Your Phone is My Proxy: Detecting and Understanding Mobile Proxy Networks

Xianghang Mi*
University at Buffalo
xmi@buffalo.edu

Siyuan Tang
Indiana University Bloomington
tangsi@iu.edu

Zhengyi Li
Indiana University Bloomington
zl11@iu.edu

Xiaojing Liao
Indiana University Bloomington
xliao@indiana.edu

Feng Qian
University of Minnesota Twin Cities
fengqian@umn.edu

XiaoFeng Wang
Indiana University Bloomington
xw7@indiana.edu

***Abstract*—Residential proxy has emerged as a service gaining popularity recently, in which proxy providers relay their customers' network traffic through millions of proxy peers under their control. We find that many of these proxy peers are mobile devices, whose role in the proxy network can have significant security implications since mobile devices tend to be privacy- and resource-sensitive. However, little effort has been made so far to understand the extent of their involvement, not to mention how these devices are recruited by the proxy network and what security and privacy risks they may pose.**

In this paper, we report the first measurement study on the mobile proxy ecosystem. Our study was made possible by a novel measurement infrastructure, which enabled us to identify proxy providers, to discover proxy SDKs (software development kits), to detect Android proxy apps built upon the proxy SDKs, to harvest proxy IP addresses, and to understand proxy traffic. The information collected through this infrastructure has brought to us new understandings of this ecosystem and important security discoveries. More specifically, 4 proxy providers were found to offer app developers mobile proxy SDKs as a competitive app monetization channel, with $50K per month per 1M MAU (monthly active users). 1,701 Android APKs (belonging to 963 Android apps) turn out to have integrated those proxy SDKs, with most of them available on Google Play with at least 300M installations in total. Furthermore, 48.43% of these APKs are flagged by at least 5 anti-virus engines as malicious, which could explain why 86.60% of the 963 Android apps have been removed from Google Play by Oct 2019. Besides, while these apps display user consent dialogs on traffic relay, our user study indicates that the user consent texts are quite confusing. We even discover a proxy SDK that stealthily relays traffic without showing any notifications. We also captured 625K cellular proxy IPs, along with a set of suspicious activities observed in proxy traffic such as ads fraud. We have reported our findings to affected parties, offered suggestions, and proposed the methodologies to detect proxy apps and proxy traffic.

*Corresponding author

Network and Distributed Systems Security (NDSS) Symposium 2021
21-24 February 2021
ISBN 1-891562-66-5
https://dx.doi.org/10.14722/ndss.2021.24008
www.ndss-symposium.org

I. Introduction

Residential proxy service is an emerging network infrastructure where customers' network traffic is relayed through millions of residential IP proxies. For example, Oxylabs [37] claims to maintain 32M residential IP proxies and Smartproxy has 10M [43]. Different from traditional web proxies (e.g., VPNs) which are typically deployed in data center networks, residential proxies reside in residential or cellular networks, and therefore can offer better anonymity and evasiveness for proxy customers [32]. Those proxies can be personal computers, mobile devices, and even IoT equipment [44] [22]. However, such a network infrastructure could be abused by cybercriminals to cover their communication with the targets. Indeed, a prior study [32] reports that ad clicking, fast fluxing, and other malicious activities are observed in the traffic relayed by residential proxy services. Another report [46] shows that a residential proxy service was utilized to issue fraudulent transactions on a large scale.

Mobile Devices as Proxy Peers. In recent years, major proxy service providers started to offer proxy SDKs to popular mobile systems (e.g., Android, iOS), and advertise their SDKs as a monetization channel. Once app developers integrate a proxy SDK into their apps, following the integration guidelines from proxy service providers, mobile devices running those apps (which we call *proxy apps*) will become *proxy peers*. Such mobile proxy SDKs are promoted to benefit both proxy service providers and app developers – the app developers get paid for distributing proxy SDKs to mobile devices, which in the meantime enables the proxy providers to offer a large mobile proxy network to their customers. However, the security implications of the proxy SDKs, both to the mobile users who install the apps and to the Internet community, have not been fully understood. Particularly, such *proxy apps* could be considered illicit, as stated in Google Play's developer policy [17]: *"Apps that facilitate proxy services to third parties may only do so in apps where that is the primary, user-facing core purpose of the app."*. Thus in our research, we focus on such *mobile proxy peers*, which are characterized by relaying traffic through mobile proxy SDKs, to provide an insider view (e.g., proxy app) of mobile proxy networks and to understand the impact of the networks on innocent mobile users.

Previous research [32] has looked into residential proxy services with a focus on their architecture and scale. Specifically, it has also explored the recruitment of proxy peers with some potentially unwanted programs (PUP) revealed to relay traffic on Windows platforms. However, this is achieved by collaboration with an anonymous security company and no detection methodology is proposed to systematically identify proxy peers and proxy SDKs on a large scale. Besides, while coarse-grained traffic logs of those PUPs were analyzed in [32] to understand the usage of residential proxy services, it is unclear whether those traffic logs are indeed relaying traffic since non-proxy components such as advertisement SDKs can co-locate in a given PUP and thus contribute to the resulting traffic. No solid methodology is available to exclude such kind of non-proxy traffic, which undermines the findings in [32]. Furthermore, little has been done so far to understand the mobile proxy ecosystem of such services. One may ask: who are the stakeholders of the ecosystem and how do they interact with each other? Particularly to what extent has the ecosystem influenced app monetization, and how important is it to residential proxy services? Also unclear is whether mobile device owners are aware of the relaying behaviors through their devices, to what extent they are willing to allow such operations, and whether their on-device resources are abused by the traffic relay, e.g., incurring high cellular data consumption. Answers to these questions are critical for demystifying the mobile proxy ecosystem, which could potentially help find an effective way to detect and mitigate its security risks.

Our Study and its Challenges. In this paper, we report the first measurement study to characterize the security and privacy risks of utilizing mobile devices as proxy peers through mobile SDKs. Our study faces several major challenges.

First, we need to discover mobile proxy SDKs and mobile proxy apps “in the wild”. However, little pubic information about such SDKs is available and it can only be acquired from residential proxy providers, who often put restrictions in place (e.g., background check to make sure that request parties are indeed app developers). Thus, it is very difficult to gather comprehensive information on mobile proxy SDKs. Further complicating the identification effort is the observation that some mobile proxy SDKs are heavily customized when provided to app developers.

Second, given a proxy app, no existing techniques can determine whether it integrates a proxy SDK or not, not to mention any attempt to identify such proxy apps on a large scale. Note that automated techniques serving such a purpose are important, not only for our measurement study but also for controlling the risks of the proxy apps, which have been classified by Google Play as potential device and network abuse [17] since June 2019.

Third, it is far from trivial to find out whether a mobile proxy SDK is involved in illicit traffic relay activities. To achieve that, we need an infrastructure to automatically run proxy apps, trigger relaying behaviors, and capture the resulting network traffic. Then, a solution should be in place to properly identify the traffic produced by mobile proxy SDKs among other traffic incurred by the host proxy apps, the OS, and other third-party libraries.

Fourth, from the end users’ perspective, understanding mobile device owners’ attitudes towards mobile proxies is also non-trivial, and it requires strategically designed user studies. We also need well-designed experiments to profile the potential security implications posed by mobile proxies to mobile devices and their owners, e.g., abusing on-device critical resources.

In our research, we addressed the above challenges through a suite of effective techniques, which enabled us to perform a large-scale study to understand the security implications of mobile proxy networks in the wild. Specifically, we develop a novel framework for semi-automated discovery of mobile proxy SDKs from commercial residential proxy services (§III-C). The key idea is that since proxy peers always communicate with the service provider, we are able to utilize passive DNS (PDNS) data to identify potential proxy programs and further recover from them mobile proxy SDKs through program analysis. The recovered mobile proxy SDKs are then be used to generate robust signatures to find Android proxy apps on a large scale. Next, the detected proxy apps are efficiently executed on a custom proxy app profiling platform developed by us. The platform captures relaying traffic (with other traffic) in a configurable execution environment with tunable settings such as the battery life, the network type as detailed in §III-D. We then devise a robust relaying traffic identification method, which effectively separates relaying traffic from other irrelevant traffic (§III-E). Furthermore, to obtain a complete picture of the mobile proxy ecosystem, we also launch a user study to understand how willingly end users would allow traffic relaying to occur (in exchange for free services or fewer in-app ads), and how well normal users can understand the traffic relaying agreement. Note that we have taken strict measures to minimize potential ethical risks. We filed several IRB applications for this study, and all of them have been approved by our institution (§III-F).

Findings. Using the above framework, we analyzed 9 leading residential proxy services and captured 625K unique cellular proxy IPs (as part of 8M residential proxy IPs) in 4 months, along with 1,701 unique Android APKs (963 Android apps) that integrate mobile proxy SDKs from the four proxy providers including MonkeySocks [33], Luminati [28], Oxylabs [37], and IPninja [23]. Our key findings are as follows.

First of all, we discovered an app monetization channel never reported in the security community before: integrating mobile proxy SDKs with mobile apps. It is found to be very profitable for app developers, with a monthly payment of $50K per 1M MAU (monthly active users) per app on average as in June 2020. We discovered 1,701 unique Android APKs integrating proxy SDKs. Among them, 1,387 (963 Android apps) have circumvented app vetting and been published in Google Play. Also, 680 (48.43%) of those APKs were flagged by at least five antivirus engines as malicious programs in different categories. Examples include adware, PUP (potentially unwanted program), trojan, fake app, virus, and clicker. This may explain why 86.60% of the 963 Android proxy apps have been removed from Google Play by Oct 2019. What’s more, the mobile proxy SDKs were found to consume GBs of WiFi data and tens of MBs of cellular data every day, and we observed that 26 Android apps integrate multiple proxy SDKs. Also, our user study indicates that the texts of user agreements of the four SDKs are ambiguous about their traffic relay behaviors and tend to cause misunderstanding among users. We even found a previously unknown proxy SDK (not belonging to the above four) stealthily relays network traffic without user agreement at all.

Then, regarding proxy IPs and proxy traffic, our analysis reveals that suspicious advertisement frauds contribute most to relaying traffic, followed by search engine, traveling, shopping, social network, and email services. We have also captured over 8M residential IPs, including 625K cellular IPs used by proxy peers. Although most of them are clean (e.g., only 0.44% of cellular IPs blacklisted), we did find that 46 IPs were involved in large-scale botnets, such as Hajime and Mirai, actively distributing attack payloads.

Contributions. The contributions of the paper are as follows.

• We report the first in-depth study on mobile devices as proxy peers, an elusive app monetization method with few details publicly available. Our findings highlight its security and privacy implications.

• We developed novel techniques for identifying mobile SDKs and Android proxy apps, profiling their behaviors, and analyzing the traffic they relay. Our techniques can be integrated into a system for detecting and monitoring Android proxy apps, as well as for mitigating their potential security threats.

• More importantly, our study brought to light a set of novel and inspiring findings, especially those highlighting the security and privacy impacts of this mobile proxy ecosystem on mobile users, app developers, app stores, and online services.

Responsible Disclosure. We have reported our findings to Google (Anti-Abuse and Google Play Protect teams), who has acknowledged the importance of the discoveries and appreciated our effort. Through a series of emails and meetings, we have also shared with them our methodologies, detection signatures, and detected samples. Besides, we have responsibly communicated our findings to relevant proxy service providers and got their acknowledged to various extents. Some of them are working to mitigate the identified security and privacy issues. More details can be found in §VI.

II. Overview

In this section, we present an overview of *the ecosystem of mobile proxy services* wherein residential proxy networks are built upon mobile SDKs and mobile apps. Particularly, our study reveals different stakeholders of the ecosystem and their interactions such as SDK promotions and app monetization.

A. The Ecosystem of Mobile Proxy Services

Figure 1 illustrates the mobile proxy service ecosystem. At the center is *proxy providers* that serve *proxy customers* through mobile/residential proxy as a service. These customers subscribe to the services and relay their traffic through the proxy networks where mobile devices are used as exit nodes (*Proxy Peers*). The proxy customers have no knowledge or control of the proxy peers, as proxy peers are hidden behind the gateway servers and only the gateway servers are visible to the proxy customers.

Unlike traditional proxy services, mobile proxy providers build up their proxy network through two unique channels.

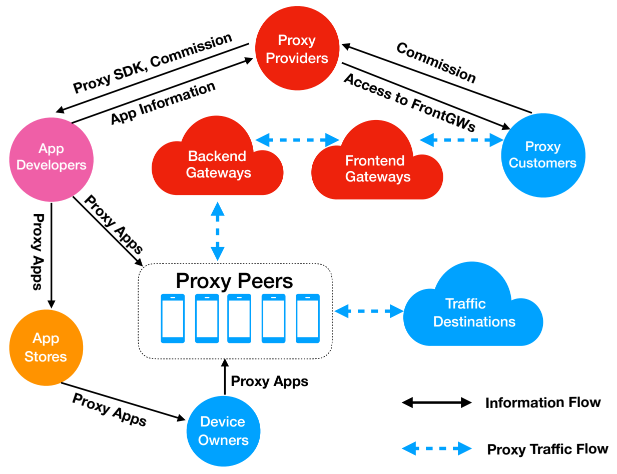


Fig. 1: The Ecosystem of Mobile Proxy Services.

TABLE I: Proxy SDKs from different proxy providers.

Proxy Provider	Supported Platforms
Luminati	Android, Windows, Mac OS
MonkeySocks	Android
Oxylabs	Android
IPninja	Android, Windows, Chrome Extension

One is to distribute their self-operated mobile apps to mobile devices. One such example is Luminati, which has control over apps such as org.hola.gpslocation and org.hola.va. The other more scalable channel is to provide *Proxy SDKs* on popular mobile platforms such as Android to monetize 3rd-party apps. Among all 38 proxy providers in our study, four were found to have proxy SDKs as listed in Table I.

App Developers can interact with the proxy providers to integrate the proxy SDK into their mobile apps. Once an integrated app is published, the revenue is calculated based on the number of installations and the scale of the active user base. Note that app developers typically have no or little control over a proxy SDK's logic such as how much traffic to relay.

Proxy Apps. To serve as the exit nodes, mobile proxy peers run *proxy apps* that are integrated with the proxy SDK. While proxy apps may ask device owners to agree with their privacy terms, it is difficult for the owners to learn that the app will relay traffic through their devices (proxy peers), as shown in our study (§IV-B). Even when device owners are aware of the relaying behaviors, they typically have no control over the relaying behaviors in terms of how much traffic to relay, what traffic to relay, and when to relay.

B. Promotion & Monetization of Proxy SDKs

Promotion of Proxy SDKs. We found in our research that all four proxy providers promote their SDKs in a very low-profile but precise manner. Interestingly, instead of utilizing online ads to promote themselves, they directly contact app developers through emails or other unsolicited methods. In particular, there are some online discussions among app developers regarding their experience of being reached out by the proxy providers [1] [12] [41].

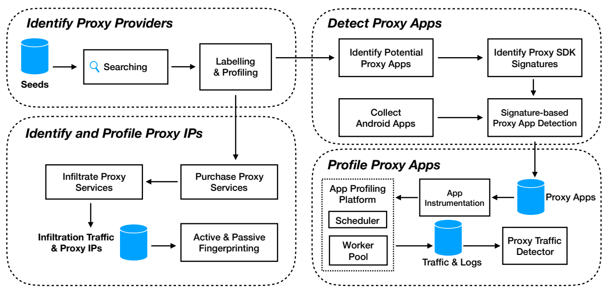


Fig. 2: Overview of Measurement Methodology.

Pricing Policy. We also looked into different proxy providers' policies for the app developers to monetize the integration of their proxy SDKs. We found that these providers tend to offer similar prices, while their commissions appear to be increasingly attractive over time. For example, in June 2019, the commission (across all four providers) was $300 for 10K users every month, in other words, $30K/1M users every month, but since September 2019, it has been raised to $500 for 10K users every month, or $50K/1M users every month. Interestingly, such a price change was quickly followed by most proxy providers, indicating their competitive nature. We also discovered that the commission varies by platforms and app user locations. An app developer may get more revenue from a mobile proxy peer (e.g., $0.025/month per user) compared to a PC proxy peer (e.g., $0.02/month per user) in certain countries.

Proxy SDKs vs. Other Popular Monetization Options. Our research shows that proxy SDK monetization has advantages over the other two popular app monetization options including ads and data monetization (collecting in-app data from apps running on mobile devices). First, proxy SDKs are a better choice for inactive apps. An app integrating a proxy SDK only needs to run as a background service or execute periodic tasks to generate revenue, incurring negligible user experience overhead. This is very different from ad SDKs, which require apps to run in foreground to display ads to users, with the revenue depending on how frequently an app is used. Second, proxy SDKs generate higher revenues compared to data monetization. For example, Luminati offers $50K per 1M MAU (monthly active users), while a popular data monetization SDK provider AppGrow only offers up to $6.6K for 1M MAUs [7].

III. Methodology

In this section, we present our measurement infrastructure as shown in Figure 2. We first explain how to find previously unknown proxy providers as detailed in §III-A, and then how to identify proxy IPs in §III-B. Further §III-C details how we detect proxy apps with high confidence. Then §III-D and §III-E describe how we profile those detected proxy apps leveraging a set of techniques including app instrumentation, dynamic analysis, and traffic analysis. Lastly §III-F discusses the methodology limitations and how we address potential ethical concerns.

A. Identify Proxy Providers

We started with the identification of residential proxy providers. Specifically, using proxy provider websites reported in a prior study [32] as "seeds", we then recursively collected search results from similar website lookup services (Alexa Find Similar Sites [2] and Similar Web [42]) until no new website is returned. After that, we manually validated each website to confirm that it indeed offers residential or mobile proxy services.

To this end, we collected 38 proxy providers (see Appendix IX-B) – more than doubling the findings made by the most recent study [32] in late 2018. Among these 38 proxy providers, many are large and previously unknown proxy providers such as Oxylabs [37] (claiming to have 32M+ residential IPs and 2M+ data center IPs) and Smartproxy [43] (claiming to have 10M+ IPs). Most importantly, four of them support mobile proxy: Oxylabs, Luminati, MonkeySocks, and IPninja. For example, Luminati [28] claims that all mobile IPs are cellular IPs. Besides this, we find some other new trends in the services provided by the 38 providers: (1) IPv6 proxy peers are offered by some providers [9]; (2) some proxy providers [34] are found to collaborate with ISPs to monetize their idle network resources by relaying traffic; (3) some providers pay exit nodes through blockchain-based currency [26]; (4) some providers (e.g., Intoli [21]) offer "add-on" services such as helping proxy customers render retrieved HTTP response in providers' headless browsers to circumvent websites' bot detection.

B. Capture IP Addresses of Peer Proxies

Among the 38 proxy providers, we selected a representative subset of 9 providers considering their scale, reputation, prices, as well as whether they have mobile proxy SDKs. Most of the selected providers claim to have the largest pool of proxy peers on the order of millions, and they are frequently recommended by proxy customers in various forums. Also some of them (e.g, Luminati, Geosurf) have been studied in previous works [32]. As shown in Table II, we purchased from 7 providers for at least one-month subscription, along with free trials of another 2 providers, due to our budget constraint. Table II lists also the cost of purchasing these services as well as the period during which we carried out the respective service infiltration. To infiltrate these proxy providers, we adopted a "traffic milking" strategy [32] wherein web clients under our control send crafted infiltration probes through the proxy networks towards our web servers (mpaas.site). In this way, public IP addresses of the exit nodes (proxy peers) can be observed. To capture both IPv6 and IPv4 peers, we assigned our web server with both IPv6 and IPv4 addresses and configured corresponding A and AAAA records in our authoritative DNS server. Once a proxy peer was detected, we performed active fingerprinting by sending network probes to its public IP address and utilized the responding network banners to infer the proxy peer's device type with Nmap [35]. We also deployed p0f [38] on our web server to passively fingerprint the OS type for each observed proxy peer. Note that this device/OS fingerprinting process was approved and guided by our institute's IRB (see details in §III-F).

Leveraging the above infiltration system, we infiltrated 9 proxy providers as listed in Table II. Overall, we captured 8,188,438 IPs (7,181,557 IPv4, and 1,006,881 IPv6), through 42M successful infiltration probes. Further, 624,989 IPv4 addresses (8.70% of all IPv4) are found to be cellular IPs by

TABLE II: Proxy providers selected for service infiltration.

Proxy Provider	Cost ($)	Period	Days
Luminati	500	05/29/2019 - 06/30/2019	33
GeoSurf	300	05/28/2019 - 07/28/2019	62
ProxyRack	30	07/11/2019 - 08/10/2019	31
Oxylabs	600	06/12/2019 - 07/25/2019	38
Smartproxy	225	04/23/2019 - 07/30/2019	92
COSMO:PROXY	100	04/23/2019 - 06/28/2019	63
Storm Proxies	35	07/11/2019 - 08/10/2019	31
Intoli	Free Trial	04/23/2019 - 04/24/2019	2
Netnut	Free Trial	04/29/2019 - 04/30/2019	2

IPinfo [24]. However, we do not know how many IPv6 proxies are cellular IPs as IPinfo's cellular dataset fails to cover IPv6.

C. Detect Proxy Apps

Here, we aim to detect the programs serving as proxy apps for those aforementioned proxy providers. To our best knowledge, we are the first to explore this problem space.

Assumptions. As shown in Figure 1, proxy peers need to contact proxy gateway(s) to relay traffic. Therefore, if a program is found to have communicated with some proxy gateway, we see a good indicator that it is likely a proxy app. This assumption underlies our detection methodology.

Identify Gateway Addresses. A prerequisite of our detection approach is to collect proxy gateway addresses (domain names). For this purpose, we take the following "snowballing" approach to obtain them. We first get a single "seed" domain for each proxy provider, and then extend the set of proxy domains using passive DNS data. Specifically, we look up IP addresses using each domain to discover more domain names by performing reverse-lookups on these addresses. We repeat these steps until no new domain or IP is found.

This seed-extension process turns out to be more complicated than it appears to be, as we discovered that those proxy providers may have their domains resolved to an IP address shared with some irrelevant domain names. For example, in Nov 2019, *proxyrack.com* is resolved to 104.26.3.97, which, however, is reversely resolved to irrelevant domains such as *delight.fit*. To address this problem, we manually picked out high-confidence domains as follows: a domain is selected only if it is semantically similar to the "seed" domain. Taking Luminati as an example, its gateway domains include *luminati.io* (the seed domain name), *luminatinet.com*, *lum-sdk.io*, *luminati-china.io*, and *lum-cn.io*, all of which demonstrate some level of connections to Luminati. This selection process was performed manually in our study given the relatively low workload of inspecting only hundreds of domain candidates.

Identifying Apps Linked to Proxy Providers. We searched the discovered gateway addresses of proxy providers on popular threat intelligence platforms including VirusTotal [48] and Hybrid Analysis [20]. Reports returned from these platforms include programs observed to be related to a given domain. Here we consider a program to be linked to a proxy provider given at least one of the following two observations: the program embeds in its payload the gateway addresses of the proxy provider (called referrer) or it communicates with the provider. By Nov 20, 2019, for all 38 proxy providers, we found 2,939 such programs, with most of them being Android (56.92%) or Windows apps (20.52%). Also, most of these Windows and Android programs are linked to the top 4 proxy providers offering proxy SDKs, as listed in Table III. Another interesting observation is that more such linked programs show up over time, from 1,810 by Feb 2019 to 2,939 by Nov 2019, with disproportional growth of Android apps, from 49.78% to 56.92% during that period. Note that the apps linked to proxy providers may not integrate their SDKs. Examples include browser apps and security apps that maintain lists of domain names containing proxy gateway addresses. So in order to find those carrying the proxy SDKs, we move on to identify a set of SDK signatures as elaborated below.

TABLE III: Windows/Android programs linked to proxy providers, through referrer (Ref) or communication (Comm).

Provider	Android			Windows		
	Ref	Comm	Linked	Ref	Comm	Linked
Luminati	547	612	1,108	257	0	257
MonkeySocks	154	115	269	1	0	1
IPninja	90	0	90	3	0	3
Oxylabs	125	0	125	10	0	10
Overall	983	741	1,673	589	14	603

Identify Proxy SDK Signatures. Given the linked Android apps discovered, we further located proxy SDKs and collected their signatures. Our key observation is that proxy SDKs of different versions can be fingerprinted by a set of robust signatures including URLs, Java namespaces, as well as Android services and broadcast receivers. For instance, many Luminati's proxy SDKs share the same namespace *io.topvpn.vpn_api*, a set of URLs (e.g., *perr.lum-sdk.io*, and *clientsdk.lum-sdk.io*), and several Android Services and BroadcastReceivers (e.g., *io.topvpn.vpn_api.bcast_recv*, *io.topvpn.vpn_api.svc*). Also, proxy SDKs are observed to keep evolving, which can potentially invalidate some SDK signatures. Therefore, a signature may only cover a subset of SDK versions while a specific SDK version may not match all signatures but only a subset. Still taking Luminati's proxy SDK as an example, its latest SDK versions have replaced package name *io.topvpn.vpn_api* with *io.lum.sdk*. However, URLs such as *clientsdk.lum-sdk.io* are still there.

Hence, we designed and implemented an extractor to automatically retrieve signature candidates from the linked Android apps. Specifically, for each proxy provider, the signature candidate extractor unpacks linked Android apps to identify the Java packages where the corresponding gateway addresses are located. From each such Java package, our approach then retrieves all IPs and domains it contains, together with its Android services and broadcast receivers. Given these signature candidates, robust signatures were selected out through a combined process involving app instrumentation and executing the linked apps, as elaborated in §III-D. In the meantime, we contacted all proxy providers to seek their mobile proxy SDKs. Two of them, Luminati and IPninja, gave us access to their latest SDK payloads along with the integration documentation, which helped us further verify that the SDK signatures selected from linked apps are robust enough to uniquely fingerprint the corresponding proxy SDKs. In our research, we identified

51 high-confidence SDK signatures (20 for MonkeySocks, 14 for Luminati, 14 for IPninja, and 3 for Oxylabs). Only 3 Oxylabs signatures were discovered since its SDKs have different namespaces and only three URLs are found to be consistent across all proxy apps.

High-Throughput Signature-Based Proxy App Detection. Leveraging the discovered signatures, our approach automatically detected proxy apps from Androzoo [3], a large repository of Android apps. Androzoo periodically downloads APKs from multiple app stores including Google Play, Anzhi [5], and AppChina [6], etc. By Nov 2019, it contains almost 10M Android APKs. The detection is based upon signature matching, which however is more complicated than it appears to be. Specifically, not only should we match the signature with the APK binary payload. but we also need to inspect uncompressed files. Also, to speed up the matching, we skip media files including images, videos, and audios. In this way, we built up the detector and an APK will be detected as a proxy app when it matches at least one signature. We ran the detector on randomly sampled 2M Android APKs downloaded from Androzoo. Although apps with multiple APK versions can get higher chances to be picked out, due to our large sampling size (2M) and the long-tailed distribution of apps over their APK versions in Androzoo, we believe that overselecting APKs from the same app should not be a major concern. Specifically, the 2M APKs we analyzed are from a diverse set of 1.3M different apps. In total, 1,378 APKs were identified as proxy apps with each integrating one or more proxy SDKs, as detailed in §IV. We then sampled 100 detected APKs for manual study and results show that all of them are true proxy apps. The precision of our detector is further verified by observing the relay traffic when running these apps (§III-E).

D. Profile Proxy Apps

Here we introduce our toolchain for dynamic Android app profiling, which serves the following three purposes. First, our approach helps manual confirmation of the linked programs (as identified in §III-C) to be indeed proxy apps, and further identification of proxy SDKs and their signatures. Then it supports dynamical verification of detected proxy APKs on a large scale, by observing their relaying behaviors and traffic. Finally, the toolchain enables us to set up control experiments to understand how proxy SDKs adjust their relaying behaviors to the evolving system environments such as battery life, network types, and system idleness.

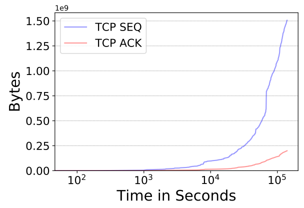
App Instrumentation. To collect detailed runtime logs for a given APK, our approach hooks its API calls using Frida-gadget [14], a dynamic instrumentation toolkit. This allows us to log API calls (parameters and payloads) when running a patched APK without interfering with the operations of its hosting system and other co-located apps. In our study, we developed hook scripts to log SQLite access, capture shared preference read/write, disable SSL pinning (so that we can access the TLS-secured control-plane communication between proxy peers and proxy gateways through MITM), and more importantly locate network API calls, etc. While our MITM experiment is focused on control-plane communication between proxy apps and proxy gateways, it may inadvertently intercept or interrupt relaying HTTPS traffic depending on whether TLS verification is enforced by proxy customers. Thus, an IRB application has been filed and got approved for this experiment. We will revisit this issue in more details in §III-F. Note that the app instrumentation component was only used during our short-term manual study of linked Android programs. For the other two scenarios (verifying detected proxy apps, and understanding behaviors of proxy SDKs), we skipped this step to run the original APKs.

Scalable App Profiling Platform. This component is at the center of our toolchain, supporting all three scenarios mentioned earlier. As shown in Figure 2, it consists of a scheduler and a pool of workers. The scheduler is used to schedule app profiling tasks to available workers while the workers are responsible for running a given app inside an Android emulator. Each worker operates as a Docker container pre-configured with a set of useful tools to achieve the following goals: (1) running an app APK inside the Android emulator with specific API levels and ABIs (armeabi-v7a, x86, etc.); (2) dumping the network traffic with tcpdump; (3) interacting with the emulator through the ADB tool so that we can change the app running environments such as battery life, screen on/off, and network types (WiFi or cellular).

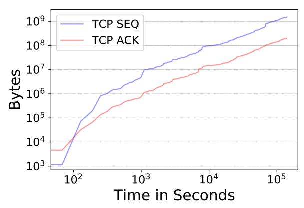
E. Proxy Traffic Detector

The network traffic captured by the aforementioned app profiling platform can come from multiple sources including the proxy SDK, other components co-located in the proxy app, apps pre-installed in the Android emulator, and the Android system itself. Our proxy traffic detector takes two steps to separate relaying traffic from the background noise. The first is to identify network traffic (*proxy connections*) between the proxy peers and the proxy gateway, and the second step is to pick out the communication (*relayed connections*) between proxy peers and traffic destinations.

Compose Groundtruth Dataset. We started with composing a groundtruth dataset consisting of proxy (between proxy peers and proxy gateways), non-proxy connections, and relayed connections (between proxy peers and traffic destinations). Specifically, non-proxy connections were collected through running the Android emulators without proxy apps. To gather proxy connections, we conducted a differential analysis on the traffic generated from the emulators with different configurations. For instance, to collect the proxy connections of Oxylabs, we ran several Android emulators with the following settings: without any proxy app, with *proxy app 1 of Oxylabs*, with *proxy app 2 of Oxylabs*, etc. From the traffic logs produced by the emulators ($T_{\text{without proxy app}}$, $T_{\text{proxy app 1 of Oxylabs}}$, $T_{\text{proxy app 2 of Oxylabs}}$, etc.), we recover the proxy traffic (proxy connections and relayed connections) associated with *Oxylabs* through $\cap_{i=1}^{n} T_{\text{proxy app-}i\text{ of Oxylabs}} \setminus T_{\text{without proxy app}}$, which allows us to exclude traffic generated by the system, co-located apps, as well as app components co-located inside the proxy apps of *Oxylabs*. From the proxy traffic, the proxy connections can be easily identified since they exhibit a strong correlation with the relayed connections in terms of timing and volume. For example, the relayed connections are established only after incoming traffic arrives through the proxy connections. In this way, we collected a dataset of 500 non-proxy connections, 50 proxy connections, and 500 relayed connections.



(a) Y-axis: # of Bytes



(b) Y-axis: Log-scaled # of Bytes

Fig. 3: TCP SEQ/ACK by Time for the Proxy Connection.

Detect Proxy Connections. To capture proxy connections, we utilize their unique characteristics as observed from the groundtruth dataset when comparing the proxy and non-proxy connections. Specifically, a proxy peer usually maintains one or more long-lived *proxy connections* to the proxy gateways through the TCP protocol. Outgoing packets of such a proxy connection are found to have a much larger TCP acknowledgement number compared to their TCP sequence number, and the gap grows as more traffic is transmitted. This indicates that the proxy peer uploads more data to the proxy gateways than downloads – much different from a typical mobile app that usually serves as the clients and consumes more data from the server-side than contributes data to the server. Figure 3 exemplifies how TCP SEQ/ACK of a proxy connection of the Oxylabs SDK changes over time. This proxy connection lasted almost 40 hours. When it terminated, the gap between the last outgoing TCP packet's SEQ and ACK becomes more than 1,200 MB. The observations allow us to accurately identify proxy connections from the network traffic dump.

Based upon the observations, we design the following metrics to profile a proxy connection: 1) the gap in bytes between TCP SEQ and TCP ACK, i.e., ConnGap $=$ Conn$_{SEQ}$ $-$ Conn$_{ACK}$; 2) the gap ratio between TCP SEQ and TCP ACK, i.e., ConnGapRatio $=$ Conn$_{SEQ}$/Conn$_{ACK}$; 3) the connection lifetime in seconds, i.e., ConnLifetime $=$ Conn$_{termination}$ $-$ Conn$_{handshake}$. Given those metrics, a lightweight threshold-based methodology was found to be very effective when evaluating on our labeled dataset. We ran a script to automatically explore different thresholds, seeking their combinations with the best performance in terms of recall and precision. The script reported that many combinations achieve both 100% recall and 100% precision, which suggests the decision boundary is wide and a threshold-based solution is good enough to handle this task. For our detection, we selected ConnGAP $\geq$ 1MB, ConnGAPRatio $\geq$ 6, ConnLifetime $\geq$ $1,000$ as the threshold combination, and it was found effective across proxy apps and proxy SDKs, which is reasonable considering that those proxy providers share the same service model.

Detect Relayed Connections. The next step is to identify the network connections (relayed connections) between the proxy peer and the traffic destinations. This step relies on the observation that relayed connections must exhibit a strong correlation with proxy connections, in terms of timing and volume. For example, relayed connections will only be set up following instructions from the proxy servers. Therefore, the handshake of the relayed connection comes very closely after the last incoming packets from the proxy connection. Also, packets received from a destination will be quickly forwarded to the proxy gateway, as the proxy peer acts as the bridge for the communication between the gateway and the destinations, leading to a strong correlation (timing and traffic size) in the traffic from the two connections. Leveraging the observations, we can effectively identify relayed connections and traffic by examining whether they exhibit a strong correlation with proxy connections reported in the first step.

Specifically, we define two metrics to identify relayed connections, *RelayedTiming* and *RelayedTrafficGapRatio*. *RelayedTiming* profiles the timing gap between the handshake of a connection and the last packet received from the proxy connection. The shorter it is, the more likely the connection is the relayed connection. *RelayedTrafficGapRatio* is defined as $\frac{\text{Traffic}_{\text{Proxy}} - \text{Traffic}_{\text{Relayed}}}{\text{Traffic}_{\text{Relayed}}}$ to profile the traffic volume difference between a given connection and the proxy connection during the lifetime of this connection. Still, the smaller it is, the more likely the connection is a relayed one. Similar to the last step, a threshold-based detector (*RelayedTiming* $\leq$ 0.05 seconds, and *RelayedTrafficGapRatio* $\leq$ 7%) was found to work effectively when evaluated on our labeled dataset.

Leveraging this two-step proxy traffic detector, we further validated the proxy app detection results by sampling 50 flagged proxy apps and observing whether there is proxy traffic when running the apps. The validation reported proxy traffic from all 50 sampled apps, indicating that our signature-based proxy app detector is of high precision. The identified relaying traffic can also help us understand how this mobile proxy ecosystem is used by proxy customers and for what purposes. Detailed results can be found in §IV and §V.

F. Discussion

Limitations. Our methodology for identifying proxy providers (§III-A) may miss underground proxy providers due to their low public visibility. Our proxy app detector (§III-C) significantly depends on robust signatures for proxy SDKs. Every time a new proxy SDK emerges, signatures should be collected before detecting. Also, it targets only the Android platform, and extra work needs to be done before it can handle the apps on other platforms such as iOS. Nevertheless, our experience indicates that: (1) the methodology for deriving signatures for one proxy SDK (e.g., Luminati) can be well generalized to other proxy SDKs; and (2) most steps in the signature generating process described in §III-C can be automated, such as identifying linked apps, verifying proxy apps, and extracting proxy SDK signatures. We therefore believe that manual efforts can be minor when extending the detector to a new proxy SDK. Another limitation is our threshold-based detectors for proxy traffic (§III-E). While they serve our purpose well to select out proxy connections and relayed connections, they may suffer from a set of limitations when applying to some real-world scenarios. First, the real-world traffic can be more complex than that produced by Android emulators, since it can be generated from many more apps and some traffic such as live streaming may also have a larger TCP SEQ than TCP ACK for outgoing packets. Also, proxy providers may keep adapting their traffic behaviors. We leave it to our future work to understand the problem and seek other solutions. Note that those potential limitations do not affect our study since we run proxy apps in emulated environments with much less background noise traffic. Another concern resides in whether proxy SDKs will distinguish emulated environments from real devices and adapt their behaviors accordingly. However, through controlled experiments on a small fraction of the apps by running them in emulators and real devices, we did not observe any proxy apps alter their behavior in an Android emulator. And we did not find proxy SDKs change their logic in an Android emulator either.

Ethical Concerns. In our research, we have filed 3 separate IRB applications for the user study (§IV-B), host fingerprinting (§III-B), short-term MITM experiment and long-term passive traffic logging (§III-D). All of them have been approved by our institution. In addition, we have enforced strict protection measures to minimize potential ethical risks in the host fingerprinting, the short-term MITM experiment, and the long-term passive traffic logging, as elaborated below.

Our host fingerprinting was configured with low frequency (at most once for each remote host per month) to avoid the burden introduced to remote hosts. In the short-term MITM experiment, relaying traffic may go through the proxy app. Thus, relayed HTTPS connections are either interrupted or intercepted depending on whether TLS verification is enforced by proxy customers. In the case of interception, we could inadvertently get access to the plaintext of relaying HTTPS traffic. To protect the privacy of proxy users, we did not look into the payload of any intercepted HTTPS connections from the users and had permanently deleted the data after the experiment. For the long-term traffic logging, we focused on the categories of traffic destinations (IPs and domains) and did not access the payload of HTTP traffic. Also, the traffic logs are stored in the physical servers at our institution and only privileged administrators have access. We plan to permanently delete them once the project finishes.

IV. Mobile Proxy Program Analysis

In this section, we report our new findings and understandings of mobile devices as residential proxy peers, a service at the center of mobile SDK-based residential proxy networks. We first present the measurement findings about the landscape and maliciousness of proxy programs (i.e., Android APKs) on mobile devices. Then we describe our discoveries through reverse engineering of proxy SDKs, and our investigation of user awareness and willingness to relay traffic.

A. Analyzing Proxy Programs

Finding I: *Proxy apps have hundreds of millions of installs in total, but many have been unpublished from Google Play, likely due to violation of Google Play's developer policy.*

Scope and Magnitude. Initially, through scanning the sampled 2M APKs from Androzoo, we discovered 1,378 Android APKs belonging to 963 Android apps (an Android app can have multiple APK versions), which have ever served as proxy peers for at least one of the four residential proxy service providers including MonkeySocks, Luminati, Oxylabs, and IPninja. Among the 1,378 Android proxy APKs, 1,374 are downloaded from Google Play by Androzoo, while the remaining 4 are from AppChina [6].

To understand how long Android apps have served as proxy peers, we collected historical APK versions for each app from Apkmonk and Google Play and discovered another 1,493 unique APKs. We then looked into whether those APKs integrate proxy SDKs using the signature-based detection (see §III-C). In total, we gathered a dataset of 963 Android proxy apps along with their 2,871 APKs, among which, 1,701 were detected as proxy APKs. Table IV presents the number of Android proxy APKs and proxy apps across proxy providers, as well as the number of Android proxy apps still available on Google Play by Oct 2019 and their installations in total. Surprisingly, 2 proxy apps were found to serve as proxy peers for multiple proxy providers. For instance, com.plonkgames.apps.iq_test (5M installs) works for both Luminati and Oxylabs.

TABLE IV: Detected Android proxy APKs and apps. GP means Google Play, and app availability and installs were captured from GP between Aug and Oct 2019.

Provider	# APKs	# Apps	# in GP	# Installs in GP
MonkeySocks	787	715	79	5,749,336
Luminati	801	204	59	229,187,000
Oxylabs	86	27	23	76,250,000
IPninja	25	18	11	2,410,075
Overall	1,701	963	171	308,596,411

Interestingly, 71 Android apps dropped out the proxy networks during the observed lifetime. Examples include com.appsomniacs.da2 with 100M installations on Google Play. Among its 21 APK versions spanning 2017-07-13 and 2019-09-12, only 6 were found to serve as proxy peers for Luminati between 2017-07-13 and 2018-05-24. We also notice that a proxy app can migrate from one proxy SDK to another. One example is com.littlepako.opusplayer3, which migrated from IPninja to Luminati. We further check the number of installations of those proxy apps. For this purpose, we crawled apps' metadata (e.g., number of installation, category, etc.) in Google Play in October 2019. Among 963 apps, 171 (79 for MonekySocks, 203 for Luminati, 27 for Oxylabs and 18 for IPninja) were found. The top 5 app categories are Education (17.69%), Tools (11.54%), Personalization (7.69%), Music & Audio (6.92%), Communication (6.15%). In total, these apps were installed 300M times and 10% of them have more than 1M installations. Besides, while MonkeySocks has the largest number of proxy Android apps found in Google Play, 90% of them have less than 20K installations. Meanwhile, most apps for Luminati and Oxylabs have more than 100K installations.

Delisted Proxy Programs. We further conducted a longitudinal study of those Google Play apps to understand their availability. Specifically, we checked each proxy app's metadata from Google Play weekly from Aug 13, 2019, to Oct 26, 2019. Interestingly, we observed that 42 apps were delisted from Google Play during our observation period and only 129 were left in the app store. Considering all 963 proxy apps once existed in Google Play, the delisting rate is as high as 86.60%. While an app can get delisted for various reasons, it still raises concerns regarding the maliciousness of proxy apps and proxy SDKs. We further checked Google Play's developer policy [17] and found that the following behavior is considered as a policy violation: "*Apps that facilitate proxy services to third parties may only do so in apps where that is the primary, user-facing core purpose of the app*". This policy has been violated by almost all detected proxy apps, since none of the apps we sampled explicitly state their functionality to be a proxy. Among the 129 apps still available in Google Play, 10 of them were still integrated with the proxy SDKs as in mid-Nov 2019.

Finding II: *Malicious mobile apps are abusing the proxy ecosystem to monetize their user base while integrating proxy SDKs can damage the reputation of benign mobile apps.*

Blacklisting. We further checked whether these proxy APKs are credible. We downloaded the analysis reports if available from VirusTotal [48] for those APKs. Among the 1,701 Android proxy APKs, 1,404 have been analyzed by VirusTotal. Among these analyzed APKs, 166 (11.82%) were alarmed by more than 10 anti-virus engines, 680 (48.43%) were hit by at least 5 and 1,212 (86.32%) were flagged by at least one as malicious. Interestingly, among 129 apps still active in Google Play, 64 were analyzed, 18 of them were flagged by at least one anti-virus engine, and one was even alarmed 15 times. Also, MonkeySocks and Luminati have a much higher ratio of malicious proxy apps, compared to Oxylabs and IPninja. One explanation is that the first two providers fail to enforce strict background checks when recruiting new proxy apps.

Table V shows the top 10 malicious categories given by the anti-virus engines. As expected, the proxy category comes as the first, which is aligned with our detection results. Interestingly, besides commonly known malicious categories such as PUP, adware, and trojan, 142 proxy APKs were detected as fake apps (a.k.a., phishing apps). Examples include com.wChatApps_5759913 (a fake Wechat app), and com.wWhatsUppMessenger_6689631 (a fake Whatsapp app). All those fake apps are served as MonkeySocks' proxy peers. Also, some of Luminati's proxy apps were reported as Tapcore [45] and Fobus [13], both of which are recently emerging malware families on the Android platform. Besides, 12 apps (16 APKs), which integrate Luminati or Oxylabs proxy SDKs, were found to be clickers (ad fraud).

To investigate whether the maliciousness stems from mobile proxy SDK or other components of the proxy program, for each proxy app, we compared the anti-virus scan results of its historical APKs with and without mobile proxy SDK. we found that 39.50% of the Android apps are blacklisted even without mobile proxy SDK by at least one anti-virus engine, while 15.75% has been alarmed by at least 5 engines. Comparing different proxy providers, Oxylabs's proxy apps look most clean with only 1.54% non-proxy APKs alarmed at least 5 times. *Those results indicate that this proxy ecosystem may be abused by malware operators as a profitable and stealthy monetization channel.* For those apps being flagged after mobile proxy SDK is added, we observe that adding MonkeySocks and Oxylabs proxy SDK leads to a higher probability for the apps to be flagged as malicious. In particular, 78.67% proxy apps of MonkeySocks were alarmed at least 5 times after integrating the SDK, while only 25.42% of these apps were flagged before the integration. For Oxylabs, the ratio increases from 1.54% to 15.79%.

TABLE V: Malicious categories of proxy APKs and apps, wherein, L denotes Luminati, M for MonkeySocks, I for IPninja, and O for Oxylabs.

	Category	% APKs	% Apps	# Providers	# AV Engines
1	Proxy	76.21%	92.02%	M-L-O	1
2	Adware	45.57%	74.02%	M-L-O	7
3	PUP	45.36%	69.33%	M-L	2
4	Trojan	45.14%	73.76%	M-I-L	6
5	Riskware	37.86%	63.50%	M-I-L-O	3
6	Fakeapp	10.14%	17.49%	M	4
7	Virus	8.29%	14.45%	M	1
8	Tapcore	4.79%	7.73%	L	3
9	Fobus	2.21%	1.27%	L	1
10	Clicker	1.14%	1.52%	L-O	2

B. Analyzing Proxy SDKs

Finding III: *Proxy SDKs deploy various tricks to keep relaying traffic in the background without users' notice.*

Workflow and Architecture. In our study, we identified proxy SDKs for four proxy providers and then reverse engineered those SDKs and some of their host APKs to understand their workflow and architecture. The typical workflow of a proxy SDK is as follows: when a proxy SDK is invoked for the first time, a traffic relaying notice will be triggered. Once users agree with the service terms, the SDK starts running in the background, along with several broadcast receivers registered to monitor device events. Those receivers allow the SDK to adapt proxy behaviors to the device status such as battery life, network type, and device idleness. Also, since a background service can be killed when the system needs to free resources, those SDKs try to restart their main services by registering restarting jobs (usually, through Android AlarmManager). Interestingly, after Android API 26, new restrictions are introduced to background services if they are from the apps with no active components running in the foreground [4]. We found that all SDKs implemented some countermeasures to circumvent the restrictions, such as starting JobService instead of Service. Below we elaborate on the user consent and relaying constraints of those proxy SDKs to further understand their impacts on device users.

User Consent. Among the four proxy service providers, we found that all of them would ask for user consent before relaying traffic, as shown in Figure 4. For instance, Luminati's proxy apps ask for user's consent through a dialog with the following statement as shown in Figure 4(a): *Your use of "App XX" is free of charge in exchange for safely using some of your device's resources (WiFi and very limited cellular data)*. To understand whether users can correctly interpret such requests along with their attitudes towards traffic relaying through their devices, we conducted a user study, which was approved by our institution's IRB. This user study involves an on-site survey. 50 participants were recruited through emails, posters and other channels, with each given a shopping gift card of $10 after participation. All of them have used mobile apps for more than three years. Their age ranges from 18 to 65 (62% are male and 38% are female), and their education level ranges from high school to graduate degrees. Before the survey, 11 of them (22% of all participants) knew the concepts of web proxy and relaying traffic. After reading our explanation hints, 49 of them understood the two concepts. Details of the survey are presented in Appendix IX-A.

Finding IV: *Proxy SDKs' user consents are ambiguous as most users cannot capture the intention of relaying traffic.*

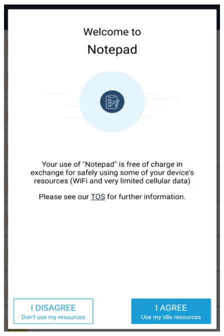
User awareness of traffic relaying consent. The survey is designed to ask users to which extent they understand that accepting user consent will lead to relaying traffic via their mobile devices. Specifically, we first randomly selected one user consent dialog from the proxy apps of all 4 proxy providers. We then asked subjects to provide their understandings on those dialogs as open-ended questions. Next if the subject knew nothing about web proxy or relaying traffic, we provided them with an explanation and asked again the readability of the user consent dialogs (e.g., "do you think the user consent dialog expresses clearly that it requests to turn your device into a web proxy and relay unknown traffic").

For the initial open-ended questions, most subjects misunderstood those user consent dialogs and only 10% of the participants expressed that the apps may want to relay traffic through their mobile devices. For the rest of the subjects, 56% mentioned that the app did nothing different from legitimate apps: e.g., "It seems that the app is free to use and will consume my cellular data, which is what every app does in general". Among all participants, 16% of them expressed their concerns that this app may collect their non-personal data: e.g., "It is free but it may collect some information like device information (device ID) and network information which is likely to be used in targeted advertising analysis and market analysis." After knowing the concepts of web proxy and relaying traffic, 72% of the participants think the dialogs are "Not at all clear" in expressing the intention of turning a user's device into a web proxy. We notice that the keywords "proxy" or "traffic relaying" *rarely* appear in the user consent dialogs. This may be because such apps try to circumvent app vetting and hide their roles of being proxy peers.

Finding V: *Most users are not willing to relay traffic through their devices, not to mention using cellular data.*

Users' unwillingness of traffic relaying in exchange for rewards. We also asked the participants to which extent they are willing to relay traffic in exchange for rewards such as free app functionality. Specifically, in the aforementioned survey, we asked them another set of questions especially ones on the acceptable traffic relaying constraint (e.g., "How much WIFI data do you allow the proxy app to consume?") and bonus (e.g., "If provided with a small amount of money, are you willing to relay traffic?").

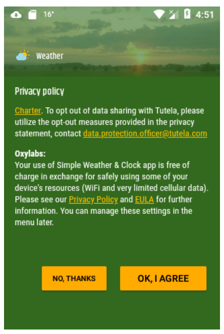
Results show that 34% of the participants are "not at all willing" to relay traffic via their mobile devices, regardless of rewards. Even for those who are willing to some extent, 69.7% would only allow the relaying activity to consume their WiFi resource for less than 1 hour every day, and 87.9% do not allow consuming their cellular data. We further asked the participants for their concerns, no matter whether they are willing or not. The top two concerns are privacy (82%) and cellular data (62%). From participants' feedback, we conclude that a large fraction of mobile users do have concerns about allowing apps to relay network traffic. This possibly explains proxy SDKs' vague user consent terms as described above.


Welcome to
Notepad

Your use of "Notepad" is free of charge in exchange for safely using some of your device's resources (WiFi and very limited cellular data)

Please see our TOS for further information.

(a) Luminati

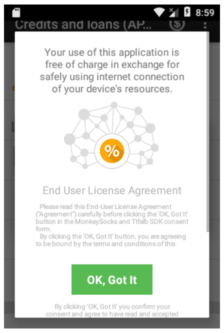


Privacy policy

Charter. To opt out of data sharing with Tutela, please utilize the opt-out measures provided in the privacy statement, contact data.protection.officer@tutela.com

Oxylabs:
Your use of Simple Weather & Clock app is free of charge in exchange for safely using some of your device's resources (WiFi and very limited cellular data). Please see our Privacy Policy and EULA for further information. You can manage these settings in the menu later.

(b) Oxylabs



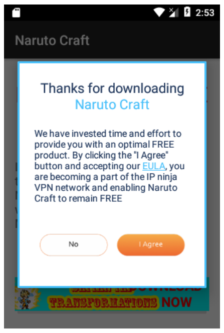
Your use of this application is free of charge in exchange for safely using internet connection of your device's resources.

End User License Agreement

Please read this End-User License Agreement ("Agreement") carefully before clicking the 'OK, Got It' button in the MonkeySocks and Ttfalb SDK consent form.
By clicking the 'OK, Got It' button, you are agreeing to be bound by the terms and conditions of this

By clicking 'OK, Got It' you confirm your consent and agree to have read and accepted

(c) MonkeySocks



Thanks for downloading
Naruto Craft

We have invested time and effort to provide you with an optimal FREE product. By clicking the "I Agree" button and accepting our EULA, you are becoming a part of the IP ninja VPN network and enabling Naruto Craft to remain FREE

(d) IPninja

Fig. 4: Dialogs of Proxy SDKs to Ask for User Consent.

Finding VI: *Proxy SDKs consume a non-negligible amount of cellular data and it is out of device owners' control.*

Traffic Relaying Policies. The app profiling platform described in §III-D enabled us to set up control experiments wherein we selectively changed system configurations to study how the relaying behavior of a proxy app adapts to the network type (WiFi or cellular), battery life, and screen on/off, etc. By combining those experiments along with reverse-engineering of the proxy SDKs, we were able to acquire a comprehensive understanding regarding how system environments affect the relaying behaviors. Specifically, we observed that *the proxy SDKs tend to avoid abusing system resources to a certain extent.* Relaying is tuned to a lower frequency when the battery life is low (e.g., for MonkeySocks, the threshold of the battery life is below 50% when not on charge, or below 15% even on charge), or the network type is cellular. Also, some proxy SDKs such as Luminati cease the relaying activities when users are performing resource-consuming operations such as calling. However, *a fair amount of cellular data is still consumed to relay traffic.* While significantly lower than the consumption of WiFi data, an average daily usage of 15 to 30MB cellular data is found in the proxy SDKs for Oxylabs, Luminati, and MonkeySocks. While for IPninja, reverse engineering reveals that the provider configures the SDK to use up to 512MB cellular data in a 30-day cycle before ceasing the relaying. Also, *these policies can be updated from the proxy gateways, but cannot be configured by device owners.* In our study, we found that the SDKs periodically retrieve updates from the server. This implies that the proxy providers have the mechanism of making the replaying policies more aggressive to serve more proxy customers. In contrast, device owners

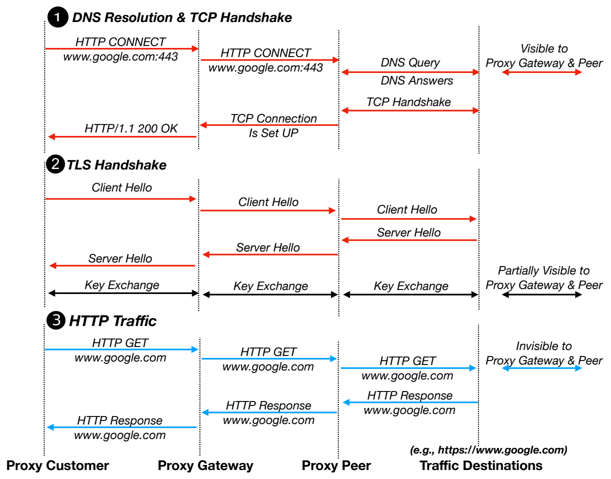


Fig. 5: The Process of Traffic Relaying.

of the proxy peers have no access to or visibility of those policies, not to mention the capability to directly configure them according to their preferences.

V. Mobile Proxy Traffic/IP Analysis

By analyzing the identified proxy apps, we took a closer look at the relaying protocol and relaying traffic, which reveals important findings regarding the proxy mechanism and proxy usage. Also, our infiltration of proxy networks (§III-B) allows us to capture and profile proxy peers attaching to cellular IPs.

A. Analyzing Mobile Proxy Traffic

Proxy Protocols. Identifying the proxy apps and SDKs enables us to understand how mobile proxy peers operate. In particular, we utilize the app profiling platform (see §III-C) to capture raw traffic and reverse engineer the mobile proxy protocol (without MITM) used by the proxy networks. Specifically, we observed a typical relaying process (see Figure 5) starting with an SDK-side DNS resolution for the destination domain name. Once the DNS resolution is done, the SDK makes a TCP connection with the destination IP, and notifies the proxy gateway. After that, each HTTP request is then relayed to the destination, and its response is forwarded back to the gateway. Underlying such interactions is a proxy protocols in line with the well-known HTTPS proxy protocol [19], which guarantees that MITM cannot access the plain traffic as long as the proxy customer has validated the TLS certificate. Once the TLS handshake is set up, HTTP request data is then relayed.

Also, we found that to relay traffic, proxy SDKs (proxy peers) need to make at least one long-term TCP connection with proxy gateways. Multiplexing protocols are used to handle multiple proxy tasks through a single TCP connection. For instance, MonkeySocks deploys HTTP/2 while the other three (Luminati, Oxylabs, and MonkeySocks) use WebSocket. In our study, we observed concurrent relaying behavior when running those SDKs. For example, more than 100 concurrent proxy tasks were found when running the Oxylabs SDK.

Finding VII: *Suspicious advertisement frauds contribute most to relaying traffic, followed by search engines, traveling, shopping, social networks, and email services.*

TABLE VI: Top domains ordered by the TCP connection count, for relay traffic of Oxylabs SDK.

Domain	% Conns	% Traffic	% HTTP	% HTTPS
www.gamepk.us	15.11%	1.31%	99.97%	0.00%
www.ltv-mob.com	3.59%	0.78%	69.87%	30.11%
www.finadvise.net	2.62%	0.20%	99.92%	0.00%
08j-0.tlnk.io	2.48%	0.32%	94.90%	5.05%
u.newspro.info	2.13%	0.15%	99.78%	0.00%
cp.effoulanponta.com	1.52%	0.82%	23.41%	76.52%
ads.pulseradius.com	1.34%	0.78%	0.00%	99.97%
okr-1.tlnk.io	1.30%	0.76%	0.24%	99.76%
adsperfection.imp2aff.com	1.27%	0.26%	65.17%	33.59%
imp.control.kochava.com	1.26%	0.70%	0.00%	100.00%

Proxy Traffic Analysis. In our research, we collected traffic logs by running proxy apps for the aforementioned proxy providers between Sept 2019 and Nov 2019, leveraging our app profiling platform (§III-D). We then utilized our relaying traffic detectors (§III-E) to recover the proxy traffic. Note that IPninja's proxy service was under maintenance during that period; as a result, we did not see its proxy traffic. Interestingly, we discovered a previously unknown proxy SDK (which we call *SDK X*) from one of the MonkeySocks proxy apps. Later we elaborate on this SDK as a case study.

For all four proxy SDKs (Oxylabs, MonkeySocks, Luminati, and SDK X), their traffic volumes vary across SDKs. Specifically, the volume is around 1GB per day per proxy app for SDK X and SDK Oxylabs, while it becomes much smaller (less than 100MB) for Luminati and MonkeySocks. For Luminati, this may be because its proxy apps run very slowly on our x86 physical servers due to the native libraries of Luminati SDK only supporting ARM architecture. For MonkeySocks, there may be few traffic relaying demands in our geolocations since it operates its proxy business mainly in Russia and our profiling system is deployed in the US.

Access to proxy traffic helps us learn more about proxy activities. When analyzing the destination domains of the proxy traffic, we found that most such domains involving the largest number of TCP connections carry out advertisement-related activities (such as ad redirection, ad impression and ad clicking), as shown in Table VI. For example, 08j-0.tlnk.io and adsperfection.imp2aff.com are related to ad tracking and ad clicking. Another example is gamepk.us whose URLs are in the form of http://www.gamepk.us/XX, where XX is a unique identifier, and it redirects its visitors to clk.myiads.com/click or cpi-offers.com, both related to ad clicking. Interestingly, although those top domains host the largest number of connections, they produce disproportionately less traffic volume, which is unusual as regular advertisement activities should incur a large traffic volume to load advertisement media files. *This observation, combined with those domains' impression/clicking URLs, indicates that those advertisement activities are related to ad frauds where instead of loading the advertisements, they only need to craft and send impression/clicking requests to the advertisement tracking system.* Taking www.gamepk.us as an example, it was involved in 38K TCP connections within a 24-hour relaying period through Oxylabs's SDK, but only generated 40MB traffic. In contrast, 248 requests/responses to www.bing.com incurred 110MB traffic. When ordering domains by their traffic volumes, other categories stand out including traveling (e.g., www.tripadvisor.com and www.expedia.com), shopping (www.amazon.com, www.bestbuy.com, etc.), searching (www.google.com and www.bing.com), and social media (connect.facebook.net and profile.meetme.com). In addition to regular web traffic, we also observed IMAP mailing traffic to popular mailing services such as imap.mail.yahoo.com, imap.gmail.com, and imap.aol.com.

Finding VIII: *Proxy SDK X has been integrated into multiple apps and stealthily relays network traffic without any user consent.*

Previously Unknown Proxy SDK. As mentioned earlier, we found a previously unknown SDK, named SDK X, when analyzing proxy traffic of MonkeySocks proxy app com.zmobile.saveplan. Specifically, our traffic selection technique (§III-E) identified an unknown proxy connection and its traffic, along with those of MonkeySocks, in the traffic logs of com.zmobile.saveplan. We found that while the SDK is obfuscated with different package names for different host apps, its plain-text server addresses (e.g., inputstreamreader.link, svc-host.net, and dll-host.cf) are hardcoded in its variations. After searching these server addresses in other apps, we found that 21 proxy apps (60 proxy APKs) for Luminati and 3 proxy apps (3 proxy APKs) for MonkeySocks also integrate this SDK. Furthermore, leveraging the technique introduced in §III-C, we identified 657 Android APKs and 34 Windows programs potentially serving as proxy programs for this SDK X.

Further profiling proxy programs with SDK X shows that it runs the WebSocket protocol as many other proxy SDKs do. However, it does not ask for user consent before relaying traffic, thus violating a set of privacy terms and regulations [15] [10]. Also, SDK X relays traffic aggressively with around 1GB of traffic volume per day per proxy app.

B. Analyzing Mobile Proxy IPs

Finding IX: *A large amount of cellular proxies and IPv6 proxies are observed in our infiltration.*

Landscape. As mentioned earlier in §III-B, we infiltrated the 9 proxy providers listed in Table II. Overall, 8,188,438 IPs (7,181,557 IPv4, and 1,006,881 IPv6) were captured through 42M infiltration probes. Among the IPv4 addresses, 624,989 (8.7%) are labeled as cellular IPs by IPInfo [24], an IP metadata database. Due to the missing IPv6 information in IPInfo, we do not know whether the IPv6 addresses are cellular or not. Also, most of the identified cellular IPs are from five proxy providers: Luminati, Oxylabs, Smartproxy, ProxyRack, and Geosurf. The fact that Luminati and Oxylabs have mobile proxy SDKs well explains the cellular IPs captured in their proxy networks. We have also contacted the other three, trying to find out whether they have mobile proxy SDKs and mobile proxy apps; only ProxyRack was confirmed to have purchased some 3rd-party apps and turned them into proxy peers. We leave it as our future research to investigate how Geosurf and Smartproxy have recruited so many cellular proxy IPs.

We found that the cellular proxy IPs are widely distributed, though following a long-tail pattern. In particular, the top 5% of IPv4 /24 entities contribute 50.64% of proxy IPv4 addresses. Also, 31.57% of cellular proxy IPs come from the top 5 countries including Turkey (12.57%), India (5.52%), Indonesia (5.39%), Russia (5.04%), and Malaysia (3.04%).

TABLE VII: Top device/OS types in activeFP for cellular IPs

Top Device Types			Top OS Types		
Name	Probes	IPs	Name	Probes	IPs
WAP	15.57%	22.59%	Linux	35.13%	45.46%
router	4.06%	2.43%	iOS	2.25%	0.29%
firewall	3.26%	0.63%	Windows	1.93%	1.06%
security-misc	0.81%	0.63%	Unix	0.28%	0.36%
webcam	0.05%	0.10%	Comware	0.14%	0.14%
broadband router	0.02%	0.07%	VRP	0.02%	0.05%
Overall	18,299	4,152	Overall	18,299	4,152

Fingerprinting Analysis. We also conducted passive fingerprinting (passiveFP) and active fingerprinting (activeFP) to infer device/OS types of proxy peers, as detailed in §III-B. PassiveFP covered 48.28% of the 2,197,019 infiltration probes (due to deployment delay between infiltration and fingerprinting) and 72.06% of the 625K unique cellular proxy IPs. Also, with the 0.85% response rate, activeFPs covered 4,152 cellular proxy IPs (0.67%), among which the OS information of 47.11% and the device information of 26.06% were reported.

Table VII presents the top device types revealed by activeFPs. The response rate of activeFPs on cellular IPs is only 0.85%, much lower than the 7.24% achieved for the overall 8M proxy IPs. This is reasonable since cellular IPs are commonly assigned to mobile devices that usually do not respond to any incoming probing traffic. However, we were still able to observe diverse device types. The most common ones are WAP, router, and firewall, which is expected as most proxy peers are likely sitting behind NAT to which cellular IPs are attached. Table VIII and Table VII show the top OSes revealed by passiveFP and activeFP. Here we can see that Linux accounts for the majority of cellular IPs, especially by passiveFPs. This is because Android is built upon Linux kernel, and therefore is reported as Linux by p0f. Besides, other OS types such as Windows, macOS, and iOS were also found.

Finding X: *Cellular proxies are highly evasive and only a negligible portion are alarmed by VirusTotal.*

Evasiveness. We further checked whether these cellular proxy IPs were ever blacklisted. For this purpose, we scanned these IPs using VirusTotal. Overall, 0.44% of the cellular proxy IPs were reported by at least one IP blacklist for either malicious URLs or download samples. The portion of such blacklisted IPs is fairly small across proxy providers, and no providers have more than 1% of cellular IPs being blacklisted. Among these services, Smartproxy has the most blacklisted IPs (0.76%), which is followed by Oxylabs (0.66%) and Luminati (0.45%). When analyzing the malicious activities they were involved in, we found that malicious subdomains of DDNS (dynamic DNS) and Trojan samples were two mostly reported. Compared to the prior research [32] that reports 2.20% of blacklisted residential IP proxies, our study discovers even fewer malicious indicators on cellular IP proxies. Also interestingly, we found that 34 IPs (not cellular but among the overall 8M IPs) were served to distribute 6 different payloads in the IoT botnet campaign of Hajime while another 12 proxy IPs were found to distribute 26 various Mirai (botnet) payloads.

TABLE VIII: Top OS types in passiveFP for cellular IPs

Top OS Types	Probes	IPs
Linux 2.2.x-3.x	71.12%	79.58%
Windows 7 or 8	8.79%	11.09%
Linux 3.1-3.10	6.08%	8.03%
Windows NT kernel	4.43%	5.10%
Linux 3.11 and newer	1.66%	0.12%
Linux 2.2.x-3.x (barebone)	0.71%	0.61%
FreeBDS 9.x or newer	0.56%	0.48%
Linux 2.2.x-3.x (no timestamps)	0.52%	0.95%
Linux 3.x	0.50%	0.50%
Mac OS X	0.25%	0.53%
Overall	1,060,771	450,375

VI. Discussion

Responsible Disclosure. We have been actively communicating with Google since December 2019. Through a series of meetings and emails, we shared with them our findings along with the methodologies, detection signatures, and detected Android APKs. Members from Google Anti-Abuse and Google Play Protect teams have acknowledged our findings, and we all agree that the mobile proxy ecosystem is incurring high security and privacy risks to Android users. While providing lists of detected APKs and apps, we didn't get the details regarding whether a specific app is PUP (potentially unwanted program) or not. However, those apps have violated Google Play's developer policy, and Google is working on profiling and detecting those proxy SDKs and apps, as learned from our communication, which therefore strongly indicates that Google considers them as PUPs. Also, we have contacted proxy providers regarding the suspicious usage of their proxy services, the ambiguity of their user notification, as well as the malicious apps integrated with their proxy SDKs. While varied by proxy providers, our findings have been well acknowledged. We are pleased to learn that some proxy providers are taking measures to mitigate the security and privacy risks. Specifically, Luminati has redesigned its user consent dialog to clearly explain the proxy function and traffic relay policies. It also claimed a strict vetting of its partners. Besides, Oxylabs claimed they had stopped offering proxy SDKs to 3rd-party app developers concerning the security and privacy risks. We haven't yet received any response from the other two proxy providers (MonkeySocks and IPninja) after contacting them several times.

Mitigation of Residential Proxy Abuse. Our study uncovered security and privacy risks incurred by the mobile proxy ecosystem. However, there is no industry-wide guideline to allow a trustworthy mobile proxy network. Through this study, with better understandings of stakeholders (i.e., app developers, app marketplace and proxy service providers), we recommend specific guidelines as below to regulate this ecosystem.

For proxy providers, we suggest three practices related to SDK development, proxy app review, and proxy customer approvals. First, we suggest having the proxy SDK configurable for app developers and device users, who can thus tune the traffic relaying policies and limit the types of data collected by the proxy SDK (e.g., geolocation, device identifiers). Also, the user consent must be specific, informed and unambiguous, considering the non-experts without the knowledge of traffic relaying. Proxy providers may carry out user studies as we did to profile and improve their user consent. Second, a strict vetting of proxy app candidates is recommended. In our study, malicious apps have been found to serve as proxy peers (§IV-A), which not only lowers the reputation of proxy providers, but also makes this ecosystem a monetization channel for malware. Also, legitimate apps may evolve to become malicious, and thus periodical inspection of vetted proxy apps is necessary. Lastly, proxy providers should deploy techniques to detect and prevent malicious traffic from being relayed through their proxy networks, which, along with proxy app vetting, will help mitigate the abuse of this ecosystem.

For the app marketplace, detecting proxy SDKs is recommended to be included in the app vetting procedure to protect device users against the potential collusion between app developers and proxy service providers. Our signature-based proxy app detector can be leveraged for this process. Also, since integrating a suspicious proxy SDK will compromise apps' reputation, app developers should carefully scrutinize a proxy SDK before integrating, especially in terms of user data access, traffic relaying policies, and user consent.

Datasets and Code Release. Relevant datasets and source code can be downloaded from https://github.com/mixianghang/mpaas. Our datasets include the proxy SDK signatures, proxy app identifiers, and captured proxy IPs, while the source code contains the signature-based proxy app detector and the proxy traffic detectors.

VII. Related Work

Studies on Web Proxies Numerous studies have looked into the security issues on web proxy services. Several works [47] [39] [30] conducted empirical studies on open web proxies to understand their usage pattern and malicious activities (e.g., ad injection). Weaver et al. [49] conducted a measurement study to understand the purpose of free proxy services based on how they modify traffic. Chung et al. [11] studied a paid proxy service to uncover content manipulation in end-to-end connections. O'Neill et al. [36] measured the prevalence of TLS proxies and identified thousands of malware intercepting TLS communications. Reaves et al. [40] studied VoIP-GSM gateways informally known as "simboxes" which are used to bridge incoming international VOIP calls with a local cellular voice network, aiming to evade costly interconnects. Some other studies focus on web proxy detection. Zhang et al. [52] proposed a proxy server detection technique leveraging the distinctive characteristics of interactive traffic such as packet size and timing. Other works [25] [49] developed techniques to detect the presence of web proxies, such as a proxy localization technique based on traceroutes of the SYN-ACK packets responding to TCP connection requests.

The closest to our study is [32], which reports an empirical study on five residential proxy services with a focus on profiling their service models and proxy IPs. Different from this work, we shift the focus to mobile proxy programs and reveal the prevalence of mobile devices as proxy peers via proxy SDKs. Also, our work achieves a series of novel findings regarding proxy SDKs, proxy apps, proxy traffic, and human factors, etc. Besides, we introduced new methodologies, such as the techniques to automatically detect and profile mobile proxy SDKs and apps, the user study that profiles users' attitudes towards mobile proxies, as well as the methodology to detect relaying traffic.

Android Malware Detection. There exists a plethora of works on Android malware detection. To name a few, Grace et al. [18] built up RiskRanker to analyze whether an app exhibits malicious behaviors. Wu et al. [50] proposed DroidMat to cluster Android apps as malicious or benign, leveraging static information such as app permissions. Yuan et al. [51] extracted more than 200 features from Android apps and used them to demonstrate the effectiveness of deep-learning-based malware detection. Mclaughlin et al. [31] proposed a malware detection system leveraging deep CNN (convolutional neural network). Instead of hand-engineered features, the authors use the static opcode sequence of dissembled programs as input for learning the decision boundary. Gong et al. [16] reported their experiences of developing, deploying, and maintaining a machine-learning-based malware detection framework for a commercial Android app marketplace.

Third-party SDK Detection and Analysis. Backes et al. [8] proposed LibScout to identify and group third-party libraries under different versions; they generate SDK profiles that can be uniquely identified by their signature trees. Ma et al. [29] proposed LibRadar, which detects libraries based on stable API features that are obfuscation-resilient. Li et al. [27] proposed LibD, which uses the internal code dependencies of an app to recognize library candidates and further classify them. In contrast to previous studies on generic third-party SDKs, our work focuses on proxy SDKs provided by mobile proxy service providers. We also investigate how these SDKs interplay with the mobile proxy ecosystem.

VIII. Concluding Remarks

We conducted to our knowledge the first systematic research on the mobile proxy ecosystem built upon mobile proxy SDKs and mobile proxy apps. In this ecosystem, mobile devices of normal end users are turned into proxy peers to relay unknown or even malicious traffic. In our study, we developed a measurement infrastructure consisting of several novel components, such as the signature-based detector that detects Android proxy apps on a large scale and the two-step proxy traffic detector built upon robust heuristics.

Leveraging the measurement platform, we make several important observations. We discover four proxy providers that offer mobile proxy SDKs as a profitable app monetization channel. The SDKs were found to have been integrated into 963 Android apps with $50K monthly payment per 1M MAU (monthly active users). However, most of those proxy apps have violated the developer policy of Google Play. Also, 48.43% proxy APKs were flagged by at least 5 anti-virus engines as malicious. Besides, our user study shows that their user consent texts are ambiguous as most subjects cannot tell the intention of relaying traffic from the consent texts. Even worse, one proxy SDK stealthily relayed traffic without showing any notifications. Furthermore, we observed a set of suspicious activities incurred by proxy traffic such as ads fraud. All the above findings highlight important security concerns of the mobile proxy ecosystem, especially for mobile device users. Moving forward, we suggest that critical stakeholders (proxy providers, app developers, and app stores) take more actions to address the security issues. We have reported our findings to relevant parties such as Google Play. Relevant datasets and source code can be accessed from https://github.com/mixianghang/mpaas.

Acknowledgment

We would like to thank the anonymous reviewers and our shepherd Prof. Christina Pöpper for their insightful comments. This project is jointly supported by the following NSF awards: 1917424, 1850725, 1618493, 1801432 and 1838083.

References

[1] A Twitter message shows up Luminati's proxy SDK promotion. https://twitter.com/malwrhunterteam/status/902965922016714753?s=20, 2017.

[2] Alexa Find Similar Sites. https://www.alexa.com/find-similar-sites.

[3] Kevin Allix, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 468–471. IEEE, 2016.

[4] Android Background Execution Limits. https://developer.android.com/about/versions/oreo/background.html.

[5] Anzhi. http://www.anzhi.com/.

[6] AppChina. http://www.appchina.com/.

[7] AppGrow. https://www.appgrow.com/.

[8] Michael Backes, Sven Bugiel, and Erik Derr. Reliable third-party library detection in android and its security applications. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 356–367. ACM, 2016.

[9] Blazing Proxies. https://blazingseollc.com/proxy/ipv6-proxies/.

[10] California Consumer Privacy Act. https://en.wikipedia.org/wiki/California_Consumer_Privacy_Act.

[11] Taejoong Chung, David Choffnes, and Alan Mislove. Tunneling for transparency: A large-scale analysis of end-to-end violations in the internet. In *Proceedings of the 2016 Internet Measurement Conference*, pages 199–213. ACM, 2016.

[12] Discussion of Luminati's SDK in an Android monetization forum. http://forums.makingmoneywithandroid.com/advertising-networks/47226-luminati-sdk.html.

[13] Fobus, the sneaky little thief that could. https://blog.avast.com/2015/01/15/fobus-the-sneaky-little-thief-that-could/.

[14] Frida Gadget. https://frida.re/docs/gadget/.

[15] General Data Protection Regulation. https://gdpr-info.eu/.

[16] Liangyi Gong, Zhenhua Li, Feng Qian, Zifan Zhang, Qi Alfred Chen, Zhiyun Qian, Hao Lin, and Yunhao Liu. Experiences of landing machine learning onto market-scale mobile malware detection. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–14, 2020.

[17] Google Play Developer Policy Center. https://play.google.com/about/privacy-security-deception/device-network-abuse/.

[18] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. Riskranker: scalable and accurate zero-day android malware detection. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 281–294. ACM, 2012.

[19] HTTP tunnel. https://en.wikipedia.org/wiki/HTTP_tunnell.

[20] Hybrid Analysis. https://www.hybrid-analysis.com/.

[21] Intoli. https://intoli.com/.

[22] IoT Devices as Proxies for Cybercrime. https://krebsonsecurity.com/2016/10/iot-devices-as-proxies-for-cybercrime/.

[23] IP Ninja. https://ipninja.io/.

[24] IPinfo. https://ipinfo.io/developers.

[25] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: illuminating the edge network. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 246–259. ACM, 2010.

[26] Lethean. https://lethean.io/.

[27] Menghao Li, Wei Wang, Pei Wang, Shuai Wang, Dinghao Wu, Jian Liu, Rui Xue, and Wei Huo. Libd: scalable and precise third-party library detection in android markets. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 335–346. IEEE, 2017.

[28] Luminati. https://luminati.io/.

[29] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen. Libradar: fast and accurate detection of third-party libraries in android apps. In *Proceedings of the 38th international conference on software engineering companion*, pages 653–656. ACM, 2016.

[30] Akshaya Mani, Tavish Vaidya, David Dworken, and Micah Sherr. An extensive evaluation of the internet's open proxies. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 252–265. ACM, 2018.

[31] Niall McLaughlin, Jesus Martinez del Rincon, BooJoong Kang, Suleiman Yerima, Paul Miller, Sakir Sezer, Yeganeh Safaei, Erik Trickel, Ziming Zhao, Adam Doupé, et al. Deep android malware detection. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 301–308. ACM, 2017.

[32] Xianghang Mi, Xuan Feng, Xiaojing Liao, Baojun Liu, XiaoFeng Wang, Feng Qian, Zhou Li, Sumayah Alrwais, Limin Sun, and Ying Liu. Resident evil: Understanding residential ip proxy as a dark service. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1185–1201. IEEE, 2019.

[33] MonkeySocks. https://monkeysocks.net/.

[34] netnut. https://netnut.io/.

[35] Nmap Service Probes. https://svn.nmap.org/nmap/nmap-service-probes.

[36] Mark O'Neill, Scott Ruoti, Kent Seamons, and Daniel Zappala. Tls proxies: Friend or foe? In *Proceedings of the 2016 Internet Measurement Conference*, pages 551–557. ACM, 2016.

[37] Oxylabs. https://oxylabs.io.

[38] p0f. http://lcamtuf.coredump.cx/p0f3/.

[39] Diego Perino, Matteo Varvello, and Claudio Soriente. Proxytorrent: Untangling the free http (s) proxy ecosystem. In *Proceedings of the 2018 World Wide Web Conference*, pages 197–206. International World Wide Web Conferences Steering Committee, 2018.

[40] Bradley Reaves, Ethan Shernan, Adam Bates, Henry Carter, and Patrick Traynor. Boxed out: Blocking cellular interconnect bypass fraud at the network edge. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 833–848, 2015.

[41] SDK for monetization of android installs - monkeysocks.net. https://forum.xda-developers.com/monetization/general/sdk-monetization-android-installs-t3814192, 2018.

[42] SimilarWeb. https://www.similarweb.com/.

[43] Smartproxy. https://smartproxy.com/.

[44] SOCKS Proxy SDKs a New Risk for Enterprises. https://securityboulevard.com/2018/04/socks-proxy-sdks-a-new-risk-for-enterprises/, 2018.

[45] Tapcore. https://tapcore.com/en.

[46] The Rise of "Bulletproof" Residential Networks. https://krebsonsecurity.com/2019/08/the-rise-of-bulletproof-residential-networks/, 2019.

[47] Giorgos Tsirantonakis, Panagiotis Ilia, Sotiris Ioannidis, Elias Athanasopoulos, and Michalis Polychronakis. A large-scale analysis of content modification by open http proxies. In *NDSS*, 2018.

[48] VirusTotal APIs. https://developers.virustotal.com/reference.

[49] Nicholas Weaver, Christian Kreibich, Martin Dam, and Vern Paxson. Here be web proxies. In *International Conference on Passive and Active Network Measurement*, pages 183–192. Springer, 2014.

[50] Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, and Kuo-Ping Wu. Droidmat: Android malware detection through manifest and api calls tracing. In *2012 Seventh Asia Joint Conference on Information Security*, pages 62–69. IEEE, 2012.

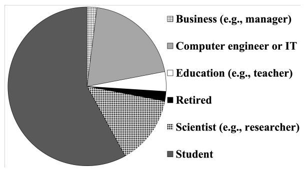
[51] Zhenlong Yuan, Yongqiang Lu, Zhaoguo Wang, and Yibo Xue. Droid-sec: deep learning in android malware detection. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 371–372. ACM, 2014.

[52] Yin Zhang and Vern Paxson. Detecting stepping stones. In *USENIX Security Symposium*, volume 171, page 184, 2000.

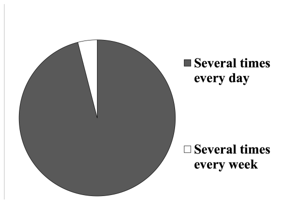
IX. Appendix

A. User Study Questions

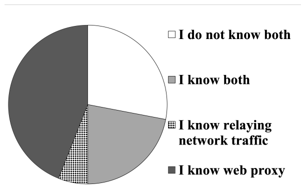
1) What is your age group?
   a) 18 ∼ 24
   b) 25 ∼ 34
   c) 35 ∼ 44
   d) 45 ∼ 54



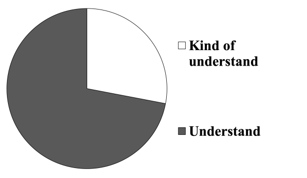
(a) *Q4: Which of the following best describes your primary occupation?*



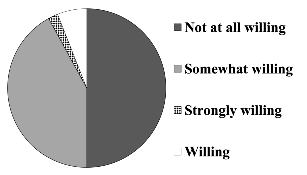
(b) *Q6: How often do you use mobile apps on your phone?*



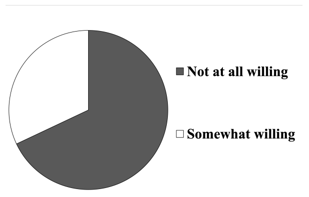
(c) *Q8: Do you know what is web proxy? Do you know what is relaying network traffic?*



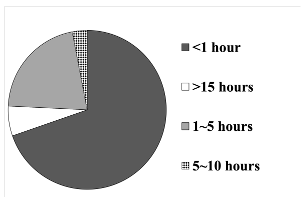
(d) *Q9: After reading the hints, do you understand what is web proxy and what is relaying network traffic?*



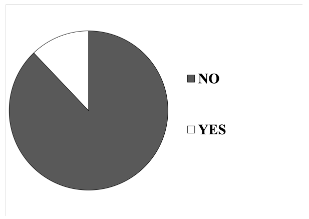
(e) *Q10: To what extent you are willing to turn your mobile device into a web proxy, if the bonus is using that app without ads or subscriptions?*



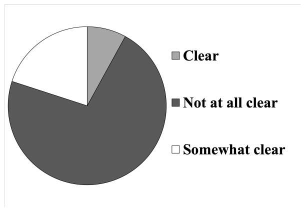
(f) *Q11: If the bonus is providing you with a small amount of money, to what extent you are willing to do so?*



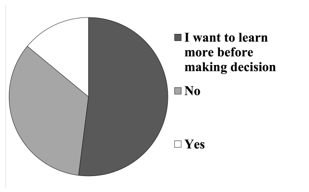
(g) *Q12: How long do you prefer to allow that app to proxy network traffic everyday?*



(h) *Q13: Would you allow proxy to consume your cellular data when you are surfing the cellular network?*



(i) *Q16: Do you think the popup expresses clearly that it requests to turn your device into a web proxy and relay unknown traffic?*



(j) *Q19: Do you agree to share your data with this app and its third-party partners?*

Fig. 6: Distributions of the user perspectives to part of user study questions

   e) $55 \sim 64$
   f) 65+

2) What is your gender?
   a) Male
   b) Female
   c) Decline to answer
3) What is the highest education level you have completed?
   a) Under high school
   b) Some high school
   c) High school graduate
   d) Some college - No degree
   e) Associates / 2-year degree
   f) Bachelor / 4-year degree
   g) Graduate degree - Master, PhD, professional, medicine, etc
4) Which of the following best describes your primary occupation?
   a) Administrative support (e.g., secretary, assistant)
   b) Art, writing, or journalism (e.g., author, reporter, sculptor)
   c) Business, management, or financial (e.g., manager, accountant, banker)
   d) Computer engineer or IT professional (e.g., systems administrator, programmer, IT consultant)
   e) Education (e.g., teacher)
   f) Legal (e.g., lawyer, law clerk)
   g) Homemaker
   h) Retired
   i) Medical (e.g., doctor, nurse, dentist)
   j) Service (e.g., retail clerks, server)
   k) Scientist (e.g., researcher, professor)
   l) Student
   m) Skilled labor (e.g., electrician, plumber, carpenter)
   n) Unemployed
   o) Decline to answer
   p) Other (Please specify)
5) Approximately how long have you been using mobile apps on your phone?
   a) 0-31 days
   b) 1-6 months
   c) 6-12 months
   d) 1-2 years
   e) 3 and more years
6) How often do you use mobile apps on your phone?
   a) Several times every day
   b) Several times every week
   c) Several times every month
   d) A few times a year
   e) Very rare
7) Suppose Figure 7(a) is a mobile app's pop-up request screen. The name "app name" is the specific app you want to use. Please describe your idea/understanding regarding the descriptive words in this pop-up using your own

words.[1]

8) Do you know what is web proxy? Do you know what is relaying network traffic? Choose all that apply.
   - ☐ I know web proxy
   - ☐ I know relaying network traffic
   - ☐ I do not know web proxy
   - ☐ I do not know relaying network traffic

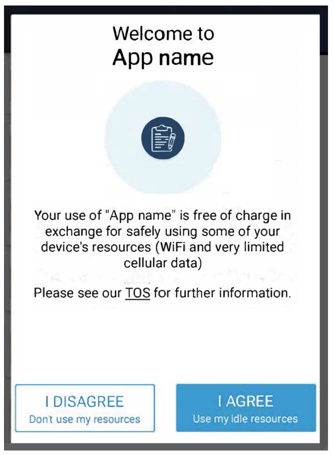
Hints.

Web proxies: web proxies are used to send web requests on behalf of web clients, to web servers. Web proxies can serve various purposes such as identity anonymization and circumventing censorship. For instance, China blocks access to Google Search. However, a user in China can send the web request through a web proxy outside China to access Google Search.

Relaying network traffic: web proxies serve to relay network traffic, sourcing from web clients, to web servers.

9) After reading the hints, do you understand what is web proxy and what is relaying network traffic?
   a) Understand
   b) Kind of understand
   c) Still don't understand at all
10) Suppose you install a mobile app on your phone, and that app requests to turn your mobile device into a web proxy and relay network traffic of unknown third parties. If that app also provides you with some bonuses, such as using that app without ads or subscriptions, to what extent are you willing to do so?
    a) Strongly willing.
    b) Willing.
    c) Somewhat willing.
    d) Not at all willing.
11) If the bonus is providing you with a small amount of money, to what extent you are willing to do so?
    a) Strongly willing.
    b) Willing.
    c) Somewhat willing.
    d) Not at all willing.
12) How long do you prefer to allow that app to proxy network traffic everyday?
    a) <1 hour.
    b) $1 \sim 5$ hours.
    c) $5 \sim 10$ hours.
    d) $10 \sim 15$ hours.
    e) >15 hours.
13) Would you allow proxy to consume your cellular data when you are surfing the cellular network?
    a) Yes.
    b) No.
14) How much data do you allow the app to use?
    a) <1 MB.

[1]This question is designed as an open-ended question.


Welcome to
App name

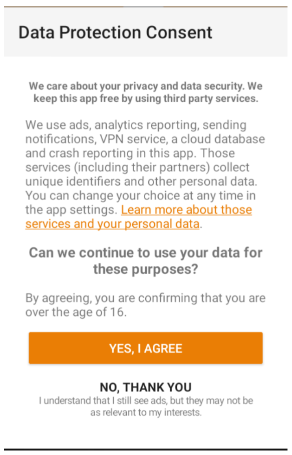
Your use of "App name" is free of charge in exchange for safely using some of your device's resources (WiFi and very limited cellular data)

Please see our TOS for further information.

I DISAGREE
Dont use my resources

I AGREE
Use my idle resources

(a) Luminati pop-up


Data Protection Consent

We care about your privacy and data security. We keep this app free by using third party services.

We use ads, analytics reporting, sending notifications, VPN service, a cloud database and crash reporting in this app. Those services (including their partners) collect unique identifiers and other personal data. You can change your choice at any time in the app settings. Learn more about those services and your personal data.

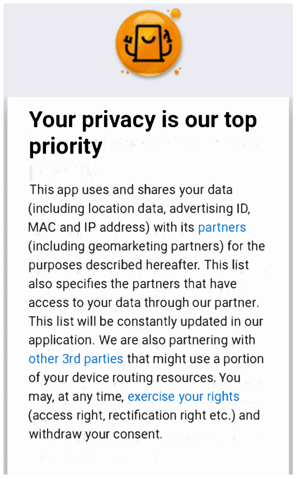
Can we continue to use your data for these purposes?

By agreeing, you are confirming that you are over the age of 16.

YES, I AGREE

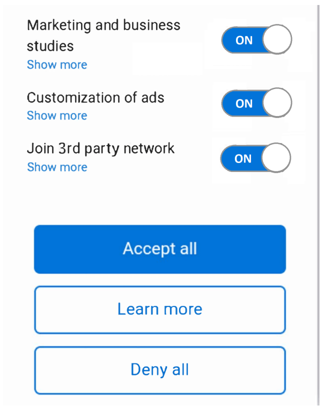
NO, THANK YOU
I understand that I still see ads, but they may not be as relevant to my interests.

(b) Ipninja pop-up


Your privacy is our top priority

This app uses and shares your data (including location data, advertising ID, MAC and IP address) with its partners (including geomarketing partners) for the purposes described hereafter. This list also specifies the partners that have access to your data through our partner. This list will be constantly updated in our application. We are also partnering with other 3rd parties that might use a portion of your device routing resources. You may, at any time, exercise your rights (access right, rectification right etc.) and withdraw your consent.

(c) Ipninja policy


Marketing and business studies — ON
Show more

Customization of ads — ON
Show more

Join 3rd party network — ON
Show more

Accept all

Learn more

Deny all

(d) Ipninja customerization

Fig. 7: Dialogs and setting figures used in user study

    b) $1 \sim 10$ MB.
    c) $10 \sim 50$ MB.
    d) $50 \sim 100$ MB.
    e) >100 MB.
15) What are your concerns regarding turning your device into a web proxy and relaying unknown traffic? Choose all that apply.
    - ☐ Traffic rate.
    - ☐ My cellular data.
    - ☐ Privacy.
    - ☐ Legal issues.
    - ☐ This may consume my phone battery.
    - ☐ Others (Please specify).
16) Do you think Figure 7(a) expresses clearly that it requests to turn your device into a web proxy and relay unknown traffic?
    a) Extremely clear.
    b) Clear.
    c) Somewhat clear.
    d) Not at all clear.

17) Have you ever met with similar kind of apps requesting turning your device into a proxy and relaying traffic?
    a) Yes.
    b) No.
18) Please specify the name of those apps.[2]
19) Figure 7(b) is another mobile app's pop-up request screen. Comparing with the previous app example, this app is requesting other privileges. Based on its description, do you agree to share your data with this app and its third-party partners?
    a) Yes.
    b) I want to learn more before making decision.
    c) No.
20) What unique identifiers and other personal data do you think they may collect?[3]
21) Suppose you want to learn more about those services and data the app and its partners collect. Figure 7(c) provides such information. Based on figure above, choose data and services that you would allow this app and third-party partners to use. Choose all that apply.
    ☐ Location data.
    ☐ Ads that I am interested in.
    ☐ My physical device number and IP address.
    ☐ Some routing resources (including but not limited to proxy network traffic, cellular data).
    ☐ None above.
22) For each choice above, please explain why you would allow or not allow.[4]
23) Figure 7(d) provides services that you could customize whether you agree to use or not. All those services turn ON by default. Which would you like to turn OFF? Choose all that apply.
    ☐ Marketing and business studies.
    ☐ Customization of ads.
    ☐ Join 3rd party network.
    ☐ I agree to use all.
24) Since you can rectify your choice whenever you want, do you think you will change your choices in question above after beginning to use the app?
    a) Yes.
    b) Maybe.
    c) No.

B. *List of Proxy Providers*

As introduced in §III-A, we have identified 38 residential proxy providers, as listed in Table IX.

TABLE IX: The list of residential proxy providers.

Provider	Website	Provider	Website
Airsocks	airsocks.in	Anonymous	anonymous-proxies.net
Apify	apify.com	Atcproxys	atcproxys.com
Blazingseollc	blazingseollc.com	Blitzproxies	blitzproxies.com
Buyproxies	buyproxies.org	Cloudproxies	cloudproxies.com
Cosmoproxy	cosmoproxy.com	Dropclub	dropclub.io
Geosurf	geosurf.com	Gimmeproxy	gimmeproxy.com
Hprox	hprox.com	Icedoutproxies	icedoutproxies.com
Infatica	infatica.io	Intoli	intoli.com
IPninja	ipninja.io	Lethean	lethean.io
Localproxies	localproxies.com	Luminati	luminati.io
Microleaves	microleaves.com	MonkeySocks	monkeysocks.net
Netnut	netnut.io	Oxylabs	oxylabs.io
Penguinproxy	penguinproxy.com	Privatix	privatix.network
Proxies	proxies.online	ProxyRack	proxyrack.com
Proxyrotator	proxyrotator.com	Residentialips	residentialips.io
Resvpn	resvpn.com	Rotatingproxies	rotatingproxies.com
Smartproxy	smartproxy.io	Sockshub	sockshub.net
Soleproxy	soleproxy.com	StormProxies	stormproxies.com
Surgeproxies	surgeproxies.com	Tuxler	tuxler.com

[2] This question is designed as an open-ended question.
[3] This question is designed as an open-ended question.
[4] This question is designed as an open-ended question. Four fields mentioned in question 21 are given