Grano-GT: A granular ground truth collection tool for encrypted browser-based Internet traffic

Faiz Zaki
Department of Computer Systems and Network
Universiti Malaya
Kuala Lumpur, Malaysia
faizzaki@um.edu.my

Abdullah Gani
Faculty of Computing and Informatics
University Malaysia Sabah
International Campus Labuan
abdullahgani@ums.edu.my

Hamid Tahaei
School of Electrical and Computer Technology
Xiamen University Malaysia
Selangor, Malaysia
email@email.com

Steven Furnell
School of Computer Science
University of Nottingham
Nottingham, United Kingdom
steven.furnell@nottingham.ac.uk

Nor Badrul Anuar
Department of Computer Systems and Network
Universiti Malaya
Kuala Lumpur, Malaysia
badrul@um.edu.my

ABSTRACT

Modern network traffic classification puts much attention toward producing a granular classification of the traffic, such as at the application service level. However, the classification process is often impaired by the lack of granular network traffic ground truth. Granular network traffic ground truth is critical to provide a benchmark for a fair evaluation of modern network traffic classification. Nevertheless, in modern network traffic classification, existing ground truth tools only managed to build the ground truth at the application name level at most. Application name level granularity is quickly becoming insufficient to address the current needs of network traffic classification and therefore; this paper presents the design, development and experimental evaluation of Grano-GT, a tool to build a reliable and highly granular network traffic ground truth for encrypted browser-based traffic at the application name and service levels. Grano-GT builds on four main engines which are packet capture, browser, application and service isolator engines. These engines work together to intercept the application requests and combine them with the support of temporal features and cascading filters to produce reliable and highly granular ground truth. Preliminary experimental results show that Grano-GT can classify the Internet traffic into respective application names with high reliability. Grano-GT achieved an average accuracy of more than 95% when validated using nDPI at the application name level. The remaining 5% loss of accuracy was primarily due to the unavailability of signatures in nDPI. In addition, Grano-GT managed to classify application service traffic with significant reliability and validated using the Kolmogorov-Smirnov Test.

KEYWORDS

Ground truth, network traffic classification, granular

1. INTRODUCTION

Network traffic classification is a fundamental process that has a wide implementation in various domains [1]. For example, traffic classification is important for network administrators to analyze the type of traffic in their networks, such as HTTP or mail traffic. It helps network administrators to optimize and prioritize their network resources based on the type of traffic classified. Besides optimizing network resources, traffic classification is also critical for network security. Network security benefits from traffic classification by having the ability to apply security policies on targeted traffic. However, in order to evaluate the accuracy of a network traffic classification model, we need a reliable network traffic ground truth. The evaluation process using the network traffic ground truth is critical to justify the performance of a proposed technique. There are a few existing tools that can build network traffic ground truth such as Traffic Labeller [2] and GT [3]. Despite that, the tools provide a rather limited level of classification granularity such as at the application protocol (e.g., HTTP and FTP) and application category levels (e.g., P2P and Mail). In modern networks, there is a demand for higher classification granularity than this sort of high-level protocol or category. Higher classification granularity such as at the application name and service level is highly valuable in modern networks. Most applications in modern networks offer various user interactions such as posting comments, liking a photo or streaming a video. Hence, the ability to classify traffic at these levels allows network administrators to be more precise when applying their network policies. Generally, we can divide the classification granularity into the 3 categories, a) coarse-grained, b) fine-grained and c) binary as shown in Figure 1.

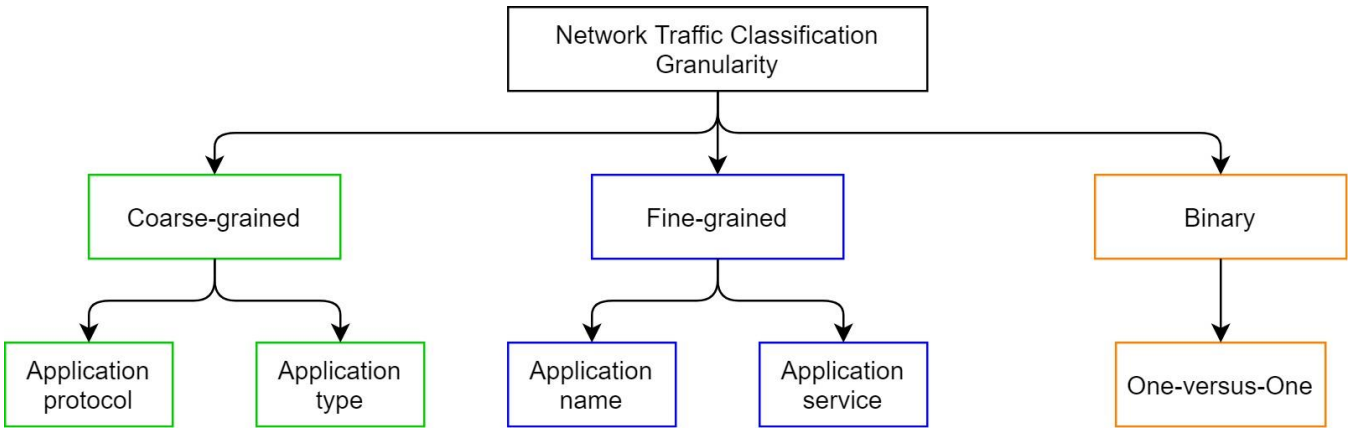

Figure 1: Classification granularity

The coarse-grained category includes application protocol (e.g., HTTP, SSH, SMTP) and type (e.g., Video, Multimedia, Games) which only provide a high-level classification. It is useful for lightweight use cases such as the traditional packet filtering firewall. On the other hand, fine-grained category classifies at application name (e.g. Facebook, Twitter) and service (e.g., Facebook-chat, Twitter-post) which gives more visibility into the network. For instance, Palo Alto's Next-Generation Firewall can classify traffic down to its application service level (e.g. Facebook-chat, Facebook-Audio) by using signature-based techniques [4]. Finally, another category of granularity is the binary category. The binary category focuses on differentiating one major traffic class from another or one-versus-one as we defined it in this paper. For example, [5] distinguished between encrypted and non-encrypted traffic while [6] applied the same classification granularity by discriminating malicious traffic from non-malicious traffic. The various levels of granularity also map to diverse classification techniques using port, deep packet inspection (DPI), statistical, machine learning and behavioral-based. Despite that, there exist several critical problems when validating classification techniques.

Among the problems when validating classification techniques is the lack of reliable and highly granular network traffic ground truth. Most network traffic traces such as from [7] and [8] only managed to build the ground truth down to the application name level at most. Therefore, it is challenging to validate the performance of classification techniques without proper ground truth which is publicly available. Consequently, researchers tend to validate their techniques using private ground truth. Using private ground truth for validation leads to another problem which is that the reliability of the technique used to collect the private ground truth is often arguable. Some of the common techniques used are port-based, DPI or manual classification. Manual classification takes place in a controlled lab environment. For example, [9] generated their ground truth by running the target application in an isolated machine. Although this approach can potentially provide the ground truth, we need to acknowledge that there is also background traffic running on the machine such as by daemons. Hence, it can contaminate the ground truth traffic thus decreasing its reliability. As a workaround, it is a common approach to apply DPI on the captured traffic. DPI operates by inspecting packet payloads to find a match of predefined signature strings. For example, HTTP traffic is easily identifiable by the signature string in its methods like GET and POST. However, DPI is less effective when dealing with encrypted traffic and applications having similar traffic signatures.

Therefore, this paper introduces Grano-GT which is a ground truth collection tool that aims to collect reliable and highly granular ground truth for encrypted browser-based traffic at the application name and service level. There are four main engines that power Grano-GT which are packet capture, browser, application and service isolator engines. These engines take advantage of Python and the Chrome DevTools Protocol [10] to isolate traffic coming from a target tab in the Chrome browser. By using this approach, Grano-GT can reliably label the application name without any contamination from background traffic. In addition, Grano-GT inspects the application layer header inside the Chrome environment for unique signatures and passes the signatures to a series of cascading filters to discriminate the application into its services. Grano-GT intelligently takes advantage of time-related features to effectively complement the process of segregating the traffic into its application services in the event of traffic encryption.

The ability to address encrypted traffic for network traffic ground truth collection demonstrates the reliability of our work. To demonstrate the reliability of Grano-GT, we evaluate it in two network environments; a) a wired campus network: UM-Net and b) a wireless home network: HOME-Net. Our experiments show that Grano-GT achieves optimum reliability at the application-name level due to the ability to intercept and isolate the IP addresses used during the browsing session. As a result, we eliminate the existence of background traffic belonging to other unrelated applications regardless of the encryption state of the traffic. Therefore, Grano-GT removes any possible contamination in the traffic capture, which was one of the problems highlighted in earlier works. We validate our claim by using nDPI which is a well-known DPI engine [11] and achieve 95% average accuracy mainly due to the unavailability of signatures in nDPI. Grano-GT addresses the remaining 5% loss of accuracy when using nDPI by using the IP isolation technique mentioned earlier. On the other hand, to

strengthen our finding at the application service level, we take advantage of the Kolmogorov-Smirnov Test to validate the distinction between the packet size distributions of different application services. The Kolmogorov-Smirnov Test is a statistical test to verify whether a sample belongs to a reference probability distribution. We adopt the Kolmogorov Smirnov Test due to its ability to ingest our packet size distribution values and conclude whether they are from the same application service. Our application of the Kolmogorov-Smirnov Test is similar to studies done by [12] and [13] in the Physics and Big Data domains. As a summary, the contributions of this paper are as follows:

a) A new approach and design for a ground truth collection tool called Grano-GT to collect reliable and highly granular network traffic ground truth for encrypted browser-based traffic at the application name and service levels.
b) A technique using IP address isolation to achieve the most optimum collection reliability at the application name level.
c) An approach using time-related features to complement the traffic segregation process at the application service level to remove the dependency on packet payload inspection, thus addressing the issue of traffic encryption and privacy concerns.

We organize the remainder of the paper as follows. Section 2 discusses the related work in the domain. We present the architecture of Grano-GT in Section 3 and provide comprehensive interpretations for each component of the architecture. Section 4 outlines our experimental setup and in Section 5, we present the experimental analysis of this paper which includes elaborating on the evaluation and validation processes of our experiments. In Section 6, we discuss key issues in the domain and the ways to move forward. Finally, Section 7 concludes the paper.

2. RELATED WORK

Reliable ground truth for network traffic classification remains as an open problem in the research community [14, 15]. It hinders proper comparison and validation between the proposed solutions. This is because most solutions utilized a private ground truth dataset for validation [16]. These private ground truths are commonly collected and labelled using less reliable techniques as described in the previous section such as port-based, DPI or manual isolation as shown in Figure 2.

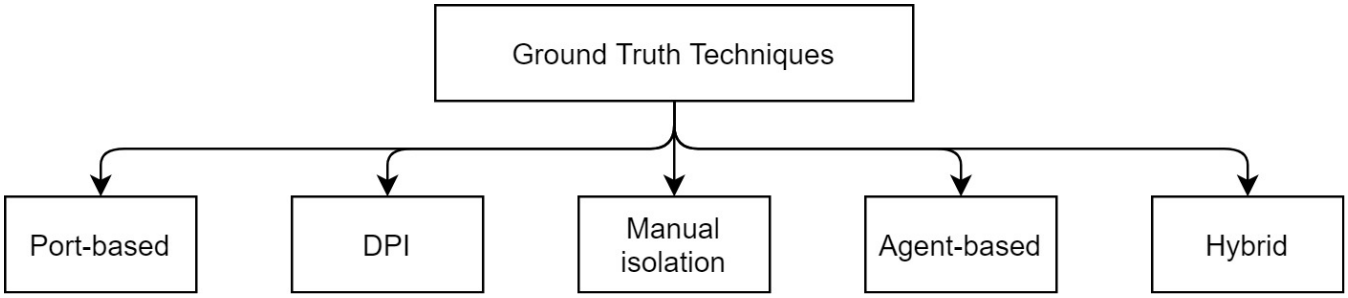

Figure 2 Ground truth techniques

Port-based is the most straightforward approach to acquire ground truth. It matches the port number of the targeted traffic against the list of registered port numbers of standard applications and protocols as published by the Internet Assigned Numbers Authority (IANA) [17]. For example, network traffic destined for port 80 matches the HTTP tag as described by IANA. However, considering the current dynamic port allocations and obfuscation technique [18], port-based is quickly becoming obsolete if used as a standalone technique. This is because modern applications tend to move away from using standard and well-known port numbers. A study by [19] proved that using the port-based as a standalone technique only achieved 70% accuracy at most. As a result of the low accuracy, researchers often reserve the port-based technique as a last resort such as during the event of unavailability of the payload information. For instance, [20] and [21] utilized the port-based technique to create their ground truths due to such an event. [20] assumed in their work that the port-based technique was largely accurate because their dataset was collected before the widespread use of dynamic port numbers.

In order to address the issue of dynamic port numbers, it is relatively common to use DPI to label the ground truth. DPI inspects the packet payload for unique application string signatures. An application string signature may be in the form of common strings such as the HTTP methods (e.g. GET and POST) that can uniquely identify an application. Due to the ability to inspect down to the packet payload, DPI emerges as a popular technique to create ground truth. [22] developed a DPI tool to match regular expression patterns against the packet payload. In [16], the authors compared the performance of multiple DPI tools and concluded that nDPI [11] and Libprotoident [23] as the most reliable open source DPI. As an example, [24] utilized nDPI to label the ground truth dataset. However, the effectiveness of DPI starts to diminish with encrypted traffic. Besides that, DPI requires frequent updates to the signature library to keep up with new applications. [25] proved this issue in their work when they were unable to label the ground truth of packets coming from new applications in one of their datasets. Hence, coupling the ineffectiveness of DPI on encrypted traffic together with the inconsistent performance among different DPI techniques, the reliability of DPI in producing ground truth is arguable.

As an alternative technique to combat the decreasing reliability of DPI, researchers consider the possibility of manually isolating the traffic. Manual isolation involves preparing machines (e.g., personal computers, virtual machines) with exclusive access only to the target application. This technique carries the assumption that all traffic captured through the machine belongs to the target application. [9] and [26] created the ground truth by manually isolating the application traffic. [9] used Docker as a virtual container to sterilize the network capture environment. Despite that, it is also worth noting that there still exists some background traffic such as from the system daemons in the machine that might contaminate the captured ground truth [3].

To lower the risk of contaminating the ground truth with background traffic, researchers introduced agent-based ground truth collection. The agents monitor the socket calls on the client machine. Sockets are endpoints in a communication flow between two applications running over a network. Therefore, monitoring the socket calls allows the extraction and association of the information on the running application process with the captured traffic thus producing accurate ground truths. For example, [8] introduced MIRAGE which tags mobile traffic with its process information through socket monitoring. Meanwhile, [3] proposed a tool named GT, which also tracks the socket activities and tags the traffic based on the information gathered from the sockets. Using this technique, GT managed to label the ground truth down to its application name (e.g., Safari, Apple Mail). A tool named Flowsing further improved GT by having the ability to label ground truth at multiple levels (i.e., application type and name) [27]. On the other hand, [2] proposed Traffic Labeller (TL) that captures all user socket calls and corresponding application process information. Then, TL tags the traffic by inserting application information into the TOS field of matching IP packets. Although it ensures high accuracy, it modifies the size of the IP packet, thus risking inaccurate capture.

As a countermeasure to inaccurate capture problems, there are also hybrid techniques that provide multiple layers of classification mechanism to increase the reliability of the network traffic ground truth. [28] introduced the Ground Truth Verification System (GTVS) which is a multi-layered hybrid framework to build ground truth datasets. It combines multiple techniques in the form of heuristics rules such as DPI and port-based classification. GTVS runs the heuristics rules in five iterations, with each iteration focusing on the remaining unclassified network traffic flow. The final iteration of GTVS involves manual human inspection to label the ground truth. Although GTVS managed to increase the reliability of the ground truth, it is not fully automated thus presenting some concerns on the scalability of the tool.

Table 1 lists the advantages and disadvantages of existing ground truth techniques while Table 2 summarizes the ground truth tools discussed in this section. All the tools offer different advantages over one another. However, they share the same limitation where the classification granularity is limited to the application name (i.e., Safari, Apple Mail). While application name granularity may have been more than sufficient in the past, [29] mentioned the need for a more granular classification, particularly at the application service level (i.e., Facebook-chat, Skype-voice) in modern networks. Besides that, existing tools put little attention to encrypted traffic because majority of the tools only deal with lower granularity levels and utilize socket monitoring through agent-based technique. Socket monitoring ignores the content of the packet hence is unaffected by the encryption state. Therefore, socket monitoring is less effective in collecting highly granular ground truth such as at the application service level as proposed by this paper.

As such, considering the current limitations of existing tools, this paper aims to pioneer the effort to build a reliable and highly granular ground truth collection tool for encrypted browser-based traffic at the application name and service levels through Grano-GT.

Table 1 Advantages and disadvantages of existing ground truth techniques

Technique	Advantages	Disadvantages
Port-based	A simple and straightforward technique by comparing the port number against a list of predefined port numbers by IANA.	Dynamic port allocation drives down the accuracy of this technique as applications move away from standard port assignments.
DPI	A very accurate technique by inspecting the contents of the network packet for matching signature strings.	Traffic encryption diminishes the effectiveness of DPI as it is unable to inspect the encrypted packet contents.
Manual isolation	The technique is easily set up by running applications in an isolated machine and network environment.	There is still a possibility for a contaminated traffic capture from the background daemon traffic.
Agent-based	A highly accurate and reliable technique by using agents to monitor network socket activities.	The classification granularity is limited to application name level at most as application service traffic rarely spawns unique socket activities.
Hybrid	Increases reliability of network traffic ground truth by using multiple levels of classification.	It involves manual human effort thus reduces the scalability of the technique for large datasets.

Table 2 Summary of existing ground truth tools

Tool Name	Technique	Granularity level	Application type	Performance
MIRAGE [8]	Agent-based	Application name	Mobile applications	Not mentioned
GT [3]	Agent-based	Application protocol and name	Desktop applications	Able to tag up to 99% of the bytes and 95% of the flows
Flowsing [27]	Agent-based	Application type and name	Desktop applications	Able to classify traffic into three different aggregation levels
Traffic Labeller [2]	Agent-based	Application type	Desktop applications	100% accuracy using port-based validation
GTVS [28]	Hybrid	Application type	Desktop applications	Outperformed L7-filter in terms of false negatives and false positives.

3. GRANO-GT ARCHITECTURE

The architecture of Grano-GT operates to build a reliable and highly granular network traffic ground truth. It sources the ground truth from live network traffic coming from a Google Chrome browser. The live network traffic goes through a series of processing engines before producing a collection of network traffic ground truth at the application name and service levels as the final output. Grano-GT works with both unencrypted and encrypted traffic as it is independent of the packet payload. As a result, Grano-GT removes the risk of tampering with user privacy.

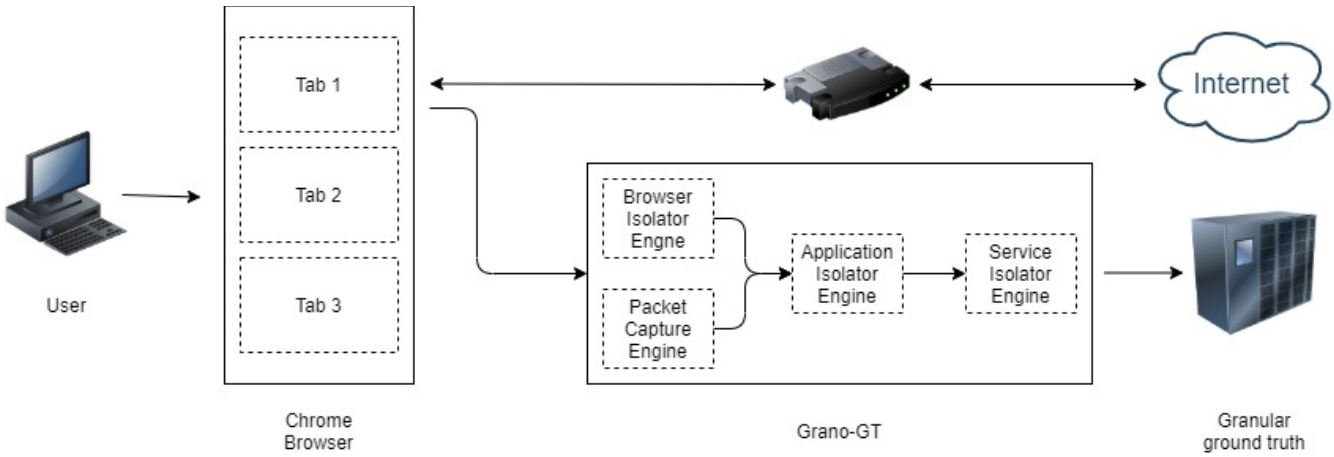

Figure 3: Grano-GT architecture

Figure 3 shows the architecture of Grano-GT, which builds on four main elements: a *browser isolator engine* which utilizes Chrome DevTools Protocol and PyChrome to isolate network responses from a target tab; a *packet capture engine* running on tshark [30], which is the command-line version of Wireshark to capture all network traffic simultaneously; an *application isolator engine* built on Python which filters only the specific target application traffic; and a *service isolator engine* to separate the target application service from the entire application traffic. The distinct components of the approach, and the main contributing technologies, are discussed in the sub-sections that follow

3.1 Browser Isolator Engine

The *browser isolator engine* runs on two main tools which are the Chrome Devtools Protocol and PyChrome. These tools ensure that the engine can effectively isolate the application traffic and remove all unrelated traffic from the capture.

3.1.1 Chrome Devtools Protocol

One of the main challenges of building a highly granular ground truth is distinguishing the raw traffic at the application service level. Application service traffic rarely spins up an entirely new network socket or uses a unique IP address that allows more straightforward

classification if compared to the application name traffic which sits at a lower granularity level. Therefore, we tackle this problem by intercepting the traffic in the browser itself which gives visibility to all browser interactions. Additionally, we selected Google Chrome as it is the most popular browser with a 67% market share according to a recent study [31]. Google Chrome also comes with the Chrome Devtools Protocol (CDP). CDP is unlike a typical network protocol (e.g. TCP). Instead, it is a set of tools or interface functions that allow rapid debugging of Chromium, Chrome and other Blink-based browsers programmatically [10]. CDP maintains a publicly available set of API. The API covers various domains such as performance, browser and network. Grano-GT mainly takes advantage of the browser and network API to profile the network activity from a specific target tab accurately.

3.1.2 PyChrome

Grano-GT builds on Python programming language. It is helpful to have a Python driver for CDP to simplify its implementation. Hence, Grano-GT utilizes PyChrome which is a Python driver for CDP [32]. It is an open-source driver that allows developers to access CDP's API using Python. For example, in this paper, PyChrome is responsible for executing all browser-based tasks such as launching the target application and intercepting network events through callback functions.

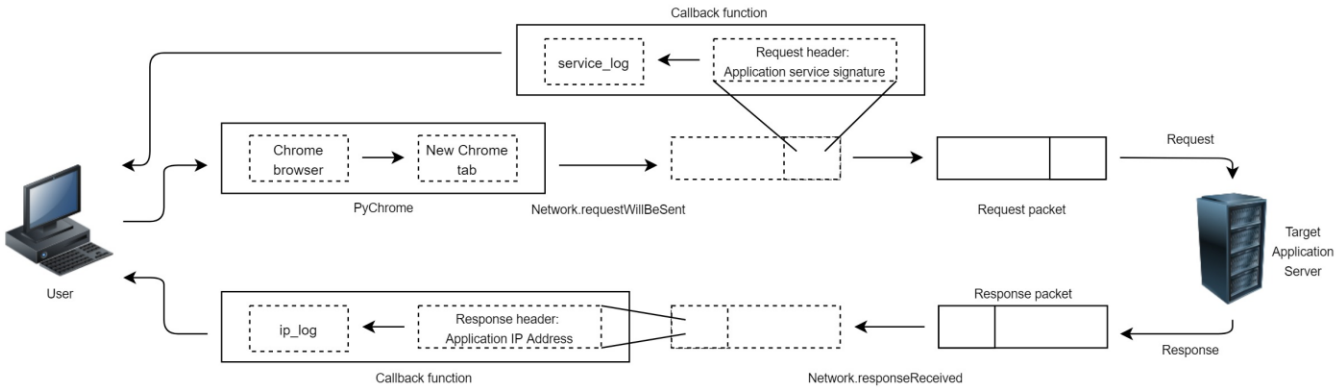

Figure 4 Workflow of browser isolator engine

Another challenge of collecting network traffic ground truth is the existence of background traffic that can contaminate the capture. To remove the risk of contamination, we separate the network responses from a specific browser tab using the API from CDP through the *browser isolator engine*. The *browser isolator engine*: 1) spawns a new tab in the Chrome browser. 2) launches a request to the target application in the new tab. These two processes make use of PyChrome to achieve the objectives programmatically.

As shown in **Figure 4**, the engine implements a callback function that triggers on *Network.responseReceived* event. In other words, the callback function executes when it receives response packets from the application server. Then, it inspects all the packet headers and logs the remote IP addresses. Consequently, it produces a list of IP addresses purely from the application running on the target tab. This is critical because modern applications typically serve their resources from multiple IP addresses.

Besides that, the *browser isolator engine* is also responsible for detecting the target application service. It implements a callback function that triggers on *Network.requestWillBeSent* event. The callback function inspects the request header and searches for matching signatures for the target application service. If there is a matching signature, the engine logs the epoch time of the match for further processing in the *service isolator engine* later on. It is also worth noting that the signatures used in Grano-GT is only limited to request headers (e.g. request URL) and is independent of the packet payload content.

3.2 Packet Capture Engine

The *packet capture engine* sits at the core of Grano-GT as it functions to capture the network traffic during the session. There are various packet capture tools available such as tcpdump and Wireshark. Grano-GT mainly utilizes the command-line version of Wireshark, called tshark [30]. Tshark retains all the advantages of Wireshark while providing a more effective platform for customizing the output programmatically. In Grano-GT, tshark serves as the network traffic capture tool to dump all the network traffic during a user's browsing session.

Tshark runs simultaneously alongside the *browser isolator engine* using Python's subprocess module. During the live capture, tshark immediately applies a filter to capture only the network traffic originating and destined to the user's machine IP address. The filter ensures that it captures all network traffic intended for the user's machine regardless of whether it is from the target application or other applications running on the machine at the time of capture (i.e. non-target application).

3.3 Application Isolator Engine

Network traffic captured through the *packet capture engine* consist of traffic from all applications intended for the user's machine. There is a need to extract only the traffic of the target application for further processing. To extract the traffic of the target application, we recall the IP log file generated from the *browser isolator engine* earlier as well as the captured traffic dump as inputs for the *application isolator engine*.

Algorithm 1: Application isolator engine

Input: captured_traffic, *CT* and IP_log_file, *Set(IP_log)*
Output: application_traffic.pcap

```
1:  for packet in CT:
2:      if packet.ip.source in IP_log or packet.ip.dest in IP_log:
3:          write(packet, application_traffic.pcap)
4:      end if
5:  end for
6:  return application_traffic.pcap
```

Algorithm 1 explains the process in further detail. The *application isolator engine* traverses each packet in the captured traffic. It verifies if the source or destination IP address matches any of the IP addresses in the IP log file. If the IP matches, it writes the packet into a new file named application_traffic.pcap. The *write* function is a self-written utility function that is based on the PcapWriter class from the Scapy library. The output of Algorithm 1 is a sanitized file containing only the target application traffic.

3.4 Service Isolator Engine

The sanitized application traffic contains the entire traffic from the target application. However, we are only interested in the traffic of the target application service. In order to extract the service, the *service isolator engine* implements a series of cascading filters. There are three levels of filtering throughout the segregation process: L1, L2 and L3 service filters.

L1 service filter aims to extract the target application service by using the *service_log* epoch time from the *browser isolator engine*. The epoch time signifies the exact time that the user executed the target application service. As such, the L1 service filter extracts all packets having epoch time more than the time recorded in the *service_log,* as shown in Algorithm 2. Using this method, the L1 service filter eliminates the unrelated packets arriving before the target application service execution.

Algorithm 2: L1 service filter

Input: application_traffic, *AT* and service_log, *SL*
Output: service_traffic, time_delta

```
1:  epoch = SL.epoch
2:  time_delta = list()
3:  for packet in AT:
4:      if packet.time_epoch > epoch:
5:          write(packet, service_traffic)
6:          time_delta.append(packet.time_delta_displayed)
7:      end if
8:  end for
9:  return  service_traffic, time_delta
```

Although the L1 service filter can extract the target application service traffic, there is still a high chance that the traffic contains other unrelated packets that arrived after the epoch time recorded in *service_log*. Therefore, the L2 service filter classifies the traffic further in order to increase precision. The process classifies the application service traffic by using the standard deviation of interarrival times as the segregation factor. Equation 1 shows the formula for the standard deviation used in this paper.

$$\sigma = \sqrt{\frac{\sum(\delta t_i - \mu)^2}{N}} + \epsilon \tag{1}$$

where δt_i = element in the *time_delta* list from L1 service filter,

μ = mean of *time_delta,*
N = length of *time_delta,*
ϵ = compensating value for *time_delta*

Algorithm 3: L2 service filter

Input: service_traffic, *ST* and time_delta, *TD*
Output: Multiple PCAP files

```
ε = 0.05
interarrival_std = std(time_delta) + ε
cnt = 1
while True:
    packet = ST.load_packet()
    while packet.time_delta_displayed < interarrival_std:
        write(packet, service_traffic_cnt)
        packet = ST.load_packet()
    end while
    cnt = cnt + 1
end while
return service_traffic_cnt
```

Based on our study, application service traffic occurs in a burst that is recognizable within the standard deviation of its interarrival times. This is also in line with a recent study that identified application traffic through its burstiness [33]. However, it is also worth noting that there is a micro delay between exact service execution and packet capture time. Therefore, we introduced the value $\boldsymbol{\epsilon}$, epsilon, which is an arbitrary small subsecond value to compensate for the delay. We selected the epsilon value of 0.05 as shown in Algorithm 3 after cross-validating the value against a set of values, namely 0.010, 0.025, 0.1, 0.125 and 0.15. Algorithm 3 also shows how it classifies the service traffic from the L1 service filter into multiple PCAP files. Each PCAP file consists of packets that are within the standard deviation threshold.

Algorithm 4: L3 service filter

Input: Multiple PCAP files and signature, *S,* and interarrival standard deviation, *interarrival_std*
Output: Pure application service PCAP files

```
for each PCAP file, f:
    keyword_cnt = 0
    for packet in f:
        time_diff = packet.time_epoch - epoch
        if S in packet.header or time_diff < interarrival_std:
            keyword_cnt = keyword_cnt + 1
            break
        end if
    end for
    if keyword_cnt == 0:
        remove(f)
    else:
        return f
```

Lastly, each of the PCAP files from the L2 service filter goes through the final stage of filtering which is the L3 service filter shown in Algorithm 4. The L3 service filter narrows down the segregation and checks each of the PCAP files for matching signatures. In the event of traffic encryption or none of the signature matches, the filter checks if the difference between the packet arrival time and the logged application service time is lower than the standard deviation of interarrival times. If none of the conditions matches, the filter drops the PCAP file, assuming that it is unrelated to the target application service. The final output of the L3 service filter is a collection of PCAP files purely of the target application service. In addition, the use of time-related features allows Grano-GT to operate in both unencrypted and encrypted environment without tampering with user privacy. Table 3 summarizes the three service filters discussed previously.

Table 3 Summary of service filters

Service filter	Objective	Output
L1	Extract related target application traffic starting from the execution of application service.	Target application traffic beginning from the time user executed application service.
L2	Partition the target application traffic into multiple smaller PCAP files containing possible target application service traffic.	Multiple PCAP files containing traffic partitioned from L1.
L3	Remove unrelated traffic from L2 and return only the target application service traffic.	PCAP files of target application service traffic.

4. TESTBED SETUP

We evaluated Grano-GT on two network environments: UM-Net which is a wired campus network at Universiti Malaya (UM) in Kuala Lumpur, Malaysia and HOME-Net which is a wireless home network provided by Telekom Malaysia, also in Kuala Lumpur, Malaysia between November 2019 to April 2020. Using two distinct network environments provides a better evaluation of the robustness of Grano-GT on different network settings.

At UM, we ran Grano-GT on a 3.4 GHz quad-core machine running Windows 10. We ran the tool and collected a total of 56 GB of network ground truth traffic for ten common and popular browser-based applications in our region [34-36]. On the other hand, we ran Grano-GT in a wireless home network to diversify the network conditions. We utilized a mid-range 2.0 GHz quad-core machine and collected a total of 36 GB of ground truth traffic for the same number of applications. Table 4 shows the complete list of applications involved in this paper. It shows the application name such as Facebook and its corresponding category which is Social Media for easier reference. In addition, Table 4 also lists the related application services for each respective application. Besides that, the Total Bytes column indicates the total size in bytes of the captured traffic for each application name and service in both network environments, UM-Net and HOME-Net. We accessed all the applications from a Google Chrome browser version 81.0.4044.92, which is the latest version at the time of writing.

Table 4 Complete list of applications

Application name	Application category	Total Bytes		Application service	Total Bytes	
		UM-Net	HOME-Net		UM-Net	HOME-Net
Facebook	Social media	308593461	265613399	comment	724882	200872
				post	1753208	219833
				react	14148	211124
				receive	697369	11226681
				send	232276	227342
				stream	239856017	7023755
Youtube	Video streaming	89032398	182437673	comment	7461	1475863
				react	10977588	9127
				stream	13727747	21328426
Twitter	Social media	400246136	377565909	tweet	60959014	448088
				retweet	4883487	3008042
				like	11018017	1016249
Spotify	Music streaming	7925388	21417369	view	50213	748038
				react	9595	313938
				stream	3877304	5825032
Web-Whatsapp	Instant messaging	20283379	18295343	send	3904	8604
				send-document	233149	3672268
				send-image	630462	1258854
Medium	Online articles	56968024	53287806	read	1548183	5259445
				react	288752	3895247
Lazada	E-commerce	91392976	173227857	view	45502056	4986109
				buy	2488849	1238459
				react	5087580	2104486
Shopee	E-commerce	49595474	65065681	view	4125964	1625425
				buy	2154626	209157
				react	4317924	204972
Reddit	Online forum	46048171	98659909	read	536632	491146
				react	61310	919485
				comment	18526	55008
Netflix	Video streaming	783922987	285698311	react	133953416	6290817
				stream	323175742	129265572

5. EXPERIMENTAL ANALYSIS

We conducted two sets of experiments to evaluate Grano-GT. The first set of experiments aims to evaluate the reliability of our traffic traces at the application name level, such as Facebook and Twitter. Although we are confident that our traffic traces are highly reliable at the application name level due to the nature of Algorithm 1 described previously, we ran our traffic traces through nDPI which served as the DPI engine to provide further validation. We chose nDPI because it can be seen as the defacto standard of DPI based on its performance described in [16]. However, nDPI only works on a finite list of application names and put less coverage on the majority of application services. Therefore, nDPI fails to recognize the application services in this paper.

Hence, our second set of experiments looks into how statistical testing can act as an alternative technique and verify the consistency of traffic traces at the application service level. As far as we are concerned, we are among the pioneer works that attempt to provide ground truth at the application service level which explains the lack of techniques for validation at this granularity level. For example, authors of MIRAGE created the ground truth for mobile application traffic, but put very little attention to the validation process of their ground truths [8]. Similarly, authors of Traffic Labeller used only the port-based method which is known to be highly restrictive to validate their ground truths [2].

5.1 VALIDATING THE APPLICATION NAME RELIABILITY USING NDPI

The algorithm running at the core of the *application isolator engine* ensures that Grano-GT produces a reliable ground truth at the application level. This is because the *application isolator engine* logs all the IP addresses coming from the target application server thus guaranteeing the extraction of pure traffic traces at the application name level. In this experiment, we validated the reliability of the traffic traces further by running them through nDPI and compared the results.

We compiled and set up nDPI (version 3.3.0-2232-942a71c7) on a Linux machine running Ubuntu 18.04 LTS before creating a script to pass traffic traces from all applications through nDPI. nDPI is an open-source DPI engine that detects various protocols and application names from traffic traces using multiple approaches such as payload based signatures, IP address blocks and host-based matching. Table 5 shows the bytes accuracies of nDPI calculated using Equation 2 for five application names.

$$Bytes\ accuracy = \frac{Total\ correctly\ classified\ bytes\ by\ nDPI}{Total\ bytes\ collected\ by\ GranoGT} \quad (2)$$

Table 5 Bytes accuracy of application name based on nDPI

Application name	UM-Net	HOME-Net
Facebook	94.4%	91.1%
Youtube	90.4%	1%
Twitter	96.2%	98.2%
Web-Whatsapp	100%	99.8%
Netflix	97.6%	99.3%

We had to exclude five of our applications which are Spotify, Medium, Lazada, Shopee and Reddit because their bytes accuracies were at 0% due to unavailability or outdated signatures in nDPI. For instance, signatures for Lazada and Shopee are unavailable because they are online shopping sites which are more common in South-East Asian countries and less known globally. Similarly, nDPI has yet to put coverage on Medium and Reddit while Spotify's signature is outdated.

Apart from outdated signatures, encrypted traffic also has an impact on the effectiveness of nDPI. nDPI labels encrypted traffic with its respective encryption protocols like TLS or utilizes other approaches, as mentioned previously. Based on Table 5, nDPI fails to achieve 100% accuracy, mostly due to its inability to detect the encrypted portion of the traffic. On the other hand, Youtube recorded only 1% accuracy in HOME-Net because it changed to using UDP protocol entirely for its video streaming service very recently and nDPI has yet to capture that change.

Hence, we can observe that the traffic traces captured by Grano-GT are reliable at the application name level. It also shows that using a DPI engine alone for building the ground truth presents its disadvantages such as unavailable and outdated signatures. Grano-GT managed to overcome these disadvantages by introducing the *application isolator engine*.

5.2 VERIFYING THE APPLICATION SERVICE USING THE KOLMOGOROV-SMIRNOV TEST

Validating the reliability of the application name is a straightforward process because there exist DPI tools that allow a particular benchmark for us to compare. However, the same process carries a slight problem when dealing with application service. Unlike nDPI which is commonly accepted as a benchmark, there is a severe lack of benchmarks for application service traces to compare.

To eliminate any research bias when using our ground truths, we need a technique that can compare our ground truths against some benchmark. We can perform a manual inspection of each packet to confirm the accuracy. The downside of manually inspecting each packet is that it is time-consuming and still introduces research bias. To tackle this problem, we look at the packet size distribution of the application service traffic. The findings in [37] showed that packets belonging to the same application protocol exhibit a similar profile in terms of packet size distribution. Using the same principle, we infer that the size distribution of the same application service should display similarities which leads us to the alternative technique to evaluate our ground truth at the application service level, which is the Kolmogorov-Smirnov (KS) Test [38].

The KS Test is a test for goodness of fit. It is a non-parametric and distribution-free test where it makes no assumption about the underlying distribution of data. It is useful in comparing whether a sample belongs to a reference probability distribution. Besides that, the KS Test has a two-sample variation where it can test whether two samples have the same distribution.

In essence, the KS Test calculates the maximum distance between the empirical distribution function of two samples. Equation 3 shows the definition of the empirical distribution function.

$$F(t) = \frac{number\ of\ elements\ in\ sample \leq t}{n} \quad (3)$$

Then, it compares the calculated maximum distance to a critical value before deciding on whether to reject or accept the null hypothesis. The critical value is defined in the table of critical values as noted in [38] where for a significance level of 0.05, the critical value, *D* is defined as follows:

$$D_{crit,0.05} = \frac{1.358}{\sqrt{n}} \tag{4}$$

As such, the Two-sample KS Test is a practical technique to verify our traffic traces at the application service level. The intuition of this experiment is that traffic traces from the same type of application service should portray similar packet size distribution. In other words, the maximum distance calculated should be lower than the critical value. For example, we calculated the KS values for 465 samples of Reddit-react traffic traces. Of these, 387 samples, or 83% recorded KS values lower than the critical value which indicates that the samples have similar packet size distribution. In contrast, traffic traces from different application services, regardless of its application names, should reflect a noticeable difference in its distribution.

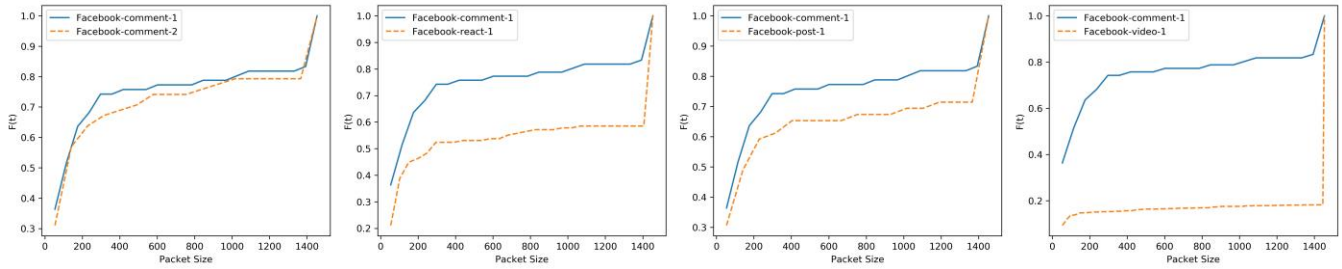

Figure 5a Comparison of cumulative distribution function values of different services within the same application

To illustrate, based on Figure 5a, we can observe that traffic traces from the same application service (i.e., Facebook-comment) show a small difference between the values of their empirical distribution function. However, Facebook-comment traffic shows a significant contrast when compared to other services although from the same Facebook application. For example, Facebook-react and Facebook-video displayed the most substantial difference between the function values. Meanwhile, Facebook-comment and Facebook-post traffic exhibit smaller differences because the nature of the service (i.e., commenting and posting on Facebook) is somewhat similar.

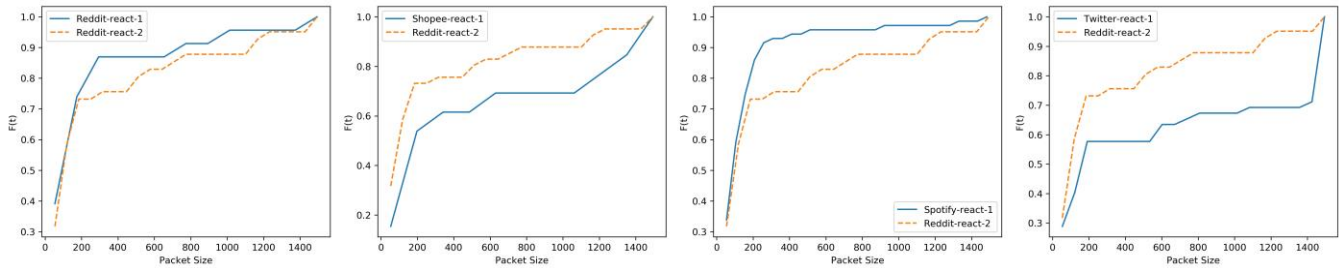

Figure 5b Comparison of cumulative distribution function values of similar services from different applications

On the other hand, Figure 5b shows a comparison of the function values between similar application services but from different applications. Intuitively, the graph of Reddit-react traffic displays a small difference between the function values as they represent the same type of application service. However, the function values of Reddit-react traffic are distinct when compared to the same service (i.e., react) from different applications. Although the KS Test only validates the goodness of fit instead of the absolute accuracy, which is of more interest, it reduces the risk of research bias by providing a technique to prove the distinction between traffic traces at the application service level.

6. DISCUSSION

In this paper, we presented Grano-GT which is a tool to build highly granular network traffic ground truth. Network traffic ground truth is critical to ensure a fair evaluation of network traffic classifiers as well as setting a performance benchmark. There are very few publicly available tools to build the ground truth as mentioned earlier in this paper. In addition, the majority of these tools put little attention on building the ground truth at the application service level.

Grano-GT attempts to address that issue by experimenting with collecting reliable and granular network traffic ground truth at the application name and service levels. However, since Grano-GT is among the pioneers in this area, there are a few fundamental issues and challenges that are worth noting. In this section, we discuss the current challenges and future works for guidance to researchers and interested parties.

6.1 Evaluating reliability at the application service level

One of the most crucial phases of any research works is evaluation. Evaluating the ground truth traffic at the application service level remains a challenge. This is because there is a severe lack of datasets at the application service level for benchmarking purposes. Hence, researchers need to come up with a reliable technique to evaluate the reliability of the traffic traces at the application service level. Nonetheless, researchers can consider a few existing techniques such as manual inspection by experts and DPI.

Manually inspecting the traffic traces will highly likely increase the chance of getting the correct evaluation. Even so, it is often the case that the manual inspection involves thousands of network packets, if not more. As a consequence, this technique becomes impractical as it takes a tremendous amount of time and effort. To overcome this issue, an automated technique like DPI is preferable.

DPI is widely used to evaluate the originating application of traffic traces. It compares the packet against a set of predefined signatures. It is highly accurate but presents one significant issue. The issue is that DPI requires constant updates to its signature bank. Moreover, application service signatures are unavailable in most DPI engines. For instance, in this paper, we are unable to use nDPI to evaluate the traffic traces at the application service level due to the unavailability of signatures. In the future, we should put more effort into creating signatures at the application service level to cope with the evolving trend of network traffic classification.

6.2 Collecting reliable ground truth with encrypted traffic

According to the recent Google Transparency Report [39], more than 90% of the traffic across Google is now encrypted. Encryption has quickly become a necessity due to rising concerns over user privacy. As such, encrypting the traffic helps in hiding user data from the public eye. Although hiding user data is an excellent move to protect user privacy, it poses a challenge to collect the network traffic ground truth because the encryption diminishes the effectiveness of the signature matching process. There are a few solutions that researchers can consider to work around the issue of traffic encryption.

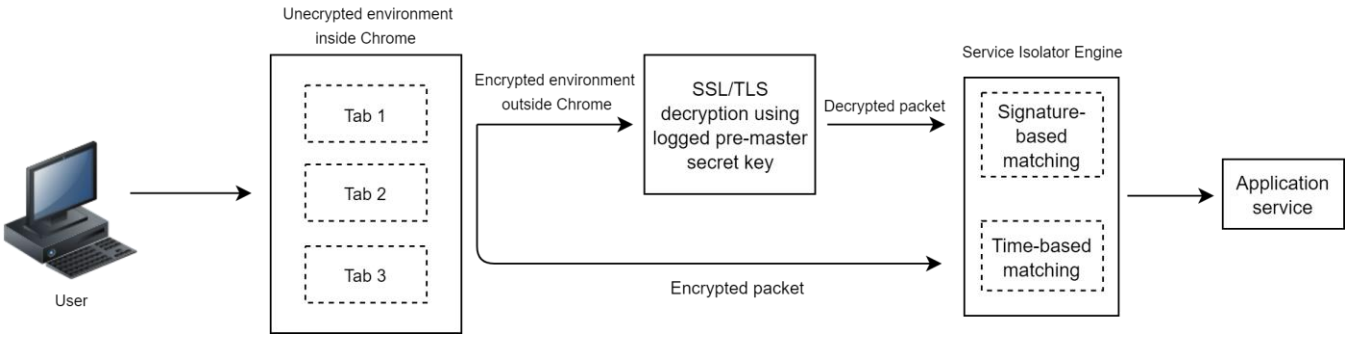

Figure 6 Grano-GT's approach to collect reliable ground truth with encrypted traffic

One of the most commonly used approaches is to introduce multiple techniques to detect the traffic besides signature matching alone. For example, based on Figure 6, we observe that the traffic remains unencrypted within the Chrome browser environment. As Grano-GT utilizes CDP in the Chrome browser environment, Grano-GT can view the traffic without any encryption and log the exact execution time of the application service upon detecting a matching signature. However, once the traffic exits the Chrome environment, the traffic gets encrypted. Hence, Grano-GT implements two techniques which are decrypting the traffic using the secret keys or using time-based matching. Grano-GT logs the secret keys from Chrome during the browsing session. Then, it utilizes the feature in tshark to decrypt the SSL/TLS traffic using the logged secret keys. If the secret keys are unable to decrypt the traffic due to limitations such as the type of protocols or ciphers used, Grano-GT switches to time-based matching. Grano-GT compares the difference between packet arrival times and the logged application service time to identify the targeted traffic without tampering with user privacy.

On the other hand, commercial solutions such as the Palo Alto Next-Generation Firewall offers traffic decryption to solve the problem. The firewall acts as a man-in-the-middle and intercepts the SSL request. Then, it returns the copy of the signed server certificate to the client thus allowing it to decrypt the traffic between the client and the server. Although this can potentially be seen as a breach in user privacy, Palo Alto claims to re-encrypt the traffic before pushing it back into the network [40]. Therefore, researchers need to put more consideration on

this issue because the portion of encrypted traffic will only keep on increasing in the future. The process of collecting network traffic ground truth should be able to maintain its reliability regardless of the state of encryption.

6.3 Publishing ground truth data without compromising privacy

Network traffic ground truth datasets are highly valuable for analysis and research. They allow researchers to evaluate and benchmark the performance of their proposed techniques, yet publicly available network ground truth datasets are scarce. Among the available datasets are by the Canadian Institute for Cybersecurity [41], WAND Network Research Group [7] and University of Brescia [42]. Despite that, the datasets are relatively outdated, with most of them being more than five years old.

The scarcity of publicly available network traffic ground truth datasets is primarily due to privacy concerns. Network traffic traces contain sensitive user data that can compromise privacy. The magnitude of privacy leaks can be devastating. For example, attackers can rebuild the network topology by using traffic traces [43]. This critical information exposes the network infrastructure to adversarial attacks. Besides that, attackers can also infer host behavior from the traffic which increases the risk of attacks.

As a result, most researchers are unable to publish their datasets without going through some anonymization process. Anonymizing network traffic traces is the process of removing identifiable relationships between two endpoints while ensuring data availability [44]. Authors in [44] presented a brief review of the typical traffic anonymity methods. For example, we can replace the IP addresses in the traffic traces with synthetic values. Besides that, we can also encrypt the IP address prefix [45]. There are also other common fields of a network packet that can undergo anonymization such as the MAC address, port number and timestamp.

Therefore, researchers should take all possible measures and considerations before releasing their datasets for public use. Future network traffic classification research may even consider moving the classification models to the traffic traces to overcome the need to make the data publicly available. As an example, Kaggle which is an online community for data scientists and machine learning practitioners [46], allows its participants to upload their machine learning models to a privately hosted data repository in a competition. As a result, participants in Kaggle competitions can evaluate their models without having full access to the data. This approach also takes inspiration from the trend in big data technology where moving data to a compute machine is computationally expensive. Instead, big data researchers move their portable compute models to the data to achieve their goals. Similarly, researchers in the network traffic classification domain can submit their classification models to the data owner and receive the results in return.

6.4 Extending usage beyond browser-based applications

In this paper, we proposed Grano-GT which is a browser-based tool to build highly granular network traffic ground truth. Grano-GT focuses on browser-based applications because it takes advantage of the Chrome Devtools Protocol to intercept application signatures. This is possible because the interception occurs at the application layer of the OSI model, thus making the traffic visible with minimal encryption. However, Grano-GT holds a limitation where it is currently unable to collect network traffic ground truth beyond browser-based applications (e.g. desktop and mobile applications). This is because discriminating network traffic between different application services from a single application beyond the browser is challenging without the appropriate traffic visibility.

Maintaining traffic visibility beyond the browser is a long-standing challenge. One of the commonly used techniques to gain traffic visibility for building ground truth is socket monitoring. The majority of ground truth tools utilize socket monitoring in their design [2, 3, 8]. While it is a reliable technique to build network traffic ground truth at the application-name level, it presents a challenge to implement it at the application-service level. The challenge is due to applications using the same socket to communicate throughout the session, thus making it difficult to distinguish between application services.

Another feasible technique is tagging the target packets with a special flag for identification. Szabo et al. proposed a tool to build network ground truth by marking the target packets with the first two characters of the application name in the IP Router Alert option field [47]. Similarly, Lizhi et al. embedded the application information into the TOS field of the IP packets [2]. This is a viable technique but it requires proper consideration to avoid degrading the performance of the network due to the increased size of the packets and as such, extending Grano-GT beyond browser-based applications remains as future work.

7. CONCLUSION

This paper presents the design, development and experimental evaluation of Grano-GT. Grano-GT reinvents the way researchers build network traffic ground truth by focusing on building highly granular ground truth. Grano-GT achieves this objective by implementing a series of cascading filters. The final output of Grano-GT is a collection of traffic traces at the application-name and service levels.

Experimental tests demonstrate the effectiveness of Grano-GT. At the application-name level, Grano-GT guarantees optimum reliability due to the *application isolator engine* that filters the application traffic based on logged IP addresses. We validated this claim by running the traffic traces through nDPI and recorded more than 95% average accuracy on both tested networks. Additionally, we used the

Kolmogorov-Smirnov Test to evaluate the reliability of the traffic traces captured at the application-service level. The results showed that Grano-GT managed to build network traffic ground truth with reliability. Besides that, Grano-GT achieved higher granularity levels (i.e. application-name and service) if compared to prior works such as mentioned in Section 2 of this paper.

We are currently working to extend Grano-GT beyond browser-based applications which will allow Grano-GT to build network traffic ground truth for other desktop applications such as Skype, Telegram and desktop games and even mobile applications. We believe Grano-GT can be useful for researchers to build their network traffic ground truth which is highly granular and reliable.

ACKNOWLEDGMENTS

This work was supported by the University Malaya Faculty Research Grants (GPF006D-2018) and Fundamental Research Grant Scheme under the Ministry of Education Malaysia (FRGS/1/2018/ICT03/UM/02/3).

REFERENCES

[1] F. Zaki, A. Gani, and N. B. Anuar, "Applications and use Cases of Multilevel Granularity for Network Traffic Classification," in *2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA)*, Langkawi, Malaysia, 28-29 Feb 2020: IEEE, pp. 75-79, doi: 10.1109/CSPA48992.2020.9068697.

[2] P. Lizhi, Z. Hongli, Y. Bo, C. Yuehui, and W. Tong, "Traffic Labeller: Collecting Internet traffic samples with accurate application information," *China Communications*, vol. 11, pp. 69-78, 2014, doi: 10.1109/CC.2014.6821309.

[3] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. C. Claffy, "GT," *ACM SIGCOMM Computer Communication Review*, vol. 39, p. 12, 2009, doi: 10.1145/1629607.1629610.

[4] P. A. Networks, "APP-ID Tech Brief - Palo Alto Networks," 2015.

[5] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, 2016: SciTePress, pp. 407-414, doi: 10.5220/0005740704070414.

[6] K. F. Yu and R. E. Harang, "Machine learning in malware traffic classifications," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, 2017: IEEE, pp. 6-10, doi: 10.1109/MILCOM.2017.8170769.

[7] W. N. R. Group. "WITS: Waikato Internet Traffic Storage." https://wand.net.nz/wits/ (accessed 1 May, 2020).

[8] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "MIRAGE: Mobile-app Traffic Capture and Ground-truth Creation," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, 10-12 Oct. 2019 2019, pp. 1-8, doi: 10.1109/CCCS.2019.8888137.

[9] J. Kampeas, A. Cohen, and O. Gurewitz, "Traffic Classification Based on Zero-Length Packets," *IEEE Transactions on Network and Service Management*, vol. 15, pp. 1049-1062, 2018, doi: 10.1109/TNSM.2018.2825881.

[10] ChromeDevTools. "Chrome DevTools Protocol Viewer." https://chromedevtools.github.io/devtools-protocol/ (accessed 21 February, 2020).

[11] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "nDPI: Open-source high-speed deep packet inspection," in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2014: IEEE, pp. 617-622, doi: 10.1109/IWCMC.2014.6906427.

[12] A. Reinhart, V. Ventura, and A. Athey, "Detecting changes in maps of gamma spectra with Kolmogorov–Smirnov tests," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 802, pp. 31-37, 2015/12/01/ 2015, doi: https://doi.org/10.1016/j.nima.2015.09.002.

[13] D. Zhao, L. Bu, C. Alippi, and Q. Wei, "A Kolmogorov-Smirnov Test to Detect Changes in Stationarity in Big Data," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14260-14265, 2017/07/01/ 2017, doi: https://doi.org/10.1016/j.ifacol.2017.08.1821.

[14] A. Dainotti, A. Pescape, and K. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, pp. 35-40, 2012, doi: 10.1109/MNET.2012.6135854.

[15] J. Yan, "A Survey of Traffic Classification Validation and Ground Truth Collection," in *2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2018: IEEE, pp. 255-259, doi: 10.1109/ICEIEC.2018.8473477.

[16] V. Carela-Español, T. Bujlow, and P. Barlet-Ros, "Is Our Ground-Truth for Traffic Classification Reliable?," in *International Conference on Passive and Active Network Measurement*, Cham, Switzerland, 2014: Springer, Cham, pp. 98-108, doi: 10.1007/978-3-319-04918-2_10.

[17] I. A. N. A. (IANA). "Service Name and Transport Protocol Port Number Registry." https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (accessed 1 May, 2020).

[18] 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, 2020/03/15/ 2020, doi: https://doi.org/10.1016/j.jnca.2020.102538.

[19] A. W. Moore and K. Papagiannaki, "Toward the Accurate Identification of Network Applications," in *Proc. Passive and Active Measurement Workshop (PAM2005)*, ed: Springer, Berlin, Heidelberg, 2005, pp. 41-54.

[20] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," presented at the Proceedings of the 2006 SIGCOMM workshop on Mining network data, Pisa, Italy, 2006. [Online]. Available: https://doi.org/10.1145/1162678.1162679.

[21] X. Yun, Y. Wang, Y. Zhang, and Y. Zhou, "A Semantics-Aware Approach to the Automated Network Protocol Identification," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 583-595, 2016, doi: 10.1109/TNET.2014.2381230.

[22] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust Network Traffic Classification," *IEEE/ACM Transactions on Networking*, vol. 23, pp. 1257-1270, 2015, doi: 10.1109/TNET.2014.2320577.

[23] S. Alcock and R. Nelson, "Libprotoident: Traffic Classification Using Lightweight Packet Inspection," in "WAND Network Research Group, Technical Report," 2012. [Online]. Available: https://research.wand.net.nz/software/libprotoident.php

[24] A. Hajjar, J. Khalife, and J. Díaz-Verdejo, "Network traffic application identification based on message size analysis," *Journal of Network and Computer Applications*, vol. 58, pp. 130-143, 2015, doi: 10.1016/J.JNCA.2015.10.003.

[25] X. Xiao, R. Li, H.-T. Zheng, R. Ye, A. KumarSangaiah, and S. Xia, "Novel dynamic multiple classification system for network traffic," *Information Sciences*, vol. 479, pp. 526-541, 2019, doi: 10.1016/J.INS.2018.10.039.

[26] S. E. Gómez, B. C. Martínez, A. J. Sánchez-Esguevillas, and L. Hernández Callejo, "Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal," *Computer Networks*, vol. 127, pp. 68-80, 2017, doi: 10.1016/J.COMNET.2017.07.018.

[27] J.-q. Lu, X. Tian, X.-h. Huang, Y.-j. Wu, Y. Ma, and Z.-s. Su, "Flowsing: a multi-agent based offline ground-truth generator," *The Journal of China Universities of Posts and Telecommunications*, vol. 18, pp. 106-111, 2011/09/01/ 2011, doi: https://doi.org/10.1016/S1005-8885(10)60201-4.

[28] M. Canini, W. Li, A. W. Moore, and R. Bolla, "GTVS: Boosting the Collection of Application Traffic Ground Truth," in *Traffic Monitoring and Analysis*, Berlin, Heidelberg, M. Papadopouli, P. Owezarski, and A. Pras, Eds., 2009// 2009: Springer Berlin Heidelberg, pp. 54-63.

[29] J. Khalife, A. Hajjar, and J. Diaz-Verdejo, "A multilevel taxonomy and requirements for an optimal traffic-classification model," *International Journal of Network Management*, vol. 24, pp. 101-120, 2014, doi: 10.1002/nem.1855.

[30] "tshark - The Wireshark Network Analyzer 3.2.2." https://www.wireshark.org/docs/man-pages/tshark.html (accessed 1 May, 2020).

[31] StatCounter. "Desktop Browser Market Share Worldwide." StatCounter. https://gs.statcounter.com/browser-market-share/desktop/worldwide (accessed 11 May, 2020).

[32] fate0. "PyChrome: A Python Package for the Google Chrome Dev Protocol." https://github.com/fate0/pychrome (accessed 1 April, 2020).

[33] H. Oudah, B. Ghita, T. Bakhshi, A. Alruban, and D. J. Walker, "Using Burstiness for Network Applications Classification," *Journal of Computer Networks and Communications*, vol. 2019, p. 10, 2019, Art no. 5758437, doi: 10.1155/2019/5758437.

[34] T. Lee. "App Annie data: Lazada was Southeast Asia's top shopping app by active users in 2018." https://www.techinasia.com/app-annie-data-lazada-southeast-asias-top-shopping-app-active-users-2018 (accessed 1 June, 2020).

[35] 42matters. "Malaysia App Market Statistics in 2020 for Android." https://42matters.com/malaysia-app-market-statistics (accessed 1 June, 2020).

[36] Advertising.my. "Top Mobile Apps in Malaysia." https://www.advertising.com.my/top-mobile-apps-in-malaysia (accessed 1 June, 2020).

[37] C.-N. Lu, C.-Y. Huang, Y.-D. Lin, and Y.-C. Lai, "High performance traffic classification based on message size sequence and distribution," *Journal of Network and Computer Applications*, vol. 76, pp. 60-74, 2016, doi: 10.1016/J.JNCA.2016.09.013.

[38] F. J. Massey, "The Kolmogorov-Smirnov Test for Goodness of Fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68-78, 1951/03/01 1951, doi: 10.1080/01621459.1951.10500769.

[39] Google. "Google Transparency Report." https://transparencyreport.google.com/https/overview?hl=en (accessed 18 May, 2020).

[40] P. A. Networks, "PAN-OS Administrator's Guide Version 9.0," Santa Clara, CA, 2020. [Online]. Available: https://docs.paloaltonetworks.com/pan-os/9-0/pan-os-admin/decryption.html
[41] C. I. f. Cybersecurity. "Datasets." https://www.unb.ca/cic/datasets/index.html (accessed 1 April, 2020).
[42] UNIBS. "UNIBS: Data sharing." http://netweb.ing.unibs.it/~ntw/tools/traces/ (accessed 1 April, 2020).
[43] J. King, K. Lakkaraju, and A. Slagell, "A taxonomy and adversarial model for attacks against network log anonymization," presented at the Proceedings of the 2009 ACM symposium on Applied Computing, Honolulu, Hawaii, 2009. [Online]. Available: https://doi.org/10.1145/1529282.1529572.
[44] X. Tian, Y. Wang, Y. Zhu, Y. Sun, and Q. Liu, "De-anonymous and Anonymous Technologies for Network Traffic Release," in *International Conference on Applications and Techniques in Information Security*, Auckland, New Zealand, 2017: SpringerLink, pp. 193-200, doi: 10.1007/978-981-10-5421-1_16. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-10-5421-1_16
[45] X. Jun, F. Jinliang, M. H. Ammar, and S. B. Moon, "Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, 12-15 Nov. 2002 2002, pp. 280-289, doi: 10.1109/ICNP.2002.1181415.
[46] K. Inc. "Kaggle." https://www.kaggle.com/ (accessed 11 September, 2020).
[47] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó, "On the Validation of Traffic Classification Algorithms," in *Passive and Active Network Measurement*, Berlin, Heidelberg, M. Claypool and S. Uhlig, Eds., 2008// 2008: Springer Berlin Heidelberg, pp. 72-81.