An Analysis of the Privacy and Security Risks of Android VPN Permission-enabled Apps

Muhammad Ikram[1,2], Narseo Vallina-Rodriguez[3], Suranga Seneviratne[1],
Mohamed Ali Kaafar[1], Vern Paxson[3,4]
[1]Data61, CSIRO [2]UNSW [3]ICSI [4]UC Berkeley

ABSTRACT

Millions of users worldwide resort to mobile VPN clients to either circumvent censorship or to access geo-blocked content, and more generally for privacy and security purposes. In practice, however, users have little if any guarantees about the corresponding security and privacy settings, and perhaps no practical knowledge about the entities accessing their mobile traffic.

In this paper we provide a first comprehensive analysis of 283 Android apps that use the Android VPN permission, which we extracted from a corpus of more than 1.4 million apps on the Google Play store. We perform a number of passive and active measurements designed to investigate a wide range of security and privacy features and to study the behavior of each VPN-based app. Our analysis includes investigation of possible malware presence, third-party library embedding, and traffic manipulation, as well as gauging user perception of the security and privacy of such apps. Our experiments reveal several instances of VPN apps that expose users to serious privacy and security vulnerabilities, such as use of insecure VPN tunneling protocols, as well as IPv6 and DNS traffic leakage. We also report on a number of apps actively performing TLS interception. Of particular concern are instances of apps that inject JavaScript programs for tracking, advertising, and for redirecting e-commerce traffic to external partners.

1. INTRODUCTION

Since the release of Android version 4.0 in October 2011, mobile app developers can use native support to create VPN clients through the Android VPN Service class. As opposed to the desktop context, where an app needs root access to create virtual interfaces, Android app developers only have to request the `BIND_VPN_SERVICE` permission (for simplicity, the "VPN permission") to create such clients.

Android's official documentation highlights the serious security concerns that the VPN permission raises: it allows an app to intercept and take full control over a user's traffic [60]. Many apps may legitimately use the VPN permission to offer (some form of) online anonymity or to enable access to censored content [84]. However, malicious app developers may abuse it to harvest users' personal information. In order to minimize possible misuse, Android alerts users about the inherent risks of the VPN permission by displaying system dialogues and notifications [60]. A large fraction of mobile users may however lack the necessary technical background to fully understand the potential implications.

The use of the VPN permission by mobile apps, many of which have been installed by millions of users worldwide, remains opaque and undocumented. In this paper, we conduct in-depth analysis of 283 Android VPN apps extracted from a population of 1.4M Google Play apps. In our efforts to illuminate and characterize the behavior of VPN apps and their impact on user's privacy and security, we develop a suite of tests that combines passive analysis of the source code (cf. Section 4) with custom-built active network measurements (cf. Section 5). The main findings of our analysis are summarized as follows:

- **Third-party user tracking and access to sensitive Android permissions:** Even though 67% of the identified VPN Android apps offer services to enhance online privacy and security, 75% of them use third-party tracking libraries and 82% request permissions to access sensitive resources including user accounts and text messages.
- **Malware presence:** While 37% of the analyzed VPN apps have more than 500K installs and 25% of them receive at least a 4-star rating, over 38% of them contain some malware presence according to VirusTotal [57]. We analyze the public user reviews available on Google Play for all the VPN apps to sense whether their users are aware of possible malicious activities in their apps. Our analysis reveals that only a marginal number of VPN users have publicly raised any security and privacy concerns in their app reviews.
- **Traffic interception modes:** The hosting infrastructure of VPN apps, which is heavily concentrated in the USA,

IMC 2016, November 14-16, 2016, Santa Monica, CA, USA
 ISBN 978-1-4503-4526-2/16/11…$15.00
DOI: http://dx.doi.org/10.1145/2987443.2987471

remains opaque for the end-user. 18% of the apps do not mention the entity hosting the terminating VPN server. Our network measurements also suggest that 16% of the analyzed apps may forward traffic through other participating users in a peer-forwarding fashion rather than using machines hosted in the cloud. This forwarding model raises a number of trust, security, and privacy concerns for participating users. Finally, 4% of the analyzed VPN apps use the VPN permission to implement localhost proxies to intercept and inspect user traffic locally, primarily for antivirus and traffic filtering purposes.

- **(Lack of) Encryption and traffic leaks:** 18% of the VPN apps implement tunneling protocols without encryption despite promising online anonymity and security to their users. In fact, approximately 84% and 66% of the analyzed VPN apps do not tunnel IPv6 and DNS traffic through the tunnel interface respectively due to lack of IPv6 support, misconfigurations or developer-induced errors. Both the lack of strong encryption and traffic leakages can ease online tracking activities performed by in-path middleboxes (*e.g.,* commercial WiFi APs harvesting user's data) and by surveillance agencies.

- **In-path proxies and traffic manipulation:** 16% of the analyzed VPN apps deploy non-transparent proxies that modify user's HTTP traffic by injecting and removing headers or performing techniques such as image transcoding. However, the artifacts implemented by VPN apps go beyond the typical features present in HTTP proxies. We identified two VPN apps actively injecting JavaScript code on user's traffic for advertisement and tracking purposes and one of them redirects e-commerce traffic to external advertising partners.

- **TLS interception:** Four of the analyzed VPN apps compromise users' root-store and actively perform TLS interception in the flight. Three of these apps claim providing traffic acceleration services and selectively intercept traffic to specific online services like social networks, banking, e-commerce sites, email and IM services and analytics services.

Our results show that — in spite of the promises for privacy, security and anonymity given by the majority of VPN apps — millions of users may be unawarely subject to poor security guarantees and abusive practices inflicted by VPN apps. However, this study has not answered several interesting research questions such as traffic discrimination [83] and the detection of side-channels to extract additional private information from user's phones.

2. ANDROID'S VPN PERMISSION

Google introduced native platform support for VPN clients through the VPN Service base class and its associated `BIND_VPN_SERVICE` permission in Android version 4.0 [60]. For simplicity, we will reference the `BIND_VPN_SERVICE` permission as the "VPN permission".

Vendor	Custom Permission
Cisco	com.cisco.anyconnect.vpn.android.MODIFY_VPN
Juniper	com.juniper.permission.JUNIPER_VPN_ACCESS
Samsung	android.permission.sec.MDM_VPN
KNOX	android.permission.sec.MDM_ENTERPRISE_VPN

Table 1: Custom VPN permissions for MDM apps.

The `BIND_VPN_SERVICE` permission is a powerful Android feature that app developers can misuse or abuse. It allows the requesting app to intercept, manipulate and forward all user's traffic to a remote proxy or VPN server of their choice or to implement proxies in localhost [93].

Android's VPN API exposes a virtual network interface to the requesting app and — if the developer configures correctly the routing tables — routes all device's traffic to it. Likewise, each write operation to the virtual interface injects a packet just like it was received from the external interface. As for any other Android permission, app developers must explicitly declare access to the VPN permission in the app's `AndroidManifest` file [2] but Android limits the creation and ownership of the virtual interface to only one app at a given time.

Due to the exceptional security and privacy risks of allowing third-party apps to intercept all user's traffic, Android generates two warnings to notify user's whenever an app creates a virtual interface using the VPN permission: (i) a system dialog seeking users approval to create a virtual interface, and (ii) a system-generated notification that informs users as long as the VPN interface remains active [60]. However, average mobile users may not fully understand, possibly due to the lack of technical background, the consequences of allowing a third-party app to read, block and/or modify their traffic.

Custom VPN permissions: Android's native VPN support has enabled proprietary VPN solutions for enterprise clients such as Cisco AnyConnect [5] and Juniper Junos [27] technologies. Enterprise solutions, also known as Mobile Device Management solutions or MDM, implement their own tunneling protocols on top on Android's VPN permission to secure and simplify remote access to enterprise or private networks. Samsung's KNOX SDK [95] is a different incarnation of proprietary MDM solutions. In that case, Samsung takes advantage from it's position as an Android OS vendor to completely replace Android's VPN implementation at the firmware level with their own solution.

Android's permission model allows MDM providers to share their VPN technologies with other apps by defining custom permissions. These are listed in Table 1. The requesting app must declare the associated custom permission on its manifest and the app providing the proprietary technology must be already installed on the device. In the case of Samsung's KNOX-enabled devices, the app developer wishing to incorporate any KNOX feature in its app must first enroll on Samsung's KNOX program and then request access to the proprietary SDK [28, 44]. As a result, Android VPN apps without KNOX support may operate incorrectly on many Samsung devices. The security guarantees that ap-

ply for the official VPN permission also apply for custom VPN permissions as the MDM solution is responsible to request Android's `BIND_VPN_SERVICE` permission.

3. DISCOVERING VPN APPS ON GOOGLE PLAY

This section describes our method for identifying and characterizing Android VPN-enabled apps on Google Play.

3.1 Detection Method

Identifying VPN-enabled apps on Google Play is not a trivial task. The list of permissions available on a given app's Google Play profile does not necessarily contain the use of the VPN permission by the app.

App developers can request the Android VPN permissions in their app manifest file in two different ways: they can request the VPN permission within the scope of the whole app or restrict its use to an specific activity or service[1] using the <*activity*> and <*service*> tags respectively. This subtle difference has an impact on any method aiming to detect VPN-enabled apps: when a developer declares the permission within the <*service*> tag, the VPN permission does not show up in the list of Android permissions available on Google Play. Consequently, in order to correctly identify VPN-enabled apps at scale — either those using Android's official permission or any of the custom VPN permissions listed in Table 1 —, we must crawl Google Play to download each app's executable and then decompile it to inspect their `AndroidManifest` file in detail.

We rely on multiple tools to fetch each app's metadata (*e.g.,* app description, installs, developer, user reviews and app rating) and to download their executables. For free apps, we use Google Play Unofficial Python API [**?**] whereas for paid apps, we use Raccoon APK Downloader to obtain the binaries after paying their required fee [42]. Finally, after having downloaded each app's executable, we use ApkTool[2] to decompile, extract and analyze each app's source code and their `AndroidManifest` file.

To increase our app coverage and maximize the number of detected VPN apps, we implemented a Google Play crawler that uses two complementary seeds. First, we obtain the app ID (or package name) from the top 100 apps for four Google Play categories likely to contain VPN and MDM apps: tools, communication, business and productivity. Second, we leverage Google Play's search feature to find apps containing VPN-related keywords like "vpn", "virtual private network", "security", "censorship", "anonymity" or "privacy" in their app description. Afterwards, our crawler fetches each app's metadata and executables. Our crawler follows a breadth-first-search approach for any other app considered as *"similar"* by Google Play and for other apps published by the same developer. In total, this method has allowed us to survey 1,488,811 apps during a three week period in September 2015.

[1]Android apps can be composed of multiple activities (*i.e.,* app components that run on the foreground on a single screen and require user interaction) and services (*i.e.,* app components that perform long-running operations in the background) [3]. Permission requests can be limited to specific app components.

[2]https://ibotpeaches.github.io/apktool

App pricing model	# of apps ($N = 283$)	# of apps analyzed in § 5
Free VPN apps with Free Services	130	130
Free VPN apps with Premium Services	153	20

Table 2: Number of VPN apps identified with our detection method.

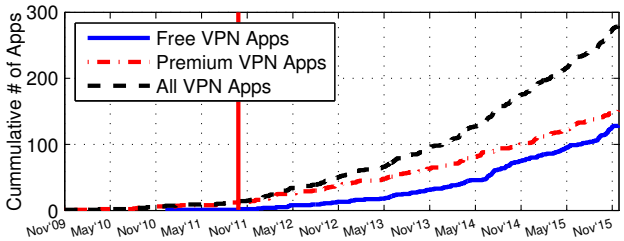

Figure 1: Evolution of VPN-enabled apps' availability on Google Play.

Our method has allowed us to identify 283 free Android apps requesting the VPN permission in their `AndroidManifest` files. 153 of free VPN apps require the user to perform in-app purchases in order to use their online VPN services. We refer to such apps as "premium VPN apps" and they typically offer weekly, monthly, quarterly and yearly subscriptions. In the case of paid apps, we relied on information available on the app description as signals to identify potential paid VPN apps. This is the result of our inability to pay the fee for downloading the executables of each paid app listed on Google Play. This approach has allowed us to find 10 potential VPN paid apps. However, after paying their fee to download their executables, only one of them actually requested the VPN permission. Therefore, we decided to exclude paid VPN apps from this study.

Our dynamic network analysis (presented in Section 5) covers the 130 free apps and 20 premium VPN apps. Unfortunately, we could not inspect the entirety of premium VPN apps as most of them are full MDM solutions which require dedicated IT and cloud support. Table 2 summarizes the scope of our static and dynamic analysis.

3.2 The Rise of VPN Apps

This section studies the presence of VPN-enabled apps available for download on Google Play over time. Given that Google Play does not report the actual release date of the apps but their last update, we use the date of their first comment as a proxy for their release date. For 9 apps without any user reviews as of this writing, we can determine the approximate release date by their last update.

Figure 1 shows the steady increase of VPN apps' listed on Google Play since November 2011 (Android 4.0 release). Note that our analysis only considers apps listed on Google Play as of September 2015 so it excludes possible VPN apps removed from Google Play. During the 2-year period that spans between November 2011 and November 2013, the number of VPN apps increased ten-fold.

App Category	% of Apps ($N = 283$)
VPN Clients	67
Enterprise	10
Traffic Optimizer	4
Communication Tools	3
Traffic filters	2
Traffic logger	2
Antivirus	1
Tor clients	1
Other	10

Table 3: Manual classification of VPN apps by their purpose.

The analysis reveals that a small group of MDM apps like Juniper's Junos Pulse and Afaria [20] were already listed on Google Play years before the release of Android v4.0 (represented in the graph with the vertical line). Unfortunately, we cannot obtain the deprecated binaries of these apps for further inspection to report how they implemented (or not) their VPN solutions before Android provided native support. We speculate that they have relied either on users to manually enter the VPN server on Android's system settings or on users with rooted phones.

During the preparations for the final manuscript on August 5, 2016, we noticed that 49 out of 283 analyzed VPN apps were no longer listed on Google Play either as a result of Google's vetting process, user complaints, or due to developer decisions.

3.3 VPN App Classification

VPN apps can provide a wide range of services to the user. Unfortunately, Google Play's categories (*e.g.,* tools and games) are too broad to capture the actual purpose of the app. In order to identify their actual intended functionality, two co-authors inspected and labeled each VPN app manually according to their Google Play app description into 9 categories that we list in Table 3. In case that an app advertises more than one functionality, we choose the most relevant one. We found no disagreements in the labeling process.

67% of Android VPN apps claim to provide traditional VPN services (labeled here as "VPN clients") including enhanced security and privacy, anti-surveillance or tunnels to access geo-filtered or censored content. Note that we consider Tor clients (*e.g.,* Orbot [37], Globus VPN [62] and TorGuard VPN client [56]) as a separate category. The second most common category is enterprise MDM solutions (10% of apps) followed by traffic optimization tools (*e.g.,* DashNet [9], 4% of apps) and communication tools (3% of apps) for tethering or for creating mesh networks and VLANs (typically for online gaming [32]).

Antivirus software apps (Qihoo 360 [41], Dr.Web Security Space [13], and TrendMicro's Mobile Security & Antivirus [29]) may also leverage the VPN permission to perform traffic analysis (*e.g.,* malware detection), to block malicious traffic and to securely forward user's traffic through trusted servers when users connect through insecure or questionable WiFi networks. Other uses of the VPN permission are traffic filters and traffic loggers (*e.g.,* NoRoot Firewall [33]) and even apps for securing online payments (*e.g.,* Fast Secure Payment [17]).

4. STATIC ANALYSIS

In this section, we analyze the source code for each VPN Android app using static analysis. In particular, we report on applications requesting sensitive permission analysis, the presence of tracking libraries in app's decompiled source code and the presence of malware activity according to the online antivirus aggregator, VirusTotal[3].

4.1 Permission Analysis

We investigate how VPN-enabled apps request other Android permissions to access sensitive system resources. We exclude network-related permissions like Internet access which are inherent to any VPN client.

Figure 2 compares the permissions requested by VPN-enabled apps with those requested by the top-1,000 free non-VPN Android apps[4], which we included for reference. We use the method-to-permission mapping provided by Au *et al.* [67] to investigate the source code segments invoking the methods protected by each Android permission. For instance, in the case of apps requesting the `READ_SMS` permission, we investigate apps' calls to associated methods such as `preSendSmsWorker` (a method used to send SMS which informs the user about the intended or wanted text) and `handleSmsReceived` (a method that handles formatting-related aspects in received SMS) in order to determine the actual use of the permission by the app.

There are Android permissions that are more common on VPN apps than in other app categories. For instance, antivirus and MDM solutions request `READ_LOGS` permission to inspect other apps' activities [2]. However, we observe that standard VPN clients like DroidVPN [12] and tigerVPN [54] also request permission to read system logs. Android documentation [2] flags this permission as highly sensitive as any app developer may carelessly misuse Android's logging capabilities and (unintentionally) expose personal information (including passwords) to any other apps requesting it. Similarly, antivirus apps request `READ_EXTERNAL_STORAGE` permission to check the stored files for possible virus and malware activity.

Many other permissions listed in Figure 2 may appear unusual requirements for VPN apps. However, VPN apps may provide additional and richer features to their users beyond a typical VPN tunnel. For each case, we manually checked the legitimacy of these requests by inspecting the API calls executed by the apps and checking the description for related functionalities without finding any evidence for deliberate abuse of granted permissions. For instance, we found that antivirus apps as well as spyware VPN apps (which we further investigate in Section 4.3) request the `READ_SMS` permission to read text messages and, in the case of antivirus apps, to scan them for possible malware presence. Similarly, apps requesting `READ_CONTACTS` incorporate functions in

[3]https://www.virustotal.com

[4]According to Google Play's ranking as of March 30, 2016.

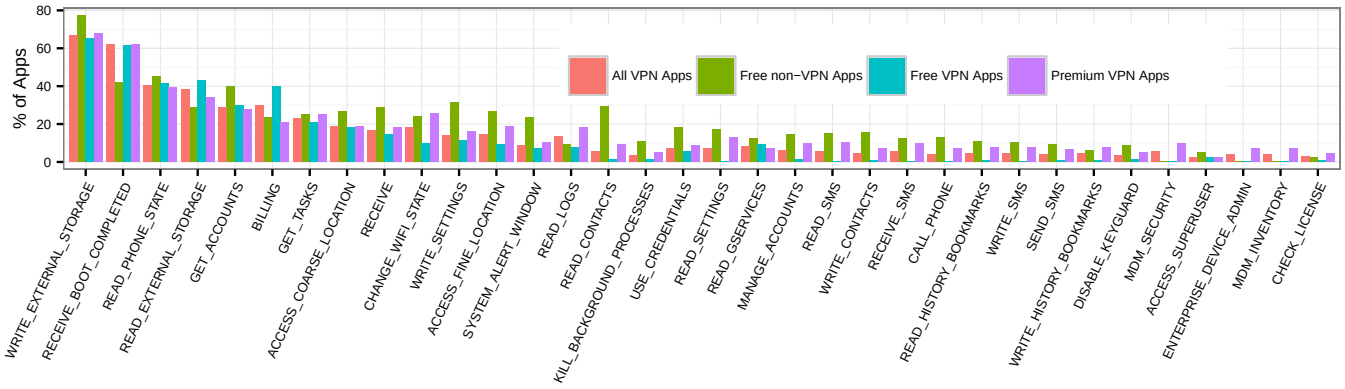

Figure 2: Detailed comparison of Android permissions (x-axis) requested by VPN apps and the top-1,000 non-VPN apps.

# Trackers	VPN Apps			Free non-VPN Apps
	Premium	Free	All	
0	65%	28%	33%	19%
1	13%	10%	8%	11%
2	10%	10%	7%	15%
3	12%	25%	13%	23%
4	2%	8%	4%	16%
≥5	5%	18%	8%	17%

Table 4: Distribution of third party trackers embedded in VPN apps.

#	App ID	Class	Rating	# Installs	AV-rank
1	OkVpn [34]	Prem.	4.2	1K	24
2	EasyVpn [15]	Prem.	4.0	50K	22
3	SuperVPN [52]	Free	3.9	10K	13
4	Betternet [19]	Free	4.3	5M	13
5	CrossVpn [7]	Free	4.2	100K	11
6	Archie VPN [4]	Free	4.3	10K	10
7	HatVPN [21]	Free	4.0	5K	10
8	sFly Network Booster [47]	Prem.	4.3	1K	10
9	One Click VPN [35]	Free	4.3	1M	6
10	Fast Secure Payment [17]	Prem.	4.1	5K	5

Table 5: VPN Apps with a VirusTotal AV-rank ≥ 5.

the likes of blocking text and calls from specific phone numbers or sharing features through SMS or email.

4.2 Tracking Libraries in VPN Apps

With the help of ApkTool, we examine the presence of embedded third-party libraries (in the form of external jar files) for analytics, tracking or advertising purposes in the source code of each VPN-enabled app. In order to identify which libraries are associated with tracking services, we use the manually curated list of 127 tracking and advertising libraries compiled by Seneviratne *et al.* [97]. Therefore, we consider our results as a lower bound of third-party tracking libraries presence in VPN apps.

Table 4 compares the number of trackers used by VPN-enabled apps with the presence of trackers in the reference set of 1,000 free non-VPN apps. 67% of the VPN apps embed at least one third-party tracking library in their source code. The use of tracking libraries in VPN apps is significantly lower than in the top 1,000 non-VPN apps with an almost 81% of the latter having at least one embedded tracking library. The fact that 65% of the premium VPN apps do not have any tracking library embedded (as opposed to only 28% of the free VPN apps) suggests that premium apps do not rely as much as free apps on revenues from advertising and analytics services.

Since most VPN apps intend to provide online anonymity (Section 3.3), the lower presence of tracking libraries is actually meaningful. However, we identified the presence of at least one tracking library in 75% of the free VPN apps claiming to protect users' privacy. 8% of all VPN apps have more than five. In particular, two VPN apps (Flash Free VPN [18] and Betternet [19]), which combined have more than 6M installs, have the highest number of embedded tracking libraries: 11 and 14 respectively.

Figure 3 ranks the top-25 popular trackers in all analyzed VPN apps. Google Ads and Google Analytics are the most popular trackers among our corpus of VPN apps. A closer examination at the long-tail of the distribution reveals however that the least popular third-party tracking libraries in our reference set of 1000 apps are instead more common in VPN apps. For instance, VPN apps like SurfEasy [53] and Ip-Shield VPN [26] integrate libraries like NativeX[5] and Appflood[6] for monetizing their apps with targeted ads.

4.3 Malware Analysis

Malware components may be designed to circumvent a specific antivirus (AV) tool [105]. As a result, it is imperative to rely upon multiple AV scanners and datasets to effectively identify the presence of malware on mobile VPN apps. We leverage the capabilities offered by VirusTotal's public API to automatize our malware detection process. VirusTotal is an online solution which aggregates the scanning capabilities provided by more than 100 AV tools, scanning engines and datasets. It has been commonly used in the academic literature to detect malicious apps, executables, software and domains [**?**, **?**, **?**].

After completing the scanning process for a given app, VirusTotal generates a report that indicates which of the participating AV scanning tools detected any malware activity in the app and the corresponding malware signature (if any). Given that a single scanning tool may produce false posi-

[5]http://www.nativex.com
[6]http://www.appflood.com

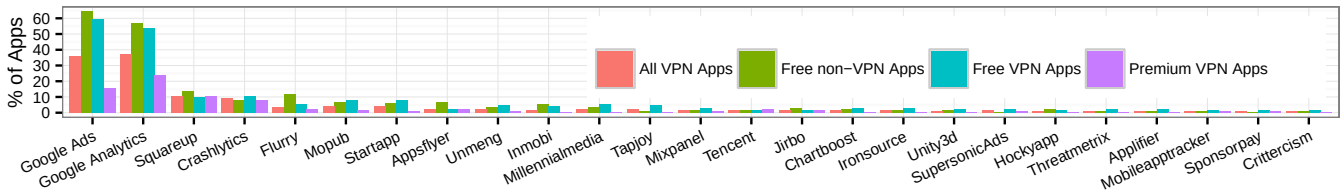

Figure 3: Top 25 third-party tracking libraries (x-axis) in VPN and non-VPN apps.

tives [105, 57], we rely on the "AV-rank" metric (*i.e.,* the number of affiliated AV tools that identified any malware activity) to reason about the maliciousness of an app. The study by Arp *et al.* [**?**] considered as a valid metric for malware presence on mobile apps an "AV-Rank" ≥ 2. Instead, we increase the 'AV-rank" to a value ≥ 5 to set a more conservative threshold for malware detection.

38% of the analyzed VPN apps have at least one positive malware report according to VirusTotal but only 4% of them have an "AV-rank" higher than 5. Table 5 ranks the top-10 VPN apps by their AV-rank. For each app, we include their Google Play rating and the number of install for reference. The malware signatures for those apps correspond to 5 different type of malware: Adware (43%), Trojan (29%), Malvertising (17%), Riskware (6%) and Spyware (5%).

OkVpn and EasyVpn, both implemented by the same app developer, incorporate Adware on their source code and both of them request the intrusive `SYSTEM_ALERT_WINDOW` permission which allows the requesting app to draw window alerts (in various forms as in the case unwanted ads) on top of any other active app. sFly Network Booster, traffic optimization VPN app, provides accelerated, worldwide content access through its dynamic routing and cloud-based accelerating system. It incorporates Spyware and requests the privacy sensitive `READ_SMS` and `SEND_SMS` permissions to read users' text messages and, potentially, send text messages to premium-rate numbers. OkVpn, EasyVPN, and sFly Network Booster are three of the 49 VPN apps that were not listed on Google Play as of August 2016 (Section 3.2).

According to the number of installs of these apps, millions of users appear to trust VPN apps despite their potential maliciousness. In fact, the high presence of malware activity in VPN apps that our analysis has revealed is worrisome given the ability that these apps already have to inspect and analyze all user's traffic with the VPN permission.

4.4 User Awareness Analysis

The previous subsection identified instances of VPN apps with malware presence. This section takes a user-centric perspective to understand if they publicly report on their Google Play reviews any of the privacy and security issues which could be present on VPN apps.

Our analysis reveals that VPN apps receive high user ratings: 37% of the VPN apps have more than 500K installs and 25% of them have at least a 4-star rating as shown in Figure 4. We cannot distinguish whether Google Play's positive installs and reviews are organic or if they were acquired using paid services to promote app installs [7].

[7] *e.g.,* http://liftoff.io

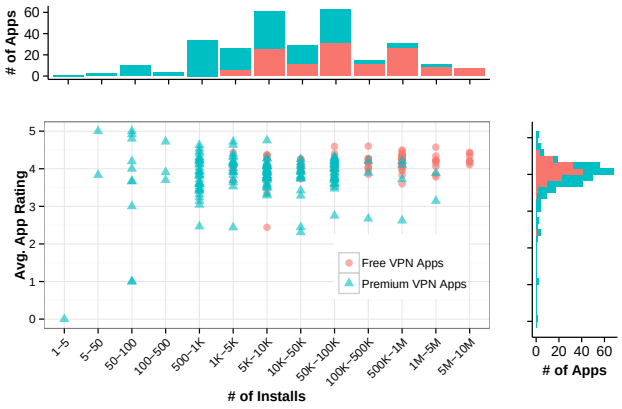

Figure 4: Distribution of app rating vs. installs per VPN app.

Complaint Category	**% of negative reviews** ($N = 4,593$)
Bugs & battery life	30%
Abusive permissions	0.5%
Privacy concerns	0.3%
Security concerns	0.4%
Malware/fraud reports	0.2%

Table 6: Classification of negative user reviews for the VPN apps with more than 1M installs in in Google Play.

To better understand whether real VPN users publicly report any security or privacy concerns after installing and using a given VPN app, we analyze (with manual supervision) 4,593 app reviews with low ratings (*i.e.,* one and two stars) for the 49 VPN apps with more than 1 million installs. Our reasoning to focus our analysis solely on negative app reviews is that users reporting concerning security-related issues will also provide a low app rating.

We classify app reviews into 5 categories (listed in Table 6) that cover from performance concerns and bugs to privacy and security concerns. We exclude from our analysis any reviews related with usability concerns. 30% of user complaints report bugs, crashes and the app's negative impact on battery-life. Only less than 1% of the negative reviews relate to security and privacy concerns, including the use of abusive or dubious permission requests and fraudulent activity, for the 9 apps listed in Table 7. Five of the apps reported as potentially malicious by app users are also flagged as such by VirusTotal (summarized in Table 5) due to malware activity (*e.g.,* EasyVPN) and trojans (*e.g.,* CrossVpn).

Summary and takeaways.

The increasing number of popular VPN apps available on Google Play and the apparent lack of user-awareness of the

App	Class	Rating	# Reviews	# Installs	AV-positive
EasyOvpn [14]	Free	4.2	84,400	5M	✓
VPN Free [58]	Prem.	4.0	15,788	1M	✓
Tigervpns [55]	Free	4.1	36,617	1M	✓
DNSet [11]	Prem.	4.0	21,699	500K	
CM Data Manager [6]	Prem.	4.3	11,005	1M	
Rocket VPN [43]	Free	4.2	11,625	500K	✓
Globus VPN [62]	Free	4.3	14,273	500K	
Spotflux VPN [50]	Free	4.0	14,095	500K	
CyberGhost [8]	Free	4.0	13,689	500K	✓

Table 7: List of VPN apps, with 500K or more number of installs, considered as malicious or intrusive by users in Google Play reviews and by VirusTotal (AV-positive column with AV-Rank ≥ 1).

security and privacy risks associated with the VPN permission indicate the urge to analyze in depth this unexplored type of mobile app. The average moble user rates VPN apps positively even when they have malware presence. According to our study, only a handful of users has raised any type of security and privacy concern in their reviews. In Section 5 we will complement the insights provided by our static analysis with a comprehensive set of active tests that aim to reveal behavioural aspects of the VPN apps during runtime at the network level.

5. NETWORK MEASUREMENTS

In this section, we investigate the runtime and network behavior of 150 VPN apps. In particular we are interested in understanding how VPN apps handle user's traffic.

We structure our analysis to illuminate the following aspects: (i) the traffic interception mechanisms implemented by each app (*i.e.,* whether the app uses the VPN permission to implement localhost proxies or to forward the traffic through a terminating end-point or another peer); (ii) the tunneling protocols implemented by each app as well as developer-induced misconfigurations which may cause traffic leaks; (iii) the presence of proxies and traffic manipulation techniques such as ad-blocking, JavaScript injection and traffic-redirection; and (iv) identify any possible occurrence of TLS interception.

We use a dedicated testbed, depicted in Figure 5, composed of a smartphone that connects to the Internet via a computer configured as a WiFi access point (AP) with dual-stack support. The WiFi AP runs `tcpdump` to intercept all the traffic being transmitted between the mobile device and the Internet. This allows us to observe the traffic generated by each VPN app as seen by an in-path observer.

We test individually each one of the 150 VPN apps under consideration. We could not fully automate our measurement efforts as one of the goals of our study is to understand and test the options offered by each VPN app in their GUI (*e.g.,* egress point diversity and supported VPN protocols). Prior to each test, we also ensure that the previous app we experimented with has not modified the root certificate store and we reboot the device to enforce the complete renewal of the virtual interface.

We run a set of purpose-built scripts (not only between the device and a server under our control but also to popular websites) and the ICSI Netalyzr tool for Android [86] to generate traffic and to analyze the different network- and traffic-related aspects of VPN apps. All the tests were conducted over a proxy-free link at Data61/CSIRO (Australia) thus the observed traffic manipulations and middleboxes can only be attributed to the VPN apps and their online infrastructure. Each subsection will describe in detail the tests used for each aforementioned analysis. The number of tests that we run per app varies with the configurability of the app (*e.g.,* whether the user can select a server in a given country) and the diversity of IP addresses that we observe. Two people executed a total of 5,340 tests manually for three months and connected to all end-points mentioned in the GUI of a given VPN app.

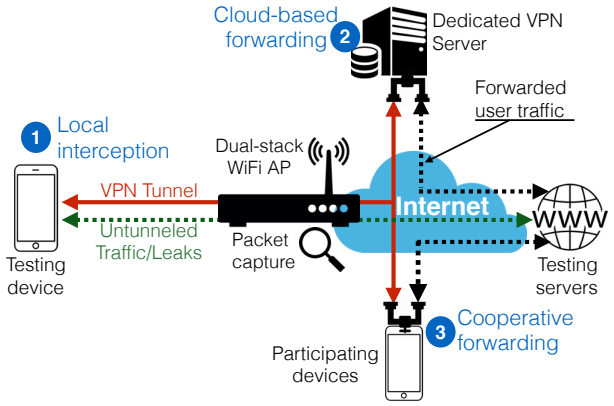

Figure 5: Our testbed and the 3 possible interception and forwarding modes for VPN apps: (1) **local interception** as a transparent proxy, (2) **cloud-based forwarding** through a VPN server, and (3) traffic forwarding through a participating node (**peer forwarding**) or other participating nodes. Our instrumented WiFi access point (AP) has the ability to observe all the traffic generated by each VPN app.

5.1 Interception and Forwarding Mechanisms

App developers can leverage the VPN permission to implement localhost proxies (Case 1 in Figure 5) or to forward user's traffic to an external machine. In the latter case, the egress point could be either a remote server hosted in the cloud (Case 2) or another participating node in a peer-forwarding fashion (Case 3). In this analysis we investigate the forwarding mechanism implemented by each VPN app according to the possible scenarios.

Our detection method relies on the client opening TCP connections to a remote dual-stack server under our control after enabling the traffic interception mode for each app. Our server records the public IP address for each TCP connection (*i.e.,* the egress point) and obtains its associated complete domain name (FQDN). We leverage *MaxMind's GeoIP* services [89] to identify the geographical location of the egress point. Our geo-location analysis is therefore limited to MaxMind's accuracy [70, 92]. We also use *Spamhaus Policy Block List* (PBL) [49] records to identify which IP addresses are associated with residential ISPs [65]. Note that Spamhaus' PBL records are populated directly by ISPs

to improve spam detection so they can be considered as an accurate proxy to identify IP addresses associated with residential end-users.

After introducing the different datasets that we will leverage to illuminate the forwarding mechanisms for each VPN app, we now define each forwarding mechanism as depicted in Figure 5. An app performs *local interception* if it operates as a localhost proxy without forwarding user's traffic to a terminating VPN server (*i.e.,* if the observed public IP address for all the TCP connections generated by our script matches the public IP address of our experimental setup). Otherwise, the VPN app implements *external forwarding*. For the latter case, we define two sub-categories: *cloud forwarding* if the VPN app uses a cloud provider to host their "terminating" VPN servers; and *peer forwarding* if the app leverages other participating users as egress points.

Local-Interception. Only 4% of the analyzed VPN apps use the VPN permission to intercept user's traffic in localhost or to implement transparent localhost proxies [93]. These VPN apps include antivirus software (*e.g., Dr.Web Security Space*), `tcpdump`-like tools that operate on user-space (*e.g., tPacketCapture*) and privacy and connection firewalls that allow users to generate connection logs or to block traffic at the flow- or app-level (*e.g., NoRoot Firewall*). Given that the traffic is intercepted locally and not forwarded through a VPN tunnel, our WiFi AP can identify side-connections generated by such apps. Notably, we observed that Dr. Web Security Space (an AV app) opens side-channel HTTPS flows to `drweb.com` and to `11t.su`[8]. To determine whether they are used to forward a copy of user's traffic, we correlate the exogenous flow sizes to the flow size (9KB) to our local web server. We observe that for 11KB and 4KB of traffic to `11t.su` and `drweb.com`, respectively. Unfortunately, given that these flows are encrypted, we could not investigate their payload to identify whether they are legitimate or not. For the remaining apps implementing local interception, we only observe traffic associated with their embedded third-party libraries for analytics and advertisement services.

External forwarding. Figure 6 shows the cumulative distribution of the number of countries hosting egress points for the remaining 96% of VPN apps. We observe a significant difference in the geographical coverage between free VPN apps and premium VPN apps. The distribution suggests that VPN servers for premium VPN apps are more scattered around the globe than for their entirely free counterparts: 80% of the free apps have their servers in less than 6 different countries, while 63% of the premium VPN apps have egress points in more than 6 countries. In fact, at least 20% of premium VPN apps have their servers located in more than 50 different countries.

The US hosts egress points for 77% of free and 90% of premium VPN apps respectively. France and the Netherlands are in second and third position for free VPN-apps (31% and 27% respectively) whereas the U.K and Germany

[8]We have noticed that Dr. Web appends the public IP address of our institution as a prefix to the domain `11t.su`.

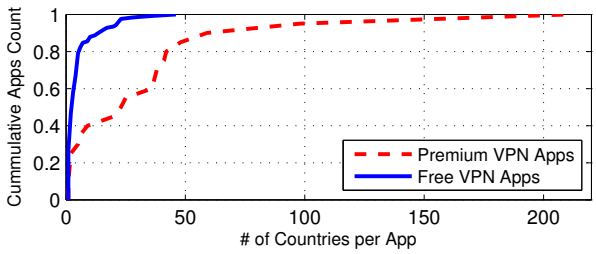

Figure 6: Distribution of the number of countries per VPN app.

Hosting Provider	Free Apps (N = 130)	# Recorded IPs
Digital Ocean	13%	74
Time Warner Cable Internet	6%	8
Amazon AWS	6%	10
JSC ER-Telecom Holding	6%	8
Saudi Telecom Comp. JSC	2%	3
Hosting Provider	**Prem. Apps (N = 20)**	**# Recorded IPs**
Leaseweb	20%	10
Reliablehosting	10%	505
Astute Hosting	10%	5
Digital Ocean	10%	2
IP-Only Networks AB	5%	3

Table 8: Top 5 VPN hosting infrastructures (by ASN) used by free VPN apps and premium VPN apps.

are second and third for premium apps (85% and 80% respectively)[9]. Notably, the top 3 countries contribute to 41% and 52% of the total number of VPN end-points for free and for premium VPN apps respectively.

A significant fraction of VPN apps concentrate all of their egress points in a single country: 16% of free VPN apps locate all of their end-points concentrated in the U.S., whereas 10% of the premium VPN apps have all their egress points concentrated in The Netherlands.

The other extreme is the VPN app *HideMyAss* [22] which provides terminating VPN servers virtually in almost every country in the world (209 countries/governments according to MaxMind's geolocation). If we look at the rank of hosting providers across VPN apps, we observe that Digital Ocean[10] (an American company) and Leaseweb[11] (a Dutch company) are the most common providers for free and premium VPN apps respectively. Table 8 shows the top 5 hosting providers by the number of VPN apps actively using their services.

Peer forwarding enables VPN apps to increase the number of egress points per country while reducing the costs of maintaining an online hosting infrastructure.[12] We attempt to identify apps implementing peer forwarding from the set of VPN apps with public IP addresses labeled as residential IPs by Spamhaus PBL. However, conducting this classification proves challenging (and prone to errors) as VPN services can deploy VPN servers in residential ISPs. This, un-

[9] Given that VPN apps can have end-points in multiple countries therefore the percentages do not add up to 100%.

[10]https://www.digitalocean.com

[11]https://www.leaseweb.com

[12]For reference, the cost per month of the hosting providers can range from 5 USD/month (Digital Ocean) to almost 200 USD/-month (Astute Hosting).

App	Class	# ASs	Residential AS(%)	Exogenous Traffic
Open Gate [36]	Free	54	70%	
VPN Gate [59]	Free	40	60%	
VyprVPN [63]	Free	2	50%	
OneClickVPN [35]	Free	57	53%	
Tigervpns [55]	Free	6	16%	✓
StrongVPN [51]	Prem.	59	14%	✓
Hola [23]	Free	41	5%	
HideMyAss [22]	Prem.	134	7%	✓
Private WiFi [40]	Free	30	7%	
VPNSecure [61]	Prem.	44	2%	

Table 9: VPN apps with egress points in residential ISPs. The last column indicates whether we have observed any possible exogenous flows for such apps.

fortunately, limits our ability to make a clear distinction between VPN apps implementing cloud- and peer-forwarding, or even hybrid approaches.

6% of the free VPN apps and 15% of the premium VPN apps relay traffic through residential ISPs. However, due to the aforementioned challenges, instead of attempting to classify each VPN app in these categories, we report in Table 9 the percentage of ASes for which we identified a residential egress point and the total number of ASes for each VPN app for reference. Out of these apps, only Hola confirms its community-powered nature (P2P) on its website.

We inspect the packets captured by our WiFi AP to identify the presence of exogenous flows which may have been forwarded through our device in a peer-to-peer fashion by the VPN engine for other participating users. While running HideMyAss we observed traffic going to JP Morgan and LinkedIn. None of these domains seem to be associated with any of the third-party libraries used by HideMyAss app. Unfortunately, we cannot entirely confirm the origin of these flows to assess whether they are endogenous to the app or not and our VPN session may have not lasted long enough[13] to capture traffic from other participating users. In the case of Tigervpns, we also identified flows to domains that no longer exist (*e.g.,* for `maxhane.com` and `qudosteam.com`, DNS lookup returned `NXDOMAIN` and `SERFAIL`, respectively.).

Nevertheless, the mere possibility of VPN apps following a peer forwarding model raises up some intriguing questions about their operational transparency and the security guarantees when forwarding traffic through (or on behalf of) other participating devices, not necessarily trustworthy.

5.2 VPN Protocols and Traffic Leaks

Ideally, the traffic forwarded through the VPN tunnel must be opaque to an in-path observer (*e.g.,* Internet service provider, commercial WiFi APs and surveillance agencies). However, there is a wide range of tunneling protocols, each with different security guarantees, that can be used by app developers to forward traffic out of the device: from secure IPSec tunnels to basic TCP tunnels without any encryption.

In addition to insecure tunneling protocols, developer-induced misconfigurations and errors may also undermine

[13]For each end-point, our session duration lasted for 180 seconds.

Protocol		Free Apps	Premium Apps
OpenVPN		14%	20%
L2TP/IPSec		5%	0%
SOCKS		4%	0%
Unidentified	UDP:80	0%	10%
	TLS (TCP:443)	15%	10%
	DTLS (UDP:443)	13%	25%
	Other ports	30%	25%
Unencrypted		19%	10%

Table 10: VPN tunneling protocols observed by our WiFi AP for the analyzed VPN apps.

user's privacy and security. VPN app developers must explicitly forward IPv6 traffic and provide the DNS settings at the time of creating the virtual interface programmatically. If not done carefully, DNS and IPv6 traffic may not be forwarded through the virtual interface [91]. In particular, DNS leakage can reveal user's networking activity and interests. The VPN API also allows app developers to overwrite user's DNS resolver with one of their choice.

All these artifacts can become a serious harm for users trying to circumvent surveillance or seeking online anonymity by using VPN apps. To investigate those crucial aspects of VPN apps, we run a script that performs crafted HTTP requests (both over IPv4 and IPv6) as well as DNS lookups to our dual-stack server under our control. In this section, we analyze the `pcaps` captured by our in-path WiFi AP to investigate the presence of tunnels without encryption in the wild (*i.e.,* we consider a tunnel implementation as unencrypted if the payload of our custom HTTP requests is seen in the clear by our WiFi AP) and to identify potential IPv6 and DNS leaks. We leverage the complementary features provided by a pcap parser [39] and Bro's comprehensive protocol analyzers (which provide support to identify some tunneling technologies) [90] to inspect in detail the traffic collected for each app.

VPN Tunnel Implementations. Table 10 shows the VPN tunneling protocols that we identified in the pcap traces gathered by our dual-stack WiFi AP. As mentioned earlier, we rely on Bro's suite of protocol parsers to identify the actual protocol used by each VPN apps. Unfortunately, Bro only provides full support for OpenVPN, L2TP/IPSec and SOCKS tunnels. For the remaining cases, we could not identify their application-layer protocol. Instead, we report the transport-layer protocol and the destination port in use. Identifying the actual protocol would have required us to decrypt the channel to inspect the payload.

Table 10 reports the different tunneling protocols that our method allowed us to identify. We observe that OpenVPN is the most popular tunneling technology both for free and premium apps (14% and 20% respectively). However, many VPN apps also use some tunneling technology over TLS and DTLS [**?**]. Of particular concern are the 19% and 10% of free and premium apps using basic TCP tunnels (also known as "port forwarders") and insecure HTTP tunnels [73]. As our WiFi AP, any in-path middlebox could inspect the payload for those apps in the clear. Therefore, the VPN apps using tunneling protocols without encryption are not protect-

ing their user-base from online surveillance and WiFi APs harvesting user's data.

IPv6 and DNS leaks. We observe that 84% of the analyzed VPN apps do not route IPv6 traffic through the VPN tunnel. Moreover, 66% of the VPN apps do not forward DNS traffic through the VPN tunnel so any in-path observer can monitor the DNS networking activity of the user. IPv6 and DNS leaks can ease user monitoring and censorship. Consequently, VPN apps like HideMyAss and VPNSecure which claim to provide security and anonymity are not effective against surveillance and malicious agents. Traffic leaks can be the result of intentional design decisions, lack of IPv6 support or even the result of developer-induced errors when configuring the routing parameters of the VPN app. Unfortunately, we cannot identify the root cause for such artifacts.

DNS redirection. For each one of the DNS lookups that we perform, we also check whether the IP address of the DNS resolver matches the one of our configured resolver's IP. Notably, 55% of the free apps (and 60% of premium apps) redirect user's DNS queries to Google DNS whereas 7% of free and 10% of premium VPN apps forward DNS traffic to their own DNS resolvers. In the latter case, users may be vulnerable to content filters and other DNS artifacts implemented by the DNS resolver such as traffic-redirection [103]. We have not further investigated the presence of traffic blockage or redirection mechanisms at the DNS level.

5.3 Traffic Manipulation

In-path proxies allow VPN services to gain control over users traffic and to manipulate traffic on the fly [102, **?**, **?**]. Moreover, many proxy features can provide an economic benefit for ISPs and network providers as in the case of HTTP header injection [**?**] or traffic redirection for advertising purposes [103].

We leverage the comprehensive network troubleshooting tool Netalyzr for Android to identify in-path flow-terminating proxies at the TCP level and, in the case of HTTP proxies, how they interfere with user's traffic. In a nutshell, Netalyzr controls both client and server side and crafts packets and HTTP requests in a way that would allow identifying non-transparent proxies along the path [**?**]. We refer the reader to Netalyzr-related bibliography for further implementation details [86, **?**, 102].

We extend the insights provided by the Netalyzr tool with custom-built tests that will allow us to identify VPN apps implementing techniques such as ad-blocking, JavaScript-injection for advertising and analytics purposes [**?**, 79], and traffic-redirection (*i.e.,* redirecting users traffic to third party advertising partners). In particular, we use two techniques to identify such proxy manipulations: First, we investigate domain mismatches between the DNS request and the service ultimately delivering the content using reverse DNS [103]. Second, we investigate content modifications for a website completely under our control, seven e-commerce websites (`alibaba.com`, `ebay.com`, `target.com`, `bestbuy.com`,`overstock.com`, `newegg.com`, and `macys.com`) and for the top-30 websites in the US, China, and Europe according to Alexa's rank [**?**]. As we demonstrate in one of our previous research efforts, the JavaScript code for two or more simultaneously accessed DOM trees' elements (*e.g.,* ads) belonging to the same website remain identical despite noticeable differences in the DOM tree elements [81]. This feature present in today's websites allows us to identify possible JavaScript injection by comparing the DOM trees for all selected websites before and after testing each VPN app. We use Selendroid [46] to fetch the rendered HTML source and extract the JavaScript as well as the DOM trees for each site.

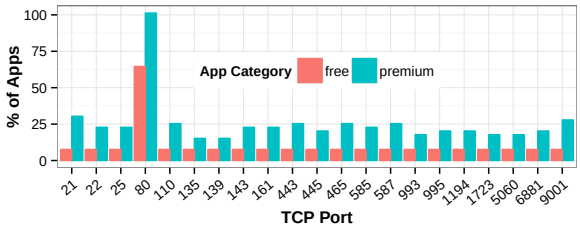

Figure 7: Distribution of in-path TCP proxy deployment per port.

In-path Proxies. The Netalyzr tests failed systematically for 34% of the analyzed VPN apps. Unfortunately, we do not have enough information to explain if such failures are caused by VPN app behavior, app bugs or if they are the result of traffic policies implemented by the VPN provider as Netalyzr generates traffic resembling BitTorrent which may be blocked by the VPN provider. We acknowledge it as a limitation of our tests to determine proxies in 34% of the analyzed VPN apps.

For the remaining 66% of VPN apps, Netalyzr results revealed the presence of flow-terminating proxies for multiple TCP ports as shown in Figure 7. According to the figure, for every port we study, in-path proxies are more common on premium VPN apps than in their free counterpart. We inspect app descriptions on Google Play store and observe that only 18% of the analyzed 66% apps provide such proxying as part of their stated purpose. The rest of the apps implement proxying as additional functionality. Nevertheless, we detected the presence of general-purpose proxies (*i.e.,* proxies listening in all the ports tested) in 8% and 15% of free and premium VPN apps respectively. Given that free VPN apps may implement peer forwarding to redirect user's traffic and the lower number of free VPN apps with premium services that we actively analyzed, in-path middleboxes and proxies may be less common in such scenarios.

In-path proxies may have additional negative effects on user's traffic which are beyond the scope of this study. Many of them may have their own particular or incomplete interpretation of transport-layer protocols [102]. In the case of HTTP proxies, Netalyzr test revealed that 47% and 55% of free and premium VPN apps actively modify HTTP traffic by default. Some proxy artifacts may have a negative impact on data fidelity and user's browsing experience as in the case of techniques like non-HTTP traffic filters over port TCP:80 (15% of VPN apps), HTTP body or header manipulations

Website	Input Point	Partner Network (click event)	Referral
alibaba.com	anchorfree.us/rdr.php	http://www.dpbolvw.net/click-7772790-12173149-1427959067000	NA
ebay.com	anchorfree.us/rdr.php	http://api.viglink.com/api/click?key=4372c7dabb08e4e38d97c4793cf6edb3	anchorfree.us/contentdiscovery2

Table 11: HotspotShield redirects user traffic to alibaba.com and ebay.com through its partner networks Conversant Media and Viglink respectively. In the case of target.com, bestbuy.com, overstock.com, newegg.com and macys.com we observed re-directions to Conversant Media.

(14% of VPN apps), and image transcoding (4% of VPN apps).

Ad-Blocking and Tracker-Blocking. Two of the analyzed VPN apps actively block ads and analytics traffic by default on our tested websites: Secure Wireless and F-Secure Freedome VPN. The apps did not explicitly mention ad-blocking feature in the Google Play store listings[14]. An analysis of the decompiled source code, using ApkTool, revealed that F-Secure Freedome VPN app blocks any traffic coming from a pre-defined list of domains associated with web and mobile tracking [16] including Google Ads, DoubleClick, and other popular tagging/analytics services such as Google Tag and comScore. However, blacklist-based ad blocking may affect the functionality of the Webpages and impairs user experience [81, **?**]. Specifically, F-Secure Freedome VPN blocks JavaScript code associated with nytimes.com's event "TaggingServices" which, as a result, prevents user access and interaction with embedded relevant video content [81].

JavaScript Injection. We identified two free VPN apps (VPN Services HotspotShield [25] by AnchorFree and WiFi Protector VPN [64]) actively injecting JavaScript codes using iframes for advertising and tracking purposes. Both apps claim to safeguard user privacy and to provide security and anonymization (cf Section 3.3). However, in the case of AnchorFree, they also provide advertising services [24]. Our static analysis of both apps' source code revealed that the actively use more than 5 different third-party tracking libraries. The developer team behind WiFi Protector VPN corroborated our observations and stated that the free version of its app injects JavaScript code for tracking and displaying their own ads to the users.

Traffic Redirection. AnchorFree's VPN app HotspotShield performs redirection of e-commerce traffic to partnering domains. When a client connects through the VPN to access specific web domains[15], the app leverages a proxy that intercepts and redirects the HTTP requests to partner websites with the following syntax: http://anchorfree.us/rdr.php?q=http://www.dpbolvw.net/click-7772790-12173149-1427959067000. As a result, user's traffic is relayed through two organizations before reaching alibaba.com: AnchorFree and dpbolvw.net, a domain owned by valueclick.com (or Conversant Media, an online advertising company[16]). Table 11 contains two samples of such requests. Our tests also identified a second partner: Viglink[17]. According to AnchorFree's website, the app provides "shielded connections, security, privacy enhancement for individuals and small businesses" and an "ad-free browsing" environment [24].

5.4 TLS Interception

VPN apps are in a privileged position to perform TLS interception [101]. They can compromise the local root certificate store of the device by injecting their own self-signed certificates using Android's KeyChain API [**?**]. Once a certificate is installed on the device, the app can intercept the TLS session establishment and generate "legit" certificates — verifiable by the self-signed root certificate injected on the trusted certificate root store — on the fly [101]. To limit potential new venues for abuse, Android requires user's consent to install root certificates and it shows an additional system notification that informs the user that a third-party can monitor their secure traffic. Only tech-savvy users may be able to fully understand the security implications of installing a root certificate.

We instrumented our Android device with OpenSSL so that we can capture a copy of the SSL/TLS server certificate when accessing more than 60 popular services operating over SSL including HTTPS, SMTP over TLS, and POP3 over TLS. The services reached in our test include diverse and popular services like Google, Gmail, Facebook, Twitter, Skype, banking services, CDNs, analytics services and e-commerce sites, many of which are associated with mobile apps implementing security countermeasures such as certificate pinning [76, 45].

We validated each server certificate against the ICSI Certificate Notary to identify possible cases of TLS interception: 3% of the TLS sessions provided certificates for which the ICSI notary could not establish a valid chain to a root certificate from the Mozilla root store. By inspecting manually each certificate, we identify 4 free VPN apps (developed by 3 different app developers) that actively intercept TLS traffic by issuing self-signed certificates as shown in Table 12. Two of the apps implementing TLS interception, DashVPN and DashNet, are implemented by the company ActMobile

A detailed inspection of the domains for which we recorded self-signed certificates, revealed that only the app Packet Capture performs TLS interception indiscriminately for all domains even if the apps perform cert pinning. The other apps, — Neopard, DashVPN and DashNet all of which

[14]Contrary to Google Play listings, F-Secure Freedome VPN mentioned its ad-blocking feature on its webiste, https://www.f-secure.com/.

[15]During our experiments redirection happened exclusively for websites categorized as e-commerce sites such as alibaba.com.

[16]http://www.conversantmedia.com

[17]http://www.viglink.com

VPN app	CA	User-warning	# Installs
Packet Capture [38]	Packet Capture	GUI	100K
DashVPN [10]	ActMobile		100K
DashNet [9]	ActMobile		10K
Exalinks Neopard [30]	Exalinks Root	Privacy Policy	10K

Table 12: VPN apps performing TLS interception, the CA signing the forged certificates and if the apps explicitly inform the user about TLS interception practices in their GUI or in their privacy policy.

claim to provide traffic acceleration — target specific services as reported in Table 13, more inclined towards email services, social networks search engines and IM. This behavior may be a consequence of the nature of the apps and the intent of the online services that they aim to optimize.

Domain (PORT)	Neopard	DashVPN	DashNet	Packet Capture
google-analytics.com	✓	✓	✓	✓
mail.google.com	✓	✓	✓	✓
mail.yahoo.com	✓	✓	✓	✓
maps.google.com	✓	✓	✓	✓
orcart.facebook.com (8883)	✓	✘	✘	✓
play.google.com	✓	✓	✓	✓
www.akamai.com	✘	✓	✘	✓
www.alcatel-lucent.com	✘	✓	✘	✓
www.amazon.com	✘	✓	✘	✓
www.avaya.com	✘	✓	✘	✓
www.bankofamerica.com	✘	✓	✓	✓
www.chase.com	✘	✓	✓	✓
www.cisco.com	✘	✓	✘	✓
www.ebay.com	✘	✓	✘	✓
www.facebook.com	✓	✓	✓	✓
www.fring.com	✘	✓	✓	✓
www.gmail.com	✓	✓	✓	✓
www.google.co.uk	✓	✓	✓	✓
www.google.com	✓	✓	✓	✓
www.hotwire.com	✘	✓	✘	✓
www.hsbc.com	✘	✓	✘	✓
www.ibm.com	✘	✓	✘	✓
www.icsi.berkeley.edu	✘	✓	✓	✓
www.linkedin.com	✓	✘	✘	✓
www.outlook.com	✘	✓	✓	✓
www.qq.com	✓	✘	✘	✓
www.seagate.com	✘	✓	✘	✓
www.simple.com	✘	✓	✘	✓
www.skype.com	✘	✓	✓	✓
www.taobao.com	✘	✓	✘	✓
www.tripadvisor.com	✘	✓	✘	✓
www.twitter.com	✓	✓	✓	✓
www.viber.com	✘	✓	✓	✓
www.yahoo.com	✓	✓	✓	✓
www.youtube.com	✓	✓	✘	✓

Table 13: Intercepted domains per VPN app. The list only prints the TCP port for those different than 443.

We manually inspected the app's GUI to check if the apps inform users about the purpose of performing TLS interception and what TLS interception implies. Packet Capture supports TLS interception (as an opt-in feature in the app) in order to expose TLS traffic to its users. Likewise, Neopard, a web acceleration app, also notifies users about the purpose of performing TLS interception in order to optimize traffic. Their privacy policy (April 2016) [31] informs users about TLS interception and lists "perform mobile usage reviews for market studies" as one of the purposed of their data collection process. In the case of DashVPN and DashNet, none of them inform users about the purpose of performing TLS interception at all.

Summary and takeaways.

Our analysis of VPN apps at the network level has revealed that the majority of VPN apps are not transparent enough about how they handle user's traffic. Despite the promises for security enhancement and online anonymity, VPN apps may forward user's traffic through other participating nodes following a peer (*e.g.,* Hola) thus opening interesting questions about the trustworthiness of the egress points and the security guarantees for users forwarding traffic for others.

Our analysis has also revealed an alarming 18% of VPN apps that implement tunneling technologies without encryption as well as 84% and 66% of apps leaking IPv6 and DNS traffic. As a result, these apps do not protect user's traffic against in-path agents performing online surveillance or user tracking. We inspect app descriptions on Google Play store and observe that 94% of the IPv6 and DNS leaking apps claim to provide privacy protection. Such traffic leaks may be associated with developer-induced errors, lack of support or even misconfigurations.

Finally, we have also identified abusive practices in our corpus of VPN apps such as JavaScript injection for tracking and advertising purposes, as well as e-commerce traffic redirection to affiliated partners and TLS interception. Only one of the apps implementing these practices (*i.e.,* Packet Capture performing TLS interception) actually inform the users about the presence of such artifacts.

5.5 Developers' responses

We contacted and shared our findings with the developers of each of the apps we observed as involved in any of the following: JavaScript injection, traffic redirection, ad-blocking and tracker-blocking, exogenous flow, peer-forwarding user traffic, and TLS interception. We also contacted app developers of apps requesting sensitive permissions, apps that are negatively reviewed by users, and the ones with embedded third-party tracking libraries. We also contacted apps which our tests revealed as possibly containing malware in their APKs.

Amongst the two apps (WiFi Protector and HotspotShield VPN) that our tests identified as performing JavaScript injection, WiFi Protector confirmed our findings and stated that the free version of their app injects JavaScript code to track users and to show their own ads. HotspotShield VPN, which we identified as also performing traffic redirection, has not responded to our correspondence.

The developer behind F-Secure Freedome VPN, we found that it blocks third-party ads and trackers, confirmed our

findings and elaborated on how they construct their blacklists for third-party trackers- and ads-blocking. The developer did not respond yet to our inquiries about the criteria used to build the blacklists. We have not received any response from Secure WiFi that our tests also identified as performing ad-blocking.

We received responses from only three developers of the apps that we observed implementing peer-forwarding of user traffic. VyperVPN and VPNSecure confirmed that they have some of their end-points located in residential ISPs as they may rely on third-party data-centers for hosting their services. Hola's developer confirmed our findings and explicitly mentioned Hola's peer-forwarding mechanism. Contacts from other apps detailed in Table 9 have not yet, as of the time of writing of this paper, responded to our requests for comments or feedback.

Neopard confirmed that they whitelist the domains for which they can optimize traffic and asked for feedback about how to increase their operational transparency and usability. ActMobile, initially, asked for further information about the purpose of this study and who has commissioned and later on, acknowledged our findings, confirmed that they disable the default TLS-interception functionality in both of the apps (DashVPN and Dashnet). They also reported that, in the new version of the apps, they ask for user consent, explicitly in the apps' GUIs, to install and to enable the ActMobile's certificates for TLS-interception and traffic acceleration, respectively. We have not received any response from the developer behind Packet Capture that our tests identified as performing TLS-interception.

Only one of the apps' developers, explicitly discussed in Section 4.1, responded to our findings and confirmed that tigerVPN requests sensitive `READ_LOG` permission to record and to use it for troubleshooting purposes. They also confirmed that, in the connection log collected via `READ_LOG` permission, they collect users' information such as end-points' IPs, wireless (mobile data connectivity (3G, 4G, and LTE) or WiFi) connectivity, and error messages.

The developer behind Ip-shield VPN that we identified as embedding less-popular tracking libraries such as Appflood for targeted ads argued that the Appflood was the best choice to monetize the app. The developer also revealed plans to update ad-free version of Ip-shield VPN on Google Play.

The rest of the developers of the apps with possibly containing malware (cf. Section 4.3), apps that are negatively reviewed by users (cf. Section 4.4), apps that are embedding third-party tracking libraries (cf. Section 4.2), and the one with exogenous traffic flows (cf. Section 5.1) have not yet, as of the time of writing of this paper, responded to our findings.

6. LIMITATIONS AND FUTURE WORK

Our method to identify and characterize VPN apps on Google Play presents several limitations, many of which are inherent to static and dynamic analysis [76]. The first limitation is app's coverage: our study is limited to Android's free Google Play apps and excludes paid apps, iOS apps and apps from alternative app stores. We also rely on a Google Play crawler to extract our corpus of VPN-enabled apps that might restrict the app coverage of our study which may miss apps that intentionally (or inadvertently) hide their use of the VPN permission. Although our apps crawler aims to capture as many VPN-enabled apps as possible, we stress that our goal is to provide an analysis of the security and privacy issues of a representative sample of VPN-enabled apps from the Google play store. Second, this paper does not consider Android apps requesting root access on rooted phones to intercept user traffic via native commands such as *tcpdump* or OpenVPN. Investigating apps falling in this category would require conducting a computational- and time-expensive static analysis. Third, we do consider runtime analysis of third-party tracking libraries and all sensitive permissions of VPN apps. Determining what an app do with sensitive permissions such as `READ_LOG`, `READ_SMS`, and `SEND_SMS` and what type of information will third-party tracking libraries collect would require fine-grained system- and network-level trace and traffic analysis.

Moreover, we identify apps implementing peer forwarding from the set of VPN apps with public IP addresses labeled as residential IPs by Spamhaus PBL. Given that VPN services can deploy VPN servers in residential ISPs and Spamhaus classification is prone to error, our analysis of peer forwarding may not be accurate. We consider it as a limitation and one possible extension of the work in this paper would be to strengthen our analysis of peer forwarding by *(i)* extending the tests duration to enable tracking of peers (running suspected VPN apps); and *(ii)* analyzing the traffic flows of an app simultaneously running on two or more mobile phones to determine if they forward traffic for each other.

Likewise, our method falls short to analyze the presence of session timeouts and apps's ability to recover from a loss of connectivity. These dynamics may cause user traffic to be exposed in the clear to any in-path middlebox for a short period of time.

This paper provides a first detailed analysis of VPN-enabled apps but it also leaves many open questions beyond the scope of our analysis. Aspects such as possible traffic or device-location discrimination practices [83] or the use of VPN apps as honeypots to harvest personal information have not been addressed in this study. In addition, reasons behind inadequacy of app actual behavior and terms of use or the the identification of side-channels for the observed data-exfiltration have been left as pending questions.

7. RELATED WORK

Several studies highlighted the privacy risks associated with Android apps over-requesting Android permissions for third-party tracking, advertising and analytic services [99, 97, 88, 71, 72] using techniques like static analysis [106, 67, 77, 78], taint analysis [75, 104], and OS modifications [82, 94, 99, 80]. Previous research also adapted techniques for malware detection such as signature analysis [69, 68, 85,

105] and anomaly detection [98, 96] to the mobile context in order to identify potential malicious activity on mobile apps.

Several research efforts leverage Android's VPN permission to accurately characterize Android's traffic and identify private data leakage inflicted by mobile apps [87, 93, 100]. More related to studying VPN apps, the study conducted by Perta *et al.* [91] is perhaps the closest one to our analysis. The paper provides a manual analysis of 14 popular VPN services and includes a study of their their mobile clients identifying developer-induced bugs and mis-configurations that lead to IPv6 and DNS leaks. Our paper provides a systematic and thorough security and privacy analysis of Android mobile apps employing the VPN permission. The study by Vallina-Rodriguez *et al.* characterized Android's root certificate store using data provided by Netalyzr for Android tool [101]. The study revealed how VPN-enabled apps could perform transparent TLS interception after compromising the root certificate store.

Finally, Appelbaum *et al.* identified security vulnerabilities on commercial and public online VPN services [66]. A survey conducted by Khattack *et al.* on VPN usage across Pakistani Internet users reported that 57% of the participants used SSL-based VPN software to access YouTube content [84]. Our paper in turn, presents a method to systematically identify and analyze security and privacy aspects of VPN-enabled apps on Android-based app stores. The implications of our analysis span to other areas such as censorship analysis and network measurements that leverage VPN services to penetrate different countries and ISPs.

8. CONCLUSIONS

Android app developers benefit from native support to implement VPN clients via the VPN permission to provide censorship circumvention, support enterprise customers and enhanced online security and privacy. However, despite the fact that Android VPN-enabled apps are being installed by millions of mobile users worldwide, their operational transparency and their possible impact on user's privacy and security remains "terra incognita" even for tech-savvy users.

In this paper, we presented a number of static and dynamic methods that allowed us to conduct in-depth analysis of VPN-enabled apps on Google Play. We investigate from the presence of tracking services and malware on VPN app binaries to artifacts implemented by these apps at the network level. Our comprehensive tests allowed us to identify instances of VPN apps embed third-party tracking services and implement abusive practices such as JavaScript-injection, ad-redirections and even TLS interception.

The ability of the `BIND_VPN_SERVICE` permission to break Android's sandboxing and the naive perception that most users have about third-party VPN apps suggest that it is urging to re-consider Android's VPN permission model to increase the control over VPN clients. Our analysis of the user reviews and the ratings for VPN apps suggested that the vast majority of users remain unaware of such practices even when considering relatively popular apps.

Acknowledgments

This work was partially supported by Data61/CSIRO and NSF grant CNS-1564329. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors or originators and do not necessarily reflect the views of the Data61/CSIRO or of the NSF. The authors would like to thank our shepherd, Ben Zhao, the anonymous reviewers for constructive feedback on preparation of the final version of this paper. We also thank Nick Kiourtis (Kryptowire) and Angelos Stavrou (Kryptowire) for valuable help.

9. REFERENCES

[1] AnchorFree Inserted Code. http://box.anchorfree.net/insert/insert.php?sn=HSSHIELD00AU&ch=HSSCNL000001&v=6231615266&b=android&ver=android&afver=362.

[2] Android Permissions. http://developer.android.com/guide/topics/security/permissions.html.

[3] Application Fundamentals. http://developer.android.com/guide/components/fundamentals.html.

[4] Archie VPN. https://play.google.com/store/apps/details?id=com.lausny.archievpnfree.go.

[5] Cisco AnyConnect. https://play.google.com/store/apps/details?id=com.cisco.anyconnect.vpn.android.avf.

[6] CM Data Manager - Speed Test. https://play.google.com/store/apps/details?id=com.cmcm.flowmonitor.

[7] CrossVpn. https://play.google.com/store/apps/details?id=com.goodyes.vpn.cn.

[8] Cyberghost - free vpn & proxy. https://play.google.com/store/apps/details?id=de.mobileconcepts.cyberghost.

[9] Dash Net Accelerated VPN . https://play.google.com/store/apps/details?id=com.actmobile.dashnet.

[10] Dash VPN | Dash Office - Speed Test. http://dashoffice.com/dash-vpn/.

[11] DNSet. https://play.google.com/store/apps/details?id=com.dnset.

[12] DroidVPN - Android VPN. https://play.google.com/store/apps/details?id=com.aed.droidvpn.

[13] Dr.Web Security Space. https://play.google.com/store/apps/details?id=com.drweb.pro.

[14] EasyOvpn - Plugin for OpenVPN. https://play.google.com/store/apps/details?id=com.easyovpn.easyovpn.

[15] EasyVpn. https://play.google.com/store/apps/details?id=yujia.easyvpn.

[16] F-Secure Freedome Anti-Tracking Feature Explained. https://community.f-secure.com/t5/F-Secure/F-Secure-Freedome-Anti-Tracking/ta-p/52153.

[17] Fast Secure Payment Service. https://play.google.com/store/apps/details?id=com.lausny.ocvpnaio.allpay.

[18] FlashVPN Free VPN Proxy. https://play.google.com/store/apps/details?id=net.flashsoft.flashvpn.activity.

[19] Free VPN Proxy by Betternet. https://play.google.com/store/apps/details?id=com.freevpnintouch.

[20] Good. Mobile Device Management (MDM). https://www1.good.com/secure-mobility-solution/mobile-device-management.html.
[21] HatVPN. https://play.google.com/store/apps/details?id=mobi.hatvpn.
[22] HideMyAss! Pro VPN for Android. https://play.google.com/store/apps/details?id=com.hidemyass.hidemyassprovpn.
[23] Hola Free VPN Proxy. https://play.google.com/store/apps/details?id=org.hola.
[24] Hotspot Shield Advertising. http://www.anchorfree.com/advertise.php.
[25] Hotspot Shield Free VPN Proxy. https://play.google.com/store/apps/details?id=hotspotshield.android.vpn.
[26] ip-shield VPN. https://play.google.com/store/apps/details?id=com.ipshield.app.
[27] Junos Pulse. https://play.google.com/store/apps/details?id=net.juniper.junos.pulse.android&hl=en.
[28] Knox Standard SDK. https://seap.samsung.com/sdk/knox-standard-android.
[29] Mobile Security & Antivirus. https://play.google.com/store/apps/details?id=com.trendmicro.tmmspersonal.
[30] NEOPARD. http://https://play.google.com/store/apps/details?id=com.exalinks.neopard/.
[31] Neopard Privacy Policy. http://neopard-mobile.com/en/about/privacy/.
[32] NeoRouter VPN Mesh. https://play.google.com/store/apps/details?id=com.neorouter.androidmesh.
[33] NoRoot Firewall. https://play.google.com/store/apps/details?id=app.greyshirts.firewall.
[34] OkVpn . https://play.google.com/store/apps/details?id=yujia.okvpn.
[35] One Click VPN. https://play.google.com/store/apps/details?id=com.lausny.ocvpn.
[36] Open Gate. https://play.google.com/store/apps/details?id=com.btzsoft.vpnclient.
[37] Orbot: Proxy with Tor. https://play.google.com/store/apps/details?id=org.torproject.android.
[38] Packet Capture. https://play.google.com/store/apps/details?id=app.greyshirts.sslcapture.
[39] pcap-parser (0.5.8). https://pypi.python.org/pypi/pcap-parser/0.5.8.
[40] Private WiFi . https://play.google.com/store/apps/details?id=com.privatewifi.pwf.hybrid.
[41] Qihoo 360. https://play.google.com/store/apps/details?id=com.qihoo360.mobilesafe.
[42] Raccon APK Downloader. http://www.onyxbits.de/raccoon.
[43] Rocket VPN âĂŞ Internet Freedom. https://play.google.com/store/apps/details?id=com.liquidum.rocketvpn.
[44] Samsung KNOX. Partnering with Samsung. https://www.samsungknox.com/en/partners.
[45] Security with HTTPS and SSL. http://developer.android.com/training/articles/security-ssl.html.
[46] Selendroid: Selenium for android. http://www.selendroid.io.
[47] sFly Network Booster, Adblocker. https://play.google.com/store/apps/details?id=com.cdnren.sfly.
[48] Shadowsocks. https://play.google.com/store/apps/details?id=com.github.shadowsocks.
[49] Spamhaus PBL. http://www.spamhaus.org/pbl/.
[50] Spotflux VPN. https://play.google.com/store/apps/details?id=com.spotflux.android.
[51] StrongVPN OpenVPN Client. https://play.google.com/store/apps/details?id=com.strongvpn.
[52] SuperVPN. https://play.google.com/store/apps/details?id=com.SuperVPN_Q0102_21.
[53] SurfEasy Secure Android VPN. https://play.google.com/store/apps/details?id=com.surfeasy.
[54] tigerVPN - Privacy Defender. https://play.google.com/store/apps/details?id=com.tigeratwork.tigervpn.
[55] Tigervpns Free VPN and Proxy. https://play.google.com/store/apps/details?id=com.tigervpns.android.
[56] TorGuard VPN. https://play.google.com/store/apps/details?id=net.torguard.openvpn.client.
[57] VirusTotal. https://www.virustotal.com.
[58] VPN Free. https://play.google.com/store/apps/details?id=com.couxin.GroxNetwork.
[59] VPN Gate. https://play.google.com/store/apps/details?id=com.lausny.vpngate.
[60] VPN Service Documentation. http://developer.android.com/reference/android/net/VpnService.html.
[61] VPNSecure OpenVPN VPN Proxy. https://play.google.com/store/apps/details?id=com.vpnsecure.pty.ltd.
[62] VPN+TOR+Cloud VPN Globus Pro! https://play.google.com/store/apps/details?id=com.globus.vpn.
[63] VyprVPN Free VPN for Privacy. https://play.google.com/store/apps/details?id=com.goldenfrog.vyprvpn.app.
[64] WiFi Protector VPN. https://play.google.com/store/apps/details?id=com.wifiprotector.android.
[65] M. Allman. Comments on bufferbloat. *SIGCOMM CCR*, 2013.
[66] J. Appelbaum, M. Ray, I. Finder, and K. Koscher. vpwns: Virtual Pwned Networks. In *USENIX FOCI*, 2012.
[67] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie. PScout: Analyzing the Android Permission Specification. In *ACM CCS*, 2012.
[68] T. Bläsing, L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak. An Android Application Sandbox System for Suspicious Software Detection. In *IEEE MALWARE*, 2010.
[69] A. Bose, X. Hu, K. G. Shin, and T. Park. Behavioral Detection of Malware on Mobile Handsets. In *ACM MobiSys*, 2008.
[70] I. Castro, J. C. Cardona, S. Gorinsky, and P. Francois. Remote peering: More peering without internet flattening. In *ACM CoNEXT*, 2014.
[71] T. Chen, I. Ullah, M. A. Kaafar, and R. Boreli. Information Leakage Through Mobile Analytics Services. In *ACM MobiSys*, 2014.
[72] P. H. Chia, Y. Yamamoto, and N. Asokan. Is this App Safe?:

A Large Scale Study on Application Permissions and Risk Signals. In *ACM WWW*, 2012.

[73] D. Crawford. PPTP vs L2TP vs OpenVPN vs SSTP vs IKEv2. https://www.bestvpn.com/blog/4147/pptp-vs-l2tp-vs-openvpn-vs-sstp-vs-ikev2/.

[74] J. Daemen and V. Rimen. Advanced Encryption Standard - Rijndael. http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf#page=1.

[75] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An Information Flow Tracking System for Real-Time Privacy Monitoring on Smartphones. *CACM*, 2014.

[76] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why Eve and Mallory love Android: An analysis of Android SSL (in) security. In *ACM CCS*, 2012.

[77] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android Permissions Demystified. In *ACM CCS*, 2011.

[78] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller. Checking App Behavior Against App Descriptions. In *ICSE*, 2014.

[79] C. Haschek. Where are free proxies free? https://blog.haschek.at/post/fd9bc.

[80] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These Aren't the Droids You're Looking for: Retrofitting Android to Protect Data from Imperious Applications. In *ACM CCS*, 2011.

[81] M. Ikram, H. J. Asghar, M. A. Kaafar, B. Krishnamurthy, and A. Mahanti. Towards seamless tracking-free web: Improved detection of trackers via one-class learning. *arXiv preprint arXiv:1603.06289*, 2016.

[82] J. Jeon, K. K. Micinski, J. A. Vaughan, A. Fogel, N. Reddy, J. S. Foster, and T. Millstein. Dr. Android and Mr. Hide: Fine-grained Permissions in Android Applications. In *ACM SPSM*, 2012.

[83] S. Khattak, D. Fifield, S. Afroz, M. Javed, S. Sundaresan, V. Paxson, S. J. Murdoch, and D. McCoy. Do you see what i see? differential treatment of anonymous users. In *NDSS*, 2016.

[84] S. Khattak, M. Javed, S. A. Khayam, Z. A. Uzmi, and V. Paxson. A look at the consequences of internet censorship through an isp lens. In *ACM IMC*, 2014.

[85] H. Kim, J. Smith, and K. G. Shin. Detecting Energy-Greedy Anomalies and Mobile Malware Variants. In *ACM MobiSys*, 2008.

[86] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the edge network. In *ACM IMC*, 2010.

[87] A. Le, J. Varmarken, S. Langhoff, A. Shuba, M. Gjoka, and A. Markopoulou. AntMonitor: A System for Monitoring from Mobile Devices. In *ACM (C2B(I)D)*, 2015.

[88] I. Leontiadis, C. Efstratiou, M. Picone, and C. Mascolo. Don't Kill my Ads!: Balancing Privacy in an Ad-supported Mobile Application Market. In *ACM HotMobile*, 2012.

[89] MaxMind. https://www.maxmind.com.

[90] V. Paxson. Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks*, 1999.

[91] V. C. Perta, M. V. Barbera, G. Tyson, H. Haddadi, and A. Mei. A Glance through the VPN Looking Glass: IPv6 Leakage and DNS Hijacking in Commercial VPN Clients. *PETS*, 2015.

[92] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye. Ip geolocation databases: Unreliable? *ACM SIGCOMM CCR*, 2011.

[93] A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson. Haystack: In Situ Mobile Traffic Analysis in User Space. *arXiv preprint arXiv:1510.01419*, 2015.

[94] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H. J. Wang, and C. Cowan. User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems. In *IEEE S&P*, 2012.

[95] Samsung KNOX. https://www.samsungknox.com/en.

[96] A.-D. Schmidt, F. Peters, F. Lamour, C. Scheel, S. A. Çamtepe, and Ş. Albayrak. Monitoring Smartphones for Anomaly Detection. *Mobile Networks and Applications*, 2009.

[97] S. Seneviratne, H. Kolamunna, and A. Seneviratne. A Measurement Study of Tracking in Paid Mobile Applications. In *ACM WiSec*, 2015.

[98] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss. "Andromaly": A Behavioral Malware Detection Framework for Android Devices. *JIIS*, 2012.

[99] S. Shekhar, M. Dietz, and D. S. Wallach. AdSplit: Separating Smartphone Advertising from Applications. In *USENIX Sec*, 2012.

[100] Y. Song and U. Hengartner. Privacyguard: A vpn-based platform to detect information leakage on android devices. In *ACM SPSM*, 2015.

[101] N. Vallina-Rodriguez, J. Amann, C. Kreibich, N. Weaver, and V. Paxson. A Tangled Mass: The Android Root Certificate Stores. In *ACM CoNEXT*, 2014.

[102] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, N. Weaver, and V. Paxson. Beyond the radio: Illuminating the higher layers of mobile networks. In *ACM MobiSys*, 2015.

[103] N. Weaver, C. Kreibich, and V. Paxson. Redirecting dns for ads and profit, 2011.

[104] L.-K. Yan and H. Yin. DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. In *USENIX Security*, 2012.

[105] Y. Zhou and X. Jiang. Dissecting Android Malware: Characterization and Evolution. In *IEEE S&P*, 2012.

[106] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh. Taming Information-stealing Smartphone Applications (on Android). In *TRUST*, 2011.