no. 8, pp. 603–605, 2016.

[99] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. Ramos, and A. Madeira, "Flowlens: Enabling efficient flow classification for ml-based network security applications," in *Proceedings of the 28th Network and Distributed System Security Symposium (San Diego, CA, USA*, 2021.

[100] Z. M. Fadlullah, T. Taleb, N. Ansari, K. Hashimoto, Y. Miyake, Y. Nemoto, and N. Kato, "Combating against attacks on encrypted protocols," in *2007 IEEE International Conference on Communications*. IEEE, 2007, pp. 1211–1216.

[101] T. Taleb, Z. M. Fadlullah, K. Hashimoto, Y. Nemoto, and N. Kato, "Tracing back attacks against encrypted protocols," in *Proceedings of the 2007 international conference on Wireless communications and mobile computing*, 2007, pp. 121–126.

[102] Y. Feng and J. Li, "Toward explainable and adaptable detection and classification of distributed denial-of-service attacks," in *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 2020, pp. 105–121.

[103] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[104] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Network and Distributed Systems Security (NDSS) Symposium*, 2018.

[105] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Mobile malware detection through analysis of deviations in application network behavior," *Computers & Security*, vol. 43, pp. 1–18, 2014.

[106] S. Wang, Z. Chen, L. Zhang, Q. Yan, B. Yang, L. Peng, and Z. Jia, "Trafficav: An effective and explainable detection of mobile malware behavior using network traffic," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 2016, pp. 1–6.

[107] A. H. Lashkari, A. F. A. Kadir, H. Gonzalez, K. F. Mbah, and A. A. Ghorbani, "Towards a network-based framework for android malware detection and characterization," in *2017 15th Annual conference on privacy, security and trust (PST)*. IEEE, 2017, pp. 233–23 309.

[108] J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. Choffnes, "Recon: Revealing and controlling pii leaks in mobile network traffic," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016, pp. 361–374.

[109] A. Continella, Y. Fratantonio, M. Lindorfer, A. Puccetti, A. Zand, C. Kruegel, and G. Vigna, "Obfuscation-resilient privacy leak detection for mobile apps through differential analysis." in *NDSS*, 2017.

[110] N. Rosner, I. B. Kadron, L. Bang, and T. Bultan, "Profit: Detecting and quantifying side channels in networked applications." in *NDSS*, 2019.

[111] R. Coulter, Q.-L. Han, L. Pan, J. Zhang, and Y. Xiang, "Data-driven cyber security in perspective—intelligent traffic analysis," *IEEE transactions on cybernetics*, vol. 50, no. 7, pp. 3081–3093, 2019.

[112] E. Papadogiannaki and S. Ioannidis, "Acceleration of intrusion detection in encrypted network traffic using heterogeneous hardware," *Sensors*, vol. 21, no. 4, p. 1140, 2021.

[113] R. Moustafa and J. Slay, "A comprehensive data set for network intrusion detection systems," *School of Engineering and Information Technology University of New South Wales at the Australian Defense Force Academy Canberra, Australia, UNSW-NB15*, 2015.

[114] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Profiledroid: Multi-layer profiling of android applications," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, 2012, pp. 137–148.

[115] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, pp. 1–29, 2014.

[116] A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson, "Haystack: In situ mobile traffic analysis in user space," *arXiv preprint arXiv:1510.01419*, pp. 1–13, 2015.

[117] Y. Song and U. Hengartner, "Privacyguard: A vpn-based platform to detect information leakage on android devices," in *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2015, pp. 15–26.

[118] A. Le, J. Varmarken, S. Langhoff, A. Shuba, M. Gjoka, and A. Markopoulou, "Antmonitor: A system for monitoring from mobile devices," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*, 2015, pp. 15–20.

[119] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[120] Y. Feng, D. Sisodia, and J. Li, "Poster: Content-agnostic identification of cryptojacking in network traffic," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 907–909.

[121] S. Khirman and P. Henriksen, "Relationship between quality-of-service and quality-of-experience for public internet service," in *In Proc. of the 3rd Workshop on Passive and Active Measurement*, vol. 1, 2002.

[122] T. Hoßfeld and A. Binzenhöfer, "Analysis of skype voip traffic in umts: End-to-end qos and qoe measurements," *Computer Networks*, vol. 52, no. 3, pp. 650–666, 2008.

[123] F. Agboma, M. Smy, and A. Liotta, "Qoe analysis of a peer-to-peer television system," in *Proceedings of IADISInt. Conf. on Telecommunications, Networks and Systems*, 2008, pp. 365–382.

[124] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 513–526.

[125] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "Youtube qoe estimation based on the analysis of encrypted network traffic using machine learning," in *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2016, pp. 1–6.

[126] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for https and quic," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1331–1339.

[127] M. J. Khokhar, T. Ehlinger, and C. Barakat, "From network traffic measurements to qoe for internet video," in *2019 IFIP Networking Conference (IFIP Networking)*. IEEE, 2019, pp. 1–9.

[128] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the tor network," in *International*

symposium on privacy enhancing technologies symposium. Springer, 2008, pp. 63–76.

[129] M. Perry, "Experimental defense for website traffic fingerprinting," https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting/, posted: 2011-09-05.

[130] "Shadowsocks - a fast tunnel proxy that helps you bypass firewalls," https://shadowsocks.org/, accessed: 2022-02-11.

[131] S. Mistry and B. Raman, "Quantifying traffic analysis of encrypted web-browsing." 1998, project paper.

[132] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proceedings 2002 IEEE Symposium on Security and Privacy*. IEEE, 2002, pp. 19–30.

[133] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 255–263.

[134] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, 2009, pp. 31–42.

[135] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, 2011, pp. 103–114.

[136] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 143–157.

[137] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 1187–1203.

[138] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," 2017.

[139] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1928–1943.

[140] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1131–1148.

[141] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Rfc2616: Hypertext transfer protocol–http/1.1," 1999.

[142] A. Hintz, "Fingerprinting websites using traffic analysis," in *International workshop on privacy enhancing technologies*. Springer, 2002, pp. 171–178.

[143] L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in *European Symposium on Research in Computer Security*. Springer, 2010, pp. 199–214.

[144] S. J. Murdoch and R. N. Watson, "Metrics for security and performance in low-latency anonymity systems," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2008, pp. 115–132.

[145] W. De la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engel, K. Wehrle, and A. Panchenko, "Trafficsliver: Fighting website fingerprinting attacks with traffic splitting," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1971–1985.

[146] R. Dingledine and N. Mathewson, "Tor protocol specification," https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt, date of visit: 2021-10-05.

[147] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE symposium on security and privacy*. IEEE, 2012, pp. 332–346.

[148] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, 2013, pp. 201–212.

[149] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 263–274.

[150] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale." in *NDSS*, 2016.

[151] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 227–238.

[152] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1375–1390.

[153] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 27–46.

[154] K. Abe and S. Goto, "Fingerprinting attack on tor anonymity using deep learning," *Proceedings of the Asia-Pacific Advanced Network*, vol. 42, pp. 15–20, 2016.

[155] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-cnn: A data-efficient website fingerprinting attack based on deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 1, p. 19.

[156] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[157] S. E. Oh, S. Sunkam, and N. Hopper, "-fp: Extraction, classification, and prediction of website fingerprints with deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 191–209, 2019.

[158] C. Wang, J. Dani, X. Li, X. Jia, and B. Wang, "Adaptive fingerprinting: website fingerprinting over few encrypted traffic," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 149–160.

[159] N. P. Hoang, A. A. Niaki, P. Gill, and M. Polychronakis, "Domain name encryption is not enough: privacy leakage via ip-based website fingerprinting," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, pp. 420–440, 2021.

[160] O. Ajao, J. Hong, and W. Liu, "A survey of location inference techniques on twitter," *Journal of Information Science*, vol. 41, no. 6, pp. 855–864, 2015.

[161] Y. Ikawa, M. Enoki, and M. Tatsubori, "Location inference using microblog messages," in *Proceedings of the 21st international conference on world wide web*, 2012, pp. 687–690.

[162] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, "Accomplice: Location inference using accelerometers on smartphones," in *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*. IEEE, 2012, pp. 1–9.

[163] A. Gallagher, D. Joshi, J. Yu, and J. Luo, "Geo-location inference from image content and user tags," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2009, pp. 55–62.

[164] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci, "Measuring serendipity: connecting people, locations and interests in a mobile 3g network," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, 2009, pp. 267–279.

[165] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Contextual localization through network traffic analysis," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 925–933.

[166] ——, "Privacy-aware contextual localization using network traffic analysis," *Computer Networks*, vol. 118, pp. 24–36, 2017.

[167] F. Xu, Y. Li, H. Wang, P. Zhang, and D. Jin, "Understanding mobile traffic patterns of large scale cellular towers in urban environment," *IEEE/ACM transactions on networking*, vol. 25, no. 2, pp. 1147–1161, 2016.

[168] H. Wang, F. Xu, Y. Li, P. Zhang, and D. Jin, "Understanding mobile traffic patterns of large scale cellular towers in urban environment," in *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 225–238.

[169] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li, "Big data driven mobile traffic understanding and forecasting: A time series approach," *IEEE transactions on services computing*, vol. 9, no. 5, pp. 796–805, 2016.

[170] R. Lippmann, D. Fried, K. Piwowarski, and W. Streilein, "Passive operating system identification from tcp/ip packet headers," in *Workshop on Data Mining for Computer Security*, vol. 40, 2003.

[171] Y.-C. Chen, Y. Liao, M. Baldi, S.-J. Lee, and L. Qiu, "Os fingerprinting and tethering detection in mobile networks," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, 2014, pp. 173–180.

[172] M. Laštovička, S. Špaček, P. Velan, and P. Čeleda, "Using tls fingerprints for os identification in encrypted traffic," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–6.

[173] N. Ruffing, Y. Zhu, R. Libertini, Y. Guan, and R. Bettati, "Smartphone reconnaissance: Operating system identification," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 1086–1091.

[174] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE access*, vol. 5, pp. 18 042–18 050, 2017.

[175] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profiliot: a machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the symposium on applied computing*, 2017, pp. 506–509.

[176] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[177] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, and Z. Han, "Capsule network assisted iot traffic classification mechanism for smart cities," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7515–7525, 2019.

[178] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, 2006.

[179] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, 2005, pp. 229–240.

[180] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *International Conference on Passive and Active Network Measurement*. Springer, 2007, pp. 165–175.

[181] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT conference*, 2006, pp. 1–12.

[182] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *International workshop on passive and active network measurement*. Springer, 2004, pp. 205–214.

[183] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2005, pp. 50–60.

[184] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks," in *2017 IEEE International conference on big data (big data)*. IEEE, 2017, pp. 1271–1276.

[185] Q. Wang, A. Yahyavi, B. Kemme, and W. He, "I know what you did on your smartphone: Inferring app usage over encrypted data traffic," in *2015 IEEE conference on communications and network security (CNS)*. IEEE, 2015, pp. 433–441.

[186] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 439–454.

[187] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Multi-classification approaches for classifying mobile app traffic," *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018.

[188] F. Aiolli, M. Conti, A. Gangwal, and M. Polato, "Mind your wallet's privacy: identifying bitcoin wallet apps and user's actions through network traffic analysis," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 1484–1491.

[189] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2017.

[190] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[191] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services: Apple imessage and beyond," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 5–11, 2014.

[192] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 35–49.

[193] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing android encrypted network traffic to identify user actions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, 2015.

[194] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic," in *10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16)*, 2016.

[195] F. Yan, M. Xu, T. Qiao, T. Wu, X. Yang, N. Zheng, and K.-K. R. Choo, "Identifying wechat red packets and fund transfers via analyzing encrypted network traffic," in *2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 1426–1432.

[196] Y. Wang, N. Zheng, M. Xu, T. Qiao, Q. Zhang, F. Yan, and J. Xu, "Hierarchical identifier: Application to user privacy eavesdropping on mobile payment app," *Sensors*, vol. 19, no. 14, p. 3052, 2019.

[197] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, "Understanding online social network usage from a network perspective," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, 2009, pp. 35–48.

[198] Y. Feng, "Botflowmon: Identify social bot traffic with netflow and machine learning," 2018.

[199] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Can't you hear me knocking: Identification of user actions on android apps via traffic analysis," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, 2015, pp. 297–304.

[200] Y. Zhao, X. Ma, J. Li, S. Yu, and W. Li, "Revisiting website fingerprinting attacks in real-world scenarios: A case study of shadowsocks," in *International Conference on Network and System Security*. Springer, 2018, pp. 319–336.

[201] A. J. Pinheiro, P. Freitas de Araujo-Filho, J. de M. Bezerra, and D. R. Campelo, "Adaptive packet padding approach for smart home networks: A tradeoff between privacy and performance," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3930–3938, 2021.

[202] J. Gong and T. Wang, "Zero-delay lightweight defenses against website fingerprinting," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 717–734.

[203] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis." in *NDSS*, vol. 9. Citeseer, 2009.

[204] X. Luo, P. Zhou, E. W. Chan, W. Lee, R. K. Chang, R. Perdisci *et al.*, "Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows." in *NDSS*, vol. 11, 2011.

[205] G. Cherubin, J. Hayes, and M. Juárez, "Website fingerprinting defenses at the application layer." *Proc. Priv. Enhancing Technol.*, vol. 2017, no. 2, pp. 186–203, 2017.

[206] Z. Deng, Z. Liu, Z. Chen, and Y. Guo, "The random forest based detection of shadowsock's traffic," in *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 2. IEEE, 2017, pp. 75–78.

[207] J. Beznazwy and A. Houmansadr, "How china detects and blocks shadowsocks," in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 111–124.

[208] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*. sn, 2016, pp. 407–414.

[209] P. Choorod and G. Weir, "Tor traffic classification based on encrypted payload characteristics," in *2021 National Computing Colleges Conference (NCCC)*. IEEE, 2021, pp. 1–6.

[210] I. Goldberg and C. A. Wood, "Network-based website fingerprinting," https://datatracker.ietf.org/doc/html/draft-wood-privsec-wfattacks-00, 2019.

[211] "New tor release: Tor 0.4.0.5," https://blog.torproject.org/new-release-tor-0405/, 2019, accessed: 2022-02-15.

[212] S. Li, H. Guo, and N. Hopper, "Measuring information leakage in website fingerprinting attacks and defenses," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1977–1992.

[213] R. Nithyanand, X. Cai, and R. Johnson, "Glove: A bespoke website fingerprinting defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014, pp. 131–134.

[214] X. Cai, R. Nithyanand, and R. Johnson, "Cs-buflo: A congestion sensitive website fingerprinting defense," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014, pp. 121–130.

[215] D. Lu, S. Bhat, A. Kwon, and S. Devadas, "Dynaflow: An efficient website fingerprinting defense based on dynamically-adjusting flows,"

in *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, 2018, pp. 109–113.

[216] S. Henri, G. Garcia-Aviles, P. Serrano, A. Banchs, and P. Thiran, "Protecting against website fingerprinting with multihoming," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 2, pp. 89–110, 2020.

[217] T. Wang, "The one-page setting: A higher standard for evaluating website fingerprinting defenses," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2794–2806.

[218] mikeperry, "Experimental defense for website traffic fingerprinting," https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting/, 2011, accessed: 2022-02-11.

[219] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, and J. Yuan, "Survey on uav cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3417–3442, 2019.

[220] Y. Feng, J. Li, and T. Nguyen, "Application-layer ddos defense with reinforcement learning," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 2020, pp. 1–10.

[221] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, "Botmark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors," *Information Sciences*, vol. 511, pp. 284–296, 2020.

[222] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Ieee communications surveys & tutorials*, vol. 16, no. 1, pp. 303–336, 2013.

[223] S. Barnum, "Standardizing cyber threat intelligence information with the structured threat information expression (stix)," *Mitre Corporation*, vol. 11, pp. 1–22, 2012.

[224] H. Jmila, G. Blanc, M. R. Shahid, and M. Lazrag, "A survey of smart home iot device classification using machine learning-based network traffic analysis," *IEEE Access*, vol. 10, pp. 97 117–97 141, 2022.

[225] T. Wu, F. Breitinger, and S. Niemann, "Iot network traffic analysis: Opportunities and challenges for forensic investigators?" *Forensic Science International: Digital Investigation*, vol. 38, p. 301123, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666281721000214

[226] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, "The rise of traffic classification in iot networks: A survey," *Journal of Network and Computer Applications*, vol. 154, p. 102538, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520300126

[227] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, "The dark side(-channel) of mobile devices: A survey on network traffic analysis," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2658–2713, 2018.

[228] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica, "Large-scale mobile traffic analysis: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 124–161, 2016.

[229] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

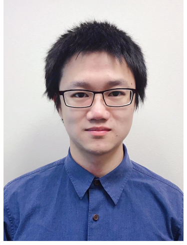
[230] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2014.

[231] J. a. V. Gomes, P. R. M. Inácio, M. Pereira, M. M. Freire, and P. P. Monteiro, "Detection and classification of peer-to-peer traffic: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, jul 2013. [Online]. Available: https://doi.org/10.1145/2480741.2480747

[232] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 37–52, 2009.

[233] B. Chandrasekaran, "Survey of network traffic models," *Waschington University in St. Louis CSE*, vol. 567, 2009.

[234] A. Erramilli, M. Roughan, D. Veitch, and W. Willinger, "Self-similar traffic and network dynamics," *Proceedings of the IEEE*, vol. 90, no. 5, pp. 800–819, 2002.



Yebo Feng is a Ph.D. candidate in the Computer Science Department at the University of Oregon (UO), where he conducts his research in the Center for Cyber Security and Privacy. He received his M.S. degree from UO in 2018, B.E. from Yangzhou University in 2016, all in computer science. His research interests include network security, anomaly detection, and network traffic analysis. He is the recipient of the Best Paper Award of 2019 IEEE CNS, Gurdeep Pall Graduate Student Fellowship of UO, and Ripple Research Fellowship. He has served as the reviewer of IEEE TDSC, IEEE TIFS, ACM TKDD, IEEE JSAC, IEEE COMST, etc. He was on the program committees of several international conferences, such as CYBER, SECURWARE, and B2C. He was also on the AE committees of USENIX OSDI and USENIX ATC.

Jun Li is a professor in the Department of Computer Science and director of the Network and Security Research Laboratory at the University of Oregon. He was also a Ripple Fellow, Narus Research Fellow, and founding director of the Center for Cyber Security and Privacy at the University of Oregon. He has received the CAREER Award from the US National Science Foundation, the Faculty Excellence Award from the University of Oregon, and the Recognition of Service Award from ACM. He received his Ph.D. with honors from UCLA in 2002. His research interests include networking, distributed systems, cybersecurity, and blockchain. He has published more than 100 peer-reviewed papers, including several Best Paper awards.

Jelena Mirkovic is Research Team Leader at USC Information Sciences Institute and Research Associate Professor of Computer Science at USC. During her professional career she published more than 100 conference and journal papers, and the first book on the denial-of-service attacks. She also pioneered use of testbeds in cybersecurity education. Her research interests span cybersecurity, networking and education.