

Exposing Android social applications: linking data leakage to privacy policies

Daniel E. Krych & Patrick McDaniel

To cite this article: Daniel E. Krych & Patrick McDaniel (2022): Exposing Android social applications: linking data leakage to privacy policies, Journal of Cyber Security Technology, DOI: 10.1080/23742917.2019.1630093

To link to this article: <https://doi.org/10.1080/23742917.2019.1630093>



Published online: 02 Mar 2022.



Submit your article to this journal [↗](#)



View related articles [↗](#)



View Crossmark data [↗](#)



Exposing Android social applications: linking data leakage to privacy policies

Daniel E. Krych and Patrick McDaniel

Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, USA

ABSTRACT

Social media applications (apps) use sensitive/private data in their nature. Android apps continue to share data with third parties and often transmit data unencrypted, leaking data directly and inadvertently. Internet Service Providers (ISPs) can legally collect and sell this leaked, sensitive user data. End users rely on privacy policies that have largely been absent, continue to lack detailed security methods, and have had inconsistencies with app actions. Overall, we lack a detailed understanding of the state of these privacy and security issues within sensitive settings such as social media apps. We aim to expose these apps, meticulously classifying and comparing unencrypted data transmitted with privacy policy disclosure, leveraging the Platform for Privacy Preferences (P3P) Specification. We develop an analysis framework and isolated testbed environments, leveraging open-source tools to enable accurate data collection through dynamic analysis. We then peer into privacy policy revisions, advertising/analytics libraries, and business relationships held by app companies. We report detailed inconsistencies between app behaviors and privacy disclosure. Most apps in our dataset transmitted the majority of their traffic unencrypted and several leaked personally identifiable information (PII)/sensitive data, while none detailed security methods or data transmission practices. Finalizing our study, we conduct brief follow-up experiments on several apps to note substantial changes in data transmission practices. We conclude that a failure to protect sensitive user data and a tendency for vague privacy policies continue to be prevalent, but over the last few years some apps have increased encryption use, thus beginning to combat some data leakage.

ARTICLE HISTORY



Received 13 May 2019
Accepted 15 May 2019

KEYWORDS

Android security; encryption; man-in-the-middle; privacy policy; social media

1. Introduction

Social media applications (apps) collect, display, and share personal information as core to their nature in providing end users with a platform on which to interact. Users trust these apps with their information, including personally identifiable information (PII) such as health information, location data, SSN, or unique device

CONTACT Daniel E. Krych  daniel.e.krych.civ@army.mil  Network Security Branch, DEVCOM Army Research Laboratory, Aberdeen Proving Ground, MD, USA

© 2019 Informa UK Limited, trading as Taylor & Francis Group

IDs. These apps also include sensitive information, which may or may not be classified as PII, such as email addresses, phone numbers, interests, usage habits, age, and gender. Android apps have been found to transmit sensitive data unencrypted and share PII with third parties and advertising companies [1–8]. These instances of data mishandling may violate the privacy and security of end users, and trends seem to indicate a continuation of these practices. Any data transmitted unencrypted results in inadvertent exposure to the network and all of the points along the network path. Many Internet Service Providers (ISPs) are and have been legally and physically capable of collecting and selling sensitive user data to advertising companies and third parties [9]. Overall, we have a limited visibility into what these app companies and other parties on the networks are doing behind-the-scenes with user data.

Users can review an app's privacy policy, which should determine what data are being collected, how they are being transmitted/handled, and who has access, but previous studies have noted the absence of app privacy policies and a difficulty in finding those that do exist [6,7,10,11]. Furthermore, if they have one at all, the majority of privacy policies only provide general information about their practices. Recent years have shown an improvement in the presence of a policy, but a lack of detail regarding data-handling and security practices seems to be the norm [11]. Additionally, inconsistencies between companies' privacy policies and the behaviors of their apps have been found for a variety of Android apps [7,12–14]. End users may not agree with companies on which types of data are worthy of encryption and which should be shared to third parties. Apps often use advertising and analytic libraries that want to obtain as much information about users as possible. The app developers themselves are unaware of the data these services collect [15]. If the developers do not know about this information, how can they define their practices in their policies and how can users know how their personal data are used? Little insight exists into whether the security and privacy of end users' data is improving or declining.

In this paper, we investigate popular Android social media apps. We hypothesize that these apps will continue to exhibit the trend of collection and direct/inadvertent exposure of privacy sensitive information, as well as a lack proper disclosure in their privacy policies. To determine this, we analyze each app's data transmission practices, privacy policies, advertising/analytics libraries used, and business relationships. We compare our findings to the apps' privacy policies to determine how well their actions match their disclosures. We leverage dynamic analysis and traffic analysis to expose the security and privacy concerns of apps transmitting data in the clear, or unencrypted, and discuss the implications which arise.

Our analysis framework entails three modules that each output security and privacy concerns of user's information. The *Application Analysis* module collects, parses, and analyzes the app traffic to determine the app's behaviors and

the types of data being transmitted unencrypted. The *Policies* module analyzes the privacy policies of each app. The outputs of these two modules are manually compared, leveraging the Platform for Privacy Preferences (P3P) Specification terminology to find inconsistencies between an app's behaviors and its policy [16]. The *Data Sharing* module also leverages the output of the *Application Analysis* to analyze the advertising and analytics libraries used by each app as well as the business relationships between app companies and third parties. For more information on the analysis framework, see the Methodology section.

We find the majority of the apps analyzed transmitted most of their data unencrypted over HTTP. Furthermore, none of the apps' privacy policies specify which data are sent unencrypted vs. encrypted. We also find inconsistencies between app behaviors and the disclosure in their privacy policies. Thus, end users rely on vague, and sometimes misleading, documentation to determine how their data are used and handled. In general, our results show that the apps that mostly used HTTPS had fewer data leaks, as expected, but even so the app that transmitted the most PII unencrypted in our emulated environment tests also used HTTPS for the majority of its transmissions. Overall, these transmissions detail which types of data these companies determine worthy of encryption. We find one app transmitting PII and sensitive data over HTTP, including but not limited to message content, location data, and phone numbers. Multiple apps leak app usage trails and the interests of the end user by transmitting pictures, videos, and webpages as they were viewed in the app. Several apps transmit location information, some more granularly than others. On top of these information leaks, more data and conclusions may be gathered through inference attacks. Previous work found similar results, without distinguishing unencrypted data from encrypted data, and noted how this information is valuable to advertisers [4]. The profiling of users, targeted advertising, and formation of an attack surface are all things that could follow from such data leaks.

The transmission of data unencrypted results in inadvertent exposure to the network. Every hop has the ability to collect this information, and Internet Service Providers (ISPs) have been known to capitalize on this fact. For a short time in 2016, AT&T allowed users to pay for privacy by opting out of its 'Internet Preferences program' for additional fees between \$531 and \$800 per year. This program collected, tracked, and monetized users' behaviors across the Internet, using deep-packet inspection to obtain data. This pay-for-privacy plan was heavily criticized then, and the extra fees were removed, but AT&T and Comcast have both publicly justified and expressed interest in standardizing these programs in the near future [17,18].

The data collected by these ISPs could be used for more than advertising purposes though, such as treating customers differently. At one point, the cable company CableONE, 'bragged that it provided worse customer service to bad credit customers [18].' Additionally, 'Comcast, AT&T, Verizon and other

large providers have repeatedly argued that privacy rules governing broadband connections are completely unnecessary [17]. A Federal Communications Commission (FCC) ruling proposed in October 2016 would have prohibited ISPs from collecting and selling user data they obtained from traffic, including sensitive information. The ruling would have imposed restrictions including an opt-in program for the collection and selling of sensitive information, and an opt-out program for non-sensitive information [19]. But this ruling never took effect. In late March 2017, the US Congress passed a bill to overturn the regulations, and soon after this bill was also signed by The President [20]. Additionally, now the FCC cannot pass similar privacy regulations in the future, and ISPs are legally allowed to continue competing in the digital advertising market, an \$83 billion dollar industry [9].

For most categories of apps, sensitive data exposure would be minimal, but social media apps are unique in that using personal data is in their nature. Therefore, when one of these apps transmits images unencrypted over HTTP, it is inherently more of a security concern since the image is more likely to be personal. A game app may transmit images unencrypted, but this is unlikely to leak any personal information, since, in general, users don't send/receive personal information in game apps. For this reason, we focus on analyzing apps in the context of their social media category, and discuss the associated implications. In terms of unencrypted sensitive data exposure in our dataset of social media apps, we find it to be minimal for some, medium for others, and particularly large for one. We make the following contributions:

- We develop an analysis framework to expose the direct/inadvertent leakage of sensitive user data by popular Android social media apps, compiling what we and others have found these apps transmitting, and investigate their privacy policies, advertising practices, and business relationships.
- Leveraging P3P, a framework for privacy policy standards, we analyze the privacy policies of the apps in our dataset. We attempt to match each app's behavior observed over HTTP (unencrypted) to corresponding P3P data type category(s) and data collection purpose(s). We also note our interpretation of whether this behavior was used for advertising purposes, if it was inferred or explicitly observed, and whether the privacy policy discloses this behavior in general. Additionally, we list every file type we observed each app transmitting unencrypted.
- We provide an overall breakdown of each app's traffic, and the detailed results of our *Application Analysis* module for each app: a graphical summary of the end points they are transmitting with, which includes the network protocols used and their percentages; how much data is transmitted; and the frequency of the data per end point. These graphs, sorted by the bidirectional data transmitted, and truncated to the top 10 connections, can be found in [Appendix A](#). Additionally, we also analyze the

network loads throughout the app's usage, and generated graphs for these during our analysis.

- We propose a unique testbed environment for investigating and analyzing Android apps, comprising an isolated and emulated environment and several publicly available analytic tools. We leverage these environments and tools in several scripts to automate the collection and initial analysis process of gathering network traffic statistics.
- We analyze the revisions of these app's privacy policies between the main experiments' time of testing and about one year later, noting how and if their practices or disclosures have changed.
- We conduct follow-up experiments at the end of our study, re-running the Application Analysis module on three apps to gather newer traffic analysis data and briefly note the changes observed within each app's data transmission practices. The results of these experiments are only discussed in [Section 4.5 End of Study Follow-up Experiments and Observations](#), while the results from the main experiments were the focus of our study and are discussed throughout.

2. Related works

A multitude of research has focused on privacy policies as well as app traffic analysis in recent years, but few have combined these for comparison. The majority of these studies used static analysis of app code to determine the app's privacy-sensitive actions. One study and its follow-up studies used dynamic analysis, as we do, to determine the app's actions. This study, however, only focused on apps geared toward children. Additionally, these studies each covered a large portion of apps and thus did not take into consideration the context of each app during analysis. Also, until recently, privacy policies had been absent for a large portion of the apps these studies covered, limiting the studies' depth and coverage.

Our investigation studies the top social media apps in depth, taking the context of each app into consideration. A game app, such as Angry Birds, sending images unencrypted is much different than a social media app, such as Tinder, sending images unencrypted, and thus these apps should be studied in context. Even two social media apps, such as Tinder and Imgur, have different implications if they were both found to send an image unencrypted.

2.1 Privacy policy analysis

Discussions of privacy policies have become more prominent in research over the last two decades, mostly focusing on those pertaining to websites. Some studies have analyzed policies and compared them to the company's data-

collection practices. Several efforts have aimed at standardizing the format of these policies, making it easier for companies to produce and maintain them, as well as increase user comprehension. In 1998, the Federal Trade Commission (FTC) reported that 85% of the websites studied collected personal information, while only 14% disclosed these practices [21]. Several years later, P3P provided websites with a standard format for privacy policies, which would enable automated interpretation and inform users of a site's data collection practices [16]. Following this, E-P3P was developed, providing enterprises with a system for formalizing and enforcing their privacy policies in an attempt to prevent privacy violations [22].

A decade later, now in the era of mobile apps, the FTC analyzed privacy policies of apps known to have a child audience in order to find violations of the Children's Online Privacy Protection Act (COPPA). Throughout their reports, they continued to find the majority of these apps did not disclose their data collection and sharing practices. In their first survey, only 16% of the 400 Android/iOS apps studied (200 from each operating system [OS]) provided a link to a privacy policy or other disclosure. Later that year, their follow-up survey found this number had risen minimally, to 20% of the 400 apps [6,7]. Soon after, the FTC released a report detailing mobile privacy suggestions for platform/OS providers, app developers, advertising networks/third parties, as well as researchers in the hopes to improve disclosures [23]. A few years later, the FTC revisited their studies, this time with 364 Android/iOS apps for kids, and found 45% with direct links to their privacy policies on the app store page; a significant increase from before, but still less than half. An additional 38 of these 364 apps had privacy policies in other places, such as in the app or on the developer's website. Of all these apps, only 48 (13%) contained 'short form disclosures in their app descriptions about the sharing of personal information with third parties, the use of persistent identifiers, in-app purchases, social network integration, or the presence of advertising' [10]. That same year a study of popular, free Android/iOS apps found that fewer than one-third of their policies specifically stated the use of encryption methods for certain types of data, and only some of these policies named the encryption technologies in use. Furthermore, most of the policies contained general disclaimers about the Internet being an insecure environment and that no guarantees could be made about the safety of end user data [11]. Around this time, another group focused on Facebook's privacy policy throughout years of revisions and found it declining in measures of privacy protection and transparency [24].

2.2 Application traffic analysis

Android app security and traffic investigations have discovered and discussed a misuse of PII and a significant amount of data being sent unencrypted via HTTP [3,25]. It has been hypothesized that the large amount of HTTP traffic seen indicates an inappropriate use of the protocol for traffic including social apps, music, and

video [25]. Studies investigating the end user data collection practices of Android apps began shortly after the birth of the Android OS. In 2010, an investigation of 101 popular iOS/Android apps used dynamic analysis to find 55% transmitting unique device IDs to third parties without disclosing this to end users [1,2]. Furthermore, of the 50 Android apps, 21 shared location data, 5 shared usernames/passwords, and 2 shared age/gender with third parties. In 2011, Enck et al. decompiled 1,100 free Android apps and used static analysis to find 51% using ad/analytics libraries, which collected various types of personal data. These libraries collected a combination of location data, phone numbers, and unique device IDs [3]. Another study that year noted similar findings and found the security methods put in place to be insufficient, leaving some data transmissions susceptible to snooping [4]. Later research focused on tracking the information flows of user's personal data using static taint analysis and continued to find instances of potential misuse [26,27]. As discussed, earlier in 2012 the FTC conducted surveys on apps geared toward children, and in their second survey, they analyzed the unencrypted traffic of the apps, finding 56% of the 400 Android/iOS apps transmitting the device ID to ad/analytics networks or other third parties. Of all the apps, 3% transmitted geolocation and 1% transmitted the phone number to the developer or third party. Additionally, every instance where the geolocation or phone number was shared also had the device ID shared, which could be used to tie together the information and maintain a profile of the user [7].

The trend continued in 2015, when Walnycky et al. analyzed 20 popular Android instant messaging apps and found that for 16 of them, they could reconstruct some unencrypted data transmitted, including text messages, multimedia content, URLs for server-side content, chat logs, and some passwords. We investigated two of the same apps: Messenger and Tinder. They were able to reconstruct images sent and received, as well as video thumbnails received through Messenger. They noted that although Tinder does not emphasize security, they found no vulnerabilities, and that it encrypted the traffic, data storage, and server storage [28]. Our findings for these two apps varied greatly, with Tinder transmitting several types of sensitive data and Messenger encrypting nearly everything [29,30]. Following this, another study of popular Android apps was published at the end of 2015, finding 73% sharing sensitive data/PII with third parties [8].

Zang et al. focused on identifying apps sharing personal data to third parties and leveraged a man-in-the-middle (MITM) attack to decrypt encrypted traffic. They researched Android and iOS apps, and had four in common with our study: Facebook, Messenger, Pinterest, and Textfree [30–33]. Their reports of the data transmitted by Pinterest and Textfree were somewhat similar to ours, with ours being a subset of the data transmissions seen, but their findings differed greatly. This is because they did not analyze the data in as great of detail and focused on all of the data transmitted, without discerning between unencrypted and encrypted traffic. We focused on investigating unencrypted traffic, which is

entirely exposed to the networks along the packet path. They also seemed to only focus on identifying specific data they inputted in the traffic, not thoroughly analyzing the traffic for other sensitive data leaks. We had few results for Facebook and Messenger, since most of their traffic was encrypted, but we did note device information being transmitted by both, which Zang et al. did not. In general, they noted a lot of additional data transmissions we did not observe since they were encrypted. A few include Pinterest sending 'Post' data and a username to a third party; Facebook, Pinterest, and Textfree transmitting passwords to their app companies; and all four transmitting 'Location' data to third parties/app companies [8].

The traffic analysis by Zang et al. was also not as granular as ours for some data fields, as they grouped together a few similar types of data and reported the findings in terms of these groups. For example, they group 'current GPS location' and 'city' into the 'Location' group. Similarly, 'texts', 'chats', and 'likes' are within the 'Post' group. Therefore, when their findings state that Textfree sends 'Post' and 'Location' to pinger.com, the app's parent company, it is unclear if this is just a 'like' and 'city' transmitted, or 'texts' and 'current GPS location'. Also, because the 'Post' data were transmitted to pinger.com and not a third party, they did not have this data finding listed in any of their tables nor discuss it further. They do mention several of the apps that were the top offenders of transmitting sensitive data and PII, noting that Textfree did transmit the current GPS location. But in general, they do not elaborate for the majority of the apps at this level of granularity. Furthermore, in the context of their findings, it is logical that the parent company receives this information in order to process and provide their services. What they didn't expose is the fact that the majority of these data are transmitted unencrypted, thus exposed directly and indirectly. In general, our findings reflect theirs; we both report that of these four apps, Facebook had the fewest sensitive data leaks, followed by Messenger, then Pinterest, and finally Textfree [8]. Since they focused only on direct data leaks to third parties, had a higher level of granularity in their findings, and did not discern between unencrypted and encrypted transmissions, they missed a multitude of pertinent findings. But since their analysis did include encrypted data transmissions, by combining their findings with ours, a general idea of the types of data the app transmits encrypted vs. unencrypted and who receives them can be determined.

2.3 Comparing actions to policies

As previously mentioned, in their 2012 follow-up study, the FTC dynamically analyzed iOS and Android apps for children and compared the app's practices to their privacy policies. Their results showed that most apps did not provide any information on their data collection practices, and without disclosing it, found many sharing information with third parties and ad networks. One

example they provided was an app transmitting the device ID, geolocation, and phone number to several advertising networks, and then stating in its privacy disclosure that it does not share or sell information to third parties. Similarly, they found another app with a vague and misleading privacy policy sharing the device ID and geolocation with advertising networks without disclosure. Overall, 'while 59% (235) of the apps transmitted device ID, geolocation, or phone number either to the developer or a third party, only 20% (81) of the apps reviewed provided any privacy disclosures to users [7].' As discussed earlier, the FTC revisited their studies a few years later, now using a proxy tool to analyze both the HTTP and HTTPS traffic. Their privacy policy analysis was explained, but the results of their traffic analysis do not seem to have been completed or released [10].

The following year, a study used static analysis to compare popular app's actions to their privacy policies, noting that 23.6% of the nearly 1,200 apps studied contained at least one inconsistency [12]. A more recent study focused on automating the detection of an app's compliance to their privacy policies and the legal issues involved. They combined machine learning-based privacy policy analysis with static analysis of app code, and noted that without notifying end users, up to 41% of the nearly 18,000 apps studied could be collecting location data and 17% could be sharing this data with third parties. Further, each app had an average of 1.83 potential inconsistencies between the app's code and the privacy policies. Furthermore, only 52% of the apps provided links on the Google Play Store to their privacy policies, and of those that did not, 71% exhibited one or more data practices [13]. Another similar study aimed to solve the same issue by linking privacy policy phrases to application programming interface (API) methods in the app code that handled sensitive data and analyzing the information flow for inconsistencies [14]. Both methods identified privacy policy violations with the former achieving better accuracy.

Until now, research comparing apps' privacy policies to their actions has almost exclusively leveraged static analysis. Furthermore, these studies have been large scale, mostly focusing on automating the analysis and covering as many apps as possible, instead of detailed analysis on each app and its inconsistencies. They also seemed to have focused more on the legal aspects of detecting these inconsistencies, rather than how users are directly affected: their personal data excessively collected and shared. Our work entails a dynamic analysis approach to expose the practices of a specific area of interest: social media apps. Using a dynamic analysis approach gives some advantages, such as collecting the network traffic to gather statistics, endpoints, and other companies the app communicated with. Also, we can directly discern between unencrypted and encrypted traffic. We consider this unencrypted data easily accessible by an adversary, and thus could be collected and abused.

The security and privacy concerns our *Application Analysis* module outputs have been gathered through analysis of HTTP traffic. Note that because we do not employ techniques to analyze the encrypted traffic, we may not expose all of the data transmission practices that are inconsistent with their app’s privacy policies. We aim instead to expose the inconsistencies found in unencrypted transmissions, as the FTC did in their 2012 follow-up survey [7]. Unlike the FTC studies, we disclose the apps in our dataset to inform users of the specific findings per app. Our work, conducted nearly four years after theirs, ensures an isolated testing environment to prevent tainting of traffic from background apps. Additionally, we aim to describe the entire picture of each app, analyzing the traffic, privacy policies, ad/analytics networks used, and even business relationships.

3. Methodology

Our analysis framework, seen in [Figure 1](#) and detailed in the section Analysis Framework, comprises the following three modules: *Application Analysis*, *Policies*, and *Data Sharing*, each of which have components focusing on a piece of the overall analysis process. Both automated and manual analysis are used in our methodology, which is feasible and required in order to analyze our dataset of social media apps in-depth, with the context of each app in mind.

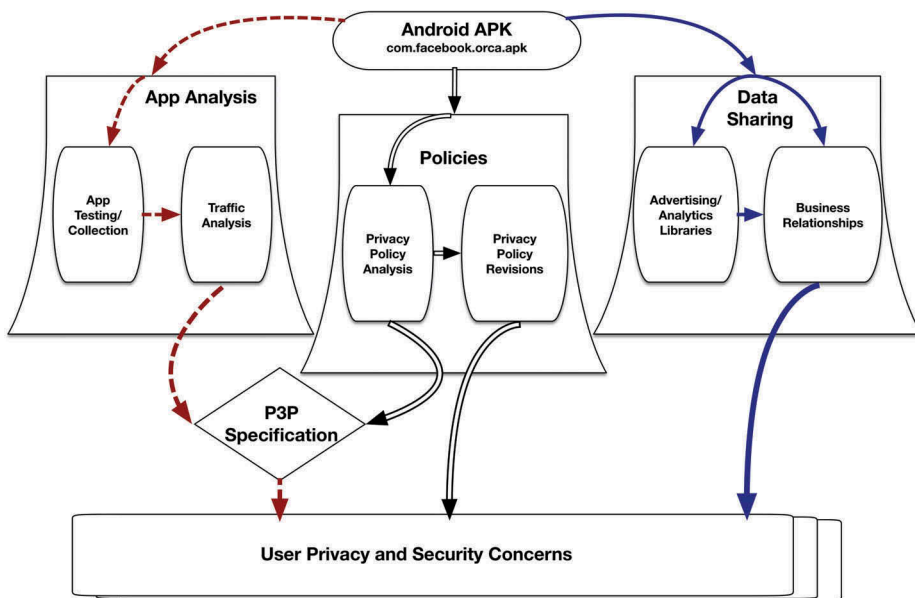


Figure 1. Analysis framework. Arrows represent the transfer of data and analysis information between the modules and components within the analysis framework. Red arrows with dotted lines, white arrows with hollow lines, and blue arrows with solid lines represent the data transferred by *Application Analysis*, *Policies*, and *Data Sharing*, respectively.

3.1 Application dataset

Our app dataset, seen in Table 1, comprises nine popular social media apps: Facebook, Imgur, Messenger, Pinterest, Textfree, Tinder, Tumblr, Twitter, and Vine [29–41]. It is important to note that the application experiments marked ‘(Follow-up)’ were conducted at the very end of our study, and the results of these are only discussed in Section 4.5 End of Study Results and Observations, while the results from the main experiments were the focus of our study and discussed throughout. Our dataset consisted of these apps in the Android Package Kit (APK) format, the package file format for Android apps used for installation. The apps that we attempted to use, but removed from our dataset, included Badoo – Free Chat & Dating App, Facebook Lite, Instagram, LinkedIn, MSQRD, Oovoo, Periscope, POF Free Dating App, Snapchat, Tango – Free Video Call & Chat, and TextNow – free text + calls. These apps were removed for various reasons such as not running properly in our environment, and are discussed in more detail later in this section. As seen in Figure 1, the APKs are the start of our analysis framework, and each APK is passed into all three modules for analysis.

When determining which social media apps to include, we went to the ‘Top Free in Android Apps’ page on the Google Play Store and chose the popular social media apps representing a wide variety of uses [42].

We focused mostly on apps we knew were popular and widely used, but also some slightly less used apps to determine how these differentiated, if at all, from the most popular. To obtain the APKs of these apps we used the

Table 1. Social media applications tested. Apr-2016 and Apr-2017 columns provide the number of cumulative downloads in millions (M) or billions (B) at this point in time.

Application	Environment	Test Date	Version #	Release		Apr-2016	Apr-2017
				Date	Date		
Facebook	Emulator	3/25/16	58.0.0.28.70	12/16/15		1–5B	1–5B
Imgur	Emulator	3/28/16	2.4.3.501	3/23/16		1–5M	5–10M
Messenger	Emulator	4/4/16	52.0.0.19.66	12/17/15		1–5B	1–5B
Pinterest	Emulator	3/28/16	5.12.1	3/24/16		100–500M	100–500M
	Routed Phone	4/7/16	5.12.1	3/24/16			
	Routed Phone (Follow-up)	11/15/17	6.43.0	11/14/17			
		12/7/17	6.43.0	11/14/17			
Textfree	Emulator	3/29/16	5.10.1	3/9/16		10–50M	10–50M
	Routed Phone	4/11/16	5.10.1	3/9/16			
	Routed Phone (MITM)	4/11/16	5.10.1	3/9/16			
	Routed Phone	2/16/17	5.10.1	3/9/16			
	Routed Phone (Follow-up)	11/15/17	8.3	11/10/17			
12/7/17		8.4	11/27/17				
Tinder	Routed Phone	4/20/16	4.4.4	12/12/15		50–100M	50–100M
	Routed Phone (Follow-up)	11/15/17	8.1.1	11/8/17			
Tumblr	Emulator	3/28/16	5.6.1.00	3/23/16		50–100M	50–100M
Twitter	Emulator	3/28/16	5.101.0	3/21/16		100–500M	0.5–1B
Vine	Emulator	3/28/16	3.3.13	11/20/15		50–100M	50–100M
	Routed Phone	4/7/16	3.3.13	11/20/15			

online tool, APK Downloader, from a website which claimed to host the latest Android APKs for a variety of apps [43]. In order to determine how recent these APKs were, we obtained the APK version numbers using the tool `aapt` included in the Android Software Development Kit (SDK), and compared these against the version numbers and release dates provided on the website AppBrain [44,45]. Additionally, to view some of the older versions which AppBrain no longer listed, we leveraged the WayBack Machine to visit the previous versions of these pages [46]. We were able to obtain recent releases of the APKs for the majority of the apps, and releases a few months old for the rest. Vine was the only case in which the privacy policy was updated between the APK release date and the date of testing. The privacy policy changes between these versions consisted of a slight change to one sentence and one additional sentence, neither which pertain to the behaviors we observed the app exhibiting. Additionally, during this study Vine first became 'Vine Camera', an add-on for the phone's camera to take and edit videos to then be shared on Twitter (suggested) or elsewhere, and then eventually moved to an archived state. This research focuses on the original version of the Vine app [41].

During our initial phase of app testing in the emulated environment, several apps did not start or run properly, leading us to drop them from our dataset. Additionally, some of these more popular apps were later tested in our rooted phone experiments, but were still excluded due to a variety of reasons. Of these apps, several did not run properly on the rooted phone, such as Snapchat, which couldn't access the camera. A few apps used IPv6 for the majority of their transmissions, which we excluded since we wanted to focus only on IPv4 for this study.

3.2 Testbed environment for app analysis

During our investigation of these social media apps, we made use of several open-source tools to create our unique testbed environment seen in [Figure 2](#). We analyzed the majority of our apps in an emulated environment and a subset of them on a rooted Android phone. For the first case, we used the Android emulator within Google's Android Software Development Kit (SDK) [45]. We felt that this was the best option over third-party emulators since it is supported by Google, widely used by Android developers, and enables emulation of most apps on the majority of Android OS versions. For both the emulator and rooted phone experiments, we leveraged the Android SDK's debugging tools to analyze apps [45]. To root the Google Nexus 6P, we followed an instructional guide online and used the tool `SuperSU` to manage root access [47,48]. Throughout the rooting process we leveraged several more programs including `fastboot`, `TWRP`, a Nexus 6P USB Driver, and

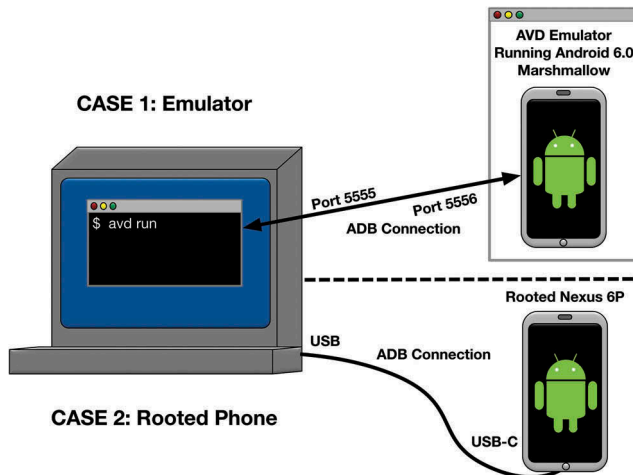


Figure 2. Our testbed environments for the emulator and rooted phone experiments.

a Nexus 6P boot image, downloaded from an Android ROM/file hosting website (<http://downloadandroidrom.com/>).

For each app, we begin by downloading the most recent APK file available [43]. Each time we test an app, we create a new Android Virtual Device (AVD) running Marshmallow OS (Version 6.0) [45]. This is done at the start of each experiment for a given app. AVDs contain the hardware profile, system image, and other properties. For our testing, we use the default profile. AVDs are then run in an emulator, which is a separate program also provided by Google that simulates a device and displays it on the computer. It supports most features of a device, which allows it to run most apps seamlessly. The emulator system leverages a version of `tcpdump`, which we used to collect every packet sent/received by the emulator between starting it up and closing it [49]. Using the Android Debug Bridge (ADB), we are able to interact with the emulator through our host computer's shell [45]. We use ADB to install, open, and uninstall apps, though it has many other features as well, including opening a shell within the emulator. We use the tool `Monkey` in the Android SDK just for starting up each app. To collect information about the specific APKs, such as permissions, we utilized the Android Assist Packaging Tool (AAPT), which is the main building tool for Android apps [45]. For each app studied in the emulator, we conducted three 15-minute-long experiments and collected the pcaps during each:

- Test 1: Open the app and idle without logging in.
- Test 2: Open the app, have the user log in, and then remain idle.
- Test 3: Open the app, have the user log in, and then perform normal activity within the app for the duration of the collection.

By collecting traffic when the app opens, but the user does not log in (Test 1), we hoped to capture most of the base traffic we could expect the app to generate. Then, by logging in and sitting idle (Test 2), we are able to focus on the initial burst of traffic when the app first loads the user data, updates feeds, etc. We also wanted to see if the app would transmit traffic after a certain period of time, following a predictable pattern, while just sitting idle. In the third capture, we executed a wide variety of actions to emulate a typical/normal use of the app (Test 3), such as liking/following pages and users, writing posts, and messaging other users. Our procedures varied for each app due to the inherent differences in functionality of these apps. Our procedures aimed to manually replicate a typical user, and the app data generated in our experiments should be taken as a general case scenario, as we did not exhaustively test each app's entire functionality. For example, during our Facebook experiments, we simulated the social media actions an average user would perform, but did not play third party games within the app [32]. As we discuss in the following section, we believe that we obtained a good snapshot of the majority of the apps tested and that our procedures performed adequately. The alternative procedure we considered was to leverage the tool *Monkey*, from the Android SDK, to execute pseudo-random user actions while running an app [45]. These pseudo-random actions were essentially touches/swipes on the screen, and few significant actions seemed to be performed, as navigating through many of these apps requires a more complex sequence of events.

To analyze our collected network traffic, we first developed a tool that summarizes the major findings in the traffic by leveraging several analytical tools, notably *Dshell* and its *ip* decoder, *tcpdump*, *matplotlib*, and *ipinfo* [49–53]. These analytic tools were additionally used for further analysis. The network forensic analysis framework *Dshell*, developed by the U.S. Army Research Laboratory, enabled us to analyze our traffic quickly and efficiently. For example, it allowed us to focus our analysis on specific traffic protocols at a time, and translate IP addresses in our traffic to their associated Autonomous System Numbers (ASNs) in order to link IP addresses to organizations. We leveraged several of *Dshell*'s decoders, including *ip*, *followstream*, *netflow*, *rip-http*, *httpdump*, *sip*, and *rtp*, to reconstruct the traffic flow from the *pcap* and manually analyze it for sensitive data and PII [50]. We used *tcpdump* mostly for the actual collection of the app traffic, and it was accessed through the tool *ADB* for use with the emulator [45,49]. It is important to note that *ADB* contributes to the traffic collected on the emulator. From the emulator's point of view, this traffic is *localhost* traffic (on a loopback interface) that is on the same subnet as the device's IP, located in the port range of 5555 to 5585. Additionally, to analyze the data more easily, we group together all outgoing and incoming traffic by server/port combinations. Thus, we view this sent and received data to/from a specific server/port as a single connection, as seen in [Figure 3](#).

Finally, we visualized our findings using graphs generated by *matplotlib*, a Python library for 2D plotting [52]. The main graph outputted is a pie chart with

multiple legends including 'Connections Made with Phone', 'Traffic Volume', 'Estimated Frequency', and 'Breakdown of Traffic' with the three main networking protocols found in our experiments: HTTPS, HTTP, and DNS. We generated several versions of these graphs, sorting the connections by frequency or bidirectional data transmitted and truncating the number of connections to the top 10 when plotting. The data-sorted graphs generated for each of the apps in both our emulator and rooted phone experiments can be found in [Appendix A](#).

All of the connections seen and the data associated with them are also recorded and stored in a corresponding text file. These text files were used for a more thorough analysis of the connections seen in each app. A frequency-sorted plot generated during Vine's emulator experiment can be seen in [Figure 3](#). The second graph we generate in our experiments contains two network load graphs, both displaying the same information but with one binned by 1 minute and the other binned by 10 seconds, as seen in [Figure 4](#). We call these the 'Network Load' graphs, displaying the packets seen vs. time into the packet capture at two different levels of granularity. From this we can see how much traffic was being sent/received throughout the duration of the test. These graphs provide a bidirectional, summarized view of the traffic seen in the app's test and enabled us to pinpoint some key connections to look into further with manual analysis, while determining the percentage of traffic sent unencrypted.

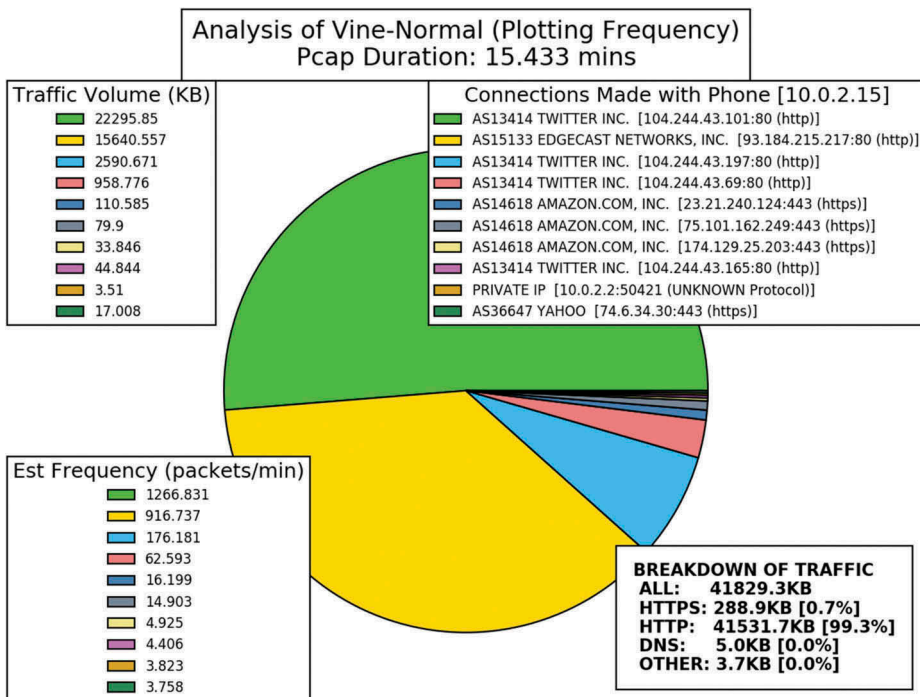


Figure 3. Analysis of Vine traffic – Android emulator.

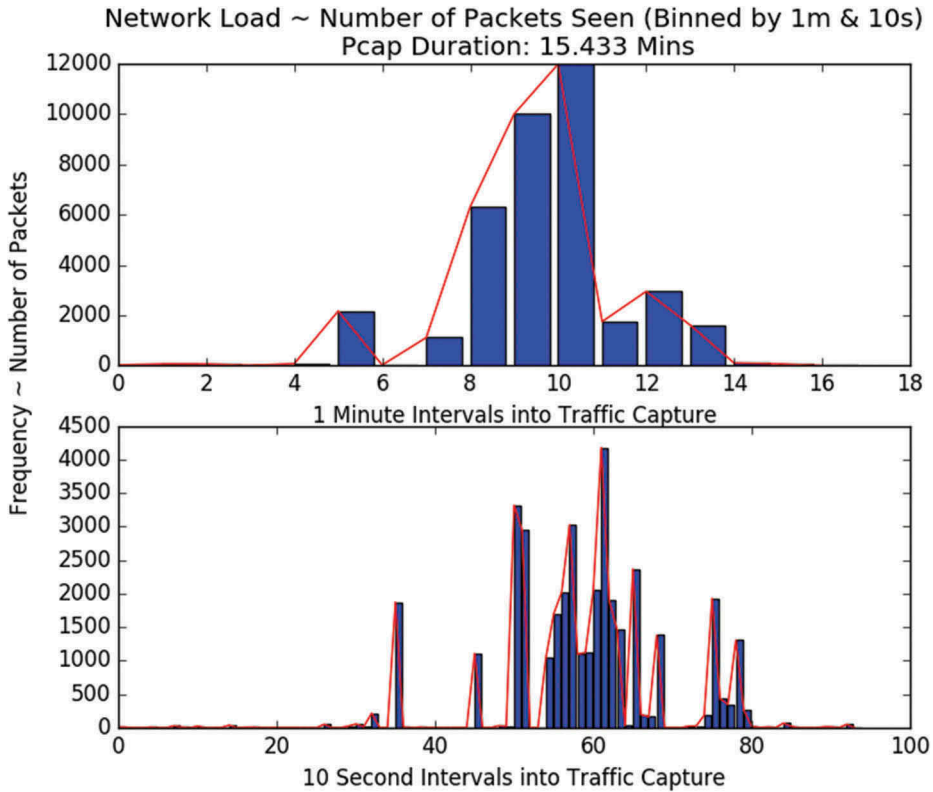


Figure 4. Network load for Vine traffic – Android emulator.

To investigate apps that did not run properly in the emulator and ensure that the emulator was producing normal and accurate app traffic, our second case of experiments were conducted with a rooted Google Nexus 6P phone. Similar to the emulator setup, we used ADB for the rooted phone experiments, but now connected the phone via USB-C to our Apple MacBook Pro laptop, as seen in Figure 2 [45]. Since the phone was rooted, we had control to install software. Using a guide online titled ‘Installing tcpdump for Android’, we installed the precompiled tcpdump binary for Android made available by a project called *Android tcpdump* [51,54]. For these experiments, we only conducted one test per app (Test 3), thus we manually installed the app, started tcpdump, opened the app, and simulated the typical/normal use [51]. Just as in the emulator environment, we made sure that each time we were collecting traffic, the only third-party app installed on the device was the one under investigation. It is important to note that the default system apps were still present on the phone and may have had a slight impact on the traffic collected.

The phone proved to be much faster than the emulator, resulting in more actions completed per minute. To compensate for this, we lowered our time from 15 minutes (in the emulator collection) to around 5-10 minutes for those apps

which had already been tested in the emulator. Additionally, we do not directly compare the results according to the amount of traffic seen, but instead by the percentage of traffic per protocol. With this in mind, we compared our results and found them to be analogous, as we discuss later in the Breakdown of Traffic section. Next, we discuss a high-level overview of our analysis framework.

3.3 Analysis framework

The following sections detail the inputs and outputs of the modules and components in our analysis framework, as also described in [Figure 1](#).

3.3.1 Application analysis module

3.3.1.1 Application testing/collection. Inputs an APK and outputs a pcap of the traffic collected. It performs dynamic analysis: running apps in an isolated, emulated environment during case one and on a rooted phone in case two.

3.3.1.2 Traffic analysis. Inputs a pcap and generates network and data statistics using our programs and scripts, which combine several publicly available analytic tools. This is then followed by manual analysis of the outputted graphs and summaries, as well as further detailed analysis of the traffic. Output is then compared with the output of the *Privacy Policy Analysis* component of the *Policies* module and the P3P Specification component in order to classify app data transmitted and determine disclosure of these practices.

3.3.2 Policies module

3.3.2.1 Privacy policy analysis. Inputs an APK and entails the manual analysis of that app's privacy policy for the main security/privacy practices it discloses and details, as well as what data are transmitted/collected and how this is performed. Output is then compared with the output of the *Traffic Analysis* component of the *Application Analysis* module and the P3P Specification component to classify app data transmitted and determine disclosure of these practices.

3.3.2.2 Privacy policy revisions. Inputs the current privacy policy and the policy in place during testing, and outputs comparisons and contrasts to determine policy changes.

3.3.3 Data sharing module

3.3.3.1 Advertising/analytics libraries. This looks at inputs an APK to determine the advertising and analytics libraries used by each of the apps. Leveraging the open-source tools *Dare* and *Soot* in several scripts, it retargets and decompiles the APKs to find which ad and analytic libraries are used in these apps [55,56]. This is followed by a quick manual investigation into the most

prominent to note which apps used them and what past studies have shown these libraries to collect/transmit, as well as their user-data handling practices.

3.3.3.2 Business relationships. This involves inputs an APK and the output from the *Advertising/Analytics Libraries* component to manually determine the relationships the app company has with other social media and advertising companies. This provides insights into how data inputted into one app could be shared among other apps/companies owned by or in business with a mutual parent company.

4. Evaluation

4.1 Breakdown of traffic

We analyzed and generated traffic analysis graphs and network load plots for each of the nine social media apps, using the methodology discussed previously. In Table 2, we organize our observations of the traffic loads seen per protocol in descending order of HTTP traffic percentages seen. We sort these data by focusing on three major protocols: HTTP, HTTPS, and DNS. Any traffic seen with a different protocol is grouped and listed as ‘Other’. It is important to note that not all of our experiments, for both the emulator and rooted phone environments, ran on the same exact network. Some experiments were run on university networks, others on apartment networks a few miles from campus. Therefore, we see different ISPs and CDNs communicated with in our experiments. It may be possible that specific networks, ISPs, or CDNs enforced more encryption and thus we cannot exactly determine if that app would have resulted in different experimental results on a different network. But, since encryption use did vary across our apps, and was mostly consistent across

Table 2. Breakdown of Application Traffic Using HTTP, HTTPS, and DNS.

App Name	HTTP	HTTPS	DNS	Other	Total KB
Vine	99.30%	0.70%	<0.1%	<0.1%	41,829.3 KB
Vine*	99.20%	0.80%	0.00%	0.00%	75,440.6 KB
Tinder*	92.10%	7.90%	<0.1%	0.00%	24,130.5 KB
Textfree*	84.30%	15.50%	<0.1%	0.20%	30,995.1 KB
Pinterest	84.20%	15.70%	0.10%	<0.1%	23,744.2 KB
Pinterest*	75.70%	24.30%	0.00%	0.00%	27,337.2 KB
Imgur	53.40%	46.60%	<0.1%	<0.1%	16,352.1 KB
Textfree* [△]	43.70%	53.70%	<0.1%	2.60%	8026.4 KB
Textfree* [▽]	41.50%	57.30%	0.40%	0.90%	4679.8 KB
Textfree	23.00%	74.10%	1.30%	1.60%	4164.8 KB
	8.50%	89.20%	0.90%	1.40%	7218.0 KB
Twitter	15.60%	84.30%	0.10%	<0.1%	12,250.0 KB
Tumblr	11.90%	88.00%	0.10%	<0.1%	24,513.6 KB
Messenger	5.80%	93.60%	0.30%	0.30%	1183.0 KB
Facebook	<0.1%	99.80%	0.10%	0.10%	5442.3 KB

* Application traffic collected on rooted Google Nexus 6P.

[△] Same APK and version retested after 10 months.

[▽] MITM (arp spoof) attack scenario.

different experiments of the same apps (and same exact versions), we believe the apps more so control the transmission practices.

As discussed earlier, we also collected traffic on a few apps using a rooted Google Nexus 6P. We set up the environment as best as possible to simulate an emulated, isolated environment by only ever having one of the apps we were testing on the phone at a time.

Unlike in the emulator case experiments, we did not perform all three tests (no login, login, and normal use) for the rooted phone. During the emulator experiments, we found that Test 1 and Test 2 provided little information compared to Test 3. Therefore, we only conducted Test 3, the normal use of apps, on the rooted phone. Additionally, we did not test every app on the rooted phone that we tested on the emulator; we only tested the apps that produced a large amount of unencrypted traffic in the emulator tests. This also enabled the verification of the accuracy of the traffic collected in the emulator compared to the real-world scenario with the rooted phone. The apps we investigated on the rooted phone were Vine, Pinterest, Textfree, and Tinder [29,33,35,41]. The latter two apps had performed poorly in the emulator experiments. Textfree was slow and crashed several times in the emulator and was attempted twice. We did notice that in the 7 MB of traffic collected in the first run and 4 MB in the second, sensitive data were being transmitted unencrypted, including entire text messages and phone numbers. Additionally, as shown in [Table 3](#), since the app did not run properly in the emulator, results varied, and little traffic was generated. The 7 MB run was 89.2% HTTPS and 8.5% HTTP, compared to the 4 MB run with 74.1% and 23%, respectively. We decided to retest this app on the rooted phone, which would allow for more accurate data for collection and analysis since the app could run properly. Tinder also had issues running in the emulator and we could not collect sufficient traffic data.

Two apps, Vine and Pinterest, were tested in both the emulator and rooted phone testbed environments. In [Table 2](#), we see that Vine running on the emulator had 99.3% of its traffic sent via HTTP, compared to Vine running on the rooted phone with 99.2%. Similarly, we see Pinterest on the emulator had 84.2% HTTP compared to 75.7% on the phone. Unlike Vine and Pinterest, Textfree's traffic breakdown varied drastically between the emulator and rooted phone runs. This was due to the fact that Textfree was very slow and crashed multiple times on the emulator, so the data collected were not an accurate representation of its normal use-case. But, the rooted phone case experiments do provide an accurate representation of the app's normal use-case, and we were able to collect about 7.5 times the amount of traffic in two-thirds the amount of time, as seen in [Figure 5](#).

Toward the end of our data collection, Google released Android Studio v2.0, which provides a significant performance boost and drastically speeds up the apps running in the emulator. Future testing in this Android emulator environment may be able to provide comparable speeds to that on the rooted phone.

Additionally, we find the emulator and rooted phone experiments to be comparable as seen in the closeness of the traffic protocol percentages across both cases for Vine and Pinterest. Thus, we conclude that the emulator is a viable environment for testing apps with considerable accuracy. As mentioned previously, all data-sorted analysis graphs can be seen in [Appendix A](#).

4.2 Classifying and comparing traffic data to policies using P3P

To investigate and analyze the dataset of social media apps at a granular level, we leveraged the P3P Specification. Specifically we use their categories and purposes elements, which provide categorical meaning to the data a service collects, as well as a purpose for why they were gathered and how they will be used. An example of a P3P category is the ‘Physical Contact Information’ category, which is described as, ‘Information that allows an individual to be contacted or located in the physical world – such as telephone number or address.’ An example of a P3P purpose is ‘Research and Development’, which is described as, ‘Information may be used to enhance, evaluate, or otherwise review the site, service, product, or market. This does not include personal information used to tailor or modify the content to the specific individual nor information used to evaluate, target, profile or contact the individual [16].’

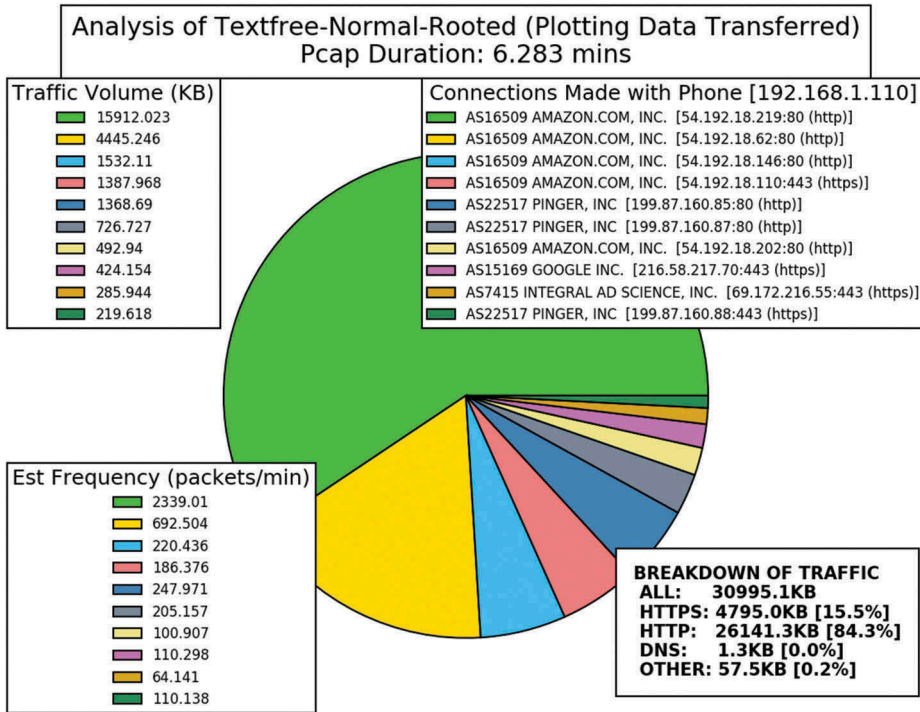


Figure 5. Analysis of Textfree traffic – rooted Nexus 6P.

By leveraging the P3P platform and its terminology we can classify the data found in the app traffic as well as map it to each app’s privacy policy. Within our framework, as seen in [Figure 1](#), the outputs of the *Traffic Analysis* component of the *Application Analysis* module and the *Privacy Policy Analysis* component of the *Policies* module are combined with the P3P Specification. The results of this app data classification are reported in [Table 4](#).

In [Table 4](#), we manually classify the app data transmitted over HTTP using P3P categories and purposes, and explain whether these data were explicitly stated or inferred, whether they were used for or with advertising, and whether it was disclosed within the app’s privacy policy. The table was created using our best interpretation of the app traffic, and the behaviors and actions these apps exhibited. For the scope of this research, we determined that the data were used for ads (transmitted to an advertising network) by the hostnames in the headers of all of the GET/POST request payloads containing the specific data. The table is broken out by app and the different types of data they each transmitted unencrypted. For each app and across all of its experiments, we state all of the file types found in the traffic (based on their file extensions). These file types are listed under the app’s name. A ✓ and x is used to represent YES and NO, respectively. This table was populated using only data apps transmitted unencrypted over HTTP. It is important to note that since our app tests were not exhaustive in testing the actions and behaviors that can be conducted within each app, our analysis is also not exhaustive. Additionally, in this research we only cover the app’s data practices within the scope of unencrypted traffic.

Table 3. Abbreviations of P3P Specification Categories and Purposes.

P3P: CATEGORIES					
PCI	Physical Contact Information	OCI	Online Contact Information	UI	Unique Identifiers
PuI	Purchase Information	FI	Financial Information	CI	Computer Information
NCD	Navigation & Click-stream Data	ID	Interactive Data	DSD	Demographic & Socioeconomic Data
C	Content	SMM	State Management Mechanisms	Pol	Political Information
HI	Health Information	PrD	Preference Data	LD	Location Data
GI	Government-issued Identifiers	O	Other		
P3P: PURPOSES					
CSA	Completion & Support of Activity for which Data was Provided	WSA	Web Site & System Administration	RD	Research & Development
OT	One-time Tailoring	PA	Pseudonymous Analysis	PsD	Pseudonymous Decision
IA	Individual Analysis	InD	Individual Decision	CVM	Contacting Visitors for Marketing of Services or Products
HP	Historical Preservation	CVMT	Contacting Visitors for Marketing of Services or Products via Telephone	OU	Other Uses



Table 4. Classifying application traffic data with P3P and determining policy disclosure. The abbreviation key is in Table 3. The list of file extensions found in the unencrypted traffic per app are listed below the app's name. Asterisks (*) denote borderline instances where the disclosure or classification could be argued, and our best judgement was used, to include instances of data transmitted that lacked disclosure and thus the P3P purpose is inferred.

Application	Data Transmitted over HTTP	P3P Category	P3P Purpose	For Ads	Disclosed
Imgur	User-agent field w/device info	CI	IA/InD	X	✓
	Referer info	NCD	CSA*	X	X*
	Content (pics) user viewed	NCD/C	CSA*	X	X
	Code files	C/O	CSA*	X	X
	HTTP cookies	SMM	CSA, RD, PsD	X	✓
	HTTP ETag	SMM	CSA, RD, PsD	X	✓*
	(Inferred) Potentially leaked user interests	PrD	N/A	X	X
Pinterest	User/real names used in profile pic filenames	OCI	N/A	X	✓*
	UID, unique identifier	UI	RD, PsD/InD	✓	✓
	Device info – OS/make/model/build	CI	CSA, RD, PsD/InD	✓	✓
	User-agent field w/device info	CI	CSA, RD, PsD/InD	✓	✓
	Referer info	NCD	CSA, RD, PsD/InD	✓	✓*
	Content (pics) user viewed	NCD/C	CSA*	X	X
	Code files	C/O	CSA*	✓	X
	HTTP cookies	SMM	CSA, RD, PsD, InD	✓	✓
	(Inferred) Potentially leaked user interests	PrD	N/A	✓	X
Textfree	Phone numbers/contacts	PCI	CSA, RD, PsD*	X	✓
	First/last name(can be nickname), sender name(name, or number)	PCI/UI/DSD	CSA, RD, PsD*	X	✓
	UID, unique identifiers	UI	CSA, RD, PsD	✓	✓
	Notification email, forgot password email	UI	CSA, RD, PsD*	X	✓
	User ID number	UI	CSA, RD, PsD*	X	✓*
	(Inferred, unpopulated fields) Facebook/Google+ IDs and tokens	UI	CSA, RD, PsD*	X	✓
	Username	UI/DSD	CSA, RD, PsD*	X	✓
	OAuth consumer key, nonce, sig, method, APNS notif. token	UI/O	CSA*	X	X
	Device info – OS/make/model/build, public/private IP addresses	CI	CSA, RD, PsD	✓	✓
	User-agent field w/device info	CI	CSA, RD, PsD	✓	✓
	Referer info	NCD	CSA, RD, PsD	✓	✓
	Account/message/call logs (msg/call ID, R/U, codec, creditBalance)	ID	CSA, RD, PsD	X	✓
	Birthday, age, gender, language, country	DSD	CSA, RD, PsD*	✓	✓
	User's Profile Picture	DSD/C	CSA, PsD*	X	✓
	SMS/MMMS messages, message signature	C	CSA, RD, PsD	X	✓
	Code files	C/O	CSA, PsD	✓	✓*
	HTTP cookies	SMM	CSA, RD, PA, PsD	✓	✓
	(Inferred) Potentially leaked user interests	PrD	N/A	X	X

(Continued)

Table 4. (Continued).

Application	Data Transmitted over HTTP	P3P Category	P3P Purpose	For Ads	Disclosed
	Location tracking, coordinates	LD	CSA, RD, PsD	✓	✓
	Location tracking, Wi-Fi mapping	LD	CSA*, PsD*	×	×
	Acc Settings: isPasswordSet, ShowAds, textfreeNotificationPrivacy, msgStatusPrivacy, textfreeIntercept, textfreeInterceptPhone	O	CSA, RD, PsD	×	✓
Tinder	IDFA, unique identifiers	UI	RD, PA/IA, PsD/InD	✓	✓
	Unique IDs per user within pic names	UI	CSA*	×	×
	Device info – OS/make/model	CI	(RD, PA/IA, PsD/InD)*	✓	×
	(Inferred) Potentially leaked Tinder ‘matches’ between users	NCD/ID	N/A	×	×
	Content (pics) user viewed	NCD/C	CSA	×	✓
	AWS (Server: AmazonS3) request ID and ID-2	ID/O	CSA	×	✓
<i>jpeg, html</i>	City/district name and zipcode	LD*	CSA, RD, PA/IA, PsD/InD	✓	✓
	(Inferred) Age-like number	DSD	CSA, RD, PA/IA, PsD/InD	✓	✓
	Code files containing city/district name, zipcode, age-like number	LD*/DSD/O	CSA, RD, PA/IA, PsD/InD	✓	✓
	HTTP ETags	SMM	CSA*	×	×
	(Inferred) Health info – sexual orientation	HI	N/A	×	×
	(Inferred) Potentially leaked user interests	PrD	N/A	×	×
	(Inferred) Location Tracking – City/district name and zipcode	LD*	CSA, RD, PA/IA, PsD/InD	✓	✓
Vine	User-agent field w/device info	CI	CSA, RD	×	✓
	Content (videos) user viewed	NCD/C	CSA, RD	×	✓
<i>jpeg, mp4.jpg, png, mp4</i>	AWS? (Server: ECAcc) request ID, version ID, ID-2	ID/O	CSA	×	✓
	HTTP ETags	SMM	CSA, RD, PA*, PsD*	×	✓
	(Inferred) Potentially leaked user interests	PrD	N/A	×	×
Tumblr	User-agent field w/device info	CI	CSA, RD	✓	✓
<i>png, js, html, css</i>	Referer info	NCD	CSA, RD	✓	✓
	Code files	C/O	CSA*	✓	×
	HTTP cookies	SMM	CSA, RD	✓	✓
Twitter	User-agent field w/device info	CI	CSA, RD, PsD/InD	×	✓
	HTTP cookies	SMM	CSA, RD, PsD/InD	×	✓
Messenger	User-agent field w/device info	CI	CSA, RD, PA/IA, PsD/InD	×	✓
<i>ogg</i>	.ogg audio file – failed call occurred	O	CSA*	×	×
Facebook	User-agent field w/device info	CI	CSA, RD, PA/IA, PsD/InD	×	✓

4.2.1 Populating and interpreting Table 4

Within the *Application Analysis* module, the *Traffic Analysis* component entails manually analyzing the data transmitted to determine the 'Data Transmitted over HTTP', 'Inferred', 'For Ads', and file extensions fields in Table 4. Next these data are cross-referenced with the P3P Specification to determine the 'P3P Category' field. Similarly, the *Policies* module's *Privacy Policy Analysis* component involves manual analysis of the app's privacy policy to determine the 'Disclosed' field. It is important to note that this column only means the data were disclosed in general, not that they were disclosed as being transmitted over HTTP, unencrypted. Next, this component's data, specifically statements from the privacy policy, are cross-referenced with the P3P Specification to determine the 'P3P Purpose' field. Thus, the data populating the 'Data Transmitted over HTTP', 'P3P Category', and 'For Ads' columns are solely based on our interpretation of the unencrypted traffic collected and for the 'P3P Category', additionally the comparison of this traffic to the P3P Specification. Similarly, the data populating the 'P3P Purpose' and 'Disclosed' columns are solely based on our interpretation of the given app's privacy policy and for the 'P3P Purpose', additionally the comparison of these statements to the P3P Specification.

To better understand how the table was populated, and which pieces of information were used to connect the traffic to the P3P terminology and the privacy policies, we go through the reasoning for the top row of the first app, Imgur:

- (1) In the *Traffic Analysis* component of the framework, we first gather all of the file extensions seen in the Imgur traffic. Next, we manually look through the unencrypted traffic and note that the OS, processor specifications, and build information of the Android device are found. In this case, the actual data found were 'User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; sdk_phone_armv7 Build/MRA44C)'. We summarize this finding as 'User-agent field w/device info' and populate the field in the 'Data Transmitted over HTTP' column. Because these data were explicitly found in the app traffic, they were not inferred. If these data had not been found in the traffic but could be inferred, we would prepend '(Inferred)' to 'User-agent field w/device info'. Next, since each of the instances of the GET Request headers containing these data had a host of 'Host: i.imgur.com', and thus in the traffic we collected it was not used for ads, a \times is placed in the 'For Ads' column, to denote NO. It is important to note that if the piece of data is found in multiple places within the traffic and used for advertising in at least one of these places, we denote the data with a YES in the 'For Ads' column. Thus, it may only be one instance of many, but we represent it as being used for advertising purposes at some point by the app. Additionally, since we are basing our analysis off of the data we have collected, if the privacy policy states that the data are used for advertising,

but we do not see this in the traffic we collected, we state that they were not used for ads.

- (2) The types of data gathered are then cross-referenced with the list of categories in the P3P Specification. We corresponded these data with the sixth P3P category 'Computer Information', which is described as, 'Information about the computer system that the individual is using to access the network – such as the IP number, domain name, browser type or operating system.'
- (3) From the manual analysis of Imgur's privacy policy within the *Privacy Policy Analysis* component of the *Policies* module, we find that they state, 'Our servers log information about each computer connecting with our site such as IP address, dates and times of each login, device characteristics, operating system, browser type, type of connection, page and image viewing statistics, and incoming and outgoing links. We associate this information with comments you post or votes you enter ... if you choose to give us personal data about you as described below, the technical information we collect that would otherwise be anonymous could instead be logged as coming from you [68].' Therefore, the transmission of these data (in general) was disclosed and a ✓ is placed in the 'Disclosed' column to denote YES. For the table, it is important to note that there may be instances where the app's privacy policy discloses the transmission of a piece of data, but does not disclose that it is used for advertising purposes. In these cases, we denote the data as *not* disclosed.
- (4) The key statements within the privacy policy are then cross-referenced with the P3P Specification to determine the P3P purpose to the best of our ability. In this case, it isn't specifically stated why the information is collected, but that it is tied together with user activity and thus creates a type of record that won't be anonymous if the user has provided personal data. Therefore, our interpretation is to loosely classify it as P3P purpose 'IA/InD', which denotes 'Individual Analysis/Individual Decision'.

As seen in the previous example, the manual population of [Table 4](#) was tedious yet necessary for this level of granularity. The information presented in [Table 4](#) is our best interpretation of the traffic and privacy policies we gathered and analyzed, and our best attempt to classify data that may be ambiguous. Due to the nature of ambiguous data and vague privacy policy statements, asterisks (*) are used to denote borderline instances where the disclosure or classification could be argued, and our best judgement was used. This additionally includes instances of data transmitted that lacked disclosure and thus the P3P purpose must be inferred.

4.3 Policies module

In the *Privacy Policy Analysis* component, we analyze the privacy policies in place for each of the apps during the time of testing. Since policies were updated between the time of analyzing the apps and their privacy policies, the WayBack Machine was used to grab the proper policy in place during the time of testing [46]. Each privacy policy was manually analyzed, but several tools such as `grep` were deployed to find terms more efficiently [57]. Each time a piece of data was uncovered during the analysis of the unencrypted app traffic, it was checked whether it was disclosed in the corresponding privacy policy. As mentioned previously, not all of the data uncovered were disclosed as being transmitted/collected, and furthermore, none of the policies properly disclosed that any data were sent unencrypted. The detailed results of this analysis can be seen in Table 4.

As part of the *Privacy Policy Analysis* component, we aimed to understand the security and privacy measures put in place by these app companies. Recall that, of all of the data transmission practices disclosed in these policies, none of the app policies discern which data are transmitted encrypted or unencrypted. Past research has shown app privacy policies use general disclaimers about the insecure nature of the Internet and state that no guarantees can be made about the confidentiality of end user data [11]. Of the nine apps in our dataset, seven loosely mention that security and privacy practices are used or attempted to protect their users' data. The remaining two apps, Pinterest and Vine, don't explicitly discuss security, or it is too vague to discern. For these two, we select quotes from them that are somewhat security related. The following statements have been taken verbatim from each app's privacy policy in place during the time of testing.

- Facebook & Messenger – 'We use the information we have to help verify accounts and activity, and to promote safety and security on and off of our Services, such as by investigating suspicious activity or violations of our terms or policies. We work hard to protect your account using teams of engineers, automated systems, and advanced technology such as encryption and machine learning. We also offer easy-to-use security tools that add an extra layer of security to your account. For more information about promoting safety on Facebook, visit the Facebook Security Help Center [58].'
- Imgur – 'We take every reasonable precaution to protect your private data from loss, misuse, unauthorized access, disclosure, alteration, or destruction [59].'
- Pinterest – 'We use the information we collect to provide our products to you and make them better, develop new products, and protect Pinterest and our users [60].'

- Textfree – ‘We have physical, electronic, and procedural safeguards to protect the loss, misuse, and alteration of the Customer Information under our control. Customer Information we collect is stored in a secure manner that is accessible only by authorized Pinger personnel, contractors, and agents bound by confidentiality obligations and is only disclosed under the provisions of this privacy policy. Our processing of Personal Information is only for the purposes for which it was collected and in accordance with this privacy policy. Periodically, Pinger undertakes a review of its privacy practices to ensure that that the information we collect is needed to provide or improve the services we provide [61].’
- Tinder – ‘We take security measures to help safeguard your personal information from unauthorized access and disclosure. However, no system can be completely secure. Therefore, although we take steps to secure your information, we do not promise, and you should not expect, that your personal information, chats, or other communications will always remain secure. Users should also take care with how they handle and disclose their personal information and should avoid sending personal information through insecure email [62].’
- Twitter – ‘We may share your private personal information with such service providers subject to obligations consistent with this privacy policy and any other appropriate confidentiality and security measures, and on the condition that the third parties use your private personal data only on our behalf and pursuant to our instructions [63].’
- Tumblr – ‘Your Account Information is protected by a password for your privacy and security. We may enable additional security features in the future, like multi-factor authentication. You need to prevent unauthorized access to your Account and information by creating a unique, secure, and protected password and limiting access to your computer and browser by signing off after you have finished accessing your Account on the Services. We seek to protect your information (including your Account Information) to ensure that it is kept private; however, we can’t guarantee the security of any information. Unauthorized entry or use, hardware or software failure, and other factors may compromise the security of user information at any time [64].’
- Vine – ‘Your public information is broadly and instantly disseminated. For instance, your profile information and public Content are immediately delivered to a wide range of users and other services that access Vine. When you share information or Content via the Services, you should think carefully about what you are making public [65].’

As Graves points out, ambiguity was prevalent throughout the app privacy policies they studied, with disclaimers explaining that data security cannot be guaranteed in this day and age. While some of those policies additionally stated

their use of encryption, others only had vague statements about their use of security/data protection practices. Graves explains that in the former case, these disclaimers may just be making it clear to end users that data cannot be completely secure, but in the latter case, it is hard to gauge the effectiveness of the security practices in place, if any at all [11]. Through our analysis of the privacy policies of apps in our dataset, and as shown in the statements quoted from these policies, we find similar results. End users are thus left with little to no answers about the security or privacy of their data. Since past research has shown the prevalence and details of privacy policies to be slowly increasing over the past several years, we aim to determine how the policies of apps in our dataset have improved, if at all, throughout the course of our study.

4.3.1 Privacy policy revisions since testing

The *Privacy Policy Revisions* component of our analysis framework entails manually finding the differences between the privacy policies in place during testing and the updated policies in place almost a year later, in February 2017. This analysis involved the use of several tools including `diff` and `grep` for data parsing and the WayBack Machine for determining and verifying when policies changed [46,57,66]. The following updates between these versions can be seen below per app:

- Facebook & Messenger – Routine maintenance. They updated the policy to adhere to the EU/Swiss framework for privacy, and explained how to contact them for questions and concerns [58,67].
- Imgur – Explained that the messaging platform is not intended to be secure, and messages are never completely hidden from public view [59,68].
- Pinterest – Removed statements explaining differences between PII and non-PII. Added ‘wholly-owned subsidiaries and affiliates’ to the parties they share the information described in the policy. Added ‘affiliates’ to the list of parties ‘[they] may get information about you and your activity off Pinterest’ from [60,69].
- Textfree – Removed the statement that Textfree complies with the COPPA. Now, ‘... as part of a request for information about Pinger products or services, they can share your provided phone number with 3rd parties (to be used to contact you/provide ads to you about Pinger products/services)’. No longer respond to ‘Do Not Track’ browser features [61,70].
- Tinder – Overall more disclosure, but now shares more PII with third parties. Now collects Personal Information and Sensitive Data: two new terms they define and use throughout the policy. Previously stated that they collect personal information such as name, email address, and other information that does not identify you. Now additionally use technology similar to cookies. Used to only provide anonymized personal information

to third parties, but now it is 'masked and obscured'. One can now opt out of cookies to third parties/ad companies. They share data with Match Group Businesses (other dating services) and updated the list of companies in this group. Now share aggregated, non-personal information, or personal information in hashed, non-human readable form for advertising and marketing purposes. Now share users' geolocation information in de-identified form with Match Group companies and third parties. They now state, 'To opt out of the sharing of your geolocation information, please discontinue use of the Tinder app.' Now explain that they transfer data to certain facilities in other countries to be processed and these locations may have different (more lenient) privacy laws. Now explain they have '... developed data practices designed to assure information is appropriately protected but we cannot always know where personal information may be accessed or processed ... While our primary data centers are in the United States, we may transfer personal information or other information to our offices outside of the United States. In addition, we may employ other companies and individuals to perform functions on our behalf. If we disclose personal information to a third party or to our employees outside of the United States, we will seek assurances that any information we may provide to them is safeguarded adequately and in accordance with this Privacy Policy and the requirements of applicable privacy laws [62,71].'

- Twitter – Removed mentions of age restrictions. Now appears to be collaborating more with ad and analytics companies, as seen in changes discussed below. Performed routine maintenance, updating terminology to more clearly express their actions. Now using more information from ad/analytics companies such as '... demographic or interest data and content viewed or actions taken on a website or app.' Added that '[you] consent to the collection, transfer, storage, disclosure, and use of your information as described in this Privacy Policy ... this includes any information you choose to provide that is deemed sensitive under applicable law.' Added a contact form for questions/comments about the policy instead of just emailing them. Seems that they improved their privacy settings and now allow users to control more. Added that the things one posts publicly is used by market research firms that analyze the information for trends and insights; before they stated, 'Twitter uses Log Data to provide, understand, and improve our Services', and now only states 'We use Log Data to make inferences.' But, in several other parts of the policy, they discuss their use of service providers '... to help provide our Services ... and to help us understand and improve the use of our Services'. Therefore, one conclusion drawn from this could be that they are moving toward having these service providers do most of the analytics work. Removed the section about 'Our Policy Toward Children' and the statement 'Our Services are not directed to persons under 13'. Did add

several sentences about ‘Privacy Shield principles’ and ‘the US-Swiss Safe Harbor Framework and principles’, but these do not appear to discuss or involve age restrictions. Changed terminology from ‘Non-Private’ to ‘Public’. Changed and added tools/capabilities in the app revisions, and updated their terminology accordingly [63,72].

- Tumblr – Same privacy policy in place as before [64].
- Vine – Routine maintenance performed [65,73].

4.4 Notable HTTP data leaks and observations

The following sections highlight prominent findings in the app traffic. More in-depth information can be found in [Table 4](#) per app.

4.4.1 Textfree

Through our analysis, we found that Textfree is sending the majority of message traffic in the clear. We collected almost all of the ‘text messages’ (including emojis) in plaintext sent to/from the app, as well as pictures sent from the app, the phone IDs/numbers (including our personal cell phone number), the contacts saved in the app, the age and gender of the Textfree user logged into the app, the Wi-Fi service set identifier (SSID) being used for the network connection, and all of the Wi-Fi SSIDs within range of the phone being broadcasted. As discussed earlier, these lists of SSIDs can be used for Wi-Fi mapping, which aims to determine the approximate location of the device even when GPS and other location services are turned off [74]. Only a few ‘text messages’ across all of the tests were not seen in the traffic collected. Pictures sent in the app were sent unencrypted and were reconstructed during traffic analysis, whereas the pictures received in the app from an outside source were hosted on an HTTP server and could be pulled down using the URL found in the unencrypted message traffic.

During the Textfree rooted phone experiment, we were able to thoroughly test the app. We had text messaging conversations through the app with a personal cell phone (Verizon carrier), and sent/received text and picture messages. On the rooted phone we took a picture and sent it to our personal cell phone, then from our personal cell phone took a picture and sent that back. In both cases, we were able to obtain the images transmitted/referenced in the HTTP traffic. An example of this in a later experiment can be seen in [Figure 6](#) test.

During our analysis we also found Textfree at times using the Session Initiation Protocol (SIP) over UDP on port 4544, a nonstandard port. According to the SIP RFC, ‘The default port value is transport and scheme dependent. The default is 5060 for sip: using UDP, TCP, or SCTP. The default is 5061 for sip: using TLS over TCP and sips: over TCP [75].’ It is unclear why this protocol was sometimes being used on a nonstandard port, but it may be an

attempt at security. We also observed some SIP traffic on the standard port of 5060, transmitted over UDP, but only during our tests on the rooted phone. We conducted a high-level analysis of this traffic, leveraging Dshell's `sip` and `rtp` decoders, as well as Wireshark [50,76]. From our analysis we concluded that Textfree registered the phone number through SIP, and continued to keep the connection alive, in which we presume would be to maintain a prepared state in case a phone call would be made. In general, this was seen through the pattern of a request to 'REGISTER', followed by multiple pairs of status of 'OK', request of 'OPTIONS'.

As a quick, additional experiment, we attempted to obtain as much traffic as possible through a MITM attack using ARP cache poisoning, targeting the rooted phone. We aimed to determine how closely our experimental results in the emulator and rooted phone tests resembled those in a realistic attack scenario. We leveraged `arp spoof` for this, a tool provided in a package of network analytic tools called `dsniff` [77]. According to the man (manual) page, 'arp spoof redirects packets from a target host (or all hosts) on the LAN intended for another host on the LAN by forging ARP replies [77].' In this scenario, the rooted phone was on Wi-Fi, connected to a router we set up. A laptop on the same network was used to determine the IP addresses of the router and rooted phone through `nmap`, an open-source network discovery tool [78]. Next, using `arp spoof`, this laptop convinced the rooted phone that the laptop was the router. Thus, the phone was tricked into passing its traffic to the laptop instead of the router, which the laptop collected and forwarded to the router, such that the traffic would still be sent to the original destination. In this real-world attack scenario, all of the unencrypted HTTP traffic sent by the phone would be seen by the adversary. Almost all of the data mentioned previously were also collected in this scenario, except the multimedia pictures were only found as URLs in the traffic (the actual images themselves were not) and the SIP traffic differed. In this case, we found several instances of GET (NO RESPONSE) in the traffic. One example was 'GET (NO RESPONSE) p.pinger.com/mms/716/IRDQTu.jpg'. Another difference was that all of the SIP traffic was on port 5060, which differed from the other Textfree experiments conducted, which additionally contained SIP traffic on the nonstandard port 4544. The SIP traffic also seemed to be slightly different, with several consecutive requests to 'REGISTER' the phone number, followed by multiple consecutive 'OK' statuses. Overall, we concluded that this MITM attack scenario yielded similar results and supported those in our emulator and rooted phone Textfree experiments.

Since Textfree yielded the most unencrypted data leaks, we retested the application about 10 months after the first test to see how the app's data handling practices had changed and test the bidirectional phone call functionality. Similar to the emulator experiments, we found SIP traffic on the nonstandard port 4544 and the standard port 5060. In this case, the phone calls placed occurred over the standard



Figure 6. (Left) Picture sent to personal phone number in Textfree app unencrypted during Feb 2017 test and (right) picture received in Textfree app from personal phone number and pulled down via HTTP from host server during Feb 2017.

SIP port 5060. We obtained detailed call logs in the SIP, Real-Time Transport Protocol (RTP), and HTTP traffic of the calls made to and from the app and our personal phone. These logs included the detailed handshakes, phone numbers, Call-IDs, nonces, proxy authentication, encodings, and codecs. We did not recover the audio from the phone calls, but believe that it may be possible with additional analysis. Using Wireshark we reconstructed the Voice over Internet Protocol (VoIP) flow graphs of the three phone calls made over SIP, seen in Figure 7 [76]. The first phone call made was from the rooted phone (in the Textfree app) to a personal phone, and it failed due to not having any calling credits available. The second call made was successful and was from our personal phone to the rooted phone, which didn't require calling credits. Finally, after watching an advertisement and earning calling credits, we made the third phone call from the rooted phone to our personal phone. The two successful calls were each a few seconds only. The SIP traffic from this experiment yielded similar results with the addition of requests and statuses relating to the phone calls, including 'INVITE', 'TRYING', 'Not enough credit for call', 'RINGING', 'SESSION PROGRESS', 'ACK', and 'BYE'. From this analysis, we conclude that the general SIP traffic seen on the nonstandard port (4544) is similar to that seen on the standard port (5060), but in our experiment, the phone calls made all used the standard port (5060). Moving forward, we continued to analyze the traffic during this new experiment, and found that the URL's for the pictures had changed from [`picUrl':http://p.pinger.com/mms/427/asQLNt.jpg`] to [`picUrl':https://picmsg.co/mms/486/ltvVUi.jpg`]. Thus, the servers used for storage were no longer part of their main website and used HTTPS instead of HTTP. At first glance, this seemed to prevent the leakage of these images, but, these HTTPS links were still backward compatible and could be accessed via HTTP by simply removing the 's'.

4.4.2 *Tinder*

Tinder used HTTP for almost all of its traffic. Analyzing this unencrypted traffic allowed us to see all of the pictures of the Tinder user and the users they communicated with. The communication messages themselves were encrypted, but the pictures and location (city/district name and zip code) of the users were

not. Corresponding with each image seen in the traffic, we found an HTTP ETag and Amazon Web Services request IDs. HTTP ETags have been shown to be used by some companies as an alternative to cookies and for tracking purposes [79]. Along with the location data was another number sent unencrypted, which may be the user’s age, but although the majority of these numbers were in a normal range, we found outliers as low as 11 and as high as 287.

Furthermore, Tinder’s privacy policy states that users must be 18 or over [62]. Although some data are transferred encrypted, it may be possible to gather more information through inference attacks. Analyzing the HTTP traffic, we broke down the URLs into the base URL, the user profile, the image size, and the picture name as they are stored on Tinder servers. We could then access these pictures stored on their servers via these URLs. These pictures were still accessible via these URLs around seven months after traffic collection, but could not be accessed when attempted again over a year after the initial test. Thus, we conclude that these images are stored and can be accessed for a significant amount of time via these URLs. By splitting up the traffic by user, we could find instances of the same user in the traffic at different times of the app’s usage. With a little understanding of the normal app usage, we know that if the user profile is repeated several times in a row with different pictures, then the Tinder user viewing the profile is browsing the other user’s pictures. Next, we know that it is supposedly unlikely to have a user profile seen again after ‘rejecting’ them, and we know that users cannot message each other or view profiles again until they have both ‘accepted’ each other. Using this information, we can infer that two users ‘matched’ (by

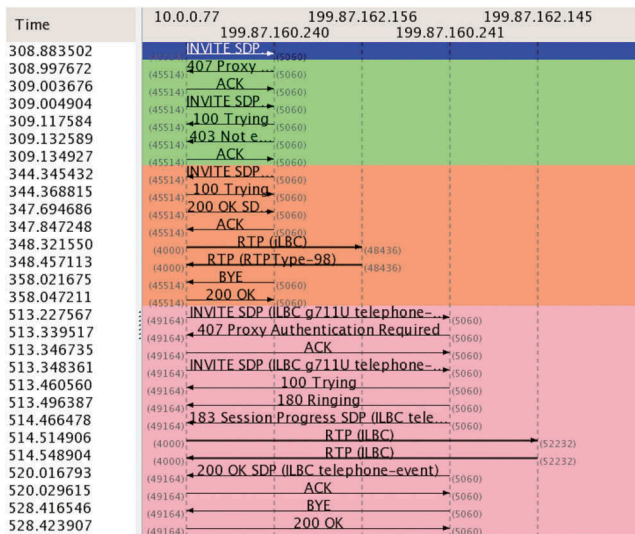


Figure 7. Wireshark’s VoIP flow analysis graph of bidirectional Textfree phone calls made over SIP in the 2/16/17 rooted phone test.

swiping right on each other) if we see a user profile repeated in the traffic, where the two points differ in a significant amount of time (such as a few minutes). We can then see if the user views the other's profile pictures again and deduce that the more times we see this, the longer their conversation. Thus, the messages themselves are encrypted, but it may be possible to infer information about the users, their usage patterns, and their personal information.

4.4.3 *Imgur*

Imgur sent a little over half of its traffic unencrypted. This entailed 197 images (including duplicates of various sizes) being transmitted in the clear during our testing in the Android emulator. These images potentially leak user interests through inference attacks, and the privacy policy does not disclose the transmission of this content unencrypted [59]. This information describes and profiles a user through their choices to view certain content. This information could be valuable to ad companies for targeted advertising or adversaries aiming to learn more about a user for phishing attacks. Additionally, HTTP cookies were found in the traffic, which was disclosed in the policy. HTTP ETags were also found and were not specifically mentioned in the privacy policy, but may be considered part of the 'other anonymous tracking information' mentioned in their statement, 'We may use cookies, web beacons, or other anonymous tracking information to improve our server's interaction with your computer [59]'. As mentioned previously, HTTP ETags have been used by some companies for tracking purposes [79]. Additionally, an mp4 video file and CSS file were found in the unencrypted traffic.

4.4.4 *Pinterest*

During our Pinterest tests in both the Android emulator (>84% HTTP) and the rooted Google Nexus 6P (>75% HTTP) environments, our analysis allowed us to see and track nearly everything the user was doing through the content they viewed. Unencrypted content in the traffic included articles and pictures users viewed in the app, including Pinterest users' profile pictures. These profile picture filenames included a Pinterest user's username, which depending on what they chose, could be their real name. Additionally, unique identifiers, device information (OS/make/model/build), the HTTP user-agent field (also including device information), website cookies, and some JavaScript and miscellaneous HTML code was sent in the clear. Similar to other apps tested, this content leaks user interests through inference. Several pieces of this content were used for advertising purposes, as seen detailed in [Table 4](#).

4.4.5 Vine

The majority of Vine's traffic was unencrypted (HTTP), which included every packet sent between the Vine app and Twitter servers, the parent company of Vine. Videos and images (video thumbnails/previews) viewed, as well as HTTP ETags and some cloud server requests, were found in the unencrypted traffic. As discussed previously, HTTP ETags have been used for tracking purposes by some companies [79]. Similar to Imgur and Pinterest, the transmission of this content unencrypted was not disclosed in the privacy policy and results in the potential leakage of user interests, which leads to multiple security/privacy concerns [65].

4.5 End of study follow-up experiments and observations

At the end of our study, we re-ran the *Application Analysis* module on the three apps that exhibited the most notable findings and were shown to use the least amount of encryption in their data transmission practices: Tinder, Textfree, and Pinterest. We downloaded the latest versions of these apps from the Google Play Store and repeated our experiments in the rooted phone environment on the Google Nexus 6P [30,34,36,37,42]. The OS version of the rooted phone and the environment remained the same as in the previous experiments. Additionally, we kept the same methodology of installing, testing, and removing each app at a time. It is important to note that these experiments were also conducted on different networks, which may or may not use different ISPs and/or CDNs, which may enforce different data transmission rules. For example, we do not know if different networks, CDNs, and ISPs force encryption on their networks. Additionally, as in the previous experiments, we were unable to control the amount of ads served during the use of the app, which could have a major effect on the breakdown of traffic. Overall, in the experiments conducted at the end of our study, we noted an increase in HTTPS and a decrease in HTTP traffic. Similar to Table 2, we provide the traffic loads seen per protocol of these final experiments in Table 5, but now sort by HTTP usage per app. Additionally, the data-sorted graphs generated during each of these follow-up experiments can be found in Appendix A.

4.5.1 Textfree

Two additional experiments, A and B, were conducted with two different versions of the app. We aimed to replicate the actions taken and messages sent during our original experiments to minimize variability. In experiment A, we observed a large increase of encryption used for data transmissions. Initial analysis showed SMS/MMS messages or URLs to the images sent/received were no longer seen in the cleartext traffic. In experiment B, we installed the latest version of the app, which was released a few weeks after the version used in experiment A, but conducted

this test on a different network. Experiment B showed another drastic increase in HTTPS traffic used, now 99%, but the cleartext content transmitted appeared to be similar to what was seen in experiment A, minus ad videos and less ad related code files. Both experiments A and B showed significantly less files being transmitted in the clear than before, with B having less than a third as much as A, yet we still found similar types of sensitive data transmitted in the clear. During initial analysis of both, despite the increased encryption, we again observed the user's age, gender, location data (city and GPS coordinates beyond city granularity), the same unique ID as in our original experiment, and device information (OS/make/model/build/language/browser version). Experiments A and B took place at different locations and on different networks, but both experiments had the same GPS coordinates, which in both cases were ~1000 ft away from the testing location. IP-based coordinates were also found but were less accurate. Additionally, we found SIP traffic of our phone calls in cleartext with very similar data as seen previously, continued to be used on the nonstandard port 4544, and included the plaintext phone number of the personal cell phone (Verizon carrier) used in our bidirectional conversations, and our call logs.

In experiment B, we noted that more ads seemed prevalent, ad videos were not transmitted unencrypted, and HTTPS connections with Akamai accounted for over 75% of the data transmitted. It is unclear if mainly data transmission practices changed between the versions used in experiments A and B, if the different networks had ads served with varying levels of encryption, or if there was another cause to the noted HTTPS increase.

4.5.2 Tinder

One additional experiment was conducted to attempt to use the majority of the app's features, as in the original experiment. We note a slight increase in HTTPS and a slight decrease in HTTP traffic. As seen previously, we still observed the Tinder profile pictures being sent unencrypted. Again, the picture URLs could be broken down into the base URL, the user profile, the image size, and the picture name as they are stored on Tinder servers, and thus we believe further data could be inferred again as before. Initial analysis showed that the location (city/

Table 5. End of study experiments: Breakdown of application traffic using HTTP, HTTPS, and DNS.

App Name	HTTP	HTTPS	DNS	Other	Total KB
Tinder ^{*^}	85.9%	14.0%	0.2%	<0.1%	17,855.3 KB
Pinterest ^{*^}	24.2%	75.7%	0.1%	<0.1%	39,521.0 KB
Pinterest ^{*^♦}	11.9%	88.0%	0.1%	<0.1%	35,478.4 KB
Textfree ^{*^}	7.6%	92.3%	0.1%	<0.1%	45,531.5 KB
Textfree ^{*^}	16.15%	82.8%	0.2%	0.5%	58,933.5 KB
Textfree ^{*♦▽}	0.5%	99.2%	0.1%	0.2%	139,657.3 KB

* Application traffic collected on rooted Google Nexus 6P.

^ Application retested at end of study with Nov 2017 application version.

♦ Application retested at end of study, now on different network.

▽ Application retested at end of study with Dec 2017 application version.

district name and zip code) information and the number which may have been 'age', are now encrypted. We note that over 85% of the data transmitted were over HTTP connections with the CDN Akamai. Comparing the Akamai CDN connections in all of the top 10 data transmitted connections throughout all of our experiments, we note mostly HTTP connections other than the Textfree 2-16-17 and 12-7-17 tests. Therefore, since Akamai's encrypted connections vary by app, and we still see such a prevalent use of HTTP in the last Tinder experiment, it seems that the encryption use may not be dictated by the CDN, but maybe more by the application companies or ISPs.

4.5.3 Pinterest

Three additional experiments, C, D, and E were conducted and analyzed. Experiments C and D occurred one after the other with the same version, but unlike all other Pinterest experiments, C did not include direct messages sent in the app during the test. Experiment E was then conducted a few weeks later, using the same app version as C and D, but on a different network. In all three, we similarly note an increase in HTTPS and decrease in HTTP traffic, and fewer files transmitted in the clear, but similar types of data to our original experiments are still transmitted unencrypted: pictures (including profile pictures) and HTTP user-agent field with device information (OS/make/model/build). In C and E we also see ad related code files, but in D we only see pictures sent in the clear. In E, ran on the same network as Textfree's follow-up experiment B, HTTPS accounted for 92.3% of the traffic, and the top data-transferred connection was with Akamai over HTTPS. We also note that even though C and D were conducted one after the other, there is still a 12.3% difference in the HTTPS traffic observed between the two experiments. This goes to show that despite aiming to replicate types of actions taken across experiments, there are still variables present, such as the amount of ads served, and potentially where those ads are served from.

4.6 Data sharing module

4.6.1 Business relationships

The business world is full of relationships, and the social media market is no different. Within our dataset of apps, we determined a few business connections, as well as some others between social media companies worth noting. For instance, Twitter, Inc., owns Vine Labs, Inc., and thus it follows that any personal data from Vine are collected by the parent company and may be used for other apps they own. Though we did not investigate Instagram, it is worth noting that Facebook, Inc., owns Instagram and according to an article from Instagram's Help Center, 'We want to show you ads from businesses that are interesting and relevant to you, and to do that, we use information about what you do on Instagram and Facebook (our parent company) and on third-party sites and apps you use' [80]. This implies that they are also obtaining

information from other sources, ‘third-party sites and apps,’ to better understand and predict a user’s actions. In addition to these relationships, we determined that Textfree is owned by Pinger, Inc., and Tumblr, Inc., is owned by Yahoo!, who is now owned by Verizon Inc. [81]. Thus, when a user believes they are only sharing their information with one of these apps/organizations, it is most likely not the case. Profiles of users maintained by these companies may be built from the aggregate information provided by all of the apps in a given business relationship.

4.6.2 Prominent advertising/analytics libraries

To analyze the APK files and obtain more in-depth information about the ad and analytics libraries they used, we needed to retarget and decompile them. To accomplish this, we made use of a number of open-source tools. First we used *Dare*, a tool that retargets APK files into .class files by retargeting Dalvik bytecode to Java bytecode [55]. After obtaining the Java bytecode, we used *Soot*, which decompiles Java bytecode into human readable .java files [56]. We could then obtain the libraries used by each app. We created a short script that performed a depth-first-search through the java directories and outputted a list of libraries. Finally, we researched each of the libraries and noted several prominent ones that past studies have investigated. These ad and analytics libraries used by the apps in our dataset can be seen in [Table 6](#).

4.6.2.1 Bolts. Bolts was developed by Parse and Facebook, Inc., originally for in-house use, but was later released as open source. Bolts does not require a developer account from either company. It can be used to enable a user to transition from one app to another. This is done through the creation of links, which contain all of the necessary data for this transition.

4.6.2.2 Flurry. Flurry is an ad and analytics service owned by Yahoo! and, as of 2017, it appears that Flurry is now owned and operated by Verizon, Inc. [82]. Grace et al. studied mobile ad networks and found that Flurry probed the phone to see what permissions an app has before acting. They also found Flurry to collect location data [5]. Flurry’s documentation states that one can, ‘set up advanced analysis of complex events, with metrics, segments and funnels to better track your users’ habits and performance’ [83].

4.6.2.3 Google Ads/Admob. Google, Inc., who owns AdMob, allows developers to analyze, monetize, and promote their apps. Grace et al. found that Google and AdMob probed permissions and collected location data [5].

4.6.2.4 Fabric. Fabric, created by Twitter, Inc., is a mobile development platform that includes crash reporting, user-tracking, user verification, tweet

embedding, monetization, AWS cloud syncing, payments, optimization, user experience (UX) analysis, maps, real-time communication, and game analytics.

These are just a prominent few of the many ad/analytics libraries used in the apps we researched. These libraries, owned by large and prominent companies, are seen throughout the apps we tested. Thus from these business relationships, it follows that a Tumblr user could have their information shared with Facebook, Yahoo!, Verizon, Google, and Twitter without even linking accounts across these platforms.

5. Conclusions

Our findings show that the trend of Android mobile apps collecting and sharing sensitive data and PII, seen often in the first few years of the OS’s existence, is still prevalent for popular social media apps. The practice of transmitting these data unencrypted, as well as to third parties and ad networks can still be observed. As the apps in our dataset were all popular social media apps, it is likely that a given end user uses or has used one of these, and thus leaked information directly and/or inadvertently to the network and hops taken by the packets, and to the companies in business with the app company. In the future, researchers should continue to investigate which apps across all domains are sending data in the clear. The inadvertent leakage of data to the networks traversed should be studied and attempted to be quantified, such as by tracing the network path of an image transmitted in an app to the destination server.

The lack of encryption technologies used by app companies could be investigated to better understand its prevalence and determine how to move toward the use of full encryption. Future work could also compare the direct/inadvertent exposure of sensitive data by free apps to that by paid versions. Apps that are transmitting over IPv6 could also be studied using our analysis framework. Future work should also investigate the Textfree app further and determine whether the audio transmitted (SIP/RTP) from the bidirectional phone calls could be recovered with additional analysis. Additionally, the use of SIP on a nonstandard port (4544) should be analyzed further, as well as further comparison of this traffic to that seen on the standard port (5060) [75].

The paid features of the Textfree app, including those which can be redeemed by watching advertisements, could be investigated further as well. The payment information and logs of these transactions may not be transmitted securely, and these data transmission practices may be different when these services are paid

Table 6. Prominent advertising/analytics libraries used by each app.

	Facebook	Imgur	Messenger	Pinterest	Textfree	Tinder	Tumblr	Twitter	Vine
Bolts					X		X		
Flurry					X		X		X
Google Ads/AdMob		X			X	X	X	X	X
Fabric						X	X	X	X

for with currency vs. paid for by watching advertisements [35]. Because passive traffic sniffing can be used to build an attack profile of a target, future work with app research could attempt to combine unencrypted data collected from a variety of apps that a typical user might have on their phone, to develop possible profiles of a target. When multiple apps and different combinations of them are involved, it may then be possible to infer additional information.

Our analysis results also show the efficacy of our framework and testbed environments for testing Android apps. Future work could leverage these and continue to investigate future versions of these social media apps, as well as other mobile apps in general. We found that for most apps, the emulator was a viable option for collecting traffic, as it was designed for developers to test apps before releasing them on the Google Play store. For this research, we used Android Studio v1.5, but with the more recent Android Studio v2.0, performance is much better and subsequent investigation is likely to have better results that more closely resemble an actual phone. The emulator provides an isolated environment that is quick and easy to instantiate for each run of the experiment, enabling fast installation and testing of one app on the device at a time. Physical phones may not be ideal for testing if apps running in the background are tainting the traffic. Additionally, running experiments on a variety of Android OSs is trivial when using the Android emulator, yet can be difficult when using physical devices. However, it is important to recall that while using Android Studio v1.5, a few of the apps we tested did not run properly in the emulator. For the vast majority of apps, the emulator results resemble those seen on the physical device. Additionally, we find very similar results during our more realistic MITM attack scenario, thus reinforcing the accuracy of the data collected in these testbed environments. In the future, we would focus even more on ensuring traffic is not being tainted from background apps on the rooted device.

Android social media apps continued to exhibit the trend of the collection and direct/inadvertent exposure of privacy sensitive information, as well as a lack of proper disclosure in their privacy policies. In our main experiments of the nine social media apps we studied, only Facebook, Messenger, Twitter, and Tumblr encrypted the majority of their traffic, which greatly reduced the sensitive data leakage. We later observed in our brief follow-up experiments, that several more apps have begun to encrypt the majority of their traffic, but that data leakage is still significantly prevalent. For example, in the 12/7/2017 Textfree experiment with over 99% of the traffic over HTTPS, we no longer found SMS/MMS messages in the clear, but still observed a significant amount of sensitive information leaked in the 0.5% of traffic transmitted over HTTP such as our GPS coordinates (~1,000 ft away), city/state/country, age, gender, device information, and more. Furthermore, in this Textfree experiment we still obtained the plaintext SIP traffic, which included our real cell phone number and call logs. If more apps moved toward encrypting the majority all of their data transmissions, this inadvertent leakage of sensitive data and PII could be avoided. Direct leakages of sensitive data would still exist to the

app companies and the parties they share data with, but the ISPs, supercomputing centers, network administrators, and router owners who inhabit and transmit data at the hops between phones and app company servers would see significantly less interpretable data. App privacy policies need to be more descriptive and explain to users better exactly how their data are transmitted, used, and secured. Simply stating that data are stored securely and that user privacy is protected means very little when the majority of the data are transmitted unencrypted and exposed to the network and hops along the data paths. Now that recent FCC rulings have officially been reversed, ISPs can continue to collect, sell, and share user data legally. Adding to this, certain geographical areas are only covered by certain ISPs, all of which may take advantage of user data to some extent. This can leave end users with no choice but to agree to the terms of these companies; pay extra fees for privacy, as Comcast and AT&T have publicly expressed interest in making standard; or forgo Internet services [17,18]. Ensuring user security and privacy should be a standard across mobile apps and should be perfected by social media apps, as transmitting and sharing sensitive data and PII is in their nature.

Acknowledgments

We would like to thank Meghan Riegel and Charles Sestito for their work and help with this research: collecting app traffic, developing scripts to automate the collection process, and investigating business relationships and ad/analytics libraries. We'd also like to thank Berkey Celik for his suggestions. Finally we thank Ryan Sheatsley for retargeting and decompiling the Android Package Kits (APKs) using *Dare* and *Soot*, and developing scripts to obtain the ad and analytics libraries as well as the namespaces used by each app [55,56].

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research was sponsored by the Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation here on. This work was also supported by the National Science Foundation [CNS-1228700 and CNS-1064900]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Notes on contributors

Daniel E. Krych is a computer scientist in the Network Security Branch at the Army Research Laboratory. He holds a B.S. (2015) in Computer Engineering and a M.S. (2017) in Computer Science and Engineering from the Pennsylvania State University. His research has been largely in the areas of network security and privacy, network forensics, intrusion detection, and cryptography.

Patrick McDaniel is the William L. Weiss Professor of Information and Communications Technology and Director of the Institute for Networking and Security Research in the School of Electrical Engineering and Computer Science at the Pennsylvania State University. Professor McDaniel is also a Fellow of the IEEE, ACM and AAAS and the director of the NSF Frontier Center for Trustworthy Machine Learning. He also served as the program manager and lead scientist for the Army Research Laboratory's Cyber-Security Collaborative Research Alliance from 2013 to 2018. Patrick's research focuses on a wide range of topics in computer and network security and technical public policy. Prior to joining Penn State in 2004, he was a senior research staff member at AT&T Labs-Research.

References

- [1] Thurm S, Kane YI. Your apps are watching you. *Wall Street Journal* 2010;17(2010):1.
- [2] The Wall Street Journal. What they know - mobile. *Wall Street Journal* 2010; Retrieved from <http://blogs.wsj.com/wtk-mobile/>
- [3] Enck W, Ocateau D, McDaniel P, et al. 2011. A study of android application security. *USENIX secur symp*, 2 (2011), 2.
- [4] Fulton E. Cellular privacy: A forensic analysis of android network traffic. In: Presented at DEF CON 19 in Las Vegas. Nevada; 2011. DEF CON Communications, Inc.; Retrieved from: <https://media.defcon.org/DEF%20CON%2019/DEF%20CON%2019%20slides/>
- [5] Grace MC, Zhou W, Jiang X, et al. Unsafe exposure analysis of mobile in-app advertisements. In: Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. 2012; Tucson, AZ; 101–112.
- [6] Federal Trade Commission. Mobile apps for kids: current privacy disclosures are disappointing. *FTC Staff Report* February 2012.
- [7] Federal Trade Commission. Mobile apps for kids: disclosures still not making the grade. *FTC Staff Report*, 2012 December.
- [8] Zang J, Dummit K, Graves J, et al. Who knows what about me? A survey of behind the scenes personal data sharing to third parties by mobile apps. Oct 2015. <http://techscience.org/a/2015103001/>
- [9] Fung B. Trump has signed repeal of the FCC privacy rules. here's what happens next. 2017 Apr. <https://www.washingtonpost.com/news/the-switch/wp/2017/04/04/trump-has-signed-repeal-of-the-fcc-privacy-rules-heres-what-happens-next/>
- [10] Cohen K, Yeung C. Sep 2015. Kids' apps disclosures revisited. Federal Trade Commission Protecting America's Consumers. <https://www.ftc.gov/news-events/blogs/business-blog/2015/09/kids-apps-disclosures-revisited>.
- [11] Graves J. An exploratory study of mobile application privacy policies. 2015 Oct. <http://techscience.org/a/2015103002/>
- [12] Yu L, Luo X, Liu X, et al. 2016. Can we trust the privacy policies of android apps? In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN); Toulouse, France; 538–549.

- [13] Zimmeck S, Wang Z, Zou L, et al. 2016. Automated analysis of privacy requirements for mobile apps. In: 2016 AAAI Fall Symposium Series; Arlington, VA.
- [14] Slavin R, Wang X, Hosseini MB, et al. 2016. Toward a framework for detecting privacy policy violations in android application code. In: Proceedings of the 38th International Conference on Software Engineering; Austin, TX, USA. 25–36.
- [15] Balebako R, Marsh A, Lin J, et al. 2014. The privacy and security behaviors of smartphone app developers. In: Proceedings of Workshop on Usable Security (USEC); San Diego, CA.
- [16] Cranor L, Langheinrich M, Marchiori M, et al. The platform for privacy preferences 1.0 (p3p1.0) specification. W3C Recomm. 16 (2002).
- [17] Bode K. Comcast says it wants to charge broadband users more for privacy. 2016 Aug. <https://www.dsreports.com/shownews/Comcast-Says-It-Wants-to-Charge-Broadband-Users-More-For-Privacy-137567>.
- [18] Bode K. AT&T says it may soon charge you extra for privacy. 2017 Jun. <https://www.dsreports.com/shownews/ATT-Says-It-May-Soon-Charge-You-Extra-For-Privacy-139840>
- [19] Hart K. FCC adopts privacy rules to give broadband consumers increased choice, transparency and security for their personal data. 2016 Oct
- [20] Reilly K. Donald Trump signs bill repealing internet privacy rules. 2017. <http://time.com/4724128/donald-trump-internet-history-isp-privacy-browser-history/>.
- [21] Federal Trade Commission. Privacy online: a report to Congress. 1998.
- [22] Ashley P, Hada S, Karjoth G, et al. 2002. E-p3p privacy policies and privacy authorization. In Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society; Washington, DC; 103–109.
- [23] Federal Trade Commission. Mobile privacy disclosures: building trust through transparency. *FTC Staff Report February*. 2013
- [24] Shore J, Steinman J. Did you really agree to that? the evolution of facebook’s privacy policy. 2015 Aug. <http://techscience.org/a/2015081102/>
- [25] Falaki H, Lymberopoulos D, Mahajan R, et al. 2010. A first look at traffic on smartphones. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement; Melbourne, Australia; 281–287.
- [26] Enck W, Gilbert P, Chun B, et al. Oct 2010. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation; Vancouver, BC.
- [27] Arzt S, Rasthofer S, Fritz C, et al. Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices*. 2014;49(6):259–269.
- [28] Walnycky D, Baggili I, Marrington A, et al. Network and device forensic analysis of android social-messaging applications. *Dig Inv*. 2015;14(2015):S77–S84.
- [29] Tinder. Version 4.4.4. 2015 Dec. <https://play.google.com/store/apps/details?id=com.tinder>
- [30] Tinder, Version 8.1.1. 2017 Nov. <https://play.google.com/store/apps/details?id=com.tinder>
- [31] Messenger - text and video chat for free. Dec. Version 52.0.0.19.66. 2015. <https://play.google.com/store/apps/details?id=com.facebook.orca>
- [32] Facebook. Version 58.0.0.28.70. 2015 Dec. <https://play.google.com/store/apps/details?id=com.facebook.katana>
- [33] Pinterest. Version 5.12.1. [Online]. 2016 Mar. <https://play.google.com/store/apps/details?id=com.pinterest>
- [34] Pinterest. Version 6.43.0. [Online]. 2017 Nov. <https://play.google.com/store/apps/details?id=com.pinterest>
- [35] Textfree - free text + call. Version 5.10.1. 2016 Mar. <https://play.google.com/store/apps/details?id=com.pinger.textfree>
- [36] Textfree - free text + call. Version 8.3. 2017 Nov. <https://play.google.com/store/apps/details?id=com.pinger.textfree>

- [37] Text free - free text + call. Version 8.4. 2017 Nov. <https://play.google.com/store/apps/details?id=com.pinger.textfree>
- [38] Imgur: Awesome images & gifs. Version 2.4.3.501. [Online]. 2016 Mar. <https://play.google.com/store/apps/details?id=com.imgur.mobile>
- [39] Tumblr. Version 5.6.1.00. [Online]. Mar 2016. <https://play.google.com/store/apps/details?id=com.tumblr>
- [40] Twitter. Mar 2016. Version 5.101.0. [Online]. <https://play.google.com/store/apps/details?id=com.twitter.android>
- [41] Vine - video entertainment. Version 3. 3.13. 2015 Nov. <https://play.google.com/store/apps/details?id=co.vine.android>
- [42] Android apps on google play. <https://play.google.com/store/apps>
- [43] APK downloader. [Online]. <https://apps.evozi.com/apk-downloader/>
- [44] Appbrain. Feb 2017. Retrieved from: <http://www.appbrain.com/>.
- [45] Android SDK. 2009. <http://developer.android.com/index.html>
- [46] Internet archive: WayBack machine. <https://archive.org/web/>
- [47] Lee M. How to root nexus 6p! [win/mac/linux]. 2015 Nov. <http://nexus6proot.com/nexus-6p-root/how-to-root-nexus-6p-winmaclinux>
- [48] Supersu v2.65. <http://www.supersu.com/download>
- [49] Jacobson V, Leres C, McCanne S. Tcpdump/libpcap. <http://www.tcpdump.org>
- [50] US Army Research Laboratory. Dshell. <https://github.com/USArmyResearchLab/Dshell>
- [51] Android tcpdump downloads. <https://www.androidtcpdump.com/android-tcpdump/downloads>.
- [52] J. D. Hunter. Matplotlib: A 2d graphics environment. *Comput in sci eng.* 2007;9(3): 90–95.
- [53] IP address details. <https://ipinfo.io/>
- [54] Troche J. Nov 2013. Installing tcpdump for Android. <https://josetrochecoder.wordpress.com/2013/11/04/installing-tcpdump-for-android/>
- [55] Oceau D, McDaniel P, Jha S. Dare: Dalvik retargeting; 2012. <http://siis.cse.psu.edu/dare>
- [56] Vallée-Rai R, Co P, Gagnon E, et al. 1999. Soot-a java bytecode optimization framework. In: Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative research; Mississauga, ON: IBM Press. 13.
- [57] Gnu grep. 2014. <https://www.gnu.org/software/grep/>
- [58] Data policy. Jan. Last Revised Jan 2015. 2015. <https://www.facebook.com/policy.php>
- [59] Privacy policy. Jan. Last Revised Jan 2016. 2016. <https://imgur.com/privacy>
- [60] Privacy policy, Dec. last Revised Dec 2015. 2015. <https://about.pinterest.com/en/privacy-policy>
- [61] Privacy policy. Feb. Last Revised Feb 2016. 2016. http://pinger.com/content/company/privacy_policy.html
- [62] Tinder, Inc. privacy policy. Last Revised Mar 2015. 2015 Mar. <https://www.gotinder.com/privacy>
- [63] Twitter privacy policy. Jan. Last Revised Jan 2016. 2016. <https://twitter.com/privacy?lang=en>
- [64] Privacy policy. Last modified Jan. Last Revised Jan 27 2014. 2014 1. <https://www.tumblr.com/policy/en/privacy>
- [65] Vine privacy policy. Dec. Last Revised Dec 2014. 2014. <https://vine.co/privacy>
- [66] GNU diffutils. 2010. <https://www.gnu.org/software/diffutils/>
- [67] Data policy. Sep. Last Revised Sep 2016. 2016. <https://m.facebook.com/policy.php>
- [68] Privacy policy. Jan. Last Revised Jan 2017. 2017. <https://imgur.com/privacy>
- [69] Privacy policy (effective November 1, 2016). Last Revised Nov 2016. Nov 2016. <https://about.pinterest.com/en/privacy-policy>

- [70] Privacy policy. <http://www.pinger.com/privacy-policy/>
- [71] Tinder, Inc. privacy policy. Last Revised. [Online]. 2017 Jan. <https://www.gotinder.com/privacy>
- [72] Twitter privacy policy. Sep. Last Revised Sep 2016. 2016. <https://twitter.com/privacy?lang=en>
- [73] Vine privacy policy. Jan. Last Revised Jan 2016. 2016. <https://vine.co/privacy>
- [74] Vaughan-Nichols SJ. How google—and everyone else—gets wi-fi location data. Nov 2011. <http://www.zdnet.com/article/how-google-and-everyone-else-gets-wi-fi-location-data/>
- [75] Rosenberg J, Schulzrinne H, Camarillo G, et al. Sip: session initiation protocol [Online]. Tech. Rep; 2002.
- [76] Wireshark GC, version 1.12.1. <https://www.wireshark.org/>
- [77] Dsniff DS. <https://www.monkey.org/~dugsong/dsniff/>
- [78] Lyon G. Nmap—free security scanner for network exploration & security audits. <https://nmap.org/>
- [79] Davis M. Etags allow tracking without cookies. 2014 Oct. <https://www.futurehosting.com/blog/etags-allow-tracking-without-cookies/>
- [80] How does Instagram decide which ads to show me? <https://help.instagram.com/173081309564229>
- [81] Doctorow C. Tumblr is now owned by a phone company, so it's stopped fighting for network neutrality. Jun 2017. <https://boingboing.net/2017/06/22/corporations-arent-people.html>
- [82] Siluk S. Yahoo becomes Altaba after Verizon buyout, Mayer out altogether. Jan 2017. http://www.cio-today.com/article/index.php?story_id=03100001X6EL
- [83] Flurry analytics. <https://developer.yahoo.com/flurry/docs/analytics/>

APPENDIX

A. Application analysis graphs

The following graphs were outputted by our *Application Analysis* module and display the summaries of the app traffic collected during our experiments. The first nine graphs are output from our Case 1 experiments using the Google’s Android emulator and the next five graphs are output from our Case 2 experiments using the rooted Google Nexus 6P phone. The final six graphs are output from the follow-up experiments at the end of our study, also using the rooted Google Nexus 6P phone.

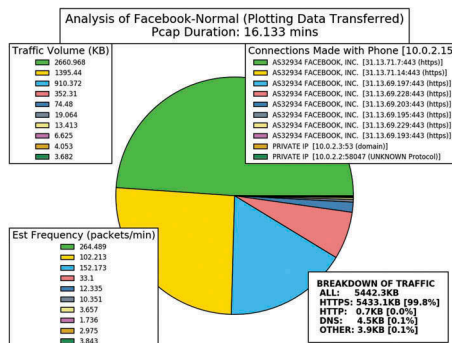


Figure A.1. Analysis of Facebook traffic – Android emulator.

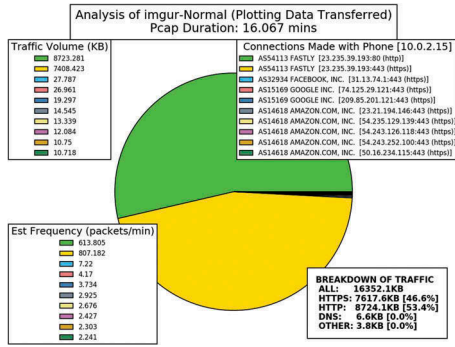


Figure A.2. Analysis of Imgur traffic – Android emulator.

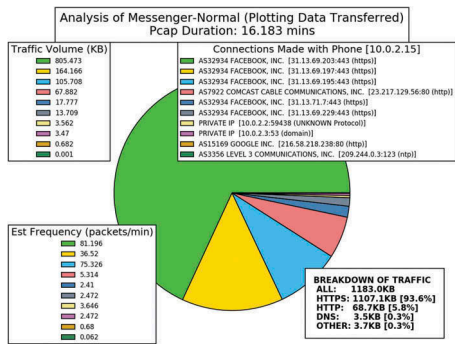


Figure A.3. Analysis of Messenger traffic – Android emulator.

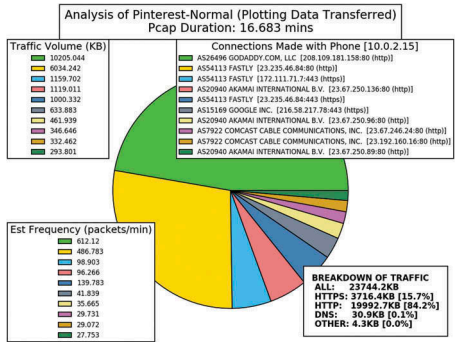


Figure A.4. Analysis of Pinterest traffic – Android emulator.

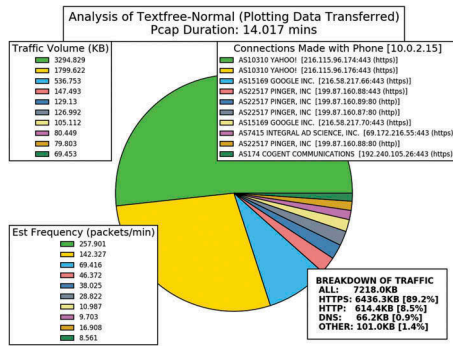


Figure A.5. Analysis of Textfree Test 1 traffic – Android emulator.

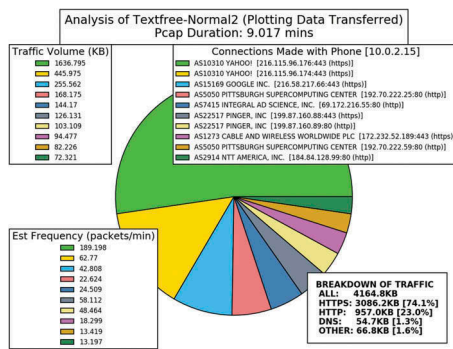


Figure A.6. Analysis of Textfree Test 2 traffic – Android emulator.

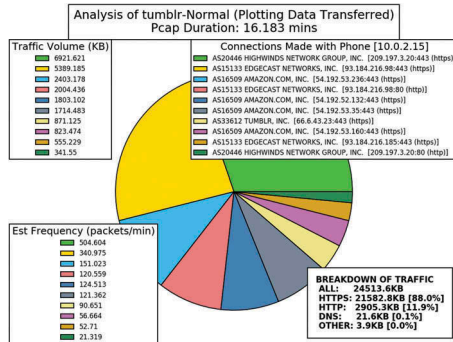


Figure A.7. Analysis of Tumblr traffic – Android emulator.

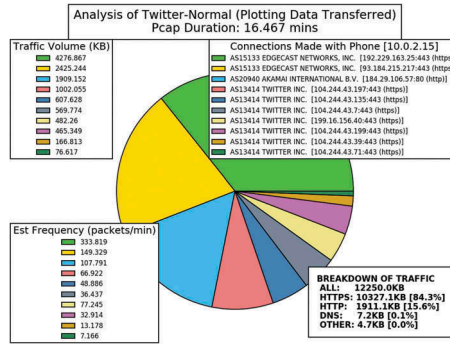


Figure A.8. Analysis of Twitter traffic – Android emulator.

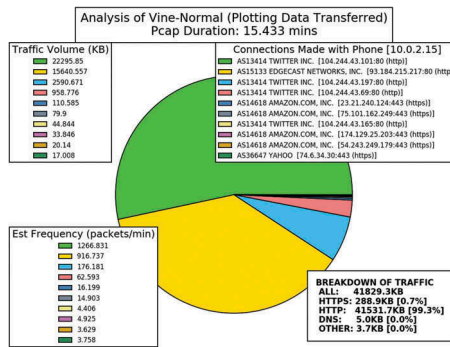


Figure A.9. Analysis of Vine traffic – Android emulator.

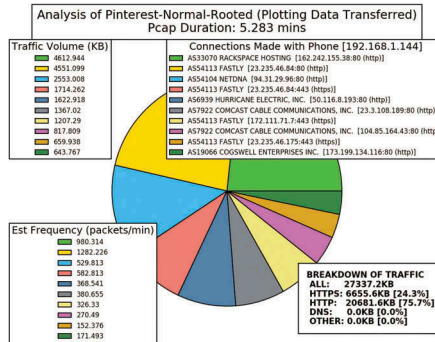


Figure A.10. Analysis of Pinterest traffic – rooted Nexus 6P.

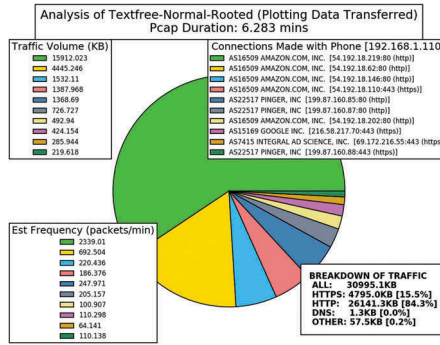


Figure A.11. Analysis of Textfree traffic – rooted Nexus 6P.

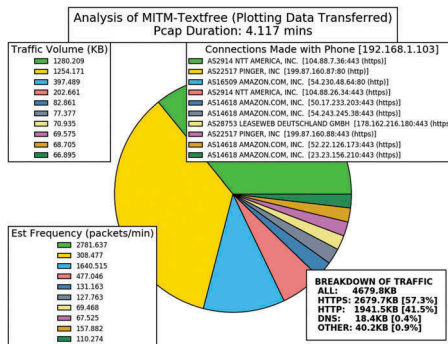


Figure A.12. Analysis of Textfree MITM (arpspoof) attack traffic – rooted Nexus 6P

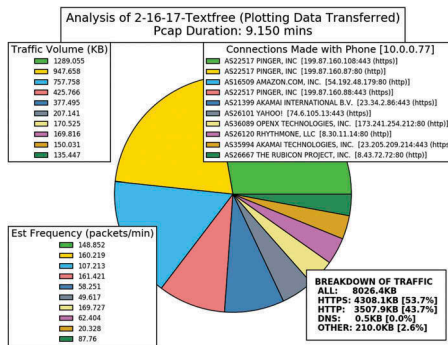


Figure A.13. Analysis of Textfree retested Feb 2017 traffic – rooted Nexus 6P.

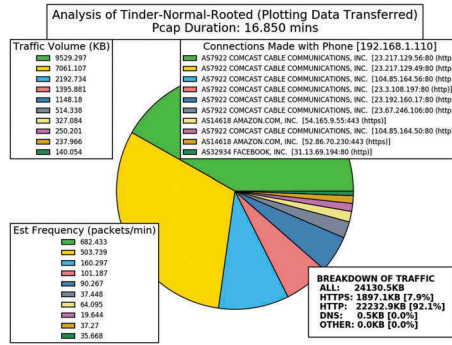


Figure A.14. Analysis of Tinder traffic – rooted Nexus 6P.

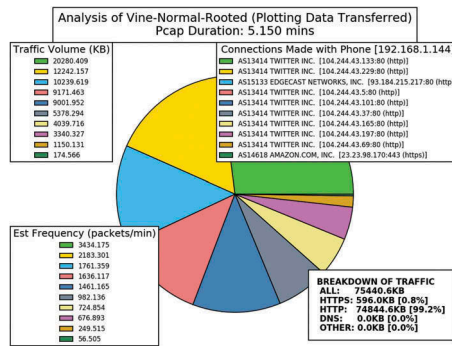


Figure A.15. Analysis of Vine traffic – rooted Nexus 6P.

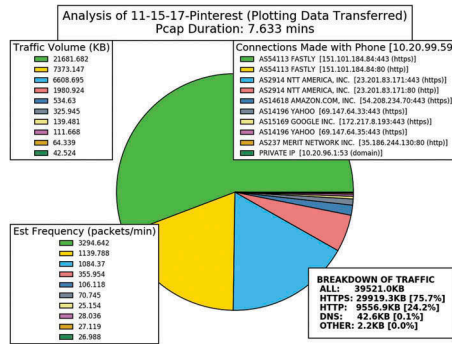


Figure A.16. Follow-up analysis of Pinterest Test 1 traffic – rooted Nexus 6P.

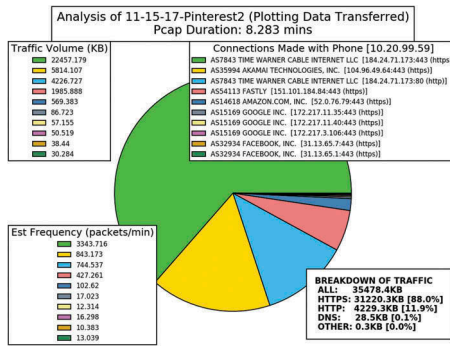


Figure A.17. Follow-up analysis of Pinterest Test 2 traffic – rooted Nexus 6P.

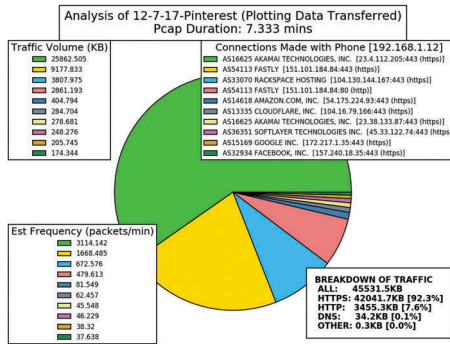


Figure A.18. Follow-up analysis of Pinterest Test 3 traffic – rooted Nexus 6P.

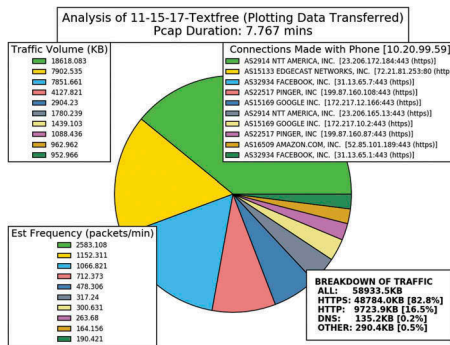


Figure A.19. Follow-up analysis of Textfree Nov 2017 traffic – rooted Nexus 6P.

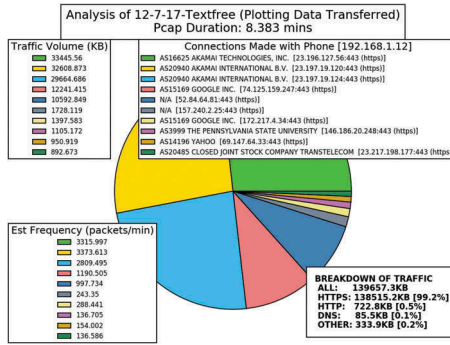


Figure A.20. Follow-up analysis of Textfree Dec 2017 traffic – rooted Nexus 6P.

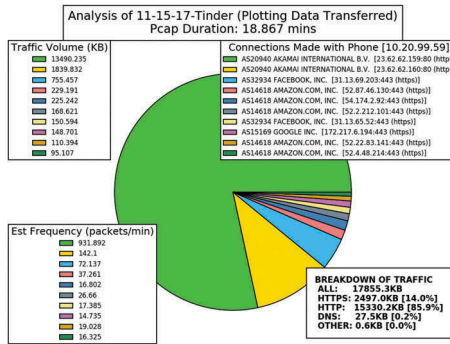


Figure A.21. Follow-up analysis of Tinder Nov 2017 traffic – rooted Nexus 6P.